PHP kézikönyv

Stig Sæther Bakken

Alexander Aulbach

Egon Schmid

Jim Winstead

Lars Torben Wilson

Rasmus Lerdorf

Andrei Zmievski

Jouni Ahto

Szerkesztette

Stig Sæther Bakken

Egon Schmid

Fordítók:
Csontos András
Heilig Szabolcs
Hojtsy Gábor
Kontra Gergely
Papp Győző
Tóth Attila
Varanka Zoltán
Másképpen segített:
Jouni Ahto
09-07-2002

PHP kézikönyv

Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Andrei Zmievski, és Jouni Ahto

Szerkesztette Stig Sæther Bakken

Szerkesztette Egon Schmid

Fordítók:
Csontos András
Heilig Szabolcs
Hojtsy Gábor
Kontra Gergely
Papp Győző
Tóth Attila
Varanka Zoltán
Másképpen segített:
Jouni Ahto

Megjelent 09-07-2002

Copyright

Tartalom

Előszó

A PHP, bővebben "PHP: Hypertext Preprocessor" egy széles körben használt, nyílt forráskódú, általános célú programozási nyelv, különösen jó web-fejlesztés támogatással, és HTML-be ágyazási képességekkel. A szintakszisa a C, Java és Perl nyelvekre épül, könnyen megtanulható. A nyelv fő célja lehetőséget teremteni dinamikusan generált weboldalak gyors készítésére, de a PHP ennél sokkal többre is képes.

Ez a kézikönyv legfőképpen függvény referenciaként használható, azonban a nyelvi elemek ismertetése is megtalálható benne, a főbb PHP szolgáltatások leírása, és más kapcsolódó információk mellett.

Ez a kézikönyv számos nyelven és formátumban letölthető a http://www.php.net/docs.php címen. A letöltések a tartalom változásával frissülnek. Ha érdekel, hogyan készül ez a kézikönyv, olvasd el az ’About the manual’ című függeléket.

A magyar kézikönyvet téritésmentesen készítjük, szabadidőnkben. A le nem fordított fejezetek mindig a legfrissebb angol szöveggel jelennek meg, ezért vegyes nyelvű oldalakkal igen ritkán fogsz találkozni. Időnként az azonban előfordulhat, hogy bizonyos már lefordított fejezetek nem teljesen aktuálisak. Emiatt ha mindig aktuális és pontos információra van szükséged, és értesz angolul, érdemes ezt a kézikönyvet az angol változattal párhuzamosan használni. Az esetleges kellemetlenségekért elnézést kérünk.

A PHP kézikönyv magyar fordításának oldala a http://weblabor.hu/ címen található. Ha szeretnél bekapcsolódni a fordításba, látogasd meg ezt a címet.

Rész I. Első lépések

Fejezet 1. Bevezetés a PHP-be

Mi az a PHP?

A PHP (rekurzív rövidítéssel "PHP: Hypertext Preprocessor") széles körben használt általános célú szkriptnyelv, amely kifejezetten alkalmas - akár HTML-be ágyazott - webalkalmazások fejlesztésére.

Egyszerű meghatározás, de mit is jelent ez valójában? Egy példán bemutatva:

Példa 1-1. Egy bevezető példa

```
<html>
    <head>
        <title>Példa</title>
    </head>
    <body>

        <?php
        echo "Helló, Én egy PHP szkript vagyok!";
        ?>

    </body>
</html>
```

Vedd észre, hogy ez mennyire különbözik más nyelveken (például Perl vagy C) írt hagyományos szkripttől. Sok parancsból álló programok helyett csak egy HTML fájlba kell egy kevés programkódot beépíteni, hogy a megfelelő HTML kimenetet produkálja. A PHP kódblokkokat speciális kezdő és befejező jelölések közé kell elhelyezni, és ezek biztosítják, hogy a feldolgozás során a váltogathasd a "PHP módot".

Az különbözteti meg a PHP-t a kliens oldali nyelvektől - pl. JavaScript -, hogy a kód a kiszolgálón fut. Az első példában látható oldal eredményét böngészőben megnézve, nem lehet megállapítani, hogy milyen kód állíthatta azt elő. Ezen felül a webszervert be lehet állítani úgy, hogy a PHP feldolgozzon minden HTML fájlt PHP blokkokat keresve, ezek után már tényleg nincs rá mód, hogy kitalálják, mit is rejt egy-egy programod.

A legjobb dolog a PHP használatában, hogy különösen egyszerű egy kezdő számára, de számos, fejlett szolgáltatást nyújt professzionális programozó számára is. Ne ijesszen el a PHP hosszú szolgáltatás listája. Gyors léptekkel lehet haladni a PHP használatában, és pár órán belül már egyszerű szkriptek írására is képes lehetsz.

Habár a PHP fejlesztésében a szerver-oldali programozás kapja a legnagyobb hangsúlyt, annál sokkal többet tud. Olvasd tovább ezt a fejezetet a következő - Mit tud a PHP? - című résszel folytatva!

Mit tud a PHP?

Bármit. A PHP főleg szerver-oldali szkriptek írására készült, azaz bármire képes, amit más CGI programok el tudnak végezni, ilyen funkciók az űrlap adatok feldolgozása, dinamikus tartalom generálása, vagy sütik küldése és fogadása. De a PHP ennél sokkal többet tud.

Három fő területen használnak PHP programokat.

- Szerver oldali programozás. Ez a hagyományos, és fő használati formája a PHP-nek. Három komponens szükséges ahhoz, hogy ezt a formát használhasd. Az első a PHP értelmező (CGI vagy szerver modul formájában), egy webszerver és egy webböngésző. Egy webszerverrel mindenképpen rendelkezni kell, megfelelően telepített és beállított PHP-vel. A PHP program kimenetét a webböngészővel lehet megtekinteni, a szerveren keresztül elérve a szkriptet. Lásd a telepítési utasításokatc. részt további információkért!
- Parancssori programozás. PHP programok szerver és böngésző nélkül is futtathatóak. Ha ilyen környezetben szeretnéd használni a PHP-t, csak a PHP értelmezőre van szükséged. Ebben a formában gyakran cron-al (ütemező Windows esetén) futtatott programokat írnak, vagy egyszerű szövegfeldolgozó szkripteket készítenek. Lásd a Parancssori használat című függeléket további információért!
- Kliens-oldali ablakozós alkalmazások írása. A PHP valószínűleg nem a legjobb nyelv ablakozós alkalmazások írásához, de ha nagyon jól ismered a PHP-t, és szeretnél néhány fejlett PHP szolgáltatást használni a kliens-oldali programjaidban, a PHP-GTK-t is használhatod ilyen programok írásához. Ezt használva lehetőséged van operációs rendszerfüggetlen programok írására is. A PHP-GTK a PHP egy kiterjesztése, nem érhető el a hivatalos PHP csomagban. Ha további információkra van szükséged látogsd meg a PHP-GTK webhelyet (http://gtk.php.net/)!

A PHP használható a legfontosabb operációs rendszereken, beleértve a Linuxot, sok Unix változatot (beleértve a HP-UX, Solaris és OpenBSD rendszereket), a Microsoft Windows-t, a Mac OS X rendszert, a RISC OS-t, és másokat. A PHP a legtöbb webszervert is támogatja, beleértve az Apache, Microsoft Internet Information Server, Personal Web Server, Netscape és iPlanet szervereket, az Oreilly Website Pro, Caudium, Xitami, OmniHTTPd, és más szervereket. A legtöbb szerverhez a PHP modul szintű támogatást nyújt, de más a CGI szabványt támogató szerverekkel is együtt tud működni CGI feldolgozóként.

Összességében a PHP használatakor szabadon választhatsz operációs rendszert és webszervert. Ráadásul a függvény-alapú és objektum orientált programozás, vagy ezek keveréke közötti választás is rajtad áll. Bár nem minden szokásos OOP szolgáltatás került megvalósításra a PHP-ben, sok eljáráskönytár és nagyobb alkalmazás is egyedüliként/kizárólagosan az OOP-t használja, például a PEAR könyvtár.

A PHP képességei nem csak HTML kimenet előállítására korlátozódnak. Képeket, PDF állományokat vagy akár Flash mozikat (libswf vagy Ming kiterjesztéssel) is létrehozhatsz futásidőben. Természetesen egyszerűen generálhatsz bármilyen szöveges kimenetet, mint az XHTML vagy bármilyen más XML. A PHP elő tudja állítani ezeket az állományokat, és el tudja menteni a szerven a közvetlen kiküldésük helyett, valamilyen szerver-oldali gyorsítótárat valósítva meg ezzel.

Az egyik legjobb és legfontosabb tulajdonsága a nyelvnek az adatbázisok széles körű támogatása. Adatbázisokat kezelő weblap készítése PHP segítségével hihetetlenül egyszerű. A következő adatbázisok támogatja jelenleg:

Adabas D	Ingres	Oracle (OCI7 és OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (csak olvasásra)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

A PHP rendelkezik egy DBX adatbázis absztrakciós kiterjesztéssel is, amellyel egyöntetűen és áttetszű módon lehet kezelni bármilyen adatbázist, amit ez a kiterjesztés támogat. Ezen kívül a PHP támogatja az ODBC-t, ezért bármilyen más, ezt a szabványt támogató adatbázishoz is lehet kapcsolódni.

A PHP támogatja a kommunikációt más szolgáltatásokkal is különböző protokollok segítségével, úgy mint LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (Windows rendszeren), és számos más. Sőt, nyithatsz hálózati foglalatokat is (socket) és kommunikálhatsz más protokolokkal is. A PHP támogatja a WDDX komplex adatcseréket, ami bármely más web programozási nyelvvel való kommunikációt megkönnyíheti. A PHP szintén rendelkezik a Java objektumok létrehozásának, és átlátszóan PHP objektummokként való kezelésének képességével. A CORBA kiterjesztés távoli objektumok elérésére használható.

A PHP rendkívül jó szövegfeldolgozó képességekkel rendelkezik, a POSIX és Perl reguláris kifejezésektől az XML állományok kezeléséig. Az XML dokumentumok feldologozásához és eléréséhez a SAX és DOM szabványok is használhatóak. Az XSLT kiterjesztés XML dokumentumok általalakítására használható.

Ha elektornikus üzleti környezetben használod a PHP-t, hasznosnak fogod találni a Cybercash, CyberMUT, VerySign Payflow Pro and CCVS függvényeket az internetes fizetést megvalósító programokban.

Végül, de nem utolsósorban a PHP számos más érdekes kiterjesztéssel szolgálhat, mint például az mnoGoSearch kereső függvények, az IRC átjáró függvények, tömörítő eszközök (gzip, bz2), naptár átalakítás, fordítás...

Ahogy látható, ez az oldal nem elegendő a PHP minden szolgáltatásának és előnyének felsorolásához. Lásd a PHP telepítése és a függvény referencia részeket további információkért!

Fejezet 2. Telepítés

Általános telepítési szempontok

Mielőtt hozzálátnál a telepítéshez, meg kell tudnod határozni, hogy mire akarod használni a PHP-t. A következő három fő területen használhatod (Mire jó a PHP?):

- Szerveroldali programozásra
- Parancssori programok írására
- Kliens oldali GUI alkalmazások kifejlesztéséhez.

Az első és legszokványosabb használatához három dologra lesz szükséged: magára a PHP-ra, egy webszerverre és egy webböngészőre. Valószínűleg már rendelkezel egy webböngészővel, és az operációs rendszer telepítésétől függően egy webszerverrel is (Apache Linux alatt vagy IIS Windows alatt). Bérelhetsz tárterületet egy cégtől is, ebben az esetben nem kell semmit magadtól telepítened, csak a PHP szkripteket megírni és feltölteni őket a bérelt webtárhelyedre és nézni a végeredményt a böngészőben. A tárterület bérlésével foglalkozó cégek listája: http://hosts.php.net/.

Ha saját magad telepíted a PHP-t, akkor kétféleképpen csatlakoztathatod a kiszolgálóhoz. A PHP-nak rengeteg kiszolgálóhoz van közvetlen modulinterfésze (SAPI). Ezek a szerverek az Apache, Microsoft Internet Information Server, Netscape és iPlanet szerverek. Ezenkívül sok szerver támogatja az ISAPI-t, a Microsoft modulinterfészét mint pl. az OmniHTTPd. Ha PHP-t nem a webszerveredhez modulként telepíteni, mindig lehetőséged van CGI feldolgozóként futtatni. Ez azt jelenti, hogy a webszerver a PHP parancssorból indítható változatát (Windows alatt `php.exe`) használja a PHP kérések lekezelésére.

Ha érdekelt vagy a PHP-s parancssori programozásban is (pl. automatikus offline képgenerálásra vagy szöveges állományok feldolgozására írt programok), akkor szükséged lesz a parancssorból futtatható változatra is. A részletesebb tájékoztatásért olvasd el a PHP parancssori alkalmazásokhoz. Ebben az esetben nincs szükség sem szerverre sem böngészőre.

A PHP és a PHP-GTK kiterjesztés segítségével kliens oldali GUI alkalmazásokat is fejleszthetsz. Ez teljesen eltérő megközelítés, mint a weboldalak programozása, mivel itt nem HTML kimenetet kell generálni, hanem ablakokat és a bennük levő objektumokat kell kezelni. A PHP-GTK-ról többet is olvashatsz, ha ellátogatsz a honlapjukra (http://gtk.php.net/). A PHP-GTK nem része a hivatalos PHP disztribuciónak.

Ettől a ponttól ez a fejezet a PHP webszerverhez történő telepítésével foglalkozik, UNIX és Windows alatt - mind modulként mind CGI feldolgozóként.

A forráskód és a futtatható állományok néhány rendszerre (beleértve a Windows-t) megtalálhatóak `http://www.php.net/` címen. Ha lehet, akkor valamelyik hozzád legközelebb eső tükörszervert (http://www.php.net/mirrors.php) vedd igénybe a letöltésekhez.

Telepítés UNIX rendszerre

Ez a fejezet bemutatja, hogy miként kell feltelepíteni és beállítani a PHP-t UNIX rendszeren. Mielőtt elkezded a telepítést, nézd meg a rendszerednek ás szerverednek megfelelő fejezetet is!

Nélkülözhetetlen előismeretek és szoftverek:

- Alapvető UNIX ismeretek ("make" és a C fordító használata, ha saját magad szeretnéd végezni a fordítást)

- Egy ANSI C fordító (ha fordítasz)
- flex (fordításhoz)
- bison (fordításhoz)
- Egy web szerver
- Bármilyen modul által igényelt elem (mint például a gd vagy pdf könyvtárak)

Számos módja van a PHP telepítésének a UNIX rendszereken, vagy egy fordítási és konfigurálási eljárással, vagy különböző előre-csomagolt megoldások használatával. Az első mód jóval bonyolultabb, így a dokumentáció inkább erre a területre fekteti a hangsúlyt.

Az alapvető konfigurálási folyamatot a `configure` számára átadott parancssori paraméterek szabályozzák. Ez a fejezet körvonalazza a leggyakoribb lehetőségeket, de számos más paraméter is rendelkezésre áll. Lásd a Configure opciók teljes listáját részletesebb információkért. Számos módja van a PHP telepítésének:

- Apache modul fordítása
- fhttpd modul fordítása
- Más szerverekre (AOLServer, NSAPI, phttpd, Pi3Web, Roxen, thttpd, vagy Zeus) fordítás
- CGI futtatható állomány készítése

Apache modul (Gyors referencia)

A PHP-t számos különböző módon lehet fordítani, de az egyik legkedveltebb forma az Apache modulként való fordítás. A következőkben egy gyors módszert mutatunk be erre.

Példa 2-1. Gyors telepítési utasítások PHP 4-hez (Apache modul verzió)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-x.x.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. ./configure --activate-module=src/modules/php4/libphp4.a
13. make
14. make install

  Ehelyett a lépés helyett talán jobb, ha a httpd futtatható állományát
  a meglévőre rámásolod. Győződj meg róla, hogy előtte leállítottad
  a szervert!

15. cd ../php-x.x.x
16. cp php.ini-dist /usr/local/lib/php.ini
```

```
17. Szerkeszd a httpd.conf vagy srm.conf fájlt és add hozzá:
      AddType application/x-httpd-php .php

18. Használd a szokásos módszert az Apache szerver újraindítására.
    (Le kell állítanod és újra kell indítanod a szervert, nem csak
    újratölteni egy HUP vagy USR1 jellel.)
```

Fordítás

Ha a PHP-t sikeresen konfiguráltad, készen állsz a fordításra. A **make** parancs elvégzi ezt a feladatot. Ha nem sikerül a fordítás, és nem tudod kideríteni, miért, olvasd el a Problémák című részt.

Telepítés UNIX/Linux rendszerre

Ez a fejezet a PHP Linux rendszerre történő telepítésének dokumentációját tartalmazza.

Csomagok használata

Számos Linux disztribúció valamilyen csomag támogatással rendelkezik, (például rpm). Ez sokat tud segíteni egy általános telepítés elvégzésében, de ha más funkciókra is vágysz (például biztonságos szerver funkciókra, vagy más adatbázis támogatására), előfordulhat, hogy mégis magadnak kell fordítanod a PHP-t és/vagy a szervert. Ha nem vagy tapasztalt az önálló fordításban, megéri utánanézni, hogy elkészítette-e már valaki más azt a csomagot, ami a kívánt funkciókkal rendelkezik.

Telepítés UNIX/HP-UX rendszerre

Ez a fejezet a PHP HP-UX rendszerre történő telepítésének dokumentációját tartalmazza.

Példa 2-2. Telepítési útmutató a HP-UX 10-hez

```
Forrás: paul_mckay@clearwater-it.co.uk
Dátum: 04-Jan-2001 09:49
(Ezek a tippek a PHP v4.0.4 és Apache v1.3.9 verziókra vonatkoznak)

Tehát szeretnéd telepíteni a PHP-t és az Apache szervert
egy HP-UX 10.20-at futtató gépen?

1. Szükséged lesz a gzip-re. Töltsd le a
http://hpux.connect.org.uk/ftp/hpux/Gnu/gzip-1.2.4a/gzip-1.2.4a-sd-10.20.depot.Z
címről, tömörítsd ki, és telepítsd swinstall-al.

2. Szükséged lesz a gcc-re. Töltsd le a
http://gatekeep.cs.utah.edu/ftp/hpux/Gnu/gcc-2.95.2/gcc-2.95.2-sd-10.20.depot.gz
címről, tömörítsd ki (most már gzip-el), és telepítsd swinstall-al.
```

3. Szükséged lesz a GNU binutils-ra. Töltsd le a
http://hpux.connect.org.uk/ftp/hpux/Gnu/binutils-2.9.1/binutils-2.9.1-sd-10.20.depot.gz
címről, tömörítsd ki, és telepítsd swinstall-al.

4. Szükséged lesz a bison-ra. Töltsd le a
http://hpux.connect.org.uk/ftp/hpux/Gnu/bison-1.28/bison-1.28-sd-10.20.depot.gz
címről, és telepítsd az eddigiek szerint.

5. Szükséged lesz a flex-re. Ennek sajnos csak a forráskódját
tudod letölteni az egyik http://www.gnu.org tükörkiszolgálóról.
A <filename>non-gnu</filename> könyvtárban találod az ftp helyen. Töltsd le az
állományt, tömörítsd ki, és a tar -xvf paranccsal megkapod
a szükséges fájlokat. Lépj be a létrejött flex könyvtárba, és
hajsd futtasd a következőket sorban egymás után: ./configure,
make, make install.

Ha hibaüzeneteket kapsz, feltehetően az a probléma, hogy gcc nincs
a PATH-ban, tehát add hozzá a PATH-hoz.

Most pedig jönnek a nehezebb dolgok.

6. Töltsd le a PHP és Apache forráskódokat.

7. Alkalmazd rájuk a gunzip és tar -xvf parancsokat.

Módosítanod kell pár állományt, hogy helyesen leforduljanak.

8. Először a configure nevű fájl szorul némi szerkesztésre, mivel
úgy tűnik nem tudja követni, hogy HP-UX gépen vagyunk.
Lesz még egy jobb módszer ennek a megoldására, de addig is
egy jól működő javítás a
    lt_target=hpux10.20
beillesztése a configure szkript 47286-adik soránál.

9. A következő lépésben az Apache GuessOS fájl szorul
javításra. Az apache_1.3.9/src/helpers fájlban írd át a
89. sort. Ez a jelenlegi tartalma:
    "echo "hp${HPUXMACH}-hpux${HPUXVER}"; exit 0"
Erre kell átirni:
    "echo "hp${HPUXMACH}-hp-hpux${HPUXVER}"; exit 0"

10. Megosztott modulként (SO) nem telepíthető a PHP HP-UX
rendszeren, ezért statikusan bele kell fordítanod a szerverbe.
Ehhez kövesd az Apache oldalon található utasításokat.

11. A PHP és Apache most már sikeresen lefordult, de az Apache
nem fog elindulni. Létre kell hoznod egy új felhasználót
az Apache számára, például www vagy apache néven. Utána
a 252-253-ik sort kell módosítanod az Apache conf/httpd.conf
állományban, tehát ehelyett a két sor helyett:
    User nobody
    Group nogroup
valami hasonlód lesz:
    User www
    Group sys

```
Ez azért szükséges, mivel az Apache nem futtatható
a nobody nevű felhasználóval HP-UX alatt.

Most már az Apache és a PHP is sikeresen működik.

Remélem ez az útmutató segített,
Paul Mckay.
```

Telepítés Unix/Solaris rendszerre

Ez a fejezet a PHP Solaris rendszerre történő telepítésének dokumentációját tartalmazza.

Szükséges programok

A Solaris telepítések gyakran nem tartalmaznak C fordítót, és a kapcsolódó eszközöket. A szükséges programok:

- gcc (javasolt, más C fordító is jó lehet)
- make
- flex
- bison
- m4
- autoconf
- automake
- perl
- gzip
- tar

Természetesen szükséged lehet további szoftverek telepítésére (és esetleg fordítására), ha plusz funkciókat szeretnél elérni (például Oracle vagy MySQL adatbázisok).

Csomagok használata

Egyszerűsítheted a Solaris telepítési eljárást, ha a pkgadd-et használod a szükséges komponensek nagyrészének telepítésénél.

Telepítés Unix/OpenBSD rendszerre

Ez a fejezet bemutatja, hogy miként kell telepíteni és beállítani a PHP-t OpenBSD (http://www.openbsd.org/) rendszeren.

Port-ok használata

Ez az ajánlott telepítési módszer OpenBSD-n, mivel így az üzemeltetők a legfrissebb biztonsági és hiba-javításokat képesek alkalmazni majd rá. Ahhoz, hogy használhasd ezt az eljárást, bizonyosodj meg, hogy megvan a legújabb port tree (http://www.openbsd.org/ports.html). Ezután találd ki, mely "flavor"-eket kívánod telepíteni, és add ki a **make install** parancsot. Itt van egy példa, hogyan is kell ezt csinálni:

Példa 2-3. OpenBSD Port telepítési példa

```
$ cd /usr/ports/www/php4
$ make show VARNAME=FLAVORS
 (milyen flavor-t akarsz a listából)
$ env FLAVOR="imap gettext ldap mysql gd" make install
$ /usr/local/sbin/php4-enable
```

Package-ek használata

Léteznek előre lefordított csomagok az általad használt OpenBSD (http://www.openbsd.org/) verziókhoz is. Ezek az operációs rendszerrel járó Apache verziókkal vannak összeillesztve. Mégis mivel a beállítható lehetőségek (*flavors*) száma nagy ehhez a porthoz, talán egyszerűbb a port tree használatával újat fordítani a forrásból. Olvasd el, hogy milyen csomagok elérhetők, man oldalak: packages(7) (http://www.openbsd.org/cgi-bin/man.cgi?query=packages)

Telepítés Mac OS X rendszerre

Ez a fejezet a PHP Mac OS X rendszerre történő telepítésének dokumentációját tartalmazza.

Csomagok használata

Rendelkezésre áll néhány előre-csomagolt és előre-fordított PHP változat a Mac OS X rendszerekre. Ez sokat tud segíteni egy általános telepítés elvégzésében, de ha más funkciókra is vágysz (például biztonságos szerver funkciókra, vagy más adatbázis támogatására), előfordulhat, hogy mégis magadnak kell fordítanod a PHP-t és/vagy a szervert. Ha nem vagy tapasztalt az önálló fordításban, megéri utánanézni, hogy elkészítette-e már valaki más azt a csomagot, ami a kívánt funkciókkal rendelkezik.

Fordítás OS X szerveren

Két - némiképp különböző - verziója érhető el a Mac OS X-nek, a kliens és szerver változat. A következők az OS X szerverre vonatkoznak.

Példa 2-4. Mac OS X szerver telepítés

```
1. Töltsd le az Apache és PHP legfrissebb változatait
2. Tömörítsd ki ezeket, és futtasd a configure programot az Apache-ra:
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache

4. Esetleg hozzáadhatod még a következő sort:
    setenv OPTIM=-O2
    Ha szerenéd, ha a fordító némi optimalizálást végezne.

5. Lépj be a PHP 4 forrás könyvtárába, és futtasd a configure programot:
    ./configure --prefix=/usr \
    --sysconfdir=/etc \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --with-xml \
    --with-apache=/src/apache_1.3.12

    Ha bármilyen más kiterjesztést szeretnél (MySQL, GD, stb.),
    szerepeltesd a megfelelő paramétereket itt. A
    --with-apache paraméternek az Apache forráskönyvtárát
    add meg, például "/src/apache_1.3.12".
6. make
7. make install
    Ez létre fog hozni egy könyvtárat az Apache forráskönyvtárában
    az src/modules/php4 alatt.

8. Most újra futtatnod kell a configure-t az Apache-ra,
   hogy beépítsd a PHP 4-et:
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache \
    --activate-module=src/modules/php4/libphp4.a

    Itt kaphatsz egy üzenetet, ami azt állítja, hogy a libmodphp4.a
    elvault. Ha ez történik, menj az src/modules/php4 könyvtárba,
    az Apache forráskönyvtár alatt, és hajtsd végre ezt:

    ranlib libmodphp4.a

    Aztáb lépj vissza az Apache forráskönyvtár gyökerébe, és
    ismét futtasd az előző configure parancssort. Ez aktualizálja
```

```
    a link táblát.

9. make

10. make install

11. Másold, és nevezd át a php.ini-dist állományt a "bin"
    könyvtárba a PHP 4 forráskönvytárból:
    cp php.ini-dist /usr/local/bin/php.ini

    vagy (ha nincs local könyvtárad)

    cp php.ini-dist /usr/bin/php.ini
```

Más példák a Mac OS X kliensre (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) és a Mac OS X szerverre (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) elérhetőek a Stepwise (http://www.stepwise.com/) honlapon.

Fordítás MacOS X kliensre

Ezeket a tippeket Marc Liyanage (http://www.entropy.ch/software/macosx/) adta szívesen közre.

A PHP Apache webszerver modulja a Mac OS X része. Ez a verzió a MySQL és PostgreSQL adatbázis-kezelők támogatását is tartalmazza.

Megjegyés: Légy körültekintő az alábbiakban, mert tönkreteheted az Apache szerveredet!

Telepítéshez az alábbiakat kell tenni:

- 1. Nyiss egy terminál ablakot
- 2. Írd be "wget http://www.diax.ch/users/liyanage/software/macosx/libphp4.so.gz", és várd meg, amíg letöltődik
- 3. Írd be "gunzip libphp4.so.gz"
- 4. Írd be "sudo apxs -i -a -n php4 libphp4.so"

Most írd be, hogy "`sudo open -a TextEdit /etc/httpd/httpd.conf`" TextEdit megnyitja a webszerver konfigurációs fájlját. Keresd meg a következő két sort a fájl vége felé (használd a Find parancsot):

```
#AddType application/x-httpd-php .php
#AddType application/x-httpd-php-source .phps
```

Töröld ki a két hashmark-ot(#), és ezután mentsd el a fájlt, lépj ki a TextEdit-ből.

Végül írd be: "`sudo apachectl graceful`" , hogy újrainduljon a webszerver.

Mostantól a PHP-nak futni kell. Ezt ellenőrizheted, ha a "Sites" könyvtáradba bemásolod a "test.php" fájlt, amiben nincs más csak ez a sor: `<?php phpinfo() ?>`.

Most nyisd meg a `127.0.0.1/~a_te_userneved/test.php` címen levő oldalt a webböngésződben. A PHP modulok információs táblázatait kell látnod ezen az oldalon.

Configure opciók teljes listája

> **Megjegyzés:** Ezek a paraméterek csak csak fordításidőben használhatóak. Ha egy lefordított PHP konfigurációján szeretnél változtatni, nézd meg a Konfiguráció című részt.

A továbbiakban egy teljes lista következik a PHP 3 és PHP 4 `configure` szkriptek által támogatott paraméterekről, amik UNIX-szerű operációs rendszereken használhatóak. A jelöléseknek megfelelően néhány a csak a PHP 3-ban elérhető, néhány csak PHP 4-ben, vannak ami mindkettőben. Sok paraméter neve megváltozott a PHP 3 és a PHP 4 között, miközben továbbra is ugyanazt a funkciót valósítják meg. Ezek kereszthivatkozásokkal vannak ellátva, ezért ha gondod akad a PHP 3 paraméterek átvételével, itt ellenőrizd, hogy megmaradtak-e a paraméterek nevei.

- Adatbázisok
- E-üzlet
- Grafika
- Máshova nem sorolható
- Hálózati
- PHP viselkedését szabályozó
- Szerver
- Szöveg és nyelv
- XML

Adatbázisok

`--with-adabas[=DIR]`

PHP 3, PHP 4: Beépíti az Adabas D támogatást. A DIR az Adabas telepítési könyvtára, alapbeállításban /usr/local.

Adabas honlap (http://www.software-ag.com/adabasd/)

`--enable-dba=shared`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A DBA megosztott modulként való beépítése

`--enable-dbx`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a DBX támogatást

`--enable-dbase`

PHP 3: Ez az opció nem áll rendelkezésre; használd helyette a --with-dbase paramétert!

PHP 4: Engedélyezi a beépített dbase támogatást. Semmilyen külső könyvtár nem szükséges hozzá.

`--with-dbase`

PHP 3: Engedélyezi a beépített dbase támogatást. Semmilyen külső könyvtár nem szükséges hozzá.

PHP 4: Ez az opció nem áll rendelkezésre; használd helyette az --enable-dbase paramétert!

`--with-db2[=DIR]`

PHP 3, PHP 4: Beépíti a Berkeley DB2 támogatást.

`--with-db3[=DIR]`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a Berkeley DB3 támogatást.

`--with-dbm[=DIR]`

PHP 3, PHP 4: Engedélyezi a DBM támogatást.

`--with-dbmaker[=DIR]`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a DBMaker támogatást. A DIR a DBMaker alap telepítési könyvtára, alapbeállításban a legfrissebben telepített DBMaker könyvtára (pl. /home/dbmaker/3.6).

`--with-empress[=DIR]`

PHP 3, PHP 4: Engedélyezi az Empress támogatást. A DIR az Empress alap telepítési könyvtára, alapbeállításban $EMPRESSPATH.

`--enable-filepro`

PHP 3: Ez az opció nem áll rendelkezésre; használd helyette a --with-filepro paramétert!

PHP 4: Engedélyezi a beépített csak olvasásra képes filePro támogatást. Semmilyen külső könyvtár nem szükséges hozzá.

`--with-filepro`

PHP 3: Engedélyezi a beépített csak olvasásra képes filePro támogatást. Semmilyen külső könyvtár nem szükséges hozzá.

PHP 4: Ez az opció nem áll rendelkezésre; használd helyette az --enable-filepro paramétert!

--with-fbsql[=DIR]

PHP 3: nem elérhető.

PHP 4: engedélyezi a FrontBase SQL támogatást. DIR a FrontBase kiinduló telepítési könyvtára, alapértelmezésben a szokásos Frontbase telepítési könyvtár. A szokásos Frontbase telepítési könyvtárak operációs rendszerenként változnak: Solaris: `/opt/FrontBase`, WinNT: `\usr\FrontBase`, Linux: `/usr/frontbase`, Mac OSX: `/Library/FrontBase`.

--with-gdbm[=DIR]

PHP 3, PHP 4: Beépíti a GDBM támogatást.

--with-hyperwave

PHP 3, PHP 4: Beépíti a Hyperwave támogatást.

--with-ibm-db2[=DIR]

PHP 3, PHP 4: Beépíti az IBM DB2 támogatást. A DIR a DB2 alap telepítési könyvtára, alapbeállításban `/home/db2inst1/sqllib`.

IBM DB2 honlap (http://www.ibm.com/db2/)

--with-informix[=DIR]

PHP 3, PHP 4: Beépíti az Informix támogatást. A DIR az Informix alap telepítési könyvtára, és nincs alapbeállítása.

--with-ingres[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti az Ingres II támogatást. A DIR az Ingres könyvtára (alapbeállításban /II/ingres).

--with-interbase[=DIR]

PHP 3, PHP 4: Beépíti az InterBase támogatást. A DIR az InterBase alap telepítési könyvtára, alapbeállításban `/usr/interbase`.

Interbase függvények

Interbase honlap (http://www.interbase.com/)

--with-ldap[=DIR]

PHP 3: Beépíti az LDAP támogatást. A DIR az LDAP alap telepítési könyvtára. Alapbeállításban `/usr` és `/usr/local`

PHP 4: Beépíti az LDAP support. A DIR az LDAP alap telepítési könyvtára.

Ez biztosítja az LDAP (Lightweight Directory Access Protocol) támogatást. Paramétere az LDAP alap telepítési könyvtára, alapbeállításban `/usr/local/ldap`.

Az LDAP-ről részletesebb információ található az RFC1777 (http://www.faqs.org/rfcs/rfc1777.html)-ben és az RFC1778 (http://www.faqs.org/rfcs/rfc1778.html)-ban.

--with-msql[=DIR]

PHP 3, PHP 4: Engedélyezi az mSQL támogatást. Paramétere az mSQL telepítési könyvtára, alapbeállításban `/usr/local/Hughes`. Ez az alap könyvtára az mSQL 2.0 disztribúciónak. A **configure** automatikusan detektálja, hogy milyen mSQL verziót használsz, és a PHP támogatja mind az 1.0-ás mind a 2.0-ás mSQL-t, de ha ha a PHP-t az 1.0-ás mSQL támogatással fordítod, csak 1.0-ás mSQL adatbázisokat tudsz majd elérni, és fordítva.

Lásd még az mSQL beállítási lehetőségeket a konfigurációs fájlban.

mSQL honlap (http://www.hughes.com.au/)

--with-mysql[=DIR]

PHP 3: Beépíti a MySQL támogatást. A DIR a MySQL alap telepítési könyvtára, alapbeállításban számos gyakori helyen keres MySQL fájlokat keresve.

PHP 4: Beépíti a MySQL támogatást. A DIR a MySQL alap könyvtára. Ha nem adod meg, a beépített MySQL könyvtár kerül felhasználásra. Ez az opció alapbeállításban be van kapcsolva.

Lásd még a MySQL beállítási lehetőségeket a konfigurációs fájlban.

MySQL home page (http://www.mysql.com/)

--with-ndbm[=DIR]

PHP 3, PHP 4: Beépíti az NDBM támogatást.

--with-ovrimos

PHP 3, PHP 4: Bekapcsolja az Ovrimos támogatást.

--with-oci8[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti az Oracle-oci8 támogatást. Alapbeállításban a DIR az ORACLE_HOME értéke.

--with-oracle[=DIR]

PHP 3: Beépíti az Oracle adatbázis támogatását. A DIR az Oracle könyvtára, alapbeállításban az $ORACLE_HOME értéke.

PHP 4: Beépíti az Oracle-oci7 támogatást. Alapbeállításban a DIR az ORACLE_HOME értéke.

Beépíti az Oracle támogatást. Oracle 7.0 és 7.3 közötti verziókkal tesztelve. A paraméter az ORACLE_HOME könyvtár. Nem kell megadnod ezt a paramétert, ha az Oracle környezeted be van állítva.

Oracle honlap (http://www.oracle.com/)

--with-pgsql[=DIR]

PHP 3: Beépíti a PostgresSQL támogatást. A DIR a PostgresSQL alap telepítési könyvtára, alapbeállításban `/usr/local/pgsql`.

PHP 4: Beépíti a PostgreSQL támogatást. A DIR a PostgreSQL alap telepítési könyvtára, alapbeállításban `/usr/local/pgsql`. Állítsd a DIR értékét shared-re, ha külső PHP modulként szeretnéd lefordítani, vagy állítsd shared,DIR-re, ha külső modulként fordítod, és megadod a könyvtárat.

Lásd a Postgres beállítási lehetőségeket a konfigurációs fájlban.

PostgreSQL honlap (http://www.postgresql.org/)

--with-solid[=DIR]

PHP 3, PHP 4: Beépíti a Solid támogatást. A DIR a Solid alap telepítési könyvtára, alapbeállításban /usr/local/solid.

Solid honlap (http://www.solidtech.com/)

--with-sybase-ct[=DIR]

PHP 3, PHP 4: Beépíti a Sybase-CT támogatást. A DIR a Sybase könyvtára, alapbeállításban /home/sybase.

Lásd még a Sybase-CT beállítási lehetőségeket a konfigurációs fájlban.

--with-sybase[=DIR]

PHP 3, PHP 4: Beépíti a Sybase-DB támogatást. A DIR a Sybase könyvtára, alapbeállításban `/home/sybase`.

Lásd még a Sybase beállítási lehetőségeket a konfigurációs fájlban.

Sybase home page (http://www.sybase.com/)

--with-openlink[=DIR]

PHP 3, PHP 4: Beépíti az OpenLink ODBC támogatást. A DIR az OpenLink alap telepítési könyvtára, alapbeállításban /usr/local/openlink.

Az OpenLink Software honlapja (http://www.openlinksw.com/)

--with-iodbc[=DIR]

PHP 3, PHP 4: Beépíti az iODBC támogatást. A DIR az iODBC alap telepítési könyvtára, alapbeállításban `/usr/local`.

Ez a szolgáltatás először az iODBC Driver Manager-hez készült, ami egy szabadon terjeszthető ODBC vezérlő számos UNIX rendszerre.

FreeODBC honlap (http://www.jepstone.net/FreeODBC/) vagy iODBC honlap (http://www.iodbc.org/)

--with-custom-odbc[=DIR]

PHP 3, PHP 4: Beépíti egy tetszőleges, egyedi ODBC könyvtár támogatását. Paramétere az bázis-könyvtár, alapbeállításban `/usr/local`.

Ez az opció elvárja, hogy előre definiáld a CUSTOM_ODBC_LIBS értéket, mielőtt elindítod a configure szkriptet. Úgyszintén szükséges, hogy legyen egy érvényes odbc.h fájlod az include path-ban. Ha nincs ilyened, hozz létre egyet, és építsd be a saját header-edet onnan. Elképzelhető, hogy a saját header fájlodnak is szüksége lesz pár extra definícióra, főleg, ha platformfüggetlen. Definiáld ezeket CFLAGS-ben.

Például QNX alatt használhatod a Sybase SQL Anywhere-t a következő sorral:

```
CFLAGS=-DODBC_QNX LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc"
./configure --with-custom-odbc=/usr/lib/sqlany50
```

--disable-unified-odbc

PHP 3: Kikapcsolja a unified ODBC támogatást. Csak akkor van értelme, ha az iODBC, Adabas, Solid, Velocis vagy egy saját ODBC felület engedélyezett.

PHP 4: Ez az opció nem áll rendelkezésre a PHP 4-ben.

A Unified ODBC modul, egy szokásos felület az ODBC-alapú megoldásokkal dolgozó adatbázisokhoz, mint a Solid, az IBM DB2 és az Adabas D. Szintén működőképes normál ODBC könyvtárak esetén. Teszteltük iODBC, Solid, Adabas D, IBM DB2 és Sybase SQL Anywhere adatbázisokkal. Elvárja, hogy egy (és csak egy) engedélyezett ezek közül a modulok közül, vagy a Velocis modul engedélyezett, esetleg egy saját ODBC könyvtár. Ez az opció csak akkor használható, ha az alábbiak közül egy paraméter már meg van adva: --with-iodbc, --with-solid, --with-ibm-db2, --with-adabas, --with-velocis, vagy --with-custom-odbc.

--with-unixODBC[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a unixODBC támogatást. A DIR a unixODBC alap telepítési könyvtára, alapbeállításban /usr/local.

--with-velocis[=DIR]

PHP 3, PHP 4: Beépíti a Velocis támogatást. A DIR a Velocis alap telepítési könyvtára, alapbeállításban /usr/local/velocis.

Velocis honlap (http://www.birdstep.com/database_technology/rdm_server.php3)

E-üzlet

--with-ccvs[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a CCVS támogatást. Add meg a CCVS telepítési könyvtárat a DIR paraméterben!

--with-cybermut[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a Cybermut támogatást. A DIR paraméter a Cybermut SDK könyvtára, ahol a `libcm-mac.a` és `cm-mac.h` állományok találhatóak.

--with-mck[=DIR]

PHP 3: Beépíti a Cybercash MCK támogatást. A DIR a cybercash mck könyvtára, ami alapbeéllításban `/usr/src/mck-3.2.0.3-linux`. Segítségért az `extra/cyberlib`-et nézd meg.

PHP 4: Ez az opció nem áll rendelkezésre; használd helyette a --with-cybercash paramétert!

--with-cybercash[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre; használd helyette a --with-mck paramétert!

PHP 4: Beépíti a CyberCash támogatást. A DIR a CyberCash MCK telepítési könyvtára.

--with-pfpro[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a Verisign Payflow Pro támogatást.

Grafika

--enable-freetype-4bit-antialias-hack

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a FreeType2 támogatást (kísérleti jellegű).

--with-gd[=DIR]

PHP 3: Beépíti a GD támogatást (a DIR a GD telepítési könyvtára).

PHP 4: Beépíti a GD támogatást (a DIR a GD telepítési könyvtára). Állítsd a DIR értékét shared-re, ha külső PHP modulként szeretnéd lefordítani, vagy állítsd shared,DIR-re, ha külső modulként fordítod, és megadod a könyvtárat.

--without-gd

PHP 3, PHP 4: Kikapcsolja a GD támogatást.

--with-imagick[=DIR]

PHP 3: Beépíti az ImageMagick támogatást. A DIR a telepítési könyvtár, ha elhagyod, a PHP megpróbálja megtalálni. [kísérleti jellegű]

PHP 4: Ez az opció nem áll rendelkezésre a PHP 4-ben.

--with-jpeg-dir[=DIR]

PHP 3: A jpeg könyvtár a pdflib 2.0-hoz.

PHP 4: A jpeg könyvtár a pdflib 3.x-hez.

--with-png-dir[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A png konyvtár a pdflib 3.x-hez.

--enable-t1lib

PHP 3: Engedélyezi a t1lib támogatást.

PHP 4: Ez az opció nem áll rendelkezésre; használd helyette a --with-t1lib paramétert!

--with-t1lib[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre; használd helyette az --enable-t1lib paramétert!

PHP 4: Beépíti a T1lib támogatást.

--with-tiff-dir[=DIR]

PHP 3: A tiff könyvtár a pdflib 2.0-hoz.

PHP 4: A tiff könyvtár a pdflib 3.x-hez.

--with-ttf[=DIR]

PHP 3, PHP 4: Beépíti a FreeType támogatást.

--with-xpm-dir[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Az xpm könyvtár a gd-1.8+-hoz.

Máshova nem sorolható

Ezek besorolásra kerülnek a jövőben további kategóriákba, értelemszerűen.

`--with-gmp`

PHP 3, PHP 4 : Beépíti a GMP támogatást.

`--disable-bcmath`

PHP 3: Fordítás BC tetszőleges pontosságú matematikai függvények nélkül. Ezek a függvények lehetőséget adnak olyan nagy számok kezelésére is, amik kiesnek az egész és lebegőpontos számok értelmezési tartományából. Lásd a BCMath tetszőleges pontosságú matematikai függvények című részt.

PHP 4: Ez az opció nem áll rendelkezésre, mivel a bcmath nem fordul be automatikusan; használd az --enable-bcmath paramétert a bekapcsoláshoz!

`--disable-display-source`

PHP 3: Forráskódmegjelenítési támogatás kikapcsolása.

PHP 4: Ez az opció nem áll rendelkezésre a PHP 4-ben.

`--disable-libtool-lock`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Lock-olás letiltása (gondot okozhat párhuzamos futásnál).

`--disable-pear`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A PEAR telepítésének kikapcsolása.

`--disable-pic`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A PIC kikapcsolása a megosztott modulokra.

`--disable-posix`

PHP 3: Ez az opció nem áll rendelkezésre; használd helyette a --without-posix paramétert!

PHP 4: A POSIX-szerű funkciók kikapcsolása.

`--enable-pcntl`

PHP 3: nem használható PHP 3-ban.

PHP 4: Engedélyezi a processz vezérlő függvényeket. (fork, waitpid, signal, stb.)

`--disable-rpath`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Addícionális futásidejű path átadás kikapcsolása.

`--disable-session`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A session támogatás kikapcsolása.

`--enable-bcmath`

PHP 3: Ez az opció nem áll rendelkezésre, mivel a bcmath automatikusan befordul; használd a --disable-bcmath paramétert a kikapcsoláshoz.

PHP 4: A BC stílusú tetszőleges pontosságú matematikai függvények engedélyezése. Olvasd el a README-BCMATH fájlt további útmutatáshoz. Ezek a függvények lehetőséget adnak olyan nagy számok kezelésére is, amik kiesnek az egész és lebegőpontos számok értelmezési tartományából. Lásd a BCMath tetszőleges pontosságú matematikai függvények című részt.

`--enable-c9x-inline`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Engedélyezi a C9x-inline semantics funkciót.

`--enable-calendar`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Engedélyezi a naptárkonvertálások támogatását.

`--enable-debug`

PHP 3, PHP 4: Debug szimbólumok engedélyezése.

`--enable-debugger`

PHP 3: Távoli debug funkciók befordítása.

PHP 4: Ez az opció nem áll rendelkezésre a PHP 4-ben.

`--enable-discard-path`

PHP 3, PHP 4: Ha ez engedélyezett, a PHP CGI állomány biztonságosan a web könyvtáron kívül helyezhető, és nem fogja tudni senki megkerülni a .htaccess biztonságot.

`--enable-dmalloc`

PHP 3, PHP 4: A dmalloc engedélyezése.

`--enable-exif`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Az exif támogatás engedélyezése.

`--enable-experimental-zts`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Ez feltehetően működésképtelenné teszi a PHP-det.

`--enable-fast-install[=PKGS]`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Gyors telepítésre optimalizálás [alapbeállítása yes]

`--enable-force-cgi-redirect`

PHP 3, PHP 4: A szerveren belüli átirányítások biztonsági ellenőrzésének bekapcsolása. Ezt kapcsold be, ha CGI módú futtatható fájlt készítesz Apache szerverrel.

`--enable-inline-optimization`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Ha sok memóriád van, és gcc-t használsz, érdemes ezt kipróbálnod.

`--enable-libgcc`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Engedélyezi a libgcc explicit linkelést.

`--enable-maintainer-mode`

PHP 3, PHP 4: Engedélyez olyan make szabályokat és függőségeket, amik többnyire nem hasznosak (sőt néha problematikusak) az átlagos telepítő számára.

`--enable-mbstr-enc-trans`

PHP 4: engedélyezi a http bemeneti karakterek automatikus felismerését és multi-byte-os karakterkódolásra fordítását.

Figyelem

Ez a kapcsoló csak PHP 4.0.6 vagy magasabb verziótól érhető el.

`--enable-mbstring`

PHP 4: engedélyezi a multi-byte karakterkódoláshoz kapcsolodó függvényeket.

Figyelem

Ez a kapcsoló csak PHP 4.0.6 vagy magasabb verziótól érhető el.

`--enable-memory-limit`

PHP 3, PHP 4: Memórialimit támogatás engedélyezése (alapbeállításban ez nincs bekapcsolva).

`--enable-safe-mode`

PHP 3, PHP 4: Safe mode engedélyezése alapbeállításban.

`--enable-satellite`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: CORBA támogatás engedélyezése Satellite-on keresztül (szükséges hozzá ORBit).

`--enable-shared[=PKGS]`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Megosztott modulok készítése [alapbeállítása yes]

`--enable-sigchild`

PHP 3, PHP 4: A PHP saját SIGCHLD kezelőjének engedélyezése.

`--enable-static[=PKGS]`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Statikus modulok készítése [alapbeállítása yes]

`--enable-sysvsem`

PHP 3, PHP 4: Engedélyezi a System V szemafor támogatást.

`--enable-sysvshm`

PHP 3, PHP 4: Engedélyezi a System V megosztott memória támogatást.

`--enable-trans-sid`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Engedélyezi az automatikus session id továbbítást.

`--with-cdb[=DIR]`

PHP 3, PHP 4: Beépíti a CDB támogatást.

--with-config-file-path=PATH

PHP 3: Beállítja a könyvtárat, ahol a php3.ini fájl keresendő. Alapbeállításban `/usr/local/lib`

PHP 4: Beállítja a könyvtárat, ahol a `php.ini` fájl keresendő. Alapbeállításban `/usr/local/lib`.

--with-cpdflib[=DIR]

PHP 3: Beépíti a ClibPDF támogatást. A DIR a ClibPDF telepítési könyvtára, alapbeállításban /usr/local.

PHP 4: Beépíti a cpdflib támogatást (szükséges hozzá cpdflib >= 2). A DIR a cpdfllib telepítési könyvtára, alapbeállításban /usr.

--with-esoob[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti az Easysoft OOB támogatást. A DIR az OOB telepítési könyvtára, alapbeállításban /usr/local/easysoft/oob/client.

--with-exec-dir[=DIR]

PHP 3, PHP 4: Csak ebben a könyvtárvban engedélyezi a futtatást safe mode esetén. Alapbeállításban /usr/local/php/bin.

--with-fdftk[=DIR]

PHP 3, PHP 4: Beépíti az fdftk támogatást. A DIR az fdftk telepítési könyvtára, alapbeállításban /usr/local.

--with-gnu-ld

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A C fordító GNU ld-t használ [alapbeállítása no]

--with-icap[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti az ICAP támogatást.

--with-imap[=DIR]

PHP 3, PHP 4: Beépíti az IMAP támogatást. A DIR az IMAP include és c-client.a könyvtára.

--with-imsp[=DIR]

PHP 3: Beépíti az IMSP támogatást (a DIR az IMSP’s include és libimsp.a könyvtára).

PHP 4: Ez az opció nem áll rendelkezésre a PHP 4-ben.

--with-java[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a Java támogatást. A DIR a JDK telepítési könyvtára. Ez csak megosztott PHP modulként fordítható.

--with-kerberos[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A Kerberos támogatás beépítése az IMAP-be.

--with-mcal[=DIR]

PHP 3, PHP 4: Beépíti az MCAL támogatást.

--with-mcrypt[=DIR]

PHP 3, PHP 4: Beépíti az mcrypt támogatást. A DIR az mcrypt telepítési könyvtára.

--with-mhash[=DIR]

PHP 3, PHP 4: Az mhash támogatás beépítése. A DIR az mhash telepítési könyvtára.

--with-mm[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti az mm támogatást a session tárolásra.

--with-mod_charset

PHP 3, PHP 4: Engedélyezi a transzfer táblákat a mod_charset-hez (Rus Apache).

--with-pdflib[=DIR]

PHP 3: Beépíti a pdflib támogatást (0.6-os és 2.0-s verzióval tesztelve). A DIR a pdflib telepítési könyvtára, alapbeállításban `/usr/local`.

PHP 4: Beépíti a pdflib 3.x/4.x támogatást. A DIR a pdflib telepítési könyvtára, alapbeállításban `/usr/local`.

PHP 4 és PDFlib 3.x/4.x megköveteli, hogy legyen JPEG és TIFF könyvtárak a rendszeredben. Amikor PDFlib támogatással fordítasz, használd a --with-jpeg-dir és --with-tiff-dir konfigurációs beállításokat. A --with-png-dir és a --with-zlib-dir beállításokkal pedig megadhatod, hogy a PDFlib támogatás PNG és Zlib támogatással együtt épüljön fel.

--enable-shared-pdflib

PHP 3, PHP 4: a pdflib mint shared library-t aktivizálja.

--with-readline[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a readline támogatást. A DIR a readline telepítési könyvtára.

--with-regex=TYPE

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Regex könyvtár típusa: system, apache vagy php

--with-servlet[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a servlet támogatást. A DIR a JSDK telepítési könyvtára. Ez az SAPI elvárja, hogy a Java modul megosztott formában legyen feltelepítve.

--with-ming

PHP 3: nem elérhető PHP 3-ban

PHP 4: Beépíti a Ming féle Flash 4 támogatást.

--with-swf[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti az swf támogatást.

--with-system-regex

PHP 3: A beépített regex könyvtár kikapcsolása.

PHP 4: (megszüntetve) A rendszer regex könyvtárának használata.

--with-tsrm-pth[=pth-config]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A GNU Pth használata.

--with-tsrm-pthreads

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: POSIX thread-ek használata (alapbeállítás)

--with-x

PHP 3: Az X Window System használata.

PHP 4: Ez az opció nem áll rendelkezésre a PHP 4-ben.

--with-bz2[=DIR]

PHP 4: Beépíti a bzip2 támogatást. A DIR paraméter a bzip2 telepítési könyvtára.

--with-zlib-dir[=DIR]

PHP 3: A zlib könyvtár a pdflib 2.0-hoz, vagy a zlib támogatás beépítése.

PHP 4: A zlib könyvtár a pdflib 3.x-hez, vagy a zlib támogatás beépítése.

--with-zlib[=DIR]

PHP 3, PHP 4: A zlib támogatás beépítése (szükséges a zlib >= 1.0.9). A DIR a zlib telepítési könyvtára, alapbeállításban /usr.

--with-zip[=DIR]

PHP 4: Beépíti a zip támogatást (a szükséges zziplib verziószám >= 0.10.6). A DIR a zziplib telepítési könyvtára, alapbeállításban /usr/local.

A legfrissebb zziplib verziót a http://zziplib.sourceforge.net/ címről tudod letölteni.

--without-pcre-regex

PHP 3: A Perl kompatibilis reguláris kifejezések kikapcsolása.

PHP 4: A Perl kompatibilis reguláris kifejezések kikapcsolása. Használd a --with-pcre-regex=DIR opciót, hogy megadd a PCRE include és library fájlok elérési útját, ha nem a beépített konyvtárat használod.

--without-posix

PHP 3: A POSIX funkciók kikapcsolása.

PHP 4: Ez az opció nem áll rendelkezésre; használd helyette a --disable-posix paramétert!

--enable-overload

PHP 3: nem elérhető PHP 3-ban

PHP 4: Engedélyezi objektum tulajdonságok és metódusok túlterhelését (overloading support).

Hálózati

--with-curl[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a CURL támogatást.

--enable-ftp

PHP 3: Ez az opció nem áll rendelkezésre; használd helyette a --with-ftp paramétert!

PHP 4: Engedélyezi az FTP támogatást.

--with-ftp

PHP 3: Engedélyezi az FTP támogatást.

PHP 4: Ez az opció nem áll rendelkezésre; használd helyette az --enable-ftp paramétert!

--disable-url-fopen-wrapper

PHP 3, PHP 4: Az URL-felismerő fopen komponensek kikapcsolása.

Figyelem

Ez a beállítási lehetőség csak a 4.0.3-as és előbbi PHP verziókban található meg. A későbbiekben az *allow_url_fopen* INI paraméter szabályozza ezt a működést. Ezért nem kell fordításidőben dönteni erről a fukcióról.

--with-mod-dav=DIR

PHP 3, PHP 4: Beépíti a DAV támogatást az Apache mod_dav segítségéve. A DIR a mod_dav telepítési könyvtára (Csak Apache modul verzióban!)

--with-openssl[=DIR]

PHP 3, PHP 4: Beépíti az SNMP OpenSSL támogatását.

--with-snmp[=DIR]

PHP 3, PHP 4: Beépíti az SNMP támogatást. A DIR az SNMP telepítési könyvtára, alapbeállításban számos tipikus helyen keresi a configure. Állítsd a DIR értékét shared-re, ha külső PHP modulként szeretnéd lefordítani, vagy állítsd shared,DIR-re, ha külső modulként fordítod, és megadod a könyvtárat.

--enable-ucd-snmp-hack

PHP 3, PHP 4: Engedélyezi az UCD SNMP hack-et.

--enable-sockets

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Engedélyezi a socket támogatást.

--with-yaz[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a YAZ támogatást (ANSI/NISO Z39.50). A DIR a YAZ bin telepítési könyvtára.

--enable-yp

PHP 3: Ez az opció nem áll rendelkezésre; használd helyette a --with-yp paramétert!

PHP 4: Beépíti az YP támogatást.

--with-yp

PHP 3: Beépíti az YP támogatást.

PHP 4: Ez az opció nem áll rendelkezésre; használd helyette az --enable-yp paramétert!

--with-mnogosearch

PHP 3, PHP 4: Beépíti az mnoGoSearch támogatást.

PHP viselkedését szabályozó

--enable-magic-quotes

PHP 3, PHP 4: Engedélyezi alapbeállításban a magic quotes szolgáltatást.

--disable-short-tags

PHP 3, PHP 4: Kikapcsolja alapbeállításban a rövid <? kezdő HTML tag-et.

--enable-track-vars

PHP 3: Engedélyezi a GET/POST/Cookie változók követését (asszociatív tömbök előállítását).

PHP 4: Ez az opció nem áll rendelkezésre a PHP 4-ben; a PHP 4.0.2-es, verziótól kezdve a track_vars szolgáltatás mindig be van kapcsolva.

Szerver

--with-aolserver-src=DIR

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Az AOLserver forrásának könyvtárát.

--with-aolserver=DIR

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A telepített AOLserver könyvtárát.

--with-apache[=DIR]

PHP 3, PHP 4: Apache modul fordítása. A DIR a felső-szintű Apache könyvtár, alapbeállításban /usr/local/etc/httpd.

--with-apxs[=FILE]

PHP 3, PHP 4: Megosztott Apache modul fordítása. A FILE az Apache apxs elérési útja, opcionális; alapbeállításban apxs.

--enable-versioning

PHP 3: A Solaris 2.x és a Linux által nyújtott versioning és scoping szolgáltatások használata.

PHP 4: Csak a szükséges szimbólumok exportálása. Lásd az INSTALL fájlt további információkért.

--with-caudium[=DIR]

PHP 3: nem elérhető PHP 3-ban

PHP 4: PHP-t Pike modulként építi fel a Caudium webszerveréhez. DIR a Caudium alapkönyvtára. Ha nincs megadva könyvtár, akkor a $prefix/caudium/server -t fogja használni, ahol a prefix a --prefix kapcsolóval szabályozható, ez alapértelmezés szerint a /usr/local.

--with-fhttpd[=DIR]

PHP 3, PHP 4: fhttpd modul fordítása. A DIR az fhttpd forrás könyvtára, alapbeállításban /usr/local/src/fhttpd.

--with-nsapi=DIR

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A telepített Netscape könyvtára.

--with-phttpd=DIR

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4:

--with-pi3web=DIR

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A PHP fordítása a Pi3Web szerver moduljaként.

--with-roxen=DIR

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A PHP Pike modulként való fordítása. A DIR a Roxen könyvtára, általában /usr/local/roxen/server.

--enable-roxen-zts

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A Roxen fordítása Zend Thread Safety kóddal.

--with-thttpd=SRCDIR

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4:

--with-zeus=DIR

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: A PHP fordítása Zeus ISAPI modulként.

Szöveg és nyelv

--with-aspell[=DIR]

PHP 3, PHP 4: Beépíti az ASPELL támogatást.

--with-gettext[=DIR]

PHP 3, PHP 4: Beépíti a GNU gettext támogatást. A DIR a gettext telepítési könyvtára, alapbeállításban /usr/local.

--with-iconv[=DIR]

PHP 3: nem elérhető PHP 3-ban.

PHP 4: Beépíti az iconv támogatást.

--with-pspell[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a PSPELL támogatást.

--with-recode[=DIR]

PHP 3: Include GNU recode support.

PHP 4: Beépíti a recode támogatást. A DIR a recode telepítési könyvtára.

--enable-shmop

PHP 3, PHP 4 : shmop támogatás aktivizálása.

XML

--with-dom[=DIR]

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a DOM támogatást (szükséges libxml >= 2.0). A DIR a libxml telepítési könyvtára, alapbeállításban `/usr`

`--enable-sablot-errors-descriptive`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Engedélyezi a leíró jellegű hibaüzeneteket.

`--with-sablot[=DIR]`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a Sablotron támogatást.

`--enable-wddx`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban.

PHP 4: Beépíti a WDDX támogatást.

`--disable-xml`

PHP 3: Ez az opció nem áll rendelkezésre a PHP 3-ban, mivel az XML funkciók nem kerülnek beépítésre alapbeállításban. Használd helyette a --with-xml paramétert a bekapcsoláshoz!

PHP 4: Kikapcsolja a beépített expat könyvtár XML támogatását.

`--with-xml`

PHP 3: Beépíti az XML támogatást.

PHP 4: Ez az opció nem áll rendelkezésre a PHP 4-ben, mivel az XML funkciók befordításra kerülnek alapbeállításban. Használd helyette a --disable-xml paramétert, hogy kikapcsold!

Telepítés Windows rendszerekre

Ez a fejezet a Windows 95/98/Me és a Windows NT/2000/XP rendszerekre történő telepítésekre vonatkozik. A PHP nem működik 16 bites környezetben mint például a Windows 3.1. Bizonyos esetekben a támogatott Windows környezetekre Win32 néven hivatkozunk.

A PHP Windows-ra telepítésének két fő módja van: vagy a kézi vagy a InstallShield telepítővarázslóval történő.

Ha van Microsoft Visual Studio telepítve a rendszereden, akkor megpróbálhatod a PHP-t az eredeti forrásból lefordítani.

Amint sikerült a PHP-t telepítened a Windows-odra, különböző kiterjesztéseket is betölthetsz további funkcionalitások eléréséhez.

Windows InstallShield telepítés

A Windows PHP telepítőprogram letölthető a http://www.php.net/ címről, ez a PHP CGI változatát telepíti is beállítja az IIS, PWS és Xitami szervereket is (ha ilyet használsz). Azt is vedd figyelembe, hogy az InstallShield telepítőprogrammal nagyon könnyű életre kelteni a PHP-t, de ez nagyon sok szempontból korlátozott változat, például a kiterjesztések automatikus telepítését nem végzi el.

Először telepítsd a választott szervert a gépedre, és ellenőrizd, hogy jól működik-e.

Futtassd a telepítő exe fájlt, és kövesd a varázsló által adott utasításokat. Kétféle telepítés közül választhatsz - a standard telepítés jól használható alapbeállításokat ad, az advanced kérdéseket tesz fel (, amelyekre tudni kell válaszolni :) .

A telepítés varázslója elég információt gyűjt ahhoz, hogy elvégezhesse a `php.ini` fájl beállítását és konfigurálja a szervert a PHP számára. IIS esetén, vagy NT Workstation alatt PWS használatakor az összes ponton fellelhető script map beállítást megmutatja, és kiválaszthatod, hogy mely pontokra kerüljön be a PHP támogatás.

Mikor a telepítés befejeződött, a varázsló informál arról, hogy szükséges-e a rendszer, ill. a szerver újraindítása, vagy rögtön elkezdheted a munkát a PHP-vel.

> **Figyelem**
>
> Légy tekintettel arra, hogy a PHP ezen telepítési módja nem biztonságos. Ha biztonságos PHP-ra vágysz, akkor jobban teszed, ha a kézikönyv további fejezeteit is elolvasod, és minden beállítást körültekintően elvégzel. Ez az automatikus telepítő egy azonnal használható PHP-t varázsol a gépedre, de nem online szerverekre szánták.

Kézi telepítés lépései

Ez a fejezet segít a PHP kézi telepítésében és Windows-os webszerverek konfigurálásában. A zippelt bináris disztribuciót kell letölteni ehhez a http://www.php.net/ címről. Ezt az útmutatót eredetileg Bob Silva (mailto:bob_silva@mail.umesd.k12.or.us) készítette. Az eredeti verzió angolul megtalálható a http://www.umesd.k12.or.us/php/win32install.html.

Ez a leírás a következő szerverek manuális PHP telepítését fedi le:

- Personal Web Server (friss verzió javasolt)
- Internet Information Server 3 vagy 4
- Apache 1.3.x
- Omni HTTPd 2.0b1 vagy újabb
- Oreilly Website Pro
- Xitami
- Netscape Enterprise Server, iPlanet

A Windows-os PHP4 disztribúciók kétféle formában kaphatók - CGI futtató (php.exe) és számos SAPI modulként (például php4isapi.dll). Az utóbbi elég új a PHP 4-ben, és jelentős teljesítményjavulást és néhány új lehetőséget biztosít az előzőhöz képest.

> **Figyelem**
>
> Mindazonáltal, figyelj arra, hogy a SAPI modulok *NEM* tekinthetők még kész minőségűnek. Különösen az ISAPI modul használatakor valószínűleg súlyos megbízhatósági problémákkal fogsz találkozni főleg Win2K-nál régebbi környezetekben. Tanúja lehetsz számos 500-as kódú szerver hibának és néhány más gyenge modultól is szenvedhetsz. Figyelmeztettünk!
>
> Ennek az az oka, hogy a SAPI modulokban a PHP újdonsült "szál-biztos" (thread-safe) változata fut, amelyet még nem teszteltek és nyúztak eleget ahhoz, hogy teljesen stabilnak nyilvánítsák, és tartalmaz néhány ismert hibát is. Másrészről, néhányan nagyon jó eredményekről számoltak be a SAPI modulok használata kapcsán, még akkor is ha, ha jelen pillanatban nem tudunk arról, hogy bárki is üzemelő webhelyén ezeket használná. Röviden: ha abszolút stabilitás kell, akkor a CGI változat stabilitása és a SAPI modulok teljesítménybeli különbsége közt kell döntened.

Ha Windows 95 alatt akarod használni valamelyik SAPI modult, akkor győződj meg arról, hogy letöltötted a DCOM frissítést a Microsoft DCOM oldaláról (http://download.microsoft.com/msdownload/dcom/95/x86/en/dcom95.exe). ISAPI modulokhoz, egy ISAPI 4.0-képes Web szerverre lesz szükséged (IIS 4.0, PWS 4.0 és IIS 5.0 alatt tesztelve). IIS 3.0 *NEM* használható. Elképzelhető, hogy le kell töltened és telepítened a Windows NT 4.0 Option Pack with IIS 4.0-t, ha natív PHP támogatást akarsz.

A következő lépéseket minden telepítés esetén a szerver specifikus beállítások előtt kell végrehajtani.

- Tömörítsd ki a disztribúciós fájlt egy tetszőleges könyvtárba. A `c:\php\` pl. jó választás.
- Biztosítanod kell, hogy a PHP az általa használandó DLL-eket meg is találja. Hogy milyen DLL-ekre vonatkozik ez, az függ attól, hogy a milyen webszervert használsz és, hogy azon a PHP modulként vagy CGIüként fog futni. A `php4ts.dll` minden esetben szükséges. Ha szerver modulként (pl. Apache vagy ISAPI) használod, akkor a `sapi` alkönyvtárból a megfelelő DLL kell. Hogy ezek a DLL-ek elérhetőek legyenek, bemásolhatod őket a rendszerkönyvtárba (pl. `winnt/system32` vagy `windows/system`), vagy rakd egy könyvtárba a fő PHP állománnyal (pl. php.exe, php4apache.dll).

  A PHP bináris, a SAPI modulok és néhány kiterjesztés használata külső DLL-ektől is függ. Bizonyosodj meg róla, hogy ezek a Windows PATH egyik könyvtárában vannak. A legjobb, ha ezeket mind a rendszer könyvtárba másolod, ami általában:
  c:\windows\system ;Windows 9x/Me esetén
  c:\winnt\system32 ;Windows NT/2000/XP esetén
  A másolandó fájlok:
  ’php4ts.dll’, ha már létezik, írd felül
  A disztribúció ’dlls’ könyvtárának fájlai. Ha már a rendszeredben ezek installálva vannak, akkor csakis akkor írd fel

  Töltsd le a legfrissebb, a saját platformodnak megfelelő Microsoft Data Access Components (MDAC) csomago(ka)t, különösen ha Microsoft Windows 9x/NT4-t használsz. MDAC elérhető a http://www.microsoft.com/data/ címen.

- Másold be a kiválasztott ini fájlt (ld lejjebb) Windows 9x/Me esetén a ’%WINDOWS%’ könyvtárba vagy Windows NT/2000/XP esetén a ’%SYSTEMROOT%’ könyvtárba, és nevezd át `php.ini`-re. A ’%WINDOWS%’ vagy ’%SYSTEMROOT%’ könyvtárak: typically:
  c:\windows ; Windows 9x/Me esetén
  c:\winnt vagy c:\winnt40 ; NT/2000/XP szerverek esetén

Két ini fájl került bele a zip-be: `php.ini-dist` és `php.ini-optimized`. Azt tanácsoljuk, hogy a `php.ini-optimized`-t használd, mert azokban az alapértékek úgy kerültek meghatározásra, hogy minél jobb teljesítménnyel és nagyobb védelemben fussanak a programjaid. A legjobb, ha áttanulmányozod az egész ini beállítások fejezetet, és magad határozod meg az összes bejegyzést. Ha a legnagyobb biztonságot szeretnéd eléni, akkor ez a járandó út, bár a PHP ezekkel php.ini-kben megadott alapbeállításokkal is ragyogóan működik.

- Szerkeszd át a saját 'php.ini' fájlodat:
  - Az 'extension_dir' bejegyzést át kell írni, hogy az arra a könyvtárra mutasson, ahol a 'php_*.dll' fájlok vannak, pl.: 'c:\php\extensions'.
  - Ha OmniHTTPd használsz, akkor a következő lépést hagyd ki. Állítsd be a 'doc_root'-ot, hogy az a webszervered document_root könyvtárára mutasson, például c:\apache\htdocs vagy c:\webroot
  - Válaszd ki, mely kiterjesztéseket töltse be a PHP induláskor. Nézd át a Windows kiterjesztések c. szakaszt, hogy melyeket kell telepíteni és melyek jönnek együtt a PHP disztribúcióval. Először érdemes tesztelni, hogy működik-e a PHP mindenféle kiterjesztés engedélyezése nélkül is.
  - PWS és IIS alatt, beállíthatod a `browscap.ini` fájlt, hogy az Windows 9x/Me esetén mutasson a 'c:\windows\system\inetsrv\browscap.ini'-re és NT/2000/XP Server esetén 'c:\winnt\system32\inetsrv\browscap.ini'-re.
  - Figyelj, hogy a Windows-os disztribúció `mibs` könyvtárában SNMP támogatáshoz szükséges fájlok vannak. Ezt a könyvtárat a `DRIVE:\usr\mibs`-be másold át (`DRIVE` az a meghajtó, ahová a PHP-t telepítetted.)

Fordítás forrásból

Mielőtt belevágunk, érdemes megválaszolni azt a kérdést, hogy: "Miért olyan nehéz lefordítani Windows alatt?". Két indokra vezethető vissza:

1. A Windows (még) nem rendelkezik fejlesztők olyan nagy lélekszámú csapatával, akik szabadon szeretnék megosztani egymás közt forrásaikat. Ennek közvetlen következménye, hogy az ilyen fejlesztésekhez szükséges, elengedhetetlen infrastruktúrális beruházások még nem történtek. Nagyjából a hozzáférhető eszközök mindegyikét Unix alól "hozták át", ezért nem kell meglepődni, ha ez az örökség időnként nagyon szembeötlő.
2. Az itt következő uasítások legtöbbje "csináld és felejtsd el" jellegű, ezért dőlj hátra, és próbáld meg a lehető leghűségesebben követni azokat.

Előkészületek

Mielőtt elkezded, le kell töltened...

- Kezdőknek, Cygwin toolkit-et a legközelebbi cygwin (http://sources.redhat.com/cygwin/download.html) tükröoldalról. A nagyon sok közkedvelt GNU segédprogramot tartalmaz, amelyek szükségesek a PHP létrehozásához.
- A maradék eszközöket töltsd le a PHP oldaláról: http://www.php.net/extra/win32build.zip.

- A PHP által használt DNS név-feloldó forrását is szerezd be a http://www.php.net/extra/bindlib_w32.zip címről. Ez a `win32build.zip` könyvtárbeli `resolv.lib` helyett kell.
- Ha még nincs zip-et kicsomagoló segédprogramod, akkor szükséged lesz egyre. Egy ingyenes: InfoZip (http://www.cdrom.com/pub/infozip/UnZip.html).

Végül kell, ugye, maga a PHP 4 forrása. A legutolsó fejlesztői változatra a szert tehetsz anonim CVS (http://www.php.net/anoncvs.php)-t használva. Ha a snapshot (http://snaps.php.net/)-ot vagy source (http://www.php.net/downloads.php) tarolt változatokat töltöd le, akkor azokat nemcsak untar-olni és ungzip-elni kell, de minden szimpla soremelést CR + LF -re kell cserélned a `*.dsp` és `*.dsw` fájlokban, mielőtt hozzálátnál használni őket a Microsoft Visual C++ -ban.

> **Megjegyzés:** Rakd a `Zend` és `TSRM` könyvtárakat a `php4` könyvtáron belülre, hogy a létrehozási lépések során megtalálja ezeket a fordító.

Az egész összerakása

- Kövesd az általad választott unzip kitömörítő program telepítési lépéseit.
- Indítsd el a `setup.exe`-t, és kövesd az ott leírtakat. Ha `c:\cygnus` útvonaltól eltérő helyre akarod telepíteni, akkor a be kell állítani a Cygwin környezeti változót arra a könyvtárra. Windows 95/98 esetén a környezeti változó beállítása történhet az autoexec.bat fájlban is, Windows NT esetén pedig, a My Computer => Control Panel => System párbeszédablakban válaszd ki az Environment nevű panelt.

> **Figyelem**
>
> Csinálj egy ideiglenes könyvtárat a Cygwin-nek, máskülönben sok parancs (különösen a bison) sikertelen lesz. Windows 95/98 esetén **`mkdir C:\TMP`**, Windows NT esetén **`mkdir %SystemDrive%\tmp`**.

- Tömörítsd ki a `win32build.zip` fájlt egy könyvtárba.
- Indítsd el a Microsoft Visual C++ -t, és válaszd ki a menüből a Tools => Options pontot. A párbeszédablakban válaszd ki a Directories panelt. Sorjában a legördülő listából válaszd ki az Executables, az Includes és Library files pontokat és győződj meg róla, hogy a hozzájuk tartozó listákban szerepel-e a `cygwin\bin`, `win32build\include` illetve `win32build\lib`. (A listához hozzáadni úgy kell, hogy a lista végén lévő üres sorra állsz, és elkezdesz gépelni.) Egy jellegzetes listabejegyzések valahogy így mutatnak:
  - `c:\cygnus\bin`
  - `c:\php-win32build\include`
  - `c:\php-win32build\lib`

  Nyomd le OK gombot, és lépj ki a Visual C++ -ból.

- Csinálj egy másik könyvtárat, ahová kitömöríted a `bindlib_w32.zip`-et. El kell döntened, hogy hibakereső szimbólumokkal (debug symbol) (bindlib - Win32 Debug) vagy anélkül (bindlib - Win32 Release) akarod fordítani a PHP-t. Állítsd be a megfelelő konfigurációt:
  - GUI -t használóknak: indítsd el a VC++ -t, nyisd meg a bindlib munkaterületetet (File => Open Workspace és válaszd ki a bindlib-et). Ezután: Build=>Set Active Configuration, és válaszd ki a kívánt konfigurációt, és végül: Build=>Rebuild All.
  - Parancssoros módban győződj meg arról, hogy a szükséges C++ környezeti változók legyenek beállítva vagy **vcvars.bat** le lett-e futtatva, és aztán add ki valamelyik alábbi parancsot:
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Debug"`**
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Release"`**
  - Ezen a ponton vagy a `Debug` vagy a `Release` alkönyvtárban kell, hogy legyen egy használható `resolv.lib`. Másold ezt rá a `win32build\lib` könyvtárban található ugyanilyen nevű fájlra.

Fordítás

A kezdésként legjobb egy önálló/CGI verziót fordítani.

- GUI -t használóknak: indítsd el a VC++ -t, nyisd meg a php4ts munkaterületetet (File => Open Workspace és válaszd ki a php4ts-t). Ezután: Build=>Set Active Configuration, és válaszd ki a kívánt konfigurációt, és végül: Build=>Rebuild All.
- Parancssoros módban győződj meg arról, hogy a szükséges C++ környezeti változók legyenek beállítva vagy **vcvars.bat** le lett-e futtatva, és aztán add ki valamelyik alábbi parancsot:
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Debug_TS"`**
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Release_TS"`**
  - Ennél a pontnál a `Debug_TS` vagy a `Release_TS` alkönyvtárban kell lennie egy használható `php.exe` fájlnak.

Ismételed meg a fenti lépéseket a `php4isapi.dsp`-re, amely a `sapi\isapi` könyvtárban található, ha a PHP-t Microsoft IIS -en akarod használni.

Windows-os kiterjesztések teleőítése

A webszerver és a PHP telepítése után valószínűleg néhány kiterjesztést is telepíteni akarsz. A következő táblázat az elérhető kiterjesztések zömét megtalálod. Megadhatod, hogy mely kiterjesztések töltődjenek be a PHP indulásakor, ha kitörlöd a megjegyzést jelentő pontosvesszőt (;) a 'extension=php_*.dll' mintájú sorok elejéről a `php.ini`-ben. A szkriptjeidben is betöltheted dinamikusan őket a dl() segítségével.

A PHP 4 kiterjesztések DLL-jei mind 'php_' taggal kezdődnek (és 'php3_' a PHP 3-éi). Ezáltal megelőzhető a PHP kiterjesztések és az általuk támogatott könyvtárak közti összevisszaság.

> **Megjegyzés:** PHP 4.0.6-ban BCMath, Calendar, COM, FTP, MySQL, ODBC, PCRE, Session, WDDX és XML támogatás *be van építve*. Semmilyen kiegészítő kiterjesztést nem kell ezek

használatához betölteni. Nézd meg a disztribúcióval jövő `README.txt` vagy `install.txt` fájlokban a beépített kiterjesztések listáját!

Megjegyzés: Néhány kiterjesztés további DLL-eket igényel a működésükhez. Néhány ezek közül a disztribúciós csomag része, a 'dlls' könyvtárban találod meg ezeket, de van néhány - pl. Oracle (php_oci8.dll) -, amelyhez szükségesek nincsenek a csomagban.

A csomagolt DLL-eket másold a 'dlls' könyvtárból valamilyen, a Windows PATH részét alkotó könyvtárba, a biztonságos helyek:

c:\windows\system ; Windows 9x/Me esetén
c:\winnt\system32 ; Windows NT/2000/XP esetén.

Ha már a rendszeredben ezek installálva vannak, akkor csakis akkor írd felül azokat, ha valami nem az elvártnak megfelelően működik. Mielőtt felülírnád a fájlokat, jó, ha csinálsz róluk biztonsági másolatot vagy átmásolod egy másik könyvtárba őket.

Táblázat 2-1. PHP kiterjesztések Windows alatt

Kiterjesztés	Leírás	Megjegyzés
php_bz2.dll	bzip2 tömörítő függvények	nincs
php_calendar.dll	Naptár átváltási függvények	PHP 4.0.3 óta beépített
php_cpdf.dll	ClibPDF függvények	nincs
php3_crypt.dll	Crypt függvények	-
php_ctype.dll	ctype féle függvények	nincs
php_curl.dll	CURL, Client URL library függvények	szükséges: libeay32.dll, ssleay32.dll (része a disztribúciónak)
php_cybercash.dll	Cybercash payment függvények	nincs
php_db.dll	DBM függvények	ellenjavallt, DBA-t használd helyette (php_dba.dll)
php_dba.dll	DBA: (dbm jellegű) adatbázis absztrakciós réteg függvényei	nincs
php_dbase.dll	dBase függvények	nincs
php3_dbm.dll	Berkeley DB2 eljáráskönyvtár	-
php_domxml.dll	DOM XML függvények	szükséges: libxml2.dll (része a disztribúciónak)
php_dotnet.dll	.NET függvények	nincs
php_exif.dll	EXIF fejlécek olvasása JPEG-ből	nincs
php_fbsql.dll	FrontBase függvények	nincs
php_fdf.dll	FDF: Forms Data Format függvények.	szükséges: fdftk.dll (része a disztribúciónak)
php_filepro.dll	filePro függvények	csak olvasható hozzáféréssel
php_ftp.dll	FTP függvények	PHP 4.0.3 óta beépített

Kiterjesztés	Leírás	Megjegyzés
php_gd.dll	GD eljáráskönyvtár képkezelő függvényei	nincs
php_gettext.dll	Gettext függvények	szükséges: gnu_gettext.dll (része a disztribúciónak)
php_hyperwave.dll	HyperWave függvények	nincs
php_iconv.dll	ICONV characterset conversion	szükséges: iconv-1.3.dll (része a disztribúciónak)
php_ifx.dll	Informix függvények	szükséges: Informix eljáráskönyvtárak
php_iisfunc.dll	IIS kezelési függvények	nincs
php_imap.dll	IMAP POP3 és NNTP függvények	PHP 3: php3_imap4r1.dll
php_ingres.dll	Ingres II függvények	szükséges: Ingres II libraries
php_interbase.dll	InterBase függvények	szükséges: gds32.dll (része a disztribúciónak)
php_java.dll	Java extension	szükséges: jvm.dll (része a disztribúciónak)
php_ldap.dll	LDAP függvények	szükséges: libsasl.dll (része a disztribúciónak)
php_mhash.dll	Mhash Functions	nincs
php_ming.dll	Ming függvények for Flash	nincs
php_msql.dll	mSQL függvények	szükséges: msql.dll (része a disztribúciónak)
php3_msql1.dll	mSQL 1 client	-
php3_msql2.dll	mSQL 2 client	-
php_mssql.dll	MSSQL függvények	szükséges: ntwdblib.dll (része a disztribúciónak)
php3_mysql.dll	MySQL függvények	PHP 4 óta beépített
php3_nsmail.dll	Netscape levelező függvények	-
php3_oci73.dll	Oracle függvények	-
php_oci8.dll	Oracle 8 függvények	szükséges: Oracle 8 kliens eljáráskönyvtárak
php_openssl.dll	OpenSSL függvények	szükséges: libeay32.dll (része a disztribúciónak)
php_oracle.dll	Oracle függvények	szükséges: Oracle 7 client kliens eljáráskönyvtárak
php_pdf.dll	PDF függvények	nincs
php_pgsql.dll	PostgreSQL függvények	nincs
php_printer.dll	Printer függvények	nincs
php_sablot.dll	XSLT függvények	szükséges: sablot.dll (része a disztribúciónak)
php_snmp.dll	SNMP get and walk függvények	csak NT -n!
php_sybase_ct.dll	Sybase függvények	szükséges: Sybase kliens eljáráskönyvtárak

Kiterjesztés	Leírás	Megjegyzés
php_yaz.dll	YAZ függvények	nincs
php_zlib.dll	ZLib tömörítő függvények	nincs

Szerverek - Apache

Ez a fejezet a PHP Apache szerverre telepítésével foglalkozik, mind a Unix, mind a Windows változatokkal.

A PHP telepítése UNIX-on futó Apache-hoz

Az alábbi 8. sor **configure** opcióihoz a Configure opciók teljes listájából választhatsz továbbiakat.

Példa 2-5. Gyors telepítési utasítások PHP 4-hez (Apache modul verzió)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-x.x.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. PHP 3 esetén: ./configure --activate-module=src/modules/php3/libphp3.a
    PHP 4 esetén: ./configure --activate-module=src/modules/php4/libphp4.a
13. make
14. make install

  Ehelyett a lépés helyett talán jobb, ha a httpd futtatható állományát
  a meglévőre rámásolod. Győződj meg róla, hogy előtte leállítottad
  a szervert!

15. cd ../php-x.x.x
16. PHP 3 esetén: cp php3.ini-dist /usr/local/lib/php3.ini
    PHP 4 esetén: cp php.ini-dist /usr/local/lib/php.ini

  Szerkeszheted az .ini állományt, beállítva az opciókat.
  Ha máshova szeretnéd tenni ezt a fájlt, add meg a
  --with-config-file-path=/eleresi/ut paramétert a 8. lépésben.

17. Szerkeszd a httpd.conf vagy srm.conf fájlt és add hozzá:

     PHP 3 esetén:   AddType application/x-httpd-php3 .php3
     PHP 4 esetén:   AddType application/x-httpd-php .php

  Itt bármilyen kiterjesztést választhatsz. A .php csak a
```

```
  mi javaslatunk. Akár a  .html kiterjesztést is megadhatod.

18. Használd a szokásos módszert az Apache szerver újraindítására.
    (Le kell állítanod és újra kell indítanod a szervert, nem csak
    újratölteni egy HUP vagy USR1 jellel.)
```

Az Apache telepítésétől és a Unix változattól függően, sok módja lehet a szerver leállításának és újraindításának. Itt néhány jellemző utasítást adunk közre, hogyan kell újraindítani a kiszolgálót, de ezek eltérhetnek apache/unix telepítéstől függően. Az `/ahol/van/`-t cseréld ki arra a könyvtárra, ahol ezek a programok vannak:

```
1. különféle Linux és SysV változatok esetén:
/etc/rc.d/init.d/httpd restart

2. apachectl szkripttel:
/ahol/van/apachectl stop
/ahol/van/apachectl start

3. httpdctl és httpsdctl (OpenSSL), hasonlóan az apachectl-hez:
/ahol/van/httpsdctl stop
/ahol/van/httpsdctl start

4. mod_ssl vagy más SSL kiszolgáló használatakor esetleg a kézi megoldás:
    a leállításra és újraindításra:

/ahol/van/apachectl stop
/ahol/van/apachectl startssl
```

Az apachectl és http(s)dctl állományok elérési útvonala gyakran eltérő. Ha tudsz használni `locate` vagy `whereis` vagy `which` parancsokat, akkor azokkal megkerestetheted ezeket a szervert irányító programokat.

A PHP Apache szerverrel történő fordításának néhány példája:

```
./configure --with-apxs --with-pgsql
```

Ez létre fog hozni egy `libphp4.so` megosztott modult, amit az Apache szerverbe egy LoadModule sorral lehet betölteni a `httpd.conf` fájlban. A PostgreSQL támogatás ebben az esetben beépített a `libphp4.so` modulba.

```
./configure --with-apxs --with-pgsql=shared
```

Ez szintén egy `libphp4.so` megosztott modult hoz létre az Apache számára, de egy `pgsql.so` modult is generál, amit később a PHP-be kell tölteni a `php.ini` extension direktívájával, vagy egy szkriptben a dl() függvénnyel.

```
./configure --with-apache=/az/apache/forras/eleresi/utja --with-pgsql
```

Ez létre fog hozni egy `libmodphp4.a` modult, egy `mod_php4.c`-t, és néhány kapcsolódó fájlt, és bemásolja az Apache forrás fába a `src/modules/php4` könyvtárba. Ezután az Apache fordítása az `--activate-module=src/modules/php4/libphp4.a` opcióval történik, és az Apache fordító rendszer elő fog állítani egy `libphp4.a` fájlt, és statikusan befordítja a `httpd` bináris állományba. A PostgreSQL támogatás közvetlenül a `httpd` futattható állomány része, tehát a végső eredmény egy egyedülálló `httpd` fájl, ami minden Apache és PHP funkciót tartalmaz.

```
./configure --with-apache=/az/apache/forras/eleresi/utja --with-pgsql=shared
```

Majdnem megegyezik az előzővel, azonban a PostgreSQL támogatás nem közvetlenül a `httpd` futtatható állományba épül be, hanem létrejön egy `pgsql.so` megosztott modul, és ezt később a PHP-be töltheted a `php.ini` extension beállítással, vagy közvetlenül a dl() függvénnyel egy PHP szkriptben.

Amikor eldöntöd, hogy melyik fordítási módszert választod, vedd számba mindeegyik lehetőség előnyeit és hátrányait is. A különálló objektumként való fordításnak megvan az az előnye, hogy lehetőséged van az Apache-ot külön lefordítani, és nem kell mindig újrafordítani, amikor lecseréled a PHP-t. Az Apache-ba építés (statikus módszer) előnye, hogy a PHP gyorsabban fog betöltődni és futni. További információkért lásd az Apache DSO támogatásról szóló oldalát (http://httpd.apache.org/docs/dso.html).

PHP telepítése Windows-on futó Apache alá

Két módja van a PHP Windows-on futó Apache 1.3.x alá telepítésének. Az egyik a CGI kezelőként futtatható php.exe, a másik az Apache modulként használandó dll. Mindkét esetben le kell állítanod az Apache-t, és az `srm.conf` vagy a `httpd.conf` fájlokat át kell szerkesztened, hogy Apache együttműködjön a PHP-val.

Kevés változata van a PHP beállításának Apache szerver alá, az alábbi verzió elég egyszerű a kezdők számára is. Nézz utána a további konfigurációs lehetőségeknek az Apache dokumentációdban.

Ha kicsomagoltad a PHP disztribúciót a c:\php\ könyvtárba a Kézi telepítés lépései fejezetnek megfelelően, akkor a következő sorokat kell beírnod az Apache konfigurációs fájlába, hogy CGI kezelőként futtassa PHP-t:

- `ScriptAlias /php/ "c:/php/"`
- `AddType application/x-httpd-php .php .phtml`

- `Action application/x-httpd-php "/php/php.exe"`

A második sorban már szerepel a `httpd.conf`-ban, csak ki van kommentezve. Miután átírtad a konfigurációs fájlt, ne felejtsd el, újraindítani a szervert, például Apache service-ként futtatásakor a `NET STOP APACHE` után egy `NET START APACHE` paranccsal, vagy használhatod a szokványos shortcut-okat.

Ha Apache modulként akarod használni a PHP-t, akkor a `php4ts.dll`-t másold a windows/system-be - Windows 9x/Me eseén vagy a winnt/system32 könyvtárba - Windows NT/2000/XP esetén (felülírva a régit). Ezután a következő két sort kell az Apache konfigurációs fájlába írni:

- `LoadModule php4_module c:/php/sapi/php4apache.dll`
- `AddType application/x-httpd-php .php .phtml`

> **Megjegyzés:** Apache 1.3.22 for Windows kezdő konfigurációs fájla (`httpd.conf-dist-win`) alapértelmezésben tartalmaz egy `ClearModuleList` direktívát. Ha ez a direktíva be van kapcsolva, akkor az `AddModule mod_php4.c` sort az `AddModule` listába kell tenni, mert máskülönben PHP nem lesz Apache modulként regisztrálva.

Ahhoz, hogy használhasd a szintaxis kiemelést, egyszerűen készíts egy PHP szkript fájlt és illeszd be ezt a kódot: `<?php show_source ("eredeti_php_szkript.php"); ?>`. Értelemszerűen helyettesítsd az `eredeti_php_script.php` részt a megmutatni kívánt forráskódot tartalmazó fájlal. (Ez az egyetlen módja ennek, mivel Windows-on nincs a .phps-hez hasonló lehetőség.)

> **Megjegyzés:** Windows-os Apache alatt minden fordított perjel (backslash) helyett szimpla perjelet kell használni egy elérési út meghatározásakor, pl. "c:\konyvtar\file.kit" helyett "c:/konyvtar/file.kit".

Szerverek - CGI/parancssori verzió

Alapbeállításban a PHP CGI programként fordul le. Ez létrehoz egy parancssorból használható értelmezőt, ami CGI feldolgozásra vagy nem webbel kapcsolatos PHP programozásra is használható. Ha egy olyan webszervert futtatsz, amelyhez a PHP modul szintű támogatással rendelkezik, akkor jobb teljesítmény eléréséhez használd inkább azt a módszert. A CGI verzió azonban lehetővé teszi az Apache-ot használóknak, hogy más-más PHP oldalakat más-más user-id-kkel futtassanak. Olvasd el a biztonságról szóló fejezetet, ha CGI-ként szeretnéd használni a PHP-t.

Tesztelés

Ha CGI programként fordítottad le a PHP-t, tesztelheted az eredményt azzal, hogy beírod **make test**. Mindig jól jön, ha leteszteled, mert így rögtön észlelhetsz olyan problémákat, amik esetleg csak később bukkantak volna fel.

Szintmérés (benchmarking)

Ha CGI programként fordítottad le a PHP 3-at, tesztelheted a sebességét azzal, hogy beírod **make bench**. Ha a "safe mode" be van kapcsolva, talán nem fog jól lefutni a benchmark, a megengedett 30 másodperc alatt. Ez azért van, mert a set_time_limit() nem használható safe mode-ban. Használd a max_execution_time konfigurációs beállítást, hogy megadhasd ezt az időt a szkriptjeidnek. A **make bench** nem veszi figyelembe a konfigurációs fájlt.

Megjegyzés: A **make bench** csak PHP 3-ban érhető el!

Szerverek - fhttpd

Ahhoz, hogy fhttpd modulként állítsd be a PHP-t, válaszolj "yes"-el a "Build as an fhttpd module?" kérdésre (ez a --with-fhttpd=*DIR* opció a configure-ban) és add meg az fhttpd könyvtárát. Ez alapbeállításban `/usr/local/src/fhttpd`. Ha fhttpd-t használsz, a PHP felépítése modulként jobb teljesítményt fog nyújtani, több beavatkozási lehetőséget és távoli futtatást biztosít.

Szerverek - Caudium

PHP 4-et Pike modulként építheted be a Caudium webszerveréhez, a PHP 3 nem támogatja ezt. Az alábbi egyszerű utasításokat követve lehet telepíteni a PHP 4-et Caudium alá.

Példa 2-6. Caudium telepítési utasítások

1. Győződj meg arról, hogy a PHP 4 telepítése előtt már telepítve lett a Caudium. A PHP 4 helyes működéséhez a Pike 7.0.268 vagy ennél újabb verziójára lesz szükséged. A példa kedvéért feltételezzük, hogy a Caudium a /opt/caudium/server/ könyvtárban van.
2. Lépj át php-x.y.z könyvtárba (x.y.z a PHP verziószáma).
3. ./configure --with-caudium=/opt/caudium/server
4. make
5. make install
6. Indítsd újra a Caudium szervert, ha jelenleg fut.
7. Lépj be a grafikus konfigurációs felületre és ott válaszd ki azt a "virtual server" pontot, amelyhez a PHP 4 támogatást akarsz adni.
8. Kattints az "Add Module"-ra és keresd meg és utána add hozzá a "PHP 4 Script Support" modult.
9. Ha a dokumentáció azt mondja, hogy ’PHP 4 interpreter isn’t available’, akkor nézd meg, biztosan újraindítottad a szervert. Ha már ellenőrizted az /opt/caudium/logs/debug/default.1 fájlt mindenféle <filename>PHP4.so</filename> -val összefüggő hiba kapcsán, akkor nézd meg azt is, hogy a <filename>caudium/server/lib/[pike-verzio]/PHP4.so</filename> létezik-e.
10. Állítsd be a PHP Script Support modult, ha szükséges.

Természetesen számos kiterjesztéssel együtt fordíthatod le a Caudium szerverhez is a PHP 4-t, a kimerítő felsorolásért a teljes konfigurációs beállítások listáját olvasd el.

> **Megjegyzés:** Ha MySQL témogatást is szeretnél, akkor figyelj arra, hogy a normál MySQL kliens kódot használd, máskülönben összeütközések léphetnek fel, ha a Pike is rendelkezik MySQL támogatással. Ezt a MySQL telepítési könyvtárának (install directory) megadásával teheted meg: --with-mysql.

Szerverek - IIS/PWS

Ez a fejezet az IIS szerverekre vonatkozó PHP telepítési útmutatókat tartalmazza, mind az IIS3, mind az IIS4 verziókhoz.

Windows 9x/NT/2000 és PWS/IIS 3

Ezen szervereken a konfigurálás a megadott REG fájl segítségével javasolt (pws-php4cgi.reg). Szerkesztheted ezt a fájlt, hogy a kiterjesztések és a PHP könyvtárait megadd. Vagy követheted az alábbi lépéseket, hogy elkészítsd ezt saját kezűleg.

Figyelem

Az alábbi lépések a windows registry-ben való közvetlen szerkesztést igénylik. Egyetlen hiba használhatatlanná teheti a rendszeredet! Mindenképpen készíts egy biztonsági másolatot, mielőtt bármi mást tennél. A PHP Team nem vállal felelősséget semmilyen kárért!

- Futtasd a Regedit-et.
- Keresd meg a `HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap` pontot.
- Az Edit menüben válaszd ki a `New->String Value` pontot.
- Írd be a fájl kiterjesztést, amit használni szeretnél, pl. `.php`
- Klikkelj kétszer az új szövegen és írd be a `php.exe` elérési útját, pl. `c:\php\php.exe %s %s`. A %s %s NAGYON fontos, enélkül a PHP nem fog működni.
- Ismételd ezeket a lépéseket az összes kiterjesztésre, amit használni szeretnél.

A következő lépések nem befolyásolják a web szerver üzembehelyezését, és csak akkor van rá szükség, ha a php szkriptjeidet parancssorból is szeretnéd futtatni - pl. `c:\myscripts\test.php`) - vagy a fájlböngészőben duplakattintásra elindítani azokat. Ugord át ezeket a lépéseket, ha azt akarod, hogy duplakattintásra inkább a szövegszerkesztődbe töltődjenek be a PHP szkriptek.

- Keresd meg a `HKEY_CLASSES_ROOT` pontot.
- Az Edit menüben válaszd a `New->Key` pontot.

- Nevezd el az új kulcsot az előzőekben megadott kiterjesztés nevére, pl. `.php`
- Válaszd ki az új kulcsot, aztán a jobb oldalon kattints kétszer a "default value" soron, és írd be, hogy `phpfile`.
- Ismételd az utóbbi lépést az összes kiterjesztésre, amit az előző részben beállítottál.
- Most hozz létre ismét egy új kulcsot (`New->Key`) a `HKEY_CLASSES_ROOT` alatt, és nevezd el `phpfile`-nak.
- Válaszd ki az új `phpfile` kulcsot, aztán a jobb oldalon kattints kétszer a "default value" soron, és írd be, hogy `PHP Script`.
- Kattints jobbgombbal a `phpfile` kulcson és válaszd ki a `New->Key` pontot. Nevezd az új kulcsot `Shell`-nek.
- Kattints jobbgombbal a `Shell` kulcson és válaszd ki a `New->Key` pontot. Nevezd az új kulcsot `open`-nek.
- Kattints jobbgombbal az `open` kulcson és válaszd ki a `New->Key` pontot. Nevezd az új kulcsot `command`-nak.
- Válaszd ki az új `command` kulcsot, aztán a jobb oldalon kattints kétszer a "default value" soron, és írd be, a `php.exe` elérési útját, pl. `c:\php\php.exe -q %1`. Ne felejtsd el a `%1`-et!).
- Lépj ki a Regedit-ből.
- Ha Windows alatt PWS szervert használsz, indítsd újra a gépet, hogy újratöltse a rendszer a registry-t.

PWS és IIS 3 használók így már rendelkeznek egy teljesen funkcionális rendszerrel. IIS 3 használóknak ajánlható Steven Genusa ötletes script map konfiguráló eszköze (http://www.genusa.com/iis/iiscfg.html).

Windows és 4-es vagy újabb PWS

Kétféle módon lehet a PHP-t 4-es vagy újabb verziójú PWS-hez installálni: ez egyik a CGI bináris, a másik az ISAPI modul dll használata.

Ha a CGI változat mellett döntesz, akkor:

- Szerkeszd át a csatolt `pws-php4cgi.reg` fájlt (sapi könyvtár), hogy abban a php.exe-re hivatkozás az installációnak megfelelő legyen. A perjeleket meg kell előzze egy maásik visszaperjel, pl.:
  `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map] ".php"="c:\\php\\php.exe"`
- A PWS Manager-ben, egy jobbegérgomb kattintás arra a könyvtárra, amelyhez PHP támogatást szeretnél rendelni, és válaszd ki a Properties menüpontot. Jelöld be az ’Execute’ négyzetet, és nyomd meg az OK-t.

Ha ISAPI modulként akarod használni, akkor:

- Szerkeszd át a csatolt `pws-php4isapi.reg` fájlt (sapi könyvtár), hogy abban a php4isapi.dll-re hivatkozás az installációnak megfelelő legyen. A perjeleket meg kell előzze egy maásik visszaperjel, pl.:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script
Map] ".php"="c:\\php\\sapi\\php4isapi.dll"
```

- A PWS Manager-ben, egy jobbegérgomb kattintás arra a könyvtárra, amelyhez PHP támogatást szeretnél rendelni, és válaszd ki a Properties menüpontot. Jelöld be az 'Execute' négyzetet, és nyomd meg az OK-t.

Windows NT/2000 és IIS 4

A PHP telepítése IIS 4-el felszerelt Windows NT/2000 szerverre a következőképpen történik:

- Az Internet Service Manager-ben (MMC) válaszd a Web Site kezdő könyvtárát.
- Nyisd meg a könyvtár Properties ablakát a jobb klikkre lenyíló menüből, és ott válaszd a Home Directory, Virtual Directory, vagy Directory fület.
- Kattints a Configuration gombra, majd az App Mappings fülre.
- Kattints az Add gombra, és az Executable box-ban írd be, hogy `c:\php-eleresi-ut\php.exe %s %s`. A %s %s MINDENFÉLEKÉPPEN szerepeljen, ha azt szeretnéd, hogy működjön a PHP.
- Az Extension box-ban, írd be a kiterjesztést, amit használni szeretnél a PHP szkriptjeidhez. Meg kell ismételned az 4-5 lépéseket minden kiterjesztésre, amit be szeretnél állítani. (`.php` és `.phtml` a szokásosak.)
- Állítsd be a megfelelő biztonságot az Internet Service Manager-ben. Ha az NT szerveren NTFS-t használsz adj futtatási jogokat I_USR_-nek arra a könyvtárra, ami a `php.exe` fájlt tartalmazza.

Szerverek - Netscape és iPlanet Enterprise

Ez a fejezet a PHP Netscape és iPlanet szerverekhez történő telepítéséről a felvilágosítást és néhány jó tanácsot, mind Sun Solaris és Windows platformokon.

PHP Netscape Enterprise Serverhez történő telepítéséről még többet olvashatsz a http://benoit.noss.free.fr/php/install-php4.html címen.

PHP telepítése Netscape-hez Sun Solaris-on

NES vagy iPlanet szerverekhez a PHP telepítésekor a megfelelő telepítési könyvtárat (install directory) kell megadni: --with-nsapi = *`DIR`* beállítás. Az alapértelmezés szerinti könyvtár általában a `/opt/netscape/suitespot/`. Lehetőleg olvasd el a `/php-xxx-version/sapi/nsapi/nsapi-readme.txt` fájlt is!

Példa 2-7. PHP telepítési példa Netscape-hez Sun Solaris-on

```
Sun Solaris 2.6-on futó Netscape Enterprise Server 3.6 telepítési utasítások
bhager@invacare.com -től
```

```
1. A következő csomagokat telepítsd a www.sunfreeware.com vagy más download oldalról:

    flex-2_5_4a-sol26-sparc-local
    gcc-2_95_2-sol26-sparc-local
    gzip-1.2.4-sol26-sparc-local
    perl-5_005_03-sol26-sparc-local
    bison-1_25-sol26-sparc-local
    make-3_76_1-sol26-sparc-local
    m4-1_4-sol26-sparc-local
    autoconf-2.13
    automake-1.4
    mysql-3.23.24-beta (ha mysql kell támogatás)
    tar-1.13 (GNU tar)

2. Győződj meg, hogy a PATH tartalmazza a helyes könyvtárakat tartalmazza:
    PATH=.:/usr/local/bin:/usr/sbin:/usr/bin:/usr/ccs/bin
    export PATH

3. gunzip php-x.x.x.tar.gz (ha .gz disztribúcióval van dolgod, egyébként ugorj a 4-
re)
4. tar xvf php-x.x.x.tar
5. cd ../php-x.x.x

6. A következő lépések előtt, győződj meg arról, hogy a Netscape szerver
   a /opt/netscape/suitespot/ könyvtárba lett telepítve. Ha nem, akkor a helyes
   útvonalat kell megadni:
   ./configure --with-mysql=/usr/local/mysql --with-nsapi=/opt/netscape/suitespot/ -
-enable-track-vars --enable-libgcc
7. make
8. make install
```

Az alaprendszer telepítése után olvasd el a megfelelő readme fájlokat, ha esetleg néhány további konfigurálási lépést kell tenned.

Elsősorban az LD_LIBRARY_PATH környezeti változóhoz kell néhány könyvtárat megadni, hogy a Netscape megtalálja a megosztott könyvtárcsomagokat (shared library). Ezt legegyszerűbb a Netscape-et indító szkriptben elhelyezni . Az indító szkript gyakran a `/utvonal/a/szerverhez/https-servername/start` könyvtárban található.

A konfigurációs fájlt is át kell írni, ami a `/utvonal/a/szerverhez/https-servername/config/` könyvtárban található.

Példa 2-8. Netscape Enterprise beállítási példa

```
Netscape Enterprise beállítási útmutató
Configuration Instructions for Netscape Enterprise Server
bhager@invacare.com -től

1. Vedd fel a következő sort a mime.types-ba:
    type=magnus-internal/x-httpd-php exts=php

2. Vedd fel a következő sort a obj.conf-ba, shlib függ az operációs rendszertől
    Unix alatt valami ehhez hasonló:
    /opt/netscape/suitespot/bin/libphp4.so.
```

```
    A következő sort a mime types init után kell beírni:
    Init fn="load-modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shli
    Init fn=php4_init errorString="Failed to initialize PHP!"

    <object name="default">
    .
    .
    .
    .#NOTE a következő sornak minden 'ObjectType' után és minden 'AddLog' sor után kell a
    Service fn="php4_execute" type="magnus-internal/x-httpd-php"
    .
    .
    </Object>


    <Object name="x-httpd-php">
    ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
    Service fn=php4_execute
    </Object>


    Hitelesítési (Authentication) beállítások

    PHP autentikáció nem használható együtt más autentikációval. Minden hite-
lesítés a saját
    PHP szkriptedhez kerül!!! A PHP hitelesítést a teljes szerverre a következő sor állít
    be:

    <Object name="default">
    AuthTrans fn=php4_auth_trans
    .
    .
    .
    .
    </Object>

    A PHP hitelesítést csak egy könyvtárra:

    <Object ppath="d:\path\to\authenticated\dir\*">
    AuthTrans fn=php4_auth_trans
    </Object>
```

Ha Netscape Enterprise 4.x -et használsz, akkor a következőket kell tenned:

Példa 2-9. Netscape Enterprise 4.x beállítási példa

```
A következő sorokat a mime types init után kell beírni, minden más az
előző példában leírthoz hasonló:
Graeme Hoose (GraemeHoose@BrightStation.com)

Init fn="load-modules" shlib="/path/to/server4/bin/libphp4.so" funcs="php4_init,php4_clos
Init fn="php4_init" LateInit="yes"
```

PHP telepítése Netscape-hez Windows-on

To Install PHP as CGI (for Netscape Enterprise Server, iPlanet, perhaps Fastrack), do the following:

- Másold be a `php4ts.dll` fájlt a systemroot könyvtárba - abba a könyvtárba, ahová a Windows-t installáltad.
- Parancssorból hozd létre a fájl hozzárendelést, ehhez a következő két sort kell begépelni:

```
assoc .php=PHPScript
ftype PHPScript=c:\php\php.exe %1 %*
```

- A Netscape Enterprise Administration Server-ben hozz létre egy üres shellcgi könyvtárat és töröld ki azon nyomban. (Ezzel a lépéssel 5 nagyon fontos sor adódik az obj.conf fájlhoz, amivel lehetővé válik, hogy a webszerver shellcgi szkripteket kezeljen.)
- A Netscape Enterprise Administration Server szakaszban hozz létre egy új mime type bejegyzést (Category: type, Content-Type: magnus-internal/shellcgi, File Suffix:php).
- Tedd meg ezt minden olyan web server példányra, amelyhez PHP támogatást akarsz adni!

A PHP CGI futtatóként való telepítéséről részletesebb leírásért lásd: http://benoit.noss.free.fr/php/install-php.html

PHP NSAPI-ként való installálásához (Netscape Enterprise Server-hez, iPlanet-hez, esetleg Fastrack-hez) a következőket kell tenni:

- Másold be a `php4ts.dll` fájlt a systemroot könyvtárba - abba a könyvtárba, ahová a Windows-t installáltad.
- Parancssorból hozd létre a fájl hozzárendelést, ehhez a következő két sort kell begépelni:

```
assoc .php=PHPScript
ftype PHPScript=c:\php\php.exe %1 %*
```

- A Netscape Enterprise Administration Server szakaszban hozz létre egy új mime type bejegyzést (Category: type, Content-Type: magnus-internal/x-httpd-php, File Suffix:php).
- Állítsd le a webszervert és szerkeszd át a `obj.conf` fájlt. Az Init szakasz végén helyezd el a következő két sort: (feltétlenül a mime type init rész után)

```
Init fn="load-modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shlib=
Init fn="php4_init" errorString="Failed to initialise PHP!"
```

- Az `<Object name="default">` szakaszban feltétlenül helyezd el a következő sort az összes 'ObjectType' után és még az összes 'AddLog' sor előtt:

```
Service fn="php4_execute" type="magnus-internal/x-httpd-php"
```

- A fájl végén hozz létre egy új, `x-httpd-php` nevű object-et. Ehhez a következő sorok beírása szükségeltetik:

```
<Object name="x-httpd-php">
ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
Service fn=php4_execute
</Object>
```

- Indítsd újra a szervert és engedélyezd a változatatásokat!
- Tedd meg ezt minden olyan web server példányra, amelyhez PHP támogatást akarsz adni!

A PHP NSAPI szűrőkénti használatának részletesebb leírásért lásd:
http://benoit.noss.free.fr/php/install-php4.html

Szerverek - OmniHTTPd

Ez a fejezet az OmniHTTPd szerverre történő telepítési útmutatókat tartalmazza.

Windows-os Omni HTTPd 2.0b1 vagy újabb verzió

A következő lépéseket kell megtenni, hogy a PHP működjön OmniHTTPd-vel. Ez a CGI futtató telepítésének lépései.Az OmniHTTPd támogatja a SAPI-t, de néhány teszt alapján nem tűnik túl stabilnak OmniHTTPd ISAPI modul.

- 1. lépés: Telepítsd fel a gépedre az OmniHTTPd szervert
- 2. lépés: Jobb gombbal klikkelj az OmniHTTPd ikonon a tálcán és válaszd ki az előbukkanó menüből a `Properties` pontot.
- 3. lépés: Kattints a `Web Server Global Settings` gombra.
- 4. lépés: Az "External" fül kiválasztása után írd be, hogy `virtual = .php | actual = c:\php-eleresi-ut\php.exe`, és nyomd meg az Add gombot!
- 5. lépés: A `Mime` fül kiválasztása után írd be: `virtual = wwwserver/stdcgi | actual = .php`, és nyomd meg az Add gombot!
- 6 lépés: Kattints az `OK`-ra.

Ismételd a 2-6. lépéseket minden kiterjesztésre, amit a PHP-hez szeretnél használni.

> **Megjegyzés:** Néhány OmniHTTPd csomag már beépített PHP támogatással jön. A telepítés elején lehet választani, hogy saját PHP telepítést akarsz a 'PHP component' kijelölés törlésével. Azt javasoljuk, hogy mindig a legfrissebb PHP binárisokat használd. Néhány OmniHTTPd a PHP 4 beta változatával kerül forgalomba, ezért nem ajánlatos a beépítettet választani. Ha a

szerveren már fut a gépen, akkor a 4. és 5. lépésben a ’Replace’ gombot kell használni az új, helyes adatok megadásához.

Szerverek - Oreilly Website Pro

Ebben a fejezetben az Oreilly Website Pro szerverhez történő telepítést tárgyaljuk.

Windows-os Oreilly Website Pro 2.5 és nagyobb verziók

Ez a lista a PHP CGI feldolgozóként ill. ISAPI modulként történő telepítésének lépéseit tartalmazza a Windows-os Oreilly Website Pro szerverekhez.

- Szerkeszd át a Server Properties-t és and válaszd ki a "Mapping" panelt!
- A List-ből válaszd ki a "Associations" pontot és írd be a kívánt fájlkiterjesztéseket (`.php`) és a CGI php.exe-hez vezető útvonalat - pl. `c:\php\php.exe` -, vagy az ISAPI DLL fájlt - pl. `c:\php\sapi\php4isapi.dll`.
- Válaszd ki a "Content Types" pontot és vedd fel ide is ugyanazokat a fájlkiterjesztéseket - `.php` megadva a content-type-ot is. Ha a CGI változatot használod, akkor ’wwwserver/shellcgi’-t kell beírni, míg ISAPI modul esetén ’wwwserver/isapi’-t. (Mindkét esetben idézőjelek nélkül.)

Szerverek - Xitami

Ebben a fejezetben az Xitami szerverhez történő telepítést tárgyaljuk.

Xitami for Windows

Ez a lista a PHP CGI feldolgozó telepítésének lépéseit írja le a Windows-os Xitami szerverekhez.

- Bizonyosodj meg, hogy fut a webszerver! Menj a Xitami adminisztrációs oldalára - általában `http://127.0.0.1/admin` -, és válaszd ki a ’Configuration’ pontot!
- Menj a Filters pontra, és add meg a File extensions mezőben milyen kiterjesztésű fájlokat rendelsz a PHP-hoz - pl.: .php.
- A ’Filter command or script’ mezőbe írd be a php állományhoz vezető útvonalat és a fájl nevét - pl. `c:\php\php.exe`.
- Nyomd le a ’Save’ ikont!

Más szerverekre fordítás

A PHP-t nagyon sok webszerverhez lehet telepíteni. Nézd meg a Szerverrel kapcsolatos beállítások c. fejezetben a szerverekhez kötödő konfigurációs beállítások teljes listáját! A CGI feldolgozó majdnem az összes, CGI szabványt támogató webszerverrel kompatibilis.

Problémák?

Olvasd el a FAQ-ot

Néhány probléma bizony gyakran előfordul. A leggyakrabban előforduló gondok és válaszok a PHP FAQ-ban olvashatóak, melynek címe: http://www.php.net/FAQ.php

Egyéb problémák

Ha még mindig elakadsz, talán valaki a PHP telepítési levelezési listán tud segíteni. Jól teszed, ha először megnézed az archívumot, hátha már valaki megválaszolt egy hasonló kérdést. Az archívum elérhető a support oldalon a http://www.php.net/ címen. Az angol PHP telepítési levelezőlistára való feliratkozáshoz küldj egy üres levelet a php-install-subscribe@lists.php.net (mailto:php-install-subscribe@lists.php.net). címre. A levelezőlista címe: `php-install@lists.php.net`. A magyar PHP levelezőlistára való feliratkozáshoz küldj egy subscribe témamegjelöléssel rendelkező levelet a wl-phplista-request@gimli.externet.hu (mailto:wl-phplista-request@gimli.externet.hu). címre. A levelezőlista címe: `wl-phplista@gimli.externet.hu`.

Ha segítséget szeretnél kapni valamelyik levelezőlistán, légy szíves próbálj meg precíz lenni, és minden fontos részletet adj meg a környezetről (operációs rendszer neve, PHP verziószám, web szerver neve, miként használod a PHP-t - CGI vagy szerver modulként, stb.). Ezenkívül adj meg elég PHP kódot, hogy a többiek reprodukálni és tesztelni tudják a problémát.

Bug jelentések

Ha úgy gondolod, hogy programhibát találtál a PHP feldolgozóban, légy szíves jelentsd a fejlesztőknek (angolul). Lehet, hogy a PHP fejlesztői semmit sem sejtenek felőle, és ha te nem jelented be, előfordulhat, hogy nem lesz kijavítva. Hibákat a bug-követő rendszeren regisztrálhatsz, melynek címe: http://bugs.php.net/.

Fejezet 3. Konfiguráció

A konfigurációs fájl

A konfigurációs fájlt (PHP 3.0-ban `php3.ini`, de PHP 4.0-től egyszerűen csak `php.ini`) a PHP induláskor olvassa be. A szerver modul verzióban ez csak egyszer történik meg, amikor a szerver elindul. A CGI verzióban ez minden meghíváskor megtörténik.

Példa 3-1. `php.ini` példa

```
; any text on a line after an unquoted semicolon (;) is ignored
[php] ; section markers (text within square brackets) are also ignored
; Boolean values can be set to either:
;    true, on, yes
; or false, off, no, none
register_globals = off
magic_quotes_gpc = yes

; you can enclose strings in double-quotes
include_path = ".:/usr/local/lib/php"

; backslashes are treated the same as any other character
include_path = ".;c:\php\lib"
```

A PHP Apache modulként történő használatakor a beállításokat az Apache konfigurációs fájljának direktíváival és .htaccess fájlokkal is megváltoztathatod. (Ehhez szükséges lesz "AllowOverride" beállításra vagy "AllowOverride All" jogosultságra.)

PHP 3.0 esetén minden `php3.ini` beállításhoz létezik egy megfelelő Apache direktíva, amely "php3_"-al kezdődik.

PHP 4.0 esetén azonban csak néhány Apache direktíva létezik, amivel a PHP beállításait közvetlenül a httpd.conf Apache konfigurációs állományban változtathatod meg.

php_value *`név érték`*

A megadott változó értékét határozza meg.

php_flag *`név on|off`*

Kétértékű logikai értéket állít be (ki- vagy bekapcsolás, azaz off vagy on)

php_admin_value *`név érték`*

A megadott változó értékét határozza meg. Az "admin" konfiguárciós beállítások csak a fő Apache konfigurációs fájlokban állíthatóak, a .htaccess fájlokban nem.

php_admin_flag *`név on|off`*

Kétértékű logikai értéket állít be (ki- vagy bekapcsolás, azaz off vagy on)

Példa 3-2. Apache konfigurációs példa

```
<IfModule mod_php4.c>
  php_value include_path ".:/usr/local/lib/php"
  php_flag safe_mode on
</IfModule>
<IfModule mod_php3.c>
  php3_include_path ".:/usr/local/lib/php"
  php3_safe_mode on
</IfModule>
```

A konfigurációs beállításokat megtekinthetők a phpinfo()-val, és egyenkénti elérést biztosít a beállításokhoz a get_cfg_var().

Általános célú beállítások

allow_url_fopen on|off

Ez a beállítás engedélyezi az URL-felismerő fopen kódot, amellyel az URL címeken elérhető objektumokat lehet megnyitni, mint pl. a fájlokat. Az alapbeállítású URL értelmezők távoli fájlok elérésére szolgálnak az FTP és HTTP protokollokkal. Néhány egyéb kiterjesztés, mint például a zlib további értelmezőket is beépíthet.

Megjegyzés: Ez a lehetőség közvetlenül a 4.0.3-as PHP kiadás után került a nyelvbe. A PHP 4.0.3 és azelőtti verziókban csak fordítási időben lehet letiltani ezt a funkciót a *--disable-url-fopen-wrapper* beállítással.

asp_tags on|off

Engedélyezi az ASP stílusú <% %> tag-ek használatát a hagyományos <?php ?> mellett. Ez a beállítás engedélyezi az érték kiíró rövidítés használatát is (<%= $valtozo %>). További információért nézd meg az Escape szekvencia HTML-ben című részt!

Megjegyzés: Az ASP stílusú tag-ek használata a 3.0.4-es verzió óta lehetséges.

auto_append_file string

Megadja a fájl nevét, amit automatikusan minden dokumentum végéhez illeszt. A beillesztés az include() függvényhez hasonlóan történik, tehát az include_path használatos az állomány eléréséhez.

A none speciális érték letiltja az automatikus hozzáfűzést.

Megjegyzés: Ha egy szkriptből exit(), segítségével lépsz ki, a hozzáfűzés *nem* működik.

auto_prepend_file string

Megadja a fájl nevét, amit automatikusan minden dokumentum elejére illeszt. A beillesztés az include() függvény használatához hasonlóan történik, tehát az include_path használatos az állomány eléréséhez.

A none speciális érték letiltja az automatikus hozzáfűzést.

cgi_ext string

display_errors on|off

Meghatározza, hogy a hibaüzenetek a HTML fájl részeként a kiírásra kerüljenek, vagy sem.

doc_root string

A PHP "gyökérkönyvtára" a szerveren. A PHP csak akkor használja, ha itt valamilyen értéket magadtál itt. Ha a PHP-t safe mode-ban használod, semmilyen fájlt nem dolgoz fel, ami ezen a könyvtáron kívül van.

engine on|off

A PHP Apache modul verziójában hasznos csak igazából. Olyan webhelyeken jöhet jól, ahol a PHP feldolgozását könyvtáranként vagy virtuális szerverenként szeretnék állítani. Az **`engine off`** direktívát a `httpd.conf` megfelelő helyeire beírva lehet a PHP-t kikapcsolni.

error_log string

A fájl neve, ahol a szkriptek hibaüzenetei tárolásra kerülnek. A `syslog` speciális érték megadásával az üzenetek rendszer naplójába kerülnek. UNIX alatt ez a syslog(3)-at jelenti, Windows NT-n az event log-ot. Ez a paraméter Windows 9x alatt nem támogatott.

error_reporting egész szám

Beállítja a hibajelzési szintet. A paraméter egy egész szám, ami egy bitmezőt reprezentál. Az alábbi értékek tetszőleges kombinációjával az értékeknek megfelelő hibákat jelezni fogja a PHP. Az összeadás eredménye a direktíva értéke.

Táblázat 3-1. Hibajelentési szintek

érték	engedélyezett hibajelzés
1	normál hibák
2	normál figyelmeztetések (warnings)
4	feldolgozási hibák
8	nem kritikus, stílussal kapcsolatos figyelmeztetések

Az alapérték 7 (normál hibák, normál figyelmeztetések és feldolgozási hibák jelennek meg).

html_errors boolean

Megadja, hogy a hibaüzenetekben szerepelhetnek-e HTML elemek.

open_basedir string

Meghatározza, hogy a PHP hol nyithat meg fájlokat a könyvtárfában.

Ha egy szkript megpróbál megnyitni egy fájlt (pl. fopen-el vagy gzopen-el), a fájl helye ellenőrzésre kerül. Ha a fájl kívül esik a megadott könyvtárból nyíló directory-fán, a PHP nem fogja megnyitni. Minden szimbolikus link feloldásra kerül, így egy symlink-kel nem lehet megkerülni ezt a korlátozást.

A . (pont) speciális érték azt jelzi, hogy a szkriptet tartalmazó könyvtárat kell alapkönyvtárnak tekinteni.

Windows alatt a több különböző könyvtárat pontosvesszővel elválasztva adhatod meg. Minden más rendszeren a megadott könyvtárakat kettősponttal kell elválasztani. Apache modul esetén az open_basedir elérési utak az egy szinttel feljebb lévő könyvtárból automatikusan öröklődnek.

Az open_basedir alatt megadottak tulajdonképpen csak előtagnak tekintendők, és nem kimondottan külön könyvtárnévnek. Ez azt jelenti, hogy az "open_basedir = /dir/incl" megengedi a hozzáférést a "/dir/include" és a "/dir/incls" könyvtárakhoz, ha azok léteznek. Ha egy adott könyvtárra kell korlátozni a hozzáférést, akkor a könyvtárnevet perjellel (/) kell lezárni, mint például: "open_basedir = /dir/incl/"

Megjegyzés: Több könyvtár megadása a 3.0.7-es verzió óta lehetséges.

Alapbeállításban a PHP semmilyen fájl megnyitását sem tiltja le.

gpc_order string

Beállítja a GET/POST/COOKIE sorrendet a változók létrehozásához. Az alapbeállítású érték "GPC". Ha például ezt "GP"-re írod át, a PHP figyelmen kívül fogja hagyni a cookie-kat, és ha van egy POST és egy GET érték ugyanazzal a névvel, a PHP a POST értéket teszi be a név által megadott változóba.

Ez az opció nem elérhető PHP 4-ben. Használd helyette a variables_order-t!

variables_order string

Beállítja az ún. EGPCS (Environment - környezeti, GET, POST, Cookie, Server) változók globális megfelelőinek létrehozási sorrendjét. Az alapbeállítás az "EGPCS". Ha például "GP"-re van beállítva, akkor a PHP figyelmen kívül hagyja a környezeti és szerver változókat valamint a cookie-kat (sütiket), és minden GET-ben kapott változót felülír a POST metódussal elküldött, azonos nevű változó.

Lásd még: register_globals!

ignore_user_abort string

Alapbeállításban "On". Kikapcsolásával (off) azonnal szkriptek leállnak, mihelyst megpróbálnak küldeni a kliensnek valamit, miután az bontotta a kapcsolat. Lásd még: ignore_user_abort()!

include_path string

Egy könyvtárlistát határoz meg, ahol a require(), include() és **fopen_with_path()** függvények a fájlokat keresik. A formátum a rendszer PATH környezeti változójának formátumával egyező: egy könyvtárlista kettőspontokkal elválasztva UNIX alatt, pontosvesszővel Windows alatt.

Példa 3-3. UNIX include_path

```
include_path=.:/home/httpd/php-lib
```

Példa 3-4. Windows include_path

```
include_path=".;c:\www\phplib"
```

Alapbeállítású érték a `.` (pont), azaz csak a szkript könyvtára.

isapi_ext string

log_errors on|off

Megadja, hogy a szkript hibaüzenetek a szerver error logjába is bekerüljenek-e vagy sem. Ez mindenképpen szerverfüggő beállítás!

magic_quotes_gpc on|off

Beállítja a "magic_quotes" állapotot a GPC (Get/Post/Cookie) műveletekhez. A bekapcsolásával minden ’ (egyszeres idézőjel), " (kétszeres idézőjel), \ (backslash) és NUL elé egy visszaperjel (\) kerül automatikusan. Ha a *magic_quotes_sybase* szintén be van kapcsolva, az egyszeres idézőjeleket megkettőzi, és nem visszaperjelet ír eléjük.

magic_quotes_runtime on|off

Ha a *magic_quotes_runtime* be van kapcsolva, a legtöbb függvény, amely külső forrásból ad vissza adatokat (beleértve adatbázisokat és szöveges fájlokat), az idézőjelek elé visszaperjelet (\) tesz. Ha a *magic_quotes_sybase* szintén be van kapcsolva, az egyszeres idézőjeleket megkettőzi, és nem visszaperjelet ír eléjük.

magic_quotes_sybase on|off

Ha a *magic_quotes_sybase* be van kapcsolva a *magic_quotes_gpc* vagy a *magic_quotes_runtime* kapsolóval együtt, akkor az egyszeres idézőjeleket megkettőzi, és nem visszaperjelet ír eléjük a megfelelő adatok használatakor.

max_execution_time egész szám

Előírja a szkriptek számára a maximális futásidőt, másodpercekben mérve. Ha ezt az időt túllépi a szkript, automatikusan megszakad a futása. Ez segíthet megvédeni a szervert a rosszul megírt, nagy erőforrásigényű szkriptektől. Az alapbeállítás `30` másodperc.

A maximum futási időbe nem számítanak be a rendszerhívások, a sleep() függvény, stb. Lásd a set_time_limit() függvény leírását további részletekért!

memory_limit egész szám

Megadja, hogy maximálisan hány byte memóriát foglalhat le egy szkript. Ez segíthet megvédeni a szervert a rosszul megírt szkriptektől, amelyek megehetik a szerver összes memóriáját.

nsapi_ext string

precision integer

A lebegőpontos számok kiírásakor megjelenő értékes számjegyek.

register_argc_argv boolean

Az argv és argc vátozók deklarálva legyenek-e az aktuális GET kérésben átadott értékekkel.

Lásd még: parancssori mód! Ez a beállítás 4.0.0-tól használható, azelőtt mindig be volt kapcsolva.

post_max_size integer

Megadja a POST metódussal maximális átadható adatmennyiséget, ebbe bele kell érteni a fájlfeltöltést is. Ahhoz, hogy nagy fájlokat is fel lehessen tölteni, ennek az értéknek nagyobb kell lennie, mint a upload_max_filesize.

Ha a memóriafelhasználás korlátozása is be lett kapcsolva a configure parancs futtattásakor, akkor a memory_limit beállítás ugyancsak hatással van a fájleltöltésre. Általánosságban memory_limit nagyobbnak kell lennie, mint a post_max_size.

register_globals on|off

Megadja, hogy az EGPCS (Environment, GET, POST, Cookie, Server) változók létrehozásra kerüljenek-e, mint globális változók. Ha nem szeretnéd, hogy mindenféle zűrös változók jöjjenek létre a globális változóid között, kapcsold ki ezt az opciót. Ez akkor jön a legjobban, ha a track_vars szolgáltatással együtt használod, amivel elérheted a EGPCS különböző változókat a `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE`, és `$_SERVER` asszociatív szuper-globális tömbökkel.

Figyelj arra, hogy ahhoz ez működjön, az apache konfigurációs fájljában a Directory blokkban az AllowOveride All beállítást kell megadni.

short_open_tag on|off

Megadja, hogy használható-e a PHP kódblokk elejét jelző rövid forma (**<? ?>**) a szkriptek futtatásakor. Ha a PHP-t az XML mellett használod, ki kell ezt kapcsolnod, és a hosszab formát kell alkalmaznod (**<?php ?>**).

sql.safe_mode on|off

track_errors on|off

Bekapcsolásával a legutóbbi hibaüzenet mindig ott lesz a `$php_errormsg` globális változóban.

track_vars on|off

Bekapcsolt állapotban a környezeti (Environment), a GET, POST, Cookie és Server változók a `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE`, és `$_SERVER` a nekik megfelelő szuper-globális asszociatív tömbökbe kerülnek értelemszerűen.

A PHP 4.0.3 és újabb verziókban a track_vars mindig be van kapcsolva.

upload_tmp_dir string

Ebbe az ideiglenes könyvtárba fogja a PHP elmenteni a weben feltöltött fájlokat. A könyvtárnak írhatónak kell lennie azon felhasználó számára, akinek alatt a PHP fut.

upload_max_filesize integer

Egy feltöltött fájl maximális mérete. Az érték bájtokban van megadva.

user_dir string

Annak a könyvtárnak a neve a felhasználók home könyvtárában, ahol a PHP fájlok vannak, például `public_html`.

warn_plus_overloading on|off

Engedélyezésével a PHP figyelmeztetést (warning) fog kiadni, ha a plusz (+) operátort használja stringekhez. Ezzel könnyebb megtalálni az újraírandó szkripteket, hogy a szövegösszefűzéshez a `.` operátort használják inkább.

Safe Mode beállítási lehetőségek

safe_mode on|off

Ki/bekapcsolja a PHP "safe mode" funkcióját. Lásd még a Biztonság és Safe Mode c. fejezeteket!

safe_mode_exec_dir string

Ha a PHP "safe mode"-ban fut, akkor system() és más külső programot futtató nyelvi elemek visszautasítják az itt megadott könyvtáron kívüli programok futtatását.

A hibakereső (debugger) beállítási lehetőségei

debugger.host string

A DNS név vagy IP cím, amit a hibakereső használ.

debugger.port string

A portszám, amit a hibakereső használ.

debugger.enabled on|off

A hibakereső engedélyezése.

Kiterjesztés-betöltés beállítási lehetőségei

enable_dl on|off

Ez a beállítás igazán csak az Apache modulban hasznos. Beállítható vele, hogy a dl() függvény használható legyen-e az egyes virtuális szervereken vagy könyvtárakban.

Biztonsági szempontból ennek a kikapcsolása a javasolt. Dinamikus betöltéssel ki lehet kerülni minden safe_mode és open_basedir megkötést.

Alapbeállításban engedélyezett a dinamikus betöltés, kivéve safe-mode-ban. Safe-mode-ban soha nem lehet használni a dl() függvényt!

extension_dir string

Az a könyvtár, ahol a dinamikusan betölthető kiterjesztések vannak.

extension string

Mely dinamikusan betölthető kiterjesztés legyen már a PHP indulásakor alapbeállításban betöltve.

mSQL beállítási lehetőségek

msql.allow_persistent on|off

Meghatározza, hogy használható-e állandó (persistent) mSQL kapcsolat.

msql.max_persistent egész szám

Az állandó (persistent) mSQL kapcsolatok maximális száma process-enként.

msql.max_links egész szám

Az mSQL kapcsolatok maximális száma process-enként, beleértve az állandó (persistent) kapcsolatokat is.

PostgreSQL beállítási lehetőségek

pgsql.allow_persistent on|off

Meghatározza, hogy használható-e állandó (persistent) PostgreSQL kapcsolat.

pgsql.max_persistent egész szám

Az állandó (persistent) PostgreSQL kapcsolatok maximális száma process-enként.

pgsql.max_links egész szám

A PostgreSQL kapcsolatok maximális száma process-enként, beleértve az állandó (persistent) kapcsolatokat is.

SESAM beállítási lehetőségek

sesam_oml string

A BS2000 PLAM library nevét állítja be, ami a betöltendő SESAM driver modulokat tartalmazza. Ez szükséges a SESAM függvények működéséhez. A BS2000 PLAM library mindenképpen ACCESS=READ,SHARE=YES beállításokkal kell rendelkezzen, mivel az apache felhasználójának kell tudni olvasni azt.

sesam_configfile string

A SESAM beállításokat tartalmaó fájl neve. Ez szükséges a SESAM függvények működéséhez. A BS2000 fájl legyen olvasható az apache felhasználó számára is.

A beállításokat tartalmazó fájl általában a következőket tartalmazza (lásd a SESAM kézikönyvet):

```
CNF=B
NAM=K
NOTYPE
```

sesam_messagecatalog string

A SESAM üzenet fájl neve. A legtöbb esetben ez a beállítás szükségtelen. Ha a SESAM üzenet fájl nincs telepítve, a rendszer BS2000 üzenet fájl táblájában, ezzel a direktívával beállítható.

Az üzenet katalógus fájl ACCESS=READ,SHARE=YES beállításokkal kell rendelkezzen, hogy az apache is tudja olvasni azt.

Sybase beállítási lehetőségek

sybase.allow_persistent on|off

Meghatározza, hogy használható-e állandó (persistent) Sybase kapcsolat.

sybase.max_persistent egész szám

Az állandó (persistent) Sybase kapcsolatok maximális száma process-enként.

sybase.max_links egész szám

A Sybase kapcsolatok maximális száma process-enként, beleértve az állandó (persistent) kapcsolatokat is.

Sybase-CT beállítási lehetőségek

sybct.allow_persistent on|off

Meghatározza, hogy létrejöhet-e állandó (persistent) Sybase-CT kapcsolat. Alapbeállításban engedélyezett.

sybct.max_persistent egész szám

Az állandó (persistent) Sybase-CT kapcsolatok maximális száma process-enként. Alapbeállításban -1, ami azt jelenti, hogy nincs korlát.

sybct.max_links egész szám

A Sybase-CT kapcsolatok maximális száma process-enként, beleértve az állandó (persistent) kapcsolatokat is. Alapbeállításban -1, ami azt jelenti, hogy nincs korlát.

sybct.min_server_severity egész szám

Azok a szerver üzenetek, amik ezzel megegyező vagy nagyobb fontossággal rendelkeznek, figyelmeztetésként (warning) jelennek meg. Ez az érték szkriptből is módosítható a sybase_min_server_severity() függvényyel. Alapbeállításban 10, ami az jelenti, hogy "information severity" és afölötti értékekkel rendelkező üzenetek hibát generálnak.

sybct.min_client_severity egész szám

Azok a kliens üzenetek, amik ezzel megegyező vagy nagyobb fontossággal rendelkeznek, figyelmeztetésként (warning) jelennek meg. Ez az érték szkriptből is módosítható a sybase_min_client_severity()függvénnyel. Alapbeállításban 10, ami gyakorlatilag kikapcsolja az ilyen üzeneteket.

sybct.login_timeout egész szám

A maximális idő (másodpercben), amit a kapcsolódási kérés befejezésére várni kell, mielőtt hiba a visszatérési érték. Fontos, hogy ha a max_execution_time értéket eléri a szkript, amikor a kapcsolati kísérlet kifut az időből, a szkript le fog állni, mielőtt a létre nem jött kapcsolat miatt elvégzendő feladatokat teljesítené! Az alapbeállítás egy perc.

sybct.timeout egész szám

A maximális idő (másodpercben), amit a select_db kérés befejezésére várni kell, mielőtt hiba a visszatérési érték. Fontos, hogy ha a max_execution_time értéket eléri a szkript, amikor a select_db kísérlet kifut az időből, a szkript le fog állni, mielőtt a létre nem jött választás miatt elvégzendő feladatokat teljesítené! Alapbeállításban nincs ilyen korlát.

sybct.hostname string

A host neve ahonnan csatlakozni szeretnél. Ezt jeleníti meg az sp_who. Nincs alapbeállítású értéke.

Informix beállítási lehetőségek

ifx.allow_persistent on|off

Meghatározza, hogy létrejöhet-e állandó (persistent) Informix kapcsolat.

ifx.max_persistent egész szám

Az állandó (persistent) Informix kapcsolatok maximális száma process-enként.

ifx.max_links egész szám

Az Informix kapcsolatok maximális száma process-enként, beleértve az állandó (persistent) kapcsolatokat.

ifx.default_host string

Alapbeállítású host, ha a szkript írója nem ad meg semmit az ifx_connect() vagy ifx_pconnect() paramétereként.

ifx.default_user string

Alapbeállítású felhasználói azonosító, ha a szkript írója nem ad meg semmit az ifx_connect() vagy ifx_pconnect() paramétereként.

ifx.default_password string

Alapbeállítású jelszó, ha a szkript írója nem ad meg semmit az ifx_connect() vagy ifx_pconnect() paramétereként.

ifx.blobinfile on|off

Állítsd igazra, ha blob oszlopokat egy fájlban szeretnéd visszakapni, hamisra, ha a memóriában. Ezt a beállítást felülbírálhatod futásidőben az ifx_blobinfile_mode()-al.

ifx.textasvarchar on|off

Állítsd igazra, ha a TEXT oszlopokat normál string-ként szeretnéd visszakapni select kérésekben, hamisra, ha ha blob id paramétereket használsz inkább. Ezt a beállítást felülbírálhatod futásidőben az ifx_textasvarchar()-al.

ifx.byteasvarchar on|off

Állítsd igazra, ha a BYTE oszlopokat normál string-ként szeretnéd visszakapni select kérésekben, hamisra, ha blob id paramétereket használsz inkább. Ezt a beállítást felülbírálhatod futásidőben az ifx_textasvarchar()-al.

ifx.charasvarchar on|off

Állítsd igazra, ha nem szeretnéd visszakapni a CHAR oszlopok záró szóközeit.

ifx.nullformat on|off

Állítsd igazra, ha a `NULL` oszlopokat a "`NULL`" string-ként szeretnéd megkapni select kérésekben, hamisra, ha üres stringként (""). Ezt a beállítást felülbírálhatod futásidőben az ifx_nullformat()-al.

Muli-Byte String beállítási lehetőségek

mbstring.internal_encoding string

Az `mbstring.internal_encoding` az alapbeállítású belső karakterkódolást adja meg.

mbstring.http_input string

Az `mbstring.http_input` az alapbeállítású HTTP bemeneti karakterkódolást adja meg.

`mbstring.http_output` string

Az `mbstring.http_output` az alapbeállítású HTTP kimeneti karakterkódolást adja meg.

`mbstring.detect_order` string

Az `mbstring.detect_order` az alapbeállítású karakterkódolás detektálási sorrendet adja meg.

`mbstring.substitute_character` string

Az `mbstring.substitute_character` az ismeretlen kódú karakterek helyettesítő karakterét adja meg.

BC Math beállítási lehetőségek

`bcmath.scale` egész szám

A decimális számjegyek száma a bcmath függvények számára.

Böngésző-képességek beállítási lehetőségei

`browscap` string

A böngészők képességeit tartalmazó fájl neve. Lásd még: get_browser().

Fejezet 4. Biztonság

A PHP egy igen hatékony nyelv és feldolgozó program, akár kiszolgálómodulként, akár egy különálló CGI futtatható állományként működik. Képes elérni fájlokat, futtatni parancsokat és hálózati kapcsolatokat nyitni a szerveren. Ezek a tulajdonságok alapesetben veszélyessé is tehetik más, a webszerveren futó alkalmazások számára. A PHP-t azonban úgy fejlesztették, hogy biztonságosabb legyen CGI programok írására, mint a Perl vagy C nyelvek. A PHP a fordítási és futásidejű beállítások helyes megválasztásával, és megfelelő programírási módszerek betartásával a szabadság és biztonság kívánt kombinációját biztosítja a fejlesztők számára.

Mivel sokféleképpen és sok mindenre lehet használni a PHP-t, számos konfigurációs lehetőség van a működésének szabályozására. A lehetőségek nagy száma garantálja, hogy a PHP-t sokféleképpen fel lehet használni, de egyben azt is jelenti, hogy ezek és a webkiszolgáló beállításainak kombinációi kritikus helyzeteket teremthetnek.

A beállítások sokszínűsége egyenlő mértékű a kódok sokszínűségével. A PHP használható teljes szerver-alkalmazások készítésére, egy shell felhasználó minden lehetőségével, vagy használható egyszerű ’server side include’-oknál, kis kockázattal egy szigorúan ellenőrzött rendszerben. Az, hogy hogyan kell kialakítani egy környezetet, milyen biztonságosan, nagyban a PHP fejlesztőn múlik.

Ez a fejezet néhány biztonsági tanácsot tárgyal, a különböző beállítási lehetőségeket és azokat a helyzeteket tárja fel, amelyekben ezeket biztonsággal lehet használni. Utána néhány kódolási szempontot is érint a különböző szintű védelem szempontjából.

Általános szempontok

A teljesen biztonságos rendszer kialakítani tulajdonképpen lehetlen, ezért a védelmi szakterületen alkalmazott megközelítés a kockázat és a használhatóság közti egyensúly megteremtésére törekszik. Ha minden a felhasználó által küldött adat két biometrikus érvényesítést (pl. retina- és ujjlenyomatvizsgálatot) igényel, akkor igen magas szintű a rendszer "felelősségre vonhatósága" (accountability). Ez azonban azt jelentené, hogy félórába telne kitölteni egy meglehetősen összetett űrlapot, ami arra ösztökélné a felhasználókat, hogy valahogy megkerüljék ezt a védelmet.

A legjobb védelem gyakran a kevésbé alkalmatlankodó és nem annyira feltűnő fajta, amely megfelel a követelményeknek anélkül, hogy megakadályozná a felhasználókat a munkájuk elvégzésében vagy túlterhelné a program íróit annak túlzott mérvű bonyolultsága. Valójában néhány biztonsági támadás pusztán a kiaknázása az olyasfajta túlságosan is kiépített védelemnek, amely hajlamos elerodálódni az idővel.

Egy mondatot érdemes megjegyezni: A rendszer csakis annyira jól védett, amennyire a leggyengébb láncszeme. Ha minden tranzakcióról feljegyzés készül idő, hely és tranzakciótípus alapján is, de a felhasználót csak egy egyszerű süti (cookie) alapján azonosítja a rendszer, akkor a felhasználók és a naplózott tranzakciók közti összefüggések érvényessége, megbízhatósága igen gyenge.

Tesztelés során figyelembe kell venni, hogy képtelenség minden lehetőséget kipróbálni már a legegyszerűbb oldalak esetén is. A programozó által várt adatok teljesen különbözőek azoktól és minden összefüggést nélkülöznek azokkal, amelyeket egy zsémbelődő alkalmazott képes elküldeni, vagy amelyeket egy szoftverkalóz (cracker) több havi munkájával állít össze, vagy amit egy házimacska a billenytyűzeten végiggyalogolva bevisz. Ezért a legjobb a programot logikai nézőpontból megközelíteni, hogy sikerüljön észrevenni, hol jöhetnek elő nem várt adatok és azok a továbbiakban hogyan módosulhatnak, tűnhetnek el vagy erősödhetnek fel a hatásuk.

Az Internet tele van olyan emberekkel, akik azzal akarnak maguknak nevet szerezni, hogy feltörik az oldalaidat, tönkreteszik a programjaidat, nem helyénvaló tartalommal töltik fel azokat, mellesleg egy - két izgalmas(?) napot szerezve ezzel Neked. Nem számít, hogy kis vagy nagy webhelyről van szó, elég indok a támadásra, hogy az rá van kapcsolva a hálóra, van egy szerver, amelyhez csatlakozni

lehet. Sok kódtörő program nem foglalkozik a méretekkel, egyszerűen csak nagy mennyiségű IP blokkokra vadászik áldozatokat keresve ezzel magának. Próbálj meg nem egy lenni közülük!

CGI futtatható állományként telepített PHP

Lehetséges támadások

A PHP CGI futtatható állományként való használata egy telepítési lehetőség azok számára, akik valami oknál fogva nem szeretnék a PHP-t modulként a szerverbe integrálni (pl. Apache), vagy a PHP-t más CGI wrapper-ekkel szeretnék használni biztonságos chroot és setuid környezet kialakítása érdekében. Ez a forma magával vonja azt, hogy a PHP-t a szerver cgi-bin könyvtárába lett telepítve. A CERT advisory CA-96.11 (http://www.cert.org/advisories/CA-96.11.interpreters_in_cgi_bin_dir.html) azt tanácsolja, hogy ne tegyél feldolgozó programot a cgi-bin könyvtárba. Bár a PHP használható mint egy egyedülálló feldolgozó program, a PHP-t úgy tervezték, hogy az ilyen telepítésekből adódó támadásokat kivédje:

- Rendszerfájlok elérése: `http://domain.nev/cgi-bin/php?/etc/passwd`

  Az URL lekérési információja (query information), ami a kérdőjel (?) után található, parancssori paraméterként kerül átadásra a feldolgozónak. Általában a feldolgozók megnyitják, és lefuttatják az első paraméterként adott fájlt.

  Ha a PHP CGI futtatható állományként hívódik meg, nem veszi figyelembe a parancssori paramétereket.

- Bármilyen web dokumentum elérése a szerveren:
  `http://domain.nev/cgi-bin/php/titkos/doc.html`

  Az elérési út információ (path information) az URL része, a futtatható fájl neve után lévő `/titkos/doc.html` a CGI program által megnyitásra és futtatásra kerülő fájl elérésének meghatározására használatos. Tipikusan néhány webkiszolgáló beállítási lehetőség (Apache-ban: Action) használatos a kérések átirányítására a dokumentumhoz, mint a `http://domain.nev/titkos/szkript.php` a PHP értelmező számára. Ezzel a beállítással a szerver először ellenőrzi az elérési engedélyeket a `/titkos` könyvtárra, és ezután állítja elő az átirányító kérést a `http://domain.nev/cgi-bin/php/titkos/szkript.php` oldalra, amit így már a PHP feldolgoz. Azonban ha eredetileg is ebben a formában volt megadva a kérés, nem történik elérési ellenőrzés a `/titkos/szkript.php` fájlra, csak a `/cgi-bin/php` fájlra. Ilyen módon bárki, aki elérheti a `/cgi-bin/php` címet, egyben tetszőleges védett dokumentumot is elérhet.

  A PHP esetében az --enable-force-cgi-redirect fordítási paraméter, a doc_root és user_dir konfigurációs lehetőségek használhatóak ennek kivédésére, ha a szerver dokumentumainak könyvtárfájában van olyan könyvtár, ami elérési korlátozásokkal bír. Nézd meg az alábbi lehetőségeket a különböző kombinációkhoz!

1. eset : csak publikus fájlok

Ha a szerveren nincs olyan tartalom, ami jelszó vagy IP alapú védelemmel van ellátva, nincs szükség ezekre a konfigurációs beállításokra. Ha a kiszolgáló nem engedélyezi az átirányításokat, illetve ha nincs módja biztonságos átirányítással küldeni a kérést a PHP számára, megadhatod az --enable-force-cgi-redirect opciót a "configure" szkript számára. Meg kell győződnöd arról, hogy a PHP szkriptjeid nem függnek egy speciális szkript-hívási formától sem, mint a `http://domain.nev/cgi-bin/php/dir/szkript.php` vagy a `http://domain.nev/dir/szkript.php`.

Az átirányítás beállítása Apache alatt az AddHandler és Action direktívákkal történik (lásd lentebb).

2. eset : az --enable-force-cgi-redirect használata

Ez a fordítási paraméter megakadályozza, hogy bárki meghívja a PHP-t egy `http://domain.nev/cgi-bin/php/titkos/szkript.php`. URL-el. Ehelyett a PHP csak akkor fog elfogadni egy ilyen kérést ha egy szerver átirányításban kapta.

Apache esetében tipikusan a következő direktívákkal történik a beállítás:

```
Action php-script /cgi-bin/php
AddHandler php-script .php
```

Ez a lehetőség csak az Apache web szerverrel tesztelt és azon múlik, hogy az Apache beállítja a nem standard REDIRECT_STATUS CGI környezeti változót ha átirányított kérésről van szó. Ha a webkiszolgálód semmilyen módon nem közli, hogy ez egy direkt vagy átirányított kérés volt-e, nem használhatod ezt az opciót, így valamelyik másik módot kell használnod.

3. eset : a doc_root vagy user_dir beállítása

Aktív tartalom elhelyezése a normál dokumentumok között, (pl. szkriptek és futtatható állományok) veszélyes gyakorlat lehet. Ha például valamilyen beállítási hiba miatt a szkriptek ahelyett, hogy lefutnának hagyományos HTML dokumentumokként jelennek meg, mindenki számára tisztán látható válnak kódolási technikáid és pélsául adatbázis jelszavaid. Ezért néhány rendszeradminisztrátor inkább egy külön könyvtárat jelöl ki, ami csak a PHP CGI által elérhető, és így mindig feldolgozásra kerül és nem jelenik meg a szkript kódja.

Ha a fent leírt átirányítás azonosítási mód nem működik, fontos, hogy egy különálló szkript doc_root-ot határozz meg, ami nem azonos a web doc_root-al.

A PHP szkript dokumentumok gyökérkönyvtárát a doc_root konfigurációs beállítással határozhatod meg a konfigurációs fájlban, vagy a PHP_DOCUMENT_ROOT környezeti változóban adhatod meg ezt az értéket. Ha ez be van állítva a PHP CGI verziója a fájl elérési útját a *`doc_root`* és a kérés elérési út információja (path information) alapján állítja elő, ami azt jelenti, hogy ezen a könyvtáron kívül nem futtatható fájl. (kivéve a *`user_dir`* esetét).

Egy másik itt használható opció a user_dir. Ha ez nincs megadva, csak a *`doc_root`* szabályozza a megnyitható fájlok körét. Ekkor egy `http://domain.nev/~user/doc.php` URL nem a "user" nevű felhasználó home könyvtárában lévő fájlt keresi, hanem a `~user/doc.php` fájlt keresi a doc_root alatt (igen, egy tilde karakterrel kezdődő könyvtárban [~]).

Ha a user_dir meg van adva, például `public_php`, akkor a fenti `http://domain.nev/~user/doc.php` kérés a `doc.php` nevű fájlt fogja megnyitni a "user" nevű felhasználó home könyvtárában lévő `public_php` könyvtárban. Ha a "user" home könyvtára `/home/user`, a lefuttatandó fájl a `/home/user/public_php/doc.php` lesz.

A *`user_dir`* kifejtés a *`doc_root`* beállítástól függetlenül működik, úgyhogy a dokumentum gyökér és felhasználói könyvtár beállításokat külön is használhatod.

4. eset : PHP feldolgozó a web könyvtárfán kívül

Rendkívül biztonságos lehetőség a PHP feldolgozót valahol a webről látható könyvtárakon kívülre tenni. Például az `/usr/local/bin` könyvtárba. Az egyetlen igazi hátránya ennek az opciónak az, hogy minden PHP szkript első sorának egy ehhez hasonló sort kell megadnod:

```
#!/usr/local/bin/php
```

ami meghatározza, hogy hol található a PHP feldolgozó, ami lefuttatja majd ezt a kódot. Ráadásul minden PHP szkriptnek futási jogot kell adni. Azaz úgy kell eljárni, mint bármilyen más nyelven megírt CGI programmal, amit Perl, sh vagy és a `#!` shell-escape mechanizmust használja önmaga futtatására.

Ahhoz, hogy ebben az esetben a PHP helyesen kezelje a PATH_INFO és a PATH_TRANSLATED információkat, a PHP feldolgozót az --enable-discard-path "configure" paraméterrel kell fordítani.

Apache modulként telepített PHP

A PHP-t Apache modulként használva örökli az Apache-t futtató felhasználó (tipikusan a "nobody") jogait. Ennek többféle hatása van a biztonságra és az azonosításra. PHP-n kersztüli adatbázis elérés esetén például, az adatbázist elérhetővé kell tenni ennek a felhasználónak számára is - annak ellenére, hogy az adatbázisnak beépített azonosítása van. Ez azt jelenti, hogy egy rosszindulatú szkript elérheti, és módosíthatja az adatbázist, akár felhasználói név és jelszó nélkül is. Lehetséges, hogy egy keresőrobot beleakadjon az adatbázis-adminisztrációs oldalak egyikébe, és kiürítse az összes adatbázist. Természetesen lehet ez ellen védekezni Apache azonosítási technikákkal, vagy LDAP segítségével megvalósítotott saját elérési modellekkel, vagy külön .htaccess fájlokkal, stb., és ezeket a saját PHP kód részevé is lehet tenni így.

Általában, ha a biztonságot akkora szintre tudjuk emelni, hogy a PHP felhasználó (ebben az esetben az Apache-é) igen kis kockázattal fut, akkor nem képes például akármilyen fájlok írására a user könyvtárakba. Letilthatjuk számára egy adatbázis elérését vagy megváltoztatását. Tipikusan ebben a helyzetben már azokat a fájlokat sem tudja írni, amit kellene, vagy egyaránt nem tud végrehajtani jó és rosszindulatú adatbázis tranzakciókat egyaránt.

Gyakori hiba ezen a ponton, hogy az Apache-nak root jogokat adnak vagy valamilyen egyéb módon bővítik az Apache jogait / lehetőségeit.

Az Apache felhasználó jogainak root szintre bővítése különösen veszélyes, és tönkreteheti a teljes rendszert, tehát sudo, chroot vagy más hasonló eszközök használata elkerülendő, ha nem vagy biztonsági szakember.

Van néhány egyszerűbb megoldás is. Az open_basedir használatával szabályozni lehet, hogy mely könyvtárakat olvashatja a PHP. Ki lehet jelölni ún. csak apache-s területeket, hogy minden web alapú művelet ide, csak ezekre a nem rendszer- és nem felhasználói fájlokra korlátozódjon.

Fájlrendszer biztonság

A PHP tiszteli a rendszerbe épített biztonsági megoldásokat, különös tekintettel a fájlok és könyvtárak hozzáférési jogosultságaira. Ez lehetőséget ad arra, hogy megszabd, mely fájlok olvashatóak a rendszerben. A mindenki számára olvasható fájloknál ügyelni kell arra, hogy ne tartalmazzanak olyan fontos adatot, amit nem szabad elolvasnia akármelyik felhasználónak a rendszeren.

Mivel a PHP úgy készült, hogy felhasználói szintű fájlrendszer hozzáférést ad, lehetséges olyan program készítése, amely a rendszerfájlokat olvassa, pl. az /etc/passwd fájlt. Ez maga után von egy nyilvánvaló következtetést, neveztesen minden esetben meg kell győződni a programokban arról, hogy a helyes fájlokat olvassa illetve írja a program.

Nézzük a következő szkriptet, ahol a felhasználó megadja, hogy le szeretne törölni egy fájlt a könyvtárában. Ez többnyire egy webes felületet jelent, ahol egy PHP program használatos fájlkezelésre, ezért az Apache-t futtató felhasználónak engedélyezni kell a fájlok törlését a felhasználó könyvtárában.

Példa 4-1. A helytelen változó használat ...

```
<?php
// egy fájl törlése a user könyvtárából
$usernev = $HTTP_POST_VARS["user_altal_beadott_nev"];
$konyvtar = "/home/$usernev";
$torlendo_file = "$userfile";
unlink ($konyvtar/$torlendo_file);
echo "$torlendo_file törölve!";
?>
```

Mivel a `$usernev` egy HTML űrlapból érkezik, a felhasználó beírhat tetszőleges felhasználói nevet és fájlnevet, így akár más könyvtárát is manipulálhatja. Ebben az esetben általában valamilyen felhasználó-azonosítási eljárást kell alkalmazni. Lássuk, mi történik, ha a beadott változók a "../etc/" és a "passwd". A kód akkor így alakulna (az adatokat behelyettesítve):

Példa 4-2. ... fájlrendszer támadáshoz vezethet

```
<?php
// egy fájl törlése akárhonnan, ahol a PHP usernek
// joga van erre. Ha a PHP root userként fut:
$usernev = "../etc/"
$konyvtar = "/home/../etc/";
$torlendo_file = "passwd";
unlink ("/home/../etc/passwd");
echo "/home/../etc/passwd törölve!";
?>
```

Két fontos komponensre kell odafigyelni, hogy megelőzhessük az ilyen problémákat:

- Csak korlátozott jogok beállítása a PHP-t futtató felhasználónak.
- Minden felhasználótól bejövő adat ellenőrzése.

Egy jobban átgondolt, tökéletesitett szkript:

Példa 4-3. Biztonságosabb fájl ellenőrzés

```
<?php
// egy fájl törlése akárhonnan, ahol a PHP usernek
// joga van erre.
$usernev = $HTTP_SERVER_VARS['REMOTE_USER']; // ez a user azonosított neve
                                             // (ha volt előtte azonosítás)

$konyvtar = "/home/$usernev";

$torlendo_file = basename("$userfile"); // elérési trükközés eldobása
unlink ($konyvtar/$torlendo_file);

$fp = fopen("/home/logging/filedelete.log","+a"); // törlés naplózása
$logstring = "$usernev $konyvtar $torlendo_file";
fputs ($fp, $logstring);
fclose($fp);

echo "$torlendo_file törölve!";
?>
```

Mindamellett, ez sem mentes a hiányosságoktól. Ha az aktuálisan használt hitelesítési módszer (authentication) megengedi a felhasználóknak, hogy saját loginnevet válasszanak, és az egyikük a "../etc/" -t választja, akkor a rendszer ugyanolyan védtelenné válik. Ebből kiindulva a jobban testreszabott ellenőrzés a következőképp alakulna:

Példa 4-4. Biztonságosabb fájlnév-ellenőrzés

```
<?php
$usernev = $HTTP_SERVER_VARS['REMOTE_USER']; // hitelesites
$homedir = "/home/$usernev"; $homedir = "/home/$usernev";

if (!ereg('^[^./][^/]*$', $userfile))
    die('rossz fájlnév'); // vége, nincs feldolgozás

if (!ereg('^[^./][^/]*$', $usernev))
    die('rossz usernev'); // vége, nincs feldolgozás
//stb...
?>
```

A használt operációs rendszertől függően széles a védeni kívánt fájlok skálája, beleértve az eszköz hivatkozásokat (/dev/ vagy COM1), konfigurációs fájlokat (/etc/ és az .ini fájlok), jól ismert tárolóhelyek (/home/, My Documents), stb. E sokaság miatt könnyebb egy olyan rendszert készíteni, ahol mindent tiltunk azon kívül, amelyet kifejezetten megengedünk.

Adatbázis biztonság

Mostanában, a dinamikus tartalmat szolgáltató web alkalmazások sarokkövének számítanak az adatbázisok. Mivel nagyon kényes, titkos adatok tárolására szolgálhatnak ezek az adatbázisok, erősen megfontolandó, miképp védjük meg ezeket.

Információk tárolásához vagy visszakereséséhez csatlakozni kell az adatbázishoz, egy érvényes lekérdezést kell küldeni, az eredményt ki kell olvasni, és le kell zárni a kapcsolatot. Manapság ebben a párbeszédben a Structured Query Language (SQL) a leggyakrabban használt lekérdezőnyelv. Figyeld meg, miként lehet SQL lekérdezéseket megbabrálni!

Mint látható, a PHP egymagában, magától nem képes megvédeni az adatbázist. A következő bekezdések célja, hogy betekintést adjanak az alapokba, hogyan kell adatbázisokat elérni és módosítani egy PHP programon belül.

Tartsd észben a következő egyszerű szabályt: tagoltan védekezni. Minél több helyen minél többet teszel a biztonság növeléséért, annál kisebb a valószínűsége, hogy a támadók sikerrel járjanak, és kiteregessék titkos adataidat, vagy visszaéljenek velük. A jó adatbázis- és alkalmazástervezés mindig a legnagyobb félelmek figyelembevételéről ismerszik meg.

Adatbázis-tervezés

Az első lépés mindig az adatbázis létrehozása, hacsak nem egy kívülállóét kell használni. Az adatbázis létrehozásakor az a tulajdonosáé lesz, azé, aki lefuttatta az utasításokat. Általában csak a tulajdonos - esetleg az ún. superuser - jogosult bármiféle az adatbázis elemeit érintő műveletre. Annak érdekében, hogy más felhasználók is hozzáférjenek, jogokat kell nekik biztosítani.

Az alkalmazásoknak soha nem szabad a tulajdonosaként vagy superuserként csatlakozni az adatbázishoz, mert ezek bármilyen utasítást és lekérdezést tetszés szerint futtathatnak, pl. a szerkezeti módosítást (táblák megszüntetése) vagy táblák komplett törlése.

Létre lehet hozni különböző, szigorúan korlátozott jogosultságú adatbázis- felhasználókat, melyek mindegyike az adatbázis manipulációnak egy-egy különböző nézőpontjáért felelősek. Mindig csak a legszükségesebb jogokat szabad engedélyezni, és el kell kerülni, hogy ugyanazt a felhasználót használjuk szerepeiben egymástól különböző esetekben. Ez azt jelenti, hogy ha a behatoló meg is szerzi valamelyik ilyen minősítést (hitelesítési információt = felhasználói név + jelszó), akkor is csak akkora változást tud okozni, mint az alkalmazás maga.

Nem kell minden feladatfüggő szabályozást a webalkalmazásban (PHP szkriptben) kódolni, ehelyett inkább használd az adatbázis lehetőségeit: view-k (nézetek), trigger-ek, rule-ok (szabályok). Ha a rendszer fejlődik, és más alkalmazásokat is csatlakoztatni kell az adatbázishoz, akkor mindegyiknél újra kellene programozni ezeket a szabályokat. Mindezen felül a triggerek arra is jók, hogy átlátszó módon és automatikusan kezeljenek egyes mezőket az adatbázisban, amelyek gyakran bepillantást adnak abba, hogy mi is történik/történt egy tranzakció közben, vagy nagyon hasznosnak bizonyulhatnak hibakeresés során.

Kapcsolódás az adatbázishoz

Elképzelhető, hogy SSL-n keresztül szeretnél kapcsolódni az adatbázishoz, hogy a kiszolgáló és ügyfél közti teljes kommunikáció titkosításával növeld a védelmet. Használhatsz ssh-t is erre a célra. Akármelyik is áll, nagyon nehéz lesz a forgalom lehallgatásából információkat kinyerni ezek után.

Titkosított tárolás

SSL/SSH az ügyfél és kiszolgáló közt mozgó adatokat védi, és nem védi az adatbázisban tárolt megmaradó adatokat. Az SSL - kapcsolati protokoll.

Mihelyst a támadó közvetlen hozzáférést szerzett az adatbázishoz - megkerülve a webszervert -, a tárolt adatok védtelenné váltak, és visszaélhet velük, ha csak maga az adatbázis nem védi valahogy azokat. Az adatok titkosítása kellőképp enyhíti ezt a veszélyt, de jelenleg nagyon kevés adatbázis kezelő támogatja a titkosítást.

Ez a legkönnyebben saját titkosító csomag írásával oldható meg, amelyet utána a PHP szkriptből el lehet érni. Ebben az esetben a PHP segítséget nyújthat néhány kiterjesztésével, mint például az Mcrypt vagy az Mhash, amelyek nagyon sokféle titkosító algoritmust fednek le. A szkript először titkosítja a tárolni kívánt adatot, majd visszakereséskor visszafejti azokat. Nézd meg a hivatkozott fejezeteket további példákért, hogyan kell a titkosítást végrehajtani.

Olyan teljesen rejtett adatok esetén, amelyeknek nyílt ábrázolásukra nincs szükség, mert nem lesznek kiíratva, a hashelés alkalmazása is meggondolandó. A hashelés jól ismert példája az, hogy a jelszavak helyett, azoknak csak MD5 hash értékét tárolják az adatbzisban. Lásd még: crypt() és md5()!

Példa 4-5. Hashelt jelszó mező használata

```
// jelszó hash értékének tárolása
$query  = sprintf("INSERT INTO users(name,pwd) VALUES('%s','%s');",
            addslashes($username), md5($password));
$result = pg_exec($connection, $query);

// lekérdezés, vajon a felhasználó a helyes jelszót adta-e meg
$query = sprintf("SELECT 1 FROM users WHERE name='%s' AND pwd='%s';",
            addslashes($username), md5($password));
$result = pg_exec($connection, $query);

if (pg_numrows($result) > 0) {
    echo "Üdvözöllek, $username!";
}
else {
    echo "$username hitelesítése nem sikerült.";
}
```

SQL "beoltás"

Sok web fejlesztő nincs tudatában annak, hogy hogyan lehet megbabrálni az SQL utasításokat, ezért az SQL utasításokat megbízható parancsoknak feltételezik. Ez azt jelenti, hogy az SQL

lekérdezésekkel ki lehet játszani a hozzáférés szabályozásokat, meg lehet kerülni a szabályos engedélyezési folyamatokat , és néha az SQL lekérdezésekkel a gazdagépen operációs rendszer szintű hozzáférést is lehet létrehozni.

A "közvetlen SQL utasítás befecskendezés" olyan módszer, amellyel a támadó a régi SQL utasításokat módosítja vagy újakat ad hozzájuk annak érdekében, hogy titkos információkhoz jusson hozzá, vagy felülírja azokat, vagy veszélyes rendszer szintű parancsokat futtasson az adatbázis gazdagépén. Ez olyan alkalmazások esetén tehető meg, amelyek a felhasználótól származó adatokból és statikus paraméterekből állítanak össze SQL lekérdezéseket. Sajnos, a következő példák mind megtörtént eseteken alapulnak.

Az, hogy az adatbázishoz superuserként (olyan személyként, aki superusert képes létrehozni) csatlakozott az alkalmazás, és a bevitt adatok ellenőrzésének hiánya odavezethet, hogy a támadó superuser hozzáférést hozhat létre az adatbázishoz.

Példa 4-6. A keresési eredmények lapokra tördelése ... és superuserek létrehozása (PostgreSQL és MySQL)

```
$offset = argv[0]; // Vigyázz, nincs beviteli ellenőrzés!
$query  = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";
// PostgreSQL
$result = pg_exec($conn, $query);
// MySQL
$result = mysql_query($query);
```

A szokványos felhasználó az ’előző’, ’következő’ linkekre kattint, ahol az `$offset` az URL-be van kódolva. A szkript azt várja, hogy `$offset` decimális szám. Mégis, valaki megpróbálhatja a következő utasítás urlencode() alakját hozzáfűzni az URL-hez:

```
// PostgreSQL
0;
insert into pg_shadow(usename,usesysid,usesuper,usecatupd,passwd)
    select ’crack’, usesysid, ’t’,’t’,’crack’
    from pg_shadow where usename=’postgres’;
--

// vagy MySQL esetén
0;
UPDATE user SET Password=PASSWORD(’crack’) WHERE user=’root’;
FLUSH PRIVILEGES;
```

Ha ez megtörténne, akkor a szkript megajándékozná a támadót egy superuser hozzáféréssel. A `0;` arra való, hogy érvényes offset-et biztosítson az eredeti lekérdezésnek.

> **Megjegyzés:** Általános módszer, hogy a `--` jellel kényszerítik ki, hogy az SQL elemző figyelmen kívül hagyja a lekérdezésként átadott string fennmaradó részét, mivel ez a megjegyzés szabványos jelölése SQL-ben.

Egy lehetséges módja a jelszavak megszerzésének, hogy kijátszák a kereső oldalak találati listájának lekérdezéseit. A támadónak mindössze annyit kell tennie, hogy végig próbálja melyik elküldött SQL lekérdezésben használt változó nincs megfelelően lekezelve. Ezeket általában egy megelőző űrlapon lehet beállítani, hogy testre szabjuk a `SELECT` utasítás `WHERE, ORDER BY, LIMIT` és `OFFSET` klauzuláit. Ha a használt adatbáziskezelő támogatja a `UNION` szerkezetet, akkor a támadó esetleg hozzáfűzhet egy teljesen új lekérdezést a már meglevőhöz, hogy kilistázza valamelyik táblában tárolt jelszavakat. Titkosított tárolás erősen ajánlott!

Példa 4-7. Árucikkek listázása ... és néhány jelszóé (valamilyen adatbázis kezelő)

```
$query  = "SELECT id, name, inserted, size FROM products
                  WHERE size = ’$size’ AND inserted BETWEEN ’$min_date’ AND ’$max_date’
                  ORDER BY $order LIMIT $limit, $offset;";
$result = odbc_exec($conn, $query);
```

A lekérdezés statikus része egy másik `SELECT` utasítással kombinálható, ami az összes jelszót kilistázza:

```
’
union select ’1’, concat(uname||’-’||passwd) as name, ’1971-01-01’, ’0’ from usertable;
--
```

Ha ezt a lekérdezést (a `’` és `--` megfelelő használatával) valamelyik `$query`-ben használt változóhoz sikerülne hozzárendelni, akkor a szörny felébredne.

SQL UPDATE parancsok ugyancsak ki vannak téve az adatbázisok elleni támadásoknak. Ezeket az utasításokat is fenyegetik az előzőekben megismert megrövidítő és hozzáfűző technikák. Ám emellett a támadó meghamisíthatja a `SET` klauzulát is. Ebben az esetben némi séma információval rendelkeznie kell a támadónak, hogy sikerrel járjon. Ezeket az információkat az űrlapváltozók neveiből szerezhetik meg, vagy egyszerűen próbálgatással. Az általánosan használt elnevezések a felhasználói névre és jelszóra nem nagyon különböznek egymástól.

Példa 4-8. Jelszó átírásától ... új jogok megszerzéséig (valamilyen adatbázis kezelő)

```
$query = "UPDATE usertable SET pwd=’$pwd’ WHERE uid=’$uid’;";
```

A rosszindulatú felhasználó a `’ or uid like’%admin%’; --` értéket adja át a `$uid` változónak, és ezzel megváltoztatja az adminisztrátor jelszavát, vagy egyszerűen a `$pwd`-nek a `"hehehe’, admin=’yes’, trusted=100 "` (lezáró szóközzel) értéket adva még több jogot szerez magának. Ezt az SQL parancsot ezek így ferdítik el:

```
// $uid == "’ or uid like’%admin%’; --"
$query = "UPDATE usertable SET pwd=’...’ WHERE uid=” or uid like ’%admin%’; -
-’;";
```

```
// $pwd == "hehehe', admin='yes', trusted=100 "
$query = "UPDATE usertable SET pwd='hehehe', admin='yes', trusted=100 WHERE ...";
```

Egy ijesztő példa, hogyan lehet az adatbázis gazdagépén operációs rendszerszintű parancsokat futtatni.

Példa 4-9. Az adatbázis-gazdagép operációs rendszere elleni támadás (MSSQL Server)

```
$query  = "SELECT * FROM products WHERE id LIKE '%$prod%'";
$result = mssql_query($query);
```

Ha a támadó az `a%' exec master..xp_cmdshell 'net user test testpass /ADD' --` értéket küldi el a `$prod` változónak, akkor a `$query` a következőképp alakul:

```
$query  = "SELECT * FROM products
                   WHERE id LIKE '%a%'
                   exec master..xp_cmdshell 'net user test testpass /ADD'-
-%'";
$result = mssql_query($query);
```

MSSQL Server futtatja a kötegbe fogott SQL utasításokat, köztük azt is, amelyik új felhasználót vesz fel az adatbázis kiszolgálógépen. Ha az alkalmazás `sa` jogosultsággal fut és az MSSQLSERVER service megfelelő jogokkal fut, akkor a támadónak most már hozzáférése van ehhez a géphez.

> **Megjegyzés:** A példák némelyike bizonyos adatbáziskezelőhöz kötődik. Ez nem azt jelenti, hogy hasonló támadás elképzelhetetlen más termékkek ellen. Az általad használt adatbázis-kezelő ugyanilyen sérülékeny lehet, akár más módon.

Elhárítási módszerek

Ellenevetésként felmerülhet, hogy a példák többségében a támadónak rendelkeznie kell valamennyi előzetes információval az adatbázis felépítéséről. Ez igaz, de soha nem lehet tudni, hogy mikor, hol, hogyan szerezhetik meg ezeket, és ha ez megtörtént, az adatbázisod védtelenné válik. A behatolók könnyen hozzájuthatnank a program egy darabjához nyílt forráskódú, vagy olyan nyilvánosan elérhető adatbázis-kezelő programcsomag használatakor, amelyik egy fórum vagy tartalomszolgáltató rendszer része. Ez különösen veszélyes lehet, ha ezek kevéssé átgondoltak és gyengén megtervezettek.

Ezek a támadások alapvetően olyan programoknak a kijátszásán alapulnak, amelyek a védelmet/biztonságot figyelmen kívül hagyva születtek. Soha nem lehet megbízni semmilyen bejövő adatban, főleg ha az a kliens oldalról érkezik, még akkor sem, ha az egy általunk megadott süti

(cookie), vagy rejtett mező (hidden input) értéke esetleg egy legördülő lista eleme. Még egy olyan ártatlan lekérdezés, mint ami az első példában látható, katasztrófát okozhat.

- Soha ne csatlakozz az adatbázishoz tulajdonosaként vagy superuser-ként. Mindig kevés jogosultsággal rendelkező, testreszabott felhasználókat használj!
- Ellenőrizd a bejövő adat típusát, hogy az a vártnak megfelelő-e! A PHP a bevitelt ellenőrző függvények széles körével rendelkezik kezdve a legegyszerűbbektől - pl.: Változókkal kapcsolatos függvények közül is_numeric() vagy a Character Type Functions közül a ctype_digit() - a Perl kompatibilis reguláris kifejezések támogatásáig.
- Ha az alkalmazás számot vár, akkor megfontolandó az is_numeric() függvénnyel ellenőrizni a típusát, vagy csendben megváltoztatni a típusát a settype() függvénnyel, vagy szám szerinti ábrázolását használni az sprintf() függvénnyel.

Példa 4-10. A lapozáshoz használt lekérdezés összeállításának biztonságosabb módja

```
settype($offset, 'integer');
$query  = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";

// Figyelj a %d -re a formázó sztringben, a %s használat értelmetlen lenne
$query  = sprintf("SELECT id, name FROM products ORDER BY name LIMIT 20 OFF-
SET %d;", $offset);
```

- Idézőjelek közé kell tenni minden nem szám jellegű, felhasználótól származó adatot, erre használható az addslashes() vagy az addcslashes(). Lásd még az első példát! Ahogy a példák is mutatják, a statikus részbe égetett idézőjelek nem elegendőek, és könnyen kijátszhatók.
- Semmilyen adatbázisra jellemző információt - különösen szerkezetit - nem szabad kiírni, ha törik, ha szakad. Lásd még: Hibajelzés és Hibakezelő és naplózó függvények!
- Tárolt eljársokat és előre definiált kurzorokat is használhatsz, hogy az adatbázis elérést absztraháld annak érdekében, hogy a felhasználók ne közvetlenül a táblákhoz vagy nézetekhez férjenek hozzá. Ennek a megoldás azonban egyéb hatásai vannak.

Ezeken kívül, hasznot hajthat a lekérdezések naplózása akár a szkripteken belül, akár ha az adatbázis kezelő maga teszi ezt. Nyilvánvalóan ez nem tud megakadályozni egyetlen ártalmas próbálkozást sem, de segítséget nyújthat annak felderítésében, hogy melyik alkalmazás lett kijátszva. A naplózás önmagában nem, csak a benne megjelenő információkon keresztül válik hasznossá: általában a több részlet, hasznosabb.

Hibakezelés

A PHP biztonsági kérdések felől a hibajelzéseknek két oldaluk van. Az egyiket nézve hasznos a védelem növelése szempontjából, a másik szemszögből viszont káros.

Egy szokásos támadási technika minél több információ begyűjtése a rendszerről. Ezt úgy próbálják megoldani, hogy helytelen adatokat küldenek be, és rögzítik a hibaüzenetek típusait és környezetüket. Ez lehetőséget ad a crackernek, hogy elég információt gyűjtsön a rendszerről, és meghatározza a lehetséges gyenge pontokat. Ha például a támadó összeszedegetett elég információt

az előző űrlap kitöltések alapján, akkor megpróbálhatja a változókat felülírni vagy megváltoztatni őket:

Példa 4-11. Változók elleni támadás egy HTML oldallal

```
<form method="post" action="tamadas_celpontja?usernev=rosszize&jelszo=rosszize">
<input type="hidden" name="usernev" value="rosszize">
<input type="hidden" name="jelszo" value="rosszize">
</form>
```

A PHP által visszaadott hibaüzenetek általában hasznosak a hibákat kereső fejlesztő számára, megjelölve a fájlt, és a függvényt, ami hibás, megadva a megfelelő programsor számát. Ez az összes információ, amit ki lehet nyerni. Nem ritka, hogy egy PHP fejlesztő a show_source(), highlight_string(), vagy highlight_file() függvényeket a fejlesztés során hibakeresésre is használja, de egy élesben lévő webhelyen ez rejtett változókat, ellenőrizetlen kódokat, és más veszélyes információkat fedhet fel. Kifejezetten veszélyes beépített hibakezelővel rendelkező ismert forrású kódok használata. Ha a támadó ráismer valamilyen általános programozási technikára, akkor megpróbálhatja a nyers erőre alapozva feltörni az oldalt a különböző megszokott hibakereső (debugging) változók elküldésével:

Példa 4-12. Szokványos hibakereső változók kihasználása

```
<form method="post" action="tamadas_celpontja?errors=Y&amp;showerrors=1&debug=1">
<input type="hidden" name="errors" value="Y">
<input type="hidden" name="showerrors" value="1">
<input type="hidden" name="debug" value="1">
</form>
```

A hibakezelés módjától függetlenül az a lehetőség, hogy egy rendszerben hibák után kuthatnak, odavezet, hogy a támadók is több információhoz jutnak. Az általános hibaüzenetek nagyrészéből például beazonosítható, hogy a rendszer PHP-t használ. Ha a támadó egy .html oldalt látott, és ismert hibákat kihasználva meg akarta tudni, hogy milyen alkalmazást használ a rendszer, hibás adatokat beküldve azonosíthatja, hogy az oldalt egy PHP program állította elő.

Egy függvényhiba elárulhatja, hogy a rendszer milyen adatbázismotort használ, vagy hogy milyen programozói stílussal készült az adott weblap. Ez mélyebb kutatásokra ad lehetőséget nyitott adatbázisportok irányában, vagy tipikus hibák illetve gyengeségek keresését jelentheti. Különböző hibás adatok küldésével a támadó meg tudja állapítani, hogy milyen sorrendben végzed az azonosításokat (a hibák sorszámaiból). Ezzel a gyenge pontok is könnyen megtalálhatóak egy szkriptben.

A fájlrendszer vagy általános PHP hibák jelezhetik, hogy milyen jogokkal rendelkezik a webszerver, és megmutathatják a fájlok elrendezését és struktúráját. A fejlesztő által írt hibás kód súlyosbíthatja a helyzetet, régebben ’rejtett’ információk könnyű kiderítését téve lehetővé.

Három megoldási lehetőség adódik erre a problémára. Az első, megvizsgálni alaposan a függvényeket, és megpróbálni elkerülni a hibákat. A második a a hibajelzés kikapcsolása a teljes

kódon belül. A harmadik a PHP testreszabható hibajelentési funkcióinak használata, hogy saját hibakezelőket definiálj. A már megtett biztonsági intézkedésektől függően esetleg mindhárom fenti módszert választható.

Megelőzendő a bajt hasznot hajthat a PHP beépített error_reporting() függvénye, amely segít biztonságosabbá tenni a programokat és megtalálni a változók vészelyeket rejtő használati formáit. A bevezetés előtti tesztelés során E_ALL beálllítással gyorsan meg lehet találni azokat a pontokat, ahol a változók könnyen és/vagy rosszindulatúan módosíthatók. Ha a program már kész bevezetésére, akkor a E_NONE-t használva teljesen leszigetelhető a kód a további vizslatásoktól.

Példa 4-13. Veszélyes változók felderítése az E_ALL segítségével

```
<?php
if ($usernev) {  // nincs inicializálva vagy ellenőrizve használat előtt
    $jo_belepes = 1;
}
if ($jo_belepes == 1) { // ha az előző feltétel hamis, nincs inicializálva vagy el-
lenőrizve használat előtt
    fpassthru ("/nagyon/kenyes/adatok/index.html");
}
?>
```

Globálisan is elérhető változók (Register Globals) használata

A biztonság növelésére használható a PHP egyik lehetősége: a register_globals = off beállítás használata. Ennek a kikapcsolása azt jelenti, hogy a felhasználótól jövő változók nem "szennyezik be" akaratlanul a PHP kódot, és így csökken a potenciális támadó által befolyásolható változók köre. Még több időt kell azzal tölteniük, hogy kitalálják a változók feltöltésének módját, és a belső változóid hatékonyan elkülönülnek a felhasználó által elküldött adatoktól.

Az ezzel járó előnyök messze felülmúlják azt a kevés kényelmetlenséget, amelyet a megnövekedett munka mennyisége okoz.

Példa 4-14. register_globals=off nélkül

```
<?php
if ($usernev) {  // felhasználható által hamisítható get/post/cookies
    $jo_belepes = 1;
}

if ($jo_belepes == 1) { // felhasználható által hamisítható get/post/cookies
    fpassthru ("/nagyon/kenyes/adatok/index.html");
}
?>
```

Példa 4-15. register_globals = off használatával

```
<?php
if($HTTP_COOKIE_VARS[’usernev’]){
    // csak sütiként jöhet, hamisítva vagy épp ellenkezőleg
    $jo_belepes = 1;
    fpassthru ("/nagyon/kenyes/adatok/index.html");
}
?>
```

Okosan használva, még azt képes lehet jelezni, ha hamisítást kíséreltek meg. Ha előre tudható, hogy mely változóknak honnan kell érkezniük, akkor azt is megvizsgálhatod, hogy vajon más módon nem próbálták-e elküldeni ezt a változót. Ez nem garantálja, hogy az adatok nem hamisíthatók, azonban megköveteli a támadótól, hogy az rátaláljon a megfelelő hamisítási módszerre.

Példa 4-16. Egyszerű változó-átvétel felfedése

```
<?php
if ($HTTP_COOKIE_VARS[’usernev’] &&
    !$HTTP_POST_VARS[’usernev’] &&
    !$HTTP_GET_VARS[’usernev’] ) {
    // egyéb usernev azonosítások elvégzése ...
    $jo_belepes = 1;
    fpassthru ("/nagyon/kenyes/adatok/index.html");
} else {
   mail("admin@example.com", "Lehetséges betörési kísérlet", $HTTP_SERVER_VARS[’REMOTE_AD
   echo "Védelmi szabályok megszegése, adminisztrátor értesítve.";
   exit;
}
?>
```

Természetesen register_globals egyszerű kikapcsolása nem jelenti azt, hogy a kód biztonságos. Minden beérkező adatot valamilyen egyéb módon ellenőrizni kell.

Felhasználótól érkező adatok

A legtöbb probléma sok PHP programban nem a nyelvben rejlik, hanem abból fakad, hogy a kód nem a biztonságosságot szem előtt tartva készült. Emiatt mindig kellő időt kell szánni annak ellenőrzésére, hogy egy adott kódrészletre milyen hatással lehet egy váratlan hibás adat.

Példa 4-17. Veszélyes változóhasználat

```
<?php
// egy fájl törlése a user könyvtárából... vagy
// talán valaki másából?
unlink ($ordogi_valtozo);
```

```
// Az elérés naplózása... vagy egy /etc/passwd bejegyzésé?
fputs ($fp, $ordogi_valtozo);

// Valami egyértelmű dolog futtatása.. vagy rm -rf *?
system ($ordogi_valtozo);
exec ($ordogi_valtozo);
?>
```

Mindig alaposan meg kell vizsgálni a felhasználók által beadott adatokat, feltéve a következő kérdéseket:

- Biztos, hogy ez a szkript csak a kívánt fájlokat fogja módosítani?
- Előfordulhat egy ponton, hogy szokatlan vagy nem kívánatos adat jelenjen meg?
- Használható-e az adott szkript nem kívánatos formában?
- Felhasználható-e más szkriptekkel együtt egy negatív hatás elérésére?
- Megfelelően naplózásra kerülnek-e a tranzakciók (elérések, változtatások)?

Kellőképpen átgondolva a fenti kérdéseket a szkript írásakor, megkímélhet attól, hogy később végiggondolva a problémákat szerencsétlen módon újra kelljen írni a teljes kódot a védelem növelése érdekében. Ezzel sem lehet garantálni a rendszer biztonságát, de segíthet annak növelésében/ fenntartásában.

Számításba lehet venni a register_globals, magic_quotes és más kényelmi szolgáltatások kikapcsolásának a gondolatát is, mivel ezek megfosztanak az adatok forrásának, helyességének, tartalmának ismeretétől. A PHP*t maximális hibajelentési szinten használva - az error_reporting E_ALL beállításával - figyelmeztetést ad előzetes érték nélküli, definiálatlan változókról, ezzel védve attól, hogy véletlenül hibás adatokkal dolgozzon a program.

A PHP elrejtése

Általában a rejtegetésen és félrevezetésen alapuló védelem az egyik leggyengébb formája a védekezésnek, azonban néha minden apró többlet kívánatos lehet.

Néhány egyszerű módszerrel elrejthető, hogy PHP-t használsz, így lassítva le a támadót, aki fel akarja deríteni a rendszer gyenge pontjait. A `php.ini`-ben az expose_php = off beállítással csökkentheted ezeket az információkat.

Másik taktika a webszerver (pl. apache) olyan beállítása a .htaccess-en vagy az apache konfigurációs fájlában, hogy különböző típusú fájlokat is PHP-n keresztül futtasson. Más félrevezető fájlkiterjesztéseket alkalmazására példa:

Példa 4-18. PHP elrejtése más nyelvként

```
# úgy tűnik, mintha PHP helyett más szerver oldali alkalmazás futna
AddType application/x-httpd-php .asp .py .pl
```

vagy egy teljesen ismeretlennel is eltakarható:

Példa 4-19. ismeretlen típusok használata PHP fájlok kiterjesztéseként

```
# PHP kódok teljesen ismeretlen típusúnak tűnnek
AddType application/x-httpd-php .bop .foo .133t
```

vagy html típus mögé is elrejthetők a PHP fájlok, ami csekély teljesítménycsökkenést okoz, mivel minden html oldal átmegy a PHP-n:

Példa 4-20. html típus használata PHP fájlkiterjesztésként

```
# PHP kódok html típusúnak tűnnek
AddType application/x-httpd-php .htm .html
```

Ahhoz, hogy ez jól működjön, minden PHP fájlt át kell nevezni a fenti kiterjesztések valamelyikének megfelelően. Mivel ez is az elrejtőzésen / ismeretlenségen alapuló védelmi forma, kevés hátránya mellett csekély megakadályozó intézkedést jelent.

Fontos aktuálisnak maradni

A PHP, mint bármilyen más nagy rendszer állandóan változások és fejlesztések alatt áll. Minden új változat kisebb-nagyobb változtatásokat tartalmaz, feljesztve a nyelvet, kijavítva biztonsági hibákat, beállítási kellemetlenségeket, és más olyan elemeket, amik a teljes rendszer stabilitására és biztonságára hatnak.

Mint más rendszerszintű nyelvek és programok esetében, a legjobb hozzáállás a gyakori frissítés, valamint a friss változatokról, és a fellépő változásokról való informálódás.

Rész II. A nyelv alapjai

Fejezet 5. Alapvető szintaxis

Escape szekvencia HTML-ben

Amikor a PHP feldolgoz egy fájlt, akkor egyszerűen a kimenetre másol minden karaktert, amíg nem találkozik valamelyik speciális jelöléssel (lásd alább!), amelyet PHP kódrészlet kezdeteként értelmez. Ekkor a PHP futtatni kezdi a zárójelölésig található összes kódot. Ezzel a működési elvvel lehet könnyen HTML oldalakba elhelyezni PHP kódot, mivel mindent, ami a PHP nyitó- és zárójelölésein kívül esik, teljes mértékben békénhagy, és csak ezeken belüli tartalmat értelmezi programkódként.

Négyféle jelöléssel lehet az állomány bizonyos részeit PHP kódként megjelölni. Ezek közül csak kettő használható bármilyen esetben: <?php. . .?> and <script language="php">. . .</script> a másik kettő ki és bekapcsolható a `php.ini` konfigurációs fájlban. Bár a rövid vagy az ASP-stílusú formák kényelmesnek tűnnek, egyszersmind kevésbé hordozhatók, mint a hosszabb változatok. Emellett, ha XML vagy XHTML fájlokba kell PHP kódot illeszteni, azt csak <?php. . .?> alakkal lehet - igazodva az XML ajánláshoz.

Példa 5-1. A PHP-módba kerülés lehetőségei

```
1.  <? echo ("Ez a legegyszerűbb, egy SGML processing utasítás\n"); ?>
    <?= $valtozo; # Ez egy rövidítése a "<? echo ..?>"-nak ?>

2.  <?php echo("Ha XHTML vagy XML dokumentumokat is akarsz szolgáltatni," .
               "biztos szeretni fogod ezt\n"); ?>

3.  <script language="php">
        echo ("Néhány szerkesztő (ilyen pl. a FrontPage) nem" .
              "szereti a processing utasításokat");
    </script>

4.  <% echo ("Használhatsz ASP-stílusú tag-eket"); %>
    <%= $valtozo; # Ez egy rövidítése a "<% echo ..%>"-nak %>
```

Az első lehetőség csak akkor elérhető, ha a rövid tag-ek engedélyezve vannak. Engedélyezhetjük - PHP 3-ban - **short_tags()** függvényhívással, a short_open_tag beállítással a PHP konfigurációs fájlban, vagy a PHP fordításánál a **configure** program --enable-short-tags opciójával.

A második lehetőség az általánosan ajánlott módszer, mivel lehetőséget ad még az új generációs XHTML dokumentumok előállítására is.

A negyedik mód csak akkor elérhető, ha az ASP-stílusú jelölés is engedélyezve van az asp_tags konfigurációs beállítással.

> **Megjegyzés:** Az ASP-stílusú tagek a 3.0.4. verziótól használhatóak

A lezárójelölés magában foglalja a jelölést közvetlenül követő újsort karaktert, ha van ilyen, emellett a PHP blokk utolsó sorát lezáró pontosvesszőt (;).

A PHP a következő struktúrák használatát is megengedi:

Példa 5-2. HTML blokkok feltételhez kötése

```
<?php if ($igazez) { ?>
<strong>Igaz a változó tartalma</strong>
<?php } else { ?>
<strong>Hamis a változó tartalma</strong>
<?php } ?>
```

Ez a várakozásoknak megfelelően működik, mivel a PHP ?> lezáró jelölést követően mindent átmásol a kimenetre, amíg egy újabb nyitó jelöléssel. A fenti példa természtesen mondvacsinált, de nagy méretű szövegblokkok kiíratásakor hatékonyabbnak bizonyulhat ideiglenesen elhagyni a PHP módot, mint mindezt a szöveget echo(), print() vagy ehhez hasonló függvényekkel kiíratni.

Utasítások elválasztása

Az utasítások - a C és a perl nyelvhez hasonlóan - pontosvesszővel végződnek.

A záró tag (?>) szintén feltételezi az utasítás végét, tehát az alábbiak egyenértékűek:

```
<?php
    echo "Ez egy teszt";
?>

<?php echo "Ez egy teszt" ?>
```

Kommentek

A PHP támogatja a ’C’, ’C++’ és Unix shell-szerű kommenteket. Például:

```
<?php
    echo "Ez egy teszt"; // Ez egy egysoros c++ szerű komment
    /* Ez egy többsoros komment
       Még egy sor komment */
    echo "Ez egy másik teszt";
    echo "Ez az utolsó teszt"; # Ez egy shell-szerű komment
?>
```

Az "egysoros" kommentek valójában csak a sor végéig, vagy az aktuális PHP kód végéig tartanak, attól függően, hogy melyik jön előbb.

```
<h1>Ez egy <?php# echo "egyszerű";?> példa.</h1>
<p>A fenti fejléc kiírja ’Ez egy egyszerű példa’.
```

Figyelj rá, hogy ne ágyazd egymásba a ’C’-stílusú kommenteket, ami akkor történhet, ha egy hosszú programblokkot kikommentezel.

```
<?php
 /*
    echo "Ez egy teszt"; /* Ebből a kommentből baj lesz */
 */
?>
```

Fejezet 6. Típusok

Bevezető

A PHP nyolc primitív típust támogat.

A négy skalár típus:

- boolean (logikai)
- integer (egész szám)
- floating-point number (float, lebegőpontos szám)
- string (karakterlánc, karaktersorozat)

A két összetett típus:

- array (tömb)
- object (objektum)

Végül két speciális típus:

- resource (erőforrás)
- NULL

> **Megjegyzés:** Ebben a kézikönyvben gyakran találkozhatsz `mixed` paraméterekkel is. Ez a kvázi-típus csak a dokumentációkban létezik, annak jelölésére szolgál, hogy többféle lehetőség adott arra a paraméterre.

A változó típusát rendszerint nem a programozó adja meg [persze van beleszólása...], hanem a PHP futási időben határozza meg a változó környezetétől függően.

> **Megjegyzés:** Ha egy kifejezés értékére és/vagy típusára vagy kíváncsi, akkor használd a var_dump() függvényt.
>
> Ha csak a típusát akarod megkapni emberek számára olvasható formában, akkor használd a gettype()-ot. Típusellenőrzésre viszont *ne* ezt, hanem a `is_type` függvényeket használd a programokban.

Ha egy változó típusát egy adott típusra kell konvertálnunk, castolhatjuk a változót, vagy alkalmazzuk rá a settype() függvényt.

A változó különbözőképp viselkedhet bizonyos helyzetekben, attól függően, hogy az adott pillanatban milyen típusú. Ha bővebb leírást szeretnél, nézd meg a Bűvészkedés a típusokkal című részt.

Logikai adattípus

Ez a legegyszerűbb típus. Egy boolean igazságértéket fejez ki. Lehet vagy `TRUE` (igaz), vagy `FALSE` (hamis).

> **Megjegyzés:** A logikai adattípus a PHP 4-esben került bevezetésre.

Szintaxis

Egy logikai érték megadásához használhatod a `TRUE` vagy `FALSE` szavakat, szükség szerint. Egyik jelentése sem függ a kis- és nagybetűs írásmódtól.

```
$igaz = True; // a logikai igaz értéket adjuk az $igaz változónak
```

Tipikus valamilyen operátor használatakor kapsz boolean típusú értéket, amit egy vezérlési szerkezetben fel tudsz használni.

```
if ($akcio == "verzio_kiirasa") { // a == operátor boolean értékkel tér vissza
    echo "Ez az 1.23-as változat";
}

// ez nem szükséges
if ($elvalaszto_kiirasa == TRUE) {
    echo "<hr>\n";
}

// mivel egyszerűen ez is működik
if ($elvalaszto_kiirasa) {
    echo "<hr>\n";
}
```

Logikai értékké alakítás

Ha kifejezetten boolean típusúvá szeretnél alakítani egy értéket, használd a `(bool)` vagy a `(boolean)` típusátalakítást. A legtöbb esetben azonban nem kell ezt alkalmaznod, mivel az érték automatikusan átalakul, ha egy operátor, függvény, vagy vezérlési szerkezet boolean típusú argumentumot vár.

Lásd még a Bűvészkedés a típusokkal című részt.

Amikor boolean típusúvá kell alakítani egy értéket, az alábbiak `FALSE` értéket adnak:

- a boolean típusú `FALSE`
- az integer (egész)típusú 0 (nulla)
- a float (lebegőpontos) 0.0 (nulla)
- az üres string, és a "0" string
- egy elemeket nem tartalmazó array (tömb)

- egy elemeket nem tartalmazó object (objektum)
- a speciális NULL érték (beleétve a nem beállított - hivatkozást megelőzően nem definiált - változókat)

Minden más érték `TRUE` lesz (beleértve bármilyen resource (előforrás) típusú értéket).

> **Figyelem**
>
> A `-1` is `TRUE` lesz, mint minden más nem nulla (akár negatív, akár pozitív) szám!

Egész számok

Egy integer a következő halmaz része: Z = {..., -2, -1, 0, 1, 2, ...}.

Lásd még a Tetszőleges pontosságú egészek és a Lebegőpontos számok című részeket.

Szintaxis

Az egészek megadhatók decimális (10 alapú), hexadecimális (16 alapú) vagy oktális (8 alapú) számként, opcionális előjellel (- vagy +).

Ha az oktális formát választod, a számot egy `0` (nulla) jeggyel kell kezdened, ha a hexadecimálisat, akkor `0x`-el.

Példa 6-1. Egész értékek

```
$a = 1234; # decimális szám
$a = -123; # negatív szám
$a = 0123; # oktális szám (megegyezik a 83 decimális számmal)
$a = 0x1A; # hexadecimális szám (megegyezik a 26 decimális számmal)
```

Az egészek maximális mérete operációs rendszertől függ, az átlagos érték a két milliárd (32 bites előjeles egész).

Egészek értelmezési határának túllépése

Az integer típus értelmezési tartományán kívül eső egész értékek float típussá alakulnak. Ha valamely művelet eredménye kívül esik a integer értelmezési tartományán, akkor az eredmény automatikusan float típusúvá konvertálódik.

```
$nagy_szam =  2147483647;
var_dump($nagy_szam);
// kimenete: int(2147483647)
$nagyobb_szam =  2147483648;
```

```
var_dump($nagyobb_szam);
// kimenete: float(2147483648)

// ez működik hexadecimálisan megadott egészekre is:
var_dump( 0x80000000 );
// kimenete: float(2147483648)

$millio = 1000000;
$nagy_szam =  50000 * $millio;
var_dump($nagy_szam);
// kimenete: float(50000000000)
```

> **Figyelem**
>
> Sajnálatosan meg kell említenünk, hogy a PHP 4.0.6-ban ez az átalakítás nem működött mindig pontosan negatív számok használatakor, például: ha a `-50000 * $millio` műveletet eredménye: `-429496728`. Ha mindkét operandus pozitív, nincs semmi probléma.
>
> Ezt a hiba ki lett javítva a PHP 4.1.0-ben.

PHP-ben nincs egész osztás. Az `1/2` művelet a `0.5` float (lebegépontos) értéket eredményezi.

```
var_dump( 25/7 );
// output: float(3.5714285714286)
```

Egész értékké alakítás

Ha kifejezetten integer típusúvá szeretnél alakítani egy értéket, használd az `(int)` vagy az `(integer)` típusátalakítást. A legtöbb esetben azonban nem kell ezt alkalmaznod, mivel az érték automatikusan átalakul, ha egy operátor, függvény, vagy vezérlési szerkezet integer típusú argumentumot vár.

Lásd még a Bűvészkedés a típusokkal című részt.

Átalakításboolean (logikai) értékekről

A `FALSE` (hamis) érték a `0` (nulla) egész számmá alakul, a `TRUE` (igaz) érték az `1` (egy) egész számot adja.

Átalakítás lebegőpontos értékekről

Lebegépontos számok egész értékké alakításakor a szám tört része elvész, azaz *lefelé kerekítés* történik.

Ha a lebegőpontos szám az egész tartományon kívül esik (általában `+/- 2.15e+9 = 2^31`), az eredmény nem definiált, mivel a lebegőpontos szám nem rendelkezik elég precizitással, hogy pontos egész eredményt kapj. Sem warning, sem notice szintű hiba nem lép fel ebben az esetben!

> **Figyelem**
>
> Soha ne alakíts egy ismeretlen törtszámot integer típusúvá, mivel ez időnként nem várt eredményekhez vezethet.
>
> ```
> echo (int) ( (0.1+0.7) * 10 ); // kiírja, hogy 7!
> ```
>
> Lásd a a lebegőpontos számok pontotsságának problémái című részt.

Átalakítás karakterláncokról

Lásd a String átalakítás című részt.

Átalakítás más típusokról

> **Figyelem**
>
> Az egésszé alakítás viselkedése más típusokra nem definiált. Jelenleg ezek az átalakítások megegyeznek azzal, mintha először logikai, majd utána egész értékké alakítottad volna a kiindulási értéket. Erre a viselkedeésre azonban *nem* szabad építeni, mivel minden figyelmeztetés nélkül megváltozhat.

Lebegőpontos számok

A lebegőpontos számok ("float", "valós szám") az alábbi szintaxisok bármelyikével hozhatóak létre:

```
$a = 1.234; $a = 1.2e3; $a = 7E-10;
```

A lebegőpontos számok mérete platformfüggő, de a maximális érték körülbelül 1.8e308, durván 14 tizedesjegy pontossággal (64 bites IEEE formátum).

Lebegőpontos számok pontossága

Elég gyakran előfordul, hogy egyszerű decimális törtek, mint a `0.1` és a `0.7` nem konvertálhatóak pontosan bináris megfelelőikre kis veszteség nélkül. Ez zavaró eredményekhez vezethet, például `floor((0.1+0.7)*10)` tipikusan `7` értékkel tér vissza, az elvárt `8` helyett, a belső ábrázolás miatt, ami valójában `7.9999999999...`.

Ez azzal a ténnyel van kapcsolatban, hogy lehetetlen megjeleníteni néhány törtet véges számú decimálisan számjeggyel. Például `1/3` decimálisan `0.3333333. . .` értékké válik.

Ezért soha ne bízz meg a törtszámok eredményeiben az utolsó jegyig, és soha se hasoníts össze két lebegőpontos számot pontos egyenlőségre. Ha nagyobb pontosságú számokra van szükséges, használd a tetszőleges pontosságú matematikai függvényeket vagy a gmp függvényeket.

Stringek

A string karakterek sorozata. PHP-ben egy karakter pontosan egy bytenak felel meg, így 256 különböző karakter lehetséges. Ez azt is jelenti, hogy a PHP-nek jelenleg nincs beépített Unicode támogatása.

> **Megjegyzés:** Nem okozhat problémát a stringek körében, hogy túl hosszúvá válnak. Semmiféle korlát nem létezik a PHP által kezelt stringek méretére, ezért nem kell tartani a hosszú stringektől.

Szintaxis

A stringeket háromféleképpen lehet létrehozni.

- aposztróffal
- idézőjellel
- heredoc szintaxissal

String létrehozása aposztróffal

A legkönnyebben úgy adhatunk meg egy egyszerű stringet, hogy aposztrófok (`'` karakterek) közé tesszük.

Ha a stringben egy aposztófot szeretnél elhelyezni, és azzal nem a string végét szeretnéd jelezni, más nyelvekhez hasonlóan egy visszaperjel karaktert kell alkalmaznod az aposztóf előtt (`\`). Ha egy aposztróf, vagy a string vége előtt el szeretnél helyezni egy visszaperjelet, meg kell dupláznod azt. Figyelj arra, hogy ha bármilyen más karakter elé teszel visszaperjelet, a visszaperjel meg fog jelenni a stringben. Emiatt gyakran nincs szükség magának a visszaperjelnek a duplázására.

> **Megjegyzés:** PHP 3 használatakor azonban egy `E_NOTICE` szintű figyelmeztetést kapsz, ha ezt kihasználod.

> **Megjegyzés:** A többi móddal ellentétben a változók értékei *nem* helyettesítődnek be, ha aposztrófos stingben változókra hivatkozol.

```
echo ’Egyszerű string’;
echo ’A stringekbe újsor karaktereket is építhetsz,
ilyen formában.’;
echo ’Arnold egyszer azt mondta: "I\’ll be back"’;
// kimenete: ... "I’ll be back"
echo ’Are you sure you want to delete C:\\*.*?’;
// kimenete: ... delete C:\*.*?
echo ’Are you sure you want to delete C:\*.*?’;
// kimenete: ... delete C:\*.*?
echo ’Ezen a ponton próbálok beépíteni \n egy újsort’;
// kimenet: ... beépíteni \n egy újsort
```

String létrehozása idézőjellel

Ha egy stringet idézőjelek (") közé helyezünk, a PHP több speciális jelölés feldolgozására lesz képes:

Táblázat 6-1. Speciális jelölések idézőjeles stringben

jelölés	jelentése
`\n`	újsor (LF vagy 0x0A (10) ASCII kódú karakter)
`\r`	kocsivissza (CR vagy 0x0D (13) ASCII kódú karakter)
`\t`	vízszintes tabulátor (HT vagy 0x09 (9) ASCII kódú karakter)
`\\`	visszaperjel
`\$`	dollárjel
`\"`	idézőjel
`\[0-7]{1,3}`	egy karaktersorozat, ami oktális számokra illeszkedik
`\x[0-9A-Fa-f]{1,2}`	egy karaktersorozat, ami hexadecimális számokra illeszkedik

Ha bármilyen más karakter elé visszaperjelet írsz, ebben az esetben is ki fog íródni a visszaperjel.

A legfontosabb előnye az idézőjeles stringeknek az, hogy a változók behelyettesítésre kerülnek. Lásd a változók behelyettesítése című részt további részletekért.

String létrehozása heredoc szintaxissal

Egy másfajta módszer a stringek megadására a heredoc szintaxis ("<<<"). [ez itt megint nem elírás, kedves unix-shell programozó!] A <<< jelzés után egy azonosítót kell megadni, majd a stringet, és végül az azonosítót mégegyszer, ezzel zárva le a stringet.

A lezáró azonosítónak *mindenképpen* a sor legelső karakterén kell kezdődnie. Ugyancsak figyelni kell arra, hogy ez az azonosító is az általános PHP elemek elnevezési korlátai alá tartozik: csak alfanumerikus karaktereket és aláhúzást tartalmazhat, és nem kezdődhet számjegy karakterrel.

> **Figyelem**
>
> Nagyon fontos, hogy odafigyelj arra, hogy a lezáró azonosítót tartalmazó sor ne tartalmazzon semmi mást, csupán *esetleg* egy idézőjel (`;`) karaktert. Ez még pontosabban azt is jelenti, hogy az azonosító *nem lehet beljebb kezdve*, és nem szabad semmilyen szóköz vagy tabulátor karaktert sem elhelyezni a pontosvessző előtt vagy után.
>
> Feltehetően a legkellemetlenebb korlátozás az, hogy még egy kocsivissza (`\r`) karakter sem lehet a sor végén, csak egy újsor (`\n`) szerepelhet. Mivel a Microsoft Windows a `\r\n` sorozatot használja a sorok végének jelzésére, a heredoc nem fog működni, ha olyan szerkesztőben készíted a programod, ami ezt a sorvég jelzést használja. A legtöbb fejlesztőeszköz azonban lehetőséget ad UNIX sorvégjelzéssel ellátott állományok mentésére.

A heredoc az idézőjeles stringekhez hasonlóan működik, az idézőjelek nélkül. Ez azt jelenti, hogy nem kell visszaperjellel jelülni az idézőjeleket a szövegben, de a fenti speciális jelölések használhatóak. A változók értékei behelyettesítődnek, de az idézőjeles karaktersorozatokhoz hasonlóan gondosan kell ügyelni a komplex változó hivatkozások megadására.

Példa 6-2. "Heredoc" string példa [VAS: Vége A Stringnek de helyette lehet bármi]

```
<?php
$str = <<<VAS
Példa egy stringre, amely
több sorban van, és heredoc
szintaxisú
VAS;

/* Komplexebb példa, változókkal. */
class ize
{
    var $ize;
    var $valami;

    function ize()
    {
        $this->ize = ’Valami’;
        $this->valami = array(’elso’, ’masodik’, ’harmadik’);
    }
}

$ize = new ize();
$nev = ’Béla’;

echo <<<VAS
```

```
A nevem "$nev". Kiírok egy értéket: $ize->ize.
Most kiírok egy tömbelemet: {$ize->valami[1]}.
Ez nagy ’A’ kell, hogy legyen: \x41
VAS;
?>
```

Megjegyzés: A heredoc a PHP 4-esben került a nyelvbe.

Változók behelyettesítése

Ha egy stringet idézőjelek között, vagy heredoc szintaxissal adsz meg, a jelölt változók értékei behelyettesítésre kerülnek.

Kétféleképpen lehet megadni egy változót, az egyszerű és a komplex formátummal. Az egyszerű forma a leggyakoribb és legkényelmesebb. Lehetőséget ad egy skalár, tömb érték vagy egy objektum tulajdonság beillesztésére.

A komplex szintaxis a PHP 4-es változatában került a nyelvbe, és a jelölésben használatos kapcsos zárójelekről ismerhető fel.

Egyszerű szintaxis

Ha dollár ($) jelet talál a PHP egy stringben, mohón megkeresi az összes ezt követő azonosítóban is használható karaktert, hogy egy érvényes változónevet alkosson. Használj kapcsos zárójeleket, ha pontosan meg szeretnéd határozni, meddig tart egy változó.

```
$ingatlan = ’ház’;
echo "kertes $ingatlan kerítéssel"; // működik, szóköz nem lehet változónévben
echo "páros $ingatlanszám"; // nem működik, az ’s’ és további karakterek lehet-
nek változónévben
echo "páros ${ingatlan}szám"; // működik, mert pontosan meg van adva a név
```

Hasonlóképpen meg lehet adni tömbindexet vagy objektum tulajdonságot is. A tömbindexek esetében a záró szögletes zárójel (]) jelöli az index végét, az objektum tulajdonságoknál az egyszerű skalárok szabályai érvényesek, habár objektum tulajdonágok esetén nem használható a fenti trükk.

```
$gyumolcsok = array( ’eper’ => ’piros’ , ’alma’ => ’zöld’ );
echo "Az alma $gyumolcsok[alma]."; // ez másképpen használandó karaktersoroza-
tokon kívül.
// Lásd az $ize[valami] karakterláncon kívüli problémáját

echo "A négyzet $negyzet->szelesseg méter széles.";

echo "A négyzet $negyzet->szelesseg00 centiméter széles."; // nem működik
// A megoldás érdekében lásd a komplex szintaxis szakaszt!
```

Bármely ennél komplexebb helyettesítéshez a komplex szintaxis használatos.

Komplex (kapcsos zárójeles) szintaxis

Ezt nem azért nevezzük komplexnek, mert a szintaxis komplex, hanem azért, mert így komplex kifejezések helyettesítésére nyílik lehetőség :).

Gyakorltilag bármilyen változó érték behelyettesíthető ezzel a szintaxissal. Egyszerűen úgy kell megadni az értéket adó kifejezést, mintha a stringen kívül dolgoznál vele, és utána { és } jelek közé kell zárni. Mivel a '{' jel megadására nincs speciális jelölés, ez a forma csak akkor fog működni, ha a { után közvetlenül egy $ jel található. (Használhatod a "{\$" vagy "\{$" jelöléseket, ha a stringben a "{$" sorozatot szeretnéd beilleszteni, és nem változóhelyettesítést adsz meg). Néhány példa, ami tisztázhatja a félreértéseket:

```
$oriasi = 'fantasztikus';
echo "Ez { $oriasi}"; // nem működik, kiírja, hogy : Ez { fantasztikus}
echo "Ez {$oriasi}";  // működik, kiírja, hogy: Ez fantasztikus
echo "Ez a négyzet {$negyzet->szelesseg}00 centiméter széles.";
echo "Működik: {$tomb[4][3]}";
echo "Hibás: {$tomb[ize][3]}";
   // ennek ugyanaz az oka, mint ami miatt
   // a $ize[valami] hibás egy stringen kívül

echo "Ebben a formában már helyes: {$tomb['ize'][3]}";
echo "Akár ezt is írhatod: {$obj->ertekek[3]-&gt;nev}";
echo "Ez a $nev nevű változó értéke: {${$nev}}";
```

String karaktereinek elérése

A string karaktereire nullától számozott indexekkel lehet hivatkozni, a string neve után megadott kapcsos zárójelek között.

> **Megjegyzés:** Kompatibilitási okokból a tömböknél használatos szögletes zárójelek is alkalmazhatóak a karakterek eléréséhez. Ez a tömb jelzés azonban nem javasolt a PHP 4-esben.

Példa 6-3. Néhány string példa

```
<?php
/* String hozzárendelése */
$str = "Ez egy string";

/* Hozzáfűzés */
```

```
$str = $str . " több szöveggel";

/* Hozzáfűzés egy más módja újsor karakterrel a végén */
$str .= " és egy újsor a végére.\n";

/* Ez a string végülis ilyen lesz: '<p>Szám: 9</p>' */
$num = 9;
$str = "<p>Szám: $num</p>";

/* Ez meg: '<p>Szám: $num</p>' */
$num = 9;
$str = '<p>Szám: $num</p>';

/* A string első karaktere kerül first-be  */
$str = 'Ez egy teszt.';
$first = $str{0};

/* Most meg az utolsó */
$str = 'Ez még mindig egy teszt';
$last = $str{strlen($str)-1};
?>
```

Hasznos függvények

Stringeket a '.' (pont) stringösszefűző operátorral tudsz összefűzni. A '+' (összeadás) operátor nem alkalmas erre a feladatra. Lásd a String operátorok című részt további információkért.

Számos hasznos függvény létezik stringek manipulálásához.

Lásd a string függvények című részt általános függvényekért, a reguláris kifejezés függvényeket fejlett kereséshez és cserékhez (két formában: Perl és POSIX kiterjesztett).

Külön függvények léteznek URL stringekhez, és stringek kódolásához/dekódolásához (mcrypt és mhash kódolások).

Végül ha ezek között sem találod, amit keresel, lásd a karakter típus függvényeket.

String konverziók

Amikor egy string számként értékelődik ki, az eredmény típusa a következőképp kerül meghatározásra.

A string lebegőpontosként értékelődik ki, ha a '.', 'e', vagy 'E' karakterek bármelyikét tartalmazza. Egyébként egészként értékelődik ki.

Az érték meghatározása a string első karakterei alapján történik. Ha a string érvényes számmal kezdődik, ez az érték kerül felhasználásra. Máskülönben az érték 0 lesz [nulla, nem nagy O betű]. Érvényes számnak tekintendő, amelynek opcionális előjelét egy vagy több számjegy követi (melyben lehet egy tizedespont), végül lehet exponense. Az exponens egy 'e' vagy 'E' karakter, melyet egy vagy több szám követ.

Ha az első tag string, a változó típusát a második kifejezés határozza meg.

```
$foo = 1 + "10.5";              // $foo most float (11.5)
$foo = 1 + "-1.3e3";            // $foo most float (-1299)
$foo = 1 + "bob-1.3e3";         // $foo most integer (1)
$foo = 1 + "bob3";              // $foo most integer (1)
$foo = 1 + "10 Kicsi Pingvin";  // $foo most integer (11)
$foo = 1 + "10 Kicsi Pingvinke" // $foo most integer (11)
$foo = "10.0 disznó " + 1;      // $foo most integer (11)
$foo = "10.0 disznó " + 0.5;    // $foo most float (10.5) [szóval 10 disznó, meg egy félc
```

Ha többet szeretnél megtudni erről a konverzióról, akkor nézd meg a Unix manualt strtod(3). [=RTFM]

Ha szeretnéd kipróbálni valamelyik fejezetbeli példát, elmásolhatod innen a példákat és beszúrhatod az alábbi sort, hogy lásd mi is történik.

[Ahhoz nem árt azért előbb érteni a beszúrt sort. A beszúrandó string elején a "\" azért van, hogy a $foo helyére NE a foo változó értéke kerüljön, hanem a \$ hatására a $ karakter íródjon ki, majd a foo karaktersor, egy egyenlőségjel, és utána a $foo változó értéke. Az első "\" hatására ugyanis a köv. karakter ("$") elveszti különleges jelentését, és kimásolódik a kimenetre, majd jön a "foo=" szöveg, ez is úgy, ahogy van a kimenetre kerül, a fordító nem is találkozott változóval, amit be kellene helyettesíteni. Aztán jön a $foo, ezt már felismeri a fordító, és a változó értékét írja ki]:

```
echo "\$foo==$foo; típusa " . gettype ($foo) . "<br>\n";
```

Tömbök

A PHP tömbjei rendezett leképezéseket valósítanak meg. A leképezés *értékeket* rendel *kulcsokhoz*. Ez a típus sokféleképpen használható, mint egy hagyományos tömb, egy lista (vektor), hash tábla, szótár, kollekció, halmaz, sor, és mások. Mivel egy újabb PHP tömb szerepelhet értékként, könnyen szimulálhatsz fákat.

Az említett struktúrák leírása ezen kézikönyv kereteibe nem fér bele, de legalább egy példát mutatunk mindegyikre. További információkért külső forrásokat kell igénybe venned erről az igen széles témáról.

Szintaxis

Tömb létrehozása az array() nyelvi elemmel

Egy array (tömb) típusú változót az array() nyelvi elemmel tudsz létrehozni, amely számos vesszővel elválasztott *`kulcs => érték`* párt vár.

Egy `kulcs` vagy egy nemnegatív integer (egész szám) vagy egy string. Ha a kulcs egy szabványos formában megadott nemnegatív egész szám (integer), az megfelelő módon kerül értelmezésre. Konkrétabban pl. a `'8'` a `8` egész szám lesz, míg a `'08'` a `'08'` string.

Az érték bármi lehet.

Ha nem adsz meg egy kulcsot, a PHP veszi az egész indexek közül a legnagyobbat, és ennél egyel nagyobb indexű elemet hoz létre. Ha még nincs egész értékkel megadott index, az új kulcs `0` (nulla) lesz. Ha olyan kulcsot adsz meg, ami már a tömb eleme, a korábbi elem felülíródik.

```
array( [kulcs =>] érték
     , ...
     )
// a kulcs vagy egy string vagy egy nemnegatív integer (egész szám)
// az érték bármi lehet
```

Létrehozás/módosítás a szögletes zárójeles formával

Meglévő tömbök is módosíthatóak konkrét elemek megadásával.

Ezt úgy tudod megtenni, hogy a tömb neve után szögletes zárójelekben megadod a kulcsot, amit módosítani szeretnél. Ha elhagyod a kulcsot, és csak egy üres szögletes zárójel párt ("`[]`") adsz meg a változó neve után, a tömb végére illeszthetsz elemet.

```
$tomb[kulcs] = érték;
$tomb[] = érték;
// a kulcs vagy egy string vagy egy nemnegatív integer (egész szám)
// az érték bármi lehet
```

Ha a `$tomb` nem létezik, ezzel létrejön. Tehát ez egy újabb alternatíva tömbök létrehozására. Ha módosítani szeretnél egy elemet, rendelj hozzá új értéket. Ha meg szeretnél szüntetni egy kulcs/érték párt, használd az unset() függvényt.

Hasznos függvények

Jónéhámy függvény létezik a tömbökkel való munka megkönnyítésére. Lásd a tömb függvények című részt.

> **Megjegyzés:** Az unset() függvénnyel lehet egy értéket a tömbből törölni. Figyelj arra, hogy a tömb nem lesz újraindexelve!

```
$a = array( 1 => 'egy', 2 => 'kettő', 3 => 'három' );
unset( $a[2] );
/* ennek eredményeképpen $a így fog kinézni:
   $a = array( 1=>'egy', 3=>'három');
   és NEM így:
   $a = array( 1 => 'egy', 2 => 'három');
*/
```

A foreach vezérlési szerkezet kifejezetten a tömbök számára jött létre. Egy egyszerű módszert ad tömbökön való végiglépkedésre.

Mit tehetünk, és mit nem a tömbökkel

Miért nem jó az `$size[valami]` forma?

Talán láttad ezt a szintaxist régi PHP programokban:

```
$ize[valami] = 'ellenség';
echo $ize[valami];
// stb.
```

Ez hibás, és mégis működik. Akkor mégis miért nem jó? Az ok az, ami már a szintaxis részben is olvasható volt, hogy a szögletes zárójelek ('`[`' és '`]`') között egy kifejezésnek kell állnia. Ez azt jelenti, hogy írhatsz ehhez hasonló dolgokat is:

```
echo $tomb[ ize(true) ];
```

Ez a példa bemutatja, hogyan használhatsz függvény visszatérési értéket tömbindexként. A PHP úgyszintén ismeri a konstansokat. Bizonyára hallottál már az `E_*` konstansokról.

```
$hiba_leiras[E_ERROR]   = "Fatális hiba történt";
$hiba_leiras[E_WARNING] = "A PHP figyelmeztetést adott";
$hiba_leiras[E_NOTICE]  = "Informális megjegyzés";
```

Figyeld meg, hogy az `E_ERROR` egy érvényes azonosító, mint a `valami` az előző példában. De a legutóbbi példa ugyanaz, mintha ezt írnánk:

```
$hiba_leiras[1] = "Fatális hiba történt";
$hiba_leiras[2] = "A PHP figyelmeztetést adott";
$hiba_leiras[8] = "Informális megjegyzés";
```

mivel az `E_ERROR` konstans értéke `1`, stb.

Akkor hogyan lehet mégis, hogy az `$size[valami]` működik? Nos, azért, mert `valami` a szintaxisa következtében egy konstanst határoz meg. Azonban ebben az esetben `valami` nevű konstans nem

létezik. A PHP ebben az esetben azt feltételezi, hogy a `valami` karaktersorozatra gondolsz (`"valami"`), csak elfejeltetted kitenni az idézőjeleket.

De miért nem jó ez?

Valamikor a jövőben a PHP fejlesztői hozzáadhatnak egy új konstanst vagy kulcsszót a nyelvhez, és akkor bajba kerülsz. Például jelenleg sem használhatóak az `empty` és `default` szavak, mivel ezek speciális kulcsszavak.

Ha ezek az érvek nem győznek meg: ez a szintaxis egyszerűen nem javasolt, és bármikor megszűnhet működni.

> **Megjegyzés:** Ha az error_reporting beállítást `E_ALL` értékre állítod be, látni fogod, hogy a PHP figyelmeztetéseket generál, ha ezt a kunstrukciót használod. Ez ugyanígy érvényes minden más nem javasolt 'szolgáltatásra'. (Add az `error_reporting(E_ALL);` sort a programod elejéhez).

> **Megjegyzés:** Idézőjeles stringekben, egy másik szintaxis érvényes. Lásd a változók behelyettesítése című részt további részletekért.

Példák

A tömb típus a PHP-ben nagyon sokoldalú, ezért összegyűjtöttünk néhány példát, hogy megmutassuk a tömbök erejét.

```
// ez
$a = array( 'szín' => 'piros',
            'íz'   => 'édes',
            'alak' => 'kerek',
            'név'  => 'alma',
                      4        // 0 lesz a kulcsa
          );

// teljesen megyezik ezzel
$a['szín'] = 'piros';
$a['íz']   = 'édes';
$a['alak'] = 'kerek';
$a['név']  = 'alma';
$a[]       = 4;        // 0 lesz a kulcsa

$b[] = 'a';
$b[] = 'b';
$b[] = 'c';
// a következő tömböt adja: array( 0 => 'a' , 1 => 'b' , 2 => 'c' )
// vagy egyszerűen: array('a', 'b', 'c')
```

Példa 6-4. Az array() használata

```
// Tömb, mint tulajdonság hozzárendelés
$map = array( 'verzió'     => 4
            , 'rendszer'   => 'Linux'
            , 'nyelv'      => 'angol'
            , 'short_tags' => true
            );

// szigorúan számokat tartalmazó tömb
$tomb = array( 7
             , 8
             , 0
             , 156
             , -10
             );
// ez ugyanaz, mint array ( 0 => 7, 1 => 8, ...)

$valtogatas = array(         10 // 0. indexű/kulcsú
                  , 5    =>  6
                  , 3    =>  7
                  , 'a'  =>  4
                  ,         11 // kulcsa 6 (a legnagyobb egész kulcs 5 volt)
                  , '8'  =>  2 // kulcsa 8 (egész!)
                  , '02' => 77 // kulcsa '02'
                  , 0    => 12 // a 10 értéket felülírjuk 12-vel
                  );

// üres tömb
$ures = array();
```

Példa 6-5. Kollekció

```
$szinek = array('piros','kék','zöld','sárga');

foreach ( $szinek as $szin ) {
    echo "Szereted a(z) $szin színt?\n";
}

/* kimenete:
Szereted a(z) piros színt?
Szereted a(z) kék színt?
Szereted a(z) zöld színt?
Szereted a(z) sárga színt?
*/
```

Figyelj arra, hogy jelenleg közvetlenül nem lehet módosítani a tömb elemeinek értékét ezzel a ciklussal. A problémát a következőképpen tudod megkerülni:

Példa 6-6. Kollekció

```
foreach ( $szinek as $kulcs => $szin )
{
    // nem működik (nem módosítja a tömböt):
    //$szin = strtoupper($szin);

    // működik (módosítja a tömböt):
    $szinek[$kulcs] = strtoupper($szin);
}
print_r($szinek);

/* kimenete:
Array
(
    [0] => PIROS
    [1] => KÉK
    [2] => ZÖLD
    [3] => SÁRGA
)
*/
```

Ebben a példában egy egytől számozott tömböt készítünk.

Példa 6-7. Egytől kezdődő index

```
$elsonegyed  = array(1 => 'Január', 'Február', 'Március');
print_r($elsonegyed);

/* kimenete:
Array
(
    [1] => 'Január'
    [2] => 'Február'
    [3] => 'Március'
)
*/
```

Példa 6-8. Felöltés valós adatokkal

```
// egy tömb felöltése a <link linkend="ref.dir">könyvtárban</link> található filenevekke
$konyvtar = opendir('.');
while ($filenev = readdir($konyvtar)) {
    $filenevek[] = $filenev;
}
closedir($konyvtar);
```

A tömbök rendezettek. A sorrendet számos függvénnyel megváltoztathatod. Lásd a tömb függvények című részt további információkért.

Példa 6-9. Tömbök rendezése

```
sort($filenevek);
print_r($filenevek);
```

Mivel a tömb egy értéke bármi lehet, értékként akár egy másik tömböt is megadhatsz. Ilyen formában készíthetsz rekurzív vagy többdimenziós tömböket.

Példa 6-10. Rekurzív és többdimenziós tömbök

```
$gyumolcsok = array ( "gyümölcsök"  => array ( "a" => "narancs"
                                     , "b" => "banán"
                                     , "c" => "alma"
                                     )
                    , "számok" => array (  1
                                     , 2
                                     , 3
                                     , 4
                                     , 5
                                     , 6
                                     )
                    , "lyukak"   => array (     "első"
                                     , 5 => "második"
                                     ,      "harmadik"
                                     )
                    );
```

Objektumok

Objektumok létrehozása

Egy objektum létrehozására a `new` operátor való, amely az adott objektumtípus egy példányát hivatott létrehozni [mivel lehet, hogy hivatkozni akarunk rá, rendszerint másolni szoktuk az objektumot (ill. a címét) egy változóba]

```
<?php
class semmi #egy objektumosztály létrehozása, semmi az osztály neve
{
```

```
    function do_semmi ()
    {
        echo "Csinálom a semmit.";
    }
}

$bar = new semmi;  #$bar most egy semmi típusú objektum
$bar->do_semmi();  #a $bar objektum do_semmi()
                   #metódusát (függvényét) hívja
?>
```

Alaposabb információkért nézd meg az Osztályok és objektumok című részt.

Erőforrások

Az erőforrás egy olyan speciális változó, ami egy külső erőforrásra tartalmaz referenciát. Az erőforrásokat speciális függvények hozzák létre és használják. Lásd az ide tartozó függeléket a különböző erőforrás típusokhoz tartozó függvények listájával.

Megjegyzés: Az erőforrás típus a PHP 4-ben került a nyelvbe.

Erőforrások felszabadítása

A PHP 4-es Zend Egine-jében bevezetett hivatkozás-számlálási technika következtében a használatlan erőforrásokat automatikusan detektálja a PHP (mint a Java). Ebben az esetben minden erőforrás által használt összes más erőforrás felszabadításra kerül a szemétgyűjtő algoritmusban. Emiatt ritkán szükséges, hogy felszabadítsuk a foglalt memóriát valamilyen free_result jellegű függvénnyel.

Megjegyzés: Az állandó adatbázis kapcsolatok speciálisak abban az értelemben, hogy *nem* szűnnek meg a szemétgyűjtés során. Lásd még az állandó kapcsolatok című részt.

NULL

A speciális `NULL` érték jelzi, hogy egy változó nem tartalmaz értéket. A `NULL` a NULL egyetlen lehetséges értéke.

Megjegyzés: A NULL típus a PHP 4-ben került bevezetésre.

Szintaxis

Csak egy értéket vehet fel a `NULL` típus, ami a kisbetűs és nagybetűs írásmódra nem érzékeny `NULL` kulcsszó.

```
$valtozo = NULL;
```

Bűvészkedés a típusokkal

A PHP nem követeli meg (nem támogatja) az explicit típusdefiníciót a változók deklalárásakor; egy változó típusát a környezet határozza meg, amiben a változót használjuk. Vagyis ha egy stringet rendelünk *`var`*-hoz, akkor *`var`* string lesz. Ha ezután egy egészet rendelünk hozzá, *`var`* egész lesz.

Egy példa a PHP automatikus típuskonverziójára az összeadás '+' operátora. Ha bármely operandusa lebegőpontos, akkor mindegyik operandusból lebegőpontos lesz, és az eredmény is az lesz. Máskülönben, ha az operandusok egészek, az eredmény is egész lesz. Figyeljünk azonban arra, hogy az operandusok típusát NEM változtatja meg; csupán a kiértékelés módját határozza meg.

```
$foo = "0";                  // $foo egy string (ASCII 48)
$foo += 2;                   // $foo most egész (2)
$foo = $foo + 1.3;           // $foo most lebegőpontos (3.3)
$foo = 5 + "10 Kis Pingvin"; // $foo egész (15)
$foo = 5 + "10 Kismalac";    // $foo egész (15)
```

Ha az utolsó két példa furcsának tűnik, nézd meg a String konverziókat.

Ha egy változót adott típussal szeretnél használni egy kifejezésben, nézd meg a Típus konverziókat. Ha a változó típusát akarod megváltoztatni, használd a settype() függvényt.

Ha az ebben a fejezetben található példákat ki szeretnéd próbálni, használd a var_dump() függvényt a típusok és értékek kiírására.

> **Megjegyzés:** Egy tömb típusának automatikus konverziója jelenleg nem definiált.

```
$a = 1;      // $a egész
$a[0] = "f"; // $a tömb lesz, $a[0]-ba "f" kerül
```

A fenti példából ugyan tisztán látszik, hogy $a-ból tömb lesz, aminek az első eleme 'f', de mi a helyzet ezzel:

```
$a = "1";    // $a egy string
$a[0] = "f"; // Mi van a string offsettel? Mi történik?
```

Mivel a PHP támogatja, hogy a stringet, mint karakterek tömbjét kezeljük, és "beleindexeljünk" a fenti példa problémához vezet: $a-ból most tömb legyen, aminek az eleme "f", vagy a string első karaktere legyen "f"?

Emiatt, a PHP 3.0.12 és PHP 4.0b3-RC4 verzióiban az automatikus típuskonverzió eredménye nem definiált. A megoldáson már dolgoznak.

Típuskonverziók

PHP-ben a típuskonverzió úgy működik, mint C-ben: a kívánt típus nevét zárójelbe írjuk az elé a változó elé, amelyet cast-olni (konvertálni) akarunk.

```
$foo = 10;            // $foo egész
$bar = (float) $foo;  // $bar lebegőpontos
```

A megengedett típusok:

- (int), (integer) - egésszé konvertál
- (bool), (boolean) - logikai értékké konvertál
- (real), (double), (float) - lebegőpontos számmá konvertál
- (string) - stringgé konvertál
- (array) - tömbbé konvertál
- (object) - objektummá konvertál

Megjegyzés: Ahelyett, hogy egy változót stringgé alakítasz, egyszerűen idézőjelbe is teheted, hogy ezt a hatást elérd.

Megjegyzés: szóközök és tabulátorok megengedettek a string belsejében, tehát az alábbiak ugyanazt csinálják:

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

Nem mindig tiszta, hogy mi történik, ha típusok közt cast-olunk [implicit cast]. További információkért lásd az alábbi fejezeteket:

- Logikai értékké alakítás
- Egész értékké alakítás

Amikor egy tömböt stringgé alakítasz, az eredmény az `Array` szó lesz. Hasonlóan, ha objektumot stringgé alakítasz, az `Object` szót kapod.

Ha skalárból, vagy stringből tömbbé szeretnénk konvertálni, a változó az első eleme lesz a tömbnek:

```
$var = 'szia';
$arr = (array) $var;
echo $arr[0]; // kiírja 'szia'
```

Ha egy skalárt vagy stringet objektummá akarjuk konvertálni, a változó az objektum egy attribútuma lesz; az attribútum neve 'scalar' lesz:

```
$var = 'szia';
$obj = (object) $var;
echo $obj->scalar; // kiírja, hogy 'szia'
```

Fejezet 7. Változók

Alapok

PHP-ban a változókat egy dollárjel utáni változónév jelöli. A változónevek érzékenyek kis- és nagybetűk különbözőségére.

A változónevekre a PHP más jelzőivel azonos szabályok vonatkoznak. Egy érvényes változónév betűvel vagy aláhúzással kezdődik, amit tetszőleges számú betű, szám vagy aláhúzás követ. Reguláris kifejezéssel kifejezve ez a következőt jelenti: ’[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*’

> **Megjegyzés:** Ebben az esetben egy betű lehet az angol abc egy betűje a-z-ig és A-Z-ig, valamint a 127-től 255-ig terjedő (0x7f-0xff) ASCII kódú karakterek.

```
$var = "Géza";
$Var = "János";
echo "$var, $Var";           // kiírja, hogy "Géza, János"

$4site        = ’ez nem jó’; // nem érvényes, mert számmal kezdődik
$_4site       = ’ez ok’;     // érvényes, aláhúzással kezdődik
$täyte        = ’mansikka’;  // érvényes, az ’ä’ az ASCII 228-as karaktere
$tükörfúrógép = "árvíztűrő"; // érvényes, ellenőrizheted egy ASCII táblában
```

PHP 3-ban a változókhoz mindig értékek tartoznak. Vagyis ha egy kifejezést rendelünk egy változóhoz, az eredeti kifejezés egészének értéke másolódik a célváltozóba. Ez azt jelenti, hogy ha például egy változó értékét egy másikhoz rendeljük, egyikük megváltozása sincs hatással a másikra. Nézd át Kifejezések c. fejezetet, ahol az ilyen jellegű hozzárendelésekről több információ található.

PHP 4-ben lehetőség van egy másik hozzárendelési módra: *változó referencia szerinti hozzárendelésére*. Ez azt jelenti, hogy az új változó egyszerűen hivatkozik (más szóval "alias lesz", vagy "rá mutat") az eredetire. Az új változón végzett változtatások az eredetit is érintik és fordítva. Ez azt is jelenti, hogy nem történik másolás; ekképpen a hozzárendelés gyorsabban történik meg. Igaz ugyan, hogy ez a sebességnövekedés csak "feszes" ciklusokban vagy nagy tömböknél ill. objektumok átadásakor jelentkezik.

Referencia szerinti értékadáshoz egyszerűen & jelet kell az átadandó változó neve elé írni. Az alábbi kód - például - kiírja kétszer, hogy ’Nevem Bob’:

```
<?php
$ize = ’Bob’;              // ’Bob’ hozzárendelése $ize-hoz
$bigyo = &$ize;            // Hivatkozás $ize-ra $bigyo-ban.
$bigyo = "Nevem $bigyo";   // $bigyo megváltoztatása...
echo $ize;                 // $ize is megváltozott
echo $bigyo;
?>
```

Fontos megjegyezni, hogy csak megnevezett változókra lehet referenciát létrehozni.

```
<?php
$ize = 25;
$bigyo = &$ize;      // Ez egy érvényes hozzárendelés.
$bigyo = &(24 * 7);  // Érvénytelen referencia egy névtelen kifejezésre.

function test()
{
   return 25;
}

$bigyo = &test();    // Érvénytelen.
?>
```

Előre definiált változók

A PHP egy számos előre definiált változót biztosít az bármely futó szkript számára. Sokat ezek közül nem lehet teljes pontossággal dokumentálni, mert függnek a a futtató szervertől, a használt verziótól, a konfigurálástól, és más gyéb tényezőktől. Néhány ilyen változó nem elérhető, ha a PHP parancssorból fut.

E tényezők ellenére az alábbi listában azok az előre definiált változók vannak felsorolva, amelyek a PHP 3 tipikus installálásakor elérhetők az Apache (http://www.apache.org/) 1.3.6-ot használva.

Az összes előre definiált változó kilistázható, és sok más hasznos információt is közöl a phpinfo().

> **Megjegyzés:** Ez a lista nem teljes, nem is ez a célja. Ez csak egy útmutató arról, hogy milyen típusú előre definiált változók lehetnek.

Apache változók

Ezek a változók az Apache (http://www.apache.org/) webszerver változói. Ha másik szerveren fut a PHP, nincs rá garancia, hogy ugyanezek a változók akkor is rendelkezésre állnak: némelyik hiányozhat, esetleg itt nem szereplőket is biztosíthat a szerver. Számos a CGI 1.1 specifikációban (http://hoohoo.ncsa.uiuc.edu/cgi/env.html) dokumentált változókra biztosan lehet alapozni.

Figyelni kell arra, hogy ha a PHP parancssorból (command line) használva a változók töredékének van csak értelme, ha egyáltalán definiáltak ilyen esetben.

$GATEWAY_INTERFACE

A server CGI specifikációjának verziója. Pl.: ’CGI/1.1’.

$SERVER_NAME

A gazdagép (host) neve, ami alatt a szkript fut. Ha a szkript virtuális host-on fut, a virtuális host neve.

$SERVER_SOFTWARE

Serverazonosító sztring, amely a kérésekre (request) adott válasz (response) fejlécében (header) szerepel.

$SERVER_PROTOCOL

Az oldal lekéréséhez használt protokoll neve és verziója pl. 'HTTP/1.0'

$REQUEST_METHOD

Az oldal eléréséhez használt kérési forma, pl.: 'GET', 'HEAD', 'POST', 'PUT'.

$QUERY_STRING

Keresési kérdés, az oldalt meghívó URL-ben a ? után szereplő rész, ha létezik.

$DOCUMENT_ROOT

A www dokumentumok főkönyvtára, amely alatt a jelenlegi szkript fut. Ez a szerver konfigurációs fájljában meghatározott érték.

$HTTP_ACCEPT

Az aktuális kérés `Accept:` HTTP fejlécének tartalma, ha van ilyen. Ebben adja meg a böngésző, hogy milyen formátumokat képes fogadni.

$HTTP_ACCEPT_CHARSET

Az aktuális kérés `Accept-Charset:` HTTP fejlécének tartalma, ha van ilyen. pl. 'iso-8859-1,*,utf-8'.

$HTTP_ENCODING

Az aktuális kérés `Accept-Encoding:` HTTP fejlécének tartalma, ha van ilyen. pl. 'gzip'.

$HTTP_ACCEPT_LANGUAGE

Az aktuális kérés `Accept-Language:` HTTP fejlécének tartalma, ha van ilyen. pl. 'en'.

$HTTP_CONNECTION

Az aktuális kérés `Connection:` HTTP fejlécének tartalma, ha van ilyen. pl. 'Keep-Alive'.

$HTTP_HOST

Az aktuális kérés `Host:` HTTP fejlécének tartalma, ha éppen van ilyen.

$HTTP_REFERER

Annak az oldalnak az URL címe, ahonnan ide került a látogató. Ezt a látogató böngészőprogramja állítja be, de ezt nem minden böngésző és nem mindig teszi meg.

$HTTP_USER_AGENT

Az aktuális `User_Agent:` HTTP fejlécének tartalma, ha van ilyen. Ez a string, amely meghatározza az oldal megtekintéséhez használt böngészőt, pl. `Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)`. Sok más mellett ezt az értéket a fel lehet használni arra, hogy a get_browser() fügvénnyel az aktuális böngésző képességeinek megfelelően lehessen kialakítani az oldal megjelenését.

$REMOTE_ADDR

Az az IP cím, ahonnan a látogató az oldalt lekérte.

$REMOTE_PORT

A látogató gépén a webszerverrel való kommunikációhoz használt port száma.

$SCRIPT_FILENAME

A PHP szkript abszolút elérési útja a szerver gépen.

$SERVER_ADMIN

A SERVER_ADMIN a kiszolgáló konfigurációs fájl direktíva értéke (Apache alatt). Ha a szkript egy virtuális host-on fut, akkor ez a virtuális host-hoz rendelt érték.

$SERVER_PORT

A kiszolgálógépen a kiszolgáló kommunikációjában használt port száma. Alapértelmezés szerint ez a ’80’, de például SSL-t használva ez a biztonságos HTTP portnak a száma lesz.

$SERVER_SIGNATURE

A kiszolgáló verziószámát és a virtuális host-ot tartalmazó sztring, ha engedélyezett. Ez a szöveg jelenik meg minden szerver által generált oldal alján.

$PATH_INFO

Hasonló értéket tartalmaz, mint a SCRIPT_NAME, de CGI verzió esetén ez jobb lehet az önhivatkozó kérdőívek készítésekor, mert egyes szervereken a SCRIPT_NAME tartalmazza a PHP bináris meghívási útvonalát is.

$PATH_TRANSLATED

A szkript a fájlrendszer szerinti (nem a document root-hoz viszonyított!) elérési útja. Ezt a kiszolgáló a virtuális->valós átalakítás után határozza meg.

$SCRIPT_NAME

Az aktuális szkript elérési útja, ahogy az az URL-ben megjelenik. Hasznos a saját magukra mutató oldalakhoz.

$REQUEST_URI

Az az URI, amellyel az oldalt lekérték. pl. ’/index.html’.

Környezeti változók

A PHP globális változói közé kerülnek a PHP feldolgozót futtató környezet változói. Számos változót a shell szolgáltat, ami alatt a PHP fut. Mivel más-más rendszerek más-más shell-eket használnak, ezért egy pontos lista összeállítása lehetetlen. A használt shell dokumentációjában nézz utána az előre definiált környezeti változóknak!

Más környezeti változók - beleértve a CGI változókat is - attól függően kerülnek be, hogy a PHP szerver modulként fut vagy CGI feldolgozóként.

PHP változók

Ezeket a PHP maga állítja elő. A `$HTTP_*_VARS` változók csak akkor állnak rendelkezésre, ha a track_vars be van kapcsolva. Ha engedélyezted ezt a beállítást, ezek a változók mindig létrejönnek, még ha üresek is. Ez nem engedi a rosszindulatú felhasználóknak, hogy meghamasítsák a változókat.

> **Megjegyzés:** A PHP 4.0.3 verziótól kezdve a track_vars mindig be van kapcsolva, függetlenül a konfigurációs fájl beállításától.

> **Megjegyzés:** Az új szuper globális (Superglobals) változók 4.1.0 verziótól használhatók. További részletekért lásd a 4.1.0 verzió bejelentését (http://www.php.net/release_4_1_0.php) Ezek a `$_GET`, `$_POST`, `$_ENV`, `$_SERVER`, `$_COOKIE`, `$_REQUEST` `$_FILES` és a `$_SESSION` tömbök, amelyek fesztelen megfogalmazásban *szuper globális* jelzővel illetik, mivel minden esetben használhatók tekintet nélkül az aktuális érvényességi körre. Emiatt a régi, nekik megfelelő `$HTTP_*_VARS` tömbök használata már nem javasolt.

Ha a register_globals be van kapcsolva, ezek a változók globális változókként is létrejönnek, függetlenül a `$HTTP_*_VARS` tömböktől. Nézd meg idevonatkozó részletes információkért a biztonságról szóló Globálisan is elérhető változók (Register Globals) használata c. fejezetet!

$argv

A szkriptnek átadott argumentumok tömbje. Ha a szkript parancssor alól futtatjuk, ez egy C-szerű hozzáférést biztosít a parancssor argumentumokhoz. Ha GET metódussal hívjuk, az URL kérdőjel utáni részét (query string) tartalmazza.

$argc

A parancssori argumentumok számát tartalmazza (ha parancssorból futtatjuk).

$PHP_SELF

A most futó szkriptfájl neve, a document root-hoz képest relatív alakban. Ha a PHP parancssorból fut, ez a változó nem elérhető. Ez a változó a kérés útvonal információját is tartalmazza, ha az létezik, azaz a $PHP_SELF a "http://example.com/test.php/foo.bar" címre a következőt fogja adni: "/test.php/foo.bar"

$HTTP_COOKIE_VARS

HTTP cookie-kat (sütiket) tartalmazó asszociatív tömb.

$_COOKIE

HTTP cookie-kat (sütiket) tartalmazó asszociatív tömb, amely minden hatókörben elérhető. PHP 4.1.0-ban került a nyelvbe.

$HTTP_GET_VARS

HTTP GET metódussal által szolgáltatott adatokat tartalmazó asszociatív tömb.

$_GET

HTTP GET metódussal által szolgáltatott adatokat tartalmazó asszociatív tömb, amely minden hatókörben elérhető. PHP 4.1.0-ban került a nyelvbe.

$HTTP_POST_VARS

HTTP POST metódussal által szolgáltatott adatokat tartalmazó asszociatív tömb.

$_POST

HTTP POST metódussal által szolgáltatott adatokat tartalmazó asszociatív tömb, amely minden hatókörben elérhető. PHP 4.1.0-ban került a nyelvbe.

$HTTP_POST_FILES

A HTTP POST metódussal feltöltött fájlokról tartalmaz információkat, asszociatív tömb. Lásd a POST metódusú feltöltések című részt további információkért a `$HTTP_POST_FILES` tartalmát és szerkezetét illetően. PHP 4.0.0-ban került a nyelvbe.

$_FILES

A HTTP POST metódussal feltöltött fájlokról tartalmaz információkat, asszociatív tömb. Lásd a POST metódusú feltöltések című részt további információkért a `$_POST` tartalmát és szerkezetét illetően. Minden hatókörben elérhető. PHP 4.1.0-ban került a nyelvbe.

$HTTP_ENV_VARS

A szülő/futtató környezet aktuális változóit tartalmazó asszociatív tömb.

$_ENV

A szülő/futtató környezet aktuális változóit tartalmazó asszociatív tömb, amely minden hatókörben elérhető. PHP 4.1.0-ban került a nyelvbe.

$HTTP_SERVER_VARS

A HTTP szervertől érkező változókkal feltöltött asszociatív tömb. Ezek a változók megfelelnek a fent említett Apache változóknak.

$_SERVER

A HTTP szervertől érkező változókkal feltöltött asszociatív tömb, amely minden hatókörben elérhető. PHP 4.1.0-ban került a nyelvbe.

$HTTP_SESSION_VARS

Az aktuális szkripthez tartozó session változókat tartalmazó asszociatív tömb.

$_SESSION

Az aktuális szkripthez tartozó session változókat tartalmazó asszociatív tömb, amely minden hatókörben elérhető. A `_SESSION` tömbben új bejegyzések létrehozásával automatikusan regisztrálhatók értékek, mint a session_register() segítségével. PHP 4.1.0-ban került a nyelvbe.

$_REQUEST

A GET, POST és Cookie változókból összeállított asszociatív tömb. Más szóval, bármilyen a felhasználótól ügyféltől érkező adatot tartalmaz, ami biztonsági szempontból nem megbízható. Minden hatókörben elérhető. PHP 4.1.0-ban került a nyelvbe.

Változók hatásköre

[Ezt a fejezetet érdemes elolvasni, még akkor is, ha profi vagy valamilyen programozási nyelvben, mert a PHP tartogat egy-két érdekes meglepetést...]

A változó hatásköre az a környezet, amelyben a változó definiált. A legtöbb esetben minden PHP változónak egyetlen hatásköre van. Ez az egyetlen hatáskör kiterjed az include és a require segítségével használt fájlokra is. Például:

```
$a = 1;
include "b.inc";
```

Itt az `$a` változó elérhető lesz az beillesztett `b.inc` szkriptben is. A felhasználói függvényekkel a lokális függvényhatáskör kerül bevezetésre. Alapértelmezés szerint minden, függvényen belül használt változó ebbe a lokális függvényhatáskörbe tartozik, például:

```
$a = 1; /* globális hatáskör */

function Test ()
{
    echo $a; /* egy helyi változót vár */
}

Test();
```

Ez a szkript nem fog semmilyen kimenetet sem eredményezni, mivel az echo kifejezés az `$a` változónak egy helyi - függvényen belüli - változatára utal, és ebben a hatáskörben ehhez nem rendeltek értéket. Ez valamelyest különbözik a C nyelv filozófiájától, ahol a globális változók automatikusan elérhetők bármely függvényből, feltéve ha a függvényben újra nem definiáltad azt a változót. Ez problémák forrása lehet, ha az ember véletlenül megváltoztat egy globális változót. A PHP-ben a globális változókat global kulcsszóval kell deklarálni a függvényekben, például:

```
$a = 1;
$b = 2;

function Osszead()
{
    global $a, $b;

    $b = $a + $b;
}

Ossszead();
echo $b;
```

A fenti szkript kiírja, hogy "3". `$a` és `$b` global-ként való deklarálásával minden utalás ezekre a változókra a globális változót fogja érinteni. Nincs megkötve, hány globális változót kezelhet egy függvény.

Globális változók elérésének másik módja a PHP által definiált speciális `$GLOBALS` tömb használata. Az előbbi példával egyenértékű megoldás:

```
$a = 1;
$b = 2;

function Osszead()
{
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Osszead();
echo $b;
```

A `$GLOBALS` asszociatív tömb, ahol a globális változó neve jelenti a kulcsot, és a változó értéke a tömbelem értéke.

A változók hatáskörének másik fontos lehetősége a *static* (statikus) változó. A statikus változó csak lokális hatáskörben él - egy függvényen belül, de két függvényhívás közt nem veszti el az értékét, a változó hatásköréből való kilépés esetén is megmarad az értéke:

```
function Test()
{
    $a = 0;
    echo $a;
    $a++;
}
```

Ez nagyon haszontalan függvény, mivel nem csinál mást, mint minden meghívásakor `$a`-t `0`-ra állítja, aztán kiírja a 0-t. Az `$a`++ teljesen felesleges, mert amint vége a függvény futásának az `$a` változó megszűnik. Ahhoz, hogy ebből értelmes számlálófüggvény legyen - megmaradjon a számláló értéke -, az `$a` változót statikusnak kell deklarálni:

```
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
```

Most már valahányszor meghívódik a Test() függvény, kiírja `$a` értékét, majd azt megnöveli eggyel.

A statikus változókat a rekurzív függvények megvalósításában is felhasználhatjuk. Rekurzívnak nevezzük azt a függvényt, amely saját magát hívja meg. Ezt természetesen feltételhez kell kötni,

nehogy végtelen rekurzióba kerüljön a vezérlés és meghatározatlan időre a függvényen belül csapdába esik. Mindig meg kell bizonyosodni arról, hogy megfelelő feltétel rendelkezésre áll a rekurzió befejezéséhez. A következő függvény rekurzívan elszámol 10-ig a statikus `$count` változó segítségével: [A static kulcsszó nagyon fontos!]

```
function Test()
{
    static $count = 0;

    $count++;
    echo $count;
    if ($count < 10) {
        Test ();
    }
    $count--;
}
```

Változó változók

Néha kényelmes változó változók használata. Ez olyan változó jelent, amelynek a nevét dinamikusan lehet beállítani. A normál változót így adunk értéket:

```
$a = "hello";
```

A változó változó veszi egy változó értékét, amelyet egy másik változó értékének tekinti. A fenti példában a *hello*, egy változó neveként használható, a $a elé még egy $-t írva.

```
$$a = "világ";
```

Ekkor már két változó van a PHP szimbólumtáblájában: `$a`, amelynek tartalma "hello", és `$hello`, amelynek a tartalma "világ". Ennélfogva a következő programsor:

```
echo "$a ${$a}";
```

pontosan ugyanazt csinálja, mintha ezt írtuk volna:

```
echo "$a $hello";
```

Mindkettő kiírja, hogy: `hello világ`.

Annak érdekében, hogy változó változókat tömbökkel együtt is használhassuk, fel kell oldani a következő kétértelműséget. A `$$a[1]` kifejezés kiértékelésekor a feldolgozónak tudnia kell, hogy ez a `$a[1]` értékét tekintse a hivatkozott változó neveként, vagy `$$a`-t - és ekkor és ennek a tömbnek 1. indexű elemére történt a hivatkozás. Az első esetben `${$a[1]}`, míg a másodikban `${$a}[1]` írandó.

Változók a PHP-n kívülről

HTML űrlapok (GET és POST)

Egy HTML űrlap (form) elküldésével az űrlap összes beviteli mezőjének értéke automatikusan elérhetővé válik a szkript számára a PHP segítségével. Ha a track_vars szolgáltatás be van kapcsolva, ezek a változók a `$HTTP_POST_VARS`, `$HTTP_GET_VARS`, és/vagy `$HTTP_POST_FILES`, asszociatív tömbökben foglalnak helyet a változók forrásának megfelelően.

További információkért ezekről a változókról olvasd el az Előre definiált változók című részt.

Példa 7-1. Egyszerű űrlap változó példája

```
<form action="foo.php" method="post">
    Name: <input type="text" name="usernev"><br>
    <input type="submit">
</form>
```

Ha elküldjük a fenti űrlapot, akkor a PHP létrehozza a `$HTTP_POST_VARS['usernev']` változót. Ha a register_globals beállítás aktív, akkor ez a változó elérhető lesz `$usernev` globális változóként is.

> **Megjegyzés:** A magic_quotes_gpc konfigurációs beállítás hatással van a GET, POST és COOKIE metódusokkal kapott értékekre. Bekapcsolva a (Ez a "PHP", biz'a!) szövegből automágikusan (Ez a \"PHP\", biz\'a!) lesz, hogy megkönnyítse az adatbázisba írást. Lásd még: addslashes(), stripslashes() és magic_quotes_sybase!

A PHP megérti és kezeli a tömbökbe rendezett űrlapváltozókat. (Lásd related faq!) Hasznos lehet csoportosítani az összetartozó változókat, vagy az olyan űrlapelemeket, ahol több lehetőség közül nem csak egy választható.

Példa 7-2. Összetettebb űrlapváltozók

```
<form action="array.php" method="post">
    Név: <input type="text" name="szemelyes[nev]"><br>
    Email: <input type="text" name="szemelyes[email]"><br>
    Sör: <br>
```

```
    <select multiple name="ital[]">
        <option value="warthog">Warthog
        <option value="guinness">Guinness
        <option value="stuttgarter">Stuttgarter Schwabenbräu
        </select>
    <input type="submit">
</form>
```

PHP 3-ban ilyen módon csak egydimenziós tömbök hozhatók létre, míg PHP 4-ben nincs ilyen korlátozás.

IMAGE SUBMIT változónevek

Az űrlap elküldésekor megoldható, hogy gomb helyett képet használunk ilyesféle jelölés segítségével:

```
<input type="image" src="image.gif" name="elkuld">
```

Ha a felhasználó a képre kattint, a kiszolgálóra a következő két változó jut el: elkuld_x-et és elkuld_y-t. Ezek tartalmazzák a kattintás képen belüli koordinátáit. A tapasztaltabbak biztos megjegyeznék, hogy a változónevek nem aláhúzást, hanem pontot tartalmaznak, de a PHP a pontot automatikusan aláhúzássá konvertálja.

HTTP sütik (cookie)

A PHP támogatja a Netscape specifikációja (http://www.netscape.com/newsref/std/cookie_spec.html) által definiált HTTP sütiket. A süti olyan mechanizmus, amely a távoli böngészőn tesz lehetővé adattárolást, és így lehetővé teszi a visszatérő felhasználók azonosítását. A sütiket a setcookie() függvénnyel lehet beállítani. A sütik a HTTP fejléc részei, így a SetCookie függvényt bármilyen kimenet generálása előtt kell meghívni. Ugyanezt a megkötést kell betartani a header() függvénnyel. Bármilyen süti, amelyet a klienstől érkezik, automatikusan egy PHP változóba kerül, úgy, mint a GET és a POST metódus adatai.

Ha több adatot akarsz rendelni egy sütihez, egyszerűen rakj *[]*-t a cookie neve után. Például:

```
setcookie("EnSutim[]", "tesztelem", time()+3600);
```

Vigyázz, mert a süti felülírja az előző azonos nevű sütit, hacsak nem különbözik a path vagy a domain. Így pl. egy bevásárlókocsi megírásakor jó egy számlálót is elhelyezni, hogy elkerüljük a problémát.

Példa 7-3. SetCookie Példa

```
$Szamlalo++;
setcookie("Szamlalo", $Szamlalo, time()+3600);
setcookie("Kocsi[$Szamlalo]", $elem, time()+3600);
```

Környezeti változók

A PHP automatikusan létrehozza az elérhető környezeti változókat, mint egyszerű PHP változókat.

```
echo $HOME;  /* Megmutatja a HOME környezeti változót, ha van. */
```

Mivel az információ ami a GET, POST és Cookie mechanizmuson keresztül jön, mind automatikusan PHP változókat generál, ezért néha jobb, ha direkt a környezeti változókból olvassuk ki, hogy tudjuk, valójában melyik változatból olvasunk. A getenv() függvényt lehet erre használni. A környezeti változókat be is lehet állítani a putenv() függvénnyel.

Pontok a bejövő változónevekben

Általában a PHP nem változtatja meg a változók neveit, amikor a szkript megkapja őket. A pont viszont nem érvényes karakter egy PHP változóneven belül. Az ok megértéséért nézd a:

```
$varname.ext;  /* érvénytelen változónév */
```

Most az elemző lát egy `$varname` nevű változót, amelyet stringösszefűző operátor (.) követ, majd egy csupasz sztring: az ext - idézőjel nélküli string, amely nem egyezik semmilyen ismert vagy lefoglalt kulcsszóval. Ez nyilván nem a kívánt eredményt adná.

Emiatt fontos, hogy PHP automatikusan helyettesíti a pontokat aláhúzásjellel.

Változótípusok meghatározása

Mivel a PHP határozza meg a változók típusát és konvertálja őket (általában) szükség szerint, nem mindig nyilvánvaló, hogy milyen típusú egy pillanatban egy adott változó. A PHP-nek számos függvénye van, amelyek egy változó típusát hivatottak eldönteni. Ezek: gettype(), is_long(), is_double(), is_string(), is_array(), és is_object().

Fejezet 8. Konstansok

A konstans egy egyszerű érték azonosítója (neve). Mint ahogy az elnevezése is mutatja, a program futása során nem változik meg az értéke (a mágikus `__FILE__` és `__LINE__` konstansok az egyedüli kivételek ez alól). A konstansok alapesetben érzékenyek a kis- és nagybetűs irásmódra. Megállapodás szerint általában csupa nagybetűs neveket adunk a konstansoknak.

A konstansok neveire a PHP más jelzőivel azonos szabályok vonatkoznak. Egy érvényes konstans név betűvel vagy aláhúzással kezdődik, amit tetszőleges számú betű, szám vagy aláhúzás követ. Reguláris kifejezéssel kifejezve ez a következőt jelenti:
`[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*`

> **Megjegyzés:** Ebben az esetben egy betű lehet az angol abc egy betűje a-z-ig és A-Z-ig, valamint a 127-től 255-ig terjedő (0x7f-0xff) ASCII kódú karakterek.

A konstansok bárhonnan elérhetőek.

Szintakszis

Konstanst a define() függvénnyel lehet létrehozni. Definiálása után később nem lehet törölni vagy megváltoztatni az értékét.

Csak skaláris adat (boolean, integer, double vagy string típusú) lehet egy konstans tartalma.

A konstans értékére a nevének megadásával lehet hivatkozni. A változókkal ellentétben *nem* szabad $ jelet tenned a konstand neve elé. Használható még a constant() függvényt is, ha például a konstans nevét egy változó adja. A get_defined_constants() függvénnyel lehet a definiált konstansok listáját megkapni.

> **Megjegyzés:** A konstansok és a (globális) változók különböző névtérben vannak. Ez azt jelenti, hogy a `TRUE` és a `$TRUE` két különböző dolgot jelent.

Ha egy definiálatlan konstanst próbálsz meg használni, a PHP a konstans nevét veszi karaktersorozatként értékül. Ilyen esetekben egy notice szintű hiba keletkezik. A defined() függvény segítségével vizsgálható a konstans létezése.

A következők a fontosabb különbségek a változókhoz képest:

- A konstansok nevét nem kell dollár jellel $ kezdeni.
- A konstansokat akárhol definiálhatók, és akárhonnan elérhetők, a változók környezeti korlátozásaitól függetlenül.
- A konstansok nem módosíthatóak, és nem törölhetőek, miután egyszer létrehozták azokat.
- A konstansok csak skaláris értékeket tartalmazhatnak.

Példa 8-1. Konstansok definiálása

```
<?php
define("KONSTANS", "Helló világ!");
echo KONSTANS; // kiírja, hogy "Helló világ!"
```

```
echo Konstans; // kiírja, hogy "Konstans" és hibát eredményez
?>
```

Előre definiált konstansok

A mindig elérhető, előre definiált konstansok:

__FILE__ (független a kis- vagy nagybetűs irásmódtól)

A szkript állomány neve. Ha egy olyan fájlon belül használjuk, amelyet include-oltunk, vagy require-ral töltöttük be, akkor az include-olt fájl nevét kapjuk, nem a futtatott szkriptét.

__LINE__ (független a kis- vagy nagybetűs irásmódtól)

Az aktuális szkripten belüli sor száma. Ha include-dal, vagy require-ral behívott fájlra alkalmazzuk, akkor azon a fájlon belüli pozíciót kapjuk.

PHP_VERSION

Az alkalmazott PHP verziószáma (string). Pl.: ’4.1.0’.

PHP_OS

Az operációs rendszer, ami alatt a PHP fut. Lehetséges értékei: "AIX", "Darwin" (MacOS), "Linux", "SunOS", "WIN32", "WINNT" Megjegyzés: elképzelhető, hogy később további értékek is elérhetőek lesznek.

`TRUE` (független a kis- vagy nagybetűs irásmódtól)

Igaz érték (lásd a boolean adattípust).

`FALSE` (független a kis- vagy nagybetűs irásmódtól)

Hamis érték (lásd a boolean adattípust).

`NULL` (független a kis- vagy nagybetűs irásmódtól)

A `NULL` érték (lásd a null adattípust).

E_ERROR

Egy nem feldolgozás során keletkezett helyrehozhatatlan hibát jelez.

E_WARNING

Egy olyan állapotot jelez, amiben a PHP tudja, hogy valami nincs rendben, de folytatja a működést, mert a szkript kijavíthatja a hibát. Ennek egy példája lehet egy érvénytelen reguláris kifejezés az ereg()-ben.

E_PARSE

A parser meghalt a szkriptbeli érvénytelen szintaxis miatt; helyrehozhatatlan hiba.

E_NOTICE

Valami történt, ami lehet, hogy hiba, de lehet, hogy nem :) A végrehajtás folytatódik. Ilyenre példa egy idézőjelek nélküli sztring, mint tömbindex, vagy egy változó elérése, amelynek még nem adtunk értéket.

E_ALL

Minden E_* formátumú konstans egyben. Ha az error_reporting() függvénnyel használják, akkor tetszőleges típusú hiba fellépését jelezni fogja a PHP.

Az E_* formátumú konstansokat tipikusan az error_reporting() függvény használja, hogy a hibajelzési szintet állítsa. Lásd az összes ilyen konstanst a hibakezelésnél.

Példa 8-2. A __FILE__ és a __LINE__ használata

```
<?php
function report_error ($file, $line, $message)
{
    echo "hiba történt a $file fájlban a $line sorban: $message.";
}

report_error(__FILE__, __LINE__, "Gáz van!");
?>
```

Fejezet 9. Kifejezések

A kifejezések a PHP legfontosabb építőkövei. A PHP-ban majdnem minden, amit leírsz, egy kifejezés. A lehető legegyszerűbb és mégis pontos definíciója a kifejezésnek: "mindaz, amelynek értéke van".

A kifejezések legalapvetőbb formái az állandók és a változók. Az "$a = 5" az ’5’-öt az $a változóhoz rendeli. Az ’5’-nek nyilván 5 az értéke, vagyis más szavakkal az ’5’ olyan kifejezés, melynek értéke 5 (itt az ’5’ egy egész típusú állandó).

Ez után a hozzárendelés után $a értéke 5 kell legyen, sőt, $b = $a esetén az eredménynek azonosnak kell lennie azzal, mint $b = 5 esetén. Másképp megfogalmazva az $a olyan kifejezés, amelynek értéke 5. Ha minden jól megy, pontosan ez fog történni.

Kissé bonyolultabb példák a kifejezésekre a függvények. Például tekintsük az alábbi függvényt:

```
function ize()
{
    return 5;
}
```

Megelőlegezzük ,hogy nem idegen Tőled a függvények koncepciója - ha nem így lenne, akkor olvasd el a függvényekről szóló fejezetet. Amint az elvárható, a `$c = foo()` lényegében azonos `$c = 5` beírásával. A függvények olyan kifejezések, amelyeknek az értéke a visszatérési értékük. Mivel foo() 5-tel tér vissza, a ’foo()’ kifejezés értéke 5. Általában a függvények nem arra használatosak, hogy egy bedrótozott számmal térjenek vissza, hanem valamilyen számolt származtatott értékkel.

Természetesen a változóknak a PHP-ben nem kell egésznek lenniük, és nagyon gyakran nem is azok. A PHP három féle skalártípust támogat: egész, lebegőpontos és sztring értékeket (skalárnak az számít, amit nem lehet kisebb darabokra bontani, eltérően - például - a tömböktől). A PHP két nem skalár típust is támogat: a tömb és objektum típust. Mindegyikükhöz lehet értéket rendelni, és mindegyikük lehet egy függvény visszatérési értéke.

Eddig a PHP/FI 2 használói nem találkoztak semmi különössel. Mindamellett a PHP továbbfejleszti a kifejezés fogalmát, ahogy azt sok hasonló nyelv teszi. A PHP egy kifejezés-orientált nyelv, abból a szempontból, hogy majdnem minden kifejezés. Tekintsük a példát, amivel eddig foglalkoztunk: ’$a = 5’. Látható, hogy két értéket szerepel benne: az egész konstans ’5’ és $a értéke, amelyet az 5 értékkel frissítettünk. Az igazság azonban az, hogy még egy érték is van a kifejezésben, a magáé a hozzárendelésé. A hozzárendelés maga kiértékeli az átadandó értéket, ebben az esetben az 5-öt. Ez azt jelenti gyakorlatilag, hogy a ’$a=5’, a hatásától eltekintve egy kifejezés, amelynek az értéke 5. Ha ezért olyasmit írunk, hogy ’$b=($a=5)’, akkor az olyasmi, mintha ’$a=5; $b=5;’-t írtunk volna (a pontosvessző jelöli az állítás végét). Mivel a hozzárendelés jobbról balra történik, ez írható zárójelek nélkül is: ’$b=$a=5’.

A kifejezés-orientáltság másik jó példája az elő- és utónövekményes operátor. A PHP/FI 2 használóinak és sok más nyelvet használónak ismerős lehet a változó++ és változó-- jelölés. Ezek az növelő és csökkentő operátorok. A PHP/FI 2-ben a ’$a++’-nek nincs értéke (mert nem kifejezés), és ezért nem is lehet semmihez hozzárendelni, vagy valamire használni. A PHP kiterjesztette az inkrementálás/dekrementálás lehetőségeit azzal, hogy ezeket is kifejezésekké alakította, mint ahogyan a C nyelvben is megtették. A PHP-ben a C nyelvhez hasonlóan kétféle növekményes operátor van: elő- és utó(növekményes). Mindkettő megnöveli a változó értékét, tehát a változó szempontjából nézve nincs különbség, a kifejezés értékében annál inkább. A ’++$változó’ formájú előnövekményes a megnövelt értéket adja vissza. (A PHP először növeli a változót, majd kiértékeli, innen jön az ’elő’). A ’$változó++’ alakú utónövekményes a változó eredeti értéket adja. (A PHP

először kiértékeli, aztán növeli a változót, innen jön az ’utó’). [A csökkentő operátorokkal értelemszerűen ugyanez a helyzet.]

A kifejezések egy gyakori típusai az összehasonlító kifejezések. Ezek a kifejezések 0-t vagy 1-et adnak, annak megfelelően, hogy hamisak (`FALSE`), vagy igazak (`TRUE`) (ebben a sorrendben). A PHP támogatja a > (nagyobb), >= (nagyobb, vagy egyenlő), == (egyenlő), != (nem egyenlő), < (kisebb) és a <= (kisebb, vagy egyenlő) operátorokat. Ezeket a kifejezéseket általában feltételes kifejezésekben szokták használni. Ilyen pl. az `if` kifejezés.

Az utolsó példa, amivel itt foglalkozunk, a kombinált hozzárendelés-operátor kifejezés. Már tudhatod, hogy ha $a-t eggyel szeretnéd megnövelni, akkor egyszerűen írhatod, hogy ’$a++’ vagy ’++$a’. Mi van akkor, hogy ha nem eggyel, hanem 3-mal kell megnövelni? Lehet háromszor egymás után írni, hogy ’$a++’, de ez nem valami hatékony és üdvözítő megoldás. Sokkal jobb gyakorlat az ’$a=$a+3’ használata. Ez kiértékeli az $a + 3-at, és hozzárendeli az $a-hoz, amelynek az lesz a végső eredménye, hogy $a értéke megnő hárommal. A PHP-ben - több más C-szerű nyelvhaz hasonlóan - ennél rövidebb formában is megadható: ’$a+=3’. Ez pontosan azt jelenti, hogy "vedd $a értékét, és adjál hozzá hármat, és írd vissza az eredményt $a-ba". Amellett, hogy rövidebb és érthetőbb, még gyorsabb is. ’$a+=3’ értéke, mint egy normál értékadásé a hozzárendelt érték. Ez persze NEM 3, hanem $a + 3 (hiszen ezt az értéket rendeltük $a-hoz). Az összes többi kétoperandusú operátort lehet ilyenformán hozzárendelő-operátorként használni, például ’$a-=5’ (kivon 5-öt $a-ból), ’$b*=7’ (megszorozza $b-t 7-tel), stb.

Létezik még egy kifejezés, amely elég különösnek tűnhet, hacsak nem láttad már más nyelvekben: a háromoperandusú feltételes operátor:

```
$elso ? $masodik : $harmadik
```

Értéke: ha az első kifejezés értéke `TRUE`(nem nulla), akkor a második részkifejezés, egyébként a harmadik részkifejezés. Egy segítő példa:

```
$b = $todo == ’növel’ ? $a+5 : $a-5; // $b $a+5 lesz, ha $todo értéke ’növel’
                                     // egyébként $b $a-5 lesz
```

Ez megegyezik az alábbi kóddal:

```
if ($todo == ’növel’) $b = $a+5 else $b = $a-5;
```

Az alábbi példa segíthet az elő- és utónövekményes illetve csökkentő operátorok és kifejezéseik jobb megértésében:

```
function duplaz($i)
{
    return $i*2;
```

```
}

$b = $a = 5;        /*ötöt rendelünk $a és $b változókhoz */

$c = $a++;          /* utónövekményes, $c 5 lesz, mert
                       először $a kiértékelődik, csak aztán nő */

$e = $d = ++$b;     /* előnövekményes, $b növelt értéke (6)
                       adódik tovább $d-nek és $e-nek */

                    /* itt $d és $e hat */

$f = duplaz($d++);  /* $d kétszeresének hozzárendelése MIELŐTT
                       inkrementálódik, 2*6 = 12 lesz $f */

$g = duplaz(++$e);   /* $e kétszeresének hozzárendelése MIUTÁN
                        inkrementálódik 2*7=14 lesz $g */

$h = $g += 10;       /* először $g 10-zel nő, így 24 lesz az értéke,
                        ez adódik tovább $h-nak, az is 24 lesz*/
```

A fejezet elején azt ígértük, hogy leírjuk a különböző utasítás-típusokat, és ígéretnek megfelelően a kifejezések lehetnek állítások. Nem minden kifejezés állítás azonban, mert utasítás, amely illeszkedik ’kif’ ’;’ mintára, vagyis a pontosvesszővel folytatodó kifejezés. A ’$b=$a=5;’-ben $a=5 érvényes kifejezés, de nem érvényes utasítás önmagában. A teljes ’$b=$a=5;’ már igen.

Még egy említésre méltó dolog a kifejezések igaz értéke. Sok esetben főleg feltételes elágazásokban és ciklusokban, nem igazán érdekes, hogy a kifejezésnek mi az értéke, csak az, hogy igaz (`TRUE`) vagy hamis (`FALSE`). A `TRUE` és `FALSE` konstansok (nevük nem függ a kis- és nagybetűs írásmódtól) a PHP két lehetséges logikai értékei. Amikor szükséges, a kifejezések automatikusan logikai értékké alakulnak. Lásd a típuskonverziókról szóló fejezetet további részletekért.

A PHP a kifejezések egy teljes és hatékony megvalósítását biztosítja, és teljes dokumentációja meghaladja e kézikönyv kereteit. A fenti példák rávilágítanak, hogy mi is az a kifejezés, és hogyan kell létrehozni hasznos kifejezéseket. A kézikönyv hátralevő részében az *`expr`* alatt egy érvényes PHP kifejezést értünk.

Fejezet 10. Operátorok

Operátorok precedenciája

Az operátorok precedenciája azt határozza meg, hogy milyen "szorosan" köt össze két kifejezést. Például az `1 + 5 * 3` kifejezésben, a kifejezés értéke 16, és nem 18, mert a szorzás operátorának, a ("*")-nak nagyobb precedenciája van, mint az összeadásének ("+"). Zárójelek segítségével tetszőleges precedenciát lehet felállítani egy kifejezésen belül, ha szükséges. Például a `(1 + 5) * 3` eredménye `18` lesz.

[Az asszociativitás talán megér egy kis magyarázatot. Ez azt határozza meg, hogy az adott szinten levő operátorok egymás utáni, zárójel nélküli használatát hogyan értelmezi a fordító. Egy példán keresztül talán érthetőbbé válik: 1-2+3 értelmezhető (1-2)+3-nak (= 2), vagy 1-(2+3)-nak (= -4). Az előbbi kiértékelés, amely *balról jobbra asszociatív*, a helyes ebben az esetben. A második kiértékelés pedig *jobbról balra asszociatív* - ilyen lenne például a hatványozás, amely nincs a nyelvben implementálva (erre szolgál pow()). Ha a ** *lenne* a hatványozás, akkor a 2**3**2 helyesen 2 a 9-en = 512 lenne. Vannak nem köthető (nem asszociatív) operátorok is, ilyenek az összehasonlító operátorok. A PHP-ban tehát *nem* értelmes a 2<$x<4 kifejezés.]

Az alábbi táblázat az operátorokat precedenciájuk szerint növekvő sorrendben tartalmazza.

Táblázat 10-1. Operátorok precedenciája

asszociativitás	operátorok
balról jobbra	,
balról jobbra	or
balról jobbra	xor
balról jobbra	and
jobbról balra	print
balról jobbra	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
balról jobbra	? :
balról jobbra	||
balról jobbra	&&
balról jobbra	|
balról jobbra	^
balról jobbra	&
nem köthető	== != === !==
nem köthető	< <= > >=
balról jobbra	<< >>
balról jobbra	+ - .
balról jobbra	* / %
jobbról balra	! ~ ++ -- (int) (float) (string) (array) (object) @
jobbról balra	[]
nem köthető	new

Aritmetikai operátorok

Emlékszel az elemi aritmetikára a suliból? Ezek pont úgy működnek, mint ott.

Táblázat 10-2. Aritmetikai operátorok

Példa	Név	Eredmény
$a + $b	Összeadás	$a és $b összege
$a - $b	Kivonás	$a és $b különbsége
$a * $b	Szorzás	$a és $b szorzata
$a / $b	Osztás	$a és $b hányadosa
$a % $b	Modulus	$a / $b maradéka

Az osztás operátor ("/") egész értékkel tér vissza (egy egész osztás eredményeképpen) ha a két operandusa egész (vagy string, ami egésszé konvertálódott) és a hányados is egész. Ha valamelyik operandus lebegőpontos szám, vagy az osztás eredménye nem egész, egy lebegőpontos szám a visszatérési érték.

Hozzárendelő operátorok

Az alapvető hozzárendelő operátor az "=". Elsőre azt hihetnénk, hogy ez az "egyenlő valamivel" jele. Valójában azt jelenti, hogy a bal oldali operandus [ami az egyenlőségjel bal oldalán áll] a jobb oldali kifejezést kapja értékül.

A hozzárendelő kifejezés értéke a bal oldalhoz rendelt érték. Vagyis a "$a = 3" értéke 3. Ez lehetőséget ad néhány trükkös dologra:

```
$a = ($b = 4) + 5; // $a most 9, és $b 4
```

Az alapvető hozzárendelő operátoron felül vannak ún. "kombinált" operátorok is az összes kétoperandusú aritmetikai és sztring operátorok számára, amelyek lehetővé teszik, hogy használjunk egy változót egy kifejezésben, majd rögtön be is állítsuk a változót a kifejezés értékére. Például:

```
$a = 3;
$a += 5; // $a-t 8-ra állítja, mintha  $a = $a + 5;-öt írtunk volna
$b = "Csecs ";
$b .= "Emő"; // $b "Csecs Emő" lesz, egyenértékű párja: $b = $b . "Emő";
```

A hozzárendelés az eredeti változót az újba másolja érték szerint, így az egyiken elvégzett változtatások a másikat nem érintik. Ezt fontos tudni, például egy sokszor végrehajtott ciklus belsejében nagy tömbök másolásakor. A PHP 4 támogatja a `$var =&$othervar;` szintaxisú referencia szerinti érték hozzárendelést is, de ez PHP 3-ban nem működik. A ’referencia szerinti

értékhozzárendelés’ azt jelenti, hogy mindkét változó ugyanarra az adatra fog mutatni, és nem történik meg a változó értékének lemásolása. További információkat a referenciákról a Referenciák részletesen című fejezetben olvashatsz.

Bitorientált operátorok

A bitorientált operátorok teszik lehetővé, hogy egész típusú számokon belül bizonyos biteket beállítsunk, vagy lefedjünk (maszkolás). Ha viszont az opertáron mindkét oldalán sztring típusú változó áll, akkor a bitorientált operátorok a sztringek karakterein dolgoznak úgy, hogy a karakterek ASCII kódjain végzik el a műveletet, és az eredményül adódó számot ASCII kóddal megadott karakternek értelmezi.

```
<?php
    echo 12 ^ 9; // ’5’ -öt ír ki

    echo "12" ^ "9"; // kiírja a visszaperjel karaktert (ASCII #8), mert
                     // (’1’ (ASCII #49)) ^ (’9’ (ASCII #57)) =  (ASCII #8)

    echo "hallo" ^ "hello"; // eredmény: #0 #4 #0 #0 #0
                            // ’a’ ^ ’e’ = #4
?>
```

Táblázat 10-3. Bitorientált operátorok

Példa	Név	Eredmény
$a & $b	És	Ott lesz ’1’ az eredményben, ahol $a és $b mindegyikében az a bit ’1’-es. Minden más biten ’0’.
$a | $b	Vagy	Ott lesz ’1’ az eredményben, ahol $a és $b közül legalább az egyik azon a bitje ’1’-es. Minden más biten ’0’.
$a ^ $b	Kizáró vagy	Ott lesz ’1’ az eredményben, ahol $a és $b közül csakis pontosan az egyikben ’1’ állt. Minden más biten ’0’. [Más közelítésben ott lesz ’1’ az eredményben, ahol különböző bitek álltak $a-ban és $b-ben; megint más közelítésben $a azon bitjei invertálódnak, amely helyeken $b-ben ’1’ áll]
~ $a	Nem	$a összes bitjét invertálja

Példa	Név	Eredmény
$a << $b	Eltolás balra	$a bitjeit $b számú bittel balra tolja (minden bitnyi eltolás 2-vel való szorzást jelent [amíg el nem fogynak a bitek, utolsó helyen előjelbit van ?!])
$a >> $b	Eltolás jobbra	$a bitjeit $b számú bittel jobbra tolja (minden bitnyi eltolás 2-vel való egész-osztást jelent. [Vigyázz, negatív számot inkább ne tolj jobbra!])

Összehasonlító operátorok

Az összehasonlító operátorok, mint nevük is sugallja, két érték összehasonlítására szolgálnak.

Táblázat 10-4. Összehasonlító operátorok

Példa	Név	Eredmény
$a == $b	Egyenlő	Igaz (TRUE), ha $a és $b értéke egyenlő
$a === $b	Azonos	Igaz (TRUE), ha $a és $b értéke egyenlő, és azonos típusúak (csak PHP 4)
$a != $b	Nem egyenlő	Igaz (TRUE), ha $a és $b értékei különbözők
$a <> $b	Nem egyenlő	Igaz (TRUE), ha $a és $b értékei különbözők
$a !== $b	Nem azonos	Igaz (TRUE), ha $a és $b értékei vagy típusai különbözők (csak PHP 4)
$a < $b	Kisebb mint	Igaz (TRUE), ha $a szigorúan kisebb, mint $b
$a > $b	Nagyobb mint	Igaz (TRUE), ha $a szigorúan nagyobb, mint $b
$a <= $b	Kisebb, vagy egyenlő	Igaz (TRUE), ha $a kisebb, vagy egyenlő, mint $b
$a >= $b	Nagyobb, vagy egyenlő	Igaz (TRUE), ha $a nagyobb, vagy egyenlő, mint $b

A feltételes "?:" (ternális) operátor úgyanúgy működik, mint C-ben és sok más nyelvben.

```
(kif1) ? (kif2) : (kif3);
```

A kifejezés *kif2*-t értékeli ki, ha *kif1* igaznak bizonyul (TRUE), és *kif3*-at, ha *kif1* hamis (FALSE).

Hibakezelő operátorok

A PHP egy hibakezelő operátort támogat, az at jelet (@ - kukac). PHP kifejezés elé írva a kifejezés által esetlegesen generált hibaüzenete(ke)t figyelmen kívül hagyja a rendszer.

Ha a track_errors szolgáltatás be van kapcsolva, bármilyen a kifejezés által generált hibaüzenet a $php_errormsg globális változóba kerül tárolásra. Ez a változó minden hiba esetén felülíródik, ezért használható információk kinyerése érdekében a kifejezést követően ezt minél hamarabb ellenőrizni kell.

```
<?php
/* Szándékos állomány megnyitási hiba */
$file = @file (’nem_letezo_allomany’) or
    die ("Nem lehet megnyitni, a hiba: ’$php_errormsg’");

// bármilyen kifejezésre működik, nem csak függvényekre
$ertek = @$tomb[$kulcs];
// nem ad notice szintű hibát, ha a $kulcs kulcs nem létezik
?>
```

Megjegyzés: A @ operátor csak kifejezésekre működik. Egyszerű ökölszabályként alkalmazandó, ha valaminek az értelmezett az értéke, akkor az elé a @ operátor is oda tehető. Ekképpen például használható változók, függvények és include() hívások, állandók neve előtt és sok más esetben. Nem használható azonban függvény és osztály definíciók vagy nyelvi szerkezetek (mint például `if` és `foreach` utasítások) előtt.

Lásd még: error_reporting()!

Figyelem

Jelenleg a "@" hibakezelő operátor kikapcsolja azon kritikus hibák jelentését is, amelyek megszakítják a szkript futását. Más problémák mellett, ha függvényből érkező hibaüzenetek elnyelésére használod a "@" jelet, le fog állni a szkript futása, ha nem létezik a megadott függvény vagy elírtad annak nevét.

Végrehajtó operátorok

A PHP-ban létezik egy program-végrehajtó operátor: a visszaidézőjel [aki tudja az igazi nevét, ne rejtse véka alá!] (“). Ezek nem szimpla idézőjelek! A PHP megpróbálja a sztring tartalmát

parancssorból futtatandó utasításként végrehajtani, amelynek a kimenete lesz az operátor értéke. Ez nem egyszerűen a kimenetre kerül, hanem hozzárendelhető egy változóhoz.

[Az alábbi kis példa az aktuális könyvtár tartalmát (hosszú lista, rejtett fájlok is) formázva írja ki (fix szélességű betűket használva, újsor karaktereket tiszteletben tartva)]

```
$output = ‘ls -al‘;
echo "<pre>$output</pre>";
```

Megjegyzés: A végrehajtó operatár nem használható, ha a safe mode be van kapcsolva.

Lásd még: escapeshellcmd(), exec(), passthru(), popen(), shell_exec() és system()!

Növelő/csökkentő operátorok

A PHP támogatja a C-szerű ún. elő- és utónövekményes ill. csökkentő operátorokat.

Táblázat 10-5. Növelő/csökkentő operátorok

Példa	Név	Hatás
++$a	előnövekményes	Növeli $a-t eggyel, majd visszaadja $a értékét
$a++	utónövekményes	Visszaadja $a értékét, majd növeli $a-t eggyel
--$a	előcsökkentő	Csökkenti $a-t eggyel, majd visszaadja $a értékét
$a--	utócsökkentő	Visszaadja $a értékét, majd csökkenti $a-t eggyel

Itt egy egyszerű példaprogram:

```
<?php
echo "<h3>Utónövekményes</h3>";
$a = 5;
echo "5-nek kell lennie: " . $a++ . "<br>\n";
echo "6-nak kell lennie: " . $a . "<br>\n";

echo "<h3>Előnövekményes</h3>";
$a = 5;
echo "6-nak kell lennie: " . ++$a . "<br>\n";
echo "6-nak kell lennie: " . $a . "<br>\n";

echo "<h3>Előcsökkentő</h3>";
$a = 5;
```

```
echo "5-nek kell lennie: " . $a-- . "<br>\n";
echo "4-nek kell lennie: " . $a . "<br>\n";

echo "<h3>Utócsökkentő</h3>";
$a = 5;
echo "4-nek kell lennie: " . --$a . "<br>\n";
echo "4-nek kell lennie: " . $a . "<br>\n";
?>
```

Logikai operátorok

Táblázat 10-6. Logikai operátorok

Példa	Név	Eredmény
$a and $b	És	Pontosan akkor igaz (TRUE), ha mind $a mind $b igazak (TRUE).
$a or $b	Vagy	Pontosan akkor igaz (TRUE), ha $a és $b között van igaz (TRUE).
$a xor $b	Kizáró vagy	Pontosan akkor igaz (TRUE), ha $a és $b közül pontosan egy igaz (TRUE).
! $a	Tagadás	Pontosan akkor igaz (TRUE), ha $a nem igaz (TRUE).
$a && $b	És	Pontosan akkor igaz (TRUE), ha mind $a mind $b igaz (TRUE).
$a || $b	Vagy	Pontosan akkor igaz (TRUE), ha $a és $b között van igaz (TRUE).

Az "és" és a "vagy" operátorok két variációjára a magyarázat az operátorok precedenciájában van. (Lásd: Operátorok Precedenciája.)

String operátorok

Két string operátor van. Az egyik az összefűzés operátor (’.’), amely bal és jobb oldali operandusának összefűzöttjével tér vissza. A második az összefűző-hozzárendelő operátor (’.=’), amely hozzáfűzi a jobb oldalon szereplő szöveges értéket a bal oldali operandus végéhez. Olvasd el a Hozzárendelő operátorok című részt!

```
$a = "Para ";
$b = $a . "Zita"; // most $b értéke "Para Zita"

$a = "Dárdarázó ";
$a .= "Vilmos"; // most $a értéke "Dárdarázó Vilmos"
```

Fejezet 11. Vezérlési szerkezetek

Az összes PHP szkript utasítások sorozatából áll. Az utasítás lehet hozzárendelő utasítás, függvényhívás, ciklus, feltételes utasítás, vagy üres utasítás. Az utasítások általában pontosvesszővel végződnek. Ezenkívül az utasításokat csoportosítani lehet; utasításblokkba foglalhatók kapcsos zárójelek segítségével. Az utasításblokkok maguk is utasítások. A különféle utasítástípusokat ebben a fejezetben tárgyaljuk.

if

Az `if` szerkezet az egyik legfontosabb szerkezete a legtöbb nyelvnek - így a PHP-nek is. A PHP a C-ben megismerthez hasonló `if` szerkezettel bír:

```
if (kifejezés)
    utasítás
```

Amint a kifejezésekről szóló fejezetben szerepel, a kifejezés logikai értéke értékelődik ki. Ha *`kifejezés`* `TRUE`, akkor a PHP végrehajtja az utasítást; ha `FALSE`, akkkor figyelmen kívül hagyja. Arról, hogy mely értékek tekinthetők `FALSE`-nak, a Logikai értékké alakítás c. fejezetben olvashatsz.

Az alábbi példa kiírja, hogy `a nagyobb, mint b`, ha *`$a`* nagyobb, mint *`$b`*:

```
if ($a > $b)
    print "a nagyobb, mint b";
```

Gyakran sok utasítást kell feltételhez kötve végrehajtani. Természetesen nem kell minden utasításhoz külön `if`-et írni. Az utasításokat utasításblokkba lehet összefogni. Az alábbi kód például kiírja, hogy `a nagyobb, mint b` ha *`$a`* nagyobb, mint *`$b`*, és utána hozzárendeli *`$a`* értékét *`$b`*-hez:

```
if ($a > $b) {
    print "a nagyobb, mint b";
    $b = $a;
}
```

A feltételes utasítások vég nélkül további `if` utasításokba ágyazhatók, amely a program különböző részeinek feltételes végrehajtását igen hatékonnyá teszi.

else

Gyakori, hogy egy bizonyos feltétel teljesülése esetén valamilyen utasítást kell végrehajtani, és valamilyen másik utasítást, ha nem teljesül a feltétel. Erre való az `else`. Az `else` kibővíti az `if` utasítást, hogy akkor hajtson végre utasítást, amikor az `if` kifejezés `FALSE`-ként értékelődik ki. Az alábbi kód például kiírja, hogy `a nagyobb, mint b` ha *$a $b*-nél nagyobb, egyébként az `a NEM nagyobb, mint b` üzenetet írja ki:

```
if ($a > $b) {
    print "a nagyobb, mint b";
} else {
    print "a NEM nagyobb, mint b";
}
```

Az `else` utasítás csak akkor hajtódik végre, ha az `if` kifejezés és az összes `elseif` kifejezés is `FALSE` értékű. Az elseif -ről most olvashatsz.

elseif

Az `elseif`, amint azt a neve is sugallja, az `if` és az `else` kombinációja. [perlesek figyelem, itt els*e*if-nek hívják!!!] Az `else`-hez hasonlóan az `if` utasítást terjeszti ki, hogy különböző utasításokat hajtson végre abban az esetben, ha az eredeti `if` kifejezés értéke `FALSE` lenne. Azonban az `else`-sel ellentétben csak akkor hajtra végre az alternatív kódrészt, ha az `elseif` kifejezés `TRUE`. Az alábbi kód például - $a értékétől függően - üdvözli Menő Manót, és Víz Eleket, vagy kiírja, hogy ismeretlen:

```
if ($a == "Menő Manó") {
   print "Szervusz Menő Manó! Rég láttalak!";
} elseif ($a == 'Víz Elek') { #szimpla idézőjel is használható
   print "Üdv Víz Elek!";
} else {
   print "Szervusz, idegen. Hát téged mi szél hozott ide?";
}
```

Egy `if` kifejezést több `elseif` követhet. Az első olyan `elseif` kifejezés hajtódik végre (ha van), amely értéke `TRUE`. A PHP-ban az 'else if' is (különírva) használható és ugyanúgy fog viselkedni, mint az 'elseif' (egybeírva). A szintaktikai jelentés 'kicsit' eltérő (ha ismered a C-t, nos ez pont úgy működik) de végülis ugyanaz lesz a végeredmény.

Az `elseif` ág csak akkor hajtódik végre, ha az őt megelőző `if` kifejezés, és az összes köztes `elseif` kifejezések `FALSE` értékűek, de az adott `elseif` kifejezése `TRUE`.

Vezérlési szerkezetek alternatív szintaxisa

A PHP bizonyos vezérlési szerkezeteihez egy alternatív szintaxist is nyújt; név szerint: az `if`, `while`, `for`, `foreach`, és `switch` számára. Minden esetben az alternatív szintaxisnál a nyitó kapcsos zárójel helyett kettőspontot (:) kell írni, a záró zárójel helyett pedig a vezérlési szerkezetnek megfelelő `endif;`, `endwhile;`, `endfor;`, `endforeach;`, vagy `endswitch;` utasításokat értelemszerűen.

```
<?php if ($a == 5): ?>
A most éppen 5.
<?php endif; ?>
```

A fenti példában az "A most éppen 5." egy alternatív szintaxisú `if` kifejezésbe van ágyazva. A HTML rész csak akkor íródik ki, ha $a egyenlő 5-tel.

Az alternatív szintaxis az `else`-re és az `elseif`-re is alkalmazható. Az alábbi példa egy `if` szerkezet, amiben van `elseif` és `else` is alternatív formában:

```
if ($a == 0.5):
   print "a most fél.";
   print "De vajon kitől?";
elseif ($a == 8):
   print "Nekem nyolc, hogy mennyi az a.";
   print "Úgyis megváltoztatom az értékét.";
   $a++;
else:
   print "Ez így nem vicces, hogy a se nem fél, se nem nyolc";
endif;
```

Lásd még a while, for, és if szerkezeteket további példák reményében!

`while`

A `while` ciklusok a PHP legegyszerűbb ciklusai. Éppen úgy viselkednek, mint a C nyelvbeli megfelelőik. A `while` általános szintaxisa:

```
while (kifejezés) utasítás
```

A `while` utasítás jelentése egyszerű. Azt mondja a PHP-nek, hogy mindaddig ismételje az utasítás(ok) végrehajtását, amíg a `while` kifejezés `TRUE`. Iterációnak nevezzük azt, amikor a PHP

egyszer végrehajtja az utasítást/utasításblokkot egy ciklus részeként. A kifejezés értéke a ciklus kezdetekor értékelődik ki, tehát még ha az utasításblokk belsejében hamissá is válik a feltétel, a blokk végrehajtása akkor sem áll meg, csak az iteráció végén [feltéve ha közben megint meg nem változik a feltétel]. Amikor a `while` kifejezés értéke már az első vizsgálatkor `FALSE`, akkor az utasítás(blokk) egyszer sem kerül végrehajtásra.

Az `if` szerkezethez hasonlóan több utasítást csoportosítani lehet a `while` ciklusban kapcsos zárójelekkel, vagy az alternatív szintaxis használatával:

```
while (kifejezés): utasítás ... endwhile;
```

Az alábbi példák ugyanazt csinálják - 1-től 10-ig kiírják a számokat:

```
/* 1. variáció */

$i = 1;
while ($i <= 10) {
    print $i++;  /* a kiírt érték $i, csak
                    utána növelünk
                    (post-inkrementáció) */
}

/* 2. variáció */

$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

`do..while`

A `do..while` ciklusok nagyon hasonlóak a `while` ciklusokhoz, a különbség mindössze annyi, hogy a kifejezés igaz volta itt az iteráció végén értékelődik ki, és nem az elején. A fő különbség a hagyományos `while` ciklushoz képest, hogy a `do..while` ciklus első iterációja garantáltan lefut (a kifejezés igazságértékét csak az iteráció végén ellenőrzi), amely nem garantált a hagyományos `while` ciklusnál (itt a kifejezés igazságértéke az iteráció kezdetén kerül kiértékelésre, ha értéke kezdetben `FALSE`, akkor a ciklus végrehajtása azonnal befejeződik).

Csak egy szintaxisa van a `do..while` ciklusnak:

```
$i = 0;
do {
   print $i;
```

```
} while ($i>0);
```

A fenti ciklus pontosan egyszer fut le, mert az első iteráció után, amikor a kifejezés igazságértéke vizsgálatra kerül, kiderül, hogy `FALSE` ($i nem nagyobb, mint 0) és a ciklus végrehajtása befejeződik.

Haladó C programozók már bizonyára jártasak a `do..while` ciklus másfajta használatában. Például utasításblokk közepén ki lehet lépni a blokkból, ha az utasításblokkot `do..while`(0), közé tesszük, és `break` utasítást használunk. A következő kódrészlet ezt szemlélteti:

```
do {
    if ($i < 5) {
        print "i nem elég nagy";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print" i most jó";

     ...i feldolgozása...

} while(0);
```

Ne aggódj, ha ezt nem azonnal vagy egyáltalán értetted meg. Lehet szkripteket - sőt hatékony szkripteket - írni ennek a lehetőségnek a használata nélkül is.

for

A `for` cilus a legbonyolultabb ciklus a PHP-ben. Éppen úgy viselkedik, mint a C nyelvbeli párja (–> C-ben értőknek tovább gomb). A `for` ciklus szintaxisa:

```
for (kif1; kif2; kif3) utasítás
```

[A fenti `for` szerkezettel megegyező az alábbi, remélhetőleg már ismerős kifejezés:

```
kif1;
while (kif2) {
  utasítás;
  kif3;
```

```
}
```

Remélem így már érthetőbb lesz az alábbi magyarázat].

Az első kifejezés (*`kif1`*) a ciklus kezdetén egyszer kerül végrehajtásra.

Minden iteráció elején *`kif2`* kiértékelődik. Ha értéke `TRUE`, akkor a ciklus folytatódik, és az *`utasítás`*ra kerül a vezérlés. Ha értéke `FALSE`, akkor a ciklus véget ér.

Minden iteráció végén *`kif3`* is végrehajtásra kerül.

Bármelyik kifejezést el lehet hagyni. Ha *`kif2`* üres, az azt jelenti, hogy a ciklus a végtelenségig fut [hacsak nem jön a jó tündér break utasítás képében...] (A PHP implicit `TRUE`-nak feltételezi az üres *`kif2`*-t, mint a C.) Ez nem annyira haszontalan, mint elsőre amennyire elsőnek tűnik, hiszen gyakran fejezheted be a ciklust egy feltételes kifejezésbe ágyazott `break` kifejezéssel a `for` feltétel kifejezésének kiértékelése helyett.

Nézd az alábbi példákat, mindegyikük kiírja a számokat 1-től 10-ig:

```
/* téma */

for ($i = 1; $i <= 10; $i++) {
    print $i;
}

/* 1. variáció */

for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}

/* 2. variáció */

$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}

/* 3. variáció - Coda :-) */

for ($i = 1; $i <= 10; print $i, $i++);
```

Természetesen "a téma" a legbarátságosabb (vagy esetleg a 3. variáció). Sok helyen hasznos azonban, hogy üres kifejezés is írható `for` ciklusba...

A PHP a `for` ciklus esetén is megengedi az alternatív szintaxishasználatát:

```
for (kif1; kif2; kif3): utasítás; ...; endfor;
```

Más nyelvekben létezik az ún. `foreach` szerkezet tömbök vagy hash-ek bejárására. A PHP 3-ban nincs ilyen, de a PHP 4-ben implementálták (lásd: foreach). PHP 3-ban a while, a list() és az each() szerkezeteket használhatod erre a célra. Nézd meg ezeknek a dokumentációját alkalmazási példákért.

`foreach`

A PHP 4-ben (nem a PHP 3-ban!) a Perlhez és más nyelvekhez hasonlóan létezik az ún. `foreach` szerkezet is. Ez jól használható eszközt ad a tömbökön végzett iterációkhoz. Két szintaxisa létezik, a második egy apró, de hasznos kiegészítéssel nyújt többet az elsőhöz képest.

```
foreach(tömb_kifejezés as $ertek) utasítás
foreach(tömb_kifejezés as $kulcs => $ertek) utasítás
```

Az első forma végigmegy a `tömb_kifejezés` szolgáltatta tömbön. Minden alkalommal az aktuális elem értéke a `$ertek` változóba kerül, és a belső tömb mutató növelésre kerül. (A következő alkalommal tehát a soron következő elemet fogja venni).

A második forma ugyanezt végzi el, de az aktuális elem kulcsa a `$kulcs` változóba kerül.

> **Megjegyzés:** Amikor a `foreach` indul, a belső tömb mutató az első elemre áll. Ez azt jelenti, hogy nem kell meghívni a reset() függvényt egy `foreach` ciklus előtt.

> **Megjegyzés:** Szintén fontos megjegyezni, hogy a `foreach` függvény a megadott tömb egy másolatával dolgozik, nem magával a tömbbel, ezért az each()-el ellentétben az eredeti tömb mutatója nem változik meg, és a tömbön végzett módosítások sem kerülnek be az eredeti tömbbe.

> **Megjegyzés:** A `foreach` nem támogatja a @ hiba-elnyelő operátor használatát!

Talán már tudod, hogy az alábbi példák egyenértékűek:

```
reset ($tomb);
while (list(, $ertek) = each ($tomb)) {
    echo "Érték: $ertek<br>\n";
}

foreach ($tomb as $ertek) {
    echo "Érték: $ertek<br>\n";
}
```

Az alábbiak is azonos eredményt szolgáltatnak:

```
reset ($tomb);
while (list($kulcs, $ertek) = each ($tomb)) {
    echo "Kulcs: $kulcs, Érték: $ertek<br>\n";
}

foreach ($tomb as $kulcs => $ertek) {
    echo "Kulcs: $kulcs, Érték: $ertek<br>\n";
}
```

Néhány további felhasználási példa:

```
/* első foreach példa: csak érték */

$tomb = array (1, 2, 3, 17);

foreach ($tomb as $ertek) {
   print "Az aktuális értéke \$tomb-nek: $ertek.\n";
}

/* második foreach példa: érték (a kulcs csak illusztráció) */

$tomb = array (1, 2, 3, 17);

$i = 0; /* csak pedagógiai okokból :) */

foreach ($tomb as $ertek) {
    print "\$tomb[$i] => $ertek.\n";
    $i++;
}

/* harmadik foreach példa: kulcs és érték */

$a = array (
    "egy" => 1,
    "kettő" => 2,
    "három" => 3,
    "tizenhét" => 17
);
```

```
foreach ($tomb as $kulcs => $ertek) {
    print "\$tomb[$kulcs] => $ertek.\n";
}

/* negyedik foreach példa: többdimenziós tömb */

$tomb[0][0] = "a";
$tomb[0][1] = "b";
$tomb[1][0] = "y";
$tomb[1][1] = "z";

foreach ($tomb as $belsotomb) {
    foreach ($belsotomb as $ertek) {
        print "$ertek\n";
    }
}

/* ötödik foreach példa: dinamikus tömbök */

foreach(array(1, 2, 3, 4, 5) as $ertek) {
    print "$ertek\n";
}
```

break

A `break` azonnal kilép az aktuális `for`, `foreach`, `while`, `do..while` ciklusból vagy `switch` szerkezetből.

A `break` elfogad egy elhagyható szám paramétert, amely megadja, hogy hány egymásba ágyazott struktúrából kell egyszerre 'kiugrani'.

```
$tomb = array ('egy', 'kettő', 'három', 'négy', 'stop', 'öt');
while (list (, $ertek) = each ($tomb)) {
    if ($ertek == 'stop') {
        break;    /* írhattál volna ide 'break 1;'-et is */
    }
    echo "$ertek<br>\n";
}

/* Az elhagyható paraméter használata */

$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "5 esetén<br>\n";
        break 1;  /* csak a switch-ből lép ki */
    case 10:
        echo "10 esetén kilépés<br>\n";
        break 2;  /* a switch és a while befejezése */
    default:
```

```
        break;
    }
}
```

continue

A `continue` ciklusok belsejében használható arra, hogy átugorjuk az aktuális iteráció hátralevő részét, és a végrehajtást a következő iterációval folytassuk.

A `continue` elfogad egy elhagyható szám paramétert, amely megadja, hogy hány egymásba ágyazott struktúrának a hátralévő részét kell átugrani.

```
while (list ($kulcs, $ertek) = each ($tomb)) {
    if (!($kulcs % 2)) { // a páros indexűek kihagyása
        continue;
    }
    valami_paratlan_dolog ($ertek);
}

$i = 0;
while ($i++ < 5) {
    echo "Külső while<br>\n";
    while (1) {
        echo "  Középső while<br>\n";
        while (1) {
            echo "  Belső while<br>\n";
            continue 3;
        }
        echo "Ezt soha nem fogja kiírni.<br>\n";
    }
    echo "Ezt sem...<br>\n";
}
```

switch

A `switch` kifejezés hasonló egy sereg IF kifejezéshez, ahol a kifejezésekben ugyanaz szerepel. [Pont olyan, mint a C-ben, C-s gyakorlattal rendelkezőknek Tovább gomb]. Gyakori, hogy ugyanazt a változót (vagy kifejezést) kell összehasonlítani több különböző értékkel, és más-más kódot végrehajtani a változó (kifejezés) értékétől függően. Pontosan erre való a `switch`.

Az alábbi két példa két különböző módon írja ki ugyanazt, az egyik egy sor `if` kifejezést használ, a másik pedig a `switch`-et:

```
if ($i == 0) {
    print "i most 0";
}
if ($i == 1) {
    print "i most 1";
}
if ($i == 2) {
    print "i most 2";
}

switch ($i) {
    case 0:
        print "i most 0";
        break;
    case 1:
        print "i most 1";
        break;
    case 2:
        print "i most 2";
        break;
}
```

A hibák elkerülése végett fontos megérteni, hogy hogyan kerül végrehajtásra a `switch` szerkezet. A `switch` vagyis utasításról utasításra hajtódik végre. Nem hajtódik végre semmilyen utasítás, csak akkor, ha egy olyan `case` kifejezést talál a PHP, amely egyezik a `switch` kifejezés értékével. Ezután a PHP addig folytatja az utasítások végrehajtását, amíg el nem éri a `switch` blokk végét, vagy nem találkozik egy `break` utasítással. FONTOS! Ha nem nincs `break` egy `case`-hez tartozó utasítás(sorozat) végén, akkor a PHP végrehajtja a soron következő `case`-hez tartozó utasításokat is! Például:

```
switch ($i) {
    case 0:
        print "i most 0";
    case 1:
        print "i most 1";
    case 2:
        print "i most 2";
}
```

Itt, ha $i értéke 0, akkor a PHP az összes kiíró utasítást végrehajtja! Ha $i értéke 1, akkor a PHP az utolsó két print-et hajtja végre, és csak ha $i értéke 2, akkor kapod a 'kívánt' eredményt (csak az 'i most 2' íródik ki). Tehát nagyon fontos nem elfelejteni a `break` utasítást (bár bizonyos körülmények között lehet, hogy pont ennek elhagyása a szándékos).

A `switch` kifejezésben a feltétel csak egyszer értékelődik ki és a kapott eredmény lesz összehasonlítva a `case` kifejezések mindegyikével. Ha `elseif` kifejezéseket használsz, a

kifejezések újra és újra kiértékelődnek. [és újra és újra be kell gépelni. Ez nem csak fárasztó, de hiba forrása is lehet.] Ha a kifejezés bonyolult, vagy egy ciklus belsejében van, a `switch` a gyorsabb.

Egy eset (case) utasításlistája üres is lehet, így a vezérlés a következő case-címkére adódik.

```
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i 3-nál kisebb, de nem negatív";
        break;
    case 3:
        print "i pont 3";
}
```

Egy különleges eset a default [alapértelmezett] címke. Ez a címke bármivel egyezik, amivel a korábbi `case` elemek nem egyeztek. Ennek kell az utolsó elemnek lennie. Például:

```
switch ($i) {
    case 0:
        print "i most 0";
        break;
    case 1:
        print "i most 1";
        break;
    case 2:
        print "i most 2";
        break;
    default:
        print "i se nem 0, se nem 1, se nem 2";
}
```

A `case` kifejezés tetszőleges kifejezés, aminek egyszerű a típusa, vagyis egész vagy lebegőpontos szám, vagy string. Tömbök és objektumok itt nem használhatók, csakis egy-egy elemük ill. változójuk egyszerű típusként.

Az alternatív szintaxis működik a switch-ekkel is. Bővebb információért lásd: Vezérlési szerkezetek alternatív szintaxisa.

```
switch ($i):
    case 0:
        print "i most 0";
        break;
    case 1:
        print "i most 1";
        break;
    case 2:
```

```
        print "i most 2";
        break;
    default:
        print "i se nem 0, se nem 1, se nem 2";
endswitch;
```

declare

A `declare` egy kódblokk számára adott futtatási direktívák beállítását teszi lehetővé. A `declare` szintaxisa hasonló a vezérlési szerkezetekéhez:

```
declare (direktíva) utasítás
```

A `direktíva` rész a `declare` blokk működését szabályozza. Jelenleg csak egy direktíva használható, a `ticks`. (Lásd lejjebb a ticks részleteit)

A `declare` blokk `utasítás` része mindig egyszer fut le. Az, hogy miképp, és milyen mellékhatásokkal, a `direktíva` részben megadottaktól függ.

Tick-ek

A tick egy olyan esemény, amely minden *`N`* db alacsony szintű utasítás végrehajtásakor bekövetkezik a `declare` blokkban. Az *`N`* értéket a `ticks=`*`N`* szintaxissal kell megadni a `declare` blokk `direktíva` részében.

Az egyes tick-ekre bekövetkező esemény(ek) a register_tick_function() függvénnyel állítható(ak) be. Lásd az alábbi példát. Akár több esemény is bekövetkezhet egy tick-re.

Példa 11-1. A PHP kód egy részének időmérése

```
<pre>
<?php
// Ez a függvény megjegyzi a hívása időpontjait
function idopontok ($visszaadni = FALSE)
{

    static $idopontok;

    // Visszaadja a $profile tartalmát, és törli
    if ($visszaadni) {
        $idok = $idopontok;
        unset ($idopontok);
        return ($idok);
```

```
    }

    $idopontok[] = microtime();
}

// A tick kezelő beállítása
register_tick_function("idopontok");

// Beállítjuk az első időpontot a declare előtt
idopontok();

// A kódblokk futtatása, minden második utasítás egy tick
declare (ticks = 2) {
    for ($x = 1; $x < 50; ++$x) {
        echo similar_text (md5($x), md5($x*$x)), "<br>";
    }
}

// Az időmérő függvény adatainak kiírása
print_r (idopontok(TRUE));
?>
</pre>
```

A fenti példa a declare blokkban lévő PHP kód sebességét méri, rögzítve minden második alacsonyszintű utasítás végrehajátásának időpontját. Ez az információ alkalmas lehet arra, hogy megtaláld a lassan futó részeket a kódodban. Ezt a hatást másképp is el lehet érni, de tick-eket használva sokkal kényelmesebb és könnyebben megvalósítható megoldást kapsz.

A tick-ek kiválóan alkalmasak hibakeresésre, egyszerű multitasking megvalósítására, háttérben futattott I/O-ra, és sok más feladatra.

Lásd még a register_tick_function() és az unregister_tick_function() függvényeket.

return

A return() utasítás függvényen belül használva azonnal befejezi a folyó függvény futását, és a paramétereként megadott érték szolgáltatja a függvény visszatérési értékét. A return() az eval() függvénnyel futatott kód vagy a szkript futását is leállítja.

A globális érvényességi körben használva a folyó szkript futását szakítja meg. Ha ez a szkript az include() vagy a require() hatására lett futtatva, akkor a vezérlés visszaadódik arra a fájlra, ahol ezek az utasítások szerepelnek, valamint include() esetén a return() paramétere lesz az include() utasítás visszatérési értéke. Ha a return() a fő szkriptben lett kiadva, akkor befejeződik a szkript futása. Ha ez a auto_prepend_file vagy auto_append_file konfigurációs beállításban szereplő fájlok valamelyikében történik (lásd: konfigurációs fájl) akkor, (csak) ezeknek a futása fejeződik be.

További magyarázatért lásd Visszatérési értékek c. fejezetet!

> **Megjegyzés:** Mivel a return() nyelvi szerkezet és nem függvény, a paraméterét körülvevő zárójelek *elhagyhatók*. Valójában az a megszokottabb, hogy nem is használják, bár ez egyáltalán nem számít.

require()

A require() beilleszti és feldolgozza a megadott fájlt. Ennek részletes mikéntjéről, lásd include()!

A require() és az include() megegyezik egymással a hibakezelését leszámítva. Az include() nem fatális hibát, figyelmeztetést generál, a require() viszont fatális hibát jelez. Másszóval, ahol az igényelt fájl nemlétekor a futást meg kell szakítani, ajánlott a require(). Az include() nem így viselkedik, a hibától függetlenül a szkript futtatása folytatódik. Bizonyosodj meg, hogy a include_path helyesen van beállítva!

Példa 11-2. Egyszerű require() példák

```
<?php

require ’prepend.php’;

require $valamifajl;

require (’valamifajl.txt’);

?>
```

Lásd az include() oldalát még több példáért!

> **Megjegyzés:** PHP 4.0.2 előtt, a következők szerint működött. A require() mindig beolvasta a kívánt fájlt, még ha az a require()-t tartalmazó sorra soha nem is került vezérlés. A feltételes szerkezetek nem befolyásolták a működését. Mégis, ha a require()-t tartalmazó sorra nem került vezérlés a megadott fájlban lévő kód nem futott le. Ehhez hasonlóan, a ciklusok sem befolyásolták a működését. Habár a fájlban szereplő kód függött az azt körülölelő ciklustól, a require() maga csak egyszer történt meg.

Lásd még: include(), require_once(), include_once(), eval(), file(), readfile(), virtual() és include_path!

include()

Az include() beilleszti és feldolgozza a megadott fájlt.

Az alábbiak igazak a require()-ra is. A require() és az include() megegyezik egymással a hibakezelését leszámítva. Az include() nem fatális hibát, figyelmeztetést generál, a require() viszont fatális hibát jelez. Magyarán, ahol az igényelt fájl nemlétekor a futást meg kell szakítani, ajánlott a require(). Az include() nem így viselkedik, a hibától függetlenül a szkript futtatása folytatódik. Bizonyosodj meg, hogy a include_path helyesen van beállítva!

A fájl beillesztése során a megadott fájl örökli az include() helyén érvényes változó hatáskört. Bármely változó, amely azon a ponton elérhető, elérhető a beillesztett fájlban is.

Példa 11-3. Egyszerű include() példa

```
valtozok.php
<?php

$szin     = ’zöld’;
$gyumolcs = ’alma’;

?>

teszt.php
<?php

echo "Egy $szin $gyumolcs"; // Egy

include ’valtozok.php’;

echo "egy $szin $gyumolcs"; // Egy zöld alma

?>
```

Függvény belsejében a megadott fájlban szereplő kód úgy fog viselkedni, mintha az magában a függvényben szerepelt volna. Ez azt jelenti, hogy a fájl örökli a változók érvényességi körét.

Példa 11-4. Függvényen belüli beillesztés

```
<?php

function ize()
{
global $szin;

    include ’valtozok.php’;

    echo "Egy $szin $gyumolcs";
}

/* valtozok.php az ize() függvény hatóköréébe esik, *
 * így a $gyumolcs nem elérhető a főggvényen kívül. *
 * A $szin igen, mivel globálisnak lett deklarálva. */

ize();                     // Egy zöld alma
echo "Egy $szin $gyumolcs"; // Egy zöld

?>
```

Ha egy fájlt beillesztünk az include()-dal vagy require()-ral, akkor a célfájl elején az elemző kilép a PHP módból HTML módba, majd visszaáll PHP módba a fájl végén. Ennek okán bármely beillesztendő fájlban levő PHP kódot közre kell fogni egy érvényes PHP kezdő- és zárójelöléssel.

Ha az include()-dal hívott fájl HTTP-n keresztül érkezik az "fopen wrapper"-ek használatával, és a célszerver PHP kódként feldolgozza a fájlt, akkor átadhatsz változókat a hívott fájlnak HTTP GET lekérési formában. Ez nem teljesen ugyanaz, mintha a include()-dal hívott fájl örökölné a helyi változókat, mivel a szkript valójában a távoli szerveren fut le, és a futási eredmény kerül beépítésre a helyi szkriptbe.

Példa 11-5. include() HTTP-n keresztül

```
/* Ezek a példák feltételezik, hogy a szerver be van állítva a .php      *
 * fájlok feldolgozására és nincs beállítva a .txt fájlok feldolgozására *
 * A 'működik' azt jelenti, hogy az $ize és $bigyo változók elérhetőek   *
 * a hívott fájlban.                                                     */

// Nem működik: a file.txt nem kerül feldolgozásra
include ("http://szerver/file.txt?ize=1&bigyo=2");

// Nem működik: egy 'file.php?ize=1&bigyo=2' nevű fájlt keres a helyi gépen
include ("file.php?ize=1&bigyo=2");

// Működik
include ("http://szerver/file.php?ize=1&bigyo=2");

$ize = 1;
$bigyo = 2;
include ("file.txt");  /* Működik */
incluie ("file.php");  /* Működik */
```

A kapcsolodó információkért lásd még: Távoli fájlok elérése, fopen() és file()

Mivel az include() és a require() különleges nyelvi elem, kapcsos zárójelekkel kell közrefogni, ha egy feltételes utasításon belül szerepel.

Példa 11-6. include() feltételes blokkon belül

```
/*Ez NEM JÓ, és nem a várt eredményt adja */

if ($feltetel)
    include($file);
else
    include($other);

/* Ez a HELYES */

if ($feltetel) {
    include($file);
} else {
    include($other);
}
```

`return` utasítást lehet elhelyezni egy include()-olt fájlban annak érdekében, hogy a kiértékelés ott befejeződjön, és visszaadjon egy értéket a hívó szkriptnek. A visszatérési értéket ugyanúgy használhatod, mint egy közönséges függvénynél.

> **Megjegyzés:** PHP 3, a return nem jelenhet meg függvény blokkon kívül máshol, amely esetben a függvénybő ltörténő visszatérést jelöli.

Példa 11-7. Az include() és a return() utasítás

```
return.php
<?php

$var = 'PHP';

return $var;

?>

noreturn.php
<?php

$var = 'PHP';

?>

testreturns.php
<?php

$ize = include 'return.php';

echo $ize; // kiírja: 'PHP'

$bigyo = include 'noreturn.php';

echo $bigyo; // kiírja: 1

?>
```

`$bigyo` értéke `1`, mert a beillesztés sikeres volt. Figyeld meg a különbséget a két fenti példa között. Az első a return() segítségével visszaadott egy értéket, a második nem. Létezik még néhány egyéb módja is változók beemelésének a fopen(), file() segítségével, vagy include() és Kimenet szabályzó függvények együttes használatával.

Lásd még: require(), require_once(), include_once(), readfile() és virtual()!

require_once()

Az require_once() beilleszt és feldolgoz fájlokat a program futása közben. Ez hasonló az require() működéséhez, azzal a fontos különbséggel, hogy ha a már egyszer beillesztésre került kódot a PHP nem próbálja meg ismét betölteni.

A require_once() használatos azokban az esetekben, amikor ugyanaz a fájl esetleg többször kerülhet beillesztésre a szkript futása során, de biztosítani kell, hogy ez ténylegesen csak egyszer történjen meg, így megelőzve a függvények újradefiniálását, változók értékének átállítását, stb.

További példákhoz az require_once() és include_once() használatához nézd meg a PEAR kódot, ami a legfrissebb PHP disztribúciókban megtalálható.

Megjegyzés: A require_once() PHP 4.0.1pl2 verzióban került a nyelvbe.

Lásd még: include(), require(), include_once(), get_required_files(), get_included_files(), readfile() és virtual()!

include_once()

Az include_once() beilleszt és feldolgoz fájlokat a program futása közben. Ez hasonló az include() működéséhez, azzal a fontos különbséggel, hogy ha a már egyszer beillesztésre került kódot a PHP nem próbálja meg ismét betölteni.

Az include_once() használatos azokban az esetekben, amikor ugyanaz a fájl esetleg többször kerülhet beillesztésre a szkript futása során, de biztosítani kell, hogy ez ténylegesen csak egyszer történjen meg, így megelőzve a függvények újradefiniálását, változók értékének átállítását, stb.

További példákhoz az require_once() és include_once() használatához nézd meg a PEAR kódot, ami a legfrissebb PHP disztribúciókban megtalálható.

Megjegyzés: Az include_once() PHP 4.0.1pl2 verzióban került a nyelvbe.

Lásd még: include(), require(), require_once(), get_required_files(), get_included_files(), readfile() és virtual()!

Fejezet 12. Függvények

Felhasználó által definiált függvények

Függvényeket a következő szintaxis szerint definiálhatod:

```
function foo ($arg_1, $arg_2, ..., $arg_n)
{
    echo "Példa függvény.\n";
    return $retval;
}
```

Bármely érvényes PHP kód megjelenhet egy függvényen belül, akár még más függvény vagy osztály definíciók is.

PHP 3-ban a függvényeket definiálni kell, mielőtt hivatkozás történik rájuk (függvényhívás előtt). PHP 4-ben nincs ez a megkötés.

A PHP nem támogatja a függvények polimorfizmusát (többalakúságát), a függvényekdefiníciókat nem lehet megszüntetni vagy újradefiniálni egy már definiált függvényeket.

A PHP 3 nem támogatja a változó számú függvényargumentumokat, bár az argumentumok kezdőértéke támogatott. Lásd az Argumentumok kezdőértéke című részt bővebb információért. A PHP 4 mindkettő lehetőséget támogatja. Lásd a Változó számú függvényargumentumok című részt és a func_num_args(), func_get_arg() és a func_get_args() függvényeket részletesebb leírásért.

Függvényargumentumok

Az információ a függvényekhez az argumentumlistán keresztül jut el, ami egy vesszővel határolt változó és/vagy konstanslista.

A PHP támogatja az érték szerinti (ez az alapértelmezett) referenciakénti paraméterátadást is, és az argumentumok kezdőértékét. A változó hosszúságú argumentumlisták csak a PHP 4-ben jelentek meg. Lásd a változó hosszúságú argumentumlistákat és a func_num_args(), func_get_arg() és a func_get_args() függvényeket részletesebb leírásért. PHP 3-ban hasonló hatás érhető el a függvénynek tömb típusú változó paraméterként történő átadásával:

```
function tombot_kezel($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

Referencia szerinti argumentumfeltöltés

Alapértelmezésben a függvény paraméterei érték szerint adódnak át (vagyis ha megváltoztatod a változót a függvényen belül, annak a függvényen kívülre nincs hatása). Ha szeretnéd megengedni, hogy a függvény módosítsa az átadott paramétereket, referencia szerint kell átadni azokat.

Ha egy függvényargumentum mindig referencia szerint kell átadni, akkor a függvénydefinícióban az argumentum neve elé egy & jelet kell írni.

```
function fgv_extrakkal(&$string)
{
    $string .= ’és a szükséges plusssz.’;
}
$str = ’Ez egy karakterfüzér, ’;
fgv_extrakkal($str);
echo $str;    // kiírja, hogy ’Ez egy karakterfüzér, és a szükséges plusssz.’
```

Argumentumok kezdőértékei

Bármely függvény skalár-argumentumainak megadhatsz kezdőértéket a C++ szintaxisnak megfelelően:

```
function kavet_csinal ($tipus = "cappucino")
{
    return "Csinálok egy pohár " . $tipus . "t.\n";
}
echo kavet_csinal ();
echo kavet_csinal ("espresso");
```

A fenti kód kimenete:

```
Csinálok egy pohár cappucinot.
Csinálok egy pohár espressot.
```

A kezdőértéknek konstans kifejezésnek kell lennie, nem lehet pl. változó vagy objektum.

Figyelj arra, hogy a kezdőértékkel rendelkező argumentumok más argumentumoktól jobbra helyezkedjenek el; különben a dolgok nem úgy mennek majd, ahogy azt várnád [hanem úgy, ahogy írtad :)]. Lásd a következő kódot:

```
function joghurtot_keszit ($type = "acidophilus", $flavour)
{
```

```
    return "Készítek egy köcsög $flavour ízű $type-t.\n";
}

echo joghurtot_keszit ("eper");   // nem úgy működik, mint szeretnéd !?!?
```

A fenti példa kimenete:

```
Warning: Missing argument 2 in call to joghurtot_keszit() in
/usr/local/etc/httpd/htdocs/phptest/functest.php on line 41
Készítek egy köcsög ízű eper-t.
```

Most hasonlítsd össze az alábbival:

```
function joghurtot_keszit ($flavour, $type = "acidophilus")
{
    return "Készítek egy köcsög $flavour ízű $type-ot.\n";
}

echo joghurtot_keszit ("eper");   // ez már jó
```

A fenti példa kimenete:

```
Készítek egy eper ízű acidophilus-t.
```

> **Megjegyzés:** [Azért vannak kiskapuk... Az argumentumokat adjuk át egy asszociatív tömbben! Az alapértékeket pedig egy másik tömbben. Ezután vegyük végig az alapértékeket tartalmazó tömb elemeit. Ha nem definiáltak a kapott tömbben, akkor írjuk felül az alapértékeket tartalmazó tömb megfelelő elemével. Így bármelyik argumentum elhagyható, sőt még annyit is nyerünk a dolgon, hogy nem kell megjegyeznünk a paraméterek sorrendjét. (forrás: egy bő lére eresztett amerikai PHP 3 könyv)]

Változó hosszúságú argumentumlista

A PHP 4 támogatja a változó hosszúságú argumentumlistát a felhasználók által definiált függvényekben. Valóban nagyon egyszerű kezelni ezt a func_num_args(), func_get_arg() és a func_get_args() függvényekkel.

Semmilyen különleges szintakszist nem igényel és az argumentumlista lehet explicit módon adott és viselkedhet úgy is, mint egy normál függvény.

Visszatérési értékek

Az elhagyható return állítást használva adhatnak vissza értéket a függvények. Bármely típus visszaadható, beleértve a listákat és az objektumokat is. A függvény végrehajtása azonnal befejeződik, és a vezérlés visszakerül a függvényhívás utáni pozícióba. További részletes információkért lásd: return()!

```
function negyzete ($num)
{
    return $num * $num;
}
echo negyzete (4);   // kiírja ’16’.
```

Megjegyzés: [Ha nincs return, az utolsó kifejezés értékével tér viszsza a függvény]

Több értéket nem tud visszaadni a függvény, de hasonló hatás érhető el ezen többszörös értékek listába szervezésével.

```
function kis_szamok()
{
    return array (0, 1, 2);
}
list ($nulla, $egy, $ketto) = kis_szamok();
```

Ha a függvénynek referenciával kell visszatérnie, akkor az & referencia operátort kell alkalmaznod a függvény deklarásákor és a a visszatérési érték megadásakor is.

```
function &referenciat_ad_vissza()
{
    return &$valtozo;
}

$hivatkozas = &referenciat_ad_vissza();
```

További információkért lásd a Referenciák fejezetet!

old_function

Az old_function kifejezés lehetővé teszi függvény deklarálását a PHP/FI2 szintakszisnak megfelelően (kivéve persze, hogy a ’function’-t ki kell cserélned ’old_function’-ra)

Ez egy ellenjavallt tulajdonság, és csak a PHP/FI2->PHP 3 konverter használhatja.

Figyelem

Az old_function segítségével deklarált függvényeket nem hívhatod a PHP belső részéből. Több minden mellett ez azt jelenti, hogy nem használhatod olyan függvényekben, mint az usort(), array_walk(), és a register_shutdown_function(). Ez a korlátozás megkerülhető egy olyan keretfüggvény megírásával (normál PHP 3 formában), amely meghívja az old_function-t.

Függvényváltozók

A PHP lehetővé teszi a függvényváltozók használatát. Ha egy változónevet kerek zárójelek követnek, akkor a PHP megkeresi a változó értékével azonos nevű függvényt, és megpróbálja azt végrehajtani. Ezt többek között visszahívandó (callback) függvények vagy függvénytáblák implementálására használható.

A függvényváltozók nem fognak működni az olyan nyelvi elemekkel, mint például az echo(), unset(), isset(), empty() vagy include(). Habár a print() nyelvi elem kivétel. Ez az egyik legjelentősebb különbség a PHP függvények és nyelvi elemek között.

Példa 12-1. Függvényváltozó példa

```
<?php
function ize()
{
    echo "Az ize()-ben<br>\n";
}

function bigyo($param = ”)
{
    echo "A bigyo()-ban; az argumentum:’$param’.<br>\n";
}

$func = ’ize’;
$func();
$func = ’bigyo’;
$func(’Stex van Boeven’);
?>
```

Lásd még a változó változókat és a function_exists() függvényt.

Fejezet 13. Osztályok, objektumok

class

Az osztály (objektumtípus) változók és rajtuk műveletet végző függvények [metódusok] együttese. Osztályt az alábbi szintakszis szerint lehet definiálni:

```
<?php
class Kosar
{
    var $dolgok;  // A kosárban levő dolgok

    // berak a kosárba $db darabot az $sorsz indexű dologból

    function berak ($sorsz, $db)
    {
        $this->dolgok[$sorsz] += $db;
    }

    // kivesz a kosárból $db darabot az $sorsz indexű dologból

    function kivesz ($sorsz, $db)
    {
        if ($this->items[$sorsz] > $db) {
            $this->items[$sorsz] -= $db;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

Ez definiál egy Kosar nevű osztályt, ami a kosárban levő áruk asszociatív tömbjéből áll, és definiál 2 funkciót hozzá, hogy bele lehessen rakni, illetve kivenni a kosárból.

Figyelem

Az alábbi figyelmeztetések a PHP 4-es verziójára érvényesek.

Az `stdClass` név le van foglalva, mivel belsőleg a Zend motor használja. Emiatt nem lehet egy `stdClass` nevű osztályod PHP-ben.

A `__sleep` és `__wakeup` nevek speciálisak a PHP osztályokban. Nem szabad ezeket a neveket használni a metódusokhoz, hacsak nem a hozzájuk tartozó speciális funkcionalitást szeretnéd felülírni. Lásd az alábbi részletes leírást.

A PHP számára lefoglalt az összes __ karakterekkel kezdődő metódusnév, mint speciális név. Nem szabad __ karakterekkel kezdődő metódusnevet adni, hacsak nem a dokumentált funkciókat szeretnéd használni.

Megjegyzés: PHP 4-ben csak állandó értékű `var` inicializáló értékek megengedettek. Ha nem állandó értéket szeretnél beállítani, írnod kell egy inicializáló metódust, ami automatikusan

meghívódik, amikor egy objektumpéldány létrejön az osztályból. Az ilyen metódusokat hívjuk konstruktoroknak (lásd lentebb).

```
/* Egyik alábbi értékadás sem működik PHP 4-ben */
class Kosar
{
    var $mai_datum = date("Y. m. d.");
    var $nev = $csaladi_nev;
    var $tulajdonos = 'Ferenc ' . 'János';
    var $termekek = array("Videó", "TV");
}

/* Így kell a fenti beállításokat elérni */
class Kosar
{
    var $mai_datum;
    var $nev;
    var $tulajdonos;
    var $termekek;

    function Kosar()
    {
        $this->mai_nap = date("Y. m. d.");
        $this->nev = $GLOBALS['csaladi_nev'];
        /* stb. . . */
    }
}
```

Az osztályok típusok, vagyis az aktuális változók tervrajzai. A kívánt típusú változót a `new` operátorral hozhatod létre.

```
$kosar = new Kosar;
$kosar->berak("10", 1);

$masik_kosar = new Kosar;
$masik_kosar->berak("0815", 3);
```

Ez létrehozza a Kosar objektumosztály $kosar és $masik_kosar nevű objektumpéldányait. A $kosar objektum berak() függvényét meghívtuk, hogy a 10-es számú árucikkből rakjon 1 darabot a kosárba. Három darab 0815 számú ternék került a $masik_kosar nevű kosárba.

Mind a $kosar, mind a $masik_kosar objektumoknak megvannak a berak() és kivesz() metódusai, és tulajdonságai. Ezek azonban egymástól független metódusok és tulajdonságok. Az objektumokról hasonlóan gondolkozhatsz, mint a könyvtárakról az állományrendszerben. Lehetséges, hogy két különböző OLVASSEL.TXT állományod van, ha ezek két különböző könyvtárban vannak. Úgy mint a könytáraknál, ahol meg kell adnod a teljes elérési utat, hogy egy állományra szeretnél hivatkozni a gyökérkönyvtárban, a teljes metódusnevet meg kell adnod, hogy meg tudd azt hívni. A PHP nyelvben a gyökérkönyvtár analógiája a globális környezet, és az elérési út elválasztója a ->. Ezért a $kosar->dolgok név és a $masik_kosar->dolgok név két különböző változót ad meg. Figyeld meg, hogy a változót $kosar->dolgok néven kell elérni, és nem $kosar->$dolgok néven, azaz a PHP változók neveiben csak egy dollárjelet kell tenned.

```
// helyes, egy dollárjel
$kosar->dolgok = array("10" => 1);

// helytelen, mivel a $kosar->$dolgok értelme $kosar->""
$kosar->$dolgok = array("10" => 1);

// helyes, de lehetséges, hogy nem a megcélzott eredmény
// $kosar->$valtozo értelme $kosar->dolgok
$valtozo = 'dolgok;
$kosar->$valtozo = array("10" => 1);
```

Egy osztály definiálásakor nem tudhatod, milyen néven lesz majd elérhető az objektumod a PHP programban: a Kosar osztály készítése idején nem volt ismert, hogy később $kosar vagy $masik_kosar néven nevezzük-e majd az objektumpéldányt. Ezért nem írhatod a Kosar osztályban, hogy $kosar->dolgok. De hogy el tudjad érni az osztály saját metódusait és tulajdonságait az objektumpéldány(ok) nevétől függetlenül, használhatod a $this kvázi-változót, amit 'a sajátom' vagy 'az aktuális objektumpéldány' értelemben alkalmazhatsz. Ezért a '$this->dolgok[$sorsz] += $db' úgy olvasható, hogy 'adj $db darab $sorsz sorszámú terméket a saját dolgok tömbömhöz', vagy 'adj $db darab $sorsz sorszámú terméket az aktuális objektumpéldány dolgok tömbjéhez'.

> **Megjegyzés:** Van pár hasznos függvény az osztályok és objektumok kezeléséhez. Lásd az Osztály és objektum függvények című részt.

extends

Gyakori, hogy szeretnél olyan osztályokat kialakítani, amelyek egy már meglévő osztályhoz hasonló tulajdonságokkal és metódusokkal rendelkeznek. Tulajdonképpen jó gyakorlat egy általános osztályt definiálni, amit minden projektedben használhatsz, és ezt az osztályt alakítani az egyes projektek igényeinek megfelelően. Ennek a megvalósítása érdekében az osztályok lehetnek más osztályok kiterjesztései. A kiterjesztett, vagy származtatott osztály minden tulajdonággal és metódussal rendelkezik, ami a kiindulási osztályban megvolt (ezt nevezzük öröklésnek, bár senki sem hal meg a folyamat során). Amit hozzáadsz a kiindulási osztályhoz, azt nevezzük kiterjesztésnek. Nem lehetséges megcsonkítani egy osztályt, azaz megszüntetni egy metódust, vagy tulajdonságot. Egy leszármazott osztály mindig pontosan egy alaposztálytól függ, azaz egyidejűleg többszörös leszármaztatás nem támogatott. A kiterjesztés kulcsszava az 'extends'.

```
class Gazdas_Kosar extends Kosar
{
    var $tulaj;

    function tulajdonosa ($nev)
    {
        $this->tulaj = $nev;
    }
}
```

Ez definiál egy Gazdas_Kosar nevű osztályt, ami a Kosar összes változójával és metódusával rendelkezik, és van egy saját változója, a $tulaj, no meg egy saját metódusa, a tulajdonosa(). A gazdás kosarat a hagyományos módon hozhatod létre, és a kosár tulajdonosát is be tudod állítani, le tudod kérdezni [ebben az esetben favágó módszerrel]. A gazdás kosarakon továbbra is lehet használni a Kosar függvényeit:

```
$gkosar = new Gazdas_Kosar;  // Gazdás kosár létrehozása
$gkosar->tulajdonosa ("Namilesz Teosztasz"); // a tulaj beállítása
print $gkosar->tulaj;        // a tulajdonos neve
$gkosar->break ("10", 1);    // (Kosar-ból öröklött funkcionalitás)
```

Konstruktor

> **Figyelem**
>
> A PHP 3 és PHP 4 konstruktorok különbözőképpen működnek. A PHP 4 megvalósítása erősen javasolt.

A konstruktorok az osztályok olyan metódusai, amelyek automatikusan meghívásra kerülnek egy új objektumpéldány new kulcsszóval történő létrehozása során. A PHP 3-ban egy metódus akkor tekinthető konstruktornak, ha a neve megyezik az osztály nevével. A PHP 4-ben egy metódus akkor lesz kostruktorrá, hogy a neve megegyezik annak az osztálynak a nevével, ahol deifiniálták. A különbség hajszálnyi, de kritikus (lásd lentebb).

```
// A PHP 3 és PHP 4 verziókban is működik
class Auto_Kosar extends Kosar
{
    function Auto_Kosar ()
    {
        $this->berak ("10", 1);
    }
}
```

Ez egy olyan Auto_Kosar nevű osztályt [objektumtípust] hoz létre, mint a Kosar, csak rendelkezik egy konstruktorral, amely inicializálja a kosarat 1 darab "10"-es áruval, valahányszor a new operátorral hozzuk létre az objektumot. [de csak akkor!!!] A konstruktoroknak is lehet átadni paramétereket, és ezek lehetnek elhagyhatók is, amely még hasznosabbá teszi őket. Ha paraméterek nélkül is használható osztályt szeretnél, állíts be minden paraméternek alapértéket.

```
// A PHP 3 és PHP 4 verziókban is működik
class Konstruktoros_Kosar extends Kosar
{
    function Konstruktoros_Kosar ($sorsz = "10", $db = 1)
    {
        $this->berak ($sorsz, $db);
```

```
    }
}

// Mindig ugyanazt az uncsi dolgot veszi...

$kiindulo_kosar = new Konstruktoros_Kosar;

// Igazi vásárlás

$masik_kosar = new Konstruktoros_kosar ("20", 17);
```

Figyelem

A PHP 3-ban a leszármazott osztályokra és konstrukrotokra számos korlátozás van. Az alábbi példákat érdemes alaposan áttekinteni, hogy megértsd ezeket a korlátozásokat.

```
class A
{
    function A()
    {
        echo "Én vagyok az A osztály konstruktora.<br>\n";
    }
}

class B extends A
{
    function C()
    {
        echo "Én egy metódus vagyok.<br>\n";
    }
}

// PHP 3-ban semmilyen konstruktor sem kerül meghívásra
$b = new B;
```

PHP 3-ban semmilyen konstruktor sem kerül meghívásra a fenti példában. A PHP 3 szabálya a következő: ’A konstruktor egy metódus, aminek ugyanaz a neve, mint az osztálynak’. Az osztály neve B, és nincs B() nevű metódus a B osztályban. Semmi sem történik.

Ez a PHP 4-ben ki van javítva egy másik szabály bevezetésével: Ha az osztályban nincs konstruktor, a szülő osztály konstruktora hívódik meg, ha létezik. A fenti példa kimenete ’Én vagyok az A osztály konstruktora.
’ lett volna PHP 4-ben.

```
class A
{
    function A()
    {
        echo "Én vagyok az A osztály konstruktora.<br>\n";
```

```
    }

    function B()
    {
        echo "Én egy B nevű metódus vagyok az A osztályban.<br>\n";
        echo "Nem vagyok A konstruktora.<br>\n";
    }
}

class B extends A
{
    function C()
    {
        echo "Én egy metódus vagyok.<br>\n";
    }
}

// Ez meghívja B()-t, mint konstruktort
$b = new B;
```

A PHP 3-ban az A osztály B() metódusa hirtelen konstruktorrá válik a B osztályban, habár ez soha sem volt cél. A PHP 3 szabálya: ’A konstruktor egy metódus, aminek ugyanaz a neve, mint az osztálynak’. A PHP 3 nem foglalkozik azzal, hogy a metódus a B osztályban van-e definiálva, vagy öröklés útján áll rendelkezésre.

Ez a PHP 4-ben ki van javítva egy másik szabály bevezetésével: ’A konstruktor egy metódus, aminek ugyanaz a neve, mint az osztálynak, ahol deifiniálták’. Ezért a PHP 4-ben a B osztálynak nincs saját konstruktora, így a szülő osztály kostruktora hívódik meg, kiírva, hogy ’Én vagyok az A osztály konstruktora.
’.

Figyelem

Sem a PHP 3, sem a PHP 4 nem hívja meg a szülő osztály konstruktorát automatikusan egy leszármazott osztály definiált konstruktorából. A te feladatod, hogy meghívd a szülő konstruktorát, ha szükséges.

Megjegyzés: Nem léteznek destruktorok sem a PHP 3 sem a PHP 4 verzióiban. Bár használhatod a register_shutdown_function() függvényt a destruktorok legtöbb viselkedésének eléréséhez.

A destruktor olyan metódus, ami automatikusan meghívódik, amikor egy objektum megszűnik, akár az unset() meghívásával, akár a környezete megszűnése miatt. PHP-ben nincsenek destruktorok.

: :

Figyelem

Az alábbiak csak PHP 4-ben érvényesek.

Időnként hasznos az ősosztályok metódusaira vagy tulajdonságaira hivatkozni, vagy olyan osztálymetódusokat meghívni, amelyek nem példányosított objektumokhoz tartoznak. A :: operátor erre használható.

```
class A
{
    function pelda()
    {
        echo "Én az eredeti A::pelda() metódus vagyok.<br>\n";
    }
}

class B extends A
{
    function pelda()
    {
        echo "Én a felüldefiniáló B::pelda() metódus vagyok.<br>\n";
        A::example();
    }
}

// nincs semmilyen objektum az A osztályból
// ez azonban ki fogja írni:
//   Én az eredeti A::pelda() metódus vagyok.<br>
A::pelda();

// B egy objektuát hozzuk létre
$b = new B;

// ez ki fogja írni:
//   Én a felüldefiniáló B::pelda() metódus vagyok.<br>
//   Én az eredeti A::pelda() metódus vagyok.<br>
$b->pelda();
```

A fenti példa meghívja az A osztály pelda() metódusát, habár nincs konkrét példányunk az A osztályból, tehát ezt nem írhatnánk le az `$a->pelda()`-hoz hasonlóan. Ehelyett a pelda() egy ’osztálymetódusként’ viselkedik, azaz az osztály egy függvényeként, és nem egy példány metódusaként.

Osztálymetódusok léteznek, de osztálytulajdonságok (változók) nem. Mivel a hívás pillanatában semmilyen objektum nem létezik, egy osztálymetódus nem használhat objektum változókat, és egyáltalán nem használhatja a $this speciális referenciát. Egy objektummetódus azonban természetesen dolgozhat globális változókkal és lokális változókkal is.

A fenti példa a B osztályban felüldefiniálja a pelda() metódust. Az A osztálytól örökölt eredeti definíció eltűnik, és többé nem érhető el, hacsak nem az A osztályban megvalósított pelda() függvényre hivatkozol közvetlenül, a :: operátor segítségével. Ennek eléréséhez A::pelda()-t kell használni (ebben az esetben írhatnál parent::pelda()-t is, ahogy az a következő szakaszban olvasható).

Ebben a környezetben van aktuálisan használt objektum, és ennek lehetnek objektum változói (tulajdonságai). Ekképpen ha egy objektummetóduson belül használod ezt az operátort, akkor alkalmazhatod a $this-t, és felhasználhatod az objektum tulajdonságait.

parent

Gyakran van szükség arra, hogy a szülő tulajdonságaira vagy metódusaira hivatkozzunk leszármazott osztályokban. Ez különösen igaz, ha a leszármazott osztály egy finomítása, vagy specializálása az alaposztálynak.

Ahelyett, hogy a szülő osztály nevét megadd minden ilyen meghíváskor (mint a hogy a :: operátor példája mutatta), használhatod a `parent` speciális nevet, ami tulajdonképpen a szülő osztály nevét jelenti, amit az `extends` kulcsszónál megadtál. Ennek a speciális névnek a használatával elkerülöd a szülő osztály nevének ismétlődését. Ha a megvalósítás során a leszármazási fát meg kell változtatni, csak egy helyen, az `extends` kulcsszónál kell átírnod a nevet.

```
class A
{
    function pelda()
    {
        echo "Én A::pelda() vagyok egyszerű funkcióval.<br>\n";
    }
}

class B extends A
{
    function pelda()
    {
        echo "Én B::pelda() vagyok több funkcióval.<br>\n";
        parent::pelda();
    }
}

$b = new B;

// Ez a B::pelda() metódust hívja, ami az A::pelda()-t hívja
$b->pelda();
```

Objektumok szerializációja, objektumok session-ökben

Megjegyzés: A PHP 3-ban az objektumok elveszítik az osztály-hozzárendelésüket a szerializációs, és deszerializációs folyamat során. Az eredmény objektum típusú, de nem tartozik semelyik osztályhoz, és nincs egy metódusa sem, tehát eléggé használhatatlan (csupán egy tömb, furcsa szintakszissal).

Figyelem

A következő információk csak a PHP 4-es változatára érvényesek.

A serialize() egy karaktersorozatot ad vissza, ami az átadott érték byte-sorozatban megadott megfelelője. Az unserialize() visszaalakít egy ilyen karaktersorozatot az eredeti értékké. A szerializációs folyamat során egy objektum átadásával elmenthetjük az objektum minden tulajdonságát (változóját). A függvények nem kerülnek elmentésre, csak az osztály neve.

Ahhoz, hogy az unserialize() segítségével vissza lehessen állítani egy objektumot, az objektum osztályának (típusának) már definiálva kell lennie. Ez a következőket jelenti egy példán keresztül megvilágítva. Ha az elso.php oldalon az A osztályú $a objektumot szerializálod, akkor kapsz egy olyan karaktersorozatot, amely az A osztályra hivatkozik, és tartalmazza az összes $a-ban lévő változó (tulajdonság) értékét. Ha ezt a karaktersorozatot a masodik.php oldalon objektummá szeretnéd alakítani, újra létrehozva az A osztályú $a nevű objektumot, akkor az A osztály definíciójának rendelkezésre kell állnia a masodik.php oldalon is. Ez úgy érhető el, hogy az A osztály definícióját egy külső állományban tárolod, és ezt alkalmazod mind az elso.php, mind a masodik.php oldalon.

```
aosztaly.inc:
  class A
  {
      var $egy = 1;

      function egyet_mutat()
      {
          echo $this->egy;
      }
  }

elso.php:
  include("aosztaly.inc");

  $a = new A;
  $s = serialize($a);
  // tároljuk az $s-t valahol, ahol masodik.php megtalálja
  $fp = fopen("tarolas", "w");
  fputs($fp, $s);
  fclose($fp);

masodik.php:
  // ez szükséges, hogy a deszerializáció rendben menjen
  include("aosztaly.inc");

  $s = implode("", @file("tarolas"));
  $a = unserialize($s);

  // most már használható az egyet_mutat() metódus
  $a->egyet_mutat();
```

Ha session-öket alkalmazol, és a session_register() függvénnyel regisztálsz objektumokat, ezek az objektumok automatikusan szerializálódnak minden PHP program futása után, és deszerializálódnak minden további programban. Ez egyszerűen azt jelenti, hogy ezek az objektumok akármelyik oldalon feltűnhetnek, miután a session részévé váltak.

Erősen javasolt, hogy minden regisztrált objektum osztály definícióját betöltsd minden oldalon, még akkor is, ha éppen nem használod azokat. Ha ezt nem teszed meg, és egy objektum úgy

deszerializálódik, hogy nem áll rendelkezésre az osztály definícója, el fogja veszteni az osztály hozzárendelését, és az `stdClass` osztály egy példánya lesz, metódusok nélkül, így használhatatlanná válik.

Ezért ha a fenti példában az $a a session részévé válik a `session_register("a")` meghívásával, akkor be kell töltened az `aosztaly.inc` külső állományt minden oldalon, nem csak az elso.php és masodik.php programokban.

A speciális `__sleep` és `__wakeup` metódusok

A serialize() ellenőrzi, hogy van-e az osztályodnak `__sleep` nevű metódusa. Ha van, ez lefut a szerializáció előtt. Ez megtisztíthatja az objektumot, és végül egy tömbbel tér vissza, amely tartalmazza az adott objektum ténylegesen szerializálandó tulajdonságainak neveit.

A `__sleep` célja, hogy bezárjon minden adatbázis kapcsolatot, a várakozó adatokat lementse, és hasonló 'tisztító' jellegű tevékenységeket végezzen. Hasznos lehet akkor is, ha nagyon nagy objektumaid vannak, amelyeket külön szeretnél lementeni.

Ezzel szemben az unserialize() a speciális `__wakeup` nevű függvényt használja. Ha ez létezik, ez a függvény alkalmazható arra, hogy visszaállítsa az objektum erőforrásait.

A `__wakeup` célja lehet például, hogy visszaállítson egy adatbázis kapcsolatot, ami a szerializáció során elveszett, és hasonló beállítási feladatokat végezzen.

Referenciák a konstruktorban

Referenciák képzése kontruktorokban problémás helyzetekhez vezethet. Ez a leírás segít a bajok elkerülésében.

```
class Ize
{
    function Ize($nev)
    {
        // egy referencia létrehozása a globális $globalref változóban
        global $globalref;
        $globalref[] = &$this;
        // a név beállítása a kapott értékre
        $this->nevBeallitas($nev);
        // és kiírás
        $this->nevKiiras();
    }

    function nevKiiras()
    {
        echo "<br>",$this->nev;
    }

    function nevBeallitas($nev)
    {
        $this->nev = $nev;
    }
}
```

Nézzük, hogy van-e különbség az `$obj1` és az `$obj2` objektum között. Az előbbi a = másoló operátorral készült, az utóbbi a =& referencia operátorral készült.

```
$obj1 = new Ize('konstruktorban beállított');
$obj1->nevKiiras();
$globalref[0]->nevKiiras();

/* kimenete:
konstruktorban beállított
konstruktorban beállított
konstruktorban beállított */

$obj2 =& new Ize('konstruktorban beállított');
$obj2->nevKiiras();
$globalref[1]->nevKiiras();

/* kimenete:
konstruktorban beállított
konstruktorban beállított
konstruktorban beállított */
```

Szemmel láthatóan nincs semmi különbség, de valójában egy nagyon fontos különbség van a két forma között: az `$obj1` és `$globalref[0]` _NEM_ referenciák, NEM ugyanaz a két változó. Ez azért történhet így, mert a "new" alapvetően nem referenciával tér vissza, hanem egy másolatot ad.

> **Megjegyzés:** Nincsenek teljesítménybeli problémák a másolatok visszaadásakor, mivel a PHP 4 és újabb verziók referencia számlálást alkalmaznak. Legtöbbször ellenben jobb másolatokkal dolgozni referenciák helyett, mivel a referenciák képzése eltart egy kis ideig, de a másolatok képzése gyakorlatilag nem igényel időt. Ha egyik sem egy nagy tömb, vagy objektum, és a változásokat nem szeretnéd mindegyik példányban egyszerre látni, akkor másolatok használatával jobban jársz.

Hogy bebizonyítsuk, amit fent írtunk, lásd az alábbi kódot:

```
// Most megváltoztatjuk a nevet. Mit várs?
// Számíthatsz arra, hogy mind $obj1 és $globalref[0] megváltozik...
$obj1->nevBeallitas('kívülről beállítva');

// mint korábban írtuk, nem ez a helyzet
$obj1->nevKiiras();
$globalref[0]->nevKiiras();

/* kimenet:
konstruktorban beállított
kívülről beállítva */
```

```
// lássuk mi a különbség az $obj2 és $globalref[1] esetén
$obj2->nevBeallitas('kívülről beállítva');

// szerencsére ezek nem csak egyenlőek, hanem éppen ugyan az
// a két változó, így $obj2->nev és $globalref[1]->nev ugyanaz
$obj2->nevKiiras();
$globalref[1]->nevKiiras();

/* kimenete:
kívülről beállítva
kívülről beállítva */
```

Végül még egy utolsó példa, próbáld meg megérteni.

```
class A
{
    function A($i)
    {
        $this->ertek = $i;
        // próbáld meg kitalálni, miért nem kell itt referencia
        $this->b = new B($this);
    }

    function refLetrehozas()
    {
        $this->c = new B($this);
    }

    function ertekKiiras()
    {
        echo "<br>",get_class($this),' osztály: ',$this->value;
    }
}


class B
{
    function B(&$a)
    {
        $this->a = &$a;
    }

    function ertekKiiras()
    {
        echo "<br>",get_class($this),' osztály: ',$this->a->value;
    }
}

// próbáld meg megérteni, hogy egy egyszerű másolás
// miért okoz nem várt eredményeket a *-al jelölt sorban
$a =& new A(10);
$a->refLetrehozas();
```

```
$a->ertekKiiras();
$a->b->ertekKiiras();
$a->c->ertekKiiras();

$a->ertek = 11;

$a->ertekKiiras();
$a->b->ertekKiiras(); // *
$a->c->ertekKiiras();

/*
kimenete:
A osztály: 10
B osztály: 10
B osztály: 10
A osztály: 11
B osztály: 11
B osztály: 11
*/
```

Fejezet 14. Referenciák

Mik a referenciák

A referenciák lehetőséget adnak PHP-ben azonos változó tartalom elérésére különböző nevek alatt. Ezek szimbólumtábla bejegyzések, nem olyanok, mint a C nyelv mutatói. PHP-ben a változók neve és tartalma két különböző dolog, tehát ugyanaz a tartalom kaphat különböző neveket. A legjobb hasonlat talán a UNIX állománynevek és állományok rendszere. A változóneveket könyvtár bejegyzésekként foghatod fel, a változók tartalmát állományokként. A referenciák olyanok, mint UNIXban a hardlinkek.

Mit lehet referenciákkal tenni

A PHP referenciák lehetőséget adnak arra, hogy egy értékhez két nevet lehessen rendelni. Ez azt jelenti, hogy a következő programban:

```
$a =& $b;
```

az `$a` és `$b` nevek ugyanarra az értékre hivatkoznak.

> **Megjegyzés:** Az `$a` és a `$b` nevek teljesen egyenrangúak. Nem arról van szó, hogy az `$a` a `$b`-re mutat, vagy fordítva, hanem arról, hogy az `$a` és a `$b` név ugyanannak az értéknek két elnevezése.

Ugyanez a forma használható az olyan függvényeknél, amelyek referenciát adnak vissza, vagy a `new` operátor használatakor (a PHP 4.0.4 és későbbi verziókban):

```
$obj =& new valamilyen_osztaly();
$ize =& valtozo_kereses ($valami);
```

> **Megjegyzés:** Ha nem használod az `&` -t, akkor az osztálypéldány másoláta adódik át. A `$this` objektumon belüli használatával ugyanazon az objektumpéldányon dolgozol. Ha az értékadás során az `&` -t elhagyod, akkor az objektumról másolat készül és a `$this` már ezen a másolaton fog dolgozni. Van, amikor ez nem kívánatos, mivel általában egy példányon szeretnénk dolgozni a jobb memóriahasználat és teljesítmény érdekében.

A a referenciákat paraméterátadáskor is lehet használni. Ebben az esetben a meghívott függvény egy lokális változója és a hívó környezet egy változója ugyanazt az értéket fogja képviselni. Például:

```
function ize (&$valtozo)
```

```
{
    $valtozo++;
}

$a = 5;
ize ($a);
```

Ez a kód az `$a` változó értékét 6-ra állítja. Ez azért történik meg, mivel az `ize` függvényben a `$valtozo` egy referencia a `$a` változó értékére. A részletes leírást a referenciakénti paraméterátadás c. fejezetben olvashatod.

A referenciák harmadik felhasználási módja a refencia visszatérési-érték.

Mit nem lehet referenciákkal tenni

Mint korábban írtuk, a referenciák nem mutatók. A következő konstrukció ezért nem a vártnak megfelelően viselkedik:

```
function ize (&$valtozo)
{
    $valtozo =& $GLOBALS["valami"];
}
ize($valami);
```

A `foo` függvényben a `$valtozo` változó a `$valami` értékéhez lesz kötve, de utána ezt megváltoztatjuk a `$GLOBALS["valami"]` értékére. Nincs lehetőség a referenciák segítségével a `$valami` más értékhez kötésére a hívó környezetben, mivel a `$valami` nem áll rendelkezésre az `ize` függvényben. Ott a `$valtozo` reprezentálja az értékét, amely csak változó tartalommal bír és nem név-érték kötéssel a hívó szimbólumtáblájában.

Referenciakénti paraméterátadás

A függvényeknek változókat referenciaként is át lehet adni, így a függvény tudja módosítani a hívó környezetben definiált értéket. Ez a következőképpen oldható meg:

```
function ize (&$valtozo)
{
    $valtozo++;
}

$a = 5;
ize ($a);
// $a itt 6
```

Figyeld meg, hogy nincs referencia jelzés a függvényhíváskor, csak a függvény definíciójában. Ez önmagában elég a megfelelő működéshez.

A következők szerepelhetnek referenciakénti paraméterátadásban:

- Változó, például `ize($a)`
- New utasítás, például `ize(new osztaly())`
- Egy függvény által visszaadott referencia, például:

```
function &valami()
{
    $a = 5;
    return $a;
}
ize(valami());
```

Lásd még a refencia visszatérési-érték leírását.

Minden más kifejezést kerülni kell referencia szerinti paraméterátadáskor, mivel az eredmény határozatlan lesz. A következő példákban a referencia szerinti paraméterátadás hibának minősül:

```
function valami() // Figyeld meg, nincs & jel!
{
    $a = 5;
    return $a;
}
ize(valami());

ize($a = 5) // Kifejezés, nem változó
ize(5) // Konstans, nem változó
```

Ezek a meghatározások a PHP 4.0.4 és későbbi verzióira érvényesek.

Refencia visszatérési-érték

A refencia visszatérési-érték pl. olyan változók megtalálásakor lehet hasznos, amelyekről referenciát kell készíteni. Ha referenciát kell visszaadni visszatérési értékként, akkor használd az alábbi formát:

```
function &valtozo_kereses ($param)
{
    ...kód...
    return $megtalalt_valtozo;
}
```

```
$ize =& valtozo_kereses ($valami);
$ize->x = 2;
```

Ebben a példában a `valtozo_kereses` egy objektumot keres meg, és a megtalált objektum egy tulajdonságát állítjuk át - helyesen. A referenciák használata nélkül a másolatának egy tulajdonságán tettük volna mindezt - hibásan.

> **Megjegyzés:** A paraméter átadással ellentétben, itt a `&` jelet mindkét helyen meg kell adnod a referenciavisszaadás jelöléséhez. Így nem egy másolatot kapsz, és az `$ize` változóra nézve referencia hozzárendelés történik, nem pedig érték hozzárendelés (értékmásolás).

Referenciák megszüntetése

Amikor megszüntetsz egy referenciát, csak megszakítod a változónév és az érték közötti kapcsolatot. Ez nem azt jelenti, hogy a váltózó értékét törlöd. Például:

```
$a = 1;
$b =& $a;
unset ($a);
```

nem fogja megszüntetni a `$b` nevet, csak az `$a` nevet, így az érték a `$b` néven továbbra is elérhető.

Ismét érdemes a Unix **unlink** parancsával és az állományrendszerrel való hasonlatosságra gondolni.

A PHP által használt referenciák

Sok konstrukció a PHP-ben referenciák segítségével valósul meg, azért minden fentebb tárgyalt kérdés ezekre az elemekre is igaz. Néhány olyan konstrukciót, mint a referencia átadást vagy visszatérést már kifejtettünk, más referenciákat használó konstrukciók:

`global` referenciák

Amikor egy változót a **global $valtozo** formával globálisként használsz, tulajdonképpen egy referenciát képzel a megfelelő globális változóra,azaz a következő kódnak megfelelő történik:

```
$valtozo =& $GLOBALS["valtozo"];
```

Ez például azt is jelenti, hogy a `$valtozo` törlése nem fogja törölni a globális változót.

`$this`

Egy objektum metódusban a `$this` mindig az aktuális példányra egy referencia.

Rész III. Szolgáltatások

Fejezet 15. Hibakezelés

A hibáknak és figyelmeztetéseknek PHP-ben számos típusa van. Ezek:

Táblázat 15-1. PHP hiba típusok

Érték	Szimbólum	Leírás	Megjegyzés
1	E_ERROR	fatális futás-idejű hibák	
2	E_WARNING	nem fatális futás-idejű hibák	
4	E_PARSE	fordítás-idejű feldolgozási hibák	
8	E_NOTICE	futás-idejű figyelmeztetések (a notice a warning-nál gyengébb)	
16	E_CORE_ERROR	fatális hibák, amik a PHP elindulásakor lépnek fel	csak a PHP 4-ben
32	E_CORE_WARNING	nem fatális hibák figyelmeztetései (warning), amik a PHP elindulásakor lépnek fel	csak a PHP 4-ben
64	E_COMPILE_ERROR	fatális fordítás-idejű hibák	csak a PHP 4-ben
128	E_COMPILE_WARNING	nem fatális fordítás-idejű figyelmeztetések (warning)	csak a PHP 4-ben
256	E_USER_ERROR	felhasználó által generált hibaüzenetek	csak a PHP 4-ben
512	E_USER_WARNING	felhasználó által generált figyelmeztetések (warning)	csak a PHP 4-ben
1024	E_USER_NOTICE	felhasználó által generált figyelmeztetések (notice)	csak a PHP 4-ben
	E_ALL	az összes fent felsorolt elem	csak a PHP 4-ben

A fenti értékek (akár a numerikusak, akár a szimbolikusak) arra használhatóak, hogy felépíts egy bitmask-ot, ami megadja, hogy mely hibákat kell jeleznie a PHP-nek. Használhatod bitszintű operátorokat, hogy összeállítsd a fenti elemekből a neked megfelelő értéket, vagy letilts egyes hibákat. Csak a '|', '~', '!', és '&' operátorok használhatóak `php.ini` fájlban, és semmilyen operátor sem használható a `php3.ini` fájlban.

PHP 4-ben az alapbeállítású error_reporting érték `E_ALL & ~E_NOTICE`, ami azt jelenti, hogy minden hiba és figyelmeztetés megjelenik az E_NOTICE-szint kivételével. PHP 3-ban az alapbeállítás `(E_ERROR | E_WARNING | E_PARSE)`, ugyanezt jelenti. Vedd figyelembe, hogy ezek a konstansok nem támogatottak a PHP 3 `php3.ini` fájljában, ezért az error_reporting beállítás a numerikus `7` érték.

Ezek a beállítások az ini fájl error_reporting direktívájával változtathatóak meg, vagy az Apache

`httpd.conf` fájlban a php_error_reporting (php3_error_reporting PHP 3 esetén) direktívával vagy végül futásidőben egy szkriptben az error_reporting() függvénnyel.

> **Figyelem**
>
> Ha a kódod vagy a szervered frissíted PHP 3-ról PHP 4-re, jól teszed, ha ellenőrzöd ezeket a beállításokat és az error_reporting() függvényhívásokat, különben akaratlanul kikapcsolod az új hibatípusokat, különösen az E_COMPILE_ERROR-t. Ez üres dokumentumokhoz vezethet, amik nem tartalmaznak semmilyen utalást arra, hogy mi történt, vagy hogy hol kellene keresni a problémát...

Minden PHP kifejezés írható a "@" előtaggal, ami kikapcsolja a hibajelentést arra a kifejezésre. Ha hiba lép fel a kifejezés kiértékelésekor, és a track_errors szolgáltatás be van kapcsolva, a hibaüzenet megtalálható a `$php_errormsg` globális változóban.

> **Megjegyzés:** A @ hibakezelő operátor nem kapcsolja ki a szkriptek feldolgozása során előforduló hibák (parse error) jelentését.

> **Figyelem**
>
> Jelenleg a "@" hibakezelő operátor kikapcsolja azon kritikus hibák jelentését is, amik megállítják a szkript futását. Más problémák mellett, ha egy függvényből érkező hibaüzenetek elnyelésére használod a "@" jelet, meg fog állni a szkript futása, ha nem létezik a megadott függvény, vagy elírtad a nevét.

Az alábbiakban láthatsz egy példát a PHP hibakezelő képességeire. Definiálunk egy hibakezelő függvényt, ami tárolja a hibákat egy fájlba (XML formátummal) és email-t küld a fejlesztőnek ha a programban kritikus hiba történik.

Példa 15-1. Hibakezelés használata egy szkriptben

```
<?php
// saját hibakezelést építünk
error_reporting(0);

// felhasználó által definiált hibakezelő függvény
function sajatHibaKezelo ($hibaszam, $hibauzenet, $filenev, $sorszam, $val-
tozok) {

    // időbélyeg a hibához
    $ido = date("Y-m-d H:i:s (T)");

    // Asszociatív tömb definiálása a hibaszövegeknek.
    // Valójában csak a 2,8,256,512 és 1024 elemeket
    // vesszük figyelembe
    $hibatipus = array (
                1   =>  "Error",
                2   =>  "Warning",
                4   =>  "Parsing Error",
                8   =>  "Notice",
                16  =>  "Core Error",
```

```
                32  =>  "Core Warning",
                64  =>  "Compile Error",
                128 =>  "Compile Warning",
                256 =>  "User Error",
                512 =>  "User Warning",
                1024=>  "User Notice"
                );

    // azok a hibatípusok, amikre a változókat is el kell menteni
    $user_hibak = array(E_USER_ERROR, E_USER_WARNING, E_USER_NOTICE);

    $hiba = "<errorentry>\n";
    $hiba .= "\t<datetime>".$ido."</datetime>\n";
    $hiba .= "\t<errornum>".$hibaszam."</errornum>\n";
    $hiba .= "\t<errortype>".$hibatipus[$hibaszam]."</errortype>\n";
    $hiba .= "\t<errormsg>".$hibauzenet."</errormsg>\n";
    $hiba .= "\t<scriptname>".$filenev."</scriptname>\n";
    $hiba .= "\t<scriptlinenum>".$sorszam."</scriptlinenum>\n";

    if (in_array($hibaszam, $user_hibak))
        $hiba .= "\t<vartrace>".wddx_serialize_value($valtozok,"Variables")."</vartrace>\
    $hiba .= "</errorentry>\n\n";

    // teszteléshez
    // echo $hiba;

    // a hibanapló elmentése, email küldés ha kritikus hiba van
    error_log($hiba, 3, "/usr/local/php4/error.log");
    if ($hibaszam == E_USER_ERROR)
        mail("phpdev@example.com","Kritikus programhiba",$hiba);
}


function tavolsag ($vektor1, $vektor2) {
    if (!is_array($vektor1) || !is_array($vektor2)) {
        trigger_error("Helytelen parameterek, tomboket varok", E_USER_ERROR);
        return NULL;
    }

    if (count($vektor1) != count($vektor2)) {
        trigger_error("A vektorok ugyanolyan dimenziojuak legyenek", E_USER_ERROR);
        return NULL;
    }

    for ($i=0; $i<count($vektor1); $i++) {
        $c1 = $vektor1[$i]; $c2 = $vektor2[$i];
        $d = 0.0;
        if (!is_numeric($c1)) {
            trigger_error("Az elso vektor $i koordinataja nem szam, nullaval szamolok",
                          E_USER_WARNING);
            $c1 = 0.0;
        }
        if (!is_numeric($c2)) {
            trigger_error("A masodik vektor $i koordinataja nem szam, nullaval szamolok",
                          E_USER_WARNING);
            $c2 = 0.0;
        }
```

```
        $d += $c2*$c2 - $c1*$c1;
    }
    return sqrt($d);
}

$regi_hiba_kezelo = set_error_handler("sajatHibaKezelo");

// nem definiált konstans, warning-ot generál
$t = NEM_VAGYOK_DEFINIALVA;

// néhány "vektor" definiálása
$a = array (2,3, "ize");
$b = array (5.5, 4.3, -1.6);
$c = array (1, -3);

// user hiba generálása
$t1 = tavolsag($c, $b)."\n";

// újabb user hiba generálása
$t2 = tavolsag($b, "ez nem tömb")."\n";

// warning generálása
$t3 = tavolsag($a, $b)."\n";
?>
```

Ez csak egy egyszerű példa, ami bemutatja, hogy hogyan kell használni a Hibakezelő és naplózó függvényeket.

Lásd még: error_reporting(), error_log(), set_error_handler(), restore_error_handler(), trigger_error(), és user_error().

Fejezet 16. Képek készítése

A PHP képességei nem korlátozódnak kizárólag HTML kimenet előállítására. Alkalmas GIF/PNG/JPEG képek készítésére is, akár szerver oldali GIF animációk (stream) megvalósítására. A PHP-t a GD könyvtári kiterjesztésekkel kell fordítanod, hogy ez a lehetőség rendelkezésre álljon. A GD és PHP a esetleg további könyvtárakat igényelhet attól függően, hogy milyen képformátumokkal szeretnél dolgozni. A GD nem támogatja a GIF formátumot az 1.6-os és későbbi verziókban.

Példa 16-1. PNG kép készítése PHP-vel

```
<?php
    Header("Content-type: image/png");
    $szoveg = implode($argv, " ");
    $kep = ImageCreateFromPng("kepek/gomb1.png");
    $narancs = ImageColorAllocate($kep, 220, 210, 60);
    $px = (imagesx($kep)-7.5*strlen($szoveg))/2;
    ImageString($kep,3,$px,9,$szoveg,$narancs);
    ImagePng($kep);
    ImageDestroy($kep);
?>
```

Ez a példa egy oldalon az <img src="gomb.php?felirat"> HTML taggel hívható meg. A fenti gomb.php szkript veszi a "felirat" szöveget, és egy létező képre írja azt (ez esetben a kép a "kepek/gomb1.png"), majd végül elküldi a böngészőnek a kész képet. Ez egy igen kellemes módszere lehet annak, hogy újabb és újabb gombokat állíts elő úgy, hogy ne kelljen újrarajzolni minden egyes alkalommal egy gombot, ha a feliratát meg szeretnéd változtatni. Így minden gomb dinamikusan kerül előállításra.

Fejezet 17. HTTP hitelesítés PHP-vel

A HTTP hitelesítési (authentication) funkciók csak akkor elérhetőek PHP-ben, ha az Apache modulként fut, bár régebben a CGI módú használat során is működött ez a funkció, azonban ez a lehetőség már megszűnt. Az Apache modulként futó PHP esetében a header() függvényt kell használni arra, hogy egy "Authentication Required" üzenetet küldjön a kliens böngészőnek, aminek hatására az egy Username/Password bemeneti ablakot nyit meg a felhasználó számára. Ha a látogató kitöltötte a username és password mezőket, az URL, ami a PHP szkriptre mutat, ismét meghívásra kerül, és rendelkezésre állnak a $PHP_AUTH_USER, $PHP_AUTH_PW és a $PHP_AUTH_TYPE változók, amik a felhasználói név, jelszó és azonosítási típus értékeket tartalmazzák értelemszerűen. Jelen pillanatban csupán a "Basic" azonosítási típus támogatott. Lásd még a header() függényvt.

Egy egyszerű példa PHP szkript, ami kliens azonosítást vált ki:

Példa 17-1. HTTP azonosítási példa

```
<?php
  if(!isset($PHP_AUTH_USER)) {
    header("WWW-Authenticate: Basic realm=\"Azonosítás indoka\"");
    header("HTTP/1.0 401 Unauthorized");
    echo "Ez jelenik meg, ha a Cancel gombot nyomja a felhasználó\n";
    exit;
  } else {
    echo "<p>Helló $PHP_AUTH_USER.</p>";
    echo "<p>A megadott jelszavad: $PHP_AUTH_PW.</p>";
  }
?>
```

> **Megjegyzés:** Vigyázz a HTTP fejlécek írásakor! A maximális kompatibilitás eléréséhez a "Basic" kulcsszót nagy B betűvel kezdd, a "realm" részt mindenképpen tedd idézőjelbe (és nem aposztrófok közé)! Végül pontosan egy szóközt hagyj ki a "401" előtt a "HTTP/1.0 401" fejléc sorban.

Egy valós esetben persze nem a $PHP_AUTH_USER és $PHP_AUTH_PW kiírása az elérni kívánt cél, így általában a usernév és jelszó ellenőrzése következik. Természetesen lehetőség van ezt egy adatbázis lekérdezéssel megoldani, vagy egy dbm fájlban utánanézni a szükséges adatoknak.

Figyelj a hibás Internet Explorer böngészőkre, amik nem fogadják el tetszőleges sorrendben a HTTP fejléceket. A tesztek azt mutatják, hogy a *WWW-Authenticate* elküldése a `HTTP/1.0 401` előtt megoldja a problémát.

Mivel a hagyományos HTTP azonosítás során a jelszó rejtett az elért szkript előtt, a PHP nem állítja be a PHP_AUTH változókat ha az adott file-ra a hagyományos azonosítás is engedélyezett, és így nem deríthető ki a user jelszava. Ebben az esetben a $REMOTE_USER változó tartalmazza a már azonosított felhasználó nevét.

> **Beállítási megjegyzés:** A PHP az `AuthType` direktíva megléte alapján dönti el, hogy rajta kívülálló azonosítás történik-e. Figyelj ennek a direktívának az elkerülésére abban a környezetben, ahol a PHP-t szeretnéd használni azonosítások kezelésére (különben minden azonosítási próbálkozás sikertelen lesz).

Vedd észre, hogy ez nem küszöböli ki azt a problémát, hogy más nem azonosítás-köteles URL címeken lévő szkriptek ugyanazon a szerveren megszerezzék a jelszavakat az azonosított URL-ekről.

A Netscape Navigator és Internet Explorer böngészők törölni fogják a böngésző adott oldalhoz tartozó azonosítási tárát (authentication cache), amennyiben egy 401-es szerver üzenetet kapnak. Ez gyakorlatilag kilépteti a user-t, ami azt jelenti, hogy legközelebb ismét meg kell adnia a nevét és jelszavát. Időnként ezt arra használják, hogy lejárati időt rendelve a belépésekhez egy idő után megszüntessék azokat, vagy egy kilépés gombot biztosítsanak.

Példa 17-2. HTTP azonosítási példa, ami új nevet és jelszót kér

```
<?php
  function azonositas() {
   header( "WWW-Authenticate: Basic realm=\"Azonosítási Rendszer Teszt\"");
   header( "HTTP/1.0 401 Unauthorized");
   echo "Érvényes nevet és jelszót kell megadnod, hogy elérd ezt a szolgáltatást!\n";
   exit;
  }

  if(!isset($PHP_AUTH_USER) || ($JartMarItt == 1 && !strcmp($RegiUser, $PHP_AUTH_USER)))
   azonositas();
  }
  else {
   echo "<p>Helló: $PHP_AUTH_USER<br>";
   echo "Régebben: $RegiUser";
   echo "<form action=\"$PHP_SELF\" method=\"post\">\n";
   echo "<input type=\"hidden\" name=\"JartMarItt\" value=\"1\">\n";
   echo "<input type=\"hidden\" name=\"RegiUser\" value=\"$PHP_AUTH_USER\">\n";
   echo "<input type=\"submit\" value=\"Újraazonosítás\">\n";
   echo "</form></p>\n";
  }
?>
```

A HTTP Basic azonosítási standard nem követeli meg ezt a viselkedést a böngészők részéről, tehát ne építs rá! Lynx-el végzett tesztek azt mutatták, hogy a Lynx nem törli az azonosítási bizonyítványokat a 401-es szerver válasz hatására, tehát egy back és forward lépéssel ismét megnyílik az oldal, feltéve, hogy az azonosítási feltételek nem változtak. Ellenben a felhasználó megnyomhatja a ’_’ billentyűt, hogy törölje az azonosítási információkat.

Szintén fontos megjegyezni, hogy ez a módszer nem vezet eredményre, ha Microsoft IIS szervert használsz CGI módú PHP-val, az IIS korlátozásai miatt.

Fejezet 18. Sütik (cookie-k)

A PHP támogatja a HTTP cookie-k kezelését. A sütik lehetőséget adnak arra, hogy adatokat tárolj a kliens gépen, így követve vagy azonosítva a visszatérő látogatókat. Sütik beállítására a setcookie() függvénnyel nyílik lehetőség. A sütik részei a HTTP fejlécnek, így a setcookie() függényt azelőtt kell meghívni, mielőtt bármilyen kimenetet küldesz a böngészőnek. Ez a header() függvénnyel megegyező korlátozást jelent. A kimenet szabályozó függvényeket használhatod a kimenet késleltetésére addig, amíg minden sütit és fejlécet elküldtél.

Minden süti, amit a klienstől visszakapsz, automatikusan PHP változóvá válik, pont úgy, mint a GET és a POST kérésekkel érkező adatok, feltéve, hogy a `register_globals` és `variables_order` php.ini beállítások ennek megfelelően vannak beállítva. Ha több értéket szeretnél adni egy sütinek, add a szokásos *[]* végződést a süti nevéhez.

A PHP 4.1.0 és későbbi változataiban a `$_COOKIE` nevű mindenhonnan látható változó mindig létrejön, tartalmazva a klienstől érkezett sütiket. A `$HTTP_COOKIE_VARS` a korábbi verziókban használható, ha a `track_vars` php.ini beállítás be van kapcsolva, bár ez a változó nem látszik mindenhonnan.

Részletesebb információkért, beleértve a böngésző hibákat, lásd a setcookie() függvényt.

Fejezet 19. Fájlfeltöltés kezelése

POST metódusú feltöltések

A PHP alkalmas fájl feltöltést fogadni bármilyen RFC-1867 kompatibilis böngészőtől (mint a Netscape Navigator 3 vagy későbbi és a Microsoft Internet Explorer 3 Microsoft javítással, vagy későbbi IE javítás nélkül). Ez a szolgáltatás egyaránt lehetővé teszi a látogatónak szöveges és bináris fájlok feltöltését. A PHP azonosítási és fájlkezelési képességeivel teljes felügyeleted van afelett, hogy ki tölthet fel fájlt, és mi történik a feltöltött fájlokkal.

Érdemes megemlíteni, hogy a PHP támogatja a PUT metódust is, amit a Netscape Composer és a W3C Amaya kliensek használnak. Lásd a PUT metódusú feltöltések részt.

A fájl feltöltési lehetőség egy speciálisan kiképzett formmal biztosítható:

Példa 19-1. Fájlfeltöltő Űrlap

```
<form enctype="multipart/form-data" action="_URL_" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="1000">
Fájl elküldése: <input name="userfile" type="file">
<input type="submit" value="OK">
</form>
```

Az _URL_ a feldolgozást végző PHP fájlra kell, hogy mutasson. A MAX_FILE_SIZE rejtett mező a fájl input mező előtt kell, hogy szerepeljen, és azt adja meg, hogy mekkora a maximális fájl méret (byte-okban megadva), amit a PHP fogad.

> **Figyelem**
>
> A MAX_FILE_SIZE egy javasolt érték a böngészők számára. Könnyű megkerülni ezt a megadott maximumot. Ezért nem szabad arra építeni, hogy a böngésző úgy viselkedik, ahogy azt te szeretnéd. A PHP-beállítások a maximális feltöltési állományméretre azonban nem kerülhetőek meg.

A változók, amelyek egy sikeres feltöltés során létrejönnek a PHP változat és beállítás függvényében mások-mások lehetnek. A következő változók a feltöltés célprogramjában jönnek létre egy sikeres feltöltés során. Ha a track_vars beállítást engedélyezed, a $HTTP_POST_FILES és $_FILES tömbök jönnek létre. Végül a kapcsolódó változók létrejöhetnek globális változókként is, ha a register_globals beállítás be van kapcsolva. Ez azonban nem javasolt.

> **Megjegyzés:** A track_vars beállítás mindig be van kapcsolva a PHP 4.0.3 vagy újabb veziókban. A PHP 4.1.0 vagy újabb változatokban a $_FILES tömböt is használhatod a `$HTTP_POST_FILES` helyett. A `$_FILES` mindig elérhető, ezért nem szabad a `global` kulcsszót használni a $_FILES változóra függvényekben.

A `$HTTP_POST_FILES` és a `$_FILES` változók használata javasolt a feltöltött állomány információk elérésére. A tömb tartalma itt következik. Ebben az esetben feltételezzük, hogy a fenti űrlapnak megfelelően a ’userfile’ nevet adtad a feltöltési input mezőnek:

`$HTTP_POST_FILES['userfile']['name']`

Az eredeti fájlnév a kliensgépen.

`$HTTP_POST_FILES['userfile']['type']`

A fájl MIME típusa, ha a böngésző megadta ezt az információt. Például egy gif kép esetében: "image/gif".

`$HTTP_POST_FILES['userfile']['size']`

A feltöltött fájl mérete, byte-ban megadva.

`$HTTP_POST_FILES['userfile']['tmp_name']`

Az ideiglenes fájl elérési útja, ahol a feltöltött fájl tárolásra került a szerveren.

Megjegyzés: A PHP 4.1.0 és újabb verziók a rövidebb nevű `$_FILES` változót is biztosítják. A PHP 3 nem biztosítja a `$HTTP_POST_FILES` változót.

Ha a register_globals be van kapcsolva a `php.ini` fájlban, a következő változók jönnek létre. Ebben az esetben is feltételezzük, hogy a fenti űrlapnak megfelelően a 'userfile' nevet adtad a feltöltési input mezőnek:

- $userfile - Az ideiglenes fájl elérési útja, ahol a feltöltött fájl tárolásra került a szerveren.
- $userfile_name - Az eredeti fájlnév a kliens gépen.
- $userfile_size - A feltöltött fájl mérete, byte-okban megadva.
- $userfile_type - A fájl MIME típusa, ha a böngésző megadta ezt az információt. Például egy gif kép esetében: "image/gif".

Figyeld meg, hogy a "$userfile" előtag minden fenti változóban az űrlapban megadott érték, amit a type="file" sornál megadott <input> elem neveként határoztál meg. A fenti kérdőívben ezt "userfile"-nak választottuk.

Megjegyzés: A `register_globals = On` beállítás nem ajánlott biztonsági és teljesítmény okok miatt.

A fájlok alapbeállításban a szerver szokásos ideiglenes könyvtárában tárolódnak, ha nem adtál meg mást az upload_tmp_dir beállítással a `php.ini` fájlban. A szerver alapbeállítású könyvtára megváltoztatható a TMPDIR környezeti változóval abban a környezetben, ahol a PHP fut. Egy PHP szkriptből a putenv()-el való átállítása nem fog menni. Ez a környezeti változó arra is használható, hogy ellenőrizd, hogy más műveletek is végezhetőek-e a feltöltött fájlokon.

Példa 19-2. Fájlfeltöltések ellenőrzése

Az alábbi példák a PHP 4.0.2-nél újabb verzióival működnek. Lásd az is_uploaded_file() és move_uploaded_file() függvényeket.

```
<?php
```

```
// A PHP 4.1.0 vagy későbbi verzióiban a $_FILES
// használandó a $HTTP_POST_FILES helyett
if (is_uploaded_file($HTTP_POST_FILES['userfile']['tmp_name'])) {
    copy($HTTP_POST_FILES['userfile']['tmp_name'], "/a/feltoltott/file/uj/helye");
} else {
    echo "Lehetséges támadás. Fájlnév: " . $HTTP_POST_FILES['userfile']['name'];
}
/* ...vagy... */
move_uploaded_file($HTTP_POST_FILES['userfile']['tmp_name'], "/a/feltoltott/file/uj/helye
?>
```

A PHP program, ami megkapja a feltöltött fájlt, gondoskodik arról is, hogy a kívánt műveleteket elvégezze a fájlal. Például törölheti a fájlt, ha az túl nagy, vagy túl kicsi, figyelembe véve a `$HTTP_POST_FILES['userfile']['size']` változót, vagy meghatározhatja a `$HTTP_POST_FILES['userfile']['type']` alapján, hogy ez a fájl megfelel-e egy meghatározott fájltípusnak, és ha nem, törölheti. Bármi is a cél a feltöltött fájlal, a PHP szkriptnek kell gondoskodnia arról, hogy elmozgassa egy biztonságos helyre, vagy törölje az ideiglenes könyvtárból az adott fájlt.

A fájl törlésre kerül az ideiglenes könyvtárból a kérés végrehajtásának végeztével, ha nem mozgatod el, vagy nem nevezed át.

Tipikus csapdák

A `MAX_FILE_SIZE` nem tartalmazhat nagyobb értéket, mint az upload_max_filesize beállítás értéke. Az alapbeállítás 2 megabyte.

Ha egy memória korlát be van állítva, esetleg növelned kell a memory_limit értékét. Gondoskodj a memory_limit kellően nagyra állításáról.

Ha a max_execution_time túl kicsire van állítva, a file feltöltési folyamat során letelhet az idő. Ezért gondoskodj a `max_execution_time` kellően nagyra állításáról.

Ha a post_max_size túl kicsi, nagy állományok nem tölthetőek fel. Gondoskodj a `post_max_size` kellően nagyra állításáról.

Ha nem ellenőrzöd a fájlokat, amin műveleteket végzel, a felhasználói esetleg más könyvtárakhoz is hozzáférhetnek...

A CERN httpd szerver úgy tűnik, hogy eldob mindent a klienstől kapott Content-type MIME fejlécben az első szóközt követően. Amíg ez fennáll, a CERN httpd szerver nem fogja támogatni a fájl feltöltéseket.

Több fájl egyidejű feltöltése

Lehetséges több fájl egyidejű feltöltése is, az `input` elemek `name` paramétereinek különböző megadásával.

Úgyszintén lehetőség van több megegyező nevű űrlap elemmel is több állomány feltöltésére. Ebben az esetben a kapcsolódó információkat tömbökben kapod meg. Ahhoz, hogy ezt elérd, a hagyományos tömb hivatkozást kell alkalmaznod, mint minden más űrlapelemnél:

Megjegyzés: Több fájl egyidejű feltöltése a PHP 3.0.10 óta lehetséges.

Példa 19-3. Több fájl egyidejű feltöltése

```
<form action="file-feltolt.php" method="post" enctype="multipart/form-data">
  Az alábbi fájlok elküldése:<br>
  <input name="userfile[]" type="file"><br>
  <input name="userfile[]" type="file"><br>
  <input type="submit" value="OK">
</form>
```

Amikor a fenti űrlap adatai elküldésre kerülnek, a `$HTTP_POST_FILES['userfile']` `$HTTP_POST_FILES['userfile']['name']` és `$HTTP_POST_FILES['userfile']['size']` változók értéket kapnak. A $_FILES tömbben ugyanezek elérhetőek a PHP 4.1.0 és újabb verziókban. Ezek mind számokkal indexelt tömbök a tömbben beküldötteknek megfelelő értékekkel. A PHP 3-asban a $HTTP_POST_VARS használható. Ha a `register_globals` be van kapcsolva, globális változókat is kapsz.

Például ha a `/home/test/review.html` és `/home/test/xwp.out` fájlok kerültek beküldésre, a `$HTTP_POST_FILES['userfile']['name'][0]` tartalma `review.html` és a `$HTTP_POST_FILES['userfile']['name'][1]` tartalma `xwp.out`. Hasonló módon a `$HTTP_POST_FILES['userfile']['size'][0]` a `review.html` fájl méretét tartalmazza, stb.

`$HTTP_POST_FILES['userfile']['name'][0]`,
`$HTTP_POST_FILES['userfile']['tmp_name'][0]`,
`$HTTP_POST_FILES['userfile']['size'][0]` és
`$HTTP_POST_FILES['userfile']['type'][0]` szintén elérhetőek.

PUT metódusú feltöltések

A PHP támogatja a HTTP PUT metódust is, amit például a Netscape Composer és a W3C Amaya használ. A PUT kérések sokkal egyszerűbbek, mint a fájl feltöltések. A következőképpen néznek ki:

```
PUT /eleresi/ut/filenev.html HTTP/1.1
```

Ez hagyományosan azt jelenti, hogy a kliens a küldött adatokat az /eleresi/ut/filenev.html fájlba szeretné elmenteni a webgyökér alatt. Az nyilvánvalóan nem lenne jó megoldás az Apache vagy a PHP részéről, ha bárkinek megengedné, hogy felülírja a fájlokat a web könyvtáradban. Éppen ezért a PUT kérések kezeléséhez be kell állítani a webszerver számára, hogy egy PHP szkriptnek küldje az ilyen bemenetet. Apache alatt ezt a *Script* direktívával teheted meg. Ez elhelyezhető szinte tetszőleges ponton az Apache konfigurációs fájlodban. Egy gyakori hely erre egy <Directory> blokk belseje, vagy esetleg egy <Virtualhost> blokk belseje. Például egy ilyen sor megoldja a feladatot:

```
Script PUT /put.php
```

Ez beállítja az Apache számára a PUT kérések kezelésére a put.php-t abban a környezetben, ahol ezt a sort elhelyezted a konfiguráláskor. Ez természetesen feltételezi, hogy a .php kiterjesztést a PHP kezeli és a PHP aktív.

A put.php fájlban aztán valami hasonlót tehetsz:

```
<?php copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>
```

Ez a kérés által meghatározott helyre másolja a küldött fájlt. Valós helyzetben természetesen szükséges valamilyen ellenőrzés, és/vagy felhasználóazonosítás, mielőtt esetleg felülírod egyik fontos fájlodat. A PHP a POST metódushoz hasonlóan egy ideiglenes fájlban tárolja a feltöltött fájlt. Amikor a kérés teljesítése befejeződött, ez az ideiglenes fájl törlődik. Ez azt jelenti, hogy a PUT kéréseket feldolgozó szkriptednek ezt a fájlt el kell mozgatnia máshova, ha meg szeretnéd tartani a feltöltött fájlt. Az ideiglenesen létrehozott fájl elérési útját a fájl nevével a $PHP_PUT_FILENAME változó tartalmazza, és a javasolt célt a $REQUEST_URI változó tartalmazza (bár ez lehet más is nem Apache szervereken). Ez a cél az, amit a kliens meghatározott. Neked nem kell feltétlenül ezt a helyet elfogadnod, lehet, hogy neked az a kényelmesebb (és biztonságosabb), hogy a feltöltött fájlokat egy speciális upload könyvtárban tárolod.

Fejezet 20. Távoli file-ok kezelése

Amennyiben az "URL fopen wrapper" támogatás be volt állítva használhatsz HTTP és FTP URL-eket majdnem minden olyan függvénnyel, ami fájlnevet kér paraméterül, beleértve a require()-t és az include()-ot is. Az "URL fopen wrapper" támogatás a PHP 4.0.3-as és korábbi verzióiban be van állítva, ha nem adtad meg a configure futtatásakor a `--disable-url-fopen-wrapper` paramétert. Későbbi verziókban ezt a szolgáltatást az *`allow_url_fopen`* `php.ini` beállítás szabályozza.

> **Megjegyzés:** Nem adhatsz meg távoli file-okat az include() és a require() használatakor Windows alatt!

Használhatod például ezt a funkciót arra, hogy egy távoli webszerveren lévő fájlt megnyiss, majd feldolgozd a kívánt adatokat, és felhasználd egy adatbázis lekérdezésben, vagy csak egyszerűen a saját site-od kinézetével, stílusával tálald.

Példa 20-1. Egy távoli weboldal címsorának megállapítása

```
<?php
$file = fopen ("http://www.example.com/", "r");
if (!$file) {
    echo "<p>Nem lehet megnyitni a külső file-t!\n";
    exit;
}
while (!feof ($file)) {
  $line = fgets ($file, 1024);
  /* Ez csak akkor jó, ha a cím és a körbezáró tag-ek egy sorban vannak */
  if (eregi ("<title>(.*)</title>", $line, $out)) {
      $title = $out[1];
      break;
  }
}
fclose($file);
?>
```

Lehetőség van arra is, hogy egy FTP szerveren fájlba írj, feltéve, hogy megfelelő jogokkal rendelkező user-ként lépsz be, és a fájl még nem létezik. Ha nem ’anonymous’ user-ként szeretnél belépni, a usernevet és jelszót az URL részeként kell megadnod a alábbi formában: ’ftp://user:jelszo@ftp.pelda.hu/eleresi/ut/alma.txt’. (Ugyanezt a módszert használhatod akkor is, ha olyan fájlokat szeretnél elérni HTTP-vel, amik a Basic azonosítást igénylik.)

Példa 20-2. Adat tárolása távoli gépen

```
<?php
$file = fopen ("ftp://ftp.example.com/incoming/outputfile", "w");
if (!$file) {
    echo "<p>Nem lehet megnyitni a külső file-t írásra.\n";
```

```
    exit;
}
/* Itt írunk a file-ba */
fputs ($file, "$HTTP_USER_AGENT\n");
fclose ($file);
?>
```

Megjegyzés: A fenti példa alapján már látható, hogy milyen technikát kell haszálni ha például távoli naplózást szeretnél alkalmazni, de mint fent is olvashattad, ez a technika csak nemlétező fájlokba való írásra alkalmas. Egy sokkal célzottab megoldás a távoli naplózásra a syslog() függvény használata.

Fejezet 21. Kapcsolatkezelés

Megjegyzés: Az alábbi fejezetek csak a PHP 3.0.7-es és későbbi verzióira vonatkoznak!

A PHP belsőleg nyilvántartja a kapcsolati státuszt. Három lehetséges állapot van:

- 0 - NORMAL (Normál)
- 1 - ABORTED (Megszakított)
- 2 - TIMEOUT (Időtúllépéses)

Amikor egy PHP szkript fut, alapállapotban a NORMAL állapot aktív. Ha a távoli kliens bontja a kapcsolatot, az ABORTED statátusz jelzése lesz aktív. Ez tipikusan akkor áll elő, ha a látogató a STOP gomb-ot használja a böngészőjében. Ha a PHP által felügyelt időkorált kerül túllépésre (lásd a set_time_limit() függvényt), a TIMEOUT állapot válik aktívvá.

Eldöntheted, hogy ha a kliens bontja a kapcsolatot, a szkript is leálljon-e vagy sem. Néha hasznos lehet, ha a szkriptjeid mindig végigfutnak, annak ellenére, hogy a kliens már nem fogadja a kimenetet. Alapbeállításban azonban a szkript is befejezi a futását, ha a kliens bontja a kapcsolatot. Ez a viselkedés az ignore_user_abort `php.ini` beállítással, valamint az ennek megfelelő "php_value ignore_user_abort" Apache .conf direktívával állítható, vagy az ignore_user_abort() függvénnyel. Ha nem konfigurálod úgy a PHP-t, hogy hagyja figyelmen kívül a kliens kapcsolatbontását, a szkriptjeid le fognak állni ilyen esetekben. Egyetlen kivétel ez alól, ha egy ’shutdown’ függvényt definiálsz a register_shutdown_function()-al. Egy ilyen beállítással, ha a látogató lenyomja a STOP gombot, a szkripted következő kimenet-küldési kísérletére a PHP a ’shutdown’ függvényt fogja meghívni. A ’shutdown’ függvény abban az esetben is meghívásra kerül, ha a szkript normálisan befejezi a futását, tehát ha valami speciálisat szeretnél tenni, amikor a kliens bontja a kapcsolatot, a connection_aborted() függvényt használhatod. Ez igazat fog visszaadni, ha a kapcsolatot a kliens bontotta.

A szkripted a belső időmérés következtében is megállhat. Alapbeállításban egy szkript maximum 30 másodpercig futhat. Ez megváltoztatható a max_execution_time `php.ini` direktívával, illetve a megfelelő "php_value max_execution_time" Apache .conf beállítással, valamint a set_time_limit() függvénnyel. Amikor ez az idő letelik, a szkript megáll, és ha a fenti esetben említett ’shutdown’ függvény definiált, az kerül meghívásra. Az időtúllépés esetét a connection_timeout() függvényel állapíthatod meg. Ez igazat fog visszaadni, ha időtúllépés miatt hívódott meg a ’shutdown’.

Fontos megjegyezni, hogy az ABORTED és TIMEOUT állapotok egyszerre is aktívak lehetnek, ha a PHP-ben a kliens kapcsolatbontásának figyelmen kívül hagyását kérted. A PHP tudni fogja, hogy a kliens már bontotta a kapcsolatot, de a szkript futni fog tovább. Ha ráadásul eléri az időkorlátot, a szkript megáll, és a ’shutdown’ függvény hívódik meg (ha beállítottál ilyet). Ezen a ponton azt fogod tapasztalni, hogy mind a connection_timeout(), mind a connection_aborted() igazat ad. Mindkettőt ellenőrizheted, ha a connection_status() függvényt hívod. Ez egy bitmezőt ad vissza, az aktív állapotokkal. Tehát ebben az esetben, mivel mindkét állapot aktív, 3-at fogsz visszakapni.

Fejezet 22. Állandó adatbázis kapcsolatok

Az állandó kapcsolatok SQL adatbázisokkal nem szűnnnek meg, ha a szkripted futása befejeződik. Ha egy állandó kapcsolatot kérsz, a PHP ellenőrzi, hogy van-e már megegyező kapcsolat (ami még az előző kérésekből maradhatott meg), és ha létezik, akkor azt használja. Ha nem talál ilyet, létrehoz egy kapcsolatot. A megegyező kapcsolat azt jelenti, hogy ugyanaz a host és ugyanaz a felhasználói név és jelszó került felhasználásra.

> **Megjegyzés:** Állandó kapcsolatokat nem csak adatbázisok szolgáltathatnak, vannak más ilyen képességű kiterjesztések, mint például az IMAP kiterjesztés.

Ha esetleg nem ismered alaposabban a webszerverek működését, hibás kép alakulhat ki benned az állandó kapcsolatokról. Ezek a kapcsolatok *nem* alkalmasak arra, hogy felhasználói ’session’-eket nyiss ugyanazon SQL linkkel. *Nem* adnak lehetőséget hatékony tranzakciók felépítésére. Egészen pontosan, hogy alaposabban tisztázzuk a kérdést, az állandó adatbázis kapcsolatok nem adnak *semmilyen* plusz lehetőséget, ami nélkülük nem létezne.

Miért?

A válaszhoz meg kell érteni, hogyan működnek együtt a webszerverek a PHP-vel. Ennek három különböző módja lehetséges.

Az első lehetőség, hogy a PHP-t CGI "wrapper"-ként használod. Ha ezt a módszert használod, minden oldal lekérésekor és feldolgozásakor egy új példány fut le a PHP feldolgozóból. Mivel a szkript futtatása után egy ilyen példány leáll, minden erőforrás, amit lefoglalt (beleértve az adatbázis kapcsolatotokat) megszűnik. Ebben az esetben semmit sem érsz azzal, hogy állandó kapcsolatot próbálsz nyitni, ez az állandóság nem valósul meg.

A népszerűbb második forma, amikor a PHP-t modulként futtatod egy több process-es webszerverben. Egy több process-es webszerver tipikusan rendelkezik egy szülő process-el, ami koordinálja a többi kapcsolódó process (a gyermekek) munkáját, amik valójában a weboldalak kiszolgálását végzik. Ha egy kérés érkezik egy klienstől, egy éppen szabad gyermekprocess kapja meg a kiszolgálásra az utasítást. Ez azt jelenti, hogy ha ugyanaz a kliens egy újabb kapcsolatot kezdeményez, esetleg egy másik gyermekprocesshez jut, mint az első alkalommal. Ebben az esetben az állandó adatbázis kapcsolat azt jelenti a számodra, hogy az egyes gyermekprocess-ek csak az első alkalommal kell, hogy létrehozzák a kapcsolatot az adatbázissal, és amennyiben egy későbbi kérés ugyanebben a gyermekprocessben ugyanazt a kapcsoltatot kívánja megnyitni, a létező kapcsolat kerül felhasználásra.

A harmadik módszer, hogy a PHP-t plug-in-ként használod egy ’multithreaded’ web szerverben. Ez azt jelenti, hogy az ISAPI, WSAPI, és NSAPI (Windows alatt) formák használhatóak a PHP-vel. Ez a PHP 4.0.0 óta lehetséges, és így a PHP alkalmas plug-in szintű együttműködésre a Netscape FastTrack (iPlanet), a Microsoft Internet Information Server (IIS), és az O’Reilly WebSite Pro szerverekkel, valamint más, a fenti standardokat támogató szerverekkel. Ebben az esetben az állandó adatbázis kapcsolatok működése megegyezik a fent leírt több process-es modellel.

Ha az állandó adatbázis kapcsolatok nem nyújtanak plusz szolgáltatásokat, mégis mire jók?

A válasz igen egyszerű: hatékonyság! Az állandó adatbázis kapcsolatok akkor lehetnek hasznosak, ha nagy a feleslegesen adatbázishoz kapcsolódással eltöltött idő. Az, hogy ez valójában milyen esetben van így, rengeteg faktoron múlik. Például azon, hogy milyen típusú adatbázisról van szó, azonos, vagy különböző gépen van-e, mint a szerver, mennyire terhelt az SQL szerver, stb. Lényegében, ha sok időt vesz igénybe a kapcsolódás, az állandó kapcsolatok jelentős segítséget nyújthatnak neked. Egy gyermekprocess így csak egy alkalommal kell, hogy kapcsolódjon, ahelyett, hogy egy ezt kérő oldal minden feldolgozásakor megtenné. Ez azt is jelenti, hogy minden gyermekprocessnek, meglesz a maga állandó kapcsolata a szerver felé. Például, ha 20 különböző

gyermekprocess dolgozott fel egy állandó adatbáziskapcsolatot kérő oldalt, 20 különböző állandó kapcsolatod lesz az SQL szerverhez, egy-egy minden gyermektől.

Fontos megjegyezni azonban, hogy ennek lehetnek hátrányos következményei is, ha az adatbázisszerver korlátozott kapcsolatainak számát az állandó kapcsolatok lefoglalják. Ha az adatbázisszervered egyidejűleg maximálisan 16 kapcsolatot képes kezelni, és egy forgalmas időszakban egyszerre 17 process próbál meg kapcsolódni az adatbázishoz, az egyik képtelen lesz erre... Ha olyan hiba van a programodban (például végetelen ciklus), ami nem hagyja a kapcsolat felbontását, egy csak 32 kapcsolattal bíró adatbázis alaposan le lesz foglalva. Nézz utána az adatbázisszervered dokumentációjában, hogy hogyan tudod lekezelni az elhagyott vagy inaktív kapcsolatokat.

Figyelem

Van még néhány faktor, amit érdemes figyelembe venned, ha állandó adatbázis kapcsolatokat használsz. Egy ilyen probléma, hogy ha tábla lezárást (lock) használsz egy állandó kapcsolaton, és a szkript valamilyen okból nem tudja feloldani a zárat, ezt a kapcsolatot használó további szkriptek nem fognak helyesen működni, és a webszerver vagy adatbázis szerver újraindítására lehet szükség a feloldáshoz. Hasonlóan ha tranzakciókat használsz, a tranzakció blokk tovább folytatódik a következő megegyező kapcsolatot használó szkriptben, ha a tranzakciót indító szkript nem tudja lezárni azt. Ezekben az esetekben a register_shutdown_function() függvényt használhatod, hogy egy egyszerű "takarító" függvényt futtass le a programod végeztével, ami visszavonja a tranzakciókat, és feloldja a tábla zárakat. Jobban teszed azonban, ha úgy kerülöd meg a problémát, hogy nem használsz állandó kapcsolatokat olyan szkriptekben, amik tábla zárakat, vagy tranzakciókat alkalmaznak.

Összefoglalva: az állandó adatbáziskapcsolatokat úgy tervezték, hogy megfeleltethetőek legyenek a hagyományos kapcsolatokkal. Ez azt jelenti, hogy *minden esetben* lehetőséged van az állandó kapcsolatokat hagyományos kapcsolatokra cserélni, és ez nem fogja megváltoztatni a szkriptjeid működését. Ez a lépés megváltoztat*hat*ja a szkripted hatékonyságát, de a viselkedését nem!

Lásd még fbsql_pconnect(), ibase_pconnect(), ifx_pconnect(), imap_popen(), ingres_pconnect(), msql_pconnect(), mssql_pconnect(), mysql_pconnect(), **OCIPLogon()**, odbc_pconnect(), **Ora_pLogon()**, pfsockopen(), pg_pconnect(), és sybase_pconnect().

Fejezet 23. Safe Mode

A safe mode egy próbálkozás a megosztott szerverek biztonsági problémáinak megoldására. Architektúrálisan nem korrekt, hogy ezt a problémát a PHP szintjén próbáljuk megoldani, de mivel a többi alternatíva a webszerver és operációs rendszer szinteken nem igazán használható, sokan - különösen az internetszolgáltatók - a safe mode-ot használják egyelőre.

A safe mode működését befolyásoló beállítások:

```
safe_mode = Off
safe_mode_gid = 0
safe_mode_include_dir =
safe_mode_exec_dir =
open_basedir =
safe_mode_allowed_env_vars = PHP_
safe_mode_protected_env_vars = LD_LIBRARY_PATH
disable_fnctions =
```

Ha a safe_mode beállítás be van kapcsolva, a PHP ellenőrzi, hogy az aktuálisan futó szkript tulajdonosa megegyezik-e a kezelésre megnyitandó file tulajdonosával. Például:

```
-rw-rw-r--    1 rasmus   rasmus       33 Jul  1 19:20 script.php
-rw-r--r--    1 root     root       1116 May 26 18:01 /etc/passwd
```

Futtatva ezt a script.php programot:

```
<?php
 readfile('/etc/passwd');
?>
```

a következő hibát kapod, ha a safe mode be van kapcsolva:

```
Warning: SAFE MODE Restriction in effect. The script whose uid is 500 is not
allowed to access /etc/passwd owned by uid 0 in /docroot/script.php on line 2
```

Ha a safe_mode helyett egy open_basedir könyvtárat állítasz be, akkor minden engedélyezett file művelet erre a könyvtárra korlátozódik. Például (Apache httpd.conf példa):

```
<Directory /docroot>
php_admin_value open_basedir /docroot
</Directory>
```

Ha a fenti script.php programot futtatod, ezzel az open_basedir beállítással, akkor a következő eredményt kapod:

```
Warning: open_basedir restriction in effect. File is in wrong directory in
/docroot/script.php on line 2
```

Le tudsz tiltani különböző függvényeket akár egyenként is. Meg kell azonban jegyezni, hogy a disable_functions beállítás csak a `php.ini`-ben használható. Ez azt jelenti, hogy nem tudsz virtuális hosztonként, vagy könyvtáranként függvényeket letiltani a httpd.conf vagy .htaccess állományokban. Ha hozzáadod ezt a php.ini állományodhoz:

```
disable_functions readfile,system
```

A következő hibát kapod:

```
Warning: readfile() has been disabled for security reasons in
/docroot/script.php on line 2
```

A safe mode használatakor tiltott/korlátozott függvények

Ez még valószínűleg nem teljes, és nem korrekt listája a függvényeknek, amiket a safe mode korlátoz.

Táblázat 23-1. Safe modeban korlátozott függvények

Függvény	Korlátozás
dbmopen()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.
dbase_open()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.
filepro()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.

Függvény	**Korlátozás**
filepro_rowcount()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.
filepro_retrieve()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.
ifx_*()	sql_safe_mode megkötések, (!= safe mode)
ingres_*()	sql_safe_mode megkötések, (!= safe mode)
mysql_*()	sql_safe_mode megkötések, (!= safe mode)
pg_loimport()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.
posix_mkfifo()	Ellenőrzi, hogy a könyvtár, amelyben dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkezik-e, mint az éppen futó program.
putenv()	Alkalmazkodik a safe_mode_protected_env_vars és safe_mode_allowed_env_vars ini beállításokhoz. Lásd még a putenv() leírását
move_uploaded_file()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.
chdir()	Ellenőrzi, hogy a könyvtár, amelyben dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkezik-e, mint az éppen futó program.
dl()	Ez a függvény nem használható, ha a safe mode be van kapcsolva.
végrehajtó operátor	Ez a függvény nem használható, ha a safe mode be van kapcsolva.
shell_exec() (a végrehajtó operátor függvény megfelelője)	Ez a függvény nem használható, ha a safe mode be van kapcsolva.
exec()	Futtatható állományok végrehajtására csak a safe_mode_exec_dir könyvtáron belül van lehetőség. Praktikus okok miatt nem lehetséges . . komponenst elhelyeni a futtatandó program elérési útjában.
system()	Futtatható állományok végrehajtására csak a safe_mode_exec_dir könyvtáron belül van lehetőség. Praktikus okok miatt nem lehetséges . . komponenst elhelyeni a futtatandó program elérési útjában.

Függvény	**Korlátozás**
passthru()	Futtatható állományok végrehajtására csak a safe_mode_exec_dir könyvtáron belül van lehetőség. Praktikus okok miatt nem lehetséges `..` komponenst elhelyeni a futtatandó program elérési útjában.
popen()	Futtatható állományok végrehajtására csak a safe_mode_exec_dir könyvtáron belül van lehetőség. Praktikus okok miatt nem lehetséges `..` komponenst elhelyeni a futtatandó program elérési útjában.
mkdir()	Ellenőrzi, hogy a könyvtár, amelyben dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkezik-e, mint az éppen futó program.
rmdir()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.
rename()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program. Ellenőrzi, hogy a könyvtár, amelyben dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkezik-e, mint az éppen futó program.
unlink()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program. Ellenőrzi, hogy a könyvtár, amelyben dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkezik-e, mint az éppen futó program.
copy()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program. Ellenőrzi, hogy a könyvtár, amelyben dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkezik-e, mint az éppen futó program. (a *`source`* és *`target`* paraméterekre)
chgrp()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.
chown()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.

Függvény	**Korlátozás**
chmod()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program. Ráadásul nem állíthatod át a SUID, SGID és sticky biteket
touch()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program. Ellenőrzi, hogy a könyvtár, amelyben dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkezik-e, mint az éppen futó program.
symlink()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program. Ellenőrzi, hogy a könyvtár, amelyben dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkezik-e, mint az éppen futó program. (csak a cél ellenőrzött)
link()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program. Ellenőrzi, hogy a könyvtár, amelyben dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkezik-e, mint az éppen futó program. (csak a cél ellenőrzött)
ob_gzhandler()	Ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.
getallheaders()	Safe modeban, semmilyen ’authorization’ (nem kisbetű/nagybetű érzékeny) kezdetű fejléc sem kerül visszaadásra. Figyelem: ez nem működik megfelelően a getallheaders() AOL szerveres megvalósításában.
Minden függvény, ami a `php4/main/fopen_wrappers.c` funkciókat használja.	??

Fejezet 24. Using PHP from the command line

Since version 4.3, PHP supports a new SAPI type (Server Application Programming Interface) named CLI which means *Command Line Interface*. As the name implies, this SAPI type main focus is on developing shell (or desktop as well) applications with PHP. There are quite some differences between the CLI SAPI and other SAPIs which are further explained throughout this chapter.

The CLI SAPI was released for the first time with PHP 4.2.0, but was still experimental back then and had to be explicitely enabled with --enable-cli when running ./configure. Since PHP 4.3.0 the CLI SAPI is no longer experimental and is therefore **always** built and installed as the php (called php.exe on Windows) binary.

Remarkable differences of the CLI SAPI compared to other SAPIs:

- Unlikely the CGI SAPI, no headers are written to the output.

  Though the CGI SAPI provies a way to suppress HTTP headers, there's not equivalent switch to enable them in the CLI SAPI.

- The are certain php.ini directives which are overriden by the CLI SAPI because the do not make sense in shell environments:

Táblázat 24-1. Overriden php.ini directives

Directive	CLI SAPI default value	Comment
html_errors	FALSE	It can be quite hard to read the error message in your shell when it's cluttered with all those meaningless HTML tags, therefore this directive defaults to FALSE.
implicit_flush	TRUE	It is desired that any output coming from print(), echo() and friends is immidiately written to the output and not cached in any buffer. You still can use output buffering if you want to defer or manipulate standard output.
max_execution_time	0 (unlimited)	Due to endless possibilities of using PHP in shell environments, the maximum execution time has been set to unlimited. Whereas applications written for the web are executed within splits of a seconds, shell application tend to have a much longer execution time.
register_argc_argv	TRUE	The global PHP variables $argc (number of arguments passed to the application) and $argv (array of the actual arguments) are always registered and filled in with the appropriate values when using the CLI SAPI.

> **Megjegyzés:** These directives cannot be initialzied with another value from the configuration file `php.ini` or a custom one (if specified). This is a limitation because those default values are applied after all configuration files have been parsed. However, their value can be changed during runtime (which does not make sense for all of those directives, e.g. register_argc_argv).

- To ease working in the shell environment, the following constants are defined:

Táblázat 24-2. CLI specific Constants

Constant	Description
`STDIN`	An already opened stream to `stdin`. This saves opening it with `$stdin = fopen('php://stdin', 'r');`
`STDOUT`	An already opened stream to `stdout`. This saves opening it with `$stdout = fopen('php://stdout', 'w');`
`STDERR`	An already opened stream to `stdout`. This saves opening it with `$stderr = fopen('php://stderr', 'w');`

Given the above, you don't need to open e.g. a stream for `stderr` yourself but simply use the constant instead of the stream resource:

```
php -r 'fwrite(STDERR, "stderr\n");'
```

You do not need to explicitly close these streams, this is automatically done by `PHP`.

- The `CLI SAPI` does **not** change the current directory to the directory of the executed script !

Example showing the difference to the `CGI SAPI`:

```
<?php
    /* Our simple test application */
    echo getcwd(), "\n";
?>
```

When using the `CGI` version, the output is

```
$ pwd
/tmp

$ php-cgi -f another_directory/test.php
/tmp/another_directory
```

This clearly shows that PHP changes its current directory to the one of the executed script.

Using the CLI SAPI yields:

```
$ pwd
/tmp

$ php -f another_directory/test.php
/tmp
```

This allows greater flexibility when writing shell tools in PHP.

> **Megjegyzés:** The CGI SAPI supports the CLI SAPI behaviour by means of the -C switch when ran from the command line.

The list of command line options provided by the PHP binary can be queried anytime by running PHP with the -h switch:

```
Usage: php [options] [-f] <file> [args...]
       php [options] -r <code> [args...]
       php [options] [-- args...]
  -s               Display colour syntax highlighted source.
  -w               Display source with stripped comments and whitespace.
  -f <file>        Parse <file>.
  -v               Version number
  -c <path>|<file> Look for php.ini file in this directory
  -a               Run interactively
  -d foo[=bar]     Define INI entry foo with value 'bar'
  -e               Generate extended information for debugger/profiler
  -z <file>        Load Zend extension <file>.
  -l               Syntax check only (lint)
  -m               Show compiled in modules
  -i               PHP information
  -r <code>        Run PHP <code> without using script tags <?..?>
  -h               This help

  args...          Arguments passed to script. Use -- args when first argument
                   starts with - or script is read from stdin
```

The CLI SAPI has three different ways of getting the PHP code you want to execute:

1. Telling PHP to execute a certain file.

   ```
   php my_script.php
   ```

```
php -f my_script.php
```

Both ways (using the `-f` switch or not) execute the given file `my_script.php`. You can choose any file to execute, your PHP scripts do not have to end with the `.php` extension but can give them any name or extension you want them to have.

2. Pass the PHP code to execute directly on the command line.

```
php -r 'print_r(get_defined_constants());'
```

Special care has to be taken in regards of shell variable substitution and quoting usage.

> **Megjegyzés:** Read the example carefully, thera are no beginning or ending tags! The `-r` switch simply does not need them. Using them will lead to a parser error.

3. Provide the PHP code to execute via standard input (`stdin`).

This gives the powerful ability to dynamically create PHP code and feed it to the binary, as shown in this (fictional) example:

```
$ some_application | some_filter | php | sort -u >final_output.txt
```

You cannot combine any of the three ways to execute code.

Like every shell application, the PHP binary accepts a number of arguments but also your PHP script can receive them. The number of arguments which can be passed to your script is not limited by PHP (the shell has a certain size limit in numbers of characters which can be passed; usually you won't hit this limit). The arguments passed to your script are available in the global array `$argv`. The zero index always contains the script name (which is `-` in case the PHP code is coming from either standard input or from the command line switch `-r`). The second registered global variable is `$argc` which contains the number of elements in the `$argv` array (**not** the number of arguments passed to the script).

As long as the arguments you want to pass to your script do not start with the `-` character, there's nothing special to watch out for. Passing an argument to your script which starts with a `-` will cause trouble because PHP itself thinks it has to handle it. To prevent this use the argument list separator `--`. After the argument has been parsed by PHP, every argument following it is passed untoched/unparsed to your script.

```
# This will not execute the given code but will show the PHP usage
$ php -r 'var_dump($argv);' -h
Usage: php [options] [-f] <file> [args...]
```

```
[...]

# This will pass the '-h' argument to your script and prevent PHP from show-
ing it's usage
$ php -r 'var_dump($argv);' -- -h
array(2) {
  [0]=>
  string(1) "-"
  [1]=>
  string(2) "-h"
}
```

However, there's another way of using PHP for shell scripting. You can write a script where the first line starts with `#!/usr/bin/php` and then following the normal PHP code included within the PHP starting and end tags and set the execution attributes of the file appropriately. This way it can be executed like a normal shell or perl script:

```
#!/usr/bin/php
<?php
    var_dump($argv);
?>
```

Assuming this file is named `test` in the current directory, we can now do the following:

```
$ chmod 755 test
$ ./test -h -- foo
array(4) {
  [0]=>
  string(6) "./test"
  [1]=>
  string(2) "-h"
  [2]=>
  string(2) "--"
  [3]=>
  string(3) "foo"
}
```

As you see no care has to be taken when passing parameters to your script which start with `-`.

Táblázat 24-3. Command line options

Option	Description

Option	Description
-s	Display colour syntax highlighted source. This option uses the internal mechanism to parse the file and produces a `HTML` highlighted version of it and writes it to standard output. Note that all it does it to generate a block of `<code> [...] </code>` `HTML` tags, no `HTML` headers. **Megjegyzés:** This option does not work together with the `-r` option.
-w	Display source with stripped comments and whitespace. **Megjegyzés:** This option does not work together with the `-r` option.
-f	Parses and executed the given filename to the `-f` option. This switch is optional and can be left out. Only providing the filename to execute is sufficient.

-v	Writes the PHP, PHP SAPI, and Zend version to standard output, e.g.
	
	`$ php -v`
	`PHP 4.3.0-dev (cli), Copyright (c) 1997-2002 The PHP Group`
	`Zend Engine v1.2.1, Copyright (c) 1998-2002 Zend Technologies`

-c	With this option one can either specify a directory where to look for `php.ini` or you can specify a custom `INI` file directly (which does not need to be named `php.ini`), e.g.:
	
	`$ php -c /custom/directory/ my_script.php`
	
	`$ php -c /custom/directory/custom-file.ini my_scrip`

-a	Runs PHP interactively.

Option	Description

-d	This option allows to set a custom value for any of the configuration directives allowed in `php.ini`. The syntax is:
	
	`-d configuration_directive[=value]`
	
	Examples:
	
	`# Ommiting the value part will set the given con-`
	`figuration directive to "1"`
	`$ php -d max_execution_time -r '$foo = ini_get("max`
	`string(1) "1"`
	
	`# Passing an empty value part will set the con-`
	`figuration directive to ""`
	`php -d max_execution_time= -r '$foo = ini_get("max_`
	`string(0) ""`
	
	`# The configuration directive will be set to any-`
	`thing passed after the '=' character`
	`$  php -d max_execution_time=20 -r '$foo = ini_get(`
	`string(2) "20"`
	`$  php -d max_execution_time=doesntmakesense -`
	`r '$foo = ini_get("max_execution_time"); var_dump($`
	`string(15) "doesntmakesense"`

-e	Generate extended information for debugger/profiler.
-z	Load Zend extension. If only a filename is given, PHP tries to load this extension from the current default library path on your system (usually specified `/etc/ld.so.conf` on Linux systems). Passing a filename with an absolute path information will not use the systems library search path. A relative filename with a directory information will tell `PHP` only to try to load the extension relative to the current directory.

Option	Description
-l	This option provides a convenient way to only perform a syntax check on the given PHP code. On succes, the text `No syntax errors detected in <filename>` is written to standard output and the shell return code is `0`. On failure, the text `Errors parsing <filename>` in addition to the internal parser error message is written to standard output and the shell return code is set to `255`. This option won't find fatal errors (like undefined functions). Use `-f` if you would like to test for fatal errors too. **Megjegyzés:** This option does not work together with the `-r` option.

-m	Using this option, PHP prints out the built in (and loaded) PHP and Zend modules:
	
	`$ php -m`
	`[PHP Modules]`
	`xml`
	`tokenizer`
	`standard`
	`session`
	`posix`
	`pcre`
	`overload`
	`mysql`
	`mbstring`
	`ctype`
	
	`[Zend Modules]`

-i	This command line option calls phpinfo(), and prints out the results. If `PHP` is not working well, it is advisable to make a `php -i` and see if any error messages are printed out before or in place of the information tables. Beware that the output is in `HTML` and therefore quite huge.

Option	Description

-r	This option allows execution of PHP right from within the command line. The PHP start and end tags (<?php and ?>) are **not needed** and will cause a parser errors.
	Megjegyzés: Care has to be taken when using this form of PHP to not collide with command line variable substitution done by the shell.
	Example showing a parser error
	`$ php -r "$foo = get_defined_constants();"`
	`Command line code(1) : Parse error - parse error, unexpected '='`
	The problem here is that the sh/bash performs variable substrution even when using double quotes ". Since the variable $foo is unlikely to be defined, it expands to nothing which results in being the code passed to PHP for executin in fact reads:
	`$ php -r " = get_defined_constants();"`
	The correct way would be to use single quotes '. variables in strings quoted with single quotes are not expanded by sh/bash.
	`$ php -r '$foo = get_defined_constants(); var_dump($`
	`array(370) {`
	`["E_ERROR"]=>`
	`int(1)`
	`["E_WARNING"]=>`
	`int(2)`
	`["E_PARSE"]=>`
	`int(4)`
	`["E_NOTICE"]=>`
	`int(8)`
	`["E_CORE_ERROR"]=>`
	`[...]`
	If you are using a shell different from sh/bash, you might experience further issues. Feel free to open a bug report or send a mail to phpdoc@lists.php.net. One still can easily run intro troubles when trying to get shell variables into the code or using backslashes for escaping. You've been warned.

-h	With this option, you can get information about the actual list of command line options and some one line descriptions about what they do.

The PHP executable can be used to run PHP scripts absolutely independent from the web server. If you are on a Unix system, you should add a special first line to your PHP script, and make it executable, so the system will know, what program should run the script. On a Windows platform you can associate `php.exe` with the double click option of the `.php` files, or you can make a batch file to run the script through PHP. The first line added to the script to work on Unix won't hurt on Windows, so you can write cross platform programs this way. A simple example of writing a command line PHP program can be found below.

Példa 24-1. Script intended to be run from command line (script.php)

```
#!/usr/bin/php
<?php

if ($argc != 2 || in_array($argv[1], array('--help', '-help', '-h', '-?'))) {
?>

This is a command line PHP script with one option.

  Usage:
  <?php echo $argv[0]; ?> <option>

  <option> can be some word you would like
  to print out. With the --help, -help, -h,
  or -? options, you can get this help.

<?php
} else {
    echo $argv[1];
}
?>
```

In the script above, we used the special first line to indicate, that this file should be run by PHP. We work with a CLI version here, so there will be no HTTP header printouts. There are two variables you can use while writing command line applications with PHP: `$argc` and `$argv`. The first is the number of arguments plus one (the name of the script running). The second is an array containing the arguments, starting with the script name as number zero (`$argv[0]`).

In the program above we checked if there are less or more than one arguments. Also if the argument was `--help`, `-help`, `-h` or `-?`, we printed out the help message, printing the script name dynamically. If we received some other argument we echoed that out.

If you would like to run the above script on Unix, you need to make it executable, and simply call it as `script.php echothis` or `script.php -h`. On Windows, you can make a batch file for this task:

Példa 24-2. Batch file to run a command line PHP script (script.bat)

```
@c:\php\php.exe script.php %1 %2 %3 %4
```

Assuming, you named the above program as `script.php`, and you have your `php.exe` in `c:\php\php.exe` this batch file will run it for you with your added options: `script.bat echothis` or `script.bat -h`.

See also the Readline extension documentation for more functions you can use to enhance your command line applications in PHP.

Rész IV. Függvény referencia

I. Apache-specifikus függvények

apache_child_terminate (PHP 4 >= 4.0.5)

terminálja az Apache processzt a kérés után

string **apache_child_terminate** (void) \linebreak

Az **apache_child_terminate()** megszünteti azt az Apache processzt, ami az aktuális PHP kérést futtatja. Ezt pl. sok memóriát fogyasztó processzek leállításra lehet használni, mivel ez memória csak belsőleg szabadul fel, és az operációs rendszer szintjén nem jelenik meg.

Lásd még: exit()!

apache_lookup_uri (PHP 3>= 3.0.4, PHP 4)

Végrehajt egy részleges kérést a meghatározott URI-re és visszatér ennek összes információjával

object **apache_lookup_uri** (string filename) \linebreak

Végrehajt egy részleges kérést a meghatározott URI-re és visszatér ennek összes információjával. Elég messze elmegy ahhoz, hogy megszerezze az összes információt az adott forrásról és visszaadja ezeket egy osztályban. A visszaadott osztály tulajdonságai:
status
the_request
status_line
method
content_type
handler
uri
filename
path_info
args
boundary
no_cache
no_local_copy
allowed
send_bodyct
bytes_sent
byterange
clength
unparsed_uri
mtime
request_time

Megjegyzés: Az **apache_lookup_uri()** csak akkor működik, ha a PHP Apache modulként van telepítve.

apache_note (PHP 3>= 3.0.2, PHP 4)

Apache kérés megjegyzéseket kér és állít be

string **apache_note** (string note_name [, string note_value]) \linebreak

Az **apache_note()** egy Apache-specifikus függvény amely egy kérésben értékeket kér le és állít be a `notes` táblában. Ha egy argumentummal hívod, akkor az aktuális `note_name` megjegyzés értékével tér vissza. Ha két argumentummal hívod, akkor beállítja a `note_name` értékét `note_value`-ra, és a `note_name` korábbi értékével tér vissza.

apache_setenv (PHP 4 >= 4.2.0)

Apache subprocess_env változóit állítja be

int **apache_setenv** (string variable, string value [, bool walk_to_top]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ascii2ebcdic (PHP 3>= 3.0.17)

Átalakít egy stringet ASCII-ből EBCDIC-be

int **ascii2ebcdic** (string ascii_str) \linebreak

Az **ascii2ebcdic()** egy Apache-specifikus függvény, és csak EBCDIC alapú operációs rendszereken érhető el (OS/390, BS2000). Egy ASCII kódolt *`ascii_str`* szöveget alakít át az annak megfelelő EBCDIC reprezentációvá (binary safe módon). Visszatérési értéke az eredmény.

Lásd még az ellentétes funkciót betöltő ebcdic2ascii() függvényt.

ebcdic2ascii (PHP 3>= 3.0.17)

Átalakít egy stringet EBCDIC-ből ASCII-be

int **ebcdic2ascii** (string ebcdic_str) \linebreak

Az **ebcdic2ascii()** egy Apache-specifikus függvény, és csak EBCDIC alapú operációs rendszereken érhető el (OS/390, BS2000). Egy EBCDIC kódolt *`ebcdic_str`* szöveget alakít át az annak megfelelő ASCII reprezentációvá (binary safe módon). Visszatérési értéke az eredmény.

Lásd még az ellentétes funkciót betöltő ascii2ebcdic() függvényt.

getallheaders (PHP 3, PHP 4)

Visszaadja az összes HTTP-kérés fejlécet

array **getallheaders** (void) \linebreak

Ez a függvény az aktuális kérés összes HTTP fejlécének asszociatív tömbjével tér vissza.

> **Megjegyzés:** A környezetből olvasva is lekérhetőek a tipikus CGI változók, és az a módszer mindenképpen működik, attól függetlenül, hogy a PHP-t Apache modulként használod, vagy sem. A phpinfo() használatával meg lehet tekinteni az ezen a módon definiált környezeti változók listáját.

Példa 1. getallheaders() példa

```
$headers = getallheaders();
while (list ($header, $value) = each ($headers)) {
    echo "$header: $value<br>;\n";
}
```

Ez a példa kijelzi az aktuális kéréshez tartozó összes fejlécsort.

> **Megjegyzés:** A **getallheaders()** jelenleg csak abban az esetben használható, ha a PHP Apache modulként fut.

virtual (PHP 3, PHP 4)

Végrehajt egy Apache allekérést

int **virtual** (string filename) \linebreak

A **virtual()** egy Apache-specifikus függvény, amely megegyezik a mod_include <!--#include virtual...--> megoldásával. Végrehajt egy Apache alkérést. Ez hasznos CGI szkriptek vagy .shtml fájlok beillesztésénél, vagy bármi másnál, amit egyébként az Apache dolgozott volna fel. Fontos a CGI szkripteknél, hogy a szkript érvényes CGI fejléceket generáljon. Ezt azt jelenti, hogy minimálisan egy Content-type fejlécet kell ki kell adnia. A PHP fájlokhoz az include() vagy require() függvényeket kell használni. A **virtual()** nem használható olyan dokumentum beillesztéséhez, amely maga is PHP fájl.

II. Tömbkezelő függvények

Ezekkel a függvényekkel különféle módon változtathatók, módosíthatók a tömbök. A tömbök elengedhetetlenek változók sorozatának tárolásához, rendezéséhez, igazgatásához és azok változtatásához.

A PHP az egy- és többdimenziós tömböket is támogatja, amelyeket akár "manuálisan" vagy valamilyen függvénnyel is létre lehet hozni. Vannak bizonyos adatbázis-kezelő függvények, amelyek adatbázis-lekérdezések alapján töltenek fel tömböket, és vannak, amelyek tömböket adnak vissza.

Nézd át a Tömbök fejezetet, ahol részletes leírást található a PHP-s tömbök megvalósításáról és működéséről.

Előfeltételek

Ezek a függvények a standard PHP részei, és így mindig használhatók.

Telepítés

Nincs szükség semmilyen telepítési lépésre, ahhoz hogy használhasd ezeket a függvényeket, mivel a standard csomag részei.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előredefiniált állandók

A `CASE_UPPER` és `CASE_LOWER` az array_change_key_case() függvénnyel használhatók együtt.

Lásd még:

is_array(), explode(), implode(), split() és join()!

array_change_key_case (PHP 4 >= 4.2.0)

visszaad egy kis vagy nagybetűsre cserélt string indexű tömböt

array **array_change_key_case** (array input [, int case]) \linebreak

A **array_change_key_case()** kis vagy nagybetűsre cseréli az *input* tömb string típusú indexeit A változtatást az elhagyható *case* paraméter szabályozza, amelynek két állandót lehet átadni: CASE_UPPER és CASE_LOWER, az alapértelmezés a CASE_LOWER. A numerikus indexet változatlanul hagyja a függvény.

Példa 1. array_change_key_case() példa

```
$bemenet = array("ElSo" => 1, "MasODIk" => 4);
print_r(array_change_key_case($bemenet, CASE_UPPER));
```

A fenti programrészlet a következőt írja ki:

```
Array
(
    [ELSO] => 1
    [MASODIK] => 2
)
```

array_chunk (PHP 4 >= 4.2.0)

feldarabolja a tömböt

array **array_chunk** (array input, int size [, bool preserve_keys]) \linebreak

A **array_chunk()** feloszt egy tömböt több tömbre a *size* értéke alapján. A legutolsó tömbben elképzelhető, hogy ennél kevesebb elem szerepel. A tömböket egy többdimenziós tömb elemeiként adja vissza, amely 0-tól kezdve numerkius indexelt.

A lehetséges *preserve_keys* paraméter TRUE-ra állításával, kikényszeríthető, hogy az eredeti tömb indexelését megtartsák a visszaadott tömbdarabok. Ha e paraméter értéke FALSE, akkor minden tömbdarabban az elemek sorszámozása 0-tól újra kezdődik. Az alapértelmezés az utóbbi: FALSE.

Példa 1. array_chunk() példa

```
$input_array = array('a', 'b', 'c', 'd', 'e');
print_r(array_chunk($input_array, 2));
print_r(array_chunk($input_array, 2, TRUE));
```

A fenti program kimenete:

```
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [0] => c
            [1] => d
        )

    [2] => Array
        (
            [0] => e
        )

)
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [2] => c
            [3] => d
        )

    [2] => Array
        (
            [4] => e
        )

)
```

array_count_values (PHP 4)

összeszámolja minden érték előfordulását a tömbben

array **array_count_values** (array input) \linebreak

Az **array_count_values()** olyan tömböt ad vissza, amelynek a kulcsai az *`input`* tömb értékei és az *`input`* tömbbeli előfordulási gyakoriságuk a hozzájuk tartozó értékek.

Példa 1. array_count_values() példa

```
$tomb = array (1, "hello", 1, "világ", "hello");
print_r(array_count_values ($tomb));
```

A fenti példa kiírja:

```
Array
(
   [1] => 2
   [hello] => 2
   [világ] => 1
)
```

array_diff (PHP 4 >= 4.0.1)

tömbök közti különbséget számolja ki

array **array_diff** (array array1, array array2 [, array ...]) \linebreak

Az **array_diff()** olyan tömböt ad vissza, amely azokat az elemeket tartalmazza, amelyek csak *`array1`*-ben szerepelnek és semelyik másik paraméterként átadott tömbben nem. Az indexelést megőrzi.

Példa 1. array_diff() példa

```
$tomb1 = array ("a" => "zöld", "vörös", "kék", "vörös");
$tomb2 = array ("b" => "zöld", "sárga", "piros");
$eredmeny = array_diff ($tomb1, $tomb2);
```

Ez azt eredményezi, hogy a `$eredmeny` értéke:

```
Array
(
    [2] => kék
)
```

Az `$tomb1`-ben előforduló többszörös értékeket ugyanúgy kezeli a függvény.

> **Megjegyzés:** Két elem akkor és csakis akkor tekinthető egyenlőnek, ha `(string) $elem1 === (string) $elem2`, azaz ha sztring ábrázolásuk megegyezik.

Figyelem

PHP 4.0.4-ben hibásan működött!

Lásd még: array_intersect()!

array_fill (PHP 4 >= 4.2.0)

feltölt egy tömböt egy megadott értékkel

array **array_fill** (int start_index, int num, mixed value) \linebreak

Az **array_fill()** a *num* paraméterben megadott számú elemmel egy tömböt hoz létre, amelyben minden elem *value* értékű és az indexelésük *start_index*-től kezdődik.

Példa 1. array_fill() példa

```
$a = array_fill(5, 6, 'banán');
```

most $a-ban a következő elemek vannak (print_r()):

```
Array
(
    [5]  => banán
    [6]  => banán
    [7]  => banán
    [8]  => banán
    [9]  => banán
    [10] => banán
)
```

array_filter (PHP 4 >= 4.0.6)

megsszűri a tömb elemeit egy függvény visszahívásával

array **array_filter** (array input [, mixed callback]) \linebreak

Az **array_filter()** olyan tömbbel tér vissza, amely az *input* tömb összes ún. 'callback' függvény által megszűrt elemét tartalmazza. Ez azt jelenti, hogy a **array_filter()** az *input* minden elemére

meghívja ezt a függvényt, amely visszatérési értékeként megadja, hogy az elem átment a szűrőn vagy sem. Ha az *`input`* tömb asszociatív, akkor a kulcsok megőrzésre kerülnek.

Példa 1. array_filter() példa

```
function paratlan($var) {
    return ($var % 2 == 1);
}

function paros($var) {
    return ($var % 2 == 0);
}

$tomb1 = array ("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
$tomb2 = array (6, 7, 8, 9, 10, 11, 12);

echo "Páratlan számok:\n";
print_r(array_filter($tomb1, "paratlan"));
echo "Páros számok:\n";
print_r(array_filter($tomb2, "paros"));
```

A fenti program kimenete:

```
Páratlan számok:
Array
(
    [a] => 1
    [c] => 3
    [e] => 5
)
Páros számok:
Array
(
    [0] => 6
    [2] => 8
    [4] => 10
    [6] => 12
)
```

> **Megjegyzés:** A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

A feldolgozandó tömböt a visszahívott függvényen belül nem lehet megváltoztatni, tehát nem lehet egyik vagy másik elemét, esetleg az egész tömböt törölni. Ha ez mégis megtörténik, akkor ennek a függvénynek a viselkedését nem lehet megjósolni.

Lásd még: array_map() és array_reduce()!

array_flip (PHP 4)

felcseréli a tömb elemeit és kulcsait

array **array_flip** (array trans) \linebreak

Az **array_flip()** felcseréli a tömb elemeit és kulcsait egymással, úgyhogy a *`trans`* tömb értékei lesznek az új kulcsok és a kulcsai pedig a hozzátartozó értékek. Látható, hogy array-t ad vissza.

Figyelj arra, hogy a *`trans`* elemértékeinek érvényes kulcsoknak kell lenniük, azaz vagy integer vagy string típusúaknak. Egy figyelmeztető "nem fatális" hibaüzenetet (warning) küld, ha az elemérték típusa nem megfelelő, és a kérdéses kulcs - érték pár *nem lesz felcserélve*.

Ha egy érték többször szerepel, akkor a legutolsó kulcsot fogja használni ahhoz az értékhez, és a többit figyelmen kívül hagyja.

Az **array_flip()** FALSE-tal tér vissza sikertelenség esetén.

Példa 1. array_flip() példa

```
$atalakitva = array_flip ($atalakitva);
$eredeti    = strtr ($str, $atalakitva);
```

Példa 2. array_flip() példa az (érték)ütközésre

```
$atalakitva = array ("a" => 1, "b" => 1, "c" => 2);
$atalakitva = array_flip ($atalakitva);
print_r($atalakitva);
```

most `$atalakitva` értéke:

```
Array
(
    [1] => b
    [2] => c
)
```

array_intersect (PHP 4 >= 4.0.1)

kiszámítja a tömbök metszetét

array **array_intersect** (array array1, array array2 [, array ...]) \linebreak

Az **array_intersect()** olyan tömböt ad vissza, amely az *array1* azon elemeit tartalmazza, amelyek minden paraméterként átadott tömbben megtalálhatók. Az indexelést megőrzi.

Példa 1. array_intersect() példa

```
$tomb1 = array ("a" => "zöld", "vörös", "kék", "vörös");
$tomb2 = array ("b" => "zöld", "sárga", "piros");
$eredmeny = array_intersect ($tomb1, $tomb2);
```

Ez a `$eredmeny` tömbben a következő értékeket képezi:

```
Array
(
    [a] => zöld
    [0] => piros
)
```

Megjegyzés: Két elem akkor és csakis akkor tekinthető egyenlőnek, ha `(string) $elem1 === (string) $elem2`, azaz ha sztring ábrázolásuk megegyezik.

Figyelem

PHP 4.0.4-ben hibásan működött!

Lásd még: array_diff()!

array_key_exists (PHP 4 >= 4.1.0)

ellenőrzi a keresett index/kulcs létezését a tömbben

bool **array_key_exists** (mixed key, array search) \linebreak

Az **array_key_exists()** `TRUE`-val tér vissza, ha az adott *key* be van állítva a *search* tömbben. A *key* bármilyen lehetséges tömbindex érték lehet.

Példa 1. array_key_exists() példa

```
$keress = array("első" => 1, "második" => 4);
if ( array_key_exists("első", $keress) ) {
    echo "Az 'első' indexű elem a tömbben van.";
```

```
}
```

Megjegyzés: A PHP 4.0.6-ban a függvény neve **key_exists()** volt.

Lásd még: isset()!

array_keys (PHP 4)

visszaadja egy tömb összes indexét

array **array_keys** (array input [, mixed search_value]) \linebreak

Az **array_keys()** az *input* tömb összes - numerikus és sztring - típusú indexét (kulcsát) adja vissza.

Ha az elhagyható *search_value* is meg an adva, akkor csak azokkal a kulcsokkal tér vissza, amelyek a megadott értékhez tartoznak. Egyébként az *input* összes kulcsa átadásra kerül.

Példa 1. array_keys() példa

```
$tomb = array (0 => 100, "szin" => "piros");
print_r(array_keys ($tomb));

$tomb = array ("kék", "piros", "zöld", "kék", "kék");
print_r(array_keys ($tomb, "kék"));

$tomb = array ("szín" => array("kék", "piros", "zöld"), "méret" => array("kicsi", "közepe
print_r(array_keys ($tomb));
```

A fenti program kimenete:

```
Array
(
  [0] => 0
  [1] => szín
)
Array
(
  [0] => 0
  [1] => 3
  [2] => 4
)
Array
(
  [0] => szín
  [1] => méret
)
```

Megjegyzés: Ez a függvény PHP 4-től használható, alább látható egy megvalósítása azok számára, akik még mindig PHP 3-t használnak.

Példa 2. array_keys() megvalósítása PHP 3-ban

```
function array_keys ($arr, $term="") {
    $t = array();
    while ( list($k,$v) = each($arr) ) {
        if ( $term && $v != $term ) {
            continue;
        }
        $t[] = $k;
    }
    return $t;
}
```

Lásd még: array_values()!

array_map (PHP 4 >= 4.0.6)

egy függvényt alkalmaz minden megadott tömb elemére

array **array_map** (mixed callback, array array1 [, array array2...]) \linebreak

Az **array_map()** olyan tömböt ad vissza, amelyben az *array1* tömb *callback* nevű függvény által kiszámolt értékei vannak. A *callback* függvény által fogadható paraméterek száma meg kell egyezzen az **array_map()**-nak átadott tömbök számával.

Példa 1. array_map() példa

```
function kob($n) {
    return $n*$n*$n;
}

$a = array(1, 2, 3, 4, 5);
$b = array_map("kob", $a);
```

Így a $b tömb értéke:

```
Array
(
  [0] => 1
  [1] => 8
  [2] => 27
  [3] => 64
  [4] => 125
)
```

Példa 2. array_map() - több tömb használata

```
function ird_spanyolul($n, $m) {
    return "A(z) $n spanyolul: $m ";
}

function osszerendel_spanyolul($n, $m) {
    return array ($n => $m);
}

$a = array(1, 2, 3, 4, 5);
$b = array("uno", "dos", "tres", "cuatro", "cinco");

$c = array_map("ird_spanyolul", $a, $b);
print_r($c);

$d = array_map("osszerendel_spanyolul", $a , $b);
print_r($d);
```

This results:

```
// $c szerkezete
Array
(
  [0] => A(z) 1 spanyolul: uno
  [1] => A(z) 2 spanyolul: dos
  [2] => A(z) 3 spanyolul: tres
  [3] => A(z) 4 spanyolul: cuatro
  [4] => A(z) 5 spanyolul: cinco
)

// $d szerkezete
Array
(
  [0] => Array
      (
          [1] => uno
      )

  [1] => Array
```

```
        (
            [2] => dos
        )

    [2] => Array
        (
            [3] => tres
        )

    [3] => Array
        (
            [4] => cuatro
        )

    [4] => Array
        (
            [5] => cinco
        )

)
```

Több tömb használata esetén a tömbök hosszának meg kell egyeznie, mert a meghívott `callback` függvény párhuzomsan halad végig rajtuk mindig a megfelelő elemeket használva fel. Ha a tömbök nem megegyező méretűek, akkor a legrövidebbek üres elemekkel lesznek feltöltve.

Ennek a függvénynek érdekes felhasználási módja, hogy tömböket tartalmazó tömböt lehet vele létrehozni, ha a `callback` függvényként a `NULL` értéket adjuk át.

Példa 3. tömböket tartalmazó tömb létrehozása

```
$a = array(1, 2, 3, 4, 5);
$b = array("egy", "kettő", "három", "négy", "öt");
$c = array("uno", "dos", "tres", "cuatro", "cinco");

$d = array_map(null, $a, $b, $c);
print_r($d);
```

A fenti program kimenete, a `$d` tömb szerkezete:

```
Array
(
    [0] => Array
        (
            [0] => 1
            [1] => egy
            [2] => uno
        )

    [1] => Array
```

```
            (
                [0] => 2
                [1] => kettő
                [2] => dos
            )

        [2] => Array
            (
                [0] => 3
                [1] => három
                [2] => tres
            )

        [3] => Array
            (
                [0] => 4
                [1] => négy
                [2] => cuatro
            )

        [4] => Array
            (
                [0] => 5
                [1] => öt
                [2] => cinco
            )

    )
```

Lásd még: array_filter() és array_reduce()!

array_merge_recursive (PHP 4 >= 4.0.1)

rekurzívan egyesít két vagy több tömböt

array **array_merge_recursive** (array array1, array array2 [, array ...]) \linebreak

Az **array_merge_recursive()** egyesíti két vagy több tömb elemeit úgy, hogy az egyik elemeit a másik után fűzi, és visszadja az eredményül kapott tömböt.

Ha a megadott tömbökben ugyanolyan string kulcsok vannak, akkor az ezekhez tartozó elemértékeket egy tömbbe gyűjti össze. Ezt rekurzívan teszi a függvény, tehát ha az elemek egyike ugyancsak tömb, akkor ennek a tömbnek az elemeire is hasonlóképp lefut az egyesítő algoritmus. Ha viszont a numerikus indexek egyeznek meg, akkor a későbbi nem felülírja a korábbi értéket, hanem azok után fűzi.

Példa 1. array_merge_recursive() példa

```
$tomb1 = array ("szín" => array ("kedvenc" => "piros"), 5);
$tomb2 = array (10, "szín" => array ("kedvenc" => "zöld", "kék"));
$eredmeny = array_merge_recursive ($tomb1, $tomb2);
```

Az `$eredmeny` tömb értéke:

```
Array
(
  [szín] => Array
      (
          [kedvenc] => Array
              (
                  [0] => piros
                  [1] => zöld
              )

          [0] => kék
      )

  [0] => 5
  [1] => 10
)
```

Lásd még: array_merge()!

array_merge (PHP 4)

egyesít két vagy több tömböt

array **array_merge** (array array1, array array2 [, array ...]) \linebreak

Az **array_merge()** egyesíti két vagy több tömb elemeit úgy, hogy a második tömb elemeit hozzáfűzi az elsőéhez, és visszadja az eredményül kapott tömböt.

Ha a megadott tömbökben ugyanolyan string kulcsok vannak, akkor a későbbi felülírja az előzőt. Ha a tömböknek ugyanolyan numerikus indexei vannak, akkor nem ez történik, hanem mindenegyiket új értékként a tömbbe illeszt.

Példa 1. array_merge() példa

```
$tomb1 = array ("szín" => "piros", 2, 4);
$tomb2 = array ("a", "b", "szín" => "zöld", "alak" => "trapéz", 4);
$eredmeny = array_merge ($tomb1, $tomb2);
```

Az `$eredmeny` tömb értéke:

```
Array
(
    [szín] => zöld
    [0] => 2
    [1] => 4
    [2] => a
    [3] => b
    [alak] => trapéz
    [4] => 4
)
```

Lásd még: array_merge_recursive()!

array_multisort (PHP 4)

egyszerre több tömböt vagy egy többdimenziós tömböt rendez

bool **array_multisort** (array ar1 [, mixed arg [, mixed ... [, array ...]]]) \linebreak

Az **array_multisort()** arra használható, hogy egyszerre több tömböt vagy egy többdimenzós tömböt rendezzen sorba valamelyik dimenziója szerint. A kulcsokat megőrzi a rendezés során.

A megadott tömböket egy képzeletbeli, sorok alapján rendezendő táblázat oszlopainak tekinthetők. Ez hasonlít az SQL ORDER BY cikkelyének feladatára. Ez első tömb, ami alapján először rendez. Az ebben a tömbben levő ugyanolyan sorok (elemek) a soron következő tömb szerint lesznek tovább rendezve, és így tovább végig a tömbökön.

Ennek a függvénynek a argumentumlistája kicsit szokatlan, de nagyon rugalmas. A legelső paraméternek mindig tömbnek kell lennie, ezután minden paraméter lehet egy tömb vagy egy a következő rendezést szabályzó jelzőértékek közül.

Sorrendi jelzők:

- SORT_ASC - növekvő sorrendbe rendez
- SORT_DESC - csökkenő sorrendbe rendez

Rendezési jelzők:

- SORT_REGULAR - megszokott módon hasonlítja össze az elemeket
- SORT_NUMERIC - számként hasonlítja össze az elemeket
- SORT_STRING - szövegként hasonlítja össze az elemeket

Nem lehet két ugyanolyan típusú jelzőt egymás után megadni. Ezek a jelzők csak a közvetlenül előttük álló tömb szerinti rendezést befolyásolják, minden tömb paraméternél az alapértelmezés lép életbe: SORT_ASC és SORT_REGULAR.

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

Példa 1. Több tömb alapján történő rendezés

```
$ar1 = array ("10", 100, 100, "a");
$ar2 = array (1, 3, "2", 1);
array_multisort ($ar1, $ar2);
```

A rendezés után a `$ar1` és a `$ar2`: (A második tömb elemei az első tömb megegyező elemeinek megfelelően (100, 100) szintén rendezettek.)

```
Array // $ar1
(
    [0] => 10
    [1] => a
    [2] => 100
    [3] => 100
)
Array // $ar2
(
    [0] => 1
    [1] => 1
    [2] => 2
    [3] => 3
)
```

Példa 2. Többdimenziós tömbök rendezése

```
$ar = array (array ("10", 100, 100, "a"), array (1, 3, "2", 1));
array_multisort ($ar[0], SORT_ASC, SORT_STRING,
                 $ar[1], SORT_NUMERIC, SORT_DESC);
```

A rendezés után a `$ar` tömb: (Az első szövegként növekvő sorrendbe rendezve, a másodikban számként csökkenő sorrendbe rendezve.)

```
Array
(
    [0] => Array
        (
            [0] => 10
            [1] => 100
            [2] => 100
```

```
            [3] => a
        )

    [1] => Array
        (
            [0] => 1
            [1] => 3
            [2] => 2
            [3] => 1
        )

)
```

array_pad (PHP 4)

adott méretűre bővít és tölt fel egy tömböt

array **array_pad** (array input, int pad_size, mixed pad_value) \linebreak

Az **array_pad()** az *`input`* tömb *`pad_size`* paraméterben megadott méretre bővített másolatát adja vissza, amelyben az új elemeket a *`pad_value`* értékkel töltötte fel. Ha a *`pad_size`* pozitív szám, akkor a tömb végére kerülnek az új elemek, ha negatív, akkor az elejére. Ha a *`pad_size`* abszolút értéke kisebb vagy egyenlő a tömb méretével, akkor az *`input`* tömbön nem történik változtatás.

Példa 1. array_pad() példa

```
$bemenet  = array (12, 10, 9);

$eredmeny = array_pad ($bemenet, 5, 0);
// array (12, 10, 9, 0, 0)

$eredmeny = array_pad ($bemenet, -7, -1);
// array (-1, -1, -1, -1, 12, 10, 9)

$eredmeny = array_pad ($bemenet, 2, "noop");
// nincs kitöltés
```

array_pop (PHP 4)

kivesz egy elemet a tömb végéről

mixed **array_pop** (array array) \linebreak

Az **array_pop()** kiveszi az *array* tömbből annak utolsó elemét, és ezt adja vissza. Ha az *array* tömb üres vagy nem tömb, akkor NULL-lal tér vissza.

Példa 1. array_pop() példa

```
$verem = array ("narancs", "banán", "alma", "málna");
$gyumolcs = array_pop ($verem);
```

Ezután a $verem csak 3 eleme lesz:

```
Array
(
    [0] => narancs
    [1] => banán
    [2] => alma
)
```

és a $gyumolcs értéke málna lesz.

> **Figyelem**
>
> Ez a függvény a logikai típusú FALSE értékkel is visszatérhet, olyan nem logikai típusú értékek mellett, amik szintén FALSE értékként jelenhetnek meg (mint pl. 0 vagy ""). Olvasd el a Logikai típusról szóló részt további információkért. Használd a === operátort a visszatérési érték ellenőrzésére.

Lásd még: array_push(), array_shift() és array_unshift()!

array_push (PHP 4)

egy vagy több elemet illeszt a tömb végére

int **array_push** (array array, mixed var [, mixed ...]) \linebreak

Az **array_push()** az *array* tömböt veremként kezeli, és annak a tetejére - a tömb végére - teszi a paraméterként megadott értékeket. Az *array* tömb mérete a hozzáfűzött elemek számával fog nőni. Ennek ugyanaz a hatása, mintha a :

```
$tomb[] = $var;
```

kódot ismételnéd meg mindenegyes *var* paraméterre.

A tömbbeli elemek új számával tér vissza.

Példa 1. array_push() példa

```
$verem = array ("narancs", "banán");
array_push ($verem, "alma", "málna");
```

E példában a `$verem` tömbnek 4 eleme lesz:

```
Array
(
    [0] => narancs
    [1] => banán
    [2] => alma
    [3] => málna
)
```

Lásd még: array_pop(), array_shift() és array_unshift()!

array_rand (PHP 4)

véletlenszerűen egy vagy több elemet hoz elő a tömbből

mixed **array_rand** (array input [, int num_req]) \linebreak

Az **array_rand()** különösen akkor hasznos, ha egy vagy több tömbbeli elemet kell véletlenszerűen kiválasztani. A függvény az *`input`* tömbből az elhagyható *`num_req`* paraméterben megadott számú elemet választja ki és adja vissza. A kívánt elemek száma alapértelmezés szerint 1.

Ha csak egy elemet választ ki, akkor az **array_rand()** ennek az elemnek az indexével tér vissza. Több elem esetén ezeknek az indexeiből alkotott tömbbel. Ezzel a megoldással kulcsokat és elemértékeket is véletlenszerűen ki lehet választani egy tömbből.

Az srand() függvényt meg kell hívni, hogy inicializálja a véletlenszám-generátort.

Példa 1. array_rand() példa

```
srand ((float) microtime() * 10000000);
$input = array ("Neo", "Morpheus", "Trinity", "Cypher", "Tank");
$rand_keys = array_rand ($input, 2);
print $input[$rand_keys[0]]."\n";
print $input[$rand_keys[1]]."\n";
```

array_reduce (PHP 4 >= 4.0.5)

elemi értékre csökkenti a tömböt tetszőleges függvény meghívásával

mixed **array_reduce** (array input, mixed callback [, int initial]) \linebreak

Az **array_reduce()** ismétlődve meghívja a *`callback`* függvényt paraméterként az *`input`* tömb elemeivel, hogy végül egy elemi értékké alakítsa át a tömböt. Az elhagyható *`intial`* megadásával beállítható egy tetszőleges kezdő érték a folyamat legelején. Ha üres a tömb, akkor ez lesz a visszadott érték is.

Példa 1. array_reduce() példa

```
function rsum($v, $w) {
    $v += $w;
    return $v;
}

function rmul($v, $w) {
    $v *= $w;
    return $v;
}

$a = array(1, 2, 3, 4, 5);
$x = array();
$b = array_reduce($a, "rsum");
$c = array_reduce($a, "rmul", 10);
$d = array_reduce($x, "rsum", 1);
```

Ennek eredménye: `$b` = `15`, `$c` = `1200` (= 1*2*3*4*5*10) és `$d` = `1`.

Lásd még: array_filter() és array_map()!

array_reverse (PHP 4)

tömb fordított sorrendű elemeivel tér vissza

array **array_reverse** (array array [, bool preserve_keys]) \linebreak

Az **array_reverse()** veszi az *`array`* tömböt, és visszaad egy olyan tömböt, amelyben elemek fordított sorrendben szerepelnek. Ha *`preserve_keys`* értéke `TRUE`, akkor megőrzi az elemek indexelését a visszaadandó tömbben.

Példa 1. array_reverse() példa

```
$bemenet  = array ("php", 4.0, array ("zöld", "piros"));
$eredmeny = array_reverse ($bemenet);
$indexelt_eredmeny = array_reverse ($bemenet, TRUE);
```

Ezután az `$eredmeny` és a `$indexelt_eredmeny` tömbnek ugyanazok az elemei lesznek, de az *indexelésük eltérő*. A `$eredmeny` és a `$indexelt_bemenet` így néz ki:

```
Array
(
  [0] => Array
      (
          [0] => zöld
          [1] => kék
      )

  [1] => 4
  [2] => php
)
Array
(
  [2] => Array
      (
          [0] => zöld
          [1] => kék
      )

  [1] => 4
  [0] => php
)
```

Megjegyzés: A második paraméter PHP 4.0.3-től használható.

array_search (PHP 4 >= 4.0.5)

adott elemet keres a tömbben és az indexével tér vissza

mixed **array_search** (mixed needle, array haystack [, bool strict]) \linebreak

Átnézi a *`haystack`* tömböt a *`needle`* elemet keresve, ha megtalálta, akkor visszaadja az indexét/kulcsát, egyébként `FALSE`-ot.

Ha a harmadik, *`strict`* paraméter értéke `TRUE`, akkor az **array_search()** a *`needle`* típusát is összeveti a *`haystack`* tömb elemeiével.

Megjegyzés: PHP 4.2.0 előtti verziókban hiba esetén az **array_search()** `NULL` adott vissza `FALSE` helyett.

> **Figyelem**
>
> Ez a függvény a logikai típusú `FALSE` értékkel is visszatérhet, olyan nem logikai típusú értékek mellett, amik szintén `FALSE` értékként jelenhetnek meg (mint pl. `0` vagy ""). Olvasd el a Logikai típusról szóló részt további információkért. Használd a === operátort a visszatérési érték ellenőrzésére.

Lásd még: array_keys() és in_array()!

array_shift (PHP 4)

kivesz egy elemet a tömb elejéről

mixed **array_shift** (array array) \linebreak

Az **array_shift()** kilépteti az *`array`* tömb első elemét, és ezt adja vissza. Az *`array`* tömb eggyel kevesebb elemet fog tartalmazni, és minden eleme eggyel előrébb (első index felé) tolódik. Ha a tömb üres volt, akkor `NULL`-lal tér vissza.

Példa 1. array_shift() példa

```
$sor      = array ("narancs", "banán", "alma", "málna");
$gyumolcs = array_shift ($sor);
```

Ez azt eredményezi, hogy `$sor` tömbnek 3 eleme marad:

```
Array
(
    [0] => banán
    [1] => alma
    [2] => málna
)
```

és a `$gyumolcs` értéke "narancs" lesz.

Lásd még: array_unshift(), array_push() és array_pop()!

array_slice (PHP 4)

a tömb egy részével tér vissza

array **array_slice** (array array, int offset [, int length]) \linebreak

Az **array_slice()** függvény az *array* tömb *offset* és *length* paraméterek által meghatározott elemsorozatával tér vissza.

Pozitív *offset* esetén, az elemsorozat ennyiedik sorszámú elemtől kezdődik, negatív *offset* esetén az *array* tömb végétől visszafelé számolva ennyiedik elemtől kezdődik.

Ha a *length* meg van adva és pozitív, akkor az elemsorozatba ennyi darab elem kerül bele. Negatív *length* esetén az elemsorozat az *array* tömb vége előtt ennyi darab elemmel "áll meg", amely azt jelenti, hogy az ezután következő elemek nem kerülnek bele a sorozatba. Ha nincs megadva a *length*, akkor az *offset* által meghatározott ponttól kezdve a tömb végéig minden elem belekerül a sorozatba.

Megjegyzés: Az **array_slice()** figyelmen kívül hagyja a tömb aktuális indexelését, és az *offset*-et és a *length*-et az elemek pillanatnyi elhelyezkedése alapján számolja.

Példa 1. array_slice() példák

```
$input = array ("a", "b", "c", "d", "e");
                                        // visszatérési értékek
$output = array_slice ($input, 2);      // "c", "d", "e"
$output = array_slice ($input, 2, -1);  // "c", "d"
$output = array_slice ($input, -2, 1);  // "d"
$output = array_slice ($input, 0, 3);   // "a", "b", "c"
```

Lásd még: array_splice()!

array_splice (PHP 4)

lecseréli a tömb egy részét

array **array_splice** (array input, int offset [, int length [, array replacement]]) \linebreak

Az **array_splice()** eltávolítja az *input* tömb *offset* és *length* paraméterek által meghatározott sorozatát, és helyébe - ha meg van adva - a *replacement* tömböt illeszti. Az eltávolított elemek tömbjével tér vissza.

Pozitív *offset* esetén, a törlendő elemsorozat ennyiedik sorszámú elemtől kezdődik, negatív *offset* esetén az *array* tömb végétől visszafelé számolva ennyiedik elemtől kezdődik.

Ha nincs megadva a *length*, akkor az *offset* által meghatározott ponttól kezdve a tömb végéig minden elemet eltávolít. Ha a *length* meg van adva és pozitív, akkor ennyi darab elem kerül

törlésre. Negatív *`length`* esetén az *`array`* tömb végétől visszafelé ennyiedik elemig törlődik minden. Jótanácsként: ha a tömb végéig kell mindent lecserélni a *`replacement`*-re, akkor a `count($input)`-ot kell a *`length`* paraméterben megadni.

Ha a *`replacement`* tömb is meg van adva, akkor a törölt elemek helyére ennek a tömbnek az elemei kerülnek. Ha *`offset`* és *`length`* alapján nem történt törlés, akkor a *`replacement`* elemei az *`offset`* által mutatott helyre kerülnek beillesztésre. Jótanácsként: ha a *`replacement`* egyelemű, akkor nem szükséges `array()`-t használni, kivéve ha ez az egy elem maga is tömb kell legyen.

A következők egyenértékűek:

```
array_push ($bemenet, $x, $y)     array_splice ($bemenet, count ($bemenet), 0, ar-
ray ($x, $y))
array_pop ($bemenet)              array_splice ($bemenet, -1)
array_shift ($bemenet)            array_splice ($bemenet, 0, 1)
array_unshift ($bemenet, $x, $y)  array_splice ($bemenet, 0, 0, array ($x, $y))
$a[$x] = $y                       array_splice ($bemenet, $x, 1, $y)
```

A törölt elemek tömbjével tér vissza a függvény.

Példa 1. array_splice() példák

```
$bemenet = array ("piros", "zöld", "kék", "sárga");
array_splice ($bemenet, 2);
// $bemenet most: array ("piros", "zöld")

$bemenet = array ("piros", "zöld", "kék", "sárga");
array_splice ($bemenet, 1, -1);
// $bemenet most: array ("piros", "sárga")

$bemenet = array ("piros", "zöld", "kék", "sárga");
array_splice ($bemenet, 1, count($bemenet), "narancs");
// $bemenet most:  array ("piros", "narancs")

$bemenet = array ("piros", "zöld", "kék", "sárga");
array_splice ($bemenet, -1, 1, array("fekete", "gesztenye"));
// $bemenet most:  array ("piros", "zöld", "kék", "fekete", "gesztenye")
```

Lásd még: array_slice()!

array_sum (PHP 4 >= 4.0.4)

tömbbeli elemek összegét számolja ki

mixed **array_sum** (array array) \linebreak

Az **array_sum()** az *array* tömb elemeinek integer vagy float típusú értékével tér vissza.

Példa 1. array_sum() példák

```
$a = array(2, 4, 6, 8);
echo "összeg(a) = ".array_sum($a);  // kiírja: összeg(a) = 20

$b = array("a"=>1.2,"b"=>2.3,"c"=>3.4);
echo "összeg(b) = ".array_sum($b);  // kiírja: összeg(b) = 6.9
```

Megjegyzés: A 4.0.6 verziót megelőzően a PHP módosította a paraméterként átadott tömböt, a sztringeket számmá konvertálta, ami majdnem mindig 0-val volt egyenlő.

array_unique (PHP 4 >= 4.0.1)

törli az ismétlődő elemeket

array **array_unique** (array array) \linebreak

Az **array_unique()** a bemenetként kapott *array* tömbből minden értéket egyszer vesz, és az így keletkezett tömbbel tér vissza.

Az indexelést megtartja. Az **array_unique()** először sorba rendezi a tömbértékeket, majd az összetartozó értékek közül az elsőként megtalált elem indexét tartja meg, a többi ezután következőt eldobja. Ez nem azt jelenti, hogy az eredeti tömbbeli legelső elem indexét tartja meg.

Megjegyzés: Két elem akkor és csakis akkor tekinthető egyenlőnek, ha `(string) $elem1 === (string) $elem2`, azaz ha sztring ábrázolásuk megegyezik.

Figyelem

PHP 4.0.4-ben hibásan működött.

Példa 1. array_unique() példa

```
$bemenet  = array ("a" => "zöld", "piros", "b" => "zöld", "kék", "piros");
$eredmeny = array_unique ($bemenet);
print_r($eredmeny);
```

A következőt írja ki: (PHP 4.0.6)

```
Array
(
    [b] => zöld
    [1] => kék
    [2] => piros
)
```

Példa 2. array_unique() és a típusok kapcsolata

```
$bemenet  = array (4,"4","3",4,3,"3");
$eredmeny = array_unique ($bemenet);
var_dump($eredmeny);
```

A következőt írja ki: (PHP 4.0.6)

```
array(2) {
  [3]=>
  int(4)
  [4]=>
  int(3)
}
```

array_unshift (PHP 4)

beszúr a tömb elejére egy vagy több elemet

int **array_unshift** (array array, mixed var [, mixed ...]) \linebreak

Az **array_unshift()** az *array* tömb után megadott elemeket a tömb elejébe beszúrja. A beszúrandó elemeket listaként kezeli a függvény, így egymáshoz képesti sorrendjük nem változik meg.

Az *array* elemeinek új számával tér vissza.

Példa 1. array_unshift() példa

```
$sor      = array ("narancs", "banán");
$gyumolcs = array_shift ($sor, "alma", "málna");
```

Ez azt eredményezi, hogy a `$sor` tömbnek 4 eleme lesz:

```
Array
(
    [0] => alma
    [1] => málna
    [2] => narancs
    [3] => banán
)
```

Lásd még: array_shift(), array_push() és array_pop()!

array_values (PHP 4)

visszadja az összes tömbben előforduló értéket

array **array_values** (array input) \linebreak

Az **array_values()** visszadja az összes *`input`* tömbben előforduló értéket.

Példa 1. array_values() példa

```
$tomb = array ("méret" => "XL", "szín" => "arany");
array_values ($tomb);
```

A fenti program kimenete:

```
Array
(
    [0] => XL
    [1] => arany
)
```

Megjegyzés: Ez a függvény PHP 4-től hozzáférhető, a PHP 3-at használók számára itt van egy lehetséges megvalósítás:

Példa 2. array_values() megvalósítása PHP 3-ban

```
function array_values ($arr) {
    $t = array();
    while (list($k, $v) = each ($arr)) {
        $t[] = $v;
    }
    return $t;
}
```

Lásd még: array_keys()!

array_walk (PHP 3>= 3.0.3, PHP 4)

felhasználói függvényt futtat a tömb minden elemén

int **array_walk** (array array, string func [, mixed userdata]) \linebreak

A *func* paraméterben megadott nevű felhasználói függvényt alkalmazza az *array* tömb minden elemére. A *func* függvény hívásakor első paraméterként a tömbérték, másodikként az indexe kerül átadásra. Ha a *userdata* is szerepel, akkor ez lesz a harmadik paraméter. A *func* nevű függvénynek a felhasználó által definiáltnak kell lennie, és nem lehet natív PHP függvény. Emiatt nem lehet az **array_walk()**-nak például közvetlenül a **str2lower()**-t meghívni, hanem először létre kell hozni egy saját függvényt, majd ennek átadni a paramétereket.

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

Ha *func* két vagy három argumentumnál többet igényel a *userdata*-tól függően, akkor figyelmeztető hibaüzenet (warning) keletkezik minden alkalommal, amikor az **array_walk()** meghívja a *func* függvényt. Ezeket a figyelmeztetések elnyomhatók a '@' karaktert az **array_walk()** elé írva vagy az error_reporting() használatával.

Megjegyzés: Ha a *func* függvénynek a tényleges tömbértékekkel kell dolgoznia, akkor a *func* függvény első paraméterét referenciaként kell átadni. Ezután az ezen a változón végzett változtatás az eredeti tömbben is változást idéz elő.

Az *array* tömb *func* függvényen belüli módosítása megjósolhatatlan viselkedést eredményezhet.

Megjegyzés: Az index és a userdata értékek átadása a *func* függvénynek a 4.0 verziótól elérhető.

PHP 4-ben reset() függvényt szükség szerint meg kell hívni, mivel az **array_walk()** nem inicializálja a tömböt alapértelmezés szerint.

A feldolgozandó tömböt a visszahívott függvényen belül nem lehet megváltoztatni, tehát nem lehet egyik vagy másik elemét, esetleg az egész tömböt törölni. Ha ez mégis megtörténik, akkor ennek a függvénynek a viselkedését nem lehet megjósolni.

Példa 1. array_walk() példa

```
$gyumolcsok = array ("d"=>"citrom", "a"=>"narancs", "b"=>"banán", "c"=>"alma");

function teszt_atir (&$elem1, $index, $prefix) {
    $elem1 = "$prefix: $elem1";
}

function teszt_kiir ($elem2, $index) {
    echo "$index. $elem2<br>\n";
}
echo "Előtte ...:\n";
array_walk ($gyumolcsok, 'test_kiir');
reset ($gyumolcsok);
array_walk ($gyumolcsok, 'test_atir', 'gyumolcs');
echo "... és utána:\n";
reset ($gyumolcsok);
array_walk ($gyumolcsok, 'test_kiir');
```

A fenti program kimenete:

```
Előtte ...:
d. citrom
a. narancs
b. banán
c. alma
... és utána:
d. gyumolcs: citrom
a. gyumolcs: narancs
b. gyumolcs: banán
c. gyumolcs: alma
```

Lásd még: each() és list()!

array (unknown)

létrehoz egy tömböt

array **array** ([mixed ...]) \linebreak

A paramétereként átadott értékekből képzett tömböt ad vissza, az elemekhez az indexük is hozzárendelhető a => operátorral.

> **Megjegyzés:** Az **array()** nyelvi építőelem tömbök létrehozására, és nem szokványos függvény.

Az "index => érték" szintaxis - egymástól vesszővel elválasztva - meghatároz egy indexet a hozzátartozó értékkel. Az index lehet sztring vagy egész szám is. Ha nincs index megadva, akkor egy automatikusan generált egész szám lesz az elem indexe. (Az automatikus indexelés 0-tól kezdődik.) Ha az index egy egész szám, akkor a következőként generált index az eddigi legnagyobb egész számú indextől eggyel nagyobb szám lesz. Figyelj arra, hogy ha két azonos indexű elemet definiálsz, akkor az utolsó felülírja az elsőt.

A következő példák segítenek megérteni, hogyan lehet kétdimenziós tömböket létrehozni, asszociatív tömbök kulcsait megadni, illetve hogyan lehet helyeket kihagyni és folytatni numerikus indexelést.

Példa 1. array() példa

```
$gyumolcsok = array (
    "gyümölcsök"  => array ("a"=>"narancs", "b"=>"banán", "c"=>"alma"),
    "számok"      => array (1, 2, 3, 4, 5, 6),
    "lyukak"      => array ("első", 5 => "második", "harmadik")
);
```

Példa 2. Automatikus indexelés az array() "függvénnyel"

```
$tomb = array( 1, 1, 1, 1,  1, 8=>1,  4=>1, 19, 3=>13);
print_r($tomb);
```

ezt fogja kiírni:

```
Array
(
    [0] => 1
    [1] => 1
    [2] => 1
    [3] => 13
    [4] => 1
    [8] => 1
    [9] => 19
)
```

Figyeld meg, hogy a 3. indexű elemet kétszer definiáltuk, és a végső értéke a 13. A 4. elemet a 8. után hoztuk létre, és a következő elemnek (19-nek) generált index értéke 9, mivel a legnagyobb létező index a tömbben a 8 volt.

Ez a példa egy olyan tömböt hoz létre, amelynek az indexelése 1-től kezdődik.

Példa 3. 1-től indexelt tömb létrehozása array() "függvénnyel"

```
$elsonegyedev  = array(1 => ’Január’, ’Február’, ’Március’);
print_r($elsonegyedev);
```

ezt írja ki:

```
Array
(
    [1] => ’Január’
    [2] => ’Február’
    [3] => ’Március’
)
```

Lásd még: array_pad(), list() és range()!

arsort (PHP 3, PHP 4)

fordított sorrendben rendez egy tömböt megtartva az indextársítást

void **arsort** (array array [, int sort_flags]) \linebreak

Ez a függvény úgy rendezi az *array* tömböt, hogy az indexek megtartják kapcsolatukat a hozzájuk rendelt tömb elemeivel. Főként akkor használják, ha az asszociatív tömbök rendezésénél az aktuális elemsorrend is fontos.

Példa 1. arsort() példa

```
$gyumolcsok = array ("d"=>"citrom", "a"=>"narancs", "b"=>"banán", "c"=>"alma");
arsort ($gyumolcsok);
 reset ($gyumolcsok);
while (list ($key, $val) = each ($gyumolcsok)) {
    echo "$key = $val\n";
}
```

Ez a példa a következőt írja ki:

```
a = narancs
d = citrom
b = banán
c = alma
```

A gyümölcsök az abc szerint fordított sorrendben vannak rendezve, és az indexek az eredeti elemekre mutatnak.

A rendezést befolyásolja a *`sort_flags`* paraméter, részletesebben lásd a sort() függvénynél!

Lásd még: asort(), rsort(), ksort() és sort()!

asort (PHP 3, PHP 4)

tömb elemeit rendezi megtartva az indextársítást

void **asort** (array array [, int sort_flags]) \linebreak

Ez a függvény úgy rendezi az *`array`* tömböt, hogy az indexek megtartják kapcsolatukat a hozzájuk rendelt tömbelemekkel. Főként akkor használják, ha az asszociatív tömbök rendezésénél az aktuális elemsorrend is fontos.

Példa 1. asort() példa

```
$gyumolcsok = array ("d"=>"citrom", "a"=>"narancs", "b"=>"banán", "c"=>"alma");
asort ($gyumolcsok);
reset ($gyumolcsok);
while (list ($key, $val) = each ($gyumolcsok)) {
    echo "$key = $val\n";
}
```

Ez a példa a következőt írja ki:

```
c = alma
b = banán
d = citrom
a = narancs
```

A gyümölcsök az abc-nek megfelelően növekvő sorrendben vannak rendezve, és az indexek az eredeti elemekre mutatnak.

A rendezést befolyásolja a *`sort_flags`* paraméter, részletesebben lásd a sort() függvénynél!

Lásd még: arsort(), rsort(), ksort() és sort()!

compact (PHP 4)

tömbbe tömöríti a változókat és értékeiket

array **compact** (mixed varname [, mixed ...]) \linebreak

A **compact()** változó számú paramétert fogad el. Minden paraméter lehet egy változónevet tartalmazó sztring vagy akár tömb is. A tömbben lehetnek újabb tömbök is, amelyek a változók neveit tárolják, a **compact()** rekurzívan kezeli ezt.

A **compact()** a neve alapján megkeresi a változót az aktuális szimbólumtáblában, és hozzáadja a kimeneti tömbhöz. A változó neve lesz a kulcs és a kulcshoz tartozó tömbérték pedig a változó értéke. Röviden, az extract()-tal ellentétesen működik. Az összes hozzáadott változót tároló kimeneti tömbbel tér vissza.

Az a sztring, amelynek megfelelő változó definiálatlan, ki lesz hagyva.

Példa 1. compact() példa

```
$varos    = "Miskolc";
$megye    = "BAZ";
$esemeny  = "P.A. elhagyja a varost";

$helyrajz = array ("varos", "megye");

$eredmeny = compact ("esemeny", "semmi_ilyen_valtozo_nincs", $helyrajz);
```

A `$eredmeny` értéke:

```
Array
(
    [esemeny] => P.A. elhagyja a varost
    [varos] => Miskolc
    [megye] => BAZ
)
```

Lásd még: extract()!

count (PHP 3, PHP 4)

megszámolja az elemeket egy változóban

int **count** (mixed var) \linebreak

Visszaadja a *var* változó elemeinek számát, amely általában array(tömb) típusú (mivel bármi másnak csak egy eleme lehet).

Ha a *var* változó nem tömb, akkor 1-et ad vissza, kivétel a `count(NULL)` ami egyenlő `0`-val.

> **Figyelem**
>
> A **count()** 0-val térhet vissza definiálatlan változók esetén és üres tömbök esetén is. Az isset() segítségével lehet megvizsgálni, hogy a változó be van-e állítva.

Olvasd át a kézikönyv Tömbök c. fejezetét a PHP-ban megvalósított tömbök részletes magyarazátáért.

Példa 1. count() példa

```
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$eredmeny = count ($a);
// $eredmeny == 3

$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$eredmeny = count ($b);
// $eredmeny == 3;
```

Megjegyzés: A sizeof() függvény a **count()** függvény álneve.

Lásd még: is_array(), isset() és strlen()!

current (PHP 3, PHP 4)

visszaadja egy tömb aktuális elemét

mixed **current** (array array) \linebreak

Minden tömbnek van egy ún. belső mutatója, amely a jelenlegi elemre mutat. A tömbbe beszúrt első elemre van inicializálva.

A **current()** függvény egyszerűen visszaadja a belső mutató által megjelölt tömbelemet. A mutató soha nem mozdul el erről az elemről. Ha a belső mutató az elemlista vége mögé mutat, akkor **current()** függvény FALSE-ot ad vissza.

> **Figyelem**
>
> Ha a tömb üres elemeket tartalmaz (0 vagy "" - üres string), akkor ez a függvény az ilyen elemeknél FALSE-ot ad vissza. Ez lehetetlenné teszi annak megállapítását a **current()** függvénnyel, hogy valóban a lista végére került-e a mutató. Az each() függvénnyel helyesen végig lehet lépkedni egy üres elemeket tartalmazó tömbön.

Lásd még: end(), next(), prev() és reset()!

each (PHP 3, PHP 4)

visszaadja a tömb pillanatnyilag kijelölt elemét és lépteti a belső mutatót

array **each** (array array) \linebreak

Az **each()** visszaadja az *array* tömb jelenlegi index-érték párját, és előre mozgatja a tömb belső mutatóját. Ez a index-érték pár egy négyelemű tömbben kerül visszaadásra az alábbi kulcsokkal: 0, 1, key, and value, valamint az alábbi elemekkel. *0*, *1*,*key* és *value*. A *0* és a *key* tartalmazza a tömbelem kulcsát, *1* és *value* pedig az adatokat.

Ha a belső mutató az elemlista vége mögé mutat, akkor **each()** függvény FALSE-ot ad vissza.

Példa 1. each() példák

```
$ize = array ("bob", "fred", "jussi", "jouni", "egon", "marliese");
$bigyo = each ($ize);
```

$bigyo tartalma most:

```
Array (
    0 => 0
    1 => 'bob'
    key => 0
    value => 'bob'
)
```

```
$ize = array ("Robert" => "Bob", "Seppo" => "Sepi");
$bigyo = each ($ize);
```

$bigyo tartalma most:

```
Array (
    0 => Robert
    1 => 'Bob'
    key => Robert
    value => 'Bob'
)
```

Az **each()**-et a list()-tel együtt szokás használni tömbök bejárására, mint például a `$HTTP_POST_VARS`:

Példa 2. `$HTTP_POST_VARS` bejárása each() függvénnyel

```
echo "POST metódussal elküldött változók:<br>";
reset ($HTTP_POST_VARS);
while (list ($key, $val) = each ($HTTP_POST_VARS)) {
    echo "$key => $val<br>";
}
```

Az **each()()** függvény lefutása után a belső tömbmutató a tömb következő elemére ugrik, vagy az utolsó elemre, ha a tömb végén áll a mutató. Használd a reset()-et, ha újból végig kell lépkedni a tömbön.

Lásd még: key(), list(), current(), reset(), next() és prev()!

end (PHP 3, PHP 4)

az utolsó elemre állítja a tömb belső mutatóját

mixed **end** (array array) \linebreak

Az **end()** a *`array`* tömb belső mutatóját annak utolsó elemére állítja, és azt adja vissza.

Lásd még: current(), each(), next() és reset()!

extract (PHP 3>= 3.0.7, PHP 4)

tömbből változókat helyez el az aktuális szimbólumtáblába

int **extract** (array var_array [, int extract_type [, string prefix]]) \linebreak

Ez a függvény arra szolgál, hogy egy tömb tartalma alapján változókat helyezzen el az aktuális szimbólumtáblában. A paraméterként kapott *var_array* tömb kulcsait tekinti a változók neveinek és a tömbelemek értékeit a változók tartalmának. Minden kulcs-érték párból készül egy változó a szimbólumtáblában, az *extract_type* és *prefix* paraméterektől függően.

Megjegyzés: PHP 4.0.5-től kezdve a beillesztett változók számával tér vissza a függvény.

Megjegyzés: EXTR_IF_EXISTS és EXTR_PREFIX_IF_EXISTS 4.2.0 verzótól használhatók.

Az **extract()** megvizsgál minden kulcsot (indexet), hogy az lehet-e egy érvényes változónév vagy sem, és hogy ilyen nevű változó létezik-e már a szimbólumtáblában (ütközés). Az érvénytelen változónevek és ütközések az *extract_type* paraméterben megadottak szerint lesz kezelve. Lehetséges értékei:

EXTR_OVERWRITE

Ütközéskor felülírja a már létező változót.

EXTR_SKIP

Ütközéskor nem írja felül a létező változót.

EXTR_PREFIX_SAME

Ütközéskor a létrehozandó változó neve elé *prefix* kerül.

EXTR_PREFIX_ALL

Minden létrehozandó változó neve elé *prefix* kerül. PHP 4.0.5-től kezdve ez érvényes a numerikus indexű elemekre is.

EXTR_PREFIX_INVALID

Minden érvénytelen nevű változónév elé a *prefix* kerül. PHP 4.0.5-től kezdve használható ez beállítás.

EXTR_IF_EXISTS

Csak akkor írja felül az adott nevű változót, ha az már létezik az aktuális szimbólimtáblában, egyébként nem csinál semmit. Ez hasznos lehet, ha : érvényes változókat előre definiálni, majd létrehozni a $_REQUEST tömbön kívül definiált változókat. PHP 4.2.0-től használható.

EXTR_PREFIX_IF_EXISTS

Csak akkor hozza létre a toldalékolt nevű változót, ha annak a toldalék nélküli megfelelője már létezik már létezik az aktuális szimbólimtáblában. PHP 4.2.0-től használható.

Ha nincs megadva az *extract_type*, akkor EXTR_OVERWRITE az alapértelmezés.

A *prefix* csak akkor szükséges, ha az *extract_type* EXTR_PREFIX_SAME, EXTR_PREFIX_ALL, EXTR_PREFIX_INVALID vagy EXTR_PREFIX_IF_EXISTS értékű. Ha prefixszel ellátott változónév sem érvényes, akkor az nem kerül bele a szimbólumtáblába.

Az **extract()** a sikeresen beillesztett változók számával tér vissza.

Az extract függvény egy lehetséges használata azoknak a változóknak a szimbólumtáblába importálása, amelyeket a wddx_deserialize() adott vissza egy asszociatív tömbben.

Példa 1. extract() példa

```
<?php

/* Feltételezi, hogy a $var_array egy a
   wddx_deserialize által visszaadott tömb*/

$méret    = "nagy";
$var_tomb = array ("szín"  => "kék",
                   "méret" => "közepes",
                   "alak"  => "gömb");
extract ($var_tomb, EXTR_PREFIX_SAME, "wddx");

print "$szín, $méret, $alak, $wddx_méret\n";
?>
```

A fenti példa ezt eredményezi:

```
kék, nagy, gömb, közepes
```

A `$méret` változó nem lett felülírva az EXTR_PREFIX_SAME megadása miatt, amely azt eredményezte, hogy ehelyett a `$wddx_méret` változóba került az új érték. Ha EXTR_SKIP-et használtunk volna, akkor a `$wddx_méret` sem került volna létrehozásra. Az EXTR_OVERWRITE pedig azt okozta volna, hogy a `$méret` értéke "közepes"-re változott volna. Az EXTR_PREFIX_ALL-lal pedig minden változó neve wddx előtaggal kezdődne: `$wddx_szín`, `$wddx_méret` és `$wddx_alak`.

Asszociatív tömböket kell használni, mert a numerikusan indexeltek nem adnak eredményt, kivéve EXTR_PREFIX_ALL vagy EXTR_PREFIX_INVALID jelzők használatakor.

Lásd még: compact()!

in_array (PHP 4)

`TRUE`-val tér vissza, ha létezik az érték a tömbben

bool **in_array** (mixed needle, array haystack [, bool strict]) \linebreak

Átnézi a *`haystack`* tömböt a *`needle`* elemet keresve, ha megtalálta, akkor `TRUE`-t ad vissza, egyébként `FALSE`-ot.

Ha a *`strict`* harmadik paraméter értéke `TRUE`, akkor az **in_array()** a *`needle`* típusát is összeveti a *`haystack`* tömb elemeiével.

Megjegyzés: Ha a *needle* paraméter sztring, akkor az összehasonlítás kis- és nagybetűk különbözőségére érzékeny.

Megjegyzés: A PHP 4.2.0 verziója előtt a *needle* paraméter nem lehetett tömb.

Példa 1. in_array() példa

```
$os = array ("Mac", "NT", "Irix", "Linux");
if (in_array ("Irix", $os)) {
    print "Got Irix";
}
if (in_array ("mac", $os)) {

    print "Got mac";
}
```

A második feltétel nem teljesül, mert az **in_array()** érzékeny a kis- és nagybetűk különbségére, ezért a program kimenete :

```
Got Irix
```

Példa 2. in_array() strict paraméterrel

```
<?php
$a = array(’1.10’, 12.4, 1.13);

if (in_array(’12.4’, $a, TRUE))
    echo "szigorúan ’12.4’-ot találtam meg.\n";
if (in_array(1.13, $a, TRUE))
    echo "szigorúan 1.13-ot találtam meg. \n";
?>
```

Ez a következőt írja ki:

```
szigorúan 1.13-ot találtam meg.
```

Példa 3. in_array() with an array as needle

```
<?php
$a = array(array('p', 'h'), array('p', 'r'), 'o');

if ( in_array(array ('p', 'h'), $a) ){
  echo "'ph'-t megtaláltam.\n";
}
if ( in_array(array ('f', 'i'), $a) ){
  echo "'fi'-t megtaláltam.\n";
}
if ( in_array('o', $a) ){
  echo "'o'-t megtaláltam.\n";
}
?>
```

Ez a következőt írja ki:

```
'ph'-t megtaláltam.
'o'-t megtaláltam.
```

Lásd még: array_search()!

key (PHP 3, PHP 4)

asszociatív tömb pillanatnyilag kiválasztott kulcsát adja vissza

mixed **key** (array array) \linebreak

A **key()** a belső tömbmutató által pillanatnyilag kijelölt elem kulcsával tér vissza.

Lásd még: current() és next()!

krsort (PHP 3>= 3.0.13, PHP 4)

kulcsok alapján fordított sorrendbe rendez egy tömböt

int **krsort** (array array [, int sort_flags]) \linebreak

Ez a függvény úgy rendezi az *array* tömböt a kulcsai alapján, hogy az indexek megtartják kapcsolatukat a hozzájuk rendelt tömb elemeivel. Főként asszociatív tömbök rendezéséhez használatos.

Példa 1. krsort() példa

```
$gyumolcsok = array ("d"=>"citrom", "a"=>"narancs", "b"=>"banán", "c"=>"alma");
krsort ($gyumolcsok);
 reset ($gyumolcsok);
while (list ($key, $val) = each ($gyumolcsok)) {
    echo "$key = $val\n";
}
```

Ez a példa a következőt írja ki:

```
d = citrom
c = alma
b = banán
a = narancs
```

A rendezést befolyásolja a *`sort_flags`* paraméter, részletesebben lásd a sort() függvénynél!

Lásd még: asort(), arsort(), ksort() sort(), natsort() és rsort()!

ksort (PHP 3, PHP 4)

kulcsok alapján rendezi a tömböt

int **ksort** (array array [, int sort_flags]) \linebreak

Ez a függvény úgy rendezi az *`array`* tömböt a kulcsai alapján, hogy az indexek megtartják kapcsolatukat a hozzájuk rendelt tömb elemeivel. Főként asszociatív tömbök rendezéséhez használatos.

Példa 1. ksort() példa

```
$gyumolcsok = array ("d"=>"citrom", "a"=>"narancs", "b"=>"banán", "c"=>"alma");
ksort ($gyumolcsok);
reset ($gyumolcsok);
while (list ($key, $val) = each ($gyumolcsok)) {
    echo "$key = $val\n";
}
```

Ez a példa a következőt írja ki:

```
a = narancs
b = banán
c = alma
d = citrom
```

A rendezést befolyásolja a *sort_flags* paraméter, részletesebben lásd a sort() függvénynél!

Lásd még: asort(), arsort(), krsort(), uksort(), sort(), natsort() és rsort()!

Megjegyzés: A második paraméter PHP 4-ben használható.

list (unknown)

értéket rendel a felsorolt változókhoz

void **list** (mixed ...) \linebreak

Az array()-hoz hasonlóan a **list()** sem valódi függvény, hanem nyelvi építőelem. A **list()** a több változó érték-hozzárendelésére szolgál egy műveleten belül.

Megjegyzés: A **list()** csak numerikus indexelésű tömbökön működik helyesen, és akkor is csak, ha az indexelés 0-tól kezdődik.

Példa 1. list() példák

```
<?php

$info = array('kávé', 'barna', 'koffein');

// az összes változó kiírása
list($ital, $szin, $ero) = $info;
print "A $ital $szin és a $ero teszi különlegessé.\n";

// csak néhány változó kiírása
list($ital, , $ero) = $info;
print "A {$ital}ban van $ero.\n";

// vagy a harmadik, legutols= kirása
list( , , $ero) = $info;
print "Kérek {$power}t!\n";

?>
```

Példa 2. Példa a list() használatára

```
<table>
 <tr>
  <th>Alkalmazott neve</th>
  <th>fizetése</th>
 </tr>

<?php

$result = mysql_query ("SELECT id, nev, fizetes FROM alkalmazottak",$conn);
while (list ($id, $nev, $fizetes) = mysql_fetch_row ($result)) {
    print (" <tr>\n".
           "  <td><a href=\"info.php?id=$id\">$nev</a></td>\n".
           "  <td>$fizetes</td>\n".
           " </tr>\n");
}

?>

</table>
```

Lásd még: each(), extract() és array()!

natcasesort (PHP 4)

tömböt rendez a "natural order" (természetes sorrend) algoritmus alapján kis-nagybetűk figyelembevétele nélkül

void **natcasesort** (array array) \linebreak

Ez a függvény olyan rendező algoritmust valósít meg, amely az alfanumerikus sztringeket úgy rendezi, ahogy azt egy ember tenné. Ezt az eljárást nevezik "natural ordering"-nek - természetes rendezésnek.

A **natcasesort()** függvény a natsort() olyan változata, amely érzéketlen a kis- és nagybetűk különbségére. Lásd a natsort() függvényt a reguláris és a természetes rendezési algortimusok közötti különbségért!

Részletesebb leírásért lásd: Martin Pool Natural Order String Comparison (http://naturalordersort.org/) c. oldalát!

Lásd még: sort(), natsort(), strnatcmp() és strnatcasecmp()!

natsort (PHP 4)

tömböt rendez a "natural order" (természetes sorrend) algoritmus alapján

void **natsort** (array array) \linebreak

Ez a függvény olyan rendező algoritmust valósít meg, amely az alfanumerikus sztringeket úgy rendezi, ahogy azt egy ember tenné. Ezt az eljárást nevezik "natural ordering"-nek - természetes rendezésnek. A sort() függvény reguláris rendezési algortimusa és eközötti különbséget szemlélteti a következő példa:

Példa 1. natsort() példa

```
$tomb1 = $tomb2 = array ("img12.png", "img10.png", "img2.png", "img1.png");

sort($tomb1);
echo "Hagyományos rendezés\n";
print_r($tomb1);

natsort($tomb2);
echo "\nTermészetes rendezés\n";
print_r($tomb2);
```

Ez a példa a következőt írja ki:

```
Hagyományos rendezés
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Természetes rendezés
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
```

Részletesebb leírásért lásd: Martin Pool Natural Order String Comparison (http://naturalordersort.org/) c. oldalát!

Lásd még: natcasesort(), strnatcmp() és strnatcasecmp()!

next (PHP 3, PHP 4)

előre mozgatja a tömb belső mutatóját

mixed **next** (array array) \linebreak

Visszaadja belső tömbmutató pozíciója utáni elemet, vagy FALSE-ot, ha nincs következő elem.

A **next()** a current()-hoz hasonlóan viselkedik egy különbséggel. A belső tömbmutatót egy hellyel előbbre állítja, mielőtt visszadná az elemet. Ez azt jelenti, hogy a mostani pozició után következő elemmel tér vissza, és a belső tömbmutatót is eggyel előre lépteti. Ha a belső tömbmutató az elemlista végére ér, akkor a **next()** FALSE-ot ad vissza.

> **Figyelem**
>
> Ha a tömb üres elemeket is tartalmaz, akkor ezeknél az elemeknél a függvény ugyanúgy FALSE-szal tér vissza. Üres elemeket tartalmazó tömb bejárásához nézd meg az each() függvényt!

Lásd még: current(), end(), prev() és reset()!

pos (PHP 3, PHP 4)

visszaadja a tömb aktuális elemét

mixed **pos** (array array) \linebreak

Ez a current() függvény álneve.

Lásd még: end(), next(), prev() és reset()!

prev (PHP 3, PHP 4)

visszlépteti a tömb belső mutatóját

mixed **prev** (array array) \linebreak

Visszaadja belső tömbmutató pozíciója előtti tömbelemet, vagy FALSE-ot ha nincs következő elem.

> **Figyelem**
>
> Ha a tömb üres elemeket is tartalmaz, akkor ezeknél az elemeknél a függvény ugyanúgy FALSE-szal tér vissza. Üres elemeket tartalmazó tömb bejárásához nézd meg az each() függvényt!

A **prev()** a next()-hez hasonlóan működik, kivéve hogy a tömbmutatót visszafelé lépteti nem pedig előre.

Lásd még: current(), end(), next() és reset()!

range (PHP 3>= 3.0.8, PHP 4)

egy tartományt lefedő tömböt készít

array **range** (mixed low, mixed high) \linebreak

A **range()** a *low* és *high* közé eső elemekből alkotott tömbbel tér vissza. Ha *low* nagyobb, mint *high*, akkor a sorozat *high*-tól indul *low* felé.

Példa 1. range() példák

```
foreach(range(0, 9) as $szam) {
    echo $szam;
}
foreach(range('a', 'z') as $betu) {
    echo $betu;
}
foreach(range('z', 'a') as $betu) {
    echo $betu;
}
```

Megjegyzés: A 4.1.0 verziót megelőzően a **range()** függvény csak a növekvő egészszámokból álló tömböt tudott generálni. A karakteres tartomány és a csökkenő sorrendű elemek támogatása a 4.1.0 verziótól létezik.

Példa 2. csökkenő és a karakteres tartományok szimulációja

```
// array_reverse használható arra, hogy megfordítsa a tömböt
foreach(array_reverse(range(0,9)) as $szam) {
  echo $szam;
}

# array_map() és chr() használható arra, hogy számokból karaktereket alkosson
foreach(array_map('chr', range(ord('a'),ord('z'))) as $karakter) {
  echo $karakter;
}
```

Lásd még a shuffle() függvényt további példákért!

reset (PHP 3, PHP 4)

a tömb belső mutatóját az első elemére állítja

mixed **reset** (array array) \linebreak

A **reset()** visszaállítja az *array* tömb belső tömbmutatóját az első elemre.

reset() a tömb első elemével tér vissza.

Lásd még: current(), each(), next() és prev()!

rsort (PHP 3, PHP 4)

fordított sorrendben rendez egy tömböt

void **rsort** (array array [, int sort_flags]) \linebreak

Ez a függvény fordított sorrendben rendezi át a tömböt: a legnagyobbtól a legkisebb felé.

Példa 1. rsort() példa

```
$gyumolcsok = array ("citrom", "narancs", "banán", "alma");
rsort ($gyumolcsok);
reset ($gyumolcsok);
while (list ($key, $val) = each ($gyumolcsok)) {
    echo "$key = $val\n";
}
```

Ez a példa a következőt írja ki:

```
0 = narancs
1 = citrom
2 = banán
3 = alma
```

A gyümölcsök az abc szerint fordított sorrendben vannak rendezve.

A rendezést befolyásolja a *sort_flags* paraméter, részletesebben lásd a sort() függvénynél!

Lásd még: arsort(), asort(), ksort(), sort() és usort()!

shuffle (PHP 3>= 3.0.8, PHP 4)

összekeveri a tömb elemeit

void **shuffle** (array array) \linebreak

Ez a függvény véletlenszerűen összekeveri a tömb elemeit. Az srand() függvényt meg kell hívni, hogy inicializálja a véletlenszám-generátort!

Példa 1. shuffle() példa

```
$numbers = range (1,20);
srand ((float)microtime()*1000000);
shuffle ($numbers);
while (list (, $number) = each ($numbers)) {
    echo "$number ";
}
```

Lásd még: arsort(), asort(), ksort(), rsort(), sort() és usort()!

sizeof (PHP 3, PHP 4)

visszadja a változó elemszámát

int **sizeof** (mixed var) \linebreak

A **sizeof()** függvény a count() álneve.

Lásd még: count()!

sort (PHP 3, PHP 4)

tömböt rendez

void **sort** (array array [, int sort_flags]) \linebreak

Ez a függvény egy tömb elemeit rendezi sorba a legkisebbtől haladva a legnagyobb felé - azaz növekvő sorrendben.

Példa 1. sort() példa

```
<?php

$gyumolcsok = array ("citrom", "narancs", "banán", "alma");
sort ($gyumolcsok);
reset ($gyumolcsok);
while (list ($key, $val) = each ($gyumolcsok)) {
    echo "gyumolcsok[".$key."] = ".$val."\n";
}

?>
```

Ez a példa a következőt írja ki:

```
gyumolcsok[0] = alma
gyumolcsok[1] = banán
gyumolcsok[2] = citrom
gyumolcsok[3] = narancs
```

A gyümölcsök az abc sorrendben vannak rendezve.

A rendezést befolyásolja a *sort_flags* paraméter, amelynek a következő értékei lehetnek:

(a rendezést befolyásoló jelzők)

- SORT_REGULAR - az elemeket hagyományosan hasonlítja össze
- SORT_NUMERIC - az elemeket számokként hasonlítja össze
- SORT_STRING - az elemeket szövegként hasonlítja össze

Lásd még: arsort(), asort(), ksort(), natsort(), natcasesort(), rsort(), usort(), array_multisort() és uksort()!

Megjegyzés: A második paraméter PHP 4-től használható.

uasort (PHP 3>= 3.0.4, PHP 4)

tömböt rendez egy felhasználó összehasonlító függvénnyel megtartva az indextársítást

void **uasort** (array array, function cmp_function) \linebreak

Ez a függvény úgy rendez egy tömböt, hogy megtartja az indexek és az értékek közti relációt. Főleg akkor használatos, amikor az asszociatív tömböket rendezzünk, ahol az elemek sorrendje fontos. Az összehasonlítást végző függvényt a felhasználó határozza meg.

Megjegyzés: További példákért lásd a usort() és uksort() példáit, hogyan kell felhasználói függvényeket használni!

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

Lásd még: usort(), uksort(), sort(), asort(), arsort(), ksort() és rsort()!

uksort (PHP 3>= 3.0.4, PHP 4)

tömböt rendez kulcsai alapján egy felhasználó összehasonlító függvénnyel

void **uksort** (array array, function cmp_function) \linebreak

Ez a függvény egy felhasználói függvénnyel átrendezi a tömböt a kulcsok szerint. Akkor használjuk ezt a függvényt, ha a rendezendő tömböt nem szokványos feltétel alapján akarjuk rendezni,

Példa 1. uksort() példa

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (4 => "négy", 3 => "három", 20 => "húsz", 10 => "tíz");

uksort ($a, "cmp");

while (list ($key, $val) = each ($a)) {
    echo "$key: $val\n";
}
```

Ez a példa a következőt írja ki:

```
20: húsz
10: tíz
4: négy
3: három
```

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

Lásd még: usort(), uasort(), sort(), asort(), arsort(), ksort(), natsort() és rsort()!

usort (PHP 3>= 3.0.3, PHP 4)

tömböt rendez elemértékei alapján egy felhasználó összehasonlító függvénnyel

void **usort** (array array, string cmp_function) \linebreak

Ez a függvény a *`cmp_function`* paraméterben megadott nevű felhasználói függvénnyel rendezi az *`array`* tömb elemeit. Ha a rendezendő tömböt nem szokványos feltétel alapján akarjuk rendezni, akkor használjuk ezt a függvényt.

Az összehasonlító függvénynek 0-nál kisebb vagy nagyobb ill. 0 értéket kell visszaadnia, ha az első elem kisebb vagy nagyobb a másodiknál, ill. ha egyenlők. Ha két elem egyenlő, sorrendjük a rendezett tömbben nem definiált.

Példa 1. usort() példa

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (3, 2, 5, 6, 1);

usort ($a, "cmp");

while (list ($key, $val) = each ($a)) {
    echo "$key: $value\n";
}
```

Ez a példa a következőt írja ki:

```
0: 6
1: 5
2: 3
3: 2
4: 1
```

> **Megjegyzés:** Ebben az egyszerű esetben nyilvánvalóan jobb választás az rsort() függvény használata.

Példa 2. Az usort() használata több dimenziós tömbökön

```
function cmp ($a, $b) {
    return strcmp($a["gyumolcs"], $b["gyumolcs"]);
}

$gyumolcsok[0]["gyumolcs"] = "citrom";
$gyumolcsok[1]["gyumolcs"] = "alma";
$gyumolcsok[2]["gyumolcs"] = "szőlő";
```

```
usort($gyumolcsok, "cmp");

while (list ($key, $val) = each ($gyumolcsok)) {
    echo "\$gyumolcsok[$key]: " . $value["gyumolcs"] . "\n";
}
```

Több dimenziós tömbökre alkalmazva `$a` és `$b` az első indexre mutató referenciákat fog tartalmazni.

Ez a példa a következőt írja ki:

```
$gyumolcsok[0]: alma
$gyumolcsok[1]: citrom
$gyumolcsok[2]: szőlő
```

> **Megjegyzés:** A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

Példa 3. Objektum tagfüggvényének (metódus) használata az usort()-tal

```
class TestObj {
    var $nev;

    function TestObj($nev)
    {
        $this->nev = $nev;
    }

    /* ez statikus összehasonlító függvény: */
    function cmp_obj($a, $b)
    {
        $al = strtolower($a->nev);
        $bl = strtolower($b->nev);
        if ($al == $bl) return 0;
        return ($al > $bl) ? +1 : -1;
    }
}

$a[] = new TestObj("c");
$a[] = new TestObj("b");
$a[] = new TestObj("d");

uasort($a, array ("TestObj", "cmp_obj"));

foreach ($a as $elem) {
    print $elem->nev."\n";
```

```
}
```

Ez a példa a következőt írja ki:

```
b
c
d
```

> **Figyelem**
>
> Néhány C könyvtárban (mint például a Solaris rendszerekben) a gyorsrendezés alapjául szolgáló függvény a PHP összeomlását okozhatja, ha az összehasonlító függvény nem következetes értéket ad vissza.

Lásd még: uasort(), uksort(), sort(), asort(), arsort(),ksort(), natsort() és rsort()!

III. Aspell függvények [ellenjavallt]

Az **aspell()** függvények ellenőrzik egy szó helyesírását és javaslatokat kínálnak fel.

> **Megjegyzés:** Az aspell csak a nagyon régi (a .27.* verzióig) aspell könyvtárakkal működik együtt. Ez a modul sem és az aspell könyvtár azon verziói sem támogatottak a továbbiakban. Ha helyesírásellenőrzési funkciókat szeretnél használni a PHP-ban, használd a pspell könyvtárat. Az a pspell könyvtárat használja, és képes az újabb aspell verzókkal együttműködni.

A függvényekhez aspell könyvár szükséges, amely beszerezhető a http://aspell.sourceforge.net/ címen.

aspell_check_raw (PHP 3>= 3.0.7, PHP 4)

Ellenőriz egy szót kis- és nagybetűk cseréje illetve a szöveg trimmelése nélkül [ellenjavallt]

bool **aspell_check_raw** (int dictionary_link, string word) \linebreak

Az **aspell_check_raw()** ellenőrzi egy szó helyesírását anélkül, hogy manipulálná (megváltoztatná a kis- és nagybetűket vagy levágná a stringet), és TRUE-t ad vissza, ha a helyesírás jó, egyébként FALSE-ot.

Példa 1. aspell_check_raw() példa

```
$aspell_link=aspell_new ("english");
if (aspell_check_raw ($aspell_link, "test")) {
    echo "A helyesírás jó";
} else {
    echo "Bocs, de a helyesírás rossz";
}
```

aspell_check (PHP 3>= 3.0.7, PHP 4)

Ellenőriz egy szót [ellenjavallt]

bool **aspell_check** (int dictionary_link, string word) \linebreak

Az **aspell_check()** ellenőrzi egy szó helyesírását és TRUE-t ad vissza, ha a helyesírás jó, egyébként FALSE-ot.

Példa 1. aspell_check() példa

```
$aspell_link=aspell_new ("english");
if (aspell_check ($aspell_link, "testt")) {
    echo "A helyesírás jó";
} else {
    echo "Bocs, de a helyesírás rossz";
}
```

aspell_new (PHP 3>= 3.0.7, PHP 4)

Betölt egy új szótárt [ellenjavallt]

int **aspell_new** (string master [, string personal]) \linebreak

Az **aspell_new()** megnyit egy új szótárt, és visszaadja a szótárkapcsolat azonosítóját, amely a többi aspell függvényben használható. Hiba esetén FALSE-szal tér vissza.

Példa 1. aspell_new() példa

```
$aspell_link=aspell_new ("english");
```

aspell_suggest (PHP 3>= 3.0.7, PHP 4)

Egy szó helyesírására tesz javaslatot [ellenjavallt]

array **aspell_suggest** (int dictionary_link, string word) \linebreak

Az **aspell_suggest()** visszaadja az adott szó helyesírási javaslatait egy tömbben.

Példa 1. aspell_suggest() példa

```
$aspell_link = aspell_new("english");

if (!aspell_check ($aspell_link, "test")) {
    $javaslatok = aspell_suggest($aspell_link, "test");

    foreach ( $javaslatok as $javaslat ) {
        echo "Lehetséges helyesírási változatok:: " . $javaslat . "&lt;br>";
    }
}
```

IV. BCMath tetszőleges pontosságú matematikai függvények

Ha tetszőleges pontosságú matematikai műveleteket szeretnél végezni, a PHP a Binary Calculator Mathematics (BCMath) függvényeket kínálja erre a célra. Ezek a függvények tetszőleges méretben és pontossággal tudnak számokat ábrázolni karaktersorozatok felhasználásával.

Előfeltételek

Licensz kérdések miatt a BCMATH könyvtár a standard PHP csomagtól különálló elemként tölthető le. Egy tar-gzippelt verzió beszerezhető a http://www.php.net/extra/number4.tar.gz címen. Olvasd el a `README.BCMATH` fájlt a PHP csomagodban további információkért.

Telepítés

PHP 4-ben ezek a függvények csak akkor érhetőek el, ha a PHP-t a `--enable-bcmath` beállítással fordítottad. PHP 3-ban ezek a függvények csak akkor érhetőek el, ha a PHP-t NEM a `--disable-bcmath` beállítással fordítottad.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előredefiniált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

bcadd (PHP 3, PHP 4)

Összead két tetszőleges pontosságú számot

string **bcadd** (string left_operand, string right_operand [, int scale]) \linebreak

Hozzáadja a *left_operand*-ot a *right_operand*-hoz, és visszatér az összeget tartalmazó stringgel. Az elhagyható *scale* paraméter határozza meg az eredményben a tizedes pont utáni számjegyek számát.

Lásd még: bcsub()!

bccomp (PHP 3, PHP 4)

Összehasonlít két tetszőleges pontosságú számot

int **bccomp** (string left_operand, string right_operand [, int scale]) \linebreak

Összehasonlítja a *left_operand*-ot a *right_operand*-al, és az eredményt egészként visszaadja. Az elhagyható *scale* paraméter beállítja a tizedes pont utáni számjegyek számát, amely felhasználásra kerül az összehasonlításban. A visszatérési érték 0, ha a két operandus egyenlő. Ha a *left_operand* nagyobb a *right_operand*-nál, az eredmény +1, de ha a *left_operand* kisebb nála, az eredmény -1.

bcdiv (PHP 3, PHP 4)

Eloszt két tetszőleges pontosságú számot

string **bcdiv** (string left_operand, string right_operand [, int scale]) \linebreak

Elosztja a *left_operand*-ot a *right_operand*-al, és visszaadja az eredményt. Az elhagyható *scale* paraméter beállítja az eredményben a tizedespont utáni számjegyek számát.

Lásd még: bcmul()!

bcmod (PHP 3, PHP 4)

Kiszámítja két tetszőleges pontosságú szám modulusát

string **bcmod** (string left_operand, string modulus) \linebreak

Kiszámítja a *left_operand*-nak a *modulus*-sal történő osztás utáni maradékát.

Lásd még: bcdiv()!

bcmul (PHP 3, PHP 4)

Összeszoroz két tetszőleges pontosságú számot

string **bcmul** (string left_operand, string right_operand [, int scale]) \linebreak

Összeszorozza a *left_operand*-ot a *right_operand*-dal, és visszaadja az eredményt. Az elhagyható *scale* paraméter határozza meg az eredményben a tizedespont utáni számjegyek számát.

Lásd még: bcdiv()!

bcpow (PHP 3, PHP 4)

Egy tetszőleges pontosságú számot egy másik hatványára emel

string **bcpow** (string x, string y [, int scale]) \linebreak

Kiszámítja *x* szám *y*. hatványát. Az elhagyható *scale* paraméter beállítja az eredményben a tizedes pont utáni számjegyek számát.

Lásd még: bcsqrt()!

bcscale (PHP 3, PHP 4)

Beállítja az alapértelmezett skálázási értéket az összes BCMath matematikai függvény részére

string **bcscale** (int scale) \linebreak

Ez a függvény beállítja a skálázási értékét minden bc matematikai függvény részére, így nem kell mindig explicit módon megadni a *scale* paramétert, csak ha szükséges.

bcsqrt (PHP 3, PHP 4)

Kiszámítja egy tetszőleges pontosságú szám négyzetgyökét

string **bcsqrt** (string operand [, int scale]) \linebreak

Visszaadja az *operand* négyzetgyökét. Az elhagyható *scale* paraméter beállítja az eredményben a tizedes pont utáni számjegyek számát.

Lásd még: bcpow()!

bcsub (PHP 3, PHP 4)

Kivon egy tetszőleges pontosságú számot egy másikból

string **bcsub** (string left_operand, string right_operand [, int scale]) \linebreak

Kivonja a *`right_operand`*-ot a *`left_operand`*-ból, és visszaadja az eredményt tartalmazó string-et. Az elhagyható *`scale`* paraméter beállítja az eredményben a tizedes pont utáni számjegyek számát.

Lásd még: bcadd()!

V. Bzip2 tömörítési függvények

Ez a modul a bzip2 (http://sources.redhat.com/bzip2/)könyvtár függvényeit használja, a bzip2 (.bz2) tömörítési eljárással tömörített állományokat és a bennük levő fájlokat írja-olvassa. A bzip promgamkönytárait Julian Seward írta.

A bzip2 támogatás nincs alapállapotban bekapcsolva a PHP-ben. Fordításkor alkalmazd a --with-bz2 kapcsolót, ha használni akarod a bzip2 függvényeket. Szükség van továbbá a bzip2/libbzip2 könyvtárak >= 1.0.x. verziójára is.

Rövid példa

A következő példaprogram megnyit egy próbafájlt, beleír egy rövid szöveget, majd kiírja a fájl tartalmát.

Példa 1. Rövid bzip2 példa

```
<?php

$filename = "/tmp/testfile.bz2";
$str = "Ez egy próbaszövegecske.\n";

// megnyitás írásra - "w" kapcsoló használatával
$bz = bzopen($filename, "w");

// beleírja a próbaszöveget a fájlba
bzwrite($bz, $str);

// bezárja a fájlot
bzclose($bz);

// újra megnyitja, de most olvasásra - "r" kapcsolót használ
$bz = bzopen($filename, "r");

// beolvas 10 karaktert a fájlból
print bzread($bz, 10);

// A fájl végéig (vagy a következő 1024 karakterig) kiírja a tartalmát, majd bezárja a fa
print bzread($bz);

bzclose($bz);

?>
```

bzclose (PHP 4 >= 4.0.4)

Bezár egy bzip2 fájl mutatót

int **bzclose** (int bz) \linebreak

Bezár egy bzip2 fájlot, amire a *bz* fájl mutatóval (file pointer) utaltunk. (A fájl mutató megértését megkönnyítheti az első, "rövid bzip2 példa" hetedik sora.)

TRUE értékkel tér vissza ha sikerül, egyébként FALSE-szal.

A fájl mutatónak egy létező és bzopen() függvénnyel sikeresen megnyitott fájlra kell mutatnia.

Lásd még: bzopen()!

bzcompress (PHP 4 >= 4.0.4)

bzip2 algoritmus szerint tömörít egy jelsorozatot

string **bzcompress** (string source [, int blocksize [, int workfactor]]) \linebreak

A **bzcompress()** függvény tömöríti a *source* paraméterben megadott forrásfájlt és visszatér ennek bzip2-ben tömörített változatával.

Az elhagyható *blocksize* paraméterrel meghatározhatjuk a tömörítéshez használt blokkméretet, 1 és 9 között. A 9 adja a legjobb tömörítést, de ez a leginkább erőforrás-igényesebb is. A *blocksize* paraméter alpértelmezett értéke: 4.

A szintén elhagyható *workfactor* paraméter a tömörítési fázisok viselkedését befolyásolja, hogyan viselkedjenek a legrosszabb esetben, vagyis amikor temérdek ismétlődő adat követi egymást. Az értéke 0 és 250 között lehet, a 250 és a 0 különleges esetekben használatos, míg a 30 az alapértelmezett érték. A *workfactor* paramétertől függetlneül, a generált kimenet ugyanaz lesz.

Példa 1. bzcompress() Példa

```
<?php
$str = "minta adat";
$bzstr = bzcompress($str, 9);
print( $bzstr );
?>
```

Lásd még: bzdecompress()!

bzdecompress (PHP 4 >= 4.0.4)

Kitcsomagol bzip2 tömörített adatokat

string **bzdecompress** (string source [, int small]) \linebreak

A **bzdecompress()** kitömöríti *source*paraméterben megadott, bzip2 tömörített adatot. Ha a *small* elhagyható paraméter TRUE (logikai IGAZ), akkor egy alternatív kicsomagoló algoritmust fog használni, ami ugyan kevesebb memóriát fogyaszt - a maximális memóriaszükséglet kb. 2300K-ra csökken -, de durván feleakkora sebességgel fut. Nézd meg a bzip2 dokumentációt (http://sources.redhat.com/bzip2/) a részletesebb információkért e témában.

Példa 1. bzdecompress()

```
<?php
$start_str = "Hát nem édes pofa?";
$bzstr = bzcompress($start_str);

print( "A tömörített szting: " );
print( $bzstr );
print( "\n<br>\n" );

$str = bzdecompress($bzstr);
print( "A kicsomagolt sztring: " );
print( $str );
print( "\n<br>\n" );
?>
```

Lásd még: bzcompress()!

bzerrno (PHP 4 >= 4.0.4)

A bzip2 hiba számával tér vissza

int **bzerrno** (int bz) \linebreak

Bármilyen bzip2 művelet során fellépő hiba számával tér vissza, amit a *bz* fájl mutató (file pointer) szolgáltat.

Lásd még: bzerror() ésbzerrstr().

bzerror (PHP 4 >= 4.0.4)

A bzip2 hiba számát és a hibás sztringet egy tömbbe pakolva tér vissza.

array **bzerror** (int bz) \linebreak

Egy asszociatív tömbbe teszi a bzip2 művelet során fellépett hibákat és a hozzájuk tartozó hibás sztringeket, amiket a *bz* fájl mutatótól (file pointer) vesz át .

Példa 1. bzerror() Példa

```
<?php
$hiba = bzerror($bz);

echo $hiba["errno"];
echo $hiba["errstr"];
?>
```

Lásd még bzerrno() és bzerrstr().

bzerrstr (PHP 4 >= 4.0.4)

A bzip2 hibasztringgel tér vissza

string **bzerrstr** (int bz) \linebreak

Egy bzip2 művelet alatt fellépő hiba hibáüzenetével tér vissza, amit a *bz* fájl mutató (file pointer) ad át a függvénynek.

Lásd még: bzerrno() és bzerror().

bzflush (PHP 4 >= 4.0.4)

Kikényszeríti az összes pufferelt adat írását

int **bzflush** (int bz) \linebreak

Az összes pufferelt bzip2 adat írását kikényszeríti a *bz* fájl mutató (file pointer) számára.

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

Lásd még: bzread() és bzwrite()!

bzopen (PHP 4 >= 4.0.4)

Megnyit egy bzip2 tömörített fájlt

int **bzopen** (string filename, string mode) \linebreak

Megnyit egy bzip2 (.bz2 kiterjesztésű) fájlt olvasásra vagy írásra. *filename* paraméter a fájl neve, amit ki szeretnénk nyitni. A *mode* paraméter hasonlóan működik, mint a fopen() függvénynél (‘r’ - olvasás, ‘w’ - írás, stb.).

Ha a művelet sikertelen, akkor FALSE-szal (logikai HAMIS), egyébként az újonnan megnyitott fájlra mutatóval (file pointer) tér vissza.

Példa 1. bzopen() Példa

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$tömörített_fájl = bzread($bz, filesize("/tmp/foo.bz2"));
bzclose($bz);

print( "Íme a /tmp/foo.bz2 tartalma: " );
print( "\n<br>\n" );
print( $tömörített_fájl );
?>
```

Lásd még: bzclose()!

bzread (PHP 4 >= 4.0.4)

Bináris bzip2 fájlokat olvas

string **bzread** (int bz [, int length]) \linebreak

A **bzread()** függvény beolvas *`length`* hosszúságú bájtot a *`bz`* paraméterben megadott bzip2 fájl mutatóból (file pointer). Addig olvas, amíg a *`length`* paraméternek megfelelő (tömörítetlen!) mennyiségű bájtot beolvassa vagy a fájlvég-jelet (EOF) eléri; már amelyik előbb bekövetkezik. Ha az elhagyható *`length`* paramétert nem állítottuk be, akkor a **bzread()** függvény 1024 (tömörítetlen) bájtot olvas egyszerre.

Példa 1. bzread() Példa

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$str = bzread($bz, 2048);
print( $str );
?>
```

Lásd még: bzwrite() és bzopen()!

bzwrite (PHP 4 >= 4.0.4)

Bináris bzip2 fájlba ír

int **bzwrite** (int bz, string data [, int length]) \linebreak

A **bzwrite()** függvény a *data* paraméterben megadott sztringet abba a bzip2 fájlba írja, amelyre a *bz* mutat. Ha az elhagyható *length* paramétert megadtuk, akkor az írás befejeződik, miután elérte a megadott (tömörítetlen) bájt hosszt ill. elérte a fájlvég-jelet (EOF), ha az hamarább bekövetkezik.

Példa 1. bzwrite() Példa

```
<?php
$str = "tömörítetlen adat";
$bz = bzopen("/tmp/foo.bz2", "w");
bzwrite($bz, $str, strlen($str));
bzclose($bz);
?>
```

Lásd még: bzread() és bzopen()!

VI. Naptár függvények

A naptár függvények csak akkor elérhetőek, ha a PHP forráskönyvtárad alatt található ’dl’ vagy ’ext’ könyvtárakban található naptár kiterjesztést lefordítottad. Mielőtt használnád, olvasd el a README nevű állományt is.

A naptár kiterjesztés egy sereg függvényt bocsát rendelkezésre két naptárformátum egyszerű konvertálásához. A közvetítő vagy standard a Julián dátumon alapul. A Julián dátum egy elég korai dátumtól kezdődik, ezért nem akadhat vele probléma (kb i.e. 4000). A naptárrendszerek közti átszámoláshoz először Julián dátumra kell átszámítani a dátumot, majd a választott rendszerre. A Julián dátum nagyon különbözik a Julián-féle naptártól! A naptárrendszerek információjához látogass el a http://genealogy.org/~scottlee/cal-overview.html címre. Erről az oldalról néhány idézett kivonat megtalálható ebben az instrukcióban.

Telepítés

E funkciók csak akkor működnek, ha a PHP-t `--enable-calendar` kapcsolóval fordítottad.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előredefiniált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

cal_days_in_month (PHP 4 >= 4.1.0)

A megadott hónap napjainak számával tér vissza az adott naptár adott évében

int **cal_days_in_month** (int calendar, int month, int year) \linebreak

Ez a függvény a napok számával tér vissza, `month` hónapban, `year` évben, és a `calendar` naptár paraméterekben megadott értékek szerint.

Lásd még: jdtounix()!

Megjegyzés: Ez a függvény csak a PHP 4RC1 verzió óta használható.

cal_from_jd (PHP 4 >= 4.1.0)

Julián naptárból egy támogatott naptártípusba konvertál és további információkat szolgáltat

array **cal_from_jd** (int jd, int calendar) \linebreak

cal_info (PHP 4 >= 4.1.0)

Különleges naptártípusokról ad információt

array **cal_info** (int calendar) \linebreak

cal_to_jd (PHP 4 >= 4.1.0)

Egy támogatott naptártípusból Julián naptárba konvertál

int **cal_to_jd** (int calendar, int month, int day, int year) \linebreak

easter_date (PHP 3>= 3.0.9, PHP 4)

Kiszámolja egy adott év húsvétjának UNIX időbélyeggel megadott éjféli időpontját

int **easter_date** (int year) \linebreak

Visszaadja az adott év húsvét éjfélt meghatározó UNIX időbélyeget (timestamp). Ha nincs megadva, az aktuális év a feltételezett.

Figyelem: A függvény figyelmeztetést generál, ha az év kivül esik az UNIX dátum értelmezési tartományán (1970 előtti vagy 2037 utáni év).

Példa 1. easter_date() példa

```
echo date ("M-d-Y", easter_date(1999));        /* "Apr-04-1999" */
echo date ("M-d-Y", easter_date(2000));        /* "Apr-23-2000" */
echo date ("M-d-Y", easter_date(2001));        /* "Apr-15-2001" */
```

Húsvét napját a Niceai Zsinat i.sz. 325-ben a tavaszi napéjegyenlőség napjára eső vagy az azutáni első telehold utáni vasárnapban határozta meg. A tavaszi napéjegyenlőséget mindig március 21-re fetételezték, hogy a számítást csökkentsék a telehold és az azt követő vasárnap meghatározására. Az algoritmust kb. 532-ben vezette be Dionyius Exiguus. A Julián naptár szerint (1753 előtti évekre vonatkoztatva) egy egyszerű 19 éves ciklust használtak a hold fázisainak nyomonkövetésére. A Clavius és Lillius által kigondolt, XIII. Gergely pápa által bevezetett és Nagy-Britanniában és gyarmatain 1752 szeptember 22.-én életbe lépett Gergely naptár szerint (1753 utáni évekre vonatkoztatva) két korrekciós tényezőt is belevettek a ciklus még pontosabbá tételéhez.

(A kód Simon Kershaw C programján alapul, <webmaster@ely.anglican.org>)

Lásd az easter_days() függvényt a húsvét kiszámításához 1970 elött és 2037 után.

easter_days (PHP 3>= 3.0.9, PHP 4)

Kiszámolja adott évben a március 21-től húsvétig terjedő napok számát

int **easter_days** (int year) \linebreak

Kiszámolja adott évben a március 21-től húsvétig terjedő napok számát. Ha nem adod meg a *year* paramétert, az aktuális évet veszi alapul.

Ez a függvény használható az easter_date() helyett, hogy kiszámolja a UNIX értelmezési tartományon kívülre eső évekre (i.sz. 1970 előtt vagy 2037 után) húsvét napját.

Példa 1. easter_days() példa

```
echo easter_days (1999);        /* 14, i.e. April 4   */
echo easter_days (1492);        /* 32, i.e. April 22  */
echo easter_days (1913);        /*  2, i.e. March 23  */
```

Húsvét napját a Niceai Zsinat i.sz. 325-ben a tavaszi napéjegyenlőség napjára eső vagy az azutáni első telehold utáni vasárnapban határozta meg. A tavaszi napéjegyenlőséget mindig március 21-re fetételezték, hogy a számítást csökkentsék a telehold és az azt követő vasárnap meghatározására. Az algoritmust kb. 532-ben vezette be Dionyius Exiguus. A Julián naptár szerint (1753 előtti évekre vonatkoztatva) egy egyszerű 19 éves ciklust használtak a hold fázisainak nyomonkövetésére. A Clavius és Lillius által kigondolt, XIII. Gergely pápa által bevezetett és Nagy-Britanniában és gyarmatain 1752 szeptember 22.-én életbe lépett Gergely naptár szerint (1753 utáni évekre vonatkoztatva) két korrekciós tényezőt is belevettek a ciklus még pontosabbá tételéhez.

(A kód Simon Kershaw C programján alapul, <webmaster@ely.anglican.org>)

Lásd még: easter_date()!

FrenchToJD (PHP 3, PHP 4)

Francia Köztársasági naptárt konvertál Julián dátumba

int **frenchtojd** (int month, int day, int year) \linebreak

Francia köztársasági dátumból konvertál egy dátumot Julián dátumba.

Ezek a műveletek csak az 1-14 év közötti dátumokat számítják át. (Gergely naptár szerint 1792 szeptember 22. - 1806 szeptember 22.). Ez hosszabb intervallum, mint ameddig a naptárt használták.

GregorianToJD (PHP 3, PHP 4)

Gergely dátumot konvertál Julián dátumba

int **gregoriantojd** (int month, int day, int year) \linebreak

A Gergely dátum értelmezési tartománya i.e. 4714 és i.sz. 9999 között van

Noha ez a szoftver a dátumokat i.e. 4714-ig képes kezelni, ilyen használata nem biztos, hogy értelmes adatot szolgáltat. A Gergely naptár 1582 október 15-től érvényes (vagy a Julián naptár szerint 1582 október 5-től). Néhány ország csak sokkal később fogadta el. Például Nagy-Britannia 1752-ben, a Szovjetúnió 1918-ban, Görögország 1923-ban ban tért át. A legtöbb európai ország a Julián naptárt használta a Gergely naptár előtt.

Példa 1. Naptár függvények

```
<?php
$jd = GregorianToJD (10,11,1970);
echo "$jd\n";
$gregorian = JDToGregorian ($jd);
echo "$gregorian\n";
?>
```

JDDayOfWeek (PHP 3, PHP 4)

Visszaadja a hét egy napjának nevét

mixed **jddayofweek** (int julianday, int mode) \linebreak

Visszaadja a hét egy napjának nevét. Sztringet vagy egészt adhat vissza a *mode* paramétertól függően.

Táblázat 1. Naptár hét módok

mode	jelentés
0	Visszaad egy egész típusú napsorszámot (0=vasárnap, 1=hétfő. stb.)
1	Visszaadja a napnevet egy sztringben (angol - Gergely naptári)
2	Visszaadja a rövidített napnevet egy sztringben (angol - Gergely naptári)

JDMonthName (PHP 3, PHP 4)

Visszaadja egy hónap nevét

string **jdmonthname** (int julianday, int mode) \linebreak

Visszaadja a hónap nevét tartalmazó sztringet. A *mode* megadja a függvénynek, hogy milyen dátumtípusba konvertálja a Julián dátumot és milyen típusú hónapneveket adjon vissza.

Táblázat 1. Naptár módok

mode	jelentés	értékek
0	Gergely naptár - rövidített	Jan, Feb, Mar, Apr
1	Gergely naptár	January, February,
2	Julián naptár - rövidített	Jan, Feb, Mar, Apr
3	Julián naptár	January, February,
4	Zsidó naptár	Tishri, Heshvan, K
5	Francia Köztársasági	Vendemiaire, Brum

JDToFrench (PHP 3, PHP 4)

Julián dátumot konvertál Francia Köztársasági naptárba

string **jdtofrench** (int juliandaycount) \linebreak

Julián dátumot konvertál Francia Köztársasági naptárba.

JDToGregorian (PHP 3, PHP 4)

Julián dátumot konvertál Gergely dátumba

string **jdtogregorian** (int julianday) \linebreak

Julián dátumot konvertál Gergely dátumot leíró formába ("hónap/nap/év").

JDToJewish (PHP 3, PHP 4)

Julián dátumot konvertál Zsidó naptárba

string **jdtojewish** (int julianday) \linebreak

Julián dátumot konvertál Zsidó naptárba.

JDToJulian (PHP 3, PHP 4)

Julián dátumot konvertál Julian-féle naptári dátumba

string **jdtojulian** (int julianday) \linebreak

Julián dátumot konvertál Julián-féle naptári dátum formátumú sztringbe ("hónap/nap/év").

jdtounix (PHP 4)

Julián dátumot konvertál UNIX időpontba

int **jdtounix** (int jday) \linebreak

Ez a függvény egy UNIX időbélyeget ad vissza a megadott *jday* Julián naphoz, vagy false-al tér vissza, ha a *jday* nincs a UNIX értelmezési tartományán belül (Gergely naptár szerint 1970 és 2037 között, azaz 2440588 <= *jday* <= 2465342)

Lásd még:unixtojd()!

JewishToJD (PHP 3, PHP 4)

Zsidó naptárt konvertál Julián dátumba

int **jewishtojd** (int month, int day, int year) \linebreak

Éretelmezési tartomány: Noha ez a szoftver a dátumokat i.e. 3761-ig képes kezelni (ez az első év), ilyen használata nem biztos, hogy értelmes adatot szolgáltat.

A zsidó naptárat évezredek óta használják, de korábban nem volt módszer egy hónap kezdetének meghatározására. Az új hónap mindig újholdkor kezdődött.

JulianToJD (PHP 3, PHP 4)

Julián-féle naptári dátumot konvertál Julián dátumba

int **juliantojd** (int month, int day, int year) \linebreak

A Julián dátum értelmezési tartománya i.e. 4713 és i.sz. 9999 között van

Noha ez a szoftver a dátumokat i.e. 4713-ig képes kezelni, ilyen használata nem biztos, hogy értelmes adatot szolgáltat. A Julián naptárt i.e. 46-ban alkották, de teljes egészében csak i.sz. 8-ban vagy csak a IV. században lett bevezetve. Hasonlóan az év kezdete kultúráról kultúrára változott nem mindenhol fogadták el a januárt az első hónapnak.

unixtojd (PHP 4)

UNIX időbélyeget konvertál Julián dátumba

int **unixtojd** ([int timestamp]) \linebreak

Visszaadja a Unix *`timestamp`*-hez (1970 január 1. óta eltelt másodpercek), vagy a mai naphoz tartozó Julián dátumot, ha nincs megadott *`timestamp`* paraméter.

Lásd még: jdtounix()!

Megjegyzés: Ez a függvény csak a PHP 4RC1 verzió óta használható.

VII. CCVS függvények

Ezek a függvények a CCVS API használatát teszik lehetővé, amik a CCVS közvetlen használatát biztosítják a PHP szriptek számára. A CCVS a RedHat (http://www.redhat.com/) megoldása a köztes pontra ("middle-man") a bankkártyák feldolgozásában. Ez lehetőséget ad a bankkártyák kiegyenlítési eljárásainak kezelésére egy *nix rendszer és egy modem segítségével. E modul segítéségével közvetlenül kezelheted a bankkártyákat a CCVS rendszer segítségével. Az alább olvasható referencia megmutatja, hogyan.

Ahhoz, hogy bekapcsold a PHP CCVS támogatását, először ellenőrizd a CCVS telepítési könyvtáradat. Utána a PHP fordításakor add meg a configure-nak a `--with-ccvs` paramétert. Ha a CCVS telepítési könyvtár megadása nélkül használod ezt a paramétert, a PHP az alapbeállítású CCVS elérési utat veszi alapul (/usr/local/ccvs). Ha a CCVS nem a hagyományos könyvtárban található, a `--with-ccvs=$ccvs_eleresi_ut` megoldást használd, ahol a $ccvs_eleresi_ut a CCVS telepítési könyvtára. Fontos megjegyezni, hogy a CCVS támogatáshoz elengedhetetlen, hogy a $ccvs_eleresi_ut/lib és a $ccvs_eleresi_ut/include létezik, és a cv_api.h magtalálható az include könyvtárban, a libccvs.a pedig a lib könyvtárban.

Ráadásul egy ccvsd a PHP-vel megegyező beállításokkal kell, hogy fusson. Úgyszintén meg kell győződnöd arról is, hogy a PHP processek ugyanazzal a userid-vel futnak, amivel a CCVS-t telepítetted. Ha például a CCVS-t a ’ccvs’ user-rel telepítetted, a PHP processek a ’ccvs’ felhasználói név alatt kell, hogy fussanak.

A CCVS-ről további információ található a http://www.redhat.com/products/ccvs címen.

Ez a dokumentáció még fejlesztések előtt áll. Addig is a Radhat által fenntartott, kicsit aktualitását vesztett, de hasznos dokumentációját tudjuk ajánlani a http://www.redhat.com/products/ccvs/support/CCVS3.3docs/ProgPHP.html címen.

A CCVS fejlesztését a Red Hat már nem végzi, és nincsenek tervek a további folytatásra. Azoknak, akik helyettesítő terméket keresnek, a Main Street Softworks MCVE (http://www.mcve.com/) termékét ajánljuk, mint lehetséges helyettesítőt. Hasonló a felépítése, és dokumentált PHP támogatással rendelkezik.

ccvs_add (PHP 4 >= 4.0.2)

Adat hozzáadása tranzakcióhoz

string **ccvs_add** (string session, string invoice, string argtype, string argval) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_auth (PHP 4 >= 4.0.2)

Tranzakció kredit azonosítás ellenőrzése

string **ccvs_auth** (string session, string invoice) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_command (PHP 4 >= 4.0.2)

Egy konkrét protokollal végrehajtható parancs futtatása, amit az általános CCVS API nem tud lekezelni

string **ccvs_command** (string session, string type, string argval) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_count (PHP 4 >= 4.0.2)

Tranzakció típusok szerinti összegzés a tranzakciók számáról

int **ccvs_count** (string session, string type) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_delete (PHP 4 >= 4.0.2)

Tranzakció törlése

string **ccvs_delete** (string session, string invoice) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_done (PHP 4 >= 4.0.2)

A CCVS motor megállítása, és felaszabadítás

string **ccvs_done** (string sess) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_init (PHP 4 >= 4.0.2)

CCVS beállítása használatra

string **ccvs_init** (string name) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_lookup (PHP 4 >= 4.0.2)

Egy megadott típusú elem megkeresése a megadott számú adatbázisban

string **ccvs_lookup** (string session, string invoice, int inum) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_new (PHP 4 >= 4.0.2)

Új, üres trantakció létrehozása

string **ccvs_new** (string session, string invoice) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_report (PHP 4 >= 4.0.2)

Egy háttérben futó kommunikációs folyamat állapotát adja vissza

string **ccvs_report** (string session, string type) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_return (PHP 4 >= 4.0.2)

Átutalás a kereskedőtől a kártyatulajdonos számára

string **ccvs_return** (string session, string invoice) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_reverse (PHP 4 >= 4.0.2)

Kezelt tranzakció teljes visszavonása

string **ccvs_reverse** (string session, string invoice) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_sale (PHP 4 >= 4.0.2)

Átutalás a kártyatulajdonostól a kereskedőnek

string **ccvs_sale** (string session, string invoice) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_status (PHP 4 >= 4.0.2)

Számla állapot ellenőrzése

string **ccvs_status** (string session, string invoice) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_textvalue (PHP 4 >= 4.0.2)

A legutóbbi függvényhívás szöveges visszetérési értéke

string **ccvs_textvalue** (string session) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ccvs_void (PHP 4 >= 4.0.2)

Befejezett tranzakció teljes visszavonása

string **ccvs_void** (string session, string invoice) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

VIII. COM támogató függvények Windowshoz

Ezek a függvények csak Windows alatt érhetőek el, a PHP 4-es verziójában kerültek bevezetésre.

COM (unknown)

COM class

Áttekintés

```
$obj = new COM("server.object")
```

The COM class provides a framework to integrate (D)COM components into your php scripts.

string **COM::COM** (string module_name [, string server_name [, int codepage]]) \linebreak

COM class constructor. Parameters:

module_name

name or class-id of the requested component.

server_name

name of the DCOM server from which the component should be fetched. If `NULL`, `localhost` is assumed. To allow DCOM `com.allow_dcom` has to be set to `TRUE` in `php.ini`.

codepage

specifies the codepage that is used to convert php-strings to unicode-strings and vice versa. Possible values are `CP_ACP`, `CP_MACCP`, `CP_OEMCP`, `CP_SYMBOL`, `CP_THREAD_ACP`, `CP_UTF7` and `CP_UTF8`.

Példa 1. COM example (1)

```
// starting word
$word = new COM("word.application") or die("Unable to instanciate Word");
print "Loaded Word, version {$word->Version}\n";

//bring it to front
$word->Visible = 1;

//open an empty document
$word->Documents->Add();

//do some weird stuff
$word->Selection->TypeText("This is a test...");
$word->Documents[1]->SaveAs("Useless test.doc");

//closing word
$word->Quit();

//free the object
$word->Release();
$word = null;
```

Példa 2. COM example (2)

```
$conn = new COM("ADODB.Connection") or die("Cannot start ADO");
$conn->Open("Provider=SQLOLEDB; Data Source=localhost;
Initial Catalog=database; User ID=user; Password=password");

$rs = $conn->Execute("SELECT * FROM sometable");    // Recordset

$num_columns = $rs->Fields->Count();
echo $num_columns . "\n";

for ($i=0; $i < $num_columns; $i++)
{
    $fld[$i] = $rs->Fields($i);
}

$rowcount = 0;
while (!$rs->EOF)
{
    for ($i=0; $i < $num_columns; $i++)
    {
        echo $fld[$i]->value . "\t";
    }
    echo "\n";
    $rowcount++;            // increments rowcount
    $rs->MoveNext();
}

$rs->Close();
$conn->Close();

$rs->Release();
$conn->Release();

$rs = null;
$conn = null;
```

VARIANT (unknown)

VARIANT class

Áttekintés

```
$vVar = new VARIANT($var)
```

A simple container to wrap variables into VARIANT structures.

string **VARIANT::VARIANT** ([mixed value [, int type [, int codepage]]]) \linebreak

VARIANT class constructor. Parameters:

value

initial value. if omitted an VT_EMPTY object is created.

type

specifies the content type of the VARIANT object. Possible values are `VT_UI1`, `VT_UI2`, `VT_UI4`, `VT_I1`, `VT_I2`, `VT_I4`, `VT_R4`, `VT_R8`, `VT_INT`, `VT_UINT`, `VT_BOOL`, `VT_ERROR`, `VT_CY`, `VT_DATE`, `VT_BSTR`, `VT_DECIMAL`, `VT_UNKNOWN`, `VT_DISPATCH` and `VT_VARIANT`. These values are mutual exclusive, but they can be combined with `VT_BYREF` to specify being a value. If omitted, the type of *`value`* is used. Consult the msdn library for additional information.

codepage

specifies the codepage that is used to convert php-strings to unicode-strings and vice versa. Possible values are `CP_ACP`, `CP_MACCP`, `CP_OEMCP`, `CP_SYMBOL`, `CP_THREAD_ACP`, `CP_UTF7` and `CP_UTF8`.

com_addref (PHP 4 >= 4.1.0)

Increases the components reference counter.

void **com_addref** (void) \linebreak

Increases the components reference counter.

com_get (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

COM komponenes egy tulajdonságának értékét adja vissza

mixed **com_get** (resource com_object, string property) \linebreak

Visszaadja a *`property`* értékét a *`com_object`*-el megadott COM komponensből. `FALSE` értékkel tér vissza hiba esetén.

com_invoke (PHP 3>= 3.0.3)

Egy COM komponens egy metódusát hívja

mixed **com_invoke** (resource com_object, string function_name [, mixed function parameters, ...]) \linebreak

A **com_invoke()** a *com_object* által mutatott COM komponens megadott metódusát hívja meg. A *function_name* függvény visszatérési értékét adja sikeres futásnál, FALSE értékkel tér vissza hiba esetén.

com_isenum (PHP 4 >= 4.1.0)

Grabs an IEnumVariant

void **com_isenum** (object com_module) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

com_load_typelib (PHP 4 >= 4.1.0)

Loads a Typelib

void **com_load_typelib** (string typelib_name [, int case_insensitive]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

com_load (PHP 3>= 3.0.3)

Létrehoz egy új referenciát egy COM komponensre

string **com_load** (string module name [, string server name]) \linebreak

A **com_load()** egy új COM komponenst hoz létre, egy erre hivatkozó referenciát ad vissza. FALSE értékkel tér vissza hiba esetén.

com_propget (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

COM komponenes egy tulajdonságának értékét adja vissza

mixed **com_propget** (resource com_object, string property) \linebreak

Ez egy alternatív név (alias) a com_get() függvényre.

com_propput (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Értéket ad COM komponenes egy tulajdonságának

void **com_propput** (resource object, string property, mixed value) \linebreak

Ez egy alternatív név (alias) a com_set() függvényre.

com_propset (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Értéket ad COM komponenes egy tulajdonságának

void **com_propset** (resource object, string property, mixed value) \linebreak

Ez egy alternatív név (alias) a com_set() függvényre.

com_release (PHP 4 >= 4.1.0)

Decreases the components reference counter.

void **com_release** (void) \linebreak

Decreases the components reference counter.

com_set (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Értéket ad COM komponenes egy tulajdonságának

void **com_set** (resource com_object, string property, mixed value) \linebreak

Beállítja a *`com_object`* COM komponenes *`property`* tulajdonságának értékét. `TRUE` értékkel tér vissza siker esetén, `FLASE` értékkel, ha hiba történt.

IX. Osztály/Objektum függvények

Bevezetés

A függvényekről

Ezek a függvények PHP osztályokról és objektumpéldányokról adnak információkat, mint például: az objektum osztályának nevét (típusát), a tagváltozóit másnéven tulajdonságait, és metódusait is. Ezekkel a függvényekkel egy objektumnak nem csak az osztály-tagságát lehet megállapítani, hanem a származását is, azaz, melyik osztályt kiterjesztése az adott objektum osztály.

Egy használati példa

Ebben a példában definiálunk egy alap osztályt, és ennek egy kiterjesztését. Az alap osztály leír egy zöldséget (Zoldseg), meghatározva, hogy ehető-e vagy sem (eheto), és hogy milyen színű. Az alosztály a paraj (`Paraj`) hozzáad egy új metódust, hogy megfőzhesd, és egy másikat, hogy megállapítsd, hogy főtt-e.

Példa 1. classes.inc

```
<?php
// alap osztály tulajdonságokkal és metódusokkal
class Zoldseg {

  var $eheto;
  var $szin;

  function Zoldseg ( $eheto, $szin="zöld" ) {
    $this->eheto = $eheto;
    $this->szin = $szin;
  }

  function eheto_e() {
    return $this->eheto;
  }

  function milyen_szinu() {
    return $this->szin;
  }

} // vége a Zoldseg osztálynak

// kiterjesztjük az alap osztályt
class Paraj extends Zoldseg {

  var $megfozve = false;

  function Paraj() {
    $this->Zoldseg ( true, "zöld" );
```

```
  }

  function fozes() {
    $this->megfozve = true;
  }

  function fott_e() {
    return $this->megfozve;
  }

} // vége a Paraj osztéynak

?>
```

Most két példányt létrehozunk a fenti osztályokból, és információkat írunk ki rólunk, beleértve az származásukat is. Definiálunk néhány hasznos függvényt, főleg azért, hogy ezeket az információkat szépen írjuk ki.

Példa 2. test_script.php

```
<pre>
<?php

include "classes.inc";

// hasznos függvények

function valtozok_kiirasa($obj) {
  $tomb = get_object_vars($obj);
  while (list($tul, $ertek) = each($tomb))
    echo "\t$tul = $ertek\n";
}

function metodusok_kiirasa($obj) {
  $tomb = get_class_methods(get_class($obj));
  foreach ($tomb as $metodus)
    echo "\tfunction $metodus()\n";
}

function os_osztaly($obj, $osztaly) {
  global $$obj;
  if (is_subclass_of($$obj, $osztaly)) {
    echo "A $obj a " . get_class($$obj) . " osztályhoz tartozik, ";
    echo "ami a $class egy alosztálya.\n";
  } else {
    echo "A $obj nem tartozik $class egy alosztályához.\n";
  }
}

// két objektumpéldány létrehozása

$karalabe  = new Zoldseg(true,"kék");
```

```
$soklevelu = new Paraj();

// objektuminformációk kiírása
echo "karalabe: CLASS " . get_class($karalabe) . "\n";
echo "soklevelu: CLASS " . get_class($soklevelu);
echo " PARENT " . get_parent_class($soklevelu) . "\n";

// a karalábé tulajdonságai
echo "\nkaralabe: Tulajdonságok\n";
valtozok_kiirasa($karalabe);

// a soklevelű paraj metódusai
echo "\nsoklevelu: Metódusok\n";
metodusok_kiirasa($soklevelu);

echo "\nSzülők:\n";
os_osztaly("soklevelu", "Paraj");
os_osztaly("soklevelu", "Zoldseg");
?>
</pre>
```

Fontos észrevenni, hogy a fenti példában a `$soklevelu` a `Paraj` objektum egy példánya, amely a `Zoldseg` egy alosztálya objektumnak, ezért a programunk utolsó része a következőt írja ki (szerencsére nincs probléma a névelőkkel :)

```
     [...]
Szülők:
A soklevelu nem tartozik Paraj egy alosztályához.
A soklevelu a paraj osztályhoz tartozik, ami a Zoldseg egy alosztálya.
```

call_user_method_array (PHP 4 >= 4.0.5)

meghívja az adott objektum egy metódusát paramétertömbbel [ellenjavallt]

mixed **call_user_method_array** (string method_name, object obj [, array paramarr]) \linebreak

> **Figyelem**
>
> A **call_user_method_array()** függvény PHP 4.1.0-től kezdve ellenjavallt. Helyette a call_user_func_array() függvényt kell használni a függvénynév paraméterében az `array(&$obj, "method_name")` szintaxissal.

Meghívja a felhasználó által definiált *`obj`* objektum *`method_name`* paraméterben megnevezett metódusát. A metóusnak átadandó paramétereket a *`paramarr`* tömbből tölti fel sorjában.

Lásd még: call_user_func_array(), call_user_func() és call_user_method()!

Megjegyzés: Ez a függvény a PHP 4.0.4pl1 kiadása után került a CVS-be.

call_user_method (PHP 3>= 3.0.3, PHP 4)

Egy megadott objektumon belül meghív egy függvényt [ellenjavallt]

mixed **call_user_method** (string method_name, object obj [, mixed parameter [, mixed ...]]) \linebreak

> **Figyelem**
>
> A **call_user_method()** függvény PHP 4.1.0-től kezdve ellenjavallt. Helyette a call_user_func() függvényt kell használni a függvénynév paraméterében az `array(&$obj, "method_name")` szintaxissal.

Meghívja a felhasználó által definiált *`obj`* objektum *`method_name`* paraméterben megnevezett metódusát. Az alábbi felhasználási példában definiálunk egy osztályt, létrehozunk egy objektumpéldányt, és a **call_user_method()** függvényt használjuk arra, hogy indirekten meghívjuk az objektum `print_info` metódusát.

```
<?php
class Orszag {
  var $NEV;
  var $TLD;

  function Orszag($nev, $tld) {
    $this->NEV = $nev;
    $this->TLD = $tld;
  }

  function print_info($prestr="") {
    echo $prestr."Ország: ".$this->NEV."\n";
```

```
    echo $prestr."Top Level Domain: ".$this->TLD."\n";
  }
}

$orsz = new Orszag("Peru","pe");

echo "* Az objektum metódusának közvetlen hívása\n";
$orsz->print_info();

echo "\n* Ugyanannak a közvetett meghívása\n";
call_user_method ("print_info", $orsz, "\t");
?>
```

Lásd még: call_user_func()!

class_exists (PHP 4)

megvizsgálja, hogy definiált-e az osztály

bool **class_exists** (string class_name) \linebreak

Ez a függvény TRUE-val tér vissza, ha a *class_name* osztály definiálva van, máskülönben FALSE-szal.

get_class_methods (PHP 4)

Osztálymetódusnevek tömbjével tér vissza.

array **get_class_methods** (mixed class_name) \linebreak

Ez a függvény a *class_name* által megadott osztály metódusainak nevei adja vissza egy tömbben.

Megjegyzés: PHP 4.0.6-tól kezdve, *class_name* osztálynév helyett közvetlenül az objektumot is át lehet adni paraméterként, például:

```
$metodusok = get_class_methods($en_objektumom); // lásd lent a teljes példát!
```

Példa 1. get_class_methods() példa

```
<?php

class en_osztalyom {
    // konstruktor
    function en_osztalyom() {
        return(true);
    }

    // 1. metódus
    function en_methodusom1() {
        return(true);
    }

    // 2. metódus
    function en_methodusom2() {
        return(true);
    }
}

$en_objektumom  = new en_osztalyom();

$methodus_nevek = get_class_methods(get_class($en_objektumom));

foreach ($metodus_nevek as $metodus_nev) {
    echo "$metodus_nev\n";
}

?>
```

A fenti példa a következőt írja ki:

```
en_osztalyom
en_methodusom1
en_methodusom2
```

Lásd még: get_class_vars() és get_object_vars()!

get_class_vars (PHP 4)

visszaadja az osztály alaptulajdonságait egy tömbben

array **get_class_vars** (string class_name) \linebreak

Ez a függvény az osztály alaptulajdonságainak tömbjével tér vissza.

Megjegyzés: Inicializálatlan - kezdő érték nélküli - alaptulajdonságokat (tagváltozókat) nem ad vissza a **get_class_vars()**.

Példa 1. get_class_vars() példa

```
<?php

class en_osztalyom {

    var $var1; // ennek nincs kezdő értéke...
    var $var2 = "xyz";
    var $var3 = 100;

    // konstruktor
    function en_osztalyom() {
        return(true);
    }

}

$en_objektumom      = new en_osztalyom();

$alap_tulajdonsagok = get_class_vars(get_class($en_objektumom));

foreach ( $alap_tulajdonsagok as $nev => $ertek ) {
    echo "$nev : $ertek\n";
}

?>
```

A fenti példa a következőt írja ki:

```
var2 : xyz
var3 : 100
```

Lásd még: get_class_methods() és get_object_vars()!

get_class (PHP 4)

visszaadja egy objektum osztályának a nevét

string **get_class** (object obj) \linebreak

Ez a függvény visszaadja annak az osztálynak a nevét, amelynek az *obj* objektum egy példánya.

> **Megjegyzés: get_class()** a felhasználó által definiált osztályok neveit mindig csupa kisbetűsként adja vissza, míg a kiterjesztésekben definiáltakat az eredeti elnevezésők szerint.

Lásd még: get_parent_class(), gettype() és is_subclass_of()!

get_declared_classes (PHP 4)

visszaadja a jelenlegi szkriptben definiált osztályok neveit

array **get_declared_classes** (void) \linebreak

Ez a függvény az aktuális szkriptben definiált osztályok neveit adja vissza egy tömbben.

> **Megjegyzés:** A PHP 4.0.1pl2-ben, három extra osztályt ad vissza a függvény a tömb elején `stdClass` (ami a `Zend/zend.c`-ben definiált), `OverloadedTestClass` (ami az `ext/standard/basic_functions.c`-ben definiált) és `Directory` (ami az `ext/standard/dir.c`-ben definiált).
>
> Attól függően, hogy milyen kiterjesztések vannak betöltve, egyéb osztályok is megjelenhetnek a listában. Ez azt jelenti, hogy saját osztályokat ilyen nevekkel már nem lehet definiálni. A függelékben vannak felsorolva az előre definiált osztályok hasonló című fejezetben.

get_object_vars (PHP 4)

Objektumtulajdonságok asszociatív tömbjével tér vissza

array **get_object_vars** (object obj) \linebreak

Ez a függvény az *obj* paraméterben átadott objektumban definiált tulajdonságokat (tagváltozókat) és értékeiket adja vissza egy asszociatív tömbben. Az *obj* típusát jelentő osztályban deklarált, de kezdőérték nélküli tagváltozókat nem adja vissza a függvény.

Példa 1. get_object_vars() használata

```
<?php
class Pont2D {
    var $x, $y;
    var $cimke;

    function Pont2D($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }

    function Cimkez($cimke) {
        $this->cimke = $cimke;
```

```
    }

    function holaPont() {
        return array("x" => $this->x,
                     "y" => $this->y,
                     "cimke" => $this->cimke);
    }
}

// $cimke deklarálva van, de nincs érték hozzárendelve
$p1 = new Pont2D(1.233, 3.445);
print_r(get_object_vars($p1));

$p1->Cimkez("pont #1");
print_r(get_object_vars($p1));

?>
```

A fenti program a következőt írja ki:

```
Array
(
    [x] => 1.233
    [y] => 3.445
)

Array
(
    [x] => 1.233
    [y] => 3.445
    [cimke] => pont #1
)
```

Lásd még: get_class_methods() és get_class_vars()!

get_parent_class (PHP 4)

visszaadja egy objektum vagy osztály szülő osztályát

string **get_parent_class** (mixed obj) \linebreak

Ha *obj* objektum típusú, akkor visszaadja az *obj* példány "típusát" jelentő osztály szülőosztályának nevét.

Ha *obj* sztring típusú, akkor visszaadja az ilyen nevű osztály szülőosztályának nevét. Ez PHP 4.0.5-től kezdve használható.

Lásd még: get_class() és is_subclass_of()!

is_a (PHP 4 >= 4.2.0)

megvizsgálja, hogy az objektum leszármazottja vagy tagja-e egy osztálynak

bool **is_a** (object object, string class_name) \linebreak

Ez a függvény TRUE-t ad vissza, ha az *object* a *class_name* osztály vagy annak egy leszármazott osztályának példánya, egyébként FALSE-t ad vissza.

Lásd még: get_class(), get_parent_class() és is_subclass_of()!

is_subclass_of (PHP 4)

megviszgálja, hogy egy objektum egy megadott osztálynak egy alosztályához tartozik-e

bool **is_subclass_of** (object obj, string class_name) \linebreak

Ez a függvény TRUE-val tér vissza, ha az *obj* objektum olyan osztálynak egy példánya, amely alosztálya/leszármazottja a *class_name* nevű osztálynak. Másik esetben FALSE-t ad vissza.

Lásd még: get_class() és get_parent_class()!

method_exists (PHP 4)

ellenőrzi az osztálymetódus létezését

bool **method_exists** (object object, string method_name) \linebreak

Ez a függvény TRUE-val tér vissza, ha a *method_name* nevű metódus definiálva van az adott *object* objektumhoz, különben FALSE-szal.

X. ClibPDF függvények

A ClibPDF segítségével PDF dokumentumokat lehet létrehozni PHP-vel. Ez elérhető a FastIO (http://www.fastio.com/)-nál, de nem szabad használatú (free) szoftver. Mielőtt dolgozni kezdesz a ClibPDF-fel tanácsos elolvasnod a licence-t. Ha nem fogadod el, használhatod Thomas Merz pdflib-jét, amelyik szintén kiválóan alkalmazható. A ClibPDF és az API hasonló Thomas Merz pdflib-jéhez de a FastIO-nak köszönhetően a ClibPDF gyorsabb, és ráadásul kisebb dokumentumokat állít elő. Ez valószinűleg változni fog a pdflib 2.0 verziójában. Egy egyszerű benchmark (a pdflib 2.0-ből php-ba fordított a pdfclock.c példa) nem mutat különbséget a sebességben. A fájlméret is hasonló, ha kikapcsoljuk a tömörítést. Tehát próbáld ki mindkettőt és nézd meg melyik jobb neked.

Ezt a dokumentációt ajánlatos a ClibPDF kézikönyvvel együtt olvasni mivel az a függvényeket nagyobb részletességgel magyarázza.

Sok függvény az alap ClibPDF-ben, a PHP modulban, valamint a pdflib-ben a ugyanazzal a névvel rendelkezik. Az összes függvény, kivéve a cpdf_open() első paramétereként a dokumentum kezelőjét (handler) kéri. Jelenleg ez a kezelőt a PHP belsőleg nem használja, mivel a ClibPDF nem támogatja több PDF dokumentum egyidejű létrehjozását. Ezt most ne is próbáld ki, mert az eredményt nem lehet előre megjósolni. Nem tudhatom, hogy a többszálas környezetben ez milyen következnényekkel járna. A ClibPDF szerzője szerint ez változni fog a következő változatnál (az írás idején a jelenlegi verzió az 1.10). Ha mégis szükséged van erre a lehetőségre, használd a pdflib modult.

> **Megjegyzés:** A cpdf_set_font() függvény megváltozott a PHP 3 óta, hogy támogassa az ázsiai betűkészleteket. Az encoding paraméter már nem egy integer, hanem egy string!

A pdflib-el szemben van egy nagy előnye a ClibPDF-nek. PDF doumentumokat tud létrehozni a memóriában ideiglenes fájlok használata nélkül. Lehetőséget ad továbbá arra, hogy koordinátákat adj át egy előre definiált mértékegységben. Ez egy nagyon jó lehetőség de szimulálható a pdf_translate()-el.

A legtöbb függvény egyszerűen használható. A legnehezebb rész valószínűleg egy nagyon egyszerű PDF létrehozása. A következő példa segíthet az elindulásban. Ez egy egy oldalas dokumentumot készít. Az oldal tartalmazza a "Times-Roman" szöveget egy 30 pont méretű körvonalas betűtípusssal. A szöveg aláhúzott.

Példa 1. Egyszerű ClibPDF példa

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
cpdf_moveto($cpdf, 50, 740);
cpdf_lineto($cpdf, 330, 740);
cpdf_stroke($cpdf);
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
```

```
?>
```

A pdflib csomag egy összetettebb példát tartalmaz, amely egy oldalsorozatot állít elő egy analóg órával. Ez itt a ClibPDF kiterjesztést használó PHP-be konvertált változat:

Példa 2. pdfclock példa a pdflib 2.0 csomagból

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 40;

$pdf = cpdf_open(0);
cpdf_set_creator($pdf, "pdf_clock.php3");
cpdf_set_title($pdf, "Analog Clock");

while($pagecount-- > 0) {
  cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius + $margin), 2 * ($radius + $mar-
gin), 1.0);

  cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* wipe */

  cpdf_translate($pdf, $radius + $margin, $radius + $margin);
  cpdf_save($pdf);
  cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

  /* minute strokes */
  cpdf_setlinewidth($pdf, 2.0);
  for ($alpha = 0; $alpha < 360; $alpha += 6)
    {
    cpdf_rotate($pdf, 6.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin/3, 0.0);
    cpdf_stroke($pdf);
    }

  cpdf_restore($pdf);
  cpdf_save($pdf);

  /* 5 minute strokes */
  cpdf_setlinewidth($pdf, 3.0);
  for ($alpha = 0; $alpha < 360; $alpha += 30)
  {
    cpdf_rotate($pdf, 30.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin, 0.0);
    cpdf_stroke($pdf);
  }

  $ltime = getdate();

  /* draw hour hand */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['minutes']/60.0) + $ltime['hours'] - 3.0) * 30.0);
```

```
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius/2, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* draw minute hand */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds']/60.0) + $ltime['minutes'] - 15.0) * 6.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius * 0.8, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* draw second hand */
  cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
  cpdf_setlinewidth($pdf, 2);
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
  cpdf_moveto($pdf, -$radius/5, 0.0);
  cpdf_lineto($pdf, $radius, 0.0);
  cpdf_stroke($pdf);
  cpdf_restore($pdf);

  /* draw little circle at center */
  cpdf_circle($pdf, 0, 0, $radius/30);
  cpdf_fill($pdf);

  cpdf_restore($pdf);

  cpdf_finalize_page($pdf, $pagecount+1);
}

cpdf_finalize($pdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($pdf);
cpdf_close($pdf);
?>
```

cpdf_add_annotation (PHP 3>= 3.0.12, PHP 4)

Megjegyzés hozzáfűzése

void **cpdf_add_annotation** (int pdf document, double llx, double lly, double urx, double ury, string title, string content, int mode) \linebreak

A **cpdf_add_annotation()** függvény egy megjegyzést helyez el a bal alsó sarkával az (*llx*, *lly*) koordinátájú pontban és jobb felső sarkával az (*urx*, *ury*) pontban.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

cpdf_add_outline (PHP 3>= 3.0.9, PHP 4)

Könyvjelzőt helyez el az aktuális oldalon

void **cpdf_add_outline** (int pdf document, string text) \linebreak

A **cpdf_add_outline()** függvény a *text* szöveggel egy könyvjelzőt helyez el, amely az aktuális oldalra mutat.

Példa 1. Oldalvázlat hozzáadása

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "1. oldal");
// ...
// némi rajzolása
// ...
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

cpdf_arc (PHP 3>= 3.0.8, PHP 4)

Ívet rajzol

void **cpdf_arc** (int pdf document, double x-koor, double y-koor, double radius, double start, double end, int mode) \linebreak

A **cpdf_arc()** függvény ívet rajzol az (*x-koor*, *y-koor*) középpont köré *radius* sugárral, *start* kezdőszögtől *end* végszögig.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

Lásd még: cpdf_circle().

cpdf_begin_text (PHP 3>= 3.0.8, PHP 4)

Szöveg szekciót kezd el

void **cpdf_begin_text** (int pdf document) \linebreak

A **cpdf_begin_text()** függvény egy szöveg szekciót kezd el. Ennek a cpdf_end_text() függvény meghívásával kell végződnie.

Példa 1. Szöveg kimenet

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

Lásd még: cpdf_end_text().

cpdf_circle (PHP 3>= 3.0.8, PHP 4)

Kört rajzol

void **cpdf_circle** (int pdf document, double x-koor, double y-koor, double radius, int mode) \linebreak

A **cpdf_circle()** függvény egy kört rajzol az (*x-koor*, *y-koor*) középpont köré *radius* sugárral.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

Lásd még: cpdf_arc().

cpdf_clip (PHP 3>= 3.0.8, PHP 4)

Kivágja az aktuális alakzatot

void **cpdf_clip** (int pdf document) \linebreak

A **cpdf_clip()** függvény vágja az összes rajzot az aktuális alakzathoz.

cpdf_close (PHP 3>= 3.0.8, PHP 4)

Bezárja a pdf dokumentumot

void **cpdf_close** (int pdf document) \linebreak

A **cpdf_close()** függvény bezárja a pdf dokumentumot. Ajánlatos, hogy ez legyen az utolsó tevékenység a cpdf_finalize(), cpdf_output_buffer() és cpdf_save_to_file() függvények után.

Lásd még: cpdf_open().

cpdf_closepath_fill_stroke (PHP 3>= 3.0.8, PHP 4)

Bezárja, kitölti és körvonalazza az aktuális alakzatot

void **cpdf_closepath_fill_stroke** (int pdf document) \linebreak

A **cpdf_closepath_fill_stroke()** függvény bezárja, kitölti az aktuális alakzatot az aktuális színező színnel és kirajzolja azt.

Lásd még: cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_closepath_stroke (PHP 3>= 3.0.8, PHP 4)

Bezárja az alakzatot és körvonalat rajzol az alakzat mentén

void **cpdf_closepath_stroke** (int pdf document) \linebreak

A **cpdf_closepath_stroke()** függvény a cpdf_closepath() és a cpdf_stroke() függvények kombinációja. Törli az alakzatot.

Lásd még: cpdf_closepath(), cpdf_stroke().

cpdf_closepath (PHP 3>= 3.0.8, PHP 4)

Befejez egy alakzatot

void **cpdf_closepath** (int pdf document) \linebreak

A **cpdf_closepath()** lezárja az aktuális alakzatot.

cpdf_continue_text (PHP 3>= 3.0.8, PHP 4)

Szöveget helyez a következő sorba

void **cpdf_continue_text** (int pdf document, string text) \linebreak

A **cpdf_continue_text()** függvény a *text* stringben megadott szöveget a következő sorba helyezi.

Lásd még: cpdf_show_xy(), cpdf_text(), cpdf_set_leading(), cpdf_set_text_pos().

cpdf_curveto (PHP 3>= 3.0.8, PHP 4)

Görbét rajzol

void **cpdf_curveto** (int pdf document, double x1, double y1, double x2, double y2, double x3, double y3, int mode) \linebreak

A **cpdf_curveto()** függvény Bezier-görbét rajzol az aktuális pozícióból a (*x3*, *y3*) pontba, felhasználva az (*x1*, *y1*) és (*x2*, *y2*) pontokat, mint kontrollpontokat.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

Lásd még: cpdf_moveto(), cpdf_rmoveto(), cpdf_rlineto(), cpdf_lineto().

cpdf_end_text (PHP 3>= 3.0.8, PHP 4)

Befejezi a szöveg szekciót

void **cpdf_end_text** (int pdf document) \linebreak

A **cpdf_end_text()** függvény befejezi a szöveg szekciót amely a cpdf_begin_text() függvényel lett megkezdve.

Példa 1. Szöveg kimenet

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

Lásd még: cpdf_begin_text().

cpdf_fill_stroke (PHP 3>= 3.0.8, PHP 4)

Kitölti és körvonalazza az aktuális alakzatot

void **cpdf_fill_stroke** (int pdf document) \linebreak

A **cpdf_fill_stroke()** függvény kitölti az aktuális alakzat belsejét az aktuális színező színnel és kirajzolja azt.

Lásd még: cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_fill (PHP 3>= 3.0.8, PHP 4)

Kitölti az aktuális alakzatot

void **cpdf_fill** (int pdf document) \linebreak

A **cpdf_fill()** függvény kitölti az aktuális alakzat belsejét az aktuális színező szinnel.

Lásd még: cpdf_closepath(), cpdf_stroke(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_finalize_page (PHP 3>= 3.0.10, PHP 4)

Befejezi az oldalt

void **cpdf_finalize_page** (int pdf document, int page number) \linebreak

A **cpdf_finalize_page()** függvény befejzi a *`page number`* számú oldalt. Ez a függvény csak memóriafelszabadításra szolgál. A befejezett oldal kevesebb memóriát foglal el de tovább már nem módosítható.

Lásd még: cpdf_page_init().

cpdf_finalize (PHP 3>= 3.0.8, PHP 4)

Befejezi a dokumentumot

void **cpdf_finalize** (int pdf document) \linebreak

A **cpdf_finalize()** függvény befejezi a dokumentumot. Ezután a cpdf_close()-t még mindenképpen meg kell hívni.

Lásd még: cpdf_close().

cpdf_global_set_document_limits (PHP 4)

A pdf dokumentumra vonatkozó korlátozások beállítására szolgál

void **cpdf_global_set_document_limits** (int maxpages, int maxfonts, int maximages, int maxannotations, int maxobjects) \linebreak

A **cpdf_global_set_document_limits()** függvény számos dokumentummal kapcsolatos korlátozást állít. A függvényt a cpdf_open() függvény meghívása előtt kell meghívni, mert csak a függvényhívás után megnyitott dokumentumok korlátait állítja.

Lásd még a cpdf_open() függvényt!

cpdf_import_jpeg (PHP 3>= 3.0.9, PHP 4)

Megnyit egy JPEG képet

int **cpdf_open_jpeg** (int pdf document, string file name, double x-koor, double y-koor, double angle, double width, double height, double x-scale, double y-scale, int mode) \linebreak

A **cpdf_import_jpeg()** függvény megnyit egy, a *file name* nevű fájlban tárolt képet. A képformátum kötelezően jpeg. A képet az aktuális oldal (*x-koor*, *y-koor*) koordinátájú pontjában helyezi el. A kép elforgatható *angle* fokkal.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

Lásd még: cpdf_place_inline_image(),

cpdf_lineto (PHP 3>= 3.0.8, PHP 4)

Egyenest rajzol

void **cpdf_lineto** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

A **cpdf_lineto()** függvény vonalat rajzol az aktuális pozícióból az (*x-koor*, *y-koor*) koordinátájú pontba.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

Lásd még: cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto().

cpdf_moveto (PHP 3>= 3.0.8, PHP 4)

Beállítja az aktuális pontot

void **cpdf_moveto** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

A **cpdf_moveto()** függvény beállítja az aktuális pozíciót az *x-koor* és *y-koor* koordinátájú pontra.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

cpdf_newpath (PHP 3>= 3.0.9, PHP 4)

Új alakzatot kezd

void **cpdf_newpath** (int pdf_document) \linebreak

A **cpdf_newpath()** új alakzatot kezd a *pdf_document* által megadott dokumentumban.

cpdf_open (PHP 3>= 3.0.8, PHP 4)

Megnyit egy új pdf dokumentumot

int **cpdf_open** (int compression, string filename) \linebreak

A **cpdf_open()** függvény megnyit egy új pdf dokumentumot. Az első paraméter bekapcsolja a a dokumentum tömörítést, ha nem egyenlő NULLával. A második elhagyható paraméter beállítja azt a fájlt, amelybe a dokumentum kerül. Ha elhagyjuk, akkor a memóriában készül el és a cpdf_save_to_file() függvénnyel fájlba írható vagy a standard kimenetre küldhető a cpdf_output_buffer()-el.

Megjegyzés: A visszaadott értékre a ClibPDF későbbi verzióinál mint első paraméterre szükség lesz minden más pdf dokumentumba író függvény esetében.

A ClibPDF könyvtár a "-" filenevet a standard kimenet szinonímájaként veszi. Ha a PHP egy apache modulként van installálva, ez nem működik, mivel a ClibPDF kiiratási módja a starndard kimenetre nem működik együtt az apache-al. Megoldhatod a problémát, ha kihagyod a filenevet és a cpdf_output_buffer() függvényt használod a pdf kiiratásához.

Lásd még: cpdf_close(), cpdf_output_buffer().

cpdf_output_buffer (PHP 3>= 3.0.9, PHP 4)

A memóriapufferből kiiratja a pdf dokumentumot

void **cpdf_output_buffer** (int pdf document) \linebreak

A **cpdf_output_buffer()** függvény a standard kimenetre küldi a pdf dokumentumot. Ehhez a dokumentumot a memóriában kell elkészíteni, azaz a cpdf_open() hívásnál nem kell filenév paramétert megadni.

Lásd még: cpdf_open().

cpdf_page_init (PHP 3>= 3.0.8, PHP 4)

Új oldalt kezd

void **cpdf_page_init** (int pdf document, int page number, int orientation, double height, double width, double unit) \linebreak

A **cpdf_page_init()** függvény új oldalt kezd *`height`* magassággal és *`width`* szélességgel. Az oldal száma *`page number`* és orientációja *`orientation`*. Az *`orientation`* paraméter 0 álló és 1 fekvő irányítottság esetén. Az utolsó elhagyható *`unit`* paraméter beállítja a koordinátarendszer egységét. Értéke a postcript pontok száma egységenként. Mivel egy hüvelyk (inch) 72 ponttal egyenlő, a 72 érték felel meg egy hüvelyknek. Az alapértelmezés szintén a 72.

Lásd még: cpdf_set_current_page().

cpdf_place_inline_image (PHP 3>= 3.0.9, PHP 4)

Képet helyez el az oldalon

void **cpdf_place_inline_image** (int pdf document, int image, double x-koor, double y-koor, double angle, double width, double height, int mode) \linebreak

A **cpdf_place_inline_image()** függvény egy php képfüggvényekkel készített képet helyez el az oldalon az (*`x-koor`*, *`y-koor`*) pontban. A kép ezzel egyidőben méretezhető.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

Lásd még: cpdf_import_jpeg(),

cpdf_rect (PHP 3>= 3.0.8, PHP 4)

Téglalapot rajzol

void **cpdf_rect** (int pdf document, double x-koor, double y-koor, double width, double height, int mode) \linebreak

A **cpdf_rect()** függvény egy téglalapot rajzol, amelynek jobb alsó sarkának koordinátája (*`x-koor`*, *`y-koor`*) pont. A téglalap szélessége *`widgth`*, magassága *`height`*.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

cpdf_restore (PHP 3>= 3.0.8, PHP 4)

Visszaállítja a korábban elmentett környezetet

void **cpdf_restore** (int pdf document) \linebreak

A **cpdf_restore()** függvény visszaállítja a cpdf_save() függvénnyel elmentett környezetet. Ez úgy működik, mint a postscript grestore parancs. Nagyon hasznos, ha egy objektumot anélkül akarunk áthelyezni vagy elforgatni, hogy az más objektumokra is hatással legyen.

Példa 1. Save/Restore

```
<?php
cpdf_save($pdf);
// forgatások, transzformációk ...
cpdf_restore($pdf)
?>
```

Lásd még: cpdf_save().

cpdf_rlineto (PHP 3>= 3.0.9, PHP 4)

Egyenest rajzol

void **cpdf_rlineto** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

A **cpdf_rlineto()** függvény vonalat rajzol az aktuális pozícióból az (*x-koor*, *y-koor*) relatív koordinátájú ponthoz.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

Lásd még: cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto().

cpdf_rmoveto (PHP 3>= 3.0.9, PHP 4)

Beállítja az aktuális pontot

void **cpdf_rmoveto** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

A **cpdf_rmoveto()** függvény beállítja az aktuális pontot az *x-koor* és *y-koor* relatív koordinátájú ponthoz.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

Lásd még: cpdf_moveto().

cpdf_rotate_text (PHP 3>= 3.0.9, PHP 4)

Sets text rotation angle

void **cpdf_rotate_text** (int pdfdoc, float angle) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

cpdf_rotate (PHP 3>= 3.0.8, PHP 4)

Beállítja a forgatást

void **cpdf_rotate** (int pdf document, double angle) \linebreak

A **cpdf_rotate()** függvény beállítja az elforgatás szögét `angle` értékre.

cpdf_save_to_file (PHP 3>= 3.0.8, PHP 4)

A pdf dokumentumot fájlba írja

void **cpdf_save_to_file** (int pdf document, string filename) \linebreak

A **cpdf_save_to_file()** függvény fájlba írja a pdf dokumentumot ha ez korábban a memóriában készült el. Nem szükséges, ha a pdf dokumentumot úgy nyitottad meg, hogy adtál filenév paramétert a cpdf_open()-nek.

Lásd még: cpdf_output_buffer(), cpdf_open().

cpdf_save (PHP 3>= 3.0.8, PHP 4)

Elmenti az aktuális környezetet

void **cpdf_save** (int pdf document) \linebreak

A **cpdf_save()** függvény elmenti az aktuális környezetet. Ez úgy működik, mint a postscript gsave parancs. Nagyon hasznos, ha egy objektumot anélkül akarunk áthelyezni vagy elforgatni, hogy az más objektumokra is hatással legyen.

Lásd még: cpdf_restore().

cpdf_scale (PHP 3>= 3.0.8, PHP 4)

Beállítja a méretezést

void **cpdf_scale** (int pdf document, double x-scale, double y-scale) \linebreak

A **cpdf_scale()** függvény beállítja a méretezési faktort mindkét irányban.

cpdf_set_action_url (PHP 3>= 3.0.9, PHP 4)

Sets hyperlink

void **cpdf_set_action_url** (int pdfdoc, float xll, float yll, float xur, float xur, string url [, int mode]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

cpdf_set_char_spacing (PHP 3>= 3.0.8, PHP 4)

Beállítja a betűtávolságot

void **cpdf_set_char_spacing** (int pdf document, double space) \linebreak

A **cpdf_set_char_spacing()** függvény beállítja a karakterek közötti távolságot.

Lásd még: cpdf_set_word_spacing(), cpdf_set_leading().

cpdf_set_creator (PHP 3>= 3.0.8, PHP 4)

A pdf dokumentumban a létrehozó(creator) mezőt állítja

void **cpdf_set_creator** (string creator) \linebreak

A **cpdf_set_creator()** függvény a pdf dokumentum szerző mezőjét állítja.

Lásd még a cpdf_set_subject(), cpdf_set_title() és a cpdf_set_keywords() függvényt!

cpdf_set_current_page (PHP 3>= 3.0.9, PHP 4)

Beállítja az aktuális oldalt

void **cpdf_set_current_page** (int pdf document, int page number) \linebreak

A **cpdf_set_current_page()** függvény beállítja azt az oldalt, amelyre az összes további művelet vonatkozik. Egy oldal választható, ameddig nincs befejezve a cpdf_finalize_page()-al.

Lásd még: cpdf_finalize_page().

cpdf_set_font_directories (PHP 4 >= 4.0.6)

Sets directories to search when using external fonts

void **cpdf_set_font_directories** (int pdfdoc, string pfmdir, string pfbdir) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

cpdf_set_font_map_file (PHP 4 >= 4.0.6)

Sets fontname to filename translation map when using external fonts

void **cpdf_set_font_map_file** (int pdfdoc, string filename) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

cpdf_set_font (PHP 3>= 3.0.8, PHP 4)

Kiválaszja az aktuális betűtípust és méretet

void **cpdf_set_font** (int pdf document, string font name, double size, string encoding) \linebreak

A **cpdf_set_font()** függvény beállítja az aktuális betűtípust, betűméretet és kódolást. Jelenleg csak a standard postscript betűtípusok támogatottak. Az utolsó *`encoding`* paraméter a következő értékeket veheti fel: "MacRomanEncoding", "MacExpertEncoding", "WinAnsiEncoding", és "`NULL`". "`NULL`" jelenti a betűtípus beépített kódolását. Lásd a ClibPDF Kézikönyvet további információért főleg az ázsiai fontok támogatását illetően.

cpdf_set_horiz_scaling (PHP 3>= 3.0.8, PHP 4)

Beállítja a szöveg vízszintes méretezését

void **cpdf_set_horiz_scaling** (int pdf document, double scale) \linebreak

A **cpdf_set_horiz_scaling()** függvény beállítja a vízszintes méretezést a *`scale`* százalékra.

cpdf_set_keywords (PHP 3>= 3.0.8, PHP 4)

Beállítja a pdf dokumentum kulcsszavak mezőjét

void **cpdf_set_keywords** (string keywords) \linebreak

A **cpdf_set_keywords()** függvény beállítja a pdf dokumentum kulcsszavait.

Lásd még: cpdf_set_title(), cpdf_set_creator(), cpdf_set_subject().

cpdf_set_leading (PHP 3>= 3.0.8, PHP 4)

Beállítja a szöveg sortávolságát

void **cpdf_set leading** (int pdf document, double distance) \linebreak

A **cpdf_set_leading()** függvény beállítja a szövegsorok közötti távolságot. Ez a cpdf_continue_text() függvénnyel elhelyezett szövegnél használatos.

Lásd még: cpdf_continue_text().

cpdf_set_page_animation (PHP 3>= 3.0.9, PHP 4)

Beállítja az oldalak közti átmeneti időtartamot

void **cpdf_set_page_animation** (int pdf document, int transition, double duration) \linebreak

A **cpdf_set_page_animation()** függvény beállítja az átmenetet két egymást követő oldal között.

A *`transition`* értéke lehet

0: nincs átmenet,
1: két, a képernyőn végigseprő vonal fedi fel az új oldalt,
2: több, a képernyőn végigseprő vonal fedi fel az új oldalt,
3: egy négyzet fedi fel az új oldalt,
4: egy, a képernyőn végigseprő vonal fedi fel az új oldalt,
5: a régi oldal szétoszolva fedi fel az új oldalt,
6: egy szétoszló effektus mozog az egyik képernyőszéltől a másikig,
7: a régi oldal egyszerűen helyettesítődik az új oldallal (alapértelmezés)

A *`duration`* értéke a képek közötti átmenet másodperceinek száma.

cpdf_set_subject (PHP 3>= 3.0.8, PHP 4)

Beállítja a pdf dokumentum tárgy mezőjét

void **cpdf_set_subject** (string subject) \linebreak

A **cpdf_set_subject()** függvény beállítja a pdf dokumentum tárgyát.

Lásd még: cpdf_set_title(), cpdf_set_creator(), cpdf_set_keywords().

cpdf_set_text_matrix (PHP 3>= 3.0.8, PHP 4)

Beállítja a szövegmátrixot

void **cpdf_set_text_matrix** (int pdf document, array matrix) \linebreak

A **cpdf_set_text_matrix()** függvény beállít egy mátrixot, amely leír egy, az aktuális betűtípusra alkalmazott transzformációt.

cpdf_set_text_pos (PHP 3>= 3.0.8, PHP 4)

Beállítja a szövegpozíciót

void **cpdf_set_text_pos** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

A **cpdf_set_text_pos()** függvény beállítja a szöveg pozícióját a következő cpdf_show() függvényhíváshoz.

Az utolsó elhagyható *mode* paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postcript pontok szerint veszi figyelembe.

Lásd még: cpdf_show(), cpdf_text().

cpdf_set_text_rendering (PHP 3>= 3.0.8, PHP 4)

Meghatározza hogyan legyen a szöveg kirajzolva

void **cpdf_set_text_rendering** (int pdf document, int mode) \linebreak

A **cpdf_set_text_rendering()** függvény meghatározza, hogy hogyan legyen egy szöveg kirajzolva. A lehetséges értékek a *mode* paraméter részére: 0=szöveg kitöltés, 1=szöveg körvonala, 2=szöveg kitöltése és körvonala, 3=láthatatlan, 4=szövegkitöltés és kivágó alakzat készítése, 5=szöveg körvonala és kivágó alakzat készítése, 6=szöveg kitöltése, körvonala és kivágó alakzat készítése, 7=csak kivágó alakzat készítése.

cpdf_set_text_rise (PHP 3>= 3.0.8, PHP 4)

Beállítja a szöveg emelkedését

void **cpdf_set_text_rise** (int pdf document, double value) \linebreak

A **cpdf_set_text_rise()** függvény beállítja a szöveg emelkedését a *value* egységre.

cpdf_set_title (PHP 3>= 3.0.8, PHP 4)

A pdf dokumentumban a cím(title) mezőt állítja

void **cpdf_set_title** (string title) \linebreak

A **cpdf_set_title()** függvény a pdf dokumentum cím mezőjét állítja.

Lásd még a cpdf_set_subject(), cpdf_set_creator(), cpdf_set_keywords() függvényeket!

cpdf_set_viewer_preferences (PHP 3>= 3.0.9, PHP 4)

How to show the document in the viewer

void **cpdf_set_viewer_preferences** (int pdfdoc, array preferences) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

cpdf_set_word_spacing (PHP 3>= 3.0.8, PHP 4)

Beállítja a szótávolságot

void **cpdf_set_word_spacing** (int pdf document, double space) \linebreak

A **cpdf_set_word_spacing()** függvény beállítja a szavak közötti távolságot.

Lásd még: cpdf_set_char_spacing(), cpdf_set_leading().

cpdf_setdash (PHP 3>= 3.0.8, PHP 4)

Beállítja a szaggatott vonalmintát

void **cpdf_setdash** (int pdf document, double white, double black) \linebreak

A **cpdf_setdash()** függvény beállítja a szaggatott vonalminta `white` fehér és `black` fekete egységre. Ha mindkettő 0, akkor sima vonalat állít be.

cpdf_setflat (PHP 3>= 3.0.8, PHP 4)

Beállítja a simaságot

void **cpdf_setflat** (int pdf document, double value) \linebreak

A **cpdf_setflat()** függvény beállítja a simaságot egy 0 - 100 közötti értékre.

cpdf_setgray_fill (PHP 3>= 3.0.8, PHP 4)

Szürke értékre állítja a kitöltő színt

void **cpdf_setgray_fill** (int pdf document, double value) \linebreak

A **cpdf_setgray_fill()** függvény beállítja az aktuális szürke értéket egy alakzat kitöltése céljából.

Lásd még: cpdf_setrgbcolor_fill().

cpdf_setgray_stroke (PHP 3>= 3.0.8, PHP 4)

Szürke értékre állítja a körvonalrajzoló színt

void **cpdf_setgray_stroke** (int pdf document, double gray value) \linebreak

A **cpdf_setgray_stroke()** függvény az adott szürke értékre állítja az aktuális rajzoló színt.

Lásd még: cpdf_setrgbcolor_stroke().

cpdf_setgray (PHP 3>= 3.0.8, PHP 4)

Szürke értékre állítja a rajzoló és kitöltő színt

void **cpdf_setgray** (int pdf document, double gray value) \linebreak

A cpdf_setgray_stroke() függvény az adott szürke színre állítja a rajzoló és kitöltő színt.

Lásd még: cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

cpdf_setlinecap (PHP 3>= 3.0.8, PHP 4)

Beállítja a vonalcsúcs paramétert

void **cpdf_setlinecap** (int pdf document, int value) \linebreak

A **cpdf_setlinecap()** függvény beállítja a vonalcsúcs paramétert 0 - 2 közötti értékre. 0 = metszett, 1 = kerek, 2 = csapott négyszög.

cpdf_setlinejoin (PHP 3>= 3.0.8, PHP 4)

Beállítja a vonalak kapcsolódási módját

void **cpdf_setlinejoin** (int pdf document, long value) \linebreak

A **cpdf_setlinejoin()** függvény beállítja a vonalcsatlakozási paramétert 0 - 2 érték közé. 0 = hegyes, 1 = kerek, 2 = tompa.

cpdf_setlinewidth (PHP 3>= 3.0.8, PHP 4)

Beállítja a vonalvastagságot

void **cpdf_setlinewidth** (int pdf document, double width) \linebreak

A **cpdf_setlinewidth()** függvény beállítja a vonalvastagságot a *`width`* értékre.

cpdf_setmiterlimit (PHP 3>= 3.0.8, PHP 4)

Beállítja a hegyesszög határát

void **cpdf_setmiterlimit** (int pdf document, double value) \linebreak

A **cpdf_setmiterlimit()** függvény beállítja a hegyesszög határának értékét 1-nél nagyobb vagy egyenlő értékre.

cpdf_setrgbcolor_fill (PHP 3>= 3.0.8, PHP 4)

Rgb színértékre állítja be a kitöltő színt

void **cpdf_setrgbcolor_fill** (int pdf document, double red value, double green value, double blue value) \linebreak

A **cpdf_setrgbcolor_fill()** függvény beállítja az aktuális rgb értéket egy alakzat kitöltése céljából.

Lásd még: cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor().

cpdf_setrgbcolor_stroke (PHP 3>= 3.0.8, PHP 4)

Rgb színértékre állítja be a körvonalrajzoló színt

void **cpdf_setrgbcolor_stroke** (int pdf document, double red value, double green value, double blue value) \linebreak

A **cpdf_setrgbcolor_stroke()** függvény beállítja az aktuális rajzolószínt az adott rgb szín értékre.

Lásd még: cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_setrgbcolor (PHP 3>= 3.0.8, PHP 4)

Rgb színértékre állítja be a körvonalrajzoló és kitöltő színt

void **cpdf_setrgbcolor** (int pdf document, double red value, double green value, double blue value) \linebreak

A cpdf_setrgbcolor_stroke() függvény beállítja az aktuális rajzoló és kitöltő színt a megadott rgb szín értékre.

Lásd még: cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

cpdf_show_xy (PHP 3>= 3.0.8, PHP 4)

Szöveget helyez adott pozícióba

void **cpdf_show_xy** (int pdf document, string text, double x-koor, double y-koor, int mode) \linebreak

A **cpdf_show_xy()** függvény a *`text`* stringben megadott szöveget az (*`x-koor`*, *`y-koor`*). kordinátájú pontba helyezi. Az utolsó elhagyható paraméter meghatározza az egységet. Ha ez 0 vagy nem szerepel, az alapértelmezett érték az oldal számára meghatározott. A koordinátákat postcript pontok szerint veszi figyelembe, az aktuális unit beállítástól függetlenül.

Megjegyzés: A **cpdf_show_xy()** függvény megegyezik a cpdf_text() függvénnyel az elhagyható paraméterektől eltekintve.

Lásd még: cpdf_text().

cpdf_show (PHP 3>= 3.0.8, PHP 4)

Szöveget helyez az aktuális pozícióba

void **cpdf_show** (int pdf document, string text) \linebreak

A **cpdf_show()** függvény a *`text`* stringben levő szöveget az aktuális pozícióba helyezi.

Lásd még: cpdf_text(), cpdf_begin_text(), cpdf_end_text().

cpdf_stringwidth (PHP 3>= 3.0.8, PHP 4)

Visszaadja a szöveg szélességét az aktuális font alapján

double **cpdf_stringwidth** (int pdf document, string text) \linebreak

A **cpdf_stringwidth()** függvény visszaadja a *text* szöveg szélességét. Előtte a betűtípust mindenképpen be kell állítani.

Lásd még: cpdf_set_font().

cpdf_stroke (PHP 3>= 3.0.8, PHP 4)

Körvonalat rajzol egy alakzat mentén

void **cpdf_stroke** (int pdf document) \linebreak

A **cpdf_stroke()** függvény egy vonalat húz az aktuális alakzat mentén.

Lásd még: cpdf_closepath(), cpdf_closepath_stroke().

cpdf_text (PHP 3>= 3.0.8, PHP 4)

Szöveget helyez el paraméterekkel

void **cpdf_text** (int pdf document, string text, double x-koor, double y-koor, int mode, double orientation, int alignmode) \linebreak

A **cpdf_text()** függvény a *text* stringben megadott szöveget a (*x-koor*, *y-koor*). által meghatározott pontba helyezi. Az elhagyható paraméter meghatározza az egységet. Ha ez 0 vagy nem szerepel, az alapértelmezett érték az oldal számára meghatározott. A koordinátákat postcript pontok szerint veszi figyelembe, az aktuális unit beállítástól függetlenül. Az elhagyható *orientation* paraméter a szöveg elforgatása fokban. Az *alignmode* elhagyható paraméter meghatározza a szöveg igazítását. Lásd a ClibPDF dokumentációt a lehetséges értékekre vonatkozólag.

Lásd még: cpdf_show_xy().

cpdf_translate (PHP 3>= 3.0.8, PHP 4)

Beállítja a koordinátarendszer kezdőpontját

void **cpdf_translate** (int pdf document, double x-koor, double y-koor, int mode) \linebreak

A **cpdf_translate()** függvény az kordinátarendszer kezdőpontját a (*x-koor*, *y-koor*) pontba helyezi.

Az utolsó elhagyható paraméter megadja a hosszúsági egységet. Ha ez 0 vagy el van hagyva, az alapértelmezett érték az oldal számára meghatározott érték. Egyébként a koordinátákat postscript pontok szerint veszi figyelembe.

XI. Crack functions

Bevezetés

These functions allow you to use the CrackLib library to test the ’strength’ of a password. The ’strength’ of a password is tested by that checks length, use of upper and lower case and checked against the specified CrackLib dictionary. CrackLib will also give helpful diagnostic messages that will help ’strengthen’ the password.

Követelmények

More information regarding CrackLib along with the library can be found at http://www.users.dircon.co.uk/~crypto/.

Telepítés

In order to use these functions, you must compile PHP with Crack support by using the `--with-crack[=DIR]` option.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

Példák

This example shows how to open a CrackLib dictionary, test a given password, retrieve any diagnostic messages, and close the dictionary.

Példa 1. CrackLib example

```
<?php
```

```
// Open CrackLib Dictionary
$dictionary = crack_opendict('/usr/local/lib/pw_dict')
     or die('Unable to open CrackLib dictionary');

// Perform password check
$check = crack_check($dictionary, 'gx9A2s0x');

// Retrieve messages
$diag = crack_getlastmessage();
echo $diag; // 'strong password'

// Close dictionary
crack_closedict($dictionary);
?>
```

Megjegyzés: If crack_check() returns `TRUE`, crack_getlastmessage() will return 'strong password'.

crack_check (PHP 4 >= 4.0.5)

Performs an obscure check with the given password

bool **crack_check** ([resource dictionary, string password]) \linebreak

Returns TRUE if *`password`* is strong, or FALSE otherwise.

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

crack_check() performs an obscure check with the given *`password`* on the specified *`dictionary`* . If *`dictionary`* is not specified, the last opened dictionary is used.

crack_closedict (PHP 4 >= 4.0.5)

Closes an open CrackLib dictionary

bool **crack_closedict** ([resource dictionary]) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

crack_closedict() closes the specified *`dictionary`* identifier. If *`dictionary`* is not specified, the current dictionary is closed.

crack_getlastmessage (PHP 4 >= 4.0.5)

Returns the message from the last obscure check

string **crack_getlastmessage** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

crack_getlastmessage() returns the message from the last obscure check.

crack_opendict (PHP 4 >= 4.0.5)

Opens a new CrackLib dictionary

resource **crack_opendict** (string dictionary) \linebreak

Returns a dictionary resource identifier on success, or FALSE on failure.

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

crack_opendict() opens the specified CrackLib *`dictionary`* for use with crack_check().

Megjegyzés: Only one dictionary may be open at a time.

See also: crack_check(), and crack_closedict().

XII. CURL, Client URL Library Functions

Bevezetés

PHP supports libcurl, a library created by Daniel Stenberg, that allows you to connect and communicate to many different types of servers with many different types of protocols. libcurl currently supports the http, https, ftp, gopher, telnet, dict, file, and ldap protocols. libcurl also supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading (this can also be done with PHP's ftp extension), HTTP form based upload, proxies, cookies, and user+password authentication.

These functions have been added in PHP 4.0.2.

Követelmények

In order to use the CURL functions you need to install the CURL (http://curl.haxx.se/) package. PHP requires that you use CURL 7.0.2-beta or higher. PHP will not work with any version of CURL below version 7.0.2-beta.

Telepítés

To use PHP's CURL support you must also compile PHP `--with-curl[=DIR]` where DIR is the location of the directory containing the lib and include directories. In the "include" directory there should be a folder named "curl" which should contain the easy.h and curl.h files. There should be a file named "libcurl.a" located in the "lib" directory.

Előre definiált állandók

Az itt listázott állandókat ez a kiterjesztés definiálja, és csak akkor elérhetőek, ha az adott

kiterjesztés be van fordítva a PHP-be, vagy dinamikusan betöltött.

`CURLOPT_PORT` (integer)

`CURLOPT_FILE` (integer)

`CURLOPT_INFILE` (integer)

`CURLOPT_INFILESIZE` (integer)

`CURLOPT_URL` (integer)

`CURLOPT_PROXY` (integer)

`CURLOPT_VERBOSE` (integer)

`CURLOPT_HEADER` (integer)

`CURLOPT_HTTPHEADER` (integer)

`CURLOPT_NOPROGRESS` (integer)

`CURLOPT_NOBODY` (integer)

`CURLOPT_FAILONERROR` (integer)

`CURLOPT_UPLOAD` (integer)

`CURLOPT_POST` (integer)

`CURLOPT_FTPLISTONLY` (integer)

`CURLOPT_FTPAPPEND` (integer)

`CURLOPT_NETRC` (integer)

`CURLOPT_FOLLOWLOCATION` (integer)

`CURLOPT_FTPASCII` (integer)

idea behind the CURL functions is that you initialize a CURL session using the curl_init(), then you can set all your options for the transfer via the curl_exec() and then you finish off your session using the curl_close(). Here is an example that uses the CURL functions to fetch the example.com homepage into a file:

Példa 1. Using PHP's CURL module to fetch the example.com homepage

```
<?php

$ch = curl_init ("http://www.example.com/");
$fp = fopen ("example_homepage.txt", "w");

curl_setopt ($ch, CURLOPT_FILE, $fp);
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);
curl_close ($ch);
fclose ($fp);
?>
```

curl_close (PHP 4 >= 4.0.2)

Close a CURL session

void **curl_close** (resource ch) \linebreak

This function closes a CURL session and frees all resources. The CURL handle, *ch*, is also deleted.

curl_errno (PHP 4 >= 4.0.3)

Return an integer containing the last error number

int **curl_errno** (resource ch) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

curl_error (PHP 4 >= 4.0.3)

Return a string containing the last error for the current session

string **curl_error** (resource ch) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

curl_exec (PHP 4 >= 4.0.2)

Perform a CURL session

bool **curl_exec** (resource ch) \linebreak

This function should be called after you initialize a CURL session and all the options for the session are set. Its purpose is simply to execute the predefined CURL session (given by the *ch*).

Tipp: Mint bármilyen más esetben, amikor a kimenet közvetlenül a böngészőhöz kerül, használhatod az kimenet szabályozó függvényeket, hogy a függvény kimenetét "elkapd", és elmentsd például egy string-ben.

curl_getinfo (PHP 4 >= 4.0.4)

Get information regarding a specific transfer

string **curl_getinfo** (resource ch, int opt) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

curl_init (PHP 4 >= 4.0.2)

Initialize a CURL session

resource **curl_init** ([string url]) \linebreak

The **curl_init()** will initialize a new session and return a CURL handle for use with the curl_setopt(), curl_exec(), and curl_close() functions. If the optional *url* parameter is supplied then the CURLOPT_URL option will be set to the value of the parameter. You can manually set this using the curl_setopt() function.

Példa 1. Initializing a new CURL session and fetching a webpage

```
<?php
$ch = curl_init();

curl_setopt ($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);

curl_close ($ch);
?>
```

See also: curl_close(), curl_setopt()

curl_setopt (PHP 4 >= 4.0.2)

Set an option for a CURL transfer

bool **curl_setopt** (resource ch, string option, mixed value) \linebreak

The **curl_setopt()** function will set options for a CURL session identified by the *ch* parameter. The *option* parameter is the option you want to set, and the *value* is the value of the option given by the *option*.

The *value* should be a long for the following options (specified in the *option* parameter):

- *CURLOPT_INFILESIZE*: When you are uploading a file to a remote site, this option should be used to tell PHP what the expected size of the infile will be.
- *CURLOPT_VERBOSE*: Set this option to a non-zero value if you want CURL to report everything that is happening.
- *CURLOPT_HEADER*: Set this option to a non-zero value if you want the header to be included in the output.
- *CURLOPT_NOPROGRESS*: Set this option to a non-zero value if you don't want PHP to display a progress meter for CURL transfers.

  Megjegyzés: PHP automatically sets this option to a non-zero parameter, this should only be changed for debugging purposes.

- *CURLOPT_NOBODY*: Set this option to a non-zero value if you don't want the body included with the output.
- *CURLOPT_FAILONERROR*: Set this option to a non-zero value if you want PHP to fail silently if the HTTP code returned is greater than 300. The default behavior is to return the page normally, ignoring the code.
- *CURLOPT_UPLOAD*: Set this option to a non-zero value if you want PHP to prepare for an upload.
- *CURLOPT_POST*: Set this option to a non-zero value if you want PHP to do a regular HTTP POST. This POST is a normal `application/x-www-form-urlencoded` kind, most commonly used by HTML forms.
- *CURLOPT_FTPLISTONLY*: Set this option to a non-zero value and PHP will just list the names of an FTP directory.
- *CURLOPT_FTPAPPEND*: Set this option to a non-zero value and PHP will append to the remote file instead of overwriting it.
- *CURLOPT_NETRC*: Set this option to a non-zero value and PHP will scan your `~./netrc` file to find your username and password for the remote site that you're establishing a connection with.
- *CURLOPT_FOLLOWLOCATION*: Set this option to a non-zero value to follow any "Location: " header that the server sends as a part of the HTTP header (note this is recursive, PHP will follow as many "Location: " headers that it is sent.)
- *CURLOPT_PUT*: Set this option to a non-zero value to HTTP PUT a file. The file to PUT must be set with the CURLOPT_INFILE and CURLOPT_INFILESIZE.

- *`CURLOPT_MUTE`*: Set this option to a non-zero value and PHP will be completely silent with regards to the CURL functions.
- *`CURLOPT_TIMEOUT`*: Pass a long as a parameter that contains the maximum time, in seconds, that you'll allow the CURL functions to take.
- *`CURLOPT_LOW_SPEED_LIMIT`*: Pass a long as a parameter that contains the transfer speed in bytes per second that the transfer should be below during CURLOPT_LOW_SPEED_TIME seconds for PHP to consider it too slow and abort.
- *`CURLOPT_LOW_SPEED_TIME`*: Pass a long as a parameter that contains the time in seconds that the transfer should be below the CURLOPT_LOW_SPEED_LIMIT for PHP to consider it too slow and abort.
- *`CURLOPT_RESUME_FROM`*: Pass a long as a parameter that contains the offset, in bytes, that you want the transfer to start from.
- *`CURLOPT_SSLVERSION`*: Pass a long as a parameter that contains the SSL version (2 or 3) to use. By default PHP will try and determine this by itself, although, in some cases you must set this manually.
- *`CURLOPT_SSL_VERIFYHOST`*: Pass a long if CURL should verify the Common name of the peer certificate in the SSL handshake. A value of 1 denotes that we should check for the existence of the common name, a value of 2 denotes that we should make sure it matches the provided hostname.
- *`CURLOPT_TIMECONDITION`*: Pass a long as a parameter that defines how the CURLOPT_TIMEVALUE is treated. You can set this parameter to TIMECOND_IFMODSINCE or TIMECOND_ISUNMODSINCE. This is a HTTP-only feature.
- *`CURLOPT_TIMEVALUE`*: Pass a long as a parameter that is the time in seconds since January 1st, 1970. The time will be used as specified by the CURLOPT_TIMEVALUE option, or by default the TIMECOND_IFMODSINCE will be used.
- *`CURLOPT_RETURNTRANSFER`*: Pass a non-zero value if you want CURL to directly return the transfer instead of printing it out directly.

The *`value`* parameter should be a string for the following values of the *`option`* parameter:

- *`CURLOPT_URL`*: This is the URL that you want PHP to fetch. You can also set this option when initializing a session with the curl_init() function.
- *`CURLOPT_USERPWD`*: Pass a string formatted in the [username]:[password] manner, for PHP to use for the connection.
- *`CURLOPT_PROXYUSERPWD`*: Pass a string formatted in the [username]:[password] format for connection to the HTTP proxy.
- *`CURLOPT_RANGE`*: Pass the specified range you want. It should be in the "X-Y" format, where X or Y may be left out. The HTTP transfers also support several intervals, separated with commas as in X-Y,N-M.
- *`CURLOPT_POSTFIELDS`*: Pass a string containing the full data to post in an HTTP "POST" operation.
- *`CURLOPT_REFERER`*: Pass a string containing the "referer" header to be used in an HTTP request.
- *`CURLOPT_USERAGENT`*: Pass a string containing the "user-agent" header to be used in an HTTP request.

- *CURLOPT_FTPPORT*: Pass a string containing the value which will be used to get the IP address to use for the ftp "POST" instruction. The POST instruction tells the remote server to connect to our specified IP address. The string may be a plain IP address, a hostname, a network interface name (under UNIX), or just a plain '-' to use the systems default IP address.
- *CURLOPT_COOKIE*: Pass a string containing the content of the cookie to be set in the HTTP header.
- *CURLOPT_SSLCERT*: Pass a string containing the filename of PEM formatted certificate.
- *CURLOPT_SSLCERTPASSWD*: Pass a string containing the password required to use the CURLOPT_SSLCERT certificate.
- *CURLOPT_COOKIEFILE*: Pass a string containing the name of the file containing the cookie data. The cookie file can be in Netscape format, or just plain HTTP-style headers dumped into a file.
- *CURLOPT_CUSTOMREQUEST*: Pass a string to be used instead of `GET` or `HEAD` when doing an HTTP request. This is useful for doing `DELETE` or other, more obscure, HTTP requests. Valid values are things like `GET`, `POST`, and so on; i.e. do not enter a whole HTTP request line here. For instance, entering 'GET /index.html HTTP/1.0\r\n\r\n' would be incorrect.

  Megjegyzés: Don't do this without making sure your server supports the command first.

- *CURLOPT_PROXY*: Give the name of the HTTP proxy to tunnel requests through.
- *CURLOPT_INTERFACE*: Pass the name of the outgoing network interface to use. This can be an interface name, an IP address or a host name.
- *CURLOPT_KRB4LEVEL*: Pass the KRB4 (Kerberos 4) security level. Anyone of the following strings (in order from least powerful, to most powerful): 'clear', 'safe', 'confidential', 'private'. If the string does not match one of these, then 'private' is used. If you set this to NULL, this disables KRB4 security. KRB4 security only works with FTP transactions currently.
- *CURLOPT_HTTPHEADER*: Pass an array of HTTP header fields to set.
- *CURLOPT_QUOTE*: Pass an array of FTP commands to perform on the server prior to the FTP request.
- *CURLOPT_POSTQUOTE*: Pass an array of FTP commands to execute on the server, after the FTP request has been performed.

The following options expect a file descriptor that is obtained by using the fopen() function:

- *CURLOPT_FILE*: The file where the output of your transfer should be placed, the default is STDOUT.
- *CURLOPT_INFILE*: The file where the input of your transfer comes from.
- *CURLOPT_WRITEHEADER*: The file to write the header part of the output into.
- *CURLOPT_STDERR*: The file to write errors to instead of stderr.

curl_version (PHP 4 >= 4.0.2)

Return the current CURL version

string **curl_version** (void) \linebreak

The **curl_version()** function returns a string containing the current CURL version.

XIII. Cybercash payment függvények

Ezek a függvények csak akkor használhatóak, ha az értelmező a `--with-cybercash=[DIR]`-rel lett lefordítva. Ezek a függvények a PHP 4-ben használhatóak.

cybercash_base64_decode (PHP 4)

base64 kódolt adatot visszafejt Cybercash számára

string **cybercash_base64_decode** (string inbuff) \linebreak

cybercash_base64_encode (PHP 4)

base64 kódolja az adatot Cybercash számára

string **cybercash_base64_encode** (string inbuff) \linebreak

cybercash_decr (PHP 4)

cybercash visszafejtés

array **cybercash_decr** (string wmk, string sk, string inbuff) \linebreak

A függvény egy asszociatív tömböt ad vissza az "errcode" elemmel, és ha az FALSE, akkor az "outbuff" (string), "outLth" (long) és "macbuff" (string) elemekkel.

cybercash_encr (PHP 4)

cybercash titkosítás

array **cybercash_encr** (string wmk, string sk, string inbuff) \linebreak

A függvény egy asszociatív tömböt ad vissza az "errcode" elemmel, és ha az FALSE, akkor az "outbuff" (string), "outLth" (long) és "macbuff" (string) elemekkel.

XIV. Crédit Mutuel CyberMUT functions

Bevezetés

This extension allows you to process credit cards transactions using Crédit Mutuel CyberMUT system (http://www.creditmutuel.fr/centre_commercial/vendez_sur_internet.html).

CyberMUT is a popular Web Payment Service in France, provided by the Crédit Mutuel bank. If you are foreign in France, these functions will not be useful for you.

The use of these functions is almost identical to the original SDK functions, except for the parameters of return for cybermut_creerformulairecm() and cybermut_creerreponsecm(), which are returned directly by functions PHP, whereas they had passed in reference in the original functions.

These functions have been added in PHP 4.0.6.

> **Megjegyzés:** These functions only provide a link to CyberMUT SDK. Be sure to read the CyberMUT Developers Guide for full details of the required parameters.

Telepítés

These functions are only available if PHP has been compiled with the `--with-cybermut[=DIR]` option, where DIR is the location of `libcm-mac.a` and `cm-mac.h`. You will require the appropriate SDK for your platform, which may be sent to you after your CyberMUT's subscription (contact them via Web, or go to the nearest Crédit Mutuel).

cybermut_creerformulairecm (PHP 4 >= 4.0.5)

Generate HTML form of request for payment

string **cybermut_creerformulairecm** (string url_CM, string version, string TPE, string montant, string ref_commande, string texte_libre, string url_retour, string url_retour_ok, string url_retour_err, string langue, string code_societe, string texte_bouton) \linebreak

cybermut_creerformulairecm() is used to generate the HTML form of request for payment.

Példa 1. First step of payment (equiv cgi1.c)

```
<?php
// Directory where the keys are located
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

  $retour =  cybermut_creerformulairecm(
  "https://www.creditmutuel.fr/test/telepaiement/paiement.cgi",
  $VERSION,
  "1234567890",
  "300FRF",
  $REFERENCE,
  $TEXTE_LIBRE,
  $URL_RETOUR,
  $URL_RETOUR_OK,
  $URL_RETOUR_ERR,
  "francais",
  "company",
  "Paiement par carte bancaire");

  echo $retour;
?>
```

See also cybermut_testmac() and cybermut_creerreponsecm().

cybermut_creerreponsecm (PHP 4 >= 4.0.5)

Generate the acknowledgement of delivery of the confirmation of payment

string **cybermut_creerreponsecm** (string phrase) \linebreak

cybermut_creerreponsecm() returns a string containing delivery acknowledgement message.

The parameter is "OK" if the message of confirmation of the payment was correctly identified by cybermut_testmac(). Any other chain is regarded as an error message.

See also cybermut_creerformulairecm() and cybermut_testmac().

cybermut_testmac (PHP 4 >= 4.0.5)

Make sure that there no was data diddling contained in the received message of confirmation

bool **cybermut_testmac** (string code_MAC, string version, string TPE, string cdate, string montant, string ref_commande, string texte_libre, string code-retour) \linebreak

cybermut_testmac() is used to make sure that there was not data diddling contained in the received message of confirmation. Pay attention to parameters *code-retour* and *texte-libre*, which cannot be evaluated as is, because of the dash. You must retrieve them by using:

```
<?php
  $code_retour = $_GET["code-retour"];
  $texte_libre = $_GET["texte-libre"];
?>
```

Példa 1. Last step of payment (equiv cgi2.c)

```
<?php
// Make sure that Enable Track Vars is ON.
// Directory where are located the keys
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

$texte_libre = $_GET["texte-libre"];
$code_retour = $_GET["code-retour"];

$mac_ok = cybermut_testmac($MAC,$VERSION,$TPE,$date,$montant,$reference,$texte_libre,$coc

if ($mac_ok) {

  //
  // insert data processing here
  //
  //

  $result=cybermut_creerreponsecm("OK");
} else {
  $result=cybermut_creerreponsecm("Document Falsifie");
}

?>
```

See also cybermut_creerformulairecm() and cybermut_creerreponsecm().

XV. Cyrus IMAP administration functions

Bevezetés

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

Előre definiált állandók

Az itt listázott állandókat ez a kiterjesztés definiálja, és csak akkor elérhetőek, ha az adott kiterjesztés be van fordítva a PHP-be, vagy dinamikusan betöltött.

`CYRUS_CONN_NONSYNCLITERAL` (integer)

`CYRUS_CONN_INITIALRESPONSE` (integer)

`CYRUS_CALLBACK_NUMBERED` (integer)

`CYRUS_CALLBACK_NOLITERAL` (integer)

cyrus_authenticate (PHP 4 >= 4.1.0)

Authenticate against a Cyrus IMAP server

bool **cyrus_authenticate** (resource connection [, string mechlist [, string service [, string user [, int minssf [, int maxssf]]]]]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

cyrus_bind (PHP 4 >= 4.1.0)

Bind callbacks to a Cyrus IMAP connection

bool **cyrus_bind** (resource connection, array callbacks) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

cyrus_close (PHP 4 >= 4.1.0)

Close connection to a Cyrus IMAP server

bool **cyrus_close** (resource connection) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

cyrus_connect (PHP 4 >= 4.1.0)

Connect to a Cyrus IMAP server

resource **cyrus_connect** ([string host [, string port [, int flags]]]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

cyrus_query (PHP 4 >= 4.1.0)

Send a query to a Cyrus IMAP server

bool **cyrus_query** (resource connection, string query) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

cyrus_unbind (PHP 4 >= 4.1.0)

Unbind ...

bool **cyrus_unbind** (resource connection, string trigger_name) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

XVI. Character type functions

Bevezetés

The functions provided by this extension check whether a character or string falls into a certain character class according to the current locale (see also setlocale()).

When called with an integer argument these functions behave exactly like their C counterparts from "ctype.h".

When called with a string argument they will check every character in the string and will only return `TRUE` if every character in the string matches the requested criteria.

Passing anything else but a string or integer will return `FALSE` immediately.

Követelmények

None besides functions from the standard C library which are always available.

Telepítés

Beginning with PHP 4.2.0 these functions are enabled by default. For older versions you have to configure and compile PHP with `--enable-ctype`.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

ctype_alnum (PHP 4 >= 4.0.4)

Check for alphanumeric character(s)

bool **ctype_alnum** (string text) \linebreak

Returns TRUE if every character in *text* is either a letter or a digit, FALSE otherwise. In the standard C locale letters are just [A-Za-z]. The function is equivalent to (ctype_alpha($text) || ctype_digit($text)).

See also ctype_alpha(), ctype_digit(), and setlocale().

ctype_alpha (PHP 4 >= 4.0.4)

Check for alphabetic character(s)

bool **ctype_alpha** (string text) \linebreak

Returns TRUE if every character in *text* is a letter from the current locale, FALSE otherwise. In the standard C locale letters are just [A-Za-z] and **ctype_alpha()** is equivalent to (ctype_upper($text) || ctype_lower($text)), but other languages have letters that are considered neither upper nor lower case.

See also ctype_upper(), ctype_lower(), and setlocale().

ctype_cntrl (PHP 4 >= 4.0.4)

Check for control character(s)

bool **ctype_cntrl** (string text) \linebreak

Returns TRUE if every character in *text* has a special control function, FALSE otherwise. Control characters are e.g. line feed, tab, esc.

ctype_digit (PHP 4 >= 4.0.4)

Check for numeric character(s)

bool **ctype_digit** (string text) \linebreak

Returns TRUE if every character in *text* is a decimal digit, FALSE otherwise.

See also ctype_alnum() and ctype_xdigit().

ctype_graph (PHP 4 >= 4.0.4)

Check for any printable character(s) except space

bool **ctype_graph** (string text) \linebreak

Returns TRUE if every character in *text* is printable and actually creates visible output (no white space), FALSE otherwise.

See also ctype_alnum(), ctype_print(), and ctype_punct().

ctype_lower (PHP 4 >= 4.0.4)

Check for lowercase character(s)

bool **ctype_lower** (string text) \linebreak

Returns TRUE if every character in *text* is a lowercase letter in the current locale.

See also ctype_alpha() and ctype_upper().

ctype_print (PHP 4 >= 4.0.4)

Check for printable character(s)

bool **ctype_print** (string text) \linebreak

Returns TRUE if every character in *text* will actually create output (including blanks). Returns FALSE if *text* contains control characters or characters that do not have any output or control function at all.

See also ctype_cntrl(), ctype_graph(), and ctype_punct().

ctype_punct (PHP 4 >= 4.0.4)

Check for any printable character which is not whitespace or an alphanumeric character

bool **ctype_punct** (string text) \linebreak

Returns TRUE if every character in *text* is printable, but neither letter, digit or blank, FALSE otherwise.

See also ctype_cntrl(), ctype_graph(), and **ctype_punct()**.

ctype_space (PHP 4 >= 4.0.4)

Check for whitespace character(s)

bool **ctype_space** (string text) \linebreak

Returns TRUE if every character in *text* creates some sort of white space, FALSE otherwise. Besides the blank character this also includes tab, vertical tab, line feed, carriage return and form feed characters.

ctype_upper (PHP 4 >= 4.0.4)

Check for uppercase character(s)

bool **ctype_upper** (string text) \linebreak

Returns TRUE if every character in *text* is a uppercase letter in the current locale.

See also ctype_alpha() and ctype_lower().

ctype_xdigit (PHP 4 >= 4.0.4)

Check for character(s) representing a hexadecimal digit

bool **ctype_xdigit** (string text) \linebreak

Returns TRUE if every character in *text* is a hexadecimal 'digit', that is a decimal digit or a character from [A-Fa-f] , FALSE otherwise.

See also ctype_digit().

XVII. Database (dbm-style) abstraction layer functions

Bevezetés

These functions build the foundation for accessing Berkeley DB style databases.

This is a general abstraction layer for several file-based databases. As such, functionality is limited to a common subset of features supported by modern databases such as Sleepycat Software's DB2 (http://www.sleepycat.com/). (This is not to be confused with IBM's DB2 software, which is supported through the ODBC functions.)

Követelmények

The behaviour of various aspects depends on the implementation of the underlying database. Functions such as dba_optimize() and dba_sync() will do what they promise for one database and will do nothing for others. You have to download and install supported dba-Handlers.

Táblázat 1. List of DBA handlers

Handler	Notes
`dbm`	Dbm is the oldest (original) type of Berkeley DB style databases. You should avoid it, if possible. We do not support the compatibility functions built into DB2 and gdbm, because they are only compatible on the source code level, but cannot handle the original dbm format.
`ndbm`	Ndbm is a newer type and more flexible than dbm. It still has most of the arbitrary limits of dbm (therefore it is deprecated).
`gdbm`	Gdbm is the GNU database manager (ftp://ftp.gnu.org/pub/gnu/gdbm/).
`db2`	DB2 is Sleepycat Software's DB2 (http://www.sleepycat.com/). It is described as "a programmatic toolkit that provides high-performance built-in database support for both standalone and client/server applications.
`db3`	DB3 is Sleepycat Software's DB3 (http://www.sleepycat.com/).
`cdb`	Cdb is "a fast, reliable, lightweight package for creating and reading constant databases." It is from the author of qmail and can be found here (http://cr.yp.to/cdb.html). Since it is constant, we support only reading operations.

When invoking the dba_open() or dba_popen() functions, one of the handler names must be supplied as an argument. The actually available list of handlers is displayed by invoking phpinfo().

Telepítés

By using the `--enable-dba=shared` configuration option you can build a dynamic loadable modul to enable PHP for basic support of dbm-style databases. You also have to add support for at least one of the following handlers by specifying the `--with-XXXX` configure switch to your PHP configure line.

Táblázat 2. Supported DBA handlers

Handler	Configure Switch
`dbm`	To enable support for dbm add `--with-dbm[=DIR]`.
`ndbm`	To enable support for ndbm add `--with-ndbm[=DIR]`.
`gdbm`	To enable support for gdbm add `--with-gdbm[=DIR]`.
`db2`	To enable support for db2 add `--with-db2[=DIR]`.
`db3`	To enable support for db3 add `--with-db3[=DIR]`.
`cdb`	To enable support for cdb add `--with-cdb[=DIR]`.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

The functions dba_open() and dba_popen() return a handle to the specified database file to access

which is used by all other dba-function calls.

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

Példák

Példa 1. DBA example

```
<?php

$id = dba_open ("/tmp/test.db", "n", "db2");

if (!$id) {
    echo "dba_open failed\n";
    exit;
}

dba_replace ("key", "This is an example!", $id);

if (dba_exists ("key", $id)) {
    echo dba_fetch ("key", $id);
    dba_delete ("key", $id);
}

dba_close ($id);
?>
```

DBA is binary safe and does not have any arbitrary limits. However, it inherits all limits set by the underlying database implementation.

All file-based databases must provide a way of setting the file mode of a new created database, if that is possible at all. The file mode is commonly passed as the fourth argument to dba_open() or dba_popen().

You can access all entries of a database in a linear way by using the dba_firstkey() and dba_nextkey() functions. You may not change the database while traversing it.

Példa 2. Traversing a database

```
<?php

// ...open database...

$key = dba_firstkey ($id);
```

```
while ($key != false) {
    if (...) {          // remember the key to perform some action later
        $handle_later[] = $key;
    }
    $key = dba_nextkey ($id);
}

for ($i = 0; $i < count($handle_later); $i++)
    dba_delete ($handle_later[$i], $id);

?>
```

dba_close (PHP 3>= 3.0.8, PHP 4)

Close database

void **dba_close** (resource handle) \linebreak

dba_close() closes the established database and frees all resources specified by *handle*.

handle is a database handle returned by dba_open().

dba_close() does not return any value.

See also: dba_open() and dba_popen()

dba_delete (PHP 3>= 3.0.8, PHP 4)

Delete entry specified by key

bool **dba_delete** (string key, resource handle) \linebreak

dba_delete() deletes the entry specified by *key* from the database specified with *handle*.

key is the key of the entry which is deleted.

handle is a database handle returned by dba_open().

dba_delete() returns TRUE or FALSE, if the entry is deleted or not deleted, respectively.

See also: dba_exists(), dba_fetch(), dba_insert(), and dba_replace().

dba_exists (PHP 3>= 3.0.8, PHP 4)

Check whether key exists

bool **dba_exists** (string key, resource handle) \linebreak

dba_exists() checks whether the specified *key* exists in the database specified by *handle*.

Key is the key the check is performed for.

Handle is a database handle returned by dba_open().

dba_exists() returns TRUE or FALSE, if the key is found or not found, respectively.

See also: dba_fetch(), dba_delete(), dba_insert(), and dba_replace().

dba_fetch (PHP 3>= 3.0.8, PHP 4)

Fetch data specified by key

string **dba_fetch** (string key, resource handle) \linebreak

dba_fetch() fetches the data specified by *key* from the database specified with *handle*.

Key is the key the data is specified by.

Handle is a database handle returned by dba_open().

dba_fetch() returns the associated string or FALSE, if the key/data pair is found or not found, respectively.

See also: dba_exists(), dba_delete(), dba_insert(), and dba_replace().

dba_firstkey (PHP 3>= 3.0.8, PHP 4)

Fetch first key

string **dba_firstkey** (resource handle) \linebreak

dba_firstkey() returns the first key of the database specified by *handle* and resets the internal key pointer. This permits a linear search through the whole database.

Handle is a database handle returned by dba_open().

dba_firstkey() returns the key or FALSE depending on whether it succeeds or fails, respectively.

See also: dba_nextkey() and example 2 in the DBA examples

dba_insert (PHP 3>= 3.0.8, PHP 4)

Insert entry

bool **dba_insert** (string key, string value, resource handle) \linebreak

dba_insert() inserts the entry described with *key* and *value* into the database specified by *handle*. It fails, if an entry with the same *key* already exists.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by dba_open().

dba_insert() returns TRUE or FALSE, depending on whether it succeeds of fails, respectively.

See also: dba_exists() dba_delete() dba_fetch() dba_replace()

dba_nextkey (PHP 3>= 3.0.8, PHP 4)

Fetch next key

string **dba_nextkey** (resource handle) \linebreak

dba_nextkey() returns the next key of the database specified by *handle* and advances the internal key pointer.

handle is a database handle returned by dba_open().

dba_nextkey() returns the key or FALSE depending on whether it succeeds or fails, respectively.

See also: dba_firstkey() and example 2 in the DBA examples

dba_open (PHP 3>= 3.0.8, PHP 4)

Open database

resource **dba_open** (string path, string mode, string handler [, ...]) \linebreak

dba_open() establishes a database instance for *path* with *mode* using *handler*.

path is commonly a regular path in your filesystem.

mode is "r" for read access, "w" for read/write access to an already existing database, "c" for read/write access and database creation if it doesn't currently exist, and "n" for create, truncate and read/write access.

handler is the name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to **dba_open()** and can act on behalf of them.

dba_open() returns a positive handle or FALSE, in the case the open is successful or fails, respectively.

See also: dba_popen() dba_close()

dba_optimize (PHP 3>= 3.0.8, PHP 4)

Optimize database

bool **dba_optimize** (resource handle) \linebreak

dba_optimize() optimizes the underlying database specified by *handle*.

handle is a database handle returned by dba_open().

dba_optimize() returns TRUE or FALSE, if the optimization succeeds or fails, respectively.

See also: dba_sync()

dba_popen (PHP 3>= 3.0.8, PHP 4)

Open database persistently

resource **dba_popen** (string path, string mode, string handler [, ...]) \linebreak

dba_popen() establishes a persistent database instance for *path* with *mode* using *handler*.

path is commonly a regular path in your filesystem.

mode is "r" for read access, "w" for read/write access to an already existing database, "c" for read/write access and database creation if it doesn't currently exist, and "n" for create, truncate and read/write access.

handler is the name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to **dba_popen()** and can act on behalf of them.

dba_popen() returns a positive handle or FALSE, in the case the open is successful or fails, respectively.

See also: dba_open() dba_close()

dba_replace (PHP 3>= 3.0.8, PHP 4)

Replace or insert entry

bool **dba_replace** (string key, string value, resource handle) \linebreak

dba_replace() replaces or inserts the entry described with *key* and *value* into the database specified by *handle*.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by dba_open().

dba_replace() returns TRUE or FALSE, depending on whether it succeeds of fails, respectively.

See also: dba_exists(), dba_delete(), dba_fetch(), and dba_insert().

dba_sync (PHP 3>= 3.0.8, PHP 4)

Synchronize database

bool **dba_sync** (resource handle) \linebreak

dba_sync() synchronizes the database specified by *handle*. This will probably trigger a physical write to disk, if supported.

handle is a database handle returned by dba_open().

dba_sync() returns TRUE or FALSE, if the synchronization succeeds or fails, respectively.

See also: dba_optimize()

XVIII. Dátummal és időponttal kapcsolatos függvények

checkdate (PHP 3, PHP 4)

Ellenőriz egy időpontot

bool **checkdate** (int month, int day, int year) \linebreak

Visszatérési értéke TRUE, ha az argumentumban megadott dátum érvényes, egyébként FALSE. Egy dátum érvénes, ha:

- az év 1 és 32767 között van
- a hónap 1 és 12 között van
- A *day* paraméter a *month* és a *year* paramétereknek megfelelő értéket vesz fel. (Szökőéveket természetesen helyesen kezeli)

Lásd még a mktime() és a strtotime() függvényt!

date (PHP 3, PHP 4)

Egy időpontstringgel tér vissza

string **date** (string format [, int timestamp]) \linebreak

Egy stringgel tér vissza, amely a formátumstring szerinti alakban tartalmazza a *timestamp* dátumot, ennek hiányában az aktuális helyi időt.

> **Megjegyzés:** Az időpont érvényes időpontja tipikusan 1901 Dec. 13. 20:25:54 és 2038. Jan. 19. 03:14:07 közötti időpontot jelent (Greenwich-i időpont szerint) (Ezek a dátumok a legkisebb és a legnagyobb 32-bites egész szám által reprezentált dátumok)
>
> Ha egy karakterlánc által leírt időpontból szeretnél időpontot gyártani, akkor használhatod a strtotime() függvényt. Ráadásul, néhány adatbázisnak van rá függvénye, hogy a belső időpont-ábrázolásukat "unix timestamp"-pé alakítják (a MySQL UNIX_TIMESTAMP függvénye ilyen).

A formátum stringben a következő karakterek lehetnek:

- a - "am" vagy "pm" (délelőtt vagy délután)
- A - "AM" vagy "PM" (ugyanez nagybetűvel)
- B - Swatch Internet time
- d - nap, 2 számjegyű (előtte 0, ha kell) pl.: "01" .. "31"
- D - a hét napja, betűvel, 3 betűs (angol[?]); pl.: "Fri"
- F - hónap, betűvel, hosszú (angol); pl.:"January"
- g - óra, 12-órás formátumban, "felesleges" nullák nélkül; pl.: "1" .. "12"
- G - óra, 24-órás formátumban, "felesleges" nullák nélkül; pl.: "0" .. "23"
- h - óra, 12-órás formátumban (2 számjegy); pl.: "01" .. "12"

- H - óra, 24-órás formátumban (2 számjegy); pl.: "00" .. "23"
- i - perc (2 számjegy); pl.: "00" .. "59"
- I (Nagy i) - "1" a nyári időszámítás alatt, "0" egyébként.
- j - hónap, számmal, "felesleges" nullák nélkül; pl.: "1" .. "31"
- l (kis 'L') - hét napja, betűvel, hosszú; pl.: "Friday"
- L - logikai változó, jelzi, hogy szökőév van-e vagy se; pl.: "0" vagy "1"
- m - hónap, számmal (2 számjegy); pl.: "01" .. "12"
- M - hónap, betűvel, 3 betűs; pl.: "Jan"
- n - hónap, számmal, "felesleges" nullák nélkül; pl.: "1" .. "12"
- O - Különbség a Greenwich-i időhöz képest; pl.: "+0200"
- r - RFC 822 formátumú dátum; pl: "Thu, 21 Dec 2000 16:01:07 +0200" (PHP 4.0.4 óta)
- s - másodperc; pl.: "00" .. "59"
- S - angol "sorszámnév-string" a hónap napjának megfelelően, betűvel, két karakteren; például "th", "nd"
- t - napok száma az adott hónapban; pl.: "28" .. "31"
- T - A gép időzóna-beállítása; pl. "MDT"
- U - eltelt másodpercek száma a UNIX Epoch óta (1970. 01. 01.)
- w - hét napja, számmal, pl.: "0" (Vasárnap) .. "6" (Szombat)
- W - ISO-8601 hét szám; a hetek hétfőn kezdődnek (PHP 4.1.0-tól) [mi szombat???]
- Y - év, 4 számjegy; pl.: "1999"
- y - év, 2 számjegy; pl.: "99"
- z - január 1 óta eltelt napok száma; pl.: "0" .. "365"
- Z - időzóna eltolódás másodpercben(pl.: "-43200" .. "43200"). Az időzónák az UTC-től nyugatra mindig negatívak, keletre mindig pozitívak.

A formátumstringben levő, egyéb karakterek egy az egyben megjelennek. A "Z" betű mindig "0"-val tér vissza, ha a gmdate()-et használjuk.

Példa 1. date() példa

```
echo (date ("l dS of F Y h:i:s A"));
echo "July 1, 2000 is on a " . date ("l", mktime(0,0,0,7,1,2000));
```

A felismert karaktereket a formátumstringben backslash-sel tudod megvédeni. Ha a karakter backslash karakter után is speciális jelentéssel bír, akkor a backslash-t is meg kell védeni a kiértékeléstől.

Példa 2. Karakterek megvédése a date() függvényben

```
echo date("l \\t\h\e jS"); // Valami olyasmit ír ki, hogy ’Saturday the 8th’
```

A **date()**-et és a mktime()-ot ügyesen együtt használva információt szerezhetsz múltbeli és jövőbeli időpontokról.

Példa 3. date() és mktime() példa

```
$tomorrow  = mktime (0,0,0,date("m")  ,date("d")+1,date("Y"));
$lastmonth = mktime (0,0,0,date("m")-1,date("d"),  date("Y"));
$nextyear  = mktime (0,0,0,date("m"),  date("d"),  date("Y")+1);
# ezek mind működnek hónap végén, ill. januárban is. Ha nem hiszed, próbáld ki!
```

> **Megjegyzés:** Ez sokkal megbízhatóbb, mint az időbélyegekkel számolni (gondolj a téli-nyári időszámításkori átállásokra)

Néhány példa a **date()** függvény formázására. Figyeld meg, hogy az összes karaktert meg kell védened a kiértékeléstől, nem csupán azokat, amlyeknek már most speciális jelentésük van, ugyanis egy jövőbeli PHP változatban ezek is jelentést kaphatnak. Amikor egy karaktert megvédesz az értelmezéstől, használj aposztrófokat a karakterlánc megadására, hogy pl. a \n újsorokká történő átalakítását megakadályozd.

Példa 4. A date() függvyény formátumstringje

```
/* Tegyük fel, hogy most March 10th, 2001, 5:16:18 pm  van*/
$today = date("F j, Y, g:i a");                 // March 10, 2001, 5:16 pm
$today = date("m.d.y");                         // 03.10.01
$today = date("j, n, Y");                       // 10, 3, 2001
$today = date("Ymd");                           // 20010310
$today = date(’h-i-s, j-m-y, it is w Day z ’);  // 05-16-17, 10-03-01, 1631 1618 6 Fripm0
$today = date(’\M\a \a \h\ó\n\a\p j. \n\a\p\j\a \v\a\n.’);   // Ma a hónap 10. napja van
$today = date("D M j G:i:s T Y");               // Sat Mar 10 15:16:08 MST 2001
$today = date(’H:m:s \m = \h\ó\n\a\p’);         // 17:03:17 m = hónap
$today = date("H:i:s");                         // 17:16:17
```

Ha a dátumokat más nyelven szeretnéd megkapni, használd a setlocale() és a strftime() függvényeket.

Lásd még a getlastmod(), gmdate(), mktime(), strftime() és a time() függvényeket.

getdate (PHP 3, PHP 4)

Dátum/idő információ

array **getdate** ([int timestamp]) \linebreak

A *timestamp*-ben megadott időpontról (vagy a jelen pillanatról, ha nincs időpont megadva) ad információt asszociatív tömb formájában. A következő elemeket tartalmazza:

- "seconds" - másodperc
- "minutes" - perc
- "hours" - óra
- "mday" - nap
- "wday" - hét napja, számmal (Vasárnap a 0, szombat a 6)
- "mon" - hónap, számmal
- "year" - év, számmal [miért, mással is lehet?]
- "yday" - január 1 óta eltelt napok száma, számmal [detto]; pl.: "299"
- "weekday" - hét napja, számmal, full; pl.: "Friday"
- "month" - hónap, betűvel, teljes; pl.: "January"

Példa 1. getdate() példa

```
$today = getdate();
$month = $today['month'];
$mday = $today['mday'];
$year = $today['year'];
echo "$month $mday, $year";
```

gettimeofday (PHP 3>= 3.0.7, PHP 4)

Aktuális idő lekérése[???]

array **gettimeofday** (void) \linebreak

Ez egy felület a gettimeofday(2)-höz. Egy asszociatív tömbbel tér vissza, tartalmazza a rendszerhívásból származó adatokat [??]

- "sec" - másodperc
- "usec" - mikroszekundum ["mikromásodperc"]
- "minuteswest" - hány percet kell hozzáadni a helyi időhöz, hogy a Greenwichit kapjuk
- "dsttime" - nyári időszámítási korrekció típusa [???]

gmdate (PHP 3, PHP 4)

Egy GMT/CUT időponttal tér vissza

string **gmdate** (string format [, int timestamp]) \linebreak

Azonos a date() függvénnyel, kivéve, hogy az időpont Greenwichi idő szerint van(GMT). Például, ha Finnországban fut (GMT +0200), az első sor lent kiírja, hogy "Jan 01 1998 00:00:00", míg a második pedig azt, hogy "Dec 31 1997 22:00:00".

Példa 1. gmdate() példa

```
echo date ("M d Y H:i:s", mktime (0,0,0,1,1,1998));
echo gmdate ("M d Y H:i:s", mktime (0,0,0,1,1,1998));
```

Lásd még a date(), mktime(), gmmktime() és a strftime() függvényeket.

gmmktime (PHP 3, PHP 4)

GMT formátumú UNIX időbélyeggel tér vissza

int **gmmktime** (int hour, int minute, int second, int month, int day, int year [, int is_dst]) \linebreak

Azonos a mktime() függvénnyel, kivéve, hogy az átadott paraméterek GMT időpontként értelmezettek.

gmstrftime (PHP 3>= 3.0.12, PHP 4)

Az aktuális helyi időből számított GMT/CUT időpontot ad vissza.

string **gmstrftime** (string format [, int timestamp]) \linebreak

A strftime()-hoz hasonlóan viselkedik, kivéve, hogy a függvény által visszaadott időpont GMT formátumban van. Például, ha "Keleti Standard Idő"-ben(GMT -0500) vagyunk, a következő sor kiírja, hogy "Dec 31 1998 20:00:00", míg a második azt írja ki, hogy "Jan 01 1999 01:00:00".

Példa 1. gmstrftime() példa

```
setlocale ('LC_TIME', 'en_US');
echo strftime ("%b %d %Y %H:%M:%S", mktime (20,0,0,12,31,98))."\n";
echo gmstrftime ("%b %d %Y %H:%M:%S", mktime (20,0,0,12,31,98))."\n";
```

Lásd még a strftime() függvényt.

localtime (PHP 4)

Helyi időpontot állít elő

array **localtime** ([int timestamp [, bool is_associative]]) \linebreak

A **localtime()** függvény egy tömböt ad vissza, ugyanolyan szerkezetben, mint azt a C nyelvbeli függvényhívás teszi. A **localtime()** függvény első argumentuma az időbélyeg; ha nincs megadva, az aktuális időt használja. A második argumentum az *`is_associative`*, ha 0, vagy nincs megadva, hagyományos tömböt (numerikusan indexelve) kapunk vissza. Ha az argumentum 1, akkor a **localtime()** függvény asszociatív tömböt ad vissza, tartalmazva az összes elemet olyan struktúrában, amint azt a C függvénye is teszi a localtime függvényben. Az asszociatív tömb kulcsainak nevei a következők:

- "tm_sec" - másodperc
- "tm_min" - perc
- "tm_hour" - óra
- "tm_mday" - nap
- "tm_mon" - hónap (!!! A Január 0 !!!)
- "tm_year" - év, az 1900 óta eltelt évek száma
- "tm_wday" - hét napja
- "tm_yday" - év napja - 1
- "tm_isdst" - nyári időszámítás aktív-e

microtime (PHP 3, PHP 4)

Az aktuális UNIX időbélyeget állítja elő mikroszekundumban

string **microtime** (void) \linebreak

Visszatér egy stringgel, melynek formája "msec sec", ahol sec az Unix Epoch (0:00:00 January 1, 1970 GMT) óta eltelt másodpercek száma, és msec a mikroszekundumok száma. Ez a függvény csak olyan operációs rendszereken elérhető, amely támogatja a gettimeofday() rendszerhívást.

Mindkét részt másodpercben kapod meg.

Példa 1. microtime() example

```
function getmicrotime(){
    list($usec, $sec) = explode(" ",microtime());
    return ((float)$usec + (float)$sec);
    }

$time_start = getmicrotime();

for ($i=0; $i < 1000; $i++){
  // helyben járunk 1000-szer
```

```
}

$time_end = getmicrotime();
$time = $time_end - $time_start;

echo "Helyben jártunk $time másodpercig";
```

Lásd még time() függvényt.

mktime (PHP 3, PHP 4)

Egy időpont UNIX időbélyegét állítja elő

int **mktime** (int hour, int minute, int second, int month, int day, int year [, int is_dst]) \linebreak

Figyeljé!!! Nézd meg, hogy az argumentumok sorrendje nem éppen szokásos, mert különbözik a hagyományos UNIX-os mktime()-étól, ezért nem igazán alkalmas arra, hogy lehagyd az utolsó néhány paramétert. Gyakori hiba scriptekben az argumentumok felcserélése.

Az argumentumok által megadott időpont UNIX időbélyegét adja. Ez tulajdonképpen egy nagy egész szám, a Unix Epoch (1970 Január 1.) és az adott idő közt eltelt másodpercek száma.

Jobbról elhagyhatsz argumentumokat; ezeket a php az aktuális helyi dátum és idő alapján pótolja.

Az *Is_dst* paramétert állíthatod 1-re, jelezve, hogy a nyári időszámításban vagy, 0-ra, hogy nem, vagy -1-re (ez az alapértelmezett), ha nem tudod.

Megjegyzés: Az *Is_dst* paraméter a 3.0.10.-es verzióban került a nyelvbe.

A **mktime()** függvény hasznos a dátumokkal való manipulálás, illetve dátumellenőrzés során, mivel automatikusan "korrekt" dátummá konvertálja a helytelenül megadott bemenetet.Például, az alábbi sorok mindegyike azt írja ki, hogy "Jan-01-1998".

Példa 1. mktime() példa

```
echo date ("M-d-Y", mktime (0,0,0,12,32,1997));
echo date ("M-d-Y", mktime (0,0,0,13,1,1997));
echo date ("M-d-Y", mktime (0,0,0,1,1,1998));
echo date ("M-d-Y", mktime (0,0,0,1,1,98));
```

A *year* 2 vagy 4 jegyű szám is lehet, ha értéke 0-69 között van, akkor a gép 2000-2069 közé teszi, ha 70-99 között van, 1970-1999-be rakja. (Azon rendszereken, ahol a time_t 32 bites egész, ami manapság a legelterjedtebb, a *year* paraméter helyes értéke kb. 1902 és 2037 között van).

Adott hónap utolsó napját a következő hónap "0." napjával fejezhetjük ki (NEM a -1.-kel!!!). Az alábbi két példa mind azt írja ki, hogy "2000 februárjának utolsó napja: 29".

Példa 2. Last day of next month

```
$lastday = mktime (0,0,0,3,0,2000);
echo strftime ("2000 februárjának utolsó napja: %d", $lastday);

$lastday = mktime (0,0,0,4,-31,2000);
echo strftime ("2000 februárjának utolsó napja: %d", $lastday);
```

Az olyan dátum, amelyben az év, hónap, és a nap is nulla, nem legális (máskülönben 1999. 11. 30.-ának kellene tekinteni, ami furcsa viselkedés lenne)

Lásd még a date() és a time() függvényeket.

strftime (PHP 3, PHP 4)

Helyi időstringet formáz a megadott formátum (és időpont) szerint

string **strftime** (string format [, int timestamp]) \linebreak

A megadott formátumstringnek megfelelően formázott dátum-stringgel tér vissza, felhasználva a megadott *timestamp* paramétert, vagy ennek hiányában a helyi időt. Hónap, és napnevek, továbbá más nyelvfüggő stringek az setlocale() függvénnyel beállított értékeknek felelnek meg.

Az alábbi formátumstringeket ismeri fel a függvény:

- %a - hét napja, rövidítve
- %A - hét napja, teljes
- %b - hónap neve, rövidítve
- %B - hónap neve, teljes
- %c - dátum és idő alapértelmezett formátumban
- %C - évszám 2 utolsó jegy nélkül (trunc(év/100), értéke 00 és 99 között)
- %d - nap, 2 számjegy ("01" .. "31")
- %D - azonos %m/%d/%y-val
- %e - nap, 2 karakter, extra szóközzel az egyjegyű számok előtt (" 1" .. "31")
- %g - mint a %G, csak évszázad nélkül
- %G - A 4-jegyű évszám, ami az ISO hétszámhoz tartozik (lásd a %V opciót). Ez azonos formátumú és értékű, mint a %Y, kivéve, hogy ha az ISO hétszám az előző, vagy a következő évhez tartozik, akkor azzal az évet adja.
- %h - ugyanaz, mint a %b
- %H - óra, 24-órás formátumban, 2 számjeggyel ("00" .. "23")
- %I - óra, 12-órás formátumban, 2 számjeggyel ("01" .. "12")
- %j - év napja, 3 számjegy ("001" .. "366" !!!)

- %m - hónap, számmal ("01" .. "12") [van kitöltő szóköz]
- %M - perc, [2 számjegy]
- %n - újsor karakter
- %p - ‘am’ vagy ‘pm’, annak megfelelően, hogy az adott időpont délelőtt, vagy délután van-e
- %r - időpont 12 órás formátumban [pl.: 12:00:00 AM]
- %R - időpont 24 órás formátumban [pl.: 00:00][Itt nincs másodperc!!!]
- %S - másodperc [2 számjegy]
- %t - tabulátor
- %T - idő, azonos a %H:%M:%S-val
- %u - hét napja, számmal [1,7], az 1 a hétfőt jelenti

Figyelem

Úgy tűnik, hogy Sun Solarison a vasárnap az 1-es, holott az ISO 9889:1999 (az aktuális C standard) világosan megmondja, hogy hétfőnek kell lennie.

- %U - az aktuális év hányadik hetében járunk, a hét vasárnappal kezdődik, vagyis az "01" hét január első vasárnapján kezdődik. [Az első hét előtti napokat "00" hétnek jelzi!!!]
- %V - Az ISO 8601:1988 hétszám ("01" .. 53), ahol az az első hét, amelynek legalább 4 napja van az újévben. A hét első napja hétfő. Használd a %G-t vagy a %g-t, ha egy év kell, ami egy időbélyeg hétszámához tartozik. [???] [Helyesen kezeli az előző évből átnyúló heteket is]
- %W - év hete, az év első hétfőjén kezdődik az "01" hét
- %w - hét napja számmal, a vasárnap a 0
- %x - alapértelmezett dátumformátum idő nélkül
- %X - alapértelmezett időpontformátum dátum nélkül
- %y - év 2 számjeggyel ("00" .. "99")
- %Y - év, évszázaddal együtt
- %Z - időzóna neve [3 betű]
- %% - a ‘%’ karakter

Megjegyzés: Nem minden formátumstring van meg az összes C könyvtárban, ami azt jelenti, hogy a PHP **strftime()** függvényében sem lesznek elérhetők.

Példa 1. strftime() példa

```
setlocale (LC_TIME, "C");
print (strftime ("A finnül "));
setlocale (LC_TIME", "fi_FI");
print (strftime ("%A, franciául "));
setlocale (LC_TIME, "fr_FR");
print (strftime ("%A és magyarul "));
```

```
setlocale (LC_TIME, "hu_HU");
print (strftime ("%A.\n"));
```

Ez a példa akkor működik jól, ha a megfelelő locales beállításokat installálod.

Lásd még setlocale() és mktime() és az Open Group specification of **strftime()()** (http://www.opengroup.org/onlinepubs/7908799/xsh/strftime.html) függvényeket.

strtotime (PHP 3>= 3.0.12, PHP 4)

Mindenféle angol szöveges időpontot UNIX időbélyeggé alakít

int **strtotime** (string time [, int now]) \linebreak

A függvény egy stringet vár, amiben egy angolul írt dátum van, és megpróbálja UNIX időbélyeggé konvertálni, a `now` paramétert, vagy ha az nincs megadva, az aktuális időpontot figyelembe véve. Ha a függvény nem tudja értelmezni a karakterláncot, `-1`-gyel tér vissza.

Mivel a **strtotime()** függvény a GNU dátum szintaxisnak megfelelően viselkedik, nézd meg a Date Input Formats (http://www.gnu.org/manual/tar-1.12/html_chapter/tar_7.html) című oldalakat a unix kézikönyvben (man). Ott le van írva a *`time`* paraméter pontos szintaxisa.

Példa 1. strtotime() példa

```
echo strtotime ("now"), "\n";
echo strtotime ("10 September 2000"), "\n";
echo strtotime ("+1 day"), "\n";
echo strtotime ("+1 week"), "\n";
echo strtotime ("+1 week 2 days 4 hours 2 seconds"), "\n";
echo strtotime ("next Thursday"), "\n";
echo strtotime ("last Monday"), "\n";
```

Példa 2. Meghiúsulás-próba

```
$str = ’Nem jó’;
if (($idobelyeg = strtotime($str)) === -1) {
    echo "A string ($str) fura, nem értem";
} else {
echo "$str == ". date(’l dS of F Y h:i:s A’,$idobelyeg);
}
```

Megjegyzés: Az időpont érvényes időpontja tipikusan 1901 Dec. 13. 20:25:54 és 2038. Jan. 19. 03:14:07 közötti időpontot jelent (Greenwich-i időpont szerint) (Ezek a dátumok a legkisebb és a legnagyobb 32-bites egész szám által reprezentált dátumok)

time (PHP 3, PHP 4)

Az aktuális UNIX időbélyeget számolja ki

int **time** (void) \linebreak

A Unix Epoch (January 1 1970 00:00:00 GMT) óta eltelt másodpercek számát adja vissza.

Lásd még a date() függvényt.

XIX. dBase functions

Bevezetés

These functions allow you to access records stored in dBase-format (dbf) databases.

There is no support for indexes or memo fields. There is no support for locking, too. Two concurrent webserver processes modifying the same dBase file will very likely ruin your database.

dBase files are simple sequential files of fixed length records. Records are appended to the end of the file and delete records are kept until you call dbase_pack().

We recommend that you do not use dBase files as your production database. Choose any real SQL server instead; MySQL or Postgres are common choices with PHP. dBase support is here to allow you to import and export data to and from your web database, because the file format is commonly understood by Windows spreadsheets and organizers.

Telepítés

In order to use these functions, you must compile PHP with the `--enable-dbase` option.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

dbase_add_record (PHP 3, PHP 4)

Add a record to a dBase database

bool **dbase_add_record** (int dbase_identifier, array record) \linebreak

Adds the data in the *record* to the database. If the number of items in the supplied record isn't equal to the number of fields in the database, the operation will fail and FALSE will be returned.

dbase_close (PHP 3, PHP 4)

Close a dBase database

bool **dbase_close** (int dbase_identifier) \linebreak

Closes the database associated with *dbase_identifier*.

dbase_create (PHP 3, PHP 4)

Creates a dBase database

int **dbase_create** (string filename, array fields) \linebreak

The *fields* parameter is an array of arrays, each array describing the format of one field in the database. Each field consists of a name, a character indicating the field type, a length, and a precision.

The types of fields available are:

L

Boolean. These do not have a length or precision.

M

Memo. (Note that these aren't supported by PHP.) These do not have a length or precision.

D

Date (stored as YYYYMMDD). These do not have a length or precision.

N

Number. These have both a length and a precision (the number of digits after the decimal point).

C

String.

If the database is successfully created, a dbase_identifier is returned, otherwise FALSE is returned.

Példa 1. Creating a dBase database file

```
// "database" name
$dbname = "/tmp/test.dbf";

// database "definition"
$def =
    array(
        array("date",     "D"),
        array("name",     "C",  50),
        array("age",      "N",   3, 0),
        array("email",    "C", 128),
        array("ismember", "L")
    );

// creation
if (!dbase_create($dbname, $def))
    print "<strong>Error!</strong>";
```

dbase_delete_record (PHP 3, PHP 4)

Deletes a record from a dBase database

bool **dbase_delete_record** (int dbase_identifier, int record) \linebreak

Marks *`record`* to be deleted from the database. To actually remove the record from the database, you must also call dbase_pack().

dbase_get_record_with_names (PHP 3>= 3.0.4, PHP 4)

Gets a record from a dBase database as an associative array

array **dbase_get_record_with_names** (int dbase_identifier, int record) \linebreak

Returns the data from *`record`* in an associative array. The array also includes an associative member named 'deleted' which is set to 1 if the record has been marked for deletion (see dbase_delete_record()).

Each field is converted to the appropriate PHP type, except:

- Dates are left as strings
- Integers that would have caused an overflow (> 32 bits) are returned as strings

dbase_get_record (PHP 3, PHP 4)

Gets a record from a dBase database

array **dbase_get_record** (int dbase_identifier, int record) \linebreak

Returns the data from *record* in an array. The array is indexed starting at 0, and includes an associative member named 'deleted' which is set to 1 if the record has been marked for deletion (see dbase_delete_record().

Each field is converted to the appropriate PHP type, except:

- Dates are left as strings
- Integers that would have caused an overflow (> 32 bits) are returned as strings

dbase_numfields (PHP 3, PHP 4)

Find out how many fields are in a dBase database

int **dbase_numfields** (int dbase_identifier) \linebreak

Returns the number of fields (columns) in the specified database. Field numbers are between 0 and dbase_numfields($db)-1, while record numbers are between 1 and dbase_numrecords($db).

Példa 1. Using dbase_numfields()

```
$rec = dbase_get_record($db, $recno);
$nf  = dbase_numfields($db);
for ($i=0; $i < $nf; $i++) {
    print $rec[$i]."<br>\n";
}
```

dbase_numrecords (PHP 3, PHP 4)

Find out how many records are in a dBase database

int **dbase_numrecords** (int dbase_identifier) \linebreak

Returns the number of records (rows) in the specified database. Record numbers are between 1 and dbase_numrecords($db), while field numbers are between 0 and dbase_numfields($db)-1.

dbase_open (PHP 3, PHP 4)

Opens a dBase database

int **dbase_open** (string filename, int flags) \linebreak

The flags correspond to those for the open() system call. (Typically 0 means read-only, 1 means write-only, and 2 means read and write.)

Returns a dbase_identifier for the opened database, or FALSE if the database couldn't be opened.

Megjegyzés: Ha a safe mode be van kapcsolva, a PHP ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.

dbase_pack (PHP 3, PHP 4)

Packs a dBase database

bool **dbase_pack** (int dbase_identifier) \linebreak

Packs the specified database (permanently deleting all records marked for deletion using dbase_delete_record()).

dbase_replace_record (PHP 3>= 3.0.11, PHP 4)

Replace a record in a dBase database

bool **dbase_replace_record** (int dbase_identifier, array record, int dbase_record_number) \linebreak

Replaces the data associated with the record *`record_number`* with the data in the *`record`* in the database. If the number of items in the supplied record is not equal to the number of fields in the database, the operation will fail and FALSE will be returned.

`dbase_record_number` is an integer which spans from 1 to the number of records in the database (as returned by dbase_numrecords()).

XX. DBM Functions

Bevezetés

These functions allow you to store records stored in a dbm-style database. This type of database (supported by the Berkeley DB, GDBM (ftp://ftp.gnu.org/pub/gnu/gdbm/), and some system libraries, as well as a built-in flatfile library) stores key/value pairs (as opposed to the full-blown records supported by relational databases).

> **Megjegyzés:** However, dbm support is deprecated and you are encourged to use the Database (dbm-style) abstraction layer functions instead.

Követelmények

To use this functions you have to compile PHP with support for an underlying database. See the list of supported Databases.

Telepítés

In order to use these functions, you must compile PHP with dbm support by using the `--with-db` option. In addtion you must ensure support for an underlying database or you can use some sytem

libraries.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

The function dbmopen() returns an database identifier which is used by the other dbm-functions.

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

Példák

Példa 1. DBM example

```
$dbm = dbmopen ("lastseen", "w");
if (dbmexists ($dbm, $userid)) {
    $last_seen = dbmfetch ($dbm, $userid);
} else {
    dbminsert ($dbm, $userid, time());
}
do_stuff();
dbmreplace ($dbm, $userid, time());
dbmclose ($dbm);
```

dblist (PHP 3, PHP 4)

Describes the DBM-compatible library being used

string **dblist** (void) \linebreak

dbmclose (PHP 3, PHP 4)

Closes a dbm database

bool **dbmclose** (resource dbm_identifier) \linebreak

Unlocks and closes the specified database.

dbmdelete (PHP 3, PHP 4)

Deletes the value for a key from a DBM database

bool **dbmdelete** (resource dbm_identifier, string key) \linebreak

Deletes the value for *key* in the database.

Returns FALSE if the key didn't exist in the database.

dbmexists (PHP 3, PHP 4)

Tells if a value exists for a key in a DBM database

bool **dbmexists** (resource dbm_identifier, string key) \linebreak

Returns TRUE if there is a value associated with the *key*.

dbmfetch (PHP 3, PHP 4)

Fetches a value for a key from a DBM database

string **dbmfetch** (resource dbm_identifier, string key) \linebreak

Returns the value associated with *key*.

dbmfirstkey (PHP 3, PHP 4)

Retrieves the first key from a DBM database

string **dbmfirstkey** (resource dbm_identifier) \linebreak

Returns the first key in the database. Note that no particular order is guaranteed since the database may be built using a hash-table, which doesn’t guarantee any ordering.

dbminsert (PHP 3, PHP 4)

Inserts a value for a key in a DBM database

int **dbminsert** (resource dbm_identifier, string key, string value) \linebreak

Adds the value to the database with the specified key.

Returns -1 if the database was opened read-only, 0 if the insert was successful, and 1 if the specified key already exists. (To replace the value, use dbmreplace().)

dbmnextkey (PHP 3, PHP 4)

Retrieves the next key from a DBM database

string **dbmnextkey** (resource dbm_identifier, string key) \linebreak

Returns the next key after *key*. By calling dbmfirstkey() followed by successive calls to **dbmnextkey()** it is possible to visit every key/value pair in the dbm database. For example:

Példa 1. Visiting every key/value pair in a DBM database

```
$key = dbmfirstkey ($dbm_id);
while ($key) {
    echo "$key = " . dbmfetch ($dbm_id, $key) . "\n";
    $key = dbmnextkey ($dbm_id, $key);
}
```

dbmopen (PHP 3, PHP 4)

Opens a DBM database

resource **dbmopen** (string filename, string flags) \linebreak

The first argument is the full-path filename of the DBM file to be opened and the second is the file open mode which is one of "r", "n", "c" or "w" for read-only, new (implies read-write, and most likely will truncate an already-existing database of the same name), create (implies read-write, and will not truncate an already-existing database of the same name) and read-write respectively.

Returns an identifier to be passed to the other DBM functions on success, or FALSE on failure.

If NDBM support is used, NDBM will actually create filename.dir and filename.pag files. GDBM only uses one file, as does the internal flat-file support, and Berkeley DB creates a `filename.db` file. Note that PHP does its own file locking in addition to any file locking that may be done by the DBM library itself. PHP does not delete the `.lck` files it creates. It uses these files simply as fixed inodes on which to do the file locking. For more information on DBM files, see your Unix man pages, or obtain GNU's GDBM (ftp://ftp.gnu.org/pub/gnu/gdbm/).

> **Megjegyzés:** Ha a safe mode be van kapcsolva, a PHP ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.

dbmreplace (PHP 3, PHP 4)

Replaces the value for a key in a DBM database

int **dbmreplace** (resource dbm_identifier, string key, string value) \linebreak

Replaces the value for the specified key in the database.

This will also add the key to the database if it didn't already exist.

XXI. dbx függvények

A dbx modul adatbázis-kezelő absztrakciós réteg, ahol az X jelenti a támogatott adatbáziskezelőket. A dbx függvényekkel azonos hívási konvenciót használva lehet elérni a támogatott adatbázisok mindegyikét. Annak érdekében, hogy ezeket a függvényeket használni lehessen,a PHP-t dbx támogatással - az `--enable-dbx` kapcsoló - és minden használni kívánt adatbázisnak megfelelő kiterjesztéssel kell fordítani, azaz MySQL esetén `--with-mysql` kapcsolót is meg kell adni. A dbx függvények önmagukban nem kommunikálnak az adatbázis-kiszolgálókkal, csak az őket támogató modulokkal/kiterjesztésekkel. Ahhoz, hogy dbx modullal lehessen elérni egy adatbázist két feltételnek kell teljesülnie: 1. a megfelelő modult be kell tölteni (dl() vagy php.ini), 2. a dbx modulnak támogatnia kell a használni kívánt adatbázis-kiszolgálót. Ezek jelenleg: MySQL, PostgreSQL, Microsoft SQL Server, FrontBase, ODBC és Sybase-CT, és majd többi követi ezeket.

Arról, hogyan kell új adatbázis támogatást implementálni a dbx modulba http://www.guidance.nl/php/dbx/doc/ címen találsz leírást.

dbx_close (PHP 4 >= 4.0.6)

lezárja a nyitott adatbázis-kapcsolatot

bool **dbx_close** (object link_identifier) \linebreak

Sikeres végrehajtás esetén `TRUE`-val tér vissza, egyébként hiba esetén `FALSE`-szal.

Példa 1. dbx_close() példa

```
<?php
$link = dbx_connect (DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Nem sikerült kapcsolódni.");

print("Sikeresen kapcsolódtam.");
dbx_close($link);
?>
```

Megjegyzés: Használd mindig az adatbázis specifikus dokumentációt is!

Lásd még: dbx_connect()!

dbx_compare (PHP 4 >= 4.1.0)

összehasonlít két sort rendezés céljából

int **dbx_compare** (array row_a, array row_b, string column_key [, int flags]) \linebreak

`0`-val tér vissza, ha a `row_a[$column_key]` egyenlő a `row_b[$column_key]` kifejezéssel, és `1`-gyel, ha az első nagyobb, illetve `-1`-gyel, ha ez kisebb. Ha *`flags`* értéke `DBX_CMP_DESC`, akkor pont fordított értékek adódnak eltérő bemenetek esetén. **dbx_compare()** kisegítő függvény a dbx_sort() használatához, hogy könnyebben lehessen írni és használni saját definiálású összehasonlító függvényeket.

A *`flags`* használható arra, hogy beállítsuk az a rendezés irányát:

- `DBX_CMP_ASC` - növekvő sorrendet ír elő
- `DBX_CMP_DESC` - csökkenő sorrendet ír elő

és az összehasonlítás típusát (milyen típusra konvertálja az értékeket):

- `DBX_CMP_NATIVE` - típuskonverzió nélkül hasonlítja össze az elemeket
- `DBX_CMP_NUMBER` - számként hasonlítja össze az elemeket
- `DBX_CMP_TEXT` - szövegként hasonlítja össze az elemeket

Egy - egy irányra és típusra vonatkozó állandót a logikai VAGY kapcsolattal lehet kombinálni (|). A *`flags`* paraméter alapértelmezése a `DBX_CMP_ASC` | `DBX_CMP_NATIVE`.

Példa 1. dbx_compare() példa

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, 'pid', DBX_CMP_DESC);
    // ha 'pid' szerint nem rendezhetőek sorba, akkor 'id' szerint
    if (!$rv) {
        $rv = dbx_compare ($a, $b, 'id', DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die ("Nem sikerült csatlakozni.");

$result = dbx_query($link, "SELECT id, pid, leiras FROM tabla ORDER BY id");
    // az eredmény most 'id' szerint van rendezve

dbx_sort ($result, "user_re_order");
    // az eredmény most csökkenő 'pid' szerint és azon túl 'id' szerint van rendezve
dbx_close ($link);
?>
```

Lásd még dbx_sort()!

dbx_connect (PHP 4 >= 4.0.6)

kapcsolódik egy adatbázishoz

object **dbx_connect** (mixed module, string host, string database, string username, string password [, int persistent]) \linebreak

Sikeres csatlakozáskor egy object-et ad vissza, hiba esetén pedig `FALSE`-t. Ha sikerült a kapcsolatot létesíteni, de a megadott adatbázis nem elérhető, akkor a kapcsolatot lezárja és `FALSE`-t ad vissza. A *`persistent`* paraméternek `DBX_PERSISTENT` értéket adva perzisztens adatbázis kapcsolatot nyit.

A *`module`* paraméter lehet sztring is, de inkább a felsorolt állandók valamelyike használandó. A lehetséges értékek alább láthatók, de figyelembe veendő, hogy csak akkor működnek, ha az adott modul be van töltve.

- `DBX_MYSQL` vagy "mysql"
- `DBX_ODBC` vagy "odbc"
- `DBX_PGSQL` vagy "pgsql"
- `DBX_MSSQL` vagy "mssql"
- `DBX_FBSQL` vagy "fbsql" (PHP 4.1.1-től kezdve)
- `DBX_SYBASECT` vagy "sybase_ct" (PHP CVS csak)

A visszaadott objektum három tagváltozóval rendelkezik:

database

Az aktuálisan kiválasztott adatbázis nevét tartalmazza.

handle

Ennek a kapcsolatazonosítónak a segítségével szükség esetén az adott modulra jellemző függvények valamelyike hívható meg.

```
<?php
$link = dbx_connect (DBX_MYSQL, "localhost", "db", "username", "password");
mysql_close ($link->handle); // dbx_close($link) használata itt sokkal jobb lenne...
?>
```

module

Ezt a dbx kiterjesztés tartja fenn belső használatra, az éppen használt modul számát tárolja.

A *`host`*, *`database`*, *`username`* és a *`password`* paraméterek kötelezőek, de nem mindig kerülnek felhasználásra a modul ténylegesen kapcsolatot létesítő függvényétől függően.

Példa 1. dbx_connect() példa

```
<?php
$link = dbx_connect (DBX_ODBC, "", "db", "username", "password", DBX_PERSISTENT)
    or die ("Nem sikerült kapcsolódni.");
print ("Sikeresen kapcsolódtam.");
dbx_close ($link);
?>
```

Megjegyzés: Használd mindig az adatbázis specifikus dokumentációt is!

Lásd még: dbx_close()!

dbx_error (PHP 4 >= 4.0.6)

visszaadja a modulban történt legutolsó hibát (nem feltétlenül a kapcsolathoz kötödő hibát)

string **dbx_error** (object link_identifier) \linebreak

A hiba szövegét tartalmazó sztringgel tér vissza, amely a modulban legutolsó függvényhívás óta történt (pl. a mysql modulban). Ha az adott modulban több kapcsolat is nyitva van, akkor is csak a legutolsó hibaüzenet kerül visszadásra. Ha több modulhoz történt csatlakozás, akkor a *`link_identifier`* által meghatározott modul utolsó hibaüzenetét adja vissza.

Példa 1. dbx_error() példa

```
<?php
$link   = dbx_connect (DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Nem sikerült kapcsolódni.");

$result = dbx_query ($link, "select id from nem_letezo_tabla");

if ( $result == 0 ) {
    print dbx_error ($link);
}
dbx_close ($link);
?>
```

Megjegyzés: Használd mindig az adatbázis specifikus dokumentációt is!

Microsoft SQL Server hibaüzenetei valójában a mssql_get_last_message() függvény visszatérési értékei.

dbx_query (PHP 4 >= 4.0.6)

végrehajt egy lekérdezést és visszaadja a teljes végeredményt (ha van)

object **dbx_query** (object link_identifier, string sql_statement [, long flags]) \linebreak

Az *`sql_statement`* sikeres végrehajtásakor `1`-gyel vagy egy objektummal tér vissza, ez utóbbival csak olyan esetben, ha SQL utasításnak van végeredménye. Hiba esetén `0`-t ad vissza.

Példa 1. A visszatérési érték lekezelésére

```
<?php
$link   = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("Could not connect");

$result = dbx_query($link, 'SELECT id, szulo_id, leiras FROM tabla');

if ( is_object($result) ) {
    // ... néhány további művelet, lásd még a lenti példákat ...
    // először, a mezőnevek és típusaik kiírása
    // azután,  egy táblába kiírni a kapott végeredményt
}
else if ( $result == 1 ) {
    echo("A lekérdezést sikeresen végrehajtva, nincs visszaadott eredményhalmaz");
```

```
}
else {
    exit("Sikertelen lekérdezés");
}

dbx_close($link);
?>
```

A *`flags`* paraméteren keresztül szabályozható a visszaadandó információ mennyisége. Ez a következő állandók logikai vagy kapcsolata lehet:

`DBX_RESULT_INDEX`

Ez *mindig* felhasználásra kerül , és hatására a visszatérési érték része lesz egy két dimenziós tömb, amelynek az első indexe a sorindex, a második az oszlopindex, azaz például a `data[2][3]`-ban a `2` jelöli a sort, a `3` az oszlopot. Az indexelés 0-tól kezdődik.

Ha a `DBX_RESULT_ASSOC` is meg van adva, akkor a minden esetben a `DBX_RESULT_INFO`-nak megfelelő értékeket is szolgáltatja a függvény.

`DBX_RESULT_INFO`

Az oszlopokról ad információt: a mezők számáról, neveikről és típusaikról.

`DBX_RESULT_ASSOC`

Ennek hatására az oszlopértékeket az oszlopnevekkel is lehet indexelni, azaz a visszaadott eredményhalmaz asszociatív tömbként is használható.

Ezek az asszociatív indexek valójában referenciák a numerikus indexelésű értékekre, így a `data[0][0]`-t módosítva a `data[0]['az_első_oszlop_mező_neve']` is megváltozik.

Megjegyzendő, hogy a `DBX_RESULT_INDEX`-et mindig használja a függvény, függetlenül a *`flags`* paraméter értékétől. Emiatt valójában csak az alábbi kombinációk értelmesek:

- `DBX_RESULT_INDEX`
- `DBX_RESULT_INDEX` | `DBX_RESULT_INFO`
- `DBX_RESULT_INDEX` | `DBX_RESULT_INFO` | `DBX_RESULT_ASSOC` - alapértelmezés, ha nincs megadva a *`flags`*.

A visszaadott objektumnak négy vagy öt tagváltozója van a *`flags`*-től függően:

handle

Érvényes kapcsolatazonosító az adatbázishoz, és így szükség esetén az adott modulra jellemző függvények hívásához használható:

```
$result = dbx_query ($link, "SELECT id FROM tabla");
$length = mysql_field_len($result->handle, 0);
```

cols and rows

A `cols` és `rows` tagváltozók tárolják az eredményhalmaz méreteit: az oszlopok/mezők és a sorok/rekordok számát.

```
$result = dbx_query ($link, "SELECT id FROM tabla");
echo "Az eredmény mérete: " . $result->rows . " x " . $result->cols . "<br/>\n";
```

info (optional)

Az `info` tagváltozó csak akkor érhető el, ha a `DBX_RESULT_INFO` és/vagy `DBX_RESULT_ASSOC` szerepel a *`flags`* paraméterben. Ez olyan kétdimenzós tömb, amelynek két nevesített sora van (`name` és `type`) a mezőinformációk eléréséhez.

Példa 2. minden mező nevének és típusának kilistázása

```
$result = dbx_query ($link, 'SELECT id FROM tabla',
        DBX_RESULT_INDEX | DBX_RESULT_INFO);

for ($i = 0; $i < $result->cols; $i++ ) {
    echo $result->info['name'][$i] . "\n";
    echo $result->info['type'][$i] . "\n";
}
```

data

A `data` tagváltozó tartalmazza a tényleges végeredményt, az eredményhalmazt, ha van ilyen. Ha a `DBX_RESULT_ASSOC` is része a *`flags`*-nek, akkor az adatok `$result->data[2]["mező_neve"]` alakban is elérhetők lesznek, asszociatív tömbként kezelve a data-t.

Példa 3. a data tagváltozó kiíratása HTML táblaként

```
$result = dbx_query ($link, 'SELECT id, szulo_id, leiras FROM tabla');

echo "<table>\n";
foreach ( $result->data as $sor ) {
    echo "<tr>\n";
    foreach ( $sor as $oszlop ) {
        echo "<td>$oszlop</td>";
    }
    echo "</tr>\n";
}
echo "</table>\n";
```

Megjegyzés: Használd mindig az adatbázis specifikus dokumentációt is!

Lásd még: dbx_connect()!

dbx_sort (PHP 4 >= 4.0.6)

dbx_query által visszaadott eredményhalmazt rendezi tetszőleges függvény segítségével

bool **dbx_sort** (object result, string user_compare_function) \linebreak

Sikeres végrehajtás esetén TRUE-val tér vissza, egyébként hiba esetén FALSE-szal.

Megjegyzés: Az SQL lekérdezésen belül használt rendezés (ORDER BY) mindig jobb hatásfokú, mint a **dbx_sort()** Ha lehet, mindig az előbbit használd!

Példa 1. dbx_sort() példa

```
<?php
function user_order ($a, $b) {
    $rv = dbx_compare($a, $b, "pid", DBX_CMP_DESC);
    // ha 'pid' szerint nem rendezhetőek sorba, akkor 'id' szerint
    if ( !$rv ) {
        $rv = dbx_compare($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die ("Nem sikerült csatlakozni.");

$result = dbx_query($link, "SELECT id, pid, leiras FROM tabla ORDER BY id");
    // az eredmény most 'id' szerint van rendezve

dbx_sort ($result, "user_re_order");
    // az eredmény most csökkenő 'pid' szerint és 'id' szerint van rendezve

dbx_close ($link);
?>
```

Lásd még dbx_compare()!

XXII. DB++ Functions

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

Bevezetés

db++, made by the German company Concept asa (http://www.concept-asa.de/), is a relational database system with high performance and low memory and disk usage in mind. While providing SQL as an additional language interface, it is not really a SQL database in the first place but provides its own AQL query language which is much more influenced by the relational algebra then SQL is.

Concept asa always had an interest in supporting open source languages, db++ has had Perl and Tcl call interfaces for years now and uses Tcl as its internal stored procedure language.

Követelmények

This extension relies on external client libraries so you have to have a db++ client installed on the system you want to use this extension on.

Concept asa (http://www.concept-asa.de/) provides db++ Demo versions (http://www.concept-asa.de/down-eng.html) and documentation (http://www.concept-asa.de/downloads/doc-eng.tar.gz) for Linux, some other UNIX versions. There is also a Windows version of db++, but this extension doesn't support it (yet).

Telepítés

In order to build this extension yourself you need the db++ client libraries and header files to be installed on your system (these are included in the db++ installation archives by default). You have to run **configure** with option `--with-dbplus` to build this extension.

configure looks for the client libraries and header files under the default paths `/usr/dbplus`, `/usr/local/dbplus` and `/opt/dblus`. If you have installed db++ in a different place you have add the installation path to the **configure** option like this:

`--with-dbplus=/your/installation/path`.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

dbplus_relation

Most db++ functions operate on or return *`dbplus_relation`* resources. A *`dbplus_relation`* is a handle to a stored relation or a relation generated as the result of a query.

Előre definiált állandók

Az itt listázott állandókat ez a kiterjesztés definiálja, és csak akkor elérhetőek, ha az adott kiterjesztés be van fordítva a PHP-be, vagy dinamikusan betöltött.

db++ error codes

Táblázat 1. DB++ Error Codes

PHP Constant	db++ constant	meaning
`DBPLUS_ERR_NOERR` (integer)	ERR_NOERR	Null error condition
`DBPLUS_ERR_DUPLICATE` (integer)	ERR_DUPLICATE	Tried to insert a duplicate tuple
`DBPLUS_ERR_EOSCAN` (integer)	ERR_EOSCAN	End of scan from rget()
`DBPLUS_ERR_EMPTY` (integer)	ERR_EMPTY	Relation is empty (server)
`DBPLUS_ERR_CLOSE` (integer)	ERR_CLOSE	The server can’t close
`DBPLUS_ERR_WLOCKED` (integer)	ERR_WLOCKED	The record is write locked
`DBPLUS_ERR_LOCKED` (integer)	ERR_LOCKED	Relation was already locked
`DBPLUS_ERR_NOLOCK` (integer)	ERR_NOLOCK	Relation cannot be locked
`DBPLUS_ERR_READ` (integer)	ERR_READ	Read error on relation
`DBPLUS_ERR_WRITE` (integer)	ERR_WRITE	Write error on relation
`DBPLUS_ERR_CREATE` (integer)	ERR_CREATE	Create() system call failed
`DBPLUS_ERR_LSEEK` (integer)	ERR_LSEEK	Lseek() system call failed

PHP Constant	db++ constant	meaning
DBPLUS_ERR_LENGTH (integer)	ERR_LENGTH	Tuple exceeds maximum length
DBPLUS_ERR_OPEN (integer)	ERR_OPEN	Open() system call failed
DBPLUS_ERR_WOPEN (integer)	ERR_WOPEN	Relation already opened for writing
DBPLUS_ERR_MAGIC (integer)	ERR_MAGIC	File is not a relation
DBPLUS_ERR_VERSION (integer)	ERR_VERSION	File is a very old relation
DBPLUS_ERR_PGSIZE (integer)	ERR_PGSIZE	Relation uses a different page size
DBPLUS_ERR_CRC (integer)	ERR_CRC	Invalid crc in the superpage
DBPLUS_ERR_PIPE (integer)	ERR_PIPE	Piped relation requires lseek()
DBPLUS_ERR_NIDX (integer)	ERR_NIDX	Too many secondary indices
DBPLUS_ERR_MALLOC (integer)	ERR_MALLOC	Malloc() call failed
DBPLUS_ERR_NUSERS (integer)	ERR_NUSERS	Error use of max users
DBPLUS_ERR_PREEXIT (integer)	ERR_PREEXIT	Caused by invalid usage
DBPLUS_ERR_ONTRAP (integer)	ERR_ONTRAP	Caused by a signal
DBPLUS_ERR_PREPROC (integer)	ERR_PREPROC	Error in the preprocessor
DBPLUS_ERR_DBPARSE (integer)	ERR_DBPARSE	Error in the parser
DBPLUS_ERR_DBRUNERR (integer)	ERR_DBRUNERR	Run error in db
DBPLUS_ERR_DBPREEXIT (integer)	ERR_DBPREEXIT	Exit condition caused by prexit() * procedure
DBPLUS_ERR_WAIT (integer)	ERR_WAIT	Wait a little (Simple only)
DBPLUS_ERR_CORRUPT_TUPLE (integer)	ERR_CORRUPT_TUPLE	A client sent a corrupt tuple
DBPLUS_ERR_WARNING0 (integer)	ERR_WARNING0	The Simple routines encountered a non fatal error which was corrected
DBPLUS_ERR_PANIC (integer)	ERR_PANIC	The server should not really die but after a disaster send ERR_PANIC to all its clients
DBPLUS_ERR_FIFO (integer)	ERR_FIFO	Can’t create a fifo
DBPLUS_ERR_PERM (integer)	ERR_PERM	Permission denied
DBPLUS_ERR_TCL (integer)	ERR_TCL	TCL_error
DBPLUS_ERR_RESTRICTED (integer)	ERR_RESTRICTED	Only two users
DBPLUS_ERR_USER (integer)	ERR_USER	An error in the use of the library by an application programmer

PHP Constant	db++ constant	meaning
`DBPLUS_ERR_UNKNOWN` (integer)	ERR_UNKNOWN	

dbplus_add (4.1.0 - 4.2.1 only)

Add a tuple to a relation

int **dbplus_add** (resource relation, array tuple) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

This function will add a tuple to a relation. The *`tuple`* data is an array of attribute/value pairs to be inserted into the given *`relation`*. After successful execution the *`tuple`* array will contain the complete data of the newly created tuple, including all implicitly set domain fields like sequences.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_aql (4.1.0 - 4.2.1 only)

Perform AQL query

resource **dbplus_aql** (string query [, string server [, string dbpath]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_aql() will execute an AQL *`query`* on the given *`server`* and *`dbpath`*.

On success it will return a relation handle. The result data may be fetched from this relation by calling dbplus_next() and **dbplus_current()**. Other relation access functions will not work on a result relation.

Further information on the AQL A... Query Language is provided in the original db++ manual.

dbplus_chdir (4.1.0 - 4.2.1 only)

Get/Set database virtual current directory

string **dbplus_chdir** ([string newdir]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_chdir() will change the virtual current directory where relation files will be looked for by dbplus_open(). **dbplus_chdir()** will return the absolute path of the current directory. Calling **dbplus_chdir()** without giving any `newdir` may be used to query the current working directory.

dbplus_close (4.1.0 - 4.2.1 only)

Close a relation

int **dbplus_close** (resource relation) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Calling **dbplus_close()** will close a relation previously opened by dbplus_open().

dbplus_curr (4.1.0 - 4.2.1 only)

Get current tuple from relation

int **dbplus_curr** (resource relation, array tuple) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_curr() will read the data for the current tuple for the given `relation` and will pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_prev(), dbplus_next(), and dbplus_last().

dbplus_errcode (4.1.0 - 4.2.1 only)

Get error string for given errorcode or last error

string **dbplus_errcode** (int errno) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_errcode() returns a cleartext error string for the error code passed as `errno` of for the result code of the last db++ operation if no parameter is given.

dbplus_errno (4.1.0 - 4.2.1 only)

Get error code for last operation

int **dbplus_errno** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_errno() will return the error code returned by the last db++ operation.

See also dbplus_errcode().

dbplus_find (4.1.0 - 4.2.1 only)

Set a constraint on a relation

int **dbplus_find** (resource relation, array constraints, mixed tuple) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_find() will place a constraint on the given relation. Further calls to functions like dbplus_curr() or dbplus_next() will only return tuples matching the given constraints.

Constraints are triplets of strings containing of a domain name, a comparison operator and a comparison value. The `constraints` parameter array may consist of a collection of string arrays, each of which contains a domain, an operator and a value, or of a single string array containing a multiple of three elements.

The comparison operator may be one of the following strings: '==', '>', '>=', '<', '<=', '!=', '~' for a regular expression match and 'BAND' or 'BOR' for bitwise operations.

See also dbplus_unselect().

dbplus_first (4.1.0 - 4.2.1 only)

Get first tuple from relation

int **dbplus_first** (resource relation, array tuple) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_curr() will read the data for the first tuple for the given `relation`, make it the current tuple and pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_curr(), dbplus_prev(), dbplus_next(), and dbplus_last().

dbplus_flush (4.1.0 - 4.2.1 only)

Flush all changes made on a relation

int **dbplus_flush** (resource relation) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_flush() will write all changes applied to `relation` since the last flush to disk.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_freealllocks (4.1.0 - 4.2.1 only)

Free all locks held by this client

int **dbplus_freealllocks** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_freeaalllocks() will free all tuple locks held by this client.

See also dbplus_getlock(), dbplus_freelock(), and dbplus_freerlocks().

dbplus_freelock (4.1.0 - 4.2.1 only)

Release write lock on tuple

int **dbplus_freelock** (resource relation, string tname) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_freelock() will release a write lock on the given `tuple` previously obtained by dbplus_getlock().

See also dbplus_getlock(), dbplus_freerlocks(), and dbplus_freealllocks().

dbplus_freerlocks (4.1.0 - 4.2.1 only)

Free all tuple locks on given relation

int **dbplus_freerlocks** (resource relation) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_freerlocks() will free all tuple locks held on the given `relation`.

See also dbplus_getlock(), dbplus_freelock(), and dbplus_freealllocks().

dbplus_getlock (4.1.0 - 4.2.1 only)

Get a write lock on a tuple

int **dbplus_getlock** (resource relation, string tname) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_getlock() will request a write lock on the specified *`tuple`*. It will return zero on success or a non-zero error code, especially DBPLUS_ERR_WLOCKED, on failure.

See also dbplus_freelock(), dbplus_freerlocks(), and dbplus_freealllocks().

dbplus_getunique (4.1.0 - 4.2.1 only)

Get a id number unique to a relation

int **dbplus_getunique** (resource relation, int uniqueid) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_getunique() will obtain a number guaranteed to be unique for the given *`relation`* and will pass it back in the variable given as *`uniqueid`*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_info (4.1.0 - 4.2.1 only)

???

int **dbplus_info** (resource relation, string key, array) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Not implemented yet.

dbplus_last (4.1.0 - 4.2.1 only)

Get last tuple from relation

int **dbplus_last** (resource relation, array tuple) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_curr() will read the data for the last tuple for the given `relation`, make it the current tuple and pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_next().

dbplus_lockrel (unknown)

Request write lock on relation

int **dbplus_lockrel** (resource relation) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_lockrel() will request a write lock on the given relation. Other clients may still query the relation, but can’t alter it while it is locked.

dbplus_next (4.1.0 - 4.2.1 only)

Get next tuple from relation

int **dbplus_next** (resource relation, array) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_curr() will read the data for the next tuple for the given `relation`, will make it the current tuple and will pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_last().

dbplus_open (4.1.0 - 4.2.1 only)

Open relation file

resource **dbplus_open** (string name) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

The relation file `name` will be opened. `name` can be either a file name or a relative or absolute path name. This will be mapped in any case to an absolute relation file path on a specific host machine and server.

On success a relation file resource (cursor) is returned which must be used in any subsequent commands referencing the relation. Failure leads to a zero return value, the actual error code may be asked for by calling dbplus_errno().

dbplus_prev (4.1.0 - 4.2.1 only)

Get previous tuple from relation

int **dbplus_prev** (resource relation, array tuple) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_curr() will read the data for the next tuple for the given `relation`, will make it the current tuple and will pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_next(), and dbplus_last().

dbplus_rchperm (4.1.0 - 4.2.1 only)

Change relation permissions

int **dbplus_rchperm** (resource relation, int mask, string user, string group) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_rchperm() will change access permissions as specified by `mask`, `user` and `group`. The values for these are operating system specific.

dbplus_rcreate (4.1.0 - 4.2.1 only)

Creates a new DB++ relation

resource **dbplus_rcreate** (string name, mixed domlist [, boolean overwrite]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_rcreate() will create a new relation named `name`. An existing relation by the same name will only be overwritten if the relation is currently not in use and `overwrite` is set to TRUE.

`domlist` should contain the domain specification for the new relation within an array of domain description strings. (**dbplus_rcreate()** will also accept a string with space delimited domain

description strings, but it is recommended to use an array). A domain description string consists of a domain name unique to this relation, a slash and a type specification character. See the db++ documentation, especially the dbcreate(1) manpage, for a description of available type specifiers and their meanings.

dbplus_rcrtexact (4.1.0 - 4.2.1 only)

Creates an exact but empty copy of a relation including indices

resource **dbplus_rcrtexact** (string name, resource relation, boolean overwrite) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_rcrtexact() will create an exact but empty copy of the given `relation` under a new `name`. An existing relation by the same `name` will only be overwritten if `overwrite` is TRUE and no other process is currently using the relation.

dbplus_rcrtlike (4.1.0 - 4.2.1 only)

Creates an empty copy of a relation with default indices

resource **dbplus_rcrtlike** (string name, resource relation, int flag) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_rcrtexact() will create an empty copy of the given `relation` under a new `name`, but with default indices. An existing relation by the same `name` will only be overwritten if `overwrite` is TRUE and no other process is currently using the relation.

dbplus_resolve (4.1.0 - 4.2.1 only)

Resolve host information for relation

int **dbplus_resolve** (string relation_name) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_resolve() will try to resolve the given *relation_name* and find out internal server id, real hostname and the database path on this host. The function will return an array containing these values under the keys 'sid', 'host' and 'host_path' or FALSE on error.

See also dbplus_tcl().

dbplus_restorepos (4.1.0 - 4.2.1 only)

???

int **dbplus_restorepos** (resource relation, array tuple) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Not implemented yet.

dbplus_rkeys (4.1.0 - 4.2.1 only)

Specify new primary key for a relation

resource **dbplus_rkeys** (resource relation, mixed domlist) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_rkeys() will replace the current primary key for *relation* with the combination of domains specified by *domlist*.

domlist may be passed as a single domain name string or as an array of domain names.

dbplus_ropen (4.1.0 - 4.2.1 only)

Open relation file local

resource **dbplus_ropen** (string name) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_ropen() will open the relation *file* locally for quick access without any client/server overhead. Access is read only and only **dbplus_current()** and dbplus_next() may be applied to the returned relation.

dbplus_rquery (4.1.0 - 4.2.1 only)

Perform local (raw) AQL query

int **dbplus_rquery** (string query, string dbpath) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_rquery() performs a local (raw) AQL query using an AQL interpreter embedded into the db++ client library. **dbplus_rquery()** is faster than dbplus_aql() but will work on local data only.

dbplus_rrename (4.1.0 - 4.2.1 only)

Rename a relation

int **dbplus_rrename** (resource relation, string name) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_rrename() will change the name of *relation* to *name*.

dbplus_rsecindex (4.1.0 - 4.2.1 only)

Create a new secondary index for a relation

resource **dbplus_rsecindex** (resource relation, mixed domlist, int type) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_rsecindex() will create a new secondary index for *`relation`* with consists of the domains specified by *`domlist`* and is of type *`type`*

`domlist` may be passed as a single domain name string or as an array of domain names.

dbplus_runlink (4.1.0 - 4.2.1 only)

Remove relation from filesystem

int **dbplus_runlink** (resource relation) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_unlink() will close and remove the *`relation`*.

dbplus_rzap (4.1.0 - 4.2.1 only)

Remove all tuples from relation

int **dbplus_rzap** (resource relation) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_rzap() will remove all tuples from *`relation`*.

dbplus_savepos (4.1.0 - 4.2.1 only)

???

int **dbplus_savepos** (resource relation) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Not implemented yet.

dbplus_setindex (4.1.0 - 4.2.1 only)

???

int **dbplus_setindex** (resource relation, string idx_name) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Not implemented yet.

dbplus_setindexbynumber (4.1.0 - 4.2.1 only)

???

int **dbplus_setindexbynumber** (resource relation, int idx_number) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Not implemented yet.

dbplus_sql (4.1.0 - 4.2.1 only)

Perform SQL query

resource **dbplus_sql** (string query, string server, string dbpath) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Not implemented yet.

dbplus_tcl (4.1.0 - 4.2.1 only)

Execute TCL code on server side

int **dbplus_tcl** (int sid, string script) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

A db++ server will prepare a TCL interpreter for each client connection. This interpreter will enable the server to execute TCL code provided by the client as a sort of stored procedures to improve the performance of database operations by avoiding client/server data transfers and context switches.

dbplus_tcl() needs to pass the client connection id the TCL `script` code should be executed by. dbplus_resolve() will provide this connection id. The function will return whatever the TCL code returns or a TCL error message if the TCL code fails.

See also dbplus_resolve().

dbplus_tremove (4.1.0 - 4.2.1 only)

Remove tuple and return new current tuple

int **dbplus_tremove** (resource relation, array tuple [, array current]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_tremove() removes *`tuple`* from *`relation`* if it perfectly matches a tuple within the relation. *`current`*, if given, will contain the data of the new current tuple after calling **dbplus_tremove()**.

dbplus_undo (4.1.0 - 4.2.1 only)

???

int **dbplus_undo** (resource relation) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Not implemented yet.

dbplus_undoprepare (4.1.0 - 4.2.1 only)

???

int **dbplus_undoprepare** (resource relation) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Not implemented yet.

dbplus_unlockrel (4.1.0 - 4.2.1 only)

Give up write lock on relation

int **dbplus_unlockrel** (resource relation) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_unlockrel() will release a write lock previously obtained by dbplus_lockrel().

dbplus_unselect (4.1.0 - 4.2.1 only)

Remove a constraint from relation

int **dbplus_unselect** (resource relation) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Calling **dbplus_unselect()** will remove a constraint previously set by dbplus_find() on `relation`.

dbplus_update (4.1.0 - 4.2.1 only)

Update specified tuple in relation

int **dbplus_update** (resource relation, array old, array new) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_update() replaces the tuple given by `old` with the data from `new` if and only if `old` completely matches a tuple within `relation`.

dbplus_xlockrel (4.1.0 - 4.2.1 only)

Request exclusive lock on relation

int **dbplus_xlockrel** (resource relation) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_xlockrel() will request an exclusive lock on *`relation`* preventing even read access from other clients.

See also dbplus_xunlockrel().

dbplus_xunlockrel (4.1.0 - 4.2.1 only)

Free exclusive lock on relation

int **dbplus_xunlockrel** (resource relation) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

dbplus_xunlockrel() will release an exclusive lock on *`relation`* previously obtained by dbplus_xlockrel().

XXIII. Direct IO functions

Bevezetés

PHP supports the direct io functions as described in the Posix Standard (Section 6) for performing I/O functions at a lower level than the C-Language stream I/O functions (fopen, fread,..).

Követelmények

Az itt leírt függvények a standard modulban találhatóak, ami mindig rendelkezésre áll.

Installation

To get these functions to work, you have to configure PHP with `--enable-dio`.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

One resource type is defined by this extension: a file descriptor returnded by dio_open().

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

dio_close (PHP 4 >= 4.2.0)

Closes the file descriptor given by fd

void **dio_close** (resource fd) \linebreak

The function **dio_close()** closes the file descriptor *resource*.

dio_fcntl (PHP 4 >= 4.2.0)

Performs a c library fcntl on fd

mixed **dio_fcntl** (resource fd, int cmd [, mixed arg]) \linebreak

The **dio_fcntl()** function performs the operation specified by *cmd* on the file descriptor *fd*. Some commands require additional arguments *args* to be supplied.

arg is an associative array, when *cmd* is F_SETLK or F_SETLLW, with the following keys:

- "start" - offset where lock begins
- "length" - size of locked area. zero means to end of file
- "wenth" - Where l_start is relative to: can be SEEK_SET, SEEK_END and SEEK_CUR
- "type" - type of lock: can be F_RDLCK (read lock), F_WRLCK (write lock) or F_UNLCK (unlock)

cmd can be one of the following operations:

- F_SETLK - Lock is set or cleared. If the lock is held by someone else **dio_fcntl()** returns -1.
- F_SETLKW - like F_SETLK, but in case the lock is held by someone else, **dio_fcntl()** waits until the lock is released.
- F_GETLK - **dio_fcntl()** returns an associative array (as described above) if someone else prevents lock. If there is no obstruction key "type" will set to F_UNLCK.
- F_DUPFD - finds the lowest numbered available file descriptor greater or equal than *arg* and returns them.

dio_open (PHP 4 >= 4.2.0)

Opens a new filename with specified permissions of flags and creation permissions of mode

resource **dio_open** (string filename, int flags [, int mode]) \linebreak

dio_open() opens a file and returns a new file descriptor for it, or -1 if any error occurred. If *flags* is O_CREAT, optional third parameter *mode* will set the mode of the file (creation permissions). The *flags* parameter can be one of the following options:

- O_RDONLY - opens the file for read access
- O_WRONLY - opens the file for write access
- O_RDWR - opens the file for both reading and writing

The *flags* parameter can also include any combination of the following flags:

- O_CREAT - creates the file, if it doesn't already exist
- O_EXCL - if both, O_CREAT and O_EXCL are set, **dio_open()** fails, if file already exists
- O_TRUNC - if file exists, and its opened for write access, file will be truncated to zero length.
- O_APPEND - write operations write data at the end of file
- O_NONBLOCK - sets non blocking mode

dio_read (PHP 4 >= 4.2.0)

Reads n bytes from fd and returns them, if n is not specified, reads 1k block

string **dio_read** (resource fd [, int n]) \linebreak

The function **dio_read()** reads and returns *n* bytes from file with descriptor *resource*. If *n* is not specified, **dio_read()** reads 1K sized block and returns them.

dio_seek (PHP 4 >= 4.2.0)

Seeks to pos on fd from whence

int **dio_seek** (resource fd, int pos, int whence) \linebreak

The function **dio_seek()** is used to change the file position of the file with descriptor *resource*. The parameter *whence* specifies how the position *pos* should be interpreted:

- SEEK_SET - specifies that *pos* is specified from the beginning of the file
- SEEK_CUR - Specifies that *pos* is a count of characters from the current file position. This count may be positive or negative
- SEEK_END - Specifies that *pos* is a count of characters from the end of the file. A negative count specifies a position within the current extent of the file; a positive count specifies a position past the current end. If you set the position past the current end, and actually write data, you will extend the file with zeros up to that position

dio_stat (PHP 4 >= 4.2.0)

Gets stat information about the file descriptor fd

array **dio_stat** (resource fd) \linebreak

Function **dio_stat()** returns information about the file with file descriptor `fd`. **dio_stat()** returns an associative array with the following keys:

- "device" - device
- "inode" - inode
- "mode" - mode
- "nlink" - number of hard links
- "uid" - user id
- "gid" - group id
- "device_type" - device type (if inode device)
- "size" - total size in bytes
- "blocksize" - blocksize
- "blocks" - number of blocks allocated
- "atime" - time of last access
- "mtime" - time of last modification
- "ctime" - time of last change

On error **dio_stat()** returns NULL.

dio_truncate (PHP 4 >= 4.2.0)

Truncates file descriptor fd to offset bytes

bool **dio_truncate** (resource fd, int offset) \linebreak

Function **dio_truncate()** causes the file referenced by `fd` to be truncated to at most `offset` bytes in size. If the file previously was larger than this size, the extra data is lost. If the file previously was shorter, it is unspecified whether the file is left unchanged or is extended. In the latter case the extended part reads as zero bytes. Returns 0 on success, otherwise -1.

dio_write (PHP 4 >= 4.2.0)

Writes data to fd with optional truncation at length

int **dio_write** (resource fd, string data [, int len]) \linebreak

The function **dio_write()** writes up to `len` bytes from `data` to file `fd`. If `len` is not specified, **dio_write()** writes all `data` to the specified file. **dio_write()** returns the number of bytes written to `fd`.

XXIV. Könyvtárkezelő függvények

chdir (PHP 3, PHP 4)

Könyvtárat vált

int **chdir** (string directory) \linebreak

Megváltoztatja a PHP aktuális könyvtárát a *directory* könyvtárra. FALSE értékkel tér vissza, ha nem lehet a könyvtárváltást teljesíteni, egyébként TRUE értéket ad.

chroot (PHP 4 >= 4.0.5)

Megváltoztatja a gyökérkönyvtárat

bool **chroot** (string directory) \linebreak

Megváltoztatja az aktuális processz gyökérkönyvtárát a *directory* paraméterben megadott értékre. FALSE értékkel tér vissza, ha nem jár sikerrel, TRUE értéket ad egyébként.

> **Megjegyzés:** Nem helyes ennek a függvénynek a használata webszerver környezetben, mivel nem lehetséges a gyökérkönyvtár visszaállítása a / könyvtárra a kérés végeztével. Ez a függvény csak CGI módban fog megfelelően funkcionálni.

dir (PHP 3, PHP 4)

Könyvtár osztály

object **dir** (string directory) \linebreak

Ez egy majdnem-objektum-orientált eljárás könyvtár listázásra. A megadott *directory* paraméter által meghatározott könyvtárat megnyitja. Ha a könyvtár megnyílt, a visszaadott objektum két tulajdonsága érhető el. A handle tulajdonság más könyvtárkezelő függvényekkel (mint pl. a readdir(), rewinddir() vagy a closedir()) való használatra szolgál. A path tulajdonság tartalmazza a könyvtár elérési útját. Három metódust alkalmazhatunk: read, rewind és close.

Példa 1. dir() példa

```
$d = dir("/etc");
echo "Handle: ".$d->handle."<br>\n";
echo "Path: ".$d->path."<br>\n";
while ($entry = $d->read()) {
    echo $entry."<br>\n";
}
$d->close();
```

closedir (PHP 3, PHP 4)

Bezárja a könyvtárat

void **closedir** (int dir_handle) \linebreak

Bezárja a *`dir_handle`* paraméterrel azonosított könyvtárat. A könyvtárat előzőleg meg kell nyitni az opendir() függvénnyel.

getcwd (PHP 4)

Az aktuális munkakönytárat adja meg

string **getcwd** (void) \linebreak

Visszaadja az aktuális munkakönyvtárat.

opendir (PHP 3, PHP 4)

Megnyit egy könyvtárat

int **opendir** (string path) \linebreak

Egy könyvtárazonosítóval tér vissza, amit későbbi closedir(), readdir(), és rewinddir() hívásokban használhatsz.

Ha a *`path`* nem egy érvényes könyvtárat ad meg, vagy a könyvtár nem megnyítható jogosultsági korlátozások, vagy filerendszer hibák miatt, az **opendir()** `FALSE` értéket ad vissza, és PHP hibát ad. Letithatod az **opendir()** során fellépő hiba kiírását, ha egy ‘@’ jelet teszel a függvény neve elé.

Példa 1. opendir() példa

```
<?php

if ($dir = @opendir("/tmp")) {
  while ($file = readdir($dir)) {
    echo "$file\n";
  }
  closedir($dir);
}

?>
```

readdir (PHP 3, PHP 4)

Adott könyvtárból beolvas egy bejegyzést

string **readdir** (int dir_handle) \linebreak

A könyvtárban levő következő file nevével tér vissza. A fileneveket nem rendezetten adja vissza.

Példa 1. Összes file kilistázása az aktuális könyvtárban

```
<?php
$handle = opendir('.');
echo "Könyvtár azonosító: $handle\n";
echo "Tartalma:\n";
while ($file = readdir($handle)) {
    echo "$file\n";
}
closedir($handle);
?>
```

Figyelj rá, hogy a **readdir()** függvény a `.` és a `..` bejegyzéseket is visszaadja. Ha ezt nem szeretnéd, akkor hagyd ki azokat így:

Példa 2. Összes file listázása az aktuális könyvtárban a `.` és a `..` bejegyzések nélkül

```
<?php
$handle = opendir('.');
while (false !== ($file = readdir($handle))) {
    if ($file != "." && $file != "..") {
        echo "$file\n";
    }
}
closedir($handle);
?>
```

rewinddir (PHP 3, PHP 4)

Visszaállítja a könyvtárkezelőt

void **rewinddir** (int dir_handle) \linebreak

Visszaállítja a *`dir_handle`* kezelővel megnyitott könyvtárat az alapállapotba, vagyis a könyvtár "elejére megy".

XXV. DOM XML függvények

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

Ezek a függvények csak akkor használhatóak, ha a PHP-t a `--with-dom=[DIR]` paraméterrel fordítottad, felhasználva a GNOME xml könyvtárat. Szükséged lesz minimum a libxml-2.0.0-ra (a béta verzióval nem fog működni!) Ezek a függvények a PHP 4-ben kerültek a nyelvbe.

Ez a modul az alábbi konstansokat definiálja:

Táblázat 1. XML konstansok

Konstans	Érték	Leírás
XML_ELEMENT_NODE	1	
XML_ATTRIBUTE_NODE	2	
XML_TEXT_NODE	3	
XML_CDATA_SECTION_NODE	4	
XML_ENTITY_REF_NODE	5	
XML_ENTITY_NODE	6	
XML_PI_NODE	7	
XML_COMMENT_NODE	8	
XML_DOCUMENT_NODE	9	
XML_DOCUMENT_TYPE_NODE	10	
XML_DOCUMENT_FRAG_NODE	11	
XML_NOTATION_NODE	12	
XML_GLOBAL_NAMESPACE	1	
XML_LOCAL_NAMESPACE	2	

Ez a modul számos osztályt definiál. A DOM XML függvények visszaadnak egy végigjárt XML dokumentum fát, amelynek minden egyes csomópontja ezen osztályok egyikéhez tartozik.

DomAttribute->name (unknown)

Returns name of attribute

bool **DomAttribute->name** (void) \linebreak

This function returns the name of the attribute.

See also DomAttribute_value().

DomAttribute->specified (unknown)

Checks if attribute is specified

bool **DomAttribute->specified** (void) \linebreak

Check DOM standard for a detailed explanation.

DomAttribute->value (unknown)

Returns value of attribute

bool **DomAttribute->value** (void) \linebreak

This function returns the value of the attribute.

See also DomAttribute_name().

DomDocument->add_root [deprecated] (unknown)

Adds a root node

resource **DomDocument->add_root** (string name) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Adds a root element node to a dom document and returns the new node. The element name is given in the passed parameter.

Példa 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->add_root("HTML");
$head = $root->new_child("HEAD", "");
$head->new_child("TITLE", "Hier der Titel");
echo htmlentities($doc->dump_mem());
?>
```

DomDocument->create_attribute (unknown)

Create new attribute

object **DomDocument->create_attribute** (string name, string value) \linebreak

This function returns a new instance of class `DomAttribute`. The name of the attribute is the value of the first parameter. The value of the attribute is the value of the second parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_cdata_section(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_cdata_section (unknown)

Create new cdata node

string **DomDocument->create_cdata_section** (string content) \linebreak

This function returns a new instance of class `DomCData`. The content of the cdata is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_comment (unknown)

Create new comment node

object **DomDocument->create_comment** (string content) \linebreak

This function returns a new instance of class `DomComment`. The content of the comment is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_element (unknown)

Create new element node

object **DomDocument->create_element** (string name) \linebreak

This function returns a new instance of class `DomElement`. The tag name of the element is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), **DomDocument_create_text()**, DomDocument_create_comment(), DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_entity_reference (unknown)

object **DomDocument->create_entity_reference** (string content) \linebreak

This function returns a new instance of class `DomEntityReference`. The content of the entity reference is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_cdata_section(), DomDocument_create_processing_instruction(), DomDocument_create_attribute(), DomNode_insert_before().

DomDocument->create_processing_instruction (unknown)

Creates new PI node

string **DomDocument->create_processing_instruction** (string content) \linebreak

This function returns a new instance of class `DomCData`. The content of the pi is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_cdata_section(), DomDocument_create_attribute(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_text_node (unknown)

Create new text node

object **DomDocument->create_text_node** (string content) \linebreak

This function returns a new instance of class `DomText`. The content of the text is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), DomDocument_create_comment(), **DomDocument_create_text()**, DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->doctype (unknown)

Returns the document type

object **DomDocument->doctype** (void) \linebreak

This function returns an object of class `DomDocumentType`. In versions of PHP before 4.3 this has been the class `Dtd`, but the DOM Standard does not know such a class.

See also the methods of class `DomDocumentType`.

DomDocument->document_element (unknown)

Returns root element node

object **DomDocument->document_element** (void) \linebreak

This function returns the root element node of a document.

The following example returns just the element with name CHAPTER and prints it. The other node -- the comment -- is not returned.

Példa 1. Retrieving root element

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
print_r($root);
?>
```

DomDocument->dump_file (unknown)

Dumps the internal XML tree back into a file

string **DomDocument->dump_file** (string filename [, bool compressionmode [, bool format]]) \linebreak

Creates an XML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below. The *format* specifies whether the output should be neatly formatted, or not. The first parameter specifies the name of the filename and the second parameter, whether it should be compressed or not.

Példa 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
$doc->dump_file("/tmp/test.xml", false, true);
?>
```

See also DomDocument_dump_mem() DomDocument_html_dump_mem().

DomDocument->dump_mem (unknown)

Dumps the internal XML tree back into a string

string **DomDocument->dump_mem** ([bool format]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Creates an XML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below. The *`format`* specifies whether the output should be neatly formatted, or not.

Példa 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
echo "<PRE>";
echo htmlentities($doc->dump_mem(true));
echo "</PRE>";
?>
```

Megjegyzés: The first parameter was added in PHP 4.3.0.

See also DomDocument_dump_file(), DomDocument_html_dump_mem().

DomDocument->get_element_by_id (unknown)

Searches for an element with a certain id

object **DomDocument->get_element_by_id** (string id) \linebreak

This function is similar to DomDocument_get_elements_by_tagname() but searches for an element with a given id. According to the DOM standard this requires a DTD which defines the attribute ID to be of type ID, though the current implementation simply does an xpath search for "//*[@ID = '%s']". This does not comply to the DOM standard which requires to return null if it is not known which attribute is of type id. This behaviour is likely to be fixed, so do not rely on the current behaviour.

See also DomDocument_get_elements_by_tagname()

DomDocument->get_elements_by_tagname (unknown)

array **DomDocument->get_elements_by_tagname** (string name) \linebreak

See also DomDocument_add_root()

DomDocument->html_dump_mem (unknown)

Dumps the internal XML tree back into a string as HTML

string **DomDocument->html_dump_mem** (void) \linebreak

Creates an HTML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below.

Példa 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
echo "<PRE>";
echo htmlentities($doc->html_dump_mem());
echo "</PRE>";
?>
```

See also DomDocument_dump_file(), DomDocument_html_dump_mem().

DomDocumentType->entities (unknown)

Returns list of entities

array **DomDocumentType->entities** (void) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomDocumentType->internal_subset (unknown)

Returns internal subset

bool **DomDocumentType->internal_subset** (void) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomDocumentType->name (unknown)

Returns name of document type

string **DomDocumentType->name** (void) \linebreak

This function returns the name of the document type.

DomDocumentType->notations (unknown)

Returns list of notations

array **DomDocumentType->notations** (void) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomDocumentType->public_id (unknown)

Returns public id of document type

string **DomDocumentType->public_id** (void) \linebreak

This function returns the public id of the document type.

The following example echos nothing.

Példa 1. Retrieving the public id

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$doctype = $dom->doctype();
echo $doctype->public_id();
?>
```

DomDocumentType->system_id (unknown)

Returns system id of document type

string **DomDocumentType->system_id** (void) \linebreak

Returns the system id of the document type.

The following example echos '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'.

Példa 1. Retrieving the system id

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$doctype = $dom->doctype();
```

```
echo $doctype->system_id();
?>
```

DomElement->get_attribute_node (unknown)

Returns value of attribute

object **DomElement->get_attribute_node** (object attr) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomElement->get_attribute (unknown)

Returns value of attribute

object **DomElement->get_attribute** (string name) \linebreak

Returns the attribute with name *name* of the current node.

See also DomElement_set_attribute()

DomElement->get_elements_by_tagname (unknown)

Adds new attribute

bool **DomElement->get_elements_by_tagname** (string name) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomElement->has_attribute (unknown)

Adds new attribute

bool **DomElement->has_attribute** (string name) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomElement->remove_attribute (unknown)

Adds new attribute

bool **DomElement->remove_attribute** (string name) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomElement->set_attribute (unknown)

Adds new attribute

bool **DomElement->set_attribute** (string name, string value) \linebreak

Sets an attribute with name *name* ot the given value. If the attribute does not exist, it will be created.

Példa 1. Setting an attribute

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

See also DomElement_get_attribute()

DomElement->tagname (unknown)

Returns name of element

string **DomElement->tagname** (void) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomNode->append_child (unknown)

Adds new child at the end of the children

object **DomNode->append_child** (object newnode) \linebreak

This functions appends a child to an existing list of children or creates a new list of children. The child can be created with e.g. DomDocument_create_element(), **DomDocument_create_text()** etc. or simply by using any other node.

Before a new child is appended it is first duplicated. Therefore the new child is a completely new copy which can be modified without changing the node which was passed to this function. If the node passed has children itself, they will be duplicated as well, which makes it quite easy to duplicate large parts of a xml document. The return value is the appended child. If you plan to do further modifications on the appended child you must use the returned node.

The following example will add a new element node to a fresh document and sets the attribute "align" to "left".

Példa 1. Adding a child

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

The above example could also be written as the following:

Példa 2. Adding a child

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
```

```
$node->set_attribute("align", "left");
$newnode = $doc->append_child($node);
?>
```

A more comples example is the one below. It first searches for a certain element, duplicates it including its children and adds it as a sibling. Finally a new attribute is added to one of the children of the new sibling and the whole document is dumped.

Példa 3. Adding a child

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$parent = $element->parent_node();
$newnode = $parent->append_child($element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

echo "<PRE>";
$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
echo "</PRE>";
?>
```

The above example could also be done with DomNode_insert_before() instead of DomNode_append_child().

See also DomNode_insert_before().

DomNode->append_sibling (unknown)

Adds new sibling to a node

object **DomNode->append_sibling** (object newnode) \linebreak

This functions appends a sibling to an existing node. The child can be created with e.g. DomDocument_create_element(), **DomDocument_create_text()** etc. or simply by using any other node.

Before a new sibling is added it is first duplicated. Therefore the new child is a completely new copy which can be modified without changing the node which was passed to this function. If the node passed has children itself, they will be duplicated as well, which makes it quite easy to duplicate large parts of a xml document. The return value is the added sibling. If you plan to do further modifications on the added sibling you must use the returned node.

This function has been added to provide the behaviour of DomNode_append_child() as it works till PHP 4.2.

See also **DomNode_append_before()**.

DomNode->attributes (unknown)

Returns list of attributes

array **DomNode->attributes** (void) \linebreak

This function only returns an array of attributes if the node is of type XML_ELEMENT_NODE.

DomNode->child_nodes (unknown)

Returns children of node

array **DomNode->child_nodes** (void) \linebreak

Returns all children of the node.

See also DomNode_next_sibling(), DomNode_previous_sibling().

DomNode->clone_node (unknown)

Clones a node

object **DomNode->clone_node** (void) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomNode->dump_node (unknown)

Dumps a single node

string **DomNode->dump_node** (void) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

See also DomDocument_dump_mem().

DomNode->first_child (unknown)

Returns first child of node

bool **DomNode->first_child** (void) \linebreak

Returns the first child of the node.

See also DomNode_last_child(), DomNode_next_sibling(), DomNode_previous_sibling().

DomNode->get_content (unknown)

Gets content of node

string **DomNode->get_content** (void) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomNode->has_attributess (unknown)

Checks if node has attributes

bool **DomNode->has_attributes** (void) \linebreak

This function checks if the node has attributes.

See also DomNode_has_child_nodes().

DomNode->has_child_nodes (unknown)

Checks if node has children

bool **DomNode->has_child_nodes** (void) \linebreak

This function checks if the node has children.

See also DomNode_child_nodes().

DomNode->insert_before (unknown)

Inserts new node as child

object **DomNode->insert_before** (object newnode, object refnode) \linebreak

This function inserts the new node *`newnode`* right before the node *`refnode`*. The return value is the inserted node. If you plan to do further modifications on the appended child you must use the returned node.

DomNode_insert_before() is very similar to DomNode_append_child() as the following example shows which does the same as the example at DomNode_append_child().

Példa 1. Adding a child

```
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$newnode = $element->insert_before($element, $element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

echo "<PRE>";
$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
echo "</PRE>";
```

See also DomNode_append_child().

DomNode->is_blank_node (unknown)

Checks if node is blank

bool **DomNode->is_blank_node** (void) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomNode->last_child (unknown)

Returns last child of node

object **DomNode->last_child** (void) \linebreak

Returns the last child of the node.

See also DomNode_first_child(), DomNode_next_sibling(), DomNode_previous_sibling().

DomNode->next_sibling (unknown)

Returns the next sibling of node

object **DomNode->next_sibling** (void) \linebreak

This function returns the next sibling of the current node. If there is no next sibling it returns false. You can use this function to iterate over all children of a node as shown in the example.

Példa 1. Iterate over children

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$child = $element->first_child();

while($child) {
   print_r($child);
   $child = $child->next_sibling();
```

```
}
?>
```

See also DomNode_previous_sibling().

DomNode->node_name (unknown)

Returns name of node

string **DomNode->node_name** (void) \linebreak

Returns name of the node. The name has different meanings for the different types of nodes as illustrated in the following table.

Táblázat 1. Meaning of value

Type	Meaning
DomAttribute	value of attribute
DomAttribute	
DomCDataSection	#cdata-section
DomComment	#comment
DomDocument	#document
DomDocumentType	document type name
DomElement	tag name
DomEntity	name of entity
DomEntityReference	name of entity reference
DomNotation	notation name
DomProcessingInstruction	target
DomText	#text

DomNode->node_type (unknown)

Returns type of node

int **DomNode->node_type** (void) \linebreak

Returns the type of the node. All possible types are listed in the table in the introduction.

DomNode->node_value (unknown)

Returns value of a node

string **DomNode->node_value** (void) \linebreak

Returns value of the node. The value has different meanings for the different types of nodes as illustrated in the following table.

Táblázat 1. Meaning of value

Type	Meaning
DomAttribute	value of attribute
DomAttribute	
DomCDataSection	content
DomComment	content of comment
DomDocument	null
DomDocumentType	null
DomElement	null
DomEntity	null
DomEntityReference	null
DomNotation	null
DomProcessingInstruction	entire content without target
DomText	content of text

DomNode->owner_document (unknown)

Returns the document this node belongs to

object **DomNode->owner_document** (void) \linebreak

This function returns the document the current node belongs to.

The following example will create two identical lists of children.

Példa 1. Finding the document of a node

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node = $doc->append_child($node);
$children = $doc->children();
print_r($children);

$doc2 = $node->owner_document();
$children = $doc2->children();
print_r($children);
```

```
?>
```

See also DomNode_insert_before().

DomNode->parent_node (unknown)

Returns the parent of the node

object **DomNode->parent_node** (void) \linebreak

This function returns the parent node.

The following example will show two identical lists of children.

Példa 1. Finding the document of a node

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node = $doc->append_child($node);
$children = $doc->children();
print_r($children);

$doc2 = $node->parent_node();
$children = $doc2->children();
print_r($children);
?>
```

DomNode->prefix (unknown)

Returns name space prefix of node

string **DomNode->prefix** (void) \linebreak

Returns the name space prefix of the node.

DomNode->previous_sibling (unknown)

Returns the previous sibling of node

object **DomNode->previous_sibling** (void) \linebreak

This function returns the previous sibling of the current node.

See also DomNode_next_sibling().

DomNode->remove_child (unknown)

Removes child from list of children

object **DomNode->remove_child** (object oldchild) \linebreak

This functions removes a child from a list of children. If child cannot be removed or is not a child the function will return false. If the child could be removed the functions returns the old child.

Példa 1. Removing a child

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$children = $element->child_nodes();
$child = $element->remove_child($children[0]);

echo "<PRE>";
$xmlfile = $dom->dump_mem(true);
echo htmlentities($xmlfile);
echo "</PRE>";
?>
```

See also DomNode_append_child().

DomNode->replace_child (unknown)

Replaces a child

object **DomNode->replace_child** (object oldnode, object newnode) \linebreak

This function replaces the child *oldnode* with the passed new node. If the new node is already a child it will not be added a second time. If the old node cannot be found the function returns false. If the replacement succeds the old node is returned.

See also DomNode_append_child()

DomNode->replace_node (unknown)

Replaces node

object **DomNode->replace_node** (object newnode) \linebreak

This function replaces an existing node with the passed new node. Before the replacement *newnode* is copied if it has a parent to make sure a node which is already in the document will not be inserted a second time. This behaviour enforces doing all modifications on the node before the replacement or to refetch the inserted node afterwards with functions like DomNode_first_child(), DomNode_child_nodes() etc..

See also DomNode_append_child()

DomNode->set_content (unknown)

Sets content of node

bool **DomNode->set_content** (void) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomNode->set_name (unknown)

Sets name of node

bool **DomNode->set_name** (void) \linebreak

Sets name of node.

See also DomNode_node_name().

DomNode->unlink_node (unknown)

Deletes node

object **DomNode->unlink_node** (void) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomProcessingInstruction->data (unknown)

Returns data of pi node

string **DomProcessingInstruction->data** (void) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

DomProcessingInstruction->target (unknown)

Returns target of pi node

string **DomProcessingInstruction->target** (void) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

domxml_new_doc (PHP 4 >= 4.2.1)

Creates new empty XML document

object **domxml_new_doc** (string version) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Creates a new dom document from scratch and returns it.

See also DomDocument_add_root()

domxml_open_file (PHP 4 >= 4.2.1)

Creates a DOM object from XML file

object **domxml_open_file** (string filename) \linebreak

The function parses the XML document in the file named *filename* and returns an object of class "Dom document", having the properties as listed above. The file is accessed read-only.

Példa 1. Opening a xml document from a file

```
<?php

if(!$dom = domxml_open_file("example.xml")) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
?>
```

See also domxml_open_mem(), domxml_new_doc().

domxml_open_mem (PHP 4 >= 4.2.1)

Creates a DOM object of an XML document

object **domxml_open_mem** (string str) \linebreak

The function parses the XML document in *str* and returns an object of class "Dom document", having the properties as listed above. This function, domxml_open_file() or domxml_new_doc() must be called before any other function calls.

Példa 1. Opening a xml document in a string

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
?>
```

See also domxml_open_file(), domxml_new_doc().

domxml_version (PHP 4 >= 4.1.0)

Get XML library version

string **domxml_version** (void) \linebreak

This function returns the version of the XML library version currently used.

domxml_xmltree (PHP 4 >= 4.2.1)

Creates a tree of PHP objects from an XML document

object **domxml_xmltree** (string str) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

The function parses the XML document in `str` and returns a tree PHP objects as the parsed document. This function is isolated from the other functions, which means you cannot access the tree with any of the other functions. Modifying it, for example by adding nodes, makes no sense since there is currently no way to dump it as an XML file. However this function may be valuable if you want to read a file and investigate the content.

xpath_eval_expression (PHP 4 >= 4.0.4)

Evaluates the XPath Location Path in the given string

array **xpath_eval_expression** (object xpath_context) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

See also xpath_eval()

xpath_eval (PHP 4 >= 4.0.4)

Evaluates the XPath Location Path in the given string

array **xpath_eval** (object xpath context) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

See also xpath_new_context()

xpath_new_context (PHP 4 >= 4.0.4)

Creates new xpath context

object **xpath_new_context** (object dom document) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

See also xpath_eval()

xptr_eval (PHP 4 >= 4.0.4)

Evaluate the XPtr Location Path in the given string

int **xptr_eval** ([object xpath_context, string eval_str]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xptr_new_context (PHP 4 >= 4.0.4)

Create new XPath Context

string **xptr_new_context** ([object doc_handle]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

XXVI. .NET functions

Bevezetés

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

dotnet_load (unknown)

Loads a DOTNET module

int **dotnet_load** (string assembly_name [, string datatype_name [, int codepage]]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

XXVII. Hibakezelő és naplózó függvények

Ebben a részben a hibák kezelésével és naplózásával kapcsolatos függvények találhatóak. Ezekkel lehetőséged van saját hibakezelő függvény létrehozására, vagy a hibák naplózási módjának megváltoztatására. Lehetőséged van megváltoztatni és kibővíteni a hibajelentést az igényeidnek megfelelően.

A hibanaplózó függvényekkel üzeneteket tudsz küldeni más gépeknek elektronikus levélben (SMS vagy pager átjárókkal hordozható eszközökre is), rendszer naplókba, stb. Ezekkel az igényeidnek megfelelő mértékben és részletességgel naplózhatod a web alkalmazásod vagy weboldalaid legfontosabb részeit.

A hibajelentő függvények lehetőséget adnak a hibajelentési szint és megjelenítési forma megváltoztatására, az egyszerű üzenetektől kezdve saját hibakezelő függvények meghívásáig.

error_log (PHP 3, PHP 4)

Hiabüzenet küldése

int **error_log** (string message [, int message_type [, string destination [, string extra_headers]]]) \linebreak

Ezzel a függvénnyel hibaüzenetet küldhetsz a kiszolgáló hiba naplójába, egy TCP portra, vagy egy állományba. Az első, *`message`* paraméterben a hibaüzenetet kell megadnod, a második, *`message_type`* paraméterben az üzenet célját:

Táblázat 1. error_log() napőló típusok

0	A *`message`* paraméterben megadott üzenet a PHP rendszer naplójába kerül, az operációs rendszer naplóját vagy egy állományt megcélozva, az error_log beállítás értékétől függően.
1	A *`message`* paraméterben megadott üzenetet elektronikus levélben a *`destination`* paraméterben megadott címre küldi. Ez az egyetlen típus, amelynél a negyedik *`extra_headers`* paraméter használható. Ez a típus a mail() által is használt belső függvényt használja.
2	A *`message`* paraméterben megadott üzenetet a PHP debug kapcsolatán keresztül küldi tovább. Ez a lehetőség csak akkor elérhető, ha a távoli debugger támogatás engedélyezett. Ebben az esetben a *`destination`* paraméter azt a hoszt nevet vagy IP címet - és opcionálisan port számot - adja meg, ami a debug információkat fogadni képes.
3	A *`message`* paraméterben megadott üzenetet a *`destination`* állomány végén hozzáfűzésre kerül.

Figyelem

A távoli debug funkviók TCP/IP kapcsolaton keresztül a PHP 3-as verziókban elérhetőek voltak. Ez a funkció *nem* használható a PHP 4-es verziókban.

Példa 1. error_log() példák

```
// Naplózzuk a rendszer naplóba a sikertelen adatbázis
// kapcsolódási kísérletet
if (!Ora_Logon ($usernev, $jelszo)) {
    error_log ("Az Oracle adatbázis nem elérhető!", 0);
}
```

```
// Figyelmeztetés az adminisztátornak, ha kifogyunk az izéből
if (!($ize = uj_ize_lefoglalasa()) {
    error_log ("Nagy gond van, végképp nincs több izé!", 1,
               "webmester@pelda.hu");
}

// Más naplózási típusok
error_log ("Általános gond van!", 2, "127.0.0.1:7000");
error_log ("Általános gond van!", 2, "loghost");
error_log ("Általános gond van!", 3, "/var/tmp/my-errors.log");
```

error_reporting (PHP 3, PHP 4)

Beállítja a hibajelentési szintet

int **error_reporting** ([int level]) \linebreak

Ez a függvény beállítja a PHP hibakezelési szintjét, és visszetér a korábbi értékkel. A hibakezelési szint vagy egy bitmaszk vagy egy hibajelentési konstans. A konstansok használata erősen javasolt, mivel a számok megváltozhatnak a későbbi verziókban.

Példa 1. Hibajelentési változások

```
error_reporting (55);   // A PHP 3-ban ez ugyanaz, mint E_ALL ^ E_NOTICE

/* PHP 4-ben viszont az 55 (E_ERROR | E_WARNING | E_PARSE |
E_CORE_ERROR | E_CORE_WARNING) jelentésű */

error_reporting (2039); // A PHP 4-ben ez ugyanaz, mint E_ALL ^ E_NOTICE

error_reporting (E_ALL ^ E_NOTICE); // Ez ugyanaz PHP 3 és 4 esetén is
```

Kövesd a hiba konstansok linkjeit bővebb információkért.

Táblázat 1. error_reporting() értékek

érték	konstans
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR

érték	konstans
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE

Példa 2. error_reporting() példák

```
error_reporting(0);
/* Minden hibajelentés kikapcsolása */

/* Az alábbi példákban először a régi szintakszist (PHP 2/3)
 * adtuk meg, utána pedig a javasolt újabb szintaxist (PHP 3/4)
 */

error_reporting (7);
error_reporting  (E_ERROR | E_WARNING | E_PARSE);
/* Ez a szint jó lehet egyszerű futási hibák kiírásához */

error_reporting  (15);
error_reporting (E_ERROR | E_WARNING | E_PARSE | E_NOTICE);
/*  Ez a szint programozásnál a nem inicializált (vagy elgépelt)
 *  változók megtalálásában segíthet
 */

error_reporting (63);
error_reporting (E_ALL);
/* Minden PHP hiba kiírása */
```

restore_error_handler (PHP 4 >= 4.0.1)

Visszaállítja a korábbi hibakezelő függvényt

void **restore_error_handler** (void) \linebreak

Ez a függvényt a set_error_handler()-el átállított hibakezelő függvény visszaállítására használható (ami lehet a beépített, vagy korábban beállított hibakezelő).

Lásd még error_reporting(), set_error_handler(), trigger_error() és user_error().

set_error_handler (PHP 4 >= 4.0.1)

Beállít egy felhasználói hibakezelő függvényt

string **set_error_handler** (string error_handler) \linebreak

A *`error_handler`* paraméterben megadott felhasználó által létrehozott hibakezelő függvényt állítja be a szkriptben felbukkanó hibák kezelésére. Visszatérési értéke a korábbi hibakazelő (ha volt ilyen), vagy `FALSE` hiba esetén. Ez a függvény használható futásidőben saját hibakezelő függvény definiálására, ha például a saját ideiglenes adatállományaid hiba esetén törlésre szorulnak, vagy ha a trigger_error() függvényt testreszabott hibaüzenetek küldésére szeretnéd használni.

A függvényt, amit megadsz két paramétert kell, hogy fogadjon, a hibakódot és a hibaüzenet karaktersorozatát. A PHP 4.0.2 verzió óta újabb három elhagyható paraméter is rendelkezésre áll: az állomány neve, ahol a hiba felbukkant, a sorszám, ahol a hiba megjelent, és az aktuális környezet (egy, a hiba létrejöttekor aktív szimbólum táblára mutató tömb).

A következő példa bemutatja, hogy hoyagn kell kezelni a belső kivételeket a felhasználói hibák keltésével, és egy felhasználó által definiált függvénnyel:

Példa 1. Hibakezelés a set_error_handler() és trigger_error() függvényekkel

```
<?php

// Rövidebb nevek a felhasználói hibákhoz - csak PHP 4-ben
define (FATAL, E_USER_ERROR);
define (ERROR, E_USER_WARNING);
define (WARNING, E_USER_NOTICE);

// A hibajelentési szint beállítása
error_reporting (FATAL | ERROR | WARNING);

// Hibakezelő függvény létrehozása
function hibaKezelo ($hszam, $hszoveg, $hfile, $hsor) {
  switch ($hszam) {
  case FATAL:
    echo "<b>FATÁLIS HIBA</b> [$errno] $errstr<br>\n";
    echo "  Fatális hiba a ".$errline.". sorban a ".$errfile;
    echo " állományban, PHP ".PHP_VERSION." (".PHP_OS.")<br>\n";
    echo "Kilépés...<br>\n";
    exit 1;
    break;
  case ERROR:
    echo "<b>HIBA</b> [$errno] $errstr<br>\n";
    break;
  case WARNING:
    echo "<b>FIGYELMEZTETÉS</b> [$errno] $errstr<br>\n";
    break;
    default:
    echo "Ismeretlen hibatípus: [$errno] $errstr<br>\n";
    break;
  }
}

// Függvény, ami teszteli a hibakezelést
function log_szorzas ($vektor, $mertek) {
  if (!is_numeric($mertek) || $mertek <= 0)
    trigger_error("log(x) x <= 0 esetén nem definiált (a megadott $mertek érték hibás)",
      FATAL);
  if (!is_array($vektor)) {
```

```
    trigger_error("Hibás vektor, értékek tömbjét kell megadni", ERROR);
    return null;
  }
  for ($i=0; $i<count($vektor); $i++) {
    if (!is_numeric($vektor[$i]))
      trigger_error("A vektor $i. eleme nem szám, nullával számolok",
        WARNING);
    $ideiglenes[$i] = log($mertek) * $vektor[$i];
  }
  return $ideiglenes;
}

// A felhasználói hibakezelő beállítása
$regi_hiba_kezelo = set_error_handler("hibaKezelo");

// Tömb nem szám értékkel
echo "'a' vektor\n";
$a = array(2,3,"izé",5.5,43.3,21.11);
print_r($a);

// Figyelmeztetés generálása, új tömb létrehozása
echo "----\n'b' vektor - figyelmeztetés (b = log(PI) * a)\n";
$b = log_szorzas($a, M_PI);
print_r($b);

// Ebből gond lesz, karaktersorozatot adunk át tömb helyett
echo "----\n'c' vektor - hiba\n";
$c = log_szorzas("ez nem tömb", 2.3);
var_dump($c);

// Ez kritikus hiba lesz, negatív szám logaritmusa nem értelmezett
echo "----\n'd' vektor - fatális hiba\n";
$d = log_szorzas($a, -2.5);

?>
```

Amikor futtatod a fenti példát, a következő kimetetet kapod:

```
'a' vektor
Array
(
    [0] => 2
    [1] => 3
    [2] => izé
    [3] => 5.5
    [4] => 43.3
    [5] => 21.11
)
----
'b' vektor - figyelmeztetés (b = log(PI) * a)
<b>FIGYELMEZTETÉS</b> [1024] A vektor 2. eleme nem szám, nullával számolok<br>
Array
(
    [0] => 2.2894597716988
```

```
    [1] => 3.4341896575482
    [2] => 0
    [3] => 6.2960143721717
    [4] => 49.566804057279
    [5] => 24.165247890281
)
----
'c' vektor - hiba
<b>HIBA</b> [512] Hibás vektor, értékek tömbjét kell megadni<br>
NULL
----
'd' vektor - fatális hiba
<b>FATÁLIS HIBA</b> [256] log(x) x <= 0 esetén nem definiált (a megadott -2.5 érték hi-
bás)<br>
  Fatális hiba a 36. sorban a trigger_error.php állományban, PHP 4.0.2 (Linux)<br>
Kilépés...<br>
```

Fontos megjegyezni, hogy a PHP belső hibakezelője teljesen figyelmen kívül marad. Az error_reporting() beállítások nem befolyásolják a felhasználói hibakezelő meghívását, minden hiba esetén lefut a hibakezelő az aktuális szinttől függetlenül. Ha ezt el szeretnéd kerülni, a hibakezelőben olvasd be az aktuális szintet (error_reporting), és írd meg úgy a függvényt, hogy annak megfelelően viselkedjen. Szintén fontos megjegyezni, hogy ez az érték 0 lesz, ha azt a parancsot, ami éppen meghívásra került a @ hibakezelő oprátorral adták meg.

Arról sem szabad elfeledkezned, hogy a Te felelősséged a szkript futás megszakítása, ha ez szükséges (például a exit() meghívásával). Ha a hibakezelő függvény visszatér, a szkript futása is folytatódni fog a következő paranccsal.

Lásd még: error_reporting(), restore_error_handler(), trigger_error() és user_error()!

trigger_error (PHP 4 >= 4.0.1)

Felhasználói szintű hibát/figyelmeztetést/megjegyzést jelez

void **trigger_error** (string error_msg [, int error_type]) \linebreak

Ez a függvény a belső vagy saját hibakezelő függvény számára küldött felhasználói szintű jelzés keltésére használható. A saját hibakezelő függvényt a set_error_handler() függvénnyel állíthatod be. A **trigger_error()** csak az E_USER hibacsaládal használható, alapbeállításban `E_USER_NOTICE` hibát jelez.

Ez a függvény hasznos, ha futás közben egy kivételt kell kezelned. Például:

```
if (assert ($nevezo == 0))
   trigger_error ("Nullával nem lehet osztani", E_USER_ERROR);
```

Megjegyzés: Lásd a set_error_handler() leírást, ahol egy bővebb példa található.

Lásd még error_reporting(), set_error_handler(), restore_error_handler() és user_error().

user_error (PHP 4)

Felhasználói szintű hibát/figyelmeztetést/megjegyzést jelez

void **user_error** (string error_msg [, int error_type]) \linebreak

Ez a függvény tulajdonképpen csak egy második név a trigger_error() függvény számára.

Lásd még error_reporting(), set_error_handler(), restore_error_handler() és trigger_error().

XXVIII. FrontBase Functions

Bevezetés

These functions allow you to access FrontBase database servers. More information about FrontBase can be found at http://www.frontbase.com/.

Documentation for FrontBase can be found at http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation.

Frontbase support has been added to PHP 4.0.6.

Követelmények

You must install the FrontBase database server or at least the fbsql client libraries to use this functions. You can get FrontBase from http://www.frontbase.com/.

Telepítés

In order to have these functions available, you must compile PHP with fbsql support by using the `--with-fbsql` option. If you use this option without specifying the path to fbsql, PHP will search for the fbsql client libraries in the default installation location for the platform. Users who installed FrontBase in a non standard directory should always specify the path to fbsql: `--with-fbsql=/path/to/fbsql`. This will force PHP to use the client libraries installed by FrontBase, avoiding any conflicts.

Futásidejű beállítások

Erőforrás típusok

Előre definiált állandók

Az itt listázott állandókat ez a kiterjesztés definiálja, és csak akkor elérhetőek, ha az adott

kiterjesztés be van fordítva a PHP-be, vagy dinamikusan betöltött.

`FBSQL_ASSOC` (integer)

`FBSQL_NUM` (integer)

`FBSQL_BOTH` (integer)

`FBSQL_LOCK_DEFERRED` (integer)

`FBSQL_LOCK_OPTIMISTIC` (integer)

`FBSQL_LOCK_PESSIMISTIC` (integer)

`FBSQL_ISO_READ_UNCOMMITTED` (integer)

`FBSQL_ISO_READ_COMMITTED` (integer)

`FBSQL_ISO_REPEATABLE_READ` (integer)

`FBSQL_ISO_SERIALIZABLE` (integer)

`FBSQL_ISO_VERSIONED` (integer)

`FBSQL_UNKNOWN` (integer)

`FBSQL_STOPPED` (integer)

`FBSQL_STARTING` (integer)

`FBSQL_RUNNING` (integer)

`FBSQL_STOPPING` (integer)

`FBSQL_NOEXEC` (integer)

`FBSQL_LOB_DIRECT` (integer)

`FBSQL_LOB_HANDLE` (integer)

fbsql_affected_rows (PHP 4 >= 4.0.6)

Get number of affected rows in previous FrontBase operation

int **fbsql_affected_rows** ([resource link_identifier]) \linebreak

fbsql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with *`link_identifier`*. If the link identifier isn't specified, the last link opened by fbsql_connect() is assumed.

> **Megjegyzés:** If you are using transactions, you need to call **fbsql_affected_rows()** after your INSERT, UPDATE, or DELETE query, not after the commit.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero.

> **Megjegyzés:** When using UPDATE, FrontBase will not update columns where the new value is the same as the old value. This creates the possibility that **fbsql_affected_rows()** may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

If the last query failed, this function will return -1.

See also: fbsql_num_rows().

fbsql_autocommit (PHP 4 >= 4.0.6)

Enable or disable autocommit

bool **fbsql_autocommit** (resource link_identifier [, bool OnOff]) \linebreak

fbsql_autocommit() returns the current autocommit status. if the optional OnOff parameter is given the auto commit status will be changed. With OnOff set to `TRUE` each statement will be committed automatically, if no errors was found. With OnOff set to `FALSE` the user must commit or rollback the transaction using either fbsql_commit() or fbsql_rollback().

See also: fbsql_commit() and fbsql_rollback()

fbsql_change_user (unknown)

Change logged in user of the active connection

resource **fbsql_change_user** (string user, string password [, string database [, resource link_identifier]]) \linebreak

fbsql_change_user() changes the logged in user of the current active connection, or the connection given by the optional parameter link_identifier. If a database is specified, this will default or current

database after the user has been changed. If the new user and password authorization fails, the current connected user stays active.

fbsql_close (PHP 4 >= 4.0.6)

Close FrontBase connection

boolean **fbsql_close** ([resource link_identifier]) \linebreak

Returns: `TRUE` on success, `FALSE` on error.

fbsql_close() closes the connection to the FrontBase server that's associated with the specified link identifier. If *`link_identifier`* isn't specified, the last opened link is used.

Using **fbsql_close()** isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

Példa 1. fbsql_close() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);
?>
```

See also: fbsql_connect() and fbsql_pconnect().

fbsql_commit (PHP 4 >= 4.0.6)

Commits a transaction to the database

bool **fbsql_commit** ([resource link_identifier]) \linebreak

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

fbsql_commit() ends the current transaction by writing all inserts, updates and deletes to the disk and unlocking all row and table locks held by the transaction. This command is only needed if autocommit is set to false.

See also: fbsql_autocommit() and fbsql_rollback()

fbsql_connect (PHP 4 >= 4.0.6)

Open a connection to a FrontBase Server

resource **fbsql_connect** ([string hostname [, string username [, string password]]]) \linebreak

Returns a positive FrontBase link identifier on success, or an error message on failure.

fbsql_connect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *`hostname`* = '`NULL`', *`username`* = '_SYSTEM' and *`password`* = empty password.

If a second call is made to **fbsql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling fbsql_close().

Példa 1. fbsql_connect() example

```
<?php

    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);

?>
```

See also fbsql_pconnect() and fbsql_close().

fbsql_create_blob (PHP 4 >= 4.2.0)

Create a BLOB

string **fbsql_create_blob** (string blob_data [, resource link_identifier]) \linebreak

Returns: A resource handle to the newly created blob.

fbsql_create_blob() creates a blob from blob_data. The returned resource handle can be used with insert and update commands to store the blob in the database.

Példa 1. fbsql_create_blob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $filename = "blobfile.bin";
    $fp = fopen($filename, "rb");
    $blobdata = fread($fp, filesize($filename));
    fclose($fp);

    $blobHandle = fbsql_create_blob($blobdata, $link);

    $sql = "INSERT INTO BLOB_TABLE (BLOB_COLUMN) VALUES ($blobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_clob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_clob (PHP 4 >= 4.2.0)

Create a CLOB

string **fbsql_create_clob** (string clob_data [, resource link_identifier]) \linebreak

Returns: A resource handle to the newly created CLOB.

fbsql_create_clob() creates a clob from clob_data. The returned resource handle can be used with insert and update commands to store the clob in the database.

Példa 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $filename = "clob_file.txt";
    $fp = fopen($filename, "rb");
    $clobdata = fread($fp, filesize($filename));
    fclose($fp);

    $clobHandle = fbsql_create_clob($clobdata, $link);

    $sql = "INSERT INTO CLOB_TABLE (CLOB_COLUMN) VALUES ($clobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_db (PHP 4 >= 4.0.6)

Create a FrontBase database

bool **fbsql_create_db** (string database name [, resource link_identifier]) \linebreak

fbsql_create_db() attempts to create a new database on the server associated with the specified link identifier.

Példa 1. fbsql_create_db() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    if (fbsql_create_db ("my_db")) {
```

```
        print("Database created successfully\n");
    } else {
        printf("Error creating database: %s\n", fbsql_error ());
    }
?>
```

See also: fbsql_drop_db().

fbsql_data_seek (PHP 4 >= 4.0.6)

Move internal result pointer

bool **fbsql_data_seek** (resource result_identifier, int row_number) \linebreak

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

fbsql_data_seek() moves the internal row pointer of the FrontBase result associated with the specified result identifier to point to the specified row number. The next call to fbsql_fetch_row() would return that row.

`Row_number` starts at 0.

Példa 1. fbsql_data_seek() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");

    fbsql_select_db ("samp_db")
        or die ("Could not select database");

    $query = "SELECT last_name, first_name FROM friends;";
    $result = fbsql_query ($query)
        or die ("Query failed");

    # fetch rows in reverse order

    for ($i = fbsql_num_rows ($result) - 1; $i >=0; $i--) {
        if (!fbsql_data_seek ($result, $i)) {
            printf ("Cannot seek to row %d\n", $i);
            continue;
        }

        if(!($row = fbsql_fetch_object ($result)))
            continue;

        printf("%s %s<BR>\n", $row->last_name, $row->first_name);
    }

    fbsql_free_result ($result);
?>
```

fbsql_database_password (PHP 4 >= 4.0.6)

Sets or retrieves the password for a FrontBase database

string **fbsql_database_password** (resource link_identifier [, string database_password]) \linebreak

Returns: The database password associated with the link identifier.

fbsql_database_password() sets and retrieves the database password used by the connection. if a database is protected by a database password, the user must call this function before calling fbsql_select_db(). if the second optional parameter is given the function sets the database password for the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if fbsql_connect() was called, and use it.

This function does not change the database password in the database nor can it be used to retrive the database password for a database.

Példa 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    fbsql_database_password($link, "secret db password");
    fbsql_select_db($database, $link);
?>
```

See also: fbsql_connect(), fbsql_pconnect() and fbsql_select_db().

fbsql_database (PHP 4 >= 4.0.6)

Get or set the database name used with a connection

string **fbsql_database** (resource link_identifier [, string database]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

fbsql_db_query (PHP 4 >= 4.0.6)

Send a FrontBase query

resource **fbsql_db_query** (string database, string query [, resource link_identifier]) \linebreak

Returns: A positive FrontBase result identifier to the query result, or FALSE on error.

fbsql_db_query() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the FrontBase server and if no such link is found it'll try to create one as if fbsql_connect() was called with no arguments

See also fbsql_connect().

fbsql_db_status (PHP 4 >= 4.1.0)

Get the status for a given database

int **fbsql_db_status** (string database_name [, resource link_identifier]) \linebreak

Returns: An integer value with the current status.

fbsql_db_status() requests the current status of the database specified by *database_name*. If the *link_identifier* is omitted the default link_identifier will be used.

The return value can be one of the following constants:

- FALSE - The exec handler for the host was invalid. This error will occur when the link_identifier connects directly to a database by using a port number. FBExec can be available on the server but no connection has been made for it.
- FBSQL_UNKNOWN - The Status is unknown.
- FBSQL_STOPPED - The database is not running. Use fbsql_start_db() to start the database.
- FBSQL_STARTING - The database is starting.
- FBSQL_RUNNING - The database is running and can be used to perform SQL operations.
- FBSQL_STOPPING - The database is stopping.
- FBSQL_NOEXEC - FBExec is not running on the server and it is not possible to get the status of the database.

See also: fbsql_start_db() and fbsql_stop_db().

fbsql_drop_db (PHP 4 >= 4.0.6)

Drop (delete) a FrontBase database

bool **fbsql_drop_db** (string database_name [, resource link_identifier]) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

fbsql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

fbsql_errno (PHP 4 >= 4.0.6)

Returns the numerical value of the error message from previous FrontBase operation

int **fbsql_errno** ([resource link_identifier]) \linebreak

Returns the error number from the last fbsql function, or `0` (zero) if no error occurred.

Errors coming back from the fbsql database backend don't issue warnings. Instead, use **fbsql_errno()** to retrieve the error code. Note that this function only returns the error code from the most recently executed fbsql function (not including fbsql_error() and **fbsql_errno()**), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

See also: fbsql_error() and fbsql_warnings().

fbsql_error (PHP 4 >= 4.0.6)

Returns the text of the error message from previous FrontBase operation

string **fbsql_error** ([resource link_identifier]) \linebreak

Returns the error text from the last fbsql function, or `''` (the empty string) if no error occurred.

Errors coming back from the fbsql database backend don't issue warnings. Instead, use **fbsql_error()** to retrieve the error text. Note that this function only returns the error text from the most recently executed fbsql function (not including **fbsql_error()** and fbsql_errno()), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
```

```
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

See also: fbsql_errno() and fbsql_warnings().

fbsql_fetch_array (PHP 4 >= 4.0.6)

Fetch a result row as an associative array, a numeric array, or both

array **fbsql_fetch_array** (resource result [, int result_type]) \linebreak

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_array() is an extended version of fbsql_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must the numeric index of the column or make an alias for the column.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

An important thing to note is that using **fbsql_fetch_array()** is NOT significantly slower than using fbsql_fetch_row(), while it provides a significant added value.

The optional second argument *result_type* in **fbsql_fetch_array()** is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

For further details, see also fbsql_fetch_row() and fbsql_fetch_assoc().

Példa 1. fbsql_fetch_array() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select user_id, fullname from table");
while ($row = fbsql_fetch_array ($result)) {
    echo "user_id: ".$row["user_id"]."<br>\n";
    echo "user_id: ".$row[0]."<br>\n";
    echo "fullname: ".$row["fullname"]."<br>\n";
    echo "fullname: ".$row[1]."<br>\n";
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_assoc (PHP 4 >= 4.0.6)

Fetch a result row as an associative array

array **fbsql_fetch_assoc** (resource result) \linebreak

Returns an associative array that corresponds to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_assoc() is equivalent to calling fbsql_fetch_array() with FBSQL_ASSOC for the optional second parameter. It only returns an associative array. This is the way fbsql_fetch_array() originally worked. If you need the numeric indices as well as the associative, use fbsql_fetch_array().

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use fbsql_fetch_array() and have it return the numeric indices as well.

An important thing to note is that using **fbsql_fetch_assoc()** is NOT significantly slower than using fbsql_fetch_row(), while it provides a significant added value.

For further details, see also fbsql_fetch_row() and fbsql_fetch_array().

Példa 1. fbsql_fetch_assoc() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select * from table");
while ($row = fbsql_fetch_assoc ($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_field (PHP 4 >= 4.0.6)

Get column information from a result and return as an object

object **fbsql_fetch_field** (resource result [, int field_offset]) \linebreak

Returns an object containing field information.

fbsql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **fbsql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- max_length - maximum length of the column
- not_null - 1 if the column cannot be NULL

- type - the type of the column

Példa 1. fbsql_fetch_field() example

```
<?php
fbsql_connect ($host, $user, $password)
    or die ("Could not connect");
$result = fbsql_db_query ("database", "select * from table")
    or die ("Query failed");
# get column metadata
$i = 0;
while ($i < fbsql_num_fields ($result)) {
    echo "Information for column $i:<BR>\n";
    $meta = fbsql_fetch_field ($result);
    if (!$meta) {
        echo "No information available<BR>\n";
    }
    echo "<PRE>
max_length:   $meta->max_length
name:         $meta->name
not_null:     $meta->not_null
table:        $meta->table
type:         $meta->type
</PRE>";
    $i++;
}
fbsql_free_result ($result);
?>
```

See also fbsql_field_seek().

fbsql_fetch_lengths (PHP 4 >= 4.0.6)

Get the length of each output in a result

array **fbsql_fetch_lengths** ([resource result]) \linebreak

Returns: An array that corresponds to the lengths of each field in the last row fetched by fbsql_fetch_row(), or FALSE on error.

fbsql_fetch_lengths() stores the lengths of each result column in the last row returned by fbsql_fetch_row(), fbsql_fetch_array() and fbsql_fetch_object() in an array, starting at offset 0.

See also: fbsql_fetch_row().

fbsql_fetch_object (PHP 4 >= 4.0.6)

Fetch a result row as an object

object **fbsql_fetch_object** (resource result [, int result_type]) \linebreak

Returns an object with properties that correspond to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_object() is similar to fbsql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

Speed-wise, the function is identical to fbsql_fetch_array(), and almost as quick as fbsql_fetch_row() (the difference is insignificant).

Példa 1. fbsql_fetch_object() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database", "select * from table");
while ($row = fbsql_fetch_object ($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
fbsql_free_result ($result);
?>
```

See also: fbsql_fetch_array() and fbsql_fetch_row().

fbsql_fetch_row (PHP 4 >= 4.0.6)

Get a result row as an enumerated array

array **fbsql_fetch_row** (resource result) \linebreak

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **fbsql_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

See also: fbsql_fetch_array(), fbsql_fetch_object(), fbsql_data_seek(), fbsql_fetch_lengths(), and fbsql_result().

fbsql_field_flags (PHP 4 >= 4.0.6)

Get the flags associated with the specified field in a result

string **fbsql_field_flags** (resource result, int field_offset) \linebreak

fbsql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using explode().

fbsql_field_len (PHP 4 >= 4.0.6)

Returns the length of the specified field

int **fbsql_field_len** (resource result, int field_offset) \linebreak

fbsql_field_len() returns the length of the specified field.

fbsql_field_name (PHP 4 >= 4.0.6)

Get the name of the specified field in a result

string **fbsql_field_name** (resource result, int field_index) \linebreak

fbsql_field_name() returns the name of the specified field index. *`result`* must be a valid result identifier and *`field_index`* is the numerical offset of the field.

> **Megjegyzés:** *`field_index`* starts at 0.
>
> e.g. The index of the third field would actually be 2, the index of the fourth field would be 3 and so on.

Példa 1. fbsql_field_name() example

```
// The users table consists of three fields:
//   user_id
//   username
//   password.

$res = fbsql_db_query("users", "select * from users", $link);

echo fbsql_field_name($res, 0) . "\n";
echo fbsql_field_name($res, 2);
```

The above example would produce the following output:

```
user_id
password
```

fbsql_field_seek (PHP 4 >= 4.0.6)

Set result pointer to a specified field offset

bool **fbsql_field_seek** (resource result, int field_offset) \linebreak

Seeks to the specified field offset. If the next call to fbsql_fetch_field() doesn't include a field offset, the field offset specified in **fbsql_field_seek()** will be returned.

See also: fbsql_fetch_field().

fbsql_field_table (PHP 4 >= 4.0.6)

Get name of the table the specified field is in

string **fbsql_field_table** (resource result, int field_offset) \linebreak

Returns the name of the table that the specified field is in.

fbsql_field_type (PHP 4 >= 4.0.6)

Get the type of the specified field in a result

string **fbsql_field_type** (resource result, int field_offset) \linebreak

fbsql_field_type() is similar to the fbsql_field_name() function. The arguments are identical, but the field type is returned instead. The field type will be one of "int", "real", "string", "blob", and others as detailed in the FrontBase documentation (http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation).

Példa 1. fbsql_field_type() example

```
<?php

fbsql_connect ("localhost", "_SYSTEM", "");
fbsql_select_db ("wisconsin");
$result = fbsql_query ("SELECT * FROM onek;");
$fields = fbsql_num_fields ($result);
$rows   = fbsql_num_rows ($result);
$i = 0;
```

```
$table = fbsql_field_table ($result, $i);
echo "Your '".$table."' table has ".$fields." fields and ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type  = fbsql_field_type  ($result, $i);
    $name  = fbsql_field_name  ($result, $i);
    $len   = fbsql_field_len   ($result, $i);
    $flags = fbsql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
fbsql_close();

?>
```

fbsql_free_result (PHP 4 >= 4.0.6)

Free result memory

bool **fbsql_free_result** (resource result) \linebreak

fbsql_free_result() will free all memory associated with the result identifier *`result`*.

fbsql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

fbsql_get_autostart_info (PHP 4 >= 4.1.0)

No description given yet

array **fbsql_get_autostart_info** ([resource link_identifier]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

fbsql_hostname (PHP 4 >= 4.0.6)

Get or set the host name used with a connection

string **fbsql_hostname** (resource link_identifier [, string host_name]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

fbsql_insert_id (PHP 4 >= 4.0.6)

Get the id generated from the previous INSERT operation

int **fbsql_insert_id** ([resource link_identifier]) \linebreak

fbsql_insert_id() returns the ID generated for an column defined as DEFAULT UNIQUE by the previous INSERT query using the given *`link_identifier`*. If *`link_identifier`* isn't specified, the last opened link is assumed.

fbsql_insert_id() returns 0 if the previous query does not generate an DEFAULT UNIQUE value. If you need to save the value for later, be sure to call fbsql_insert_id() immediately after the query that generates the value.

> **Megjegyzés:** The value of the FrontBase SQL function `LAST_INSERT_ID()` always contains the most recently generated DEFAULT UNIQUE value, and is not reset between queries.

fbsql_list_dbs (PHP 4 >= 4.0.6)

List databases available on a FrontBase server

resource **fbsql_list_dbs** ([resource link_identifier]) \linebreak

fbsql_list_dbs() will return a result pointer containing the databases available from the current fbsql daemon. Use the fbsql_tablename() function to traverse this result pointer.

Példa 1. fbsql_list_dbs() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');
$db_list = fbsql_list_dbs($link);

while ($row = fbsql_fetch_object($db_list)) {
    echo $row->Database . "\n";
}
```

The above example would produce the following output:

```
database1
database2
database3
...
```

Megjegyzés: The above code would just as easily work with fbsql_fetch_row() or other similar functions.

fbsql_list_fields (PHP 4 >= 4.0.6)

List FrontBase result fields

resource **fbsql_list_fields** (string database_name, string table_name [, resource link_identifier]) \linebreak

fbsql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with fbsql_field_flags(), fbsql_field_len(), fbsql_field_name(), and fbsql_field_type().

A result identifier is a positive integer. The function returns -1 if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@fbsql()` then this error string will also be printed out.

Példa 1. fbsql_list_fields() example

```
$link = fbsql_connect(’localhost’, ’myname’, ’secret’);

$fields = fbsql_list_fields("database1", "table1", $link);
$columns = fbsql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
    echo fbsql_field_name($fields, $i) . "\n";;
}
```

The above example would produce the following output:

```
field1
field2
field3
...
```

fbsql_list_tables (PHP 4 >= 4.0.6)

List tables in a FrontBase database

resource **fbsql_list_tables** (string database [, resource link_identifier]) \linebreak

fbsql_list_tables() takes a database name and returns a result pointer much like the fbsql_db_query() function. The fbsql_tablename() function should be used to extract the actual table names from the result pointer.

fbsql_next_result (PHP 4 >= 4.0.6)

Move the internal result pointer to the next result

bool **fbsql_next_result** (resource result_id) \linebreak

When sending more than one SQL statement to the server or executing a stored procedure with multiple results will cause the server to return multiple result sets. This function will test for additional results available form the server. If an additional result set exists it will free the existing result set and prepare to fetch the words from the new result set. The function will return `TRUE` if an additional result set was available or `FALSE` otherwise.

Példa 1. fbsql_next_result() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret");
    fbsql_select_db("MyDB", $link);
    $SQL = "Select * from table1; select * from table2;";
    $rs = fbsql_query($SQL, $link);
    do {
        while ($row = fbsql_fetch_row($rs)) {
        }
    } while (fbsql_next_result($rs));
    fbsql_free_result($rs);
    fbsql_close ($link);
?>
```

fbsql_num_fields (PHP 4 >= 4.0.6)

Get number of fields in result

int **fbsql_num_fields** (resource result) \linebreak

fbsql_num_fields() returns the number of fields in a result set.

See also: fbsql_db_query(), fbsql_query(), fbsql_fetch_field(), and fbsql_num_rows().

fbsql_num_rows (PHP 4 >= 4.0.6)

Get number of rows in result

int **fbsql_num_rows** (resource result) \linebreak

fbsql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements. To retrieve the number of rows returned from a INSERT, UPDATE or DELETE query, use fbsql_affected_rows().

Példa 1. fbsql_num_rows() example

```
<?php

$link = fbsql_connect("localhost", "username", "password");
fbsql_select_db("database", $link);

$result = fbsql_query("SELECT * FROM table1;", $link);
$num_rows = fbsql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

See also: fbsql_affected_rows(), fbsql_connect(), fbsql_select_db(), and fbsql_query().

fbsql_password (PHP 4 >= 4.0.6)

Get or set the user password used with a connection

string **fbsql_password** (resource link_identifier [, string password]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

fbsql_pconnect (PHP 4 >= 4.0.6)

Open a persistent connection to a FrontBase Server

resource **fbsql_pconnect** ([string hostname [, string username [, string password]]]) \linebreak

Returns: A positive FrontBase persistent link identifier on success, or FALSE on error.

fbsql_pconnect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *host* = 'localhost', *username* = "_SYSTEM" and *password* = empty password.

fbsql_pconnect() acts very much like fbsql_connect() with two major differences.

To set Frontbase server port number, use fbsql_select_db().

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use.

This type of links is therefore called 'persistent'.

fbsql_query (PHP 4 >= 4.0.6)

Send a FrontBase query

resource **fbsql_query** (string query [, resource link_identifier]) \linebreak

fbsql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if fbsql_connect() was called with no arguments, and use it.

> **Megjegyzés:** The query string shall always end with a semicolon.

fbsql_query() returns TRUE (non-zero) or FALSE to indicate whether or not the query succeeded. A return value of TRUE means that the query was legal and could be executed by the server. It does not indicate anything about the number of rows affected or returned. It is perfectly possible for a query to succeed but affect no rows or return no rows.

The following query is syntactically invalid, so **fbsql_query()** fails and returns FALSE:

Példa 1. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

The following query is semantically invalid if `my_col` is not a column in the table `my_tbl`, so **fbsql_query()** fails and returns `FALSE`:

Példa 2. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

fbsql_query() will also fail and return `FALSE` if you don't have permission to access the table(s) referenced by the query.

Assuming the query succeeds, you can call fbsql_num_rows() to find out how many rows were returned for a SELECT statement or fbsql_affected_rows() to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

For SELECT statements, **fbsql_query()** returns a new result identifier that you can pass to fbsql_result(). When you are done with the result set, you can free the resources associated with it by calling fbsql_free_result(). Although, the memory will automatically be freed at the end of the script's execution.

See also: fbsql_affected_rows(), fbsql_db_query(), fbsql_free_result(), fbsql_result(), fbsql_select_db(), and fbsql_connect().

fbsql_read_blob (PHP 4 >= 4.2.0)

Read a BLOB from the database

string **fbsql_read_blob** (string blob_handle [, resource link_identifier]) \linebreak

Returns: A string containing the BLOB specified by blob_handle.

fbsql_read_blob() reads BLOB data from the database. If a select statement contains BLOB and/or BLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with fbsql_set_lob_mode() so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_blob()** to get the actual BLOB data from the database.

Példa 1. fbsql_read_blob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $sql = "SELECT BLOB_COLUMN FROM BLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the blob data for teh first row
```

```
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the BLOB data in the first row
    $blob_data = fbsql_read_blob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: fbsql_create_blob(), **fbsql_read_blob()**, fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_read_clob (PHP 4 >= 4.2.0)

Read a CLOB from the database

string **fbsql_read_clob** (string clob_handle [, resource link_identifier]) \linebreak

Returns: A string containing the CLOB specified by clob_handle.

fbsql_read_clob() reads CLOB data from the database. If a select statement contains BLOB and/or CLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with fbsql_set_lob_mode() so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_clob()** to get the actual CLOB data from the database.

Példa 1. fbsql_read_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $sql = "SELECT CLOB_COLUMN FROM CLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the clob data for teh first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the CLOB data in the first row
    $clob_data = fbsql_read_clob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), **fbsql_read_clob()**, and fbsql_set_lob_mode().

fbsql_result (PHP 4 >= 4.0.6)

Get result data

mixed **fbsql_result** (resource result, int row [, mixed field]) \linebreak

fbsql_result() returns the contents of one cell from a FrontBase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tabledname.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **fbsql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls to **fbsql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: fbsql_fetch_row(), fbsql_fetch_array(), and fbsql_fetch_object().

fbsql_rollback (PHP 4 >= 4.0.6)

Rollback a transaction to the database

bool **fbsql_rollback** ([resource link_identifier]) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

fbsql_rollback() ends the current transaction by rolling back all statements issued since last commit. This command is only needed if autocommit is set to false.

See also: fbsql_autocommit() and fbsql_commit()

fbsql_select_db (PHP 4 >= 4.0.6)

Select a FrontBase database

bool **fbsql_select_db** (string database_name [, resource link_identifier]) \linebreak

Returns: TRUE on success, FALSE on error.

fbsql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if fbsql_connect() was called, and use it.

The client contacts FBExec to obtain the port number to use for the connection to the database. If the database name is a number the system will use that as a port number and it will not ask FBExec for the port number. The FrontBase server can be stared as FRontBase -FBExec=No -port=<port number> <database name>.

Every subsequent call to fbsql_query() will be made on the active database.

if the database is protected with a database password, the user must call fbsql_database_password() before selecting the database.

See also: fbsql_connect(), fbsql_pconnect(), fbsql_database_password() and fbsql_query().

fbsql_set_lob_mode (PHP 4 >= 4.2.0)

Set the LOB retrieve mode for a FrontBase result set

bool **fbsql_set_lob_mode** (resource result, string database_name) \linebreak

Returns: `TRUE` on success, `FALSE` on error.

fbsql_set_lob_mode() sets the mode for retrieving LOB data from the database. When BLOB and CLOB data is stored in FrontBase it can be stored direct or indirect. Direct stored LOB data will always be fetched no matter the setting of the lob mode. If the LOB data is less than 512 bytes it will always be stored directly.

- FBSQL_LOB_DIRECT - LOB data is retrieved directly. When data is fetched from the database with fbsql_fetch_row(), and other fetch functions, all CLOB and BLOB columns will be returned as ordinary columns. This is the default value on a new FrontBase result.
- FBSQL_LOB_HANDLE - LOB data is retrieved as handles to the data. When data is fetched from the database with **fbsql_fetch_row ()**, and other fetch functions, LOB data will be returned as a handle to the data if the data is stored indirect or the data if it is stored direct. If a handle is returned it will be a 27 byte string formatted as "@’00000000000000000000000’".

See also: fbsql_create_blob(), fbsql_create_clob(), fbsql_read_blob(), and fbsql_read_clob().

fbsql_set_transaction (PHP 4 >= 4.2.0)

Set the transaction locking and isolation

void **fbsql_set_transaction** (resource link_identifier, int Locking, int Isolation) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

fbsql_start_db (PHP 4 >= 4.0.6)

Start a database on local or remote server

bool **fbsql_start_db** (string database_name [, resource link_identifier]) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

fbsql_start_db()

See also: fbsql_db_status() and fbsql_stop_db().

fbsql_stop_db (PHP 4 >= 4.0.6)

Stop a database on local or remote server

bool **fbsql_stop_db** (string database_name [, resource link_identifier]) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

fbsql_stop_db()

See also: fbsql_db_status() and fbsql_start_db().

fbsql_tablename (PHP 4 >= 4.2.0)

Get table name of field

string **fbsql_tablename** (resource result, int i) \linebreak

fbsql_tablename() takes a result pointer returned by the fbsql_list_tables() function as well as an integer index and returns the name of a table. The fbsql_num_rows() function may be used to determine the number of tables in the result pointer.

Példa 1. fbsql_tablename() example

```
<?php
fbsql_connect ("localhost", "_SYSTEM", "");
$result = fbsql_list_tables ("wisconsin");
$i = 0;
while ($i < fbsql_num_rows ($result)) {
    $tb_names[$i] = fbsql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

fbsql_username (PHP 4 >= 4.0.6)

Get or set the host user used with a connection

string **fbsql_username** (resource link_identifier [, string username]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

fbsql_warnings (PHP 4 >= 4.0.6)

Enable or disable FrontBase warnings

bool **fbsql_warnings** ([bool OnOff]) \linebreak

Returns TRUE if warnings is turned on otherwise FALSE.

fbsql_warnings() enables or disables FrontBase warnings.

XXIX. filePro függvények

Ezek a függvények lehetőséget adnak a filePro adatbázisok olvasásására.

A filePro az fP Technologies, Inc. bejegyzett védjegye. További információk a filePro-ról a http://www.fptech.com/ címen találhatóak.

filepro_fieldcount (PHP 3, PHP 4)

Visszaadja a mezők számát

int **filepro_fieldcount** (void) \linebreak

A megnyitott filePro adatbázis mezőinek (oszlopainak) számával tér vissza.

Lásd még a filepro() függvényt.

filepro_fieldname (PHP 3, PHP 4)

Visszaadja a mező nevét

string **filepro_fieldname** (int field_number) \linebreak

Visszaadja a *`field_number`* számú mező nevét.

filepro_fieldtype (PHP 3, PHP 4)

Visszaadja a mező típusát

string **filepro_fieldtype** (int field_number) \linebreak

Visszaadja a *`field_number`* számú mező típusát.

filepro_fieldwidth (PHP 3, PHP 4)

Visszaadja a mező hosszát

int **filepro_fieldwidth** (int field_number) \linebreak

Visszaadja a *`field_number`* számú mező hosszát.

filepro_retrieve (PHP 3, PHP 4)

Adatok lekérése a filePro adatbázisból

string **filepro_retrieve** (int row_number, int field_number) \linebreak

Visszaadja az adatbázis megadott elérésén található adatot.

filepro_rowcount (PHP 3, PHP 4)

Visszaadja a sorok számát

int **filepro_rowcount** (void) \linebreak

Visszaadja a megnyitott filePro adatbázis sorainak számát.

Lásd még a filepro() függvényt.

filepro (PHP 3, PHP 4)

Beolvassa, és ellenőrzi a map fájlt

bool **filepro** (string directory) \linebreak

Beolvassa, és ellenőrzi a map fájlt, ami a mezőszámot és információkat tartalmazza.

Nem történik kizárólagos lefoglalás (locking), ezért nem szabad a PHP által megnyitott filePro adatbázisokat módosítani.

XXX. Filesystem functions

Bevezetés

Követelmények

Az itt leírt függvények a standard modulban találhatóak, ami mindig rendelkezésre áll.

Telepítés

Semmilyen telepítés nem szükséges ezen függvények használatához, a PHP alapelemei.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

Lásd még

For related functions, see also the Directory and Program Execution sections.

basename (PHP 3, PHP 4)

Returns filename component of path

string **basename** (string path [, string suffix]) \linebreak

Given a string containing a path to a file, this function will return the base name of the file. If the filename ends in *`suffix`* this will also be cut off.

On Windows, both slash (`/`) and backslash (`\`) are used as path separator character. In other environments, it is the forward slash (`/`).

Példa 1. basename() example

```
$path = "/home/httpd/html/index.php";
$file = basename ($path);        // $file is set to "index.php"
$file = basename ($path,".php"); // $file is set to "index"
```

Megjegyzés: The *`suffix`* parameter was added in PHP 4.1.0.

See also: dirname()

chgrp (PHP 3, PHP 4)

Changes file group

int **chgrp** (string filename, mixed group) \linebreak

Attempts to change the group of the file *`filename`* to *`group`* (specified by name or number). Only the superuser may change the group of a file arbitrarily; other users may change the group of a file to any group of which that user is a member.

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

See also chown() and chmod().

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

chmod (PHP 3, PHP 4)

Changes file mode

int **chmod** (string filename, int mode) \linebreak

Attempts to change the mode of the file specified by *filename* to that given in *mode*.

Note that *mode* is not automatically assumed to be an octal value, so strings (such as "g+w") will not work properly. To ensure the expected operation, you need to prefix *mode* with a zero (0):

```
chmod ("/somedir/somefile", 755);   // decimal; probably incorrect
chmod ("/somedir/somefile", "u+rwx,go+rx"); // string; incorrect
chmod ("/somedir/somefile", 0755);  // octal; correct value of mode
```

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

See also chown() and chgrp().

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

chown (PHP 3, PHP 4)

Changes file owner

int **chown** (string filename, mixed user) \linebreak

Attempts to change the owner of the file *filename* to user *user* (specified by name or number). Only the superuser may change the owner of a file.

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

See also **chown()** and chmod().

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

clearstatcache (PHP 3, PHP 4)

Clears file stat cache

void **clearstatcache** (void) \linebreak

Invoking the stat or lstat system call on most systems is quite expensive. Therefore, the result of the last call to any of the status functions (listed below) is stored for use on the next such call using the same filename. If you wish to force a new status check, for instance if the file is being checked many times and may change or disappear, use this function to clear the results of the last call from memory.

This value is only cached for the lifetime of a single request.

Affected functions include stat(), lstat(), file_exists(), is_writable(), is_readable(), is_executable(), is_file(), is_dir(), is_link(), filectime(), fileatime(), filemtime(), fileinode(), filegroup(), fileowner(), filesize(), filetype(), and fileperms().

copy (PHP 3, PHP 4)

Copies file

int **copy** (string source, string dest) \linebreak

Makes a copy of a file. Returns TRUE if the copy succeeded, FALSE otherwise.

Példa 1. copy() example

```
if (!copy($file, $file.'.bak')) {
    print ("failed to copy $file...<br>\n");
}
```

Megjegyzés: As of PHP 4.3.0, both *source* and *dest* may be URLs if the "fopen wrappers" have been enabled. See fopen() for more details.

Figyelem

If the destination file already exists, it will be overwritten.

See also move_uploaded_file(), rename(), and the section of the manual about handling file uploads.

delete (unknown)

A dummy manual entry

void **delete** (string file) \linebreak

This is a dummy manual entry to satisfy those people who are looking for unlink() or unset() in the wrong place.

See also: unlink() to delete files, unset() to delete variables.

dirname (PHP 3, PHP 4)

Returns directory name component of path

string **dirname** (string path) \linebreak

Given a string containing a path to a file, this function will return the name of the directory.

On Windows, both slash (/) and backslash (\) are used as path separator character. In other environments, it is the forward slash (/).

Példa 1. dirname() example

```
$path = "/etc/passwd";
$file = dirname ($path); // $file is set to "/etc"
```

Megjegyzés: In PHP 4.0.3, **dirname()** was fixed to be POSIX-compliant. Essentially, this means that if there are no slashes in *path* , a dot ('.') is returned, indicating the current directory. Otherwise, the returned string is *path* with any trailing /component removed. Note that this means that you will often get a slash or a dot back from **dirname()** in situations where the older functionality would have given you the empty string.

See also: basename(), pathinfo(), and realpath().

disk_free_space (PHP 4 >= 4.1.0)

Returns available space in directory

float **disk_free_space** (string directory) \linebreak

Given a string containing a directory, this function will return the number of bytes available on the corresponding filesystem or disk partition.

Példa 1. disk_free_space() example

```
$df = disk_free_space("/"); // $df contains the number of bytes
                            // available on "/"
```

disk_total_space (PHP 4 >= 4.1.0)

Returns the total size of a directory

float **disk_total_space** (string directory) \linebreak

Given a string containing a directory, this function will return the total number of bytes on the corresponding filesystem or disk partition.

Példa 1. disk_total_space() example

```
$df = disk_total_space("/"); // $df contains the total number of
                             // bytes available on "/"
```

diskfreespace (PHP 3>= 3.0.7, PHP 4)

Alias of disk_free_space()

float **diskfreespace** (string directory) \linebreak

This is a deprecated alias of disk_free_space(). Use that function instead.

fclose (PHP 3, PHP 4)

Closes an open file pointer

bool **fclose** (int fp) \linebreak

The file pointed to by *fp* is closed.

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

The file pointer must be valid, and must point to a file successfully opened by fopen() or fsockopen().

feof (PHP 3, PHP 4)

Tests for end-of-file on a file pointer

int **feof** (int fp) \linebreak

Returns TRUE if the file pointer is at EOF or an error occurs; otherwise returns FALSE.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen().

fflush (PHP 4 >= 4.0.1)

Flushes the output to a file

int **fflush** (int fp) \linebreak

This function forces a write of all buffered output to the to the resource pointed to by the file handle *`fp`*. Returns `TRUE` if successful, `FALSE` otherwise.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen().

fgetc (PHP 3, PHP 4)

Gets character from file pointer

string **fgetc** (int fp) \linebreak

Returns a string containing a single character read from the file pointed to by fp. Returns `FALSE` on EOF.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen().

See also fread(), fopen(), popen(), fsockopen(), and fgets().

fgetcsv (PHP 3>= 3.0.8, PHP 4)

Gets line from file pointer and parse for CSV fields

array **fgetcsv** (int fp, int length [, string delimiter [, string enclosure]]) \linebreak

Similar to fgets() except that **fgetcsv()** parses the line it reads for fields in CSV format and returns an array containing the fields read. The field delimiter is a comma, unless you specify another delimiter with the optional third parameter. The enclosure character is double quote, unless it is specified. *`Delimiter`* and *`enclosure`* cannot be null and only first character is used when they are specified.

`Fp` must be a valid file pointer to a file successfully opened by fopen(), popen(), or fsockopen()

`Length` must be greater than the longest line to be found in the CSV file (allowing for trailing line-end characters).

fgetcsv() returns `FALSE` on error, including end of file.

> **Megjegyzés:** A blank line in a CSV file will be returned as an array comprising a single `NULL` field, and will not be treated as an error.
>
> *`enclosure`* is added from PHP 4.3.0.

Példa 1. fgetcsv() example - Read and print entire contents of a CSV file

```
$row = 1;
$fp = fopen ("test.csv","r");
while ($data = fgetcsv ($fp, 1000, ",")) {
    $num = count ($data);
```

```
    print "<p> $num fields in line $row: <br>";
    $row++;
    for ($c=0; $c < $num; $c++) {
        print $data[$c] . "<br>";
    }
}
fclose ($fp);
```

fgets (PHP 3, PHP 4)

Gets line from file pointer

string **fgets** (int fp [, int length]) \linebreak

Returns a string of up to length - 1 bytes read from the file pointed to by fp. Reading ends when length - 1 bytes have been read, on a newline (which is included in the return value), or on EOF (whichever comes first). If no length is specified, the length defaults to 1k, or 1024 bytes.

If an error occurs, returns `FALSE`.

Common Pitfalls:

People used to the 'C' semantics of **fgets()** should note the difference in how `EOF` is returned.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen().

A simple example follows:

Példa 1. Reading a file line by line

```
$fd = fopen ("/tmp/inputfile.txt", "r");
while (!feof ($fd)) {
    $buffer = fgets($fd, 4096);
    echo $buffer;
}
fclose ($fd);
```

Megjegyzés: The *`length`* parameter became optional in PHP 4.2.0

See also fread(), fopen(), popen(), fgetc(), fsockopen(), and socket_set_timeout().

fgetss (PHP 3, PHP 4)

Gets line from file pointer and strip HTML tags

string **fgetss** (int fp, int length [, string allowable_tags]) \linebreak

Identical to fgets(), except that fgetss attempts to strip any HTML and PHP tags from the text it reads.

You can use the optional third parameter to specify tags which should not be stripped.

Megjegyzés: *allowable_tags* was added in PHP 3.0.13, PHP 4.0.0.

See also fgets(), fopen(), fsockopen(), popen(), and strip_tags().

file_exists (PHP 3, PHP 4)

Checks whether a file exists

bool **file_exists** (string filename) \linebreak

Returns TRUE if the file specified by *filename* exists; FALSE otherwise.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

The results of this function are cached. See clearstatcache() for more details.

On windows use "//computername/share/filename" or "\\\\computername\share\filename" to check files on network shares.

file_get_contents (PHP 4 CVS only)

Reads entire file into a string

string **file_get_contents** (string filename [, int use_include_path]) \linebreak

Identical to readfile(), except that **file_get_contents()** returns the file in a string.

Megjegyzés: This function was introduced in PHP 4.3.0.

Megjegyzés: Ez a függvény "binary-safe", azaz helyesen kezeli a 127-es ASCII kód feletti karaktereket is.

Tipp: URL-t is megadhatsz az állomány név paraméter helyén ha az "fopen wrappers" szolgáltatás be van kapcsolva. Bővebb információk az fopen() leírásánál olvashatóak.

See also: fgets(), file(), fread(), include(), and readfile().

file_get_wrapper_data (PHP 4 CVS only)

Retrieves header/meta data from "wrapped" file pointers

mixed **file_get_wrapper_data** (int fp) \linebreak

This function returns header or meta data from files opened with fopen(). This is useful to return the response headers for HTTP connections, or some other statistics for other resources.

The format of the returned data is deliberately undocumented at this time, and depends on which wrapper(s) were used to open the file.

Megjegyzés: This function was introduced in PHP 4.3.0.

file_register_wrapper (PHP 4 CVS only)

Register a URL wrapper implemented as a PHP class

boolean **file_register_wrapper** (string protocol, string classname) \linebreak

This function is currently only documented by the example below:

Példa 1. Implementing a base64 encoding protocol

```
class Base64EncodingStream {
    var $fp = null;

    function stream_open($path, $mode, $options, &$opened_path)
    {
        $this->fp = fopen($path, $mode);
        return is_resource($this->fp);
    }
    function stream_close()
    {
        fclose($this->fp);
    }
    function stream_read($count)
    {
        return false; // We only allow writing
    }
    function stream_write($data)
    {
        return fwrite($this->fp, base64_encode($data));
    }
    function stream_flush()
```

```
    {
        fflush($this->fp);
        return true;
    }
    function stream_seek($offset, $whence)
    {
        return false;
    }
    function stream_gets()
    {
        return false;
    }
    function stream_tell()
    {
        return false;
    }
    function stream_eof()
    {
        return false;
    }
}
file_register_wrapper("base64", "Base64EncodingStream")
    or die("Failed to register protocol");

copy("/tmp/inputfile.txt", "base64:///tmp/outputfile.txt");
readfile("/tmp/outputfile");
```

file_register_wrapper() will return false if the *`protocol`* already has a handler, or if "fopen wrappers" are disabled.

> **Megjegyzés:** This function was introduced in PHP 4.3.0.

file (PHP 3, PHP 4)

Reads entire file into an array

array **file** (string filename [, int use_include_path]) \linebreak

Identical to readfile(), except that **file()** returns the file in an array. Each element of the array corresponds to a line in the file, with the newline still attached.

> **Megjegyzés:** Each line in the resulting array will include the line ending, so you still need to use trim() if you do not want the line ending present.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

```
<?php
// get a web page into an array and print it out
$fcontents = file ('http://www.example.com/');
while (list ($line_num, $line) = each ($fcontents)) {
    echo "<b>Line $line_num:</b>; ", htmlspecialchars ($line), "<br>\n";
}

// get a web page into a string
$fcontents = implode ('', file ('http://www.example.com/'));
?>
```

Megjegyzés: As of PHP 4.3.0 you can use file_get_contents() to return the contents of a file as a string in a binary safe manner.

Figyelem

Ez a függvény (még) nem "binary safe", azaz nem helyesen kezeli a 127-es ASCII kód feletti karaktereket!

Tipp: URL-t is megadhatsz az állomány név paraméter helyén ha az "fopen wrappers" szolgáltatás be van kapcsolva. Bővebb információk az fopen() leírásánál olvashatóak.

See also readfile(), fopen(), fsockopen(), and popen().

fileatime (PHP 3, PHP 4)

Gets last access time of file

int **fileatime** (string filename) \linebreak

Returns the time the file was last accessed, or FALSE in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See clearstatcache() for more details.

Note: The atime of a file is supposed to change whenever the data blocks of a file are being read. This can be costly performancewise when an application regularly accesses a very large number of files or directories. Some Unix filesystems can be mounted with atime updates disabled to increase the performance of such applications; USENET news spools are a common example. On such filesystems this function will be useless.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

filectime (PHP 3, PHP 4)

Gets inode change time of file

int **filectime** (string filename) \linebreak

Returns the time the file was last changed, or FALSE in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See clearstatcache() for more details.

Note: In most Unix filesystems, a file is considered changed when its inode data is changed; that is, when the permissions, owner, group, or other metadata from the inode is updated. See also filemtime() (which is what you want to use when you want to create "Last Modified" footers on web pages) and fileatime().

Note also that in some Unix texts the ctime of a file is referred to as being the creation time of the file. This is wrong. There is no creation time for Unix files in most Unix filesystems.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

filegroup (PHP 3, PHP 4)

Gets file group

int **filegroup** (string filename) \linebreak

Returns the group ID of the owner of the file, or FALSE in case of an error. The group ID is returned in numerical format, use posix_getgrgid() to resolve it to a group name.

The results of this function are cached. See clearstatcache() for more details.

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

Megjegyzés: This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

fileinode (PHP 3, PHP 4)

Gets file inode

int **fileinode** (string filename) \linebreak

Returns the inode number of the file, or FALSE in case of an error.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

filemtime (PHP 3, PHP 4)

Gets file modification time

int **filemtime** (string filename) \linebreak

Returns the time the file was last modified, or FALSE in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

Note: This function returns the time when the data blocks of a file were being written to, that is, the time when the content of the file was changed. Use date() on the result of this function to get a printable modification date for use in page footers.

See also filectime() and touch().

fileowner (PHP 3, PHP 4)

Gets file owner

int **fileowner** (string filename) \linebreak

Returns the user ID of the owner of the file, or FALSE in case of an error. The user ID is returned in numerical format, use posix_getpwuid() to resolve it to a username.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

fileperms (PHP 3, PHP 4)

Gets file permissions

int **fileperms** (string filename) \linebreak

Returns the permissions on the file, or FALSE in case of an error.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

The results of this function are cached. See clearstatcache() for more details.

filesize (PHP 3, PHP 4)

Gets file size

int **filesize** (string filename) \linebreak

Returns the size of the file in bytes, or FALSE in case of an error.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

filetype (PHP 3, PHP 4)

Gets file type

string **filetype** (string filename) \linebreak

Returns the type of the file. Possible values are fifo, char, dir, block, link, file, and unknown.

Returns FALSE if an error occurs.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also: is_dir(), is_file(), is_link(), file_exists(), and stat().

flock (PHP 3>= 3.0.7, PHP 4)

Portable advisory file locking

bool **flock** (int fp, int operation [, int &wouldblock]) \linebreak

PHP supports a portable way of locking complete files in an advisory way (which means all accessing programs have to use the same way of locking or it will not work).

flock() operates on *fp* which must be an open file pointer. *operation* is one of the following values:

- To acquire a shared lock (reader), set *operation* to LOCK_SH (set to 1 prior to PHP 4.0.1).
- To acquire an exclusive lock (writer), set *operation* to LOCK_EX (set to 2 prior to PHP 4.0.1).
- To release a lock (shared or exclusive), set *operation* to LOCK_UN (set to 3 prior to PHP 4.0.1).
- If you don't want **flock()** to block while locking, add LOCK_NB (4 prior to PHP 4.0.1) to *operation*.

flock() allows you to perform a simple reader/writer model which can be used on virtually every platform (including most Unix derivatives and even Windows). The optional third argument is set to TRUE if the lock would block (EWOULDBLOCK errno condition)

flock() returns TRUE on success and FALSE on error (e.g. when a lock could not be acquired).

> **Megjegyzés:** Because **flock()** requires a file pointer, you may have to use a special lock file to protect access to a file that you intend to truncate by opening it in write mode (with a "w" or "w+" argument to fopen()).

> **Figyelem**
>
> **flock()** will not work on NFS and many other networked file systems. Check your operating system documentation for more details.
>
> On some operating systems **flock()** is implemented at the process level. When using a multithreaded server API like ISAPI you may not be able to rely on **flock()** to protect files against other PHP scripts running in parallel threads of the same server instance!
>
> **flock()** is not supported on antiquated filesystems like FAT and its derivates and will therefore always return FALSE under this environments (this is especially true for Windows 98 users).

fopen (PHP 3, PHP 4)

Opens file or URL

int **fopen** (string filename, string mode [, int use_include_path]) \linebreak

If *filename* begins with "http://" (not case sensitive), an HTTP 1.0 connection is opened to the specified server, the page is requested using the HTTP GET method, and a file pointer is returned to the beginning of the body of the response. A 'Host:' header is sent with the request in order to handle name-based virtual hosts.

As of PHP 4.3.0 (not yet released), if you have compiled in support for OpenSSL, you may use "https://" to open an HTTP connection over SSL.

Note that the file pointer allows you to retrieve only the *body* of the response; to retrieve the HTTP response header you need to be using PHP 4.0.5 or later; The headers will be stored in the $http_response_header variable. As of PHP 4.3.0 (not yet released), the header information can be retrieved using the **file_get_wrapper_data()**.

HTTP connections are read-only; you cannot write data or copy files to an HTTP resource.

Versions prior to PHP 4.0.5 do not handle HTTP redirects. Because of this, directories must include trailing slashes.

If *filename* begins with "ftp://" (not case sensitive), an ftp connection to the specified server is opened and a pointer to the requested file is returned. If the server does not support passive mode ftp, this will fail. You can open files for either reading or writing via ftp (but not both simultaneously). If the remote file already exists on the ftp server and you attempt to open it for writing, this will fail. If you need to update existing files over ftp, use ftp_connect().

If *filename* is one of "php://stdin", "php://stdout", or "php://stderr", the corresponding stdio stream will be opened. (This was introduced in PHP 3.0.13; in earlier versions, a filename such as "/dev/stdin" or "/dev/fd/0" must be used to access the stdio streams.)

If *filename* begins with anything else, the file will be opened from the filesystem, and a file pointer to the file opened is returned.

If the open fails, the function returns FALSE.

mode may be any of the following:

- 'r' - Open for reading only; place the file pointer at the beginning of the file.
- 'r+' - Open for reading and writing; place the file pointer at the beginning of the file.
- 'w' - Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- 'w+' - Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- 'a' - Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
- 'a+' - Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.

> **Megjegyzés:** The *mode* may contain the letter 'b'. This is useful only on systems which differentiate between binary and text files (i.e. Windows. It's useless on Unix). If not needed, this will be ignored.

You can use the optional third parameter and set it to "1", if you want to search for the file in the include_path, too.

Példa 1. fopen() example

```
$fp = fopen ("/home/rasmus/file.txt", "r");
$fp = fopen ("/home/rasmus/file.gif", "wb");
$fp = fopen ("http://www.example.com/", "r");
$fp = fopen ("ftp://user:password@example.com/", "w");
```

If you are experiencing problems with reading and writing to files and you're using the server module version of PHP, remember to make sure that the files and directories you're using are accessible to the server process.

On the Windows platform, be careful to escape any backslashes used in the path to the file, or use forward slashes.

```
$fp = fopen ("c:\\data\\info.txt", "r");
```

See also fclose(), fsockopen(), socket_set_timeout(), and popen().

fpassthru (PHP 3, PHP 4)

Output all remaining data on a file pointer

int **fpassthru** (int fp) \linebreak

Reads to EOF on the given file pointer from the current position and writes the results to standard output.

If an error occurs, **fpassthru()** returns FALSE.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen(). You may need to call rewind() to reset the file pointer to the beginning of the file if you have already written data to the file. The file is closed when **fpassthru()** is done reading it (leaving *fp* useless).

If you just want to dump the contents of a file to stdout you may want to use the readfile(), which saves you the fopen() call.

> **Megjegyzés:** When using **fpassthru()** on a binary file on Windows systems, you should make sure to open the file in binary mode by appending a b to the mode used in the call to fopen().

See also readfile(), fopen(), popen(), and fsockopen()

fputs (PHP 3, PHP 4)

Writes to a file pointer

int **fputs** (int fp, string str [, int length]) \linebreak

fputs() is an alias to fwrite(), and is identical in every way. Note that the *length* parameter is optional and if not specified the entire string will be written.

fread (PHP 3, PHP 4)

Binary-safe file read

string **fread** (int fp, int length) \linebreak

fread() reads up to *length* bytes from the file pointer referenced by *fp*. Reading stops when *length* bytes have been read or EOF is reached, whichever comes first.

```
// get contents of a file into a string
$filename = "/usr/local/something.txt";
$fd = fopen ($filename, "r");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

Megjegyzés: On systems which differentiate between binary and text files (i.e. Windows) the file must be opened with 'b' included in fopen() mode parameter.

```
$filename = "c:\\files\\somepic.gif";
$fd = fopen ($filename, "rb");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

See also fwrite(), fopen(), fsockopen(), popen(), fgets(), fgetss(), fscanf(), file(), and fpassthru().

fscanf (PHP 4 >= 4.0.1)

Parses input from a file according to a format

mixed **fscanf** (int handle, string format [, string var1]) \linebreak

The function **fscanf()** is similar to sscanf(), but it takes its input from a file associated with *`handle`* and interprets the input according to the specified *`format`*. If only two parameters were passed to this function, the values parsed will be returned as an array. Otherwise, if optional parameters are passed, the function will return the number of assigned values. The optional parameters must be passed by reference.

Any whitespace in the format string matches any whitespace in the input stream. This means that even a tab \n in the format string can match a single space character in the input stream.

Példa 1. fscanf() Example

```
$fp = fopen ("users.txt","r");
while ($userinfo = fscanf ($fp, "%s\t%s\t%s\n")) {
    list ($name, $profession, $countrycode) = $userinfo;
    //... do something with the values
}
fclose($fp);
```

Példa 2. users.txt

```
javier  argonaut        pe
hiroshi sculptor        jp
robert  slacker us
luigi   florist it
```

See also fread(), fgets(), fgetss(), sscanf(), printf(), and sprintf().

fseek (PHP 3, PHP 4)

Seeks on a file pointer

int **fseek** (int fp, int offset [, int whence]) \linebreak

Sets the file position indicator for the file referenced by *fp*.The new position, measured in bytes from the beginning of the file, is obtained by adding *offset* to the position specified by *whence*, whose values are defined as follows:
SEEK_SET - Set position equal to *offset* bytes.
SEEK_CUR - Set position to current location plus *offset*.
SEEK_END - Set position to end-of-file plus *offset*. (To move to a position before the end-of-file, you need to pass a

If *whence* is not specified, it is assumed to be SEEK_SET.

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

May not be used on file pointers returned by fopen() if they use the "http://" or "ftp://" formats.

Megjegyzés: The *whence* argument was added after PHP 4.0.0.

See also ftell() and rewind().

fstat (PHP 4)

Gets information about a file using an open file pointer

array **fstat** (int fp) \linebreak

Gathers the statistics of the file opened by the file pointer fp. This function is similar to the stat() function except that it operates on an open file pointer instead of a filename.

Returns an array with the statistics of the file with the following elements:

1. device

2. inode
3. number of links
4. user id of owner
5. group id owner
6. device type if inode device *
7. size in bytes
8. time of last access
9. time of last modification
10. time of last change
11. blocksize for filesystem I/O *
12. number of blocks allocated

* - only valid on systems supporting the st_blksize type--other systems (i.e. Windows) return -1

The results of this function are cached. See clearstatcache() for more details.

ftell (PHP 3, PHP 4)

Tells file pointer read/write position

int **ftell** (int fp) \linebreak

Returns the position of the file pointer referenced by fp; i.e., its offset into the file stream.

If an error occurs, returns `FALSE`.

The file pointer must be valid, and must point to a file successfully opened by fopen() or popen().

See also fopen(), popen(), fseek(), and rewind().

ftruncate (PHP 4)

Truncates a file to a given length

int **ftruncate** (int fp, int size) \linebreak

Takes the filepointer, *`fp`*, and truncates the file to length, *`size`*. This function returns `TRUE` on success and `FALSE` on failure.

fwrite (PHP 3, PHP 4)

Binary-safe file write

int **fwrite** (int fp, string string [, int length]) \linebreak

fwrite() writes the contents of *`string`* to the file stream pointed to by *`fp`*. If the *`length`* argument is given, writing will stop after *`length`* bytes have been written or the end of *`string`* is reached, whichever comes first.

fwrite() returns the number of bytes written, or -1 on error.

Note that if the *`length`* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *`string`*.

> **Megjegyzés:** On systems which differentiate between binary and text files (i.e. Windows) the file must be opened with 'b' included in fopen() mode parameter.

See also fread(), fopen(), fsockopen(), popen(), and fputs().

glob (PHP 4 CVS only)

Find pathnames matching a pattern

array **glob** (string pattern [, int flags]) \linebreak

The **glob()** function searches for all the pathnames matching *`pattern`* according to the rules used by the shell. No tilde expansion or parameter substitution is done.

Returns an array containing the matched files/directories or FALSE on error.

> **Megjegyzés:** This function is disabled in safe mode and therefore will always return FALSE in safe mode.

Példa 1. Convenient way how glob() can replace opendir() and friends.

```
foreach (glob("*.txt") as $filename) {
    echo "$filename size " . filesize($filename) . "\n";
}
```

This could result in the following output:

```
funclist.txt size 44686
funcsummary.txt size 267625
quickref.txt size 137820
```

See also opendir(), readdir() and closedir().

is_dir (PHP 3, PHP 4)

Tells whether the filename is a directory

bool **is_dir** (string filename) \linebreak

Returns TRUE if the filename exists and is a directory. If *filename* is a relative filename, it will be checked relative to the current working directory.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also chdir(), is_file() and is_link().

is_executable (PHP 3, PHP 4)

Tells whether the filename is executable

bool **is_executable** (string filename) \linebreak

Returns TRUE if the filename exists and is executable.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also is_file() and is_link().

is_file (PHP 3, PHP 4)

Tells whether the filename is a regular file

bool **is_file** (string filename) \linebreak

Returns TRUE if the filename exists and is a regular file.

The results of this function are cached. See clearstatcache() for more details.

See also is_dir() and is_link().

is_link (PHP 3, PHP 4)

Tells whether the filename is a symbolic link

bool **is_link** (string filename) \linebreak

Returns TRUE if the filename exists and is a symbolic link.

The results of this function are cached. See clearstatcache() for more details.

See also is_dir(), is_file(), and readlink().

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

is_readable (PHP 3, PHP 4)

Tells whether the filename is readable

bool **is_readable** (string filename) \linebreak

Returns TRUE if the filename exists and is readable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also is_writable().

is_uploaded_file (PHP 3>= 3.0.17, PHP 4 >= 4.0.3)

Tells whether the file was uploaded via HTTP POST

bool **is_uploaded_file** (string filename) \linebreak

Returns TRUE if the file named by `filename` was uploaded via HTTP POST. This is useful to help ensure that a malicious user hasn't tried to trick the script into working on files upon which it should not be working--for instance, `/etc/passwd`.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

is_uploaded_file() is available only in versions of PHP 3 after PHP 3.0.16, and in versions of PHP 4 after 4.0.2. If you are stuck using an earlier version, you can use the following function to help protect yourself:

Megjegyzés: The following example will *not* work in versions of PHP 4 after 4.0.2. It depends on internal functionality of PHP which changed after that version.

```
<?php
/* Userland test for uploaded file. */
function is_uploaded_file($filename) {
    if (!$tmp_file = get_cfg_var('upload_tmp_dir')) {
        $tmp_file = dirname(tempnam(", "));
    }
    $tmp_file .= '/' . basename($filename);
    /* User might have trailing slash in php.ini... */
    return (ereg_replace('/+', '/', $tmp_file) == $filename);
}
```

```
/* This is how to use it, since you also don't have
 * move_uploaded_file() in these older versions: */
if (is_uploaded_file($HTTP_POST_FILES['userfile'])) {
    copy($HTTP_POST_FILES['userfile'], "/place/to/put/uploaded/file");
} else {
    echo "Possible file upload attack: filename '$HTTP_POST_FILES[userfile]'.";
}
?>
```

See also move_uploaded_file(), and the section Handling file uploads for a simple usage example.

is_writable (PHP 4)

Tells whether the filename is writable

bool **is_writable** (string filename) \linebreak

Returns `TRUE` if the filename exists and is writable. The filename argument may be a directory name allowing you to check if a directory is writeable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also is_readable().

is_writeable (PHP 3, PHP 4)

Tells whether the filename is writable

bool **is_writeable** (string filename) \linebreak

This is an alias for is_writable()

link (PHP 3, PHP 4)

Create a hard link

int **link** (string target, string link) \linebreak

link() creates a hard link.

See also the symlink() to create soft links, and readlink() along with linkinfo().

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

linkinfo (PHP 3, PHP 4)

Gets information about a link

int **linkinfo** (string path) \linebreak

linkinfo() returns the st_dev field of the UNIX C stat structure returned by the lstat system call. This function is used to verify if a link (pointed to by *path*) really exists (using the same method as the S_ISLNK macro defined in stat.h). Returns 0 or FALSE in case of error.

See also symlink(), link(), and readlink().

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

lstat (PHP 3>= 3.0.4, PHP 4)

Gives information about a file or symbolic link

array **lstat** (string filename) \linebreak

Gathers the statistics of the file or symbolic link named by filename. This function is identical to the stat() function except that if the *filename* parameter is a symbolic link, the status of the symbolic link is returned, not the status of the file pointed to by the symbolic link.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. inode protection mode
4. number of links
5. user id of owner
6. group id owner
7. device type if inode device *
8. size in bytes
9. time of last access
10. time of last modification
11. time of last change
12. blocksize for filesystem I/O *
13. number of blocks allocated

* - only valid on systems supporting the st_blksize type--other systems (i.e. Windows) return -1.

lstat() cannot be used on remote files.

The results of this function are cached. See clearstatcache() for more details.

mkdir (PHP 3, PHP 4)

Makes directory

int **mkdir** (string pathname, int mode) \linebreak

Attempts to create the directory specified by pathname.

Note that you probably want to specify the mode as an octal number, which means it should have a leading zero. The mode is also modified by the current umask, which you can change using umask().

```
mkdir ("/path/to/my/dir", 0700);
```

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

See also rmdir().

move_uploaded_file (PHP 4 >= 4.0.3)

Moves an uploaded file to a new location

bool **move_uploaded_file** (string filename, string destination) \linebreak

This function checks to ensure that the file designated by *filename* is a valid upload file (meaning that it was uploaded via PHP's HTTP POST upload mechanism). If the file is valid, it will be moved to the filename given by *destination*.

If *filename* is not a valid upload file, then no action will occur, and **move_uploaded_file()** will return FALSE.

If *filename* is a valid upload file, but cannot be moved for some reason, no action will occur, and **move_uploaded_file()** will return FALSE. Additionally, a warning will be issued.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

> **Megjegyzés:** Ha a safe mode be van kapcsolva, a PHP ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.

Megjegyzés: move_uploaded_file() is not affected by the normal safe-mode UID-restrictions. This is not unsafe because **move_uploaded_file()** only operates on files uploaded via PHP.

Figyelem

If the destination file already exists, it will be overwritten.

See also is_uploaded_file(), and the section Handling file uploads for a simple usage example.

parse_ini_file (PHP 4)

Parse a configuration file

array **parse_ini_file** (string filename [, bool process_sections]) \linebreak

parse_ini_file() loads in the ini file specified in *`filename`*, and returns the settings in it in an associative array. By setting the last *`process_sections`* parameter to TRUE, you get a multidimensional array, with the section names and settings included. The default for *`process_sections`* is FALSE

Megjegyzés: This function has nothing to do with the `php.ini` file. It is already processed, the time you run your script. This function can be used to read in your own application's configuration files.

Megjegyzés: If a value in the ini file contains any non-alphanumeric characters it needs to be enclosed in double-quotes (").

Megjegyzés: Since PHP 4.2.1 this function is also affected by safe_mode and open_basedir.

The structure of the ini file is similar to that of the `php.ini`'s.

Figyelem

If the ini file you are trying to parse is malformed, PHP will exit.

Példa 1. Contents of sample.ini

```
; This is a sample configuration file
; Comments start with ';', as in php.ini

[first_section]
one = 1
five = 5
```

```
[second_section]
path = /usr/local/bin
URL = "http://www.example.com/~username"
```

Példa 2. parse_ini_file() example

```
<?php

// Parse without sections
$ini_array = parse_ini_file("sample.ini");
print_r($ini_array);

// Parse with sections
$ini_array = parse_ini_file("sample.ini", TRUE);
print_r($ini_array);

?>
```

Would produce:

```
Array
(
    [one] => 1
    [five] => 5
    [path] => /usr/local/bin
    [URL] => http://www.example.com/~username
)
Array
(
    [first_section] => Array
        (
            [one] => 1
            [five] => 5
        )

    [second_section] => Array
        (
            [path] => /usr/local/bin
            [URL] => http://www.example.com/~username
        )

)
```

pathinfo (PHP 4 >= 4.0.3)

Returns information about a file path

array **pathinfo** (string path) \linebreak

pathinfo() returns an associative array containing information about *`path`*. The following array elements are returned: *`dirname`*, *`basename`* and *`extension`*.

Példa 1. pathinfo() Example

```
<?php

$path_parts = pathinfo("/www/htdocs/index.html");

echo $path_parts["dirname"] . "\n";
echo $path_parts["basename"] . "\n";
echo $path_parts["extension"] . "\n";

?>
```

Would produce:

```
/www/htdocs
index.html
html
```

See also dirname(), basename(), parse_url() and realpath().

pclose (PHP 3, PHP 4)

Closes process file pointer

int **pclose** (int fp) \linebreak

Closes a file pointer to a pipe opened by popen().

The file pointer must be valid, and must have been returned by a successful call to popen().

Returns the termination status of the process that was run.

See also popen().

popen (PHP 3, PHP 4)

Opens process file pointer

int **popen** (string command, string mode) \linebreak

Opens a pipe to a process executed by forking the command given by command.

Returns a file pointer identical to that returned by fopen(), except that it is unidirectional (may only be used for reading or writing) and must be closed with pclose(). This pointer may be used with fgets(), fgetss(), and fputs().

If an error occurs, returns FALSE.

```
$fp = popen ("/bin/ls", "r");
```

Megjegyzés: If the command to be executed could not be found, a valid resource is returned. This may seem odd, but makes sense; it allows you to access any error message returned by the shell:

```
<?php
error_reporting(E_ALL);

/* Add redirection so we can get stderr. */
$fp = popen('/path/to/spooge 2>&1', 'r');
echo "'$fp'; " . gettype($fp) . "\n";
$read = fread($fp, 2096);
echo $read;
pclose($fp);
?>
```

See also pclose().

readfile (PHP 3, PHP 4)

Outputs a file

int **readfile** (string filename [, int use_include_path]) \linebreak

Reads a file and writes it to standard output.

Returns the number of bytes read from the file. If an error occurs, FALSE is returned and unless the function was called as @readfile, an error message is printed.

If *`filename`* begins with "http://" (not case sensitive), an HTTP 1.0 connection is opened to the specified server and the text of the response is written to standard output.

Versions prior to PHP 4.0.5 do not handle HTTP redirects. Because of this, directories must include trailing slashes.

If *`filename`* begins with "ftp://" (not case sensitive), an ftp connection to the specified server is opened and the requested file is written to standard output. If the server does not support passive mode ftp, this will fail.

If *`filename`* begins with neither of these strings, the file will be opened from the filesystem and its contents written to standard output.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also fpassthru(), file(), fopen(), include(), require(), and virtual().

readlink (PHP 3, PHP 4)

Returns the target of a symbolic link

string **readlink** (string path) \linebreak

readlink() does the same as the readlink C function and returns the contents of the symbolic link path or 0 in case of error.

See also is_link(), symlink(), and linkinfo().

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

realpath (PHP 4)

Returns canonicalized absolute pathname

string **realpath** (string path) \linebreak

realpath() expands all symbolic links and resolves references to '/./', '/../' and extra '/' characters in the input *`path`* and return the canonicalized absolute pathname. The resulting path will have no symbolic link, '/./' or '/../' components.

realpath() returns `FALSE` on failure, e.g. if the file does not exists.

Példa 1. realpath() example

```
$real_path = realpath ("../../index.php");
```

See also: basename(), dirname(), and pathinfo().

rename (PHP 3, PHP 4)

Renames a file

bool **rename** (string oldname, string newname) \linebreak

Attempts to rename *oldname* to *newname*.

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

rewind (PHP 3, PHP 4)

Rewind the position of a file pointer

int **rewind** (int fp) \linebreak

Sets the file position indicator for fp to the beginning of the file stream.

If an error occurs, returns 0.

The file pointer must be valid, and must point to a file successfully opened by fopen().

> **Megjegyzés:** If you have opened the file in append ("a") mode, any data you write to the file will always be appended, regardless of the file position.

See also fseek() and ftell().

rmdir (PHP 3, PHP 4)

Removes directory

bool **rmdir** (string dirname) \linebreak

Attempts to remove the directory named by *dirname*. The directory must be empty, and the relevant permissions must permit this. Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

See also mkdir() and unlink().

set_file_buffer (PHP 3>= 3.0.8, PHP 4 >= 4.0.1)

Sets file buffering on the given file pointer

int **set_file_buffer** (int fp, int buffer) \linebreak

Output using fwrite() is normally buffered at 8K. This means that if there are two processes wanting to write to the same output stream (a file), each is paused after 8K of data to allow the other to write. **set_file_buffer()** sets the buffering for write operations on the given filepointer *`fp`* to *`buffer`* bytes. If *`buffer`* is 0 then write operations are unbuffered. This ensures that all writes with fwrite() are completed before other processes are allowed to write to that output stream.

The function returns 0 on success, or EOF if the request cannot be honored.

The following example demonstrates how to use **set_file_buffer()** to create an unbuffered stream.

Példa 1. set_file_buffer() example

```
$fp=fopen($file, "w");
if($fp){
  set_file_buffer($fp, 0);
  fputs($fp, $output);
  fclose($fp);
}
```

See also fopen(), fwrite().

stat (PHP 3, PHP 4)

Gives information about a file

array **stat** (string filename) \linebreak

Gathers the statistics of the file named by filename.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. inode protection mode
4. number of links
5. user id of owner
6. group id owner
7. device type if inode device *
8. size in bytes
9. time of last access
10. time of last modification
11. time of last change
12. blocksize for filesystem I/O *
13. number of blocks allocated

* - only valid on systems supporting the st_blksize type--other systems (i.e. Windows) return -1.

Returns FALSE in case of error.

stat() cannot be used on remote files.

The results of this function are cached. See clearstatcache() for more details.

symlink (PHP 3, PHP 4)

Creates a symbolic link

int **symlink** (string target, string link) \linebreak

symlink() creates a symbolic link from the existing *target* with the specified name *link*.

See also link() to create hard links, and readlink() along with linkinfo().

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

tempnam (PHP 3, PHP 4)

Create file with unique file name

string **tempnam** (string dir, string prefix) \linebreak

Creates a file with a unique filename in the specified directory. If the directory does not exist, **tempnam()** may generate a file in the system's temporary directory, and return the name of that.

Prior to PHP 4.0.6, the behaviour of the **tempnam()** function was system dependent. On Windows the TMP environment variable will override the *dir* parameter, on Linux the TMPDIR environment variable has precedence, while SVR4 will always use your *dir* parameter if the directory it points to exists. Consult your system documentation on the tempnam(3) function if in doubt.

Returns the new temporary filename, or the FALSE string on failure.

Példa 1. tempnam() example

```
$tmpfname = tempnam ("/tmp", "FOO");

$fp = fopen($tmpfname, "w");
fwrite($fp, "writing to tempfile");
fclose($fp);

// do here something

unlink($tmpfname);
```

> **Megjegyzés:** This function's behavior changed in 4.0.3. The temporary file is also created to avoid a race condition where the file might appear in the filesystem between the time the string was generated and before the the script gets around to creating the file. Note, that you need to remove the file in case you need it no more, it is not done automatically.

See also tmpfile() and unlink().

tmpfile (PHP 3>= 3.0.13, PHP 4)

Creates a temporary file

int **tmpfile** (void) \linebreak

Creates a temporary file with an unique name in write mode, returning a file handle similar to the one returned by fopen(). The file is automatically removed when closed (using fclose()), or when the script ends.

For details, consult your system documentation on the `tmpfile(3)` function, as well as the `stdio.h` header file.

Példa 1. tmpfile() example

```
$temp = tmpfile();
fwrite($temp, "writing to tempfile");
fclose($temp); // this removes the file
```

See also tempnam().

touch (PHP 3, PHP 4)

Sets access and modification time of file

int **touch** (string filename [, int time [, int atime]]) \linebreak

Attempts to set the access and modification time of the file named by filename to the value given by time. If the option *`time`* is not given, uses the present time. This is equivalent to what utime (sometimes referred to as utimes) does. If the third option *`atime`* is present, the access time of the given filename is also modified.

If the file does not exist, it is created.

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

Példa 1. touch() example

```
if (touch ($FileName)) {
    print "$FileName modification time has been
           changed to todays date and time";
} else {
    print "Sorry Could Not change modification time of $FileName";
}
```

umask (PHP 3, PHP 4)

Changes the current umask

int **umask** (int mask) \linebreak

umask() sets PHP's umask to mask & 0777 and returns the old umask. When PHP is being used as a server module, the umask is restored when each request is finished.

umask() without arguments simply returns the current umask.

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

unlink (PHP 3, PHP 4)

Deletes a file

int **unlink** (string filename) \linebreak

Deletes *`filename`*. Similar to the Unix C unlink() function.

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

See also rmdir() for removing directories.

XXXI. Forms Data Format functions

Bevezetés

Forms Data Format (FDF) is a format for handling forms within PDF documents. You should read the documentation at http://partners.adobe.com/asn/developer/acrosdk/forms.html for more information on what FDF is and how it is used in general.

The general idea of FDF is similar to HTML forms. The difference is basically the format how data is transmitted to the server when the submit button is pressed (this is actually the Form Data Format) and the format of the form itself (which is the Portable Document Format, PDF). Processing the FDF data is one of the features provided by the fdf functions. But there is more. One may as well take an existing PDF form and populated the input fields with data without modifying the form itself. In such a case one would create a FDF document (fdf_create()) set the values of each input field (fdf_set_value()) and associate it with a PDF form (fdf_set_file()). Finally it has to be sent to the browser with MimeType `application/vnd.fdf`. The Acrobat reader plugin of your browser recognizes the MimeType, reads the associated PDF form and fills in the data from the FDF document.

If you look at an FDF-document with a text editor you will find a catalogue object with the name `FDF`. Such an object may contain a number of entries like `Fields`, `F`, `Status` etc.. The most commonly used entries are `Fields` which points to a list of input fields, and `F` which contains the filename of the PDF-document this data belongs to. Those entries are referred to in the FDF documentation as /F-Key or /Status-Key. Modifying this entries is done by functions like fdf_set_file() and fdf_set_status(). Fields are modified with fdf_set_value(), fdf_set_opt() etc..

Követelmények

You must download the FDF toolkit from http://partners.adobe.com/asn/developer/acrosdk/forms.html.

Telepítés

You must compile PHP with `--with-fdftk[=DIR]`.

> **Megjegyzés:** If you run into problems configuring PHP with fdftk support, check whether the header file FdfTk.h and the library libFdfTk.so are at the right place. They should be in

fdftk-dir/include and fdftk-dir/lib. This will not be the case if you just unpack the FdfTk distribution.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Előre definiált állandók

Az itt listázott állandókat ez a kiterjesztés definiálja, és csak akkor elérhetőek, ha az adott

kiterjesztés be van fordítva a PHP-be, vagy dinamikusan betöltött.

`FDFValue` (integer)

`FDFStatus` (integer)

`FDFFile` (integer)

`FDFID` (integer)

`FDFFf` (integer)

`FDFSetFf` (integer)

`FDFClearFf` (integer)

`FDFFlags` (integer)

`FDFSetF` (integer)

`FDFClrF` (integer)

`FDFAP` (integer)

`FDFAS` (integer)

`FDFAction` (integer)

`FDFAA` (integer)

`FDFAPRef` (integer)

`FDFIF` (integer)

`FDFEnter` (integer)

`FDFExit` (integer)

`FDFDown` (integer)

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Open temp file and evaluate data
// The pdf form contained several input text fields with the names
// volume, date, comment, publisher, preparer, and two checkboxes
// show_publisher and show_preparer.
$fdf = fdf_open("test.fdf");
$volume = fdf_get_value($fdf, "volume");
echo "The volume field has the value '<B>$volume</B>'<BR>";

$date = fdf_get_value($fdf, "date");
echo "The date field has the value '<B>$date</B>'<BR>";

$comment = fdf_get_value($fdf, "comment");
echo "The comment field has the value '<B>$comment</B>'<BR>";

if(fdf_get_value($fdf, "show_publisher") == "On") {
  $publisher = fdf_get_value($fdf, "publisher");
  echo "The publisher field has the value '<B>$publisher</B>'<BR>";
} else
  echo "Publisher shall not be shown.<BR>";

if(fdf_get_value($fdf, "show_preparer") == "On") {
  $preparer = fdf_get_value($fdf, "preparer");
  echo "The preparer field has the value '<B>$preparer</B>'<BR>";
} else
  echo "Preparer shall not be shown.<BR>";
fdf_close($fdf);
?>
```

fdf_add_template (PHP 3>= 3.0.13, PHP 4)

Adds a template into the FDF document

bool **fdf_add_template** (int fdfdoc, int newpage, string filename, string template, int rename) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

fdf_close (PHP 3>= 3.0.6, PHP 4)

Close an FDF document

bool **fdf_close** (int fdf_document) \linebreak

The **fdf_close()** function closes the FDF document.

See also fdf_open().

fdf_create (PHP 3>= 3.0.6, PHP 4)

Create a new FDF document

int **fdf_create** (void) \linebreak

The **fdf_create()** creates a new FDF document. This function is needed if one would like to populate input fields in a PDF document with data.

Példa 1. Populating a PDF document

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, "volume", $volume, 0);

fdf_set_file($outfdf, "http:/testfdf/resultlabel.pdf");
fdf_save($outfdf, "outtest.fdf");
fdf_close($outfdf);
Header("Content-type: application/vnd.fdf");
$fp = fopen("outtest.fdf", "r");
fpassthru($fp);
unlink("outtest.fdf");
?>
```

See also fdf_close(), fdf_save(), fdf_open().

fdf_get_file (PHP 3>= 3.0.6, PHP 4)

Get the value of the /F key

string **fdf_get_file** (int fdf_document) \linebreak

The fdf_set_file() returns the value of the /F key.

See also fdf_set_file().

fdf_get_status (PHP 3>= 3.0.6, PHP 4)

Get the value of the /STATUS key

string **fdf_get_status** (int fdf_document) \linebreak

The **fdf_get_status()** returns the value of the /STATUS key.

See also fdf_set_status().

fdf_get_value (PHP 3>= 3.0.6, PHP 4)

Get the value of a field

string **fdf_get_value** (int fdf_document, string fieldname) \linebreak

The **fdf_get_value()** function returns the value of a field.

See also fdf_set_value().

fdf_next_field_name (PHP 3>= 3.0.6, PHP 4)

Get the next field name

string **fdf_next_field_name** (int fdf_document [, string fieldname]) \linebreak

The **fdf_next_field_name()** function returns the name of the field after the field in *`fieldname`* or the field name of the first field if the second parameter is `NULL`.

See also fdf_set_value(), fdf_get_value().

fdf_open (PHP 3>= 3.0.6, PHP 4)

Open a FDF document

int **fdf_open** (string filename) \linebreak

The **fdf_open()** function opens a file with form data. This file must contain the data as returned from a PDF form. Currently, the file has to be created 'manually' by using fopen() and writing the content of HTTP_FDF_DATA with fwrite() into it. A mechanism like for HTML form data where for each input field a variable is created does not exist.

Példa 1. Accessing the form data

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Open temp file and evaluate data
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

See also fdf_close().

fdf_save (PHP 3>= 3.0.6, PHP 4)

Save a FDF document

int **fdf_save** (string filename) \linebreak

The **fdf_save()** function saves a FDF document. The FDF Toolkit provides a way to output the document to stdout if the parameter *filename* is '.'. This does not work if PHP is used as an apache module. In such a case one will have to write to a file and use e.g. fpassthru() to output it.

See also fdf_close() and example for fdf_create().

fdf_set_ap (PHP 3>= 3.0.6, PHP 4)

Set the appearance of a field

bool **fdf_set_ap** (int fdf_document, string field_name, int face, string filename, int page_number) \linebreak

The **fdf_set_ap()** function sets the appearance of a field (i.e. the value of the /AP key). The possible values of *face* are 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

fdf_set_encoding (PHP 4 >= 4.1.0)

Sets FDF character encoding

bool **fdf_set_encoding** (int fdf_document, string encoding) \linebreak

fdf_set_encoding() sets the character encoding in FDF document *fdf_document*. *encoding* should be the valid encoding name. The valid encoding name in Acrobat 5.0 are "Shift-JIS", "UHC", "GBK","BigFive".

The **fdf_set_encoding()** is available in PHP 4.1.0 or later.

fdf_set_file (PHP 3>= 3.0.6, PHP 4)

Set the value of the /F key

bool **fdf_set_file** (int fdf_document, string filename) \linebreak

The **fdf_set_file()** sets the value of the /F key. The /F key is just a reference to a PDF form which is to be populated with data. In a web environment it is a URL (e.g. http:/testfdf/resultlabel.pdf).

See also fdf_get_file() and example for fdf_create().

fdf_set_flags (PHP 4 >= 4.0.2)

Sets a flag of a field

bool **fdf_set_flags** (int fdf_document, string fieldname, int whichFlags, int newFlags) \linebreak

The **fdf_set_flags()** sets certain flags of the given field *fieldname*.

See also fdf_set_opt().

fdf_set_javascript_action (PHP 4 >= 4.0.2)

Sets an javascript action of a field

bool **fdf_set_javascript_action** (int fdf_document, string fieldname, int trigger, string script) \linebreak

fdf_set_javascript_action() sets a javascript action for the given field *fieldname*.

See also fdf_set_submit_form_action().

fdf_set_opt (PHP 4 >= 4.0.2)

Sets an option of a field

bool **fdf_set_opt** (int fdf_document, string fieldname, int element, string str1, string str2) \linebreak

The **fdf_set_opt()** sets options of the given field *fieldname*.

See also fdf_set_flags().

fdf_set_status (PHP 3>= 3.0.6, PHP 4)

Set the value of the /STATUS key

bool **fdf_set_status** (int fdf_document, string status) \linebreak

The **fdf_set_status()** sets the value of the /STATUS key.

See also fdf_get_status().

fdf_set_submit_form_action (PHP 4 >= 4.0.2)

Sets a submit form action of a field

bool **fdf_set_submit_form_action** (int fdf_document, string fieldname, int trigger, string script, int flags) \linebreak

The **fdf_set_submit_form_action()** sets a submit form action for the given field *fieldname*.

See also fdf_set_javascript_action().

fdf_set_value (PHP 3>= 3.0.6, PHP 4)

Set the value of a field

bool **fdf_set_value** (int fdf_document, string fieldname, string value, int isName) \linebreak

The **fdf_set_value()** function sets the value of a field. The last parameter determines if the field value is to be converted to a PDF Name (*isName* = 1) or set to a PDF String (*isName* = 0).

See also fdf_get_value().

XXXII. FriBiDi functions

Bevezetés

Követelmények

Telepítés

Futásidejű beállítások

Erőforrás típusok

Előre definiált állandók

Az itt listázott állandókat ez a kiterjesztés definiálja, és csak akkor elérhetőek, ha az adott kiterjesztés be van fordítva a PHP-be, vagy dinamikusan betöltött.

`FRIBIDI_CHARSET_UTF8` (integer)

`FRIBIDI_CHARSET_8859_6` (integer)

`FRIBIDI_CHARSET_8859_8` (integer)

`FRIBIDI_CHARSET_CP1255` (integer)

`FRIBIDI_CHARSET_CP1256` (integer)

`FRIBIDI_CHARSET_ISIRI_3342` (integer)

fribidi_log2vis (PHP 4 >= 4.0.4)

Convert a logical string to a visual one

string **fribidi_log2vis** (string str, string direction, int charset) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

XXXIII. FTP functions

Bevezetés

The functions in this extension implement client access to file servers speaking the File Transfer Protocol FTP as defined in http://www.faqs.org/rfcs/rfc959.html.

Követelmények

Az itt leírt függvények a standard modulban találhatóak, ami mindig rendelkezésre áll.

Telepítés

In order to use FTP functions with your PHP configuration, you should add the `--enable-ftp` option when installing PHP 4, and `--with-ftp` when using PHP 3.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

This extension uses one resource-type, which is the link-identifier of the ftp-connection.

Előre definiált állandók

Az itt listázott állandókat ez a kiterjesztés definiálja, és csak akkor elérhetőek, ha az adott

kiterjesztés be van fordítva a PHP-be, vagy dinamikusan betöltött.

`FTP_ASCII` (integer)

`FTP_TEXT` (integer)

`FTP_BINARY` (integer)

`FTP_IMAGE` (integer)

`FTP_TIMEOUT_SEC` (integer)

Példák

Példa 1. FTP example

```
<?php
// set up basic connection
$conn_id = ftp_connect($ftp_server);

// login with username and password
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// check connection
if ((!$conn_id) || (!$login_result)) {
        echo "FTP connection has failed!";
        echo "Attempted to connect to $ftp_server for user $ftp_user_name";
        die;
    } else {
        echo "Connected to $ftp_server, for user $ftp_user_name";
    }

// upload the file
$upload = ftp_put($conn_id, $destination_file, $source_file, FTP_BINARY);

// check upload status
if (!$upload) {
        echo "FTP upload has failed!";
    } else {
        echo "Uploaded $source_file to $ftp_server as $destination_file";
    }

// close the FTP stream
ftp_close($conn_id);
?>
```

ftp_cdup (PHP 3>= 3.0.13, PHP 4)

Changes to the parent directory

bool **ftp_cdup** (resource ftp_stream) \linebreak

Changes to the parent directory.

Returns TRUE on success, FALSE on error.

ftp_chdir (PHP 3>= 3.0.13, PHP 4)

Changes directories on a FTP server

bool **ftp_chdir** (resource ftp_stream, string directory) \linebreak

Changes to the specified *directory*.

Returns TRUE on success, FALSE on error.

ftp_close (PHP 4 >= 4.2.0)

Closes an FTP connection

void **ftp_close** (resource ftp_stream) \linebreak

Megjegyzés: This function is only available in CVS.

ftp_close() closes *ftp_stream* and releases the resource. You can't reuse this resource but have to create a new one with ftp_connect().

ftp_connect (PHP 3>= 3.0.13, PHP 4)

Opens up an FTP connection

resource **ftp_connect** (string host [, int port [, int timeout]]) \linebreak

Returns a FTP stream on success, FALSE on error.

ftp_connect() opens up a FTP connection to the specified *host*. The *port* parameter specifies an alternate port to connect to. If it is omitted or zero, then the default FTP port, 21, will be used.

The *timeout* parameter specifies the timeout for all subsequent network operations. If omitted, the default value is 90 seconds. The timeout can be changed and queried anytime with ftp_set_option() and ftp_get_option().

Megjegyzés: This parameter is only available in CVS.

ftp_delete (PHP 3>= 3.0.13, PHP 4)

Deletes a file on the FTP server

bool **ftp_delete** (resource ftp_stream, string path) \linebreak

ftp_delete() deletes the file specified by *path* from the FTP server.

Returns TRUE on success, FALSE on error.

ftp_exec (PHP 4 >= 4.0.3)

Request execution of a program on the FTP server

bool **ftp_exec** (resource ftp_stream, string command) \linebreak

Sends a SITE EXEC *command* request to the FTP server. Returns FALSE if the request fails, returns the output of the command otherwise.

ftp_fget (PHP 3>= 3.0.13, PHP 4)

Downloads a file from the FTP server and saves to an open file

bool **ftp_fget** (resource ftp_stream, resource fp, string remote_file, int mode) \linebreak

ftp_fget() retrieves *remote_file* from the FTP server, and writes it to the given file pointer, *fp*. The transfer *mode* specified must be either FTP_ASCII or FTP_BINARY.

Returns TRUE on success, FALSE on error.

ftp_fput (PHP 3>= 3.0.13, PHP 4)

Uploads from an open file to the FTP server

bool **ftp_fput** (resource ftp_stream, string remote_file, resource fp, int mode) \linebreak

ftp_fput() uploads the data from the file pointer *fp* until end of file. The results are stored in *remote_file* on the FTP server. The transfer *mode* specified must be either FTP_ASCII or FTP_BINARY

Returns TRUE on success, FALSE on error.

ftp_get_option (PHP 4 >= 4.2.0)

Retrieves various runtime behaviours of the current FTP stream

mixed **ftp_get_option** (resource ftp_stream, int option) \linebreak

Megjegyzés: This function is only available in CVS.

Returns the value on success, or FALSE if the given *option* is not supported. In the latter case, a warning message is also thrown.

This function returns the value for the requested *option* from the specified *ftp_stream* . Currently, the following options are supported:

Táblázat 1. Supported runtime FTP options

FTP_TIMEOUT_SEC	Returns the current timeout used for network related operations.

Példa 1. ftp_get_option() example

```
// Get the timeout of the given FTP stream
$timeout = ftp_get_option($conn_id, FTP_TIMEOUT_SEC);
```

ftp_get (PHP 3>= 3.0.13, PHP 4)

Downloads a file from the FTP server

bool **ftp_get** (resource ftp_stream, string local_file, string remote_file, int mode) \linebreak

ftp_get() retrieves *remote_file* from the FTP server, and saves it to *local_file* locally. The transfer *mode* specified must be either FTP_ASCII or FTP_BINARY.

Returns TRUE on success, FALSE on error.

See also ftp_fget().

ftp_login (PHP 3>= 3.0.13, PHP 4)

Logs in an FTP connection

bool **ftp_login** (resource ftp_stream, string username, string password) \linebreak

Logs in the given FTP stream.

Returns TRUE on success, FALSE on error.

ftp_mdtm (PHP 3>= 3.0.13, PHP 4)

Returns the last modified time of the given file

int **ftp_mdtm** (resource ftp_stream, string remote_file) \linebreak

ftp_mdtm() checks the last-modified time for a file, and returns it as a UNIX timestamp. If an error occurs, or the file does not exist, -1 is returned. Note that not all servers support this feature.

Returns a UNIX timestamp on success, or -1 on error.

Megjegyzés: ftp_mdtm() does not work with directories.

ftp_mkdir (PHP 3>= 3.0.13, PHP 4)

Creates a directory

string **ftp_mkdir** (resource ftp_stream, string directory) \linebreak

Creates the specified *`directory`*.

Returns the newly created directory name on success, FALSE on error.

ftp_nlist (PHP 3>= 3.0.13, PHP 4)

Returns a list of files in the given directory

array **ftp_nlist** (resource ftp_stream, string directory) \linebreak

Returns an array of filenames from the specified directory on success, FALSE on error.

ftp_pasv (PHP 3>= 3.0.13, PHP 4)

Turns passive mode on or off

bool **ftp_pasv** (resource ftp_stream, bool pasv) \linebreak

ftp_pasv() turns on passive mode if the *`pasv`* parameter is TRUE (it turns off passive mode if *`pasv`* is FALSE.) In passive mode, data connections are initiated by the client, rather than by the server.

Returns TRUE on success, FALSE on error.

ftp_put (PHP 3>= 3.0.13, PHP 4)

Uploads a file to the FTP server

bool **ftp_put** (resource ftp_stream, string remote_file, string local_file, int mode) \linebreak

ftp_put() stores *`local_file`* on the FTP server, as *`remote_file`*. The transfer *`mode`* specified must be either `FTP_ASCII` or `FTP_BINARY`.

Returns `TRUE` on success, `FALSE` on error.

Példa 1. ftp_put() example

```
$upload = ftp_put($conn_id, $destination_file, $source_file, FTP_ASCII);
```

ftp_pwd (PHP 3>= 3.0.13, PHP 4)

Returns the current directory name

string **ftp_pwd** (resource ftp_stream) \linebreak

Returns the current directory, or `FALSE` on error.

ftp_quit (PHP 3>= 3.0.13, PHP 4)

Closes an FTP connection

void **ftp_quit** (resource ftp_stream) \linebreak

ftp_quit() is an alias for ftp_close().

ftp_rawlist (PHP 3>= 3.0.13, PHP 4)

Returns a detailed list of files in the given directory

array **ftp_rawlist** (resource ftp_stream, string directory) \linebreak

ftp_rawlist() executes the FTP LIST command, and returns the result as an array. Each array element corresponds to one line of text. The output is not parsed in any way. The system type identifier returned by ftp_systype() can be used to determine how the results should be interpreted.

ftp_rename (PHP 3>= 3.0.13, PHP 4)

Renames a file on the FTP server

bool **ftp_rename** (resource ftp_stream, string from, string to) \linebreak

ftp_rename() renames the file or directory currently named *`from`* to the new name *`to`*, on the FTP stream *`ftp_stream`*.

Returns TRUE on success, FALSE on error.

ftp_rmdir (PHP 3>= 3.0.13, PHP 4)

Removes a directory

bool **ftp_rmdir** (resource ftp_stream, string directory) \linebreak

Removes the specified *`directory`*.

Returns TRUE on success, FALSE on error.

ftp_set_option (PHP 4 >= 4.2.0)

Set miscellaneous runtime FTP options

bool **ftp_set_option** (resource ftp_stream, int option, mixed value) \linebreak

Megjegyzés: This function is only available in CVS.

Returns TRUE if the option could be set; FALSE if not. A warning message will be thrown if the *`option`* is not supported or the passed *`value`* doesn't match the expected value for the given *`option`*.

This function controls various runtime options for the specified FTP stream. The *`value`* parameter depends on which *`option`* parameter is chosen to be altered. Currently, the following options are supported:

Táblázat 1. Supported runtime FTP options

FTP_TIMEOUT_SEC	Changes the timeout in seconds used for all network related functions. Parameter *`value`* has be to of type int and must be greater than 0. The default timeout is 90 seconds.

Példa 1. ftp_set_option() example

```
// Set the network timeout down to 10 seconds
ftp_set_option($conn_id, FTP_TIMEOUT_SEC, 10);
```

ftp_site (PHP 3>= 3.0.15, PHP 4)

Sends a SITE command to the server

bool **ftp_site** (resource ftp_stream, string cmd) \linebreak

ftp_site() sends the command specified by *cmd* to the FTP server. SITE commands are not standardized, and vary from server to server. They are useful for handling such things as file permissions and group membership.

Returns TRUE on success, FALSE on error.

ftp_size (PHP 3>= 3.0.13, PHP 4)

Returns the size of the given file

int **ftp_size** (resource ftp_stream, string remote_file) \linebreak

ftp_size() returns the size of a file in bytes. If an error occurs, of if the file does not exist, -1 is returned. Not all servers support this feature.

Returns the file size on success, or -1 on error.

ftp_systype (PHP 3>= 3.0.13, PHP 4)

Returns the system type identifier of the remote FTP server

string **ftp_systype** (resource ftp_stream) \linebreak

Returns the remote system type, or FALSE on error.

XXXIV. Függvénykezelő függvények

E függvények mindegyike különféle, a PHP függvényekhez szorosan kapcsolodó műveletekben segédkezik.

call_user_func_array (PHP 4 >= 4.0.4)

Felhasználói függvényt hív meg a paraméterek egy tömbjével

mixed **call_user_func_array** (string function_name [, array paramarr]) \linebreak

A *function_name*-ban megadott nevű felhasználói függvényt hívja meg *paramarr*-ben megadott paraméterekkel. Például:

```
function debug($var, $val)
    echo "***Bogarásszunk\nVÁLTOZÓ: $var\nÉRTÉKE:";
    if (is_array($val) || is_object($val) || is_resource($val))
        print_r($val);
    else
        echo "\n$val\n";
    echo "***\n";
}

$c = mysql_connect();
$host = $HTTP_SERVER_VARS["SERVER_NAME"];

call_user_func_array (’debug’, array("host", $host));
call_user_func_array (’debug’, array("c", $c));
call_user_func_array (’debug’, array("HTTP_POST_VARS", $HTTP_POST_VARS));
```

Lásd még: call_user_func(), call_user_method() és call_user_method_array()!

Megjegyzés: Ez a függvény a PHP 4.0.4pl1 kiadása után került a CVS-be.

call_user_func (PHP 3>= 3.0.3, PHP 4)

Felhasználói függvényt hív meg.

mixed **call_user_func** (string function_name [, mixed parameter [, mixed ...]]) \linebreak

A *function_name*-ban megadott nevű felhasználói függvényt hívja meg. Az ezután álló *parameter* paraméterek a megadott sorrendben lesznek átadva a meghívott függvénynek. Nézzük a kövekező példát:

```
function borbely ($tipus) {
    print " Szóval $tipus akar, semmi gond.";
}
call_user_func (’borbely’, "gomba frizurát");
call_user_func (’borbely’, "nyírást");
```

Lásd még: call_user_func_array(), call_user_method() és call_user_method_array()!

create_function (PHP 4 >= 4.0.1)

Név nélküli (lambda típusú) függvényt definiál

string **create_function** (string args, string code) \linebreak

Az átadott paraméterek alapján egy anonim - név nélküli függvényt hoz létre. Az *`args`* paramétert általában egyszeres idézőjelek közt szokás átadni, és ez ajánlott a *`code`*-ra is. Ennek az az oka, hogy az egyszeres idézőjelek közé fogott sztringekre nem aktiválódik a változó-behelyettesítés, máskülönben macskakörmök használata esetén a változó neveket visszaperjelezned kellene, mint például: `\$avar`.

Ezzel a függvénnyel - például futási időben nyert információk alapján - definiálhatsz új függvényeket:

Példa 1. A create_function() segítségével definiált függvények

```
$ujfgv = create_function('$a,$b','return "ln($a) + ln($b) = ".log($a * $b);');
echo "Új névtelen függvényünk: $ujfgv\n"; // azért valami neve csak van :)
echo $ujfgv(2,M_E)."\n";
// eredményül kírja:
// Új névtelen függvényünk: lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
```

Vagy egy általános kezelőfüggvény megléte esetén műveletek sorozata alkalmazható egy adott paraméterlistára, pl:

Példa 2. Általános feldolgozó függvény létrehozása create_function()-nel.

```
function feldolgoz($valtozo1, $valtozo2, $fvtomb) {
    for ($f=0; $f < count($fvtomb); $f++)
    echo $fvtomb[$f]($valtozo1,$valtozo2)."\n";
}

// egy csomó matematikai függvény létrehozása
$f1 = 'if ($a >=0) {return "b*a^2 = ".$b*sqrt($a);} else {return false;}';
$f2 = "return \"min(b^2+a, a^2,b) = \".min(\$a*\$a+\$b,\$b*\$b+\$a);";
$f3 = 'if ($a > 0 && $b != 0) {return "ln(a)/b = ".log($a)/$b;} else {return false;}';
$fv_tomb = array(
        create_function('$x,$y', 'return "ezt nem tudom, mi: ".(sin($x) + $x*cos($y));'),
        create_function('$x,$y', 'return "az átfogó: ".sqrt($x*$x + $y*$y);'),
        create_function('$a,$b', $f1),
        create_function('$a,$b', $f2),
        create_function('$a,$b', $f3)
    );
```

```
echo "\nAz első tömb anonim függvényekkel\n";
echo "paraméterek: 2.3445, M_PI\n";
feldolgoz(2.3445, M_PI, $fv_tomb);

// most egy csomó sztringkezelő függvény létrehozása
$fv_tomb2 = array(
        create_function('$b,$a','if (strncmp($a,$b,3) == 0) return "** \"$a\" '.
        'és \"$b\"\n** nekem azonosnak tűnik! (az első 3 karakter után)";'),
        create_function('$a,$b','; return "CRCs: ".crc32($a)." , ".crc32(b);'),
        create_function('$a,$b','; return "hasonló(a,b) = ".similar_text($a,$b,&$p)."($p%
    );
echo "\nA második tömb anonim függvényekkel\n";
feldolgoz("Akkor csillogó és fényes volt.", "Akkor este volt.", $fv_tomb2);
```

a fenti kód lefuttatása után a következő fog megjelenni:

```
Az első tömb anonim függvényekkel
paraméterek: 2.3445, M_PI
ezt nem tudom, mi: -1.6291725057799
az átfogó: 3.9199852871011
b*a^2 = 4.8103313314525
min(b^2+a, a^2,b) = 8.6382729035898
ln(a/b) = 0.27122299212594

A második tömb anonim függvényekkel
** "Akkor csillogó és fényes volt." és "Akkor este volt."
** nekem azonosnak tunik! (az elso 3 karakter utan)
CRCs: -725381282 , 1908338681
hasonló(a,b) = 11(45.833333333333%)
```

Valószínűleg az anonim függvények legáltalánosabb felhasználási módja az ún. 'callback' függvények létrehozása, pl. az array_walk() vagy az usort() számára.

Példa 3. Anonim függvények használata 'callback' függvényként

```
$nevelok = array("a ","egy ","az ","ez ");
array_walk($nevelok, create_function('&$v,$k','$v = $v."mangó";'));
print_r($av);  // PHP 3-soknak var_dump()
// kiírja, hogy :
// Array
// (
//   [0] => a mangó
//   [1] => egy mangó
//   [2] => az mangó
//   [3] => ez mangó
// )

// szrtinggel teli tömb a rövidebbtől a hosszabb felé rendezve
$sv = array("rovid","hosszabb","egy nagy string","ez valami sztring dolog");
print_r($sv);
```

```
// kiírja, hogy:
// Array
// (
//   [0] => rovid
//   [1] => hosszabb
//   [2] => egy nagy sztring
//   [3] => ez valami sztring dolog
// )

// hosszúság szerinti csökkenő sorrendben rendezve:
usort($sv, create_function('$a,$b','return strlen($b) - strlen($a);'));
print_r($sv);
// kiírja, hogy:
// Array
// (
//   [0] => ez valami sztring dolog
//   [1] => egy nagy sztring
//   [2] => hosszabb
//   [3] => rovid
// )
```

func_get_arg (PHP 4)

Visszaadja a paraméter lista egy elemét

mixed **func_get_arg** (int arg_num) \linebreak

Visszadja a függvénynek átadott paraméterlista *arg_num*. helyén álló paramétert. A paraméterek számozása 0-tól kezdődik. A **func_get_arg()** "nem fatális" hibát (warning) jelez, ha függvényen kívül hívod meg.

Ha az *arg_num* a pillanatnyilag átadott paraméterek számánál nagyobb, akkor egy "nem fatális" hiba (warning) generálódik és a **func_get_arg()** FALSE-szal tér vissza.

```
<?php
function ize() {
     $parameterek_szama = func_num_args();
     echo "argumentumok száma: {$parameterek_szama}<br/>\n";
     if ($parameterek_szama >= 2) {
     echo "A második paraméter: " . func_get_arg (1) . "<br/>\n";
     }
}

ize (1, 2, 3);
?>
```

A **func_get_arg()** a func_num_args() és a func_get_args() függvényekkel együtt olyan függvények írását segítik elő, amelyek különböző hosszúságú paraméterlistákat képesek kezelni.

Megjegyzés: Ez a függvény PHP 4-től használható.

func_get_args (PHP 4)

Visszatér a függvénynek átadott paramétereket tartalmazó tömbbel

array **func_get_args** (void) \linebreak

Visszaad egy tömböt az aktuális függvénynek átadott paraméterekkel. A **func_get_args()** "nem fatális" hibát (warning) jelez, ha függvénydefiníción kívül hívod meg.

```
<?php
function ize() {
    $pm_szama = func_num_args();
    echo "A parameterek szama: {$pm_szama}<br/>\n";
    if ($pm_szama >= 2) {
    echo "A masodik parameter: " . func_get_arg (1) . "<br/>\n";
    }
    $pm_lista = func_get_args();
    for ($i = 0; $i < $pm_szama; $i++) {
    echo "$i. parameter : " . $pm_lista[$i] . "<br/>\n";
    }
}

ize (1, 2, 3);
?>
```

A **func_get_args()**-t a func_get_arg() és func_num_args() függvényekkel együtt használva a felhasználói függvények is képesek kezelni a változó hosszúságú paraméterlistákat.

Megjegyzés: Ez a függvény PHP 4-től használható.

func_num_args (PHP 4)

Visszaadja a függvénynek átadott paraméterek számát

int **func_num_args** (void) \linebreak

Visszaadja az aktuális függvénynek átadott paraméterek számát. A **func_num_args()** "nem fatális" hibát (warning) jelez, ha függvénydefiníción kívül hívod meg.

```
<?php
function ize() {
    $pm_szama = func_num_args();
    echo "A parameterek szama: {$pm_szama}\n";
}

ize (1, 2, 3);    // kiírja, hogy : 'A parameterek szama: 3'
?>
```

A **func_num_args()**-t a func_get_arg() és func_get_args() függvényekkel együtt használva a felhasználói függvények is képesek kezelni a változó hosszúságú paraméterlistákat.

function_exists (PHP 3>= 3.0.7, PHP 4)

TRUE-val tér vissza, ha az adott nevű függvény definiálva van

bool **function_exists** (string function_name) \linebreak

A beépített és a felhasználó által definiált függvények között megkeresi a *function_name* nevűt. Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

```
if (function_exists('imap_open')) {
    echo "IMAP függvények használhatók.<br/>\n";
} else {
    echo "IMAP függvények _nem_ használhatók.<br/>\n";
}
```

Vigyázat, mert egy függvény neve létezhet még akkor is, ha a függvény maga nem használható valamilyen konfigurációs vagy fordítási beálltás következtében. Ilyenek például a képkezelő függvéyek. Szintén fontos megjegyezni, hogy a **function_exists()** FALSE értéket fog adni a nyelvi elemekre, amilyen például az include_once() vagy az echo().

Lásd még: method_exists() és get_defined_functions()!

get_defined_functions (PHP 4 >= 4.0.4)

visszatér az összes definiált függvény nevével

array **get_defined_functions** (void) \linebreak

Ez a függvény egy többdimenziós tömböt ad vissza, amelyben mind a beépített mind a felhasználó által definiált függvények nevei szerepelnek. A beépített függvények nevei `$arr["internal"]`-n keresztül érhetők el, amelyeket a felhasználó definiált, azok pedig az `$arr["user"]`-n keresztül. (lásd a lenti példát!)

```
function az_en_sorom($id, $adat) {
    return "<tr><th>$id</th><td>$adat</td></tr>\n";
}

$arr = get_defined_functions();

print_r($arr);
```

valami hasonlót fog kírni:

```
Array
(
    [internal] => Array
        (
            [0] => zend_version
            [1] => func_num_args
            [2] => func_get_arg
            [3] => func_get_args
            [4] => strlen
            [5] => strcmp
            [6] => strncmp
            ...
            [750] => bcscale
            [751] => bccomp
        )

    [user] => Array
        (
            [0] => az_en_sorom
        )

)
```

Lásd még: get_defined_vars() és get_defined_constants()!

register_shutdown_function (PHP 3>= 3.0.4, PHP 4)

Regiszrálja a függvényt a szkript befejezésekor történő végrehajtásra

int **register_shutdown_function** (string func) \linebreak

A *func* nevű függvényt regisztrálja, hogy az majd a szkript feldolgozása után lefusson.

Ha egymás után többször is meghívásra kerül a **register_shutdown_function()**, akkor a paraméterként átadott függvények ebben a sorrendben lesznek regisztrálva és meghívva. Ha valamelyik regisztrált függvényben az exit()-re kerül a vezérlés, akkor a feldolgozás megszakad és az ezután következő regisztrált függvények nem kerülnek meghívásra.

Mivel a regisztrált "lezáró" függvények a kérés kiszolgálása (kimeneti tartalom elküldése) után kerülnek meghívásra , ezért nem lehet semmilyen adatot küldeni a kliensnek pl. echo() vagy print() függvényekkel, és a kimeneti tárolók tartalmát sem lehet olvasni ob_get_contents() segítségével.

> **Megjegyzés:** A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

register_tick_function (PHP 4 >= 4.0.3)

Regisztrálja a függvényt a tick eseményhez

void **register_tick_function** (string func [, mixed arg]) \linebreak

Regisztrálja a *func* nevű függvényt, hogy az meghívásra kerüljön minden egyes tick bekövetkeztekor.

unregister_tick_function (PHP 4 >= 4.0.3)

Megszünteti a függvény a tick bekövetkeztekor történő futtatását

void **unregister_tick_function** (string func [, mixed arg]) \linebreak

A *func* nevű függvényt kiiktatja, hogy az ne fusson le, amikor egy tick esemény bekövetkezik.

XXXV. GNU Gettext

A gettext függvények az NLS (Native Language Support) API-t valósítják meg, ami a PHP szkriptek nemzetközivé tételéhez használható. Lásd még a GNU gettext dokumentációt egy alaposabb leírásért ezen függvényekkel kapcsolatban.

bind_textdomain_codeset (PHP 4 >= 4.2.0)

Specify the character encoding in which the messages from the DOMAIN message catalog will be returned

string **bind_textdomain_codeset** (string domain, string codeset) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

bindtextdomain (PHP 3>= 3.0.7, PHP 4)

Beállítja egy domain elérési útját

string **bindtextdomain** (string domain, string directory) \linebreak

A **bindtextdomain()** függvény beállítja egy domain elérési útját.

dcgettext (PHP 3>= 3.0.7, PHP 4)

Felülbírálja a domain-t egy lekérés erejéig

string **dcgettext** (string domain, string message, int category) \linebreak

Ez a függvény lehetőséget ad az aktuális domain felülbírálására egy lekérés erejéig. Egy kategória megadására is módot ad.

dcngettext (PHP 4 >= 4.2.0)

Plural version of dcgettext

string **dcngettext** (string domain, string msgid1, string msgid2, int n, int category) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

dgettext (PHP 3>= 3.0.7, PHP 4)

Az aktuális domain felülbírálása

string **dgettext** (string domain, string message) \linebreak

A **dgettext()** függvény lehetőséget ad az aktuális domain felülbírálására egy lekérés erejéig.

dngettext (PHP 4 >= 4.2.0)

Plural version of dgettext

string **dngettext** (string domain, string msgid1, string msgid2, int n) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

gettext (PHP 3>= 3.0.7, PHP 4)

Egy üzenet lekérése az aktuális domain-ben

string **gettext** (string message) \linebreak

Ez a függvény visszaadja a lefordított stringet, ha található ilyen a fordítási táblában. Egyébként az átadott stringet változtatás adja vissza. Haználhatod az aláhúzás karaktert, mint e függvény alias-át.

Példa 1. gettext() ellenőrzés

```
<?php
// Németre állítjuk a nyelvet
putenv ("LANG=de");

// A fordítási táblák elérésnek megadása
bindtextdomain ("enPHPAlkalmazasom", "./locale");

// Domain kiválasztása
textdomain ("enPHPAlkalmazasom");

// Szöveges üzenet kiírása
print (gettext ("Welcome to My PHP Application"));
?>
```

ngettext (PHP 4 >= 4.2.0)

Plural version of gettext

string **ngettext** (string msgid1, string msgid2, int n) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

textdomain (PHP 3>= 3.0.7, PHP 4)

Beállítja az alapértelmezett domain-t

string **textdomain** (string text_domain) \linebreak

Ez a függvény beállítja, hogy melyik domain-ben kell keresni a gettext() hívások esetén. Az előző beállítást adja vissza. Ha nem adsz át paramétert, az aktuális beállításokkal tér vissza.

XXXVI. GMP functions

Bevezetés

These functions allow you to work with arbitrary-length integers using the GNU MP library.

These functions have been added in PHP 4.0.4.

> **Megjegyzés:** Most GMP functions accept GMP number arguments, defined as `resource` below. However, most of these functions will also accept numeric and string arguments, given that it is possible to convert the latter to a number. Also, if there is a faster function that can operate on integer arguments, it would be used instead of the slower function when the supplied arguments are integers. This is done transparently, so the bottom line is that you can use integers in every function that expects GMP number. See also the gmp_init() function.

> **Figyelem**
>
> If you want to explicitly specify a large integer, specify it as a string. If you don't do that, PHP will interpret the integer-literal first, possibly resulting in loss of precision, even before `GMP` comes into play.

Követelmények

You can download the GMP library from http://www.swox.com/gmp/. This site also has the GMP manual available.

You will need GMP version 2 or better to use these functions. Some functions may require more recent version of the GMP library.

Telepítés

In order to have these functions available, you must compile PHP with GMP support by using the

`--with-gmp` option.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előre definiált állandók

Az itt listázott állandókat ez a kiterjesztés definiálja, és csak akkor elérhetőek, ha az adott kiterjesztés be van fordítva a PHP-be, vagy dinamikusan betöltött.

`GMP_ROUND_ZERO` (integer)

`GMP_ROUND_PLUSINF` (integer)

`GMP_ROUND_MINUSINF` (integer)

Példák

Példa 1. Factorial function using GMP

```
<?php
function fact ($x) {
    if ($x <= 1)
        return 1;
    else
        return gmp_mul ($x, fact ($x-1));
}

print gmp_strval (fact (1000)) . "\n";
?>
```

This will calculate factorial of 1000 (pretty big number) very fast.

Lásd még

More mathmatical functions can be found in the sections BCMath Arbitrary Precision Mathematics Functions and Mathematical Functions.

gmp_abs (PHP 4 >= 4.0.4)

Absolute value

resource **gmp_abs** (resource a) \linebreak

Returns absolute value of *a*.

gmp_add (PHP 4 >= 4.0.4)

Add numbers

resource **gmp_add** (resource a, resource b) \linebreak

Add two GMP numbers. The result will be a GMP number representing the sum of the arguments.

gmp_and (PHP 4 >= 4.0.4)

Logical AND

resource **gmp_and** (resource a, resource b) \linebreak

Calculates logical AND of two GMP numbers.

gmp_clrbit (PHP 4 >= 4.0.4)

Clear bit

resource **gmp_clrbit** (resource &a, int index) \linebreak

Clears (sets to 0) bit *index* in *a*.

gmp_cmp (PHP 4 >= 4.0.4)

Compare numbers

int **gmp_cmp** (resource a, resource b) \linebreak

Returns a positive value if `a > b`, zero if `a = b` and negative value if `a < b`.

gmp_com (PHP 4 >= 4.0.4)

Calculates one’s complement of a

resource **gmp_com** (resource a) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

gmp_div_q (PHP 4 >= 4.0.4)

Divide numbers

resource **gmp_div_q** (resource a, resource b [, int round]) \linebreak

Divides *a* by *b* and returns the integer result. The result rounding is defined by the *round*, which can have the following values:

- *GMP_ROUND_ZERO*: The result is truncated towards 0.
- *GMP_ROUND_PLUSINF*: The result is rounded towards `+infinity`.
- *GMP_ROUND_MINUSINF*: The result is rounded towards `-infinity`.

This function can also be called as gmp_div().

See also gmp_div_r(), gmp_div_qr()

gmp_div_qr (PHP 4 >= 4.0.4)

Divide numbers and get quotient and remainder

array **gmp_div_qr** (resource n, resource d [, int round]) \linebreak

The function divides *n* by *d* and returns array, with the first element being `[n/d]` (the integer result of the division) and the second being `(n - [n/d] * d)` (the remainder of the division).

See the gmp_div_q() function for description of the *round* argument.

Példa 1. Division of GMP numbers

```
<?php
    $a = gmp_init ("0x41682179fbf5");
    $res = gmp_div_qr ($a, "0xDEFE75");
    printf("Result is: q - %s, r - %s",
            gmp_strval ($res[0]), gmp_strval ($res[1]));
?>
```

See also gmp_div_q(), gmp_div_r().

gmp_div_r (PHP 4 >= 4.0.4)

Remainder of the division of numbers

resource **gmp_div_r** (resource n, resource d [, int round]) \linebreak

Calculates remainder of the integer division of *n* by *d*. The remainder has the sign of the *n* argument, if not zero.

See the gmp_div_q() function for description of the *round* argument.

See also gmp_div_q(), gmp_div_qr()

gmp_div (PHP 4 >= 4.0.4)

Divide numbers

resource **gmp_div** (resource a, resource b) \linebreak

This function is an alias to gmp_div_q().

gmp_divexact (PHP 4 >= 4.0.4)

Exact division of numbers

resource **gmp_divexact** (resource n, resource d) \linebreak

Divides *n* by *d*, using fast "exact division" algorithm. This function produces correct results only when it is known in advance that *d* divides *n*.

gmp_fact (PHP 4 >= 4.0.4)

Factorial

resource **gmp_fact** (int a) \linebreak

Calculates factorial (a!) of *a*.

gmp_gcd (PHP 4 >= 4.0.4)

Calculate GCD

resource **gmp_gcd** (resource a, resource b) \linebreak

Calculate greatest common divisor of *a* and *b*. The result is always positive even if either of, or both, input operands are negative.

gmp_gcdext (PHP 4 >= 4.0.4)

Calculate GCD and multipliers

array **gmp_gcdext** (resource a, resource b) \linebreak

Calculates g, s, and t, such that `a*s + b*t = g = gcd(a,b)`, where gcd is the greatest common divisor. Returns an array with respective elements g, s and t.

gmp_hamdist (PHP 4 >= 4.0.4)

Hamming distance

int **gmp_hamdist** (resource a, resource b) \linebreak

Returns the hamming distance between *a* and *b*. Both operands should be non-negative.

gmp_init (PHP 4 >= 4.0.4)

Create GMP number

resource **gmp_init** (mixed number) \linebreak

Creates a GMP number from an integer or string. String representation can be decimal or hexadecimal. In the latter case, the string should start with `0x`.

Példa 1. Creating GMP number

```
<?php
    $a = gmp_init (123456);
    $b = gmp_init ("0xFFFFDEBACDFEDF7200");
?>
```

Megjegyzés: It is not necessary to call this function if you want to use integer or string in place of GMP number in GMP functions, like gmp_add(). Function arguments are automatically converted to GMP numbers, if such conversion is possible and needed, using the same rules as **gmp_init()**.

gmp_intval (PHP 4 >= 4.0.4)

Convert GMP number to integer

int **gmp_intval** (resource gmpnumber) \linebreak

This function allows to convert GMP number to integer.

> **Figyelem**
>
> This function returns a useful result only if the number actually fits the PHP integer (i.e., signed long type). If you want just to print the GMP number, use gmp_strval().

gmp_invert (PHP 4 >= 4.0.4)

Inverse by modulo

resource **gmp_invert** (resource a, resource b) \linebreak

Computes the inverse of *a* modulo *b*. Returns FALSE if an inverse does not exist.

gmp_jacobi (PHP 4 >= 4.0.4)

Jacobi symbol

int **gmp_jacobi** (resource a, resource p) \linebreak

Computes Jacobi symbol (http://www.utm.edu/research/primes/glossary/JacobiSymbol.html) of *a* and *p*. *p* should be odd and must be positive.

gmp_legendre (PHP 4 >= 4.0.4)

Legendre symbol

int **gmp_legendre** (resource a, resource p) \linebreak

Compute the Legendre symbol (http://primes.utm.edu/glossary/page.php/LegendreSymbol.html) of *a* and *p*. *p* should be odd and must be positive.

gmp_mod (PHP 4 >= 4.0.4)

Modulo operation

resource **gmp_mod** (resource n, resource d) \linebreak

Calculates *n* modulo *d*. The result is always non-negative, the sign of *d* is ignored.

gmp_mul (PHP 4 >= 4.0.4)

Multiply numbers

resource **gmp_mul** (resource a, resource b) \linebreak

Multiplies *a* by *b* and returns the result.

gmp_neg (PHP 4 >= 4.0.4)

Negate number

resource **gmp_neg** (resource a) \linebreak

Returns -*a*.

gmp_or (PHP 4 >= 4.0.4)

Logical OR

resource **gmp_or** (resource a, resource b) \linebreak

Calculates logical inclusive OR of two GMP numbers.

gmp_perfect_square (PHP 4 >= 4.0.4)

Perfect square check

bool **gmp_perfect_square** (resource a) \linebreak

Returns TRUE if *a* is a perfect square, FALSE otherwise.

See also: gmp_sqrt(), gmp_sqrtrm().

gmp_popcount (PHP 4 >= 4.0.4)

Population count

int **gmp_popcount** (resource a) \linebreak

Return the population count of *a*.

gmp_pow (PHP 4 >= 4.0.4)

Raise number into power

resource **gmp_pow** (resource base, int exp) \linebreak

Raise *base* into power *exp*. The case of 0^0 yields 1. *exp* cannot be negative.

gmp_powm (PHP 4 >= 4.0.4)

Raise number into power with modulo

resource **gmp_powm** (resource base, resource exp, resource mod) \linebreak

Calculate (*base* raised into power *exp*) modulo *mod*. If *exp* is negative, result is undefined.

gmp_prob_prime (PHP 4 >= 4.0.4)

Check if number is "probably prime"

int **gmp_prob_prime** (resource a [, int reps]) \linebreak

If this function returns 0, *a* is definitely not prime. If it returns 1, then *a* is "probably" prime. If it returns 2, then *a* is surely prime. Reasonable values of *reps* vary from 5 to 10 (default being 10); a higher value lowers the probability for a non-prime to pass as a "probable" prime.

The function uses Miller-Rabin’s probabilistic test.

gmp_random (PHP 4 >= 4.0.4)

Random number

resource **gmp_random** (int limiter) \linebreak

Generate a random number. The number will be between *limiter* and zero in value. If *limiter* is negative, negative numbers are generated.

gmp_scan0 (PHP 4 >= 4.0.4)

Scan for 0

int **gmp_scan0** (resource a, int start) \linebreak

Scans *a*, starting with bit *start*, towards more significant bits, until the first clear bit is found. Returns the index of the found bit.

gmp_scan1 (PHP 4 >= 4.0.4)

Scan for 1

int **gmp_scan1** (resource a, int start) \linebreak

Scans *a*, starting with bit *start*, towards more significant bits, until the first set bit is found. Returns the index of the found bit.

gmp_setbit (PHP 4 >= 4.0.4)

Set bit

resource **gmp_setbit** (resource &a, int index [, bool set_clear]) \linebreak

Sets bit *index* in *a*. *set_clear* defines if the bit is set to 0 or 1. By default the bit is set to 1.

gmp_sign (PHP 4 >= 4.0.4)

Sign of number

int **gmp_sign** (resource a) \linebreak

Return sign of *a* - 1 if *a* is positive and -1 if it's negative.

gmp_sqrt (PHP 4 >= 4.0.4)

Square root

resource **gmp_sqrt** (resource a) \linebreak

Calculates square root of *a*.

gmp_sqrtrm (unknown)

Square root with remainder

array **gmp_sqrtrm** (resource a) \linebreak

Returns array where first element is the integer square root of *a* (see also gmp_sqrt()), and the second is the remainder (i.e., the difference between *a* and the first element squared).

gmp_strval (PHP 4 >= 4.0.4)

Convert GMP number to string

string **gmp_strval** (resource gmpnumber [, int base]) \linebreak

Convert GMP number to string representation in base *base*. The default base is 10. Allowed values for the base are from 2 to 36.

Példa 1. Converting a GMP number to a string

```
<?php
    $a = gmp_init("0x41682179fbf5");
    printf ("Decimal: %s, 36-based: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_sub (PHP 4 >= 4.0.4)

Subtract numbers

resource **gmp_sub** (resource a, resource b) \linebreak

Subtracts *b* from *a* and returns the result.

gmp_xor (PHP 4 >= 4.0.4)

Logical XOR

resource **gmp_xor** (resource a, resource b) \linebreak

Calculates logical exclusive OR (XOR) of two GMP numbers.

XXXVII. HTTP functions

Bevezetés

These functions let you manipulate the output sent back to the remote browser right down to the HTTP protocol level.

Követelmények

Az itt leírt függvények a standard modulban találhatóak, ami mindig rendelkezésre áll.

Telepítés

Semmilyen telepítés nem szükséges ezen függvények használatához, a PHP alapelemei.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

header (PHP 3, PHP 4)

Send a raw HTTP header

int **header** (string string [, bool replace [, int http_reponse_code]]) \linebreak

header() is used to send raw HTTP headers. See the HTTP/1.1 specification (http://www.w3.org/Protocols/rfc2616/rfc2616) for more information on HTTP headers.

The optional *replace* parameter indicates whether the header should replace a previous similar header, or add a second header of the same type. By default it will replace, but if you pass in FALSE as the second argument you can force multiple headers of the same type. For example:

```
header('WWW-Authenticate: Negotiate');
header('WWW-Authenticate: NTLM', FALSE);
```

The second optional *http_response_code* force the HTTP response code to the specified value. (This parameter is available in PHP 4.3.0 and higher.)

There are two special-case header calls. The first is a header that starts with the string "HTTP/" (case is not significant), which will be used to figure out the HTTP status code to send. For example, if you have configured Apache to use a PHP script to handle requests for missing files (using the ErrorDocument directive), you may want to make sure that your script generates the proper status code.

```
<?php
  header("HTTP/1.0 404 Not Found");
?>
```

Megjegyzés: The HTTP status header line will always be the first sent to the client, regardless of the actual **header()** call beeing the first or not. The status may be overridden by calling **header()** with a new status line at any time unless the HTTP headers have already been sent.

Megjegyzés: In PHP 3, this only works when PHP is compiled as an Apache module. You can achieve the same effect using the Status header.

```
header("Status: 404 Not Found");
```

The second special case is the "Location:" header. Not only does it send this header back to the browser, but it also returns a `REDIRECT` (302) status code to the browser unless some `3xx` status code has already been set.

```
header("Location: http://www.example.com/"); /* Redirect browser */
exit;                 /* Make sure that code below does
                         not get executed when we redirect. */
```

> **Megjegyzés:** HTTP/1.1 requires an absolute URI as argument to Location: (http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.30) including the scheme, hostname and absolute path, but some clients accept relative URIs. You can usually use `$_SERVER['HTTP_HOST']`, `$_SERVER['PHP_SELF']` and dirname() to make an absolute URI from a relative one yourself:
>
> ```
> header("Location: http://".$_SERVER['HTTP_HOST']
>                       .dirname($_SERVER['PHP_SELF'])
>                       ."/".$relative_url);
> ```

PHP scripts often generate dynamic content that must not be cached by the client browser or any proxy caches between the server and the client browser. Many proxies and clients can be forced to disable caching with

```
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");    // Date in the past
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
                                                     // always modified
header("Cache-Control: no-store, no-cache, must-revalidate");  // HTTP/1.1
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");                          // HTTP/1.0
```

> **Megjegyzés:** You may find that your pages aren't cached even if you don't output all of the headers above. There are a number of options that users may be able to set for their browser that change its default caching behavior. By sending the headers above, you should override any settings that may otherwise cause the output of your script to be cached.
>
> Additionally, session_cache_limiter() and the `session.cache_limiter` configuration setting can be used to automatically generate the correct caching-related headers when sessions are being used.

Remember that **header()** must be called before any actual output is sent, either by normal HTML tags, blank lines in a file, or from PHP. It is a very common error to read code with include(), or require(), functions, or another file access function, and have spaces or empty lines that are output before **header()** is called. The same problem exists when using a single PHP/HTML file.

```
<?php header ("Content-type: audio/x-pn-realaudio"); ?>
// Broken, note the blank line above
```

> **Megjegyzés:** In PHP 4, you can use output buffering to get around this problem, with the overhead of all of your output to the browser being buffered in the server until you send it. You can do this by calling ob_start() and ob_end_flush() in your script, or setting the `output_buffering` configuration directive on in your `php.ini` or server configuration files.

If you want the user to be prompted to save the data you are sending, such as a generated PDF file, you can use the Content-Disposition (http://www.ietf.org/rfc/rfc2183.txt) header to supply a recommended filename and force the browser to display the save dialog.

```
<?php
header("Content-type: application/pdf");
header("Content-Disposition: attachment; filename=downloaded.pdf");

/* ... output pdf file ... */
```

> **Megjegyzés:** There is a bug in Microsoft Internet Explorer 4.01 that prevents this from working. There is no workaround. There is also a bug in Microsoft Internet Explorer 5.5 that interferes with this, which can be resolved by upgrading to Service Pack 2 or later.

> **Megjegyzés:** If safe mode is enabled the uid of the script is added to the `realm` part of the `WWW-Authenticate` header if you set this header (used for HTTP Authentication).

See also headers_sent(), setcookie(), and the section on HTTP authentication.

headers_sent (PHP 3>= 3.0.8, PHP 4)

Returns `TRUE` if headers have been sent

bool **headers_sent** (void) \linebreak

This function returns `TRUE` if the HTTP headers have already been sent, `FALSE` otherwise.

See also header()

setcookie (PHP 3, PHP 4)

Send a cookie

int **setcookie** (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]]) \line-break

setcookie() defines a cookie to be sent along with the rest of the header information. Cookies must be sent *before* any other headers are sent (this is a restriction of cookies, not PHP). This requires you to place calls to this function before any `<html>` or `<head>` tags.

All the arguments except the *`name`* argument are optional. If only the name argument is present, the cookie by that name will be deleted from the remote client. You may also replace any argument with an empty string (*""*) in order to skip that argument. The *`expire`* and *`secure`* arguments are integers and cannot be skipped with an empty string. Use a zero (*0*) instead. The *`expire`* argument is a regular Unix time integer as returned by the time() or mktime() functions. The *`secure`* indicates that the cookie should only be transmitted over a secure HTTPS connection.

Once the cookies have been set they can be accessed on the next page load with the `$_COOKIE` array (which used to be called `$HTTP_COOKIE_VARS` in PHP versions prior to 4.1.0).

Common Pitfalls:

- Cookies will not become visible until the next loading of a page that the cookie should be visible for.
- Cookies must be deleted with the same parameters as they were set with.

In PHP 3, multiple calls to **setcookie()** in the same script will be performed in reverse order. If you are trying to delete one cookie before inserting another you should put the insert before the delete. In PHP 4, multiple calls to **setcookie()** are performed in the order called.

Some examples follow how to send cookies:

Példa 1. setcookie() send examples

```
setcookie ("TestCookie", $value);
setcookie ("TestCookie", $value,time()+3600);  /* expire in 1 hour */
setcookie ("TestCookie", $value,time()+3600, "/~rasmus/", ".utoronto.ca", 1);
```

When deleting a cookie you should assure that the expiration date is in the past, to trigger the removal mechanism in your browser. Examples follow how to delete cookies sent in previous example:

Példa 2. setcookie() delete examples

```
// set the expiration date to one hour ago
setcookie ("TestCookie", "", time() - 3600);
setcookie ("TestCookie", "", time() - 3600, "/~rasmus/", ".utoronto.ca", 1);
```

Note that the value portion of the cookie will automatically be urlencoded when you send the cookie, and when it is received, it is automatically decoded and assigned to a variable by the same name as the cookie name. To see the contents of our test cookie in a script, simply use one of the following examples:

```
echo $TestCookie;
echo $_COOKIE["TestCookie"];
```

You may also set array cookies by using array notation in the cookie name. This has the effect of setting as many cookies as you have array elements, but when the cookie is received by your script, the values are all placed in an array with the cookie's name:

```
setcookie ("cookie[three]", "cookiethree");
setcookie ("cookie[two]", "cookietwo");
setcookie ("cookie[one]", "cookieone");
if (isset ($cookie)) {
    while (list ($name, $value) = each ($cookie)) {
        echo "$name == $value<br>\n";
    }
}
```

For more information on cookies, see Netscape's cookie specification at http://www.netscape.com/newsref/std/cookie_spec.html.

Microsoft Internet Explorer 4 with Service Pack 1 applied does not correctly deal with cookies that have their path parameter set.

Netscape Communicator 4.05 and Microsoft Internet Explorer 3.x appear to handle cookies incorrectly when the path and time are not set.

XXXVIII. Hyperwave functions

Introduction

Hyperwave has been developed at IICM (http://www.iicm.edu/) in Graz. It started with the name Hyper-G and changed to Hyperwave when it was commercialised (If I remember properly it was in 1996).

Hyperwave is not free software. The current version, 4.1, is available at www.hyperwave.com (http://www.hyperwave.com/). A time limited version can be ordered for free (30 days).

Hyperwave is an information system similar to a database (HIS, Hyperwave Information Server). Its focus is the storage and management of documents. A document can be any possible piece of data that may as well be stored in file. Each document is accompanied by its object record. The object record contains meta data for the document. The meta data is a list of attributes which can be extended by the user. Certain attributes are always set by the Hyperwave server, other may be modified by the user. An attribute is a name/value pair of the form name=value. The complete object record contains as many of those pairs as the user likes. The name of an attribute does not have to be unique, e.g. a title may appear several times within an object record. This makes sense if you want to specify a title in several languages. In such a case there is a convention, that each title value is preceded by the two letter language abbreviation followed by a colon, e.g. ’en:Title in English’ or ’ge:Titel in deutsch’. Other attributes like a description or keywords are potential candidates. You may also replace the language abbreviation by any other string as long as it separated by colon from the rest of the attribute value.

Each object record has native a string representation with each name/value pair separated by a newline. The Hyperwave extension also knows a second representation which is an associated array with the attribute name being the key. Multilingual attribute values itself form another associated array with the key being the language abbreviation. Actually any multiple attribute forms an associated array with the string left to the colon in the attribute value being the key. (This is not fully implemented. Only the attributes Title, Description and Keyword are treated properly yet.)

Besides the documents, all hyper links contained in a document are stored as object records as well. Hyper links which are in a document will be removed from it and stored as individual objects, when the document is inserted into the database. The object record of the link contains information about where it starts and where it ends. In order to gain the original document you will have to retrieve the plain document without the links and the list of links and reinsert them (The functions hw_pipedocument() and hw_gettext() do this for you. The advantage of separating links from the document is obvious. Once a document to which a link is pointing to changes its name, the link can easily be modified accordingly. The document containing the link is not affected at all. You may even add a link to a document without modifying the document itself.

Saying that hw_pipedocument() and hw_gettext() do the link insertion automatically is not as simple as it sounds. Inserting links implies a certain hierarchy of the documents. On a web server this is given by the file system, but Hyperwave has its own hierarchy and names do not reflect the position of an object in that hierarchy. Therefore creation of links first of all requires a mapping from the Hyperwave hierarchy and namespace into a web hierarchy respective web namespace. The fundamental difference between Hyperwave and the web is the clear distinction between names and hierarchy in Hyperwave. The name does not contain any information about the objects position in the hierarchy. In the web the name also contains the information on where the object is located in the hierarchy. This leads to two possibles ways of mapping. Either the Hyperwave hierarchy and name of the Hyperwave object is reflected in the URL or the name only. To make things simple the second approach is used. Hyperwave object with name ’my_object’ is mapped to ’http://host/my_object’

disregarding where it resides in the Hyperwave hierarchy. An object with name ’parent/my_object’ could be the child of ’my_object’ in the Hyperwave hierarchy, though in a web namespace it appears to be just the opposite and the user might get confused. This can only be prevented by selecting reasonable object names.

Having made this decision a second problem arises. How do you involve PHP? The URL http://host/my_object will not call any PHP script unless you tell your web server to rewrite it to e.g. ’http://host/php3_script/my_object’ and the script ’php3_script’ evaluates the $PATH_INFO variable and retrieves the object with name ’my_object’ from the Hyperwave server. Their is just one little drawback which can be fixed easily. Rewriting any URL would not allow any access to other document on the web server. A PHP script for searching in the Hyperwave server would be impossible. Therefore you will need at least a second rewriting rule to exclude certain URLS like all e.g. starting with http://host/Hyperwave. This is basically sharing of a namespace by the web and Hyperwave server.

Based on the above mechanism links are insert into documents.

It gets more complicated if PHP is not run as a server module or CGI script but as a standalone application e.g. to dump the content of the Hyperwave server on a CD-ROM. In such a case it makes sense to retain the Hyperwave hierarchy and map in onto the file system. This conflicts with the object names if they reflect its own hierarchy (e.g. by choosing names including ’/’). Therefore ’/’ has to be replaced by another character, e.g. ’_’. to be continued.

The network protocol to communicate with the Hyperwave server is called HG-CSP (http://www.hyperwave.com/7.17-hg-prot) (Hyper-G Client/Server Protocol). It is based on messages to initiate certain actions, e.g. get object record. In early versions of the Hyperwave Server two native clients (Harmony, Amadeus) were provided for communication with the server. Those two disappeared when Hyperwave was commercialised. As a replacement a so called wavemaster was provided. The wavemaster is like a protocol converter from HTTP to HG-CSP. The idea is to do all the administration of the database and visualisation of documents by a web interface. The wavemaster implements a set of placeholders for certain actions to customise the interface. This set of placeholders is called the PLACE Language. PLACE lacks a lot of features of a real programming language and any extension to it only enlarges the list of placeholders. This has led to the use of JavaScript which IMO does not make life easier.

Adding Hyperwave support to PHP should fill in the gap of a missing programming language for interface customisation. It implements all the messages as defined by the HG-CSP but also provides more powerful commands to e.g. retrieve complete documents.

Hyperwave has its own terminology to name certain pieces of information. This has widely been taken over and extended. Almost all functions operate on one of the following data types.

- object ID: An unique integer value for each object in the Hyperwave server. It is also one of the attributes of the object record (ObjectID). Object ids are often used as an input parameter to specify an object.
- object record: A string with attribute-value pairs of the form attribute=value. The pairs are separated by a carriage return from each other. An object record can easily be converted into an object array with **hw_object2array()**. Several functions return object records. The names of those functions end with obj.
- object array: An associated array with all attributes of an object. The key is the attribute name. If an attribute occurs more than once in an object record it will result in another indexed or associated array. Attributes which are language depended (like the title, keyword, description) will form an associated array with the key set to the language abbreviation. All other multiple

attributes will form an indexed array. PHP functions never return object arrays.

- hw_document: This is a complete new data type which holds the actual document, e.g. HTML, PDF etc. It is somewhat optimised for HTML documents but may be used for any format.

Several functions which return an array of object records do also return an associated array with statistical information about them. The array is the last element of the object record array. The statistical array contains the following entries:

Hidden

Number of object records with attribute PresentationHints set to Hidden.

CollectionHead

Number of object records with attribute PresentationHints set to CollectionHead.

FullCollectionHead

Number of object records with attribute PresentationHints set to FullCollectionHead.

CollectionHeadNr

Index in array of object records with attribute PresentationHints set to CollectionHead.

FullCollectionHeadNr

Index in array of object records with attribute PresentationHints set to FullCollectionHead.

Total

Total: Number of object records.

Integration with Apache

The Hyperwave extension is best used when PHP is compiled as an Apache module. In such a case the underlying Hyperwave server can be hidden from users almost completely if Apache uses its rewriting engine. The following instructions will explain this.

Since PHP with Hyperwave support built into Apache is intended to replace the native Hyperwave solution based on Wavemaster I will assume that the Apache server will only serve as a Hyperwave web interface. This is not necessary but it simplifies the configuration. The concept is quite simple. First of all you need a PHP script which evaluates the PATH_INFO variable and treats its value as the name of a Hyperwave object. Let's call this script 'Hyperwave'. The URL http://your.hostname/Hyperwave/name_of_object would than return the Hyperwave object with the name 'name_of_object'. Depending on the type of the object the script has to react accordingly. If it is a collection, it will probably return a list of children. If it is a document it will return the mime type and the content. A slight improvement can be achieved if the Apache rewriting engine is used. From the users point of view it would be more straight forward if the URL http://your.hostname/name_of_object would return the object. The rewriting rule is quite easy:

```
RewriteRule ^/(.*) /usr/local/apache/htdocs/HyperWave/$1 [L]
```

Now every URL relates to an object in the Hyperwave server. This causes a simple to solve problem. There is no way to execute a different script, e.g. for searching, than the ’Hyperwave’ script. This can be fixed with another rewriting rule like the following:

```
RewriteRule ^/hw/(.*) /usr/local/apache/htdocs/hw/$1 [L]
```

This will reserve the directory `/usr/local/apache/htdocs/hw` for additional scripts and other files. Just make sure this rule is evaluated before the one above. There is just a little drawback: all Hyperwave objects whose name starts with ’hw/’ will be shadowed. So, make sure you don’t use such names. If you need more directories, e.g. for images just add more rules or place them all in one directory. Finally, don’t forget to turn on the rewriting engine with

```
RewriteEngine on
```

My experiences have shown that you will need the following scripts:

- to return the object itself
- to allow searching
- to identify yourself
- to set your profile
- one for each additional function like to show the object attributes, to show information about users, to show the status of the server, etc.

Todo

There are still some things todo:

- The hw_InsertDocument has to be split into hw_insertobject() and **hw_putdocument()**.
- The names of several functions are not fixed, yet.
- Most functions require the current connection as its first parameter. This leads to a lot of typing, which is quite often not necessary if there is just one open connection. A default connection will improve this.
- Conversion form object record into object array needs to handle any multiple attribute.

hw_Array2Objrec (PHP 3>= 3.0.4, PHP 4)

convert attributes from object array to object record

strin **hw_array2objrec** (array object_array) \linebreak

Converts an *`object_array`* into an object record. Multiple attributes like 'Title' in different languages are treated properly.

See also hw_objrec2array().

hw_changeobject (PHP 3>= 3.0.3, PHP 4)

Changes attributes of an object (obsolete)

void **hw_changeobject** (int link, int objid, array attributes) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

hw_Children (PHP 3>= 3.0.3, PHP 4)

object ids of children

array **hw_children** (int connection, int objectID) \linebreak

Returns an array of object ids. Each id belongs to a child of the collection with ID *`objectID`*. The array contains all children both documents and collections.

hw_ChildrenObj (PHP 3>= 3.0.3, PHP 4)

object records of children

array **hw_childrenobj** (int connection, int objectID) \linebreak

Returns an array of object records. Each object record belongs to a child of the collection with ID *`objectID`*. The array contains all children both documents and collections.

hw_Close (PHP 3>= 3.0.3, PHP 4)

closes the Hyperwave connection

int **hw_close** (int connection) \linebreak

Returns FALSE if connection is not a valid connection index, otherwise TRUE. Closes down the connection to a Hyperwave server with the given connection index.

hw_Connect (PHP 3>= 3.0.3, PHP 4)

opens a connection

int **hw_connect** (string host, int port, string username, string password) \linebreak

Opens a connection to a Hyperwave server and returns a connection index on success, or FALSE if the connection could not be made. Each of the arguments should be a quoted string, except for the port number. The *`username`* and *`password`* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple connections open at once. Keep in mind, that the password is not encrypted.

See also hw_pconnect().

hw_connection_info (PHP 3>= 3.0.3, PHP 4)

Prints information about the connection to Hyperwave server

void **hw_connection_info** (int link) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

hw_Cp (PHP 3>= 3.0.3, PHP 4)

copies objects

int **hw_cp** (int connection, array object_id_array, int destination id) \linebreak

Copies the objects with object ids as specified in the second parameter to the collection with the id *`destination id`*.

The value return is the number of copied objects.

See also hw_mv().

hw_Deleteobject (PHP 3>= 3.0.3, PHP 4)

deletes object

int **hw_deleteobject** (int connection, int object_to_delete) \linebreak

Deletes the object with the given object id in the second parameter. It will delete all instances of the object.

Returns TRUE if no error occurs otherwise FALSE.

See also hw_mv().

hw_DocByAnchor (PHP 3>= 3.0.3, PHP 4)

object id object belonging to anchor

int **hw_docbyanchor** (int connection, int anchorID) \linebreak

Returns an th object id of the document to which *anchorID* belongs.

hw_DocByAnchorObj (PHP 3>= 3.0.3, PHP 4)

object record object belonging to anchor

string **hw_docbyanchorobj** (int connection, int anchorID) \linebreak

Returns an th object record of the document to which *anchorID* belongs.

hw_Document_Attributes (PHP 3>= 3.0.3, PHP 4)

object record of hw_document

string **hw_document_attributes** (int hw_document) \linebreak

Returns the object record of the document.

For backward compatibility, **hw_documentattributes()** is also accepted. This is deprecated, however.

See also hw_document_bodytag(), and hw_document_size().

hw_Document_BodyTag (PHP 3>= 3.0.3, PHP 4)

body tag of hw_document

string **hw_document_bodytag** (int hw_document) \linebreak

Returns the BODY tag of the document. If the document is an HTML document the BODY tag should be printed before the document.

See also hw_document_attributes(), and hw_document_size().

For backward compatibility, **hw_documentbodytag()** is also accepted. This is deprecated, however.

hw_Document_Content (PHP 3>= 3.0.3, PHP 4)

returns content of hw_document

string **hw_document_content** (int hw_document) \linebreak

Returns the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record.

See also hw_document_attributes(), hw_document_size(), and hw_document_setcontent().

hw_Document_SetContent (PHP 4)

sets/replaces content of hw_document

string **hw_document_setcontent** (int hw_document, string content) \linebreak

Sets or replaces the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record. If you provide this information in the content of the document too, the Hyperwave server will change the object record accordingly when the document is inserted. Probably not a very good idea. If this functions fails the document will retain its old content.

See also hw_document_attributes(), hw_document_size(), and hw_document_content().

hw_Document_Size (PHP 3>= 3.0.3, PHP 4)

size of hw_document

int **hw_document_size** (int hw_document) \linebreak

Returns the size in bytes of the document.

See also hw_document_bodytag(), and hw_document_attributes().

For backward compatibility, **hw_documentsize()** is also accepted. This is deprecated, however.

hw_dummy (PHP 3>= 3.0.3, PHP 4)

Hyperwave dummy function

string **hw_dummy** (int link, int id, int msgid) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

hw_EditText (PHP 3>= 3.0.3, PHP 4)

retrieve text document

int **hw_edittext** (int connection, int hw_document) \linebreak

Uploads the text document to the server. The object record of the document may not be modified while the document is edited. This function will only works for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also hw_pipedocument(), hw_free_document(), hw_document_bodytag(), hw_document_size(), hw_output_document(), hw_gettext().

hw_Error (PHP 3>= 3.0.3, PHP 4)

error number

int **hw_error** (int connection) \linebreak

Returns the last error number. If the return value is 0 no error has occurred. The error relates to the last command.

hw_ErrorMsg (PHP 3>= 3.0.3, PHP 4)

returns error message

string **hw_errormsg** (int connection) \linebreak

Returns a string containing the last error message or 'No Error'. If FALSE is returned, this function failed. The message relates to the last command.

hw_Free_Document (PHP 3>= 3.0.3, PHP 4)

frees hw_document

int **hw_free_document** (int hw_document) \linebreak

Frees the memory occupied by the Hyperwave document.

hw_GetAnchors (PHP 3>= 3.0.3, PHP 4)

object ids of anchors of document

array **hw_getanchors** (int connection, int objectID) \linebreak

Returns an array of object ids with anchors of the document with object ID *`objectID`*.

hw_GetAnchorsObj (PHP 3>= 3.0.3, PHP 4)

object records of anchors of document

array **hw_getanchorsobj** (int connection, int objectID) \linebreak

Returns an array of object records with anchors of the document with object ID *`objectID`*.

hw_GetAndLock (PHP 3>= 3.0.3, PHP 4)

return bject record and lock object

string **hw_getandlock** (int connection, int objectID) \linebreak

Returns the object record for the object with ID *`objectID`*. It will also lock the object, so other users cannot access it until it is unlocked.

See also hw_unlock(), and hw_getobject().

hw_GetChildColl (PHP 3>= 3.0.3, PHP 4)

object ids of child collections

array **hw_getchildcoll** (int connection, int objectID) \linebreak

Returns an array of object ids. Each object ID belongs to a child collection of the collection with ID *`objectID`*. The function will not return child documents.

See also hw_children(), and hw_getchilddoccoll().

hw_GetChildCollObj (PHP 3>= 3.0.3, PHP 4)

object records of child collections

array **hw_getchildcollobj** (int connection, int objectID) \linebreak

Returns an array of object records. Each object records belongs to a child collection of the collection with ID *objectID*. The function will not return child documents.

See also hw_childrenobj(), and hw_getchilddoccollobj().

hw_GetChildDocColl (PHP 3>= 3.0.3, PHP 4)

object ids of child documents of collection

array **hw_getchilddoccoll** (int connection, int objectID) \linebreak

Returns array of object ids for child documents of a collection.

See also hw_children(), and hw_getchildcoll().

hw_GetChildDocCollObj (PHP 3>= 3.0.3, PHP 4)

object records of child documents of collection

array **hw_getchilddoccollobj** (int connection, int objectID) \linebreak

Returns an array of object records for child documents of a collection.

See also hw_childrenobj(), and hw_getchildcollobj().

hw_GetObject (PHP 3>= 3.0.3, PHP 4)

object record

array **hw_getobject** (int connection, [int|array] objectID, string query) \linebreak

Returns the object record for the object with ID *objectID* if the second parameter is an integer. If the second parameter is an array of integer the function will return an array of object records. In such a case the last parameter is also evaluated which is a query string.

The query string has the following syntax:

<expr> ::= "(" <expr> ")" |

"!" <expr> | /* NOT */

<expr> "||" <expr> | /* OR */

<expr> "&&" <expr> | /* AND */

<attribute> <operator> <value>

<attribute> ::= /* any attribute name (Title, Author, DocumentType ...) */

<operator> ::= "=" | /* equal */

"<" | /* less than (string compare) */

">" | /* greater than (string compare) */

"~" /* regular expression matching */

The query allows to further select certain objects from the list of given objects. Unlike the other query functions, this query may use not indexed attributes. How many object records are returned depends on the query and if access to the object is allowed.

See also hw_getandlock(), and hw_getobjectbyquery().

hw_GetObjectByQuery (PHP 3>= 3.0.3, PHP 4)

search object

array **hw_getobjectbyquery** (int connection, string query, int max_hits) \linebreak

Searches for objects on the whole server and returns an array of object ids. The maximum number of matches is limited to *`max_hits`*. If *`max_hits`* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyqueryobj().

hw_GetObjectByQueryColl (PHP 3>= 3.0.3, PHP 4)

search object in collection

array **hw_getobjectbyquerycoll** (int connection, int objectID, string query, int max_hits) \linebreak

Searches for objects in collection with ID *`objectID`* and returns an array of object ids. The maximum number of matches is limited to *`max_hits`*. If *`max_hits`* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyquerycollobj().

hw_GetObjectByQueryCollObj (PHP 3>= 3.0.3, PHP 4)

search object in collection

array **hw_getobjectbyquerycollobj** (int connection, int objectID, string query, int max_hits) \linebreak

Searches for objects in collection with ID *`objectID`* and returns an array of object records. The maximum number of matches is limited to *`max_hits`*. If *`max_hits`* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyquerycoll().

hw_GetObjectByQueryObj (PHP 3>= 3.0.3, PHP 4)

search object

array **hw_getobjectbyqueryobj** (int connection, string query, int max_hits) \linebreak

Searches for objects on the whole server and returns an array of object records. The maximum number of matches is limited to *`max_hits`*. If *`max_hits`* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyquery().

hw_GetParents (PHP 3>= 3.0.3, PHP 4)

object ids of parents

array **hw_getparents** (int connection, int objectID) \linebreak

Returns an indexed array of object ids. Each object id belongs to a parent of the object with ID *`objectID`*.

hw_GetParentsObj (PHP 3>= 3.0.3, PHP 4)

object records of parents

array **hw_getparentsobj** (int connection, int objectID) \linebreak

Returns an indexed array of object records plus an associated array with statistical information about the object records. The associated array is the last entry of the returned array. Each object record belongs to a parent of the object with ID *`objectID`*.

hw_getrellink (PHP 3>= 3.0.3, PHP 4)

Get link from source to dest relative to rootid

string **hw_getrellink** (int link, int rootid, int sourceid, int destid) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

hw_GetRemote (PHP 3>= 3.0.3, PHP 4)

Gets a remote document

int **hw_getremote** (int connection, int objectID) \linebreak

Returns a remote document. Remote documents in Hyperwave notation are documents retrieved from an external source. Common remote documents are for example external web pages or queries in a database. In order to be able to access external sources throught remote documents Hyperwave introduces the HGI (Hyperwave Gateway Interface) which is similar to the CGI. Currently, only ftp, http-servers and some databases can be accessed by the HGI. Calling **hw_getremote()** returns the document from the external source. If you want to use this function you should be very familiar with HGIs. You should also consider to use PHP instead of Hyperwave to access external sources. Adding database support by a Hyperwave gateway should be more difficult than doing it in PHP.

See also hw_getremotechildren().

hw_GetRemoteChildren (PHP 3>= 3.0.3, PHP 4)

Gets children of remote document

int **hw_getremotechildren** (int connection, string object record) \linebreak

Returns the children of a remote document. Children of a remote document are remote documents itself. This makes sense if a database query has to be narrowed and is explained in Hyperwave Programmers' Guide. If the number of children is 1 the function will return the document itself formated by the Hyperwave Gateway Interface (HGI). If the number of children is greater than 1 it will return an array of object record with each maybe the input value for another call to **hw_getremotechildren()**. Those object records are virtual and do not exist in the Hyperwave server, therefore they do not have a valid object ID. How exactely such an object record looks like is up to the HGI. If you want to use this function you should be very familiar with HGIs. You should also consider to use PHP instead of Hyperwave to access external sources. Adding database support by a Hyperwave gateway should be more difficult than doing it in PHP.

See also hw_getremote().

hw_GetSrcByDestObj (PHP 3>= 3.0.3, PHP 4)

Returns anchors pointing at object

array **hw_getsrcbydestobj** (int connection, int objectID) \linebreak

Returns the object records of all anchors pointing to the object with ID *`objectID`*. The object can either be a document or an anchor of type destination.

See also hw_getanchors().

hw_GetText (PHP 3>= 3.0.3, PHP 4)

retrieve text document

int **hw_gettext** (int connection, int objectID [, mixed rootID/prefix]) \linebreak

Returns the document with object ID *`objectID`*. If the document has anchors which can be inserted, they will be inserted already. The optional parameter *`rootID/prefix`* can be a string or an integer. If it is an integer it determines how links are inserted into the document. The default is 0 and will result in links that are constructed from the name of the link's destination object. This is useful for web applications. If a link points to an object with name 'internet_movie' the HTML link will be <A HREF="/internet_movie">. The actual location of the source and destination object in the document hierachy is disregarded. You will have to set up your web browser, to rewrite that URL to for example '/my_script.php3/internet_movie'. 'my_script.php3' will have to evaluate $PATH_INFO and retrieve the document. All links will have the prefix '/my_script.php3/'. If you do not want this you can set the optional parameter *`rootID/prefix`* to any prefix which is used instead. Is this case it has to be a string.

If *`rootID/prefix`* is an integer and unequal to 0 the link is constructed from all the names starting at the object with the id *`rootID/prefix`* separated by a slash relative to the current object. If for example the above document 'internet_movie' is located at 'a-b-c-internet_movie' with '-' being the seperator between hierachy levels on the Hyperwave server and the source document is located at 'a-b-d-source' the resulting HTML link would be: <A HREF="../c/internet_movie">. This is useful if you want to download the whole server content onto disk and map the document hierachy onto the file system.

This function will only work for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also hw_pipedocument(), hw_free_document(), hw_document_bodytag(), hw_document_size(), and hw_output_document().

hw_getusername (PHP 3>= 3.0.3, PHP 4)

name of currently logged in user

string **hw_getusername** (int connection) \linebreak

Returns the username of the connection.

hw_Identify (PHP 3>= 3.0.3, PHP 4)

identifies as user

int **hw_identify** (string username, string password) \linebreak

Identifies as user with *`username`* and *`password`*. Identification is only valid for the current session. I do not thing this function will be needed very often. In most cases it will be easier to identify with the opening of the connection.

See also hw_connect().

hw_InCollections (PHP 3>= 3.0.3, PHP 4)

check if object ids in collections

array **hw_incollections** (int connection, array object_id_array, array collection_id_array, int return_collections) \linebreak

Checks whether a set of objects (documents or collections) specified by the *object_id_array* is part of the collections listed in *collection_id_array*. When the fourth parameter *return_collections* is 0, the subset of object ids that is part of the collections (i.e., the documents or collections that are children of one or more collections of collection ids or their subcollections, recursively) is returned as an array. When the fourth parameter is 1, however, the set of collections that have one or more objects of this subset as children are returned as an array. This option allows a client to, e.g., highlight the part of the collection hierarchy that contains the matches of a previous query, in a graphical overview.

hw_Info (PHP 3>= 3.0.3, PHP 4)

info about connection

string **hw_info** (int connection) \linebreak

Returns information about the current connection. The returned string has the following format: <Serverstring>, <Host>, <Port>, <Username>, <Port of Client>, <Byte swapping>

hw_InsColl (PHP 3>= 3.0.3, PHP 4)

insert collection

int **hw_inscoll** (int connection, int objectID, array object_array) \linebreak

Inserts a new collection with attributes as in *object_array* into collection with object ID *objectID*.

hw_InsDoc (PHP 3>= 3.0.3, PHP 4)

insert document

int **hw_insdoc** (int connection, int parentID, string object_record, string text) \linebreak

Inserts a new document with attributes as in *object_record* into collection with object ID *parentID*. This function inserts either an object record only or an object record and a pure ascii text in *text* if *text* is given. If you want to insert a general document of any kind use hw_insertdocument() instead.

See also hw_insertdocument(), and hw_inscoll().

hw_insertanchors (PHP 4 >= 4.0.4)

Inserts only anchors into text

string **hw_insertanchors** (int hwdoc, array anchorecs, array dest [, array urlprefixes]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

hw_InsertDocument (PHP 3>= 3.0.3, PHP 4)

upload any document

int **hw_insertdocument** (int connection, int parent_id, int hw_document) \linebreak

Uploads a document into the collection with *parent_id*. The document has to be created before with hw_new_document(). Make sure that the object record of the new document contains at least the attributes: Type, DocumentType, Title and Name. Possibly you also want to set the MimeType. The functions returns the object id of the new document or FALSE.

See also hw_pipedocument().

hw_InsertObject (PHP 3>= 3.0.3, PHP 4)

inserts an object record

int **hw_insertobject** (int connection, string object rec, string parameter) \linebreak

Inserts an object into the server. The object can be any valid hyperwave object. See the HG-CSP documentation for a detailed information on how the parameters have to be.

Note: If you want to insert an Anchor, the attribute Position has always been set either to a start/end value or to 'invisible'. Invisible positions are needed if the annotation has no correspondig link in the annotation text.

See also hw_pipedocument(), hw_insertdocument(), hw_insdoc(), and hw_inscoll().

hw_mapid (PHP 3>= 3.0.13, PHP 4)

Maps global id on virtual local id

int **hw_mapid** (int connection, int server id, int object id) \linebreak

Maps a global object id on any hyperwave server, even those you did not connect to with hw_connect(), onto a virtual object id. This virtual object id can then be used as any other object id, e.g. to obtain the object record with hw_getobject(). The server id is the first part of the global object id (GOid) of the object which is actually the IP number as an integer.

Note: In order to use this function you will have to set the F_DISTRIBUTED flag, which can currently only be set at compile time in hg_comm.c. It is not set by default. Read the comment at the beginning of hg_comm.c

hw_Modifyobject (PHP 3>= 3.0.7, PHP 4)

modifies object record

int **hw_modifyobject** (int connection, int object_to_change, array remove, array add, int mode) \linebreak

This command allows to remove, add, or modify individual attributes of an object record. The object is specified by the Object ID *`object_to_change`*. The first array *`remove`* is a list of attributes to remove. The second array *`add`* is a list of attributes to add. In order to modify an attribute one will have to remove the old one and add a new one. **hw_modifyobject()** will always remove the attributes before it adds attributes unless the value of the attribute to remove is not a string or array.

The last parameter determines if the modification is performed recursively. 1 means recurive modification. If some of the objects cannot be modified they will be skiped without notice. hw_error() may not indicate an error though some of the objects could not be modified.

The keys of both arrays are the attributes name. The value of each array element can either be an array, a string or anything else. If it is an array each attribute value is constructed by the key of each element plus a colon and the value of each element. If it is a string it is taken as the attribute value. An empty string will result in a complete removal of that attribute. If the value is neither a string nor an array but something else, e.g. an integer, no operation at all will be performed on the attribute. This is neccessary if you want to to add a completely new attribute not just a new value for an existing attribute. If the remove array contained an empty string for that attribute, the attribute would be tried to be removed which would fail since it doesn't exist. The following addition of a new value for that attribute would also fail. Setting the value for that attribute to e.g. 0 would not even try to remove it and the addition will work.

If you would like to change the attribute 'Name' with the current value 'books' into 'articles' you will have to create two arrays and call **hw_modifyobject()**.

Példa 1. modifying an attribute

```
// $connect is an existing connection to the Hyperwave server
// $objid is the ID of the object to modify
$remarr = array("Name" => "books");
$addarr = array("Name" => "articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

In order to delete/add a name=value pair from/to the object record just pass the remove/add array and set the last/third parameter to an empty array. If the attribute is the first one with that name to add, set attribute value in the remove array to an integer.

Példa 2. adding a completely new attribute

```
// $connect is an existing connection to the Hyperwave server
// $objid is the ID of the object to modify
$remarr = array("Name" => 0);
$addarr = array("Name" => "articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

Megjegyzés: Multilingual attributes, e.g. 'Title', can be modified in two ways. Either by providing the attributes value in its native form 'language':'title' or by providing an array with elements for each language as described above. The above example would than be:

Példa 3. modifying Title attribute

```
$remarr = array("Title" => "en:Books");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

or

Példa 4. modifying Title attribute

```
$remarr = array("Title" => array("en" => "Books"));
$addarr = array("Title" => array("en" => "Articles", "ge"=>"Artikel"));
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

This removes the english title 'Books' and adds the english title 'Articles' and the german title 'Artikel'.

Példa 5. removing attribute

```
$remarr = array("Title" => "");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

Megjegyzés: This will remove all attributes with the name 'Title' and adds a new 'Title' attribute. This comes in handy if you want to remove attributes recursively.

Megjegyzés: If you need to delete all attributes with a certain name you will have to pass an empty string as the attribute value.

Megjegyzés: Only the attributes 'Title', 'Description' and 'Keyword' will properly handle the language prefix. If those attributes don't carry a language prefix, the prefix 'xx' will be assigned.

Megjegyzés: The 'Name' attribute is somewhat special. In some cases it cannot be complete removed. You will get an error message 'Change of base attribute' (not clear when this happens). Therefore you will always have to add a new Name first and than remove the old one.

Megjegyzés: You may not suround this function by calls to hw_getandlock() and hw_unlock(). **hw_modifyobject()** does this internally.

Returns TRUE if no error occurs otherwise FALSE.

hw_Mv (PHP 3>= 3.0.3, PHP 4)

moves objects

int **hw_mv** (int connection, array object id array, int source id, int destination id) \linebreak

Moves the objects with object ids as specified in the second parameter from the collection with id *`source id`* to the collection with the id *`destination id`*. If the destination id is 0 the objects will be unlinked from the source collection. If this is the last instance of that object it will be deleted. If you want to delete all instances at once, use hw_deleteobject().

The value return is the number of moved objects.

See also hw_cp(), and hw_deleteobject().

hw_New_Document (PHP 3>= 3.0.3, PHP 4)

create new document

int **hw_new_document** (string object_record, string document_data, int document_size) \linebreak

Returns a new Hyperwave document with document data set to *`document_data`* and object record set to *`object_record`*. The length of the *`document_data`* has to passed in *`document_size`*This function does not insert the document into the Hyperwave server.

See also hw_free_document(), hw_document_size(), hw_document_bodytag(), hw_output_document(), and hw_insertdocument().

hw_Objrec2Array (PHP 3>= 3.0.3, PHP 4)

convert attributes from object record to object array

array **hw_objrec2array** (string object_record [, array format]) \linebreak

Converts an *object_record* into an object array. The keys of the resulting array are the attributes names. Multi-value attributes like 'Title' in different languages form its own array. The keys of this array are the left part to the colon of the attribute value. This left part must be two characters long. Other multi-value attributes without a prefix form an indexed array. If the optional parameter is missing the attributes 'Title', 'Description' and 'Keyword' are treated as language attributes and the attributes 'Group', 'Parent' and 'HtmlAttr' as non-prefixed multi-value attributes. By passing an array holding the type for each attribute you can alter this behaviour. The array is an associated array with the attribute name as its key and the value being one of HW_ATTR_LANG or HW_ATTR_NONE

See also hw_array2objrec().

hw_Output_Document (PHP 3>= 3.0.3, PHP 4)

prints hw_document

int **hw_output_document** (int hw_document) \linebreak

Prints the document without the BODY tag.

For backward compatibility, **hw_outputdocument()** is also accepted. This is deprecated, however.

hw_pConnect (PHP 3>= 3.0.3, PHP 4)

make a persistent database connection

int **hw_pconnect** (string host, int port, string username, string password) \linebreak

Returns a connection index on success, or FALSE if the connection could not be made. Opens a persistent connection to a Hyperwave server. Each of the arguments should be a quoted string, except for the port number. The *username* and *password* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple persistent connections open at once.

See also hw_connect().

hw_PipeDocument (PHP 3>= 3.0.3, PHP 4)

retrieve any document

int **hw_pipedocument** (int connection, int objectID) \linebreak

Returns the Hyperwave document with object ID *`objectID`*. If the document has anchors which can be inserted, they will have been inserted already. The document will be transfered via a special data connection which does not block the control connection.

See also hw_gettext() for more on link insertion, hw_free_document(), hw_document_size(), hw_document_bodytag(), and hw_output_document().

hw_Root (PHP 3>= 3.0.3, PHP 4)

root object id

int **hw_root** ( ) \linebreak

Returns the object ID of the hyperroot collection. Currently this is always 0. The child collection of the hyperroot is the root collection of the connected server.

hw_setlinkroot (PHP 3>= 3.0.3, PHP 4)

Set the id to which links are calculated

void **hw_setlinkroot** (int link, int rootid) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

hw_stat (PHP 3>= 3.0.3, PHP 4)

Returns status string

string **hw_stat** (int link) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

hw_Unlock (PHP 3>= 3.0.3, PHP 4)

unlock object

int **hw_unlock** (int connection, int objectID) \linebreak

Unlocks a document, so other users regain access.

See also hw_getandlock().

hw_Who (PHP 3>= 3.0.3, PHP 4)

List of currently logged in users

int **hw_who** (int connection) \linebreak

Returns an array of users currently logged into the Hyperwave server. Each entry in this array is an array itself containing the elements id, name, system, onSinceDate, onSinceTime, TotalTime and self. 'self' is 1 if this entry belongs to the user who initianted the request.

XXXIX. Hyperwave API functions

Introduction

Hyperwave has been developed at IICM (http://www.iicm.edu/) in Graz. It started with the name Hyper-G and changed to Hyperwave when it was commercialised (If I remember properly it was in 1996).

Hyperwave is not free software. The current version, 5.5, is available at www.hyperwave.com (http://www.hyperwave.com/). A time limited version can be ordered for free (30 days).

Hyperwave is an information system similar to a database (HIS, Hyperwave Information Server). Its focus is the storage and management of documents. A document can be any possible piece of data that may as well be stored in file. Each document is accompanied by its object record. The object record contains meta data for the document. The meta data is a list of attributes which can be extended by the user. Certain attributes are always set by the Hyperwave server, other may be modified by the user.

Requirements

Since 2001 there is a Hyperwave SDK available. It supports Java, JavaScript and C++. This PHP Extension is based on the C++ interface. In order to activate the hwapi support in PHP you will have to install the Hyperwave SDK first and configure PHP with `--with-hwapi=<dir$gt;`.

Classes

The API provided by the HW_API extension is fully object oriented. It is very similar to the C++ interface of the Hyperwave SDK. It consist of the following classes.

- `HW_API`
- `HW_API_Object`
- `HW_API_Attribute`
- `HW_API_Error`
- `HW_API_Content`
- `HW_API_Reason`

Some basic classes like `HW_API_String`, `HW_API_String_Array`, etc., which exist in the Hyperwave SDK have not been implemented since PHP has powerful replacements for them.

Each class has certain method, whose names are identical to its counterparts in the Hyperwave SDK. Passing arguments to this function differs from all the other PHP Extension but is close to the C++ API of the HW SDK. Instead of passing serval parameters they are all put into an associated array and passed as one paramter. The names of the keys are identical to those documented in the HW

SDK. The common parameters are listed below. If other parameters are required they will be documented if needed.

- `objectIdentifier` The name or id of an object, e.g. "rootcollection", "0x873A8768 0x00000002".
- `parentIdentifier` The name or id of an object which is considered to be a parent.
- `object` An instance of class HW_API_Object.
- `parameters` An instance of class HW_API_Object.
- `version` The version of an object.
- `mode` An integer value determine the way an operation is executed.
- `attributeSelector` Any array of strings, each containing a name of an attribute. This is used if you retrieve the object record and want to include certain attributes.
- `objectQuery` A query to select certain object out of a list of objects. This is used to reduce the number of objects which was delivered by a function like **hw_api->children()** or **hw_api->find()**.

Integration with Apache

The integration with Apache and possible other servers is already described in the Hyperwave Modul which has been the first extension to connect a Hyperwave Server.

hw_api_attribute->key (unknown)

Returns key of the attribute

string **key** (void) \linebreak

Returns the name of the attribute.

See also hwapi_attribute_value().

hw_api_attribute->langdepvalue (unknown)

Returns value for a given language

string **langdepvalue** (string language) \linebreak

Returns the value in the given language of the attribute.

See also hwapi_attribute_value().

hw_api_attribute->value (unknown)

Returns value of the attribute

string **value** (void) \linebreak

Returns the value of the attribute.

See also hwapi_attribute_key(), hwapi_attribute_values().

hw_api_attribute->values (unknown)

Returns all values of the attribute

array **values** (void) \linebreak

Returns all values of the attribute as an array of strings.

See also hwapi_attribute_value().

hw_api_attribute (unknown)

Creates instance of class hw_api_attribute

object **attribute** ([string name [, string value]]) \linebreak

Creates a new instance of hw_api_attribute with the given name and value.

hw_api->checkin (unknown)

Checks in an object

object **checkin** (array parameter) \linebreak

This function checks in an object or a whole hiearchie of objects. The parameters array contains the required element 'objectIdentifier' and the optional element 'version', 'comment', 'mode' and 'objectQuery'. 'version' sets the version of the object. It consists of the major and minor version separated by a period. If the version is not set, the minor version is incremented. 'mode' can be one of the following values:

HW_API_CHECKIN_NORMAL

Checks in and commits the object. The object must be a document.

HW_API_CHECKIN_RECURSIVE

If the object to check in is a collection, all children will be checked in recursively if they are documents. Trying to check in a collection would result in an error.

HW_API_CHECKIN_FORCE_VERSION_CONTROL

Checks in an object even if it is not under version control.

HW_API_CHECKIN_REVERT_IF_NOT_CHANGED

Check if the new version is different from the last version. Unless this is the case the object will be checked in.

HW_API_CHECKIN_KEEP_TIME_MODIFIED

Keeps the time modified from the most recent object.

HW_API_CHECKIN_NO_AUTO_COMMIT

The object is not automatically commited on checkin.

See also hwapi_checkout().

hw_api->checkout (unknown)

Checks out an object

object **checkout** (array parameter) \linebreak

This function checks out an object or a whole hiearchie of objects. The parameters array contains the required element 'objectIdentifier' and the optional element 'version', 'mode' and 'objectQuery'. 'mode' can be one of the following values:

HW_API_CHECKIN_NORMAL

Checks out an object. The object must be a document.

HW_API_CHECKIN_RECURSIVE

If the object to check out is a collection, all children will be checked out recursively if they are documents. Trying to check out a collection would result in an error.

See also hwapi_checkin().

hw_api->children (unknown)

Returns children of an object

array **children** (array parameter) \linebreak

Retrieves the children of a collection or the attributes of a document. The children can be further filtered by specifying an object query. The parameter array contains the required elements 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

The return value is an array of objects of type `HW_API_Object` or `HW_API_Error`.

See also hwapi_parents().

hw_api_content->mimetype (unknown)

Returns mimetype

string **mimetype** (void) \linebreak

Returns the mimetype of the content.

hw_api_content->read (unknown)

Read content

string **read** (string buffer, integer len) \linebreak

Reads *`len`* bytes from the content into the given buffer.

hw_api->content (unknown)

Returns content of an object

object **content** (array parameter) \linebreak

This function returns the content of a document as an object of type `hw_api_content`. The parameter array contains the required elements 'objectidentifier' and the optional element 'mode'. The mode can be one of the constants HW_API_CONTENT_ALLLINKS, HW_API_CONTENT_REACHABLELINKS or HW_API_CONTENT_PLAIN.

HW_API_CONTENT_ALLLINKS means to insert all anchors even if the destination is not reachable. HW_API_CONTENT_REACHABLELINKS tells **hw_api_content()** to insert only reachable links and HW_API_CONTENT_PLAIN will lead to document without any links.

hw_api->copy (unknown)

Copies physically

object **copy** (array parameter) \linebreak

This function will make a physical copy including the content if it exists and returns the new object or an error object. The parameter array contains the required elements 'objectIdentifier' and 'destinationParentIdentifier'. The optional parameter is 'attributeSelector''

See also hwapi_move(), hwapi_link().

hw_api->dbstat (unknown)

Returns statistics about database server

object **dbstat** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_hwstat(), hwapi_ftstat().

hw_api->dcstat (unknown)

Returns statistics about document cache server

object **dcstat** (array parameter) \linebreak

See also hwapi_hwstat(), hwapi_dbstat(), hwapi_ftstat().

hw_api->dstanchors (unknown)

Returns a list of all destination anchors

object **dstanchors** (array parameter) \linebreak

Retrieves all destination anchors of an object. The parameter array contains the required element 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

See also hwapi_srcanchors().

hw_api->dstofsrcanchors (unknown)

Returns destination of a source anchor

object **dstofsrcanchors** (array parameter) \linebreak

Retrieves the destination object pointed by the specified source anchors. The destination object can either be a destination anchor or a whole document. The parameters array contains the required element 'objectIdentifier' and the optional element 'attributeSelector'.

See also hwapi_srcanchors(), hwapi_dstanchors(), hwapi_objectbyanchor().

hw_api_error->count (unknown)

Returns number of reasons

int **count** (void) \linebreak

Returns the number of error reasons.

See also hwapi_error_reason().

hw_api_error->reason (unknown)

Returns reason of error

object **reason** (void) \linebreak

Returns the first error reason.

See also hwapi_error_count().

hw_api->find (unknown)

Search for objects

array **find** (array parameter) \linebreak

This functions searches for objects either by executing a key or/and full text query. The found objects can further be filtered by an optional object query. They are sorted by their importance. The second search operation is relatively slow and its result can be limited to a certain number of hits. This allows to perform an incremental search, each returning just a subset of all found documents, starting at a given index. The parameter array contains the 'keyquery' or/and 'fulltextquery' depending on who you would like to search. Optional parameters are 'objectquery', 'scope', 'lanugages' and 'attributeselector'. In case of an incremental search the optional parameters 'startIndex', numberOfObjectsToGet' and 'exactMatchUnit' can be passed.

hw_api->ftstat (unknown)

Returns statistics about fulltext server

object **ftstat** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_dbstat(), hwapi_hwstat().

hwapi_hgcsp (unknown)

Returns object of class hw_api

object **hwapi_hgcsp** (string hostname [, int port]) \linebreak

Opens a connection to the Hyperwave server on host *`hostname`*. The protocol used is HGCSP. If you do not pass a port number, 418 is used.

See also **hwapi_hwtp()**.

hw_api->hwstat (unknown)

Returns statistics about Hyperwave server

object **hwstat** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_dbstat(), hwapi_ftstat().

hw_api->identify (unknown)

Log into Hyperwave Server

object **identify** (array parameter) \linebreak

Logs into the Hyperwave Server. The parameter array must contain the elements 'username' und 'password'.

The return value will be an object of type `HW_API_Error` if identification failed or TRUE if it was successful.

hw_api->info (unknown)

Returns information about server configuration

object **info** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_dbstat(), hwapi_ftstat(), hwapi_hwstat().

hw_api->insert (unknown)

Inserts a new object

object **insert** (array parameter) \linebreak

Insert a new object. The object type can be user, group, document or anchor. Depending on the type other object attributes has to be set. The parameter array contains the required elements 'object' and 'content' (if the object is a document) and the optional parameters 'parameters', 'mode' and 'attributeSelector'. The 'object' must contain all attributes of the object. 'parameters' is an object as well holding futher attributes like the destination (attribute key is 'Parent'). 'content' is the content of the document. 'mode' can be a combination of the following flags:

HW_API_INSERT_NORMAL

The object in inserted into the server.

HW_API_INSERT_FORCE-VERSION-CONTROL

HW_API_INSERT_AUTOMATIC-CHECKOUT

HW_API_INSERT_PLAIN

HW_API_INSERT_KEEP_TIME_MODIFIED

HW_API_INSERT_DELAY_INDEXING

See also hwapi_replace().

hw_api->insertanchor (unknown)

Inserts a new object of type anchor

object **insertanchor** (array parameter) \linebreak

This function is a shortcut for hwapi_insert(). It inserts an object of type anchor and sets some of the attributes required for an anchor. The parameter array contains the required elements 'object' and 'documentIdentifier' and the optional elements 'destinationIdentifier', 'parameter', 'hint' and 'attributeSelector'. The 'documentIdentifier' specifies the document where the anchor shall be inserted. The target of the anchor is set in 'destinationIdentifier' if it already exists. If the target does not exists the element 'hint' has to be set to the name of object which is supposed to be inserted later. Once it is inserted the anchor target is resolved automatically.

See also hwapi_insertdocument(), hwapi_insertcollection(), hwapi_insert().

hw_api->insertcollection (unknown)

Inserts a new object of type collection

object **insertcollection** (array parameter) \linebreak

This function is a shortcut for hwapi_insert(). It inserts an object of type collection and sets some of the attributes required for a collection. The parameter array contains the required elements ’object’ and ’parentIdentifier’ and the optional elements ’parameter’ and ’attributeSelector’. See hwapi_insert() for the meaning of each element.

See also hwapi_insertdocument(), hwapi_insertanchor(), hwapi_insert().

hw_api->insertdocument (unknown)

Inserts a new object of type document

object **insertdocument** (array parameter) \linebreak

This function is a shortcut for hwapi_insert(). It inserts an object with content and sets some of the attributes required for a document. The parameter array contains the required elements ’object’, ’parentIdentifier’ and ’content’ and the optional elements ’mode’, ’parameter’ and ’attributeSelector’. See hwapi_insert() for the meaning of each element.

See also hwapi_insert() hwapi_insertanchor(), hwapi_insertcollection().

hw_api->link (unknown)

Creates a link to an object

object **link** (array parameter) \linebreak

Creates a link to an object. Accessing this link is like accessing the object to links points to. The parameter array contains the required elements ’objectIdentifier’ and ’destinationParentIdentifier’. ’destinationParentIdentifier’ is the target collection.

The function returns true on success or an error object.

See also hwapi_copy().

hw_api->lock (unknown)

Locks an object

object **lock** (array parameter) \linebreak

Locks an object for exclusive editing by the user calling this function. The object can be only unlocked by this user or the sytem user. The parameter array contains the required element ’objectIdentifier’ and the optional parameters ’mode’ and ’objectquery’. ’mode’ determines how an object is locked. HW_API_LOCK_NORMAL means, an object is locked until it is unlocked.

HW_API_LOCK_RECURSIVE is only valid for collection and locks all objects within th collection und possible subcollections. HW_API_LOCK_SESSION means, an object is locked only as long as the session is valid.

See also hwapi_unlock().

hw_api->move (unknown)

Moves object between collections

object **move** (array parameter) \linebreak

See also hw_objrec2array().

hw_api_content (unknown)

Create new instance of class hw_api_content

string **content** (string content, string mimetype) \linebreak

Creates a new content object from the string *`content`*. The mimetype is set to *`mimetype`*.

hw_api_object->assign (unknown)

Clones object

object **assign** (array parameter) \linebreak

Clones the attributes of an object.

hw_api_object->attreditable (unknown)

Checks whether an attribute is editable

bool **attreditable** (array parameter) \linebreak

hw_api_object->count (unknown)

Returns number of attributes

int **count** (array parameter) \linebreak

hw_api_object->insert (unknown)

Inserts new attribute

bool **insert** (object attribute) \linebreak

Adds an attribute to the object. Returns true on success and otherwise false.

See also hwapi_object_remove().

hw_api_object (unknown)

Creates a new instance of class hw_api_object

object **hw_api_object** (array parameter) \linebreak

See also hwapi_lock().

hw_api_object->remove (unknown)

Removes attribute

bool **remove** (string name) \linebreak

Removes the attribute with the given name. Returns true on success and otherwise false.

See also hwapi_object_insert().

hw_api_object->title (unknown)

Returns the title attribute

string **title** (array parameter) \linebreak

hw_api_object->value (unknown)

Returns value of attribute

string **value** (string name) \linebreak

Returns the value of the attribute with the given name or false if an error occured.

hw_api->object (unknown)

Retrieve attribute information

object **hw_api->object** (array parameter) \linebreak

This function retrieves the attribute information of an object of any version. It will not return the document content. The parameter array contains the required elements ’objectIdentifier’ and the optional elements ’attributeSelector’ and ’version’.

The returned object is an instance of class `HW_API_Object` on success or `HW_API_Error` if an error occured.

This simple example retrieves an object and checks for errors.

Példa 1. Retrieve an object

```
<?php
function handle_error($error) {
  $reason = $error->reason(0);
  echo "Type: <B>";
  switch($reason->type()) {
    case 0:
      echo "Error";
      break;
    case 1:
      echo "Warning";
      break;
    case 2:
      echo "Message";
      break;
  }
  echo "</B><BR>\n";
  echo "Description: ".$reason->description("en")."<BR>\n";
}

function list_attr($obj) {
  echo "<TABLE>\n";
  $count = $obj->count();
  for($i=0; $i<$count; $i++) {
    $attr = $obj->attribute($i);
    printf("  <TR><TD ALIGN=right bgcolor=#c0c0c0><B>%s</B></TD><TD bgcolor=#F0F0F0>%s</T
             $attr->key(), $attr->value());
  }
  echo "</TABLE>\n";
}

$hwapi = hwapi_hgcsp($g_config[HOSTNAME]);
$parms = array("objectIdentifier"=>"rootcollection, "attributeSelector"=>array("Title", "
$root = $hwapi->object($parms);
if(get_class($root) == "HW_API_Error") {
  handle_error($root);
  exit;
}
list_attr($root);
?>
```

See also hwapi_content().

hw_api->objectbyanchor (unknown)

Returns the object an anchor belongs to

object **objectbyanchor** (array parameter) \linebreak

This function retrieves an object the specified anchor belongs to. The parameter array contains the required element 'objectIdentifier' and the optional element 'attributeSelector'.

See also **hwapi_dstofsrcanchor()**, hwapi_srcanchors(), hwapi_dstanchors().

hw_api->parents (unknown)

Returns parents of an object

array **parents** (array parameter) \linebreak

Retrieves the parents of an object. The parents can be further filtered by specifying an object query. The parameter array contains the required elements 'objectidentifier' and the optional elements 'attributeselector' and 'objectquery'.

The return value is an array of objects of type `HW_API_Object` or `HW_API_Error`.

See also hwapi_children().

hw_api_reason->description (unknown)

Returns description of reason

string **description** (void) \linebreak

Returns the description of a reason

hw_api_reason->type (unknown)

Returns type of reason

object **type** (void) \linebreak

Returns the type of a reason.

hw_api->remove (unknown)

Delete an object

object **remove** (array parameter) \linebreak

This function removes an object from the specified parent. Collections will be removed recursively. You can pass an optional object query to remove only those objects which match the query. An object will be deleted physically if it is the last instance. The parameter array contains the required elements ’objectidentifier’ and ’parentidentifier’. If you want to remove a user or group ’parentidentifier’ can be skiped. The optional parameter ’mode’ determines how the deletion is performed. In normal mode the object will not be removed physically until all instances are removed. In physical mode all instances of the object will be deleted imediately. In removelinks mode all references to and from the objects will be deleted as well. In nonrecursive the deletion is not performed recursive. Removing a collection which is not empty will cause an error.

See also hwapi_move().

hw_api->replace (unknown)

Replaces an object

object **replace** (array parameter) \linebreak

Replaces the attributes and the content of an object The parameter array contains the required elements ’objectIdentifier’ and ’object’ and the optional parameters ’content’, ’parameters’, ’mode’ and ’attributeSelector’. ’objectIdentifier’ contains the object to be replaced. ’object’ contains the new object. ’content’ contains the new content. ’parameters’ contain extra information for HTML documents. HTML_Language is the letter abbreviation of the language of the title. HTML_Base sets the base attribute of the HTML document. ’mode’ can be a combination of the following flags:

HW_API_REPLACE_NORMAL

The object on the server is replace with the object passed.

HW_API_REPLACE_FORCE_VERSION_CONTROL

HW_API_REPLACE_AUTOMATIC_CHECKOUT

HW_API_REPLACE_AUTOMATIC_CHECKIN

HW_API_REPLACE_PLAIN

HW_API_REPLACE_REVERT_IF_NOT_CHANGED

HW_API_REPLACE_KEEP_TIME_MODIFIED

See also hwapi_insert().

hw_api->setcommitedversion (unknown)

Commits version other than last version

object **setcommitedversion** (array parameter) \linebreak

Commits a version of a docuemnt. The commited version is the one which is visible to users with read access. By default the last version is the commited version.

See also hwapi_checkin(), hwapi_checkout(), **hwapi_revert()**.

hw_api->srcanchors (unknown)

Returns a list of all source anchors

object **srcanchors** (array parameter) \linebreak

Retrieves all source anchors of an object. The parameter array contains the required element ’objectIdentifier’ and the optional elements ’attributeSelector’ and ’objectQuery’.

See also hwapi_dstanchors().

hw_api->srcsofdst (unknown)

Returns source of a destination object

object **srcsofdst** (array parameter) \linebreak

Retrieves all the source anchors pointing to the specified destination. The destination object can either be a destination anchor or a whole document. The parameters array contains the required element ’objectIdentifier’ and the optional element ’attributeSelector’ and ’objectQuery’. The function returns an array of objects or an error.

See also **hwapi_dstofsrcanchor()**.

hw_api->unlock (unknown)

Unlocks a locked object

object **unlock** (array parameter) \linebreak

Unlocks a locked object. Only the user who has locked the object and the system user may unlock an object. The parameter array contains the required element ’objectIdentifier’ and the optional parameters ’mode’ and ’objectquery’. The meaning of ’mode’ is the same as in function hwapi_lock().

Returns true on success or an object of class HW_API_Error.

See also hwapi_lock().

hw_api->user (unknown)

Returns the own user object

object **user** (array parameter) \linebreak

See also hwapi_userlist().

hw_api->userlist (unknown)

Returns a list of all logged in users

object **userlist** (array parameter) \linebreak

See also hwapi_user().

XL. ICAP Functions [deprecated]

Bevezetés

> **Megjegyzés:** Icap will be removed in near future. Neither this module, nor those versions of icap library are supported any longer. If you want to use calendar capabilities in PHP, use mcal instead.

Követelmények

The icap library has to be installed, which is not longer supported and available.

Telepítés

To get these functions to work, you have to compile PHP with `--with-icap`.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

icap_close (unknown)

Close an ICAP stream

int **icap_close** (int icap_stream [, int flags]) \linebreak

Closes the given icap stream.

icap_create_calendar (PHP 4)

Create a new calendar

string **icap_create_calendar** (int stream_id, string calendar) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

icap_delete_calendar (PHP 4)

Delete a calendar

string **icap_delete_calendar** (int stream_id, string calendar) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

icap_delete_event (PHP 4)

Delete an event from an ICAP calendar

string **icap_delete_event** (int stream_id, int uid) \linebreak

icap_delete_event() deletes the calendar event specified by the *uid*.

Returns TRUE.

icap_fetch_event (PHP 4)

Fetches an event from the calendar stream/

int **icap_fetch_event** (int stream_id, int event_id [, int options]) \linebreak

icap_fetch_event() fetches an event from the calendar stream specified by *event_id*.

Returns an event object consisting of:

- int id - ID of that event.
- int public - TRUE if the event if public, FALSE if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds

icap_list_alarms (PHP 4)

Return a list of events that has an alarm triggered at the given datetime

int **icap_list_alarms** (int stream_id, array date, array time) \linebreak

Returns an array of event ID's that has an alarm going off at the given datetime.

icap_list_alarms() function takes in a datetime for a calendar stream. An array of event id's that has an alarm should be going off at the datetime are returned.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour

- int min - minutes
- int sec - seconds

icap_list_events (PHP 4)

Return a list of events between two given datetimes

array **icap_list_events** (int stream_id, int begin_date [, int end_date]) \linebreak

Returns an array of event ID's that are between the two given datetimes.

icap_list_events() function takes in a beginning datetime and an end datetime for a calendar stream. An array of event id's that are between the given datetimes are returned.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds

icap_open (PHP 4)

Opens up an ICAP connection

stream **icap_open** (string calendar, string username, string password, string options) \linebreak

Returns an ICAP stream on success, FALSE on error.

icap_open() opens up an ICAP connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also.

icap_rename_calendar (PHP 4)

Rename a calendar

string **icap_rename_calendar** (int stream_id, string old_name, string new_name) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

icap_reopen (PHP 4)

Reopen ICAP stream to new calendar

int **icap_reopen** (int stream_id, string calendar [, int options]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

icap_snooze (PHP 4)

Snooze an alarm

string **icap_snooze** (int stream_id, int uid) \linebreak

icap_snooze() turns on an alarm for a calendar event specified by the *`uid`*.

Returns `TRUE`.

icap_store_event (PHP 4)

Store an event into an ICAP calendar

string **icap_store_event** (int stream_id, object event) \linebreak

icap_store_event() Stores an event into an ICAP calendar. An event object consists of:

- int public - 1 if public, 0 if private;
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - Number of minutes before the event to sned out an alarm.
- datetime start - datetime object of the start of the event.
- datetime end - datetime object of the end of the event.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds

Returns `TRUE` on success and `FALSE` on error.

XLI. iconv functions

Bevezetés

This module contains an interface to the iconv library functions. The Iconv library functions convert files between various character sets encodings. The supported character sets depend on iconv() implementation on your system. Note that iconv() function in some system is not work well as you expect. In this case, you should install libiconv library.

Követelmények

You must have iconv() function in standard C library or libiconv installed on your system. libiconv library is available from http://www.gnu.org/software/libiconv/.

Telepítés

To be able to use the functions defined in this module you must compile your PHP interpreter using the configure line `--with-iconv`.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

iconv_get_encoding (PHP 4 >= 4.0.5)

Get current setting for character encoding conversion

array **iconv_get_encoding** ([string type]) \linebreak

It returns the current settings of ob_iconv_handler() as array or FALSE in failure.

See also: iconv_set_encoding() and ob_iconv_handler().

iconv_set_encoding (PHP 4 >= 4.0.5)

Set current setting for character encoding conversion

array **iconv_set_encoding** (string type, string charset) \linebreak

It changes the value of *type* to *charset* and returns TRUE in success or FALSE in failure.

Példa 1. iconv_set_encoding() example:

```
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
```

See also: iconv_get_encoding() and ob_iconv_handler().

iconv (PHP 4 >= 4.0.5)

Convert string to requested character encoding

string **iconv** (string in_charset, string out_charset, string str) \linebreak

It converts the string *string* encoded in *in_charset* to the string encoded in *out_charset*. It returns the converted string or FALSE, if it fails.

Példa 1. iconv() example:

```
echo iconv("ISO-8859-1","UTF-8","This is test.");
```

ob_iconv_handler (PHP 4 >= 4.0.5)

Convert character encoding as output buffer handler

array **ob_iconv_handler** (string contents, int status) \linebreak

It converts the string encoded in *internal_encoding* to *output_encoding*.

internal_encoding and *output_encoding* should be defined by iconv_set_encoding() or in configuration file.

Példa 1. ob_iconv_handler() example:

```
ob_start("ob_iconv_handler"); // start output buffering
```

See also: iconv_get_encoding() and iconv_set_encoding().

XLII. Képmanipuláló függvények

A PHP képmanipuláló függvényeit használhatod JPEG, GIF, PNG és SWF képek méreteinek lekérdezésére, és ha van GD könyvtárad, mely a http://www.boutell.com/gd/ címen érhető el, tudsz vele képeket készíteni és manipulálni.

exif_imagetype (PHP 4 CVS only)

Determine the type of an image

int|false **exif_imagetype** (string filename) \linebreak

exif_imagetype() reads the first bytes of an image and checks its signature. When a correct signature is found a constant will be returned otherwise the return value is FALSE. The return value is the same value that getimagesize() returns in index 2 but this function is much faster.

The following constants are defined: 1 = IMAGETYPE_GIF, 2 = IMAGETYPE_JPG, 3 = IMAGETYPE_PNG, 4 = IMAGETYPE_SWF, 5 = IMAGETYPE_PSD, 6 = IMAGETYPE_BMP, 7 = IMAGETYPE_TIFF_II (intel byte order), 8 = IMAGETYPE_TIFF_MM (motorola byte order), 9 = IMAGETYPE_JPC, 10 = IMAGETYPE_JP2, 11 = IMAGETYPE_JPX, 12 = IMAGETYPE_SWC.

This function can be used to avoid calls to other exif functions with unsupported file teypes or in conjunction with $_SERVER['HTTP_ACCEPT'] to check whether or not the viewer is able to see a specific image in his browser.

Megjegyzés: This function is only available in PHP 4 compiled using `--enable-exif`.

This function does not require the GD image library.

See also getimagesize().

exif_read_data (PHP 4 >= 4.2.0)

Read the EXIF headers from JPEG or TIFF

array **exif_read_data** (string filename [, string sections [, bool arrays [, bool thumbnail]]]) \linebreak

The **exif_read_data()** function reads the EXIF headers from a JPEG or TIFF image file. It returns an associative array where the indexes are the header names and the values are the values associated with those headers. If no data can be returned the result is FALSE.

`filename` is the name of the file to read. This cannot be a url.

`sections` a comma separated list of sections that need to be present in file to produce a result array.

FILE	FileName, FileSize, FileDateTime, SectionsFound
COMPUTED	html, Width, Height, IsColor and some more if available.
ANY_TAG	Any information that has a Tag e.g. IFD0, EXIF, ...
IFD0	All tagged data of IFD0. In normal imagefiles this contains image size and so forth.

THUMBNAIL	A file is supposed to contain a thumbnail if it has a second IFD. All tagged information about the embedded thumbnail is stored in this section.
COMMENT	Comment headers of JPEG images.
EXIF	The EXIF section is a sub section of IFD0. It contains more detailed information about an image. Most of these entries are digital camera related.

arrays specifies whether or not each section becomes an array. The sections *FILE*, *COMPUTED* and *THUMBNAIL* allways become arrays as they may contain values whose names are conflict with other sections.

thumbnail whether or not to read the thumbnail itself and not only its tagged data.

> **Megjegyzés:** Exif headers tend to be present in JPEG/TIFF images generated by digital cameras, but unfortunately each digital camera maker has a different idea of how to actually tag their images, so you can't always rely on a specific Exif header being present.

Példa 1. exif_read_data() example

```
<?php
echo "test1.jpg:<br />\n";
$exif = exif_read_data ('tests/test1.jpg','IFD0');
echo $exif===false ? "No header data found.<br />\n" : "Image contains head-
ers<br />";
$exif = exif_read_data ('tests/test2.jpg',0,true);
echo "test2.jpg:<br />\n";
foreach($exif as $key=>$section) {
    foreach($section as $name=>$val) {
        echo "$key.$name: $val<br />\n";
    }
}?>
```

The first call fails because the image has no header information.

```
test1.jpg:
No header data found.
test2.jpg:
FILE.FileName: test2.jpg
FILE.FileDateTime: 1017666176
FILE.FileSize: 1240
FILE.FileType: 2
FILE.SectionsFound: ANY_TAG, IFD0, THUMBNAIL, COMMENT
COMPUTED.html: width="1" height="1"
COMPUTED.Height: 1
COMPUTED.Width: 1
COMPUTED.IsColor: 1
```

```
COMPUTED.ByteOrderMotorola: 1
COMPUTED.UserComment: Exif test image.
COMPUTED.UserCommentEncoding: ASCII
COMPUTED.Copyright: Photo (c) M.Boerger, Edited by M.Boerger.
COMPUTED.Copyright.Photographer: Photo (c) M.Boerger
COMPUTED.Copyright.Editor: Edited by M.Boerger.
IFD0.Copyright: Photo (c) M.Boerger
IFD0.UserComment: ASCII
THUMBNAIL.JPEGInterchangeFormat: 134
THUMBNAIL.JPEGInterchangeFormatLength: 523
COMMENT.0: Comment #1.
COMMENT.1: Comment #2.
COMMENT.2: Comment #3end
```

Megjegyzés: If the image contains any IFD0 data then COMPUTED contains the entry ByteOrderMotorola which is 0 for little-endian (intel) and 1 for big-endian (motorola) byte order. This was added in PHP 4.3.

When an Exif header contains a Copyright note this itself can contain two values. As the solution is inconsitent in the Exif 2.10 standard the COMPUTED section will return both entries *Copyright.Photographer* and *Copyright.Editor* while the IFD0 sections contains the byte array with the NULL character that splits both entries. Or just the first entry if the datatype was wrong (normal behaviour of Exif). The COMPUTED will contain also an entry *Copyright* Which is either the original copyright string or it is a comma separated list of photo and editor copyright.

Megjegyzés: The tag UserComment has the same problem as the Copyright tag. It can store two values first the encoding used and second the value itself. If so the IFD section only contains the encoding or a byte array. The COMPUTED section will store both in the entries *UserCommentEncoding* and *UserComment*. The entry *UserComment* is available in both cases so it should be used in preference to the value in IFD0 section.

If the user comment uses Unicode or JIS encoding and the module mbstring is available this encoding will automatically changed according to the exif ini settings. This was added in PHP 4.3.

Megjegyzés: Height and Width are computed the same way getimagesize() does so their values must not be part of any header returned. Also html is a height/width text string to be used inside normal HTML.

Megjegyzés: Starting from PHP 4.3 the function can read all embedded IFD data including arrays (returned as such). Also the size of an embedded thumbnail is returned in *THUMBNAIL* subarray and the function **exif_read_data()** can return thumbnails in TIFF format. Last but not least there is no longer a maximum legth for returned values (not until memory limit is reached).

Megjegyzés: This function is only available in PHP 4 compiled using `--enable-exif`. Its functionality and behaviour has changed in PHP 4.2. Earlier versions are very unstable.

Since PHP 4.3 user comment can automatically change encoding if PHP 4 was compiled using `--enable-mbstring`.

This function does not require the GD image library.

See also exif_thumbnail() and getimagesize().

exif_thumbnail (PHP 4 >= 4.2.0)

Retrieve the embedded thumbnail of a TIFF or JPEG image

string **exif_thumbnail** (string filename [, int &width [, int &height]]) \linebreak

exif_thumbnail() reads the embedded thumbnail of a TIFF or JPEG image. If the image contains no thumbnail `FALSE` will be returned.

Both parameters *`width`* and *`height`* are available since PHP 4.3 and return the size of the thumbnail. It is possible that **exif_thumbnail()** cannot create an image but determine its size. In this case the return value is `FALSE` but *`width`* and *`height`* are set.

Starting from version PHP 4.3 the function **exif_thumbnail()** can return thumbnails in TIFF format.

Megjegyzés: This function is only available in PHP 4 compiled using `--enable-exif`. Its functionality and behaviour has changed in PHP 4.2

This function does not require the GD image library.

See also exif_read_data().

GetImageSize (PHP 3, PHP 4)

Egy GIF, JPEG, PNG vagy SWF kép méretét adja vissza

array **getimagesize** (string filename [, array imageinfo]) \linebreak

A **GetImageSize()** függvény megállapítja a méretét bármely GIF, JPG, PNG vagy SWF filenak és visszaadja a kép méreteit along with a file type and a height/width text string to be used inside a normal HTML `IMG` tag.

Egy 4 elemű tömbbel tér vissza. A tömb első eleme (indexe 0) a kép szélessége pixelben. A következő a kép magassága. A harmadik pedig egy flag, amely a kép típusára utal. 1 = GIF, 2 = JPG, 3 = PNG, 4 = SWF. A negyedik pedig egy string, amely tartalmazza a megfelelő "height=xxx width=xxx" string, amit egy az egyben bele tudsz írni egy IMG tagbe.

Példa 1. GetImageSize

```
<?php $size = GetImageSize ("img/flag.jpg"); ?>
<IMG SRC="img/flag.jpg" <?php echo $size[3]; ?>
```

Az opcionális *`imageinfo`* paraméter lehetővé teszi néhány plusz információ kinyerését a fileból. Jelenleg ez a különböző JPG APP jelzéseket asszociatív tömb formájában adja vissza. Néhány program ezeket az APP jelzéseket használja, hogy szöveges információt rejtsen a képekbe. A legismertebb az IPTC http://www.iptc.org/ információ belarakása az APP13 jelzésbe. Az iptcparse() funkciót használhatod arra, hogy a bináris APP13 jelzést valami olvashatóvá alakítsd.

Példa 2. IPTC információ kinyerése a GetImageSize segítségével

```
<?php
    $size = GetImageSize ("testimg.jpg",&$info);
    if (isset ($info["APP13"])) {
        $iptc = iptcparse ($info["APP13"]);
        var_dump ($iptc);
    }
?>
```

Megjegyzés: Ehhez a függvényhez nem kell a GD könyvtár.

image_type_to_mime_type (unknown)

Get Mime-Type for image-type returned by getimagesize, exif_read_data, exif_thumbnail, exif_imagetype

string **image_type_to_mime_type** (int imagetype) \linebreak

The **image_type_to_mime_type()** function will determine the Mime-Type for an IMAGETYPE constant.

Példa 1. image_type_to_mime_type (file)

```
<?php
header ("Content-type: ".image_type_to_mime_type (IMAGETYPE_PNG));
?>
```

Megjegyzés: This function does not require the GD image library.

See also getimagesize(), exif_imagetype(), exif_read_data() and exif_thumbnail().

image2wbmp (PHP 4 >= 4.0.5)

Output image to browser or file

int **image2wbmp** (resource image [, string filename [, int threshold]]) \linebreak

image2wbmp() creates the WBMP file in filename from the image *`image`*. The *`image`* argument is the return from imagecreate().

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using header(), you can create a PHP script that outputs WBMP images directly.

Megjegyzés: WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also imagewbmp().

imagealphablending (PHP 4 >= 4.0.6)

Set the blending mode for an image

int **imagealphablending** (resource image, bool blendmode) \linebreak

imagealphablending() allows for two different modes of drawing on truecolor images. In blending mode, the alpha channel component of the color supplied to all drawing function, such as imagesetpixel() determines how much of the underlying color should be allowed to shine through. As a result, gd automatically blends the existing color at that point with the drawing color, and stores the result in the image. The resulting pixel is opaque. In non-blending mode, the drawing color is copied literally with its alpha channel information, replacing the destination pixel. Blending mode is not available when drawing on palette images. If *`blendmode`* is `TRUE`, then blending mode is enabled, otherwise disabled.

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.1

ImageArc (PHP 3, PHP 4)

Egy ellipszisdarabot rajzol

int **imagearc** (int im, int cx, int cy, int w, int h, int s, int e, int col) \linebreak

Az **ImageArc()** függvény egy ellipszisdarabot rajzol az im-mel azonosított képre *cx*, *cy* középponttal (a kép bal felső széle a 0, 0). A *w* és a *h* paraméterek az ellipszis szélességét illetve magasságát határozzák meg. Az ellipszisdarab kezdő és végpontját fokban a *s* és az *e* adják meg.

ImageChar (PHP 3, PHP 4)

Egy karaktert rajzol vízszintesen

int **imagechar** (int im, int font, int x, int y, string c, int col) \linebreak

Az **ImageChar()** függvény megrajzolja a *c* string első betűjét *col* színnel az *id*-vel azonosított képen a betű bal felső sarkát az *x*,*y* koordinátához igazítva (a kép bal felső széle a 0, 0 pont). with a color . Ha *font* értéke 1, 2, 3, 4 vagy 5, egy beépített betűtípus kerül használatra (nagyobb szám nagyobb fontot jelent).

Lásd még az imageloadfont() funkciót.

ImageCharUp (PHP 3, PHP 4)

Egy karaktert rajzol függőlegesen

int **imagechar** (int im, int font, int x, int y, string c, int col) \linebreak

Az **ImageChar()** függvény megrajzolja a *c* string első betűjét *col* színnel az *id*-vel azonosított képen a betű bal felső sarkát az *x*,*y* koordinátához igazítva (a kép bal felső széle a 0, 0 pont). with a color . Ha *font* értéke 1, 2, 3, 4 vagy 5, egy beépített font kerül használatra (nagyobb szám nagyobb fontot jelent).

Lásd még az imageloadfont() funkciót.

ImageColorAllocate (PHP 3, PHP 4)

Egy színt foglal le egy képben

int **imagecolorallocate** (int im, int red, int green, int blue) \linebreak

Az **ImageColorAllocate()** egy színkóddal tér vissza, amely a megadotott RGB komponensekből áll. Az *im* paraméter az imagecreate() funkció által visszaadott érték. A *red*, *green* és *blue* paraméterek a szín vörös, zöld és a kék komponenseit határozzák meg. A komponensek értékei 0 és 255 közötti egész számok. Az **ImageColorAllocate()** függvényt meg kell hívnod az összes olyan színre, amelyet az *im*-mel azonosított képben használni akarsz.

```
$feher = ImageColorAllocate ($im, 255, 255, 255);
$fekete = ImageColorAllocate ($im, 0, 0, 0);
```

ImageColorAt (PHP 3, PHP 4)

Egy képpont színének indexét adja vissza

int **imagecolorat** (int im, int x, int y) \linebreak

A paraméterek által meghatározott képen és helyen lévő képpont színének indexével tér vissza.

Lásd még a imagecolorset() és a imagecolorsforindex() függvényeket.

ImageColorClosest (PHP 3, PHP 4)

Az adott színhez legközelebb álló szín indexét adja vissza

int **imagecolorclosest** (int im, int red, int green, int blue) \linebreak

A kép palettájának azon indexét adja vissza, amely a paraméterekben megadott RGB értékhez a "legközelebb" van.

A "távolság" a kívánt szín és a többi palettaszín között úgy kerül meghatározásra, mintha a színek RGB komponensét egy háromdimenziós térben ábrázolnánk, s a köztük levő távolságot mérnénk.

Lásd még a imagecolorexact() függvényt.

imagecolorclosestalpha (PHP 4 >= 4.0.6)

Get the index of the closest color to the specified color + alpha

int **imagecolorclosestalpha** (resource image, int red, int green, int blue, int alpha) \linebreak

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value and *alpha* level.

See also imagecolorexactalpha().

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.1

imagecolorclosesthwb (PHP 4 >= 4.0.1)

Get the index of the color which has the hue, white and blackness nearest to the given color

int **imagecolorclosesthwb** (resource image, int red, int green, int blue) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ImageColorDeAllocate (PHP 3>= 3.0.6, PHP 4)

Egy színt szabadít fel egy képből

int **imagecolordeallocate** (int im, int index) \linebreak

Az **ImageColorDeAllocate()** funkció felszabadít egy korábban az **ImageColorAllocate()** által lefoglalt színt.

```
$feher = ImageColorAllocate($im, 255, 255, 255);
ImageColorDeAllocate($im, $feher);
```

ImageColorExact (PHP 3, PHP 4)

A megadott szín palettabeli indexét adja vissza

int **imagecolorexact** (int im, int red, int green, int blue) \linebreak

Az adott szín palettablei indexét adja vissza. image.

Ha a szín nem szerepel a kép palettájában, -1-gyel tér vissza.

Lásd még a imagecolorclosest() függvényt.

imagecolorexactalpha (PHP 4 >= 4.0.6)

Get the index of the specified color + alpha

int **imagecolorexactalpha** (resource image, int red, int green, int blue, int alpha) \linebreak

Returns the index of the specified color+alpha in the palette of the image.

If the color does not exist in the image’s palette, -1 is returned.

See also imagecolorclosestalpha().

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

ImageColorResolve (PHP 3>= 3.0.2, PHP 4)

Az adott szín, vagy ahhoz legközelebbi szín palettaindexét adja vissza.

int **imagecolorresolve** (int im, int red, int green, int blue) \linebreak

A függvény garantáltan visszaad egy indexet a kívánt színre válaszul, vagy a szín palettaindexét, vagy a legközelebbi lehetséges alternatíváét.

Lásd még a imagecolorclosest() függvényt.

imagecolorresolvealpha (PHP 4 >= 4.0.6)

Get the index of the specified color + alpha or its closest possible alternative

int **imagecolorresolvealpha** (resource image, int red, int green, int blue, int alpha) \linebreak

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

See also imagecolorclosestalpha().

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.1

ImageColorSet (PHP 3, PHP 4)

Adott indexű palettaszín megváltoztatása

bool **imagecolorset** (int im, int index, int red, int green, int blue) \linebreak

A függvény az adott palettaindexű színt a paraméterekben megadott színűvé változtatja. Hasznos kitöltésszerű műveletekhez indexelt szervezésű képekben, mert nem kell a tényleges kitöltéshez szükséges plusz műveleteket végrehajtani.

Lásd még a imagecolorat() függvényt.

ImageColorsForIndex (PHP 3, PHP 4)

Adott indexű szín RGB komponensei

array **imagecolorsforindex** (int im, int index) \linebreak

Egy asszociatív tömbbel tér vissza, mely az adott indexű paletta vörös, zöld és kék komponenseit tartalmazza.

Lásd még a imagecolorat() és imagecolorexact() függvényeket.

ImageColorsTotal (PHP 3, PHP 4)

A kép palettájában levő színek számát adja

int **imagecolorstotal** (int im) \linebreak

A megadott kép színeinek számát adja vissza.

Lásd még a imagecolorat() és a imagecolorsforindex() függvényeket.

ImageColorTransparent (PHP 3, PHP 4)

Egy színt átlátszónak definiál

int **imagecolortransparent** (int im [, int col]) \linebreak

Az **ImageColorTransparent()** az *im* képben az átlátszó színt *col*-ra állítja. Az *im* az **ImageCreate()** által visszaadott érték és *col* pedig egy **ImageColorAllocate()** által visszaadott azonosító.

Az új vagy az aktuális átlátszó szín azonosítójával tér vissza.

ImageCopy (PHP 3>= 3.0.6, PHP 4)

Kép egy részét másolja

int **ImageCopy** (int dst_im, int src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h) \linebreak

A *src_im* kép egy részét *dst_im*-be másolja, a részlet másolását a *src_x*, *src_y* koordinátákkal kezdődik, a képrészlet szélessége *src_w*, magassága *src_h*. A részlet bal felső sarka a *dst_x*, *dst_y* koordinátára kerül.

imagecopymerge (PHP 4 >= 4.0.1)

Copy and merge part of an image

int **imagecopymerge** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h, int pct) \linebreak

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to imagecopy().

Megjegyzés: This function was added in PHP 4.0.6

imagecopymergegray (PHP 4 >= 4.0.6)

Copy and merge part of an image with gray scale

int **imagecopymergegray** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h, int pct) \linebreak

imagecopymergegray() copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to imagecopy().

This function is identical to imagecopymerge() except that when merging it preservese the hue of the source by converting the destination pixels to gray scale before the copy operation.

Megjegyzés: This function was added in PHP 4.0.6

imagecopyresampled (PHP 4 >= 4.0.6)

Copy and resize part of an image with resampling

int **imagecopyresampled** (resource dst_im, resource src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH) \linebreak

imagecopyresampled() copies a rectangular portion of one image to another image, smoothly interpolating pixel values so that, in particular, reducing the size of an image still retains a great deal of clarity. *Dst_im* is the destination image, *src_im* is the source image identifier. If the source and destination coordinates and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left corner. This function can be used to copy regions within the same image (if *dst_im* is the same as *src_im*) but if the regions overlap the results will be unpredictable.

See also imagecopyresized().

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

ImageCopyResized (PHP 3, PHP 4)

Másolja és átméretezni egy kép részletét

int **imagecopyresized** (int dst_im, int src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH) \linebreak

Az **ImageCopyResized()** függvény egy kép téglalap alakú részét másolja át egy másik képbe. *Dst_im* a tárgykép, *src_im* a forráskép azonosítója. Ha a forrás és a célkép koordinátái,

szélessége, magassága különbözik, a képdarabon nyújtás, összenyomás hajtódik végre. A koordináták a képdarabok bal felső koordinátái. Ezt a funkciót egyazon képen belüli másolásra is lehet használni (ha a *dst_im* és a *src_im* prarméter azonos), de ha az egyes részletek átfedik egymást, az eredmény kiszámíthatatlan.

ImageCreate (PHP 3, PHP 4)

Egy új képet hoz létre

int **imagecreate** (int x_size, int y_size) \linebreak

Az **ImageCreate()** függvény egy képazonosítóval tér vissza, mely üres, és mérete *x_size* * *y_size*.

imagecreatefromgd2 (PHP 4 >= 4.1.0)

Create a new image from GD2 file or URL

resource **imagecreatefromgd2** (string filename) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

imagecreatefromgd2part (PHP 4 >= 4.1.0)

Create a new image from a given part of GD2 file or URL

resource **imagecreatefromgd2part** (string filename, int srcX, int srcY, int width, int height) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

imagecreatefromgd (PHP 4 >= 4.1.0)

Create a new image from GD file or URL

resource **imagecreatefromgd** (string filename) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ImageCreateFromGif (PHP 3, PHP 4)

Fileból vagy URL-ből új képet hoz létre

int **imagecreatefromgif** (string filename) \linebreak

Az **ImageCreateFromGif()** egy képazonosítóval tér vissza, mely az adott filenévből létrehozott képet tartalmazza.

Az **ImageCreateFromGif()** üres stringgel tér vissza hiba esetén. Hibaüzenetet is ad, amit hibás linkként látunk a böngészőben A nyomkövetés megkönnyítésére a következő példa egy hibát jelző GIF-et állít elő.

Példa 1. Létrehozás közbeni hiba kezelése (vic@zymsys.com jóvoltából)

```
function LoadGif ($imgname) {
    $im = @imagecreatefromgif ($imgname); /* Kísérlet a megnyitásra */
    if ($im == "") { /* Megnézzük, hogy sikerült-e */
        $im = ImageCreate (150, 30); /* Üres kép létrehozása */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Hibaüzenet */
        ImageString($im, 1, 5, 5, "Hiba $imgname kép betöltésénél", $tc);
    }
    return $im;
}
```

Megjegyzés: Mivel mindenféle GIF támogatást az 1.6-os verzióban töröltek, ez a funkció nem elérhető, ha épp azt a verziót használod.

ImageCreateFromJpeg (PHP 3>= 3.0.16, PHP 4)

Fileból vagy URL-ből új képet hoz létre

int **imagecreatefromjpeg** (string filename) \linebreak

Az **ImageCreateFromJpeg()** egy képazonosítóval tér vissza, mely az adott filenévből létrehozott képet tartalmazza.

Az **ImageCreateFromJpeg()** üres stringgel tér vissza hiba esetén. Hibaüzenetet is ad, amit hibás linkként látunk a böngészőben A nyomkövetés megkönnyítésére a következő példa egy hibát jelző JPEG-et állít elő.

Példa 1. Létrehozás közbeni hiba kezelése (vic@zymsys.com jóvoltából)

```
function LoadJpeg ($imgname) {
    $im = @imagecreatefromjpeg ($imgname); /* Kísérlet a megnyitásra */
    if ($im == "") { /* Megnézzük, hogy sikerült-e */
        $im = ImageCreate (150, 30); /* Üres kép létrehozása */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Hibaüzenet */
        ImageString($im, 1, 5, 5, "Hiba $imgname kép betöltésénél", $tc);
    }
    return $im;
}
```

ImageCreateFromPng (PHP 3>= 3.0.13, PHP 4)

Fájlból vagy URL-ből új képet hoz létre

int **imagecreatefrompng** (string filename) \linebreak

Az **ImageCreateFromPng()** egy képazonosítóval tér vissza, mely az adott filenévből létrehozott képet tartalmazza.

Az **ImageCreateFromPng()** üres stringgel tér vissza hiba esetén. Hibaüzenetet is ad, amit hibás linkként látunk a böngészőben A nyomkövetés megkönnyítésére a következő példa egy hibát jelző PNG-t állít elő.

Példa 1. Létrehozás közbeni hiba kezelése (vic@zymsys.com jóvoltából)

```
function LoadPNG ($imgname) {
    $im = @imagecreatefrompng ($imgname); /* Kísérlet a megnyitásra */
    if ($im == "") { /* Megnézzük, hogy sikerült-e */
        $im = ImageCreate (150, 30); /* Üres kép létrehozása */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Hibaüzenet */
        ImageString($im, 1, 5, 5, "Hiba $imgname kép betöltésénél", $tc);
    }
    return $im;
}
```

imagecreatefromstring (PHP 4 >= 4.0.4)

Create a new image from the image stream in the string

resource **imagecreatefromstring** (string image) \linebreak

imagecreatefromstring() returns an image identifier representing the image obtained from the given string.

imagecreatefromwbmp (PHP 4 >= 4.0.1)

Create a new image from file or URL

resource **imagecreatefromwbmp** (string filename) \linebreak

imagecreatefromwbmp() returns an image identifier representing the image obtained from the given filename.

imagecreatefromwbmp() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error WBMP:

Példa 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadWBMP ($imgname) {
    $im = @imagecreatefromwbmp ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = imagecreate (20, 20); /* Create a blank image */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 10, 10, $bgc);
        /* Output an errmsg */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

Megjegyzés: WBMP support is only available if PHP was compiled against GD-1.8 or later.

imagecreatefromxbm (PHP 4 >= 4.0.1)

Create a new image from file or URL

resource **imagecreatefromxbm** (string filename) \linebreak

imagecreatefromxbm() returns an image identifier representing the image obtained from the given filename.

imagecreatefromxpm (PHP 4 >= 4.0.1)

Create a new image from file or URL

resource **imagecreatefromxpm** (string filename) \linebreak

imagecreatefromxpm() returns an image identifier representing the image obtained from the given filename.

imagecreatetruecolor (PHP 4 >= 4.0.6)

Create a new true color image

resource **imagecreatetruecolor** (int x_size, int y_size) \linebreak

imagecreatetruecolor() returns an image identifier representing a black image of size *`x_size`* by *`y_size`*.

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

ImageDashedLine (PHP 3, PHP 4)

Szaggatott vonalat rajzol

int **imagedashedline** (int im, int x1, int y1, int x2, int y2, int col) \linebreak

Az **ImageDashedLine()** függvény szaggatott vonalat rajzol az *`x1`*, *`y1`* pontból az *`x2`*, *`y2`* pontba az *`im`* képen *`col`* színnel. (A kép bal felső sarka a 0, 0 koordinátájú pont.)

Lásd még az **ImageLine()** függvényt.

ImageDestroy (PHP 3, PHP 4)

Megsemmisít egy képet

int **imagedestroy** (int im) \linebreak

Az **ImageDestroy()** függvény felszabadítja az *`im`* kép által lefoglalt memóriát. *`Im`* egy **ImageCreate()** függvény által visszaadott kép azonosító.

imageellipse (PHP 4 >= 4.0.6)

Draw an ellipse

int **imageellipse** (resource image, int cx, int cy, int w, int h, int col) \linebreak

imageellipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively. The color of the ellipse is specified by *color*.

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.2 or later

See also imagearc().

ImageFill (PHP 3, PHP 4)

Zárt terület kifestése

int **imagefill** (int im, int x, int y, int col) \linebreak

Az **ImageFill()** függvény az *x*, *y* koordinátától kezdve átszínez egy zárt területet az *im* képen *col* színűvé. (A kép bal felső sarka a 0, 0 koordinátájú pont.)

imagefilledarc (PHP 4 >= 4.0.6)

Draw a partial ellipse and fill it

int **imagefilledarc** (resource image, int cx, int cy, int w, int h, int s, int e, int col, int style) \linebreak

imagefilledarc() draws a partial ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively while the start and end points are specified in degrees indicated by the *s* and *e* arguments. *style* is a bitwise OR of the following possibilities:

1. IMG_ARC_PIE
2. IMG_ARC_CHORD
3. IMG_ARC_NOFILL
4. IMG_ARC_EDGED

IMG_ARC_PIE and IMG_ARC_CHORD are mutually exclusive; IMG_ARC_CHORD just connects the starting and ending angles with a straight line, while IMG_ARC_PIE produces a rounded edge. IMG_ARC_NOFILL indicates that the arc or chord should be outlined, not filled. IMG_ARC_EDGED, used together with IMG_ARC_NOFILL, indicates that the beginning and ending angles should be connected to the center - this is a good way to outline (rather than fill) a 'pie slice'.

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.1

imagefilledellipse (PHP 4 >= 4.0.6)

Draw a filled ellipse

int **imagefilledellipse** (resource image, int cx, int cy, int w, int h, int col) \linebreak

imagefilledellipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively. The ellipse is filled using *color*

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

See also imagefilledarc().

ImageFilledPolygon (PHP 3, PHP 4)

Kitöltött sokszöget rajzol

int **imagefilledpolygon** (int im, array points, int num_points, int col) \linebreak

Az **ImageFilledPolygon()** függvény az *im* képre egy kitöltött sokszöget rajzol. A *points* egy PHP tömb, amely a sokszög csúcspontjait tartalmazza. Például: points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, stb. *Num_points* a sokszög csúcsainak száma.

ImageFilledRectangle (PHP 3, PHP 4)

Kitöltött téglalapot rajzol

int **imagefilledrectangle** (int im, int x1, int y1, int x2, int y2, int col) \linebreak

Az **ImageFilledRectangle()** függvény egy **col()** színű kitöltött téglalapot rajzol az *im* képre. A téglalap bal felső koordinátája *x1*, *y1*, jobb alsó koordinátája pedig *x2*, *y2*. (A kép bal felső sarka a 0, 0 koordinátájú pont.)

Megjegyzés: Ha nem bal felső és jobb alsó koordinátákat adsz meg, akkor lehal a program, és a képből semmit sem fogsz látni (lehet, hogy bug...)!!! [angol doksiba!]

ImageFillToBorder (PHP 3, PHP 4)

Kitöltés adott határoló színig

int **imagefilltoborder** (int im, int x, int y, int border, int col) \linebreak

Az **ImageFillToBorder()** függvény a `border` szín eléréséig kitölt egy területet. A kitöltést az `x`, `y` koordinátában kezdi, a kitöltés színe `col` lesz. (A kép bal felső sarka a 0, 0 koordinátájú pont.)

ImageFontHeight (PHP 3, PHP 4)

Adott betűtípus magassága

int **imagefontheight** (int font) \linebreak

A függvény a megadott betűkészlet egy karakterének magasságát adja vissza pixelben.

Lásd még a **ImageFontWidth()** és a **ImageLoadFont()** függvényeket!

ImageFontWidth (PHP 3, PHP 4)

Adott betűtípus szélessége

int **imagefontwidth** (int font) \linebreak

A függvény a megadott betűkészlet egy karakterének szélességét adja vissza pixelben.

Lásd még a **ImageFontHeight()** és a **ImageLoadFont()** függvényeket.

imageftbbox (PHP 4 >= 4.1.0)

Give the bounding box of a text using fonts via freetype2

array **imageftbbox** (int size, int angle, string font_file, string text [, array extrainfo]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

imagefttext (PHP 4 >= 4.1.0)

Write text to the image using fonts using FreeType 2

array **imagefttext** (resource image, int size, int angle, int x, int y, int col, string font_file, string text [, array extrainfo]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ImageGammaCorrect (PHP 3>= 3.0.13, PHP 4)

A GD képre gamma korrekciót alkalmaz

int **imagegammacorrect** (int im, double inputgamma, double outputgamma) \linebreak

A **ImageGammaCorrect()** függvény gamma korrekciót alkalmaz egy gd képfolyamra (*`im`*). A korrekció mértékét a bemeneti és a kimeneti gamma értéke határozza meg (*`inputgamma`* és az *`outputgamma`*).

imagegd2 (PHP 4 >= 4.1.0)

Output GD2 image to browser or file

int **imagegd2** (resource image [, string filename]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

imagegd (PHP 4 >= 4.1.0)

Output GD image to browser or file

int **imagegd** (resource image [, string filename]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ImageGIF (PHP 3, PHP 4)

Kirajzol egy képet, vagy fájlba menti

int **imagegif** (int im [, string filename]) \linebreak

Az **ImageGIF()** függvény egy filename nevű GIF fájlt hoz létre az *`im`* képből. Az *`im`* paraméter az imagecreate() függvény által visszaadott azonosító.

A kép formátuma GIF87a lesz, ha a kép nem tartalmaz **ImageColorTransparent()** függvénnyel létrehozott átlátszó színt és GIF89a formátumú, ha tartalmaz átlátszó színt.

A filename paraméter opcionális, ha nem adod meg, akkor a kép közvetlenül a kimenetre kerül. Ha a lap tartalmát a header() függvénnyel image/gif-re állítod, akkor a PHP lap közvetlenül egy GIF típusú képet jelenít meg.

> **Megjegyzés:** Mivel minden GIF támogatást a GD könyvtár 1.6-os verziójából eltávolítottak, így ez a függvény nem elérhető, ha a GD e változatát használod.

ImageInterlace (PHP 3, PHP 4)

Interlace bit állítása

int **imageinterlace** (int im [, int interlace]) \linebreak

Az **ImageInterlace()** függvény az interlace bitet hivatott állítani. Ha értéke 1, az image "interlaced" lesz, ha 0, akkor nem lesz az.

A függvény visszatérési értéke az interlace bit aktuális értéke.

ImageJPEG (PHP 3>= 3.0.16, PHP 4)

Kirajzol egy képet, vagy fájlba menti

int **imagejpeg** (int im [, string filename [, int quality]]) \linebreak

Az **ImageJPEG()** függvény egy JPEG fájlt hoz létre az *`im`* képből. Az *`im`* paraméter a **ImageCreate()** függvény által visszaadott azonosító.

Ha a filename paraméter elmarad, akkor a kép a szabványos kimenetre kerül. Ha szeretnéd megadni a kép minőségét akkor, amikor a képet a kimenetre szeretnéd küldeni, akkor a kép nevének üres karakterláncot (”) adj meg! Ha egy image/jpg értékű content-type fejlécet írsz ki a header() függvénnyel, akkor a JPEG típusú kép közvetlenül a kimenetre küldhető.

> **Megjegyzés:** A JPEG támogatás csak akkor elérhető, ha a PHP-t GD-1.8 (vagy későbbi) változatával fordítod.

ImageLine (PHP 3, PHP 4)

Vonalat rajzol

int **imageline** (int im, int x1, int y1, int x2, int y2, int col) \linebreak

Az **ImageLine()** függvény vonalat rajzol az *x1*, *y1* pontból az *x2*, *y2* pontba *col* színnel. (A kép bal felső sarka a 0, 0 koordinátájú pont.)

Lásd még az **ImageCreate()** és a **ImageColorAllocate()** függvényeket!

ImageLoadFont (PHP 3, PHP 4)

Betölt egy betűkészletet

int **imageloadfont** (string file) \linebreak

Az **ImageLoadFont()** függvény egy felhasználói pixelgrafikus betűkészletet tölt be és a betöltött betűtípus azonosítóját adja vissza. (Ez mindig nagyobb, mint 5, hogy a beépített betűkészletekkel ne ütközzön.)

A font file formátuma jelenleg bináris és gépfüggő. Ez azt jelenti, hogy a betűkészletet azon a gépen kell létrehozni, mint amilyen típusú CPU-n fog futni a PHP, ami a betűkészletet használja.

Táblázat 1. Betűkészletek formátuma

byte pozíció	C adattípus	leírás
byte 0-3	int	a betűtípusban levő karakterek száma
byte 4-7	int	az első karakter kódja (gyakran 32, ez a szóköz kódja)
byte 8-11	int	egy karakter szélessége
byte 12-15	int	egy karakter magassága
byte 16-	char	tömb, amely az egyes karakterek adatait tartalmazza. Minden karakter egy pontot reprezentál, így az egész tömb hossza (nchars*width*height) byte hosszú.

Lásd még az **ImageFontWidth()** és a **ImageFontHeight()** függvényeket!

imagepalettecopy (PHP 4 >= 4.0.1)

Copy the palette from one image to another

int **imagepalettecopy** (resource destination, resource source) \linebreak

imagepalettecopy() copies the palette from the *source* image to the *destination* image.

ImagePng (PHP 3>= 3.0.13, PHP 4)

Kirajzol PNG egy képet, vagy fájlba menti

int **imagepng** (int im [, string filename]) \linebreak

Az **ImagePng()** függvény az *im* képet írja ki a szabványos kimenetre (a böngészőbe), vagy, ha *filename* paramétert is megadtál, akkor a kép tartalma abba a fájlba kerül.

```
$im = ImageCreateFromPng("test.png");
ImagePng($im);
```

ImagePolygon (PHP 3, PHP 4)

Sokszöget rajzol

int **imagepolygon** (int im, array points, int num_points, int col) \linebreak

Az **ImagePolygon()** függvény az id azonosítójú képre egy sokszöget rajzol. A *Points* tömb tartalmazza a sokszög csúcsait a következőképp: points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, stb. A *Num_points* adja meg a csúcspontok számát.

Lásd még az imagecreate() függvényt!

ImagePSBBox (PHP 3>= 3.0.9, PHP 4)

Szöveg köré rajzolt legkisebb téglalap méreteit adja PostScript Type1 betűkészletet használva

array **imagepsbbox** (string text, int font, int size [, int space [, int tightness [, float angle]]]) \linebreak

A *Size* pixelben értendő.

A *Space* paraméter segítségével megváltoztathatjuk a betűtípus alapértelmezett szóközét. Ez az érték a normál értékhez hozzáadódik, így van értelme negatív értéket is megadni.

A *Tightness* paraméterrel pedig a betűk közötti távolságot változtathatjuk. A paraméter értéke a karakter méretéhez hozzáadódik, így lehet negatív is.

Az *Angle* paramétert fokban kell megadni.

A *space* és a *tightness* paraméterek mértékegysége egy ezred em. (ennek az em-nek az m betű magasságához van köze)

A *space*, *tightness* és a *angle* paraméterek elhagyhatók.

A határoló doboz méreteit a függvény a fontból számítja, és sajnos néha eléggé különbözik ez az érték a ténylegesen megrajzolt szöveg méretétől. Ha a megadott szög 0 fok, akkor előfordulhat, hogy a szöveg a valóságban 1 pixelllel nagyobb lesz mindkét irányban.

A függvény egy tömböt ad vissza, melynek elemei:

0	bal alsó x-koordináta
1	bal alsó y-koordináta
2	jobb felső x-koordináta
3	jobb felső y-koordináta

Lásd még az imagepstext() függvényt!

ImagePSCopyFont (PHP 3>= 3.0.9, PHP 4)

Make a copy of an already loaded font for further modification

int **imagepscopyfont** (int fontindex) \linebreak

Use this function if you need make further modifications to a font, for example extending/condensing, slanting it or changing it's character encoding vector, but need to keep a original along as well. Note that a font you want to copy must be one obtained using **ImagePSLoadFont()**, not a font that is itself a copied one. You can although make modifications to it before copying.

If you use this function, you *must* free a fonts obtained this way yourself and in reverse order. Otherwise your script *will* hang.

In a case everything went right, a valid font index will be returned and can be used for further purposes. Otherwise a function returns false and prints a message describing what went wrong.

Lásd még a **ImagePSLoadFont()**.

ImagePSEncodeFont (PHP 3>= 3.0.9, PHP 4)

Megváltoztatja a betűkészlet egy karakterkódolási vektorát

int **imagepsencodefont** (string encodingfile) \linebreak

Fájlból betölt egy karakter-kódolási vektort és a font kódolási-vektorát erre változtatja. Mivel a PostScript betűkészletek alapértelmezett vektorában a legtöbb 127 feletti karakter máshol van, mint szeretnéd, így muszáj lesz megváltoztatnod, hacsak nem az angol nyelvet használod [nem azt használod, mert ezt a magyar leírást használod]. Ennek a betöltendő fájlnak a pontos formátuma megtalálható a T1libs dokumentációjában. [Aki nem hiszi, annak utánajárunk :)] A T1lib két használható fájllal rukkolt ki: az IsoLatin1.enc és az IsoLatin2.enc nevűekkel. [Ékes anyanyelvünk lelkes híveinek az utóbbit ajánlom!]

Ha azon kapod magad, hogy folyton ezt a függvényt használod, akkor jobban jársz, ha a kódolást a ps.default_encoding-ot a konfigurációs fájlban a megfelelő kódolási fájlra állítod, így betűkészleteid automatikusan jó kódolásúak lesznek.

ImagePsExtendFont (PHP 3>= 3.0.9, PHP 4)

Széthúz vagy zsugorít egy betűkészletet

bool **imagepsextendfont** (int font_index, double extend) \linebreak

Széthúz vagy zsugorít egy betűkészletet (a *font_index* paraméterben megadottat). Ha az *extend* paraméter értéke egynél kisebb, a betűkészletet összenyomja.

ImagePSFreeFont (PHP 3>= 3.0.9, PHP 4)

Egy PostScript Type 1 betűtípus által használt memóriát szabadít fel

void **imagepsfreefont** (int fontindex) \linebreak

Lásd még a **ImagePSLoadFont()** függvényt!

ImagePSLoadFont (PHP 3>= 3.0.9, PHP 4)

Fájlból betölt egy PostScript Type 1 betűtípust

int **imagepsloadfont** (string filename) \linebreak

Ha minden jól megy, akkor egy érvényes betűtípus-azonosítót kapsz vissza. Ha nincs szerencséd, a függvény hamissal tér vissza, és kiírja, hogy mi a baj. [ennek nem sok hasznát veszed, hiszen a kimenet típusa kép...]

Lásd még a **ImagePSFreeFont()** függvényt!

ImagePsSlantFont (PHP 3>= 3.0.9, PHP 4)

Dönt egy betűkészletet

bool **imagepsslantfont** (int font_index, double slant) \linebreak

A *font_index* paraméterben megadott betűkészletet dönti a *slant* parametérben megadott értékkel.

ImagePSText (PHP 3>= 3.0.9, PHP 4)

Egy képbe PostScript Type1 típusú szöveget ír

array **imagepstext** (int image, string text, int font, int size, int foreground, int background, int x, int y [, int space [, int tightness [, float angle [, int antialias_steps]]]]) \linebreak

`Size` pixelben.

A `foreground` az a szín, amivel a szöveg kiíródik. A `background` az a szín, amelyet háttérszínnek feltételez a rajzolóalgoritmus (antialiasing miatt kell). A függvény egyetlen `background` színű pontot sem fog rajzolni, így a háttérnek nem kell egyszínűnek lennie.

Az `x`, `y` által adott paraméterek fogják a szöveg kezdetét meghatározni (durván az első karakter bal alsó sarkát). Ez egy különbség a **ImageString()**, függvénytől, ahol az `x`, `y` paraméterek az első karakter jobb felső [vagy bal felső] sarkát határozzák meg. Ha nem érted, hogy miért úgy rajzol a függvény, olvasd a PostScript dokumentációját a betűkészletekkel meg a koordinátarendszerekkel kapcsolatban!

A `space` paraméter segítségével a betűkészlet alapértelmezett térközének méretét tudod szabályozni. A megadott mennyiség a normál térközhöz hozzáadódik, így van értelme negatív térközt is megadni.

A `tightness` paraméter segítségével a betűk közötti térköz nagyságán tudsz változtatni. A megadott mennyiség a normál térközhöz hozzáadódik, így negatív is lehet.

A `angle` paraméter fokban értendő.

Az `antialias_steps` paraméter segítségével az antialiasing-hoz használt színek számát tudod szabályozni. A megengedett értékek: 4 és 16. 20-nál kisebb betűméret esetén a nagyobb értéket célszerű használni. Nagyobb betűkre használj 4-et, mert így gyorsabb a betűk rajzolása!

A `space` és a `tightness` paraméterek egysége a "karakter távolság egység", ami ezrede az em-nek [0.001em, jól mondom?].

A `space`, `tightness`, `angle` és a `antialias` paraméterek elhagyhatók.

A függvény egy tömbbel tér vissza, melynek az alábbi elemei vannak:

0	bal alsó x-koordináta
1	bal alsó y-koordináta
2	jobb felső x-koordináta
3	jobb felső y-koordináta

Lásd még az imagepsbbox() függvényt!

ImageRectangle (PHP 3, PHP 4)

Téglalapot rajzol

int **imagerectangle** (int im, int x1, int y1, int x2, int y2, int col) \linebreak

A **ImageRectangle()** függvény egy col színű téglalapot rajzol az im képre. A téglalap bal felső koordinátái x1, y1, a jobb alsó koordináták pedig x2, y2. A kép bal felső sarka a 0,0 koordináta.

imagesetbrush (PHP 4 >= 4.0.6)

Set the brush image for line drawing

int **imagesetbrush** (resource image, resource brush) \linebreak

imagesetbrush() sets the brush image to be used by all line drawing functions (such as imageline() and imagepolygon()) when drawing with the special colors `IMG_COLOR_BRUSHED` or `IMG_COLOR_STYLEDBRUSHED`.

Megjegyzés: You need not take special action when you are finished with a brush, but if you destroy the brush image, you must not use the `IMG_COLOR_BRUSHED` or `IMG_COLOR_STYLEDBRUSHED` colors until you have set a new brush image!

Megjegyzés: This function was added in PHP 4.0.6

ImageSetPixel (PHP 3, PHP 4)

Egy pont színét megváltoztatja

int **imagesetpixel** (int im, int x, int y, int col) \linebreak

A **ImageSetPixel()** függvény az *`im`* képben *`x`*, *`y`* pont (A kép bal felső sarka a 0,0 pont) színét a *`col`* színűre változtatja.

Lásd még a **ImageCreate()** és a **ImageColorAllocate()** függvényeket.

imagesetstyle (PHP 4 >= 4.0.6)

Set the style for line drawing

int **imagesetstyle** (resource image, array style) \linebreak

imagesetstyle() sets the style to be used by all line drawing functions (such as imageline() and imagepolygon()) when drawing with the special color `IMG_COLOR_STYLED` or lines of images with color `IMG_COLOR_STYLEDBRUSHED`.

The *`style`* parameter is an array of pixels. Following example script draws a dashed line from upper left to lower right corner of the canvas:

Példa 1. imagesetstyle

```
<?php
header ("Content-type: image/png");
$im  = imagecreate (100, 100);
$w   = imagecolorallocate ($im, 255, 255, 255);
```

```
$red = imagecolorallocate ($im, 255, 0, 0);

/* Draw a dashed line, 5 red pixels, 5 white pixels */
$style = array ($red,$red,$red,$red,$red,$w,$w,$w,$w,$w);
imagesetstyle ($im, $style);
imageline ($im, 0, 0, 100, 100, IMG_COLOR_STYLED);

/* Draw a line of happy faces using imagesetbrush() with imagesetstyle */
$style = array ($w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$red);
imagesetstyle ($im, $style);

$brush = imagecreatefrompng ("http://www.libpng.org/pub/png/images/smile.happy.png");
imagecolortransparent ($brush, $w);
imagesetbrush ($im, $brush);
imageline ($im, 100, 0, 0, 100, IMG_COLOR_STYLEDBRUSHED);

imagepng ($im);
imagedestroy ($im);
?>
```

See also imagesetbrush(), imageline().

Megjegyzés: This function was added in PHP 4.0.6

imagesetthickness (PHP 4 >= 4.0.6)

Set the thickness for line drawing

void **imagesetthickness** (resource image, int thickness) \linebreak

imagesetthickness() sets the thickness of the lines drawn when drawing rectangles, polygons, ellipses etc. etc. to *`thickness`* pixels.

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

imagesettile (PHP 4 >= 4.0.6)

Set the tile image for filling

int **imagesettile** (resource image, resource tile) \linebreak

imagesettile() sets the tile image to be used by all region filling functions (such as imagefill() and imagefilledpolygon()) when filling with the special color `IMG_COLOR_TILED`.

A tile is an image used to fill an area with a repeated pattern. *Any* GD image can be used as a tile, and by setting the transparent color index of the tile image with imagecolortransparent(), a tile allows certain parts of the underlying area to shine through can be created.

Megjegyzés: You need not take special action when you are finished with a tile, but if you destroy the tile image, you must not use the `IMG_COLOR_TILED` color until you have set a new tile image!

Megjegyzés: This function was added in PHP 4.0.6

ImageString (PHP 3, PHP 4)

Karakterláncot ír ki (vízszintesen)

int **imagestring** (int im, int font, int x, int y, string s, int col) \linebreak

Az **ImageString()** függvény kiírja az *s* karakterláncot az *im* képen az *x*, *y* (a bal felső sarok a 0, 0 pont) ponttól *col* színnel. Ha a *font* értéke 1, 2, 3, 4 vagy 5, egy beépített betűkészlet kerül felhasználásra.

Lásd még az **ImageLoadFont()** függvényt!

ImageStringUp (PHP 3, PHP 4)

Kiír egy karakterláncot lentről fel [mint ahogy a :-)-t kell nézni]

int **imagestringup** (int im, int font, int x, int y, string s, int col) \linebreak

Az **ImageStringUp()** függvény kiírja az *s* karakterláncot függőlegesen az *im* képen az *x*, *y* (a bal felső sarok a 0, 0 pont) ponttól *col* színnel. Ha a *font* értéke 1, 2, 3, 4 vagy 5, egy beépített betűkészlet kerül felhasználásra.

Lásd még az **ImageLoadFont()** függvényt!

ImageSX (PHP 3, PHP 4)

Kép szélessége

int **imagesx** (int im) \linebreak

Az **ImageSX()** függvény az *im* álala azonosított kép szélességét adja vissza.

Lásd még a **ImageCreate()** és a **ImageSY()** függvényeket!

ImageSY (PHP 3, PHP 4)

Kép magassága

int **imagesy** (int im) \linebreak

Az **ImageSY()** függvény az *`im`* álala azonosított kép magasságát adja vissza.

Lásd még a **ImageCreate()** és a **ImageSX()** függvényeket!

imagetruecolortopalette (PHP 4 >= 4.0.6)

Convert a true color image to a palette image

void **imagetruecolortopalette** (resource image, bool dither, int ncolors) \linebreak

imagetruecolortopalette() converts a truecolor image to a palette image. The code for this function was originally drawn from the Independent JPEG Group library code, which is excellent. The code has been modified to preserve as much alpha channel information as possible in the resulting palette, in addition to preserving colors as well as possible. This does not work as well as might be hoped. It is usually best to simply produce a truecolor output image instead, which guarantees the highest output quality.

`dither` indicates if the image should be dithered - if it is `TRUE` then dithering will be used which will result in a more speckled image but with better color approximation.

`ncolors` sets the maximum number of colors that should be retained in the palette.

Megjegyzés: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

ImageTTFBBox (PHP 3>= 3.0.1, PHP 4)

TruType betűtípusú szöveg befoglaló téglalapját adja vissza.

array **imagettfbbox** (int size, int angle, string fontfile, string text) \linebreak

A függvény kiszámítja és visszaadja a TrueType szöveg befoglaló téglalap méretét (képpontban).

`text`

a megmérendő szöveg

`size`

a betűk nagysága

`fontfile`

A TrueType betűtípus fájl neve. (Lehet URL is)

angle

Szög fokban, amiben a *text* paramétert mérjük. [???]

Az **ImageTTFBBox()** függvény egy 8-elemű tömbbel tér vissza, amely a befoglaló négy pont koordinátáit tartalmazza:

0	bal alsó sarok, X koordináta
1	bal alsó sarok, Y koordináta
2	jobb alsó sarok, X koordináta
3	jobb alsó sarok, Y koordináta
4	jobb felső sarok, X koordináta
5	jobb felső sarok, Y koordináta
6	bal felső sarok, X koordináta
7	bal felső sarok, Y koordináta

A pontok a *szöveghez* képest értendők, tehát a "bal felső" azt jelenti, hogy a szöveget normálisan nézve bal felső.

E függvény használatához a GD és a FreeType könyvtárak is kellenek.

Lásd még az **ImageTTFText()** függvényt!

ImageTTFText (PHP 3, PHP 4)

TrueType típusú szöveget ír ki egy képre

array **imagettftext** (int im, int size, int angle, int x, int y, int col, string fontfile, string text) \linebreak

A **ImageTTFText()** függvény az *im* képre írja ki a *text* szöveget az *x*, *y* koordinátákra *angle* szöggel elforgatva *col* színnel a *fontfile* betűtípust felhasználva.

Az *x*, *y* által adott paraméterek fogják a szöveg kezdetét meghatározni (durván az első karakter bal alsó sarkát). Ez egy különbség a **ImageString()**, függvénytől, ahol az *x*, *y* paraméterek az első karakter jobb felső [talán bal felső] sarkát határozzák meg.

Az *angle* paraméter értéke fokban értendő. Ha értéke 0, a szöveg "normális" balról-jobbra olvasható szöveg. Pozitív értékek a szöveget pozitív (óramutató járásával ellentétes) irányba forgatja. (Pl.: 90-es elforgatásnál a szöveget lentről felfelé kell olvasni)

A *fontfile* paraméter egy TrueType betűtípus teljes elérési útvonala.

A *text* paraméter a szöveg, mely tartalmazhat UTF-8 karaktersorozatokat ({ formában) a 255 feletti kódú karakterek eléréséhez.

A *col* paraméter egy szín-index. Ha értéke negatív, az kikapcsolja az antialiasing-ot.

Az **ImageTTFText()** függvény egy 8-elemű tömböt ad vissza, mely a szöveget befoglaló 4 pont koordinátáit tartalmazza. A pontok sorrendje: bal felső, jobb felső, jobb alsó, bal alsó. A pontok a szöveghez képest relatívok, vagyis a szöveget olvasva értendők.

Az alábbi példaprogram egy fekete 400x30-as GIF képet eredményez egy fehér Arial típusú "Próba" kezdetű szöveggel. [Feltéve, ha nem túl új a GD-d mert a gif támogatást kiszedték :(]

Példa 1. ImageTTFText

```
<?php
Header ("Content-type: image/gif");
$im = imagecreate (400, 30);
$black = ImageColorAllocate ($im, 0, 0, 0);
$white = ImageColorAllocate ($im, 255, 255, 255);
ImageTTFText ($im, 20, 0, 10, 20, $white, "/path/arial.ttf",
              "Próba Omega: &#937;");
ImageGif ($im);
ImageDestroy ($im);
?>
```

A függvény használatához a GD könyvtáron kívül a FreeType (http://www.freetype.org/) könyvtár is szükséges.

Lásd még a **ImageTTFBBox()** függvényt!

imagetypes (PHP 3 CVS only, PHP 4 >= 4.0.2)

Return the image types supported by this PHP build

int **imagetypes** (void) \linebreak

This function returns a bit-field corresponding to the image formats supported by the version of GD linked into PHP. The following bits are returned, `IMG_GIF` | `IMG_JPG` | `IMG_PNG` | `IMG_WBMP`. To check for PNG support, for example, do this:

Példa 1. imagetypes

```
<?php
if (imagetypes() & IMG_PNG) {
    echo "PNG Support is enabled";
}
?>
```

imagewbmp (PHP 3>= 3.0.15, PHP 4 >= 4.0.1)

Output image to browser or file

int **imagewbmp** (resource image [, string filename [, int foreground]]) \linebreak

imagewbmp() creates the WBMP file in filename from the image *`image`*. The *`image`* argument is the return from the imagecreate() function.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using header(), you can create a PHP script that outputs WBMP images directly.

Megjegyzés: WBMP support is only available if PHP was compiled against GD-1.8 or later.

Using the optional *foreground* parameter, you can set the foreground color. Use an identifier obtained from imagecolorallocate(). The default foreground color is black.

See also image2wbmp(), imagepng(), imagegif(), imagejpeg(), imagetypes().

iptcembed (PHP 3>= 3.0.7, PHP 4)

Embed binary IPTC data into a JPEG image

array **iptcembed** (string iptcdata, string jpeg_file_name [, int spool]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

iptcparse (PHP 3>= 3.0.6, PHP 4)

Parse a binary IPTC http://www.iptc.org/ block into single tags.

array **iptcparse** (string iptcblock) \linebreak

This function parses a binary IPTC block into its single tags. It returns an array using the tagmarker as an index and the value as the value. It returns FALSE on error or if no IPTC data was found. See getimagesize() for a sample.

jpeg2wbmp (PHP 4 >= 4.0.5)

Convert JPEG image file to WBMP image file

int **jpeg2wbmp** (string jpegname, string wbmpname, int d_height, int d_width, int threshold) \linebreak

Converts the *jpegname* JPEG file to WBMP format, and saves it as *wbmpname*. With the *d_height* and *d_width* you specify the height and width of the destination image.

Megjegyzés: WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also png2wbmp().

png2wbmp (PHP 4 >= 4.0.5)

Convert PNG image file to WBMP image file

int **png2wbmp** (string pngname, string wbmpname, int d_height, int d_width, int threshold) \linebreak

Converts the *`pngname`* PNG file to WBMP format, and saves it as *`wbmpname`*. With the *`d_height`* and *`d_width`* you specify the height and width of the destination image.

Megjegyzés: WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also jpeg2wbmp().

read_exif_data (PHP 4 >= 4.0.1)

Reads header information stored in TIFF and JPEG images

array **exif_read_data** (string filename, string sections, bool arrays, bool thumbnail) \linebreak

Megjegyzés: The **read_exif_data()** function is an alias for exif_read_data().

See also exif_thumbnail().

XLIII. IMAP, POP3 and NNTP functions

Bevezetés

These functions are not limited to the IMAP protocol, despite their name. The underlying c-client library also supports NNTP, POP3 and local mailbox access methods.

Követelmények

This extension requires the c-client library to be installed. Grab the latest version from ftp://ftp.cac.washington.edu/imap/ and compile it.

Telepítés

To get these functions to work, you have to compile PHP with `--with-imap`.

Then copy `c-client/c-client.a` to `/usr/local/lib/libc-client.a` or some other directory on your link path and copy `c-client/c-client.h`, `c-client/imap4r1.h`, `c-client/rfc-882.h`, `c-client/mail.h` and `c-client/linkage.h` to `/usr/local/include` or some other directory in your include path.

> **Megjegyzés:** Depending how the c-client was configured, you might also need to add `--with-imap-ssl=/path/to/openssl/` and/or `--with-kerberos` into the PHP configure line.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Előre definiált állandók

Az itt listázott állandókat ez a kiterjesztés definiálja, és csak akkor elérhetőek, ha az adott

kiterjesztés be van fordítva a PHP-be, vagy dinamikusan betöltött.

`NIL` (integer)

`OP_DEBUG` (integer)

`OP_READONLY` (integer)

`OP_ANONYMOUS` (integer)

`OP_SHORTCACHE` (integer)

`OP_SILENT` (integer)

`OP_PROTOTYPE` (integer)

`OP_HALFOPEN` (integer)

`OP_EXPUNGE` (integer)

`OP_SECURE` (integer)

`CL_EXPUNGE` (integer)

`FT_UID` (integer)

`FT_PEEK` (integer)

`FT_NOT` (integer)

`FT_INTERNAL` (integer)

`FT_PREFETCHTEXT` (integer)

`ST_UID` (integer)

`ST_SILENT` (integer)

`ST_SET` (integer)

information is provided by the documentation of the c-client library source (`docs/internal.txt`). and the following RFC documents:

- RFC2821 (http://www.faqs.org/rfcs/rfc2821.html): Simple Mail Transfer Protocol (SMTP).
- RFC2822 (http://www.faqs.org/rfcs/rfc2822.html): Standard for ARPA internet text messages.
- RFC2060 (http://www.faqs.org/rfcs/rfc2060.html): Internet Message Access Protocol (IMAP) Version 4rev1.
- RFC1939 (http://www.faqs.org/rfcs/rfc1939.html): Post Office Protocol Version 3 (POP3).
- RFC977 (http://www.faqs.org/rfcs/rfc977.html): Network News Transfer Protocol (NNTP).
- RFC2076 (http://www.faqs.org/rfcs/rfc2076.html): Common Internet Message Headers.
- RFC2045 (http://www.faqs.org/rfcs/rfc2045.html) , RFC2046 (http://www.faqs.org/rfcs/rfc2046.html) , RFC2047 (http://www.faqs.org/rfcs/rfc2047.html) , RFC2048 (http://www.faqs.org/rfcs/rfc2048.html) & RFC2049 (http://www.faqs.org/rfcs/rfc2049.html): Multipurpose Internet Mail Extensions (MIME).

A detailed overview is also available in the books Programming Internet Email (http://www.oreilly.com/catalog/progintemail/noframes.html) by David Wood and Managing IMAP (http://www.oreilly.com/catalog/mimap/noframes.html) by Dianna Mullet & Kevin Mullet.

Figyelem

Crashes and startup problems of PHP may be encountered when loading this extension in conjunction with the recode extension. See the recode extension for more information.

imap_8bit (PHP 3, PHP 4)

Convert an 8bit string to a quoted-printable string

string **imap_8bit** (string string) \linebreak

Convert an 8bit string to a quoted-printable string (according to RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), section 6.7).

Returns a quoted-printable string.

See also imap_qprint().

imap_alerts (PHP 3>= 3.0.12, PHP 4)

This function returns all IMAP alert messages (if any) that have occurred during this page request or since the alert stack was reset

array **imap_alerts** (void) \linebreak

This function returns an array of all of the IMAP alert messages generated since the last **imap_alerts()** call, or the beginning of the page. When **imap_alerts()** is called, the alert stack is subsequently cleared. The IMAP specification requires that these messages be passed to the user.

imap_append (PHP 3, PHP 4)

Append a string message to a specified mailbox

int **imap_append** (int imap_stream, string mbox, string message [, string flags]) \linebreak

Returns `TRUE` on sucess, `FALSE` on error.

imap_append() appends a string message to the specified mailbox *`mbox`*. If the optional *`flags`* is specified, writes the *`flags`* to that mailbox also.

When talking to the Cyrus IMAP server, you must use "\r\n" as your end-of-line terminator instead of "\n" or the operation will fail.

Példa 1. imap_append() example

```
$stream = imap_open("{your.imap.host}INBOX.Drafts","username", "password");

$check = imap_check($stream);
print "Msg Count before append: ". $check->Nmsgs."\n";

imap_append($stream,"{your.imap.host}INBOX.Drafts"
                  ,"From: me@my.host\r\n"
                  ."To: you@your.host\r\n"
                  ."Subject: test\r\n"
                  ."\r\n"
                  ."this is a test message, please ignore\r\n"
```

```
                    );

$check = imap_check($stream);
print "Msg Count after append : ". $check->Nmsgs."\n";

imap_close($stream);
```

imap_base64 (PHP 3, PHP 4)

Decode BASE64 encoded text

string **imap_base64** (string text) \linebreak

imap_base64() function decodes BASE-64 encoded text (see RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), Section 6.8). The decoded message is returned as a string.

See also imap_binary().

imap_binary (PHP 3>= 3.0.2, PHP 4)

Convert an 8bit string to a base64 string

string **imap_binary** (string string) \linebreak

Convert an 8bit string to a base64 string (according to RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), Section 6.8).

Returns a base64 string.

See also imap_base64().

imap_body (PHP 3, PHP 4)

Read the message body

string **imap_body** (int imap_stream, int msg_number [, int flags]) \linebreak

imap_body() returns the body of the message, numbered *`msg_number`* in the current mailbox. The optional *`flags`* are a bit mask with one or more of the following:

- FT_UID - The *`msgno`* is a UID
- FT_PEEK - Do not set the \Seen flag if not already set
- FT_INTERNAL - The return string is in internal format, will not canonicalize to CRLF.

imap_body() will only return a verbatim copy of the message body. To extract single parts of a multipart MIME-encoded message you have to use imap_fetchstructure() to analyze its structure and imap_fetchbody() to extract a copy of a single body component.

imap_bodystruct (PHP 3>= 3.0.4, PHP 4)

Read the structure of a specified body section of a specific message

object **imap_bodystruct** (int stream_id, int msg_no, int section) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

imap_check (PHP 3, PHP 4)

Check current mailbox

object **imap_check** (int imap_stream) \linebreak

Returns information about the current mailbox. Returns FALSE on failure.

The **imap_check()** function checks the current mailbox status on the server and returns the information in an object with following properties:

- Date - last change of mailbox contents
- Driver - protocol used to access this mailbox: POP3, IMAP, NNTP
- Mailbox - the mailbox name
- Nmsgs - number of messages in the mailbox
- Recent - number of recent messages in the mailbox

imap_clearflag_full (PHP 3>= 3.0.3, PHP 4)

Clears flags on messages

string **imap_clearflag_full** (int stream, string sequence, string flag, string options) \linebreak

This function causes a store to delete the specified flag to the flags set for the messages in the specified sequence. The flags which you can unset are "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", and "\\Draft" (as defined by RFC2060).

The options are a bit mask with one or more of the following:

ST_UID The sequence argument contains UIDs instead of sequence numbers

imap_close (PHP 3, PHP 4)

Close an IMAP stream

int **imap_close** (int imap_stream [, int flags]) \linebreak

Close the imap stream. Takes an optional *flag* CL_EXPUNGE, which will silently expunge the mailbox before closing, removing all messages marked for deletion.

imap_createmailbox (PHP 3, PHP 4)

Create a new mailbox

int **imap_createmailbox** (int imap_stream, string mbox) \linebreak

imap_createmailbox() creates a new mailbox specified by *mbox*. Names containing international characters should be encoded by imap_utf7_encode()

Returns TRUE on success and FALSE on error.

See also imap_renamemailbox(), imap_deletemailbox() and imap_open() for the format of *mbox* names.

Példa 1. imap_createmailbox() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
     or die("can't connect: ".imap_last_error());

$name1 = "phpnewbox";
$name2 = imap_utf7_encode("phpnewb&ouml;x");

$newname = $name1;

echo "Newname will be '$name1'<br>\n";

# we will now create a new mailbox "phptestbox" in your inbox folder,
# check its status after creation and finaly remove it to restore
# your inbox to its initial state
if(@imap_createmailbox($mbox,imap_utf7_encode("{your.imap.host}INBOX.$newname"))) {
  $status = @imap_status($mbox,"{your.imap.host}INBOX.$newname",SA_ALL);
  if($status) {
    print("your new mailbox '$name1' has the following status:<br>\n");
    print("Messages:   ". $status->messages   )."<br>\n";
```

```
    print("Recent:     ". $status->recent     )."<br>\n";
    print("Unseen:     ". $status->unseen     )."<br>\n";
    print("UIDnext:    ". $status->uidnext    )."<br>\n";
    print("UIDvalidity:". $status->uidvalidity)."<br>\n";

    if(imap_renamemailbox($mbox,"{your.imap.host}INBOX.$newname","{your.imap.host}INBOX.$
      echo "renamed new mailbox from 'name1' to 'name2'<br>\n";
      $newname=$name2;
    } else {
      print "imap_renamemailbox on new mailbox failed: ".imap_last_error()."<br>\n";
    }
  } else {
    print  "imap_status on new mailbox failed: ".imap_last_error()."<br>\n";
  }
  if(@imap_deletemailbox($mbox,"{your.imap.host}INBOX.$newname")) {
    print "new mailbox removed to restore initial state<br>\n";
  } else {
    print  "imap_deletemailbox on new mailbox failed: ".implode("<br>\n",imap_errors()).'
  }

} else {
  print  "could not create new mailbox: ".implode("<br>\n",imap_errors())."<br>\n";
}

imap_close($mbox);
```

imap_delete (PHP 3, PHP 4)

Mark a messge for deletion from current mailbox

int **imap_delete** (int imap_stream, int msg_number [, int flags]) \linebreak

Returns TRUE.

imap_delete() marks messages listed in *msg_number* for deletion. The optional *flags* parameter only has a single option, *FT_UID*, which tells the function to treat the *msg_number* argument as a *UID*. Messages marked for deletion will stay in the mailbox until either imap_expunge() is called or imap_close() is called with the optional parameter CL_EXPUNGE.

Megjegyzés: POP3 mailboxes do not have their message flags saved between connections, so imap_expunge() must be called during the same connection in order for messages marked for deletion to actually be purged.

Példa 1. imap_delete() Beispiel

```
$mbox = imap_open ("{your.imap.host}INBOX", "username", "password")
    or die ("can't connect: " . imap_last_error());

$check = imap_mailboxmsginfo ($mbox);
print "Messages before delete: " . $check->Nmsgs . "<br>\n" ;
imap_delete ($mbox, 1);
$check = imap_mailboxmsginfo ($mbox);
print "Messages after  delete: " . $check->Nmsgs . "<br>\n" ;
imap_expunge ($mbox);
$check = imap_mailboxmsginfo ($mbox);
print "Messages after expunge: " . $check->Nmsgs . "<br>\n" ;
imap_close ($mbox);
```

imap_deletemailbox (PHP 3, PHP 4)

Delete a mailbox

int **imap_deletemailbox** (int imap_stream, string mbox) \linebreak

imap_deletemailbox() deletes the specified mailbox (see imap_open() for the format of *mbox* names).

Returns TRUE on success and FALSE on error.

See also imap_createmailbox(), imap_renamemailbox(), and imap_open() for the format of *mbox*.

imap_errors (PHP 3>= 3.0.12, PHP 4)

This function returns all of the IMAP errors (if any) that have occurred during this page request or since the error stack was reset.

array **imap_errors** (void) \linebreak

This function returns an array of all of the IMAP error messages generated since the last **imap_errors()** call, or the beginning of the page. When **imap_errors()** is called, the error stack is subsequently cleared.

imap_expunge (PHP 3, PHP 4)

Delete all messages marked for deletion

int **imap_expunge** (int imap_stream) \linebreak

imap_expunge() deletes all the messages marked for deletion by imap_delete(), imap_mail_move(), or imap_setflag_full().

Returns TRUE.

imap_fetch_overview (PHP 3>= 3.0.4, PHP 4)

Read an overview of the information in the headers of the given message

array **imap_fetch_overview** (int imap_stream, string sequence [, int flags]) \linebreak

This function fetches mail headers for the given *sequence* and returns an overview of their contents. *sequence* will contain a sequence of message indices or UIDs, if *flags* contains FT_UID. The returned value is an array of objects describing one message header each:

- subject - the messages subject
- from - who sent it
- date - when was it sent
- message_id - Message-ID
- references - is a reference to this message id
- size - size in bytes
- uid - UID the message has in the mailbox
- msgno - message sequence number in the maibox
- recent - this message is flagged as recent
- flagged - this message is flagged
- answered - this message is flagged as answered
- deleted - this message is flagged for deletion
- seen - this message is flagged as already read
- draft - this message is flagged as being a draft

Példa 1. imap_fetch_overview() example

```
$mbox = imap_open("{your.imap.host:143}","username","password")
     or die("can’t connect: ".imap_last_error());

$overview = imap_fetch_overview($mbox,"2,4:6",0);

if(is_array($overview)) {
        reset($overview);
        while( list($key,$val) = each($overview)) {
                print     $val->msgno
                . " - " . $val->date
                . " - " . $val->subject
                . "\n";
        }
```

```
}

imap_close($mbox);
```

imap_fetchbody (PHP 3, PHP 4)

Fetch a particular section of the body of the message

string **imap_fetchbody** (int imap_stream, int msg_number, string part_number [, flags flags]) \linebreak

This function causes a fetch of a particular section of the body of the specified messages as a text string and returns that text string. The section specification is a string of integers delimited by period which index into a body part list as per the IMAP4 specification. Body parts are not decoded by this function.

The options for **imap_fetchbody()** is a bitmask with one or more of the following:

- FT_UID - The *msg_number* is a UID
- FT_PEEK - Do not set the \Seen flag if not already set
- FT_INTERNAL - The return string is in "internal" format, without any attempt to canonicalize CRLF.

See also: imap_fetchstructure().

imap_fetchheader (PHP 3>= 3.0.3, PHP 4)

Returns header for a message

string **imap_fetchheader** (int imap_stream, int msgno, int flags) \linebreak

This function causes a fetch of the complete, unfiltered RFC2822 (http://www.faqs.org/rfcs/rfc2822.html) format header of the specified message as a text string and returns that text string.

The options are:

FT_UID The msgno argument is a UID
FT_INTERNAL The return string is in "internal" format, without any attempt to canonicalize to CRLF newlines
FT_PREFETCHTEXT The RFC822.TEXT should be pre-fetched at the same time. This avoids an extra RTT on an IMAP connection if a full message text is desired (e.g. in a "save to local file" operation)

imap_fetchstructure (PHP 3, PHP 4)

Read the structure of a particular message

object **imap_fetchstructure** (int imap_stream, int msg_number [, int flags]) \linebreak

This function fetches all the structured information for a given message. The optional *`flags`* parameter only has a single option, *`FT_UID`*, which tells the function to treat the *`msg_number`* argument as a *`UID`*. The returned object includes the envelope, internal date, size, flags and body structure along with a similar object for each mime attachement. The structure of the returned objects is as follows:

Táblázat 1. Returned Objects for imap_fetchstructure()

type	Primary body type
encoding	Body transfer encoding
ifsubtype	TRUE if there is a subtype string
subtype	MIME subtype
ifdescription	TRUE if there is a description string
description	Content description string
ifid	TRUE if there is an identification string
id	Identification string
lines	Number of lines
bytes	Number of bytes
ifdisposition	TRUE if there is a disposition string
disposition	Disposition string
ifdparameters	TRUE if the dparameters array exists
dparameters	An array of objects where each object has an "attribute" and a "value" property corresponding to the parameters on the Content-disposition MIMEheader.
ifparameters	TRUE if the parameters array exists
parameters	An array of objects where each object has an "attribute" and a "value" property.
parts	An array of objects identical in structure to the top-level object, each of which corresponds to a MIME body part.

Táblázat 2. Primary body type

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	video
7	other

Táblázat 3. Transfer encodings

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

See also: imap_fetchbody().

imap_get_quota (PHP 4 >= 4.0.5)

Retrieve the quota level settings, and usage statics per mailbox

array **imap_get_quota** (int imap_stream, string quota_root) \linebreak

Returns an array with integer values limit and usage for the given mailbox. The value of limit represents the total amount of space allowed for this mailbox. The usage value represents the mailboxes current level of capacity. Will return FALSE in the case of failure.

This function is currently only available to users of the c-client2000 library.

imap_stream should be the value returned from an imap_status() call. This stream is required to be opened as the mail admin user for the quota function to work. *quota_root* should normally be in the form of user.name where name is the mailbox you wish to retrieve information about.

Példa 1. imap_get_quota() example

```
$mbox = imap_open("{your.imap.host}","mailadmin","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$quota_value = imap_get_quota($mbox, "user.kalowsky");
if(is_array($quota_value)) {
```

```
    print "Usage level is: " . $quota_value['usage'];
    print "Limit level is: " . $quota_value['limit'];
}

imap_close($mbox);
```

See also imap_open(), imap_set_quota().

imap_getmailboxes (PHP 3>= 3.0.12, PHP 4)

Read the list of mailboxes, returning detailed information on each one

array **imap_getmailboxes** (int imap_stream, string ref, string pattern) \linebreak

Returns an array of objects containing mailbox information. Each object has the attributes *name*, specifying the full name of the mailbox; *delimiter*, which is the hierarchy delimiter for the part of the hierarchy this mailbox is in; and *attributes*. *Attributes* is a bitmask that can be tested against:

- LATT_NOINFERIORS - This mailbox has no "children" (there are no mailboxes below this one).
- LATT_NOSELECT - This is only a container, not a mailbox - you cannot open it.
- LATT_MARKED - This mailbox is marked. Only used by UW-IMAPD.
- LATT_UNMARKED - This mailbox is not marked. Only used by UW-IMAPD.

Mailbox names containing international Characters outside the printable ASCII range will be encoded and may be decoded by imap_utf7_decode().

ref should normally be just the server specification as described in imap_open(), and *pattern* specifies where in the mailbox hierarchy to start searching. If you want all mailboxes, pass '*' for *pattern*.

There are two special characters you can pass as part of the *pattern*: '*' and '%'. '*' means to return all mailboxes. If you pass *pattern* as '*', you will get a list of the entire mailbox hierarchy. '%' means to return the current level only. '%' as the *pattern* parameter will return only the top level mailboxes; '~/mail/%' on UW_IMAPD will return every mailbox in the ~/mail directory, but none in subfolders of that directory.

Példa 1. imap_getmailboxes() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$list = imap_getmailboxes($mbox,"{your.imap.host}","*");
if(is_array($list)) {
  reset($list);
  while (list($key, $val) = each($list))
  {
```

```
    print "($key) ";
    print imap_utf7_decode($val->name).",";
    print "'".$val->delimiter."',";
    print $val->attributes."<br>\n";
  }
} else
  print "imap_getmailboxes failed: ".imap_last_error()."\n";

imap_close($mbox);
```

See also imap_getsubscribed().

imap_getsubscribed (PHP 3>= 3.0.12, PHP 4)

List all the subscribed mailboxes

array **imap_getsubscribed** (int imap_stream, string ref, string pattern) \linebreak

This function is identical to imap_getmailboxes(), except that it only returns mailboxes that the user is subscribed to.

imap_header (PHP 3, PHP 4)

Read the header of the message

object **imap_header** (int imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]]) \linebreak

This is an alias to imap_headerinfo() and is identical to this in any way.

imap_headerinfo (PHP 3, PHP 4)

Read the header of the message

object **imap_headerinfo** (int imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]]) \linebreak

This function returns an object of various header elements.

remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references

message flags:
Recent - 'R' if recent and seen,

’N’ if recent and not seen,
’ ’ if not recent
Unseen - ’U’ if not seen AND not recent,
’ ’ if seen OR not seen and recent
Answered -’A’ if answered,
’ ’ if unanswered
Deleted - ’D’ if deleted,
’ ’ if not deleted
Draft - ’X’ if draft,
’ ’ if not draft
Flagged - ’F’ if flagged,
’ ’ if not flagged

NOTE that the Recent/Unseen behavior is a little odd. If you want to know if a message is Unseen, you must check for

Unseen == ’U’ || Recent == ’N’

toaddress (full to: line, up to 1024 characters)

to[] (returns an array of objects from the To line, containing):
personal
adl
mailbox
host

fromaddress (full from: line, up to 1024 characters)

from[] (returns an array of objects from the From line, containing):
personal
adl
mailbox
host

ccaddress (full cc: line, up to 1024 characters)
cc[] (returns an array of objects from the Cc line, containing):
personal
adl
mailbox
host

bccaddress (full bcc line, up to 1024 characters)
bcc[] (returns an array of objects from the Bcc line, containing):
personal
adl
mailbox
host

reply_toaddress (full reply_to: line, up to 1024 characters)
reply_to[] (returns an array of objects from the Reply_to line, containing):
personal

adl
mailbox
host

senderaddress (full sender: line, up to 1024 characters)
sender[] (returns an array of objects from the sender line, containing):
personal
adl
mailbox
host

return_path (full return-path: line, up to 1024 characters)
return_path[] (returns an array of objects from the return_path line,
containing):
personal
adl
mailbox
host

udate (mail message date in unix time)

fetchfrom (from line formatted to fit *fromlength*
characters)

fetchsubject (subject line formatted to fit *subjectlength* characters)

imap_headers (PHP 3, PHP 4)

Returns headers for all messages in a mailbox

array **imap_headers** (int imap_stream) \linebreak

Returns an array of string formatted with header info. One element per mail message.

imap_last_error (PHP 3>= 3.0.12, PHP 4)

This function returns the last IMAP error (if any) that occurred during this page request

string **imap_last_error** (void) \linebreak

This function returns the full text of the last IMAP error message that occurred on the current page. The error stack is untouched; calling **imap_last_error()** subsequently, with no intervening errors, will return the same error.

imap_listmailbox (PHP 3, PHP 4)

Read the list of mailboxes

array **imap_listmailbox** (int imap_stream, string ref, string pattern) \linebreak

Returns an array containing the names of the mailboxes. See imap_getmailboxes() for a description of *ref* and *pattern*.

Példa 1. imap_listmailbox() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$list = imap_listmailbox($mbox,"{your.imap.host}","*");
if(is_array($list)) {
  reset($list);
  while (list($key, $val) = each($list))
    print imap_utf7_decode($val)."<br>\n";
} else
  print "imap_listmailbox failed: ".imap_last_error()."\n";

imap_close($mbox);
```

imap_listsubscribed (PHP 3, PHP 4)

List all the subscribed mailboxes

array **imap_listsubscribed** (int imap_stream, string ref, string pattern) \linebreak

Returns an array of all the mailboxes that you have subscribed. This is almost identical to imap_listmailbox(), but will only return mailboxes the user you logged in as has subscribed.

imap_mail_compose (PHP 3>= 3.0.5, PHP 4)

Create a MIME message based on given envelope and body sections

string **imap_mail_compose** (array envelope, array body) \linebreak

Példa 1. imap_mail_compose() example

```
<?php

$envelope["from"]="musone@afterfive.com";
$envelope["to"]="musone@darkstar";
$envelope["cc"]="musone@edgeglobal.com";

$part1["type"]=TYPEMULTIPART;
$part1["subtype"]="mixed";

$filename="/tmp/imap.c.gz";
$fp=fopen($filename,"r");
$contents=fread($fp,filesize($filename));
fclose($fp);

$part2["type"]=TYPEAPPLICATION;
$part2["encoding"]=ENCBINARY;
$part2["subtype"]="octet-stream";
$part2["description"]=basename($filename);
$part2["contents.data"]=$contents;

$part3["type"]=TYPETEXT;
$part3["subtype"]="plain";
$part3["description"]="description3";
$part3["contents.data"]="contents.data3\n\n\n\t";

$body[1]=$part1;
$body[2]=$part2;
$body[3]=$part3;

echo nl2br(imap_mail_compose($envelope,$body));

?>
```

imap_mail_copy (PHP 3, PHP 4)

Copy specified messages to a mailbox

int **imap_mail_copy** (int imap_stream, string msglist, string mbox [, int flags]) \linebreak

Returns TRUE on success and FALSE on error.

Copies mail messages specified by *msglist* to specified mailbox. *msglist* is a range not just message numbers (as described in RFC2060 (http://www.faqs.org/rfcs/rfc2060.html)).

Flags is a bitmask of one or more of

- CP_UID - the sequence numbers contain UIDS
- CP_MOVE - Delete the messages from the current mailbox after copying

imap_mail_move (PHP 3, PHP 4)

Move specified messages to a mailbox

int **imap_mail_move** (int imap_stream, string msglist, string mbox [, int flags]) \linebreak

Moves mail messages specified by *msglist* to specified mailbox. *msglist* is a range not just message numbers (as described in RFC2060 (http://www.faqs.org/rfcs/rfc2060.html)).

Flags is a bitmask and may contain the single option

- CP_UID - the sequence numbers contain UIDS

Returns TRUE on success and FALSE on error.

imap_mail (PHP 3>= 3.0.14, PHP 4)

Send an email message

string **imap_mail** (string to, string subject, string message [, string additional_headers [, string cc [, string bcc [, string rpath]]]]) \linebreak

This function allows sending of emails with correct handling of Cc and Bcc receivers. The parameters to, cc and bcc are all strings and are all parsed as rfc822 address lists. The receivers specified in bcc will get the mail, but are excluded from the headers. Use the rpath parameter to specify return path. This is useful when using php as a mail client for multiple users.

imap_mailboxmsginfo (PHP 3>= 3.0.2, PHP 4)

Get information about the current mailbox

object **imap_mailboxmsginfo** (int imap_stream) \linebreak

Returns information about the current mailbox. Returns FALSE on failure.

The **imap_mailboxmsginfo()** function checks the current mailbox status on the server. It is similar to imap_status(), but will additionally sum up the size of all messages in the mailbox, which will take some additional time to execute. It returns the information in an object with following properties.

Táblázat 1. Mailbox properties

Date	date of last change
Driver	driver
Mailbox	name of the mailbox

Nmsgs	number of messages
Recent	number of recent messages
Unread	number of unread messages
Deleted	number of deleted messages
Size	mailbox size

Példa 1. imap_mailboxmsginfo() example

```
<?php

$mbox = imap_open("{your.imap.host}INBOX","username", "password")
      or die("can’t connect: ".imap_last_error());

$check = imap_mailboxmsginfo($mbox);

if($check) {
    print "Date: "    . $check->Date    ."<br>\n" ;
    print "Driver: "  . $check->Driver  ."<br>\n" ;
    print "Mailbox: " . $check->Mailbox ."<br>\n" ;
    print "Messages: ". $check->Nmsgs   ."<br>\n" ;
    print "Recent: "  . $check->Recent  ."<br>\n" ;
    print "Unread: "  . $check->Unread  ."<br>\n" ;
    print "Deleted: " . $check->Deleted ."<br>\n" ;
    print "Size: "    . $check->Size    ."<br>\n" ;
} else {
    print "imap_check() failed: ".imap_last_error(). "<br>\n";
}

imap_close($mbox);

?>
```

imap_mime_header_decode (PHP 3>= 3.0.17, PHP 4)

Decode MIME header elements

array **imap_mime_header_decode** (string text) \linebreak

imap_mime_header_decode() function decodes MIME message header extensions that are non ASCII text (see RFC2047 (http://www.faqs.org/rfcs/rfc2047.html)) The decoded elements are returned in an array of objects, where each object has two properties, "charset" & "text". If the element hasn’t been encoded, and in other words is in plain US-ASCII,the "charset" property of that element is set to "default".

Példa 1. imap_mime_header_decode() example

```
$text="=?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld@dkuug.dk>";

$elements=imap_mime_header_decode($text);
for($i=0;$i<count($elements);$i++) {
    echo "Charset: {$elements[$i]->charset}\n";
    echo "Text: {$elements[$i]->text}\n\n";
}
```

In the above example we would have two elements, whereas the first element had previously been encoded with ISO-8859-1, and the second element would be plain US-ASCII.

imap_msgno (PHP 3>= 3.0.3, PHP 4)

This function returns the message sequence number for the given UID

int **imap_msgno** (int imap_stream, int uid) \linebreak

This function returns the message sequence number for the given UID. It is the inverse of imap_uid().

imap_num_msg (PHP 3, PHP 4)

Gives the number of messages in the current mailbox

int **imap_num_msg** (int imap_stream) \linebreak

Return the number of messages in the current mailbox.

See also: imap_num_recent() and imap_status().

imap_num_recent (PHP 3, PHP 4)

Gives the number of recent messages in current mailbox

int **imap_num_recent** (int imap_stream) \linebreak

Returns the number of recent messages in the current mailbox.

See also: imap_num_msg() and imap_status().

imap_open (PHP 3, PHP 4)

Open an IMAP stream to a mailbox

int **imap_open** (string mailbox, string username, string password [, int flags]) \linebreak

Returns an IMAP stream on success and FALSE on error. This function can also be used to open streams to POP3 and NNTP servers, but some functions and features are only available on IMAP servers.

A mailbox name consists of a server part and a mailbox path on this server. The special name INBOX stands for the current users personal mailbox. The server part, which is enclosed in '{' and '}', consists of the servers name or ip address, an optional port (prefixed by ':'), and an optional protocol specification (prefixed by '/'). The server part is mandatory in all mailbox parameters. Mailbox names that contain international characters besides those in the printable ASCII space have to be encoded with imap_utf7_encode().

The options are a bit mask with one or more of the following:

- OP_READONLY - Open mailbox read-only
- OP_ANONYMOUS - Dont use or update a .newsrc for news (NNTP only)
- OP_HALFOPEN - For IMAP and NNTP names, open a connection but dont open a mailbox
- CL_EXPUNGE - Expunge mailbox automatically upon mailbox close

To connect to an IMAP server running on port 143 on the local machine, do the following:

```
$mbox = imap_open ("{localhost:143}INBOX", "user_id", "password");
```

To connect to a POP3 server on port 110 on the local server, use:

```
$mbox = imap_open ("{localhost:110/pop3}INBOX", "user_id", "password");
```

To connect to an SSL IMAP or POP3 server, add /ssl after the protocol specification:

```
$mbox = imap_open ("{localhost:993/imap/ssl}INBOX", "user_id", "password");
```

To connect to an SSL IMAP or POP3 server with a self-signed certificate, add /ssl/novalidate-cert after the protocol specification:

```
$mbox = imap_open ("{localhost:995/pop3/ssl/novalidate-cert}", "user_id", "password");
```

To connect to an NNTP server on port 119 on the local server, use:

```
$nntp = imap_open ("{localhost:119/nntp}comp.test", "", "");
```

To connect to a remote server replace "localhost" with the name or the IP address of the server you want to connect to.

Példa 1. imap_open() example

```
$mbox = imap_open ("{your.imap.host:143}", "username", "password");

echo "<p><h1>Mailboxes</h1>\n";
$folders = imap_listmailbox ($mbox, "{your.imap.host:143}", "*");

if ($folders == false) {
    echo "Call failed<br>\n";
} else {
    while (list ($key, $val) = each ($folders)) {
        echo $val."<br>\n";
    }
}

echo "<p><h1>Headers in INBOX</h1>\n";
$headers = imap_headers ($mbox);

if ($headers == false) {
    echo "Call failed<br>\n";
} else {
    while (list ($key,$val) = each ($headers)) {
        echo $val."<br>\n";
    }
}

imap_close($mbox);
```

imap_ping (PHP 3, PHP 4)

Check if the IMAP stream is still active

int **imap_ping** (int imap_stream) \linebreak

Returns TRUE if the stream is still alive, FALSE otherwise.

imap_ping() function pings the stream to see it is still active. It may discover new mail; this is the preferred method for a periodic "new mail check" as well as a "keep alive" for servers which have inactivity timeout. (As PHP scripts do not tend to run that long, i can hardly imagine that this function will be usefull to anyone.)

imap_popen (PHP 3>= 3.0.12, PHP 4)

Open a persistant IMAP stream to a mailbox

int **imap_popen** (string mailbox, string user, string password [, int options]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

imap_qprint (PHP 3, PHP 4)

Convert a quoted-printable string to an 8 bit string

string **imap_qprint** (string string) \linebreak

Convert a quoted-printable string to an 8 bit string (according to RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), section 6.7).

Returns an 8 bit (binary) string.

See also imap_8bit().

imap_renamemailbox (PHP 3, PHP 4)

Rename an old mailbox to new mailbox

int **imap_renamemailbox** (int imap_stream, string old_mbox, string new_mbox) \linebreak

This function renames on old mailbox to new mailbox (see imap_open() for the format of *mbox* names).

Returns TRUE on success and FALSE on error.

See also imap_createmailbox(), imap_deletemailbox(), and imap_open() for the format of *mbox*.

imap_reopen (PHP 3, PHP 4)

Reopen IMAP stream to new mailbox

int **imap_reopen** (int imap_stream, string mailbox [, string flags]) \linebreak

This function reopens the specified stream to a new mailbox on an IMAP or NNTP server.

The options are a bit mask with one or more of the following:

- OP_READONLY - Open mailbox read-only
- OP_ANONYMOUS - Dont use or update a `.newsrc` for news (NNTP only)
- OP_HALFOPEN - For IMAP and NNTP names, open a connection but dont open a mailbox.
- CL_EXPUNGE - Expunge mailbox automatically upon mailbox close (see also imap_delete() and imap_expunge())

Returns `TRUE` on success and `FALSE` on error.

imap_rfc822_parse_adrlist (PHP 3>= 3.0.2, PHP 4)

Parses an address string

array **imap_rfc822_parse_adrlist** (string address, string default_host) \linebreak

This function parses the address string as defined in RFC2822 (http://www.faqs.org/rfcs/rfc2822.html) and for each address, returns an array of objects. The objects properties are:

- mailbox - the mailbox name (username)
- host - the host name
- personal - the personal name
- adl - at domain source route

Példa 1. imap_rfc822_parse_adrlist() example

```
$address_string = "Hartmut Holzgraefe <hartmut@cvs.php.net>, postmaster@somedomain.net, r
$address_array  = imap_rfc822_parse_adrlist($address_string,"somedomain.net");
if(! is_array($address_array)) die("somethings wrong\n");

reset($address_array);
while(list($key,$val)=each($address_array)){
  print "mailbox : ".$val->mailbox."<br>\n";
  print "host    : ".$val->host."<br>\n";
  print "personal: ".$val->personal."<br>\n";
  print "adl     : ".$val->adl."<p>\n";
}
```

imap_rfc822_parse_headers (PHP 4)

Parse mail headers from a string

object **imap_rfc822_parse_headers** (string headers [, string defaulthost]) \linebreak

This function returns an object of various header elements, similar to imap_header(), except without the flags and other elements that come from the IMAP server.

imap_rfc822_write_address (PHP 3>= 3.0.2, PHP 4)

Returns a properly formatted email address given the mailbox, host, and personal info.

string **imap_rfc822_write_address** (string mailbox, string host, string personal) \linebreak

Returns a properly formatted email address as defined in RFC2822 (http://www.faqs.org/rfcs/rfc2822.html) given the mailbox, host, and personal info.

Példa 1. imap_rfc822_write_address() example

```
print imap_rfc822_write_address("hartmut","cvs.php.net","Hartmut Holzgraefe")."\n";
```

imap_scanmailbox (PHP 3, PHP 4)

Read the list of mailboxes, takes a string to search for in the text of the mailbox

array **imap_scanmailbox** (int imap_stream, string ref, string pattern, string content) \linebreak

Returns an array containing the names of the mailboxes that have *content* in the text of the mailbox. This function is similar to imap_listmailbox(), but it will additionally check for the presence of the string *content* inside the mailbox data. See imap_getmailboxes() for a description of *ref* and *pattern*.

imap_search (PHP 3>= 3.0.12, PHP 4)

This function returns an array of messages matching the given search criteria

array **imap_search** (int imap_stream, string criteria, int flags) \linebreak

This function performs a search on the mailbox currently opened in the given imap stream. `criteria` is a string, delimited by spaces, in which the following keywords are allowed. Any multi-word arguments (eg. FROM "joey smith") must be quoted.

- ALL - return all messages matching the rest of the criteria
- ANSWERED - match messages with the \\ANSWERED flag set
- BCC "string" - match messages with "string" in the Bcc: field
- BEFORE "date" - match messages with Date: before "date"
- BODY "string" - match messages with "string" in the body of the message
- CC "string" - match messages with "string" in the Cc: field
- DELETED - match deleted messages
- FLAGGED - match messages with the \\FLAGGED (sometimes referred to as Important or Urgent) flag set
- FROM "string" - match messages with "string" in the From: field
- KEYWORD "string" - match messages with "string" as a keyword
- NEW - match new messages
- OLD - match old messages
- ON "date" - match messages with Date: matching "date"
- RECENT - match messages with the \\RECENT flag set
- SEEN - match messages that have been read (the \\SEEN flag is set)
- SINCE "date" - match messages with Date: after "date"
- SUBJECT "string" - match messages with "string" in the Subject:
- TEXT "string" - match messages with text "string"
- TO "string" - match messages with "string" in the To:
- UNANSWERED - match messages that have not been answered
- UNDELETED - match messages that are not deleted
- UNFLAGGED - match messages that are not flagged
- UNKEYWORD "string" - match messages that do not have the keyword "string"
- UNSEEN - match messages which have not been read yet

For example, to match all unanswered messages sent by Mom, you'd use: "UNANSWERED FROM mom". Searches appear to be case insensitive. This list of criteria is from a reading of the UW c-client source code and may be uncomplete or inaccurate (see also RFC2060, section 6.4.4).

Valid values for flags are SE_UID, which causes the returned array to contain UIDs instead of messages sequence numbers.

imap_set_quota (PHP 4 >= 4.0.5)

Sets a quota for a given mailbox

int **imap_set_quota** (int imap_stream, string quota_root, int quota_limit) \linebreak

Sets an upper limit quota on a per mailbox basis. This function requires the *imap_stream* to have been opened as the mail administrator account. It will not work if opened as any other user.

This function is currently only available to users of the c-client2000 library.

imap_stream is the stream pointer returned from a imap_open() call. This stream must be opened as the mail administrator, other wise this function will fail. *quota_root* is the mailbox to have a quota set. This should follow the IMAP standard format for a mailbox, 'user.name'. *quota_limit* is the maximum size (in KB) for the *quota_root*.

Returns TRUE on success and FALSE on error.

Példa 1. imap_set_quota() example

```
$mbox = imap_open ("{your.imap.host:143}", "mailadmin", "password");

if(!imap_set_quota($mbox, "user.kalowsky", 3000)) {
    print "Error in setting quota\n";
    return;
}

imap_close($mbox);
```

See also imap_open(), **imap_set_quota()**.

imap_setacl (PHP 4 >= 4.1.0)

Sets the ACL for a giving mailbox

int **imap_setacl** (int stream_id, string mailbox, string id, string rights) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

imap_setflag_full (PHP 3>= 3.0.3, PHP 4)

Sets flags on messages

string **imap_setflag_full** (int stream, string sequence, string flag, string options) \linebreak

This function causes a store to add the specified flag to the flags set for the messages in the specified sequence.

The flags which you can set are "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", and "\\Draft" (as defined by RFC2060).

The options are a bit mask with one or more of the following:

ST_UID The sequence argument contains UIDs instead of sequence numbers

Példa 1. imap_setflag_full() example

```
$mbox = imap_open("{your.imap.host:143}","username","password")
     or die("can't connect: ".imap_last_error());

$status = imap_setflag_full($mbox,"2,5","\\Seen \\Flagged");

print gettype($status)."\n";
print $status."\n";

imap_close($mbox);
```

imap_sort (PHP 3>= 3.0.3, PHP 4)

Sort an array of message headers

array **imap_sort** (int stream, int criteria, int reverse, int options) \linebreak

Returns an array of message numbers sorted by the given parameters.

Reverse is 1 for reverse-sorting.

Criteria can be one (and only one) of the following:

SORTDATE message Date
SORTARRIVAL arrival date
SORTFROM mailbox in first From address
SORTSUBJECT message Subject
SORTTO mailbox in first To address

SORTCC mailbox in first cc address
SORTSIZE size of message in octets

The flags are a bitmask of one or more of the following:

SE_UID Return UIDs instead of sequence numbers
SE_NOPREFETCH Don’t prefetch searched messages.

imap_status (PHP 3>= 3.0.4, PHP 4)

This function returns status information on a mailbox other than the current one

object **imap_status** (int imap_stream, string mailbox, int options) \linebreak

This function returns an object containing status information. Valid flags are:

- SA_MESSAGES - set status->messages to the number of messages in the mailbox
- SA_RECENT - set status->recent to the number of recent messages in the mailbox
- SA_UNSEEN - set status->unseen to the number of unseen (new) messages in the mailbox
- SA_UIDNEXT - set status->uidnext to the next uid to be used in the mailbox
- SA_UIDVALIDITY - set status->uidvalidity to a constant that changes when uids for the mailbox may no longer be valid
- SA_ALL - set all of the above

status->flags is also set, which contains a bitmask which can be checked against any of the above constants.

Példa 1. imap_status() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      or die("can’t connect: ".imap_last_error());

$status = imap_status($mbox,"{your.imap.host}INBOX",SA_ALL);
if($status) {
  print("Messages:   ". $status->messages   ."<br>\n";
  print("Recent:     ". $status->recent     ."<br>\n";
  print("Unseen:     ". $status->unseen     ."<br>\n";
  print("UIDnext:    ". $status->uidnext    ."<br>\n";
  print("UIDvalidity:". $status->uidvalidity)."<br>\n";
} else
  print "imap_status failed: ".imap_last_error()."\n";

imap_close($mbox);
```

imap_subscribe (PHP 3, PHP 4)

Subscribe to a mailbox

int **imap_subscribe** (int imap_stream, string mbox) \linebreak

Subscribe to a new mailbox.

Returns TRUE on success and FALSE on error.

imap_thread (PHP 4 >= 4.1.0)

Return threaded by REFERENCES tree

int **imap_thread** (int stream_id [, int flags]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

imap_uid (PHP 3>= 3.0.3, PHP 4)

This function returns the UID for the given message sequence number

int **imap_uid** (int imap_stream, int msgno) \linebreak

This function returns the UID for the given message sequence number. An UID is an unique identifier that will not change over time while a message sequence number may change whenever the content of the mailbox changes. This function is the inverse of imap_msgno().

Megjegyzés: This is not supported by POP3 mailboxes.

imap_undelete (PHP 3, PHP 4)

Unmark the message which is marked deleted

int **imap_undelete** (int imap_stream, int msg_number) \linebreak

This function removes the deletion flag for a specified message, which is set by imap_delete() or imap_mail_move().

Returns TRUE on success and FALSE on error.

imap_unsubscribe (PHP 3, PHP 4)

Unsubscribe from a mailbox

int **imap_unsubscribe** (int imap_stream, string mbox) \linebreak

Unsubscribe from a specified mailbox.

Returns TRUE on success and FALSE on error.

imap_utf7_decode (PHP 3>= 3.0.15, PHP 4)

Decodes a modified UTF-7 encoded string.

string **imap_utf7_decode** (string text) \linebreak

Decodes modified UTF-7 *text* into 8bit data.

Returns the decoded 8bit data, or FALSE if the input string was not valid modified UTF-7. This function is needed to decode mailbox names that contain international characters outside of the printable ASCII range. The modified UTF-7 encoding is defined in RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), section 5.1.3 (original UTF-7 was defned in RFC1642 (http://www.faqs.org/rfcs/rfc1642.html)).

imap_utf7_encode (PHP 3>= 3.0.15, PHP 4)

Converts 8bit data to modified UTF-7 text.

string **imap_utf7_encode** (string data) \linebreak

Converts 8bit *data* to modified UTF-7 text. This is needed to encode mailbox names that contain international characters outside of the printable ASCII range. The modified UTF-7 encoding is defined in RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), section 5.1.3 (original UTF-7 was defned in RFC1642 (http://www.faqs.org/rfcs/rfc1642.html)).

Returns the modified UTF-7 text.

imap_utf8 (PHP 3>= 3.0.13, PHP 4)

Converts text to UTF8

string **imap_utf8** (string text) \linebreak

Converts the given *text* to UTF8 (as defined in RFC2044 (http://www.faqs.org/rfcs/rfc2044.html)).

XLIV. Informix functions

Bevezetés

The Informix driver for Informix (IDS) 7.x, SE 7.x, Universal Server (IUS) 9.x and IDS 2000 is implemented in "ifx.ec" and "php3_ifx.h" in the informix extension directory. IDS 7.x support is fairly complete, with full support for BYTE and TEXT columns. IUS 9.x support is partly finished: the new data types are there, but SLOB and CLOB support is still under construction.

Követelmények

Configuration notes: You need a version of ESQL/C to compile the PHP Informix driver. ESQL/C versions from 7.2x on should be OK. ESQL/C is now part of the Informix Client SDK.

Make sure that the "INFORMIXDIR" variable has been set, and that $INFORMIXDIR/bin is in your PATH before you run the "configure" script.

Telepítés

Megjegyzés: The configure script will autodetect the libraries and include directories, if you run configure `--with_informix=yes`. You can override this detection by specifying "IFX_LIBDIR", "IFX_LIBS" and "IFX_INCDIR" in the environment. The configure script will also try to detect your Informix server version. It will set the "HAVE_IFX_IUS" conditional compilation variable if your Informix version >= 9.00.

Futásidejű beállítások

Megjegyzés: Make sure that the Informix environment variables INFORMIXDIR and INFORMIXSERVER are available to the PHP ifx driver, and that the INFORMIX bin directory is in the PATH. Check this by running a script that contains a call to phpinfo() before you start testing. The phpinfo() output should list these environment variables. This is TRUE for both CGI php and Apache mod_php. You may have to set these environment variables in your Apache startup script.

The Informix shared libraries should also be available to the loader (check LD_LINBRARY_PATH or ld.so.conf/ldconfig).

Some notes on the use of BLOBs (TEXT and BYTE columns): BLOBs are normally addressed by BLOB identifiers. Select queries return a "blob id" for every BYTE and TEXT column. You can get at the contents with "string_var = ifx_get_blob($blob_id);" if you choose to get the BLOBs in memory (with : "ifx_blobinfile(0);"). If you prefer to receive the content of BLOB

columns in a file, use "ifx_blobinfile(1);", and "ifx_get_blob($blob_id);" will get you the filename. Use normal file I/O to get at the blob contents.

For insert/update queries you must create these "blob id's" yourself with "ifx_create_blob();". You then plug the blob id's into an array, and replace the blob columns with a question mark (?) in the query string. For updates/inserts, you are responsible for setting the blob contents with ifx_update_blob().

The behaviour of BLOB columns can be altered by configuration variables that also can be set at runtime :

configuration variable : ifx.textasvarchar

configuration variable : ifx.byteasvarchar

runtime functions :

ifx_textasvarchar(0) : use blob id's for select queries with TEXT columns

ifx_byteasvarchar(0) : use blob id's for select queries with BYTE columns

ifx_textasvarchar(1) : return TEXT columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

ifx_byteasvarchar(1) : return BYTE columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

configuration variable : ifx.blobinfile

runtime function :

ifx_blobinfile_mode(0) : return BYTE columns in memory, the blob id lets you get at the contents.

ifx_blobinfile_mode(1) : return BYTE columns in a file, the blob id lets you get at the file name.

If you set ifx_text/byteasvarchar to 1, you can use TEXT and BYTE columns in select queries just like normal (but rather long) VARCHAR fields. Since all strings are "counted" in PHP, this remains "binary safe". It is up to you to handle this correctly. The returned data can contain anything, you are responsible for the contents.

If you set ifx_blobinfile to 1, use the file name returned by ifx_get_blob(..) to get at the blob contents. Note that in this case YOU ARE RESPONSIBLE FOR DELETING THE TEMPORARY FILES CREATED BY INFORMIX when fetching the row. Every new row fetched will create new temporary files for every BYTE column.

The location of the temporary files can be influenced by the environment variable "blobdir", default is "." (the current directory). Something like : putenv(blobdir=tmpblob"); will ease the cleaning up of temp files accidentally left behind (their names all start with "blb").

Automatically trimming "char" (SQLCHAR and SQLNCHAR) data: This can be set with the configuration variable

ifx.charasvarchar : if set to 1 trailing spaces will be automatically trimmed, to save you some "chopping".

`NULL` values: The configuration variable ifx.nullformat (and the runtime function ifx_nullformat()) when set to `TRUE` will return `NULL` columns as the string "`NULL`", when set to `FALSE` they return

the empty string. This allows you to discriminate between `NULL` columns and empty columns.

Erőforrás típusok

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

ifx_affected_rows (PHP 3>= 3.0.3, PHP 4)

Get number of rows affected by a query

int **ifx_affected_rows** (int result_id) \linebreak

result_id is a valid result id returned by ifx_query() or ifx_prepare().

Returns the number of rows affected by a query associated with *result_id*.

For inserts, updates and deletes the number is the real number (sqlerrd[2]) of affected rows. For selects it is an estimate (sqlerrd[0]). Don't rely on it. The database server can never return the actual number of rows that will be returned by a SELECT because it has not even begun fetching them at this stage (just after the "PREPARE" when the optimizer has determined the query plan).

Useful after ifx_prepare() to limit queries to reasonable result sets.

See also: ifx_num_rows()

Példa 1. Informix affected rows

```
$rid = ifx_prepare ("select * from emp
                     where name like " . $name, $connid);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
```

ifx_blobinfile_mode (PHP 3>= 3.0.4, PHP 4)

Set the default blob mode for all select queries

void **ifx_blobinfile_mode** (int mode) \linebreak

Set the default blob mode for all select queries. Mode "0" means save Byte-Blobs in memory, and mode "1" means save Byte-Blobs in a file.

ifx_byteasvarchar (PHP 3>= 3.0.4, PHP 4)

Set the default byte mode

void **ifx_byteasvarchar** (int mode) \linebreak

Sets the default byte mode for all select-queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_close (PHP 3>= 3.0.3, PHP 4)

Close Informix connection

int **ifx_close** ([int link_identifier]) \linebreak

Returns: always TRUE.

ifx_close() closes the link to an Informix database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

ifx_close() will not close persistent links generated by ifx_pconnect().

See also: ifx_connect(), and ifx_pconnect().

Példa 1. Closing a Informix connection

```
$conn_id = ifx_connect ("mydb@ol_srv", "itsme", "mypassword");
... some queries and stuff ...
ifx_close($conn_id);
```

ifx_connect (PHP 3>= 3.0.3, PHP 4)

Open Informix server connection

int **ifx_connect** ([string database [, string userid [, string password]]]) \linebreak

Returns a connection identifier on success, or FALSE on error.

ifx_connect() establishes a connection to an Informix server. All of the arguments are optional, and if they're missing, defaults are taken from values supplied in configuration file (ifx.default_host for the host (Informix libraries will use INFORMIXSERVER environment value if not defined), ifx.default_user for user, ifx.default_password for the password (none if not defined).

In case a second call is made to **ifx_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling ifx_close().

See also ifx_pconnect(), and ifx_close().

Példa 1. Connect to a Informix database

```
$conn_id = ifx_connect ("mydb@ol_srv1", "imyself", "mypassword");
```

ifx_copy_blob (PHP 3>= 3.0.4, PHP 4)

Duplicates the given blob object

int **ifx_copy_blob** (int bid) \linebreak

Duplicates the given blob object. *`bid`* is the ID of the blob object.

Returns FALSE on error otherwise the new blob object-id.

ifx_create_blob (PHP 3>= 3.0.4, PHP 4)

Creates an blob object

int **ifx_create_blob** (int type, int mode, string param) \linebreak

Creates an blob object.

type: 1 = TEXT, 0 = BYTE

mode: 0 = blob-object holds the content in memory, 1 = blob-object holds the content in file.

param: if mode = 0: pointer to the content, if mode = 1: pointer to the filestring.

Return FALSE on error, otherwise the new blob object-id.

ifx_create_char (PHP 3>= 3.0.6, PHP 4)

Creates an char object

int **ifx_create_char** (string param) \linebreak

Creates an char object. *`param`* should be the char content.

ifx_do (PHP 3>= 3.0.4, PHP 4)

Execute a previously prepared SQL-statement

int **ifx_do** (int result_id) \linebreak

Returns TRUE on success, FALSE on error.

Executes a previously prepared query or opens a cursor for it.

Does NOT free *`result_id`* on error.

Also sets the real number of ifx_affected_rows() for non-select statements for retrieval by ifx_affected_rows()

See also: ifx_prepare(). There is a example.

ifx_error (PHP 3>= 3.0.3, PHP 4)

Returns error code of last Informix call

string **ifx_error** (void) \linebreak

The Informix error codes (SQLSTATE & SQLCODE) formatted as follows :

x [SQLSTATE = aa bbb SQLCODE=cccc]

where x = space : no error

E : error

N : no more data

W : warning

? : undefined

If the "x" character is anything other than space, SQLSTATE and SQLCODE describe the error in more detail.

See the Informix manual for the description of SQLSTATE and SQLCODE

Returns in a string one character describing the general results of a statement and both SQLSTATE and SQLCODE associated with the most recent SQL statement executed. The format of the string is "(char) [SQLSTATE=(two digits) (three digits) SQLCODE=(one digit)]". The first character can be ’ ’ (space) (success), ’W’ (the statement caused some warning), ’E’ (an error happened when executing the statement) or ’N’ (the statement didn’t return any data).

See also: ifx_errormsg()

ifx_errormsg (PHP 3>= 3.0.4, PHP 4)

Returns error message of last Informix call

string **ifx_errormsg** ([int errorcode]) \linebreak

Returns the Informix error message associated with the most recent Informix error, or, when the optional "*errorcode*" param is present, the error message corresponding to "*errorcode*".

See also: ifx_error()

```
printf("%s\n&lt;br>", ifx_errormsg(-201));
```

ifx_fetch_row (PHP 3>= 3.0.3, PHP 4)

Get row as enumerated array

array **ifx_fetch_row** (int result_id [, mixed position]) \linebreak

Returns an associative array that corresponds to the fetched row, or FALSE if there are no more rows.

Blob columns are returned as integer blob id values for use in ifx_get_blob() unless you have used ifx_textasvarchar(1) or ifx_byteasvarchar(1), in which case blobs are returned as string values. Returns FALSE on error

result_id is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

position is an optional parameter for a "fetch" operation on "scroll" cursors: "NEXT", "PREVIOUS", "CURRENT", "FIRST", "LAST" or a number. If you specify a number, an "absolute" row fetch is executed. This parameter is optional, and only valid for SCROLL cursors.

ifx_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0, with the column name as key.

Subsequent calls to **ifx_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

Példa 1. Informix fetch rows

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                      $connid, IFX_SCROLL);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do ($rid)) {
   ... error ...
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for(reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("\n<br>");
    $row = ifx_fetch_row ($rid, "NEXT");
}
ifx_free_result ($rid);
```

ifx_fieldproperties (PHP 3>= 3.0.3, PHP 4)

List of SQL fieldproperties

array **ifx_fieldproperties** (int result_id) \linebreak

Returns an associative array with fieldnames as key and the SQL fieldproperties as data for a query with *result_id*. Returns FALSE on error.

Returns the Informix SQL fieldproperties of every field in the query as an associative array. Properties are encoded as: "SQLTYPE;length;precision;scale;ISNULLABLE" where SQLTYPE = the Informix type like "SQLVCHAR" etc. and ISNULLABLE = "Y" or "N".

Példa 1. Informix SQL fieldproperties

```
$properties = ifx_fieldproperties ($resultid);
if (! isset($properties)) {
  ... error ...
}
for ($i = 0; $i < count($properties); $i++) {
    $fname = key ($properties);
    printf ("%s:\t type =  %s\n", $fname, $properties[$fname]);
    next ($properties);
}
```

ifx_fieldtypes (PHP 3>= 3.0.3, PHP 4)

List of Informix SQL fields

array **ifx_fieldtypes** (int result_id) \linebreak

Returns an associative array with fieldnames as key and the SQL fieldtypes as data for query with *result_id*. Returns FALSE on error.

Példa 1. Fieldnames and SQL fieldtypes

```
$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
  ... error ...
}
for ($i = 0; $i < count($types); $i++) {
    $fname = key($types);
    printf("%s :\t type =  %s\n", $fname, $types[$fname]);
    next($types);
}
```

ifx_free_blob (PHP 3>= 3.0.4, PHP 4)

Deletes the blob object

int **ifx_free_blob** (int bid) \linebreak

Deletes the blobobject for the given blob object-id *`bid`*. Returns `FALSE` on error otherwise `TRUE`.

ifx_free_char (PHP 3>= 3.0.6, PHP 4)

Deletes the char object

int **ifx_free_char** (int bid) \linebreak

Deletes the charobject for the given char object-id *`bid`*. Returns `FALSE` on error otherwise `TRUE`.

ifx_free_result (PHP 3>= 3.0.3, PHP 4)

Releases resources for the query

int **ifx_free_result** (int result_id) \linebreak

Releases resources for the query associated with *`result_id`*. Returns `FALSE` on error.

ifx_get_blob (PHP 3>= 3.0.4, PHP 4)

Return the content of a blob object

int **ifx_get_blob** (int bid) \linebreak

Returns the content of the blob object for the given blob object-id *`bid`*.

ifx_get_char (PHP 3>= 3.0.6, PHP 4)

Return the content of the char object

int **ifx_get_char** (int bid) \linebreak

Returns the content of the char object for the given char object-id *`bid`*.

ifx_getsqlca (PHP 3>= 3.0.8, PHP 4)

Get the contents of sqlca.sqlerrd[0..5] after a query

array **ifx_getsqlca** (int result_id) \linebreak

result_id is a valid result id returned by ifx_query() or ifx_prepare().

Returns a pseudo-row (associative array) with sqlca.sqlerrd[0] ... sqlca.sqlerrd[5] after the query associated with *result_id*.

For inserts, updates and deletes the values returned are those as set by the server after executing the query. This gives access to the number of affected rows and the serial insert value. For SELECTs the values are those saved after the PREPARE statement. This gives access to the *estimated* number of affected rows. The use of this function saves the overhead of executing a "select dbinfo('sqlca.sqlerrdx')" query, as it retrieves the values that were saved by the ifx driver at the appropriate moment.

Példa 1. Retrieve Informix sqlca.sqlerrd[x] values

```
/* assume the first column of 'sometable' is a serial */
$qid = ifx_query("insert into sometable
                  values (0, '2nd column', 'another column') ", $connid);
if (! $qid) {
    ... error ...
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "The serial value of the inserted row is : " . $serial_value<br>\n";
```

ifx_htmltbl_result (PHP 3>= 3.0.3, PHP 4)

Formats all rows of a query into a HTML table

int **ifx_htmltbl_result** (int result_id [, string html_table_options]) \linebreak

Returns the number of rows fetched or FALSE on error.

Formats all rows of the *result_id* query into a html table. The optional second argument is a string of <table> tag options

Példa 1. Informix results as HTML table

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                     $connid, IFX_SCROLL);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do($rid) {
  ... error ...
```

```
}

ifx_htmltbl_result ($rid, "border=\"2\"");

ifx_free_result($rid);
```

ifx_nullformat (PHP 3>= 3.0.4, PHP 4)

Sets the default return value on a fetch row

void **ifx_nullformat** (int mode) \linebreak

Sets the default return value of a NULL-value on a fetch row. Mode "0" returns "", and mode "1" returns "NULL".

ifx_num_fields (PHP 3>= 3.0.3, PHP 4)

Returns the number of columns in the query

int **ifx_num_fields** (int result_id) \linebreak

Returns the number of columns in query for *result_id* or FALSE on error

After preparing or executing a query, this call gives you the number of columns in the query.

ifx_num_rows (PHP 3>= 3.0.3, PHP 4)

Count the rows already fetched from a query

int **ifx_num_rows** (int result_id) \linebreak

Gives the number of rows fetched so far for a query with *result_id* after a ifx_query() or ifx_do() query.

ifx_pconnect (PHP 3>= 3.0.3, PHP 4)

Open persistent Informix connection

int **ifx_pconnect** ([string database [, string userid [, string password]]]) \linebreak

Returns: A positive Informix persistent link identifier on success, or FALSE on error

ifx_pconnect() acts very much like ifx_connect() with two major differences.

This function behaves exactly like ifx_connect() when PHP is not running as an Apache module. First, when connecting, the function would first try to find a (persistent) link that's already open with

the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (ifx_close() will not close links established by **ifx_pconnect()**).

This type of links is therefore called 'persistent'.

See also: ifx_connect().

ifx_prepare (PHP 3>= 3.0.4, PHP 4)

Prepare an SQL-statement for execution

int **ifx_prepare** (string query, int conn_id [, int cursor_def, mixed blobidarray]) \linebreak

Returns a integer *`result_id`* for use by ifx_do(). Sets *`affected_rows`* for retrieval by the ifx_affected_rows() function.

Prepares *`query`* on connection *`conn_id`*. For "select-type" queries a cursor is declared and opened. The optional *`cursor_type`* parameter allows you to make this a "scroll" and/or "hold" cursor. It's a bitmask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together.

For either query type the estimated number of affected rows is saved for retrieval by ifx_affected_rows().

If you have BLOB (BYTE or TEXT) columns in the query, you can add a *`blobidarray`* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: ifx_do().

ifx_query (PHP 3>= 3.0.3, PHP 4)

Send Informix query

int **ifx_query** (string query, int link_identifier [, int cursor_type [, mixed blobidarray]]) \linebreak

Returns: A positive Informix result identifier on success, or FALSE on error.

A "result_id" resource used by other functions to retrieve the query results. Sets "affected_rows" for retrieval by the ifx_affected_rows() function.

ifx_query() sends a query to the currently active database on the server that's associated with the specified link identifier.

Executes *query* on connection *conn_id*. For "select-type" queries a cursor is declared and opened. The optional *cursor_type* parameter allows you to make this a "scroll" and/or "hold" cursor. It's a bitmask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together. Non-select queries are "execute immediate". IFX_SCROLL and IFX_HOLD are symbolic constants and as such shouldn't be between quotes. I you omit this parameter the cursor is a normal sequential cursor.

For either query type the number of (estimated or real) affected rows is saved for retrieval by ifx_affected_rows().

If you have BLOB (BYTE or TEXT) columns in an update query, you can add a *blobidarray* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: ifx_connect().

Példa 1. Show all rows of the "orders" table as a html table

```
ifx_textasvarchar(1);      // use "text mode" for blobs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
    printf("Can't select orders : %s\n<br>%s<br>\n", ifx_error());
    ifx_errormsg();
    die;
}
ifx_htmltbl_result($res_id, "border=\"1\"");
ifx_free_result($res_id);
```

Példa 2. Insert some values into the "catalog" table

```
                  // create blob id's for a byte and text column
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");
                     // store blob id's in a blobid array
$blobidarray[] = $textid;
$blobidarray[] = $byteid;
                      // launch query
$query = "insert into catalog (stock_num, manu_code, " .
         "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
if (! $res_id) {
  ... error ...
}
                    // free result id
```

```
ifx_free_result($res_id);
```

ifx_textasvarchar (PHP 3>= 3.0.4, PHP 4)

Set the default text mode

void **ifx_textasvarchar** (int mode) \linebreak

Sets the default text mode for all select-queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_update_blob (PHP 3>= 3.0.4, PHP 4)

Updates the content of the blob object

ifx_update_blob (int bid, string content) \linebreak

Updates the content of the blob object for the given blob object *`bid`*. *`content`* is a string with new data. Returns FALSE on error otherwise TRUE.

ifx_update_char (PHP 3>= 3.0.6, PHP 4)

Updates the content of the char object

int **ifx_update_char** (int bid, string content) \linebreak

Updates the content of the char object for the given char object *`bid`*. *`content`* is a string with new data. Returns FALSE on error otherwise TRUE.

ifxus_close_slob (PHP 3>= 3.0.4, PHP 4)

Deletes the slob object

int **ifxus_close_slob** (int bid) \linebreak

Deletes the slob object on the given slob object-id *`bid`*. Return FALSE on error otherwise TRUE.

ifxus_create_slob (PHP 3>= 3.0.4, PHP 4)

Creates an slob object and opens it

int **ifxus_create_slob** (int mode) \linebreak

Creates an slob object and opens it. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> or-mask. You can also use constants named IFX_LO_RDONLY, IFX_LO_WRONLY etc. Return FALSE on error otherwise the new slob object-id.

ifxus_free_slob (PHP 3>= 3.0.4, PHP 4)

Deletes the slob object

int **ifxus_free_slob** (int bid) \linebreak

Deletes the slob object. *bid* is the Id of the slob object. Returns FALSE on error otherwise TRUE.

ifxus_open_slob (PHP 3>= 3.0.4, PHP 4)

Opens an slob object

int **ifxus_open_slob** (long bid, int mode) \linebreak

Opens an slob object. *bid* should be an existing slob id. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> or-mask. Returns FALSE on error otherwise the new slob object-id.

ifxus_read_slob (PHP 3>= 3.0.4, PHP 4)

Reads nbytes of the slob object

int **ifxus_read_slob** (long bid, long nbytes) \linebreak

Reads nbytes of the slob object. *bid* is a existing slob id and *nbytes* is the number of bytes read. Return FALSE on error otherwise the string.

ifxus_seek_slob (PHP 3>= 3.0.4, PHP 4)

Sets the current file or seek position

int **ifxus_seek_slob** (long bid, int mode, long offset) \linebreak

Sets the current file or seek position of an open slob object. *bid* should be an existing slob id. Modes: 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END and *offset* is an byte offset. Return FALSE on error otherwise the seek position.

ifxus_tell_slob (PHP 3>= 3.0.4, PHP 4)

Returns the current file or seek position

int **ifxus_tell_slob** (long bid) \linebreak

Returns the current file or seek position of an open slob object *`bid`* should be an existing slob id. Return FALSE on error otherwise the seek position.

ifxus_write_slob (PHP 3>= 3.0.4, PHP 4)

Writes a string into the slob object

int **ifxus_write_slob** (long bid, string content) \linebreak

Writes a string into the slob object. *`bid`* is a existing slob id and *`content`* the content to write. Return FALSE on error otherwise bytes written.

XLV. InterBase functions

Bevezetés

InterBase is a popular database put out by Borland/Inprise. More information about InterBase is available at http://www.interbase.com/. Oh, by the way, InterBase just joined the open source movement!

Megjegyzés: Full support for InterBase 6 was added in PHP 4.0.

This database uses a single quote (') character for escaping, a behavior similar to the Sybase database, add to your `php.ini` the following directive:

```
magic_quotes_sybase = On
```

Követelmények

Telepítés

Futásidejű beállítások

Erőforrás típusok

Előre definiált állandók

Az itt listázott állandókat ez a kiterjesztés definiálja, és csak akkor elérhetőek, ha az adott

kiterjesztés be van fordítva a PHP-be, vagy dinamikusan betöltött.

`IBASE_DEFAULT` (integer)

`IBASE_TEXT` (integer)

`IBASE_UNIXTIME` (integer)

`IBASE_READ` (integer)

`IBASE_COMMITTED` (integer)

`IBASE_CONSISTENCY` (integer)

`IBASE_NOWAIT` (integer)

`IBASE_TIMESTAMP` (integer)

`IBASE_DATE` (integer)

`IBASE_TIME` (integer)

ibase_blob_add (PHP 3>= 3.0.7, PHP 4)

Add data into created blob

int **ibase_blob_add** (int blob_id, string data) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ibase_blob_cancel (PHP 3>= 3.0.7, PHP 4)

Cancel creating blob

int **ibase_blob_cancel** (int blob_id) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ibase_blob_close (PHP 3>= 3.0.7, PHP 4)

Close blob

int **ibase_blob_close** (int blob_id) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ibase_blob_create (PHP 3>= 3.0.7, PHP 4)

Create blob for adding data

int **ibase_blob_create** ([int link_identifier]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ibase_blob_echo (PHP 3>= 3.0.7, PHP 4)

Output blob contents to browser

int **ibase_blob_echo** (string blob_id_str) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ibase_blob_get (PHP 3>= 3.0.7, PHP 4)

Get len bytes data from open blob

string **ibase_blob_get** (int blob_id, int len) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ibase_blob_import (PHP 3>= 3.0.7, PHP 4)

Create blob, copy file in it, and close it

string **ibase_blob_import** ([int link_identifier, int file_id]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ibase_blob_info (PHP 3>= 3.0.7, PHP 4)

Return blob length and other useful info

object **ibase_blob_info** (string blob_id_str) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ibase_blob_open (PHP 3>= 3.0.7, PHP 4)

Open blob for retrieving data parts

int **ibase_blob_open** (string blob_id) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ibase_close (PHP 3>= 3.0.6, PHP 4)

Close a connection to an InterBase database

int **ibase_close** ([int connection_id]) \linebreak

Closes the link to an InterBase database that’s associated with a connection id returned from ibase_connect(). If the connection id is omitted, the last opened link is assumed. Default transaction on link is committed, other transactions are rolled back.

ibase_commit (PHP 3>= 3.0.7, PHP 4)

Commit a transaction

int **ibase_commit** ([int link_identifier, int trans_number]) \linebreak

Commits transaction *trans_number* which was created with ibase_trans().

ibase_connect (PHP 3>= 3.0.6, PHP 4)

Open a connection to an InterBase database

int **ibase_connect** (string database [, string username [, string password [, string charset [, int buffers [, int dialect [, string role]]]]]]) \linebreak

Establishes a connection to an InterBase server. The *database* argument has to be a valid path to database file on the server it resides on. If the server is not local, it must be prefixed with either 'hostname:' (TCP/IP), '//hostname/' (NetBEUI) or 'hostname@' (IPX/SPX), depending on the connection protocol used. *username* and *password* can also be specified with PHP configuration directives ibase.default_user and ibase.default_password. *charset* is the default character set for a database. *buffers* is the number of database buffers to allocate for the server-side cache. If 0 or omitted, server chooses its own default. *dialect* selects the default SQL dialect for any statement executed within a connection, and it defaults to the highest one supported by client libraries.

In case a second call is made to **ibase_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned. The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling ibase_close().

Példa 1. ibase_connect() example

```
<?php
    $dbh = ibase_connect($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query($dbh, $stmt);
    while ($row = ibase_fetch_object($sth)) {
        echo $row->email, "\n";
    }
    ibase_free_result($sth);
    ibase_close($dbh);
?>
```

Megjegyzés: *buffers* was added in PHP 4.0RC2.

Megjegyzés: *dialect* was added in PHP 4.0RC2. It is functional only with InterBase 6 and versions higher than that.

Megjegyzés: *role* was added in PHP 4.0RC2. It is functional only with InterBase 5 and versions higher than that.

See also ibase_pconnect().

ibase_errmsg (PHP 3>= 3.0.7, PHP 4)

Returns error messages

string **ibase_errmsg** (void) \linebreak

Returns a string containing an error message.

ibase_execute (PHP 3>= 3.0.6, PHP 4)

Execute a previously prepared query

int **ibase_execute** (int query [, int bind_args]) \linebreak

Execute a query prepared by ibase_prepare(). This is a lot more effective than using ibase_query() if you are repeating a same kind of query several times with only some parameters changing.

```
<?php
    $updates = array(
        1 => 'Eric',
        5 => 'Filip',
        7 => 'Larry'
    );

    $query = ibase_prepare("UPDATE FOO SET BAR = ? WHERE BAZ = ?");

    while (list($baz, $bar) = each($updates)) {
        ibase_execute($query, $bar, $baz);
    }
?>
```

ibase_fetch_object (PHP 3>= 3.0.7, PHP 4)

Get an object from a InterBase database

object **ibase_fetch_object** (int result_id) \linebreak

Fetches a row as a pseudo-object from a *`result_id`* obtained either by ibase_query() or ibase_execute().

```
<php
    $dbh = ibase_connect ($host, $username, $password);
```

```
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query ($dbh, $stmt);
    while ($row = ibase_fetch_object ($sth)) {
        print $row->email . "\n";
    }
    ibase_close ($dbh);
?>
```

Subsequent call to **ibase_fetch_object()** would return the next row in the result set, or FALSE if there are no more rows.

See also ibase_fetch_row().

ibase_fetch_row (PHP 3>= 3.0.6, PHP 4)

Fetch a row from an InterBase database

array **ibase_fetch_row** (int result_identifier) \linebreak

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

ibase_fetch_row() fetches one row of data from the result associated with the specified *`result_identifier`*. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **ibase_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

ibase_field_info (PHP 3>= 3.0.7, PHP 4)

Get information about a field

array **ibase_field_info** (int result, int field number) \linebreak

Returns an array with information about a field after a select query has been run. The array is in the form of name, alias, relation, length, type.

```
$rs=ibase_query("SELECT * FROM tablename");
$coln = ibase_num_fields($rs);
for ($i=0; $i < $coln; $i++) {
    $col_info = ibase_field_info($rs, $i);
    echo "name: ".$col_info['name']."\n";
    echo "alias: ".$col_info['alias']."\n";
    echo "relation: ".$col_info['relation']."\n";
    echo "length: ".$col_info['length']."\n";
    echo "type: ".$col_info['type']."\n";
    }
```

ibase_free_query (PHP 3>= 3.0.6, PHP 4)

Free memory allocated by a prepared query

int **ibase_free_query** (int query) \linebreak

Free a query prepared by ibase_prepare().

ibase_free_result (PHP 3>= 3.0.6, PHP 4)

Free a result set

int **ibase_free_result** (int result_identifier) \linebreak

Free’s a result set the has been created by ibase_query().

ibase_num_fields (PHP 3>= 3.0.7, PHP 4)

Get the number of fields in a result set

int **ibase_num_fields** (int result_id) \linebreak

Returns an integer containing the number of fields in a result set.

```
<?php
$dbh = ibase_connect ($host, $username, $password);
$stmt = ’SELECT * FROM tblname’;
$sth = ibase_query ($dbh, $stmt);

if (ibase_num_fields($sth) > 0) {
while ($row = ibase_fetch_object ($sth)) {
print $row->email . "\n";
}
} else {
die ("No Results were found for your query");
}

ibase_close ($dbh);
?>
```

See also: ibase_field_info().

ibase_pconnect (PHP 3>= 3.0.6, PHP 4)

Creates an persistent connection to an InterBase database

int **ibase_pconnect** (string database [, string username [, string password [, string charset [, int buffers [, int dialect [, string role]]]]]]) \linebreak

ibase_pconnect() acts very much like ibase_connect() with two major differences. First, when connecting, the function will first try to find a (persistent) link that's already opened with the same parameters. If one is found, an identifier for it will be returned instead of opening a new connection. Second, the connection to the InterBase server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (ibase_close() will not close links established by **ibase_pconnect()**). This type of link is therefore called 'persistent'.

Megjegyzés: `buffers` was added in PHP4-RC2.

Megjegyzés: `dialect` was added in PHP4-RC2. It is functional only with InterBase 6 and versions higher than that.

Megjegyzés: `role` was added in PHP4-RC2. It is functional only with InterBase 5 and versions higher than that.

See also ibase_connect() for the meaning of parameters passed to this function. They are exactly the same.

ibase_prepare (PHP 3>= 3.0.6, PHP 4)

Prepare a query for later binding of parameter placeholders and execution

int **ibase_prepare** ([int link_identifier, string query]) \linebreak

Prepare a query for later binding of parameter placeholders and execution (via ibase_execute()).

ibase_query (PHP 3>= 3.0.6, PHP 4)

Execute a query on an InterBase database

int **ibase_query** ([int link_identifier, string query [, int bind_args]]) \linebreak

Performs a query on an InterBase database. If the query is not successful, returns `FALSE`. If it is successful and there are resulting rows (such as with a SELECT query), returns a result identifier. If the query was successful and there were no results, returns `TRUE`. Returns `FALSE` if the query fails.

See also ibase_errmsg(), ibase_fetch_row(), ibase_fetch_object(), and ibase_free_result().

ibase_rollback (PHP 3>= 3.0.7, PHP 4)

Rolls back a transaction

int **ibase_rollback** ([int link_identifier, int trans_number]) \linebreak

Rolls back transaction *`trans_number`* which was created with ibase_trans().

ibase_timefmt (PHP 3>= 3.0.6, PHP 4)

Sets the format of timestamp, date and time type columns returned from queries

int **ibase_timefmt** (string format [, int columntype]) \linebreak

Sets the format of timestamp, date or time type columns returned from queries. Internally, the columns are formatted by c-function strftime(), so refer to it's documentation regarding to the format of the string. *`columntype`* is one of the constants IBASE_TIMESTAMP, IBASE_DATE and IBASE_TIME. If omitted, defaults to IBASE_TIMESTAMP for backwards compatibility.

```
<?php
// InterBase 6 TIME-type columns will be returned in
// the form '05 hours 37 minutes'.
ibase_timefmt("%H hours %M minutes", IBASE_TIME);
?>
```

You can also set defaults for these formats with PHP configuration directives ibase.timestampformat, ibase.dateformat and ibase.timeformat.

Megjegyzés: *`columntype`* was added in PHP 4.0. It has any meaning only with InterBase version 6 and higher.

Megjegyzés: A backwards incompatible change happened in PHP 4.0 when PHP configuration directive ibase.timeformat was renamed to ibase.timestampformat and directives ibase.dateformat and ibase.timeformat were added, so that the names would match better their functionality.

ibase_trans (PHP 3>= 3.0.7, PHP 4)

Begin a transaction

int **ibase_trans** ([int trans_args [, int link_identifier]]) \linebreak

Begins a transaction.

XLVI. Ingres II functions

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

These functions allow you to access Ingres II database servers.

In order to have these functions available, you must compile php with Ingres support by using the `--with-ingres` option. You need the Open API library and header files included with Ingres II. If the II_SYSTEM environment variable isn't correctly set you may have to use `--with-ingres=DIR` to specify your Ingres installation directory.

When using this extension with Apache, if Apache does not start and complains with "PHP Fatal error: Unable to start ingres_ii module in Unknown on line 0" then make sure the environement variable II_SYSTEM is correctly set. Adding "export II_SYSTEM="/home/ingres/II" in the script that starts Apache, just before launching httpd, should be fine.

> **Megjegyzés:** If you already used PHP extensions to access other database servers, note that Ingres doesn't allow concurrent queries and/or transaction over one connection, thus you won't find any result or transaction handle in this extension. The result of a query must be treated before sending another query, and a transaction must be commited or rolled back before opening another transaction (which is automaticaly done when sending the first query).

ingres_autocommit (PHP 4 >= 4.0.2)

Switch autocommit on or off

bool **ingres_autocommit** ([resource link]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_autocommit() is called before opening a transaction (before the first call to ingres_query() or just after a call to ingres_rollback() or **ingres_autocommit()**) to switch the "autocommit" mode of the server on or off (when the script begins the autocommit mode is off).

When the autocommit mode is on, every query is automaticaly commited by the server, as if ingres_commit() was called after every call to ingres_query().

See also ingres_query(), ingres_rollback(), and ingres_commit().

ingres_close (PHP 4 >= 4.0.2)

Close an Ingres II database connection

bool **ingres_close** ([resource link]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Returns TRUE on success, or FALSE on failure.

ingres_close() closes the connection to the Ingres server that's associated with the specified link. If the *link* parameter isn't specified, the last opened link is used.

ingres_close() isn't usually necessary, as it won't close persistent connections and all non-persistent connections are automatically closed at the end of the script.

See also ingres_connect() and ingres_pconnect().

ingres_commit (PHP 4 >= 4.0.2)

Commit a transaction

bool **ingres_commit** ([resource link]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_commit() commits the currently open transaction, making all changes made to the database permanent.

This closes the transaction. A new one can be open by sending a query with ingres_query().

You can also have the server commit automaticaly after every query by calling ingres_autocommit() before opening the transaction.

See also ingres_query(), ingres_rollback(), and ingres_autocommit().

ingres_connect (PHP 4 >= 4.0.2)

Open a connection to an Ingres II database

resource **ingres_connect** ([string database [, string username [, string password]]]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Returns a Ingres II link resource on success, or FALSE on failure.

ingres_connect() opens a connection with the Ingres database designated by *database*, which follows the syntax *[node_id::]dbname[/svr_class]*.

If some parameters are missing, **ingres_connect()** uses the values in php.ini for *ingres.default_database*, *ingres.default_user*, and *ingres.default_password*.

The connection is closed when the script ends or when ingres_close() is called on this link.

All the other ingres functions use the last opened link as a default, so you need to store the returned value only if you use more than one link at a time.

Példa 1. ingres_connect() example

```
<?php
    $link = ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ($link);
?>
```

Példa 2. ingres_connect() example using default link

```
<?php
    ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ();
?>
```

See also ingres_pconnect() and ingres_close().

ingres_fetch_array (PHP 4 >= 4.0.2)

Fetch a row of result into an array

array **ingres_fetch_array** ([int result_type [, resource link]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_fetch_array() Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

This function is an extended version of ingres_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use the numeric index of the column or make an alias for the column.

```
ingres_query(select t1.f1 as foo t2.f1 as bar from t1, t2);
$result = ingres_fetch_array();
$foo = $result["foo"];
$bar = $result["bar"];
```

result_type can be INGRES_NUM for enumerated array, INGRES_ASSOC for associative array, or INGRES_BOTH (default).

Speed-wise, the function is identical to ingres_fetch_object(), and almost as quick as ingres_fetch_row() (the difference is insignificant).

Példa 1. ingres_fetch_array() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_array()) {
    echo $row["user_id"];  # using associative array
    echo $row["fullname"];
    echo $row[1];          # using enumerated array
    echo $row[2];
}
?>
```

See also ingres_query(), ingres_num_fields(), ingres_field_name(), ingres_fetch_object(), and ingres_fetch_row().

ingres_fetch_object (PHP 4 >= 4.0.2)

Fetch a row of result into an object.

object **ingres_fetch_object** ([int result_type [, resource link]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_fetch_object() Returns an object that corresponds to the fetched row, or FALSE if there are no more rows.

This function is similar to ingres_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: INGRES_ASSOC, INGRES_NUM, and INGRES_BOTH.

Speed-wise, the function is identical to ingres_fetch_array(), and almost as quick as ingres_fetch_row() (the difference is insignificant).

Példa 1. ingres_fetch_object() example

```
<?php
ingres_connect ($database, $user, $password);
ingres_query ("select * from table");
```

```
while ($row = ingres_fetch_object()) {
    echo $row->user_id;
    echo $row->fullname;
}
?>
```

See also ingres_query(), ingres_num_fields(), ingres_field_name(), ingres_fetch_array(), and ingres_fetch_row().

ingres_fetch_row (PHP 4 >= 4.0.2)

Fetch a row of result into an enumerated array

array **ingres_fetch_row** ([resource link]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_fetch_row() returns an array that corresponds to the fetched row, or FALSE if there are no more rows. Each result column is stored in an array offset, starting at offset 1.

Subsequent call to **ingres_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

Példa 1. ingres_fetch_row() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also ingres_num_fields(), ingres_query(), ingres_fetch_array(), and ingres_fetch_object().

ingres_field_length (PHP 4 >= 4.0.2)

Get the length of a field

int **ingres_field_length** (int index [, resource link]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_field_length() returns the length of a field. This is the number of bytes used by the server to store the field. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_field_name (PHP 4 >= 4.0.2)

Get the name of a field in a query result.

string **ingres_field_name** (int index [, resource link]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_field_name() returns the name of a field in a query result, or FALSE on failure.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() and ingres_fetch_row().

ingres_field_nullable (PHP 4 >= 4.0.2)

Test if a field is nullable

bool **ingres_field_nullable** (int index [, resource link]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_field_nullable() returns `TRUE` if the field can be set to the `NULL` value and `FALSE` if it can't.

`index` is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_field_precision (PHP 4 >= 4.0.2)

Get the precision of a field

int **ingres_field_precision** (int index [, resource link]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_field_precision() returns the precision of a field. This value is used only for decimal, float and money SQL data types. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

`index` is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_field_scale (PHP 4 >= 4.0.2)

Get the scale of a field

int **ingres_field_scale** (int index [, resource link]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_field_scale() returns the scale of a field. This value is used only for the decimal SQL data type. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

`index` is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_field_type (PHP 4 >= 4.0.2)

Get the type of a field in a query result

string **ingres_field_type** (int index [, resource link]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_field_type() returns the type of a field in a query result, or FALSE on failure. Examples of types returned are "IIAPI_BYTE_TYPE", "IIAPI_CHA_TYPE", "IIAPI_DTE_TYPE", "IIAPI_FLT_TYPE", "IIAPI_INT_TYPE", "IIAPI_VCH_TYPE". Some of these types can map to more than one SQL type depending on the length of the field (see ingres_field_length()). For example "IIAPI_FLT_TYPE" can be a float4 or a float8. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_num_fields (PHP 4 >= 4.0.2)

Get the number of fields returned by the last query

int **ingres_num_fields** ([resource link]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_num_fields() returns the number of fields in the results returned by the Ingres server after a call to ingres_query()

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_num_rows (PHP 4 >= 4.0.2)

Get the number of rows affected or returned by the last query

int **ingres_num_rows** ([resource link]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

For delete, insert or update queries, **ingres_num_rows()** returns the number of rows affected by the query. For other queries, **ingres_num_rows()** returns the number of rows in the query's result.

> **Megjegyzés:** This function is mainly meant to get the number of rows modified in the database. If this function is called before using ingres_fetch_array(), ingres_fetch_object() or ingres_fetch_row() the server will delete the result's data and the script won't be able to get them.
>
> You should instead retrieve the result's data using one of these fetch functions in a loop until it returns FALSE, indicating that no more results are available.

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_pconnect (PHP 4 >= 4.0.2)

Open a persistent connection to an Ingres II database

resource **ingres_pconnect** ([string database [, string username [, string password]]]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Returns a Ingres II link resource on success, or FALSE on failure.

See ingres_connect() for parameters details and examples. There are only 2 differences between **ingres_pconnect()** and ingres_connect() : First, when connecting, the function will first try to find a (persistent) link that's already opened with the same parameters. If one is found, an identifier for it will be returned instead of opening a new connection. Second, the connection to the Ingres server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (ingres_close() will not close links established by **ingres_pconnect()**). This type of link is therefore called 'persistent'.

See also ingres_connect() and ingres_close().

ingres_query (PHP 4 >= 4.0.2)

Send a SQL query to Ingres II

bool **ingres_query** (string query [, resource link]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Returns TRUE on success, or FALSE on failure.

ingres_query() sends the given *query* to the Ingres server. This query must be a valid SQL query (see the Ingres SQL reference guide)

The query becomes part of the currently open transaction. If there is no open transaction, **ingres_query()** opens a new transaction. To close the transaction, you can either call ingres_commit() to commit the changes made to the database or ingres_rollback() to cancel these changes. When the script ends, any open transaction is rolled back (by calling ingres_rollback()). You can also use ingres_autocommit() before opening a new transaction to have every SQL query immediatly commited.

Some types of SQL queries can’t be sent with this function:

- close (see ingres_close())
- commit (see ingres_commit())
- connect (see ingres_connect())
- disconnect (see ingres_close())
- get dbevent
- prepare to commit
- rollback (see ingres_rollback())
- savepoint
- set autocommit (see ingres_autocommit())
- all cursor related queries are unsupported

Példa 1. ingres_query() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also ingres_fetch_array(), ingres_fetch_object(), ingres_fetch_row(), ingres_commit(), ingres_rollback(), and ingres_autocommit().

ingres_rollback (PHP 4 >= 4.0.2)

Roll back a transaction

bool **ingres_rollback** ([resource link]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ingres_rollback() rolls back the currently open transaction, actually canceling all changes made to the database during the transaction.

This closes the transaction. A new one can be open by sending a query with ingres_query().

See also ingres_query(), ingres_commit(), and ingres_autocommit().

XLVII. IRC Gateway Functions

What is ircg?

With ircg you can build powerful, fast and scalable webchat solutions in conjunction with dedicated or public IRC servers.

Platforms

IRCG runs under

- AIX
- FreeBSD
- HP-UX
- Irix
- Linux
- Solaris
- Tru64

Requirements

To install and use IRCG you need the following software:

1. IRCG-Library (http://www.schumann.cx/ircg/) from Sascha Schumann.
2. SGI Static Threads Library (http://sourceforge.net/projects/state-threads/)
3. thttpd (http://www.acme.com/software/thttpd/) webserver

Installation

A detailed installation instruction can be found here (http://lxr.php.net/source/php4/ext/ircg/README.txt).

ircg_channel_mode (PHP 4 >= 4.0.5)

Set channel mode flags for user

boolean **ircg_channel_mode** (resource connection, string channel, string mode_spec, string nick) \linebreak

Set channel mode flags for *`channel`* on server connected to by *`connection`*. Mode flags are passed in *`mode_spec`* and are applied to the user specified by *`nick`*.

Mode flags are set or cleared by specifind a mode character and prepending it with a plus or minus character respectively. E.g. operator mode is granted by '+o' and revoked by '-o' passed as *`mode_spec`*.

ircg_disconnect (PHP 4 >= 4.0.4)

Close connection to server

boolean **ircg_disconnect** (resource connection, string reason) \linebreak

ircg_disconnect() will close a *`connection`* to a server previously established with ircg-pconnect().

See also: ircg_pconnect().

ircg_fetch_error_msg (PHP 4 >= 4.1.0)

Returns the error from previous ircg operation

array **ircg_fetch_error_msg** (resource connection) \linebreak

ircg_fetch_error_msg() returns the error from the last called ircg function.

Megjegyzés: Errorcode is stored in first array element, errortext in second.

Példa 1. ircg_fetch_error_msg() example

```
if (!ircg_join ($id, "#php")) {
    $error = ircg_fetch_error_msg($id);
    print ("Can't join channel #php. Errorcode:
            $error[0] Description: $error[1]");
}
```

ircg_get_username (PHP 4 >= 4.1.0)

Get username for connection

string **ircg_get_username** (int connection) \linebreak

Function **ircg_get_username()** returns the username for specified connection *connection*. Returns FALSE if *connection* died or is not valid.

ircg_html_encode (PHP 4 >= 4.0.5)

Encodes HTML preserving output

boolean **ircg_html_encode** (string html_string) \linebreak

Encodes a HTML string *html_string* for output. This feature could be usable, e.g. if someone wants to discuss about an html problem.

ircg_ignore_add (PHP 4 >= 4.0.5)

Add a user to your ignore list on a server

boolean **ircg_ignore_add** (resource connection, string nick) \linebreak

This function will add user *nick* to your ignore list on the server connected to by *connection*. By doing so you will suppress all messages from this user from being send to you.

See also: ircg_ignore_del().

ircg_ignore_del (PHP 4 >= 4.0.5)

Remove a user from your ignore list on a server

boolean **ircg_ignore_del** (resource connection, string nick) \linebreak

This function remove user *nick* from your ignore list on the server connected to by *connection*.

See also: ircg_ignore_add().

ircg_is_conn_alive (PHP 4 >= 4.0.5)

Check connection status

boolean **ircg_is_conn_alive** (resource connection) \linebreak

ircg_is_conn_alive() returns TRUE if *connection* is still alive and working or FALSE if the server no longer talks to us.

ircg_join (PHP 4 >= 4.0.4)

Join a channel on a connected server

boolean **ircg_join** (resource connection, string channel) \linebreak

Join the channel *channel* on the server connected to by *connection*.

ircg_kick (PHP 4 >= 4.0.5)

Kick a user out of a channel on server

boolean **ircg_kick** (resource connection, string channel, string nick, string reason) \linebreak

Kick user *nick* from *channel* on server connected to by *connection*. *reason* should give a short message describing why this action was performed.

ircg_lookup_format_messages (PHP 4 >= 4.0.5)

Select a set of format strings for display of IRC messages

boolean **ircg_lookup_format_messages** (string name) \linebreak

Select the set of format strings to use for display of IRC messages and events. Sets may be registered with ircg_register_format_messages(), a default set named `ircg` is always available.

See also: ircg_register_format_messages()

ircg_msg (PHP 4 >= 4.0.4)

Send message to channel or user on server

boolean **ircg_msg** (resource connection, string recipient, string message [, boolean suppress]) \linebreak

ircg_msg() will send the message to a channel or user on the server connected to by *connection*. A *recipient* starting with # or & will send the *message* to a channel, anything else will be interpreted as a username.

Setting the optional parameter *suppress* to a TRUE value will suppress output of your message to your own *connection*.

ircg_nick (PHP 4 >= 4.0.5)

Change nickname on server

boolean **ircg_nick** (resource connection, string nick) \linebreak

Change your nickname on the given *connection* to the one given in *nick* if possible.

Will return TRUE on success and FALSE on failure.

ircg_nickname_escape (PHP 4 >= 4.0.6)

Encode special characters in nickname to be IRC-compliant

string **ircg_nickname_escape** (string nick) \linebreak

Function **ircg_nickname_escape()** returns a decoded nickname specified by *nick* wich is IRC-compliant.

See also: ircg_nickname_unescape()

ircg_nickname_unescape (PHP 4 >= 4.0.6)

Decodes encoded nickname

string **ircg_nickname_unescape** (string nick) \linebreak

Function **ircg_nickname_unescape()** returns a decoded nickname, which is specified in *nick*.

See also: ircg_nickname_escape()

ircg_notice (PHP 4 >= 4.0.5)

Send a notice to a user on server

boolean **ircg_notice** (resource connection, string , string message) \linebreak

This function will send the *message* text to the user *nick* on the server connected to by *connection*. Query your IRC documentation of choice for the exact difference between a MSG and a NOTICE.

ircg_part (PHP 4 >= 4.0.4)

Leave a channel on server

boolean **ircg_part** (resource connection, string channel) \linebreak

Leave the channel *channel* on the server connected to by *connection*.

ircg_pconnect (PHP 4 >= 4.0.4)

Connect to an IRC server

resource **ircg_pconnect** (string username [, string server_ip [, int server_port [, string msg_format [, array ctcp_messages [, array user_settings]]]]]) \linebreak

ircg_pconnect() will try to establish a connection to an IRC server and return a connection resource handle for further use.

The only mandatory parameter is *`username`*, this will set your initial nickname on the server. *`server_ip`* and *`server_port`* are optional and default to `127.0.0.1` and `6667`.

Megjegyzés: For now parameter *`server_ip`* will not do any hostname lookups and will only accept IP addresses in numerical form.

Currently **ircg_pconnect()** always returns a valid handle, even if the connection failed.

You can customize the output of IRC messages and events by selection a format string set previously created with ircg_register_format_messages() by specifing the sets name in *`msg_format`*.

`ctcp_messages`

`user_settings`

See also: ircg_disconnect(), ircg_is_conn_alive(), ircg_register_format_messages().

ircg_register_format_messages (PHP 4 >= 4.0.5)

Register a set of format strings for display of IRC messages

boolean **ircg_register_format_messages** (string name, array messages) \linebreak

With **ircg_register_format_messages()** you can customize the way your IRC output looks like. You can even register different format string sets and switch between them on the fly with ircg_lookup_format_messages().

- Plain channel message
- Private message received
- Private message sent
- Some user leaves channel
- Some user enters channel
- Some user was kicked from the channel
- Topic has been changed
- Error
- Fatal error
- Join list end(?)

- Self part(?)
- Some user changes his nick
- Some user quits his connection
- Mass join begin
- Mass join element
- Mass join end
- Whois user
- Whois server
- Whois idle
- Whois channel
- Whois end
- Voice status change on user
- Operator status change on user
- Banlist
- Banlist end

- %f - from
- %t - to
- %c - channel
- %r - plain message
- %m - encoded message
- %j - js encoded message

- 1 - mod encode
- 2 - nickname decode

See also: ircg_lookup_format_messages().

ircg_set_current (PHP 4 >= 4.0.4)

Set current connection for output

boolean **ircg_set_current** (resource connection) \linebreak

Select the current connection for output in this execution context. Every output sent from the server connected to by *`connection`* will be copied to standard output while using default formatting or a format string set specified by ircg_register_format_messages() and selected by ircg_lookup_format_messages().

See also: ircg_register_format_messages() and ircg_lookup_format_messages().

ircg_set_file (PHP 4 >= 4.2.0)

Set logfile for connection

bool **ircg_set_file** (int connection, string path) \linebreak

Function **ircg_set_file()** specifies a logfile *path* in which all output from connection *connection* will be logged. Returns TRUE on success, otherwise FALSE.

ircg_set_on_die (PHP 4 >= 4.2.0)

Set hostaction to be execute when connection dies

bool **ircg_set_on_die** (int connection, string host, int port, string data) \linebreak

In case of the termination of connection *connection* IRCG will connect to *host* at *port* (Note: host must be an IPv4 address, IRCG does not resolve host-names due to blocking issues), send *data* to the new host connection and will wait until the remote part closes connection. This can be used to trigger a php script for example.

ircg_topic (PHP 4 >= 4.0.5)

Set topic for channel on server

boolean **ircg_topic** (resource connection, string channel, string new_topic) \linebreak

Change the topic for channel *channel* on the server connected to by *connection* to *new_topic*.

ircg_whois (PHP 4 >= 4.0.5)

Query user information for nick on server

boolean **ircg_whois** (resource connection, string nick) \linebreak

Sends a query to the connected server *connection* to send information for the specified user *nick*.

XLVIII. Java

There are two possible ways to bridge PHP and Java: you can either integrate PHP into a Java Servlet environment, which is the more stable and efficient solution, or integrate Java support into PHP. The former is provided by a SAPI module that interfaces with the Servlet server, the latter by the Java extension.

PHP 4 ext/java provides a simple and effective means for creating and invoking methods on Java objects from PHP. The JVM is created using JNI, and everything runs in-process. Build instructions for ext/java can be found in `php4/ext/java/README`.

Példa 1. Java Example

```
<?php
  // get instance of Java class java.lang.System in PHP
  $system = new Java('java.lang.System');

  // demonstrate property access
  print 'Java version='.$system->getProperty('java.version').' <br>';
  print 'Java vendor=' .$system->getProperty('java.vendor').'  <br>';
  print 'OS='.$system->getProperty('os.name').' '.
              $system->getProperty('os.version').' on '.
              $system->getProperty('os.arch').' <br>';

  // java.util.Date example
  $formatter = new Java('java.text.SimpleDateFormat',
                        "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");

  print $formatter->format(new Java('java.util.Date'));
?>
```

Példa 2. AWT Example

```
<?php
  // This example is only intented to be run as a CGI.

  $frame  = new Java('java.awt.Frame', 'PHP');
  $button = new Java('java.awt.Button', 'Hello Java World!');

  $frame->add('North', $button);
  $frame->validate();
  $frame->pack();
  $frame->visible = True;

  $thread = new Java('java.lang.Thread');
  $thread->sleep(10000);

  $frame->dispose();
?>
```

Notes:

- `new Java()` will create an instance of a class if a suitable constructor is available. If no parameters are passed and the default constructor is useful as it provides access to classes like `java.lang.System` which expose most of their functionallity through static methods.
- Accessing a member of an instance will first look for bean properties then public fields. In other words, `print $date.time` will first attempt to be resolved as `$date.getTime()`, then as `$date.time`.
- Both static and instance members can be accessed on an object with the same syntax. Furthermore, if the java object is of type `java.lang.Class`, then static members of the class (fields and methods) can be accessed.
- Exceptions raised result in PHP warnings, and `NULL` results. The warnings may be eliminated by prefixing the method call with an "@" sign. The following APIs may be used to retrieve and reset the last error:
  - java_last_exception_get()
  - java_last_exception_clear()
- Overload resolution is in general a hard problem given the differences in types between the two languages. The PHP Java extension employs a simple, but fairly effective, metric for determining which overload is the best match.

  Additionally, method names in PHP are not case sensitive, potentially increasing the number of overloads to select from.

  Once a method is selected, the parameters are cooerced if necessary, possibly with a loss of data (example: double precision floating point numbers will be converted to boolean).

- In the tradition of PHP, arrays and hashtables may pretty much be used interchangably. Note that hashtables in PHP may only be indexed by integers or strings; and that arrays of primitive types in Java can not be sparse. Also note that these constructs are passed by value, so may be expensive in terms of memory and time.

sapi/servlet builds upon the mechanism defined by ext/java to enable the entire PHP processor to be run as a servlet. The primary advanatage of this from a PHP perspective is that web servers which support servlets typically take great care in pooling and reusing JVMs. Build instructions for the Servlet SAPI module can be found in `php4/sapi/README`. Notes:

- While this code is intended to be able to run on any servlet engine, it has only been tested on Apache’s Jakarta/tomcat to date. Bug reports, success stories and/or patches required to get this code to run on other engines would be appreciated.
- PHP has a habit of changing the working directory. sapi/servlet will eventually change it back, but while PHP is running the servlet engine may not be able to load any classes from the CLASSPATH which are specified using a relative directory syntax, or find the work directory used for administration and JSP compilation tasks.

java_last_exception_clear (PHP 4 >= 4.0.2)

Clear last Java exception

void **java_last_exception_clear** (void) \linebreak

java_last_exception_get (PHP 4 >= 4.0.2)

Get last Java exception

exception **java_last_exception_get** (void) \linebreak

The following example demonstrates the usage of Java's exception handler from within PHP:

Példa 1. Java exception handler

```
<?php
  $stack = new Java('java.util.Stack');
  $stack->push(1);

  // This should succeed
  $result = $stack->pop();
  $ex = java_last_exception_get();
  if (!$ex) print "$result\n";

  // This should fail (error suppressed by @)
  $result = @$stack->pop();
  $ex = java_last_exception_get();
  if ($ex) print $ex->toString();

  // Clear last exception
  java_last_exception_clear();
?>
```

XLIX. LDAP functions

Introduction to LDAP

LDAP is the Lightweight Directory Access Protocol, and is a protocol used to access "Directory Servers". The Directory is a special kind of database that holds information in a tree structure.

The concept is similar to your hard disk directory structure, except that in this context, the root directory is "The world" and the first level subdirectories are "countries". Lower levels of the directory structure contain entries for companies, organisations or places, while yet lower still we find directory entries for people, and perhaps equipment or documents.

To refer to a file in a subdirectory on your hard disk, you might use something like

```
/usr/local/myapp/docs
```

The forwards slash marks each division in the reference, and the sequence is read from left to right.

The equivalent to the fully qualified file reference in LDAP is the "distinguished name", referred to simply as "dn". An example dn might be.

```
cn=John Smith,ou=Accounts,o=My Company,c=US
```

The comma marks each division in the reference, and the sequence is read from right to left. You would read this dn as ..

```
country = US
organization = My Company
organizationalUnit = Accounts
commonName = John Smith
```

In the same way as there are no hard rules about how you organise the directory structure of a hard disk, a directory server manager can set up any structure that is meaningful for the purpose. However, there are some conventions that are used. The message is that you can not write code to access a directory server unless you know something about its structure, any more than you can use a database without some knowledge of what is available.

Complete code example

Retrieve information for all entries where the surname starts with "S" from a directory server, displaying an extract with name and email address.

Példa 1. LDAP search example

```
<?php
// basic sequence with LDAP is connect, bind, search, interpret search
// result, close connection
```

```
echo "<h3>LDAP query test</h3>";
echo "Connecting ...";
$ds=ldap_connect("localhost");  // must be a valid LDAP server!
echo "connect result is ".$ds."<p>";

if ($ds) {
    echo "Binding ...";
    $r=ldap_bind($ds);     // this is an "anonymous" bind, typically
                           // read-only access
    echo "Bind result is ".$r."<p>";

    echo "Searching for (sn=S*) ...";
    // Search surname entry
    $sr=ldap_search($ds,"o=My Company, c=US", "sn=S*");
    echo "Search result is ".$sr."<p>";

    echo "Number of entires returned is ".ldap_count_entries($ds,$sr)."<p>";

    echo "Getting entries ...<p>";
    $info = ldap_get_entries($ds, $sr);
    echo "Data for ".$info["count"]." items returned:<p>";

    for ($i=0; $i<$info["count"]; $i++) {
        echo "dn is: ". $info[$i]["dn"] ."<br>";
        echo "first cn entry is: ". $info[$i]["cn"][0] ."<br>";
        echo "first email entry is: ". $info[$i]["mail"][0] ."<p>";
    }

    echo "Closing connection";
    ldap_close($ds);

} else {
    echo "<h4>Unable to connect to LDAP server</h4>";
}
?>
```

Using the PHP LDAP calls

You will need to get and compile LDAP client libraries from either the University of Michigan ldap-3.3 package or the Netscape Directory SDK 3.0. You will also need to recompile PHP with LDAP support enabled before PHP’s LDAP calls will work.

Before you can use the LDAP calls you will need to know ..

- The name or address of the directory server you will use
- The "base dn" of the server (the part of the world directory that is held on this server, which could be "o=My Company,c=US")
- Whether you need a password to access the server (many servers will provide read access for an

"anonymous bind" but require a password for anything else)

The typical sequence of LDAP calls you will make in an application will follow this pattern:

```
ldap_connect()    // establish connection to server
  |
ldap_bind()       // anonymous or authenticated "login"
  |
do something like search or update the directory
and display the results
  |
ldap_close()      // "logout"
```

More Information

Lots of information about LDAP can be found at

- Netscape (http://developer.netscape.com/tech/directory/)
- University of Michigan (http://www.umich.edu/~dirsvcs/ldap/index.html)
- OpenLDAP Project (http://www.openldap.org/)
- LDAP World (http://www.innosoft.com/ldapworld)

The Netscape SDK contains a helpful Programmer’s Guide in .html format.

ldap_8859_to_t61 (PHP 4 >= 4.0.2)

Translate 8859 characters to t61 characters

string **ldap_8859_to_t61** (string value) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ldap_add (PHP 3, PHP 4)

Add entries to LDAP directory

bool **ldap_add** (resource link_identifier, string dn, array entry) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

The **ldap_add()** function is used to add entries in the LDAP directory. The DN of the entry to be added is specified by *dn*. Array *entry* specifies the information about the entry. The values in the entries are indexed by individual attributes. In case of multiple values for an attribute, they are indexed using integers starting with 0.

```
entry["attribute1"] = value
entry["attribute2"][0] = value1
entry["attribute2"][1] = value2
```

Példa 1. Complete example with authenticated bind

```
<?php
$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {
    // bind with appropriate dn to give update access
    $r=ldap_bind($ds,"cn=root, o=My Company, c=US", "secret");

    // prepare data
    $info["cn"]="John Jones";
    $info["sn"]="Jones";
    $info["mail"]="jonj@here.and.now";
    $info["objectclass"]="person";

    // add data to directory
    $r=ldap_add($ds, "cn=John Jones, o=My Company, c=US", $info);

    ldap_close($ds);
} else {
    echo "Unable to connect to LDAP server";
```

```
}
?>
```

ldap_bind (PHP 3, PHP 4)

Bind to LDAP directory

bool **ldap_bind** (resource link_identifier [, string bind_rdn [, string bind_password]]) \linebreak

Binds to the LDAP directory with specified RDN and password. Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

ldap_bind() does a bind operation on the directory. *bind_rdn* and *bind_password* are optional. If not specified, anonymous bind is attempted.

ldap_close (PHP 3, PHP 4)

Close link to LDAP server

bool **ldap_close** (resource link_identifier) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

ldap_close() closes the link to the LDAP server that's associated with the specified *link_identifier*.

This call is internally identical to ldap_unbind(). The LDAP API uses the call ldap_unbind(), so perhaps you should use this in preference to **ldap_close()**.

Megjegyzés: This function is an alias of ldap_unbind().

ldap_compare (PHP 4 >= 4.0.2)

Compare value of attribute found in entry specified with DN

bool **ldap_compare** (resource link_identifier, string dn, string attribute, string value) \linebreak

Returns TRUE if *value* matches otherwise returns FALSE. Returns -1 on error.

ldap_compare() is used to compare *value* of *attribute* to value of same attribute in LDAP directory entry specified with *dn*.

The following example demonstrates how to check whether or not given password matches the one defined in DN specified entry.

Példa 1. Complete example of password check

```
<?php

$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {

    // bind
    if(ldap_bind($ds)) {

        // prepare data
        $dn = "cn=Matti Meikku, ou=My Unit, o=My Company, c=FI";
        $value = "secretpassword";
        $attr = "password";

        // compare value
        $r=ldap_compare($ds, $dn, $attr, $value);

        if ($r === -1) {
            echo "Error: ".ldap_error($ds);
        } elseif ($r === TRUE) {
            echo "Password correct.";
        } elseif ($r === FALSE) {
            echo "Wrong guess! Password incorrect.";
        }

    } else {
        echo "Unable to bind to LDAP server.";
    }

    ldap_close($ds);

} else {
    echo "Unable to connect to LDAP server.";
}
?>
```

Figyelem

ldap_compare() can NOT be used to compare BINARY values!

Megjegyzés: This function was added in 4.0.2.

ldap_connect (PHP 3, PHP 4)

Connect to an LDAP server

resource **ldap_connect** ([string hostname [, int port]]) \linebreak

Returns a positive LDAP link identifier on success, or FALSE on error.

ldap_connect() establishes a connection to a LDAP server on a specified *hostname* and *port*. Both the arguments are optional. If no arguments are specified then the link identifier of the already opened link will be returned. If only *hostname* is specified, then the port defaults to 389.

If you are using OpenLDAP 2.x.x you can specify a URL instead of the hostname. To use LDAP with SSL, compile OpenLDAP 2.x.x with SSL support, configure PHP with SSL, and use ldaps://hostname/ as host parameter. The port parameter is not used when using URLs.

Megjegyzés: URL and SSL support were added in 4.0.4.

ldap_count_entries (PHP 3, PHP 4)

Count the number of entries in a search

int **ldap_count_entries** (resource link_identifier, resource result_identifier) \linebreak

Returns number of entries in the result or FALSE on error.

ldap_count_entries() returns the number of entries stored in the result of previous search operations. *result_identifier* identifies the internal ldap result.

ldap_delete (PHP 3, PHP 4)

Delete an entry from a directory

bool **ldap_delete** (resource link_identifier, string dn) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

ldap_delete() function delete a particular entry in LDAP directory specified by *dn*.

ldap_dn2ufn (PHP 3, PHP 4)

Convert DN to User Friendly Naming format

string **ldap_dn2ufn** (string dn) \linebreak

ldap_dn2ufn() function is used to turn a DN, specified by *dn*, into a more user-friendly form, stripping off type names.

ldap_err2str (PHP 3>= 3.0.13, PHP 4)

Convert LDAP error number into string error message

string **ldap_err2str** (int errno) \linebreak

Returns string error message.

This function returns the string error message explaining the error number *errno*. While LDAP errno numbers are standardized, different libraries return different or even localized textual error messages. Never check for a specific error message text, but always use an error number to check.

See also ldap_errno() and ldap_error().

Példa 1. Enumerating all LDAP error messages

```
<?php
  for($i=0; $i<100; $i++) {
    printf("Error $i: %s<br>\n", ldap_err2str($i));
  }
?>
```

ldap_errno (PHP 3>= 3.0.12, PHP 4)

Return the LDAP error number of the last LDAP command

int **ldap_errno** (resource link_identifier) \linebreak

Return the LDAP error number of the last LDAP command for this link.

This function returns the standardized error number returned by the last LDAP command for the given *link_identifier*. This number can be converted into a textual error message using ldap_err2str().

Unless you lower your warning level in your `php.ini` sufficiently or prefix your LDAP commands with @ (at) characters to suppress warning output, the errors generated will also show up in your HTML output.

Példa 1. Generating and catching an error

```
<?php
// This example contains an error, which we will catch.
$ld = ldap_connect("localhost");
$bind = ldap_bind($ld);
// syntax error in filter expression (errno 87),
// must be "objectclass=*" to work.
$res =  @ldap_search($ld, "o=Myorg, c=DE", "objectclass");
if (!$res) {
    printf("LDAP-Errno: %s<br>\n", ldap_errno($ld));
    printf("LDAP-Error: %s<br>\n", ldap_error($ld));
```

```
    die("Argh!<br>\n");
}
$info = ldap_get_entries($ld, $res);
printf("%d matching entries.<br>\n", $info["count"]);
?>
```

See also ldap_err2str() and ldap_error().

ldap_error (PHP 3>= 3.0.12, PHP 4)

Return the LDAP error message of the last LDAP command

string **ldap_error** (resource link_identifier) \linebreak

Returns string error message.

This function returns the string error message explaining the error generated by the last LDAP command for the given *link_identifier* While LDAP errno numbers are standardized, different libraries return different or even localized textual error messages. Never check for a specific error message text, but always use an error number to check.

Unless you lower your warning level in your `php.ini` sufficiently or prefix your LDAP commands with `@` (at) characters to suppress warning output, the errors generated will also show up in your HTML output.

See also ldap_err2str() and ldap_errno().

ldap_explode_dn (PHP 3, PHP 4)

Splits DN into its component parts

array **ldap_explode_dn** (string dn, int with_attrib) \linebreak

ldap_explode_dn() function is used to split the DN returned by ldap_get_dn() and breaks it up into its component parts. Each part is known as Relative Distinguished Name, or RDN. **ldap_explode_dn()** returns an array of all those components. *with_attrib* is used to request if the RDNs are returned with only values or their attributes as well. To get RDNs with the attributes (i.e. in attribute=value format) set *with_attrib* to 0 and to get only values set it to 1.

ldap_first_attribute (PHP 3, PHP 4)

Return first attribute

string **ldap_first_attribute** (resource link_identifier, resource result_entry_identifier, int ber_identifier) \linebreak

Returns the first attribute in the entry on success and FALSE on error.

Similar to reading entries, attributes are also read one by one from a particular entry. **ldap_first_attribute()** returns the first attribute in the entry pointed by the *result_entry_identifier*. Remaining attributes are retrieved by calling ldap_next_attribute() successively. *ber_identifier* is the identifier to internal memory location pointer. It is passed by reference. The same *ber_identifier* is passed to the ldap_next_attribute() function, which modifies that pointer.

See also ldap_get_attributes()

ldap_first_entry (PHP 3, PHP 4)

Return first result id

resource **ldap_first_entry** (resource link_identifier, resource result_identifier) \linebreak

Returns the result entry identifier for the first entry on success and FALSE on error.

Entries in the LDAP result are read sequentially using the **ldap_first_entry()** and ldap_next_entry() functions. **ldap_first_entry()** returns the entry identifier for first entry in the result. This entry identifier is then supplied to **lap_next_entry()** routine to get successive entries from the result.

See also ldap_get_entries().

ldap_first_reference (PHP 4 >= 4.0.5)

Return first reference

resource **ldap_first_reference** (resource link, resource result) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ldap_free_result (PHP 3, PHP 4)

Free result memory

bool **ldap_free_result** (resource result_identifier) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

ldap_free_result() frees up the memory allocated internally to store the result and pointed by the *result_identifier*. All result memory will be automatically freed when the script terminates.

Typically all the memory allocated for the ldap result gets freed at the end of the script. In case the script is making successive searches which return large result sets, **ldap_free_result()** could be called to keep the runtime memory usage by the script low.

ldap_get_attributes (PHP 3, PHP 4)

Get attributes from a search result entry

array **ldap_get_attributes** (resource link_identifier, resource result_entry_identifier) \linebreak

Returns a complete entry information in a multi-dimensional array on success and FALSE on error.

ldap_get_attributes() function is used to simplify reading the attributes and values from an entry in the search result. The return value is a multi-dimensional array of attributes and values.

Having located a specific entry in the directory, you can find out what information is held for that entry by using this call. You would use this call for an application which "browses" directory entries and/or where you do not know the structure of the directory entries. In many applications you will be searching for a specific attribute such as an email address or a surname, and won’t care what other data is held.

return_value["count"] = number of attributes in the entry
return_value[0] = first attribute
return_value[n] = nth attribute

return_value["attribute"]["count"] = number of values for attribute
return_value["attribute"][0] = first value of the attribute
return_value["attribute"][i] = (i+1)th value of the attribute

Példa 1. Show the list of attributes held for a particular directory entry

```
// $ds is the link identifier for the directory

// $sr is a valid search result from a prior call to
// one of the ldap directory search calls

$entry = ldap_first_entry($ds, $sr);

$attrs = ldap_get_attributes($ds, $entry);

echo $attrs["count"]." attributes held for this entry:<p>";

for ($i=0; $i<$attrs["count"]; $i++)
    echo $attrs[$i]."<br>";
```

See also ldap_first_attribute() and ldap_next_attribute()

ldap_get_dn (PHP 3, PHP 4)

Get the DN of a result entry

string **ldap_get_dn** (resource link_identifier, resource result_entry_identifier) \linebreak

Returns the DN of the result entry and FALSE on error.

ldap_get_dn() function is used to find out the DN of an entry in the result.

ldap_get_entries (PHP 3, PHP 4)

Get all result entries

array **ldap_get_entries** (resource link_identifier, resource result_identifier) \linebreak

Returns a complete result information in a multi-dimenasional array on success and FALSE on error.

ldap_get_entries() function is used to simplify reading multiple entries from the result, specified with *result_identifier*, and then reading the attributes and multiple values. The entire information is returned by one function call in a multi-dimensional array. The structure of the array is as follows.

The attribute index is converted to lowercase. (Attributes are case-insensitive for directory servers, but not when used as array indices.)

return_value["count"] = number of entries in the result
return_value[0] : refers to the details of first entry

return_value[i]["dn"] = DN of the ith entry in the result

return_value[i]["count"] = number of attributes in ith entry
return_value[i][j] = jth attribute in the ith entry in the result

return_value[i]["attribute"]["count"] = number of values for
attribute in ith entry
return_value[i]["attribute"][j] = jth value of attribute in ith entry

See also ldap_first_entry() and ldap_next_entry()

ldap_get_option (PHP 4 >= 4.0.4)

Get the current value for given option

bool **ldap_get_option** (resource link_identifier, int option, mixed retval) \linebreak

Sets *retval* to the value of the specified option. Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN. These are described in draft-ietf-ldapext-ldap-c-api-xx.txt (http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt)

> **Megjegyzés:** This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4

Példa 1. Check protocol version

```
// $ds is a valid link identifier for a directory server
if (ldap_get_option($ds, LDAP_OPT_PROTOCOL_VERSION, $version))
    echo "Using protocol version $version";
else
    echo "Unable to determine protocol version";
```

See also ldap_set_option().

ldap_get_values_len (PHP 3>= 3.0.13, PHP 4)

Get all binary values from a result entry

array **ldap_get_values_len** (resource link_identifier, resource result_entry_identifier, string attribute) \line-break

Returns an array of values for the attribute on success and FALSE on error.

ldap_get_values_len() function is used to read all the values of the attribute in the entry in the result. entry is specified by the *result_entry_identifier*. The number of values can be found by indexing "count" in the resultant array. Individual values are accessed by integer index in the array. The first index is 0.

This function is used exactly like ldap_get_values() except that it handles binary data and not string data.

> **Megjegyzés:** This function was added in 4.0.

ldap_get_values (PHP 3, PHP 4)

Get all values from a result entry

array **ldap_get_values** (resource link_identifier, resource result_entry_identifier, string attribute) \linebreak

Returns an array of values for the attribute on success and FALSE on error.

ldap_get_values() function is used to read all the values of the attribute in the entry in the result. entry is specified by the *result_entry_identifier*. The number of values can be found by indexing "count" in the resultant array. Individual values are accessed by integer index in the array. The first index is 0.

This call needs a *result_entry_identifier*, so needs to be preceded by one of the ldap search calls and one of the calls to get an individual entry.

You application will either be hard coded to look for certain attributes (such as "surname" or "mail") or you will have to use the ldap_get_attributes() call to work out what attributes exist for a given entry.

LDAP allows more than one entry for an attribute, so it can, for example, store a number of email addresses for one person’s directory entry all labeled with the attribute "mail"

return_value["count"] = number of values for attribute
return_value[0] = first value of attribute
return_value[i] = ith value of attribute

Példa 1. List all values of the "mail" attribute for a directory entry

```
// $ds is a valid link identifier for a directory server

// $sr is a valid search result from a prior call to
//     one of the ldap directory search calls

// $entry is a valid entry identifier from a prior call to
//        one of the calls that returns a directory entry

$values = ldap_get_values($ds, $entry,"mail");

echo $values["count"]." email addresses for this entry.<p>";

for ($i=0; $i < $values["count"]; $i++)
    echo $values[$i]."<br>";
```

ldap_list (PHP 3, PHP 4)

Single-level search

resource **ldap_list** (resource link_identifier, string base_dn, string filter [, array attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]]) \linebreak

Returns a search result identifier or FALSE on error.

ldap_list() performs the search for a specified *filter* on the directory with the scope LDAP_SCOPE_ONELEVEL.

LDAP_SCOPE_ONELEVEL means that the search should only return information that is at the level immediately below the *base_dn* given in the call. (Equivalent to typing "ls" and getting a list of files and folders in the current working directory.)

This call takes 5 optional parameters. See ldap_search() notes.

> **Megjegyzés:** These optional parameters were added in 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

Példa 1. Produce a list of all organizational units of an organization

```
// $ds is a valid link identifier for a directory server

$basedn = "o=My Company, c=US";
$justthese = array("ou");

$sr=ldap_list($ds, $basedn, "ou=*", $justthese);

$info = ldap_get_entries($ds, $sr);

for ($i=0; $i<$info["count"]; $i++)
    echo $info[$i]["ou"][0] ;
```

> **Megjegyzés:** From 4.0.5 on it's also possible to do parallel searches. See ldap_search() for details.

ldap_mod_add (PHP 3>= 3.0.8, PHP 4)

Add attribute values to current attributes

bool **ldap_mod_add** (resource link_identifier, string dn, array entry) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

This function adds attribute(s) to the specified *dn*. It performs the modification at the attribute level as opposed to the object level. Object-level additions are done by the ldap_add() function.

ldap_mod_del (PHP 3>= 3.0.8, PHP 4)

Delete attribute values from current attributes

bool **ldap_mod_del** (resource link_identifier, string dn, array entry) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

This function removes attribute(s) from the specified *dn*. It performs the modification at the attribute level as opposed to the object level. Object-level deletions are done by the ldap_delete() function.

ldap_mod_replace (PHP 3>= 3.0.8, PHP 4)

Replace attribute values with new ones

bool **ldap_mod_replace** (resoure link_identifier, string dn, array entry) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

This function replaces attribute(s) from the specified *dn*. It performs the modification at the attribute level as opposed to the object level. Object-level modifications are done by the ldap_modify() function.

ldap_modify (PHP 3, PHP 4)

Modify an LDAP entry

bool **ldap_modify** (resource link_identifier, string dn, array entry) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

ldap_modify() function is used to modify the existing entries in the LDAP directory. The structure of the entry is same as in ldap_add().

ldap_next_attribute (PHP 3, PHP 4)

Get the next attribute in result

string **ldap_next_attribute** (resource link_identifier, resource result_entry_identifier, resource ber_identifier) \linebreak

Returns the next attribute in an entry on success and FALSE on error.

ldap_next_attribute() is called to retrieve the attributes in an entry. The internal state of the pointer is maintained by the *ber_identifier*. It is passed by reference to the function. The first call to **ldap_next_attribute()** is made with the *result_entry_identifier* returned from ldap_first_attribute().

See also ldap_get_attributes()

ldap_next_entry (PHP 3, PHP 4)

Get next result entry

resource **ldap_next_entry** (resource link_identifier, resource result_entry_identifier) \linebreak

Returns entry identifier for the next entry in the result whose entries are being read starting with ldap_first_entry(). If there are no more entries in the result then it returns `FALSE`.

ldap_next_entry() function is used to retrieve the entries stored in the result. Successive calls to the **ldap_next_entry()** return entries one by one till there are no more entries. The first call to **ldap_next_entry()** is made after the call to ldap_first_entry() with the *`result_entry_identifier`* as returned from the ldap_first_entry().

See also ldap_get_entries()

ldap_next_reference (PHP 4 >= 4.0.5)

Get next reference

resource **ldap_next_reference** (resource link, resource entry) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ldap_parse_reference (PHP 4 >= 4.0.5)

Extract information from reference entry

bool **ldap_parse_reference** (resource link, resource entry, array referrals) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ldap_parse_result (PHP 4 >= 4.0.5)

Extract information from result

bool **ldap_parse_result** (resource link, resource result, int errcode, string matcheddn, string errmsg, array referrals) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ldap_read (PHP 3, PHP 4)

Read an entry

resource **ldap_read** (resource link_identifier, string base_dn, string filter [, array attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]]) \linebreak

Returns a search result identifier or FALSE on error.

ldap_read() performs the search for a specified *`filter`* on the directory with the scope LDAP_SCOPE_BASE. So it is equivalent to reading an entry from the directory.

An empty filter is not allowed. If you want to retrieve absolutely all information for this entry, use a filter of "objectClass=*". If you know which entry types are used on the directory server, you might use an appropriate filter such as "objectClass=inetOrgPerson".

This call takes 5 optional parameters. See ldap_search() notes.

> **Megjegyzés:** These optional parameters were added in 4.0.2: *`attrsonly`*, *`sizelimit`*, *`timelimit`*, *`deref`*.

From 4.0.5 on it’s also possible to do parallel searches. See ldap_search() for details.

ldap_rename (PHP 4 >= 4.0.5)

Modify the name of an entry

bool **ldap_rename** (resource link_identifier, string dn, string newrdn, string newparent, bool deleteoldrdn) \linebreak

The entry specified by *`dn`* is renamed/moved. The new RDN is specified by *`newrdn`* and the new parent/superior entry is specified by *`newparent`*. If the parameter *`deleteoldrdn`* is TRUE the old RDN value(s) is removed, else the old RDN value(s) is retained as non-distinguished values of the entry. Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

> **Megjegyzés:** This function currently only works with LDAPv3. You may have to use ldap_set_option() prior to binding to use LDAPv3. This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.5.

ldap_search (PHP 3, PHP 4)

Search LDAP tree

resource **ldap_search** (resource link_identifier, string base_dn, string filter [, array attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]]) \linebreak

Returns a search result identifier or FALSE on error.

ldap_search() performs the search for a specified filter on the directory with the scope of LDAP_SCOPE_SUBTREE. This is equivalent to searching the entire directory. *base_dn* specifies the base DN for the directory.

There is a optional fourth parameter, that can be added to restrict the attributes and values returned by the server to just those required. This is much more efficient than the default action (which is to return all attributes and their associated values). The use of the fourth parameter should therefore be considered good practice.

The fourth parameter is a standard PHP string array of the required attributes, eg array("mail","sn","cn") Note that the "dn" is always returned irrespective of which attributes types are requested.

Note too that some directory server hosts will be configured to return no more than a preset number of entries. If this occurs, the server will indicate that it has only returned a partial results set. This occurs also if the sixth parameter *sizelimit* has been used to limit the count of fetched entries.

The fifth parameter *attrsonly* should be set to 1 if only attribute types are wanted. If set to 0 both attributes types and attribute values are fetched which is the default behaviour.

With the sixth parameter *sizelimit* it is possible to limit the count of entries fetched. Setting this to 0 means no limit. NOTE: This parameter can NOT override server-side preset sizelimit. You can set it lower though.

The seventh parameter *timelimit* sets the number of seconds how long is spend on the search. Setting this to 0 means no limit. NOTE: This parameter can NOT override server-side preset timelimit. You can set it lower though.

The eigth parameter *deref* specifies how aliases should be handled during the search. It can be one of the following:

- LDAP_DEREF_NEVER - (default) aliases are never dereferenced.
- LDAP_DEREF_SEARCHING - aliases should be dereferenced during the search but not when locating the base object of the search.
- LDAP_DEREF_FINDING - aliases should be dereferenced when locating the base object but not during the search.
- LDAP_DEREF_ALWAYS - aliases should be dereferenced always.

> **Megjegyzés:** These optional parameters were added in 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

The search filter can be simple or advanced, using boolean operators in the format described in the LDAP doumentation (see the Netscape Directory SDK

(http://developer.netscape.com/docs/manuals/directory/41/ag/find.htm) for full information on filters).

The example below retrieves the organizational unit, surname, given name and email address for all people in "My Company" where the surname or given name contains the substring $person. This example uses a boolean filter to tell the server to look for information in more than one attribute.

Példa 1. LDAP search

```
// $ds is a valid link identifier for a directory server

// $person is all or part of a person’s name, eg "Jo"

$dn = "o=My Company, c=US";
$filter="(|(sn=$person*)(givenname=$person*))";
$justthese = array( "ou", "sn", "givenname", "mail");

$sr=ldap_search($ds, $dn, $filter, $justthese);

$info = ldap_get_entries($ds, $sr);

print $info["count"]." entries returned<p>";
```

From 4.0.5 on it’s also possible to do parallel searches. To do this you use an array of link identifiers, rather than a single identifier, as the first argument. If you don’t want the same base DN and the same filter for all the searches, you can also use an array of base DNs and/or an array of filters. Those arrays must be of the same size as the link identifier array since the first entries of the arrays are used for one search, the second entries are used for another, and so on. When doing parallel searches an array of search result identifiers is returned, except in case of error, then the entry corresponding to the search will be `FALSE`. This is very much like the value normally returned, except that a result identifier is always returned when a search was made. There are some rare cases where the normal search returns `FALSE` while the parallel search returns an identifier.

ldap_set_option (PHP 4 >= 4.0.4)

Set the value of the given option

bool **ldap_set_option** (resource link_identifier, int option, mixed newval) \linebreak

Sets the value of the specified option to be *`newval`*. Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad. on error.

The parameter *`option`* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN, LDAP_OPT_SERVER_CONTROLS, LDAP_OPT_CLIENT_CONTROLS. Here’s a brief description, see draft-ietf-ldapext-ldap-c-api-xx.txt

(http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt) for
details.

The options LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION and LDAP_OPT_ERROR_NUMBER have integer value, LDAP_OPT_REFERRALS and LDAP_OPT_RESTART have boolean value, and the options LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING and LDAP_OPT_MATCHED_DN have string value. The first example illustrates their use. The options LDAP_OPT_SERVER_CONTROLS and LDAP_OPT_CLIENT_CONTROLS require a list of controls, this means that the value must be an array of controls. A control consists of an *oid* identifying the control, an optional *value*, and an optional flag for *criticality*. In PHP a control is given by an array containing an element with the key *oid* and string value, and two optional elements. The optional elements are key *value* with string value and key *iscritical* with boolean value. *iscritical* defaults to `FALSE` if not supplied. See also the second example below.

> **Megjegyzés:** This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4.

Példa 1. Set protocol version

```
// $ds is a valid link identifier for a directory server
if (ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3))
    echo "Using LDAPv3";
else
    echo "Failed to set protocol version to 3";
```

Példa 2. Set server controls

```
// $ds is a valid link identifier for a directory server
// control with no value
$ctrl1 = array("oid" => "1.2.752.58.10.1", "iscritical" => TRUE);
// iscritical defaults to FALSE
$ctrl2 = array("oid" => "1.2.752.58.1.10", "value" => "magic");
// try to set both controls
if (!ldap_set_option($ds, LDAP_OPT_SERVER_CONTROLS, array($ctrl1, $ctrl2)))
    echo "Failed to set server controls";
```

See also ldap_get_option().

ldap_set_rebind_proc (PHP 4 >= 4.2.0)

Set a callback function to do re-binds on referral chasing.

bool **ldap_set_rebind_proc** (resource link, string callback) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ldap_sort (PHP 4 >= 4.2.0)

Sort LDAP result entries

bool **ldap_sort** (resource link, resource result, string sortfilter) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ldap_start_tls (PHP 4 >= 4.2.0)

Start TLS

bool **ldap_start_tls** (resource link) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ldap_t61_to_8859 (PHP 4 >= 4.0.2)

Translate t61 characters to 8859 characters

string **ldap_t61_to_8859** (string value) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ldap_unbind (PHP 3, PHP 4)

Unbind from LDAP directory

bool **ldap_unbind** (resource link_identifier) \linebreak

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

ldap_unbind() function unbinds from the LDAP directory.

L. Mail függvények

A mail() függvény ad lehetőséget email küldésére.

Levelezés beállítási lehetőségei

SMTP string

A DNS neve vagy IP címe annak az SMTP szervernek, amit a PHP Windows alatt a mail küldésre használ, amikor a mail() függvényt hívod meg.

sendmail_from string

Megadja, hogy milyen "From:" email címet használjon a PHP mail küldésekor Windows alatt.

sendmail_path string

Megadja, hogy hol található a **sendmail** program. Ez általában `/usr/sbin/sendmail` vagy `/usr/lib/sendmail`. A **configure** megpróbálja megtalálni, és beállítani, de ha ez nem sikerül, itt te is beállíthatod.

Azokon a rendszereken, ahol nem a sendmail használatos, ez a beállítás a sendmail wrapper/helyettesítő beállítására szolgál, ha van ilyen. Például a Qmail (http://www.qmail.org/) használók általában a `/var/qmail/bin/sendmail` beállítást használják.

ezmlm_hash (PHP 3>= 3.0.17, PHP 4 >= 4.0.2)

Kiszámítja a hash értéket, amit az EZMLM vár

int **ezmlm_hash** (string addr) \linebreak

Az **ezmlm_hash()** kiszámítja a szükséges hash értékeket, amik az EZMLM levelezőlisták MySQL adatbázisban való tárolásához szükségesek.

Példa 1. A hash kiszámítása és a felhasználó felírása a listára

```
$felhasznalo = "joecool@example.com";
$hash = ezmlm_hash ($felhasznalo);
$query = sprintf ("INSERT INTO minta VALUES (%s, '%s')", $hash, $felhasznalo);
$db->query($query); // itt éppen a PHPLIB db felületet használjuk
```

mail (PHP 3, PHP 4)

Levelet küld

bool **mail** (string to, string subject, string message [, string additional_headers]) \linebreak

A **mail()** automatikusan a *`message`* paraméterben megadott levelet a *`to`* paraméterben megadott címzettnek. A levél témamegjelölését a *`subject`* határozza meg. Több címzettet is megadhatsz, ha vesszőt teszel a címzettek listájában az email címek közé a *`to`* paraméterében. Levélmellékletek és egyéb különleges tartalmú levelek küldésére is használható ez a függvény. Ezeket a MIME kódolással érhetők el, amelyről részletesebb leírást egy Zend cikkben (http://www.zend.com/zend/spotlight/sendmimeemailpart1.php) vagy a PEAR Mime Classes (http://pear.php.net/Mail_Mime) c. található.

A következő RFC-k is számos segítséget nyújthatnak: RFC 1896 (http://www.ietf.org/rfc/rfc1896.txt), RFC 2045 (http://www.ietf.org/rfc/rfc2045.txt), RFC 2046 (http://www.ietf.org/rfc/rfc2046.txt), RFC 2047 (http://www.ietf.org/rfc/rfc2047.txt), RFC 2048 (http://www.ietf.org/rfc/rfc2048.txt) és RFC 2049 (http://www.ietf.org/rfc/rfc2049.txt).

A **mail()** `TRUE`-val tér vissza, ha levelet sikeresen elküldte máskülönben `FALSE`-szal.

Példa 1. Email küldése

```
mail("joecool@example.com", "Témamegjelölés", "Első sor\nMásodik sor\nHarmadik sor");
```

A negyedik paraméter megadott szöveg az email fejléc végére kerül. Ez nagyon gyakran újabb fejlécek elküldésére használható, több ilyen esetén azokat az újsorkarakterrel kell elválasztani.

Megjegyzés: A fejlécek elválasztására a `\r\n` karaktersorozatot kell használni még akkor is, ha néhány Unix levéltovábbító elem az egyszerű újsorkarakterrel (`\n`) is beéri. A Cc: fejléc kis- és nagy betűre érzékeny, így `Cc:` kell írni Win32 rendszereken. A Bcc: fejlécet nem támogatják a Win32 rendszerek.

Példa 2. e-mail küldése kiegészítő fejlécekkel

```
mail("nobody@example.com", "Téma", $uzenet,
     "From: webmaster@$SERVER_NAME\r\n".
     "Reply-To: webmaster@$SERVER_NAME\r\n".
     "X-Mailer: PHP/" . phpversion());
```

Az *`additional_parameters`* paraméter használható arra, hogy a levélküldőként beállított program számára további paramétereket lehessen átadni, lásd: `sendmail_path`). Ezzel például a levél mellé a feladó 'borítékot' is küldhet a sendmail-t használva. Szükség lehet a webszervert futtató felhasználót sendmail beállításai közé felvenni, hogy a levelekbe ne kerüljenek `X-Warning` fejlécek, ha ezzel a módszerrel küldesz 'borítékot' a levél mellé.

Példa 3. Kiegészítő fejléces levél küldése és további parancssori paraméterek átadása

```
mail("nobody@example.com", "a tárgy", $uzenet,
     "From: webmaster@$SERVER_NAME", "-fwebmaster@$SERVER_NAME");
```

Megjegyzés: Az ötödik paraméter PHP 4.0.5-től elérhető.

Egyszerű sztring kezelő módszerekkel lehet összetettebb e-mail üzenetek összeállításához:

Példa 4. Komplexebb email küldése

```
/* címzettek */
$kinek  = "Mari <mari@example.com>" . ", " ; // figyelj a vesszőre!
$kinek .= "Kolos <kolos@example.com>";

/* tárgy */
$targy = "Augusztusi Születésnapi Emlékeztető";

/* üzenet */
$uzenet = '
<html>
<head>
 <title>Augusztusi Születésnapi Emlékeztető</title>
```

```
</head>
<body>
<p>Itt van az augusztusi születésnapok listája!</p>
<table>
 <tr>
  <th>Személy</th><th>Év</th><th>Hónap</th><th>Nap</th>
 </tr>
 <tr>
  <td>Jocó</td><td>1970</td><td>augusztus</td><td>3.</td>
  <td>Saci</td><td>1973</td><td>augusztus</td><td>17.</td>
 </tr>
</table>
</body>
</html>
';

/* HTML levél küldése a Content-type fejléc megadásával */
$fejlec  = "MIME-Version: 1.0\r\n";
$fejlec .= "Content-type: text/html; charset=iso-8859-2\r\n";

/* további fejlécek */
$fejlec .= "From: Szuletesnapi Emlekezteto <szulinap@example.com>\r\n";

$fejlec .= "Cc: szulinaptar@example.com\r\n";
$fejlec .= "Bcc: szulinapell@example.com\r\n";

/* és most küldjük el! */
mail($kinek, $targy, $uzenet, $fejlec);
```

Megjegyzés: Biztosítani kell, hogy a ne legyen egyetlen újsorkarakter sem a *`to`* vagy a *`subject`* fejlécben, mert különben nem megfelelően lesz kiküldve a levél.

LI. mailparse functions

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

mailparse_determine_best_xfer_encoding (4.1.0 - 4.1.2 only)

Figures out the best way of encoding the content read from the file pointer fp, which must be seek-able

int **mailparse_determine_best_xfer_encoding** (resource fp) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_msg_create (4.1.0 - 4.1.2 only)

Returns a handle that can be used to parse a message

int **mailparse_msg_create** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_msg_extract_part_file (4.1.0 - 4.1.2 only)

Extracts/decodes a message section, decoding the transfer encoding

string **mailparse_msg_extract_part_file** (resource rfc2045, string filename [, string callbackfunc]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_msg_extract_part (4.1.0 - 4.1.2 only)

Extracts/decodes a message section. If callbackfunc is not specified, the contents will be sent to "stdout"

void **mailparse_msg_extract_part** (resource rfc2045, string msgbody [, string callbackfunc]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_msg_free (4.1.0 - 4.1.2 only)

Frees a handle allocated by mailparse_msg_crea

void **mailparse_msg_free** (resource rfc2045buf) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_msg_get_part_data (4.1.0 - 4.1.2 only)

Returns an associative array of info about the message

array **mailparse_msg_get_part_data** (resource rfc2045) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_msg_get_part (4.1.0 - 4.1.2 only)

Returns a handle on a given section in a mimemessage

int **mailparse_msg_get_part** (resource rfc2045, string mimesection) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_msg_get_structure (4.1.0 - 4.1.2 only)

Returns an array of mime section names in the supplied message

array **mailparse_msg_get_structure** (resource rfc2045) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_msg_parse_file (4.1.0 - 4.1.2 only)

Parse file and return a resource representing the structure

resource **mailparse_msg_parse_file** (string filename) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_msg_parse (4.1.0 - 4.1.2 only)

Incrementally parse data into buffer

void **mailparse_msg_parse** (resource rfc2045buf, string data) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_rfc822_parse_addresses (4.1.0 - 4.1.2 only)

Parse addresses and returns a hash containing that data

array **mailparse_rfc822_parse_addresses** (string addresses) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_stream_encode (4.1.0 - 4.1.2 only)

Streams data from source file pointer, apply encoding and write to destfp

bool **mailparse_stream_encode** (resource sourcefp, resource destfp, string encoding) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mailparse_uudecode_all (unknown)

Scans the data from fp and extract each embedded uuencoded file. Returns an array listing filename information

array **mailparse_uudecode_all** (resource fp) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

LII. Matematikai függvények

Előszó

Ezek a matematikai függvények csak a gépeden ábrázolható integer és float típusok értéktartományába eső értékeket képes kezelni (ez jelenleg azonos a C beli long és double típusokéval). Ha nagyobb számokat kell kezelni, akkor a nézd meg a BCMath tetszőleges pontosságú matematikai függvényeket..

Matematikai állandók

Ez a matematikai kiterjesztés a következő értékeket hozza létre a PHP-ban:

Táblázat 1. Matematikai állandók

Állandó	Érték	Leírás
M_PI	3.14159265358979323846	Pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [4.0.2]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT3	1.73205080756887729352	sqrt(3) [4.0.2]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [4.0.2]
M_EULER	0.57721566490153286061	Euler constant [4.0.2]

PHP 4.0.0 verziójáig csak az M_PI használható. Minden más állandó PHP 4.0.0-tól kezdve elérhető, kivéve amelyiknél a [4.0.2] címke olvasható: ezek csak PHP 4.0.2-tól kezdve.

abs (PHP 3, PHP 4)

abszolút érték

mixed **abs** (mixed number) \linebreak

A szám abszolút értékével tér vissza. Ha paraméterben átadott szám float típusú, akkor a visszaadott érték is float típusú lesz, egyébként integer. (mivel egy float-nak általában nagyobb az értéke, mint egy integer-nek).

Példa 1. abs() példa

```
$abs  = abs(-4.2); // $abs  = 4.2; (double/float)
$abs2 = abs(5);    // $abs2 = 5; (integer)
$abs3 = abs(-5);   // $abs3 = 5; (integer)
```

acos (PHP 3, PHP 4)

arkusz koszinusz

float **acos** (float arg) \linebreak

Az *arg* arkusz koszinuszát adja vissza radiánban.

Lásd még: acosh(), asin() és atan()!

acosh (PHP 4 >= 4.1.0)

area hiperbolikus koszinusz

float **acosh** (float arg) \linebreak

Az *arg* area hiperbolikus koszinuszát adja vissza, azt az értéket, amelynek hiperbolikus koszinusza egyenlő az *arg* paraméterrel.

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

Lásd még: acos(), asin() és atan()!

asin (PHP 3, PHP 4)

arkusz szinusz

float **asin** (float arg) \linebreak

Az `arg` arkusz szinuszát adja vissza radiánban.

Lásd még: acos() és atan()!

asinh (PHP 4 >= 4.1.0)

area hiperbolikus szinusz

float **asinh** (float arg) \linebreak

Az `arg` area hiperbolikus szinuszával tér vissza, azzal az értékkel, amelynek hiperbolikus szinusza egyenlő az `arg` paraméterrel.

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

Lásd még: asin(), acos() és atan()!

atan2 (PHP 3>= 3.0.5, PHP 4)

két változó arkusz tangense

float **atan2** (float y, float x) \linebreak

Ez a függvény az `x` és `y` által meghatározott érték arkusz tangensét adja vissza radiánban. Ez hasonló az `y` / `x` arkusz tangenséhez, attól eltekintve hogy a paraméterek előjele meghatározza, hogy az eredmény melyik körnegyedbe esik.

A függvény eredményét radiánban adja vissza, ami -PI és +PI közé esik (beleértve a határokat is).

Lásd még: acos() és atan()!

atan (PHP 3, PHP 4)

arkusz tangens

float **atan** (float arg) \linebreak

A `arg` arkusz tangensét adja vissza radiánban.

Lásd még: asin() és acos()!

atanh (PHP 4 >= 4.1.0)

area hiperbolikus tangens

float **atanh** (float arg) \linebreak

Az *arg* area hiperbolikus tangensével tér vissza, azzal az értékkel, amelynek hiiperbolikus tangense egyenlő az *arg* paraméterrel.

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

Lásd még: atan(), asin() és acos()!

base_convert (PHP 3>= 3.0.6, PHP 4)

tetszőleges számrendszerbe vált át egy számot

string **base_convert** (string number, int frombase, int tobase) \linebreak

A *number*-ben átadott számot váltja át a *tobase*-ben megadott alapszámú számrendszerbe, és ezzel tér vissza. A *number* alapszáma a *frombase*-ben adható meg. Mind a *frombase* és *tobase* paraméternek 2 és 36 közé kell esnie. A 10-nél nagyobb értékű számjegyeket betűkkel ábrázoljuk, ahol az "a" 10-et jelent, a "b" 11-et és a "z" 35-öt.

Példa 1. base_convert()

```
$binaris = base_convert ($hexadecimalis, 16, 2);
```

bindec (PHP 3, PHP 4)

binárisból tizes számrendszerbe vált át

int **bindec** (string binary_string) \linebreak

A *binary_string* kettes számrendszerbeli (bináris) szám tizes számrendszerbeli (decimális) megfelelőjét adja vissza.

A **bindec()** függvény integer típusúvá konvertálja át a bináris számot. A legnagyobb konvertálható szám a 31 db 1 bitből álló szám, vagyis 2147483647 (10-es számrendszerben).

Lásd még: decbin()!

ceil (PHP 3, PHP 4)

törtrésszel bíró számot felfelé kerekíti

float **ceil** (float value) \linebreak

Ha szükséges, akkor a következő legnagyobb egész számmal tér vissza a *value*-t felfelé kerekítve. A **ceil()** által visszadott érték float típusú, mivel a float értékészlete általában nagyobb, mint int-é.

Példa 1. ceil() példák

```
$ot  = ceil(4.3);   // $ot  =  5.0;
$tiz = ceil(9.999); // $tiz = 10.0;
```

Lásd még: floor() és round()!

cos (PHP 3, PHP 4)

koszinusz

float **cos** (float arg) \linebreak

A radiában megadott *arg* koszinuszát adja vissza.

Lásd még: sin() és tan()!

cosh (PHP 4 >= 4.1.0)

hiperbolikus koszinusz

float **cosh** (float arg) \linebreak

Az *arg* hiperbolikus koszinuszát adja vissza, amelyet a `(exp(arg) + exp(-arg))/2` kifejezéssel határozható meg.

Lásd még: cos(), acosh(), sin() és tan()!

decbin (PHP 3, PHP 4)

tízes számrendszerből kettesbe vált át

string **decbin** (int number) \linebreak

A *number* kettes számrendszerbeli ábrázolását sztringként adja vissza. A legnagyobb átváltható szám 4294967295 (tízes számrendszerben), ami 31 db 1-esből álló sztringet eredményez.

Lásd még: bindec()!

dechex (PHP 3, PHP 4)

tízes számrendszerből tizenhatosba vált át

string **dechex** (int number) \linebreak

A *number* tizenhatos számrendszerbeli ábrázolását sztringként adja vissza. A legnagyobb átváltható szám 2147483647 (tízes számrendszerben), ami "7fffffff"-et ad eredményül.

Lásd még: hexdec()!

decoct (PHP 3, PHP 4)

tízes számrendszerből nyolcasba vált át

string **decoct** (int number) \linebreak

A *number* nyolcas számrendszerbeli ábrázolását sztringként adja vissza. A legnagyobb átváltható szám 2147483647 (tízes számrendszerben), ami "17777777777"-et ad eredményül.

Lásd még: octdec()!

deg2rad (PHP 3>= 3.0.4, PHP 4)

fokból radiánba vált át

float **deg2rad** (float number) \linebreak

Ez a függvény a *number* paraméterben megadott fokokat radiánba váltja át.

Lásd még: rad2deg()!

exp (PHP 3, PHP 4)

e a(z) ... -re emelve

float **exp** (float arg) \linebreak

Visszaadja, hogy mennyivel egyenlő az e az *arg*-ban megadott hatványon.

Lásd még: pow()!

expm1 (PHP 4 >= 4.1.0)

exp(number) - 1 pontos értéke, még ha a number 0-hoz közeli értékű is

float **expm1** (float number) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

floor (PHP 3, PHP 4)

lefelé kerekíti a törtrésszel bíró számot

float **floor** (float value) \linebreak

Ha szükséges, akkor a megelőző legkisebb egész számmal tér vissza a *value*-t lefelé kerekítve. A **floor()** által visszadott érték float típusú, mivel a float értékészlete általában nagyobb, mint int-é.

Példa 1. floor() példa

```
$negy   = floor(4.3);   // $negy   = 4.0;
$kilenc = floor(9.999); // $kilenc = 9.0;
```

Lásd még: ceil() és round()!

getrandmax (PHP 3, PHP 4)

a legnagyobb lehetséges véletlen számot adja vissza

int **getrandmax** (void) \linebreak

A legnagyobb lehetséges véletlen számot adja vissza, amit a rand() a meghívásakor visszaadhat.

Lásd még: rand(), srand(), mt_rand(), mt_srand() és mt_getrandmax()!

hexdec (PHP 3, PHP 4)

tizenhatos számrendszerből tízesbe vált át

int **hexdec** (string hex_string) \linebreak

A tizenhatos számrendszerbeli *`hex_string`* tízes számrendszerbeli megfelelőjével tér vissza. A legnagyobb átváltható szám a 7fffffff, azaz 2147483647 (tízes számrendszerben).

A **hexdec()** a *`hex_string`* nem hexadecimális karaktereit `0`-ra cseréli, és minden baloldali nullát figyelmen kívül hagy, de a jobboldaliak beszámítanak az eredménybe.

Példa 1. hexdec() példa

```
var_dump(hexdec("hee"));
var_dump(hexdec("ee"));
// mindkettő: int(238)

var_dump(hexdec("az"));
var_dump(hexdec("a0"));
// mindkettő: int(160)
```

Lásd még: dechex()!

hypot (PHP 4 >= 4.1.0)

a sqrt(num1*num1 + num2*num2) értékét adja vissza

float **hypot** (float num1, float num2) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

is_finite (PHP 4 >= 4.2.0)

bool **is_finite** (float val) \linebreak

`TRUE`-val tér vissza, ha a *`val`* olan érvényes véges számot, amely belefér a PHP-t futtató platform lebegőpontos ábrázolási tartományába.

is_infinite (PHP 4 >= 4.2.0)

bool **is_infinite** (float val) \linebreak

`TRUE`-val tér vissza, ha a *`val`* pozitív vagy negatív végtelen, mint pl. a `log(0)` értéke, vagy bármely más érték, amely *nem* belefér a PHP-t futtató platform lebegőpontos ábrázolási tartományába.

is_nan (PHP 4 >= 4.2.0)

bool **is_nan** (float val) \linebreak

`TRUE`-val tér vissza, ha *`val`* nem szám, mint pl. a `acos(1.01)` eredménye.

> **Megjegyzés:** Az előző példából láthatóan ez akkor fordulhat elő, ha például valamilyen matematikai műveletnek értelmezési tartományán kívüli értéket adunk át, pl. arkusza csak `-1` és `+1` közé eső számoknak lehet.

lcg_value (PHP 4)

Kombinált lineáris kongruencia generátor

float **lcg_value** (void) \linebreak

A **lcg_value()** ál véletlenszámot ad vissza a (0,1) intervallumból. A függvény két, 2^31 - 85 és 2^31 - 249 periódusú CG-t kombinál, és a periódusa ennek a két prímnek a szorzatával egyenlő.

log10 (PHP 3, PHP 4)

tízes alapú logaritmus

float **log10** (float arg) \linebreak

Az *`arg`* tízes alapú logaritmusát adja vissza.

log1p (PHP 4 >= 4.1.0)

log(1 + number) pontos értékét adja vissza, akkor is ha a paraméter értéke 0-hoz közeli

float **log1p** (float number) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

log (PHP 3, PHP 4)

természetes alapú logaritmus

float **log** (float arg) \linebreak

Az `arg` természetes (e) alapú logaritmusát adja vissza.

max (PHP 3, PHP 4)

megkeresi a legnagyobb értéket

mixed **max** (mixed arg1, mixed arg2, mixed argn) \linebreak

A **max()** a paraméterként átadott, számként értelmezett értékek legnagyobbjával tér vissza.

Ha az első paraméter tömb típusú, akkor a tömbbeli legnagyobb értéket adja vissza. Ha az első paraméter integer, float vagy string típusú, akkor legalább két paraméterrel kell meghívni a függvényt, és az a legnagyobb értékű számmal tér vissza. Korlátlan számú paraméterrel hívható meg ez a függvény.

Ha legalább egy összehasonlítandó érték float típusú, akkor mindegyik float-ként lesz kezelve és a visszatérési érték is float lesz. Ha egyik sem float, akkor mindegyik integer-ként lesz kezelve, és a visszatérési érték is integer lesz.

min (PHP 3, PHP 4)

megkeresi a legkisebb értéket

number **min** (number arg1, number arg2 [, ...]) \linebreak number **min** (array numbers) \linebreak

A **min()** a paraméterként átadott, számként értelmezett értékek legkisebbjével tér vissza.

Ha az első paraméter tömb típusú, akkor a tömbbeli legkisebb értéket adja vissza. Ha az első paraméter integer, float vagy string típusú, akkor legalább két paraméterrel kell meghívni a függvényt, és az a legkisebb értékű számmal tér vissza. Korlátlan számú paraméterrel hívható meg ez a függvény.

Ha legalább egy összehasonlítandó érték float típusú, akkor mindegyik float-ként lesz kezelve és a visszatérési érték is float lesz. Ha egyik sem float, akkor mindegyik integer-ként lesz kezelve, és a visszatérési érték is integer lesz.

mt_getrandmax (PHP 3>= 3.0.6, PHP 4)

visszaadja a lehetséges legnagyobb véletlenszámot

int **mt_getrandmax** (void) \linebreak

Visszaadja azt a legnagyobb véletlenszámot, amit az mt_rand() a meghívásakor visszadhat.

Lásd még: mt_rand(), mt_srand(), rand(), srand() és getrandmax()!

mt_rand (PHP 3>= 3.0.6, PHP 4)

véletlenszámot generál - jobban :)

int **mt_rand** ([int min, int max]) \linebreak

A régebbi libc-k véletlenszám generátorai kétes vagy bizonytalan jellegűek és lassúak voltak. Alapértelmezés szerint, a PHP a libc véletlenszám generátorát használja a rand() függvényben. A **mt_rand()** függvény egy "beugrós" helyettesítése ennek. Ez egy ismert jellemvonásokkal bíró véletlenszám generátor, a Mersenne Twister, ami többféle kriptogáfiai eljárás inicializálásához (seeding) is megfelelő véletlenszámokat generál kb. négyszer gyorsabban, mint az átlagos libc rand függvények. (Részletesebb információkért nézd meg a Mersenne Twister honlapját, amely http://www.math.keio.ac.jp/~matumoto/emt.html , az MT forrás optimalizált változata http://www.scp.syr.edu/~marc/hawk/twister.html pedig címen található.

Ha a *`min`* és *`max`* paraméterek nélkül hívod meg az **mt_rand()** függvényt, akkor az ál véletlenszám értéke 0 és `RAND_MAX` közé fog esni. Ha pl. 5 és 15 közé (bezárólag) eső véletlen számot akarsz kapni, akkor `mt_rand(5, 15)` kell írni.

Ne felejtsd el a véletlenszám generátor kiinduló értékét beállítani a mt_srand() függvénnyel.

> **Megjegyzés:** A 3.0.7 verziók előtt a *`max`* paraméter helyett *`range`* (tartomány) szerepelt. Ezekben a verziókban az előző példával egyenértékű megoldás a `mt_rand (5, 11)`, hogy 5 és 15 közé eső véletlen számot kapj.

Lásd még: mt_srand(), mt_getrandmax(), srand(), rand() és getrandmax()!

mt_srand (PHP 3>= 3.0.6, PHP 4)

beállítja a "jobbik" véletlenszám generátor kiinduló értékét

void **mt_srand** (int seed) \linebreak

Beállítja a véletlenszám generátor kiinduló értékét *`seed`*-re.

```
// beállítás ezredmásodpercekkel
function make_seed() {
    list($usec, $sec) = explode(' ', microtime());
    return (float) $sec + ((float) $usec * 100000);
}
mt_srand(make_seed());
$randval = mt_rand();
```

Lásd még: mt_rand(), mt_getrandmax(), srand(), rand() és getrandmax()!

number_format (PHP 3, PHP 4)

ezres csoportosítással formázza a számot

string **number_format** (float number [, int decimals [, string dec_point [, string thousands_sep]]]) \linebreak

A **number_format()** a *`number`* paraméter formázott értékével tér vissza. A függvény egy, két vagy négy paramétert fogad el (de hármat nem):

Ha csak egy paraméter van megadva, akkor *`number`* törtrész nélkül, de ezresenkénti csoportosítással kerül formázásra. ("," választva el a csoportokat)

Két paraméter esetén a *`number`* a *`decimals`*-ba megadott számú tizedesre lesz formázva, ponttal (".") a tizedesek előtt és vesszővel (",") választva el a csoportokat.

Négy paraméter esetén a *`number`* a *`decimals`*-ba megadott számú tizedesre lesz formázva. A tizedesek előtt a *`dec_point`* áll a pont (".") helyett, és a *`thousands_sep`* választja el a csoprtokat a vessző (",") helyett.

> **Megjegyzés:** A *`thousands_sep`*-nak csak az első karakterét használja a függvény. Például ha `ize`-t adsz meg mint *`thousands_sep`*, akkor az `1000` számhoz, akkor **number_format()** `1i000`-t fog visszaadni.

Példa 1. number_format() példa

Példaként, a francia jelölés általában két tizedest tartalmaz, és vesszőt használ (',') a tizedesek elválasztására, és szóközzel (' ') tagolja az egész részt. Ez a következőképpen érhető el:

```
<?php
```

```
    $szam = 1234.56;

    // angol jelölés (alapértelmezés)
    $angol_szam_forma = number_format($szam);
    // 1,234.56

    // francia jelölés
    $francia_szam_forma = number_format($szam, 2, ',', ' ');
    // 1 234,56

    $szam = 1234.5678;

    // angol jelölés ezresek csoportosítása nélkül
    $angol_szam_forma = number_format($szam, 2, '.', '');
    // 1234.56

?>
```

Megjegyzés: Lásd még: sprintf(), printf() és sscanf()!

octdec (PHP 3, PHP 4)

nyolcas számrendszerből tízesbe vált át

int **octdec** (string octal_string) \linebreak

A nyolcas számrendszerbeli *`octal_string`* tízes számrendszerbeli megfelelőjével tér vissza. A legnagyobb átváltható szám "17777777777", ami 2147483647-et (tízes számrendszerben) ad eredményül.

Lásd még: decoct()!

pi (PHP 3, PHP 4)

visszaadja pi értékét

float **pi** (void) \linebreak

PI megközelítőleg pontos értékét adja vissza. A visszaadott float érték pontossága a `php.ini`-ben megadott precision beállítástól függ, amely alapértelmezésben `14`. A `M_PI` állandó használata ugyanazt az eredményt adja, mint a **pi()** függvény.

```
echo pi(); // 3.1415926535898
```

```
echo M_PI; // 3.1415926535898
```

pow (PHP 3, PHP 4)

hatványozás

number **pow** (number base, number exp) \linebreak

A *base*-t az *exp* hatványra emelt értékével tér vissza. Ha lehetséges, akkor a függvény visszatérési értéke integer lesz.

Ha a hatványérték nem számítható ki, akkor egy "nem fatális" (warning) hibát jelez a **pow()** függvény, és FALSE-szal tér vissza.

Példa 1. Néhány példa a pow() alkalmazására

```
<?php

var_dump( pow(2,8) ); // int(256)
echo pow(-1,20); // 1
echo pow( 0, 0); // 1

echo pow(-1, 5.5); // hiba

?>
```

Figyelem

PHP 4.0.6-ban és korábbi verziókban a **pow()** mindig float típussal tért vissza, és soha nem jelzett hibát.

Lásd még: exp() és sqrt()!

rad2deg (PHP 3>= 3.0.4, PHP 4)

radiánból fokokra vált

float **rad2deg** (float number) \linebreak

Ez a függvény a radiánban értlemezett *number* paramétert fokokba váltja.

Lásd még: deg2rad()!

rand (PHP 3, PHP 4)

véletlenszámot generál

int **rand** (void) \linebreak int **rand** ([int min, int max]) \linebreak

Ha a *`min`* és *`max`* paraméterek nélkül hívod meg az mt_rand() függvényt, akkor az ál véletlenszám értéke 0 és `RAND_MAX` közé fog esni. Ha pl. 5 és 15 közé (bezárólag) eső véletlen számot akarsz kapni, akkor `mt_rand(5, 15)` kell írni.

Ne felejtsd el a véletlenszám generátor kiinduló értékét beállítani a srand() függvénnyel.

Megjegyzés: A 3.0.7 verziók előtt a *`max`* paraméter helyett *`range`* (tartomány) szerepelt. Ezekben a verziókban az előző példával egyenértékű megoldás a `mt_rand (5, 11)`, hogy 5 és 15 közé eső véletlen számot kapj.

Lásd még: srand(), getrandmax(), mt_rand(), mt_srand() és mt_getrandmax().

round (PHP 3, PHP 4)

lebegőpontos számot kerekít

float **round** (float val [, int precision]) \linebreak

A *`val`* paraméter *`precision`*-ben megadott pontosságra kerekített értékével tér vissza. A *`precision`* a tizedespont után (ill. előtte) álló számjegyek számát jelenti. A *`precision`* nulla vagy negatív szám is lehet.

Figyelem

PHP nem kezeli megfelelően a `"12,300.2"`-hez hasonló sztringeket alapértelmezés szerint. Lásd string konverziók!

```
$ize = round(3.4);  // $ize == 3.0
$ize = round(3.5);  // $ize == 4.0
$ize = round(3.6);  // $ize == 4.0
$ize = round(3.6, 0);  // a fentivel azonos

$ize = round(1.95583, 2);  // $ize == 1.96

$ize = round(1241757, -3);  // $ize == 1242000
```

Megjegyzés: A *`precision`* paraméter csak PHP 4-től használható.

Lásd még: ceil() és floor()!

sin (PHP 3, PHP 4)

szinusz

float **sin** (float arg) \linebreak

A radiánban megadott *arg* szinuszát adja vissza.

Lásd még: cos() és tan()!

sinh (PHP 4 >= 4.1.0)

hiperbolikus szinusz

float **sinh** (float arg) \linebreak

Az *arg* hiperbolikus szinuszát adja vissza, amelyet a `(exp(arg) - exp(-arg))/2` kifejezés határoz meg.

Lásd még: sin(), asinh(), cos() és tan()!

sqrt (PHP 3, PHP 4)

négyzetgyököt von

float **sqrt** (float arg) \linebreak

Az *arg* négyzetgyökét adja vissza.

Lásd még: pow()!

srand (PHP 3, PHP 4)

a véletlenszámgenerátor kiinduló értékét állítja be

void **srand** (int seed) \linebreak

Beállítja a véletlenszám generátor kiinduló értékét *seed*-re.

```
// beállítás ezredmásodpercekkel
function make_seed() {
    list($usec, $sec) = explode(' ', microtime());
    return (float) $sec + ((float) $usec * 100000);
}
```

```
srand(make_seed());
$randval = rand();
```

Lásd még: rand(), getrandmax(), mt_rand(), mt_srand() és mt_getrandmax()!

tan (PHP 3, PHP 4)

tangens

float **tan** (float arg) \linebreak

A radiánban megadott *`arg`* tangensét adja vissza.

Lásd még: sin() és cos()!

tanh (PHP 4 >= 4.1.0)

hiperbolikus tangens

float **tanh** (float arg) \linebreak

Az *`arg`* hiperbolikus tangensét adja vissza, amelyet a `sinh(arg)/cosh(arg)` kifejezés definiál.

Lásd még: tan(), atanh(), sin() és cos()!

LIII. Multi-Byte String Functions

Introduction

There are many languages in which all characters can be expressed by single byte. Multi-byte character codes are used to express many characters for many languages. `mbstring` is developed to handle Japanese characters. However, many `mbstring` functions are able to handle character encoding other than Japanese.

A multi-byte character encoding represents single character with consecutive bytes. Some character encoding has shift(escape) sequences to start/end multi-byte character strings. Therefore, a multi-byte character string may be destroyed when it is divided and/or counted unless multi-byte character encoding safe method is used. This module provides multi-byte character safe string functions and other utility functions such as conversion functions.

Since PHP is basically designed for ISO-8859-1, some multi-byte character encoding does not work well with PHP. Therefore, it is important to set `mbstring.internal_encoding` to a character encoding that works with PHP.

PHP4 Character Encoding Requirements

- Per byte encoding
- Single byte characters in range of `00h-7fh` which is compatible with `ASCII`
- Multi-byte characters without `00h-7fh`

These are examples of internal character encoding that works with PHP and does NOT work with PHP.

```
Character encodings work with PHP:
ISO-8859-*, EUC-JP, UTF-8

Character encodings do NOT work with PHP:
JIS, SJIS
```

Character encoding, that does not work with PHP, may be converted with `mbstring`'s HTTP input/output conversion feature/function.

> **Megjegyzés:** SJIS should not be used for internal encoding unless the reader is familiar with parser/compiler, character encoding and character encoding issues.

> **Megjegyzés:** If you use database with PHP, it is recommended that you use the same character encoding for both database and `internal encoding` for ease of use and better performance.
>
> If you are using PostgreSQL, it supports character encoding that is different from backend

character encoding. See the PostgreSQL manual for details.

How to Enable mbstring

`mbstring` is an extended module. You must enable module with `configure` script. Refer to the Install section for details.

The following configure options are related to `mbstring` module.

- `--enable-mbstring` : Enable `mbstring` functions. This option is required to use `mbstring` functions.
- `--enable-mbstr-enc-trans` : Enable HTTP input character encoding conversion using `mbstring` conversion engine. If this feature is enabled, HTTP input character encoding may be converted to `mbstring.internal_encoding` automatically.

HTTP Input and Output

HTTP input/output character encoding conversion may convert binary data also. Users are supposed to control character encoding conversion if binary data is used for HTTP input/output.

If `enctype` for HTML form is set to `multipart/form-data`, `mbstring` does not convert character encoding in POST data. If it is the case, strings are needed to be converted to internal character encoding.

- HTTP Input

  There is no way to control HTTP input character conversion from PHP script. To disable HTTP input character conversion, it has to be done in `php.ini`.

  Példa 1. Disable HTTP input conversion in `php.ini`

  ```
  ;; Disable HTTP Input conversion
  mbstring.http_input = pass
  ```

  When using PHP as an Apache module, it is possible to override PHP ini setting per Virtual Host in `httpd.conf` or per directory with `.htaccess`. Refer to the Configuration section and Apache Manual for details.

- HTTP Output

  There are several ways to enable output character encoding conversion. One is using `php.ini`, another is using ob_start() with mb_output_handler() as `ob_start` callback function.

  Megjegyzés: For PHP3-i18n users, `mbstring`'s output conversion differs from PHP3-i18n.

Character encoding is converted using output buffer.

Példa 2. `php.ini` setting example

```
;; Enable output character encoding conversion for all PHP pages

;; Enable Output Buffering
output_buffering    = On

;; Set mb_output_handler to enable output conversion
output_handler      = mb_output_handler
```

Példa 3. Script example

```
<?php

// Enable output character encoding conversion only for this page

// Set HTTP output character encoding to SJIS
mb_http_output('SJIS');

// Start buffering and specify "mb_output_handler" as
// callback function
ob_start('mb_output_handler');

?>
```

Supported Character Encoding

Currently, the following character encoding is supported by `mbstring` module. Caracter encoding may be specified for `mbstring` functions' `encoding` parameter.

The following character encoding is supported in this PHP extension :

`UCS-4`, `UCS-4BE`, `UCS-4LE`, `UCS-2`, `UCS-2BE`, `UCS-2LE`, `UTF-32`, `UTF-32BE`, `UTF-32LE`, `UCS-2LE`, `UTF-16`, `UTF-16BE`, `UTF-16LE`, `UTF-8`, `UTF-7`, `ASCII`, `EUC-JP`, `SJIS`, `eucJP-win`, `SJIS-win`, `ISO-2022-JP`, `JIS`, `ISO-8859-1`, `ISO-8859-2`, `ISO-8859-3`, `ISO-8859-4`, `ISO-8859-5`, `ISO-8859-6`, `ISO-8859-7`, `ISO-8859-8`, `ISO-8859-9`, `ISO-8859-10`,

`ISO-8859-13`, `ISO-8859-14`, `ISO-8859-15`, `byte2be`, `byte2le`, `byte4be`, `byte4le`, `BASE64`, `7bit`, `8bit` and `UTF7-IMAP`.

`php.ini` entry, which accepts encoding name, accepts "`auto`" and "`pass`" also. `mbstring` functions, which accepts encoding name, and accepts "`auto`".

If "`pass`" is set, no character encoding conversion is performed.

If "`auto`" is set, it is expanded to "`ASCII,JIS,UTF-8,EUC-JP,SJIS`".

See also mb_detect_order()

> **Megjegyzés:** "Supported character encoding" does not mean that it works as internal character code.

`php.ini` settings

- `mbstring.internal_encoding` defines default internal character encoding.
- `mbstring.http_input` defines default HTTP input character encoding.
- `mbstring.http_output` defines default HTTP output character encoding.
- `mbstring.detect_order` defines default character code detection order. See also mb_detect_order().
- `mbstring.substitute_character` defines character to substitute for invalid character encoding.

Web Browsers are supposed to use the same character encoding when submitting form. However, browsers may not use the same character encoding. See mb_http_input() to detect character encoding used by browsers.

If `enctype` is set to `multipart/form-data` in HTML forms, `mbstring` does not convert character encoding in POST data. The user must convert them in the script, if conversion is needed.

Although, browsers are smart enough to detect character encoding in HTML. `charset` is better to be set in HTTP header. Change `default_charset` according to character encoding.

Példa 4. `php.ini` setting example

```
;; Set default internal encoding
;; Note: Make sure to use character encoding works with PHP
mbstring.internal_encoding    = UTF-8  ; Set internal encoding to UTF-8

;; Set default HTTP input character encoding
;; Note: Script cannot change http_input setting.
mbstring.http_input           = pass    ; No conversion.
mbstring.http_input           = auto    ; Set HTTP input to auto
                                ; "auto" is expanded to "ASCII,JIS,UTF-8,EUC-
JP,SJIS"
mbstring.http_input           = SJIS    ; Set HTTP2 input to  SJIS
mbstring.http_input           = UTF-8,SJIS,EUC-JP ; Specify order
```

```
;; Set default HTTP output character encoding
mbstring.http_output          = pass    ; No conversion
mbstring.http_output          = UTF-8   ; Set HTTP output encoding to UTF-8

;; Set default character encoding detection order
mbstring.detect_order         = auto    ; Set detect order to auto
mbstring.detect_order         = ASCII,JIS,UTF-8,SJIS,EUC-JP ; Specify order

;; Set default substitute character
mbstring.substitute_character = 12307   ; Specify Unicode value
mbstring.substitute_character = none    ; Do not print character
mbstring.substitute_character = long    ; Long Example: U+3000,JIS+7E7E
```

Példa 5. `php.ini` setting for `EUC-JP` users

```
;; Disable Output Buffering
output_buffering      = Off

;; Set HTTP header charset
default_charset       = EUC-JP

;; Set HTTP input encoding conversion to auto
mbstring.http_input   = auto

;; Convert HTTP output to EUC-JP
mbstring.http_output  = EUC-JP

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Példa 6. `php.ini` setting for `SJIS` users

```
;; Enable Output Buffering
output_buffering     = On

;; Set mb_output_handler to enable output conversion
output_handler       = mb_output_handler

;; Set HTTP header charset
default_charset      = Shift_JIS

;; Set http input encoding conversion to auto
```

```
mbstring.http_input  = auto

;; Convert to SJIS
mbstring.http_output = SJIS

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Overload of PHP string functions by mbstring functions with multibyte support

Because almost PHP application written for language using single-byte character encoding, there are some difficulties for multibyte string handling including japanese. Almost PHP string functions such as substr() do not support multibyte string.

Multibyte extension (mbstring) has some PHP string functions with multibyte support (ex. substr() supports mb_substr()).

Multibyte extension (mbstring) also supports ’function overloading’ to add multibyte string functionality without code modification. Using function overloading, some PHP string functions will be oveloaded multibyte string functions. For example, mb_substr() is called instead of substr() if function overloading is enabled. Function overload makes easy to port application supporting only single-byte encoding for multibyte application.

`mbstring.func_overload` in `php.ini` should be set some positive value to use function overloading. The value should specify the category of overloading functions, sbould be set 1 to enable mail function overloading. 2 to enable string functions, 4 to regular expression functions. For example, if is set for 7, mail, strings, regex functions should be overloaded. The list of overloaded functions are shown in below.

Táblázat 1. Functions to be overloaded

value of mbstring.func_overload	original function	overloaded function
1	mail()	mb_send_mail()
2	strlen()	mb_strlen()
2	strpos()	mb_strpos()
2	strrpos()	mb_strrpos()
2	substr()	mb_substr()
4	ereg()	mb_ereg()
4	eregi()	mb_eregi()
4	ereg_replace()	mb_ereg_replace()
4	eregi_replace()	mb_eregi_replace()
4	split()	mb_split()

Basics for Japanese multi-byte character

Most Japanese characters need more than 1 byte per character. In addition, several character encoding schemas are used under a Japanese environment. There are EUC-JP, Shift_JIS(SJIS) and ISO-2022-JP(JIS) character encoding. As Unicode becomes popular, UTF-8 is used also. To develop Web applications for a Japanese environment, it is important to use the character set for the task in hand, whether HTTP input/output, RDBMS and E-mail.

- Storage for a character can be up to six bytes
- A multi-byte character is usually twice of the width compared to single-byte characters. Wider characters are called "zen-kaku" - meaning full width, narrower characters are called "han-kaku" - meaning half width. "zen-kaku" characters are usually fixed width.
- Some character encoding defines shift(escape) sequence for entering/exiting multi-byte character strings.
- ISO-2022-JP must be used for SMTP/NNTP.
- "i-mode" web site is supposed to use SJIS.

References

Multi-byte character encoding and its related issues are very complex. It is impossible to cover in sufficient detail here. Please refer to the following URLs and other resources for further readings.

- Unicode/UTF/UCS/etc

  `http://www.unicode.org/`

- Japanese/Korean/Chinese character information

  `ftp://ftp.ora.com/pub/examples/nutshell/ujip/doc/cjk.inf`

mb_convert_encoding (PHP 4 >= 4.0.6)

Convert character encoding

string **mb_convert_encoding** (string str, string to-encoding [, mixed from-encoding]) \linebreak

mb_convert_encoding() converts character encoding of string *`str`* from *`from-encoding`* to *`to-encoding`*.

`str` : String to be converted.

`from-encoding` is specified by character code name before conversion. it can be array or string - comma separated enumerated list. If it is not specified, the internal encoding will be used.

Példa 1. mb_convert_encoding() example

```
/* Convert internal character encoding to SJIS */
$str = mb_convert_encoding($str, "SJIS");

/* Convert EUC-JP to UTF-7 */
$str = mb_convert_encoding($str, "UTF-7", "EUC-JP");

/* Auto detect encoding from JIS, eucjp-win, sjis-win, then convert str to UCS-
2LE */
$str = mb_convert_encoding($str, "UCS-2LE", "JIS, eucjp-win, sjis-win");

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
$str = mb_convert_encoding($str, "EUC-JP", "auto");
```

See also: mb_detect_order().

mb_convert_kana (PHP 4 >= 4.0.6)

Convert "kana" one from another ("zen-kaku" ,"han-kaku" and more)

string **mb_convert_kana** (string str, string option [, mixed encoding]) \linebreak

mb_convert_kana() performs "han-kaku" - "zen-kaku" conversion for string *`str`*. It returns converted string. This function is only useful for Japanese.

`option` is conversion option. Default value is `"KV"`.

`encoding` is character encoding. If it is omitted, internal character encoding is used.

```
Applicable Conversion Options

option : Specify with conversion of following options. Default "KV"
"r" :  Convert "zen-kaku" alphabets to "han-kaku"
"R" :  Convert "han-kaku" alphabets to "zen-kaku"
```

```
        "n" :  Convert "zen-kaku" numbers to "han-kaku"
        "N" :  Convert "han-kaku" numbers to "zen-kaku"
        "a" :  Convert "zen-kaku" alphabets and numbers to "han-kaku"
        "A" :  Convert "zen-kaku" alphabets and numbers to "han-kaku"
        (Characters included in "a", "A" options are
        U+0021 - U+007E excluding U+0022, U+0027, U+005C, U+007E)
        "s" :  Convert "zen-kaku" space to "han-kaku" (U+3000 -> U+0020)
        "S" :  Convert "han-kaku" space to "zen-kaku" (U+0020 -> U+3000)
        "k" :  Convert "zen-kaku kata-kana" to "han-kaku kata-kana"
        "K" :  Convert "han-kaku kata-kana" to "zen-kaku kata-kana"
        "h" :  Convert "zen-kaku hira-gana" to "han-kaku kata-kana"
        "H" :  Convert "han-kaku kata-kana" to "zen-kaku hira-gana"
        "c" :  Convert "zen-kaku kata-kana" to "zen-kaku hira-gana"
        "C" :  Convert "zen-kaku hira-gana" to "zen-kaku kata-kana"
        "V" :  Collapse voiced sound notation and convert them into a charac-
ter. Use with "K","H"
```

Példa 1. mb_convert_kana() example

```
/* Convert all "kana" to "zen-kaku" "kata-kana" */
$str = mb_convert_kana($str, "KVC");

/* Convert "han-kaku" "kata-kana" to "zen-kaku" "kata-kana"
   and "zen-kaku" alpha-numeric to "han-kaku" */
$str = mb_convert_kana($str, "KVa");
```

mb_convert_variables (PHP 4 >= 4.0.6)

Convert character code in variable(s)

string **mb_convert_variables** (string to-encoding, mixed from-encoding, mixed vars) \linebreak

mb_convert_variables() convert character encoding of variables *`vars`* in encoding *`from-encoding`* to encoding *`to-encoding`*. It returns character encoding before conversion for success, `FALSE` for failure.

mb_convert_variables() join strings in Array or Object to detect encoding, since encoding detection tends to fail for short strings. Therefore, it is impossible to mix encoding in single array or object.

It *`from-encoding`* is specified by array or comma separated string, it tries to detect encoding from *`from-coding`*. When *`encoding`* is omitted, `detect_order` is used.

`vars (3rd and larger)` is reference to variable to be converted. String, Array and Object are accepted. **mb_convert_variables()** assumes all parameters have the same encoding.

Példa 1. mb_convert_variables() example

```
/* Convert variables $post1, $post2 to internal encoding */
$interenc = mb_internal_encoding();
$inputenc = mb_convert_variables($interenc, "ASCII,UTF-8,SJIS-win", $post1, $post2);
```

mb_decode_mimeheader (PHP 4 >= 4.0.6)

Decode string in MIME header field

string **mb_decode_mimeheader** (string str) \linebreak

mb_decode_mimeheader() decodes encoded-word string *`str`* in MIME header.

It returns decoded string in internal character encoding.

See also mb_encode_mimeheader().

mb_decode_numericentity (PHP 4 >= 4.0.6)

Decode HTML numeric string reference to character

string **mb_decode_numericentity** (string str, array convmap [, string encoding]) \linebreak

Convert numeric string reference of string *`str`* in specified block to character. It returns converted string.

`array` is array to specifies code area to convert.

`encoding` is character encoding. If it is omitted, internal character encoding is used.

Példa 1. *`convmap`* example

```
$convmap = array (
   int start_code1, int end_code1, int offset1, int mask1,
   int start_code2, int end_code2, int offset2, int mask2,
   ........
   int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN,
// then convert value to numeric string reference.
```

See also: mb_encode_numericentity().

mb_detect_encoding (PHP 4 >= 4.0.6)

Detect character encoding

string **mb_detect_encoding** (string str [, mixed encoding-list]) \linebreak

mb_detect_encoding() detects character encoding in string *`str`*. It returns detected character encoding.

`encoding-list` is list of character encoding. Encoding order may be specified by array or comma separated list string.

If *`encoding_list`* is omitted, detect_order is used.

Példa 1. mb_detect_encoding() example

```
/* Detect character encoding with current detect_order */
echo mb_detect_encoding($str);

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
echo mb_detect_encoding($str, "auto");

/* Specify encoding_list character encoding by comma separated list */
echo mb_detect_encoding($str, "JIS, eucjp-win, sjis-win");

/* Use array to specify encoding_list  */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
echo mb_detect_encoding($str, $ary);
```

See also: mb_detect_order().

mb_detect_order (PHP 4 >= 4.0.6)

Set/Get character encoding detection order

array **mb_detect_order** ([mixed encoding-list]) \linebreak

mb_detect_order() sets automatic character encoding detection order to *`encoding-list`*. It returns `TRUE` for success, `FALSE` for failure.

`encoding-list` is array or comma separated list of character encoding. ("auto" is expanded to "ASCII, JIS, UTF-8, EUC-JP, SJIS")

If *`encoding-list`* is omitted, it returns current character encoding detection order as array.

This setting affects mb_detect_encoding() and mb_send_mail().

> **Megjegyzés:** `mbstring` currently implements following encoding detection filters. If there is a invalid byte sequence for following encoding, encoding detection will fail.

Megjegyzés: `UTF-8`, `UTF-7`, `ASCII`, `EUC-JP`,`SJIS`, `eucJP-win`, `SJIS-win`, `JIS`, `ISO-2022-JP`

For `ISO-8859-*`, `mbstring` always detects as `ISO-8859-*`.

For `UTF-16`, `UTF-32`, `UCS2` and `UCS4`, encoding detection will fail always.

Példa 1. Useless detect order example

```
; Always detect as ISO-8859-1
detect_order = ISO-8859-1, UTF-8

; Always detect as UTF-8, since ASCII/UTF-7 values are
; valid for UTF-8
detect_order = UTF-8, ASCII, UTF-7
```

Példa 2. mb_detect_order() examples

```
/* Set detection order by enumerated list */
mb_detect_order("eucjp-win,sjis-win,UTF-8");

/* Set detection order by array */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
mb_detect_order($ary);

/* Display current detection order */
echo implode(", ", mb_detect_order());
```

See also mb_internal_encoding(), mb_http_input(), mb_http_output() mb_send_mail().

mb_encode_mimeheader (PHP 4 >= 4.0.6)

Encode string for MIME header

string **mb_encode_mimeheader** (string str [, string charset [, string transfer-encoding [, string linefeed]]]) \linebreak

mb_encode_mimeheader() converts string *`str`* to encoded-word for header field. It returns converted string in ASCII encoding.

`charset` is character encoding name. Default is `ISO-2022-JP`.

transfer-encoding is transfer encoding. It should be one of "B" (Base64) or "Q" (Quoted-Printable). Default is "B".

linefeed is end of line marker. Default is "\r\n" (CRLF).

Példa 1. mb_convert_kana() example

```
$name = ""; // kanji
$mbox = "kru";
$doma = "gtinn.mon";
$addr = mb_encode_mimeheader($name, "UTF-7", "Q") . " <" . $mbox . "@" . $doma . ">";
echo $addr;
```

See also mb_decode_mimeheader().

mb_encode_numericentity (PHP 4 >= 4.0.6)

Encode character to HTML numeric string reference

string **mb_encode_numericentity** (string str, array convmap [, string encoding]) \linebreak

mb_encode_numericentity() converts specified character codes in string *str* from HTML numeric character reference to character code. It returns converted string.

array is array specifies code area to convert.

encoding is character encoding.

Példa 1. *convmap* example

```
$convmap = array (
 int start_code1, int end_code1, int offset1, int mask1,
 int start_code2, int end_code2, int offset2, int mask2,
 ........
 int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN, then
// it converts value to numeric string reference.
```

Példa 2. mb_encode_numericentity() example

```
/* Convert Left side of ISO-8859-1 to HTML numeric character reference */
$convmap = array(0x80, 0xff, 0, 0xff);
```

```
$str = mb_encode_numericentity($str, $convmap, "ISO-8859-1");

/* Convert user defined SJIS-win code in block 95-104 to numeric
   string reference */
$convmap = array(
       0xe000, 0xe03e, 0x1040, 0xffff,
       0xe03f, 0xe0bb, 0x1041, 0xffff,
       0xe0bc, 0xe0fa, 0x1084, 0xffff,
       0xe0fb, 0xe177, 0x1085, 0xffff,
       0xe178, 0xe1b6, 0x10c8, 0xffff,
       0xe1b7, 0xe233, 0x10c9, 0xffff,
       0xe234, 0xe272, 0x110c, 0xffff,
       0xe273, 0xe2ef, 0x110d, 0xffff,
       0xe2f0, 0xe32e, 0x1150, 0xffff,
       0xe32f, 0xe3ab, 0x1151, 0xffff );
$str = mb_encode_numericentity($str, $convmap, "sjis-win");
```

See also: mb_decode_numericentity().

mb_ereg_match (PHP 4 >= 4.2.0)

Regular expression match for multibyte string

bool **mb_ereg_match** (string pattern, string string [, string option]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg_match() returns TRUE if *`string`* matches regular expression *`pattern`*, FALSE if not.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_ereg_replace (PHP 4 >= 4.2.0)

Replace regular expression with multibyte support

string **mb_ereg_replace** (string pattern, string replacement, string string [, array option]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg_replace() scans *`string`* for matches to *`pattern`*, then replaces the matched text with *`replacement`* and returns the result string or `FALSE` on error. Multibyte character can be used in *`pattern`*.

Matching condition can be set by *`option`* parameter. If `i` is specified for this parameter, the case will be ignored. If `x` is specified, white space will be ignored. If `m` is specified, match will be executed in multiline mode and line break will be included in '.'. If `p` is specified, match will be executed in POSIX mode, line break will be considered as normal character. If `e` is specified, *`replacement`* string will be evaluated as PHP expression.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_eregi_replace().

mb_ereg_search_getpos (PHP 4 >= 4.2.0)

Returns start point for next regular expression match

array **mb_ereg_search_getpos** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg_search_getpos() returns the point to start regular expression match for mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). The position is represented by bytes from the head of string.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_setpos().

mb_ereg_search_getregs (PHP 4 >= 4.2.0)

Retrive the result from the last multibyte regular expression match

array **mb_ereg_search_getregs** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg_search_getregs() returns an array including the sub-string of matched part by last mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). If there are some maches, the first element will have the matched sub-string, the second element will have the first part grouped with brackets, the third element will have the second part grouped with brackets, and so on. It returns FALSE on error;

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_init (PHP 4 >= 4.2.0)

Setup string and regular expression for multibyte regular expression match

array **mb_ereg_search_init** (string string [, string pattern [, string option]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg_search_init() sets *`string`* and *`pattern`* for multibyte regular expression. These values are used for mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). It returns TRUE for success, FALSE for error.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_regs().

mb_ereg_search_pos (PHP 4 >= 4.2.0)

Return position and length of matched part of multibyte regular expression for predefined multibyte string

array **mb_ereg_search_pos** ([string pattern [, string option]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg_search_pos() returns an array including position of matched part for multibyte regular expression. The first element of the array will be the beggining of matched part, the second element will be length (bytes) of matched part. It returns FALSE on error.

The string for match is specified by mb_ereg_search_init(). It it is not specified, the previous one will be used.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_regs (PHP 4 >= 4.2.0)

Returns the matched part of multibyte regular expression

array **mb_ereg_search_regs** ([string pattern [, string option]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg_search_regs() executes the multibyte regular expression match, and if there are some matched part, it returns an array including substring of matched part as first element, the first grouped part with brackets as second element, the second grouped part as third element, and so on. It returns FALSE on error.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_setpos (PHP 4 >= 4.2.0)

Set start point of next regular expression match

array **mb_ereg_search_setpos** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg_search_setpos() sets the starting point of match for mb_ereg_search().

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search (PHP 4 >= 4.2.0)

Multibyte regular expression match for predefined multibyte string

bool **mb_ereg_search** ([string pattern [, string option]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg_search() returns TRUE if the multibyte string matches with the regular expression, FALSE for otherwise. The string for matching is set by mb_ereg_search_init(). If `pattern` is not specified, the previous one is used.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg (PHP 4 >= 4.2.0)

Regular expression match with multibyte support

int **mb_ereg** (string pattern, string string [, array regs]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg() executes the regular expression match with multibyte support, and returns 1 if matches are found. If the optional third parameter was specified, the function returns the byte length of matched part, and therarray `regs` will contain the substring of matched string. The functions returns 1 if it matches with the empty string. It no matche found or error happend, FALSE will be returned.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_eregi()

mb_eregi_replace (PHP 4 >= 4.2.0)

Replace regular expression with multibyte support ignoring case

string **mb_eregi_replace** (string pattern, string replace, string string) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_ereg_replace() scans `string` for matches to `pattern`, then replaces the matched text with `replacement` and returns the result string or FALSE on error. Multibyte character can be used in `pattern`. The case will be ignored.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_replace().

mb_eregi (PHP 4 >= 4.2.0)

Regular expression match ignoring case with multibyte support

int **mb_eregi** (string pattern, string string [, array regs]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_eregi() executes the regular expression match with multibyte support, and returns 1 if matches are found. This function ignore case. If the optional third parameter was specified, the function returns the byte length of matched part, and therarray `regs` will contain the substring of matched string. The functions returns 1 if it matches with the empty string. It no matche found or error happend, FALSE will be returned.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_get_info (PHP 4 >= 4.2.0)

Get internal settings of mbstring

string **mb_get_info** ([string type]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_get_info() returns internal setting parameter of mbstring.

If `type` isn't specified or is specified to "all", an array having the elements "internal_encoding", "http_output", "http_input", "func_overload" will be returned.

If *type* is specified for "http_output", "http_input", "internal_encoding", "func_overload", the specified setting parameter will be returned.

See also mb_internal_encoding(), mb_http_output().

mb_http_input (PHP 4 >= 4.0.6)

Detect HTTP input character encoding

string **mb_http_input** ([string type]) \linebreak

mb_http_input() returns result of HTTP input character encoding detection.

type: Input string specifies input type. "G" for GET, "P" for POST, "C" for COOKIE. If type is omitted, it returns last input type processed.

Return Value: Character encoding name. If **mb_http_input()** does not process specified HTTP input, it returns FALSE.

See also mb_internal_encoding(), mb_http_output(), mb_detect_order().

mb_http_output (PHP 4 >= 4.0.6)

Set/Get HTTP output character encoding

string **mb_http_output** ([string encoding]) \linebreak

If *encoding* is set, **mb_http_output()** sets HTTP output character encoding to *encoding*. Output after this function is converted to *encoding*. **mb_http_output()** returns TRUE for success and FALSE for failure.

If *encoding* is omitted, **mb_http_output()** returns current HTTP output character encoding.

See also mb_internal_encoding(), mb_http_input(), mb_detect_order().

mb_internal_encoding (PHP 4 >= 4.0.6)

Set/Get internal character encoding

string **mb_internal_encoding** ([string encoding]) \linebreak

mb_internal_encoding() sets internal character encoding to *encoding* If parameter is omitted, it returns current internal encoding.

encoding is used for HTTP input character encoding conversion, HTTP output character encoding conversion and default character encoding for string functions defined by mbstring module.

encoding: Character encoding name

Return Value: If *encoding* is set,**mb_internal_encoding()** returns TRUE for success, otherwise returns FALSE. If *encoding* is omitted, it returns current character encoding name.

Példa 1. mb_internal_encoding() example

```
/* Set internal character encoding to UTF-8 */
mb_internal_encoding("UTF-8");

/* Display current internal character encoding */
echo mb_internal_encoding();
```

See also mb_http_input(), mb_http_output(), mb_detect_order().

mb_language (PHP 4 >= 4.0.6)

Set/Get current language

string **mb_language** ([string language]) \linebreak

mb_language() sets language. If *language* is omitted, it returns current language as string.

language setting is used for encoding e-mail messages. Valid languages are "Japanese", "ja","English","en" and "uni" (UTF-8). mb_send_mail() uses this setting to encode e-mail.

Language and its setting is ISO-2022-JP/Base64 for Japanese, UTF-8/Base64 for uni, ISO-8859-1/quoted printable for English.

Return Value: If *language* is set and *language* is valid, it returns `TRUE`. Otherwise, it returns `FALSE`. When *language* is omitted, it returns language name as string. If no language is set previously, it returns `FALSE`.

See also mb_send_mail().

mb_output_handler (PHP 4 >= 4.0.6)

Callback function converts character encoding in output buffer

string **mb_output_handler** (string contents, int status) \linebreak

mb_output_handler() is ob_start() callback function. **mb_output_handler()** converts characters in output buffer from internal character encoding to HTTP output character encoding.

4.1.0 or later version, this hanlder adds charset HTTP header when following conditions are met:

- Does not set `Content-Type` by header()
- Default MIME type begins with `text/`
- `http_output` setting is other than pass

contents : Output buffer contents

status : Output buffer status

Return Value: String converted

Példa 1. mb_output_handler() example

```
mb_http_output("UTF-8");
ob_start("mb_output_handler");
```

Megjegyzés: If you want to output some binary data such as image from PHP script, you must set output encoding to "pass" using mb_http_output().

See also ob_start().

mb_parse_str (PHP 4 >= 4.0.6)

Parse GET/POST/COOKIE data and set global variable

boolean **mb_parse_str** (string encoded_string [, array result]) \linebreak

mb_parse_str() parses GET/POST/COOKIE data and sets global variables. Since PHP does not provide raw POST/COOKIE data, it can only used for GET data for now. It preses URL encoded data, detects encoding, converts coding to internal encoding and set values to *result* array or global variables.

encoded_string: URL encoded data.

result: Array contains decoded and character encoding converted values.

Return Value: It returns TRUE for success or FALSE for failure.

See also mb_detect_order(), mb_internal_encoding().

mb_preferred_mime_name (PHP 4 >= 4.0.6)

Get MIME charset string

string **mb_preferred_mime_name** (string encoding) \linebreak

mb_preferred_mime_name() returns MIME charset string for character encoding *encoding*. It returns charset string.

Példa 1. mb_preferred_mime_string() example

```
$outputenc = "sjis-win";
mb_http_output($outputenc);
ob_start("mb_output_handler");
header("Content-Type: text/html; charset=" . mb_preferred_mime_name($outputenc));
```

mb_regex_encoding (PHP 4 >= 4.2.0)

Returns current encoding for multibyte regex as string

string **mb_regex_encoding** ([string encoding]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_regex_encoding() returns the character encoding used by multibyte regex functions.

If the optional parameter *encoding* is specified, it is set to the character encoding for multibyte regex. The default value is the internal character encoding.

Megjegyzés: This function is supported in PHP 4.2.0 or higher.

See also: mb_internal_encoding(), mb_ereg()

mb_send_mail (PHP 4 >= 4.0.6)

Send encoded mail.

boolean **mb_send_mail** (string to, string subject, string message [, string additional_headers [, string additional_parameter]]) \linebreak

mb_send_mail() sends email. Headers and message are converted and encoded according to mb_language() setting. **mb_send_mail()** is wrapper function of mail(). See mail() for details.

to is mail addresses send to. Multiple recipients can be specified by putting a comma between each address in to. This parameter is not automatically encoded.

subject is subject of mail.

message is mail message.

additional_headers is inserted at the end of the header. This is typically used to add extra headers. Multiple extra headers are separated with a newline ("\n").

additional_parameter is a MTA command line parameter. It is useful when setting the correct Return-Path header when using sendmail.

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

See also mail(), mb_encode_mimeheader(), and mb_language().

mb_split (PHP 4 >= 4.2.0)

Split multibyte string using regular expression

array **mb_split** (string pattern, string string [, int limit]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

mb_split() split multibyte *string* using regular expression *pattern* and returns the result as an array.

If optional parameter *limit* is specified, it will be split in *limit* elements as maximum.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Megjegyzés:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_strcut (PHP 4 >= 4.0.6)

Get part of string

string **mb_strcut** (string str, int start [, int length [, string encoding]]) \linebreak

mb_strcut() returns the portion of *str* specified by the *start* and *length* parameters.

mb_strcut() performs equivalent operation as mb_substr() with different method. If *start* position is multi-byte character's second byte or larger, it starts from first byte of multi-byte character.

It subtracts string from *str* that is shorter than *length* AND character that is not part of multi-byte string or not being middle of shift sequence.

encoding is character encoding. If it is not set, internal character encoding is used.

See also mb_substr(), mb_internal_encoding().

mb_strimwidth (PHP 4 >= 4.0.6)

Get truncated string with specified width

string **mb_strimwidth** (string str, int start, int width, string trimmarker [, string encoding]) \linebreak

mb_strimwidth() truncates string *str* to specified *width*. It returns truncated string.

If *trimmarker* is set, *trimmarker* is appended to return value.

start is start position offset. Number of characters from the beginning of string. (First character is 0)

trimmarker is string that is added to the end of string when string is truncated.

encoding is character encoding. If it is omitted, internal encoding is used.

Példa 1. mb_strimwidth() example

```
$str = mb_strimwidth($str, 0, 40, "..>");
```

See also: mb_strwidth(), mb_internal_encoding().

mb_strlen (PHP 4 >= 4.0.6)

Get string length

string **mb_strlen** (string str [, string encoding]) \linebreak

mb_strlen() returns number of characters in string *str* having character encoding *encoding*. A multi-byte character is counted as 1.

encoding is character encoding for *str*. If *encoding* is omitted, internal character encoding is used.

See also mb_internal_encoding(), strlen().

mb_strpos (PHP 4 >= 4.0.6)

Find position of first occurrence of string in a string

int **mb_strpos** (string haystack, string needle [, int offset [, string encoding]]) \linebreak

mb_strpos() returns the numeric position of the first occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns FALSE.

mb_strpos() performs multi-byte safe strpos() operation based on number of characters. *needle* position is counted from the beginning of the *haystack*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal character encoding is used. mb_strrpos() accepts `string` for *needle* where strrpos() accepts only character.

offset is search offset. If it is not specified, 0 is used.

encoding is character encoding name. If it is omitted, internal character encoding is used.

See also **mb_strpos()**, mb_internal_encoding(), strpos()

mb_strrpos (PHP 4 >= 4.0.6)

Find position of last occurrence of a string in a string

int **mb_strrpos** (string haystack, string needle [, string encoding]) \linebreak

mb_strrpos() returns the numeric position of the last occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns `FALSE`.

mb_strrpos() performs multi-byte safe strrpos() operation based on number of characters. *needle* position is counted from the beginning of *haystack*. First character's position is 0. Second character position is 1.

If *encoding* is omitted, internal encoding is assumed. **mb_strrpos()** accepts `string` for *needle* where strrpos() accepts only character.

encoding is character encoding. If it is not specified, internal character encoding is used.

See also mb_strpos(), mb_internal_encoding(), strrpos().

mb_strwidth (PHP 4 >= 4.0.6)

Return width of string

int **mb_strwidth** (string str [, string encoding]) \linebreak

mb_strwidth() returns width of string *str*.

Multi-byte character usually twice of width compare to single byte character.

```
Character width

U+0000 - U+0019   0
U+0020 - U+1FFF   1
U+2000 - U+FF60   2
U+FF61 - U+FF9F   1
U+FFA0 -          2
```

encoding is character encoding. If it is omitted, internal encoding is used.

See also: mb_strimwidth(), mb_internal_encoding().

mb_substitute_character (PHP 4 >= 4.0.6)

Set/Get substitution character

mixed **mb_substitute_character** ([mixed substrchar]) \linebreak

mb_substitute_character() specifies substitution character when input character encoding is invalid or character code is not exist in output character encoding. Invalid characters may be substituted `NULL`(no output), string or integer value (Unicode character code value).

This setting affects mb_detect_encoding() and mb_send_mail().

`substchar` : Specify Unicode value as integer or specify as string as follows

- "none" : no output
- "long" : Output character code value (Example: U+3000,JIS+7E7E)

Return Value: If *`substchar`* is set, it returns `TRUE` for success, otherwise returns `FALSE`. If *`substchar`* is not set, it returns Unicode value or "`none`"/"`long`".

Példa 1. mb_substitute_character() example

```
/* Set with Unicode U+3013 (GETA MARK) */
mb_substitute_character(0x3013);

/* Set hex format */
mb_substitute_character("long");

/* Display current setting */
echo mb_substitute_character();
```

mb_substr (PHP 4 >= 4.0.6)

Get part of string

string **mb_substr** (string str, int start [, int length [, string encoding]]) \linebreak

mb_substr() returns the portion of *`str`* specified by the *`start`* and *`length`* parameters.

mb_substr() performs multi-byte safe substr() operation based on number of characters. Position is counted from the beginning of *`str`*. First character's position is 0. Second character position is 1, and so on.

If *`encoding`* is omitted, internal encoding is assumed.

`encoding` is character encoding. If it is omitted, internal character encoding is used.

See also mb_strcut(), mb_internal_encoding().

LIV. MCAL functions

MCAL stands for Modular Calendar Access Library.

Libmcal is a C library for accessing calendars. It’s written to be very modular, with pluggable drivers. MCAL is the calendar equivalent of the IMAP module for mailboxes.

With mcal support, a calendar stream can be opened much like the mailbox stream with the IMAP support. Calendars can be local file stores, remote ICAP servers, or other formats that are supported by the mcal library.

Calendar events can be pulled up, queried, and stored. There is also support for calendar triggers (alarms) and recurring events.

With libmcal, central calendar servers can be accessed, removing the need for any specific database or local file programming.

To get these functions to work, you have to compile PHP with `--with-mcal`. That requires the mcal

library to be installed. Grab the latest version from http://mcal.chek.com/ and compile and install it.

The following constants are defined when using the MCAL module. For weekdays :

- MCAL_SUNDAY
- MCAL_MONDAY
- MCAL_TUESDAY
- MCAL_WEDNESDAY
- MCAL_THURSDAY
- MCAL_FRIDAY
- MCAL_SATURDAY

For recurrence :

- MCAL_RECUR_NONE
- MCAL_RECUR_DAILY
- MCAL_RECUR_WEEKLY
- MCAL_RECUR_MONTHLY_MDAY
- MCAL_RECUR_MONTHLY_WDAY
- MCAL_RECUR_YEARLY

For months :

- MCAL_JANUARY
- MCAL_FEBRUARY
- MCAL_MARCH
- MCAL_APRIL
- MCAL_MAY
- MCAL_JUNE
- MCAL_JULY
- MCAL_AUGUST
- MCAL_SEPTEMBER
- MCAL_OCTOBER
- MCAL_NOVEMBER
- MCAL_DECEMBER

Most of the functions use an internal event structure that is unique for each stream. This alleviates the need to pass around large objects between functions. There are convenience functions for setting, initializing, and retrieving the event structure values.

mcal_append_event (PHP 4)

Store a new event into an MCAL calendar

int **mcal_append_event** (int mcal_stream) \linebreak

mcal_append_event() Stores the global event into an MCAL calendar for the given stream.

Returns the id of the newly inserted event.

mcal_close (PHP 3>= 3.0.13, PHP 4)

Close an MCAL stream

int **mcal_close** (int mcal_stream, int flags) \linebreak

Closes the given mcal stream.

mcal_create_calendar (PHP 3>= 3.0.13, PHP 4)

Create a new MCAL calendar

string **mcal_create_calendar** (int stream, string calendar) \linebreak

Creates a new calendar named *calendar*.

mcal_date_compare (PHP 3>= 3.0.13, PHP 4)

Compares two dates

int **mcal_date_compare** (int a_year, int a_month, int a_day, int b_year, int b_month, int b_day) \linebreak

mcal_date_compare() Compares the two given dates, returns <0, 0, >0 if a<b, a==b, a>b respectively.

mcal_date_valid (PHP 3>= 3.0.13, PHP 4)

Returns TRUE if the given year, month, day is a valid date

int **mcal_date_valid** (int year, int month, int day) \linebreak

mcal_date_valid() Returns TRUE if the given year, month and day is a valid date, FALSE if not.

mcal_day_of_week (PHP 3>= 3.0.13, PHP 4)

Returns the day of the week of the given date

int **mcal_day_of_week** (int year, int month, int day) \linebreak

mcal_day_of_week() returns the day of the week of the given date. Possible return values range from 0 for Sunday through 6 for Saturday.

mcal_day_of_year (PHP 3>= 3.0.13, PHP 4)

Returns the day of the year of the given date

int **mcal_** (int year, int month, int day) \linebreak

mcal_day_of_year() returns the day of the year of the given date.

mcal_days_in_month (PHP 3>= 3.0.13, PHP 4)

Returns the number of days in the given month

int **mcal_days_in_month** (int month, int leap year) \linebreak

mcal_days_in_month() Returns the number of days in the given month, taking into account if the given year is a leap year or not.

mcal_delete_calendar (PHP 3>= 3.0.13, PHP 4)

Delete an MCAL calendar

string **mcal_delete_calendar** (int stream, string calendar) \linebreak

Deletes the calendar named *calendar*.

mcal_delete_event (PHP 3>= 3.0.13, PHP 4)

Delete an event from an MCAL calendar

int **mcal_delete_event** (int mcal_stream [, int event_id]) \linebreak

mcal_delete_event() deletes the calendar event specified by the event_id.

Returns TRUE.

mcal_event_add_attribute (PHP 3>= 3.0.15, PHP 4)

Adds an attribute and a value to the streams global event structure

void **mcal_event_add_attribute** (int stream, string attribute, string value) \linebreak

mcal_event_add_attribute() adds an attribute to the stream's global event structure with the value given by "value".

mcal_event_init (PHP 3>= 3.0.13, PHP 4)

Initializes a streams global event structure

int **mcal_event_init** (int stream) \linebreak

mcal_event_init() initializes a streams global event structure. this effectively sets all elements of the structure to 0, or the default settings.

Returns TRUE.

mcal_event_set_alarm (PHP 3>= 3.0.13, PHP 4)

Sets the alarm of the streams global event structure

int **mcal_event_set_alarm** (int stream, int alarm) \linebreak

mcal_event_set_alarm() sets the streams global event structure's alarm to the given minutes before the event.

Returns TRUE.

mcal_event_set_category (PHP 3>= 3.0.13, PHP 4)

Sets the category of the streams global event structure

int **mcal_event_set_category** (int stream, string category) \linebreak

mcal_event_set_category() sets the streams global event structure's category to the given string.

Returns TRUE.

mcal_event_set_class (PHP 3>= 3.0.13, PHP 4)

Sets the class of the streams global event structure

int **mcal_event_set_class** (int stream, int class) \linebreak

mcal_event_set_class() sets the streams global event structure's class to the given value. The class is either 1 for public, or 0 for private.

Returns TRUE.

mcal_event_set_description (PHP 3>= 3.0.13, PHP 4)

Sets the description of the streams global event structure

int **mcal_event_set_description** (int stream, string description) \linebreak

mcal_event_set_description() sets the streams global event structure's description to the given string.

Returns TRUE.

mcal_event_set_end (PHP 3>= 3.0.13, PHP 4)

Sets the end date and time of the streams global event structure

int **mcal_event_set_end** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]]) \linebreak

mcal_event_set_end() sets the streams global event structure's end date and time to the given values.

Returns TRUE.

mcal_event_set_recur_daily (PHP 3>= 3.0.13, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_daily** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_daily() sets the streams global event structure's recurrence to the given value to be reoccuring on a daily basis, ending at the given date.

mcal_event_set_recur_monthly_mday (PHP 3>= 3.0.13, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_monthly_mday** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_monthly_mday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by month day basis, ending at the given date.

mcal_event_set_recur_monthly_wday (PHP 3>= 3.0.13, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_monthly_wday** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_monthly_wday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by week basis, ending at the given date.

mcal_event_set_recur_none (PHP 3>= 3.0.15, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_none** (int stream) \linebreak

mcal_event_set_recur_none() sets the streams global event structure to not recur (event->recur_type is set to `MCAL_RECUR_NONE`).

mcal_event_set_recur_weekly (PHP 3>= 3.0.13, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_weekly** (int stream, int year, int month, int day, int interval, int weekdays) \linebreak

mcal_event_set_recur_weekly() sets the streams global event structure's recurrence to the given value to be reoccuring on a weekly basis, ending at the given date.

mcal_event_set_recur_yearly (PHP 3>= 3.0.13, PHP 4)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_yearly** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_yearly() sets the streams global event structure's recurrence to the given value to be reoccuring on a yearly basis,ending at the given date.

mcal_event_set_start (PHP 3>= 3.0.13, PHP 4)

Sets the start date and time of the streams global event structure

int **mcal_event_set_start** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]]) \linebreak

mcal_event_set_start() sets the streams global event structure's start date and time to the given values.

Returns TRUE.

mcal_event_set_title (PHP 3>= 3.0.13, PHP 4)

Sets the title of the streams global event structure

int **mcal_event_set_title** (int stream, string title) \linebreak

mcal_event_set_title() sets the streams global event structure's title to the given string.

Returns TRUE.

mcal_expunge (unknown)

Deletes all events marked for being expunged.

int **mcal_expunge** (int stream) \linebreak

mcal_expunge() Deletes all events which have been previously marked for deletion.

mcal_fetch_current_stream_event (PHP 3>= 3.0.13, PHP 4)

Returns an object containing the current streams event structure

object **mcal_fetch_current_stream_event** (int stream) \linebreak

mcal_fetch_current_stream_event() returns the current stream's event structure as an object containing:

- int id - ID of that event.
- int public - TRUE if the event if public, FALSE if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year

- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

mcal_fetch_event (PHP 3>= 3.0.13, PHP 4)

Fetches an event from the calendar stream

object **mcal_fetch_event** (int mcal_stream, int event_id [, int options]) \linebreak

mcal_fetch_event() fetches an event from the calendar stream specified by *id*.

Returns an event object consisting of:

- int id - ID of that event.
- int public - TRUE if the event if public, FALSE if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

The possible values for recur_type are:

- 0 - Indicates that this event does not recur
- 1 - This event recurs daily
- 2 - This event recurs on a weekly basis
- 3 - This event recurs monthly on a specific day of the month (e.g. the 10th of the month)
- 4 - This event recurs monthly on a sequenced day of the week (e.g. the 3rd Saturday)
- 5 - This event recurs on an annual basis

mcal_is_leap_year (PHP 3>= 3.0.13, PHP 4)

Returns if the given year is a leap year or not

int **mcal_is_leap_year** (int year) \linebreak

mcal_is_leap_year() returns 1 if the given year is a leap year, 0 if not.

mcal_list_alarms (PHP 3>= 3.0.13, PHP 4)

Return a list of events that has an alarm triggered at the given datetime

array **mcal_list_alarms** (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]]) \linebreak

Returns an array of event ID's that has an alarm going off between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_list_events (PHP 3>= 3.0.13, PHP 4)

Return a list of IDs for a date or a range of dates.

array **mcal_list_events** (int mcal_stream, object begin_date [, object end_date]) \linebreak

Returns an array of ID's that are between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an beginning date and an optional end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_next_recurrence (PHP 3>= 3.0.13, PHP 4)

Returns the next recurrence of the event

int **mcal_next_recurrence** (int stream, int weekstart, array next) \linebreak

mcal_next_recurrence() returns an object filled with the next date the event occurs, on or after the supplied date. Returns empty date field if event does not occur or something is invalid. Uses weekstart to determine what day is considered the beginning of the week.

mcal_open (PHP 3>= 3.0.13, PHP 4)

Opens up an MCAL connection

int **mcal_open** (string calendar, string username, string password [, int options]) \linebreak

Returns an MCAL stream on success, FALSE on error.

mcal_open() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_popen (PHP 3>= 3.0.13, PHP 4)

Opens up a persistent MCAL connection

int **mcal_popen** (string calendar, string username, string password [, int options]) \linebreak

Returns an MCAL stream on success, FALSE on error.

mcal_popen() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_rename_calendar (PHP 3>= 3.0.13, PHP 4)

Rename an MCAL calendar

string **mcal_rename_calendar** (int stream, string old_name, string new_name) \linebreak

Renames the calendar *old_name* to *new_name*.

mcal_reopen (PHP 3>= 3.0.13, PHP 4)

Reopens an MCAL connection

int **mcal_reopen** (string calendar [, int options]) \linebreak

Reopens an MCAL stream to a new calendar.

mcal_reopen() reopens an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also.

mcal_snooze (PHP 3>= 3.0.13, PHP 4)

Turn off an alarm for an event

int **mcal_snooze** (int id) \linebreak

mcal_snooze() turns off an alarm for a calendar event specified by the id.

Returns TRUE.

mcal_store_event (PHP 3>= 3.0.13, PHP 4)

Modify an existing event in an MCAL calendar

int **mcal_store_event** (int mcal_stream) \linebreak

mcal_store_event() Stores the modifications to the current global event for the given stream.

Returns the event id of the modified event on success and FALSE on error.

mcal_time_valid (PHP 3>= 3.0.13, PHP 4)

Returns TRUE if the given year, month, day is a valid time

int **mcal_time_valid** (int hour, int minutes, int seconds) \linebreak

mcal_time_valid() Returns TRUE if the given hour, minutes and seconds is a valid time, FALSE if not.

mcal_week_of_year (PHP 4)

Returns the week number of the given date

int **mcal_week_of_year** (int day, int month, int year) \linebreak

LV. Mcrypt Encryption Functions

This is an interface to the mcrypt library, which supports a wide variety of block algorithms such as DES, TripleDES, Blowfish (default), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 and GOST in CBC, OFB, CFB and ECB cipher modes. Additionally, it supports RC6 and IDEA which are considered "non-free".

Mcrypt can be used to encrypt and decrypt using the above mentioned ciphers. If you linked against libmcrypt-2.2.x, the four important mcrypt commands (mcrypt_cfb(), mcrypt_cbc(), mcrypt_ecb(), and mcrypt_ofb()) can operate in both modes which are named MCRYPT_ENCRYPT and MCRYPT_DECRYPT, respectively.

Példa 1. Encrypt an input value with TripleDES under 2.2.x in ECB mode

```
<?php
$key = "this is a very secret key";
$input = "Let us meet at 9 o’clock at the secret place.";

$encrypted_data = mcrypt_ecb (MCRYPT_3DES, $key, $input, MCRYPT_ENCRYPT);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`.

If you linked against libmcrypt 2.4.x or 2.5.x, these functions are still available, but it is recommended that you use the advanced functions.

Példa 2. Encrypt an input value with TripleDES under 2.4.x and higher in ECB mode

```
<?php
    $key = "this is a very secret key";
    $input = "Let us meet at 9 o’clock at the secret place.";

    $td = mcrypt_module_open (’tripledes’, ”, ’ecb’, ”);
    $iv = mcrypt_create_iv (mcrypt_enc_get_iv_size ($td), MCRYPT_RAND);
    mcrypt_generic_init ($td, $key, $iv);
    $encrypted_data = mcrypt_generic ($td, $input);
    mcrypt_generic_end ($td);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`. For a full example see mcrypt_module_open().

Requirements

These functions work using mcrypt (http://mcrypt.hellug.gr/).

If you linked against libmcrypt 2.4.x or higher, the following additional block algorithms are supported: CAST, LOKI97, RIJNDAEL, SAFERPLUS, SERPENT and the following stream

ciphers: ENIGMA (crypt), PANAMA, RC4 and WAKE. With libmcrypt 2.4.x or higher another cipher mode is also available; nOFB.

Installation

To use it, download libmcrypt-x.x.tar.gz from here (http://mcrypt.hellug.gr/) and follow the included installation instructions. You need to compile PHP with the `--with-mcrypt` parameter to enable this extension. Make sure you compile libmcrypt with the option `--disable-posix-threads`.

Runtime Configuration

Resource types

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Predefined constants

Mcrypt can operate in four block cipher modes (CBC, OFB, CFB, and ECB). If linked against libmcrypt-2.4.x or higher the functions can also operate in the block cipher mode nOFB and in STREAM mode. Below you find a list with all supported encryption modes together with the constants that are defines for the encryption mode. For a more complete reference and discussion see Applied Cryptography by Schneier (ISBN 0-471-11709-9).

- MCRYPT_MODE_ECB (electronic codebook) is suitable for random data, such as encrypting other keys. Since data there is short and random, the disadvantages of ECB have a favorable negative effect.
- MCRYPT_MODE_CBC (cipher block chaining) is especially suitable for encrypting files where the security is increased over ECB significantly.
- MCRYPT_MODE_CFB (cipher feedback) is the best mode for encrypting byte streams where single bytes must be encrypted.
- MCRYPT_MODE_OFB (output feedback, in 8bit) is comparable to CFB, but can be used in applications where error propagation cannot be tolerated. It’s insecure (because it operates in 8bit mode) so it is not recommended to use it.
- MCRYPT_MODE_NOFB (output feedback, in nbit) is comparable to OFB, but more secure because it operates on the block size of the algorithm.
- MCRYPT_MODE_STREAM is an extra mode to include some stream algorithms like WAKE or RC4.

Here is a list of ciphers which are currently supported by the mcrypt extension. For a complete list of supported ciphers, see the defines at the end of `mcrypt.h`. The general rule with the mcrypt-2.2.x API is that you can access the cipher from PHP with MCRYPT_ciphername. With the

libmcrypt-2.4.x and libmcrypt-2.5.x API these constants also work, but it is possible to specify the

name of the cipher as a string with a call to mcrypt_module_open().

- MCRYPT_3DES
- MCRYPT_ARCFOUR_IV (libmcrypt > 2.4.x only)
- MCRYPT_ARCFOUR (libmcrypt > 2.4.x only)
- MCRYPT_BLOWFISH
- MCRYPT_CAST_128
- MCRYPT_CAST_256
- MCRYPT_CRYPT
- MCRYPT_DES
- MCRYPT_DES_COMPAT (libmcrypt 2.2.x only)
- MCRYPT_ENIGMA (libmcrypt > 2.4.x only, alias for MCRYPT_CRYPT)
- MCRYPT_GOST
- MCRYPT_IDEA (non-free)
- MCRYPT_LOKI97 (libmcrypt > 2.4.x only)
- MCRYPT_MARS (libmcrypt > 2.4.x only, non-free)
- MCRYPT_PANAMA (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_128 (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_192 (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_256 (libmcrypt > 2.4.x only)
- MCRYPT_RC2
- MCRYPT_RC4 (libmcrypt 2.2.x only)
- MCRYPT_RC6 (libmcrypt > 2.4.x only)
- MCRYPT_RC6_128 (libmcrypt 2.2.x only)
- MCRYPT_RC6_192 (libmcrypt 2.2.x only)
- MCRYPT_RC6_256 (libmcrypt 2.2.x only)
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_SAFERPLUS (libmcrypt > 2.4.x only)
- MCRYPT_SERPENT(libmcrypt > 2.4.x only)
- MCRYPT_SERPENT_128 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_192 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_256 (libmcrypt 2.2.x only)
- MCRYPT_SKIPJACK (libmcrypt > 2.4.x only)
- MCRYPT_TEAN (libmcrypt 2.2.x only)
- MCRYPT_THREEWAY
- MCRYPT_TRIPLEDES (libmcrypt > 2.4.x only)
- MCRYPT_TWOFISH (for older mcrypt 2.x versions, or mcrypt > 2.4.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx are available in newer 2.x versions, but not in the 2.4.x

versions)

- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_WAKE (libmcrypt > 2.4.x only)
- MCRYPT_XTEA (libmcrypt > 2.4.x only)

You must (in CFB and OFB mode) or can (in CBC mode) supply an initialization vector (IV) to the respective cipher function. The IV must be unique and must be the same when decrypting/encrypting. With data which is stored encrypted, you can take the output of a function of the index under which the data is stored (e.g. the MD5 key of the filename). Alternatively, you can transmit the IV together with the encrypted data (see chapter 9.3 of Applied Cryptography by Schneier (ISBN 0-471-11709-9) for a discussion of this topic).

mcrypt_cbc (PHP 3>= 3.0.8, PHP 4)

Encrypt/decrypt data in CBC mode

string **mcrypt_cbc** (int cipher, string key, string data, int mode [, string iv]) \linebreak string **mcrypt_cbc** (string cipher, string key, string data, int mode [, string iv]) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mcrypt_cfb (PHP 3>= 3.0.8, PHP 4)

Encrypt/decrypt data in CFB mode

string **mcrypt_cfb** (int cipher, string key, string data, int mode, string iv) \linebreak string **mcrypt_cfb** (string cipher, string key, string data, int mode [, string iv]) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mcrypt_create_iv (PHP 3>= 3.0.8, PHP 4)

Create an initialization vector (IV) from a random source

string **mcrypt_create_iv** (int size, int source) \linebreak

mcrypt_create_iv() is used to create an IV.

mcrypt_create_iv() takes two arguments, *`size`* determines the size of the IV, *`source`* specifies the source of the IV.

The source can be MCRYPT_RAND (system random number generator), MCRYPT_DEV_RANDOM (read data from /dev/random) and MCRYPT_DEV_URANDOM (read data from /dev/urandom). If you use MCRYPT_RAND, make sure to call srand() before to initialize the random number generator.

Példa 1. mcrypt_create_iv() example

```
<?php
    $size = mcrypt_get_iv_size (MCRYPT_CAST_256, MCRYPT_MODE_CFB);
    $iv = mcrypt_create_iv ($size, MCRYPT_DEV_RANDOM);
?>
```

The IV is only meant to give an alternative seed to the encryption routines. This IV does not need to be secret at all, though it can be desirable. You even can send it along with your ciphertext without loosing security.

More information can be found at http://www.ciphersbyritter.com/GLOSSARY.HTM#IV, http://fn2.freenet.edmonton.ab.ca/~jsavard/crypto/co0409.htm and in chapter 9.3 of Applied Cryptography by Schneier (ISBN 0-471-11709-9) for a discussion of this topic.

mcrypt_decrypt (PHP 4 >= 4.0.2)

Decrypts crypttext with given parameters

string **mcrypt_decrypt** (string cipher, string key, string data, string mode [, string iv]) \linebreak

mcrypt_decrypt() decrypts the data and returns the unencrypted data.

`Cipher` is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

`Key` is the key with which the data is encrypted. If it's smaller that the required keysize, it is padded with '`\0`'.

`Data` is the data that will be decrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '`\0`'.

`Mode` is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *`IV`* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '`\0`'.

mcrypt_ecb (PHP 3>= 3.0.8, PHP 4)

Encrypt/decrypt data in ECB mode

string **mcrypt_ecb** (int cipher, string key, string data, int mode) \linebreak string **mcrypt_ecb** (string cipher, string key, string data, int mode [, string iv]) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mcrypt_enc_get_algorithms_name (PHP 4 >= 4.0.2)

Returns the name of the opened algorithm

string **mcrypt_enc_get_algorithms_name** (resource td) \linebreak

This function returns the name of the algorithm.

Példa 1. mcrypt_enc_get_algorithms_name() example

```
<?php
    $td = mcrypt_module_open (MCRYPT_CAST_256, ”, MCRYPT_MODE_CFB, ”);
    echo mcrypt_enc_get_algorithms_name($td). "\n";

    $td = mcrypt_module_open (’cast-256’, ”, MCRYPT_MODE_CFB, ”);
    echo mcrypt_enc_get_algorithms_name($td). "\n";
?>

Prints:
CAST-256
CAST-256
```

mcrypt_enc_get_block_size (PHP 4 >= 4.0.2)

Returns the blocksize of the opened algorithm

int **mcrypt_enc_get_block_size** (resource td) \linebreak

This function returns the block size of the algorithm specified by the encryption descriptor *td* in bytes.

mcrypt_enc_get_iv_size (PHP 4 >= 4.0.2)

Returns the size of the IV of the opened algorithm

int **mcrypt_enc_get_iv_size** (resource td) \linebreak

This function returns the size of the iv of the algorithm specified by the encryption descriptor in bytes. If it returns ’0’ then the IV is ignored in the algorithm. An IV is used in cbc, cfb and ofb modes, and in some algorithms in stream mode.

mcrypt_enc_get_key_size (PHP 4 >= 4.0.2)

Returns the maximum supported keysize of the opened mode

int **mcrypt_enc_get_key_size** (resource td) \linebreak

This function returns the maximum supported key size of the algorithm specified by the encryption descriptor *td* in bytes.

mcrypt_enc_get_modes_name (PHP 4 >= 4.0.2)

Returns the name of the opened mode

string **mcrypt_enc_get_modes_name** (resource td) \linebreak

This function returns the name of the mode.

Példa 1. mcrypt_enc_get_modes_name() example

```
<?php
    $td = mcrypt_module_open (MCRYPT_CAST_256, ”, MCRYPT_MODE_CFB, ”);
    echo mcrypt_enc_get_modes_name($td). "\n";

    $td = mcrypt_module_open (’cast-256’, ”, ’ecb’, ”);
    echo mcrypt_enc_get_modes_name($td). "\n";
?>

Prints:
CFB
ECB
```

mcrypt_enc_get_supported_key_sizes (PHP 4 >= 4.0.2)

Returns an array with the supported keysizes of the opened algorithm

array **mcrypt_enc_get_supported_key_sizes** (resource td) \linebreak

Returns an array with the key sizes supported by the algorithm specified by the encryption descriptor. If it returns an empty array then all key sizes between 1 and mcrypt_enc_get_key_size() are supported by the algorithm.

Példa 1. mcrypt_enc_get_supported_key_sizes() example

```
<?php
    $td = mcrypt_module_open (’rijndael-256’, ”, ’ecb’, ”);
    var_dump (mcrypt_enc_get_supported_key_sizes($td));
?>

This will print:

array(3) {
  [0]=>
  int(16)
  [1]=>
  int(24)
  [2]=>
```

```
  int(32)
}
?>
```

mcrypt_enc_is_block_algorithm_mode (PHP 4 >= 4.0.2)

Checks whether the encryption of the opened mode works on blocks

bool **mcrypt_enc_is_block_algorithm_mode** (resource td) \linebreak

This function returns TRUE if the mode is for use with block algorithms, otherwise it returns FALSE. (eg. FALSE for stream, and TRUE for cbc, cfb, ofb).

mcrypt_enc_is_block_algorithm (PHP 4 >= 4.0.2)

Checks whether the algorithm of the opened mode is a block algorithm

bool **mcrypt_enc_is_block_algorithm** (resource td) \linebreak

This function returns TRUE if the algorithm is a block algorithm, or FALSE if it is a stream algorithm.

mcrypt_enc_is_block_mode (PHP 4 >= 4.0.2)

Checks whether the opened mode outputs blocks

bool **mcrypt_enc_is_block_mode** (resource td) \linebreak

This function returns TRUE if the mode outputs blocks of bytes or FALSE if it outputs bytes. (eg. TRUE for cbc and ecb, and FALSE for cfb and stream).

mcrypt_enc_self_test (PHP 4 >= 4.0.2)

This function runs a self test on the opened module

bool **mcrypt_enc_self_test** (resource td) \linebreak

This function runs the self test on the algorithm specified by the descriptor *td*. If the self test succeeds it returns FALSE. In case of an error, it returns TRUE.

mcrypt_encrypt (PHP 4 >= 4.0.2)

Encrypts plaintext with given parameters

string **mcrypt_encrypt** (string cipher, string key, string data, string mode [, string iv]) \linebreak

mcrypt_encrypt() encrypts the data and returns the encrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data will be encrypted. If it's smaller that the required keysize, it is padded with '\0'. It is better not to use ASCII strings for keys. It is recommended to use the mhash functions to create a key from a string.

Data is the data that will be encrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '\0'. The returned crypttext can be larger that the size of the data that is given by *data*.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '\0'.

Példa 1. mcrypt_encrypt() Example

```
<?php
    $iv = mcrypt_create_iv (mcrypt_get_iv_size (MCRYPT_RIJNDAEL_256, MCRYPT_MODE_ECB), MC
    $key = "This is a very secret key";
    $text = "Meet me at 11 o'clock behind the monument.";
    echo strlen ($text)."\n";

    $crypttext = mcrypt_encrypt (MCRYPT_RIJNDAEL_256, $key, $text, MCRYPT_MODE_ECB, $iv);
    echo strlen ($crypttext)."\n";
?>
```

The above example will print out:

```
42
64
```

See also mcrypt_module_open() for a more advanced API and an example.

mcrypt_generic_deinit (PHP 4 >= 4.1.1)

This function deinitializes an encryption module

bool **mcrypt_generic_deinit** (resource td) \linebreak

This function terminates encryption specified by the encryption descriptor (*td*). It clears all buffers, but does not close the module. You need to call mcrypt_module_close() yourself. (But PHP does this for you at the end of the script. Returns FALSE on error, or TRUE on succes.

See for an example mcrypt_module_open() and the entry on mcrypt_generic_init().

mcrypt_generic_end (PHP 4 >= 4.0.2)

This function terminates encryption

bool **mcrypt_generic_end** (resource td) \linebreak

This function is deprecated, use mcrypt_generic_deinit() instead. It can cause crashes when used with mcrypt_module_close() due to multiple buffer frees.

This function terminates encryption specified by the encryption descriptor (*td*). Actually it clears all buffers, and closes all the modules used. Returns FALSE on error, or TRUE on succes.

mcrypt_generic_init (PHP 4 >= 4.0.2)

This function initializes all buffers needed for encryption

int **mcrypt_generic_init** (resource td, string key, string iv) \linebreak

The maximum length of the key should be the one obtained by calling mcrypt_enc_get_key_size() and every value smaller than this is legal. The IV should normally have the size of the algorithms block size, but you must obtain the size by calling mcrypt_enc_get_iv_size(). IV is ignored in ECB. IV MUST exist in CFB, CBC, STREAM, nOFB and OFB modes. It needs to be random and unique (but not secret). The same IV must be used for encryption/decryption. If you do not want to use it you should set it to zeros, but this is not recommended. The function returns a negative value on error.

You need to call this function before every call to mcrypt_generic() or mdecrypt_generic().

See for an example mcrypt_module_open() and the entry on mcrypt_generic_deinit().

mcrypt_generic (PHP 4 >= 4.0.2)

This function encrypts data

string **mcrypt_generic** (resource td, string data) \linebreak

This function encrypts data. The data is padded with "\0" to make sure the length of the data is n * blocksize. This function returns the encrypted data. Note that the length of the returned string can in fact be longer then the input, due to the padding of the data.

The encryption handle should alwayws be initialized with mcrypt_generic_init() with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling mcrypt_generic_deinit(). See mcrypt_module_open() for an example.

See also mdecrypt_generic(), mcrypt_generic_init() and mcrypt_generic_deinit().

mcrypt_get_block_size (PHP 3>= 3.0.8, PHP 4)

Get the block size of the specified cipher

int **mcrypt_get_block_size** (int cipher) \linebreak int **mcrypt_get_block_size** (string cipher, string module) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or 2.5.x.

mcrypt_get_block_size() is used to get the size of a block of the specified *`cipher`* (in combination with an encryption mode).

This example shows how to use this function when linked against libmcrypt 2.4.x and 2.5.x.

Példa 1. mcrypt_get_block_size() example

```
<?php
    echo mcrypt_get_block_size (’tripledes’, ’ecb’);
?>

Prints:
8
```

See also: mcrypt_get_key_size() and mcrypt_encrypt().

mcrypt_get_cipher_name (PHP 3>= 3.0.8, PHP 4)

Get the name of the specified cipher

string **mcrypt_get_cipher_name** (int cipher) \linebreak string **mcrypt_get_cipher_name** (string cipher) \linebreak

mcrypt_get_cipher_name() is used to get the name of the specified cipher.

mcrypt_get_cipher_name() takes the cipher number as an argument (libmcrypt 2.2.x) or takes the cipher name as an argument (libmcrypt 2.4.x or higher) and returns the name of the cipher or `FALSE`, if the cipher does not exist.

Példa 1. mcrypt_get_cipher_name() Example

```
<?php
   $cipher = MCRYPT_TripleDES;
```

```
   echo mcrypt_get_cipher_name ($cipher);
?>
```

The above example will produce:

```
3DES
```

mcrypt_get_iv_size (PHP 4 >= 4.0.2)

Returns the size of the IV belonging to a specific cipher/mode combination

int **mcrypt_get_iv_size** (resource td) \linebreak int **mcrypt_get_iv_size** (string cipher, string mode) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

mcrypt_get_iv_size() returns the size of the Initialisation Vector (IV) in bytes. On error the function returns `FALSE`. If the IV is ignored in the specified cipher/mode combination zero is returned.

`cipher` is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

`mode` is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

`td` is the resource that is returned by mcrypt_module_open().

Példa 1. mcrypt_create_iv() example

```
<?php
    $size = mcrypt_get_iv_size (MCRYPT_CAST_256, MCRYPT_MODE_CFB);

    $size = mcrypt_get_iv_size (’des’, ’ecb’);
?>
```

See also: mcrypt_create_iv()

mcrypt_get_key_size (PHP 3>= 3.0.8, PHP 4)

Get the key size of the specified cipher

int **mcrypt_get_key_size** (int cipher) \linebreak int **mcrypt_get_key_size** (string cipher, string module) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or 2.5.x.

mcrypt_get_key_size() is used to get the size of a key of the specified *`cipher`* (in combination with an encryption mode).

This example shows how to use this function when linked against libmcrypt 2.4.x and 2.5.x.

Példa 1. mcrypt_get_block_size() example

```
<?php
    echo mcrypt_get_key_size (’tripledes’, ’ecb’);
?>

Prints:
24
```

See also: mcrypt_get_block_size() and mcrypt_encrypt().

mcrypt_list_algorithms (PHP 4 >= 4.0.2)

Get an array of all supported ciphers

array **mcrypt_list_algorithms** ([string lib_dir]) \linebreak

mcrypt_list_algorithms() is used to get an array of all supported algorithms in the *`lib_dir`* parameter.

mcrypt_list_algorithms() takes an optional *`lib_dir`* parameter which specifies the directory where all algorithms are located. If not specifies, the value of the mcrypt.algorithms_dir `php.ini` directive is used.

Példa 1. mcrypt_list_algorithms() Example

```
<?php
    $algorithms = mcrypt_list_algorithms ("/usr/local/lib/libmcrypt");

    foreach ($algorithms as $cipher) {
        echo "$cipher<br />\n";
    }
?>
```

The above example will produce a list with all supported algorithms in the "/usr/local/lib/libmcrypt" directory.

mcrypt_list_modes (PHP 4 >= 4.0.2)

Get an array of all supported modes

array **mcrypt_list_modes** ([string lib_dir]) \linebreak

mcrypt_list_modes() is used to get an array of all supported modes in the *lib_dir*.

mcrypt_list_modes() takes as optional parameter a directory which specifies the directory where all modes are located. If not specifies, the value of the mcrypt.modes_dir `php.ini` directive is used.

Példa 1. mcrypt_list_modes() Example

```
<?php
    $modes = mcrypt_list_modes ();

    foreach ($modes as $mode) {
        echo "$mode <br />\n";
    }
?>
```

The above example will produce a list with all supported algorithms in the default mode directory. If it is not set with the ini directive mcrypt.modes_dir, the default directory of mcrypt is used (which is /usr/local/lib/libmcrypt).

mcrypt_module_close (PHP 4 >= 4.0.2)

Close the mcrypt module

bool **mcrypt_module_close** (resource td) \linebreak

This function closes the specified encryption handle.

See mcrypt_module_open() for an example.

mcrypt_module_get_algo_block_size (PHP 4 >= 4.0.2)

Returns the blocksize of the specified algorithm

int **mcrypt_module_get_algo_block_size** (string algorithm [, string lib_dir]) \linebreak

This function returns the block size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_algo_key_size (PHP 4 >= 4.0.2)

Returns the maximum supported keysize of the opened mode

int **mcrypt_module_get_algo_key_size** (string algorithm [, string lib_dir]) \linebreak

This function returns the maximum supported key size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_supported_key_sizes (PHP 4 >= 4.0.2)

Returns an array with the supported keysizes of the opened algorithm

array **mcrypt_module_get_supported_key_sizes** (string algorithm [, string lib_dir]) \linebreak

Returns an array with the key sizes supported by the specified algorithm. If it returns an empty array then all key sizes between 1 and mcrypt_module_get_algo_key_size() are supported by the algorithm. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

See also mcrypt_enc_get_supported_key_sizes() which is used on open encryption modules.

mcrypt_module_is_block_algorithm_mode (PHP 4 >= 4.0.2)

This function returns if the the specified module is a block algorithm or not

bool **mcrypt_module_is_block_algorithm_mode** (string mode [, string lib_dir]) \linebreak

This function returns TRUE if the mode is for use with block algorithms, otherwise it returns FALSE. (eg. FALSE for stream, and TRUE for cbc, cfb, ofb). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_is_block_algorithm (PHP 4 >= 4.0.2)

This function checks whether the specified algorithm is a block algorithm

bool **mcrypt_module_is_block_algorithm** (string algorithm [, string lib_dir]) \linebreak

This function returns TRUE if the specified algorithm is a block algorithm, or FALSE is it is a stream algorithm. The optional *lib_dir* parameter can contain the location where the algorithm module is on the system.

mcrypt_module_is_block_mode (PHP 4 >= 4.0.2)

This function returns if the the specified mode outputs blocks or not

bool **mcrypt_module_is_block_mode** (string mode [, string lib_dir]) \linebreak

This function returns `TRUE` if the mode outputs blocks of bytes or `FALSE` if it outputs just bytes. (eg. `TRUE` for cbc and ecb, and `FALSE` for cfb and stream). The optional *`lib_dir`* parameter can contain the location where the mode module is on the system.

mcrypt_module_open (PHP 4 >= 4.0.2)

This function opens the module of the algorithm and the mode to be used

resource **mcrypt_module_open** (string algorithm, string algorithm_directory, string mode, string mode_directory) \linebreak

This function opens the module of the algorithm and the mode to be used. The name of the algorithm is specified in algorithm, eg. "twofish" or is one of the MCRYPT_ciphername constants. The module is closed by calling mcrypt_module_close(). Normally it returns an encryption descriptor, or `FALSE` on error.

The *`algorithm_directory`* and *`mode_directory`* are used to locate the encryption modules. When you supply a directory name, it is used. When you set one of these to the empty string (""), the value set by the *`mcrypt.algorithms_dir`* or *`mcrypt.modes_dir`* ini-directive is used. When these are not set, the default directories that are used are the ones that were compiled in into libmcrypt (usally /usr/local/lib/libmcrypt).

Példa 1. mcrypt_module_open() Example

```
<?php
    $td = mcrypt_module_open (MCRYPT_DES, ”, MCRYPT_MODE_ECB, ’/usr/lib/mcrypt-
modes’);
    $td = mcrypt_module_open (’rijndael-256’, ”, ’ofb’, ”);
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory `/usr/lib/mcrypt-modes`. The second example uses strings as name for the cipher an dmode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

Példa 2. Using mcrypt_module_open() in encryption

```
<?php
    /* Open the cipher */
    $td = mcrypt_module_open (’rijndael-256’, ”, ’ofb’, ”);
```

```
    /* Create the IV and determine the keysize length */
    $iv = mcrypt_create_iv (mcrypt_enc_get_iv_size($td), MCRYPT_DEV_RANDOM);
    $ks = mcrypt_enc_get_key_size ($td);

    /* Create key */
    $key = substr (md5 ('very secret key'), 0, $ks);

    /* Intialize encryption */
    mcrypt_generic_init ($td, $key, $iv);

    /* Encrypt data */
    $encrypted = mcrypt_generic ($td, 'This is very important data');

    /* Terminate encryption handler */
    mcrypt_generic_deinit ($td);

    /* Initialize encryption module for decryption */
    mcrypt_generic_init ($td, $key, $iv);

    /* Decrypt encrypted string */
    $decrypted = mdecrypt_generic ($td, $encrypted);

    /* Terminate decryption handle and close module */
    mcrypt_generic_deinit ($td);
    mcrypt_module_close ($td);

    /* Show string */
    echo trim ($decrypted)."\n";
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory `/usr/lib/mcrypt-modes`. The second example uses strings as name for the cipher an dmode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

See also mcrypt_module_close(), mcrypt_generic(), mdecrypt_generic(), mcrypt_generic_init() and mcrypt_generic_deinit().

mcrypt_module_self_test (PHP 4 >= 4.0.2)

This function runs a self test on the specified module

bool **mcrypt_module_self_test** (string algorithm [, string lib_dir]) \linebreak

This function runs the self test on the algorithm specified. The optional *`lib_dir`* parameter can contain the location of where the algorithm module is on the system.

The function returns `TRUE` if the self test succeeds, or `FALSE` when if fails.

mcrypt_ofb (PHP 3>= 3.0.8, PHP 4)

Encrypt/decrypt data in OFB mode

string **mcrypt_ofb** (int cipher, string key, string data, int mode, string iv) \linebreak string **mcrypt_ofb** (string cipher, string key, string data, int mode [, string iv]) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mdecrypt_generic (PHP 4 >= 4.0.2)

This function decrypts data

string **mdecrypt_generic** (resource td, string data) \linebreak

This function decrypts data. Note that the length of the returned string can in fact be longer then the unencrypted string, due to the padding of the data.

Példa 1. mdecrypt_generic() Example

```
<?php
    /* Data */
    $key = 'this is a very long key, even too long for the cipher';
    $plain_text = 'very important data';

    /* Open module, and create IV */
    $td = mcrypt_module_open ('des', '', 'ecb', '');
    $key = substr ($key, 0, mcrypt_enc_get_key_size ($td));
    $iv_size = mcrypt_enc_get_iv_size ($td);
    $iv = mcrypt_create_iv ($iv_size, MCRYPT_RAND);

    /* Initialize encryption handle */
    if (mcrypt_generic_init ($td, $key, $iv) != -1) {

        /* Encrypt data */
        $c_t = mcrypt_generic ($td, $plain_text);
        mcrypt_generic_deinit ($td);

        /* Reinitialize buffers for decryption */
        mcrypt_generic_init ($td, $key, $iv);
        $p_t = mdecrypt_generic ($td, $c_t);

        /* Clean up */
        mcrypt_generic_deinit ($td);
        mcrypt_module_close ($td);
    }

    if (strncmp ($p_t, $plain_text, strlen($plain_text)) == 0) {
        echo "ok\n";
```

```
    } else {
        echo "error\n";
    }
?>
```

The above example shows how to check if the data before the encryption is the same as the data after the decryption. It is very important to reinitialize the encryption buffer with mcrypt_generic_init() before you try to decrypt the data.

The decryption handle should alwayws be initialized with mcrypt_generic_init() with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling mcrypt_generic_deinit(). See mcrypt_module_open() for an example.

See also mcrypt_generic(), mcrypt_generic_init() and mcrypt_generic_deinit().

LVI. Mhash függvények

Ezek a függvények az mhash (http://mhash.sourceforge.net/) eljáráskönyvtárral működnek együtt.

Ezek egy felületet biztosítanak az mhash könyvtárhoz. Az mhash széles skáláját támogatja a hash algoritmusoknak, mint például az MD5, SHA1, GOST és még számos más. Az mhash például ellenőrzőösszegek, kivonatok, azonosítási kódok előállítására is használható.

A használatukhoz le kell tölteni az mhash csomagot az mhash webhelyről (http://mhash.sourceforge.net/) és a csomagban található a telepítési utasítások szerint installálni kell. A PHP-t a `--with-mhash` paraméterrel kell fordítani, amivel engedélyezzük ezt a kiterjesztést.

Példa 1. Az MD5 kivonat és a hmac előállítása és kiírása hexa formában

```
<?php
$input = "mit kérsz a semmiért?";
$hash = mhash (MHASH_MD5, $input);
print "A 'hash' értéke: ".bin2hex ($hash)."\n<br />";
$hash = mhash (MHASH_MD5, $input, "Jefe");
print "A 'hmac' értéke: ".bin2hex ($hash)."\n<br />";
?>
```

Ez a következő kimenetet eredményezi:

```
A 'hash' értéke: 08dd459b941ee225cc6b33717f87d9a4
A 'hash' értéke: d6044d59c420fecb5e5e2910d2134129
```

A támogatott hash algoritmusok teljes listája az mhash dokumentációban található. Az általános szabály, hogy a hash algoritmust az MHASH_HASHNEVE formában lehet elérni PHP-ből, például a TIGER eléréséhez az MHASH_TIGER PHP konstans használható.

Itt következnek az mhash által jelenleg támogatott hash-ek. Az mhash dokumentációban szereplő, de itt fel nem soroltak is nyugodtan használhatók, ebben az esetben ez a dokumentáció már nem

tekinthető aktuálisnak.

- `MHASH_MD5`
- `MHASH_SHA1`
- `MHASH_HAVAL256`
- `MHASH_HAVAL192`
- `MHASH_HAVAL160`
- `MHASH_HAVAL128`
- `MHASH_RIPEMD160`
- `MHASH_GOST`
- `MHASH_TIGER`
- `MHASH_CRC32`
- `MHASH_CRC32B`

mhash_count (PHP 3>= 3.0.9, PHP 4)

Visszaadja a legnagyobb elérhető hash azonosítót

int **mhash_count** (void) \linebreak

Az **mhash_count()** visszaadja a legnagyobb elérhető hash azonosítót. A használható hash-ek azonosító számai nullától eddig a számig terjednek.

Példa 1. Az összes hash használata

```
<?php

$nr = mhash_count();

for ($i = 0; $i &lt;= $nr; $i++) {
    echo sprintf ("The blocksize of %s is %d\n",
        mhash_get_hash_name ($i),
        mhash_get_block_size ($i));
}
?>
```

mhash_get_block_size (PHP 3>= 3.0.9, PHP 4)

A megadott hash blokkméretével tér vissza

int **mhash_get_block_size** (int hash) \linebreak

Az **mhash_get_block_size()** a megadott *hash* blokkméretét adja vissza byte-okban mérve.

Ha az **mhash_get_block_size()** függvénynek érvénytelen *hash* paramétert adsz, FALSE értékkel tér vissza.

mhash_get_hash_name (PHP 3>= 3.0.9, PHP 4)

A megadott hash nevét adja vissza

string **mhash_get_hash_name** (int hash) \linebreak

Az **mhash_get_hash_name()** a megadott hash nevével tér vissza.

Ha az **mhash_get_hash_name()** függvénynek átadott hash azonosító egy nemlétező hash-re hivatkozik, FALSE értékkel tér vissza.

Példa 1. mhash_get_hash_name() példa

```
<?php
$hash = MHASH_MD5;

print mhash_get_hash_name ($hash);
?>
```

Ez a következő kimenetet eredményezi:

```
MD5
```

mhash_keygen_s2k (PHP 4 >= 4.0.4)

Kulcsot generál

string **mhash_keygen_s2k** (int hash, string password, string salt, int bytes) \linebreak

Az **mhash_keygen_s2k()** *bytes* méretű kulcsot generál a megadott *password* alapján. Az S2K algoritmust használja, amely az OpenPGP leírásban (RFC 2440) van definiálva. A megadott *hash* algoritmust használja a kulcs előállítására. A *salt* minden generált kulcsra más-más kell legyen, kellőképpen véletlenszerű értékekkel, hogy különböző kulcsok álljanak elő. A *salt*-nak ismertnek kell lennie a kulcsok ellenőrzésekor, ezért jó ötlet a kulcsokat hozzáfűzni ehhez. A *salt* mindig nyolc bájt hosszú legyen, ha ennél kevesebb, automatikusan nullákkal töltődik fel.

Figyelembe veendő mindenek előtt, hogy a felhasználók által beadott jelszavak nem túlságosan alkalmasak arra, hogy titkosítási rendszerek kulcsai legyenek, mivel a felhasználók általában jól begépelhető kulcsot választanak. Ezek a jelszavak csak 6-7 bitet (vagy még kevesebbet) használnak ki karakterenként. Nagyon ajánlott a beadott jelszóra olyan átalakítás használata, mint ez a függvény.

mhash (PHP 3>= 3.0.9, PHP 4)

Hash számítás

string **mhash** (int hash, string data [, string key]) \linebreak

Az **mhash()** alkalmazza a *hash* által megadott algoritmust a *data* paraméterre, és visszatér az eredmény hash-el (amit kivonatnak [digest-nek] is hívnak). A *key* paraméter megadásával visszaadja az keletkező HMAC-ot. A HMAC egy kulcsos hash-elés üzenet azonosításra, vagy csak egyszerűen egy üzenet kivonat, amely a megadott kulcstól függ. Nem minden mhash algoritmus használható HMAC módban. Hiba esetén a függvény FALSE értéket ad.

LVII. Mimetype Functions

Bevezetés

The functions in this module try to guess the content type and encoding of a file by looking for certain *magic* byte sequences at specific positions within the file. While this is not a bullet proof approach the heuristics used do a very good job.

This extension is derivated from Apache mod_mime_magic, which is itself based on the `file` command maintaind by Ian F. Darwin. See the source code for further historic and copyright information.

Követelmények

Az itt leírt függvények a standard modulban találhatóak, ami mindig rendelkezésre áll.

Telepítés

The extension needs a copy of the `magic.mime` as distributed with the `file` command. This file also part of most recent Linux distributions and usually stored in the `/usr/share/misc` directory.

Futásidejű beállítások

Táblázat 1. MIME magic Configuration options

Name	Default	Changeable
mime_magic.magicfile	"/usr/share/misc/magic.mime"	PHP_INI_SYSTEM

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előre definiált állandók

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

mime_content_type (PHP 4 CVS only)

Detect MIME Content-type for a file

string **mime_content_type** (string filename) \linebreak

Returns the MIME content type for a file as determined by using information from the `magic.mime` file. Content types are returned in MIME format, like `text/plain` or `application/octet-stream`.

LVIII. Microsoft SQL Server functions

The MSSQL extension is available on Win32 systems only. You can use the Sybase extension to connect to MSSQL databases from other platforms.

These functions allow you to access MS SQL Server database. The extension requires the MS SQL Client Tools to be installed on the system where PHP is installed. The Client Tools can be installed from the MS SQL Server CD or by copying ntwdblib.dll from \winnt\system32 on the server to \winnt\system32 on the PHP box. Copying ntwdblib.dll will only provide access. Configuration of the client will require installation of all the tools.

The MSSQL extension is enabled by adding extension=php_mssql.dll to `php.ini`.

mssql_bind (PHP 4 >= 4.1.0)

Adds a parameter to a stored procedure or a remote stored procedure

int **mssql_bind** (int stmt, string param_name, mixed var, int type [, int is_output [, int is_null [, int maxlen]]]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mssql_close (PHP 3, PHP 4)

Close MS SQL Server connection

int **mssql_close** ([int link_identifier]) \linebreak

Returns: TRUE on success, FALSE on error.

mssql_close() closes the link to a MS SQL Server database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

mssql_close() will not close persistent links generated by mssql_pconnect().

See also: mssql_connect(), mssql_pconnect().

mssql_connect (PHP 3, PHP 4)

Open MS SQL server connection

int **mssql_connect** ([string servername [, string username [, string password]]]) \linebreak

Returns: A positive MS SQL link identifier on success, or FALSE on error.

mssql_connect() establishes a connection to a MS SQL server. The servername argument has to be a valid servername that is defined in the 'interfaces' file.

In case a second call is made to **mssql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling mssql_close().

See also mssql_pconnect(), mssql_close().

mssql_data_seek (PHP 3, PHP 4)

Move internal row pointer

int **mssql_data_seek** (int result_identifier, int row_number) \linebreak

Returns: TRUE on success, FALSE on failure.

mssql_data_seek() moves the internal row pointer of the MS SQL result associated with the specified result identifier to point to the specified row number. The next call to mssql_fetch_row() would return that row.

See also: **mssql_data_seek()**.

mssql_execute (PHP 4 >= 4.1.0)

Executes a stored procedure on a MS SQL server database

int **mssql_execute** (int stmt) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mssql_fetch_array (PHP 3, PHP 4)

Fetch row as array

int **mssql_fetch_array** (int result) \linebreak

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

mssql_fetch_array() is an extended version of mssql_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using **mssql_fetch_array()** is NOT significantly slower than using mssql_fetch_row(), while it provides a significant added value.

For further details, also see mssql_fetch_row().

mssql_fetch_assoc (PHP 4 >= 4.2.0)

Returns an associative array of the current row in the result set specified by result_id

array **mssql_fetch_assoc** (int result_id [, int result_type]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mssql_fetch_batch (PHP 4 >= 4.0.4)

Returns the next batch of records

int **mssql_fetch_batch** (string result_index) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mssql_fetch_field (PHP 3, PHP 4)

Get field information

object **mssql_fetch_field** (int result [, int field_offset]) \linebreak

Returns an object containing field information.

mssql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **mssql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.
- column_source - the table from which the column was taken
- max_length - maximum length of the column
- numeric - 1 if the column is numeric

See also mssql_field_seek().

mssql_fetch_object (PHP 3)

Fetch row as object

int **mssql_fetch_object** (int result) \linebreak

Returns: An object with properties that correspond to the fetched row, or FALSE if there are no more rows.

mssql_fetch_object() is similar to mssql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to mssql_fetch_array(), and almost as quick as mssql_fetch_row() (the difference is insignificant).

See also: mssql_fetch-array() and mssql_fetch-row().

mssql_fetch_row (PHP 3, PHP 4)

Get row as enumerated array

array **mssql_fetch_row** (int result) \linebreak

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

mssql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **mssql_fetch_rows()** would return the next row in the result set, or FALSE if there are no more rows.

See also: mssql_fetch_array(), mssql_fetch_object(), mssql_data_seek(), **mssql_fetch_lengths()**, and mssql_result().

mssql_field_length (PHP 3>= 3.0.3, PHP 4)

Get the length of a field

int **mssql_field_length** (int result [, int offset]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mssql_field_name (PHP 3>= 3.0.3, PHP 4)

Get the name of a field

int **mssql_field_name** (int result [, int offset]) \linebreak

mssql_field_seek (PHP 3, PHP 4)

Set field offset

int **mssql_field_seek** (int result, int field_offset) \linebreak

Seeks to the specified field offset. If the next call to mssql_fetch_field() won't include a field offset, this field would be returned.

See also: mssql_fetch_field().

mssql_field_type (PHP 3>= 3.0.3, PHP 4)

Get the type of a field

string **mssql_field_type** (int result [, int offset]) \linebreak

mssql_free_result (PHP 3, PHP 4)

Free result memory

int **mssql_free_result** (int result) \linebreak

mssql_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script ends. You may call **mssql_free_result()** with the result identifier as an argument and the associated result memory will be freed.

mssql_get_last_message (PHP 3, PHP 4)

Returns the last message from server (over min_message_severity?)

string **mssql_get_last_message** (void) \linebreak

mssql_guid_string (PHP 4 >= 4.1.0)

Converts a 16 byte binary GUID to a string

string **mssql_guid_string** (string binary [, int short_format]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mssql_init (PHP 4 >= 4.1.0)

Initializes a stored procedure or a remote stored procedure

int **mssql_init** (string sp_name [, int conn_id]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mssql_min_error_severity (PHP 3, PHP 4)

Sets the lower error severity

void **mssql_min_error_severity** (int severity) \linebreak

mssql_min_message_severity (PHP 3, PHP 4)

Sets the lower message severity

void **mssql_min_message_severity** (int severity) \linebreak

mssql_next_result (PHP 4 >= 4.0.5)

Move the internal result pointer to the next result

bool **mssql_next_result** (int result_id) \linebreak

When sending more than one SQL statement to the server or executing a stored procedure with multiple results, it will cause the server to return multiple result sets. This function will test for additional results available form the server. If an additional result set exists it will free the existing result set and prepare to fetch the rows from the new result set. The function will return `TRUE` if an additional result set was available or `FALSE` otherwise.

Példa 1. mssql_next_result() example

```
<?php
    $link = mssql_connect ("localhost", "userid", "secret");
    mssql_select_db("MyDB", $link);
```

```
    $SQL = "Select * from table1 select * from table2";
    $rs = mssql_query($SQL, $link);
    do {
        while ($row = mssql_fetch_row($rs)) {
        }
    } while (mssql_next_result($rs));
    mssql_free_result($rs);
    mssql_close ($link);
?>
```

mssql_num_fields (PHP 3, PHP 4)

Get number of fields in result

int **mssql_num_fields** (int result) \linebreak

mssql_num_fields() returns the number of fields in a result set.

See also: **mssql_db_query()**, mssql_query(), mssql_fetch_field(), and mssql_num_rows().

mssql_num_rows (PHP 3, PHP 4)

Get number of rows in result

int **mssql_num_rows** (string result) \linebreak

mssql_num_rows() returns the number of rows in a result set.

See also: **mssql_db_query()**, mssql_query(), and mssql_fetch_row().

mssql_pconnect (PHP 3, PHP 4)

Open persistent MS SQL connection

int **mssql_pconnect** ([string servername [, string username [, string password]]]) \linebreak

Returns: A positive MS SQL persistent link identifier on success, or FALSE on error.

mssql_pconnect() acts very much like mssql_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (mssql_close() will not close links established by **mssql_pconnect()**).

This type of links is therefore called 'persistent'.

mssql_query (PHP 3, PHP 4)

Send MS SQL query

int **mssql_query** (string query [, int link_identifier]) \linebreak

Returns: A positive MS SQL result identifier on success, or FALSE on error.

mssql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if mssql_connect() was called, and use it.

See also: **mssql_db_query()**, mssql_select_db(), and mssql_connect().

mssql_result (PHP 3, PHP 4)

Get result data

int **mssql_result** (int result, int i, mixed field) \linebreak

mssql_result() returns the contents of one cell from a MS SQL result set. The field argument can be the field's offset, the field's name or the field's table dot field's name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), it uses the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **mssql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: mssql_fetch_row(), mssql_fetch_array(), and mssql_fetch_object().

mssql_rows_affected (PHP 4 >= 4.0.4)

Returns the number of records affected by the query

int **mssql_rows_affected** (int conn_id) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

mssql_select_db (PHP 3, PHP 4)

Select MS SQL database

int **mssql_select_db** (string database_name [, int link_identifier]) \linebreak

Returns: TRUE on success, FALSE on error

mssql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if mssql_connect() was called, and use it.

Every subsequent call to mssql_query() will be made on the active database.

See also: mssql_connect(), mssql_pconnect(), and mssql_query()

LIX. Ming függvénykönyvtár Flash mozik előállításához

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

Bevezetés

A Ming egy nyílt forráskódu (LGPL) függvénykönyvtár, amely segítségével SWF formátumú (Flash) mozikat tudsz létrehozni. A Ming függvényeivel nagyjából lefedi a Flash 4 összes képeségét, mint: alakzatok, színátmenetek, bitmap képek (png és jpeg formátum), alakváltozások, szövegek, gombok, akciók, moziklipek, mp3 aláfestőzenék (stream típus), és színtranszformációk. Az egyedüli hiányosság, hogy eseményekhez rendelt hangkeltést nem tudunk vele létrehozni.

A "Ming" nem mozaikszó.

Érdemes még most, mindenek előtt tisztázni: a különféle hosszokat, távolságokat, méreteket mind "twip" (twenty units per pixel) mértékegységben kell megadni. Ez annyit jelent, hogy egy eredeti méreteiben megjelenített Flash moziban egy képpont húsz twinpek felel meg. Így lehet csak pontos, jól nagyítható mozikat létrehozni.

A már korábban létező PHP/libswf modulhoz képest a Ming több előrelépést is jelent. Egyrészt a Ming nyílt forráskódu, így lehetőség van bárhol, bármilyen rendszeren annak használatára, ahol a forrásból fordítást végre tudod hajtani. A libswf ezzel szemben nem publikus forrással rendelkezik, és csak a jelentősebb platformokra fordítottak belőle bináris függvénykönyvtárt. Ezen platformok közt a Windows nincs jelen. A Ming ráadásul jóval emberközelibb módon kínálja fel a Flash generáláshoz szükséges eljárásokat, ezeket különféle objektumokon keresztül tudjuk hívni. A Ming jelenleg is fejlesztés és karbantartás alatt áll. Ha van olyan, amit a Ming nem tud, de szükségesnek érzed, nosza tudasd velünk ezt a következő címen: ming@opaque.net (mailto:ming@opaque.net).

A Ming modul a 4.0.5-ös php verziótól szerepel a hivatalos kiadásban.

Telepítés

A Ming használatbavételéhez először is le kell fordítanod, és telepítened azt. Letölthető forráskódot, és telepítési útmutatót találsz a Ming saját honlapján: http://www.opaque.net/ming/. Ugyanitt találhasz példákat, egy kissebb gyorstalpalót, és a legfrissebb Ming-kapcsolatos híreket.

Töltsd le a Ming forráskódjának tömörített állományát, csomagold ki. Lépj bele a Ming könyvtárba, majd a következp parancsokat add ki: make, majd ha ez végzett: make isntall.

Ezen műveletek során létrejön a `libming.so` bináris állomány és a megfelelő helyre is telepedik: `/usr/lib/`. Emellett egy ming.h állomány is helyet kap az `/usr/include/` könyvtárban. Ha

máshová szeretnéd telepíteni, akkor fordítás előtt módosítsd a `Makefile` állomány `PREFIX=` bejegyzését, hogy a telepítés helyét megjelöld magad.

php-be kövezett változat (unix)

```
mkdir <phpdir>/ext/ming
cp php_ext/* <phpdir>/ext/ming
cd <phpdir>
./buildconf
./configure --with-ming <egyéb konfig beállításaid>
```

Fordítsd és telepítsd szokás szerint a php-t, szükség esetén indítsd újra a webszervert.

előfordított állomány telepítése (unix)

Töltsd le a `php_ming.so.gz` állományt, csomagold ki, és tedd a php moduljait tároló könyvtárba. Ezt könnyen megtudhatod, ha kiadod a következő parancssori utasítást: **php-config --extension-dir**). Ezekután izlés szerint vagy beteszed a `php.ini` fájlodba a `extension=php_ming.so` sort, vagy minden olyan oldalon, ahol szükséges, magad töltöd be a modult a `dl('php_ming.so');` paranccsal.

A Ming használata

A Ming 13 új osztályt hoz be a PHP rendszerbe, ezek mind saját tulajdonságokkal és eljárásokkal rendelkeznek. A használatba vételükhöz érdemes tehát ismerni, hogy kezeli a PHP az objektum

orientált programozást.

- swfmovie().
- swfshape().
- swfdisplayitem().
- swfgradient().
- swfbitmap().
- swffill().
- swfmorph().
- swftext().
- swffont().
- swftextfield().
- swfsprite().
- swfbutton().
- swfaction().

ming_setcubicthreshold (PHP 4 >= 4.0.5)

Set cubic threshold (?)

void **ming_setcubicthreshold** (int threshold) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ming_setscale (PHP 4 >= 4.0.5)

Set scale (?)

void **ming_setscale** (int scale) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

ming_useswfversion (PHP 4 >= 4.2.0)

Use SWF version (?)

void **ming_useswfversion** (int version) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

SWFAction (PHP 4 >= 4.0.5)

Creates a new Action.

new **swfaction** (string script) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfaction() creates a new Action, and compiles the given script into an SWFAction object.

The script syntax is based on the C language, but with a lot taken out- the SWF bytecode machine is just too simpleminded to do a lot of things we might like. For instance, we can't implement function calls without a tremendous amount of hackery because the jump bytecode has a hardcoded offset value. No pushing your calling address to the stack and returning- every function would have to know exactly where to return to.

So what's left? The compiler recognises the following tokens:

- break
- for
- continue
- if
- else
- do
- while

There is no typed data; all values in the SWF action machine are stored as strings. The following functions can be used in expressions:

time()

Returns the number of milliseconds (?) elapsed since the movie started.

random(seed)

Returns a pseudo-random number in the range 0-seed.

length(expr)

Returns the length of the given expression.

int(number)

Returns the given number rounded down to the nearest integer.

concat(expr, expr)

Returns the concatenation of the given expressions.

ord(expr)

Returns the ASCII code for the given character

chr(num)

Returns the character for the given ASCII code

substr(string, location, length)

Returns the substring of length length at location location of the given string string.

Additionally, the following commands may be used:

duplicateClip(clip, name, depth)

Duplicate the named movie clip (aka sprite). The new movie clip has name name and is at depth depth.

removeClip(expr)

Removes the named movie clip.

trace(expr)

Write the given expression to the trace log. Doubtful that the browser plugin does anything with this.

startDrag(target, lock, [left, top, right, bottom])

Start dragging the movie clip target. The lock argument indicates whether to lock the mouse (?)- use 0 (`FALSE`) or 1 (`TRUE`). Optional parameters define a bounding area for the dragging.

stopDrag()

Stop dragging my heart around. And this movie clip, too.

callFrame(expr)

Call the named frame as a function.

getURL(url, target, [method])

Load the given URL into the named target. The target argument corresponds to HTML document targets (such as "_top" or "_blank"). The optional method argument can be POST or GET if you want to submit variables back to the server.

loadMovie(url, target)

Load the given URL into the named target. The target argument can be a frame name (I think), or one of the magical values "_level0" (replaces current movie) or "_level1" (loads new movie on top of current movie).

nextFrame()

Go to the next frame.

prevFrame()

Go to the last (or, rather, previous) frame.

play()

Start playing the movie.

stop()

Stop playing the movie.

toggleQuality()

Toggle between high and low quality.

stopSounds()

Stop playing all sounds.

gotoFrame(num)

Go to frame number num. Frame numbers start at 0.

gotoFrame(name)

Go to the frame named name. Which does a lot of good, since I haven't added frame labels yet.

setTarget(expr)

Sets the context for action. Or so they say- I really have no idea what this does.

And there's one weird extra thing. The expression frameLoaded(num) can be used in if statements and while loops to check if the given frame number has been loaded yet. Well, it's supposed to, anyway, but I've never tested it and I seriously doubt it actually works. You can just use /:framesLoaded instead.

Movie clips (all together now- aka sprites) have properties. You can read all of them (or can you?), you can set some of them, and here they are:

- x
- y
- xScale
- yScale
- currentFrame - (read-only)
- totalFrames - (read-only)
- alpha - transparency level
- visible - 1=on, 0=off (?)
- width - (read-only)
- height - (read-only)
- rotation
- target - (read-only) (???)
- framesLoaded - (read-only)
- name
- dropTarget - (read-only) (???)
- url - (read-only) (???)
- highQuality - 1=high, 0=low (?)
- focusRect - (???)
- soundBufTime - (???)

So, setting a sprite's x position is as simple as `/box.x = 100;`. Why the slash in front of the box, though? That's how flash keeps track of the sprites in the movie, just like a unix filesystem- here it

shows that box is at the top level. If the sprite named box had another sprite named biff inside of it, you'd set its x position with /box/biff.x = 100;. At least, I think so; correct me if I'm wrong here.

This simple example will move the red square across the window.

Példa 1. swfaction() example

```
<?php
 $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for($n=0; $n<5; ++$n)
  {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000,4000);

  $i = $m->add($p);
  $i->setDepth(1);
  $i->moveTo(-500,2000);
  $i->setName("box");

  $m->add(new SWFAction("/box.x += 3;"));
  $m->nextFrame();
  $m->add(new SWFAction("gotoFrame(0); play();"));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example tracks down your mouse on the screen.

Példa 2. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setRate(36.0);
  $m->setDimension(1200, 800);
  $m->setBackground(0, 0, 0);

  /* mouse tracking sprite - empty, but follows mouse so we can
     get its x and y coordinates */

  $i = $m->add(new SWFSprite());
  $i->setName('mouse');

  $m->add(new SWFAction("
    startDrag('/mouse', 1); /* '1' means lock sprite to the mouse */
  "));

  /* might as well turn off antialiasing, since these are just squares. */

  $m->add(new SWFAction("
    this.quality = 0;
  "));

  /* morphing box */
  $r = new SWFMorph();
  $s = $r->getShape1();

  /* Note this is backwards from normal shapes.  No idea why. */
  $s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
  $s->movePenTo(-40, -40);
  $s->drawLine(80, 0);
  $s->drawLine(0, 80);
  $s->drawLine(-80, 0);
  $s->drawLine(0, -80);

  $s = $r->getShape2();

  $s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
  $s->movePenTo(-1, -1);
  $s->drawLine(2, 0);
  $s->drawLine(0, 2);
  $s->drawLine(-2, 0);
  $s->drawLine(0, -2);

  /* sprite container for morphing box -
     this is just a timeline w/ the box morphing */

  $box = new SWFSprite();
  $box->add(new SWFAction("
    stop();
  "));
  $i = $box->add($r);

  for($n=0; $n<=20; ++$n)
```

```
  {
    $i->setRatio($n/20);
    $box->nextFrame();
  }

  /* this container sprite allows us to use the same action code many times */

  $cell = new SWFSprite();
  $i = $cell->add($box);
  $i->setName('box');

  $cell->add(new SWFAction("

    setTarget('box');

    /* ...x means the x coordinate of the parent, i.e. (..).x */
    dx = (/mouse.x + random(6)-3 - ...x)/5;
    dy = (/mouse.y + random(6)-3 - ...y)/5;
    gotoFrame(int(dx*dx + dy*dy));

  "));

  $cell->nextFrame();
  $cell->add(new SWFAction("

    gotoFrame(0);
    play();

  "));

  $cell->nextFrame();

  /* finally, add a bunch of the cells to the movie */

  for($x=0; $x<12; ++$x)
  {
    for($y=0; $y<8; ++$y)
    {
      $i = $m->add($cell);
      $i->moveTo(100*$x+50, 100*$y+50);
    }
  }

  $m->nextFrame();

  $m->add(new SWFAction("

    gotoFrame(1);
    play();

  "));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Same as above, but with nice colored balls...

Példa 3. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(11000, 8000);
  $m->setBackground(0x00, 0x00, 0x00);

  $m->add(new SWFAction("

this.quality = 0;
/frames.visible = 0;
startDrag('/mouse', 1);

  "));

  // mouse tracking sprite
  $t = new SWFSprite();
  $i = $m->add($t);
  $i->setName('mouse');

  $g = new SWFGradient();
  $g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
  $g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

  // gradient shape thing
  $s = new SWFShape();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.03);
  $s->setRightFill($f);
  $s->movePenTo(-600, -600);
  $s->drawLine(1200, 0);
  $s->drawLine(0, 1200);
  $s->drawLine(-1200, 0);
  $s->drawLine(0, -1200);

  // need to make this a sprite so we can multColor it
  $p = new SWFSprite();
  $p->add($s);
  $p->nextFrame();

  // put the shape in here, each frame a different color
  $q = new SWFSprite();
  $q->add(new SWFAction("gotoFrame(random(7)+1); stop();"));
  $i = $q->add($p);

  $i->multColor(1.0, 1.0, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 0.5);
  $q->nextFrame();
```

```
  $i->multColor(1.0, 0.75, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 0.5, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 1.0);
  $q->nextFrame();

  // finally, this one contains the action code
  $p = new SWFSprite();
  $i = $p->add($q);
  $i->setName('frames');
  $p->add(new SWFAction("

dx = (/:mousex-/:lastx)/3 + random(10)-5;
dy = (/:mousey-/:lasty)/3;
x = /:mousex;
y = /:mousey;
alpha = 100;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("

this.x = x;
this.y = y;
this.alpha = alpha;
x += dx;
y += dy;
dy += 3;
alpha -= 8;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("prevFrame(); play();"));
  $p->nextFrame();

  $i = $m->add($p);
  $i->setName('frames');
  $m->nextFrame();

  $m->add(new SWFAction("

lastx = mousex;
lasty = mousey;
mousex = /mouse.x;
mousey = /mouse.y;

++num;

if(num == 11)
  num = 1;
```

```
removeClip('char' & num);
duplicateClip(/frames, 'char' & num, num);

  "));

  $m->nextFrame();
  $m->add(new SWFAction("prevFrame(); play();"));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will handles keyboard actions. (You'll probably have to click in the window to give it focus. And you'll probably have to leave your mouse in the frame, too. If you know how to give buttons focus programatically, feel free to share, won't you?)

Példa 4. swfaction() example

```
<?php

  /* sprite has one letter per frame */

  $p = new SWFSprite();
  $p->add(new SWFAction("stop();"));

  $chars = "abcdefghijklmnopqrstuvwxyz".
       "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
           "1234567890!@#$%^&/*()_+-=/[]{}|;:,.<>¿~";

  $f = new SWFFont("_sans");

  for($n=0; $nremove($i);
    $t = new SWFTextField();
    $t->setFont($f);
    $t->setHeight(240);
    $t->setBounds(600,240);
    $t->align(SWFTEXTFIELD_ALIGN_CENTER);
    $t->addString($c);
    $i = $p->add($t);
    $p->labelFrame($c);
    $p->nextFrame();
  }

  /* hit region for button - the entire frame */

  $s = new SWFShape();
  $s->setFillStyle0($s->addSolidFill(0, 0, 0, 0));
  $s->drawLine(600, 0);
  $s->drawLine(0, 400);
  $s->drawLine(-600, 0);
  $s->drawLine(0, -400);
```

```
  /* button checks for pressed key, sends sprite to the right frame */

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT);

  for($n=0; $naddAction(new SWFAction("

setTarget('/char');
gotoFrame('$c');

    "), SWFBUTTON_KEYPRESS($c));
  }

  $m = new SWFMovie();
  $m->setDimension(600,400);
  $i = $m->add($p);
  $i->setName('char');
  $i->moveTo(0,80);

  $m->add($b);

  header('Content-type: application/x-shockwave-flash');
  $m->output();

?>
```

SWFBitmap->getHeight (unknown)

Returns the bitmap's height.

int **swfbitmap->getheight** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbitmap->getheight() returns the bitmap's height in pixels.

See also **swfbitmap->getwidth()**.

SWFBitmap->getWidth (unknown)

Returns the bitmap's width.

int **swfbitmap->getwidth** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbitmap->getwidth() returns the bitmap's width in pixels.

See also **swfbitmap->getheight()**.

SWFBitmap (PHP 4 >= 4.0.5)

Loads Bitmap object

new **swfbitmap** (string filename [, int alphafilename]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbitmap() creates a new SWFBitmap object from the Jpeg or DBL file named *`filename`*. *`alphafilename`* indicates a MSK file to be used as an alpha mask for a Jpeg image.

> **Megjegyzés:** We can only deal with baseline (frame 0) jpegs, no baseline optimized or progressive scan jpegs!

SWFBitmap has the following methods : **swfbitmap->getwidth()** and **swfbitmap->getheight()**.

You can't import png images directly, though- have to use the png2dbl utility to make a dbl ("define bits lossless") file from the png. The reason for this is that I don't want a dependency on the png library in ming- autoconf should solve this, but that's not set up yet.

Példa 1. Import PNG files

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(new SWFBitmap("png.dbl"));
  $s->setRightFill($f);

  $s->drawLine(32, 0);
  $s->drawLine(0, 32);
  $s->drawLine(-32, 0);
  $s->drawLine(0, -32);

  $m = new SWFMovie();
```

```
  $m->setDimension(32, 32);
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

And you can put an alpha mask on a jpeg fill.

Példa 2. swfbitmap() example

```
<?php

  $s = new SWFShape();

  // .msk file generated with "gif2mask" utility
  $f = $s->addFill(new SWFBitmap("alphafill.jpg", "alphafill.msk"));
  $s->setRightFill($f);

  $s->drawLine(640, 0);
  $s->drawLine(0, 480);
  $s->drawLine(-640, 0);
  $s->drawLine(0, -480);

  $c = new SWFShape();
  $c->setRightFill($c->addFill(0x99, 0x99, 0x99));
  $c->drawLine(40, 0);
  $c->drawLine(0, 40);
  $c->drawLine(-40, 0);
  $c->drawLine(0, -40);

  $m = new SWFMovie();
  $m->setDimension(640, 480);
  $m->setBackground(0xcc, 0xcc, 0xcc);

  // draw checkerboard background
  for($y=0; $y<480; $y+=40)
  {
    for($x=0; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }

    $y+=40;

    for($x=40; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }
  }
```

```
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

swfbutton_keypress (PHP 4 >= 4.0.5)

Returns the action flag for keyPress(char)

int **swfbutton_keypress** (string str) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

SWFbutton->addAction (unknown)

Adds an action

void **swfbutton->addaction** (ressource action, int flags) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbutton->addaction() adds the action *`action`* to this button for the given conditions. The following *`flags`* are valid: SWFBUTTON_MOUSEOVER, SWFBUTTON_MOUSEOUT, SWFBUTTON_MOUSEUP, SWFBUTTON_MOUSEUPOUTSIDE, SWFBUTTON_MOUSEDOWN, SWFBUTTON_DRAGOUT and SWFBUTTON_DRAGOVER.

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->addShape (unknown)

Adds a shape to a button

void **swfbutton->addshape** (ressource shape, int flags) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbutton->addshape() adds the shape *`shape`* to this button. The following *`flags`*' values are valid: SWFBUTTON_UP, SWFBUTTON_OVER, SWFBUTTON_DOWN or SWFBUTTON_HIT. SWFBUTTON_HIT isn't ever displayed, it defines the hit region for the button. That is, everywhere the hit shape would be drawn is considered a "touchable" part of the button.

SWFbutton->setAction (unknown)

Sets the action

void **swfbutton->setaction** (ressource action) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbutton->setaction() sets the action to be performed when the button is clicked. Alias for addAction(shape, SWFBUTTON_MOUSEUP). *`action`* is a swfaction().

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setdown (unknown)

Alias for addShape(shape, SWFBUTTON_DOWN))

void **swfbutton->setdown** (ressource shape) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbutton->setdown() alias for addShape(shape, SWFBUTTON_DOWN).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setHit (unknown)

Alias for addShape(shape, SWFBUTTON_HIT)

void **swfbutton->sethit** (ressource shape) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbutton->sethit() alias for addShape(shape, SWFBUTTON_HIT).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setOver (unknown)

Alias for addShape(shape, SWFBUTTON_OVER)

void **swfbutton->setover** (ressource shape) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbutton->setover() alias for addShape(shape, SWFBUTTON_OVER).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setUp (unknown)

Alias for addShape(shape, SWFBUTTON_UP)

void **swfbutton->setup** (ressource shape) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbutton->setup() alias for addShape(shape, SWFBUTTON_UP).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton (PHP 4 >= 4.0.5)

Creates a new Button.

new **swfbutton** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfbutton() creates a new Button. Roll over it, click it, see it call action code. Swank.

SWFButton has the following methods : **swfbutton->addshape()**, **swfbutton->setup()**, **swfbutton->setover() swfbutton->setdown()**, **swfbutton->sethit() swfbutton->setaction()** and **swfbutton->addaction()**.

This simple example will show your usual interactions with buttons : rollover, rollon, mouseup, mousedown, noaction.

Példa 1. swfbutton() example

```
<?php

  $f = new SWFFont("_serif");

  $p = new SWFSprite();

  function label($string)
  {
    global $f;

    $t = new SWFTextField();
    $t->setFont($f);
    $t->addString($string);
    $t->setHeight(200);
    $t->setBounds(3200,200);
    return $t;
  }
  function addLabel($string)
  {
    global $p;

    $i = $p->add(label($string));
    $p->nextFrame();
    $p->remove($i);
  }

  $p->add(new SWFAction("stop();"));
  addLabel("NO ACTION");
  addLabel("SWFBUTTON_MOUSEUP");
  addLabel("SWFBUTTON_MOUSEDOWN");
  addLabel("SWFBUTTON_MOUSEOVER");
```

```
  addLabel("SWFBUTTON_MOUSEOUT");
  addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
  addLabel("SWFBUTTON_DRAGOVER");
  addLabel("SWFBUTTON_DRAGOUT");

  function rect($r, $g, $b)
  {
    $s = new SWFShape();
    $s->setRightFill($s->addFill($r, $g, $b));
    $s->drawLine(600,0);
    $s->drawLine(0,600);
    $s->drawLine(-600,0);
    $s->drawLine(0,-600);

    return $s;
  }

  $b = new SWFButton();
  $b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
  $b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
  $b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(1);"),
            SWFBUTTON_MOUSEUP);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(2);"),
        SWFBUTTON_MOUSEDOWN);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(3);"),
        SWFBUTTON_MOUSEOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(4);"),
        SWFBUTTON_MOUSEOUT);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(5);"),
        SWFBUTTON_MOUSEUPOUTSIDE);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(6);"),
        SWFBUTTON_DRAGOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(7);"),
        SWFBUTTON_DRAGOUT);

  $m = new SWFMovie();
  $m->setDimension(4000,3000);

  $i = $m->add($p);
  $i->setName("label");
  $i->moveTo(400,1900);

  $i = $m->add($b);
  $i->moveTo(400,900);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will enables you to drag draw a big red button on the windows. No drag-and-drop, just moving around.

Példa 2. swfbutton->addaction() example

```
<?php

  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->drawLine(1000,0);
  $s->drawLine(0,1000);
  $s->drawLine(-1000,0);
  $s->drawLine(0,-1000);

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP | SWFBUTTON_DOWN | SWFBUTTON_OVER);

  $b->addAction(new SWFAction("startDrag('/test', 0);"), // '0' means don't lock to mouse
        SWFBUTTON_MOUSEDOWN);

  $b->addAction(new SWFAction("stopDrag();"),
        SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

  $p = new SWFSprite();
  $p->add($b);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->setName('test');
  $i->moveTo(1000,1000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->addColor (unknown)

Adds the given color to this item's color transform.

void **swfdisplayitem->addcolor** ([int red [, int green [, int blue [, int a]]]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->addcolor() adds the color to this item's color transform. The color is given in its RGB form.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->move (unknown)

Moves object in relative coordinates.

void **swfdisplayitem->move** (int dx, int dy) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->move() moves the current object by (dx,dy) from its current position.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->moveto()**.

SWFDisplayItem->moveTo (unknown)

Moves object in global coordinates.

void **swfdisplayitem->moveto** (int x, int y) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->moveto() moves the current object to (x,y) in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->move()**.

SWFDisplayItem->multColor (unknown)

Multiplies the item's color transform.

void **swfdisplayitem->multcolor** ([int red [, int green [, int blue [, int a]]]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->multcolor() multiplies the item's color transform by the given values.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This simple example will modify your picture's atmospher to Halloween (use a landscape or bright picture).

Példa 1. swfdisplayitem->multcolor() example

```
<?php

  $b = new SWFBitmap("backyard.jpg");
  // note use your own picture :-)
  $s = new SWFShape();
  $s->setRightFill($s->addFill($b));
  $s->drawLine($b->getWidth(), 0);
  $s->drawLine(0, $b->getHeight());
  $s->drawLine(-$b->getWidth(), 0);
  $s->drawLine(0, -$b->getHeight());

  $m = new SWFMovie();
  $m->setDimension($b->getWidth(), $b->getHeight());

  $i = $m->add($s);

  for($n=0; $n<=20; ++$n)
  {
    $i->multColor(1.0-$n/10, 1.0, 1.0);
    $i->addColor(0xff*$n/20, 0, 0);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->remove (unknown)

Removes the object from the movie

void **swfdisplayitem->remove** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->remove() removes this object from the movie's display list.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfmovie->add()**.

SWFDisplayItem->Rotate (unknown)

Rotates in relative coordinates.

void **swfdisplayitem->rotate** (float ddegrees) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->rotate() rotates the current object by `ddegrees` degrees from its current rotation.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->rotateto()**.

SWFDisplayItem->rotateTo (unknown)

Rotates the object in global coordinates.

void **swfdisplayitem->rotateto** (float degrees) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->rotateto() set the current object rotation to *`degrees`* degrees in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This example bring three rotating string from the background to the foreground. Pretty nice.

Példa 1. swfdisplayitem->rotateto() example

```
<?php
  $thetext =  "ming!";

  $f = new SWFFont("Bauhaus 93.fdb");

  $m = new SWFMovie();
  $m->setRate(24.0);
  $m->setDimension(2400, 1600);
  $m->setBackground(0xff, 0xff, 0xff);

  // functions with huge numbers of arbitrary
  // arguments are always a good idea!  Really!

  function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
  {
    global $f, $m;

    $t = new SWFText();
    $t->setFont($f);
    $t->setColor($r, $g, $b, $a);
    $t->setHeight(960);
    $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
    $t->addString($string);

    // we can add properties just like a normal php var,
    // as long as the names aren't already used.
    // e.g., we can't set $i->scale, because that's a function

    $i = $m->add($t);
    $i->x = $x;
    $i->y = $y;
    $i->rot = $rot;
    $i->s = $scale;
    $i->rotateTo($rot);
    $i->scale($scale, $scale);

    // but the changes are local to the function, so we have to
    // return the changed object.  kinda weird..
```

```
    return $i;
 }

  function step($i)
  {
    $oldrot = $i->rot;
    $i->rot = 19*$i->rot/20;
    $i->x = (19*$i->x + 1200)/20;
    $i->y = (19*$i->y + 800)/20;
    $i->s = (19*$i->s + 1.0)/20;

    $i->rotateTo($i->rot);
    $i->scaleTo($i->s, $i->s);
    $i->moveTo($i->x, $i->y);

    return $i;
  }

  // see?  it sure paid off in legibility:

  $i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
  $i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
  $i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);

  for($i=1; $i<=100; ++$i)
  {
    $i1 = step($i1);
    $i2 = step($i2);
    $i3 = step($i3);

    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

See also **swfdisplayitem->rotate()**.

SWFDisplayItem->scale (unknown)

Scales the object in relative coordinates.

void **swfdisplayitem->scale** (int dx, int dy) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->scale() scales the current object by (*dx*,*dy*) from its current size.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scaleto()**.

SWFDisplayItem->scaleTo (unknown)

Scales the object in global coordinates.

void **swfdisplayitem->scaleto** (int x, int y) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->scaleto() scales the current object to (*x*,*y*) in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scale()**.

SWFDisplayItem->setDepth (unknown)

Sets z-order

void **swfdisplayitem->setdepth** (float depth) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->rotate() sets the object's z-order to *depth*. Depth defaults to the order in which instances are created (by add'ing a shape/text to a movie)- newer ones are on top of older ones. If two objects are given the same depth, only the later-defined one can be moved.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setName (unknown)

Sets the object's name

void **swfdisplayitem->setname** (string name) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->setname() sets the object's name to *`name`*, for targetting with action script. Only useful on sprites.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setRatio (unknown)

Sets the object's ratio.

void **swfdisplayitem->setratio** (float ratio) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->setratio() sets the object's ratio to *`ratio`*. Obviously only useful for morphs.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This simple example will morph nicely three concentric circles.

Példa 1. swfdisplayitem->setname() example

```
<?php

  $p = new SWFMorph();

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
```

```
  $g->addEntry(0.16, 0xff, 0xff, 0xff);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0xff, 0xff, 0xff);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0xff, 0xff, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape1();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0, 0);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0, 0xff, 0);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0, 0, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape2();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $f->skewXTo(1.0);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m = new SWFMovie();
  $m->setDimension(320, 240);
  $i = $m->add($p);
  $i->moveTo(160, 120);

  for($n=0; $n<=1.001; $n+=0.01)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->skewX (unknown)

Sets the X-skew.

void **swfdisplayitem->skewx** (float ddegrees) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->skewx() adds *ddegrees* to current x-skew.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewXTo (unknown)

Sets the X-skew.

void **swfdisplayitem->skewxto** (float degrees) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->skewxto() sets the x-skew to *degrees*. For *degrees* is 1.0, it means a 45-degree forward slant. More is more forward, less is more backward.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewY (unknown)

Sets the Y-skew.

void **swfdisplayitem->skewy** (float ddegrees) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->skewy() adds *ddegrees* to current y-skew.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewyto()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->skewYTo (unknown)

Sets the Y-skew.

void **swfdisplayitem->skewyto** (float degrees) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem->skewyto() sets the y-skew to *degrees*. For *degrees* is 1.0, it means a 45-degree forward slant. More is more upward, less is more downward.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewy()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem (unknown)

Creates a new displayitem object.

new **swfdisplayitem** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfdisplayitem() creates a new swfdisplayitem object.

Here's where all the animation takes place. After you define a shape, a text object, a sprite, or a button, you add it to the movie, then use the returned handle to move, rotate, scale, or skew the thing.

SWFDisplayItem has the following methods : **swfdisplayitem->move()**, **swfdisplayitem->moveto()**, **swfdisplayitem->scaleto()**, **swfdisplayitem->scale()**, **swfdisplayitem->rotate()**, **swfdisplayitem->rotateto()**, **swfdisplayitem->skewxto()**, **swfdisplayitem->skewx()**, **swfdisplayitem->skewyto() swfdisplayitem->skewyto()**, **swfdisplayitem->setdepth() swfdisplayitem->remove()**, **swfdisplayitem->setname() swfdisplayitem->setratio()**, **swfdisplayitem->addcolor()** and **swfdisplayitem->multcolor()**.

SWFFill->moveTo (unknown)

Moves fill origin

void **swffill->moveto** (int x, int y) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swffill->moveto() moves fill's origin to (*x*,*y*) in global coordinates.

SWFFill->rotateTo (unknown)

Sets fill's rotation

void **swffill->rotateto** (float degrees) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swffill->rotateto() sets fill's rotation to *degrees* degrees.

SWFFill->scaleTo (unknown)

Sets fill's scale

void **swffill->scaleto** (int x, int y) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swffill->scaleto() sets fill's scale to *x* in the x-direction, *y* in the y-direction.

SWFFill->skewXTo (unknown)

Sets fill x-skew

void **swffill->skewxto** (float x) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swffill->skewxto() sets fill x-skew to *x*. For *x* is 1.0, it is a is a 45-degree forward slant. More is more forward, less is more backward.

SWFFill->skewYTo (unknown)

Sets fill y-skew

void **swffill->skewyto** (float y) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swffill->skewyto() sets fill y-skew to *y*. For *y* is 1.0, it is a is a 45-degree upward slant. More is more upward, less is more downward.

SWFFill (PHP 4 >= 4.0.5)

Loads SWFFill object

The **swffill()** object allows you to transform (scale, skew, rotate) bitmap and gradient fills. **swffill()** objects are created by the **swfshape->addfill()** methods.

SWFFill has the following methods : **swffill->moveto()** and **swffill->scaleto()**, **swffill->rotateto()**, **swffill->skewxto()** and **swffill->skewyto()**.

swffont->getwidth (unknown)

Returns the string's width

int **swffont->getwidth** (string string) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swffont->getwidth() returns the string *`string`*'s width, using font's default scaling. You'll probably want to use the **SWFText()** version of this method which uses the text object's scale.

SWFFont (PHP 4 >= 4.0.5)

Loads a font definition

new **swffont** (string filename) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

If *`filename`* is the name of an FDB file (i.e., it ends in ".fdb"), load the font definition found in said file. Otherwise, create a browser-defined font reference.

FDB ("font definition block") is a very simple wrapper for the SWF DefineFont2 block which contains a full description of a font. One may create FDB files from SWT Generator template files with the included makefdb utility- look in the util directory off the main ming distribution directory.

Browser-defined fonts don't contain any information about the font other than its name. It is assumed that the font definition will be provided by the movie player. The fonts _serif, _sans, and _typewriter should always be available. For example:

```
<?php
$f = newSWFFont("_sans");
?>
```

will give you the standard sans-serif font, probably the same as what you'd get with `<font name="sans-serif">` in HTML.

swffont() returns a reference to the font definition, for use in the **SWFText->setFont()** and the **SWFTextField->setFont()** methods.

SWFFont has the following methods : **swffont->getwidth()**.

SWFGradient->addEntry (unknown)

Adds an entry to the gradient list.

void **swfgradient->addentry** (float ratio, int red, int green, int blue [, int a]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfgradient->addentry() adds an entry to the gradient list. *`ratio`* is a number between 0 and 1 indicating where in the gradient this color appears. Thou shalt add entries in order of increasing ratio.

`red`, *`green`*, *`blue`* is a color (RGB mode). Last parameter *`a`* is optional.

SWFGradient (PHP 4 >= 4.0.5)

Creates a gradient object

new **swfgradient** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfgradient() creates a new SWFGradient object.

After you've added the entries to your gradient, you can use the gradient in a shape fill with the **swfshape->addfill()** method.

SWFGradient has the following methods : **swfgradient->addentry()**.

This simple example will draw a big black-to-white gradient as background, and a redish disc in its center.

Példa 1. swfgradient() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(320, 240);

  $s = new SWFShape();

  // first gradient- black to white
  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(1.0, 0xff, 0xff, 0xff);

  $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
  $f->scaleTo(0.01);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  $s = new SWFShape();

  // second gradient- radial gradient from red to transparent
  $g = new SWFGradient();
  $g->addEntry(0.0, 0xff, 0, 0, 0xff);
  $g->addEntry(1.0, 0xff, 0, 0, 0);

  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.005);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMorph->getshape1 (unknown)

Gets a handle to the starting shape

mixed **swfmorph->getshape1** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmorph->getshape1() gets a handle to the morph's starting shape. **swfmorph->getshape1()** returns an swfshape() object.

SWFMorph->getshape2 (unknown)

Gets a handle to the ending shape

mixed **swfmorph->getshape2** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmorph->getshape2() gets a handle to the morph's ending shape. **swfmorph->getshape2()** returns an swfshape() object.

SWFMorph (PHP 4 >= 4.0.5)

Creates a new SWFMorph object.

new **swfmorph** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmorph() creates a new SWFMorph object.

Also called a "shape tween". This thing lets you make those tacky twisting things that make your computer choke. Oh, joy!

The methods here are sort of weird. It would make more sense to just have newSWFMorph(shape1, shape2);, but as things are now, shape2 needs to know that it’s the second part of a morph. (This, because it starts writing its output as soon as it gets drawing commands- if it kept its own description of its shapes and wrote on completion this and some other things would be much easier.)

SWFMorph has the following methods : **swfmorph->getshape1()** and **swfmorph->getshape1()**.

This simple example will morph a big red square into a smaller blue black-bordered square.

Példa 1. swfmorph() example

```
<?php
  $p = new SWFMorph();

  $s = $p->getShape1();
  $s->setLine(0,0,0,0);

  /* Note that this is backwards from normal shapes (left instead of right).
     I have no idea why, but this seems to work.. */

  $s->setLeftFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-1000,-1000);
  $s->drawLine(2000,0);
  $s->drawLine(0,2000);
  $s->drawLine(-2000,0);
  $s->drawLine(0,-2000);

  $s = $p->getShape2();
  $s->setLine(60,0,0,0);
  $s->setLeftFill($s->addFill(0, 0, 0xff));
  $s->movePenTo(0,-1000);
  $s->drawLine(1000,1000);
  $s->drawLine(-1000,1000);
  $s->drawLine(-1000,-1000);
  $s->drawLine(1000,-1000);

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setBackground(0xff, 0xff, 0xff);

  $i = $m->add($p);
  $i->moveTo(1500,1000);

  for($r=0.0; $r<=1.0; $r+=0.1)
  {
    $i->setRatio($r);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMovie->add (unknown)

Adds any type of data to a movie.

void **swfmovie->add** (ressource instance) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie->add() adds `instance` to the current movie. `instance` is any type of data : Shapes, text, fonts, etc. must all be add'ed to the movie to make this work.

For displayable types (shape, text, button, sprite), this returns an **SWFDisplayItem()**, a handle to the object in a display list. Thus, you can add the same shape to a movie multiple times and get separate handles back for each separate instance.

See also all other objects (adding this later), and **swfmovie->remove()**

See examples in : **swfdisplayitem->rotateto()** and **swfshape->addfill()**.

SWFMovie->nextframe (unknown)

Moves to the next frame of the animation.

void **swfmovie->nextframe** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie->setframes() moves to the next frame of the animation.

SWFMovie->output (unknown)

Dumps your lovingly prepared movie out.

void **swfmovie->output** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie->output() dumps your lovingly prepared movie out. In PHP, preceding this with the command

```
<?php
header('Content-type: application/x-shockwave-flash');
?>
```

convinces the browser to display this as a flash movie.

See also **swfmovie->save()**.

See examples in : **swfmovie->streammp3()**, **swfdisplayitem->rotateto()**, swfaction()... Any example will use this method.

SWFMovie->remove (unknown)

Removes the object instance from the display list.

void **swfmovie->remove** (resource instance) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie->remove() removes the object instance *`instance`* from the display list.

See also **swfmovie->add()**.

SWFMovie->save (unknown)

Saves your movie in a file.

void **swfmovie->save** (string filename) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie->save() saves your movie to the file named *`filename`*.

See also **output()**.

SWFMovie->setbackground (unknown)

Sets the background color.

void **swfmovie->setbackground** (int red, int green, int blue) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie->setbackground() sets the background color. Why is there no rgba version? Think about it. (Actually, that's not such a dumb question after all- you might want to let the html background show through. There's a way to do that, but it only works on IE4. Search the http://www.macromedia.com/ site for details.)

SWFMovie->setdimension (unknown)

Sets the movie's width and height.

void **swfmovie->setdimension** (int width, int height) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie->setdimension() sets the movie's width to *`width`* and height to *`height`*.

SWFMovie->setframes (unknown)

Sets the total number of frames in the animation.

void **swfmovie->setframes** (string numberofframes) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie->setframes() sets the total number of frames in the animation to *`numberofframes`*.

SWFMovie->setrate (unknown)

Sets the animation's frame rate.

void **swfmovie->setrate** (int rate) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie->setrate() sets the frame rate to *`rate`*, in frame per seconds. Animation will slow down if the player can't render frames fast enough- unless there's a streaming sound, in which case display frames are sacrificed to keep sound from skipping.

SWFMovie->streammp3 (unknown)

Streams a MP3 file.

void **swfmovie->streammp3** (string mp3FileName) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie->streammp3() streams the mp3 file *`mp3FileName`*. Not very robust in dealing with oddities (can skip over an initial ID3 tag, but that's about it). Like **SWFShape->addJpegFill()**, this

isn’t a stable function- we’ll probably need to make a separate SWFSound object to contain sound types.

Note that the movie isn’t smart enough to put enough frames in to contain the entire mp3 stream- you’ll have to add (length of song * frames per second) frames to get the entire stream in.

Yes, now you can use ming to put that rock and roll devil worship music into your SWF files. Just don’t tell the RIAA.

Példa 1. swfmovie->streammp3() example

```
<?php
  $m = new SWFMovie();
  $m->setRate(12.0);
  $m->streamMp3("distortobass.mp3");
  // use your own MP3

  // 11.85 seconds at 12.0 fps = 142 frames
  $m->setFrames(142);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMovie (PHP 4 >= 4.0.5)

Creates a new movie object, representing an SWF version 4 movie.

new **swfmovie** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfmovie() creates a new movie object, representing an SWF version 4 movie.

SWFMovie has the following methods : **swfmovie->output()**,**swfmovie->save()**, **swfmovie->add()**, **swfmovie->remove()**, **swfmovie->nextframe()**, **swfmovie->setbackground()**, **swfmovie->setrate()**, **swfmovie->setdimension()**, **swfmovie->setframes()** and **swfmovie->streammp3()**.

See examples in : **swfdisplayitem->rotateto()**, **swfshape->setline()**, **swfshape->addfill()**... Any example will use this object.

SWFShape->addFill (unknown)

Adds a solid fill to the shape.

void **swfshape->addfill** (int red, int green, int blue [, int a]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

void **swfshape->addfill** (SWFbitmap bitmap [, int flags]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

void **swfshape->addfill** (SWFGradient gradient [, int flags]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfshape->addfill() adds a solid fill to the shape's list of fill styles. **swfshape->addfill()** accepts three different types of arguments.

`red`, *`green`*, *`blue`* is a color (RGB mode). Last parameter *`a`* is optional.

The *`bitmap`* argument is an swfbitmap() object. The *`flags`* argument can be one of the following values : SWFFILL_CLIPPED_BITMAP or SWFFILL_TILED_BITMAP. Default is SWFFILL_TILED_BITMAP. I think.

The *`gradient`* argument is an swfgradient() object. The flags argument can be one of the following values : SWFFILL_RADIAL_GRADIENT or SWFFILL_LINEAR_GRADIENT. Default is SWFFILL_LINEAR_GRADIENT. I'm sure about this one. Really.

swfshape->addfill() returns an swffill() object for use with the **swfshape->setleftfill()** and **swfshape->setrightfill()** functions described below.

See also **swfshape->setleftfill()** and **swfshape->setrightfill()**.

This simple example will draw a frame on a bitmap. Ah, here's another buglet in the flash player- it doesn't seem to care about the second shape's bitmap's transformation in a morph. According to spec, the bitmap should stretch along with the shape in this example..

Példa 1. swfshape->addfill() example

```
<?php

  $p = new SWFMorph();

  $b = new SWFBitmap("alphafill.jpg");
  // use your own bitmap
  $width = $b->getWidth();
  $height = $b->getHeight();

  $s = $p->getShape1();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);
  $f->moveTo(-$width/2, -$height/4);
  $f->scaleTo(1.0, 0.5);
  $s->setLeftFill($f);
  $s->movePenTo(-$width/2, -$height/4);
  $s->drawLine($width, 0);
  $s->drawLine(0, $height/2);
  $s->drawLine(-$width, 0);
  $s->drawLine(0, -$height/2);

  $s = $p->getShape2();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);

  // these two have no effect!
  $f->moveTo(-$width/4, -$height/2);
  $f->scaleTo(0.5, 1.0);

  $s->setLeftFill($f);
  $s->movePenTo(-$width/4, -$height/2);
  $s->drawLine($width/2, 0);
  $s->drawLine(0, $height);
  $s->drawLine(-$width/2, 0);
  $s->drawLine(0, -$height);

  $m = new SWFMovie();
  $m->setDimension($width, $height);
  $i = $m->add($p);
  $i->moveTo($width/2, $height/2);

  for($n=0; $n<1.001; $n+=0.03)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->drawCurve (unknown)

Draws a curve (relative).

void **swfshape->drawcurve** (int controldx, int controldy, int anchordx, int anchordy) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfshape->drawcurve() draws a quadratic curve (using the current line style,set by **swfshape->setline()**) from the current pen position to the relative position (*anchorx*,*anchory*) using relative control point (*controlx*,*controly*). That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawCurveTo (unknown)

Draws a curve.

void **swfshape->drawcurveto** (int controlx, int controly, int anchorx, int anchory) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfshape->drawcurveto() draws a quadratic curve (using the current line style, set by **swfshape->setline()**) from the current pen position to (*anchorx*,*anchory*) using (*controlx*,*controly*) as a control point. That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawLine (unknown)

Draws a line (relative).

void **swfshape->drawline** (int dx, int dy) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfshape->drawline() draws a line (using the current line style set by **swfshape->setline()**) from the current pen position to displacement (*dx*,*dy*).

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawlineto()**.

SWFShape->drawLineTo (unknown)

Draws a line.

void **swfshape->drawlineto** (int x, int y) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfshape->setrightfill() draws a line (using the current line style, set by **swfshape->setline()**) from the current pen position to point (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawline()**.

SWFShape->movePen (unknown)

Moves the shape's pen (relative).

void **swfshape->movepen** (int dx, int dy) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfshape->setrightfill() move the shape's pen from coordinates (current x,current y) to (current x + *dx*, current y + *dy*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->movePenTo (unknown)

Moves the shape's pen.

void **swfshape->movepento** (int x, int y) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfshape->setrightfill() move the shape's pen to (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepen()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->setLeftFill (unknown)

Sets left rasterizing color.

void **swfshape->setleftfill** (swfgradient fill) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

void **swfshape->setleftfill** (int red, int green, int blue [, int a]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

What this nonsense is about is, every edge segment borders at most two fills. When rasterizing the object, it's pretty handy to know what those fills are ahead of time, so the swf format requires these to be specified.

swfshape->setleftfill() sets the fill on the left side of the edge- that is, on the interior if you're defining the outline of the shape in a counter-clockwise fashion. The fill object is an SWFFill object returned from one of the addFill functions above.

This seems to be reversed when you're defining a shape in a morph, though. If your browser crashes, just try setting the fill on the other side.

Shortcut for `swfshape->setleftfill($s->addfill($r, $g, $b [, $a]));`.

See also **swfshape->setrightfill()**.

SWFShape->setLine (unknown)

Sets the shape's line style.

void **swfshape->setline** (int width [, int red [, int green [, int blue [, int a]]]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfshape->setline() sets the shape's line style. *`width`* is the line's width. If *`width`* is 0, the line's style is removed (then, all other arguments are ignored). If *`width`* > 0, then line's color is set to *`red`*, *`green`*, *`blue`*. Last parameter *`a`* is optional.

swfshape->setline() accepts 1, 4 or 5 arguments (not 3 or 2).

You must declare all line styles before you use them (see example).

This simple example will draw a big "!#%*@", in funny colors and gracious style.

Példa 1. swfshape->setline() example

```
<?php
  $s = new SWFShape();
  $f1 = $s->addFill(0xff, 0, 0);
  $f2 = $s->addFill(0xff, 0x7f, 0);
  $f3 = $s->addFill(0xff, 0xff, 0);
  $f4 = $s->addFill(0, 0xff, 0);
  $f5 = $s->addFill(0, 0, 0xff);

  // bug: have to declare all line styles before you use them
  $s->setLine(40, 0x7f, 0, 0);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->setLine(40, 0, 0x7f, 0);
  $s->setLine(40, 0, 0, 0x7f);

  $f = new SWFFont('Techno.fdb');

  $s->setRightFill($f1);
  $s->setLine(40, 0x7f, 0, 0);
  $s->drawGlyph($f, '!');
  $s->movePen($f->getWidth('!'), 0);

  $s->setRightFill($f2);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->drawGlyph($f, '#');
  $s->movePen($f->getWidth('#'), 0);
```

```
  $s->setRightFill($f3);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->drawGlyph($f, '%');
  $s->movePen($f->getWidth('%'), 0);

  $s->setRightFill($f4);
  $s->setLine(40, 0, 0x7f, 0);
  $s->drawGlyph($f, '*');
  $s->movePen($f->getWidth('*'), 0);

  $s->setRightFill($f5);
  $s->setLine(40, 0, 0, 0x7f);
  $s->drawGlyph($f, '@');

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setRate(12.0);
  $i = $m->add($s);
  $i->moveTo(1500-$f->getWidth("!#%*@")/2, 1000+$f->getAscent()/2);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setRightFill (unknown)

Sets right rasterizing color.

void **swfshape->setrightfill** (swfgradient fill) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

void **swfshape->setrightfill** (int red, int green, int blue [, int a]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

See also **swfshape->setleftfill()**.

Shortcut for `swfshape->setrightfill($s->addfill($r, $g, $b [, $a]));`.

SWFShape (PHP 4 >= 4.0.5)

Creates a new shape object.

new **swfshape** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfshape() creates a new shape object.

SWFShape has the following methods : **swfshape->setline()**, **swfshape->addfill()**, **swfshape->setleftfill()**, **swfshape->setrightfill()**, **swfshape->movepento()**, **swfshape->movepen()**, **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->drawcurveto()** and **swfshape->drawcurve()**.

This simple example will draw a big red elliptic quadrant.

Példa 1. swfshape() example

```
<?php
  $s = new SWFShape();
  $s->setLine(40, 0x7f, 0, 0);
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(200, 200);
  $s->drawLineTo(6200, 200);
  $s->drawLineTo(6200, 4600);
  $s->drawCurveTo(200, 4600, 200, 200);

  $m = new SWFMovie();
  $m->setDimension(6400, 4800);
  $m->setRate(12.0);
  $m->add($s);
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFSprite->add (unknown)

Adds an object to a sprite

void **swfsprite->add** (resource object) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfsprite->add() adds a swfshape(), a swfbutton(), a swftext(), a swfaction() or a swfsprite() object.

For displayable types (swfshape(), swfbutton(), swftext(), swfaction() or swfsprite()), this returns a handle to the object in a display list.

SWFSprite->nextframe (unknown)

Moves to the next frame of the animation.

void **swfsprite->nextframe** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfsprite->setframes() moves to the next frame of the animation.

SWFSprite->remove (unknown)

Removes an object to a sprite

void **swfsprite->remove** (ressource object) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfsprite->remove() remove a swfshape(), a swfbutton(), a swftext(), a swfaction() or a swfsprite() object from the sprite.

SWFSprite->setframes (unknown)

Sets the total number of frames in the animation.

void **swfsprite->setframes** (int numberofframes) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfsprite->setframes() sets the total number of frames in the animation to *`numberofframes`*.

SWFSprite (PHP 4 >= 4.0.5)

Creates a movie clip (a sprite)

new **swfsprite** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swfsprite() are also known as a "movie clip", this allows one to create objects which are animated in their own timelines. Hence, the sprite has most of the same methods as the movie.

swfsprite() has the following methods : **swfsprite->add()**, **swfsprite->remove()**, **swfsprite->nextframe()** and **swfsprite->setframes()**.

This simple example will spin gracefully a big red square.

Példa 1. swfsprite() example

```
<?php
  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
```

```
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->moveTo(1500,1000);
  $i->setName("blah");

  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(3000,2000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFText->addString (unknown)

Draws a string

void **swftext->addstring** (string string) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftext->addstring() draws the string *`string`* at the current pen (cursor) location. Pen is at the baseline of the text; i.e., ascending text is in the -y direction.

SWFText->getWidth (unknown)

Computes string's width

void **swftext->addstring** (string string) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftext->addstring() returns the rendered width of the *string* string at the text object's current font, scale, and spacing settings.

SWFText->moveTo (unknown)

Moves the pen

void **swftext->moveto** (int x, int y) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftext->moveto() moves the pen (or cursor, if that makes more sense) to (*x*,*y*) in text object's coordinate space. If either is zero, though, value in that dimension stays the same. Annoying, should be fixed.

SWFText->setColor (unknown)

Sets the current font color

void **swftext->setcolor** (int red, int green, int blue [, int a]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftext->setspacing() changes the current text color. Default is black. I think. Color is represented using the RGB system.

SWFText->setFont (unknown)

Sets the current font

void **swftext->setfont** (string font) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftext->setfont() sets the current font to *`font`*.

SWFText->setHeight (unknown)

Sets the current font height

void **swftext->setheight** (int height) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftext->setheight() sets the current font height to *`height`*. Default is 240.

SWFText->setSpacing (unknown)

Sets the current font spacing

void **swftext->setspacing** (float spacing) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftext->setspacing() sets the current font spacing to *`spacingspacing`*. Default is 1.0. 0 is all of the letters written at the same point. This doesn't really work that well because it inflates the advance across the letter, doesn't add the same amount of spacing between the letters. I should try and explain that better, prolly. Or just fix the damn thing to do constant spacing. This was really just a way to figure out how letter advances work, anyway.. So nyah.

SWFText (PHP 4 >= 4.0.5)

Creates a new SWFText object.

new **swftext** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftext() creates a new SWFText object, fresh for manipulating.

SWFText has the following methods : **swftext->setfont()**, **swftext->setheight()**, **swftext->setspacing()**, **swftext->setcolor()**, **swftext->moveto()**, **swftext->addstring()** and **swftext->getwidth()**.

This simple example will draw a big yellow "PHP generates Flash with Ming" text, on white background.

Példa 1. swftext() example

```
<?php
  $f = new SWFFont("Techno.fdb");
  $t = new SWFText();
  $t->setFont($f);
  $t->moveTo(200, 2400);
  $t->setColor(0xff, 0xff, 0);
  $t->setHeight(1200);
  $t->addString("PHP generates Flash with Ming!!");

  $m = new SWFMovie();
  $m->setDimension(5400, 3600);

  $m->add($t);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFTextField->addstring (unknown)

Concatenates the given string to the text field

void **swftextfield->addstring** (string string) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setname() concatenates the string *`string`* to the text field.

SWFTextField->align (unknown)

Sets the text field alignment

void **swftextfield->align** (int alignement) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->align() sets the text field alignment to *`alignement`*. Valid values for *`alignement`* are : SWFTEXTFIELD_ALIGN_LEFT, SWFTEXTFIELD_ALIGN_RIGHT, SWFTEXTFIELD_ALIGN_CENTER and SWFTEXTFIELD_ALIGN_JUSTIFY.

SWFTextField->setbounds (unknown)

Sets the text field width and height

void **swftextfield->setbounds** (int width, int height) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setbounds() sets the text field width to *`width`* and height to *`height`*. If you don't set the bounds yourself, Ming makes a poor guess at what the bounds are.

SWFTextField->setcolor (unknown)

Sets the color of the text field.

void **swftextfield->setcolor** (int red, int green, int blue [, int a]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setcolor() sets the color of the text field. Default is fully opaque black. Color is represented using RGB system.

SWFTextField->setFont (unknown)

Sets the text field font

void **swftextfield->setfont** (string font) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setfont() sets the text field font to the [browser-defined?] *`font`* font.

SWFTextField->setHeight (unknown)

Sets the font height of this text field font.

void **swftextfield->setheight** (int height) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setheight() sets the font height of this text field font to the given height *`height`*. Default is 240.

SWFTextField->setindentation (unknown)

Sets the indentation of the first line.

void **swftextfield->setindentation** (int width) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setindentation() sets the indentation of the first line in the text field, to *`width`*.

SWFTextField->setLeftMargin (unknown)

Sets the left margin width of the text field.

void **swftextfield->setleftmargin** (int width) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setleftmargin() sets the left margin width of the text field to *`width`*. Default is 0.

SWFTextField->setLineSpacing (unknown)

Sets the line spacing of the text field.

void **swftextfield->setlinespacing** (int height) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setlinespacing() sets the line spacing of the text field to the height of *`height`*. Default is 40.

SWFTextField->setMargins (unknown)

Sets the margins width of the text field.

void **swftextfield->setmargins** (int left, int right) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setmargins() set both margins at once, for the man on the go.

SWFTextField->setname (unknown)

Sets the variable name

void **swftextfield->setname** (string name) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setname() sets the variable name of this text field to *`name`*, for form posting and action scripting purposes.

SWFTextField->setrightMargin (unknown)

Sets the right margin width of the text field.

void **swftextfield->setrightmargin** (int width) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield->setrightmargin() sets the right margin width of the text field to *`width`*. Default is 0.

SWFTextField (PHP 4 >= 4.0.5)

Creates a text field object

new **swftextfield** ([int flags]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

swftextfield() creates a new text field object. Text Fields are less flexible than swftext() objects- they can't be rotated, scaled non-proportionally, or skewed, but they can be used as form entries, and they can use browser-defined fonts.

The optional flags change the text field's behavior. It has the following possibles values :

- SWFTEXTFIELD_NOEDIT indicates that the field shouldn't be user-editable
- SWFTEXTFIELD_PASSWORD obscures the data entry
- SWFTEXTFIELD_DRAWBOX draws the outline of the textfield
- SWFTEXTFIELD_MULTILINE allows multiple lines
- SWFTEXTFIELD_WORDWRAP allows text to wrap
- SWFTEXTFIELD_NOSELECT makes the field non-selectable

Flags are combined with the bitwise OR operation. For example,

```
<?php
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
?>
```

creates a totally useless non-editable password field.

SWFTextField has the following methods : **swftextfield->setfont()**, **swftextfield->setbounds()**, **swftextfield->align()**, **swftextfield->setheight()**, **swftextfield->setleftmargin()**, **swftextfield->setrightmargin()**, **swftextfield->setmargins()**, **swftextfield->setindentation()**, **swftextfield->setlinespacing()**, **swftextfield->setcolor()**, **swftextfield->setname()** and **swftextfield->addstring()**.

LX. Miscellaneous functions

These functions were placed here because none of the other categories seemed to fit.

connection_aborted (PHP 3>= 3.0.7, PHP 4)

Returns TRUE if client disconnected

int **connection_aborted** (void) \linebreak

Returns TRUE if client disconnected. See the Connection Handling description in the Features chapter for a complete explanation.

connection_status (PHP 3>= 3.0.7, PHP 4)

Returns connection status bitfield

int **connection_status** (void) \linebreak

Returns the connection status bitfield. See the Connection Handling description in the Features chapter for a complete explanation.

connection_timeout (PHP 3>= 3.0.7, PHP 4 <= 4.0.4)

Return TRUE if script timed out

bool **connection_timeout** (void) \linebreak

Returns TRUE if script timed out.

Deprecated

This function is deprecated, and doesn't even exist anymore as of 4.0.5.

See the Connection Handling description in the Features chapter for a complete explanation.

constant (PHP 4 >= 4.0.4)

Returns the value of a constant

mixed **constant** (string name) \linebreak

constant() will return the value of the constant indicated by *name*.

constant() is useful if you need to retrieve the value of a constant, but do not know it's name. i.e. It is stored in a variable or returned by a function.

Példa 1. constant() example

```
<?php
```

```
define ("MAXSIZE", 100);

echo MAXSIZE;
echo constant("MAXSIZE"); // same thing as the previous line

?>
```

See also define(), defined() and the section on Constants.

define (PHP 3, PHP 4)

Defines a named constant.

bool **define** (string name, mixed value [, bool case_insensitive]) \linebreak

Defines a named constant. See the section on constants for more details.

The name of the constant is given by *name*; the value is given by *value*.

The optional third parameter *case_insensitive* is also available. If the value TRUE is given, then the constant will be defined case-insensitive. The default behaviour is case-sensitive; i.e. CONSTANT and Constant represent different values.

Példa 1. Defining Constants

```
<?php
define ("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.

define ("GREETING", "Hello you.",TRUE);
echo GREETING; // outputs "Hello you."
echo Greeting; // outputs "Hello you."

?>
```

define() returns TRUE on success and FALSE if an error occurs.

See also defined(), constant() and the section on Constants.

defined (PHP 3, PHP 4)

Checks whether a given named constant exists

bool **defined** (string name) \linebreak

Returns TRUE if the named constant given by *name* has been defined, FALSE otherwise.

Példa 1. Checking Constants

```
<?php
if (defined("CONSTANT")){ // Note that it should be quoted
    echo CONSTANT; //
    }
?>
```

See also define(), constant(), get_defined_constants() and the section on Constants.

die (unknown)

Alias of exit()

This function is an alias of exit().

eval (unknown)

Evaluate a string as PHP code

mixed **eval** (string code_str) \linebreak

eval() evaluates the string given in *code_str* as PHP code. Among other things, this can be useful for storing code in a database text field for later execution.

There are some factors to keep in mind when using **eval()**. Remember that the string passed must be valid PHP code, including things like terminating statements with a semicolon so the parser doesn't die on the line after the **eval()**, and properly escaping things in *code_str*.

Also remember that variables given values under **eval()** will retain these values in the main script afterwards.

A return statement will terminate the evaluation of the string immediately. In PHP 4, **eval()** returns FALSE unless return() is called in the evaluated code, in which case the value passed to return() is returned. In PHP 3, **eval()** does not return a value.

Példa 1. eval() example - simple text merge

```
<?php
$string = 'cup';
$name = 'coffee';
$str = 'This is a $string with my $name in it.<br>';
```

```
echo $str;
eval ("\$str = \"$str\";");
echo $str;
?>
```

The above example will show:

```
This is a $string with my $name in it.
This is a cup with my coffee in it.
```

Tipp: Mint bármilyen más esetben, amikor a kimenet közvetlenül a böngészőhöz kerül, használhatod az kimenet szabályozó függvényeket, hogy a függvény kimenetét "elkapd", és elmentsd például egy string-ben.

exit (unknown)

Output a message and terminate the current script

void **exit** ([string status]) \linebreak void **exit** (int status) \linebreak

Megjegyzés: This is not a real function, but a language construct.

The **exit()** function terminates execution of the script. It prints *`status`* just before exiting.

If *`status`* is an integer, that value will also be used as the exit status. Exit statuses should be in the range 1 to 254, the exit status 255 is reserved by PHP and shall not be used.

Megjegyzés: The current CVS version does NOT print the *`status`* if it is an integer.

Megjegyzés: The die() function is an alias for **exit()**.

Példa 1. exit() example

```
<?php

$filename = '/path/to/data-file';
$file = fopen ($filename, 'r')
```

```
    or exit("unable to open file ($filename)");

?>
```

get_browser (PHP 3, PHP 4)

Tells what the user's browser is capable of

object **get_browser** ([string user_agent]) \linebreak

get_browser() attempts to determine the capabilities of the user's browser. This is done by looking up the browser's information in the `browscap.ini` file. By default, the value of $HTTP_USER_AGENT is used; however, you can alter this (i.e., look up another browser's info) by passing the optional *`user_agent`* parameter to **get_browser()**.

The information is returned in an object, which will contain various data elements representing, for instance, the browser's major and minor version numbers and ID string; `TRUE`/false values for features such as frames, JavaScript, and cookies; and so forth.

While `browscap.ini` contains information on many browsers, it relies on user updates to keep the database current. The format of the file is fairly self-explanatory.

The following example shows how one might list all available information retrieved about the user's browser.

Példa 1. get_browser() example

```
<?php
function list_array ($array) {
    while (list ($key, $value) = each ($array)) {
    $str .= "<b>$key:</b> $value<br />\n";
    }
    return $str;
}
echo "$HTTP_USER_AGENT<hr />\n";
$browser = get_browser();
echo list_array ((array) $browser);
?>
```

The output of the above script would look something like this:

```
Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)<hr />
<b>browser_name_pattern:</b> Mozilla/4\.5.*<br />
<b>parent:</b> Netscape 4.0<br />
<b>platform:</b> Unknown<br />
<b>majorver:</b> 4<br />
```

```
<b>minorver:</b> 5<br />
<b>browser:</b> Netscape<br />
<b>version:</b> 4<br />
<b>frames:</b> 1<br />
<b>tables:</b> 1<br />
<b>cookies:</b> 1<br />
<b>backgroundsounds:</b> <br />
<b>vbscript:</b> <br />
<b>javascript:</b> 1<br />
<b>javaapplets:</b> 1<br />
<b>activexcontrols:</b> <br />
<b>beta:</b> <br />
<b>crawler:</b> <br />
<b>authenticodeupdate:</b> <br />
<b>msn:</b> <br />
```

In order for this to work, your browscap configuration file setting must point to the correct location of the `browscap.ini` file.

For more information (including locations from which you may obtain a `browscap.ini` file), check the PHP FAQ at http://www.php.net/FAQ.php.

highlight_file (PHP 4)

Syntax highlighting of a file

mixed **highlight_file** (string filename [, bool return]) \linebreak

The **highlight_file()** function prints out a syntax higlighted version of the code contained in *`filename`* using the colors defined in the built-in syntax highlighter for PHP.

If the second parameter *`return`* is set to `TRUE` then **highlight_file()** will return the highlighted code as a string instead of printing it out. If the second parameter is not set to `TRUE` then **highlight_file()** will return `TRUE` on success, `FALSE` on failure.

> **Megjegyzés:** The *`return`* parameter became available in PHP 4.2.0. Before this time it behaved like the default, which is `FALSE`

> **Megjegyzés:** Care should be taken when using the show_source() and **highlight_file()** functions to make sure that you do not inadvertently reveal sensitive information such as passwords or any other type of information that might create a potential security risk.

> **Megjegyzés:** Since PHP 4.2.1 this function is also affected by safe_mode and open_basedir.

Példa 1. Creating a source highlighting URL

To setup a URL that can code hightlight any script that you pass to it, we will make use of the "ForceType" directive in Apache to generate a nice URL pattern, and use the function **highlight_file()** to show a nice looking code list.

In your httpd.conf you can add the following:

```
<Location /source>
    ForceType application/x-httpd-php
</Location>
```

And then make a file named "source" and put it in your web root directory.

```
<HTML>
<HEAD>
<TITLE>Source Display</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<?php
    $script = getenv ("PATH_TRANSLATED");
    if(!$script) {
    echo "<BR><B>ERROR: Script Name needed</B><BR>";
    } else {
    if (ereg("(\.php|\.inc)$",$script)) {
    echo "<H1>Source of: $PATH_INFO</H1>\n<HR>\n";
    highlight_file($script);
    } else {
    echo "<H1>ERROR: Only PHP or include script names are allowed</H1>";
    }
    }
    echo "<HR>Processed: ".date("Y/M/d H:i:s",time());
?>
</BODY>
</HTML>
```

Then you can use an URL like the one below to display a colorized version of a script located in "/path/to/script.php" in your web site.

```
http://your.server.com/source/path/to/script.php
```

See also highlight_string(), show_source().

highlight_string (PHP 4)

Syntax highlighting of a string

mixed **highlight_string** (string str [, bool return]) \linebreak

The **highlight_string()** function outputs a syntax highlighted version of *str* using the colors defined in the built-in syntax highlighter for PHP.

If the second parameter *return* is set to TRUE then **highlight_string()** will return the highlighted code as a string instead of printing it out. If the second parameter is not set to TRUE then **highlight_string()** will return TRUE on success, FALSE on failure.

> **Megjegyzés:** The *return* parameter became available in PHP 4.2.0. Before this time it behaved like the default, which is FALSE

See also highlight_file(), and show_source().

ignore_user_abort (PHP 3>= 3.0.7, PHP 4)

Set whether a client disconnect should abort script execution

int **ignore_user_abort** ([int setting]) \linebreak

This function sets whether a client disconnect should cause a script to be aborted. It will return the previous setting and can be called without an argument to not change the current setting and only return the current setting. See the Connection Handling section in the Features chapter for a complete description of connection handling in PHP.

leak (PHP 3, PHP 4)

Leak memory

void **leak** (int bytes) \linebreak

leak() leaks the specified amount of memory.

This is useful when debugging the memory manager, which automatically cleans up "leaked" memory when each request is completed.

pack (PHP 3, PHP 4)

Pack data into binary string.

string **pack** (string format [, mixed args]) \linebreak

Pack given arguments into binary string according to *`format`*. Returns binary string containing data.

The idea to this function was taken from Perl and all formatting codes work the same as there, however, there are some formatting codes that are missing such as Perl’s "u" format code. The format string consists of format codes followed by an optional repeater argument. The repeater argument can be either an integer value or * for repeating to the end of the input data. For a, A, h, H the repeat count specifies how many characters of one data argument are taken, for @ it is the absolute position where to put the next data, for everything else the repeat count specifies how many data arguments are consumed and packed into the resulting binary string. Currently implemented are

- a NUL-padded string
- A SPACE-padded string
- h Hex string, low nibble first
- H Hex string, high nibble first
- c signed char
- C unsigned char
- s signed short (always 16 bit, machine byte order)
- S unsigned short (always 16 bit, machine byte order)
- n unsigned short (always 16 bit, big endian byte order)
- v unsigned short (always 16 bit, little endian byte order)
- i signed integer (machine dependent size and byte order)
- I unsigned integer (machine dependent size and byte order)
- l signed long (always 32 bit, machine byte order)
- L unsigned long (always 32 bit, machine byte order)
- N unsigned long (always 32 bit, big endian byte order)
- V unsigned long (always 32 bit, little endian byte order)
- f float (machine dependent size and representation)
- d double (machine dependent size and representation)
- x NUL byte
- X Back up one byte
- @ NUL-fill to absolute position

Példa 1. pack() format string

```
$binarydata = pack ("nvc*", 0x1234, 0x5678, 65, 66);
```

The resulting binary string will be 6 bytes long and contain the byte sequence 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.

Note that the distinction between signed and unsigned values only affects the function unpack(), where as function **pack()** gives the same result for signed and unsigned format codes.

Also note that PHP internally stores integer values as signed values of a machine dependent size. If you give it an unsigned integer value too large to be stored that way it is converted to a float which often yields an undesired result.

show_source (PHP 4)

Syntax highlighting of a file

bool **show_source** (string filename [, bool return]) \linebreak

This function is an alias to highlight_file(). For more information see the documentation there.

See also highlight_string() and highlight_file().

sleep (PHP 3, PHP 4)

Delay execution

void **sleep** (int seconds) \linebreak

The sleep function delays program execution for the given number of *seconds*.

See also usleep() and set_time_limit()

uniqid (PHP 3, PHP 4)

Generate a unique ID

string **uniqid** (string prefix [, bool lcg]) \linebreak

uniqid() returns a prefixed unique identifier based on the current time in microseconds. The prefix can be useful for instance if you generate identifiers simultaneously on several hosts that might happen to generate the identifier at the same microsecond. *Prefix* can be up to 114 characters long.

If the optional *lcg* parameter is TRUE, **uniqid()** will add additional "combined LCG" entropy at the end of the return value, which should make the results more unique.

With an empty *prefix*, the returned string will be 13 characters long. If *lcg* is TRUE, it will be 23 characters.

Megjegyzés: The *lcg* parameter is only available in PHP 4 and PHP 3.0.13 and later.

If you need a unique identifier or token and you intend to give out that token to the user via the network (i.e. session cookies), it is recommended that you use something along the lines of

```
$token = md5(uniqid("")); // no prefix
$better_token = md5(uniqid(rand(),1)); // better, difficult to guess
```

This will create a 32 character identifier (a 128 bit hex number) that is extremely difficult to predict.

unpack (PHP 3, PHP 4)

Unpack data from binary string

array **unpack** (string format, string data) \linebreak

unpack() from binary string into array according to *`format`*. Returns array containing unpacked elements of binary string.

unpack() works slightly different from Perl as the unpacked data is stored in an associative array. To accomplish this you have to name the different format codes and separate them by a slash /.

Példa 1. unpack() format string

```
$array = unpack ("c2chars/nint", $binarydata);
```

The resulting array will contain the entries "chars1", "chars2" and "int".

For an explanation of the format codes see also: pack()

Note that PHP internally stores integral values as signed. If you unpack a large unsigned long and it is of the same size as PHP internally stored values the result will be a negative number even though unsigned unpacking was specified.

usleep (PHP 3, PHP 4)

Delay execution in microseconds

void **usleep** (int micro_seconds) \linebreak

The **usleep()** function delays program execution for the given number of *`micro_seconds`*. A microsecond is one millionth of a second.

See also sleep() and set_time_limit().

Megjegyzés: This function does not work on Windows systems.

LXI. mnoGoSearch Functions

These functions allow you to access the mnoGoSearch (former UdmSearch) free search engine. In order to have these functions available, you must compile PHP with mnogosearch support by using the `--with-mnogosearch` option. If you use this option without specifying the path to mnogosearch, PHP will look for mnogosearch under /usr/local/mnogosearch path by default. If you installed mnogosearch at other path you should specify it: `--with-mnogosearch=DIR`.

mnoGoSearch is a full-featured search engine software for intranet and internet servers, distributed under the GNU license. mnoGoSearch has number of unique features, which makes it appropriate for a wide range of application from search within your site to a specialized search system such as cooking recipes or newspaper search, FTP archive search, news articles search, etc. It offers full-text indexing and searching for HTML, PDF, and text documents. mnoGoSearch consists of two parts. The first is an indexing mechanism (indexer). The purpose of indexer is to walk through HTTP, FTP, NEWS servers or local files, recursively grabbing all the documents and storing meta-data about that documents in a SQL database in a smart and effective manner. After every document is referenced by its corresponding URL, meta-data collected by indexer is used later in a search process. The search is performed via Web interface. C CGI, PHP and Perl search front ends are included.

> **Megjegyzés:** PHP contains built-in mysql access library, which can be used to access MySQL. It is known that mnoGoSearch is not compatible with this built-in library and can work only with generic MySQL libraries. Thus, if you use mnoGoSearch with mysql, during PHP configuration you have to indicate directory of MySQL installation, that was used during mnoGoSearch configuration, i.e. for example: `--with-mnogosearch --with-mysql=/usr`.

You need at least version 3.1.10 of mnoGoSearch installed to use these functions.

More information about mnoGoSearch can be found at http://www.mnogosearch.ru/.

udm_add_search_limit (PHP 4 >= 4.0.5)

Add various search limits

int **udm_add_search_limit** (int agent, int var, string val) \linebreak

udm_add_search_limit() returns `TRUE` on success, `FALSE` on error. Adds search restrictions.

`agent` - a link to Agent, received after call to udm_alloc_agent().

`var` - defines parameter, indicating limit.

`val` - defines value of the current parameter.

Possible *`var`* values:

- UDM_LIMIT_URL - defines document URL limitations to limit search through subsection of database. It supports SQL % and _ LIKE wildcards, where % matches any number of characters, even zero characters, and _ matches exactly one character. E.g. http://my.domain.__/catalog may stand for http://my.domain.ru/catalog and http://my.domain.ua/catalog.
- UDM_LIMIT_TAG - defines site TAG limitations. In indexer-conf you can assign specific TAGs to various sites and parts of a site. Tags in mnoGoSearch 3.1.x are lines, that may contain metasymbols % and _. Metasymbols allow searching among groups of tags. E.g. there are links with tags ABCD and ABCE, and search restriction is by ABC_ - the search will be made among both of the tags.
- UDM_LIMIT_LANG - defines document language limitations.
- UDM_LIMIT_CAT - defines document category limitations. Categories are similar to tag feature, but nested. So you can have one category inside another and so on. You have to use two characters for each level. Use a hex number going from 0-F or a 36 base number going from 0-Z. Therefore a top-level category like 'Auto' would be 01. If it has a subcategory like 'Ford', then it would be 01 (the parent category) and then 'Ford' which we will give 01. Put those together and you get 0101. If 'Auto' had another subcategory named 'VW', then it's id would be 01 because it belongs to the 'Ford' category and then 02 because it's the next category. So it's id would be 0102. If VW had a sub category called 'Engine' then it's id would start at 01 again and it would get the 'VW' id 02 and 'Auto' id of 01, making it 010201. If you want to search for sites under that category then you pass it cat=010201 in the url.
- UDM_LIMIT_DATE - defines limitation by date document was modified.

  Format of parameter value: a string with first character < or >, then with no space - date in unixtime format, for example:

  Udm_Add_Search_Limit($udm,UDM_LIMIT_DATE,"<908012006");

  If > character is used, then search will be restricted to those documents having modification date greater than entered. If <, then smaller.

udm_alloc_agent (PHP 4 >= 4.0.5)

Allocate mnoGoSearch session

int **udm_alloc_agent** (string dbaddr [, string dbmode]) \linebreak

udm_alloc_agent() returns mnogosearch agent identifier on success, `FALSE` on error. This function creates a session with database parameters.

`dbaddr` - URL-style database description. Options (type, host, database name, port, user and password) to connect to SQL database. Do not matter for built-in text files support. Format: DBAddr DBType:[//[DBUser[:DBPass]@]DBHost[:DBPort]]/DBName/ Currently supported DBType values are: mysql, pgsql, msql, solid, mssql, oracle, ibase. Actually, it does not matter for native libraries support. But ODBC users should specify one of supported values. If your database type is not supported, you may use "unknown" instead.

`dbmode` - You may select SQL database mode of words storage. When "single" is specified, all words are stored in the same table. If "multi" is selected, words will be located in different tables depending of their lengths. "multi" mode is usually faster but requires more tables in database. If "crc" mode is selected, mnoGoSearch will store 32 bit integer word IDs calculated by CRC32 algorythm instead of words. This mode requres less disk space and it is faster comparing with "single" and "multi" modes. "crc-multi" uses the same storage structure with the "crc" mode, but also stores words in different tables depending on words lengths like "multi" mode. Format: DBMode single/multi/crc/crc-multi

Megjegyzés: *`dbaddr`* and *`dbmode`* must match those used during indexing.

Megjegyzés: In fact this function does not open connection to database and thus does not check entered login and password. Actual connection to database and login/password verification is done by udm_find().

udm_api_version (PHP 4 >= 4.0.5)

Get mnoGoSearch API version.

int **udm_api_version** (void) \linebreak

udm_api_version() returns mnoGoSearch API version number. E.g. if mnoGoSearch 3.1.10 API is used, this function will return `30110`.

This function allows user to identify which API functions are available, e.g. udm_get_doc_count() function is only available in mnoGoSearch 3.1.11 or later.

Example:

```
if (udm_api_version() >= 30111) {
print  "Total number of urls in database: ".udm_get_doc_count($udm)."<br>\n";
}
```

udm_cat_list (PHP 4 >= 4.0.6)

Get all the categories on the same level with the current one.

array **udm_cat_list** (int agent, string category) \linebreak

udm_cat_list() returns array listing all categories of the same level as current category in the categories tree.

The function can be useful for developing categories tree browser.

Returns array with the following format:

The array consists of pairs. Elements with even index numbers contain category paths, odd elements contain corresponding category names.

$array[0] will contain ’020300’
$array[1] will contain ’Audi’
$array[2] will contain ’020301’
$array[3] will contain ’BMW’
$array[4] will contain ’020302’
$array[5] will contain ’Opel’
...
etc.

Following is an example of displaying links of the current level in format:
Audi
BMW
Opel
...

```
<?php
 $cat_list_arr = udm_cat_list($udm_agent,$cat);
 $cat_list = ”;
 for ($i=0; $i<count($cat_list_arr); $i+=2) {
   $path = $cat_list_arr[$i];
   $name = $cat_list_arr[$i+1];
   $cat_list .= "<a href=\"$PHP_SELF?cat=$path\">$name</a><br>";
 }
?>
```

udm_cat_path (PHP 4 >= 4.0.6)

Get the path to the current category.

array **udm_cat_path** (int agent, string category) \linebreak

udm_cat_path() returns array describing path in the categories tree from the tree root to the current category.

agent - agent link identifier.

category - current category - the one to get path to.

Returns array with the following format:

The array consists of pairs. Elements with even index numbers contain category paths, odd elements contain corresponding category names.

For example, the call `$array=udm_cat_path($agent, '02031D');` may return the following array:

$array[0] will contain ”
$array[1] will contain ’Root’
$array[2] will contain ’02’
$array[3] will contain ’Sport’
$array[4] will contain ’0203’
$array[5] will contain ’Auto’
$array[4] will contain ’02031D’
$array[5] will contain ’Ferrari’

Példa 1. Specifying path to the current category in the following format: ’> Root > Sport > Auto > Ferrari’

```
<?php
  $cat_path_arr = udm_cat_path($udm_agent,$cat);
  $cat_path = ”;
  for ($i=0; $i<count($cat_path_arr); $i+=2) {
    $path = $cat_path_arr[$i];
    $name = $cat_path_arr[$i+1];
    $cat_path .= " > <a href=\"$PHP_SELF?cat=$path\">$name</a> ";
  }
?>
```

udm_check_charset (PHP 4 >= 4.2.0)

Check if the given charset is known to mnogosearch

int **udm_check_charset** (int agent, string charset) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

udm_check_stored (PHP 4 >= 4.2.0)

Check connection to stored

int **udm_check_stored** (int agent, int link, string doc_id) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

udm_clear_search_limits (PHP 4 >= 4.0.5)

Clear all mnoGoSearch search restrictions

int **udm_clear_search_limits** (int agent) \linebreak

udm_clear_search_limits() resets defined search limitations and returns TRUE.

udm_close_stored (PHP 4 >= 4.2.0)

Close connection to stored

int **udm_close_stored** (int agent, int link) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

udm_crc32 (PHP 4 >= 4.2.0)

Return CRC32 checksum of gived string

int **udm_crc32** (int agent, string str) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

udm_errno (PHP 4 >= 4.0.5)

Get mnoGoSearch error number

int **udm_errno** (int agent) \linebreak

udm_errno() returns mnoGoSearch error number, zero if no error.

agent - link to agent identifier, received after call to udm_alloc_agent().

Receiving numeric agent error code.

udm_error (PHP 4 >= 4.0.5)

Get mnoGoSearch error message

string **udm_error** (int agent) \linebreak

udm_error() returns mnoGoSearch error message, empty string if no error.

agent - link to agent identifier, received after call to udm_alloc_agent().

Receiving agent error message.

udm_find (PHP 4 >= 4.0.5)

Perform search

int **udm_find** (int agent, string query) \linebreak

udm_find() returns result link identifier on success, FALSE on error.

The search itself. The first argument - session, the next one - query itself. To find something just type words you want to find and press SUBMIT button. For example, "mysql odbc". You should not use quotes " in query, they are written here only to divide a query from other text. mnoGoSearch will find all documents that contain word "mysql" and/or word "odbc". Best documents having bigger weights will be displayed first. If you use search mode ALL, search will return documents that

contain both (or more) words you entered. In case you use mode ANY, the search will return list of documents that contain any of the words you entered. If you want more advanced results you may use query language. You should select "bool" match mode in the search from.

mnoGoSearch understands the following boolean operators:

& - logical AND. For example, "mysql & odbc". mnoGoSearch will find any URLs that contain both "mysql" and "odbc".

| - logical OR. For example "mysql|odbc". mnoGoSearch will find any URLs, that contain word "mysql" or word "odbc".

~ - logical NOT. For example "mysql & ~odbc". mnoGoSearch will find URLs that contain word "mysql" and do not contain word "odbc" at the same time. Note that ~ just excludes given word from results. Query "~odbc" will find nothing!

() - group command to compose more complex queries. For example "(mysql | msql) & ~postgres". Query language is simple and powerful at the same time. Just consider query as usual boolean expression.

udm_free_agent (PHP 4 >= 4.0.5)

Free mnoGoSearch session

int **udm_free_agent** (int agent) \linebreak

udm_free_agent() returns TRUE on success, FALSE on error.

agent - link to agent identifier, received ‘ after call to udm_alloc_agent().

Freeing up memory allocated for agent session.

udm_free_ispell_data (PHP 4 >= 4.0.5)

Free memory allocated for ispell data

int **udm_free_ispell_data** (int agent) \linebreak

udm_free_ispell_data() always returns TRUE.

agent - agent link identifier, received after call to udm_alloc_agent().

> **Megjegyzés:** This function is supported beginning from version 3.1.12 of mnoGoSearch and it does not do anything in previous versions.

udm_free_res (PHP 4 >= 4.0.5)

Free mnoGoSearch result

int **udm_free_res** (int res) \linebreak

udm_free_res() returns TRUE on success, FALSE on error.

res - a link to result identifier, received after call to udm_find().

Freeing up memory allocated for results.

udm_get_doc_count (PHP 4 >= 4.0.5)

Get total number of documents in database.

int **udm_get_doc_count** (int agent) \linebreak

udm_get_doc_count() returns number of documents in database.

agent - link to agent identifier, received after call to udm_alloc_agent().

Megjegyzés: This function is supported only in mnoGoSearch 3.1.11 or later.

udm_get_res_field (PHP 4 >= 4.0.5)

Fetch mnoGoSearch result field

string **udm_get_res_field** (int res, int row, int field) \linebreak

udm_get_res_field() returns result field value on success, FALSE on error.

res - a link to result identifier, received after call to udm_find().

row - the number of the link on the current page. May have values from 0 to *UDM_PARAM_NUM_ROWS-1*.

field - field identifier, may have the following values:

- UDM_FIELD_URL - document URL field
- UDM_FIELD_CONTENT - document Content-type field (for example, text/html).
- UDM_FIELD_CATEGORY - document category field. Use udm_cat_path() to get full path to current category from the categories tree root. (This parameter is available only in PHP 4.0.6 or later).
- UDM_FIELD_TITLE - document title field.
- UDM_FIELD_KEYWORDS - document keywords field (from META KEYWORDS tag).
- UDM_FIELD_DESC - document description field (from META DESCRIPTION tag).
- UDM_FIELD_TEXT - document body text (the first couple of lines to give an idea of what the document is about).
- UDM_FIELD_SIZE - document size.
- UDM_FIELD_URLID - unique URL ID of the link.
- UDM_FIELD_RATING - page rating (as calculated by mnoGoSearch).
- UDM_FIELD_MODIFIED - last-modified field in unixtime format.

- UDM_FIELD_ORDER - the number of the current document in set of found documents.
- UDM_FIELD_CRC - document CRC.

udm_get_res_param (PHP 4 >= 4.0.5)

Get mnoGoSearch result parameters

string **udm_get_res_param** (int res, int param) \linebreak

udm_get_res_param() returns result parameter value on success, FALSE on error.

res - a link to result identifier, received after call to udm_find().

param - parameter identifier, may have the following values:

- UDM_PARAM_NUM_ROWS - number of received found links on the current page. It is equal to UDM_PARAM_PAGE_SIZE for all search pages, on the last page - the rest of links.
- UDM_PARAM_FOUND - total number of results matching the query.
- UDM_PARAM_WORDINFO - information on the words found. E.g. search for "a good book" will return "a: stopword, good:5637, book: 120"
- UDM_PARAM_SEARCHTIME - search time in seconds.
- UDM_PARAM_FIRST_DOC - the number of the first document displayed on current page.
- UDM_PARAM_LAST_DOC - the number of the last document displayed on current page.

udm_load_ispell_data (PHP 4 >= 4.0.5)

Load ispell data

int **udm_load_ispell_data** (int agent, int var, string val1, string val2, int flag) \linebreak

udm_load_ispell_data() loads ispell data. Returns TRUE on success, FALSE on error.

agent - agent link identifier, received after call to udm_alloc_agent().

var - parameter, indicating the source for ispell data. May have the following values:

After using this function to free memory allocated for ispell data, please use udm_free_ispell_data(), even if you use UDM_ISPELL_TYPE_SERVER mode.

The fastest mode is UDM_ISPELL_TYPE_SERVER. UDM_ISPELL_TYPE_TEXT is slower and UDM_ISPELL_TYPE_DB is the slowest. The above pattern is TRUE for mnoGoSearch 3.1.10 - 3.1.11. It is planned to speed up DB mode in future versions and it is going to be faster than TEXT mode.

- UDM_ISPELL_TYPE_DB - indicates that ispell data should be loaded from SQL. In this case, parameters *val1* and *val2* are ignored and should be left blank. *flag* should be equal to 1.

> **Megjegyzés:** *flag* indicates that after loading ispell data from defined source it sould be sorted (it is necessary for correct functioning of ispell). In case of loading ispell data from files there may be several calls to **udm_load_ispell_data()**, and there is no sense to sort data

after every call, but only after the last one. Since in db mode all the data is loaded by one call, this parameter should have the value `1`. In this mode in case of error, e.g. if ispell tables are absent, the function will return `FALSE` and code and error message will be accessible through udm_error() and udm_errno().

Example:

```
if (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_DB,”,”,1)) {
  printf("Error #%d: ’%s’\n", udm_errno($udm), udm_error($udm));
  exit;
}
```

- UDM_ISPELL_TYPE_AFFIX - indicates that ispell data should be loaded from file and initiates loading affixes file. In this case *`val1`* defines double letter language code for which affixes are loaded, and *`val2`* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return `FALSE`, and an error message will be displayed. Error message text cannot be accessed through udm_error() and udm_errno(), since those functions can only return messages associated with SQL. Please, see *`flag`* parameter description in UDM_ISPELL_TYPE_DB.

  Example:

  ```
  if ((! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_AFFIX,’en’,’/opt/ispell/en.aff’,0
     (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_AFFIX,’ru’,’/opt/ispell/ru.aff’,0)
     (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SPELL,’en’,’/opt/ispell/en.dict’,0
     (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SPELL,’ru’,’/opt/ispell/ru.dict’,1
  exit;
  }
  ```

  Megjegyzés: *`flag`* is equal to `1` only in the last call.

- UDM_ISPELL_TYPE_SPELL - indicates that ispell data should be loaded from file and initiates loading of ispell dictionary file. In this case *`val1`* defines double letter language code for which affixes are loaded, and *`val2`* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return `FALSE`, and an error message will be displayed. Error message text cannot be accessed through udm_error() and udm_errno(), since those functions can only return messages associated with SQL. Please, see *`flag`* parameter description in UDM_ISPELL_TYPE_DB.

  ```
  if ((! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,’en’,’/opt/ispell/en.aff’,0
  ```

```
  (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'ru','/opt/ispell/ru.aff',0)
  (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'en','/opt/ispell/en.dict',0
  (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'ru','/opt/ispell/ru.dict',1
exit;
}
```

Megjegyzés: *flag* is equal to 1 only in the last call.

- UDM_ISPELL_TYPE_SERVER - enables spell server support. *val1* parameter indicates address of the host running spell server. *val2* ' is not used yet, but in future releases it is going to indicate number of port used by spell server. *flag* parameter in this case is not needed since ispell data is stored on spellserver already sorted.

  Spelld server reads spell-data from a separate configuration file (/usr/local/mnogosearch/etc/spelld.conf by default), sorts it and stores in memory. With clients server communicates in two ways: to indexer all the data is transferred (so that indexer starts faster), from search.cgi server receives word to normalize and then passes over to client (search.cgi) list of normalized word forms. This allows fastest, compared to db and text modes processing of search queries (by omitting loading and sorting all the spell data).

  udm_load_ispell_data() function in UDM_ISPELL_TYPE_SERVER mode does not actually load ispell data, but only defines server address. In fact, server is automatically used by udm_find() function when performing search. In case of errors, e.g. if spellserver is not running or invalid host indicated, there are no messages returned and ispell conversion does not work.

  Megjegyzés: This function is available in mnoGoSearch 3.1.12 or later.

  Example:

```
if (!udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SERVER,"","",1)) {
    printf("Error loading ispell data from server<br>\n");
    exit;
}
```

udm_open_stored (PHP 4 >= 4.2.0)

Open connection to stored

int **udm_open_stored** (int agent, string storedaddr) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

udm_set_agent_param (PHP 4 >= 4.0.5)

Set mnoGoSearch agent session parameters

int **udm_set_agent_param** (int agent, int var, string val) \linebreak

udm_set_agent_param() returns `TRUE` on success, `FALSE` on error. Defines mnoGoSearch session parameters.

The following parameters and their values are available:

- UDM_PARAM_PAGE_NUM - used to choose search results page number (results are returned by pages beginning from 0, with UDM_PARAM_PAGE_SIZE results per page).
- UDM_PARAM_PAGE_SIZE - number of search results displayed on one page.
- UDM_PARAM_SEARCH_MODE - search mode. The following values available: UDM_MODE_ALL - search for all words; UDM_MODE_ANY - search for any word; UDM_MODE_PHRASE - phrase search; UDM_MODE_BOOL - boolean search. See udm_find() for details on boolean search.
- UDM_PARAM_CACHE_MODE - turns on or off search result cache mode. When enabled, the search engine will store search results to disk. In case a similar search is performed later, the engine will take results from the cache for faster performance. Available values: UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - turns on or off trackquery mode. Since version 3.1.2 mnoGoSearch has a query tracking support. Note that tracking is implemented in SQL version only and not available in built-in database. To use tracking, you have to create tables for tracking support. For MySQL, use create/mysql/track.txt. When doing a search, front-end uses those tables to store query words, a number of found documents and current UNIX timestamp in seconds. Available values: UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.
- UDM_PARAM_PHRASE_MODE - defines whether index database using phrases ("phrase" parameter in indexer.conf). Possible values: UDM_PHRASE_ENABLED and UDM_PHRASE_DISABLED. Please note, that if phrase search is enabled (UDM_PHRASE_ENABLED), it is still possible to do search in any mode (ANY, ALL, BOOL or PHRASE). In 3.1.10 version of mnoGoSearch phrase search is supported only in sql and built-in database modes, while beginning with 3.1.11 phrases are supported in cachemode as well.

  Examples of phrase search:

  "Arizona desert" - This query returns all indexed documents that contain "Arizona desert" as a phrase. Notice that you need to put double quotes around the phrase

- UDM_PARAM_CHARSET - defines local charset. Available values: set of charsets supported by mnoGoSearch, e.g. koi8-r, cp1251, ...

- UDM_PARAM_STOPFILE - Defines name and path to stopwords file. (There is a small difference with mnoGoSearch - while in mnoGoSearch if relative path or no path entered, it looks for this file in relation to UDM_CONF_DIR, the module looks for the file in relation to current path, i.e. to the path where the php script is executed.)
- UDM_PARAM_STOPTABLE - Load stop words from the given SQL table. You may use several StopwordTable commands. This command has no effect when compiled without SQL database support.
- UDM_PARAM_WEIGHT_FACTOR - represents weight factors for specific document parts. Currently body, title, keywords, description, url are supported. To activate this feature please use degrees of 2 in *Weight commands of the indexer.conf. Let's imagine that we have these weights:

  URLWeight 1
  BodyWeight 2
  TitleWeight 4
  KeywordWeight 8
  DescWeight 16

  As far as indexer uses bit OR operation for word weights when some word presents several time in the same document, it is possible at search time to detect word appearance in different document parts. Word which appears only in the body will have 00000010 argegate weight (in binary notation). Word used in all document parts will have 00011111 aggregate weight.

  This parameter's value is a string of hex digits ABCDE. Each digit is a factor for corresponding bit in word weight. For the given above weights configuration:

  E is a factor for weight 1 (URL Weight bit)
  D is a factor for weight 2 (BodyWeight bit)
  C is a factor for weight 4 (TitleWeight bit)
  B is a factor for weight 8 (KeywordWeight bit)
  A is a factor for weight 16 (DescWeight bit)

  Examples:

  UDM_PARAM_WEIGHT_FACTOR=00001 will search through URLs only.

  UDM_PARAM_WEIGHT_FACTOR=00100 will search through Titles only.

  UDM_PARAM_WEIGHT_FACTOR=11100 will search through Title,Keywords,Description but not through URL and Body.

  UDM_PARAM_WEIGHT_FACTOR=F9421 will search through:

  Description with factor 15 (F hex)
  Keywords with factor 9
  Title with factor 4
  Body with factor 2
  URL with factor 1

  If UDM_PARAM_WEIGHT_FACTOR variable is ommited, original weight value is taken to sort results. For a given above weight configuration it means that document description has a most big weight 16.

- UDM_PARAM_WORD_MATCH - word match. You may use this parameter to choose word match type. This feature works only in "single" and "multi" modes using SQL based and built-in

database. It does not work in cachemode and other modes since they use word CRC and do not support substring search. Available values:

UDM_MATCH_BEGIN - word beginning match;

UDM_MATCH_END - word ending match;

UDM_MATCH_WORD - whole word match;

UDM_MATCH_SUBSTR - word substring match.

- UDM_PARAM_MIN_WORD_LEN - defines minimal word length. Any word shorter this limit is considered to be a stopword. Please note that this parameter value is inclusive, i.e. if UDM_PARAM_MIN_WORD_LEN=3, a word 3 characters long will not be considered a stopword, while a word 2 characters long will be. Default value is 1.
- UDM_PARAM_ISPELL_PREFIXES - Possible values: UDM_PREFIXES_ENABLED and UDM_PREFIXES_DISABLED, that respectively enable or disable using prefixes. E.g. if a word "tested" is in search query, also words like "test", "testing", etc. Only suffixes are supported by default. Prefixes usually change word meanings, for example if somebody is searching for the word "tested" one hardly wants "untested" to be found. Prefixes support may also be found useful for site’s spelling checking purposes. In order to enable ispell, you have to load ispell data with udm_load_ispell_data().
- UDM_PARAM_CROSS_WORDS - enables or disables crosswords support. Possible values: UDM_CROSS_WORDS_ENABLED and UDM_CROSS_WORDS_DISABLED.

  The corsswords feature allows to assign words between and also to a document this link leads to. It works in SQL database mode and is not supported in built-in database and Cachemode.

  Megjegyzés: Crosswords are supported only in mnoGoSearch 3.1.11 or later.

- UDM_PARAM_VARDIR - specifies a custom path to directory where indexer stores data when using built-in database and in cache mode. By default `/var` directory of mnoGoSearch installation is used. Can have only string values. The parameter is available in PHP 4.1.0 or later.
- UDM_PARAM_VARDIR - specifies a custom path to directory where indexer stores data when using built-in database and in cache mode. By default `/var` directory of mnoGoSearch installation is used. Can have only string values. The parameter is available in PHP 4.1.0 or later.

LXII. mSQL functions

These functions allow you to access mSQL database servers. In order to have these functions available, you must compile PHP with msql support by using the `--with-msql[=dir]` option. The default location is /usr/local/Hughes.

More information about mSQL can be found at http://www.hughes.com.au/.

msql_affected_rows (PHP 3>= 3.0.6, PHP 4)

Returns number of affected rows

int **msql_affected_rows** (int query_identifier) \linebreak

Returns number of affected ("touched") rows by a specific query (i.e. the number of rows returned by a SELECT, the number of rows modified by an update, or the number of rows removed by a delete).

See also: msql_query().

msql_close (PHP 3, PHP 4)

Close mSQL connection

int **msql_close** (int link_identifier) \linebreak

Returns TRUE on success, FALSE on error.

msql_close() closes the link to a mSQL database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

msql_close() will not close persistent links generated by msql_pconnect().

See also: msql_connect() and msql_pconnect().

msql_connect (PHP 3, PHP 4)

Open mSQL connection

int **msql_connect** ([string hostname [, string server [, string username [, string password]]]]) \linebreak

Returns a positive mSQL link identifier on success, or FALSE on error.

msql_connect() establishes a connection to a mSQL server. The *server* parameter can also include a port number. eg. "hostname:port". It defaults to 'localhost'.

In case a second call is made to **msql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling msql_close().

See also msql_pconnect(), msql_close().

msql_create_db (PHP 3, PHP 4)

Create mSQL database

int **msql_create_db** (string database name [, int link_identifier]) \linebreak

msql_create_db() attempts to create a new database on the server associated with the specified link identifier.

See also: msql_drop_db().

msql_createdb (PHP 3, PHP 4)

Create mSQL database

int **msql_createdb** (string database name [, int link_identifier]) \linebreak

Identical to msql_create_db().

msql_data_seek (PHP 3, PHP 4)

Move internal row pointer

int **msql_data_seek** (int query_identifier, int row_number) \linebreak

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

msql_data_seek() moves the internal row pointer of the mSQL result associated with the specified query identifier to pointer to the specifyed row number. The next call to msql_fetch_row() would return that row.

See also: msql_fetch_row().

msql_dbname (PHP 3, PHP 4)

Get current mSQL database name

string **msql_dbname** (int query_identifier, int i) \linebreak

msql_dbname() returns the database name stored in position *`i`* of the result pointer returned from the msql_listdbs() function. The msql_numrows() function can be used to determine how many database names are available.

msql_drop_db (PHP 3, PHP 4)

Drop (delete) mSQL database

int **msql_drop_db** (string database_name, int link_identifier) \linebreak

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

msql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

See also: msql_create_db().

msql_dropdb (PHP 3, PHP 4)

Drop (delete) mSQL database

See msql_drop_db().

msql_error (PHP 3, PHP 4)

Returns error message of last msql call

string **msql_error** ([int link_identifier]) \linebreak

Errors coming back from the mSQL database backend no longer issue warnings. Instead, use these functions to retrieve the error string.

msql_fetch_array (PHP 3, PHP 4)

Fetch row as array

int **msql_fetch_array** (int query_identifier [, int result_type]) \linebreak

Returns an array that corresponds to the fetched row, or `FALSE` if there are no more rows.

msql_fetch_array() is an extended version of msql_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

The second optional argument *`result_type`* in **msql_fetch_array()** is a constant and can take the following values: MSQL_ASSOC, MSQL_NUM, and MSQL_BOTH.

Be careful if you are retrieving results from a query that may return a record that contains only one field that has a value of 0 (or an empty string, or `NULL`).

An important thing to note is that using **msql_fetch_array()** is NOT significantly slower than using msql_fetch_row(), while it provides a significant added value.

For further details, also see msql_fetch_row().

msql_fetch_field (PHP 3, PHP 4)

Get field information

object **msql_fetch_field** (int query_identifier, int field_offset) \linebreak

Returns an object containing field information

msql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by **msql_fetch_field()** is retreived.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- not_null - 1 if the column cannot be `NULL`
- primary_key - 1 if the column is a primary key
- unique - 1 if the column is a unique key
- type - the type of the column

See also msql_field_seek().

msql_fetch_object (PHP 3, PHP 4)

Fetch row as object

int **msql_fetch_object** (int query_identifier [, int result_type]) \linebreak

Returns an object with properties that correspond to the fetched row, or `FALSE` if there are no more rows.

msql_fetch_object() is similar to msql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional second argument *`result_type`* in msql_fetch_array() is a constant and can take the following values: MSQL_ASSOC, MSQL_NUM, and MSQL_BOTH.

Speed-wise, the function is identical to msql_fetch_array(), and almost as quick as msql_fetch_row() (the difference is insignificant).

See also: msql_fetch_array() and msql_fetch_row().

msql_fetch_row (PHP 3, PHP 4)

Get row as enumerated array

array **msql_fetch_row** (int query_identifier) \linebreak

Returns an array that corresponds to the fetched row, or `FALSE` if there are no more rows.

msql_fetch_row() fetches one row of data from the result associated with the specified query identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **msql_fetch_row()** would return the next row in the result set, or `FALSE` if there are no more rows.

See also: msql_fetch_array(), msql_fetch_object(), msql_data_seek(), and msql_result().

msql_field_seek (PHP 3, PHP 4)

Set field offset

int **msql_field_seek** (int query_identifier, int field_offset) \linebreak

Seeks to the specified field offset. If the next call to msql_fetch_field() won't include a field offset, this field would be returned.

See also: msql_fetch_field().

msql_fieldflags (PHP 3, PHP 4)

Get field flags

string **msql_fieldflags** (int query_identifier, int i) \linebreak

msql_fieldflags() returns the field flags of the specified field. Currently this is either, "not `NULL`", "primary key", a combination of the two or "" (an empty string).

msql_fieldlen (PHP 3, PHP 4)

Get field length

int **msql_fieldlen** (int query_identifier, int i) \linebreak

msql_fieldlen() returns the length of the specified field.

msql_fieldname (PHP 3, PHP 4)

Get field name

string **msql_fieldname** (int query_identifier, int field) \linebreak

msql_fieldname() returns the name of the specified field. *`query_identifier`* is the query identifier, and *`field`* is the field index. `msql_fieldname($result, 2);` will return the name of the second field in the result associated with the result identifier.

msql_fieldtable (PHP 3, PHP 4)

Get table name for field

int **msql_fieldtable** (int query_identifier, int field) \linebreak

Returns the name of the table *`field`* was fetched from.

msql_fieldtype (PHP 3, PHP 4)

Get field type

string **msql_fieldtype** (int query_identifier, int i) \linebreak

msql_fieldtype() is similar to the msql_fieldname() function. The arguments are identical, but the field type is returned. This will be one of "int", "char" or "real".

msql_free_result (PHP 3, PHP 4)

Free result memory

int **msql_free_result** (int query_identifier) \linebreak

msql_free_result() frees the memory associated with *`query_identifier`*. When PHP completes a request, this memory is freed automatically, so you only need to call this function when you want to make sure you don’t use too much memory while the script is running.

msql_freeresult (PHP 3, PHP 4)

Free result memory

See msql_free_result()

msql_list_dbs (PHP 3, PHP 4)

List mSQL databases on server

int **msql_list_dbs** (void) \linebreak

msql_list_dbs() will return a result pointer containing the databases available from the current msql daemon. Use the msql_dbname() function to traverse this result pointer.

msql_list_fields (PHP 3, PHP 4)

List result fields

int **msql_list_fields** (string database, string tablename) \linebreak

msql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with msql_fieldflags(), msql_fieldlen(), msql_fieldname(), and msql_fieldtype(). A query identifier is a positive integer. The function returns `-1` if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@msql_list_fields()` then this error string will also be printed out.

See also msql_error().

msql_list_tables (PHP 3, PHP 4)

List tables in an mSQL database

int **msql_list_tables** (string database) \linebreak

msql_list_tables() takes a database name and result pointer much like the msql() function. The msql_tablename() function should be used to extract the actual table names from the result pointer.

msql_listdbs (PHP 3, PHP 4)

List mSQL databases on server

See msql_list_dbs().

msql_listfields (PHP 3, PHP 4)

List result fields

See msql_list_fields().

msql_listtables (PHP 3, PHP 4)

List tables in an mSQL database

See msql_list_tables().

msql_num_fields (PHP 3, PHP 4)

Get number of fields in result

int **msql_num_fields** (int query_identifier) \linebreak

msql_num_fields() returns the number of fields in a result set.

See also: msql(), msql_query(), msql_fetch_field(), and msql_num_rows().

msql_num_rows (PHP 3, PHP 4)

Get number of rows in result

int **msql_num_rows** (int query_identifier) \linebreak

msql_num_rows() returns the number of rows in a result set.

See also: msql(), msql_query(), and msql_fetch_row().

msql_numfields (PHP 3, PHP 4)

Get number of fields in result

int **msql_numfields** (int query_identifier) \linebreak

Identical to msql_num_fields().

msql_numrows (PHP 3, PHP 4)

Get number of rows in result

int **msql_numrows** (void) \linebreak

Identical to msql_num_rows().

msql_pconnect (PHP 3, PHP 4)

Open persistent mSQL connection

int **msql_pconnect** ([string server [, string username [, string password]]]) \linebreak

Returns a positive mSQL persistent link identifier on success, or FALSE on error.

msql_pconnect() acts very much like msql_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (msql_close() will not close links established by **msql_pconnect()**).

This type of links is therefore called 'persistent'.

msql_query (PHP 3, PHP 4)

Send mSQL query

int **msql_query** (string query, int link_identifier) \linebreak

msql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if msql_connect() was called, and use it.

Returns a positive mSQL query identifier on success, or `FALSE` on error.

Példa 1. msql_query()

```
<?php
$link = msql_connect("dbserver")
   or die("unable to connect to msql server: ".msql_error());
msql_select_db("db", $link)
   or die("unable to select database 'db': ".msql_error());

$result = msql_query("SELECT * FROM table WHERE id=1", $link);
if (!$result) {
   die("query failed: ".msql_error());
}

while ($row = msql_fetch_array($result)) {
    echo $row["id"];
}
?>
```

See also: msql(), msql_select_db(), and msql_connect().

msql_regcase (PHP 3, PHP 4)

Make regular expression for case insensitive match

See sql_regcase().

msql_result (PHP 3, PHP 4)

Get result data

int **msql_result** (int query_identifier, int i, mixed field) \linebreak

Returns the contents of the cell at the row and offset in the specified mSQL result set.

msql_result() returns the contents of one cell from a mSQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (fieldname.tablename). If the column name has been aliased ('select foo as bar from ...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **msql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: msql_fetch_row(), msql_fetch_array(), and msql_fetch_object().

msql_select_db (PHP 3, PHP 4)

Select mSQL database

int **msql_select_db** (string database_name, int link_identifier) \linebreak

Returns TRUE on success, FALSE on error.

msql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if msql_connect() was called, and use it.

Every subsequent call to msql_query() will be made on the active database.

See also: msql_connect(), msql_pconnect(), and msql_query().

msql_selectdb (PHP 3, PHP 4)

Select mSQL database

See msql_select_db().

msql_tablename (PHP 3, PHP 4)

Get table name of field

string **msql_tablename** (int query_identifier, int field) \linebreak

msql_tablename() takes a result pointer returned by the msql_list_tables() function as well as an integer index and returns the name of a table. The msql_numrows() function may be used to determine the number of tables in the result pointer.

Példa 1. msql_tablename() example

```
<?php
msql_connect ("localhost");
$result = msql_list_tables ("wisconsin");
```

```
$i = 0;
while ($i < msql_numrows ($result)) {
    $tb_names[$i] = msql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

msql (PHP 3, PHP 4)

Send mSQL query

int **msql** (string database, string query, int link_identifier) \linebreak

Returns a positive mSQL query identifier to the query result, or FALSE on error.

msql() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the mSQL server and if no such link is found it'll try to create one as if msql_connect() was called with no arguments (see msql_connect()).

LXIII. MySQL függvények

E függvények segítségével érheted el a MySQL adatbázisokat. Ahhoz, hogy ezeket a függvényeket használhasd, a php-t a `--with-mysql` opcióval kell lefordítanod. Ha nem adod meg az opció után a mysql elérési útvonalát, akkor a php a beépített mysql kliens könyvtárat fogja használni. Ha a felhasználók más alkalmazásokat is használnak, amelyek mysql-t használnak (például a php 3-as és 4-es verzióját, vagy az auth-mysql-t), akkor mindig meg kell adni a mysql elérési útvonalát a `--with-mysql=/path/to/mysql` formában. Ez ki fogja kényszeríteni, hogy a meglévő mysql kliens könyvtárat használja a php, elkerülve az ebből adódó ütközéseket.

A MySQL-ről további információt a http://www.mysql.com/ címen találsz.

A MySQL dokumentációját pedig a http://www.mysql.com/documentation/.

Az alábbi kis példa bemutatja, hogyan lehet MySQL adatbázisokhoz csatlakozni, kérést végrehajtani, kiíratni az eredményt és megszüntetni a kapcsolatot.

Példa 1. MySQL modul áttekintő példa

```
<?php
// Csatlakozás, adatbázis kiválasztása
$kapcsolat = mysql_connect("mysql_hoszt", "mysql_azonosito", "mysql_jelszo")
    or die("Nem tudok csatlakozni");
print "A kapcsolódás sikerül";
mysql_select_db("az_en_adatbazisom")
    or die("Nem sikerült kiválasztanom az adatbázist");

// SQL kérés végrehajtása
$keres = "SELECT * FROM az_en_tablam";
$eredmeny = mysql_query($keres) or die("Hiba a kérésben");

// Az eredmény kiírása HTML-ben
print "<table>\n";
while ($line = mysql_fetch_array($eredmeny, MYSQL_ASSOC)) {
    print "\t<tr>\n";
    foreach ($sor as $egy_oszlop) {
        print "\t\t<td>$egy_oszlop</td>\n";
    }
    print "\t</tr>\n";
}
print "</table>\n";

// Kapcsolat lezárása
mysql_close($kapcsolat);
?>
```

mysql_affected_rows (PHP 3, PHP 4)

Az előző MySQL műveletben érintett sorok számát adja meg

int **mysql_affected_rows** ([resource link_identifier]) \linebreak

A **mysql_affected_rows()** függvény az előző *link_identifier*-rel azonosított kapcsolatban kiadott INSERT, UPDATE vagy DELETE kérésben érintett sorok számát adja vissza. Ha nem adsz meg kapcsolat-azonosítót, akkor az utolsó mysql_connect()-tel megnyitott kapcsolatról kapsz információt.

Megjegyzés: Ha transzakciókezelést használsz, akkor a **mysql_affected_rows()** függvényt az INSERT, UPDATE, vagy DELETE kérés után közvetlenül, nem a kommit után kell meghívni.

Ha az utolsó lekérdezés egy WHERE feltétel nélküli DELETE volt, akkor bár a táblából minden sor törlődik, a függvény nullát fog visszaadni.

Megjegyzés: Ha UPDATE-tel használod, a MySQL nem fogja azokat a sorokat frissíteni, ahol a sor régi és új értéke megegyezik. Így nem kizárt, hogy a **mysql_affected_rows()** függvény nem pont az egyező sorok számát adja vissza, hanem csak a ténylegesen megváltoztatott sorok számát.

A függvény nem működik a lekérdező SELECT kéréssel együtt használva, csak azokkal a lekérdezésekkel, melyek rekordokat módosítanak. Ha egy SELECT lekérdezésben kapott sorok számát szeretnéd megkapni, akkor használd a mysql_num_rows() függvényt.

Ha az utolsó kérés meghiusult, akkor a függvény -1-gyel tér vissza.

Lásd még amysql_num_rows() függvényt!

mysql_change_user (PHP 3>= 3.0.13)

Aktív kapcsolat felhasználójának módosítása

int **mysql_change_user** (string user, string password [, string database [, resource link_identifier]]) \linebreak

A **mysql_change_user()** függvény megváltoztatja az aktuális, vagy az opcionális *link_identifier* paraméterben megadott azonosítójú kapcsolat felhasználóját. Ha egy adatbázis nevét megadod, akkor ez lesz az alapértelmezett, illetve az aktuális adatbázis, miután a mysql felhasználó megváltozott. Ha az új név-jelszó páros helytelen, akkor a függvényhívás előtti kapcsolat él tovább.

Megjegyzés: A függvény a PHP 3.0.13 változatával került a nyelvbe és MySQL 3.23.3, vagy későbbi változata szükséges a használatához.

mysql_character_set_name (PHP 4 CVS only)

Returns the name of the character set

int **mysql_character_set_name** ([resource link_identifier]) \linebreak

mysql_character_set_name() returns the default character set name for the current connection.

Példa 1. mysql_character_set_name() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$charset = mysql_character_set_name($link);
printf ("current character set is %s\n", $charset);
?>
```

The above example would produce the following output:

```
latin1
```

See also: mysql_real_escape_string()

mysql_close (PHP 3, PHP 4)

Lezár egy MySQL kapcsolatot

bool **mysql_close** ([resource link_identifier]) \linebreak

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

A **mysql_close()** függvény bezárja az adott azonosítójú MySQL kapcsolatot. Ha nem adsz meg *`link_identifier`* paramétert, akkor az utoljára megnyitott kapcsolatot zárja le.

A **mysql_close()** függvény használata általában szükségtelen, mert a nem perzisztens kapcsolatok a szkript végén bezáródnak. Lásd még: erőforrások felszabadítása

> **Megjegyzés:** A **mysql_close()** függvény nem zárja be a mysql_pconnect() függvénnyel megnyitott kapcsolatokat.

Példa 1. MySQL close példa

```
<?php
    $link = mysql_connect ("kraemer", "marliesle", "titok")
        or die ("Nem lehet csatlakozni");
    print ("A csatlakozás sikerült");
```

```
    mysql_close ($link);
?>
```

Lásd még a mysql_connect() és a mysql_pconnect() függvényeket!

mysql_connect (PHP 3, PHP 4)

Kapcsolatot nyit meg egy MySQL szerverhez

resource **mysql_connect** ([string server [, string username [, string password]]]) \linebreak

Pozitív MySQL azonosítóval tér vissza, ha a csatlakozás sikerült, FALSE-sal ha nem.

A **mysql_connect()** függvény kapcsolatot nyit meg egy MySQL szerverhez. A paramétereket elhagyhatod. Az alapértelmezett értékek: *server* = 'localhost:3306', *username* = a folyamat tulajdonosának belépési neve *password* = üres karakterlánc.

A *server* paraméter tartalmazhat egy portszámot is, például: "hostname:port" vagy tartalmazhatja a MySQL socket elérési útvonalát, például: ":/path/to/socket". Az utóbbi hostname paramétert használva is a helyi MySQL szerverhez próbál majd kapcsolódni a függvény.

> **Megjegyzés:** A ":port" formátumú kiegészítés a PHP 3.0B4-es változatában került a nyelvbe.
>
> A ":/path/to/socket" formát pedig a PHP 3.0.10-es verziójától kezdve használhatjuk.
>
> A sikertelen kapcsolatkor kiírt hibaüzenetet elnyomhatod, ha '@' jelet írsz a függvény elé.

Ha a **mysql_connect()** függvényt kétszer ugyanazokkal a paraméterekkel hívod meg, akkor nem jön létre újabb kapcsolat; a függvény a már meglévő kapcsolat azonosítóját fogja visszaadni.

A kapcsolat a PHP program végén bezárul, ha előbb nem zártuk volna le a mysql_close() függvénnyel.

Példa 1. MySQL connect példa

```
<?php
    $link = mysql_connect ("localhost", "felhasználónév", "titok")
        or die ("Nem lehet csatlakozni");
    print ("A csatlakozás sikerült");
    mysql_close ($link);
?>
```

Lásd még a mysql_pconnect() és a mysql_close() függvényeket.

mysql_create_db (PHP 3, PHP 4)

Létrehoz egy MySQL adatbázist

bool **mysql_create_db** (string database name [, resource link_identifier]) \linebreak

A **mysql_create_db()** függvény megkísérel létrehozni egy új adatbázist a megadott kapcsolatazonosítón keresztül.

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

Példa 1. MySQL 'create database' példa

```
<?php
    $link = mysql_pconnect ("kron", "jutta", "geheim")
        or die ("Nem lehet csatlakozni")
    if (mysql_create_db ("my_db")) {
        print ("Az adatbázist létrehoztam\n");
    } else {
        printf ("Hiba az adatbázis létrehozásakor: %s\n", mysql_error ());
    }
?>
```

Kompatíbilitási okokból a **mysql_createdb()** függvény is használható, de nem javasolt.

Lásd még a: mysql_drop_db() függvényt.

mysql_data_seek (PHP 3, PHP 4)

Belső eredménymutató mozgatása

bool **mysql_data_seek** (resource result_identifier, int row_number) \linebreak

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

A **mysql_data_seek()** függvény az adott azonosítójú ereményhalmazban ugrik a megadott pozícióra. A mysql_fetch_row() függvény következő hívásakor ezt a sorszámú sort kapod meg.

Ha az első sorra akarsz pozicionálni, akkor *`Row_number`* 0 legyen!

Példa 1. MySQL data seek példa

```
<?php
    $link = mysql_pconnect ("weblabor", "kgergely", "titok")
        or die ("Nem tudok csatlakozni");

    mysql_select_db ("samp_db")
        or die ("Nem tudok belépni az adatbázisba");

    $query = "SELECT last_name, first_name FROM friends";
    $eredmeny = mysql_query ($query)
```

```
        or die ("Rossz kérés");

    // fetch rows in reverse order

    for ($i = mysql_num_rows ($eredmeny) - 1; $i &gt;=0; $i--) {
        if (!mysql_data_seek ($eredmeny, $i)) {
            echo ("Nem tudok a $i. sorra ugorni\n");
            continue;
        }

        if(!($sor = mysql_fetch_object ($eredmeny)))
            continue;

            echo "$sor->vezeteknev $sor->keresztnev<br />\n");
    }

    mysql_free_result ($eredmeny);
?>
```

mysql_db_name (PHP 3>= 3.0.6, PHP 4)

Get result data

string **mysql_db_name** (resource result, int row [, mixed field]) \linebreak

A **mysql_db_name()** függvény első paramétere egy mysql_list_dbs() által visszaadott érték. A *row* paraméter az eredményhalmaz indexe.

Ha hiba történt, a függvény FALSE-sal tér vissza. Használd a mysql_errno() és a mysql_error() függvényeket a hiba eredetének megállapítására!

Példa 1. Mysql_db_name() példa

```
<?php
error_reporting(E_ALL);

mysql_connect(’dbhost’, ’név’, ’jelszó’);
$db_list = mysql_list_dbs();

$i = 0;
$cnt = mysql_num_rows($db_list);
while ($i < $cnt) {
    echo mysql_db_name($db_list, $i) . "\n";
    $i++;
}
?>
```

Kompatíbilitási okokból a **mysql_dbname()** függvény is használható, de használata nem javasolt.

mysql_db_query (PHP 3, PHP 4)

MySQL kérést küld az adatbázisnak

resource **mysql_db_query** (string database, string query [, resource link_identifier]) \linebreak

Visszatérési értéke egy pozitív MySQL eredmény-azonosító, ha a kérés sikeres volt; FALSE, ha a kérés hibás volt.

A **mysql_db_query()** függvény kiválaszt egy adatbázist, majd azon végrehajt egy kérést. Ha a kapcsolat azonosító nincs megadva, akkor a függvény keres egyet. Ha nem talál ilyet, akkor megpróbál létrehozni egyet oly módon, mintha meghívta volna a mysql_connect() függvényt paraméterek nélkül.

Lásd még a mysql_connect() és a mysql_query() függvényeket.

> **Megjegyzés:** A függvény használata a PHP 4.0.6-os változata óta nem javasolt, ezért NE használd ezt a függvényt. Inkább használd a mysql_select_db() és a mysql_query() függvényeket.

mysql_drop_db (PHP 3, PHP 4)

Töröl egy MySQL adatbázist

bool **mysql_drop_db** (string database_name [, resource link_identifier]) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

A **mysql_drop_db()** függvény megpróbálja az egész adatbázist kitörölni, melyet az adott kapcsolat-azonosító határoz meg.

Kompatíbilitási okokból **mysql_dropdb()** néven is hívható a függvény, de használata nem javasolt.

Lásd még a mysql_create_db() függvényt!

mysql_errno (PHP 3, PHP 4)

Az előző MySQL művelet hibakódját tartalmazza

int **mysql_errno** ([resource link_identifier]) \linebreak

Az utoljára végrehajtott MySQL hiba kódjával tér vissza, vagy ha az sikeres volt, akkor 0-t (nullát) ad vissza.

A MySQL hibái már nem jelennek meg figyelmeztetésként. [orig.: Errors coming back from the mySQL database backend no longer issue warnings.] Ha szeretnéd megkapni a hiba kódját, használd a **mysql_errno()** függvényt! Ez a függvény csak az utolsó MySQL művelet hibakódját adja vissza [(not including mysql_error() and **mysql_errno()**)], ha használni szeretnéd, ügyelj rá, hogy még azelőtt hívd meg a függvényt, mielőtt egy másik MySQL függvényt meghívsz.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Lásd még a mysql_error() függvényt

mysql_error (PHP 3, PHP 4)

Az előző MySQL művelet hibaszövegét adja

string **mysql_error** ([resource link_identifier]) \linebreak

Az utolsó MySQL művelet hibaszövegével tér vissza, vagy ″ (üres karakterlánccal) ha nem volt hiba.

A MySQL hibái már nem jelennek meg figyelmeztetésként. [orig.: Errors coming back from the mySQL database backend no longer issue warnings.] Ha szeretnéd megkapni a hiba kódját, használd a **mysql_error()** függvényt! Ez a függvény csak az utolsó MySQL művelet hibakódját adja vissza [(not including **mysql_error()** and mysql_errno())], ha használni szeretnéd, ügyelj rá, hogy még azelőtt hívd meg a függvényt, mielőtt egy másik MySQL függvényt meghívsz.

```
<?php
mysql_connect("mcl");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nemletezoadatbazis");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nemletezotabla");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Lásd még a mysql_errno() függvényt!

mysql_escape_string (PHP 4 >= 4.0.3)

Egy karakerláncban szereplő speciális karakterek értelmezését szünteti meg.

string **mysql_escape_string** (string unescaped_string) \linebreak

A függvény az *`unescaped_string`* karakterláncot átalakítja úgy, hogy az biztonságosan elhelyezhető legyen egy mysql_query() függvényben.

Megjegyzés: A **mysql_escape_string()** függvény nem nyúl a `%` és a _ jelekhez.

mysql_fetch_array (PHP 3, PHP 4)

Kérés egy sorát adja vissza (tetszőleges) tömb formájában.

array **mysql_fetch_array** (resource result [, int result_type]) \linebreak

Az eredmény következő sorával tér vissza tömb formájában, vagy `FALSE`-sal, ha már nincs több sor.

A **mysql_fetch_array()** függvény a mysql_fetch_row() függvény kiterjesztett változata. Ráadásul az eredményt nem csak számokkal indexelt tömbbe írja, hanem asszociatív tömbbe is, ahol a mező nevei a kulcsok.

Ha az eredmény több sorának ugyanaz a neve, akkor a később szereplő oszlop marad meg. Ha szeretnéd az összes mezőt elérni ilyenkor is, akkor számmal indexeld a tömböt, vagy az SQL parancsban kérj aliast az oszlopra. Az alias-olt [ez mi magyarul?] oszlopok tartalmát nem tudod az oszlopok eredeti nevével elérni (pl. itt a `'field'` használatával).

```
select tik.f1 as ize tak.f1 as bigyo from tik, tak
```

Jó tudni, hogy a **mysql_fetch_array()** függvény használata *nem jelentősen* lassabb a mysql_fetch_row() használatánál, de a kapott eredmény feldolgozása jóval kényelmesebb.

A **mysql_fetch_array()** függvény elhagyható *`result_type`* paramétere a következő lehet: MYSQL_ASSOC, MYSQL_NUM, vagy MYSQL_BOTH. Ez a lehetőség a PHP 3.0.7-es változatában került a nyelvbe. A paraméter alapértelmezett értéke a MYSQL_BOTH.

A MYSQL_BOTH használatával egy olyan tömböt kapsz, amelyben az elemek számmal és karakterlánccal is indexelve vannak. MYSQL_ASSOC értékkel használva csak asszociatív tömböt kapsz (mint a mysql_fetch_assoc() függvénnyel), MYSQL_NUM értékkel meghívva a fügvényt számozott indexű tömböt kapsz (mint a mysql_fetch_row() függvénnyel).

További részletekért olvasd még a mysql_fetch_row() és a mysql_fetch_assoc() függvényeket!

Példa 1. Mysql_fetch_array() példa

```
<?php
mysql_connect ($host, $felhasznalo, $jelszo);
$eredmeny = mysql_db_query ("adatbázis","select user_id, teljesnev from table");
while ($sor = mysql_fetch_array ($eredmeny)) {
    echo "felhasznalonev: ".$sor["felhasznalonev"]."<br>\n";
    echo "felhasznalonev: ".$sor[0]."<br>\n";
    echo "nev: ".$sor["nev"]."<br>\n";
```

```
    echo "nev: ".$sor[1]."<br>\n";
}
mysql_free_result ($eredmeny);
?>
```

mysql_fetch_assoc (PHP 4 >= 4.0.3)

Az eredmény egy sorát asszociatív tömbként adja vissza.

array **mysql_fetch_assoc** (resource result) \linebreak

Az eredmény következő sorával, mint asszociatív tömbbel tér vissza, vagy FALSE-sal, ha nincs több sor.

A **mysql_fetch_assoc()** függvény hívásával ugyanazt érjük el, mintha a mysql_fetch_array() függvényt hívtuk volna meg MYSQL_ASSOC második paraméterrel. Ez a függvény egy tisztán asszociatív tömböt ad vissza. Eredetileg a mysql_fetch_array() függvény így működött. Ha a számokkal indexelt elemekre is szükséged van, akkor használd a mysql_fetch_array() függvényt.

Ha az eredmény több oszlopának ugyanaz a neve, akkor a később szereplő oszlop marad meg. Ha szeretnéd az összes mezőt elérni, akkor vagy számokkal indexszel érd el az eredményt a mysql_fetch_array() függvény használatával, vagy az SQL parancsban kérj aliast az oszlopra. Lásd még a mysql_fetch_array() függvénynél szereplő példát az aliasok leírásáért.

Jó tudni, hogy a **mysql_fetch_assoc()** függvény használata NEM jelentősen lassabb a mysql_fetch_row() használatánál, de a kapott eredmény feldolgozása jóval kényelmesebb.

További részletekért lásd még a mysql_fetch_row() és a mysql_fetch_array() függvényeket.

Példa 1. mysql_fetch_assoc()

```
<?php
mysql_connect ($host, $felhasznalo, $jelszo);
$eredmeny = mysql_db_query ("adatbázis","select * from table");
while ($sor = mysql_fetch_assoc ($eredmeny)) {
    echo $sor["felhasznalonev"];
    echo $sor["nev"];
}
mysql_free_result ($eredmeny);
?>
```

mysql_fetch_field (PHP 3, PHP 4)

Eredményhalmaz egy oszlopáról ad információt objektum formájában

object **mysql_fetch_field** (resource result [, int field_offset]) \linebreak

Egy mezőinformációt tartalmazó objektummal tér vissza.

A **mysql_fetch_field()** függvényt arra használhatod, hogy az eredmény egy oszlopáról információt kaphass. Ha a mező sorszámát nem adod meg, akkor a következő olyan oszlopról kapsz információt, amelyre még nem hívtad meg a **mysql_fetch_field()** függvényt.

Az objektum tulajdonságai:

- name - az oszlop neve
- table - a tábla neve, amelyhez az oszlop tartozik
- max_length - az oszlop maximális hossza
- not_null - értéke 1, ha az oszlop nem veheti fel a `NULL` értéket
- primary_key - értéke 1, ha az oszlop elsődleges kulcs
- unique_key - értéke 1, ha az oszlop egyszerű kulcs
- multiple_key - értéke 1, ha az oszlop összetett kulcs része
- numeric - értéke 1, ha az oszlop szám
- blob - értéke 1, ha az oszlop BLOB típusú
- type - az oszlop típusa
- unsigned - értéke 1, ha az oszlop nem előjeles
- zerofill - értéke 1, ha az oszlop nullával feltöltött

Példa 1. mysql_fetch_field()

```
<?php
mysql_connect ($host, $felhasznalo, $jelszo)
    or die ("Nem lehet csatlakozni az adatbázishoz");
$eredmeny = mysql_db_query ("adatbázis", "select * from table")
    or die ("Hibás kérés");
# oszlopinformáció lekérése
$i = 0;
while ($i < mysql_num_fields ($eredmeny)) {
    echo "Információ a $i. oszlopról:<BR>\n";
    $meta = mysql_fetch_field ($eredmeny);
    if (!$meta) {
        echo "Nem tudok információt lekérni<BR>gt;\n";
    }
    echo "<PRE>
blob:         $meta->blob
max_length:   $meta->max_length
multiple_key: $meta->multiple_key
name:         $meta->name
not_null:     $meta->not_null
numeric:      $meta->numeric
primary_key:  $meta->primary_key
table:        $meta->table
type:         $meta->type
unique_key:   $meta->unique_key
unsigned:     $meta->unsigned
zerofill:     $meta->zerofill
```

```
</PRE>";
    $i++;
}
mysql_free_result ($eredmeny);
?>
```

Lásd még a mysql_field_seek() függvényt!

mysql_fetch_lengths (PHP 3, PHP 4)

Az eredmény oszlopainak hosszáról nyújt információt

array **mysql_fetch_lengths** (resource result) \linebreak

Egy tömbbel tér vissza. A tömb elemei az utolsó mysql_fetch_row() függvény által betöltött oszlopok hosszait tartalmazza. Hiba esetén FALSE értékkel tér vissza.

A **mysql_fetch_lengths()** függvény a mysql_fetch_row(), mysql_fetch_array() vagy a mysql_fetch_object() által visszaadott eredmény-oszlopok hosszát tárolja, illetve megmondja számmal indexelt tömb formájában.

Lásd még a mysql_fetch_row() függvényt!

mysql_fetch_object (PHP 3, PHP 4)

Az eredmény következő sorát objektum formájában adja vissza

object **mysql_fetch_object** (resource result [, int result_type]) \linebreak

Egy objektummal tér vissza, amelynek tulajdonságai a betöltött sor oszlopait tartalmazzák. Ha nincs több sor, a függvény FALSE-at ad eredményül.

A **mysql_fetch_object()** függvény hasonló a mysql_fetch_array() függvényhez, egy különbséggel: nem tömböt, hanem objektumot ad vissza a függvény. Ez azt jelenti, hogy a mezőket csak a nevük alapján tudod elérni, mivel a számok nem érvényes tulajdonságnevek.

Az elhagyható *result_type* parméter egy konstans, ami MYSQL_ASSOC, MYSQL_NUM vagy MYSQL_BOTH lehet. Lásd még a mysql_fetch_array() függvényt, ha kíváncsi vagy az állandók leírására.

Sebesség szempontjából a mysql_fetch_array() függvénnyel azonos sebességű és majdnem olyan gyors, mint a mysql_fetch_row() függvény (a különbség elhanyagolható)

Példa 1. mysql_fetch_object() példa

```
<?php
mysql_connect ($host, $felhasznalo, $jelszo);
$eredmeny = mysql_db_query ("adatbázis", "select * from table");
while ($sor = mysql_fetch_object ($eredmeny)) {
    echo $sor-&gt;user_id;
```

```
    echo $sor-&gt;fullname;
}
mysql_free_result ($eredmeny);
?>
```

Lásd még a mysql_fetch_array() és a mysql_fetch_row() függvényt!

mysql_fetch_row (PHP 3, PHP 4)

Az eredmény következő sorát adja vissza számmal indexelt tömb formájában

array **mysql_fetch_row** (resource result) \linebreak

Egy számmal indexelt tömbbel tér vissza, ami a betöltött sorokat tartalmazza, vagy hamissal, ha nincs több sor.

A **mysql_fetch_row()** függvény betölt egy sort a megadott eredmény-azonosítójú eredményhalmazból. A sort (számmal indexelt) tömb formájában adja vissza. Az oszlopok számozása 0-tól indul.

A **mysql_fetch_row()** többszöri alkamazásával végig tudjuk járni az eredményhalmazt, mivel az mindig a következő sort tölti be, majd ha elfogytak a sorok, FALSE-at ad vissza.

Lásd még a mysql_fetch_array(), mysql_fetch_object(), mysql_data_seek(), mysql_fetch_lengths() és a mysql_result() függvényeket!

mysql_field_flags (PHP 3, PHP 4)

Adott eredmény adott mezejének flagjeit adja vissza.

string **mysql_field_flags** (resource result, int field_offset) \linebreak

A **mysql_field_flags()** függvény az adott mező flagjeit adja vissza. A flagek egyszerű szavak, melyeket egyetlen szóköz választ el, így a kapott értéket feldolgozhatod a explode() függvény segítségével.

A MySQL a következő flageket támogatja (feltéve, hogy elég friss a használt MySQL, amit használsz): "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

Kompatíbilitási okokból a **mysql_fieldflags()** függvény is használható, de nem javasolt.

mysql_field_len (PHP 3, PHP 4)

A kérdezett mező hosszát adja vissza

int **mysql_field_len** (resource result, int field_offset) \linebreak

A **mysql_field_len()** függvény a kérdezett mező hosszával tér vissza.

Kompatíbilitási okokból a **mysql_fieldlen()** függvény is használható, de nem ajánlott.

mysql_field_name (PHP 3, PHP 4)

Adott mező nevét adja vissza

string **mysql_field_name** (resource result, int field_index) \linebreak

A **mysql_field_name()** függvény az eredményben szereplő megadott indexű mező nevét adja vissza. A *`result`* paraméternek érvényes eredményazonosítónak kell lennie, a *`field_index`* paraméter pedig a mező indexe.

Megjegyzés: A *`field_index`* paraméter számozása 0-tól kezdődik.

Például a harmadik mező indexe 2, a negyedik mezőé pedig 3.

Példa 1. mysql_field_name() példa

```
// A felhasznalok tábla három mezőből áll:
//   azonosito
//   felhasznalonev
//   jelszo.

$eredmeny = mysql_db_query("felhasznalok", "select * from felhasznalok", $link);

echo mysql_field_name($eredmeny, 0) . "\n";
echo mysql_field_name($eredmeny, 2);
```

A fenti példa ay alábbi kimenetet eredményezi:

```
azonosito
jelszo
```

Kompatíbilitási okokból a **mysql_fieldname()** függvény is használható, de nem ajánlott.

mysql_field_seek (PHP 3, PHP 4)

Az eredménymutatót a kívánt pozícióba mozgatja

int **mysql_field_seek** (resource result, int field_offset) \linebreak

A megadott helyre mozgatja az eredmény-mutatót. Ha a következő mysql_fetch_field() függvényhívás nem tartalmaz mező pozíciót, akkor a **mysql_field_seek()** függvény által visszaadott mezőpozíciót fogja visszaadni.

Lásd még a mysql_fetch_field() függvényt!

mysql_field_table (PHP 3, PHP 4)

Visszaadja annak a táblának a nevét, melyben a megadott mező található

string **mysql_field_table** (resource result, int field_offset) \linebreak

A függvény annak a táblának a nevével tér vissza, melyben a megadott mező szerepel.

Kompatíbilitási okokból a **mysql_fieldtable()** függvény is használható, de nem ajánlott.

mysql_field_type (PHP 3, PHP 4)

Az eredmény adott mezejének típusát adja vissza

string **mysql_field_type** (resource result, int field_offset) \linebreak

A **mysql_field_type()** függvény hasonló a mysql_field_name() függvényhez. A paraméterek megegyeznek, de a függvény a mező típusával tér vissza. A mezőtípus az "int", "real", "string", "blob" vagy hasonló, a MySQL dokumentációban (http://www.mysql.com/documentation/) található típusok közül való.

Példa 1. mysql mezőtípusok

```
<?php

mysql_connect ("localhost:3306");
mysql_select_db ("wisconsin");
$eredmeny = mysql_query ("SELECT * FROM onek");
$mezo = mysql_num_fields ($eredmeny);
$sor   = mysql_num_rows ($eredmeny);
$i = 0;
$table = mysql_field_table ($eredmeny, $i);
echo "A ’$table’ nevű táblának $mezo mezője és $sor sora van <BR>";
echo "A táblának az alábbi mezői vannak<BR>";
while ($i < $mezo) {
    $tipus = mysql_field_type  ($eredmeny, $i);
    $nev   = mysql_field_name  ($eredmeny, $i);
    $hossz = mysql_field_hossz ($eredmeny, $i);
    $flags = mysql_field_flags ($eredmeny, $i);
    echo "$tipus $nev $hossz $flags."<BR>";
    $i++;
}
mysql_close();

?>
```

Kompatíbilitási okokból a **mysql_fieldtype()** függvény is használható, de nem ajánlott.

mysql_free_result (PHP 3, PHP 4)

Felszabadítja az eredmény által lefoglalt memóriát

bool **mysql_free_result** (resource result) \linebreak

A **mysql_free_result()** függvény az összes *`result`* eredményazonosító által használt memóriát felszabadítja.

A **mysql_free_result()** függvényt csak akkor kell meghívnod, ha nagyon aggódsz a használt memória mennyisége miatt. A memória a php program végén úgyis felszabadul.

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

Kompatíbilitási okokból a **mysql_freeresult()** függvény is használható, de nem ajánlott.

mysql_get_client_info (PHP 4 >= 4.0.5)

MySQL kliens információ

string **mysql_get_client_info** (void) \linebreak

A **mysql_get_client_info()** függvény a kliensprogram által használt könyvtár verziójáról ad információt.

A **mysql_get_client_info()** függvény a PHP 4.0.5-ös verziójában került a nyelvbe.

mysql_get_host_info (PHP 4 >= 4.0.5)

MySQL kiszolgáló információ

string **mysql_get_host_info** ([resource link_identifier]) \linebreak

A **mysql_get_host_info()** függvény egy karakterlánccal tér vissza, ami a *`link_identifier`* által használt kapcsolatról ad információt, például a kiszolgáló gép nevét. Ha a *`link_identifier`* paramétert elhagyjuk, akkor a legutóbb megnyitott kapcsolatról kapunk információt.

A **mysql_get_host_info()** függvény a PHP 4.0.5-ös változatával került a nyelvbe.

mysql_get_proto_info (PHP 4 >= 4.0.5)

MySQL protokoll információ

int **mysql_get_proto_info** ([resource link_identifier]) \linebreak

A **mysql_get_proto_info()** függvény a *`link_identifier`* által használt protokoll verzióját adja vissza. Ha a *`link_identifier`* paramétert nem adod meg, akkor az utoljára használt kapcsolatról kapsz információt.

mysql_get_proto_info() függvény a PHP 4.0.5-ös változatával került a nyelvbe.

mysql_get_server_info (PHP 4 >= 4.0.5)

MySQL szerver információ

int **mysql_get_server_info** ([resource link_identifier]) \linebreak

A **mysql_get_server_info()** függvény a *`link_identifier`*-ban megadott kapcsolat által használt szerver verzióját adja. Ha nincs megadva *`link_identifier`*, akkor az utoljára használt kapcsolatról kapsz információt.

A **mysql_get_server_info()** függvény a PHP 4.0.5-ös változatával került a nyelvbe.

mysql_info (PHP 4 CVS only)

Get information about the most recent query

string **mysql_info** ([resource link_identifier]) \linebreak

mysql_info() returns detailed information about the last query using the given *`link_identifier`*. If *`link_identifier`* isn't specified, the last opened link is assumed.

mysql_info() returns a string for all statements listed below. For all other `FALSE`. The string format depends on the given statement.

Példa 1. Relevant MySQL Statements

```
INSERT INTO ... SELECT ...
String format: Records: 23 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),(...)...
String format: Records: 37 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...
String format: Records: 42 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE
String format: Records: 60 Duplicates: 0 Warnings: 0
UPDATE
String format: Rows matched: 65 Changed: 65 Warnings: 0
```

The numbers are only for illustrating purpose; their values will correspond to the query.

Megjegyzés: mysql_info() returns a non-`FALSE` value for the INSERT ... VALUES statement only if multiple value lists are specified in the statement.

See also: mysql_affected_rows()

mysql_insert_id (PHP 3, PHP 4)

Visszaadja az előző INSERT művelet által előállított id-t

int **mysql_insert_id** ([resource link_identifier]) \linebreak

A **mysql_insert_id()** függvény az előző INSERT műveletben szereplő AUTO_INCREMENT tulajdonságú mező értékét adja vissza. Ha megadjuk *`link_identifier`*-t az adott linkkel rendelekező, ha nem adunk meg paramétert, az utolsó megnyitott linkhez tartozó eredményt szolgáltatja.

A **mysql_insert_id()** függvény 0-val tér vissza, ha az előző kérésben nem jött létre AUTO_INCREMENT érték. Ha szükséged van később az értékre, hívd meg a **mysql_insert_id()** függvényt rögtön az értéket generáló lekérdezés után!

Megjegyzés: A `LAST_INSERT_ID()` függvény értéke mindig az utoljára előállított AUTO_INCREMENT értéket tartalmazza, nem törlődik az értéke lekérdezések között.

Figyelem

A **mysql_insert_id()** függvény a MySQL C API függvény `mysql_insert_id()` visszatérési értékét `long`-gá (PHP-ban ez az int) konvertálja. Ha az AUTO_INCREMENT meződ mondjuk BIGINT típusú, akkor a **mysql_insert_id()** függvény rossz eredményt adhat vissza. Ezért inkább az SQL-be beépített `LAST_INSERT_ID()` MySQL függvényt használd!

mysql_list_dbs (PHP 3, PHP 4)

Kilistázza a MySQL szerveren elérhető adatbázisokat

resource **mysql_list_dbs** ([resource link_identifier]) \linebreak

A **mysql_list_dbs()** függvény egy eredménymutatót ad vissza, ami a mysql démontól elérhető adatbázisokat tartalmazza. Használd a mysql_tablename() függvényt az eredménymutatón történő végigjárásra, vagy bármely függvényt az eredménytáblákra!

Példa 1. mysql_list_dbs() példa

```
<?php
$link = mysql_connect('localhost', 'nevem', 'titok');
$db_list = mysql_list_dbs($link);

while ($sor = mysql_fetch_object($db_list)) {
```

```
  echo $sor->Database . "\n";
}
?>
```

A fenti példa az alábbi kimenetet eredményezi.

```
adatbázis1
adatbázis2
adatbázis3
...
```

> **Megjegyzés:** A fenti program egyszerűen bővíthető a mysql_fetch_row() függvénnyel vagy más hasonló függvénnyel.

Kompatíbilitási okokból a **mysql_listdbs()** függvény is használható, de nem javasolt.

Lásd még a mysql_db_name() függvényt!

mysql_list_fields (PHP 3, PHP 4)

Kilistázza a MySQL eredmény mezőit

resource **mysql_list_fields** (string database_name, string table_name [, resource link_identifier]) \linebreak

A **mysql_list_fields()** függvény az adott tábláról szolgáltat információt. A paraméterek az adatbázis és a tábla neve. A függvény egy eredmény-azonosító tér vissza, amit a mysql_field_flags(), mysql_field_len(), mysql_field_name(), és a mysql_field_type() függvényekkel hámozhatsz meg.

Példa 1. mysql_list_fields() példa

```
<?php
$link = mysql_connect(’localhost’, ’nevem’, ’titok’);

$mezo = mysql_list_fields("adatbázis1", "tabla1", $link);
$oszlop = mysql_num_fields($mezo);

for ($i = 0; $i < $oszlop; $i++) {
  echo mysql_field_name($mezo, $i) . "\n";;
}
```

A fenti példa az alábbi kimenetet eredményezi:

```
mezo1
```

```
mezo2
mezo3
...
```

Kompatíbilitási okokból a **mysql_listfields()** függvény is használható, de nem ajánlott.

mysql_list_processes (PHP 4 CVS only)

List MySQL processes

resource **mysql_list_processes** ([resource link_identifier]) \linebreak

mysql_list_processes() returns a result pointer describing the current server threads.

Példa 1. mysql_list_processes() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');

$result = mysql_list_processes($link);
while ($row = mysql_fetch_row($result)){
    printf("%s %s %s %s %s\n", $row["Id"], $row["Host"], $row["db"],
       $row["Command"], $row["Time"]);
}
mysql_free_result ($result);
?>
```

The above example would produce the following output:

```
1 localhost test Processlist 0
4 localhost mysql sleep 5
```

See also: mysql_thread_id()

mysql_list_tables (PHP 3, PHP 4)

Adott adatbázisbeli táblaneveket listázza

resource **mysql_list_tables** (string database [, resource link_identifier]) \linebreak

A **mysql_list_tables()** függvény veszi az adatbázisnevet, és egy eredmény-mutatót ad vissza, a mysql_db_query() függvényhez hasonlóan. A mysql_tablename() függvényt használhatod arra, hogy az eredmény-mutatóból, vagy más tábla eredményéből a tényleges táblaneveket megállapítsd.

Kompatíbilitási okokból a **mysql_listtables()** függvény is használható, de nem ajánlott.

mysql_num_fields (PHP 3, PHP 4)

Az eredményben szereplő mezők számát adja

int **mysql_num_fields** (resource result) \linebreak

A **mysql_num_fields()** függvény az eredményhalmazban levő mezők számát adja vissza.

Lásd még a mysql_db_query(), mysql_query(), mysql_fetch_field() és a mysql_num_rows() függvényeket!

Kompatíbilitási okokból a **mysql_numfields()** függvény is használható, de nem ajánlott.

mysql_num_rows (PHP 3, PHP 4)

Az eredményben szereplő sorok számát adja vissza.

int **mysql_num_rows** (resource result) \linebreak

A **mysql_num_rows()** függvény az eredményhalmazban szereplő sorok számát adja vissza. A függvény csak SELECT kérésre használható. Ha egy INSERT, UPDATE vagy DELETE kérésben érintett sorok számát szeretnéd megtudni, használd a DELETE, use mysql_affected_rows() függvényt!

Példa 1. mysql_num_rows() példa

```
<?php
$conn = mysql_connect("mlabdial", "juzer", "jelszo");
mysql_select_db("adatbázis",$conn);

$eredmeny = mysql_query("SELECT * FROM tabla1", $conn);
$sorok_szama = mysql_num_rows($eredmeny);

echo "$sorok_szama sor van";
?>
```

Lásd még a mysql_affected_rows(), mysql_connect(), mysql_select_db() és a mysql_query() függvényeket!

Kompatíbilitási okokból a **mysql_numrows()** függvény is használható, de nem javasolt.

mysql_pconnect (PHP 3, PHP 4)

Perzisztens kapcsolatot nyit meg a MySQL szerverhez

resource **mysql_pconnect** ([string server [, string username [, string password]]]) \linebreak

Visszatérési értéke: Egy pozitív MySQL perzisztens kapcsolatazonosító siker esetén, vagy FALSE, ha hiba történt.

A **mysql_pconnect()** függvény létrehoz egy kapcsolatot a MySQL szerverhez. Az elhagyott paraméterek az alábbi értékeket veszik fel: *server* = 'localhost:3306', *username* = annak a felhasználónak a neve, akié a szerver folyamat és végül: *password* = üres jelszó.

A *server* karakterlánc a port számát is tartalmazhatja; pl.: "hostname:port" vagy a socket teljes elérési útvonalát pl.: ":/path/to/socket". Ez persze csak a helyi gépen működik.

Megjegyzés: A ":port" használata a 3.0B4-es verzió óta lehetséges.

A ":/path/to/socket" forma használata a 3.0.10-es verzió óta támogatott.

A **mysql_pconnect()** függvény a mysql_connect() függvényhez hasonlóan működik, csupán két nagyobb különbség van:

Először is, a kapcsolat felvételekor a függvény először megpróbál egy (perzisztens) kapcsolatot találni, amely már nyitva van arra a hostra ugyanazzal a név/jelszó párossal. Ha talál ilyet, akkor ennek a kapocsnak az azonosítóját adja vissza és nem nyit újat.

Másodszor pedig a kapcsolat az SQL szerverhez nem kerül bezárásra, amikor a php program véget ér, hanem megmarad későbbi felhasználásra. (A mysql_close() függvény nem fogja neked lezárni a **mysql_pconnect()** függvénnyel megnyitott kapcsolatokat).

Az ilyen típusú kapcsolatot ezért hívják 'perzisztensnek'.

Megjegyzés: Ügyelj rá, hogy ezek a fajta kapcsolatok csak a PHP betölthető modulos változatában működnek. Lást a Perzisztens Adatbázis Kapcsolatok fejezetet további információkért!

mysql_ping (PHP 4 CVS only)

Ping a server connection or reconnect if there is no connection

bool **mysql_ping** ([resource link_identifier]) \linebreak

mysql_ping() checks whether or not the connection to the server is working. If it has gone down, an automatic reconnection is attempted. This function can be used by scripts that remain idle for a long while, to check whether or not the server has closed the connection and reconnect if necessary. **mysql_ping()** returns TRUE if the connection to the server is working, otherwise FALSE.

See also: mysql_thread_id(), mysql_list_processes().

mysql_query (PHP 3, PHP 4)

MySQL kérést küld a szervernek

resource **mysql_query** (string query [, resource link_identifier]) \linebreak

A **mysql_query()** függvény kérést küld a megadott kapcsolat-azonosítójú szerver aktív adatbázisához. Ha nem adsz meg *`link_identifier`*-t, akkor a legutóbb megnyitott kapcsolatot használja a függvény. Ha nincs nyitva ilyen kapcsolat, akkor a függvény megpróbál nyitni egyet, mintha a mysql_connect() függvényt hívtuk volna paraméterek nélkül.

Megjegyzés: A kérésnek nem szabad pontosvesszővel végződnie.

A SELECT utasításra alkalmazott **mysql_query()** függvény eredményazonosítóval vagy `FALSE`-sal tér vissza a kérés végrehajtásától függően. Egyéb esetekben **mysql_query()** függvény TRUE-val (nemnulla) vagy FALSE-szal tér vissza, attól függően, hogy a kérés teljesítése sikeres volt-e. A TRUE visszatérési érték azt jelenti, hogy a kérés szintaktikailag helyes volt, és lefuttatta a szerver. Az érintett sorok számáról azonban nem mond semmit. Előfordulhat ugyanis, hogy a kérés sikeresen lefutott, de nem érintett egyetlen sort sem, vagy az eredményben egyetlen sor sincs.

A következő kérés szintaktikailag rossz, igy a **mysql_query()** függvény meghiúsul és `FALSE` eredményt ad:

Példa 1. mysql_query()

```
<?php
$eredmeny = mysql_query ("SELECT * WHERE 1=1")
  or die ("Érvénytelen lekérdezés");
?>
```

Az alábbi kérés szemantikailag helytelen, ha nincs `my_col` nevű oszlop a `my_tbl` nevű táblában, így a **mysql_query()** meghiúsul és `FALSE` értékkel tér vissza.

Példa 2. mysql_query()

```
<?php
$eredmeny = mysql_query ("SELECT my_col FROM my_tbl")
    or die ("Érvénytelen kérés");
?>
```

A **mysql_query()** függvény akkor is meghiúsul és `FALSE` értékkel tér vissza, ha nincs megfelelő engedélyed a kérés által hivatkozott táblá(k)hoz.

Feltéve, hogy a kérés sikeres, meghívhatod a mysql_num_rows() függvényt, hogy kiderítsd, hány sort kaptál vissza a SELECT kérésedre, vagy a mysql_affected_rows() függvényt, hogy megtudd hány sort érintett a DELETE, NSERT, REPLACE vagy UPDATE kérésed.

SELECT kérés esetében a **mysql_query()** függvény egy új eredmény-azonosítót ad vissza, amit később a mysql_fetch_array() vagy más eredménytáblákkal foglalkozó függvénnyel használhatsz. Ha az eredményre már nincs szükség és memóriafóbiád van, akkor a mysql_free_result() függvénnyel a foglalt memóriát felszabadíthatod, normál esetben megvárod a program végét és a memória felszabadul.

Lásd még a mysql_num_rows(), mysql_affected_rows(), mysql_db_query(), mysql_unbuffered_query(), mysql_free_result(), mysql_fetch_array(), mysql_fetch_row(), mysql_fetch_assoc(), mysql_result(), mysql_select_db() és a mysql_connect() függvényeket!

mysql_real_escape_string (PHP 4 CVS only)

Escapes special characters in a string for use in a SQL statement, taking into account the current charset of the connection.

string **mysql_real_escape_string** (string unescaped_string [, resource link_identifier]) \linebreak

This function will escape special characters in the *`unescaped_string`*, taking into account the current charset of the connection so that it is safe to place it in a mysql_query().

> **Megjegyzés: mysql_real_escape_string()** does not escape `%` and `_`.

Példa 1. mysql_real_escape_string() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$item = "Zak's and Derick's Laptop";
$escaped_item = mysql_real_escape_string($item);
printf ("Escaped string: %s\n", $escaped_item);
?>
```

The above example would produce the following output:

```
Escaped string: Zak\'s and Derick\'s Laptop
```

See also: mysql_escape_string(), mysql_character_set_name().

mysql_result (PHP 3, PHP 4)

Eredmény egy sorának lekérdezése

mixed **mysql_result** (resource result, int row [, mixed field]) \linebreak

A **mysql_result()** függvény egy MySQL eredményhalmaz egy elemét adja vissza. A mező paraméter a mező sorszáma vagy neve lehet vagy táblanév pont mezőnév (táblanév.mezőnév). Ha az oszlop más néven lett lekérve, mint ami az oszlop eredeti neve volt (’select izé as bigyó from...’), akkor használd azt a bizonyos ’más nevet’!

Amikor nagy eredményhalmazokkal dolgozol, akkor mérlegelned kell az olyan függvyének használatát, amelyek az egész eredménysort betöltik. Mivel ezek több cella tartalmát olvassák be egy függvényhívással, így ezek SOKKAL gyorsabbak, mint a **mysql_result()** függvény hívogatása. Említést érdemel még, hogy a numerikus index használata jóval gyorsabb, mint a mezőnév vagy a táblanév.mezőnév forma használata.

A **mysql_result()** függvényhívásokat más, ugyanazon eredményhalmazzal foglalkozó függvényhívásokkal nem illik (gyk. TILOS) együtt használni!

Ajánlott az alábbi, gyors [és programozóbarát] függvények használata: mysql_fetch_row(), mysql_fetch_array() és mysql_fetch_object().

mysql_select_db (PHP 3, PHP 4)

Kiválaszt egy MySQL adatbázist

bool **mysql_select_db** (string database_name [, resource link_identifier]) \linebreak

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

A **mysql_select_db()** függvény az adott kapcsolat-azonosítójú szerverkapcsolat adatbázisát módosítja. Ha nincs kapcsolat-azonosító megadva, akkor az utoljára megnyitott kapcsolatban választ adatbázist. Ha ilyen sincs, akkor megpróbál a MySQL szerverhez kapcsolódni úgy, mintha a mysql_connect() függvény lett volna meghívva paraméterek nélkül.

Az összes további mysql_query() függvény az aktív adatbázison fog történni.

Lásd még a mysql_connect(), mysql_pconnect() és a mysql_query() függvényeket!

Kompatíbilitási okokból a **mysql_selectdb()** függvény is használható, de használata nem ajánlott.

mysql_stat (PHP 4 CVS only)

Get current system status

string **mysql_stat** ([resource link_identifier]) \linebreak

mysql_stat() returns the current server status.

> **Megjegyzés: mysql_stat()** currently only returns status for uptime, threads, queries, open tables, flush tables and queries per second. For a complete list of other status variables you have to use the SHOW STATUS SQL command.

Példa 1. mysql_stat() example

```
<?php
$link = mysql_connect('localhost', "mysql_user", "mysql_password");
printf("%s\n", mysql_stat($link));
?>
```

The above example would produce the following output:

```
Uptime: 5380  Threads: 1  Questions: 1321299  Slow queries: 1  Opens: 26 Flush ta-
bles: 1  Open tables: 17  Queries per second avg: 245.595
```

mysql_tablename (PHP 3, PHP 4)

Egy tábla nevével tér vissza

string **mysql_tablename** (resource result, int i) \linebreak

A **mysql_tablename()** függvény veszi a mysql_list_tables() függvény által visszaadott eredmény-mutatót és egy indexet, majd megmondja az oszlophoz tartozó tábla nevét. A mysql_num_rows() függvény használható arra, hogy megmondja hány tábla van az eredményhalmazban.

Példa 1. Mysql_tablename() példa

```
<?php
mysql_connect ("localhost:3306");
$eredmeny = mysql_list_tables ("wisconsin");
$i = 0;
while ($i < mysql_num_rows ($eredmeny)) {
    $tb_nevek[$i] = mysql_tablename ($eredmeny, $i);
    echo $tb_nevek[$i] . "<BR>";
    $i++;
}
?>
```

mysql_thread_id (PHP 4 CVS only)

Return the current thread ID

int **mysql_thread_id** ([resource link_identifier]) \linebreak

mysql_thread_id() returns the current thread ID. If the connection is lost and you reconnect with mysql_ping(), the thread ID will change. This means you should not get the thread ID and store it for later. You should get it when you need it.

Példa 1. mysql_thread_id() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$thread_id = mysql_thread_id($link);
if ($thread_id){
    printf ("current thread id is %d\n", $thread_id);
}
?>
```

The above example would produce the following output:

```
current thread id is 73
```

See also: mysql_ping(), mysql_list_processes().

mysql_unbuffered_query (PHP 4 >= 4.0.6)

SQL kérést küld küld a MySQL-nek anélkül, hogy az eredményt előfeldolgozná.

resource **mysql_unbuffered_query** (string query [, resource link_identifier]) \linebreak

A **mysql_unbuffered_query()** függvény a *`query`* SQL kérést küldi a MySQL-nek anélkül, hogy bármi módon betöltené, vagy pufferelni, mint azt a mysql_query() függvény teszi. Egyfelől, ez elég jelentős memória-megtakarítást tesz lehetővé az olyan SQL kérések számára, amelyek nagy eredményhalmazt állítanak elő. Másfelől, az eredményhalmazzal rögtön dolgozni kezdhetsz, amint az első sor megérkezik: nem kell megvárnod a teljes SQL kérés lefutását. Ha több adatbáziskapcsolatot használsz, meg kell mondanod a *`link_identifier`* paraméterben, hogy melyik kapcsolatra vonatkozik a kérés.

> **Megjegyzés:** A **mysql_unbuffered_query()** függvénynek az a hátránya, hogy nem használhatod a mysql_num_rows() függvényt a visszakapott eredményhalmazra. A bufferelés nélküli SQL kérés minden sorát be kell töltened, mielőtt egy új SQL kérést küldesz a MySQL-nek.

Lásd még a mysql_query()függvényt!

LXIV. Mohawk Software session handler functions

msession is an interface to a high speed session daemon which can run either locally or remotely. It is designed to provide consistent session management for a PHP web farm.

The session server software can be found at http://www.mohawksoft.com/phoenix/.

msession_connect (PHP 4 >= 4.2.0)

Connect to msession server

bool **msession_connect** (string host, string port) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_count (PHP 4 >= 4.2.0)

Get session count

int **msession_count** (void) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_create (PHP 4 >= 4.2.0)

Create a session

bool **msession_create** (string session) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_destroy (PHP 4 >= 4.2.0)

Destroy a session

bool **msession_destroy** (string name) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_disconnect (PHP 4 >= 4.2.0)

Close connection to msession server

void **msession_disconnect** (void) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_find (PHP 4 >= 4.2.0)

Find value

array **msession_find** (string name, string value) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_get_array (PHP 4 >= 4.2.0)

Get array of ... ?

array **msession_get_array** (string session) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_get (PHP 4 >= 4.2.0)

Get value from session

string **msession_get** (string session, string name, string value) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_getdata (unknown)

Get data ... ?

string **msession_getdata** (string session) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_inc (PHP 4 >= 4.2.0)

Increment value in session

string **msession_inc** (string session, string name) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_list (PHP 4 >= 4.2.0)

List ... ?

array **msession_list** (void) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_listvar (PHP 4 >= 4.2.0)

List sessions with variable

array **msession_listvar** (string name) \linebreak

Returns an associative array of value, session for all sessions with a variable named *name*.

Used for searching sessions with common attributes.

msession_lock (PHP 4 >= 4.2.0)

Lock a session

int **msession_lock** (string name) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_plugin (PHP 4 >= 4.2.0)

Call an escape function within the msession personality plugin

string **msession_plugin** (string session, string val [, string param]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_randstr (PHP 4 >= 4.2.0)

Get random string

string **msession_randstr** (int param) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_set_array (PHP 4 >= 4.2.0)

Set array of ...

bool **msession_set_array** (string session, array tuples) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_set (PHP 4 >= 4.2.0)

Set value in session

bool **msession_set** (string session, string name, string value) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_setdata (unknown)

Set data ... ?

bool **msession_setdata** (string session, string value) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_timeout (PHP 4 >= 4.2.0)

Set/get session timeout

int **msession_timeout** (string session [, int param]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_uniq (PHP 4 >= 4.2.0)

Get uniq id

string **msession_uniq** (int param) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msession_unlock (PHP 4 >= 4.2.0)

Unlock a session

int **msession_unlock** (string session, int key) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

LXV. muscat functions

muscat_close (4.0.5 - 4.2.1 only)

Shuts down the muscat session and releases any memory back to PHP.

int **muscat_close** (resource muscat_handle) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

[Not back to the system, note!]

muscat_get (4.0.5 - 4.2.1 only)

Gets a line back from the core muscat API.

string **muscat_get** (resource muscat_handle) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

Returns a literal FALSE when there is no more to get (as opposed to ""). Use === FALSE or !== FALSE to check for this.

muscat_give (4.0.5 - 4.2.1 only)

Sends string to the core muscat API

int **muscat_give** (resource muscat_handle, string string) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

muscat_setup_net (4.0.5 - 4.2.1 only)

Creates a new muscat session and returns the handle.

resource **muscat_setup_net** (string muscat_host, int port) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

muscat_host is the hostname to connect to port is the port number to connect to - actually takes exactly the same args as fsockopen

muscat_setup (4.0.5 - 4.2.1 only)

Creates a new muscat session and returns the handle.

resource **muscat_setup** (int size [, string muscat_dir]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

Size is the ammount of memory in bytes to allocate for muscat muscat_dir is the muscat installation dir e.g. "/usr/local/empower", it defaults to the compile time muscat directory

LXVI. Hálózatkezelési Függvények

checkdnsrr (PHP 3, PHP 4)

Adott Internet domainnév vagy IP címnek megfelelő bejegyzés létének vizsgálata a DNS rekordokban

int **checkdnsrr** (string host [, string típus]) \linebreak

Keresést kér a névszervertől, a *típus* paraméterben megadott típusú rekordokban, a *host* paraméternek megfelelő bejegyzésekre. A visszatérési érték TRUE lesz, amennyiben volt találat. Ha nem volt találat, avagy hiba merült fel a művelet során, a visszatérési érték FALSE lesz.

A *típus* paraméter a következők közül választható: A, MX, NS, SOA, PTR, CNAME, vagy ANY. Az alapértelmezés az MX.

A keresett *host* lehet IP cím és domainnév is.

Megjegyzés: Ez a függvény nem működik Windows operációs rendszereken!

Lásd még: getmxrr(), gethostbyaddr(), gethostbyname(), gethostbynamel(), valamint a man named(8) oldalát!

closelog (PHP 3, PHP 4)

Syslog (rendszer napló) kapcsolat zárása

int **closelog** (void) \linebreak

A **closelog()** függvény zárja a rendszer naplózóval való kapcsolatot. A függvény használata opcionális.

Lásd még: define_syslog_variables(), syslog() és openlog()!

debugger_off (PHP 3)

A beépített PHP debugger letiltása

int **debugger_off** (void) \linebreak

Letiltja beépített PHP debuggert (hibakeresőt). A debugger jelen pillanatban is fejlesztés alatt áll.

debugger_on (PHP 3)

A beépített PHP debugger engedélyezése

int **debugger_on** (string address) \linebreak

Engedélyezi a beépített PHP debugger működését, az a *address* paraméterben megadott címhez kapcsolódik. A debugger jelen pillanatban is fejlesztés alatt áll.

define_syslog_variables (PHP 3, PHP 4)

A rendszer naplózóval kapcsolatos konstansok inicializálása

void **define_syslog_variables** (void) \linebreak

Minden állandót, amelyeknek csak köze van a syslog függvényekhez, újrainicializál.

Lásd még: openlog(), syslog() és closelog()!

fsockopen (PHP 3, PHP 4)

Internet vagy Unix domain socket megnyitása

int **fsockopen** (string hostname, int port [, int errno [, string errstr [, double timeout]]]) \linebreak

Stream típusú kapcsolatot kezdeményez egy Internet (AF_INET, TCP vagy UDP protokollal) vagy Unix (AF_UNIX) domain felé. Internet domain megadása esetén TCP kapcsolatot nyit a *`hostname`* paraméterben megadott géppel, a *`port`* paraméterben megadott porton. Ez esetben a *`hostnév`* lehet domainnév vagy IP cím is. UDP kapcsolat esetén explicit meg kell adni a protokollt a *`hostname`* paraméterben, a következő módon: `udp://hostname`. Unix domain esetén a *`hostname`* paraméternek kell tartalmaznia a socket elérési útját, ekkor a *`port`* értékének nullának kell lennie. A *`timeout`* paraméter elhagyható, értéke másodpercekben megadandó.

A PHP 4.3.0 verziótól kezdve, ha az OpenSSL támogatást befordítottad a PHP-be, a *`hostname`* elé írhatsz '`ssl://`' vagy '`tls://`' karaktersorozatot is, ha SSL vagy TLS kliens kapcsolatot szeretnél létesíteni a TCP/IP felett a távoli hoszttal.

A **fsockopen()** függvény egy fájl-azonosítót ad vissza, ami a rendes fájlkezelő parancsokkal együtt használható átlátszó módon. Ezek a következőek: fgets(), fgetss(), fputs(), fclose(), és feof()).

Ha a kapcsolatfelvétel meghiúsul, a visszatérési érték `FALSE` lesz, és amennyiben az *`errno`* és *`errstr`* opcionális paraméterek meg lettek adva, ezekben visszakapjuk a `connect()` rendszerhívás hibaértékét és hibaüzenetét. Amennyiben az *`errno`* nulla értékkel jön vissza, miközben a függvény értéke is `FALSE`, akkor a hiba még valahol a `connect()` rendszerhívás előtt következett be. Ez általában annak köszönhető, hogy valami probléma merül fel már az inicializáció közben. Az *`errno`* és *`errstr`* argumentumok referenciaként adják vissza a visszatérési értéket.

Bizonyos rendszereken előfordulhat, hogy a Unix domain illetve a *`timeout`* nem használható.

Alapesetben a socket blokkolt módban nyílik meg. Amennyiben ez nem megfelelő, ez átkapcsolható a socket_set_blocking() segítségével.

Példa 1. fsockopen() Példa

```
$fp = fsockopen ("www.php.net", 80, $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br>\n";
} else {
    fputs ($fp, "GET / HTTP/1.0\r\nHost: www.php.net\r\n\r\n");
    while (!feof($fp)) {
        echo fgets ($fp,128);
    }
    fclose ($fp);
```

```
}
?>
```

A következő példa bemutatja, hogy gyűjthető be a saját gépünkön levő "daytime" nevű, 13-as porton figyelő szolgáltatástól információ.

Példa 2. UDP kapcsolat használata

```
<?php
$fp = fsockopen("udp://127.0.0.1", 13, $errno, $errstr);
if (!$fp) {
    echo "Hiba: $errno - $errstr<br>\n";
} else {
    fwrite($fp,"\n");
    echo fread($fp, 26);
    fclose($fp);
}
?>
```

Megjegyzés: A timeout paraméter a 3.0.9-es verziótól felfele, míg az UDP támogatás csak a 4-es verziókban elérhető.

Lásd még: pfsockopen(), socket_set_blocking(), socket_set_timeout(), fgets(), fgetss(), fputs(), fclose() és feof()!

gethostbyaddr (PHP 3, PHP 4)

Adott IP címnek megfelelő hostnév kinyerése

string **gethostbyaddr** (string ip_cím) \linebreak

Az *ip_cím*-nek megfelelő hostnevet adja vissza. Hiba esetén a kapott IP címet adja vissza.

Lásd még: gethostbyname()!

gethostbyname (PHP 3, PHP 4)

Kinyeri a megadott Internet hostnak megfelelő IP címet

string **gethostbyname** (string hostnév) \linebreak

A *hostname* paraméter által megadott Internet host IP címét adja vissza.

Lásd még: gethostbyaddr()!

gethostbynamel (PHP 3, PHP 4)

Adott hostnévnek megfelelő összes IP cím listájának kinyerése

array **gethostbynamel** (string hostnév) \linebreak

Tömbben adja vissza a megadott *hostnév*-re illő IP címek listáját.

Lásd még: gethostbyname(), gethostbyaddr(), checkdnsrr(), getmxrr(), valamint a man `named(8)` oldala!

getmxrr (PHP 3, PHP 4)

Adott Internet hostnévhez tartozó MX rekordok kigyűjtése

int **getmxrr** (string hostnév, tömb mxhosts [, array weight]) \linebreak

Névszerver keresést kezdeményez a megadott *hostnév*-re, MX (Mail Exchanger - levélszerver) rekordok után kutatva. Találat esetén `TRUE` értékkel tér vissza, míg hiba estén vagy találat hiányában `FALSE` értékkel.

A talált MX rekordokat az *mxhosts* argumentumban megadott tömbben kapjuk vissza. A *weight* opcionális paramétert megadva, az ott megjelölt tömbben viszzakapjuk az egyes MX rekordok súlyozottságát.

Lásd még: checkdnsrr(), gethostbyname(), gethostbynamel(), gethostbyaddr(), valamint a man `named(8)` oldala!

getprotobyname (PHP 4)

Névvel megadott protokol számát adja vissza

int **getprotobyname** (string név) \linebreak

A **getprotobyname()** függvény a *név* paraméterben megadott protokol számát adja vissza. A *név* paraméternek az `/etc/protocols`-ban megadott névnek kell, hogy megfeleljen.

Lásd még: getprotobynumber()!

getprotobynumber (PHP 4)

Számmal megadott protokol nevét adja vissza

string **getprotobynumber** (int protokolszám) \linebreak

A **getprotobynumber()** visszaadja a *protokolszám* paraméternek megfelelő protokol `/etc/protocols` szerinti nevét.

Lásd még: getprotobyname()!

getservbyname (PHP 4)

Adott szolgáltatástípus portszámának megallapítása név alapján

int **getservbyname** (string szolgáltatás, string protokol) \linebreak

A **getservbyname()** a *`szolgáltatás`* és a *`protokol`* paraméterekre illő portszámot adja vissza. A szolgáltatás neve az /etc/services-nek megfelelő kell, hogy legyen. A *`protokol`* paraméter értéke `"tcp"` avagy `"ucp"` lehet (csupa kisbetűvel!).

Lásd még: getservbyport()!

getservbyport (PHP 4)

Adott porton levő szolgáltatás nevének megállapítása

string **getservbyport** (int port, string protokol) \linebreak

A **getservbyport()** /etc/services szerű nevét adja vissza az adott *`port`*-on levő, adott *`protokol`*-ú szolgáltatásnak. A *`protokol`* paraméter `"tcp"` vagy `"udp"` értéket vehet fel (csupa kisbetűvel).

Lásd még: getservbyname()!

ip2long (PHP 4)

Pontozott IP címet tartalmazó karakterláncot valódi IP cimmé alakít

int **ip2long** (string ip_cím) \linebreak

Az **ip2long()** függvény IPv4 formátumú IP címmé alakítja az *`ip_cím`* paraméterben megadott szabványos, pontozott formátumú IP-t. Az IP címek valójában ebben a 4 byteos egész szám formátumban tárolódnak.

Példa 1. ip2long() Példa

```
<?php
$ip = gethostbyname("www.php.net");
$out = "Az alábbi URL-ek egyenértékűek:<br>\n";
$out .= "http://hu.php.net/, http://".$ip."/, és http://".ip2long($ip)."/<br>\n";
echo $out;
?>
```

Megjegyzés: Mivel a PHP előjeles egésszám ábrázolást alkalmaz, és az IP címek negatív számokat eredményezhetnek ezáltal, szükséged lehet a sprintf() vagy printf() függvény "%u" formázó elemére. Ezáltal a megjelenített szám ábrázolása előjel nélküli lesz.

Eme második példa pedig azt mutatja meg, hogy a printf() segítségével hogy tudjuk ez a lekonvertált címet kiíratni:

Példa 2. IP cím kiíratása

```
<?php
$ip = gethostbyname("hu.php.net");
printf ("%u\n", ip2long($ip));
echo $out;
?>
```

Lásd még: long2ip()!

long2ip (PHP 4)

Négy byteon tárolt (IPv4) hálózatcímet alakít az Internet szabványnak megfelelő pontozott formára

string **long2ip** (int proper_address) \linebreak

A **long2ip()** függvény pontozott formátumú (aaa.bbb.ccc.ddd) Internet címet hoz létre a paraméterben megadott valódi IP hálózati címből.

Lásd még: ip2long()!

openlog (PHP 3, PHP 4)

A rendszer naplózóval való kapcsolat nyitása

int **openlog** (string ident, int option, int facility) \linebreak

Az **openlog()** függvény segítségével nyithatunk egy program számára kapcsolatot a rendszer naplózóval. Az *`ident`* argumentumban megadott szöveg minden naplóüzenet elejére kerül. Az *`option`* és a *`facility`* lehetséges értékei alant találhatóak. Az *`option`* paraméterrel jelezhetőek a különleges opciók, amik belejáthszhatnak a naplózás mikéntjébe. Kicsit lejjebb található róluk táblázat. A *`facility`* paraméterben adható meg, milyen csoporthoz tartozó program küldte ezt a naplóüzenetet. Azt, hogy milyen *`facility`* típussal hogy bánjon a syslog, azt a gépünk syslog konfigurációjában lehet részletezni. Az **openlog()** függvény használata nem szükségszerű, mivel a syslog() ezt amúgy is megteszi, ha szükség van rá. Ebben az esetben az *`ident`* paraméter `FALSE` lesz.

Táblázat 1. openlog() Opciók

Konstans	Leírás
LOG_CONS	bárminemű hiba esetén, ami a rendszer naplózó felé adatot küldés folyamán felmerül, küldje azt egyenesen a rendszer konzoljára
LOG_NDELAY	azonnali kapcsolatnyitás a rendszer-naplózóhoz

Konstans	Leírás
LOG_ODELAY	(alapértelmezett) várjon a csatlakozással, amíg az első naplóüzenet meg nem érkezik
LOG_PERROR	a naplóüzeneteket az alapértelmezett hibakimenetre is küldje
LOG_PID	minden üzenethez mellékelje a processzazonosítót (PID) is

Ezen opciók közül egyet, de akár többet is fel lehet használni. Több opció megadása esetén az opciók közt 'vagy' kapcsolatot kell létrehozni. Amennyiben például egy azonnal nyitandó kapcsolat kell, ami írjon a konzolra és a PID-eket is mellékelje, azt igy kell megadni: `LOG_CONS | LOG_NDELAY | LOG_PID`

Táblázat 2. openlog() Szolgáltatás csoportok (facility paraméter)

Konstans	Leírás
LOG_AUTH	biztonsági/authorizációs üzenetek (ehelyett a LOG_AUTHPRIV-et használja, amely rendszerekben ez lehetséges)
LOG_AUTHPRIV	biytonsági/authorizációs üzenetek (privát)
LOG_CRON	clock daemon (cron és at)
LOG_DAEMON	egyéb rendszer daemon-ok
LOG_KERN	kernel üzenetek
LOG_LOCAL0 ... LOG_LOCAL7	helyi használatra fentartott
LOG_LPR	soros nyomtató alrendszer
LOG_MAIL	levél alrendszer
LOG_NEWS	USENET news alrendszer
LOG_SYSLOG	a syslogd saját belső üzenetei
LOG_USER	általános, felhasználó szinten történt bejegyzés
LOG_UUCP	UUCP alrendszer

Lásd még: define_syslog_variables(), syslog() és closelog()!

pfsockopen (PHP 3>= 3.0.7, PHP 4)

Tartós Internet vagy Unix domain socket-kapcsolat megnyitása

int **pfsockopen** (string hostnév, int port [, int errno [, string errstr [, int timeout]]]) \linebreak

Teljességgel megegyezik eme függvény és a fsockopen(), csupán annyi különbséggel, hogy a **pfsockopen()** esetén a kapcsolat a php script futásának végeztével nem szűnik meg.

socket_get_status (PHP 4)

Egy már meglévő socket adatainak kinyerése

array **socket_get_status** (resource socket_get_status) \linebreak

A megadott socketről ad vissta információt. Jelen pillanatban ez egy 4 elemes tömböt jelent:

- *`timed_out`* (bool) - időtúllépés történt a socketen, adatra várás közben
- *`blocked`* (bool) - A socket blokkolt
- *`eof`* (bool) - fájl vége jelzés volt
- *`unread_bytes`* (int) - A még be nem olvasott byte-ok száma a pufferben

Lásd még: socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_strerror(), és a Socket kiegészítések!

socket_set_blocking (PHP 4)

Adott socketen a blokkoló/nem blokkoló mód kapcsolgatása

int **socket_set_blocking** (int socket azonosító, int mód) \linebreak

Ha a *`mód`* paraméter értéke `FALSE`, a megadott socket nem blokkolt módba kapcsolódik. Ha ez a paraméter `TRUE` értéket vesz fel, az esetben blokkoló módba lesz kapcsolva az adott socket, melyet a *`socket azonosító`* paraméterrel adhatunk meg. Ez például egy adott socketről fgets()-el való olvasásra lehet hatással. Nem blokkolt módban azonnal visszatér az fgets() valamiféle értékkel, míg blokkolt módban vár, amíg adatot nem kap.

Korábban ezt a függvényt **set_socket_blocking()** vezették be, de annak használata nem javallott.

socket_set_timeout (PHP 4)

Adott socket időtúllépési periódusának állítása

bool **socket_set_timeout** (int socket azonosító, int másodpercek, int ezredmásodpercek) \linebreak

A *`socket azonosító`* paraméterben azonosított socket időtúllépési tartamát adhatjuk meg vele, melyek értékét a *`másodpercek`*, valamint az *`ezredmásodpercek`* paraméterben adhatunk meg.

Példa 1. socket_set_timeout() Example

```
<?php
$fp = fsockopen("www.php.net", 80);
if(!$fp) {
    echo "Nem sikerült a webhely megnyitása\n";
} else {
    fputs($fp,"GET / HTTP/1.0\n\n");
    $start = time();
```

```
    socket_set_timeout($fp, 2);
    $res = fread($fp, 2000);
    var_dump(socket_get_status($fp));
    fclose($fp);
    print $res;
}
?>
```

E függvény korábbi neve **set_socket_timeout()** volt, annak használata nem javallott.

Lásd még: fsockopen() és fopen()!

syslog (PHP 3, PHP 4)

Syslog üzenet létrehozása

int **syslog** (int prioritás, string üzenet) \linebreak

A **syslog()** egy naplóüzenetet hoz létre, amit a rendszer naplózó rendszeren keresztül tesz közzé. A *`prioritás`* paraméter két dolog keverékéből jön, egyrészt az állapotból, másrészt a bejegyzési szintből, ennek részletezése az alant látható táblázatban látható. A másik paraméterben adható meg az üzenet konkrét szövege, amit egy az egyben továbbít, kivéve a `%m` karaktersot, ennek helyére az errno aktuális értékének megfelelő hibaüzenetet helyettesíti be.

Táblázat 1. syslog() prioritási szintek (csökkenő sorrendben)

Konstans	Leírás
LOG_EMERG	a rendszer használhatatlan
LOG_ALERT	azonnali beavatkozás szükségeltetik
LOG_CRIT	kritikus állapot
LOG_ERR	hibajellentést okozó állapot
LOG_WARNING	csak figyelmeztetést okozó állapot
LOG_NOTICE	normál, de említésre méltó körülmény
LOG_INFO	információs üzenet
LOG_DEBUG	debug szintű üzenet

Példa 1. A syslog() használata

```
<?php
define_syslog_variables();
// syslog nyitása, processz azonosítóstól. A log
// menjen ki az alapértelmezett hibakimenetre is,
// valamint a felhasználó által definiált naplózó
// mechanizmus is kapja meg a logot
```

```
openlog("myScripLog", LOG_PID | LOG_PERROR, LOG_LOCAL0);

// egy kis programkód

if (authorized_client()) {
    // itt lehet csinálni valami jogosultat :)
} else {
    // nem azonosított felhsználó!
    // a próbálkozás naplózásra kerül
    $access = date("Y/m/d H:i:s");

    syslog(LOG_WARNING,"Unauthorized client: $access $REMOTE_ADDR ($HTTP_USER_AGENT)");
}
closelog();
?>
```

Saját definiálású syslog kezelő felállításához érdemes a syslog.conf(5) Unix man oldalt olvasgatni. További információ a syslog többi állapotjelző lehetőségeiről a syslog(3) man oldalon található

Windows NT alatt a syslog emulálva van, az Event Log segítségével.

Lásd még: define_syslog_variables(), openlog() és closelog()!

LXVII. Ncurses terminal screen control functions

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

What is ncurses?

ncurses (new curses) is a free software emulation of curses in System V Rel 4.0 (and above). It uses terminfo format, supports pads, colors, multiple highlights, form characters and function key mapping.

Platforms

Ncurses is available for the following platforms:

- AIX
- BeOS
- Cygwin
- Digital Unix (aka OSF1)
- FreeBSD
- GNU/Linux
- HPUX
- IRIX
- OS/2
- SCO OpenServer
- Solaris
- SunOS

Requirements

You need the ncurses libraries and headerfiles. Download the latest version from the

ftp://ftp.gnu.org/pub/gnu/ncurses/ or from an other GNU-Mirror.

Installation

To get these functions to work, you have to compile the CGI version of PHP with `--with-ncurses`.

Ncurses predefined constants

Error codes

On error ncurses functions return NCURSES_ERR.

Colors

Táblázat 1. ncurses color constants

constant	meaning
NCURSES_COLOR_BLACK	no color (black)
NCURSES_COLOR_WHITE	white
NCURSES_COLOR_RED	red - supported when terminal is in color mode
NCURSES_COLOR_GREEN	green - supported when terminal is in color mod
NCURSES_COLOR_YELLOW	yellow - supported when terminal is in color mod
NCURSES_COLOR_BLUE	blue - supported when terminal is in color mod
NCURSES_COLOR_CYAN	cyan - supported when terminal is in color mod
NCURSES_COLOR_MAGENTA	magenta - supported when terminal is in color mod

Keys

Táblázat 2. ncurses key constants

constant	meaning
NCURSES_KEY_F0 - NCURSES_KEY_F64	function keys F1 - F64
NCURSES_KEY_DOWN	down arrow
NCURSES_KEY_UP	up arrow
NCURSES_KEY_LEFT	left arrow
NCURSES_KEY_RIGHT	right arrow
NCURSES_KEY_HOME	home key (upward+left arrow)
NCURSES_KEY_BACKSPACE	backspace
NCURSES_KEY_DL	delete line

constant	**meaning**
NCURSES_KEY_IL	insert line
NCURSES_KEY_DC	delete character
NCURSES_KEY_IC	insert char or enter insert mode
NCURSES_KEY_EIC	exit insert char mode
NCURSES_KEY_CLEAR	clear screen
NCURSES_KEY_EOS	clear to end of screen
NCURSES_KEY_EOL	clear to end of line
NCURSES_KEY_SF	scroll one line forward
NCURSES_KEY_SR	scroll one line backward
NCURSES_KEY_NPAGE	next page
NCURSES_KEY_PPAGE	previous page
NCURSES_KEY_STAB	set tab
NCURSES_KEY_CTAB	clear tab
NCURSES_KEY_CATAB	clear all tabs
NCURSES_KEY_SRESET	soft (partial) reset
NCURSES_KEY_RESET	reset or hard reset
NCURSES_KEY_PRINT	print
NCURSES_KEY_LL	lower left
NCURSES_KEY_A1	upper left of keypad
NCURSES_KEY_A3	upper right of keypad
NCURSES_KEY_B2	center of keypad
NCURSES_KEY_C1	lower left of keypad
NCURSES_KEY_C3	lower right of keypad
NCURSES_KEY_BTAB	back tab
NCURSES_KEY_BEG	beginning
NCURSES_KEY_CANCEL	cancel
NCURSES_KEY_CLOSE	close
NCURSES_KEY_COMMAND	cmd (command)
NCURSES_KEY_COPY	copy
NCURSES_KEY_CREATE	create
NCURSES_KEY_END	end
NCURSES_KEY_EXIT	exit
NCURSES_KEY_FIND	find
NCURSES_KEY_HELP	help
NCURSES_KEY_MARK	mark
NCURSES_KEY_MESSAGE	message
NCURSES_KEY_MOVE	move
NCURSES_KEY_NEXT	next
NCURSES_KEY_OPEN	open
NCURSES_KEY_OPTIONS	options
NCURSES_KEY_PREVIOUS	previous
NCURSES_KEY_REDO	redo

constant	**meaning**
NCURSES_KEY_REFERENCE	ref (reference)
NCURSES_KEY_REFRESH	refresh
NCURSES_KEY_REPLACE	replace
NCURSES_KEY_RESTART	restart
NCURSES_KEY_RESUME	resume
NCURSES_KEY_SAVE	save
NCURSES_KEY_SBEG	shiftet beg (beginning)
NCURSES_KEY_SCANCEL	shifted cancel
NCURSES_KEY_SCOMMAND	shifted command
NCURSES_KEY_SCOPY	shifted copy
NCURSES_KEY_SCREATE	shifted create
NCURSES_KEY_SDC	shifted delete char
NCURSES_KEY_SDL	shifted delete line
NCURSES_KEY_SELECT	select
NCURSES_KEY_SEND	shifted end
NCURSES_KEY_SEOL	shifted end of line
NCURSES_KEY_SEXIT	shifted exit
NCURSES_KEY_SFIND	shifted find
NCURSES_KEY_SHELP	shifted help
NCURSES_KEY_SHOME	shifted home
NCURSES_KEY_SIC	shifted input
NCURSES_KEY_SLEFT	shifted left arrow
NCURSES_KEY_SMESSAGE	shifted message
NCURSES_KEY_SMOVE	shifted move
NCURSES_KEY_SNEXT	shifted next
NCURSES_KEY_SOPTIONS	shifted options
NCURSES_KEY_SPREVIOUS	shifted previous
NCURSES_KEY_SPRINT	shifted print
NCURSES_KEY_SREDO	shifted redo
NCURSES_KEY_SREPLACE	shifted replace
NCURSES_KEY_SRIGHT	shifted right arrow
NCURSES_KEY_SRSUME	shifted resume
NCURSES_KEY_SSAVE	shifted save
NCURSES_KEY_SSUSPEND	shifted suspend
NCURSES_KEY_UNDO	undo
NCURSES_KEY_MOUSE	mouse event has occured
NCURSES_KEY_MAX	maximum key value

Mouse

Táblázat 3. mouse constants

Constant	meaning
NCURSES_BUTTON1_RELEASED - NCURSES_BUTTON4_RELEASED	button (1-4) released
NCURSES_BUTTON1_PRESSED - NCURSES_BUTTON4_PRESSED	button (1-4) pressed
NCURSES_BUTTON1_CLICKED - NCURSES_BUTTON4_CLICKED	button (1-4) clicked
NCURSES_BUTTON1_DOUBLE_CLICKED - NCURSES_BUTTON4_DOUBLE_CLICKED	button (1-4) double clicked
NCURSES_BUTTON1_TRIPLE_CLICKED - NCURSES_BUTTON4_TRIPLE_CLICKED	button (1-4) triple clicked
NCURSES_BUTTON_CTRL	ctrl pressed during click
NCURSES_BUTTON_SHIFT	shift pressed during click
NCURSES_BUTTON_ALT	alt pressed during click
NCURSES_ALL_MOUSE_EVENTS	report all mouse events
NCURSES_REPORT_MOUSE_POSITION	report mouse position

ncurses_addch (PHP 4 >= 4.1.0)

Add character at current position and advance cursor

int **ncurses_addch** (int ch) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_addchnstr (PHP 4 >= 4.2.0)

Add attributed string with specified length at current position

int **ncurses_addchnstr** (string s, int n) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_addchstr (PHP 4 >= 4.2.0)

Add attributed string at current position

int **ncurses_addchstr** (string s) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_addnstr (PHP 4 >= 4.2.0)

Add string with specified length at current position

int **ncurses_addnstr** (string s, int n) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_addstr (PHP 4 >= 4.2.0)

Output text at current position

int **ncurses_addstr** (string text) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_assume_default_colors (PHP 4 >= 4.2.0)

Define default colors for color 0

int **ncurses_assume_default_colors** (int fg, int bg) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_attroff (PHP 4 >= 4.1.0)

Turn off the given attributes

int **ncurses_attroff** (int attributes) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_attron (PHP 4 >= 4.1.0)

Turn on the given attributes

int **ncurses_attron** (int attributes) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_attrset (PHP 4 >= 4.1.0)

Set given attributes

int **ncurses_attrset** (int attributes) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_baudrate (PHP 4 >= 4.1.0)

Returns baudrate of terminal

int **ncurses_baudrate** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_beep (PHP 4 >= 4.1.0)

Let the terminal beep

int **ncurses_beep** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_beep() sends an audlible alert (bell) and if its not possible flashes the screen. Returns FALSE on success, otherwise TRUE.

See also: ncurses_flash()

ncurses_bkgd (PHP 4 >= 4.1.0)

Set background property for terminal screen

int **ncurses_bkgd** (int attrchar) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_bkgdset (PHP 4 >= 4.1.0)

Control screen background

void **ncurses_bkgdset** (int attrchar) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_border (PHP 4 >= 4.2.0)

Draw a border around the screen using attributed characters

int **ncurses_border** (int left, int right, int top, int bottom, int tl_corner, int tr_corner, int bl_corner, int br_corner) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_can_change_color (PHP 4 >= 4.1.0)

Check if we can change terminals colors

bool **ncurses_can_change_color** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

The function **ncurses_can_change_color()** returns TRUE or FALSE, depending on whether the terminal has color capabilities and whether the programmer can change the colors.

ncurses_cbreak (PHP 4 >= 4.1.0)

Switch of input buffering

bool **ncurses_cbreak** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_cbreak() disables line buffering and character processing (interrupt and flow control characters are unaffected), making characters typed by the user immediately available to the program.

ncurses_cbreak() returns TRUE or NCURSES_ERR if any error occured.

See also: ncurses_nocbreak()

ncurses_clear (PHP 4 >= 4.1.0)

Clear screen

bool **ncurses_clear** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_clear() clears the screen completely without setting blanks. Returns FALSE on success, otherwise TRUE.

Note: **ncurses_clear()** clears the screen without setting blanks, which have the current background rendition. To clear screen with blanks, use ncurses_erase().

See also: ncurses_erase()

ncurses_clrtobot (PHP 4 >= 4.1.0)

Clear screen from current position to bottom

bool **ncurses_clrtobot** (void) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_clrtobot() erases all lines from cursor to end of screen and creates blanks. Blanks created by **ncurses_clrtobot()** have the current background rendition. Returns TRUE if any error occured, otherwise FALSE.

See also: ncurses_clear(), ncurses_clrtoeol()

ncurses_clrtoeol (PHP 4 >= 4.1.0)

Clear screen from current position to end of line

bool **ncurses_clrtoeol** (void) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_clrtoeol() erases the current line from cursor position to the end. Blanks created by **ncurses_clrtoeol()** have the current background rendition. Returns TRUE if any error occured, otherwise FALSE.

See also: ncurses_clear(), ncurses_clrtobot()

ncurses_color_set (PHP 4 >= 4.1.0)

Set fore- and background color

int **ncurses_color_set** (int pair) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_curs_set (PHP 4 >= 4.1.0)

Set cursor state

int **ncurses_curs_set** (int visibility) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_def_prog_mode (PHP 4 >= 4.1.0)

Saves terminals (program) mode

bool **ncurses_def_prog_mode** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_def_prog_mode() saves the current terminal modes for program (in curses) for use by **ncurses_reset_prog_mode()**. Returns FALSE on success, otherwise TRUE.

See also: **ncurses_reset_prog_mode()**

ncurses_def_shell_mode (PHP 4 >= 4.1.0)

Saves terminals (shell) mode

bool **ncurses_def_shell_mode** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_def_shell_mode() saves the current terminal modes for shell (not in curses) for use by **ncurses_reset_shell_mode()**. Returns FALSE on success, otherwise TRUE.

See also: **ncurses_reset_shell_mode()**

ncurses_define_key (PHP 4 >= 4.2.0)

Define a keycode

int **ncurses_define_key** (string definition, int keycode) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_delay_output (PHP 4 >= 4.1.0)

Delay output on terminal using padding characters

int **ncurses_delay_output** (int milliseconds) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_delch (PHP 4 >= 4.1.0)

Delete character at current position, move rest of line left

bool **ncurses_delch** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_delch() deletes the character under the cursor. All characters to the right of the cursor on the same line are moved to the left one position and the last character on the line is filled with a blank. The cursor position does not change. Returns FALSE on success, otherwise TRUE.

See also: ncurses_deleteln()

ncurses_deleteln (PHP 4 >= 4.1.0)

Delete line at current position, move rest of screen up

bool **ncurses_deleteln** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_deleteln() deletes the current line under cursorposition. All lines below the current line are moved up one line. The bottom line of window is cleared. Cursor position does not change. Returns FALSE on success, otherwise TRUE.

See also: ncurses_delch()

ncurses_delwin (PHP 4 >= 4.1.0)

Delete a ncurses window

int **ncurses_delwin** (resource window) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_doupdate (PHP 4 >= 4.1.0)

Write all prepared refreshes to terminal

bool **ncurses_doupdate** (void) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_doupdate()() compares the virtual screen to the physical screen and updates the physical screen. This way is more effective than using multiple refresh calls. Returns FALSE on success, TRUE if any error occured.

ncurses_echo (PHP 4 >= 4.1.0)

Activate keyboard input echo

bool **ncurses_echo** (void) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_echo() enables echo mode. All characters typed by user are echoed by ncurses_getch(). Returns FALSE on success, TRUE if any error occured.

To disable echo mode use ncurses_noecho().

ncurses_echochar (PHP 4 >= 4.1.0)

Single character output including refresh

int **ncurses_echochar** (int character) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_end (PHP 4 >= 4.1.0)

Stop using ncurses, clean up the screen

int **ncurses_end** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_erase (PHP 4 >= 4.1.0)

Erase terminal screen

bool **ncurses_erase** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_erase() fills the terminal screen with blanks. Created blanks have the current background rendition, set by ncurses_bkgd(). Returns FALSE on success, TRUE if any error occured.

See also: ncurses_bkgd(), ncurses_clear()

ncurses_erasechar (PHP 4 >= 4.1.0)

Returns current erase character

string **ncurses_erasechar** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_erasechar() returns the current erase char character.

See also: ncurses_killchar()

ncurses_filter (PHP 4 >= 4.1.0)

int **ncurses_filter** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_flash (PHP 4 >= 4.1.0)

Flash terminal screen (visual bell)

bool **ncurses_flash** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_flash() flashes the screen, and if its not possible, sends an audible alert (bell). Returns FALSE on success, otherwise TRUE.

See also: ncurses_beep()

ncurses_flushinp (PHP 4 >= 4.1.0)

Flush keyboard input buffer

bool **ncurses_flushinp** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

The **ncurses_flushinp()** throws away any typeahead that has been typed and has not yet been read by your program. Returns FALSE on success, otherwise TRUE.

ncurses_getch (PHP 4 >= 4.1.0)

Read a character from keyboard

int **ncurses_getch** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_getmouse (PHP 4 >= 4.2.0)

Reads mouse event

bool **ncurses_getmouse** (array mevent) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_getmouse() reads mouse event out of queue. Function **ncurses_getmouse()** will return ;FALSE if a mouse event is actually visible in the given window, otherwise it will return TRUE. Event options will be delivered in parameter *mevent*, which has to be an array, passed by reference (see example below). On success an associative array with following keys will be delivered:

- "id" : Id to distinguish multiple devices
- "x" : screen relative x-position in character cells
- "y" : screen relative y-position in character cells
- "z" : currently not supported
- "mmask" : Mouse action

Példa 1. ncurses_getmouse() example

```
switch (ncurses_getch){
  case NCURSES_KEY_MOUSE:
    if (!ncurses_getmouse(&$mevent)){
```

```
      if ($mevent["mmask"] & NCURSES_MOUSE_BUTTON1_PRESSED){
        $mouse_x = $mevent["x"]; // Save mouse position
        $mouse_y = $mevent["y"];
      }
    }
  break;

  default:
    ....
}
```

See also: ncurses_ungetmouse()

ncurses_halfdelay (PHP 4 >= 4.1.0)

Put terminal into halfdelay mode

int **ncurses_halfdelay** (int tenth) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_has_colors (PHP 4 >= 4.1.0)

Check if terminal has colors

bool **ncurses_has_colors** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_has_colors() returns `TRUE` or `FALSE` depending on whether the terminal has color capacitites.

See also: ncurses_can_change_color()

ncurses_has_ic (PHP 4 >= 4.1.0)

Check for insert- and delete-capabilities

bool **ncurses_has_ic** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_has_ic() checks terminals insert- and delete capabilitites. It returns TRUE when terminal has insert/delete-capabilities, otherwise FALSE.

See also: ncurses_has_il()

ncurses_has_il (PHP 4 >= 4.1.0)

Check for line insert- and delete-capabilities

bool **ncurses_has_il** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_has_il() checks terminals insert- and delete-line-capabilities. It returns TRUE when terminal has insert/delete-line capabilities, otherwise FALSE

See also: ncurses_has_ic()

ncurses_has_key (PHP 4 >= 4.1.0)

Check for presence of a function key on terminal keyboard

int **ncurses_has_key** (int keycode) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_hline (PHP 4 >= 4.2.0)

Draw a horizontal line at current position using an attributed character and max. n characters long

int **ncurses_hline** (int charattr, int n) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_inch (PHP 4 >= 4.1.0)

Get character and attribute at current position

string **ncurses_inch** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_inch() returns the character from the current position.

ncurses_init_color (PHP 4 >= 4.2.0)

Set new RGB value for color

int **ncurses_init_color** (int color, int r, int g, int b) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_init_pair (PHP 4 >= 4.1.0)

Allocate a color pair

int **ncurses_init_pair** (int pair, int fg, int bg) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_init (PHP 4 >= 4.1.0)

Initialize ncurses

int **ncurses_init** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_insch (PHP 4 >= 4.1.0)

Insert character moving rest of line including character at current position

int **ncurses_insch** (int character) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_insdelln (PHP 4 >= 4.1.0)

Insert lines before current line scrolling down (negative numbers delete and scroll up)

int **ncurses_insdelln** (int count) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_insertln (PHP 4 >= 4.1.0)

Insert a line, move rest of screen down

bool **ncurses_insertln** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_insertln() inserts a new line above the current line. The bottom line will be lost.

ncurses_insstr (PHP 4 >= 4.2.0)

Insert string at current position, moving rest of line right

int **ncurses_insstr** (string text) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_instr (PHP 4 >= 4.2.0)

Reads string from terminal screen

int **ncurses_instr** (string buffer) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_instr() returns the number of charaters read from the current character position until end of line. *buffer* contains the characters. Atrributes are stripped from the characters.

ncurses_isendwin (PHP 4 >= 4.1.0)

Ncurses is in endwin mode, normal screen output may be performed

bool **ncurses_isendwin** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_isendwin() returns TRUE, if **ncurses_endwin()** has been called without any subsequent calls to ncurses_wrefresh(), otherwise FALSE.

See also: **ncurses_endwin() ncurses_wrefresh()()**

ncurses_keyok (PHP 4 >= 4.2.0)

Enable or disable a keycode

int **ncurses_keyok** (int keycode, bool enable) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_killchar (PHP 4 >= 4.1.0)

Returns current line kill character

bool **ncurses_killchar** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_killchar() returns the current line kill character.

See also: ncurses_erasechar()

ncurses_longname (PHP 4 >= 4.2.0)

Returns terminals description

string **ncurses_longname** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_longname() returns a verbose description of the terminal. The descritpion is truncated to 128 characters. On Error **ncurses_longname()** returns NULL.

See also: ncurses_termname()

ncurses_mouseinterval (PHP 4 >= 4.1.0)

Set timeout for mouse button clicks

int **ncurses_mouseinterval** (int milliseconds) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mousemask (PHP 4 >= 4.2.0)

Sets mouse options

int **ncurses_mousemask** (int newmask, int oldmask) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Function **ncurses_mousemask()** will set mouse events to be reported. By default no mouse events will be reported. The function **ncurses_mousemask()** will return a mask to indicated which of the in parameter *newmask* specified mouse events can be reported. On complete failure, it returns 0. In parameter *oldmask*, which is passed by reference **ncurses_mousemask()** returns the previous value of mouse event mask. Mouse events are represented bei NCURSES_KEY_MOUSE in the **ncurses_wgetch()** input stream. To read the event data and pop the event of of queue, call ncurses_getmouse().

As a side effect, setting a zero mousemask in *newmask* turns off the mouse pointer. Setting a non zero value turns mouse pointer on.

mouse mask options can be set with the following predefined constants:

- NCURSES_BUTTON1_PRESSED
- NCURSES_BUTTON1_RELEASED
- NCURSES_BUTTON1_CLICKED
- NCURSES_BUTTON1_DOUBLE_CLICKED
- NCURSES_BUTTON1_TRIPLE_CLICKED
- NCURSES_BUTTON2_PRESSED
- NCURSES_BUTTON2_RELEASED
- NCURSES_BUTTON2_CLICKED
- NCURSES_BUTTON2_DOUBLE_CLICKED
- NCURSES_BUTTON2_TRIPLE_CLICKED
- NCURSES_BUTTON3_PRESSED
- NCURSES_BUTTON3_RELEASED
- NCURSES_BUTTON3_CLICKED
- NCURSES_BUTTON3_DOUBLE_CLICKED
- NCURSES_BUTTON3_TRIPLE_CLICKED
- NCURSES_BUTTON4_PRESSED
- NCURSES_BUTTON4_RELEASED
- NCURSES_BUTTON4_CLICKED
- NCURSES_BUTTON4_DOUBLE_CLICKED

- NCURSES_BUTTON4_TRIPLE_CLICKED
- NCURSES_BUTTON_SHIFT>
- NCURSES_BUTTON_CTRL
- NCURSES_BUTTON_ALT
- NCURSES_ALL_MOUSE_EVENTS
- NCURSES_REPORT_MOUSE_POSITION

See also: ncurses_getmouse(), ncurses_ungetmouse() **ncurese_getch()**

Példa 1. ncurses_mousemask() example

```
$newmask = NCURSES_BUTTON1_CLICKED + NCURSES_BUTTON1_RELEASED;
$mask = ncurses_mousemask($newmask, &$oldmask);
if ($mask & $newmask){
  printf ("All specified mouse options will be supported\n");
}
```

ncurses_move (PHP 4 >= 4.1.0)

Move output position

int **ncurses_move** (int y, int x) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK* . A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvaddch (PHP 4 >= 4.2.0)

Move current position and add character

int **ncurses_mvaddch** (int y, int x, int c) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvaddchnstr (PHP 4 >= 4.2.0)

Move position and add attrributed string with specified length

int **ncurses_mvaddchnstr** (int y, int x, string s, int n) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvaddchstr (PHP 4 >= 4.2.0)

Move position and add attributed string

int **ncurses_mvaddchstr** (int y, int x, string s) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvaddnstr (PHP 4 >= 4.2.0)

Move position and add string with specified length

int **ncurses_mvaddnstr** (int y, int x, string s, int n) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvaddstr (PHP 4 >= 4.2.0)

Move position and add string

int **ncurses_mvaddstr** (int y, int x, string s) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvcur (PHP 4 >= 4.2.0)

Move cursor immediately

int **ncurses_mvcur** (int old_y, int old_x, int new_y, int new_x) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvdelch (PHP 4 >= 4.2.0)

Move position and delete character, shift rest of line left

int **ncurses_mvdelch** (int y, int x) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvgetch (PHP 4 >= 4.2.0)

Move position and get character at new position

int **ncurses_mvgetch** (int y, int x) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvhline (PHP 4 >= 4.2.0)

Set new position and draw a horizontal line using an attributed character and max. n characters long

int **ncurses_mvhline** (int y, int x, int attrchar, int n) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvinch (PHP 4 >= 4.2.0)

Move position and get attributed character at new position

int **ncurses_mvinch** (int y, int x) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvvline (unknown)

Set new position and draw a vertical line using an attributed character and max. n characters long

int **ncurses_mvvline** (int y, int x, int attrchar, int n) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_mvwaddstr (PHP 4 >= 4.2.0)

Add string at new position in window

int **ncurses_mvwaddstr** (resource window, int y, int x, string text) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_napms (PHP 4 >= 4.1.0)

Sleep

int **ncurses_napms** (int milliseconds) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_newwin (PHP 4 >= 4.1.0)

Create a new window

int **ncurses_newwin** (int rows, int cols, int y, int x) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_nl (PHP 4 >= 4.1.0)

Translate newline and carriage return / line feed

bool **ncurses_nl** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_nocbreak (PHP 4 >= 4.1.0)

Switch terminal to cooked mode

bool **ncurses_nocbreak** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_nocbreak() routine returns terminal to normal (cooked) mode. Initially the terminal may or may not in cbreak mode as the mode is inherited. Therefore a program should call ncurses_cbreak() and **ncurses_nocbreak()** explicitly. Returns TRUE if any error occured, otherwise FALSE.

See also: ncurses_cbreak()

ncurses_noecho (PHP 4 >= 4.1.0)

Switch off keyboard input echo

bool **ncurses_noecho** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_noecho() prevents echoing of user typed characters. Returns TRUE if any error occured, otherwise FALSE.

See also: ncurses_echo(), ncurses_getch()

ncurses_nonl (PHP 4 >= 4.1.0)

Do not translate newline and carriage return / line feed

bool **ncurses_nonl** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_noqiflush (PHP 4 >= 4.1.0)

Do not flush on signal characters

int **ncurses_noqiflush** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_noraw (PHP 4 >= 4.1.0)

Switch terminal out of raw mode

bool **ncurses_noraw** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_noraw() switches the terminal out of raw mode. Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences that are that in raw mode, the interrupt, quit, suspend and flow control characters are all passed through uninterpreted, instead of generating a signal. Returns TRUE if any error occured, otherwise FALSE.

See also: ncurses_raw(), ncurses_cbreak(), ncurses_nocbreak()

ncurses_putp (PHP 4 >= 4.2.0)

int **ncurses_putp** (string text) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_qiflush (PHP 4 >= 4.1.0)

Flush on signal characters

int **ncurses_qiflush** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_raw (PHP 4 >= 4.1.0)

Switch terminal into raw mode

bool **ncurses_raw** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_raw() places the terminal in raw mode. Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences that are that in raw mode, the interrupt, quit, suspend and flow control characters are all passed through uninterpreted, instead of generating a signal. Returns TRUE if any error occured, otherwise FALSE.

See also: ncurses_noraw(), ncurses_cbreak(), ncurses_nocbreak()

ncurses_refresh (PHP 4 >= 4.1.0)

Refresh screen

int **ncurses_refresh** (int ch) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_resetty (PHP 4 >= 4.1.0)

Restores saved terminal state

bool **ncurses_resetty** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Function **ncurses_resetty()** restores the terminal state, which was previously saved by calling ncurses_savetty(). This function always returns FALSE.

See also: ncurses_savetty()

ncurses_savetty (PHP 4 >= 4.1.0)

Saves terminal state

bool **ncurses_savetty** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Function **ncurses_savetty()** saves the current terminal state. The saved terminal state can be restored with function ncurses_resetty(). **ncurses_savetty()** always returns FALSE.

See also: ncurses_resetty()

ncurses_scr_dump (PHP 4 >= 4.2.0)

Dump screen content to file

int **ncurses_scr_dump** (string filename) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_scr_init (PHP 4 >= 4.2.0)

Initialize screen from file dump

int **ncurses_scr_init** (string filename) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_scr_restore (PHP 4 >= 4.2.0)

Restore screen from file dump

int **ncurses_scr_restore** (string filename) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_scr_set (PHP 4 >= 4.2.0)

Inherit screen from file dump

int **ncurses_scr_set** (string filename) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_scrl (PHP 4 >= 4.1.0)

Scroll window content up or down without changing current position

int **ncurses_scrl** (int count) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_slk_attr (PHP 4 >= 4.1.0)

Returns current soft label key attribute

bool **ncurses_slk_attr** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_slk_attr() returns the current soft label key attribute. On error returns TRUE, otherwise FALSE.

ncurses_slk_attroff (PHP 4 >= 4.1.0)

int **ncurses_slk_attroff** (int intarg) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_slk_attron (PHP 4 >= 4.1.0)

int **ncurses_slk_attron** (int intarg) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_slk_attrset (PHP 4 >= 4.1.0)

int **ncurses_slk_attrset** (int intarg) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_slk_clear (PHP 4 >= 4.1.0)

Clears soft labels from screen

bool **ncurses_slk_clear** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

The function **ncurses_slk_clear()** clears soft label keys from screen. Returns `TRUE` on error, otherwise `FALSE`.

ncurses_slk_color (PHP 4 >= 4.1.0)

Sets color for soft label keys

int **ncurses_slk_color** (int intarg) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_slk_init (PHP 4 >= 4.1.0)

Initializes soft label key functions

bool **ncurses_slk_init** (int format) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Funtion **ncurses_slk_init()** must be called before **ncurses_initscr()** or **ncurses_newterm()** is called. If **ncurses_initscr()** eventually uses a line from stdscr to emulate the soft labels, then *format* determines how the labels are arranged of the screen. Setting *format* to 0 indicates a 3-2-3 arrangement of the labels, 1 indicates a 4-4 arrangement and 2 indicates the PC like 4-4-4 mode, but in addition an index line will be created.

ncurses_slk_noutrefresh (PHP 4 >= 4.1.0)

Copies soft label keys to virtual screen

bool **ncurses_slk_noutrefresh** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_slk_refresh (PHP 4 >= 4.1.0)

Copies soft label keys to screen

bool **ncurses_slk_refresh** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_slk_refresh() copies soft label keys from virtual screen to physical screen. Returns TRUE on error, otherwise FALSE.

ncurses_slk_restore (PHP 4 >= 4.1.0)

Restores soft label keys

bool **ncurses_slk_restore** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

The function **ncurses_slk_restore()** restores the soft label keys after ncurses_slk_clear() has been performed.

ncurses_slk_touch (PHP 4 >= 4.1.0)

Fources output when ncurses_slk_noutrefresh is performed

bool **ncurses_slk_touch** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

The **ncurses_slk_touch()** function forces all the soft labels to be output the next time a ncurses_slk_noutrefresh() is performed.

ncurses_standend (PHP 4 >= 4.1.0)

Stop using ’standout’ attribute

int **ncurses_standend** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_standout (PHP 4 >= 4.1.0)

Start using ’standout’ attribute

int **ncurses_standout** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_start_color (PHP 4 >= 4.1.0)

Start using colors

int **ncurses_start_color** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_termattrs (PHP 4 >= 4.1.0)

Returns a logical OR of all attribute flags supported by terminal

bool **ncurses_termattrs** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_termname (PHP 4 >= 4.2.0)

Returns terminals (short)-name

string **ncurses_termname** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_termname() returns terminals shortname. The shortname is truncated to 14 characters. On error **ncurses_termname()** returns NULL.

See also: ncurses_longname()

ncurses_timeout (PHP 4 >= 4.1.0)

Set timeout for special key sequences

void **ncurses_timeout** (int millisec) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_typeahead (PHP 4 >= 4.1.0)

Specify different filedescriptor for typeahead checking

int **ncurses_typeahead** (int fd) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_ungetch (PHP 4 >= 4.1.0)

Put a character back into the input stream

int **ncurses_ungetch** (int keycode) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_ungetmouse (PHP 4 >= 4.2.0)

Pushes mouse event to queue

bool **ncurses_ungetmouse** (array mevent) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

ncurses_getmouse() pushes a KEY_MOUSE event onto the unput queue and associates with this event the given state sata and screen-relative character cell coordinates, specified in `mevent`. Event options will be specified in associative array `mevent`:

- "id" : Id to distinguish multiple devices
- "x" : screen relative x-position in character cells
- "y" : screen relative y-position in character cells
- "z" : currently not supported
- "mmask" : Mouse action

ncurses_ungetmouse() returns FALSE on success, otherwise TRUE.

See also: ncurses_getmouse()

ncurses_use_default_colors (PHP 4 >= 4.1.0)

Assign terminal default colors to color id -1

bool **ncurses_use_default_colors** (void) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_use_env (PHP 4 >= 4.1.0)

Control use of environment information about terminal size

void **ncurses_use_env** (bool flag) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_use_extended_names (PHP 4 >= 4.1.0)

Control use of extended names in terminfo descriptions

int **ncurses_use_extended_names** (bool flag) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_vidattr (PHP 4 >= 4.1.0)

int **ncurses_vidattr** (int intarg) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_vline (PHP 4 >= 4.2.0)

Draw a vertical line at current position using an attributed character and max. n characters long

int **ncurses_vline** (int charattr, int n) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

ncurses_wrefresh (PHP 4 >= 4.2.0)

Refresh window on terminal screen

int **ncurses_wrefresh** (resource window) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

undocumented

LXVIII. Lotus Notes functions

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

notes_body (PHP 4 >= 4.0.5)

Open the message msg_number in the specified mailbox on the specified server (leave serv

array **notes_body** (string server, string mailbox, int msg_number) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_copy_db (PHP 4 >= 4.0.5)

Create a note using form form_name

string **notes_copy_db** (string from_database_name, string to_database_name) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_create_db (PHP 4 >= 4.0.5)

Create a Lotus Notes database

bool **notes_create_db** (string database_name) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_create_note (PHP 4 >= 4.0.5)

Create a note using form form_name

string **notes_create_note** (string database_name, string form_name) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_drop_db (PHP 4 >= 4.0.5)

Drop a Lotus Notes database

bool **notes_drop_db** (string database_name) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_find_note (PHP 4 >= 4.0.5)

Returns a note id found in database_name. Specify the name of the note. Leaving type bla

bool **notes_find_note** (string database_name, string name [, string type]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_header_info (PHP 4 >= 4.0.5)

Open the message msg_number in the specified mailbox on the specified server (leave serv

object **notes_header_info** (string server, string mailbox, int msg_number) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_list_msgs (PHP 4 >= 4.0.5)

Returns the notes from a selected database_name

bool **notes_list_msgs** (string db) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_mark_read (PHP 4 >= 4.0.5)

Mark a note_id as read for the User user_name

string **notes_mark_read** (string database_name, string user_name, string note_id) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_mark_unread (PHP 4 >= 4.0.5)

Mark a note_id as unread for the User user_name

string **notes_mark_unread** (string database_name, string user_name, string note_id) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_nav_create (PHP 4 >= 4.0.5)

Create a navigator name, in database_name

bool **notes_nav_create** (string database_name, string name) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_search (PHP 4 >= 4.0.5)

Find notes that match keywords in database_name

string **notes_search** (string database_name, string keywords) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_unread (PHP 4 >= 4.0.5)

Returns the unread note id’s for the current User user_name

string **notes_unread** (string database_name, string user_name) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

notes_version (PHP 4 >= 4.0.5)

Get the version Lotus Notes

string **notes_version** (string database_name) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

LXIX. Unified ODBC functions

In addition to normal ODBC support, the Unified ODBC functions in PHP allow you to access several databases that have borrowed the semantics of the ODBC API to implement their own API. Instead of maintaining multiple database drivers that were all nearly identical, these drivers have been unified into a single set of ODBC functions.

> **Megjegyzés:** There is no ODBC involved when connecting to the above databases. The functions that you use to speak natively to them just happen to share the same names and syntax as the ODBC functions. The exception to this is iODBC. Building PHP with iODBC support enables you to use any ODBC-compliant drivers with your PHP applications. iODBC is maintained by OpenLink Software (http://www.openlinksw.com/). More information on iODBC, as well as a HOWTO, is available at www.iodbc.org (http://www.iodbc.org/).

Requirements

The following databases are supported by the Unified ODBC functions: Adabas D (http://www.software-ag.com/adabasd/), IBM DB2 (http://www.ibm.com/db2/), iODBC (http://www.iodbc.org/), Solid (http://www.solidtech.com/), and Sybase SQL Anywhere (http://www.sybase.com/). To access these databases you need to have the required libraries installed.

Installation

Please see the Installation on Unix Systems chapter for more information about configuring PHP with these databases.

Runtime Configuration

The behaviour of the ODBC functions is affected by settings in the global configuration file `php.ini`.

Táblázat 1. Unified ODBC Configuration Options

Name	Default	Changeable
odbc.default_db *	NULL	PHP_INI_ALL
odbc.default_user *	NULL	PHP_INI_ALL
odbc.default_pw *	NULL	PHP_INI_ALL
odbc.allow_persistent	"1"	PHP_INI_SYSTEM
odbc.check_persistent	"1"	PHP_INI_SYSTEM
odbc.max_persistent	"-1"	PHP_INI_SYSTEM
odbc.max_links	"-1"	PHP_INI_SYSTEM
odbc.defaultlrl	"4096"	PHP_INI_ALL
odbc.defaultbinmode	"1"	PHP_INI_ALL

Megjegyzés: Entries marked with * are not implemented yet.

For further details and definition of the PHP_INI_* constants see ini_set().

Here is a short explanation of the configuration directives.

odbc.default_db string

ODBC data source to use if none is specified in odbc_connect() or odbc_pconnect().

odbc.default_user string

User name to use if none is specified in odbc_connect() or odbc_pconnect().

odbc.default_pw string

Password to use if none is specified in odbc_connect() or odbc_pconnect().

odbc.allow_persistent boolean

Whether to allow persistent ODBC connections.

odbc.check_persistent boolean

Check that a connection is still valid before reuse.

odbc.max_persistent integer

The maximum number of persistent ODBC connections per process.

odbc.max_links integer

The maximum number of ODBC connections per process, including persistent connections.

odbc.defaultlrl integer

Handling of LONG fields. Specifies the number of bytes returned to variables.

odbc.defaultbinmode integer

Handling of binary data.

Resource types

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Predefined constants

Ez a kiterjesztés semmilyen konstans értéket nem definiál.

odbc_autocommit (PHP 3>= 3.0.6, PHP 4)

Toggle autocommit behaviour

bool **odbc_autocommit** (resource connection_id [, bool OnOff]) \linebreak

Without the *OnOff* parameter, this function returns auto-commit status for *connection_id*. TRUE is returned if auto-commit is on, FALSE if it is off or an error occurs.

If *OnOff* is TRUE, auto-commit is enabled, if it is FALSE auto-commit is disabled. Returns TRUE on success, FALSE on failure.

By default, auto-commit is on for a connection. Disabling auto-commit is equivalent with starting a transaction.

See also odbc_commit() and odbc_rollback().

odbc_binmode (PHP 3>= 3.0.6, PHP 4)

Handling of binary column data

int **odbc_binmode** (resource result_id, int mode) \linebreak

(ODBC SQL types affected: BINARY, VARBINARY, LONGVARBINARY)

- ODBC_BINMODE_PASSTHRU: Passthru BINARY data
- ODBC_BINMODE_RETURN: Return as is
- ODBC_BINMODE_CONVERT: Convert to char and return

When binary SQL data is converted to character C data, each byte (8 bits) of source data is represented as two ASCII characters. These characters are the ASCII character representation of the number in its hexadecimal form. For example, a binary 00000001 is converted to "01" and a binary 11111111 is converted to "FF".

Táblázat 1. LONGVARBINARY handling

binmode	longreadlen	result
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_RETURN	0	passthru
ODBC_BINMODE_CONVERT	0	passthru
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_PASSTHRU	>0	passthru
ODBC_BINMODE_RETURN	>0	return as is
ODBC_BINMODE_CONVERT	>0	return as char

If odbc_fetch_into() is used, passthru means that an empty string is returned for these columns.

If *result_id* is 0, the settings apply as default for new results.

Megjegyzés: Default for longreadlen is 4096 and binmode defaults to ODBC_BINMODE_RETURN. Handling of binary long columns is also affected by odbc_longreadlen()

odbc_close_all (PHP 3>= 3.0.6, PHP 4)

Close all ODBC connections

void **odbc_close_all** (void) \linebreak

odbc_close_all() will close down all connections to database server(s).

Megjegyzés: This function will fail if there are open transactions on a connection. This connection will remain open in this case.

odbc_close (PHP 3>= 3.0.6, PHP 4)

Close an ODBC connection

void **odbc_close** (resource connection_id) \linebreak

odbc_close() will close down the connection to the database server associated with the given connection identifier.

Megjegyzés: This function will fail if there are open transactions on this connection. The connection will remain open in this case.

odbc_columnprivileges (PHP 4)

Returns a result identifier that can be used to fetch a list of columns and associated privileges

int **odbc_columnprivileges** (resource connection_id [, string qualifier [, string owner [, string table_name [, string column_name]]]]) \linebreak

Lists columns and associated privileges for the given table. Returns an ODBC result identifier or FALSE on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *column_name* argument accepts search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_columns (PHP 4)

Lists the column names in specified tables. Returns a result identifier containing the information.

int **odbc_columns** (resource connection_id [, string qualifier [, string owner [, string table_name [, string column_name]]]]) \linebreak

Lists all columns in the requested range. Returns an ODBC result identifier or FALSE on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *owner*, *table_name* and *column_name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

See also odbc_columnprivileges() to retrieve associated privileges.

odbc_commit (PHP 3>= 3.0.6, PHP 4)

Commit an ODBC transaction

bool **odbc_commit** (resource connection_id) \linebreak

Returns: TRUE on success, FALSE on failure. All pending transactions on *`connection_id`* are committed.

odbc_connect (PHP 3>= 3.0.6, PHP 4)

Connect to a datasource

resource **odbc_connect** (string dsn, string user, string password [, int cursor_type]) \linebreak

Returns an ODBC connection id or 0 (FALSE) on error.

The connection id returned by this functions is needed by other ODBC functions. You can have multiple connections open at once. The optional fourth parameter sets the type of cursor to be used for this connection. This parameter is not normally needed, but can be useful for working around problems with some ODBC drivers.

With some ODBC drivers, executing a complex stored procedure may fail with an error similar to: "Cannot open a cursor on a stored procedure that has anything other than a single select statement in it". Using SQL_CUR_USE_ODBC may avoid that error. Also, some drivers don’t support the optional row_number parameter in odbc_fetch_row(). SQL_CUR_USE_ODBC might help in that case, too.

The following constants are defined for cursortype:

- SQL_CUR_USE_IF_NEEDED
- SQL_CUR_USE_ODBC
- SQL_CUR_USE_DRIVER
- SQL_CUR_DEFAULT

For persistent connections see odbc_pconnect().

odbc_cursor (PHP 3>= 3.0.6, PHP 4)

Get cursorname

string **odbc_cursor** (resource result_id) \linebreak

odbc_cursor will return a cursorname for the given result_id.

odbc_do (PHP 3>= 3.0.6, PHP 4)

Synonym for odbc_exec()

resource **odbc_do** (resource conn_id, string query) \linebreak

odbc_do() will execute a query on the given connection.

odbc_error (PHP 4 >= 4.0.5)

Get the last error code

string **odbc_error** ([resource connection_id]) \linebreak

Returns a six-digit ODBC state, or an empty string if there has been no errors. If *`connection_id`* is specified, the last state of that connection is returned, else the last state of any connection is returned.

See also: odbc_errormsg() and odbc_exec().

odbc_errormsg (PHP 4 >= 4.0.5)

Get the last error message

string **odbc_errormsg** ([resource connection_id]) \linebreak

Returns a string containing the last ODBC error message, or an empty string if there has been no errors. If *`connection_id`* is specified, the last state of that connection is returned, else the last state of any connection is returned.

See also: odbc_error() and odbc_exec().

odbc_exec (PHP 3>= 3.0.6, PHP 4)

Prepare and execute a SQL statement

resource **odbc_exec** (resource connection_id, string query_string) \linebreak

Returns FALSE on error. Returns an ODBC result identifier if the SQL command was executed successfully.

odbc_exec() will send an SQL statement to the database server specified by *`connection_id`*. This parameter must be a valid identifier returned by odbc_connect() or odbc_pconnect().

See also: odbc_prepare() and odbc_execute() for multiple execution of SQL statements.

odbc_execute (PHP 3>= 3.0.6, PHP 4)

Execute a prepared statement

resource **odbc_execute** (resource result_id [, array parameters_array]) \linebreak

Executes a statement prepared with odbc_prepare(). Returns TRUE on successful execution; FALSE otherwise. The array *parameters_array* only needs to be given if you really have parameters in your statement.

Parameters in *parameter_array* will be substituted for placeholders in the prepared statement in order.

Any parameters in *parameter_array* which start and end with single quotes will be taken as the name of a file to read and send to the database server as the data for the appropriate placeholder.

Megjegyzés: As of PHP 4.1.1, this file reading functionality has the following restrictions:

- File reading is *not* subject to any safe mode or safe mode restrictions. This is fixed in PHP 4.2.0.
- Remote files are not supported.
- If you wish to store a string which actually begins and ends with single quotes, you must escape them or add a space or other non-single-quote character to the beginning or end of the parameter, which will prevent the parameter's being taken as a file name. If this is not an option, then you must use another mechanism to store the string, such as executing the query directly with odbc_exec()).

odbc_fetch_array (PHP 4 >= 4.0.2)

Fetch a result row as an associative array

array **odbc_fetch_array** (resource result [, int rownumber]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

odbc_fetch_into (PHP 3>= 3.0.6, PHP 4)

Fetch one result row into array

bool **odbc_fetch_into** (resource result_id [, int rownumber, array result_array]) \linebreak resource **odbc_fetch_into** (resource result_id, array result_array [, int rownumber]) \linebreak

Returns the number of columns in the result; FALSE on error. *result_array* must be passed by reference, but it can be of any type since it will be converted to type array. The array will contain the column values starting at array index 0.

As of PHP 4.0.5 the *result_array* does not need to be passed by reference any longer.

As of PHP 4.0.6 the *rownumber* cannot be passed as a constant, but rather as a variable.

As of PHP 4.2.0 the *result_array* and *rownumber* have been swapped. This allows the rownumber to be a constant again. This change will also be the last one to this function.

Példa 1. odbc_fetch_into() pre 4.0.6 example

```
$rc = odbc_fetch_into($res_id, $my_array);
```

or

```
$rc = odbc_fetch_into($res_id, $row, $my_array);

$rc = odbc_fetch_into($res_id, 1, $my_array);
```

Példa 2. odbc_fetch_into() 4.0.6 example

```
$rc = odbc_fetch_into($res_id, $my_array);
```

or

```
$row = 1;
$rc = odbc_fetch_into($res_id, $row, $my_array);
```

Példa 3. odbc_fetch_into() 4.2.0 example

```
$rc = odbc_fetch_into($res_id, $my_array);
```

or

```
$rc = odbc_fetch_into($res_id,$my_array, 2);
```

odbc_fetch_object (PHP 4 >= 4.0.2)

Fetch a result row as an object

object **odbc_fetch_object** (resource result [, int rownumber]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

odbc_fetch_row (PHP 3>= 3.0.6, PHP 4)

Fetch a row

bool **odbc_fetch_row** (resource result_id [, int row_number]) \linebreak

If **odbc_fetch_row()** was succesful (there was a row), TRUE is returned. If there are no more rows, FALSE is returned.

odbc_fetch_row() fetches a row of the data that was returned by odbc_do() / odbc_exec(). After **odbc_fetch_row()** is called, the fields of that row can be accessed with odbc_result().

If *row_number* is not specified, **odbc_fetch_row()** will try to fetch the next row in the result set. Calls to **odbc_fetch_row()** with and without *row_number* can be mixed.

To step through the result more than once, you can call **odbc_fetch_row()** with *row_number* 1, and then continue doing **odbc_fetch_row()** without *row_number* to review the result. If a driver doesn't support fetching rows by number, the *row_number* parameter is ignored.

odbc_field_len (PHP 3>= 3.0.6, PHP 4)

Get the length (precision) of a field

int **odbc_field_len** (resource result_id, int field_number) \linebreak

odbc_field_len() will return the length of the field referecend by number in the given ODBC result identifier. Field numbering starts at 1.

See also: odbc_field_scale() to get the scale of a floating point number.

odbc_field_name (PHP 3>= 3.0.6, PHP 4)

Get the columnname

string **odbc_field_name** (resource result_id, int field_number) \linebreak

odbc_field_name() will return the name of the field occupying the given column number in the given ODBC result identifier. Field numbering starts at 1. FALSE is returned on error.

odbc_field_num (PHP 3>= 3.0.6, PHP 4)

Return column number

int **odbc_field_num** (resource result_id, string field_name) \linebreak

odbc_field_num() will return the number of the column slot that corresponds to the named field in the given ODBC result identifier. Field numbering starts at 1. FALSE is returned on error.

odbc_field_precision (PHP 4)

Synonym for odbc_field_len()

string **odbc_field_precision** (resource result_id, int field_number) \linebreak

odbc_field_precision() will return the precision of the field referecend by number in the given ODBC result identifier.

See also: odbc_field_scale() to get the scale of a floating point number.

odbc_field_scale (PHP 4)

Get the scale of a field

string **odbc_field_scale** (resource result_id, int field_number) \linebreak

odbc_field_precision() will return the scale of the field referecend by number in the given ODBC result identifier.

odbc_field_type (PHP 3>= 3.0.6, PHP 4)

Datatype of a field

string **odbc_field_type** (resource result_id, int field_number) \linebreak

odbc_field_type() will return the SQL type of the field referecend by number in the given ODBC result identifier. Field numbering starts at 1.

odbc_foreignkeys (PHP 4)

Returns a list of foreign keys in the specified table or a list of foreign keys in other tables that refer to the primary key in the specified table

resource **odbc_foreignkeys** (resource connection_id, string pk_qualifier, string pk_owner, string pk_table, string fk_qualifier, string fk_owner, string fk_table) \linebreak

odbc_foreignkeys() retrieves information about foreign keys. Returns an ODBC result identifier or FALSE on failure.

The result set has the following columns:

- PKTABLE_QUALIFIER
- PKTABLE_OWNER
- PKTABLE_NAME
- PKCOLUMN_NAME
- FKTABLE_QUALIFIER
- FKTABLE_OWNER
- FKTABLE_NAME
- FKCOLUMN_NAME
- KEY_SEQ
- UPDATE_RULE
- DELETE_RULE
- FK_NAME
- PK_NAME

If *pk_table* contains a table name, **odbc_foreignkeys()** returns a result set containing the primary key of the specified table and all of the foreign keys that refer to it.

If *fk_table* contains a table name, **odbc_foreignkeys()** returns a result set containing all of the foreign keys in the specified table and the primary keys (in other tables) to which they refer.

If both *pk_table* and *fk_table* contain table names, **odbc_foreignkeys()** returns the foreign keys in the table specified in *fk_table* that refer to the primary key of the table specified in *pk_table*. This should be one key at most.

odbc_free_result (PHP 3>= 3.0.6, PHP 4)

Free resources associated with a result

bool **odbc_free_result** (resource result_id) \linebreak

Always returns TRUE.

odbc_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But,

if you are sure you are not going to need the result data anymore in a script, you may call **odbc_free_result()**, and the memory associated with *result_id* will be freed.

> **Megjegyzés:** If auto-commit is disabled (see odbc_autocommit()) and you call **odbc_free_result()** before commiting, all pending transactions are rolled back.

odbc_gettypeinfo (PHP 4)

Returns a result identifier containing information about data types supported by the data source.

int **odbc_gettypeinfo** (resource connection_id [, int data_type]) \linebreak

Retrieves information about data types supported by the data source. Returns an ODBC result identifier or FALSE on failure. The optional argument *data_type* can be used to restrict the information to a single data type.

The result set has the following columns:

- TYPE_NAME
- DATA_TYPE
- PRECISION
- LITERAL_PREFIX
- LITERAL_SUFFIX
- CREATE_PARAMS
- NULLABLE
- CASE_SENSITIVE
- SEARCHABLE
- UNSIGNED_ATTRIBUTE
- MONEY
- AUTO_INCREMENT
- LOCAL_TYPE_NAME
- MINIMUM_SCALE
- MAXIMUM_SCALE

The result set is ordered by DATA_TYPE and TYPE_NAME.

odbc_longreadlen (PHP 3>= 3.0.6, PHP 4)

Handling of LONG columns

int **odbc_longreadlen** (resource result_id, int length) \linebreak

(ODBC SQL types affected: LONG, LONGVARBINARY) The number of bytes returned to PHP is controled by the parameter length. If it is set to 0, Long column data is passed thru to the client.

Megjegyzés: Handling of LONGVARBINARY columns is also affected by odbc_binmode().

odbc_next_result (PHP 4 >= 4.0.5)

Checks if multiple results are avaiable

bool **odbc_next_result** (resource result_id) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

odbc_num_fields (PHP 3>= 3.0.6, PHP 4)

Number of columns in a result

int **odbc_num_fields** (resource result_id) \linebreak

odbc_num_fields() will return the number of fields (columns) in an ODBC result. This function will return -1 on error. The argument is a valid result identifier returned by odbc_exec().

odbc_num_rows (PHP 3>= 3.0.6, PHP 4)

Number of rows in a result

int **odbc_num_rows** (resource result_id) \linebreak

odbc_num_rows() will return the number of rows in an ODBC result. This function will return -1 on error. For INSERT, UPDATE and DELETE statements **odbc_num_rows()** returns the number of rows affected. For a SELECT clause this can be the number of rows available.

Note: Using **odbc_num_rows()** to determine the number of rows available after a SELECT will return -1 with many drivers.

odbc_pconnect (PHP 3>= 3.0.6, PHP 4)

Open a persistent database connection

int **odbc_pconnect** (string dsn, string user, string password [, int cursor_type]) \linebreak

Returns an ODBC connection id or 0 (FALSE) on error. This function is much like odbc_connect(), except that the connection is not really closed when the script has finished. Future requests for a connection with the same *dsn*, *user*, *password* combination (via odbc_connect() and **odbc_pconnect()**) can reuse the persistent connection.

Megjegyzés: Persistent connections have no effect if PHP is used as a CGI program.

For information about the optional cursor_type parameter see the odbc_connect() function. For more information on persistent connections, refer to the PHP FAQ.

odbc_prepare (PHP 3>= 3.0.6, PHP 4)

Prepares a statement for execution

resource **odbc_prepare** (resource connection_id, string query_string) \linebreak

Returns FALSE on error.

Returns an ODBC result identifier if the SQL command was prepared successfully. The result identifier can be used later to execute the statement with odbc_execute().

odbc_primarykeys (PHP 4)

Returns a result identifier that can be used to fetch the column names that comprise the primary key for a table

resource **odbc_primarykeys** (resource connection_id, string qualifier, string owner, string table) \linebreak

Returns the column names that comprise the primary key for a table. Returns an ODBC result identifier or FALSE on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- KEY_SEQ
- PK_NAME

odbc_procedurecolumns (PHP 4)

Retrieve information about parameters to procedures

resource **odbc_procedurecolumns** (resource connection_id [, string qualifier [, string owner [, string proc [, string column]]]]) \linebreak

Returns the list of input and output parameters, as well as the columns that make up the result set for the specified procedures. Returns an ODBC result identifier or FALSE on failure.

The result set has the following columns:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- COLUMN_NAME
- COLUMN_TYPE
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

The result set is ordered by PROCEDURE_QUALIFIER, PROCEDURE_OWNER, PROCEDURE_NAME and COLUMN_TYPE.

The *owner*, *proc* and *column* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_procedures (PHP 4)

Get the list of procedures stored in a specific data source. Returns a result identifier containing the information.

resource **odbc_procedures** (resource connection_id [, string qualifier [, string owner [, string name]]]) \linebreak

Lists all procedures in the requested range. Returns an ODBC result identifier or FALSE on failure.

The result set has the following columns:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- NUM_INPUT_PARAMS
- NUM_OUTPUT_PARAMS
- NUM_RESULT_SETS
- REMARKS
- PROCEDURE_TYPE

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_result_all (PHP 3>= 3.0.6, PHP 4)

Print result as HTML table

int **odbc_result_all** (resource result_id [, string format]) \linebreak

Returns the number of rows in the result or FALSE on error.

odbc_result_all() will print all rows from a result identifier produced by odbc_exec(). The result is printed in HTML table format. With the optional string argument *format*, additional overall table formatting can be done.

odbc_result (PHP 3>= 3.0.6, PHP 4)

Get result data

string **odbc_result** (resource result_id, mixed field) \linebreak

Returns the contents of the field.

field can either be an integer containing the column number of the field you want; or it can be a string containing the name of the field. For example:

```
$item_3 = odbc_result ($Query_ID, 3);
$item_val = odbc_result ($Query_ID, "val");
```

The first call to **odbc_result()** returns the value of the third field in the current record of the query result. The second function call to **odbc_result()** returns the value of the field whose field name is "val" in the current record of the query result. An error occurs if a column number parameter for a field is less than one or exceeds the number of columns (or fields) in the current record. Similarly, an

error occurs if a field with a name that is not one of the fieldnames of the table(s) that is(are) being queried.

Field indices start from 1. Regarding the way binary or long column data is returned refer to odbc_binmode() and odbc_longreadlen().

odbc_rollback (PHP 3>= 3.0.6, PHP 4)

Rollback a transaction

int **odbc_rollback** (resource connection_id) \linebreak

Rolls back all pending statements on *`connection_id`*. Returns `TRUE` on success, `FALSE` on failure.

odbc_setoption (PHP 3>= 3.0.6, PHP 4)

Adjust ODBC settings. Returns `FALSE` if an error occurs, otherwise `TRUE`.

int **odbc_setoption** (resource id, int function, int option, int param) \linebreak

This function allows fiddling with the ODBC options for a particular connection or query result. It was written to help find work arounds to problems in quirky ODBC drivers. You should probably only use this function if you are an ODBC programmer and understand the effects the various options will have. You will certainly need a good ODBC reference to explain all the different options and values that can be used. Different driver versions support different options.

Because the effects may vary depending on the ODBC driver, use of this function in scripts to be made publicly available is strongly discouraged. Also, some ODBC options are not available to this function because they must be set before the connection is established or the query is prepared. However, if on a particular job it can make PHP work so your boss doesn't tell you to use a commercial product, that's all that really matters.

`id` is a connection id or result id on which to change the settings.For SQLSetConnectOption(), this is a connection id. For SQLSetStmtOption(), this is a result id.

`Function` is the ODBC function to use. The value should be 1 for SQLSetConnectOption() and 2 for SQLSetStmtOption().

Parameter *`option`* is the option to set.

Parameter *`param`* is the value for the given *`option`*.

Példa 1. ODBC Setoption Examples

```
// 1. Option 102 of SQLSetConnectOption() is SQL_AUTOCOMMIT.
//    Value 1 of SQL_AUTOCOMMIT is SQL_AUTOCOMMIT_ON.
//    This example has the same effect as
//    odbc_autocommit($conn, true);

odbc_setoption ($conn, 1, 102, 1);
```

```
// 2. Option 0 of SQLSetStmtOption() is SQL_QUERY_TIMEOUT.
//    This example sets the query to timeout after 30 seconds.

$result = odbc_prepare ($conn, $sql);
odbc_setoption ($result, 2, 0, 30);
odbc_execute ($result);
```

odbc_specialcolumns (PHP 4)

Returns either the optimal set of columns that uniquely identifies a row in the table or columns that are automatically updated when any value in the row is updated by a transaction

resource **odbc_specialcolumns** (resource connection_id, int type, string qualifier, string owner, string table, int scope, int nullable) \linebreak

When the type argument is SQL_BEST_ROWID, **odbc_specialcolumns()** returns the column or columns that uniquely identify each row in the table.

When the type argument is SQL_ROWVER, **odbc_specialcolumns()** returns the optimal column or set of columns that, by retrieving values from the column or columns, allows any row in the specified table to be uniquely identified.

Returns an ODBC result identifier or `FALSE` on failure.

The result set has the following columns:

- SCOPE
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- PSEUDO_COLUMN

The result set is ordered by SCOPE.

odbc_statistics (PHP 4)

Retrieve statistics about a table

resource **odbc_statistics** (resource connection_id, string qualifier, string owner, string table_name, int unique, int accuracy) \linebreak

Get statistics about a table and it's indexes. Returns an ODBC result identifier or FALSE on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- NON_UNIQUE
- INDEX_QUALIFIER
- INDEX_NAME
- TYPE
- SEQ_IN_INDEX
- COLUMN_NAME
- COLLATION
- CARDINALITY
- PAGES
- FILTER_CONDITION

The result set is ordered by NON_UNIQUE, TYPE, INDEX_QUALIFIER, INDEX_NAME and SEQ_IN_INDEX.

odbc_tableprivileges (PHP 4)

Lists tables and the privileges associated with each table

int **odbc_tableprivileges** (resource connection_id [, string qualifier [, string owner [, string name]]]) \line-break

Lists tables in the requested range and the privileges associated with each table. Returns an ODBC result identifier or FALSE on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_tables (PHP 3>= 3.0.17, PHP 4)

Get the list of table names stored in a specific data source. Returns a result identifier containing the information.

int **odbc_tables** (resource connection_id [, string qualifier [, string owner [, string name [, string types]]]]) \linebreak

Lists all tables in the requested range. Returns an ODBC result identifier or FALSE on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- TABLE_TYPE
- REMARKS

The result set is ordered by TABLE_TYPE, TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

To support enumeration of qualifiers, owners, and table types, the following special semantics for the *qualifier*, *owner*, *name*, and *table_type* are available:

- If *qualifier* is a single percent character (%) and *owner* and *name* are empty strings, then the result set contains a list of valid qualifiers for the data source. (All columns except the TABLE_QUALIFIER column contain NULLs.)
- If *owner* is a single percent character (%) and *qualifier* and *name* are empty strings, then the result set contains a list of valid owners for the data source. (All columns except the TABLE_OWNER column contain NULLs.)
- If *table_type* is a single percent character (%) and *qualifier*, *owner* and *name* are empty strings, then the result set contains a list of valid table types for the data source. (All columns except the TABLE_TYPE column contain NULLs.)

If *table_type* is not an empty string, it must contain a list of comma-separated values for the types of interest; each value may be enclosed in single quotes (') or unquoted. For example, "'TABLE','VIEW'" or "TABLE, VIEW". If the data source does not support a specified table type, **odbc_tables()** does not return any results for that type.

See also odbc_tableprivileges() to retrieve associated privileges.

LXX. Oracle 8 függvények

Ezekkel a függvényekkel az Oracle8 és Oracle7 adatbázisokat érheted el, az Oracle8 Call-Interface (OCI8) használatával. Az Oracle8 kliens könyvtárakra szükséged lesz, ha szeretnéd ezt a kiterjesztést használni.

Ez a kiterjesztés rugalmasabb, mint a hagyományos Oracle kiterjesztés. Lehetővé teszi globális és lokális PHP változók kötését Oracle értékekhez, teljes LOB, FILE és ROWID támogatással rendelkezik, és lehetővé teszi a felhasználó által megadott változók használatát.

Mielőtt elkezdenéd használni ezt a kiterjesztést, győződj meg róla, hogy az Oracle környezeti változók helyesen be vannak állítva az Oracle és a webszerver felhasználó számára. Az értékek, amiket esetleg be kell állítanod a kövezkezők:

- ORACLE_HOME
- ORACLE_SID
- LD_PRELOAD
- LD_LIBRARY_PATH
- NLS_LANG
- ORA_NLS33

Miután beállítottad a könyezeti változókat a webszerver felhasználó számára, add hozzá a webszerver felhasználót (nobody, www) az oracle csoporthoz.

Ha a webszerver nem indul el, vagy lefagy induláskor: Ellenőrizd, hogy az Apache szervert a pthread könyvtárral fordítottad-e:

```
# ldd /www/apache/bin/httpd
    libpthread.so.0 => /lib/libpthread.so.0 (0x4001c000)
    libm.so.6 => /lib/libm.so.6 (0x4002f000)
    libcrypt.so.1 => /lib/libcrypt.so.1 (0x4004c000)
    libdl.so.2 => /lib/libdl.so.2 (0x4007a000)
    libc.so.6 => /lib/libc.so.6 (0x4007e000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Ha a libpthread nincs bent a listában, újra kell telepítened az Apache szervert:

```
# cd /usr/src/apache_1.3.xx
# make clean
# LIBS=-lpthread ./config.status
# make
# make install
```

Példa 1. OCI tippek

```
<?php
// A tippeket sergo@bacup.ru küldte be

// Használd az OCI_DEFAULT opciót a végrehajtásnál,
// ha késleltetni szeternéd a végrehjtást
OCIExecute($stmt, OCI_DEFAULT);

// Lekérdezett adatok használatához (fetch után):

$result = OCIResult($stmt, $n);
if (is_object($result)) { $result = $result->load(); }

// INSERT vagy UPDATE parancsokhoz:

$sql = "insert into table (mezo1, mezo2) values (mezo1 = 'érték',
 mezo2 = empty_clob()) returning mezo2 into :mezo2";
OCIParse($conn, $sql);
$clob = OCINewDescriptor($conn, OCI_D_LOB);
OCIBindByName($stmt, ":mezo2", &$clob, -1, OCI_B_CLOB);
OCIExecute($stmt, OCI_DEFAULT);
$clob->save("valami szöveg");
OCICommit($conn);

?>
```

Könnyen elérheted a tárolt eljárásokat, éppen úgy, mint parancssorból.

Példa 2. Tárolt eljárások használata

```
<?php
// webmaster@remoterealty.com küldte be a következő kódot:

$sth = OCIParse ( $dbh, "begin sp_newaddress( :cimid, '$keresztnev',
 '$csaladinev', '$ceg', '$cim1', '$cim2', '$varos', '$megye',
 '$irszam', '$orszag', :hibakod );end;" );

// Ez a parancs meghívja az sp_newaddress nevű tárolt eljárást,
// a :cimid ki és bemeneti változóval és a :hibakod kimeneti
// változóval. Ezután a változó kötéseket kell elvégezni:

   OCIBindByName ( $sth, ":cimid", $cimid, 10 );
   OCIBindByName ( $sth, ":hibakod", $hibakod, 10 );
   OCIExecute ( $sth );
```

```
?>
```

OCIBindByName (PHP 3>= 3.0.4, PHP 4)

PHP változó kötése egy Oracle értékhez

int **OCIBindByName** (int stmt, string ph_name, mixed & variable, int length [, int type]) \linebreak

Az **OCIBindByName()** a *variable* paraméterben megadott PHP változót köti a *ph_name* paraméterben megadott Oracle értékez (placeholder-hez). Akár bemeneti, akár kimeneti értékként kerül felhasználásra, ez futásidőben dől el, és a szükséges tárolási kapacitás lefoglalásra kerül. A *length* paraméter adja meg a kötés maximális méretét. Ha a *length* paramétert -1 értékkel adod meg, az **OCIBindByName()** a *variable* paraméterben megadott változó aktuális hosszát veszi maximum méretnek.

Ha absztrakt adattípust kell kötnöd (például LOB, ROWID vagy BFILE), először az **OCINewDescriptor()** függvényt kell használnod. A *length* paraméternek nincs jelentősége absztrakt adattípusok használatakor, mindenképpen -1 értékre kell állítani ebben az esetben. A *type* paraméter adja meg az Oracle-nek, hogy milyen leírót szeretnél használni. Lehetséges értékek: OCI_B_FILE (Binary-File), OCI_B_CFILE (Character-File), OCI_B_CLOB (Character-LOB), OCI_B_BLOB (Binary-LOB) és OCI_B_ROWID (ROWID).

Példa 1. OCIBindByName

```
<?php
/* OCIBindByName példa - thies@thieso.net (980221)
  Három rekord felvétele a "dolgozok" táblába, és a ROWID
  használata az adatok frissítésére rögtön a felvitel után.
*/

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"insert into dolgozok (dolgozoid, dnev) ".
   "values (:dolgozoid,:dnev) ".
   "returning ROWID into :rid");

$adatok = array(1111 => "Larry", 2222 => "Bill", 3333 => "Jim");

$rowid = OCINewDescriptor($conn, OCI_D_ROWID);

OCIBindByName($stmt,":dolgozoid",&$dolgozoid, 32);
OCIBindByName($stmt,":dnev",     &$dnev,      32);
OCIBindByName($stmt,":rid",      &$rowid,     -1, OCI_B_ROWID);

$update = OCIParse($conn,
              "update dolgozok set fizetes = :fizetes ".
              "where ROWID = :rid");
OCIBindByName($update,":rid",     &$rowid,     -1, OCI_B_ROWID);
OCIBindByName($update,":fizetes", &$fizetes,   32);

$fizetes = 10000;

while (list($dolgozoid,$dnev) = each($adatok)) {
OCIExecute($stmt);
OCIExecute($update);
}
```

```
$rowid->free();

OCIFreeStatement($update);
OCIFreeStatement($stmt);

$stmt = OCIParse($conn,"select * from dolgozok where dolgozoid in (1111,2222,3333)");
OCIExecute($stmt);
while (OCIFetchInto($stmt, &$tomb, OCI_ASSOC)) {
var_dump($tomb);
}
OCIFreeStatement($stmt);

/* Töröljük a "szemetünket" a dolgozok táblából.... */
$stmt = OCIParse($conn,"delete from dolgozok where dolgozoid in (1111,2222,3333)");
OCIExecute($stmt);
OCIFreeStatement($stmt);

OCILogoff($conn);
?>
```

Figyelem

Nem jó ötlet a "magic quotes" szolgáltatást, és az **OciBindByName()** függvényt egyszerre használni, mivel semmilyen átalakítás nem szükséges a változókon, és bármilyen extra idézőjel, ami az értékekben megjelenik az adatbázisba kerül. Az **OciBindByName()** nem tud különbséget tenni a korábban a karaktersorozatban lévő és a "magic quotes" átalakítás során bekerült idézőjelek között.

OCICancel (PHP 3>= 3.0.8, PHP 4)

Kurzorral olvasás befejezése

int **OCICancel** (int stmt) \linebreak

Ha nem szeretnél több adatot olvasni egy kurzorral, hívd meg ezt a függényt.

OCICollAppend (PHP 4 >= 4.0.6)

[Eddig] nem dokumentált

string **OCICollAppend** (object collection, object object) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCICollAssign (PHP 4 >= 4.0.6)

[Eddig] nem dokumentált

string **OCICollAssign** (object collection, object object) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCICollAssignElem (PHP 4 >= 4.0.6)

[Eddig] nem dokumentált

string **OCICollAssignElem** (object collection, string ndx, string val) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCICollGetElem (PHP 4 >= 4.0.6)

[Eddig] nem dokumentált

string **OCICollGetElem** (object collection, string ndx) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCICollMax (PHP 4 >= 4.0.6)

[Eddig] nem dokumentált

string **OCICollMax** (object collection) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCICollSize (PHP 4 >= 4.0.6)

[Eddig] nem dokumentált

string **OCICollSize** (object collection) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCICollTrim (PHP 4 >= 4.0.6)

[Eddig] nem dokumentált

string **OCICollTrim** (object collection, int num) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCIColumnIsNULL (PHP 3>= 3.0.4, PHP 4)

`NULL` értékű mező azonosítása

int **OCIColumnIsNULL** (int stmt, mixed column) \linebreak

Az **OCIColumnIsNULL()** `TRUE` értéket ad vissza, ha a visszaadott *`column`* mező az *`stmt`* eredménytáblájában `NULL` értékű. Használhatod az oszlop sorszámot (1-től számozva), vagy az oszlop nevét a *`col`* paraméterben.

OCIColumnName (PHP 3>= 3.0.4, PHP 4)

Oszlop neve egy eredménytáblában

string **OCIColumnName** (int stmt, int col) \linebreak

Az **OCIColumnName()** visszaadja a *`col`* paraméterben megadott oszlop nevét. A *`col`* egy sorszám, a számozás 1-től kezdődik.

Példa 1. OCIColumnName

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from dolgozok");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Név</TH>";
    print "<TH>Típus</TH>";
    print "<TH>Méret</TH>";
    print "</TR>";
    $oszlopszam = OCINumCols($stmt);
    for ( $i = 1; $i <= $oszlopszam; $i++ ) {
        $oszlopnev   = OCIColumnName($stmt,$i);
        $oszloptipus = OCIColumnType($stmt,$i);
        $oszlopmeret = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$oszlopnev</TD>";
        print "<TD>$oszloptipus</TD>";
        print "<TD>$oszlopmeret</TD>";
        print "</TR>";
    }
    print "</TABLE>\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Lásd még **OCINumCols()**, **OCIColumnType()** és **OCIColumnSize()**.

OCIColumnPrecision (PHP 4)

[Eddig] nem dokumentált

int **OCIColumnPrecision** (int stmt, int col) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCIColumnScale (PHP 4)

[Eddig] nem dokumentált

int **OCIColumnScale** (int stmt, int col) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCIColumnSize (PHP 3>= 3.0.4, PHP 4)

Oszlop mérete egy eredménytáblában

int **OCIColumnSize** (int stmt, mixed column) \linebreak

Az **OCIColumnSize()** az Oracle által visszaadott oszlop mérettel tér vissza. Használhatod az oszlop sorszámot (1-től számozva), vagy az oszlop nevét a *col* paraméterben.

Példa 1. OCIColumnSize

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from dolgozok");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Név</TH>";
    print "<TH>Típus</TH>";
    print "<TH>Méret</TH>";
```

```
    print "</TR>";
    $oszlopszam = OCINumCols($stmt);
    for ( $i = 1; $i <= $oszlopszam; $i++ ) {
        $oszlopnev   = OCIColumnName($stmt,$i);
        $oszloptipus = OCIColumnType($stmt,$i);
        $oszlopmeret = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$oszlopnev</TD>";
        print "<TD>$oszloptipus</TD>";
        print "<TD>$oszlopmeret</TD>";
        print "</TR>";
    }
    print "</TABLE>\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Lásd még **OCINumCols()**, **OCIColumnName()** és **OCIColumnSize()**.

OCIColumnType (PHP 3>= 3.0.4, PHP 4)

Oszlop adattípusa egy eredménytáblában

mixed **OCIColumnType** (int stmt, int col) \linebreak

Az **OCIColumnType()** az adott sorszámú (1-től számozva) oszlop adattípusát adja vissza.

Példa 1. OCIColumnType

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from dolgozok");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Név</TH>";
    print "<TH>Típus</TH>";
    print "<TH>Méret</TH>";
    print "</TR>";
    $oszlopszam = OCINumCols($stmt);
    for ( $i = 1; $i <= $oszlopszam; $i++ ) {
        $oszlopnev   = OCIColumnName($stmt,$i);
        $oszloptipus = OCIColumnType($stmt,$i);
        $oszlopmeret = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$oszlopnev</TD>";
        print "<TD>$oszloptipus</TD>";
```

```
        print "<TD>$oszlopmeret</TD>";
        print "</TR>";
    }
    print "</TABLE>\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Lásd még **OCINumCols()**, **OCIColumnName()** és **OCIColumnSize()**.

OCIColumnTypeRaw (PHP 4)

[Eddig] nem dokumentált

mixed **OCIColumnTypeRaw** (int stmt, int col) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCICommit (PHP 3>= 3.0.7, PHP 4)

Függőben lévő tranzakciók végrehajtása

int **OCICommit** (int connection) \linebreak

Az **OCICommit()** minden a *connection* Oracle kapcsolaton függőben lévő tranzakciót végrehajt.

OCIDefineByName (PHP 3>= 3.0.7, PHP 4)

PHP változó használata a definiálási lépésben egy SELECT számára

int **OCIDefineByName** (int stmt, string Column-Name, mixed variable [, int type]) \linebreak

Az **OCIDefineByName()** SQL oszlopok lekérdezését köti felhasználó által definiált PHP változókhoz. Figyelj arra, hogy az Oracle CSUPA NAGYBETŰS neveket használ, de ennek ellenére írhatsz a SELECT-ben kisbetűs neveket is. Az **OCIDefineByName()** függvény a *Column-Name* paraméterben a helyes nagybetűs írást várja. Ha olyan változót definiálsz, ami nem létezik a SELECT eredménytáblájában, semmilyen hibát sem kapsz!

Ha absztrakt adattípust kell definiálnod (például LOB, ROWID vagy BFILE), először az **OCINewDescriptor()** függvényt kell használnod. Lásd még az **OCIBindByName()** függvényt.

Példa 1. OCIDefineByName

```
<?php
/* OCIDefineByName példa - thies@thieso.net (980219) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select dolgozoid, dnev from dolgozok");

/* ennek az ociexecute ELŐTT KELL szerepelnie! */

OCIDefineByName($stmt, "DOLGOZOID", $dolgozoid);
OCIDefineByName($stmt, "DNEV",      $dnev);

OCIExecute($stmt);

while (OCIFetch($stmt)) {
    echo "Dolgozó azonosító:" . $dolgozoid . "\n";
    echo "Dolgozó név:"       . $dnev      . "\n";
}

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIError (PHP 3>= 3.0.7, PHP 4)

A legutóbbi parancs / kapcsolat / általános hiba visszaadása

array **OCIError** ([int stmt|conn|global]) \linebreak

Az **OCIError()** a legutóbbi hibát adja vissza. Ha az opcionális *stmt|conn|global* paramétert nem adod meg, a legutóbbi hibát kapod. Ha nem volt hiba, FALSE értéket kapsz. Az **OCIError()** egy asszociatív tömbben adja vissza a hibát. Ebben a tömbben a *code* az Oracle hibakód, a *message* az Oracle hibaüzenet karaktersorozat.

OCIExecute (PHP 3>= 3.0.4, PHP 4)

Parancs futtatása

int **OCIExecute** (int statement [, int mode]) \linebreak

Az **OCIExecute()** futtat egy korábban feldolgozott parancsot. (Lásd az **OCIParse()** függvényt.) Az opcionális *mode* paraméterrel tudod beállítani a végrehajtás módját. Az alapértelmezés az

OCI_COMMIT_ON_SUCCESS. Ha nem szeretnéd, hogy a parancs rögtön egy tranzakció végrehajtást is végezzen, add meg az OCI_DEFAULT módot.

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

OCIFetch (PHP 3>= 3.0.4, PHP 4)

A következő eredmény sor lekérdezése

int **OCIFetch** (int statement) \linebreak

Az **OCIFetch()** lekérdezi a következő eredménytábla sort, és egy belső eredmény bufferbe helyezi. SELECT parancsok eredménytábláján használható.

OCIFetchInto (PHP 3>= 3.0.4, PHP 4)

A következő sor lekérdezése tömbbe

int **OCIFetchInto** (int stmt, array & result [, int mode]) \linebreak

Az **OCIFetchInto()** lekérdezi a következő sort, és a *`result`* tömbbe tölti be. SELECT parancsok eredménytábláján használható. A függvény felülírja a *`result`* korábbi tartalmát, ha volt ilyen. Alapértelmezésben a *`result`* egy számindexes tömb lesz minden mezőértékkel, ami nem `NULL`.

A *`mode`* paraméterrel tudod szabályozni a result szerkezetét és tartalmát. Több opciót is megadhatsz, egyszerű összeadással, például írhatod, hogy OCI_ASSOC + OCI_RETURN_NULLS. A használható opciók:

`OCI_ASSOC` Asszociatív tömb formátum.
`OCI_NUM` Számindexes tömb formátum (alapértelmezés).
`OCI_RETURN_NULLS` A `NULL` értékű mezők visszaadása.
`OCI_RETURN_LOBS` A LOB érték visszaadása a leíró helyett.

OCIFetchStatement (PHP 3>= 3.0.8, PHP 4)

Miden eredménysor lekérdezése egy tömbbe

int **OCIFetchStatement** (int stmt, array & variable) \linebreak

Az **OCIFetchStatement()** lekérdezi az összes eredménysort, és egy PHP tömbbe helyezi. A függvény a lekrédezett sorok számával tér vissza.

Példa 1. OCIFetchStatement

```
<?php
/* OCIFetchStatement példa - mbritton@verinet.com (990624) */

$conn = OCILogon("scott","tiger");
```

```
$stmt = OCIParse($conn,"select * from dolgozok");

OCIExecute($stmt);

$sorokszama = OCIFetchStatement($stmt, $tabla);
if ( $sorokszama > 0 ) {
   print "<TABLE BORDER=\"1\">\n";
   print "<TR>\n";
   while ( list( $kulcs, $ertek ) = each( $tabla ) ) {
      print "<TH>$kulcs</TH>\n";
   }
   print "</TR>\n";

   for ( $i = 0; $i < $sorokszama; $i++ ) {
      reset($tabla);
      print "<TR>\n";
      while ( $mezo = each($tabla) ) {
         $adat = $mezo['value'];
         print "<TD>$adat[$i]</TD>\n";
      }
      print "</TR>\n";
   }
   print "</TABLE>\n";
} else {
   echo "Nincs adat a táblában<BR>\n";
}
print "$sorokszama sort sikerült lekérdezni<BR>\n";

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIFreeCollection (PHP 4 >= 4.1.0)

[Eddig] nem dokumentált

string **OCIFreeCollection** (object lob) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCIFreeCursor (PHP 3>= 3.0.8, PHP 4)

Kurzorhoz rendelt erőforrások felszabadítása

int **OCIFreeCursor** (int stmt) \linebreak

Az **OCIFreeCursor()** TRUE értéket ad siker, FALSE értéket ad kudarc esetén.

OCIFreeDesc (PHP 4)

Felszabadít egy LOB leírót

int **OCIFreeDesc** (object lob) \linebreak

Az **OCIFreeDesc()** TRUE értéket ad siker, FALSE értéket ad kudarc esetén.

OCIFreeStatement (PHP 3>= 3.0.5, PHP 4)

Parancshoz rendelt erőforrások felszabadítása

int **OCIFreeStatement** (int stmt) \linebreak

Az **OCIFreeStatement()** TRUE értéket ad siker, FALSE értéket ad kudarc esetén.

OCIInternalDebug (PHP 3>= 3.0.4, PHP 4)

A belső debug kimenet engedélyezése / letiltása

void **OCIInternalDebug** (int onoff) \linebreak

Az **OCIInternalDebug()** engedélyezi a belső debug kimenetet ha az *onoff* paramétert 1 értékre állítod. Tiltásra a 0 értékkel van lehetőség.

OCILoadLob (PHP 4)

[Eddig] nem dokumentált

string **OCILoadLob** (object lob) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCILogOff (PHP 3>= 3.0.4, PHP 4)

Oracle kapcsolat bontása

int **OCILogOff** (int connection) \linebreak

Az **OCILogOff()** függvény lezár egy Oracle kapcsolatot.

OCILogon (PHP 3>= 3.0.4, PHP 4)

Oracle adatbázis kapcsolat létesítése

int **OCILogon** (string username, string password [, string db]) \linebreak

Az **OCILogon()** egy kapcsolat azonosítóval tér vissza, amire a legtöbb OCI hívásban szükséged lesz. Az opcionális harmadik paraméter vagy a helyi Oracle példány nevét adja meg, vagy egy tnsnames.ora bejegyzést, amihez csatlakozni szeretnél. Ha nem adod meg ezt a paramétert, a PHP az ORACLE_SID (Oracle példány) vagy a TWO_TASK (tnsnames.ora) környezeti változókat használja, hogy eldöntse melyik adatbázishoz kell csatlakozni.

Az **OCILogon()** által visszaadott kapcsolatot az aktuális oldalon megosztva használja a PHP. Ez azt jelenti, hogy a tranzakciók végrehjatása vagy visszavonása minden nyitott tranzakcióra vonatkozik az adott oldalon, még akkor is, ha több adatbázis kapcsolattal is rendelkezel.

Ez a példa bemutatja a kapcsolatok megosztását.

Példa 1. OCILogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocilogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hello (proba varchar2(64))");
  ociexecute($stmt);
  echo $conn." tábla létrehozva\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hello");
  ociexecute($stmt);
  echo $conn." tábla törölve\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hello
            values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." beillesztés végrehajtva\n\n";
}
```

```
function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hello");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." törlés végrehajtva\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." tranzakció végrehajtva\n\n";
}

function rollback($conn)
{ ocirollback($conn);
  echo $conn." tranzakció visszavonva\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hello");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."----select végrehajtása\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"PROBA").">\n\n";
  echo $conn."----kész\n\n";
}

create_table($c1);
insert_data($c1);   // Új sor beillesztése c1 felhasználásával
insert_data($c2);   // Új sor beillesztése c2 felhasználásával

select_data($c1);   // Mindkét insert eredményét megkapjuk
select_data($c2);

rollback($c1);      // Visszavonjuk a c1-en végzett műveletet

select_data($c1);   // Mindkét műveletet visszavontuk
select_data($c2);

insert_data($c2);   // Sor beillesztése c2-vel
commit($c2);        // Tranzakció végrehajtása c2-vel

select_data($c1);   // A c2 beillesztés eredményét visszakapjuk

delete_data($c1);   // A c1-el töröljük a sorokat
select_data($c1);   // Egy sor sincs
select_data($c2);   // Egy sor sincs
commit($c1);        // A c1-el végrehajtjuk a tranzakciót

select_data($c1);   // Egy sor sincs
select_data($c2);   // Egy sor sincs

drop_table($c1);
print "</PRE></HTML>";
?>
```

Lásd még **OCIPLogon()** és **OCINLogon()**.

OCINewCollection (PHP 4 >= 4.0.6)

[Eddig] nem dokumentált

string **OCINewCollection** (int conn, string tdo [, string shema]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCINewCursor (PHP 3>= 3.0.8, PHP 4)

Új kurzor (parancs eredmény kezelő) megnyitása

int **OCINewCursor** (int conn) \linebreak

Az **OCINewCursor()** egy új eredménytábla kezelő kurzort nyit meg a megadott kapcsolaton.

Példa 1. REF CURSOR használata egy tárolt eljárásból

```
<?php
// feltesszük, hogy az info.kimenet tárolt eljárásod egy "ref cursor"-t
// ad vissza az :adat-ban

$conn = OCILogon("scott","tiger");
$kurzor = OCINewCursor($conn);
$stmt = OCIParse($conn,"begin info.kimenet(:adat); end;");

ocibindbyname($stmt,"adat", &$kurzor, -1, OCI_B_CURSOR);
ociexecute($stmt);
ociexecute($kurzor);

while (OCIFetchInto($kurzor, &$adat)) {
    var_dump($adat);
}

OCIFreeStatement($stmt);
OCIFreeCursor($kurzor);
OCILogoff($conn);
?>
```

Példa 2. REF CURSOR használata select parancsban

```
<?php
print "<HTML><BODY>";
$conn = OCILogon("scott","tiger");
$szamlalo_kurzor =
    "CURSOR(select count(dolgozoid) dolgozo_szam from dolgozok " .
    "where dolgozok.reszleg = reszleg.reszlegid) as DSZAM from reszleg";
$stmt = OCIParse($conn,"select reszlegid,rnev,$szamlalo_kurzor");

ociexecute($stmt);
print "<TABLE BORDER=\"1\">";
print "<TR>";
print "<TH>RÉSZLEG NÉV</TH>";
print "<TH>RÉSZLEG SZÁM</TH>";
print "<TH>DOLGOZÓK SZÁMA</TH>";
print "</TR>";

while (OCIFetchInto($stmt, &$adat, OCI_ASSOC)) {
    print "<TR>";
    $rnev  = $adat["RNEV"];
    $rszam = $adat["RESZLEGID"];
    print "<TD>$rnev</TD>";
    print "<TD>$rszam</TD>";
    ociexecute($adat["DSZAM"]);
    while (OCIFetchInto($adat["DSZAM"], &$reszadat, OCI_ASSOC)) {
        $dszam = $reszadat["DOLGOZO_SZAM"];
        print  "<TD>$dszam</TD>";
    }
    print "</TR>";
}
print "</TABLE>";
print "</BODY></HTML>";
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCINewDescriptor (PHP 3>= 3.0.7, PHP 4)

Egy új üres LOB vagy FILE leírót hoz létre

string **OCINewDescriptor** (int connection [, int type]) \linebreak

Az **OCINewDescriptor()** tárterületet foglal le leírók vagy LOB lokátorok tárolására. A *type* lehetséges értékei OCI_D_FILE, OCI_D_LOB és OCI_D_ROWID. LOB leírók esetén a a load, save és savefile metódusokat használhatod, a BFILE csak a load metódust támogatja. Lásd a második példa használati tippjeit.

Példa 1. OCINewDescriptor

```
<?php
    /* Ez a program űrlapból történő meghíváshoz készült.
     * A $felhasznalo, $jelszo, $tabla, $where és $commitmeret
     * értékeket várja az űrlapból. A program letörli a kiválasztott
     * sorokat a ROWID felhasználásval, és minden $commitmeret sor
     * után egy tranzakció végrehajtást eszközöl. Használd óvatosan,
     * a műveletek nem visszavonhatóak.
     */
    $conn = OCILogon($felhasznalo, $jelszo);
    $stmt = OCIParse($conn,"select rowid from $tabla $where");
    $rowid = OCINewDescriptor($conn,OCI_D_ROWID);
    OCIDefineByName($stmt,"ROWID",&$rowid);
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
       $sorokszama = OCIRowCount($stmt);
       $delete = OCIParse($conn,"delete from $tabla where ROWID = :rid");
       OCIBindByName($delete,":rid",&$rowid,-1,OCI_B_ROWID);
       OCIExecute($delete);
       print "$sorokszama\n";
       if ( ($sorokszama % $commitmeret) == 0 ) {
           OCICommit($conn);
       }
    }
    $sorokszama = OCIRowCount($stmt);
    print "$sorokszama sor törölve...\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
?>
```

```
<?php
    /* Ez a program LOB mezőbe állomány feltöltést mutat be.
     * Az űrlapmező, amit a példa feltételez a következő:
     * <form action="upload.php" method="post" enctype="multipart/form-data">
     * <input type="file" name="lob_upload">
     * ...
     */
  if(!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload"><br>
<input type="submit" value="Feltöltés"> - <input type="reset">
</form>
<?php
  } else {

     // $lob_upload tartalmazza a feltöltött file ideiglenes állomány nevét

     // lásd még az állomány feltöltésről szóló részt biztonságos
     // file feltöltés példák érdekében

     $conn = OCILogon($felhasznalo, $jelszo);
     $lob = OCINewDescriptor($conn, OCI_D_LOB);
```

```
     $stmt = OCIParse($conn,"insert into $tabla (id, a_blob)
               values(my_seq.NEXTVAL, EMPTY_BLOB()) returning a_blob into :a_blob");
     OCIBindByName($stmt, ':a_blob', &$lob, -1, OCI_B_BLOB);
     OCIExecute($stmt, OCI_DEFAULT);
     if($lob->savefile($lob_upload)){
        OCICommit($conn);
        echo "A Blob feltöltése sikeres\n";
     }else{
        echo "Nem sikerült feltölteni a Blobot\n";
     }
     OCIFreeDesc($lob);
     OCIFreeStatement($stmt);
     OCILogoff($conn);
  }
?>
```

Példa 2. OCINewDescriptor

```
<?php
    /* PL/SQL tárolt eljárások hívása, amik clobs bemeneti paramétereket
     * tartalmaznak (PHP 4 >= 4.0.6).
     *
     * PROCEDURE adat_mentes
     *   Argumentum neve                Típus                    KI/BE  Alapérték
     *   ------------------------------ ------------------------ ------ ----
-----
     *   KULCS                          NUMBER(38)               BE
     *   ADAT                           CLOB                     BE
     *
     */

    $conn = OCILogon($felhasznalo, $jelszo);
    $stmt = OCIParse($conn, "begin adat_mentes(:kulcs, :adat); end;");
    $clob = OCINewDescriptor($conn, OCI_D_LOB);
OCIBindByName($stmt, ':kulcs', $kulcs);
OCIBindByName($stmt, ':adat',  $clob, -1, OCI_B_CLOB);
$clob->WriteTemporary($adat);
OCIExecute($stmt, OCI_DEFAULT);
OCICommit($conn);
$clob->close();
$clob->free();
OCIFreeStatement($stmt);
?>
```

OCINLogon (PHP 3>= 3.0.8, PHP 4)

Új Oracle adatbázis kapcsolat létrehozása

int **OCINLogon** (string username, string password [, string db]) \linebreak

Az **OCINLogon()** egy új Oracle 8 adatbázis kapcsolatot hoz létre, és bejelentkezik. Az opcionális harmadik paraméter vagy a helyi Oracle példány nevét adja meg, vagy egy tnsnames.ora bejegyzést, amihez csatlakozni szeretnél. Ha nem adod meg ezt a paramétert, a PHP az ORACLE_SID (Oracle példány) vagy a TWO_TASK (tnsnames.ora) környezeti változókat használja, hogy eldöntse melyik adatbázishoz kell csatlakozni.

Az **OCINLogon()** mindenképpen egy új kapcsolatot hoz létre. Erre akkor van szükséged, ha el kell különítened a tranzakcióidat egymástól. Alapértelmezésben a kapcsolatokat a PHP megosztja egy oldalon belül, ha az **OCILogon()** függvényt használod. Az **OCIPLogon()** használata esetén pedig már a web szerver process szintjén érvényesül a megosztás. Ha több kapcsolatod van az **OCINLogon()** függvénnyel megnyitva, minden tranzakció végrehajtás és visszavonás cska az adott kapcsolatra vonatkozik.

A következő példa bemutatja a kapcsolatok elkülönülését.

Példa 1. OCINLogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocinlogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hello (proba varchar2(64))");
  ociexecute($stmt);
  echo $conn." tábla létrehozva\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hello");
  ociexecute($stmt);
  echo $conn." tábla törölve\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hello
            values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." beillesztés végrehajtva\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hello");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." törlés végrehajtva\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." tranzakció végrehajtva\n\n";
}
```

```
function rollback($conn)
{ ocirollback($conn);
  echo $conn." tranzakció visszavonva\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hello");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."----select végrehajtása\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"PROBA").">\n\n";
  echo $conn."----kész\n\n";
}

create_table($c1);
insert_data($c1);

select_data($c1);
select_data($c2);

rollback($c1);

select_data($c1);
select_data($c2);

insert_data($c2);
commit($c2);

select_data($c1);

delete_data($c1);
select_data($c1);
select_data($c2);
commit($c1);

select_data($c1);
select_data($c2);

drop_table($c1);
print "</PRE></HTML>";
?>
```

Lásd még **OCILogon()** és **OCIPLogon()**.

OCINumCols (PHP 3>= 3.0.4, PHP 4)

Az eredményoszlopok száma egy eredménytáblában

int **OCINumCols** (int stmt) \linebreak

Az **OCINumCols()** visszaadja az eredménytábla oszlopainak a számát.

Példa 1. OCINumCols

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from dolgozok");
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
        print "\n";
        $ncols = OCINumCols($stmt);
        for ( $i = 1; $i <= $ncols; $i++ ) {
            $oszlop_neve  = OCIColumnName($stmt,$i);
            $mezo_erteke = OCIResult($stmt,$i);
            print $oszlop_neve . ': ' . $mezo_erteke . "\n";
        }
        print "\n";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

OCIParse (PHP 3>= 3.0.4, PHP 4)

SQL lekérdezés feldolgozása, parancs visszaadása

int **OCIParse** (int conn, string query) \linebreak

Az **OCIParse()** feldolgozz a *`query`* paraméterben megadott SQL lekérdezést a *`conn`* kapcsolaton. A parancs azonosítóval tér vissza, ha a lekérdezés érvényes volt, `FALSE` értéket ad egyébként. A *`query`* paramétert bármilyen érvényes SQL parancs vagy PL/SQL blokk lehet.

OCIPLogon (PHP 3>= 3.0.8, PHP 4)

Oracle adatbáziscsatlakozás állandó kapcsolattal

int **OCIPLogon** (string username, string password [, string db]) \linebreak

Az **OCIPLogon()** egy állandó (persistent) adatbázis kapcsolatot hoz létre az Oracle8 adatbázishoz, és bejelentkezik. Az opcionális harmadik paraméter vagy a helyi Oracle példány nevét adja meg, vagy egy tnsnames.ora bejegyzést, amihez csatlakozni szeretnél. Ha nem adod meg ezt a paramétert, a PHP az ORACLE_SID (Oracle példány) vagy a TWO_TASK (tnsnames.ora) környezeti változókat használja, hogy eldöntse melyik adatbázishoz kell csatlakozni.

Lásd még **OCILogon()** és **OCINLogon()**.

OCIResult (PHP 3>= 3.0.4, PHP 4)

Mező értékének visszaadása lekérdezett sorból

mixed **OCIResult** (int statement, mixed column) \linebreak

Az **OCIResult()** a *`column`* oszlopban lévő adatot adja vissza az aktuális sorban. Lásd az **OCIFetch()** függvényt. Az **OCIResult()** mindent karaktersorozatként ad vissza, kivéve az absztrakt adattípusokat (ROWID, LOB és FILE típusok).

OCIRollback (PHP 3>= 3.0.7, PHP 4)

Függőben lévő tranzakciók visszavonása

int **OCIRollback** (int connection) \linebreak

Az **OCIRollback()** minden a *`connection`* Oracle kapcsolaton függőben lévő tranzakciót visszavon.

OCIRowCount (PHP 3>= 3.0.7, PHP 4)

Az érintett sorok száma

int **OCIRowCount** (int statement) \linebreak

Az **OCIRowCount()** visszaadja az érintett sorok számát például egy update parancs után. Ez a függvény nem adja vissza a select lekérdezés által visszaadott sorok számát!

Példa 1. OCIRowCount

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $stmt = OCIParse($conn,"create table dolgozok2 as select * from dolgozok");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " sor beillesztve.<BR>";
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"delete from dolgozok2");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " sor törölve.<BR>";
    OCICommit($conn);
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"drop table dolgozok2");
    OCIExecute($stmt);
    OCIFreeStatement($stmt);
    OCILogOff($conn);
    print "</PRE></HTML>";
?>
```

OCISaveLob (PHP 4)

[Eddig] nem dokumentált

string **OCISaveLob** (object lob) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCISaveLobFile (PHP 4)

[Eddig] nem dokumentált

string **OCISaveLobFile** (object lob) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

OCIServerVersion (PHP 3>= 3.0.4, PHP 4)

Szerver verzió információ visszaadása

string **OCIServerVersion** (int conn) \linebreak

Példa 1. OCIServerVersion

```
<?php
   $conn = OCILogon("scott","tiger");
   print "Szerver verzió: " . OCIServerVersion($conn);
   OCILogOff($conn);
?>
```

OCISetPrefetch (PHP 3>= 3.0.12, PHP 4)

Az előzetesen lehívott sorok számának beállítása

int **OCISetPrefetch** (int stmt, int rows) \linebreak

Beállítja az előzetesen lehívott sorok számát. Az alapértelmezés 1 sor.

OCIStatementType (PHP 3>= 3.0.5, PHP 4)

OCI parancs típusát adja vissza

string **OCIStatementType** (int stmt) \linebreak

Az **OCIStatementType()** a parancs típusát adja vissza. A következők a lehetséges visszatérési értékek:

1. "SELECT"
2. "UPDATE"
3. "DELETE"
4. "INSERT"
5. "CREATE"
6. "DROP"
7. "ALTER"
8. "BEGIN"
9. "DECLARE"
10. "UNKNOWN"

Példa 1. OCIStatementType() példák

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $sql  = "delete from dolgozok where reszleg = 10";

    $stmt = OCIParse($conn,$sql);
    if ( OCIStatementType($stmt) == "DELETE" ) {
        die "Nincs jogod törölni ebből a táblából<BR>";
    }

    OCILogoff($conn);
    print "</PRE></HTML>";
?>
```

OCIWriteLobToFile (PHP 4)

[Eddig] nem dokumentált

void **OCIWriteLobToFile** (object lob [, string filename [, int start [, int lenght]]]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

LXXI. OpenSSL functions

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

Introduction

This module uses the functions of OpenSSL (http://www.openssl.org/) for generation and verification of signatures and for sealing (encrypting) and opening (decrypting) data. PHP-4.0.4pl1 requires OpenSSL >= 0.9.6, but PHP-4.0.5 and greater with also work with OpenSSL >= 0.9.5.

Megjegyzés: Please keep in mind that this extension is still considered experimental!

OpenSSL offers many features that this module currently doesn't support. Some of these may be added in the future.

Key/Certificate parameters

Quite a few of the openssl functions require a key or a certificate parameter. PHP 4.0.5 and earlier have to use a key or certificate resource returned by one of the openssl_get_xxx functions. Later versions may use one of the following methods:

- Certificates
  1. An X.509 resource returned from openssl_x509_read
  2. A string having the format `file://path/to/cert.pem`; the named file must contain a PEM encoded certificate
  3. A string containing the content of a certificate, PEM encoded
- Public/Private Keys
  1. A key resource returned from openssl_get_publickey() or openssl_get_privatekey()
  2. For public keys only: an X.509 resource
  3. A string having the format `file://path/to/file.pem` - the named file must contain a PEM encoded certificate/private key (it may contain both)
  4. A string containing the content of a certificate/key, PEM encoded
  5. For private keys, you may also use the syntax *array($key, $passphrase)* where $key represents a key specified using the file:// or textual content notation above, and $passphrase

represents a string containing the passphrase for that private key

Certificate Verification

When calling a function that will verify a signature/certificate, the *cainfo* parameter is an array containing file and directory names the specify the locations of trusted CA files. If a directory is specified, then it must be a correctly formed hashed directory as the **openssl** command would use.

PKCS7 Flags/Constants

The S/MIME functions make use of flags which are specified using a bitfield which can include one or more of the following values:

Táblázat 1. PKCS7 CONSTANTS

Constant	Description
PKCS7_TEXT	adds text/plain content type headers to encrypted/signed message. If decrypting or verifying, it strips those headers from the output - if the decrypted or verified message is not of MIME type text/plain then an error will occur.
PKCS7_BINARY	normally the input message is converted to "canonical" format which is effectively using CR and LF as end of line: as required by the S/MIME specification. When this options is present, no translation occurs. This is useful when handling binary data which may not be in MIME format.
PKCS7_NOINTERN	when verifying a message, certificates (if any) included in the message are normally searched for the signing certificate. With this option only the certificates specified in the `extracerts` parameter of openssl_pkcs7_verify() are used. The supplied certificates can still be used as untrusted CAs however.
PKCS7_NOVERIFY	do not verify the signers certificate of a signed message.
PKCS7_NOCHAIN	do not chain verification of signers certificates: that is don’t use the certificates in the signed message as untrusted CAs.

Constant	Description
PKCS7_NOCERTS	when signing a message the signer's certificate is normally included - with this option it is excluded. This will reduce the size of the signed message but the verifier must have a copy of the signers certificate available locally (passed using the `extracerts` to openssl_pkcs7_verify() for example.
PKCS7_NOATTR	normally when a message is signed, a set of attributes are included which include the signing time and the supported symmetric algorithms. With this option they are not included.
PKCS7_DETACHED	When signing a message, use cleartext signing with the MIME type multipart/signed. This is the default if the `flags` parameter to openssl_pkcs7_sign() if you do not specify any flags. If you turn this option off, the message will be signed using opaque signing, which is more resistant to translation by mail relays but cannot be read by mail agents that do not support S/MIME.
PKCS7_NOSIGS	Don't try and verify the signatures on a message

Megjegyzés: These constants were added in 4.0.6.

openssl_csr_export_to_file (PHP 4 >= 4.2.0)

Exports a CSR to file or a var

bool **openssl_csr_export_to_file** (resource csr, string outfilename [, bool notext]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_csr_export (PHP 4 >= 4.2.0)

Exports a CSR to file or a var

bool **openssl_csr_export** (resource csr, string out [, bool notext]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_csr_new (PHP 4 >= 4.2.0)

Generates a privkey and CSR

bool **openssl_csr_new** (array dn, resource privkey [, array extraattribs [, array configargs]]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_csr_sign (PHP 4 >= 4.2.0)

Signs a cert with another CERT

resource **openssl_csr_sign** (mixed csr, mixed x509, mixed priv_key, long days) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_error_string (PHP 4 >= 4.0.6)

Return openSSL error message

mixed **openssl_error_string** (void) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Returns an error message string, or FALSE if there are no more error messages to return.

openssl_error_string() returns the last error from the openSSL library. Error messages are stacked, so this function should be called multiple times to collect all of the information.

The parameters/return type of this function may change before it appears in a release version of PHP

Példa 1. openssl_error_string() example

```
// lets assume you just called an openssl function that failed
while($msg = openssl_error_string())
    echo $msg . "<br />\n";
```

Megjegyzés: This function was added in 4.0.6.

openssl_free_key (PHP 4 >= 4.0.4)

Free key resource

void **openssl_free_key** (resource key_identifier) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

openssl_free_key() frees the key associated with the specified `key_identifier` from memory.

openssl_get_privatekey (PHP 4 >= 4.0.4)

Prepare a PEM formatted private key for use

resource **openssl_get_privatekey** (mixed key [, string passphrase]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Returns a positive key resource identifier on success, or FALSE on error.

openssl_get_privatekey() parses the PEM formatted private key specified by `key` and prepares it for use by other functions. The optional parameter `passphrase` must be used if the specified key is encrypted (protected by a passphrase).

openssl_get_publickey (PHP 4 >= 4.0.4)

Extract public key from certificate and prepare it for use

resource **openssl_get_publickey** (mixed certificate) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Returns a positive key resource identifier on success, or FALSE on error.

openssl_get_publickey() extracts the public key from an X.509 certificate specified by `certificate` and prepares it for use by other functions.

openssl_open (PHP 4 >= 4.0.4)

Open sealed data

bool **openssl_open** (string sealed_data, string open_data, string env_key, mixed priv_key_id) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad. If successful the opened data is returned in `open_data`.

openssl_open() opens (decrypts) `sealed_data` using the private key associated with the key identifier `priv_key_id` and the envelope key `env_key`, and fills `open_data` with the

decrypted data. The envelope key is generated when the data are sealed and can only be used by one specific private key. See openssl_seal() for more information.

Példa 1. openssl_open() example

```
// $sealed and $env_key are assumed to contain the sealed data
// and our envelope key, both given to us by the sealer.

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// decrypt the data and store it in $open
if (openssl_open($sealed, $open, $env_key, $pkeyid))
    echo "here is the opened data: ", $open;
else
    echo "failed to open data";

// free the private key from memory
openssl_free_key($pkeyid);
```

See also openssl_seal().

openssl_pkcs7_decrypt (PHP 4 >= 4.0.6)

Decrypts an S/MIME encrypted message

bool **openssl_pkcs7_decrypt** (string infilename, string outfilename, mixed recipcert, mixed recipkey) \line-break

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Decrypts the S/MIME encrypted message contained in the file specified by *`infilename`* using the certificate and it's associated private key specified by *`recipcert`* and *`recipkey`*.

The decrypted message is output to the file specified by *`outfilename`*

Példa 1. openssl_pkcs7_decrypt() example

```
// $cert and $key are assumed to contain your personal certificate and private
// key pair, and that you are the recipient of an S/MIME message
$infilename = "encrypted.msg";  // this file holds your encrypted message
$outfilename = "decrypted.msg"; // make sure you can write to this file

if (openssl_pkcs7_decrypt($infilename, $outfilename, $cert, $key))
    echo "decrypted!";
else
    echo "failed to decrypt!";
```

Megjegyzés: This function was added in 4.0.6.

openssl_pkcs7_encrypt (PHP 4 >= 4.0.6)

Encrypt an S/MIME message

bool **openssl_pkcs7_encrypt** (string infile, string outfile, mixed recipcerts, array headers [, long flags]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

openssl_pkcs7_encrypt() takes the contents of the file named *`infile`* and encrypts them using an RC2 40-bit cipher so that they can only be read by the intended recipients specified by *`recipcerts`*, which is either a lone X.509 certificate, or an array of X.509 certificates. *`headers`* is an array of headers that will be prepended to the data after it has been encrypted. *`flags`* can be used to specify options that affect the encoding process - see PKCS7 constants. *`headers`* can be either an associative array keyed by header name, or an indexed array, where each element contains a single header line.

Példa 1. openssl_pkcs7_encrypt() example

```
// the message you want to encrypt and send to your secret agent
// in the field, known as nighthawk.  You have his certificate
// in the file nighthawk.pem
$data = <<<EOD
Nighthawk,

Top secret, for your eyes only!
```

```
The enemy is closing in! Meet me at the cafe at 8.30am
to collect your forged passport!

HQ
EOD;

// load key
$key = implode("", file("nighthawk.pem"));

// save message to file
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);

// encrypt it
if (openssl_pkcs7_encrypt("msg.txt", "enc.txt", $key,
    array("To" => "nighthawk@example.com", // keyed
syntax
          "From: HQ <hq@example.com>", // indexed syntax
          "Subject" => "Eyes only")))
{
    // message encrypted - send it!
    exec(ini_get("sendmail_path") . " < enc.txt");
}
```

openssl_pkcs7_sign (PHP 4 >= 4.0.6)

sign an S/MIME message

bool **openssl_pkcs7_sign** (string infilename, string outfilename, mixed signcert, mixed privkey, array headers [, long flags [, string extracertsfilename]]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

openssl_pkcs7_sign() takes the contents of the file named *infilename* and signs them using the certificate and it's matching private key specified by *signcert* and *privkey* parameters.

headers is an array of headers that will be prepended to the data after it has been signed (see openssl_pkcs7_encrypt() for more information about the format of this parameter.

flags can be used to alter the output - see PKCS7 constants - if not specified, it defaults to PKCS7_DETACHED.

extracerts specifies the name of a file containing a bunch of extra certificates to include in the signature which can for example be used to help the recipient to verify the certificate that you used.

Példa 1. openssl_pkcs7_sign() example

```
// the message you want to sign so that recipient can be sure it was you that
// sent it
$data = <<<EOD

You have my authorization to spend $10,000 on dinner expenses.

The CEO
EOD;
// save message to file
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);
// encrypt it
if (openssl_pkcs7_sign("msg.txt", "signed.txt", "mycert.pem",
    array("mycert.pem", "mypassphrase"),
    array("To" => "joes@sales.com", // keyed syntax
          "From: HQ <ceo@sales.com>", // indexed syntax
          "Subject" => "Eyes only"))
{
    // message signed - send it!
    exec(ini_get("sendmail_path") . " < signed.txt");
}
```

Megjegyzés: This function was added in 4.0.6.

openssl_pkcs7_verify (PHP 4 >= 4.0.6)

Verifies the signature of an S/MIME signed message

bool **openssl_pkcs7_verify** (string filename, int flags [, string outfilename [, array cainfo [, string extracerts]]]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

openssl_pkcs7_verify() reads the S/MIME message contained in the filename specified by *`filename`* and examines the digital signature. It returns `TRUE` if the signature is verified, `FALSE` if it is not correct (the message has been tampered with, or the signing certificate is invalid), or -1 on error.

`flags` can be used to affect how the signature is verified - see PKCS7 constants for more information.

If the `outfilename` is specified, it should be a string holding the name of a file into which the certificates of the persons that signed the messages will be stored in PEM format.

If the `cainfo` is specified, it should hold information about the trusted CA certificates to use in the verification process - see certificate verification for more information about this parameter.

If the `extracerts` is specified, it is the filename of a file containing a bunch of certificates to use as untrusted CAs.

Megjegyzés: This function was added in 4.0.6.

openssl_pkey_export_to_file (PHP 4 >= 4.2.0)

Gets an exportable representation of a key into a file

bool **openssl_pkey_export_to_file** (mixed key, string outfilename [, string passphrase [, array config_args]]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_pkey_export (PHP 4 >= 4.2.0)

Gets an exportable representation of a key into a string or file

bool **openssl_pkey_export** (mixed key, mixed out [, string passphrase [, array config_args]]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_pkey_new (PHP 4 >= 4.2.0)

Generates a new private key

resource **openssl_pkey_new** ([array configargs]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_private_decrypt (PHP 4 >= 4.0.6)

Decrypts data with private key

bool **openssl_private_decrypt** (string data, string crypted, mixed key [, int padding]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_private_encrypt (PHP 4 >= 4.0.6)

Encrypts data with private key

bool **openssl_private_encrypt** (string data, string crypted, mixed key [, int padding]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_public_decrypt (PHP 4 >= 4.0.6)

Decrypts data with public key

bool **openssl_public_decrypt** (string data, string crypted, resource key [, int padding]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_public_encrypt (PHP 4 >= 4.0.6)

Encrypts data with public key

bool **openssl_public_encrypt** (string data, string crypted, mixed key [, int padding]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ez a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_seal (PHP 4 >= 4.0.4)

Seal (encrypt) data

int **openssl_seal** (string data, string sealed_data, array env_keys, array pub_key_ids) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ez a függvényt csak a saját felelősségedre használd!

Returns the length of the sealed data on success, or FALSE on error. If successful the sealed data is returned in *sealed_data*, and the envelope keys in *env_keys*.

openssl_seal() seals (encrypts) *data* by using RC4 with a randomly generated secret key. The key is encrypted with each of the public keys associated with the identifiers in *pub_key_ids* and each encrypted key is returned in *env_keys*. This means that one can send sealed data to multiple recipients (provided one has obtained their public keys). Each recipient must receive both the sealed data and the envelope key that was encrypted with the recipient's public key.

Példa 1. openssl_seal() example

```
// $data is assumed to contain the data to be sealed

// fetch public keys for our recipients, and ready them
$fp = fopen("/src/openssl-0.9.6/demos/maurice/cert.pem", "r");
```

```
$cert = fread($fp, 8192);
fclose($fp);
$pk1 = openssl_get_publickey($cert);
// Repeat for second recipient
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk2 = openssl_get_publickey($cert);

// seal message, only owners of $pk1 and $pk2 can decrypt $sealed with keys
// $ekeys[0] and $ekeys[1] respectively.
openssl_seal($data, $sealed, $ekeys, array($pk1,$pk2));

// free the keys from memory
openssl_free_key($pk1);
openssl_free_key($pk2);
```

See also openssl_open().

openssl_sign (PHP 4 >= 4.0.4)

Generate signature

bool **openssl_sign** (string data, string signature, mixed priv_key_id) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad. If successful the signature is returned in *signature*.

openssl_sign() computes a signature for the specified *data* by using SHA1 for hashing followed by encryption using the private key associated with *priv_key_id*. Note that the data itself is not encrypted.

Példa 1. openssl_sign() example

```
// $data is assumed to contain the data to be signed

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);
```

```
// compute signature
openssl_sign($data, $signature, $pkeyid);

// free the key from memory
openssl_free_key($pkeyid);
```

See also openssl_verify().

openssl_verify (PHP 4 >= 4.0.4)

Verify signature

int **openssl_verify** (string data, string signature, mixed pub_key_id) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Returns 1 if the signature is correct, 0 if it is incorrect, and -1 on error.

openssl_verify() verifies that the *`signature`* is correct for the specified *`data`* using the public key associated with *`pub_key_id`*. This must be the public key corresponding to the private key used for signing.

Példa 1. openssl_verify() example

```
// $data and $signature are assumed to contain the data and the signature

// fetch public key from certificate and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pubkeyid = openssl_get_publickey($cert);

// state whether signature is okay or not
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1)
    echo "good";
elseif ($ok == 0)
    echo "bad";
else
    echo "ugly, error checking signature";

// free the key from memory
openssl_free_key($pubkeyid);
```

See also openssl_sign().

openssl_x509_check_private_key (PHP 4 >= 4.2.0)

Checks if a private key corresponds to a CERT

bool **openssl_x509_check_private_key** (mixed cert, mixed key) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_x509_checkpurpose (PHP 4 >= 4.0.6)

Verifies if a certificate can be used for a particular purpose

bool **openssl_x509_checkpurpose** (mixed x509cert, int purpose, array cainfo [, string untrustedfile]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Returns TRUE if the certificate can be used for the intended purpose, FALSE if it cannot, or -1 on error.

openssl_x509_checkpurpose() examines the certificate specified by *`x509cert`* to see if it can be used for the purpose specified by *`purpose`*.

`cainfo` should be an array of trusted CA files/dirs as described in Certificate Verification.

untrustedfile, if specified, is the name of a PEM encoded file holding certificates that can be used to help verify the certificate, although no trust in placed in the certificates that come from that file.

Táblázat 1. openssl_x509_checkpurpose() purposes

Constant	Description
X509_PURPOSE_SSL_CLIENT	Can the certificate be used for the client side of an SSL connection?
X509_PURPOSE_SSL_SERVER	Can the certificate be used for the server side of an SSL connection?
X509_PURPOSE_NS_SSL_SERVER	Can the cert be used for Netscape SSL server?
X509_PURPOSE_SMIME_SIGN	Can the cert be used to sign S/MIME email?
X509_PURPOSE_SMIME_ENCRYPT	Can the cert be used to encrypt S/MIME email?
X509_PURPOSE_CRL_SIGN	Can the cert be used to sign a certificate revocation list (CRL)?
X509_PURPOSE_ANY	Can the cert be used for Any/All purposes?

These options are not bitfields - you may specify one only!

Megjegyzés: This function was added in 4.0.6.

openssl_x509_export_to_file (PHP 4 >= 4.2.0)

Exports a CERT to file or a var

bool **openssl_x509_export_to_file** (mixed x509, string outfilename [, bool notext]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_x509_export (PHP 4 >= 4.2.0)

Exports a CERT to file or a var

bool **openssl_x509_export** (mixed x509, string outfilename [, bool notext]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

openssl_x509_free (PHP 4 >= 4.0.6)

Free certificate resource

void **openssl_x509_free** (resource x509cert) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

openssl_x509_free() frees the certificate associated with the specified `x509cert` resource from memory.

Megjegyzés: This function was added in 4.0.6.

openssl_x509_parse (PHP 4 >= 4.0.6)

Parse an X509 certificate and return the information as an array

array **openssl_x509_parse** (mixed x509cert [, bool shortnames]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

openssl_x509_parse() returns information about the supplied *x509cert*, including fields such as subject name, issuer name, purposes, valid from and valid to dates etc. *shortnames* controls how the data is indexed in the array - if *shortnames* is TRUE (the default) then fields will be indexed with the short name form, otherwise, the long name form will be used - e.g.: CN is the shortname form of commonName.

The structure of the returned data is (deliberately) not yet documented, as it is still subject to change.

Megjegyzés: This function was added in 4.0.6.

openssl_x509_read (PHP 4 >= 4.0.6)

Parse an X.509 certificate and return a resource identifier for it

resource **openssl_x509_read** (mixed x509certdata) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

openssl_x509_read() parses the certificate supplied by *x509certdata* and returns a resource identifier for it.

Megjegyzés: This function was added in 4.0.6.

LXXII. Oracle függvények

Ora_Bind (PHP 3, PHP 4)

PHP változó kötése egy Oracle paraméterhez

int **ora_bind** (int cursor, string PHP variable name, string SQL parameter name, int length [, int type]) \linebreak

TRUE értékkel tér vissza, ha sikeres a kötés, egyébként FALSE értéket ad. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

Ez a függvény összeköti a megnevezett PHP változót egy SQL paraméterrel. Az SQL paraméternek ":név" formátumúnak kell lennie. Az opcionális típus paraméterrel lehet meghatározni, hogy az SQL paraméter egy bemeneti/kimeneti (0, alapértelmezett), bemeneti (1), vagy kimeneti (2) paraméter. A PHP 3.0.1 verziójától az ORA_BIND_INOUT, ORA_BIND_IN és ORA_BIND_OUT konstansokat is lehet a számok helyett használni.

Az **ora_bind()** függvényt az ora_parse() után és az ora_exec() előtt kell meghívni. A bemeneti értékeket az összekapcsolt PHP változóknak való értékadással lehet megadni, az ora_exec() meghívása után pedig az összekapcsolt PHP változók tartalmazzák majd a kimeneti értékeket, ha vannak kimeneti értékek.

```
<?php
ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;");
ora_bind ($curs, "result", ":x",   $len, 2);
ora_bind ($curs, "input",  ":in",  5,    1);
ora_bind ($curs, "output", ":out", 5,    2);
$input = 765;
ora_exec($curs);
echo "Result: $result<br>Out: $output<br>In: $input";
?>
```

Ora_Close (PHP 3, PHP 4)

Oracle kurzor bezárása

int **ora_close** (int cursor) \linebreak

TRUE értékkel tér vissza, ha sikeres a lezárás. Ellenkező esetben FALSE értéket ad. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

Ez a függvény lezár egy adat kurzort, amit az ora_open() függvénnyel nyitottál.

Ora_ColumnName (PHP 3, PHP 4)

Oracle eredményoszlop nevét adja vissza

string **Ora_ColumnName** (int cursor, int column) \linebreak

Visszaadja a *column* mező/oszlop nevét, amelyen a *cursor* kurzor áll. A visszaadott név csupa nagybetűkből áll. Az első oszlop sorszáma a nulla.

Ora_ColumnSize (PHP 3, PHP 4)

Oracle eredményoszlop méretét adja vissza

int **Ora_ColumnSize** (int cursor, int column) \linebreak

Visszaadja a *column* mező/oszlop méretét, amelyen a *cursor* kurzor áll. Az első oszlop sorszáma a nulla.

Ora_ColumnType (PHP 3, PHP 4)

Oracle eredményoszlop típusát adja vissza

string **Ora_ColumnType** (int cursor, int column) \linebreak

Visszaadja a *column* mező/oszlop Oracle adattípusát, amelyen a *cursor* kurzor áll. Az első oszlop sorszáma a nulla. A visszaadott típus a következők valamelyike lesz:

```
"VARCHAR2"
"VARCHAR"
"CHAR"
"NUMBER"
"LONG"
"LONG RAW"
"ROWID"
"DATE"
"kurzor"
```

Ora_Commit (PHP 3, PHP 4)

Oracle tranzakció végrehajtása

int **ora_commit** (int conn) \linebreak

`TRUE` értéket ad vissza, ha sikeres, egyébként `FALSE` értékkel tér vissza. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

Ez a függvény végrehajt egy Oracle tranzakciót. A tranzakciót úgy definiálhatjuk, mint az adott kapcsolaton bekövetkezett változásokat a legutolsó végrehajtás/visszavonás vagy a kapcsolat létrejötte óta, feltéve, hogy a változások automatikus végrehjatása ki volt kapcsolva.

Ora_CommitOff (PHP 3, PHP 4)

Automatikus végrehajtás letiltása

int **ora_commitoff** (int conn) \linebreak

TRUE értéket ad vissza, ha sikeres, egyébként FALSE értéket kapunk. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

Ez a függvény kikapcsolja az ora_exec() utáni automatikus végrehajtást.

Ora_CommitOn (PHP 3, PHP 4)

Automatikus végrehajtás engedélyezése

int **ora_commiton** (int conn) \linebreak

Ez a függvény bekapcsolja az automatikus elküldést minden ora_exec() után az adott kapcsolaton.

TRUE értéket ad vissza, ha sikeres, egyébként FALSE értéket kapunk. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

Ora_Do (PHP 3, PHP 4)

Parse, Exec és Fetch egyben

int **ora_do** (int conn, string query) \linebreak

Ez a függvény egy gyors kombinációja az ora_parse(), ora_exec() és ora_fetch() függvényeknek. Feldolgoz és végrehajt egy parancsot, majd letölti az első eredmény sort.

TRUE értéket ad vissza, ha sikeres, egyébként FALSE értéket kapunk. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

Lásd még az ora_parse(),ora_exec(), és ora_fetch() függvényeket.

Ora_Error (PHP 3, PHP 4)

Oracle hibaüzenet lekérése

string **Ora_Error** (int cursor_or_connection) \linebreak

Hibaüzenetet ad vissza az *XXX-NNNNN* formában, ahol *XXX* a hiba forrása, *NNNNN* pedig a hiba azonosítója.

Megjegyzés: A csatlakozás azonosítók támogatása a 3.0.4 verzióban került a nyelvbe.

Az Oracle Unix verzióin egy hibáról a következőképpen lehet részleteket megtudni: `$ oerr ora 00001 00001, 00000, "unique constraint (%s.%s) violated" // *Cause: An update or insert statement attempted to insert a duplicate key // For Trusted ORACLE configured in DBMS MAC mode, you may see // this message if`

```
a duplicate entry exists at a different level. // *Action: Either remove
the unique restriction or do not insert the key
```

Ora_ErrorCode (PHP 3, PHP 4)

Oracle hibakód lekérdezése

int **Ora_ErrorCode** (int cursor_or_connection) \linebreak

Az adott kurzoron, vagy kapcsolaton végrehajtott utolsó parancs hibakódját adja vissza.

Megjegyzés: A csatlakozás azonosítók támogatása a 3.0.4 verzióban került a nyelvbe.

Ora_Exec (PHP 3, PHP 4)

Feldolgozott parancs végrehajtása egy Oracle kurzoron

int **ora_exec** (int cursor) \linebreak

`TRUE` értéket ad vissza, ha sikeres, egyébként `FALSE` értéket kapunk. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

Lásd még ora_parse(), ora_fetch() és ora_do().

Ora_Fetch_Into (PHP 3, PHP 4)

Sor betöltése megadott eredmény tömbbe

int **ora_fetch_into** (int cursor, array result [, int flags]) \linebreak

Az **Ora_Fetch_Into()** egy sort tölt be a második paraméterben megadott tömbbe.

Példa 1. Oracle sor betöltése tömbbe

```
<?php
array($results);
ora_fetch_into($cursor, &$results);
echo $results[0];
echo $results[1];
?>
```

Fontos, hogy a tömböt referencia szerint kell átadnod.

Lásd még az ora_parse(), ora_exec(), ora_fetch() és ora_do() függvényeket.

Ora_Fetch (PHP 3, PHP 4)

Egy adatsor lekérdezése egy kurzorról

int **ora_fetch** (int cursor) \linebreak

TRUE értéket ad vissza, ha sikeres (lekérdezett egy sort), FALSE értéket kapunk, ha nincs több sor, vagy hiba lépett fel. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni. Ha nam volt hiba, a ora_errorcode() nullát ad vissza.

Lehív egy sornyi adatot a megadott kurzoron.

Lásd még az ora_parse(), ora_exec() és ora_do() függvényeket.

Ora_GetColumn (PHP 3, PHP 4)

Adat lekérése egy lekérdezett oszlopból

mixed **ora_getcolumn** (int cursor, mixed column) \linebreak

Az oszlophoz tartozó adatatot adja vissza. Ha hiba lép fel, FALSE értékkel tér vissza, és az ora_errorcode() nem-nulla értéke jelzi a hibát. Azonban a függvény adhat vissza FALSE értékű, de nem boolean típusú értéket is, még akkor is, ha nem történt hiba, (pl: NULL eredmény, üres sztring, az érték 0, a sztring "0").

Lekérdezi egy oszlop, vagy függvény-eredmény értékét.

Ora_Logoff (PHP 3, PHP 4)

Oracle kapcsolat lezárása

int **ora_logoff** (int conn) \linebreak

TRUE értéket ad vissza, ha sikeres, egyébként FALSE értéket kapunk. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

Kilépteti a felhasználót és bontja a szerverrel a kapcsolatot.

Lásd még az ora_logon() függvényt.

Ora_Logon (PHP 3, PHP 4)

Oracle kapcsolat megnyitása

int **ora_logon** (string user, string password) \linebreak

Létrehoz egy kapcsolatot a PHP és egy Oracle adatbázis szerver között, a felhasználónév és jelszó felhasználásával.

A kapcsolatokat az SQL*Net segítségével is létre lehet hozni, a felhasználó TNS név megadásával:

```
$conn = Ora_Logon("user<emphasis>@TNSNAME</emphasis>", "pass");
```

Amennyiben nem ASCII karakteres adattal kell dolgoznod, mindenképpen gondoskodj róla, hogy az NLS_LANG be legyen állítva a környezetben. Szerver modulok esetén a szerver környzetében kell beállítnai a szerver indítása előtt.

A függvény egy csatlakozás azonosítót ad vissza siker esetén, vagy FALSE értéket kudarc esetén. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

Ora_Numcols (PHP 3, PHP 4)

Az oszlopok száma egy eredménytáblában

int **ora_numcols** (int cursor_ind) \linebreak

Az **ora_numcols()** visszaadja az eredméyntáblában lévő oszlopok számát. Értelmes eredményt csak egy parse/exec/fetch függvényhívási sorozat után ad.

Lásd még az ora_parse(), ora_exec(), ora_fetch() és ora_do() függvényeket.

Ora_Numrows (PHP 3, PHP 4)

A sorok száma egy eredménytáblában

int **ora_numrows** (int cursor_ind) \linebreak

Az **ora_numrows()** a sorok számát adja vissza egy eredménytáblában.

Ora_Open (PHP 3, PHP 4)

Oracle kurzor megnyitása

int **ora_open** (int conn) \linebreak

Megnyit egy a csatlakozással társított Oracle kurzort.

Visszaadja a kurzor azonosítóját (kurzor indexét), vagy FALSE értéket hiba esetén. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

Ora_Parse (PHP 3, PHP 4)

SQL kifejezés feldolgozása

int **ora_parse** (int cursor_ind, string sql_statement, int defer) \linebreak

Ez a függvény feldolgoz egy SQL kifejezést, vagy egy PL/SQL blokkot és társítja a megadott kurzorral.

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

Lásd még az ora_exec(), ora_fetch() és ora_do() függvényeket.

Ora_pLogon (PHP 3, PHP 4)

Állandó Oracle kapcsolat megnyitása

int **ora_plogon** (string user, string pass) \linebreak

Állandó kapcsolatot hoz létre a PHP és egy Oracle adatbázis között az adott felhasználónévvel és jelszóval.

Lásd még az ora_logon() függvényt.

Ora_Rollback (PHP 3, PHP 4)

Visszavon egy tranzakciót

int **ora_rollback** (int conn) \linebreak

Az **Ora_Rollback()** visszavon egy Oracle tranzakciót. Lásd még az ora_commit() függvényt a tranzakció leírásáért.

TRUE értéket ad vissza, ha sikeres, egyébként FALSE értéket kapunk. A hibáról további részleteket az ora_error() és ora_errorcode() függvényekkel lehet megtudni.

LXXIII. Ovrimos SQL functions

Ovrimos SQL Server, is a client/server, transactional RDBMS combined with Web capabilities and fast transactions.

Ovrimos SQL Server is available at www.ovrimos.com (http://www.ovrimos.com/). To enable ovrimos support in PHP just compile PHP with the `--with-ovrimos` parameter to configure script. You'll need to install the sqlcli library available in the Ovrimos SQL Server distribution.

Példa 1. Connect to Ovrimos SQL Server and select from a system table

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo ("Connection ok!");
    $res = ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will just connect to an Ovrimos SQL server.

ovrimos_close (PHP 4 >= 4.0.3)

Closes the connection to ovrimos

void **ovrimos_close** (int connection) \linebreak

ovrimos_close() is used to close the specified connection.

ovrimos_close() closes a connection to Ovrimos. This has the effect of rolling back uncommitted transactions.

ovrimos_commit (PHP 4 >= 4.0.3)

Commits the transaction

int **ovrimos_commit** (int connection_id) \linebreak

ovrimos_commit() is used to commit the transaction.

ovrimos_commit() commits the transaction.

ovrimos_connect (PHP 4 >= 4.0.3)

Connect to the specified database

int **ovrimos_connect** (string host, string db, string user, string password) \linebreak

ovrimos_connect() is used to connect to the specified database.

ovrimos_connect() returns a connection id (greater than 0) or 0 for failure. The meaning of 'host' and 'port' are those used everywhere in Ovrimos APIs. 'Host' is a host name or IP address and 'db' is either a database name, or a string containing the port number.

Példa 1. ovrimos_connect() Example

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

The above example will connect to the database and print out the specified table.

ovrimos_cursor (PHP 4 >= 4.0.3)

Returns the name of the cursor

int **ovrimos_cursor** (int result_id) \linebreak

ovrimos_cursor() is used to get the name of the cursor.

ovrimos_cursor() returns the name of the cursor. Useful when wishing to perform positioned updates or deletes.

ovrimos_exec (PHP 4 >= 4.0.3)

Executes an SQL statement

int **ovrimos_exec** (int connection_id, string query) \linebreak

ovrimos_exec() is used to execute an SQL statement.

ovrimos_exec() executes an SQL statement (query or update) and returns a result_id or FALSE. Evidently, the SQL statement should not contain parameters.

ovrimos_execute (PHP 4 >= 4.0.3)

Executes a prepared SQL statement

bool **ovrimos_execute** (int result_id [, array parameters_array]) \linebreak

ovrimos_execute() is used to execute an SQL statement.

ovrimos_execute() executes a prepared statement. Returns TRUE or FALSE. If the prepared statement contained parameters (question marks in the statement), the correct number of parameters should be passed in an array. Notice that I don’t follow the PHP convention of placing just the name of the optional parameter inside square brackets. I couldn’t bring myself on liking it.

ovrimos_fetch_into (PHP 4 >= 4.0.3)

Fetches a row from the result set

bool **ovrimos_fetch_into** (int result_id, array result_array [, string how [, int rownumber]]) \linebreak

ovrimos_fetch_into() is used to fetch a row from the result set.

ovrimos_fetch_into() fetches a row from the result set into ’result_array’, which should be passed by reference. Which row is fetched is determined by the two last parameters. ’how’ is one of ’Next’ (default), ’Prev’, ’First’, ’Last’, ’Absolute’, corresponding to forward direction from current

position, backward direction from current position, forward direction from the start, backward direction from the end and absolute position from the start (essentially equivalent to ’first’ but needs ’rownumber’). Case is not significant. ’Rownumber’ is optional except for absolute positioning. Returns TRUE or FALSE.

Példa 1. A fetch into example

```
<?php
$conn=ovrimos_connect ("neptune", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn,"select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_into ($res, &$row)) {
            list ($table_id, $table_name) = $row;
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_into ($res, &$row)) {
                list ($table_id, $table_name) = $row;
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This example will fetch a row.

ovrimos_fetch_row (PHP 4 >= 4.0.3)

Fetches a row from the result set

bool **ovrimos_fetch_row** (int result_id [, int how [, int row_number]]) \linebreak

ovrimos_fetch_row() is used to fetch a row from the result set.

ovrimos_fetch_row() fetches a row from the result set. Column values should be retrieved with other calls. Returns TRUE or FALSE.

Példa 1. A fetch row example

```
<?php
$conn = ovrimos_connect ("remote.host", "8001", "admin", "password");
```

```
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_row ($res, "First")) {
            $table_id = ovrimos_result ($res, 1);
            $table_name = ovrimos_result ($res, 2);
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_row ($res, "Next")) {
                $table_id = ovrimos_result ($res, "table_id");
                $table_name = ovrimos_result ($res, "table_name");
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will fetch a row and print the result.

ovrimos_field_len (PHP 4 >= 4.0.3)

Returns the length of the output column

int **ovrimos_field_len** (int result_id, int field_number) \linebreak

ovrimos_field_len() is used to get the length of the output column with number *`field_number`*, in result *`result_id`*.

ovrimos_field_len() returns the length of the output column at the (1-based) index specified.

ovrimos_field_name (PHP 4 >= 4.0.3)

Returns the output column name

int **ovrimos_field_name** (int result_id, int field_number) \linebreak

ovrimos_field_name() is used to get the output column name.

ovrimos_field_name() returns the output column name at the (1-based) index specified.

ovrimos_field_num (PHP 4 >= 4.0.3)

Returns the (1-based) index of the output column

int **ovrimos_field_num** (int result_id, string field_name) \linebreak

ovrimos_field_num() is used to get the (1-based) index of the output column.

ovrimos_field_num() returns the (1-based) index of the output column specified by name, or FALSE.

ovrimos_field_type (PHP 4 >= 4.0.3)

Returns the (numeric) type of the output column

int **ovrimos_field_type** (int result_id, int field_number) \linebreak

ovrimos_field_type() is used to get the (numeric) type of the output column.

ovrimos_field_type() returns the (numeric) type of the output column at the (1-based) index specified.

ovrimos_free_result (PHP 4 >= 4.0.3)

Frees the specified result_id

bool **ovrimos_free_result** (int result_id) \linebreak

ovrimos_free_result() is used to free the result_id.

ovrimos_free_result() frees the specified result_id *`result_id`*. Returns TRUE.

ovrimos_longreadlen (PHP 4 >= 4.0.3)

Specifies how many bytes are to be retrieved from long datatypes

int **ovrimos_longreadlen** (int result_id, int length) \linebreak

ovrimos_longreadlen() is used to specify how many bytes are to be retrieved from long datatypes.

ovrimos_longreadlen() specifies how many bytes are to be retrieved from long datatypes (long varchar and long varbinary). Default is zero. It currently sets this parameter the specified result set. Returns TRUE.

ovrimos_num_fields (PHP 4 >= 4.0.3)

Returns the number of columns

int **ovrimos_num_fields** (int result_id) \linebreak

ovrimos_num_fields() is used to get the number of columns.

ovrimos_num_fields() returns the number of columns in a result_id resulting from a query.

ovrimos_num_rows (PHP 4 >= 4.0.3)

Returns the number of rows affected by update operations

int **ovrimos_num_rows** (int result_id) \linebreak

ovrimos_num_rows() is used to get the number of rows affected by update operations.

ovrimos_num_rows() returns the number of rows affected by update operations.

ovrimos_prepare (PHP 4 >= 4.0.3)

Prepares an SQL statement

int **ovrimos_prepare** (int connection_id, string query) \linebreak

ovrimos_prepare() is used to prepare an SQL statement.

ovrimos_prepare() prepares an SQL statement and returns a result_id (or `FALSE` on failure).

Példa 1. Connect to Ovrimos SQL Server and prepare a statement

```
<?php
$conn=ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_prepare ($conn, "select table_id, table_name
                                  from sys.tables where table_id=1");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res)) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close($conn);
}
?>
```

This will connect to Ovrimos SQL Server, prepare a statement and the execute it.

ovrimos_result_all (PHP 4 >= 4.0.3)

Prints the whole result set as an HTML table

bool **ovrimos_result_all** (int result_id [, string format]) \linebreak

ovrimos_result_all() is used to print an HTML table containing the whole result set.

ovrimos_result_all() prints the whole result set as an HTML table. Returns `TRUE` or `FALSE`.

Példa 1. Prepare a statement, execute, and view the result

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_prepare ($conn, "select table_id, table_name
                                    from sys.tables where table_id = 7");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res, array(3))) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close($conn);
}
?>
```

This will execute an SQL statement and print the result in an HTML table.

Példa 2. Ovrimos_result_all with meta-information

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "select table_id, table_name
                                 from sys.tables where table_id = 1")
    if ($res != 0) {
        echo "Statement ok! cursor=".ovrimos_cursor ($res)."\n";
        $colnb = ovrimos_num_fields ($res);
        echo "Output columns=".$colnb."\n";
        for ($i=1; $i <= $colnb; $i++) {
            $name = ovrimos_field_name ($res, $i);
            $type = ovrimos_field_type ($res, $i);
            $len = ovrimos_field_len ($res, $i);
```

```
            echo "Column ".$i." name=".$name." type=".$type." len=".$len."\n";
        }
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

Példa 3. ovrimos_result_all example

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "update test set i=5");
    if ($res != 0) {
        echo "Statement ok!";
        echo ovrimos_num_rows ($res)." rows affected\n";
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

ovrimos_result (PHP 4 >= 4.0.3)

Retrieves the output column

int **ovrimos_result** (int result_id, mixed field) \linebreak

ovrimos_result() is used to retrieve the output column.

ovrimos_result() retrieves the output column specified by ’field’, either as a string or as an 1-based index.

ovrimos_rollback (PHP 4 >= 4.0.3)

Rolls back the transaction

int **ovrimos_rollback** (int connection_id) \linebreak

ovrimos_rollback() is used to roll back the transaction.

ovrimos_rollback() rolls back the transaction.

LXXIV. Kimenet Szabályozó Függvények

A kimenet szabályzó függvények teszik lehetővé, hogy teljes kontrolt szerezzünk afelett, mikor kerüljon a PHP script futásának eredménye a kimenetre, azaz Apache modulként pl. mikor adja azt át ezt a webszervernek. Hasznos ez akkor, amikor fejlécadatokat akarunk még létrehozni/módosítani, miközben már a html lap genegrálásának közepén járunk. Eme kimenetvezérlő függvények nincsenek hatással a header() vagy a setcookie() által létrehozott fejléc adatokra, csakis a törzsben található, echo() vagy egyéb függvények által generált kimenetre, valamint a php kódon kívüli statikus részekre (azaz amik a <? > jeleken kívül találhatóak).

Példa 1. Kimenet Szabályzás Példa

```
<?php

ob_start();
echo "Szia\n";

setcookie ("sutineve", "sutiadat");

ob_end_flush();

?>
```

A fenti példában az echo által generált szöveg a pufferben várakozik mindaddig, amíg egy ob_end_flush() parancs nem érkezik. Mindeközben a setcookie() által generált fejlécsor minden hiba nélül elmegy a böngésző felé (ez amúgy lehetetlen volna, mivel az echo már a webszerver felé küldött adatok törzsébe dolgozna, így a fejlécbe írni már lehetetlenség lenne).

Lásd még: header() és setcookie().

flush (PHP 3, PHP 4)

A kimeneti puffer üritése

void **flush** (void) \linebreak

Ezzel a függvénnyel tudjuk kényszeríteni a php kimenetét esetlegesen pufferelő programnak (ez lehet a CGI réteg, avagyon a webszerver például), hogy azonnal ürítse ezt a puffert. Gyakorlatilag ez annyit jelent, hogy az adatok azonnal elindulnak a böngésző fele.

Megjegyzés: A **flush()** nincs hatással a php saját pufferelési rendszerére, sem a böngésző esetleges pufferelési mechanizmusára.

Bizonyos szerverek, példának okáért a Win32 alatt futók ennek ellenére is pufferelik a php-től kapott kimenetet, amíg annak futása be nem fejeződik.

Az Apache szerver moduljai, mint például a mod_gzip is rendelkezhetnek saját pufferelési rendszerrel. Ezáltal lehetséges, hogy a **flush()** meghívása nem eredményez azonnali adatküldést a böngésző fele.

Az is lehetséges még, hogy a böngésző maga puffereli a kapott adatokat. Például a Netscape a kapott html-t puffereli, amíg egy sorvége jellel nem találkozik, vagy egy html címke kezdetét nem látja. Emellett a táblázatokat csak a </table> zárócímke megérkezése után rajzolja ki.

ob_clean (PHP 4 >= 4.2.0)

A kimeneti puffer törlése

void **ob_clean** (void) \linebreak

E függvény alkalmazásával teljesen megsemmisíthetjük a kimeneti puffer tartalmát.

Ezt alkalmazva nem szűnik meg a kimeneti puffer, az ob_end_clean() függvénnyel ellentétben.

Lásd még: ob_flush(), ob_end_flush() és ob_end_clean().

ob_end_clean (PHP 4)

A kimeneti puffer törlése, és egyidejűleg a pufferelés kikapcsolása

void **ob_end_clean** (void) \linebreak

E függvény alkalmazásával teljesen megsemmisíthetjük a kimeneti puffer tartalmát, valamint megszüntethetjük a pufferelést is egyben.

Lásd még: ob_start(), ob_clean() és ob_end_flush().

ob_end_flush (PHP 4)

A kimeneti puffer ürítése (kiküldése), és a kimeneti pufferelés lekapcsolása

void **ob_end_flush** (void) \linebreak

E függvény elküldi a kimeneti puffer tartamát a kimenetre, majd kikapcsolja a pufferelést. Ha szükség lenne még a puffer tartalmára esetleges későbbi műveletekhez, az ob_get_contents() függvény segítségével tudjuk változóba menteni azt, mielőtt még az **ob_end_flush()** meghívásra kerülne.

Lásd még: ob_start(), ob_get_contents(), ob_flush() és ob_end_clean().

ob_flush (PHP 4 >= 4.2.0)

A kimeneti puffer ürítése (kiküldése)

void **ob_flush** (void) \linebreak

E függvény elküldi a kimeneti puffer tartamát a kimenetre, Ha szükség lenne még a puffer tartalmára esetleges későbbi műveletekhez, az ob_get_contents() függvény segítségével tudod változóba menteni azt, mielőtt még az **ob_flush()** meghívásra kerülne.

Ezt alkalmazva nem szűnik meg a kimeneti puffer, az ob_end_flush() függvénnyel ellentétben.

Lásd még: ob_get_contents(), ob_clean(), ob_end_flush(), és ob_end_clean().

ob_get_contents (PHP 4)

A kimeneti puffer tartalmának kinyerése

string **ob_get_contents** (void) \linebreak

A kimeneti puffer tartalmát adja vissza, vagy FALSE értéket, ha a kimeneti puffer mechanizmus nem aktív.

Lásd még: ob_start() és ob_get_length().

ob_get_length (PHP 4 >= 4.0.2)

A kimeneti puffer aktuális méretének kiolvasása

string **ob_get_length** (void) \linebreak

Ezzel a függvénnyel tudjuk megállapítani, milyen méretű jelenleg a kimeneti puffer, azaz mennyi kiküldésre váró adat van benne. Ha nem aktív a kimeneti pufferelés a visszatérési érték FALSE lesz.

Lásd még: ob_start() és ob_get_contents().

ob_get_level (PHP 4 >= 4.2.0)

Visszaadja a vizsgált puffer egymásba ágyazottsági szintjét.

int **ob_get_level** (void) \linebreak

Több, egymásba ágyazott pufferelést is aktiválhatunk egy php oldalon belül. Ekkor ezzel a függvénnyel tudjuk egy pufferől megállapítani annak beágyazottsági mélységét.

Lásd még: ob_start() és ob_get_contents().

ob_get_status (PHP 4 >= 4.2.0)

Get status of output buffers

array **ob_get_status** ([bool full_status]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

This will return the current status of output buffers. It returns array contains buffer status or FALSE for error.

See also ob_get_level().

ob_gzhandler (PHP 4 >= 4.0.4)

ob_start függvényhez használatos, kimenet gzip-elő függvény

string **ob_gzhandler** (string puffer [, int mód]) \linebreak

Megjegyzés: A *mód* paraméter a 4.0.5-ös PHP verziótól felfele használható.

Az **ob_gzhandler()** függvény arra lett létrehozva, hogy az ob_start() második paramétereként megadva megpróbálkozzon gzip tömörítettre alakítva továbbítani a kimeneti puffert. Próbálkozásról azért van csak szó, mert ezt a kódolást csak akkor hajtja végre, ha a böngésző elfogadja ezt a formátumot. Ezt az információt HTTP kérés fejlécében küldi a böngésző, "Accept-Encoding" név alatt. Ha van ilyen információ a böngészőről, és ennek értéke "gzip" vagy "deflate", ennek megfelelően küldi a hozzá való "Content-Type" fejlécet és a kódolt weblapot. Ez böngészőfüggetlen meoldás, hisz a böngésző maga közli, hogy hajlandó-e ilyen formában adatot fogadni, vagy sem.

Példa 1. ob_gzhandler() Példa

```
<?php

ob_start("ob_gzhandler");

?>
<html>
<body>
<p>Ez egy tömörített oldal lesz (talán). :)
</html>
</body>
```

Lásd még: ob_start() és ob_end_flush().

ob_implicit_flush (PHP 4)

Az implicit pufferürítés ki-be kapcsolása

void **ob_implicit_flush** ([int flag]) \linebreak

Az **ob_implicit_flush()** segítségével kapcsolható be vagy ki az implicit pufferürítés. Az itt *`flag`* néven nevezett opcionális paraméter elhagyása esetén a függvény hívása bekapcsolást jelent, amúgy értéke "on" vagy "off" kell legyen. Ha az implicit pufferürítés be van kapcsolva, minden olyan mívelet után amely kimenetet generál, meghívódik a flush() függvény, ezáltal a webszerver minden PHP kimeneti parancs után továbbítani fogja azonnal a böngésző fele az adott dokumentumrészletet. Ekkor a flush() hívogatására természetszerűleg nincs szükség.

Ha bekapcsoljuk ezt az implicit pufferürítési opciót aktív kimeneti pufferelés közben, tehát már volt meghívva ob_start() függvény, ez a kimeneti puffer azonnali kiküldésre kerül. Emellett a pufferelés le is kapcsolódik, mintha egy ob_end_flush() függvényhívást alkalmaztunk volna.

Lásd még: flush(), ob_start(), és ob_end_flush().

ob_start (PHP 4)

A kimenet pufferelés bekapcsolása

void **ob_start** ([string output_callback]) \linebreak

Ezzel a függvénnyel kapcsolhatjuk be a PHP saját belső kimenet pufferelési mechanizmusát. Ameddig a kimenet pufferelés be van kapcsolva, a fejlécadatokon kívül semmiféle kimenet nem hagyja el a PHP háza tájékát, az egy belső pufferben tárolódik.

Ennek a puffernek a tartalma az ob_get_contents() függvény segítségével változóba másolható. Ha végül szeretnénk a puffer tartalmát kiadni a php kezéből, akkor jön szóba az ob_end_flush(). Ha valamiért menet közben úgy döntünk, a puffer tartalma mégis felesleges, az ob_end_clean()

csendben és titokban megsemmisíti a pufferünket tartalmastól, nem hagyja el egy bájt sem ekkor a php-t.

Az itt *output_callback* néven nevezett második, opcionális paraméter segítségével megadhatunk egy függvényt, melynek egy string típusú adatot kell paraméterként fogadnia, és ugyancsak stringet kell visszaadnia. Ez a megadható függvény akkor hívódik meg, amikor az ob_end_flush() meghívásra kerül, vagy a script végeztével a kimeneti puffer ürítésekor, automatikusan. Ekkor megkapja ez az *output_callback* paraméterben megadott függvény a puffer tartalmát, ezzel eljátszhadozhat, majd visszaadja a játszadozás eredményét, és ez kerül végül a php kimenetére.

> **Megjegyzés:** A 4.0.4-es PHP verziótól felfele létezik egy ob_gzhandler() nevű beépített php függvény, mely segítségével gz tömörített formában küldhetjük el adatainkat a böngésző fele. Az ob_gzhandler() erre csak akkor vetemedik, ha úgy találja, a böngésző ezt a gzip kódolt formátumot elfogadja. Ezt a függvényt tehát az **ob_start()** második paramétereként megadva használhatjuk fel.

A pufferek egymásba ágyazhatóak, azaz miközben már van aktív kimeneti pufferünk, hozhatunk létre még egyet. Viszont ez esetben biztosnak kell lennünk abban, hogy az ob_end_flush() megfelelő számban meg legyen hívva. Ha több *output_callback* is meg van ilyenkor adva, mindegyik pufferhez másfajta, akkor azok mind sorban végrehajtódnak az egymásba ágyazottság sorrendjében.

Az ob_end_clean(), ob_end_flush(), ob_clean(), ob_flush() és **ob_start()** nem hívható "callback" függvényekből. Ha callback függvényből hívod ezeket, a viselkedés nem definiált. Ha törölni szeretnéd egy buffer tartalmát, ad vissza "" (üres karaktersorozatot) a callback függvényből.

Példa 1. Felhasználó által írt puffermódosító függvény használata

```
<?php

function callback($buffer) {

  // minden sört borra cserélünk
  return (ereg_replace("Sörre", "Borra", $buffer));

}

ob_start("callback");

?>
<html>
<body>
<p>Sörre bor, mindenkor!
</body>
</html>

<?php

ob_end_flush();

?>
```

A következő kimenet jön létre:

```
<html>
<body>
<p>Borra bor, mindenkor!
</body>
</html>
```

Lásd még: ob_get_contents(), ob_end_flush(), ob_end_clean(), ob_implicit_flush() és ob_gzhandler().

LXXV. Object property and method call overloading

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

The purpose of this extension is to allow overloading of object property access and method calls. Only one function is defined in this extension, overload() which takes the name of the class that should have this functionality enabled. The class named has to define appropriate methods if it wants to have this functionality: `__get()`, `__set()` and `__call()` respectively for getting/setting a property, or calling a method. This way overloading can be selective. Inside these handler functions the overloading is disabled so you can access object properties normally.

Some simple examples on using the overload() function:

Példa 1. Overloading a PHP class

```
<?php

class OO
{
    var $a = 111;
    var $elem = array('b' => 9, 'c' => 42);

    // Callback method for getting a property
    function __get($prop_name, &$prop_value)
    {
        if (isset($this->elem[$prop_name])) {
            $prop_value = $this->elem[$prop_name];
            return true;
        } else {
            return false;
        }
    }

    // Callback method for setting a property
    function __set($prop_name, $prop_value)
    {
        $this->elem[$prop_name] = $prop_value;
        return true;
    }
}

// Here we overload the OO object
overload('OO');

$o = new OO;
print "\$o->a: $o->a\n"; // print: $o->a:
```

```
print "\$o->b: $o->b\n"; // print: $o->b: 9
print "\$o->c: $o->c\n"; // print: $o->c: 42
print "\$o->d: $o->d\n"; // print: $o->d:

// add a new item to the $elem array in OO
$o->x = 56;

// instantiate stdclass (it is built-in in PHP 4)
// $val is not overloaded!
$val = new stdclass;
$val->prop = 555;

// Set "a" to be an array with the $val object in it
// But __set() will put this in the $elem array
$o->a = array($val);
var_dump($o->a[0]->prop);

?>
```

Figyelem

As this is an experimental extension, not all things work. There is no `__call()` support currently, you can only overload the get and set operations for properties. You cannot invoke the original overloading handlers of the class, and `__set()` only works to one level of property access.

overload (PHP 4 >= 4.2.0)

Enable property and method call overloading for a class

void **overload** ([string class_name]) \linebreak

The **overload()** function will enable property and method call overloading for a class identified by *`class_name`*. See an example in the introductory section of this part.

LXXVI. PDF functions

Introduction

The PDF functions in PHP can create PDF files using the PDFlib library created by Thomas Merz (http://www.pdflib.com/corporate/tm.html). PDFlib is available for download at http://www.pdflib.com/pdflib/index.html, but requires that you purchase a license for commercial use. The JPEG (ftp://ftp.uu.net/graphics/jpeg/) and TIFF (http://www.libtiff.org/) libraries are required to compile this extension. Please see the PDFlib installation section for more information about compiling PDF support into PHP.

The documentation in this section is only meant to be an overview of the available functions in the PDFlib library and should not be considered an exhaustive reference. Please consult the documentation included in the source distribution of PDFlib for the full and detailed explanation of each function here. It provides a very good overview of what PDFlib is capable of doing and contains the most up-to-date documentation of all functions.

All of the functions in PDFlib and the PHP module have identical function names and parameters. You will need to understand some of the basic concepts of PDF and PostScript to efficiently use this extension. All lengths and coordinates are measured in PostScript points. There are generally 72 PostScript points to an inch, but this depends on the output resolution. Please see the PDFlib documentation included with the source distribution of PDFlib for a more thorough explanation of the coordinate system used.

Please note that most of the PDF functions require a *`pdf object`* as it's first parameter. Please see the examples below for more information.

> **Megjegyzés:** An alternative PHP module for PDF document creation based on FastIO's (http://www.fastio.com/) ClibPDF is available. Please see the ClibPDF section for details. Note that ClibPDF has a slightly different API compared to PDFlib.

Confusion with old PDFlib versions

Starting with PHP 4.0.5, the PHP extension for PDFlib is officially supported by PDFlib GmbH. This means that all the functions described in the PDFlib manual (V3.00 or greater) are supported by PHP 4 with exactly the same meaning and the same parameters. Only the return values may differ from the PDFlib manual, because the PHP convention of returning `FALSE` was adopted. For compatibility reasons this binding for PDFlib still supports the old functions, but they should be replaced by their new versions. PDFlib GmbH will not support any problems arising from the use of these deprecated functions.

Táblázat 1. Deprecated functions and its replacements

Old function	Replacement
pdf_put_image()	Not needed anymore.
pdf_execute_image()	Not needed anymore.
pdf_get_annotation()	**pdf_get_bookmark()** using the same parameters.

Old function	Replacement
pdf_get_font()	pdf_get_value() passing "font" as the second parameter.
pdf_get_fontsize()	pdf_get_value() passing "fontsize" as the second parameter.
pdf_get_fontname()	pdf_get_parameter() passing "fontname" as the second parameter.
pdf_set_info_creator()	pdf_set_info() passing "Creator" as the second parameter.
pdf_set_info_title()	pdf_set_info() passing "Title" as the second parameter.
pdf_set_info_subject()	pdf_set_info() passing "Subject" as the second parameter.
pdf_set_info_author()	pdf_set_info() passing "Author" as the second parameter.
pdf_set_info_keywords()	pdf_set_info() passing "Keywords" as the second parameter.
pdf_set_leading()	pdf_set_value() passing "leading" as the second parameter.
pdf_set_text_rendering()	pdf_set_value() passing "textrendering" as the second parameter.
pdf_set_text_rise()	pdf_set_value() passing "textrise" as the second parameter.
pdf_set_horiz_scaling()	pdf_set_value() passing "horizscaling" as the second parameter.
pdf_set_text_matrix()	Not available anymore
pdf_set_char_spacing()	pdf_set_value() passing "charspacing" as the second parameter.
pdf_set_word_spacing()	pdf_set_value() passing "wordspacing" as the second parameter.
pdf_set_transition()	pdf_set_parameter() passing "transition" as the second parameter.
pdf_open()	pdf_new() plus an subsequent call of pdf_open_file()
pdf_set_font()	pdf_findfont() plus an subsequent call of pdf_setfont()
pdf_set_duration()	pdf_set_value() passing "duration" as the second parameter.
pdf_open_gif()	pdf_open_image_file() passing "gif" as the second parameter.
pdf_open_jpeg()	pdf_open_image_file() passing "jpeg" as the second parameter.
pdf_open_tiff()	pdf_open_image_file() passing "tiff" as the second parameter.
pdf_open_png()	pdf_open_image_file() passing "png" as the second parameter.

Old function	Replacement
pdf_get_image_width()	pdf_get_value() passing `"imagewidth"` as the second parameter and the image as the third parameter.
pdf_get_image_height()	pdf_get_value() passing `"imageheight"` as the second parameter and the image as the third parameter.

PDFlib 3.x Installation Hints

When using version 3.x of PDFlib, you should configure PDFlib with the option `--enable-shared-pdflib`.

Issues with older versions of PDFlib

Any version of PHP 4 after March 9, 2000 does not support versions of PDFlib older than 3.0.

PDFlib 3.0 or greater is supported by PHP 3.0.19 and later.

Examples

Most of the functions are fairly easy to use. The most difficult part is probably creating a very simple PDF document at all. The following example should help to get started. It creates `test.pdf` with one page. The page contains the text "Times Roman outlined" in an outlined, 30pt font. The text is also underlined.

Példa 1. Creating a PDF document with PDFlib

```
<?php
$pdf = pdf_new();
pdf_open_file($pdf, "test.pdf");
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Test for PHP wrapper of PDFlib 2.0");
pdf_set_info($pdf, "Creator", "See Author");
pdf_set_info($pdf, "Subject", "Testing");
pdf_begin_page($pdf, 595, 842);
pdf_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
pdf_end_page($pdf);
pdf_close($pdf);
pdf_delete($pdf);
```

```
echo "<A HREF=getpdf.php>finished</A>";
?>
```

The script `getpdf.php` just returns the pdf document.

```
<?php
$len = filesize($filename);
header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=foo.pdf");
readfile($filename);
?>
```

The PDFlib distribution contains a more complex example which creates a page with an analog clock. Here we use the in memory creation feature of PDFlib to alleviate the need to use temporary files. The example, converted to PHP from the PDFlib example, is as follows: (The same example is available in the CLibPDF documentation.)

Példa 2. pdfclock example from PDFlib distribution

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 10;

$pdf = pdf_new();

if (!pdf_open_file($pdf, "")) {
    print error;
    exit;
};

pdf_set_parameter($pdf, "warning", "true");

pdf_set_info($pdf, "Creator", "pdf_clock.php");
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Analog Clock");

while($pagecount-- > 0) {
    pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));

    pdf_set_parameter($pdf, "transition", "wipe");
    pdf_set_value($pdf, "duration", 0.5);

    pdf_translate($pdf, $radius + $margin, $radius + $margin);
    pdf_save($pdf);
    pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);
```

```
/* minute strokes */
pdf_setlinewidth($pdf, 2.0);
for ($alpha = 0; $alpha < 360; $alpha += 6) {
    pdf_rotate($pdf, 6.0);
    pdf_moveto($pdf, $radius, 0.0);
    pdf_lineto($pdf, $radius-$margin/3, 0.0);
    pdf_stroke($pdf);
}

pdf_restore($pdf);
pdf_save($pdf);

/* 5 minute strokes */
pdf_setlinewidth($pdf, 3.0);
for ($alpha = 0; $alpha < 360; $alpha += 30) {
    pdf_rotate($pdf, 30.0);
    pdf_moveto($pdf, $radius, 0.0);
    pdf_lineto($pdf, $radius-$margin, 0.0);
    pdf_stroke($pdf);
}

$ltime = getdate();

/* draw hour hand */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['minutes']/60.0)+$ltime['hours']-3.0)*30.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius/2, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* draw minute hand */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['seconds']/60.0)+$ltime['minutes']-15.0)*6.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius * 0.8, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* draw second hand */
pdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
pdf_setlinewidth($pdf, 2);
pdf_save($pdf);
pdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
pdf_moveto($pdf, -$radius/5, 0.0);
pdf_lineto($pdf, $radius, 0.0);
pdf_stroke($pdf);
pdf_restore($pdf);

/* draw little circle at center */
pdf_circle($pdf, 0, 0, $radius/30);
pdf_fill($pdf);
```

```
    pdf_restore($pdf);

    pdf_end_page($pdf);

    # to see some difference
    sleep(1);
}

pdf_close($pdf);

$buf = pdf_get_buffer($pdf);
$len = strlen($buf);

header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=foo.pdf");
print $buf;

pdf_delete($pdf);
?>
```

pdf_add_annotation (PHP 3>= 3.0.12, PHP 4)

Deprecated: Adds annotation

pdf_add_outline() is replaced by pdf_add_note()

See also pdf_add_note().

pdf_add_bookmark (PHP 4 >= 4.0.1)

Adds bookmark for current page

int **pdf_add_bookmark** (int pdf object, string text [, int parent [, int open]]) \linebreak

Add a nested bookmark under *parent*, or a new top-level bookmark if *parent* = 0. Returns a bookmark descriptor which may be used as parent for subsequent nested bookmarks. If open = 1, child bookmarks will be folded out, and invisible if open = 0.

pdf_add_launchlink (PHP 4 >= 4.0.5)

Add a launch annotation for current page

int **pdf_add_launchlink** (int pdf object, float llx, float lly, float urx, float ury, string filename) \linebreak

Add a launch annotation (to a target of arbitrary file type).

pdf_add_locallink (PHP 4 >= 4.0.5)

Add a link annotation for current page

int **pdf_add_locallink** (int pdf object, float llx, float lly, float urx, float ury, int page, string dest) \linebreak

Add a link annotation to a target within the current PDF file.

pdf_add_note (PHP 4 >= 4.0.5)

Add a note annotation for current page

int **pdf_add_note** (int pdf object, float llx, float lly, float urx, float ury, string contents, string title, string icon, int open) \linebreak

Add a note annotation. icon is one of of "comment, "insert", "note", "paragraph", "newparagraph", "key", or "help".

pdf_add_outline (PHP 3>= 3.0.6, PHP 4)

Deprecated: Adds bookmark for current page

Deprecated.

See pdf_add_bookmark().

pdf_add_pdflink (PHP 3>= 3.0.12, PHP 4)

Adds file link annotation for current page

int **pdf_add_pdflink** (int pdf object, float llx, float lly, float urx, float ury, string filename, int page, string dest) \linebreak

Add a file link annotation (to a PDF target).

pdf_add_thumbnail (PHP 4 >= 4.0.5)

Adds thumbnail for current page

int **pdf_add_thumbnail** (int pdf object, int image) \linebreak

Add an existing image as thumbnail for the current page.

pdf_add_weblink (PHP 3>= 3.0.12, PHP 4)

Adds weblink for current page

int **pdf_add_weblink** (int pdf object, float llx, float lly, float urx, float ury, string url) \linebreak

Add a weblink annotation to a target URL on the Web.

pdf_arc (PHP 3>= 3.0.6, PHP 4)

Draws an arc (counterclockwise)

void **pdf_arc** (resource pdf object, float x, float y, float r, float alpha, float beta) \linebreak

Draw a counterclockwise circular arc from alpha to beta degrees

See also: pdf_arcn()

pdf_arcn (PHP 4 >= 4.0.5)

Draws an arc (clockwise)

void **pdf_arc** (resource pdf object, float x, float y, float r, float alpha, float beta) \linebreak

Draw a clockwise circular arc from alpha to beta degrees

See also: pdf_arc()

pdf_attach_file (PHP 4 >= 4.0.5)

Adds a file attachement for current page

int **pdf_attach_file** (int pdf object, float llx, float lly, float urx, float ury, string filename, string description, string author, string mimetype, string icon) \linebreak

Add a file attachment annotation. icon is one of "graph, "paperclip", "pushpin", or "tag".

pdf_begin_page (PHP 3>= 3.0.6, PHP 4)

Starts new page

void **pdf_begin_page** (int pdf object, float width, float height) \linebreak

Add a new page to the document. The *width* and *height* are specified in points, which are 1/72 of an inch.

Táblázat 1. Common Page Sizes in Points

name	size
A0	2380✗3368
A1	1684✗2380
A2	1190✗1684
A3	842✗1190
A4	595✗842
A5	421✗595
A6	297✗421
B5	501✗709
letter (8.5"✗11")	612✗792
legal (8.5"✗14")	612✗1008
ledger (17"✗11")	1224✗792
11"✗17"	792✗1224

pdf_begin_pattern (PHP 4 >= 4.0.5)

Starts new pattern

int **pdf_begin_pattern** (int pdf object, float width, float height, float xstep, float ystep, int painttype) \linebreak

Starts a new pattern definition and returns a pattern handle. *width*, and *height* define the bounding box for the pattern. *xstep* and *ystep* give the repeated pattern offsets. *painttype*=1 means that the pattern has its own colour settings whereas a value of 2 indicates that the current colour is used when the pattern is applied.

pdf_begin_template (PHP 4 >= 4.0.5)

Starts new template

void **pdf_begin_template** (int pdf object, float width, float height) \linebreak

Start a new template definition.

pdf_circle (PHP 3>= 3.0.6, PHP 4)

Draws a circle

void **pdf_circle** (int pdf object, float x, float y, float r) \linebreak

Draw a circle with center (x, y) and radius r.

pdf_clip (PHP 3>= 3.0.6, PHP 4)

Clips to current path

void **pdf_clip** (int pdf object) \linebreak

Use the current path as clipping path.

pdf_close_image (PHP 3>= 3.0.7, PHP 4)

Closes an image

void **pdf_close_image** (int pdf object, int image) \linebreak

Close an *image* retrieved with one of the **pdf_open_image*()** functions.

pdf_close_pdi_page (PHP 4 >= 4.0.5)

Close the page handle

void **pdf_close_pdi_page** (int pdf object, int pagehandle) \linebreak

Close the page handle, and free all page-related resources.

pdf_close_pdi (PHP 4 >= 4.0.5)

Close the input PDF document

void **pdf_close_pdi** (int pdf object, int dochandle) \linebreak

Close all open page handles, and close the input PDF document.

pdf_close (PHP 3>= 3.0.6, PHP 4)

Closes a pdf object

void **pdf_close** (int pdf object) \linebreak

Close the generated PDF file, and free all document-related resources.

pdf_closepath_fill_stroke (PHP 3>= 3.0.6, PHP 4)

Closes, fills and strokes current path

void **pdf_closepath_fill_stroke** (int pdf object) \linebreak

Close the path, fill, and stroke it.

pdf_closepath_stroke (PHP 3>= 3.0.6, PHP 4)

Closes path and draws line along path

void **pdf_closepath_stroke** (int pdf object) \linebreak

Close the path, and stroke it.

pdf_closepath (PHP 3>= 3.0.6, PHP 4)

Closes path

void **pdf_closepath** (int pdf object) \linebreak

Close the current path.

pdf_concat (PHP 4 >= 4.0.5)

Concatenate a matrix to the CTM

void **pdf_concat** (int pdf object, float a, float b, float c, float d, float e, float f) \linebreak

Concatenate a matrix to the CTM.

pdf_continue_text (PHP 3>= 3.0.6, PHP 4)

Outputs text in next line

void **pdf_continue_text** (int pdf object, string text) \linebreak

Print text at the next line. The spacing between lines is determined by the `leading` parameter.

pdf_curveto (PHP 3>= 3.0.6, PHP 4)

Draws a curve

void **pdf_curveto** (int pdf object, float x1, float y1, float x2, float y2, float x3, float y3) \linebreak

Draw a Bezier curve from the current point, using 3 more control points.

pdf_delete (PHP 4 >= 4.0.5)

Deletes a PDF object

void **pdf_delete** (int pdf object) \linebreak

Delete the PDF object, and free all internal resources.

pdf_end_page (PHP 3>= 3.0.6, PHP 4)

Ends a page

void **pdf_end_page** (int pdf object) \linebreak

Finish the page.

pdf_end_pattern (PHP 4 >= 4.0.5)

Finish pattern

void **pdf_end_pattern** (int pdf object) \linebreak

Finish the pattern definition.

pdf_end_template (PHP 4 >= 4.0.5)

Finish template

void **pdf_end_template** (int pdf object) \linebreak

Finish the template definition.

pdf_endpath (PHP 3>= 3.0.6, PHP 4)

Deprecated: Ends current path

Deprecated, use one of the stroke, fill, or clip functions instead.

pdf_fill_stroke (PHP 3>= 3.0.6, PHP 4)

Fills and strokes current path

void **pdf_fill_stroke** (int pdf object) \linebreak

Fill and stroke the path with the current fill and stroke color.

pdf_fill (PHP 3>= 3.0.6, PHP 4)

Fills current path

void **pdf_fill_stroke** (int pdf object) \linebreak

Fill the interior of the path with the current fill color.

pdf_findfont (PHP 4 >= 4.0.5)

Prepare font for later use with pdf_setfont().

int **pdf_findfont** (int pdf object, string fontname, string encoding, int embed) \linebreak

Prepare a font for later use with pdf_setfont(). The metrics will be loaded, and if embed is nonzero, the font file will be checked, but not yet used. *encoding* is one of "builtin", "macroman", "winansi", "host", or a user-defined encoding name, or the name of a CMap.

pdf_findfont() returns a font handle or FALSE on error.

Példa 1. pdf_findfont() example

```
<?php

$font = pdf_findfont($pdf, "Times New Roman", "winansi", 1);
if ($font) {
    pdf_setfont($pdf, $font, 10);
}

?>
```

pdf_get_buffer (PHP 4 >= 4.0.5)

Fetch the buffer containig the generated PDF data.

string **pdf_get_buffer** (int pdf object) \linebreak

Get the contents of the PDF output buffer. The result must be used by the client before calling any other PDFlib function.

pdf_get_font (PHP 4)

Deprecated: font handling

Deprecated.

See pdf_get_value().

pdf_get_fontname (PHP 4)

Deprecated: font handling

Deprecated.

See pdf_get_parameter().

pdf_get_fontsize (PHP 4)

Deprecated: font handling

Deprecated.

See pdf_get_value().

pdf_get_image_height (PHP 3>= 3.0.12, PHP 4)

Returns height of an image

string **pdf_get_image_height** (int pdf object, int image) \linebreak

pdf_get_image_height() is deprecated, use pdf_get_value() instead.

pdf_get_image_width (PHP 3>= 3.0.12, PHP 4)

Returns width of an image

string **pdf_get_image_width** (int pdf object, int image) \linebreak

The **pdf_get_image_width()** is deprecated, use pdf_get_value() instead.

pdf_get_majorversion (PHP 4 >= 4.2.0)

Returns the major version number of the PDFlib

int **pdf_get_majorversion** (void) \linebreak

Returns the major version number of the PDFlib.

pdf_get_minorversion (PHP 4 >= 4.2.0)

Returns the minor version number of the PDFlib

int **pdf_get_majorversion** (void) \linebreak

Returns the minor version number of the PDFlib.

pdf_get_parameter (PHP 4 >= 4.0.1)

Gets certain parameters

string **pdf_get_parameter** (int pdf object, string key [, float modifier]) \linebreak

Get the contents of some PDFlib parameter with string type.

pdf_get_pdi_parameter (PHP 4 >= 4.0.5)

Get some PDI string parameters

string **pdf_get_pdi_parameter** (int pdf object, string key, int doc, int page, int index) \linebreak

Get the contents of some PDI document parameter with string type.

pdf_get_pdi_value (PHP 4 >= 4.0.5)

Gets some PDI numerical parameters

string **pdf_get_pdi_value** (int pdf object, string key, int doc, int page, int index) \linebreak

Get the contents of some PDI document parameter with numerical type.

pdf_get_value (PHP 4 >= 4.0.1)

Gets certain numerical value

float **pdf_get_value** (int pdf object, string key [, float modifier]) \linebreak

Get the contents of some PDFlib parameter with float type.

pdf_initgraphics (PHP 4 >= 4.0.5)

Resets graphic state

void **pdf_initgraphics** (int pdf object) \linebreak

Reset all implicit color and graphics state parameters to their defaults.

pdf_lineto (PHP 3>= 3.0.6, PHP 4)

Draws a line

void **pdf_lineto** (int pdf object, float x, float y) \linebreak

Draw a line from the current point to (x, y).

pdf_makespotcolor (PHP 4 >= 4.0.5)

Makes a spotcolor

void **pdf_makespotcolor** (int pdf object, string spotname) \linebreak

Make a named spot color from the current color.

pdf_moveto (PHP 3>= 3.0.6, PHP 4)

Sets current point

void **pdf_moveto** (int pdf object, float x, float y) \linebreak

Set the current point.

> **Megjegyzés:** The current point for graphics and the current text output position are maintained separately. See pdf_set_text_pos() to set the text output position.

pdf_new (PHP 4 >= 4.0.5)

Creates a new pdf object

int **pdf_new** ( ) \linebreak

Create a new PDF object, using default error handling and memory management.

pdf_open_CCITT (PHP 4 >= 4.0.5)

Opens a new image file with raw CCITT data

int **pdf_open_CCITT** (int pdf object, string filename, int width, int height, int BitReverse, int k, int Blackls1) \linebreak

Open a raw CCITT image.

pdf_open_file (PHP 4 >= 4.0.5)

Opens a new pdf object

int **pdf_open_file** (int pdf object [, string filename]) \linebreak

Create a new PDF file using the supplied file name. If *`filename`* is empty the PDF document will be generated in memory instead of on file. The result must be fetched by the client with the pdf_get_buffer() function.

The following example shows how to create a pdf document in memory and how to output it correctly.

Példa 1. Creating a PDF document in memory

```
<?php

$pdf = pdf_new();

pdf_open_file($pdf);
pdf_begin_page($pdf, 595, 842);
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "A PDF document created in memory!", 50, 750);
pdf_end_page($pdf);
pdf_close($pdf);

$data = pdf_get_buffer($pdf);

header("Content-type: application/pdf");
header("Content-disposition: inline; filename=test.pdf");
header("Content-length: " . strlen($data));

echo $data;

?>
```

pdf_open_gif (PHP 3>= 3.0.7, PHP 4)

Deprecated: Opens a GIF image

Deprecated.

See pdf_open_image(),

pdf_open_image_file (PHP 3 CVS only, PHP 4)

Reads an image from a file

int **pdf_open_image_file** (int PDF-document, string imagetype, string filename [, string stringparam [, string intparam]]) \linebreak

Open an image file. Supported types are "jpeg", "tiff", "gif", and "png". *`stringparam`* is either "", "mask", "masked", or "page". *`intparam`*is either 0, the image id of the applied mask, or the page.

pdf_open_image (PHP 4 >= 4.0.5)

Versatile function for images

int **pdf_open_image** (int PDF-document, string imagetype, string source, string data, long length, int width, int height, int components, int bpc, string params) \linebreak

Use image data from a variety of data sources. Supported types are "jpeg", "ccitt", "raw". Supported sources are "memory", "fileref", "url". len is only used for type="raw", params is only used for type="ccitt".

pdf_open_jpeg (PHP 3>= 3.0.7, PHP 4)

Deprecated: Opens a JPEG image

Deprecated.

See also pdf_open_image(),

pdf_open_memory_image (PHP 3>= 3.0.10, PHP 4)

Opens an image created with PHP's image functions

int **pdf_open_memory_image** (int pdf object, int image) \linebreak

The **pdf_open_memory_image()** function takes an image created with the PHP's image functions and makes it available for the pdf object. The function returns a pdf image identifier.

Példa 1. Including a memory image

```
<?php
$im = ImageCreate(100, 100);
$col = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col);
$pim = pdf_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
?>
```

See also pdf_close_image(), pdf_place_image().

pdf_open_pdi_page (PHP 4 >= 4.0.5)

Prepare a page

int **pdf_open_pdi_page** (int pdf object, int dochandle, int pagenumber, string pagelabel) \linebreak

Prepare a page for later use with pdf_place_image()

pdf_open_pdi (PHP 4 >= 4.0.5)

Opens a PDF file

int **pdf_open_pdi** (int pdf object, string filename, string stringparam, int intparam) \linebreak

Open an existing PDF document for later use.

pdf_open_png (PHP 4)

Deprecated: Opens a PNG image

Deprecated.

See pdf_open_image().

pdf_open_tiff (PHP 4)

Deprecated: Opens a TIFF image

int **pdf_open_tiff** (int PDF-document, string filename) \linebreak

Deprecated.

See also pdf_open_image(),

pdf_open (PHP 3>= 3.0.6, PHP 4)

Deprecated: Open a new pdf object

pdf_open() is deprecated, use pdf_new() plus pdf_open_file() instead.

See also pdf_new(), pdf_open_file().

pdf_place_image (PHP 3>= 3.0.7, PHP 4)

Places an image on the page

void **pdf_place_image** (int pdf object, int image, float x, float y, float scale) \linebreak

Place an image with the lower left corner at (x, y), and scale it.

pdf_place_pdi_page (PHP 4 >= 4.0.6)

Places an image on the page

void **pdf_place_pdi_page** (int pdf object, int page, float x, float y, float sx, float sy) \linebreak

Place a PDF page with the lower left corner at (x, y), and scale it.

pdf_rect (PHP 3>= 3.0.6, PHP 4)

Draws a rectangle

void **pdf_rect** (int pdf object, float x, float y, float width, float height) \linebreak

Draw a rectangle at lower left (x, y) with width and height.

pdf_restore (PHP 3>= 3.0.6, PHP 4)

Restores formerly saved environment

void **pdf_restore** (int pdf object) \linebreak

Restore the most recently saved graphics state.

pdf_rotate (PHP 3>= 3.0.6, PHP 4)

Sets rotation

void **pdf_rotate** (int pdf object, float phi) \linebreak

Rotate the coordinate system by phi degrees.

pdf_save (PHP 3>= 3.0.6, PHP 4)

Saves the current environment

void **pdf_save** (int pdf object) \linebreak

Save the current graphics state.

pdf_scale (PHP 3>= 3.0.6, PHP 4)

Sets scaling

void **pdf_scale** (int pdf object, float x-scale, float y-scale) \linebreak

Scale the coordinate system.

pdf_set_border_color (PHP 3>= 3.0.12, PHP 4)

Sets color of border around links and annotations

void **pdf_set_border_color** (int pdf object, float red, float green, float blue) \linebreak

Set the border color for all kinds of annotations.

pdf_set_border_dash (PHP 4 >= 4.0.1)

Sets dash style of border around links and annotations

void **pdf_set_border_dash** (int pdf object, float black, float white) \linebreak

Set the border dash style for all kinds of annotations. See pdf_setdash().

pdf_set_border_style (PHP 3>= 3.0.12, PHP 4)

Sets style of border around links and annotations

void **pdf_set_border_style** (int pdf object, string style, float width) \linebreak

Set the border style for all kinds of annotations. *`style`* is "solid" or "dashed".

pdf_set_char_spacing (PHP 3>= 3.0.6, PHP 4)

Deprecated: Sets character spacing

Deprecated.

See also pdf_set_value(),

pdf_set_duration (PHP 3>= 3.0.6, PHP 4)

Deprecated: Sets duration between pages

Deprecated.
See pdf_set_value().

pdf_set_font (PHP 3>= 3.0.6, PHP 4)

Deprecated: Selects a font face and size

Deprecated. You should use pdf_findfont() plus pdf_setfont() instead.
See pdf_findfont(), pdf_setfont().

pdf_set_horiz_scaling (PHP 3>= 3.0.6, PHP 4)

Sets horizontal scaling of text

void **pdf_set_horiz_scaling** (int pdf object, float scale) \linebreak
Deprecated.
See also pdf_set_value(),

pdf_set_info_author (PHP 3>= 3.0.6, PHP 4)

Fills the author field of the document

bool **pdf_set_info_author** (int pdfdoc, string author) \linebreak
This function is deprecate, use pdf_set_info() instead.

pdf_set_info_creator (PHP 3>= 3.0.6, PHP 4)

Fills the creator field of the document

bool **pdf_set_info_creator** (int pdfdoc, string creator) \linebreak
This function is deprecate, use pdf_set_info() instead.

pdf_set_info_keywords (PHP 3>= 3.0.6, PHP 4)

Fills the keywords field of the document

bool **pdf_set_info_keywords** (int pdfdoc, string keywords) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info_subject (PHP 3>= 3.0.6, PHP 4)

Fills the subject field of the document

bool **pdf_set_info_subject** (int pdfdoc, string subject) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info_title (PHP 3>= 3.0.6, PHP 4)

Fills the title field of the document

bool **pdf_set_info_title** (int pdfdoc, string title) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info (PHP 4 >= 4.0.1)

Fills a field of the document information

void **pdf_set_info** (int pdf object, string key, string value) \linebreak

Fill document information field key with value. *key* is one of "Subject", "Title", "Creator", "Author", "Keywords", or a user-defined key.

pdf_set_leading (PHP 3>= 3.0.6, PHP 4)

Deprecated: Sets distance between text lines

Deprecated.

See also pdf_set_value(),

pdf_set_parameter (PHP 4)

Sets certain parameters

void **pdf_set_parameter** (int pdf object, string key, string value) \linebreak

Set some PDFlib parameter with string type.

pdf_set_text_matrix (PHP 3>= 3.0.6)

Deprecated: Sets the text matrix

See **pdf_set_paramter()**.

pdf_set_text_pos (PHP 3>= 3.0.6, PHP 4)

Sets text position

void **pdf_set_text_pos** (int pdf object, float x, float y) \linebreak

Set the text output position.

pdf_set_text_rendering (PHP 3>= 3.0.6, PHP 4)

Deprecated: Determines how text is rendered

Deprecated.

See pdf_set_value(),

pdf_set_text_rise (PHP 3>= 3.0.6, PHP 4)

Deprecated: Sets the text rise

Deprecated.

See pdf_set_value(),

pdf_set_value (PHP 4 >= 4.0.1)

Sets certain numerical value

void **pdf_set_value** (int pdf object, string key, float value) \linebreak

Set the value of some PDFlib parameter with float type.

pdf_set_word_spacing (PHP 3>= 3.0.6, PHP 4)

Depriciated: Sets spacing between words

Deprecated.

See also pdf_set_value(),

pdf_setcolor (PHP 4 >= 4.0.5)

Sets fill and stroke color

void **pdf_setcolor** (int pdf object, string type, string colorspace, float c1 [, float c2 [, float c3 [, float c4]]]) \linebreak

Set the current color space and color. The parameter *type* can be "fill", "stroke", or "both" to specify that the color is set for filling, stroking or both filling and stroking. The parameter *colorspace* can be `gray`, `rgb`, `cmyk`, `spot` or `pattern`. The parameters *c1*, *c2*, *c3* and *c4* represent the color components for the color space specified by *colorspace*. Except as otherwise noted, the color components are floating-point values that range from 0 to 1.

For `gray` only *c1* is used.

For `rgb` parameters *c1*, *c2*, and *c3* specify the red, green and blue values respectively.

```
// Set fill and stroke colors to white.
pdf_setcolor($pdf, "both", "rgb", 1, 1, 1);
```

For `cmyk`, parameters *c1*, *c2*, *c3*, and *c4* are the cyan, magenta, yellow and black values, respectively.

```
// Set fill and stroke colors to white.
pdf_setcolor($pdf, "both", "cmyk", 0, 0, 0, 1);
```

For `spot`, *c1* should be a spot color handles returned by pdf_makespotcolor() and *c2* is a tint value between 0 and 1.

For `pattern`, *c1* should be a pattern handle returned by pdf_begin_pattern().

pdf_setdash (PHP 3>= 3.0.6, PHP 4)

Sets dash pattern

void **pdf_setdash** (int pdf object, float b, float w) \linebreak

Set the current dash pattern to *b* black and *w* white units.

pdf_setflat (PHP 3>= 3.0.6, PHP 4)

Sets flatness

void **pdf_setflat** (int pdf object, float flatness) \linebreak

Set the flatness to a value between 0 and 100 inclusive.

pdf_setfont (PHP 4 >= 4.0.5)

Set the current font

void **pdf_setfont** (int pdf object, int font, float size) \linebreak

Set the current font in the given size, using a *font* handle returned by pdf_findfont()

See Also: pdf_findfont().

pdf_setgray_fill (PHP 3>= 3.0.6, PHP 4)

Sets filling color to gray value

void **pdf_setgray_fill** (int pdf object, float gray) \linebreak

Set the current fill color to a gray value between 0 and 1 inclusive.

Megjegyzés: PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_setgray_stroke (PHP 3>= 3.0.6, PHP 4)

Sets drawing color to gray value

void **pdf_setgray_stroke** (int pdf object, float gray) \linebreak

Set the current stroke color to a gray value between 0 and 1 inclusive

Megjegyzés: PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_setgray (PHP 3>= 3.0.6, PHP 4)

Sets drawing and filling color to gray value

void **pdf_setgray** (int pdf object, float gray) \linebreak

Set the current fill and stroke color.

Megjegyzés: PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_setlinecap (PHP 3>= 3.0.6, PHP 4)

Sets linecap parameter

void **pdf_setlinecap** (int pdf object, int linecap) \linebreak

Set the *linecap* parameter to a value between 0 and 2 inclusive.

pdf_setlinejoin (PHP 3>= 3.0.6, PHP 4)

Sets linejoin parameter

void **pdf_setlinejoin** (int pdf object, long linejoin) \linebreak

Set the line join parameter to a value between 0 and 2 inclusive.

pdf_setlinewidth (PHP 3>= 3.0.6, PHP 4)

Sets line width

void **pdf_setlinewidth** (int pdf object, float width) \linebreak

Set the current linewidth to width.

pdf_setmatrix (PHP 4 >= 4.0.5)

Sets current transformation matrix

void **pdf_setmatrix** (int pdf object, float a, float b, float c, float d, float e, float f) \linebreak

Explicitly set the current transformation matrix.

pdf_setmiterlimit (PHP 3>= 3.0.6, PHP 4)

Sets miter limit

void **pdf_setmiterlimit** (int pdf object, float miter) \linebreak

Set the miter limit to a value greater than or equal to 1.

pdf_setpolydash (PHP 4 >= 4.0.5)

Sets complicated dash pattern

void **pdf_setpolydash** (int pdf object, float * dasharray) \linebreak

Set a more complicated dash pattern defined by an array.

pdf_setrgbcolor_fill (PHP 3>= 3.0.6, PHP 4)

Sets filling color to rgb color value

void **pdf_setrgbcolor_fill** (int pdf object, float red value, float green value, float blue value) \linebreak

Set the current fill color to the supplied RGB values.

Megjegyzés: PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_setrgbcolor_stroke (PHP 3>= 3.0.6, PHP 4)

Sets drawing color to rgb color value

void **pdf_setrgbcolor_stroke** (int pdf object, float red value, float green value, float blue value) \linebreak

Set the current stroke color to the supplied RGB values.

Megjegyzés: PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_setrgbcolor (PHP 3>= 3.0.6, PHP 4)

Sets drawing and filling color to rgb color value

void **pdf_setrgbcolor** (int pdf object, float red value, float green value, float blue value) \linebreak

Set the current fill and stroke color to the supplied RGB values.

Megjegyzés: PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_show_boxed (PHP 4)

Output text in a box

int **pdf_show_boxed** (int pdf object, string text, float left, float top, float width, float height, string hmode [, string feature]) \linebreak

Format text in the current font and size into the supplied text box according to the requested formatting mode, which must be one of "left", "right", "center", "justify", or "fulljustify". If width and height are 0, only a single line is placed at the point (left, top) in the requested mode.

Returns the number of characters that did not fit in the specified box. Returns 0 if all characters fit or the *`width`* and *`height`* parameters were set to 0 for single-line formattting.

pdf_show_xy (PHP 3>= 3.0.6, PHP 4)

Output text at given position

void **pdf_show_xy** (int pdf object, string text, float x, float y) \linebreak

Print text in the current font at (x, y).

pdf_show (PHP 3>= 3.0.6, PHP 4)

Output text at current position

void **pdf_show** (int pdf object, string text) \linebreak

Print text in the current font and size at the current position.

pdf_skew (PHP 4)

Skews the coordinate system

void **pdf_skew** (int pdf object, float alpha, float beta) \linebreak

Skew the coordinate system in x and y direction by alpha and beta degrees.

pdf_stringwidth (PHP 3>= 3.0.6, PHP 4)

Returns width of text using current font

float **pdf_stringwidth** (int pdf object, string text [, int font [, float size]]) \linebreak

Returns the width of *text* using the last font set by pdf_setfont(). If the optional parameters *font* and *size* are specified, the width will be calculated using that font and size instead. Please note that *font* is a font handle returned by pdf_findfont().

Megjegyzés: Both the *font* and *size* parameters must used together.

See Also: pdf_setfont() and pdf_findfont().

pdf_stroke (PHP 3>= 3.0.6, PHP 4)

Draws line along path

void **pdf_stroke** (int pdf object) \linebreak

Stroke the path with the current color and line width, and clear it.

pdf_translate (PHP 3>= 3.0.6, PHP 4)

Sets origin of coordinate system

void **pdf_translate** (int pdf object, float tx, float ty) \linebreak

Translate the origin of the coordinate system.

LXXVII. Verisign Payflow Pro functions

This extension allows you to process credit cards and other financial transactions using Verisign Payment Services, formerly known as Signio (http://www.verisign.com/products/payflow/pro/index.html).

These functions are only available if PHP has been compiled with the `--with-pfpro[=DIR]` option. You will require the appropriate SDK for your platform, which may be downloaded from within the manager interface (https://manager.verisign.com/) once you have registered. If you are going to use this extension in an SSL-enabled webserver or with other SSL components (such as the CURL+SSL extension) you MUST get the beta SDK.

Once you have downloaded the SDK you should copy the files from the `lib` directory of the distribution. Copy the header file `pfpro.h` to `/usr/local/include` and the library file `libpfpro.so` to `/usr/local/lib`.

When using these functions, you may omit calls to pfpro_init() and pfpro_cleanup() as this extension will do so automatically if required. However the functions are still available in case you are processing a number of transactions and require fine control over the library. You may perform any number of transactions using pfpro_process() between the two.

These functions were added in PHP 4.0.2.

> **Megjegyzés:** These functions only provide a link to Verisign Payment Services. Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

pfpro_cleanup (PHP 4 >= 4.0.2)

Shuts down the Payflow Pro library

void **pfpro_cleanup** (void) \linebreak

pfpro_cleanup() is used to shutdown the Payflow Pro library cleanly. It should be called after you have processed any transactions and before the end of your script. However you may omit this call, in which case this extension will automatically call **pfpro_cleanup()** after your script terminates.

See also pfpro_init().

pfpro_init (PHP 4 >= 4.0.2)

Initialises the Payflow Pro library

void **pfpro_init** (void) \linebreak

pfpro_init() is used to initialise the Payflow Pro library. You may omit this call, in which case this extension will automatically call **pfpro_init()** before the first transaction.

See also pfpro_cleanup().

pfpro_process_raw (PHP 4 >= 4.0.2)

Process a raw transaction with Payflow Pro

string **pfpro_process_raw** (string parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]]) \linebreak

Returns: A string containing the response.

pfpro_process_raw() processes a raw transaction string with Payflow Pro. You should really use pfpro_process() instead, as the encoding rules of these transactions are non-standard.

The first parameter in this case is a string containing the raw transaction request. All other parameters are the same as with pfpro_process(). The return value is a string containing the raw response.

> **Megjegyzés:** Be sure to read the Payflow Pro Developers Guide for full details of the required parameters and encoding rules. You would be well advised to use pfpro_process() instead.

Példa 1. Payflow Pro raw example

```
<?php

pfpro_init();

$response = pfpro_process("USER=mylogin&PWD[5]=m&ndy&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=411
```

```
if (!$response) {
  die("Couldn't establish link to Verisign.\n");
}

echo "Verisign raw response was ".$response;

pfpro_cleanup();

?>
```

pfpro_process (PHP 4 >= 4.0.2)

Process a transaction with Payflow Pro

array **pfpro_process** (array parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]]) \linebreak

Returns: An associative array containing the response

pfpro_process() processes a transaction with Payflow Pro. The first parameter is an associative array containing keys and values that will be encoded and passed to the processor.

The second parameter is optional and specifies the host to connect to. By default this is "test.signio.com", so you will certainly want to change this to "connect.signio.com" in order to process live transactions.

The third parameter specifies the port to connect on. It defaults to 443, the standard SSL port.

The fourth parameter specifies the timeout to be used, in seconds. This defaults to 30 seconds. Note that this timeout appears to only begin once a link to the processor has been established and so your script could potentially continue for a very long time in the event of DNS or network problems.

The fifth parameter, if required, specifies the hostname of your SSL proxy. The sixth parameter specifies the port to use.

The seventh and eighth parameters specify the logon identity and password to use on the proxy.

The function returns an associative array of the keys and values in the response.

> **Megjegyzés:** Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

Példa 1. Payflow Pro example

```
<?php

pfpro_init();

$transaction = array(USER => 'mylogin',
     PWD => 'mypassword',
     TRXTYPE => 'S',
```

```
    TENDER => 'C',
    AMT => 1.50,
    ACCT => '4111111111111111',
    EXPDATE => '0904'
    );

$response = pfpro_process($transaction);

if (!$response) {
  die("Couldn't establish link to Verisign.\n");
}

echo "Verisign response code was ".$response[RESULT];
echo ", which means: ".$response[RESPMSG]."\n";

echo "\nThe transaction request: ";
print_r($transaction);

echo "\nThe response: ";
print_r($response);

pfpro_cleanup();

?>
```

pfpro_version (PHP 4 >= 4.0.2)

Returns the version of the Payflow Pro software

string **pfpro_version** (void) \linebreak

pfpro_version() returns the version string of the Payflow Pro library. At the time of writing, this was L211.

LXXVIII. PHP opciók és információk

assert_options (PHP 4)

Set/get the various assert flags

mixed **assert_options** (int what [, mixed value]) \linebreak

Using **assert_options()** you may set the various assert() control options or just query their current settings.

Táblázat 1. Assert Options

option	ini-parameter	default	description
ASSERT_ACTIVE	assert.active	1	enable assert() evaluation
ASSERT_WARNING	assert.warning	1	issue a PHP warning for each failed assertion
ASSERT_BAIL	assert.bail	0	terminate execution on failed assertions
ASSERT_QUIET_EVAL	assert.quiet_eval	0	disable error_reporting during assertion expression evaluation
ASSERT_CALLBACK	assert_callback	(NULL)	user function to call on failed assertions

assert_options() will return the original setting of any option or FALSE on errors.

assert (PHP 4)

Checks if assertion is FALSE

int **assert** (string|bool assertion) \linebreak

assert() will check the given *assertion* and take appropriate action if its result is FALSE.

If the *assertion* is given as a string it will be evaluated as PHP code by **assert()**. The advantages of a string *assertion* are less overhead when assertion checking is off and messages containing the *assertion* expression when an assertion fails.

Assertions should be used as a debugging feature only. You may use them for sanity-checks that test for conditions that should always be TRUE and that indicate some programming errors if not or to check for the presence of certain features like extension functions or certain system limits and features.

Assertions should not be used for normal runtime operations like input parameter checks. As a rule of thumb your code should always be able to work correctly if assertion checking is not activated.

The behavior of **assert()** may be configured by assert_options() or by .ini-settings described in that functions manual page.

The assert_options() function and/or ASSERT_CALLBACK configuration directive allow a callback function to be set to handle failed assertions.

assert() callbacks are particularly useful for building automated test suites because they allow you to easily capture the code passed to the assertion, along with information on where the assertion was made. While this information can be captured via other methods, using assertions makes it much faster and easier!

The callback function should accept three arguments. The first argument will contain the file the assertion failed in. The second argument will contain the line the assertion failed on and the third argument will contain the expression that failed (if any - literal values such as 1 or "two" will not be passed via this argument)

Példa 1. Handle a failed assertion with a custom handler

```
<?php
// Active assert and make it quiet
assert_options (ASSERT_ACTIVE, 1);
assert_options (ASSERT_WARNING, 0);
assert_options (ASSERT_QUIET_EVAL, 1);

// Create a handler function
function my_assert_handler ($file, $line, $code) {
    echo "<hr>Assertion Failed:
        File '$file'<br>
        Line '$line'<br>
        Code '$code'<br><hr>";
}

// Set up the callback
assert_options (ASSERT_CALLBACK, 'my_assert_handler');

// Make an assertion that should fail
assert ('mysql_query ("")');
?>
```

dl (PHP 3, PHP 4)

egy PHP kiterjesztés betöltése futásidőben

int **dl** (string library) \linebreak

Betölti azt a PHP kiterjesztést, amit a *`library`* paraméterben adtál meg. Lásd még az extension_dir konfigurációs direktívát.

extension_loaded (PHP 3>= 3.0.10, PHP 4)

Visszaadja, hogy a megadott kiterjesztés be van-e töltve

bool **extension_loaded** (string name) \linebreak

Igazat ad vissza ha a *name* paraméterben megadott kiterjesztés be van töltve. A különböző kiterjesztések neveit a phpinfo() függvénnyel is kiderítheted.

Lásd még: phpinfo().

Megjegyzés: Ez a függvény a 3.0.10-es verzióval került a nyelvbe.

get_cfg_var (PHP 3, PHP 4)

Visszaadja egy PHP konfigurációs beállítás értékét.

string **get_cfg_var** (string varname) \linebreak

Visszaadja a *varname* paraméterben megadott konfigurációs beállítás értékét, vagy FALSE értéket, ha hiba keletkezik.

Nem adja vissza a fordításkor beállított értékeket, vagy az Apache konfigurációs file-ból beolvasottakat (php3_configuration_option illetve php_configuration_option direktívák).

Ha le szeretnéd ellenőrizni, hogy a rendszer egy konfigurációs file-t használ, próbáld meg a cfg_file_path értéket lekérdezni. Ha ez elérhető, akkor egy konfigurációs file-t használ a PHP.

get_current_user (PHP 3, PHP 4)

Az aktuális PHP script tulajdonosának a nevét adja vissza.

string **get_current_user** (void) \linebreak

Visszaadja az aktuális PHP script tulajdonosának user nevét.

Lásd még: getmyuid(), getmypid(), getmyinode(), és getlastmod().

get_defined_constants (PHP 4 >= 4.1.0)

Returns an associative array with the names of all the constants and their values

array **get_defined_constants** (void) \linebreak

This function returns the names and values of all the constants currently defined. This includes those created by extensions as well as those created with the define() function.

For example the line below

```
print_r (get_defined_constants());
```

will print a list like:

```
Array
(
    [E_ERROR] => 1
    [E_WARNING] => 2
    [E_PARSE] => 4
    [E_NOTICE] => 8
    [E_CORE_ERROR] => 16
    [E_CORE_WARNING] => 32
    [E_COMPILE_ERROR] => 64
    [E_COMPILE_WARNING] => 128
    [E_USER_ERROR] => 256
    [E_USER_WARNING] => 512
    [E_USER_NOTICE] => 1024
    [E_ALL] => 2047
    [TRUE] => 1
)
```

See also get_loaded_extensions(), get_defined_functions() and get_defined_vars().

get_extension_funcs (PHP 4)

Visszaadja egy megadott modul függvényeinek a nevét

array **get_extension_funcs** (string module_name) \linebreak

Ez a függvény visszaadja egy tömbben az összes függvény nevét, amit a *`module_name`* modul definiál.

Például az alábbi sorok

```
print_r(get_extension_funcs("xml"));
print_r(get_extension_funcs("gd"));
```

ki fogják írni az `xml` és `gd` modulok függvényeinek neveit értelemszerűen.

Lásd még: get_loaded_extensions()

get_included_files (PHP 4)

Visszaad egy tömböt a filenevekkel, amiket a include_once()-al használtál

array **get_included_files** (void) \linebreak

Ez a függvény visszaadja egy asszociatív tömbben a file-ok neveit, amiket a scriptedbe töltöttél az include_once() használatával. A tömb indexei a filenevek, ahogy az include_once()-al használtad őket, a ".php" kiterjesztés nélkül.

> **Megjegyzés:** A PHP 4.0.1pl2-ben ez a függvény feltételezi, hogy az `include_once`-al beszúrt file-ok a ".php" kiterjesztéssel rendelkeznek. Más kiterjesztéssel a függvény nem fog működni.

Lásd még: require_once(), include_once(), get_required_files()

get_loaded_extensions (PHP 4)

Visszaad egy tömböt, ami tartalmazza a fordított és betöltött modulok nevét

array **get_loaded_extensions** (void) \linebreak

Ez a függvény visszaadja a PHP-be fordított és a betöltött modulok neveit egy tömmbben

Például az alábbi sor

```
print_r(get_loaded_extensions());
```

egy ilyen listát ír ki:

```
Array
(
    [0] => xml
    [1] => wddx
    [2] => standard
    [3] => session
    [4] => posix
    [5] => pgsql
    [6] => pcre
    [7] => gd
    [8] => ftp
    [9] => db
    [10] => Calendar
    [11] => bcmath
)
```

Lásd még: get_extension_funcs()

get_magic_quotes_gpc (PHP 3>= 3.0.6, PHP 4)

A 'magic_quotes_gpc' beállítás aktuális értékét adja vissza.

long **get_magic_quotes_gpc** (void) \linebreak

Visszaadja az aktív konfigurációs beállítást a magic_quotes_gpc-hez. (0 ha ki van kapcsolva, 1 ha be van kapcsolva).

Lásd még: get_magic_quotes_runtime() és set_magic_quotes_runtime().

get_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4)

A 'magic_quotes_runtime' beállítás aktuális értékét adja vissza.

long **get_magic_quotes_runtime** (void) \linebreak

Visszaadja az aktív konfigurációs beállítást a magic_quotes_runtime-hoz. (0 ha ki van kapcsolva, 1 ha be van kapcsolva).

Lásd még: get_magic_quotes_gpc() és set_magic_quotes_runtime().

get_required_files (PHP 4)

Visszaad egy tömböt a filenevekkel, amiket a require_once()-al használtál

array **get_required_files** (void) \linebreak

Ez a függvény visszaadja egy asszociatív tömbben a file-ok neveit, amiket a scriptedbe töltöttél a require_once() használatával. A tömb indexei a filenevek, ahogy a require_once()-al használtad őket, a ".php" kiterjesztés nélkül.

Az alábbi példa

Példa 1. A require-al és include-al behívott file-ok

```
<?php

require_once("local.php");
require_once("../inc/global.php");

for ($i=1; $i<5; $i++)
  include "util".$i."php";

echo "Required_once files\n";
print_r(get_required_files());
```

```
echo "Included_once files\n";
print_r(get_included_files());
?>
```

ezt a kimenetet generálja:

```
Required_once files
Array
(
    [local] => local.php
    [../inc/global] => /full/path/to/inc/global.php
)

Included_once files
Array
(
    [util1] => util1.php
    [util2] => util2.php
    [util3] => util3.php
    [util4] => util4.php
)
```

Megjegyzés: A PHP 4.0.1pl2-ben ez a függvény feltételezi, hogy a `required_once`-al beszúrt file-ok a ".php" kiterjesztéssel rendelkeznek. Más kiterjesztéssel a függvény nem fog működni.

Lásd még: require_once(), include_once(), get_included_files()

getenv (PHP 3, PHP 4)

Egy környezeti változó értékét adja vissza

string **getenv** (string varname) \linebreak

Visszaadja a *`varname`* környezeti változó értékét, vagy `FALSE` értéket hiba esetén.

```
$ipcim = getenv("REMOTE_ADDR"); // visszaadja az IP cimét a látogatónak
```

Az összes környezeti változót a phpinfo() listázza ki. Ha mélyebben bele szeretnél merülni a témába, nézd meg a CGI specifikációt (http://hoohoo.ncsa.uiuc.edu/cgi/), különösen a környezeti változókról szóló oldalt (http://hoohoo.ncsa.uiuc.edu/cgi/env.html).

getlastmod (PHP 3, PHP 4)

Visszaadja az oldal legutóbbi módosításának idejét.

int **getlastmod** (void) \linebreak

Visszaadja az oldal legutóbbi módosításának idejét. A visszaadott érték egy Unix timestamp, azért alkalmas pl. a date() függvény bemeneteként való alkalmazásra. FALSE értékkel tér vissza hiba esetén.

Példa 1. getlastmod() példa

```
// kiírja pl., hogy 'Legutóbbi módosítás: 2000. 07. 30. 16:49:43.'
echo "Legutóbbi módosítás: ".date( "Y. m. d. H:i:s.", getlastmod() );
```

Lásd még: date(), getmyuid(), get_current_user(), getmyinode() és getmypid().

getmygid (PHP 4 >= 4.1.0)

Get PHP script owner's GID

int **getmygid** (void) \linebreak

Returns the group ID of the current script, or FALSE on error.

See also getmyuid(), getmypid(), get_current_user(), getmyinode(), and getlastmod().

getmyinode (PHP 3, PHP 4)

Visszaadja az aktuális oldal inode-ját.

int **getmyinode** (void) \linebreak

Visszaadja az aktuális PHP oldal inode-ját, vagy FALSE értéket hiba esetén.

Lásd még: getmyuid(), get_current_user(), getmypid() és getlastmod().

Megjegyzés: Ez a függvény természetesen nem működik Windows alatt.

getmypid (PHP 3, PHP 4)

Visszaadja a PHP process ID-jét.

int **getmypid** (void) \linebreak

Visszaadja az aktuális PHP process ID-t, vagy FALSE értéket hiba esetén.

Amikor a PHP szerver modulként üzemel, egy script több meghívása nem garantált, hogy más ID-t kap.

Lásd még: getmyuid(), get_current_user(), getmyinode() és getlastmod().

getmyuid (PHP 3, PHP 4)

Visszadja a PHP script tulajdonosának userID-ját

int **getmyuid** (void) \linebreak

Visszaadja az aktuális script UID-ját, vagy FALSE értéket hiba esetén.

Lásd még: getmypid(), get_current_user(), getmyinode() és getlastmod().

getrusage (PHP 3>= 3.0.7, PHP 4)

Visszadja az aktuális erőforrás-terheltségeket

array **getrusage** ([int who]) \linebreak

Ez egy felület a getrusage(2)-hez. Egy asszociatív tömböt ad vissza, ami a rendszerhívás által adott adatokat tartalmazza. Ha a who 1, a getrusage az RUSAGE_CHILDREN-el hívódik meg.

Minden bejegyzés a dokumentált elem nevekkel érhető el.

Példa 1. Getrusage példa

```
$dat = getrusage();
echo $dat["ru_nswap"];         # swap-ek száma
echo $dat["ru_majflt"];        # page fault-ok száma
echo $dat["ru_utime.tv_sec"];  # felhasznált user idő (másodpercek)
echo $dat["ru_utime.tv_usec"]; # felhasznált user idő (egymilliomod másodperc)
```

Lásd a rendszered manpage-ét bővebb információért.

ini_alter (PHP 4)

Changes the value of a configuration option

string **ini_alter** (string varname, string newvalue) \linebreak

Changes the value of a configuration option, returns `FALSE` on failure, and the previous value of the configuration option on success.

Megjegyzés: This is an alias of ini_set()

See also ini_get(), ini_get_all(), ini_restore(), and ini_set()

ini_get_all (PHP 4 >= 4.2.0)

Gets all configuration options

array **ini_get_all** ([string extension]) \linebreak

Returns all the registered configuration options as an associative array. If optional *`extension`* parameter is set, returns only options specific for that extension.

See also: ini_alter(), ini_restore(), ini_get(), and ini_set()

ini_get (PHP 4)

Gets the value of a configuration option

string **ini_get** (string varname) \linebreak

Returns the value of the configuration option on success, an empty string on failure.

See also get_cfg_var(), ini_get_all(), ini_alter(), ini_restore(), and ini_set()

ini_restore (PHP 4)

Restores the value of a configuration option

string **ini_restore** (string varname) \linebreak

Restores a given configuration option to its original value.

See also: ini_alter(), ini_get(), ini_get_all(), and ini_set()

ini_set (PHP 4)

Sets the value of a configuration option

string **ini_set** (string varname, string newvalue) \linebreak

Sets the value of the given configuration option. Returns the old value on success, FALSE on failure. The configuration option will keep this new value during the script's execution, and will be restored at the script's ending.

Not all the available options can be changed using **ini_set()**. Below is a table with a list of all PHP options (as of PHP 4.2.0), indicating which ones can be changed/set and at what level.

Táblázat 1. Configuration options

Name	Default	Changeable
com.allow_dcom	"0"	PHP_INI_SYSTEM
com.autoregister_typelib	"0"	PHP_INI_SYSTEM
com.autoregister_verbose	"0"	PHP_INI_SYSTEM
com.autoregister_casesensitive	"1"	PHP_INI_SYSTEM
com.typelib_file	""	PHP_INI_SYSTEM
crack.default_dictionary	NULL	PHP_INI_SYSTEM
exif.encode_unicode	"ISO-8859-15"	PHP_INI_ALL
exif.decode_unicode_motorola	"UCS-2BE"	PHP_INI_ALL
exif.decode_unicode_intel	"UCS-2LE"	PHP_INI_ALL
exif.encode_jis	""	PHP_INI_ALL
exif.decode_jis_motorola	"JIS"	PHP_INI_ALL
exif.decode_jis_intel	"JIS"	PHP_INI_ALL
fbsql.allow_persistent	"1"	PHP_INI_SYSTEM
fbsql.generate_warnings	"0"	PHP_INI_SYSTEM
fbsql.autocommit	"1"	PHP_INI_SYSTEM
fbsql.max_persistent	"-1"	PHP_INI_SYSTEM
fbsql.max_links	"128"	PHP_INI_SYSTEM
fbsql.max_connections	"128"	PHP_INI_SYSTEM
fbsql.max_results	"128"	PHP_INI_SYSTEM
fbsql.batchSize	"1000"	PHP_INI_SYSTEM
fbsql.default_host	NULL	PHP_INI_SYSTEM
fbsql.default_user	"_SYSTEM"	PHP_INI_SYSTEM
fbsql.default_password	""	PHP_INI_SYSTEM
fbsql.default_database	""	PHP_INI_SYSTEM
fbsql.default_database_password	""	PHP_INI_SYSTEM
hwapi.allow_persistent	"0"	PHP_INI_SYSTEM
hyerwave.allow_persistent	"0"	PHP_INI_SYSTEM
hyperwave.default_port	"418"	PHP_INI_ALL
iconv.input_encoding	ICONV_INPUT_ENCODING	PHP_INI_ALL
iconv.output_encoding	ICONV_OUTPUT_ENCODING	PHP_INI_ALL
iconv.internal_encoding	ICONV_INTERNAL_ENCODING	PHP_INI_ALL
ifx.allow_persistent	"1"	PHP_INI_SYSTEM

Name	Default	Changeable
ifx.max_persistent	"-1"	PHP_INI_SYSTEM
ifx.max_links	"-1"	PHP_INI_SYSTEM
ifx.default_host	NULL	PHP_INI_SYSTEM
ifx.default_user	NULL	PHP_INI_SYSTEM
ifx.default_password	NULL	PHP_INI_SYSTEM
ifx.blobinfile	"1"	PHP_INI_ALL
ifx.textasvarchar	"0"	PHP_INI_ALL
ifx.byteasvarchar	"0"	PHP_INI_ALL
ifx.charasvarchar	"0"	PHP_INI_ALL
ifx.nullformat	"0"	PHP_INI_ALL
ingres.allow_persistent	"1"	PHP_INI_SYSTEM
ingres.max_persistent	"-1"	PHP_INI_SYSTEM
ingres.max_links	"-1"	PHP_INI_SYSTEM
ingres.default_database	NULL	PHP_INI_ALL
ingres.default_user	NULL	PHP_INI_ALL
ingres.default_password	NULL	PHP_INI_ALL
ibase.allow_persistent	"1"	PHP_INI_SYSTEM
ibase.max_persistent	"-1"	PHP_INI_SYSTEM
ibase.max_links	"-1"	PHP_INI_SYSTEM
ibase.default_user	NULL	PHP_INI_ALL
ibase.default_password	NULL	PHP_INI_ALL
ibase.timestampformat	"%m/%d/%Y%H:%M:%S"	PHP_INI_ALL
ibase.dateformat	"%m/%d/%Y"	PHP_INI_ALL
ibase.timeformat	"%H:%M:%S"	PHP_INI_ALL
java.class.path	NULL	PHP_INI_ALL
java.home	NULL	PHP_INI_ALL
java.library.path	NULL	PHP_INI_ALL
java.library	JAVALIB	PHP_INI_ALL
java.library	NULL	PHP_INI_ALL
ldap.max_links	"-1"	PHP_INI_SYSTEM
mbstring.detect_order	NULL	PHP_INI_ALL
mbstring.http_input	NULL	PHP_INI_ALL
mbstring.http_output	NULL	PHP_INI_ALL
mbstring.internal_encoding	NULL	PHP_INI_ALL
mbstring.substitute_character	NULL	PHP_INI_ALL
mbstring.func_overload	"0"	PHP_INI_SYSTEM
mcrypt.algorithms_dir	NULL	PHP_INI_ALL
mcrypt.modes_dir	NULL	PHP_INI_ALL
mime_magic.magicfile	"/usr/share/misc/magic.mime"	PHP_INI_SYSTEM
mssql.allow_persistent	"1"	PHP_INI_SYSTEM
mssql.max_persistent	"-1"	PHP_INI_SYSTEM
mssql.max_links	"-1"	PHP_INI_SYSTEM

Name	Default	Changeable
mssql.max_procs	"25"	PHP_INI_ALL
mssql.min_error_severity	"10"	PHP_INI_ALL
mssql.min_message_severity	"10"	PHP_INI_ALL
mssql.compatability_mode	"0"	PHP_INI_ALL
mssql.connect_timeout	"5"	PHP_INI_ALL
mssql.timeout	"60"	PHP_INI_ALL
mssql.textsize	"-1"	PHP_INI_ALL
mssql.textlimit	"-1"	PHP_INI_ALL
mssql.batchsize	"0"	PHP_INI_ALL
mssql.datetimeconvert	"1"	PHP_INI_ALL
mysql.allow_persistent	"1"	PHP_INI_SYSTEM
mysql.max_persistent	"-1"	PHP_INI_SYSTEM
mysql.max_links	"-1"	PHP_INI_SYSTEM
mysql.default_host	NULL	PHP_INI_ALL
mysql.default_user	NULL	PHP_INI_ALL
mysql.default_password	NULL	PHP_INI_ALL
mysql.default_port	NULL	PHP_INI_ALL
mysql.default_socket	NULL	PHP_INI_ALL
ncurses.value	"42"	PHP_INI_ALL
ncurses.string	"foobar"	PHP_INI_ALL
odbc.allow_persistent	"1"	PHP_INI_SYSTEM
odbc.max_persistent	"-1"	PHP_INI_SYSTEM
odbc.max_links	"-1"	PHP_INI_SYSTEM
odbc.default_db	NULL	PHP_INI_ALL
odbc.default_user	NULL	PHP_INI_ALL
odbc.default_pw	NULL	PHP_INI_ALL
odbc.defaultlrl	"4096"	PHP_INI_ALL
odbc.defaultbinmode	"1"	PHP_INI_ALL
odbc.check_persistent	"1"	PHP_INI_SYSTEM
pfpro.defaulthost	"test.signio.com"	
pfpro.defaulthost	"test-payflow.verisign.com"	
pfpro.defaultport	"443"	PHP_INI_ALL
pfpro.defaulttimeout	"30"	PHP_INI_ALL
pfpro.proxyaddress	""	PHP_INI_ALL
pfpro.proxyport	""	PHP_INI_ALL
pfpro.proxylogon	""	PHP_INI_ALL
pfpro.proxypassword	""	PHP_INI_ALL
pgsql.allow_persistent	"1"	PHP_INI_SYSTEM
pgsql.max_persistent	"-1"	PHP_INI_SYSTEM
pgsql.max_links	"-1"	PHP_INI_SYSTEM
pgsql.auto_reset_persistent	"0"	PHP_INI_SYSTEM
pgsql.ignore_notice	"0"	PHP_INI_ALL

Name	Default	Changeable
pgsql.log_notice	"0"	PHP_INI_ALL
session.save_path	"/tmp"	PHP_INI_ALL
session.name	"PHPSESSID"	PHP_INI_ALL
session.save_handler	"files"	PHP_INI_ALL
session.auto_start	"0"	PHP_INI_ALL
session.gc_probability	"1"	PHP_INI_ALL
session.gc_maxlifetime	"1440"	PHP_INI_ALL
session.serialize_handler	"php"	PHP_INI_ALL
session.cookie_lifetime	"0"	PHP_INI_ALL
session.cookie_path	"/"	PHP_INI_ALL
session.cookie_domain	""	PHP_INI_ALL
session.cookie_secure	""	PHP_INI_ALL
session.use_cookies	"1"	PHP_INI_ALL
session.use_only_cookies	"0"	PHP_INI_ALL
session.referer_check	""	PHP_INI_ALL
session.entropy_file	""	PHP_INI_ALL
session.entropy_length	"0"	PHP_INI_ALL
session.cache_limiter	"nocache"	PHP_INI_ALL
session.cache_expire	"180"	PHP_INI_ALL
session.use_trans_sid	"1"	PHP_INI_ALL
session.encode_sources	"globals	track"
extname.global_value	"42"	PHP_INI_ALL
extname.global_string	"foobar"	PHP_INI_ALL
assert.active	"1"	PHP_INI_ALL
assert.bail	"0"	PHP_INI_ALL
assert.warning	"1"	PHP_INI_ALL
assert.callback	NULL	PHP_INI_ALL
assert.quiet_eval	"0"	PHP_INI_ALL
safe_mode_protected_env_vars	SAFE_MODE_PROTECTED_ENV_VARS	PHP_INI_SYSTEM
safe_mode_allowed_env_vars	SAFE_MODE_ALLOWED_ENV_VARS	PHP_INI_SYSTEM
url_rewriter.tags	"a=href	area=href
url_rewriter.tags	"a=href	area=href
sybct.allow_persistent	"1"	PHP_INI_SYSTEM
sybct.max_persistent	"-1"	PHP_INI_SYSTEM
sybct.max_links	"-1"	PHP_INI_SYSTEM
sybct.min_server_severity	"10"	PHP_INI_ALL
sybct.min_client_severity	"10"	PHP_INI_ALL
sybct.hostname	NULL	PHP_INI_ALL
tokenizer.global_value	"42"	PHP_INI_ALL
tokenizer.global_string	"foobar"	PHP_INI_ALL

Name	Default	Changeable
vpopmail.directory	""	PHP_INI_ALL
zlib.output_compression	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR
zlib.output_compression_level	"-1"	PHP_INI_ALL
define_syslog_variables	"0"	PHP_INI_ALL
highlight.bg	HL_BG_COLOR	PHP_INI_ALL
highlight.comment	HL_COMMENT_COLOR	PHP_INI_ALL
highlight.default	HL_DEFAULT_COLOR	PHP_INI_ALL
highlight.html	HL_HTML_COLOR	PHP_INI_ALL
highlight.keyword	HL_KEYWORD_COLOR	PHP_INI_ALL
highlight.string	HL_STRING_COLOR	PHP_INI_ALL
allow_call_time_pass_reference	"1"	PHP_INI_SYSTEM|PHP_INI_PERDIR
asp_tags	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR
display_errors	"1"	PHP_INI_ALL
display_startup_errors	"0"	PHP_INI_ALL
enable_dl	"1"	PHP_INI_SYSTEM
expose_php	"1"	PHP_INI_SYSTEM
html_errors	"1"	PHP_INI_SYSTEM
xmlrpc_errors	"0"	PHP_INI_SYSTEM
xmlrpc_error_number	"0"	PHP_INI_ALL
ignore_user_abort	"0"	PHP_INI_ALL
implicit_flush	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
log_errors	"0"	PHP_INI_ALL
log_errors_max_len	"1024"	PHP_INI_ALL
ignore_repeated_errors	"0"	PHP_INI_ALL
ignore_repeated_source	"0"	PHP_INI_ALL
magic_quotes_gpc	"1"	PHP_INI_ALL
magic_quotes_runtime	"0"	PHP_INI_ALL
magic_quotes_sybase	"0"	PHP_INI_ALL
output_buffering	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
output_handler	NULL	PHP_INI_PERDIR|PHP_INI_SYSTEM
register_argc_argv	"1"	PHP_INI_ALL
register_globals	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
safe_mode	"1"	PHP_INI_SYSTEM
safe_mode	"0"	PHP_INI_SYSTEM
safe_mode_include_dir	NULL	PHP_INI_SYSTEM
safe_mode_gid	"0"	PHP_INI_SYSTEM

Name	Default	Changeable
short_open_tag	DEFAULT_SHORT_OPEN_TAG	PHP_INI_SYSTEM|PHP_INI_PERDIR
sql.safe_mode	"0"	PHP_INI_SYSTEM
track_errors	"0"	PHP_INI_ALL
y2k_compliance	"0"	PHP_INI_ALL
unserialize_callback_func	NULL	PHP_INI_ALL
arg_separator.output	"&"	PHP_INI_ALL
arg_separator.input	"&"	PHP_INI_SYSTEM|PHP_INI_PERDIR
auto_append_file	NULL	PHP_INI_ALL
auto_prepend_file	NULL	PHP_INI_ALL
doc_root	NULL	PHP_INI_SYSTEM
default_charset	SAPI_DEFAULT_CHARSET	PHP_INI_ALL
default_mimetype	SAPI_DEFAULT_MIMETYPE	PHP_INI_ALL
error_log	NULL	PHP_INI_ALL
extension_dir	PHP_EXTENSION_DIR	PHP_INI_SYSTEM
gpc_order	"GPC"	PHP_INI_ALL
include_path	PHP_INCLUDE_PATH	PHP_INI_ALL
max_execution_time	"30"	PHP_INI_ALL
open_basedir	NULL	PHP_INI_SYSTEM
safe_mode_exec_dir	"1"	PHP_INI_SYSTEM
upload_max_filesize	"2M"	PHP_INI_SYSTEM
file_uploads	"1"	PHP_INI_ALL
post_max_size	"8M"	PHP_INI_SYSTEM
upload_tmp_dir	NULL	PHP_INI_SYSTEM
user_dir	NULL	PHP_INI_SYSTEM
variables_order	NULL	PHP_INI_ALL
error_append_string	NULL	PHP_INI_ALL
error_prepend_string	NULL	PHP_INI_ALL
SMTP	"localhost"	PHP_INI_ALL
smtp_port	25	PHP_INI_ALL
browscap	NULL	PHP_INI_SYSTEM
error_reporting	NULL	PHP_INI_ALL
memory_limit	"8M"	PHP_INI_ALL
precision	"14"	PHP_INI_ALL
sendmail_from	NULL	PHP_INI_ALL
sendmail_path	DEFAULT_SENDMAIL_PATH	PHP_INI_SYSTEM
disable_functions	""	PHP_INI_SYSTEM
allow_url_fopen	"1"	PHP_INI_ALL
always_populate_raw_post_data	"0"	PHP_INI_ALL
xbithack	"0"	PHP_INI_ALL
engine	"1"	PHP_INI_ALL

Name	Default	Changeable
last_modified	"0"	PHP_INI_ALL
child_terminate	"0"	PHP_INI_ALL
async_send	"0"	PHP_INI_ALL

Táblázat 2. Definition of PHP_INI_* constants

Constant	Value	Meaning
PHP_INI_USER	1	Entry can be set in user scripts
PHP_INI_PERDIR	2	Entry can be set in `.htaccess`
PHP_INI_SYSTEM	4	Entry can be set in `php.ini` or `httpd.conf`
PHP_INI_ALL	7	Entry can be set anywhere

See also: ini_alter(), ini_get(), and ini_restore()

php_logo_guid (PHP 4)

Visszaadja a logo guid-t.

string **php_logo_guid** (void) \linebreak

Megjegyzés: Ez a függvény a PHP 4 Beta 4 verzióban került a nyelvbe.

php_sapi_name (PHP 4 >= 4.0.1)

Visszaadja, hogy milyen módon fut a PHP

string **php_sapi_name** (void) \linebreak

A **php_sapi_name()** visszad egy kisbetűs string-et, ami megadja, hogy milyen felület van a szerver és a PHP között (Server API, SAPI). CGI PHP esetében, ez a string "cgi", Apache mod_php esetén "apache", stb.

Példa 1. php_sapi_name() példa

```
$inter_type = php_sapi_name();
if ($inter_type == "cgi")
    print "Te CGI PHP-t használsz\n";
```

```
else
    print "Te nem CGI PHP-t használsz\n";
```

php_uname (PHP 4 >= 4.0.2)

Returns information about the operating system PHP was built on

string **php_uname** (void) \linebreak

php_uname() returns a string with a description of the operating system PHP is built on.

Példa 1. php_uname() Example

```
if (substr(php_uname(), 0, 7) == "Windows") {
   die ("Sorry, this script doesn’t run on Windows.\n");
}
```

phpcredits (PHP 4)

Prints out the credits for PHP

void **phpcredits** ([int flag]) \linebreak

This function prints out the credits listing the PHP developers, modules, etc. It generates the appropriate HTML codes to insert the information in a page. *`flag`* is optional, and it defaults to `CREDITS_ALL`. To generate a custom credits page, you may want to use the *`flag`* parameter. For example to print the general credits, you will use somewhere in your code:

```
...
phpcredits(CREDITS_GENERAL);
...
```

And if you want to print the core developers and the documentation group, in a page of its own, you will use:

```
<?php
phpcredits(CREDITS_GROUP + CREDITS_DOCS + CREDITS_FULLPAGE);
?>
```

And if you feel like embedding all the credits in your page, then code like the one below will do it:

```
<html>
 <head>
  <title>My credits page</title>
 </head>
 <body>
  <?php
 // some code of your own
 phpcredits(CREDITS_ALL);
 // some more code
 ?>
 </body>
</html>
```

Táblázat 1. Pre-defined phpcredits() flags

name	description
CREDITS_ALL	All the credits, equivalent to using: CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGE. It generates a complete stand-alone HTML page with the appropriate tags.
CREDITS_DOCS	The credits for the documentation team
CREDITS_FULLPAGE	Usually used in combination with the other flags. Indicates that the a complete stand-alone HTML page needs to be printed including the information indicated by the other flags.
CREDITS_GENERAL	General credits: Language design and concept, PHP 4.0 authors and SAPI module.
CREDITS_GROUP	A list of the core developers
CREDITS_MODULES	A list of the extension modules for PHP, and their authors
CREDITS_SAPI	A list of the server API modules for PHP, and their authors

See also: phpinfo(), phpversion(), and php_logo_guid().

phpinfo (PHP 3, PHP 4)

Rengeteg PHP információt ad vissza.

int **phpinfo** (void) \linebreak

Rengeteg hasznos információt ad vissza a PHP aktuális állapotáról. Ez magában foglalja a PHP konfigurációs beállításait, a kiterjesztéseket, a PHP verziószámát, szerver és környezeti információkat (ha a PHP szerver modulként működik), a PHP környezeti információit, operációs rendszer verziószámot, elérési utakat, php3.ini (php.ini PHP 4 esetén) beállításokat, helyi konfigurációs beállításokat, HTTP fejléceket, és a GNU Public Licence-et.

Lásd még: phpversion().

phpversion (PHP 3, PHP 4)

Visszadja az aktuális PHP verziószámot.

string **phpversion** (void) \linebreak

Visszad egy stringet, ami az éppen futó PHP feldolgozó verzóját mutatja.

Példa 1. phpversion() példa

```
// Kiírja, hogy pl. 'Aktuális PHP verzió: 4.0.0'
echo "Aktuális PHP verzió: ".phpversion();
```

Lásd még: phpinfo().

putenv (PHP 3, PHP 4)

Beállíja egy környezeti változó értékét.

void **putenv** (string setting) \linebreak

Hozzáadja a *`setting`* beállítást a szerver környezethez.

Példa 1. Egy környezeti változó beállítása

```
putenv("EGYEDIID=$egyediid");
```

set_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4)

Beállítja az aktív értéket a magic_quotes_runtime-hoz.

long **set_magic_quotes_runtime** (int new_setting) \linebreak

Beállítja az aktív konfigurációs értéket a magic_quotes_runtime-hoz. (0 kikapcsolja, 1 bekapcsolja).

Lásd még: get_magic_quotes_gpc(), get_magic_quotes_runtime().

set_time_limit (PHP 3, PHP 4)

Beállítja a maximális futási időt

void **set_time_limit** (int seconds) \linebreak

Beállítja, hogy mennyi másodperc adott maximálisan egy script futásához. Ha ezt az értéket a script eléri, fatális hiba keletkezik. Az alapérték 30 másodperc, vagy ha létezik, akkor a max_execution_time érték, amit a konfigurációs file-ban állíthatsz be. Ha nulla az értéke, nincs időkorlát.

Amikor meghívod, a **set_time_limit()** újraindítja az időmérést `NULL`áról. Ez azt jelenti, hogy 30 másodperces alapbeállítással a script futásának 25. másodpercében átállítod ezt egy set_time_limit(20) hívással, a script végülis összesen 45 másodpercig futhat.

Azonban a **set_time_limit()**-nek nincs hatása, ha ’safe mode’-ot használsz. Ilyenkor nincs más mód, mint kikapcsolni a ’safe mode’-ot, vagy átállítani a konfigurációs file-ban ezt az értéket.

version_compare (PHP 4 >= 4.1.0)

Compares two "PHP-standardized" version number strings

int **version_compare** (string version1, string version2 [, string operator]) \linebreak

version_compare() compares two "PHP-standardized" version number strings. This is useful if you would like to write programs working only on some versions of PHP.

version_compare() returns -1 if the first version is lower than the second, 0 if they are equal, and +1 if the second is lower.

If you specify the third optional *`operator`* argument, you can test for a particular relationship. The possible operators are: `<`, `lt`, `<=`, `le`, `>`, `gt`, `>=`, `ge`, `==`, `=`, `eq`, `!=`, `<>`, `ne` respectively. Using this argument, the function will return 1 if the relationship is the one specified by the operator, 0 otherwise.

Példa 1. version_compare() Example

```
// prints -1
echo version_compare("4.0.4", "4.0.6");

// these all print 1
```

```
echo version_compare("4.0.4", "4.0.6", "<");
echo version_compare("4.0.6", "4.0.6", "eq");
```

zend_logo_guid (PHP 4)

Visszaadja a Zend logo guid-t.

string **zend_logo_guid** (void) \linebreak

Megjegyzés: Ez a függvény a PHP 4 Beta 4 verzióban került a nyelvbe.

zend_version (PHP 4)

Gets the version of the current Zend engine

string **zend_version** (void) \linebreak

Returns a string containing the version of the currently running PHP parser.

Példa 1. zend_version() Example

```
// prints e.g. 'Zend engine version: 1.0.4'
echo "Zend engine version: " . zend_version();
```

See also phpinfo(), phpcredits(), php_logo_guid(), and phpversion().

LXXIX. POSIX functions

This module contains an interface to those functions defined in the IEEE 1003.1 (POSIX.1) standards document which are not accessible through other means. POSIX.1 for example defined the open(), read(), write() and close() functions, too, which traditionally have been part of PHP 3 for a long time. Some more system specific functions have not been available before, though, and this module tries to remedy this by providing easy access to these functions.

> **Figyelem**
>
> Sensitive data can be retrieved with the POSIX functions, e.g. posix_getpwnam() and friends. None of the POSIX function perform any kind of access checking when safe mode is enabled. It's therefore **strongly** advised to disable the POSIX extension at all (use `--disable-posix` in your configure line) if you're operating in such an environment.

Megjegyzés: The POSIX extension is **not** available on the Windows platform.

posix_ctermid (PHP 3>= 3.0.13, PHP 4)

Get path name of controlling terminal

string **posix_ctermid** (void) \linebreak

Needs to be written.

posix_getcwd (PHP 3>= 3.0.13, PHP 4)

Pathname of current directory

string **posix_getcwd** (void) \linebreak

Needs to be written ASAP.

posix_getegid (PHP 3>= 3.0.10, PHP 4)

Return the effective group ID of the current process

int **posix_getegid** (void) \linebreak

Return the numeric effective group ID of the current process. See also posix_getgrgid() for information on how to convert this into a useable group name.

posix_geteuid (PHP 3>= 3.0.10, PHP 4)

Return the effective user ID of the current process

int **posix_geteuid** (void) \linebreak

Return the numeric effective user ID of the current process. See also posix_getpwuid() for information on how to convert this into a useable username.

posix_getgid (PHP 3>= 3.0.10, PHP 4)

Return the real group ID of the current process

int **posix_getgid** (void) \linebreak

Return the numeric real group ID of the current process. See also posix_getgrgid() for information on how to convert this into a useable group name.

posix_getgrgid (PHP 3>= 3.0.13, PHP 4)

Return info about a group by group id

array **posix_getgrgid** (int gid) \linebreak

Needs to be written.

posix_getgrnam (PHP 3>= 3.0.13, PHP 4)

Return info about a group by name

array **posix_getgrnam** (string name) \linebreak

Needs to be written.

posix_getgroups (PHP 3>= 3.0.10, PHP 4)

Return the group set of the current process

array **posix_getgroups** (void) \linebreak

Returns an array of integers containing the numeric group ids of the group set of the current process. See also posix_getgrgid() for information on how to convert this into useable group names.

posix_getlogin (PHP 3>= 3.0.13, PHP 4)

Return login name

string **posix_getlogin** (void) \linebreak

Returns the login name of the user owning the current process. See posix_getpwnam() for information how to get more information about this user.

posix_getpgid (PHP 3>= 3.0.10, PHP 4)

Get process group id for job control

int **posix_getpgid** (int pid) \linebreak

Returns the process group identifier of the process *pid*.

This is not a POSIX function, but is common on BSD and System V systems. If your system does not support this function at system level, this PHP function will always return FALSE.

posix_getpgrp (PHP 3>= 3.0.10, PHP 4)

Return the current process group identifier

int **posix_getpgrp** (void) \linebreak

Return the process group identifier of the current process. See POSIX.1 and the getpgrp(2) manual page on your POSIX system for more information on process groups.

posix_getpid (PHP 3>= 3.0.10, PHP 4)

Return the current process identifier

int **posix_getpid** (void) \linebreak

Return the process identifier of the current process.

posix_getppid (PHP 3>= 3.0.10, PHP 4)

Return the parent process identifier

int **posix_getppid** (void) \linebreak

Return the process identifier of the parent process of the current process.

posix_getpwnam (PHP 3>= 3.0.13, PHP 4)

Return info about a user by username

array **posix_getpwnam** (string username) \linebreak

Returns an associative array containing information about a user referenced by an alphanumeric username, passed in the *username* parameter.

The array elements returned are:

Táblázat 1. The user information array

Element	Description
name	The name element contains the username of the user. This is a short, usually less than 16 character "handle" of the user, not her real, full name. This should be the same as the *username* parameter used when calling the function, and hence redundant.

Element	Description
passwd	The passwd element contains the user’s password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
uid	User ID of the user in numeric form.
gid	The group ID of the user. Use the function posix_getgrgid() to resolve the group name and a list of its members.
gecos	GECOS is an obsolete term that refers to the finger information field on a Honeywell batch processing system. The field, however, lives on, and its contents have been formalized by POSIX. The field contains a comma separated list containing the user’s full name, office phone, office number, and home phone number. On most systems, only the user’s full name is available.
dir	This element contains the absolute path to the home directory of the user.
shell	The shell element contains the absolute path to the executable of the user’s default shell.

posix_getpwuid (PHP 3>= 3.0.13, PHP 4)

Return info about a user by user id

array **posix_getpwuid** (int uid) \linebreak

Returns an associative array containing information about a user referenced by a numeric user ID, passed in the *uid* parameter.

The array elements returned are:

Táblázat 1. The user information array

Element	Description
name	The name element contains the username of the user. This is a short, usually less than 16 character "handle" of the user, not her real, full name.
passwd	The passwd element contains the user’s password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
uid	User ID, should be the same as the *uid* parameter used when calling the function, and hence redundant.

Element	Description
gid	The group ID of the user. Use the function posix_getgrgid() to resolve the group name and a list of its members.
gecos	GECOS is an obsolete term that refers to the finger information field on a Honeywell batch processing system. The field, however, lives on, and its contents have been formalized by POSIX. The field contains a comma separated list containing the user's full name, office phone, office number, and home phone number. On most systems, only the user's full name is available.
dir	This element contains the absolute path to the home directory of the user.
shell	The shell element contains the absolute path to the executable of the user's default shell.

posix_getrlimit (PHP 3>= 3.0.10, PHP 4)

Return info about system ressource limits

array **posix_getrlimit** (void) \linebreak

Needs to be written ASAP.

posix_getsid (PHP 3>= 3.0.10, PHP 4)

Get the current sid of the process

int **posix_getsid** (int pid) \linebreak

Return the sid of the process *pid*. If *pid* is 0, the sid of the current process is returned.

This is not a POSIX function, but is common on System V systems. If your system does not support this function at system level, this PHP function will always return FALSE.

posix_getuid (PHP 3>= 3.0.10, PHP 4)

Return the real user ID of the current process

int **posix_getuid** (void) \linebreak

Return the numeric real user ID of the current process. See also posix_getpwuid() for information on how to convert this into a useable username.

posix_isatty (PHP 3>= 3.0.13, PHP 4)

Determine if a file descriptor is an interactive terminal

bool **posix_isatty** (int fd) \linebreak

Needs to be written.

posix_kill (PHP 3>= 3.0.13, PHP 4)

Send a signal to a process

bool **posix_kill** (int pid, int sig) \linebreak

Send the signal *`sig`* to the process with the process identifier *`pid`*. Returns `FALSE`, if unable to send the signal, `TRUE` otherwise.

See also the kill(2) manual page of your POSIX system, which contains additional information about negative process identifiers, the special pid 0, the special pid -1, and the signal number 0.

posix_mkfifo (PHP 3>= 3.0.13, PHP 4)

Create a fifo special file (a named pipe)

bool **posix_mkfifo** (string pathname, int mode) \linebreak

posix_mkfifo() creates a special `FIFO` file which exists in the file system and acts as a bidirectional communication endpoint for processes.

The second parameter *`mode`* has to be given in octal notation (e.g. 0644). The permission of the newly created `FIFO` also depends on the setting of the current umask(). The permissions of the created file are (mode & ~umask).

> **Megjegyzés:** Ha a safe mode be van kapcsolva, a PHP ellenőrzi, hogy a könyvtár, amelyben dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkezik-e, mint az éppen futó program.

posix_setegid (PHP 4 >= 4.0.2)

Set the effective GID of the current process

bool **posix_setegid** (int gid) \linebreak

Set the effective group ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns `TRUE` on success, `FALSE` otherwise.

posix_seteuid (PHP 4 >= 4.0.2)

Set the effective UID of the current process

bool **posix_seteuid** (int uid) \linebreak

Set the real user ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns TRUE on success, FALSE otherwise. See also posix_setgid().

posix_setgid (PHP 3>= 3.0.13, PHP 4)

Set the GID of the current process

bool **posix_setgid** (int gid) \linebreak

Set the real group ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function. The appropriate order of function calls is **posix_setgid()** first, posix_setuid() last.

Returns TRUE on success, FALSE otherwise.

posix_setpgid (PHP 3>= 3.0.13, PHP 4)

set process group id for job control

int **posix_setpgid** (int pid, int pgid) \linebreak

Let the process *pid* join the process group *pgid*. See POSIX.1 and the setsid(2) manual page on your POSIX system for more informations on process groups and job control. Returns TRUE on success, FALSE otherwise.

posix_setsid (PHP 3>= 3.0.13, PHP 4)

Make the current process a session leader

int **posix_setsid** (void) \linebreak

Make the current process a session leader. See POSIX.1 and the setsid(2) manual page on your POSIX system for more informations on process groups and job control. Returns the session id.

posix_setuid (PHP 3>= 3.0.13, PHP 4)

Set the UID of the current process

bool **posix_setuid** (int uid) \linebreak

Set the real user ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns TRUE on success, FALSE otherwise. See also posix_setgid().

posix_times (PHP 3>= 3.0.13, PHP 4)

Get process times

array **posix_times** (void) \linebreak

Returns a hash of strings with information about the current process CPU usage. The indices of the hash are

- ticks - the number of clock ticks that have elapsed since reboot.
- utime - user time used by the current process.
- stime - system time used by the current process.
- cutime - user time used by current process and children.
- cstime - system time used by current process and children.

posix_ttyname (PHP 3>= 3.0.13, PHP 4)

Determine terminal device name

string **posix_ttyname** (int fd) \linebreak

Needs to be written.

posix_uname (PHP 3>= 3.0.10, PHP 4)

Get system name

array **posix_uname** (void) \linebreak

Returns a hash of strings with information about the system. The indices of the hash are

- sysname - operating system name (e.g. Linux)
- nodename - system name (e.g. valiant)
- release - operating system release (e.g. 2.2.10)
- version - operating system version (e.g. #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - system architecture (e.g. i586)
- domainname - DNS domainname (e.g. php.net)

domainname is a GNU extension and not part of POSIX.1, so this field is only available on GNU systems or when using the GNU libc.

Posix requires that you must not make any assumptions about the format of the values, e.g. you cannot rely on three digit version numbers or anything else returned by this function.

LXXX. PostgreSQL függvények

Postgres, amit eredetileg a UC Berkeley Computer Science Department fejlesztett ki, úttörő volt az objektum-relációs adatmodellek területén, és most elérhetővé válik több kereskedelmi adatbázisban is. Támogatja az SQL92/SQL3 nyelv használatát, tranzakciók integritását és a típusok kiterjeszthetőségét. A PostgreSQL egy public-domain, nyílt forrású leszármazottja ennek az eredeti Berkeley kódnak.

A PostgreSQL ingyenes. A legújabb verzió a www.PostgreSQL.org (http://www.postgresql.org/) címen érhető el.

A 6.3 verzió óta (03/02/1998) a PostgreSQL unix socketeket használ. A lenti táblázat mutatja az új kapcsolatteremtési lehetőségeket. Ez a socket a `/tmp/.s.PGSQL.5432` néven érhető el. Ezt az opciót a **postmaster** parancs ’-i’ kapcsolójával érheted el, a jelentése pedig: "figyeld a TCP/IP socketeket is úgy, mint a unix socketeket".

Táblázat 1. Postmaster és a PHP

Postmaster	PHP	Állapot
postmaster &	pg_connect("", "", "", "", "dbname");	OK
postmaster -i &	pg_connect("", "", "", "", "dbname");	OK
postmaster &	pg_connect("localhost", "", "", "", "dbname");	Unable to connect to PostgreSQL server: connectDB() failed: Is the postmaster running and accepting TCP/IP (with -i) connection at ’localhost’ on port ’5432’? in /path/to/file.php3 on line 20. vagyis: Nem lehet kapcsolódni a PostgreSQL szerverhez: a connectDB() hívás meghiúsult. Fut a postmaster, és fogadja a TCP/IP kapcsolatokat (-i) a helyi gépen az 5432-es porton? a /elérési_út/a/file.php3 a 20-as sorban
postmaster -i &	pg_connect("localhost", "", "", "", "dbname");	OK

A következőképp is kezdhetsz kapcsolatot: **$conn = pg_Connect("host=localhost port=5432 dbname=chris");**

Annak érdekében, hogy használhassunk nagy objektum felületet (lo), szükséges az egészet egy tranzakciós blokkba foglalni. A tranzakciós blokk egy **begin**-nel kezdődik, és ha a tranzakció érvényes, egy **commit**-tal vagy egy **end**-del végződik. Ha a tranzakció meghiúsul, akkor **rollback** vagy **abort** paranccsal kell végződnie.

Példa 1. Nagy objektumok használata

```
<?php
```

```
    $database = pg_Connect ("", "", "", "", "jacarta");
    pg_exec ($database, "begin");
    $oid = pg_locreate ($database);
    echo ("$oid\n");
    $handle = pg_loopen ($database, $oid, "w");
    echo ("$handle\n");
    pg_lowrite ($handle, "gaga");
    pg_loclose ($handle);
    pg_exec ($database, "commit");
?>
```

pg_affected_rows (PHP 4 >= 4.2.0)

Returns number of affected records(tuples)

int **pg_affected_rows** (resource result) \linebreak

pg_affected_rows() returns the number of tuples (instances/records/rows) affected by INSERT, UPDATE, and DELETE queries executed by pg_query(). If no tuple is affected by this function, it will return 0.

Példa 1. pg_affected_rows()

```
<?php
     $result = pg_query ($conn, "INSERT INTO publisher VALUES (’Author’)");
     $cmdtuples = pg_affected_rows ($result);
     echo $cmdtuples . " tuples are affected.";
?>
```

Megjegyzés: This function used to be called `pg_cmdtuples()`.

See also pg_query() and pg_num_rows().

pg_cancel_query (PHP 4 >= 4.2.0)

Cancel async query

bool **pg_cancel_query** (resource connection) \linebreak

pg_cancel_query() cancel asynchronous query sent by pg_send_query(). You cannot cancel query executed by pg_query().

See also pg_send_query() and pg_connection_busy()

pg_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Get the client encoding

string **pg_client_encoding** ([resource connection]) \linebreak

pg_client_encoding() returns the client encoding as the string. The returned string should be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

Megjegyzés: This function requires PHP-4.0.3 or higher and PostgreSQL-7.0 or higher. If libpq is compiled without multibyte encoding support, pg_set_client_encoding() always return

"SQL_ASCII". Supported encoding depends on PostgreSQL version. Refer to PostgreSQL manual for details to enable multibyte support and encoding supported.

The function used to be called **pg_clientencoding()**.

See also pg_set_client_encoding().

pg_Close (PHP 3, PHP 4)

lezár egy PostgreSQL kapcsolatot

bool **pg_close** (int connection) \linebreak

Hamissal tér vissza, ha a connection érvénytelen kapcsolatazonosító, egyébként igazzal. Lezárja az adott azonosítójú kapcsolatot a PostgreSQL adatbázissal.

pg_Connect (PHP 3, PHP 4)

megnyit egy kapcsolatot

int **pg_connect** (string host, string port, string options, string tty, string dbname) \linebreak

Sikeres végrehajtás esetén egy kapcsolat-azonosítóval tér vissza, vagy hamissal, ha a kapcsolat nem hozható létre. Kapcsolatot nyit egy PostgreSQL adatbázishoz. Az összes argumentum változó-interpolált, beleértve a porszámot is. Az options és a tty argumentumok opcionálisak, vagyis elhagyhatók. Ez a függvény egy kapcsolat-azonosítóval tér vissza, mely más PostgreSQL függvények használatához szükséges. Egyszerre több kapcsolatot is tudsz nyitni.

Kapcsolat létrehozására alkalmas az alábbi parancs is: **$conn = pg_connect("dbname=marliese port=5432");** A *`dbname`* és a *`port`* paraméter mellett még használható a *`host`*, *`tty`*, *`options`*, *`user`* és a *`password`*.

Lásd még a **pg_pConnect()** függvényt.

pg_connection_busy (PHP 4 >= 4.2.0)

Get connection is busy or not

bool **pg_connection_busy** (resource connection) \linebreak

pg_connection_busy() returns `TRUE` if the connection is busy. If it is busy, a previous query is still executing. If pg_get_result() is called, it will be blocked.

See also pg_connection_status() and pg_get_result()

pg_connection_reset (PHP 4 >= 4.2.0)

Reset connection (reconnect)

bool **pg_connection_reset** (resource connection) \linebreak

pg_connection_reset() resets the connection. It is useful for error recovery. Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

See also pg_connect(), pg_pconnect() and pg_connection_status()

pg_connection_status (PHP 4 >= 4.2.0)

Get connection status

int **pg_connection_status** (resource connection) \linebreak

pg_connection_status() returns a connection status. Possible statuses are `PGSQL_CONNECTION_OK` and `PGSQL_CONNECTION_BAD`.

See also pg_connection_busy().

pg_convert (PHP 4 CVS only)

Convert associative array value into suitable for SQL statement.

array **pg_convert** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_convert() check and convert `assoc_array` suitable for SQL statement.

Megjegyzés: This function is experimental.

See also pg_metadata()

pg_copy_from (PHP 4 >= 4.2.0)

Insert records into a table from an array

int **pg_copy_from** (int connection, string table_name, array rows [, string delimiter [, string null_as]]) \linebreak

pg_copy_from() insert records into a table from *`rows`*. It issues `COPY` command internally to insert records. Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

See also pg_copy_to()

pg_copy_to (PHP 4 >= 4.2.0)

Copy a table to an array

int **pg_copy_to** (int connection, string table_name [, string delimiter [, string null_as]]) \linebreak

pg_copy_to() copies a table to an array. The resulting array is returned. It returns `FALSE` on failure.

See also pg_copy_from()

pg_DBname (PHP 3, PHP 4)

adatbázis neve

string **pg_dbname** (int connection) \linebreak

Az adott kapcsolatazonosítójú PostgreSQL adatbázisnevével tér vissza, vagy hamissal, ha a kapcsolat-azonosító érvénytelen.

pg_delete (PHP 4 CVS only)

Delete records.

long **pg_delete** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_delete() deletes record condition specified by `assoc_array` which has `field=>value`. If `option` is specified, pg_convert() is applied to `assoc_array` with specified option.

Példa 1. pg_delete

```
<?php
    $db = pg_connect (’dbname=foo’);
    // This is safe, since $_POST is converted automatically
    $res = pg_delete($db, ’post_log’, $_POST);
    if ($res) {
        echo "POST data is deleted: $res\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Megjegyzés: This function is experimental.

See also pg_convert()

pg_end_copy (PHP 4 >= 4.0.3)

Sync with PostgreSQL backend

bool **pg_end_copy** ([resource connection]) \linebreak

pg_end_copy() syncs the PostgreSQL frontend (usually a web server process) with the PostgreSQL server after doing a copy operation performed by pg_put_line(). **pg_end_copy()** must be issued, otherwise the PostgreSQL server may get out of sync with the frontend and will report an error. Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad.

For further details and an example, see also pg_put_line().

pg_escape_bytea (PHP 4 >= 4.2.0)

Escape binary for bytea type

string **pg_escape_bytea** (string data) \linebreak

pg_escape_bytea() escapes string for bytea datatype. It returns escaped string.

> **Megjegyzés:** When you SELECT bytea type, PostgreSQL returns octal byte value prefixed by \ (e.g. \032). Users are supposed to convert back to binary format by yourself.
>
> This function requires PostgreSQL 7.2 or later. With PostgreSQL 7.2.0 and 7.2.1, bytea type must be casted when you enable multi-byte support. i.e. `INSERT INTO test_table (image) VALUES ('$image_escaped'::bytea);` PostgreSQL 7.2.2 or later does not need cast. Exception is when client and backend character encoding does not match, there may be multi-byte stream error. User must cast to bytea to avoid this error.
>
> Newer PostgreSQL will support unescape function. Support for built-in unescape function will be added when it's available.

See also pg_escape_string()

pg_escape_string (PHP 4 >= 4.2.0)

Escape string for text/char type

string **pg_escape_string** (string data) \linebreak

pg_escape_string() escapes string for text/char datatype. It returns escaped string for PostgreSQL. Use of this functon is recommended instead of addslashes().

> **Megjegyzés:** This function requires PostgreSQL 7.2 or later.

See also pg_escape_bytea()

pg_Fetch_Array (PHP 3>= 3.0.1, PHP 4)

beolvas egy sort egy tömbbe

array **pg_fetch_array** (int result, int row [, int result_type]) \linebreak

Az adatbázis következő sorával tér vissza tömb formában, vagy hamissal, ha már nincs több sor.

A **pg_fetch_array()** függvény a pg_fetch_row() kiterjesztett változata. Amellett, hogy a tömb numerikusan indexelhető, az adatokat asszociatív indexszel is tárolja a mezőneveket használva kulcsnak.

A harmadik, *result_type* nevű argumentum a következő értékeket veheti fel: PGSQL_ASSOC, PGSQL_NUM, and PGSQL_BOTH.

Megjegyzés: A *result_type* paraméter a PHP 4.0-ás változatában került a nyelvbe.

Jó tudni, hogy a **pg_fetch_array()** használata NEM jelentősen lassabb, mint a pg_fetch_row() használata, míg az eredmény érthetőbb.

További részletekért lásd még a pg_fetch_row() függvényt.

Példa 1. A pg_fetch_array használata

```
<?php
$conn = pg_pconnect("","","","","publisher");
if (!$conn) {
    echo "Hiba történt.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "Hiba történt.\n";
    exit;
}

$arr = pg_fetch_array ($result, 0);
echo $arr[0] . " <- array\n"; #hiba < kell!!!

$arr = pg_fetch_array ($result, 1);
echo $arr["author"] . " <- array\n"; # itt is
?>
```

pg_Fetch_Object (PHP 3>= 3.0.1, PHP 4)

sor beolvasása objektumként

object **pg_fetch_object** (int result, int row [, int result_type]) \linebreak

Egy objektummal tér vissza, aminek a tulajdonságai megegyeznek a beolvasott sor mezőivel, hamissal, ha nincs több sor.

pg_fetch_object() hasonló a pg_fetch_array()-hoz, egy különbséget kivéve - objektummal tér vissza, nem tömbbel. Vagyis adatot csak mezőnevekkel érhetsz el, indexszel (számokkal) nem (a számok ugyanis illegális mezőnevek).

A harmadik, *result_type* nevű argumentum a következő értékeket veheti fel: PGSQL_ASSOC, PGSQL_NUM, and PGSQL_BOTH.

> **Megjegyzés:** A *result_type* paraméter a PHP 4.0-ás változatában került a nyelvbe.

Sebesség szempontjából, a függvény azonos a pg_fetch_array() függvénnyel, és majdnem olyan gyors, mint a pg_fetch_row() (a különbség jelentéktelen).

Lásd még a pg_fetch_array() és a pg_fetch_row() függvényeket.

Példa 1. pg_fetch_object alkalmazása

```
<?php
$database = "verlag";
$db_conn = pg_connect ("localhost", "5432", "", "", $database);
if (!$db_conn): ?>
    <H1>Nem lehet kapcsolódni a <? echo $database ?> nevű adatbázishoz.</H1> <?
    exit;
endif;

$qu = pg_exec ($db_conn, "SELECT * FROM verlag ORDER BY autor");
$row = 0; // A postgresnek kell egy sorszámláló, ami más adatbáziskezelőnél talán nem

while ($data = pg_fetch_object ($qu, $row)):
    echo $data->autor." (";
    echo $data->jahr ."): ";
    echo $data->titel."<BR>";
    $row++;
endwhile; ?>

<PRE><?php
$fields[] = Array ("autor", "Author");
$fields[] = Array ("jahr",  "  Year");
$fields[] = Array ("titel", " Title");

$row= 0; // A postgresnek kell egy sorszámláló, ami más adatbáziskezelőnél talán nem
while ($data = pg_fetch_object ($qu, $row)):
    echo "----------\n";
    reset ($fields);
    while (list (,$item) = each ($fields)):
        echo $item[1].": ".$data->$item[0]."\n";
    endwhile;
    $row++;
endwhile;
echo "----------\n"; ?>
</PRE> <?php
pg_freeResult ($qu);
pg_close ($db_conn);
```

```
?>
```

pg_fetch_result (PHP 4 >= 4.2.0)

Returns values from a result resource

mixed **pg_fetch_result** (resource result, int row, mixed field) \linebreak

pg_fetch_result() returns values from a *result* resource returned by pg_query(). *row* is integer. *field* is field name(string) or field index (integer). The *row* and *field* specify what cell in the table of results to return. Row numbering starts from 0. Instead of naming the field, you may use the field index as an unquoted number. Field indices start from 0.

PostgreSQL has many built in types and only the basic ones are directly supported here. All forms of integer, boolean and void types are returned as integer values. All forms of float, and real types are returned as float values. All other types, including arrays are returned as strings formatted in the same default PostgreSQL manner that you would see in the **psql** program.

pg_Fetch_Row (PHP 3>= 3.0.1, PHP 4)

következő sor beolvasása numerikusan indexelt tömbbe

array **pg_fetch_row** (int result, int row) \linebreak

A beolvasott sorral tér vissza numerikusan indexelt (hagyományos) tömb formájában, vagy hamissal, ha nincs több sor.

A **pg_fetch_row()** függvény betölti a megadott eredmény-azonosítónak megfelelő sort. Az eredményt eredmény formában adja vissza. Az egyes oszlopokat indexszel (számmal) lehet elérni. Az első oszlop indexe 0.

A **pg_fetch_row()** egymás utáni alkalmazásával az eredményhalmaz következő sorát kapjuk, amíg van következő sor, majd hamist, ha már nincs több sor.

Lásd még a pg_fetch_array(), pg_fetch_object() és a **pg_result()** függvényeket.

Példa 1. pg_fetch_row alkalmazás

```
<?php
$conn = pg_pconnect("","","","","publisher");
if (!$conn) {
    echo "Gáz van.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "Baj van, nagy baj van.\n";
    exit;
}
```

```
$row = pg_fetch_row ($result, 0);
echo $row[0] . " <- row\n";  # < helyett valami más kell

$row = pg_fetch_row ($result, 1);
echo $row[0] . " <- row\n"; # itt is

$row = pg_fetch_row ($result, 2);
echo $row[1] . " <- row\n"; # detto
?>
```

pg_field_is_null (PHP 4 >= 4.2.0)

Test if a field is `NULL`

int **pg_field_is_null** (resource result, int row, mixed field) \linebreak

pg_field_is_null() test if a field is `NULL` or not. It returns 1 if the field in the given row is `NULL`. It returns 0 if the field in the given row is NOT `NULL`. Field can be specified as column index (number) or fieldname (string). Row numbering starts at 0.

Megjegyzés: This function used to be called `pg_fieldisnull()`.

pg_field_name (PHP 4 >= 4.2.0)

Returns the name of a field

string **pg_field_name** (resource result, int field_number) \linebreak

pg_field_name() returns the name of the field occupying the given *`field_number`* in the given PostgreSQL *`result`* resource. Field numbering starts from 0.

Megjegyzés: This function used to be called `pg_fieldname()`.

See also pg_field_num().

pg_field_num (PHP 4 >= 4.2.0)

Returns the field number of the named field

int **pg_field_num** (resource result, string field_name) \linebreak

pg_field_num() will return the number of the column (field) slot that corresponds to the *field_name* in the given PostgreSQL *result* resource. Field numbering starts at 0. This function will return -1 on error.

Megjegyzés: This function used to be called pg_fieldnum().

See also pg_field_name().

pg_field_prtlen (PHP 4 >= 4.2.0)

Returns the printed length

int **pg_field_prtlen** (resource result, int row_number, string field_name) \linebreak

pg_field_prtlen() returns the actual printed length (number of characters) of a specific value in a PostgreSQL *result*. Row numbering starts at 0. This function will return -1 on an error.

Megjegyzés: This function used to be called pg_field_prtlen().

See also pg_field_size().

pg_field_size (PHP 4 >= 4.2.0)

Returns the internal storage size of the named field

int **pg_field_size** (resource result, int field_number) \linebreak

pg_field_size() returns the internal storage size (in bytes) of the field number in the given PostgreSQL *result*. Field numbering starts at 0. A field size of -1 indicates a variable length field. This function will return FALSE on error.

Megjegyzés: This function used to be called pg_fieldsize().

See also **pg_field_len()** and pg_field_type().

pg_field_type (PHP 4 >= 4.2.0)

Returns the type name for the corresponding field number

string **pg_field_type** (resource result, int field_number) \linebreak

pg_field_type() returns a string containing the type name of the given *field_number* in the given PostgreSQL *result* resource. Field numbering starts at 0.

Megjegyzés: This function used to be called `pg_fieldtype()`.

See also **pg_field_len()** and pg_field_name().

pg_free_result (PHP 4 >= 4.2.0)

Free result memory

bool **pg_free_result** (resource result) \linebreak

pg_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But, if you are sure you are not going to need the result data anymore in a script, you may call **pg_free_result()** with the *`result`* resource as an argument and the associated result memory will be freed. It returns true on success and false if an error occurs.

Megjegyzés: This function used to be called `pg_freeresult()`.

See also pg_query().

pg_get_result (PHP 4 >= 4.2.0)

Get asynchronous query result

resource **pg_get_result** ([resource connection]) \linebreak

pg_get_result() get result resource from async query executed by pg_send_query(). pg_send_query() can send multiple queries to PostgreSQL server and **pg_get_result()** is used to get query result one by one. It returns result resource. If there is no more results, it returns `FALSE`.

pg_Host (PHP 3, PHP 4)

A hostnevet adja vissza

string **pg_host** (int connection_id) \linebreak

A **pg_Host()** függvény az adott kapcsolat-azonosítójú PostgreSQL host nevét adja vissza.

pg_insert (PHP 4 CVS only)

Insert array into table.

bool **pg_insert** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_insert() inserts `assoc_array` which has `field=>value` into table specified as `table_name`. If `options` is specified, pg_convert() is applied to `assoc_array` with specified option.

Példa 1. pg_insert

```
<?php
    $db = pg_connect ('dbname=foo');
    // This is safe, since $_POST is converted autotmatically
    $res = pg_insert($db, 'post_log', $_POST);
    if ($res) {
        echo "POST data is succesfully logged\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Megjegyzés: This function is experimental.

See also pg_convert()

pg_last_error (PHP 4 >= 4.2.0)

Get the last error message string of a connection

string **pg_last_error** (resource connection) \linebreak

pg_last_error() returns the last error message for given *`connection`*.

Error messages may be overwritten by internal PostgreSQL(libpq) function calls. It may not return appropriate error message, if multiple errors are occured inside a PostgreSQL module function.

Use pg_result_error(), pg_result_status() and pg_connection_status() for better error handling.

Megjegyzés: This function used to be called `pg_errormessage()`.

See also pg_result_error().

pg_last_notice (PHP 4 >= 4.0.6)

Returns the last notice message from PostgreSQL server

string **pg_last_notice** (resource connection) \linebreak

pg_last_notice() returns the last notice message from the PostgreSQL server specified by *`connection`*. The PostgreSQL server sends notice messages in several cases, e.g. if the

transactions can’t be continued. With **pg_last_notice()**, you can avoid issuing useless queries, by checking whether the notice is related to the transaction or not.

> **Figyelem**
>
> This function is EXPERIMENTAL and it is not fully implemented yet. **pg_last_notice()** was added in PHP 4.0.6. However, PHP 4.0.6 has problem with notice message handling. Use of the PostgreSQL module with PHP 4.0.6 is not recommended even if you are not using **pg_last_notice()**.
>
> This function is fully implemented in PHP 4.3.0. PHP earlier than PHP 4.3.0 ignores database connection parameter.

Notice message tracking can be set to optional by setting 1 for `pgsql.ignore_notice` ini from PHP 4.3.0.

Notice message logging can be set to optional by setting 0 for `pgsql.log_notice` ini from PHP 4.3.0. Unless `pgsql.ignore_notice` is set to 0, notice message cannot be logged.

See also pg_query() and pg_last_error().

pg_last_oid (PHP 4 >= 4.2.0)

Returns the last object’s oid

int **pg_last_oid** (resource result) \linebreak

pg_last_oid() is used to retrieve the `oid` assigned to an inserted tuple (record) if the result resource is used from the last command sent via pg_query() and was an SQL INSERT. Returns a positive integer if there was a valid `oid`. It returns `FALSE` if an error occurs or the last command sent via pg_query() was not an INSERT or INSERT is failed.

OID field became an optional field from PostgreSQL 7.2. When OID field is not defined in a table, programmer must use pg_result_status() to check if record is is inserted successfully or not.

> **Megjegyzés:** This function used to be called `pg_getlastoid()`.

See also pg_query() and pg_result_status()

pg_lo_close (PHP 4 >= 4.2.0)

Close a large object

bool **pg_lo_close** (resource large_object) \linebreak

pg_lo_close() closes a Large Object. *`large_object`* is a resource for the large object from pg_lo_open().

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Megjegyzés:** This function used to be called `pg_loclose()`.

See also pg_lo_open(), pg_lo_create() and pg_lo_import().

pg_lo_create (PHP 4 >= 4.2.0)

Create a large object

int **pg_lo_create** (resource connection) \linebreak

pg_lo_create() creates a Large Object and returns the `oid` of the large object. *`connection`* specifies a valid database connection opened by pg_connect() or pg_pconnect(). PostgreSQL access modes INV_READ, INV_WRITE, and INV_ARCHIVE are not supported, the object is created always with both read and write access. INV_ARCHIVE has been removed from PostgreSQL itself (version 6.3 and above). It returns large object oid otherwise. It returns `FALSE`, if an error occurred,

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Megjegyzés: This function used to be called `pg_locreate()`.

pg_lo_export (PHP 4 >= 4.2.0)

Export a large object to file

bool **pg_lo_export** (int oid, string pathname [, resource connection]) \linebreak

The *`oid`* argument specifies oid of the large object to export and the *`pathname`* argument specifies the pathname of the file. It returns `FALSE` if an error occurred, `TRUE` otherwise.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Megjegyzés: This function used to be called `pg_loexport()`.

See also pg_lo_import().

pg_lo_import (PHP 4 >= 4.2.0)

Import a large object from file

int **pg_lo_import** ([resource connection, string pathname]) \linebreak

In versions before PHP 4.2.0 the syntax of this function was different, see the following definition:

int **pg_lo_import** ([resource connection, string pathname]) \linebreak

The *`pathname`* argument specifies the pathname of the file to be imported as a large object. It returns `FALSE` if an error occurred, oid of the just created large object otherwise.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Megjegyzés:** Ha a safe mode be van kapcsolva, a PHP ellenőrzi, hogy az állományok/könyvtárak, amelyekkel dolgozni szeretnél, ugyanazzal a felhasználói azonosítóval (UID) rendelkeznek-e, mint az éppen futó program.

> **Megjegyzés:** This function used to be called `pg_loimport()`.

See also pg_lo_export() and pg_lo_open().

pg_lo_open (PHP 4 >= 4.2.0)

Open a large object

resource **pg_lo_open** (resource connection, int oid, string mode) \linebreak

pg_lo_open() open a Large Object and returns large object resource. The resource encapsulates information about the connection. *`oid`* specifies a valid large object oid and *`mode`* can be either "r", "w", or "rw". It returns FALSE if there is an error.

> **Figyelem**
>
> Do not close the database connection before closing the large object resource.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Megjegyzés:** This function used to be called `pg_loopen()`.

See also pg_lo_close() and pg_lo_create().

pg_lo_read_all (PHP 4 >= 4.2.0)

Read a entire large object and send straight to browser

int **pg_lo_read_all** (resource large_object) \linebreak

pg_lo_read_all() reads a large object and passes it straight through to the browser after sending all pending headers. Mainly intended for sending binary data like images or sound. It returns number of bytes read. It returns FALSE, if an error occurred.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Megjegyzés:** This function used to be called `pg_loreadall()`.

See also pg_lo_read().

pg_lo_read (PHP 4 >= 4.2.0)

Read a large object

string **pg_lo_read** (resource large_object, int len) \linebreak

pg_lo_read() reads at most *len* bytes from a large object and returns it as a string. *large_object* specifies a valid large object resource and *len* specifies the maximum allowable size of the large object segment. It returns FALSE if there is an error.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Megjegyzés: This function used to be called `pg_loread()`.

See also pg_lo_read_all().

pg_lo_seek (PHP 4 >= 4.2.0)

Seeks position of large object

bool **pg_lo_seek** (resource large_object, int offset [, int whence]) \linebreak

pg_lo_seek() seeks position of large object resource. *whence* is PGSQL_SEEK_SET, PGSQL_SEEK_CUR or PGSQL_SEEK_END.

See also pg_lo_tell().

pg_lo_tell (PHP 4 >= 4.2.0)

Returns current position of large object

int **pg_lo_tell** (resource large_object) \linebreak

pg_lo_tell() returns current position (offset from the beginning of large object).

See also pg_lo_seek().

pg_lo_unlink (PHP 4 >= 4.2.0)

Delete a large object

bool **pg_lo_unlink** (resource connection, int oid) \linebreak

pg_lo_unlink() deletes a large object with the *oid*. It returns TRUE on success, otherwise returns FALSE.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Megjegyzés: This function used to be called pg_lo_unlink().

See also pg_lo_create() and pg_lo_import().

pg_lo_write (PHP 4 >= 4.2.0)

Write a large object

int **pg_lo_write** (resource large_object, string data) \linebreak

pg_lo_write() writes at most to a large object from a variable *data* and returns the number of bytes actually written, or FALSE in the case of an error. *large_object* is a large object resource from pg_lo_open().

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Megjegyzés: This function used to be called pg_lo_write().

See also pg_lo_create() and pg_lo_open().

pg_metadata (PHP 4 CVS only)

Get metadata for table.

array **pg_metadata** (resource connection, string table_name) \linebreak

pg_metadata() returns table definition for table_name as array. If there is error, it returns FALSE

Megjegyzés: This function is experimental.

See also pg_convert()

pg_num_fields (PHP 4 >= 4.2.0)

Returns the number of fields

int **pg_num_fields** (resource result) \linebreak

pg_num_fields() returns the number of fields (columns) in a PostgreSQL *result*. The argument is a result resource returned by pg_query(). This function will return -1 on error.

Megjegyzés: This function used to be called `pg_numfields()`.

See also pg_num_rows() and pg_affected_rows().

pg_num_rows (PHP 4 >= 4.2.0)

Returns the number of rows

int **pg_num_rows** (resource result) \linebreak

pg_num_rows() will return the number of rows in a PostgreSQL *`result`* resource. *`result`* is a query result resource returned by pg_query(). This function will return -1 on error.

Megjegyzés: Use pg_affected_rows() to get number of rows affected by INSERT, UPDATE and DELETE query.

Megjegyzés: This function used to be called `pg_numrows()`.

See also pg_num_fields() and pg_affected_rows().

pg_Options (PHP 3, PHP 4)

Returns options

string **pg_options** (int connection_id) \linebreak

A **pg_Options()** függvény egy stringgel tér vissza, amiben az adott PostgreSQL azonosító kapcsolat opcióit tartalmazza.

pg_pConnect (PHP 3, PHP 4)

Tartós adatbázis-kapcsolatot hoz létre

int **pg_pconnect** (string host, string port, string options, string tty, string dbname) \linebreak

Siker esetén egy kapcsolat-indexszel tér vissza, vagy hamissal, ha a kapcsolat nem hozható létre. Tartós kapcsolatot nyit a PostgreSQL adatbázishoz. Minden paraméternek idézőjelezettnek kell lennie, beleértve a portszámot. Az options és a tty paraméterek elhagyhatók. A függvény egy olyan kapcsolat-azonosítóval tér vissza, amely szükséges más PostgreSQL függvények végrehajtásához. Egyszerre több tartós kapcsolatod is lehet nyitva egyszerre. [Hát igen, ez a Postgres még a valós életet is túlszárnyalja...] Lásd még a **pg_Connect()** függvényt.

Kapcsolat a következő módon is létrehozható: **$conn = pg_pconnect("dbname=marliese port=5432");** Egyéb paraméterek *dbname* és *port* are[???] *host*, *tty*, *options*, *user* és *password*.

pg_Port (PHP 3, PHP 4)

Melyik porton van a kapcsolat

int **pg_port** (int connection_id) \linebreak

A **pg_Port()** függvény az adott PostgreSQL kapcsolat-azonosítójú összeköttetés portszámát adja meg.

pg_put_line (PHP 4 >= 4.0.3)

Send a NULL-terminated string to PostgreSQL backend

bool **pg_put_line** ([resource connection, string data]) \linebreak

pg_put_line() sends a NULL-terminated string to the PostgreSQL backend server. This is useful for example for very high-speed inserting of data into a table, initiated by starting a PostgreSQL copy-operation. That final NULL-character is added automatically. Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

> **Megjegyzés:** The application must explicitly send the two characters "\." on the last line to indicate to the backend that it has finished sending its data.

See also pg_end_copy().

Példa 1. High-speed insertion of data into a table

```
<?php
    $conn = pg_pconnect ("dbname=foo");
    pg_query($conn, "create table bar (a int4, b char(16), d float8)");
    pg_query($conn, "copy bar from stdin");
    pg_put_line($conn, "3\thello world\t4.5\n");
    pg_put_line($conn, "4\tgoodbye world\t7.11\n");
    pg_put_line($conn, "\\.\n");
    pg_end_copy($conn);
?>
```

pg_query (PHP 4 >= 4.2.0)

Execute a query

resource **pg_query** (resource connection, string query) \linebreak

pg_query() returns a query result resource if query could be executed. It returns FALSE on failure or if connection is not a valid connection. Details about the error can be retrieved using the pg_last_error() function if connection is valid. pg_last_error() sends an SQL statement to the PostgreSQL database specified by the *connection* resource. The *connection* must be a valid connection that was returned by pg_connect() or pg_pconnect(). The return value of this function is an query result resource to be used to access the results from other PostgreSQL functions such as pg_fetch_array().

Megjegyzés: *connection* is a optional parameter for **pg_query()**. If *connection* is not set, default connection is used. Default connection is the last connection made by pg_connect() or pg_pconnect().

Although *connection* can be omitted, it is not recommended, since it could be a cause of hard to find bug in script.

Megjegyzés: This function used to be called pg_exec(). pg_exec() is still available for compatibility reasons but users are encouraged to use the newer name.

See also pg_connect(), pg_pconnect(), pg_fetch_array(), pg_fetch_object(), pg_num_rows(), and pg_affected_rows().

pg_result_error (PHP 4 >= 4.2.0)

Get error message associated with result

string **pg_result_error** (resource result) \linebreak

pg_result_error() returns error message associated with *result* resource. Therefore, user has better chance to get better error message than pg_last_error().

See also pg_query(), pg_send_query(), pg_get_result(), pg_last_error() and pg_last_notice()

pg_result_status (PHP 4 >= 4.2.0)

Get status of query result

int **pg_result_status** (resource result) \linebreak

pg_result_status() returns status of result resource. Possible retun values are PGSQL_EMPTY_QUERY, PGSQL_COMMAND_OK, PGSQL_TUPLES_OK,

PGSQL_COPY_TO, PGSQL_COPY_FROM, PGSQL_BAD_RESPONSE, PGSQL_NONFATAL_ERROR and PGSQL_FATAL_ERROR.

See also pg_connection_status().

pg_select (PHP 4 CVS only)

Select records.

array **pg_select** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_select() selects records specified by `assoc_array` which has `field=>value`. For successful query, it returns array contains all records and fields that match the condition specified by `assoc_array`. If `options` is specified, pg_convert() s applied to `assoc_array` with specified option.

Példa 1. pg_select

```
<?php
    $db = pg_connect (’dbname=foo’);
    // This is safe, since $_POST is converted autotmatically
    $rec = pg_select($db, ’post_log’, $_POST);
    if ($rec) {
        echo "Records selected\n";
        var_dump($rec);
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Megjegyzés: This function is experimental.

See also pg_convert()

pg_send_query (PHP 4 >= 4.2.0)

Send asynchronous query

bool **pg_send_query** (resource connection, string query) \linebreak bool **pg_send_query** (string query) \linebreak

pg_send_query() send asynchronous query to the *`connection`*. Unlike pg_query(), it can send multiple query to PostgreSQL and get the result one by one using pg_get_result(). Script execution is not block while query is executing. Use pg_connection_busy() to check connection is busy (i.e. query is executing) Query may be canceled by calling pg_cancel_query().

Although, user can send multiple query at once. User cannot send multiple query over busy connection. If query is sent while connection is busy, it waits until last query is finished and discards all result.

See also pg_query(), pg_cancel_query(), pg_get_result() and pg_connection_busy()

pg_set_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Set the client encoding

int **pg_set_client_encoding** ([resource connection, string encoding]) \linebreak

pg_set_client_encoding() sets the client encoding and return 0 if success or -1 if error.

`encoding` is the client encoding and can be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250. Available encoding depends on your PostgreSQL and libpq version. Refer to PostgreSQL manual for supported encodings for your PostgreSQL.

> **Megjegyzés:** This function requires PHP-4.0.3 or higher and PostgreSQL-7.0 or higher. Supported encoding depends on PostgreSQL version. Refer to PostgreSQL manual for details.
>
> The function used to be called **pg_setclientencoding()**.

See also pg_client_encoding().

pg_trace (PHP 4 >= 4.0.1)

PostgreSQL szerver kapcsolatot követ nyomon

bool **pg_trace** (string filename [, string mode [, int connection]]) \linebreak

Engedélyezi a PostgreSQL felület kommunikációinak nyomkövetését egy fileba. Ahhoz, hogy megértsd az eredményeket, meg kell barátkoznod a PostgreSQL kommunikációs protokolljával. Ha nem ismeret a protokollt, hasznos lehet akkor is nyomon követni a szerverhez küldött kérések hibáit; például 'grep '^To backend' trace.log' paranccsal megnézni, hogy milyen kéréseket küldtek a PostgreSQL szervernek.

A *`filename`* és a *`mode`* paraméterek azonosak a fopen() függvényéhez (a *`mode`* alapértelmezése 'w'), *`connection`* határozza meg a nyomon követendő kapcsolatot; az utoljára megnyitot kapcsolat az alapértelmezett.

TRUE-val tér vissza, ha a *`filename`* paraméterben megadott filet meg lehet nyitni loggolásra, egyébként FALSE-szal tér vissza.

Lásd még a fopen() és a pg_untrace() függvényeket.

pg_tty (PHP 3, PHP 4)

A tty nevét adja vissza

string **pg_tty** (int connection_id) \linebreak

A **pg_tty()** függvény a tty nevét adja vissza, amelyre az adott azonosítójú kapcsolat szerver oldali nyomkövetésének kimenete kerül.

pg_untrace (PHP 4 >= 4.0.1)

Abbahagyja az adott kapcsolat nyomkövetését

bool **pg_untrace** ([int connection]) \linebreak

A pg_trace() által elindított nyomkövetést szünteti meg. A *`connection`* paraméter határozza meg, hogy mely kapcsolat nyomkövetését kell abbahagyni. Az alapértelmezés az utoljára megnyitott kapcsolat.

Mindig TRUE-val tér vissza.

Lásd még a pg_trace() függvényt.

pg_update (PHP 4 CVS only)

Update table.

long **pg_update** (resource connection, string table_name, array condition, array data [, int options]) \linebreak

pg_update() updates records that matches `condition` with `data` If `options` is specified, pg_convert() is applied to `assoc_array` with specified options.

Példa 1. pg_update

```
<?php
    $db = pg_connect (’dbname=foo’);
    $data = array(’field1’=>’AA’, ’field2’=>’BB’);
    // This is safe, since $_POST is converted automatically
    $res = pg_update($db, ’post_log’, $_POST, $data);
    if ($res) {
        echo "Data is updated: $res\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Megjegyzés: This function is experimental.

See also pg_convert()

LXXXI. Process Control Functions

Process Control support in PHP implements the Unix style of process creation, program execution, signal handling and process termination. Process Control should not be enabled within a webserver environment and unexpected results may happen if any Process Control functions are used within a webserver environment.

This documentation is intended to explain the general usage of each of the Process Control functions. For detailed information about Unix process control you are encouraged to consult your systems documentation including fork(2), waitpid(2) and signal(2) or a comprehensive reference such as Advanced Programming in the UNIX Environment by W. Richard Stevens (Addison-Wesley).

Process Control support in PHP is not enabled by default. You will need to use the --enable-pcntl configuration option when compiling PHP to enable Process Control support.

> **Megjegyzés:** Currently, this module will not function on non-Unix platforms (Windows).

The following list of signals are supported by the Process Control functions. Please see your systems signal(7) man page for details of the default behavior of these signals.

Táblázat 1. Supported Signals

SIGFPE	SIGCONT	SIGKILL
SIGSTOP	SIGUSR1	SIGTSTP
SIGHUP	SIGUSR2	SIGTTIN
SIGINT	SIGSEGV	SIGTTOU
SIGQUIT	SIGPIPE	SIGURG
SIGILL	SIGALRM	SIGXCPU
SIGTRAP	SIGTERM	SIGXFSZ
SIGABRT	SIGSTKFLT	SIGVTALRM
SIGIOT	SIGCHLD	SIGPROF
SIGBUS	SIGCLD	SIGWINCH
SIGPOLL	SIGIO	SIGPWR
SIGSYS		

Process Control Example

This example forks off a daemon process with a signal handler.

Példa 1. Process Control Example

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
```

```
     die("could not fork");
} else if ($pid) {
     exit(); // we are the parent
} else {
     // we are the child
}

// detatch from the controlling terminal
if (!posix_setsid()) {
    die("could not detach from terminal");
}

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");

// loop forever performing tasks
while(1) {

    // do something interesting here

}

function sig_handler($signo) {

     switch($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
             // handle restart tasks
             break;
         default:
             // handle all other signals
     }

}

?>
```

pcntl_exec (PHP 4 >= 4.2.0)

Executes specified program in current process space

bool **pcntl_exec** (string path [, array args [, array envs]]) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

pcntl_fork (PHP 4 >= 4.1.0)

Forks the currently running process

int **pcntl_fork** (void) \linebreak

The **pcntl_fork()** function creates a child process that differs from the parent process only in it's PID and PPID. Please see your system's fork(2) man page for specific details as to how fork works on your system.

On success, the PID of the child process is returned in the parent's thread of execution, and a 0 is returned in the child's thread of execution. On failure, a -1 will be returned in the parent's context, no child process will be created, and a PHP error is raised.

Példa 1. pcntl_fork() Example

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
     die("could not fork");
} else if ($pid) {
     // we are the parent
} else {
     // we are the child
}

?>
```

See also pcntl_waitpid() and pcntl_signal().

pcntl_signal (PHP 4 >= 4.1.0)

Installs a signal handler

bool **pcntl_signal** (int signo, mixed handle) \linebreak

The **pcntl_signal()** function installs a new signal handler for the signal indicated by *signo*. The signal handler is set to *handler* which may be the name of a user created function, or either of the two global constants SIG_IGN or SIG_DFL.

pcntl_signal() returns TRUE on success or FALSE on failure.

Példa 1. pcntl_signal() Example

```
<?php

// signal handler function
function sig_handler($signo) {

     switch($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
             // handle restart tasks
             break;
         case SIGUSR1:
             print "Caught SIGUSR1...\n";
             break;
         default:
             // handle all other signals
     }

}

print "Installing signal handler...\n";

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");
pcntl_signal(SIGUSR1, "sig_handler");

print "Generating signal SIGTERM to self...\n";

// send SIGUSR1 to current process id
posix_kill(posix_getpid(), SIGUSR1);

print "Done\n"

?>
```

See also pcntl_fork() and pcntl_waitpid().

pcntl_waitpid (PHP 4 >= 4.1.0)

Waits on or returns the status of a forked child

int **pcntl_waitpid** (int pid, int status, int options) \linebreak

The **pcntl_waitpid()** function suspends execution of the current process until a child as specified by the *pid* argument has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal handling function. If a child as requested by *pid* has already exited by the time of the call (a so-called "zombie" process), the function returns immediately. Any system resources used by the child are freed. Please see your system's waitpid(2) man page for specific details as to how waitpid works on your system.

pcntl_waitpid() returns the process ID of the child which exited, -1 on error or zero if WNOHANG was used and no child was available

The value of *pid* can be one of the following:

Táblázat 1. possible values for *pid*

< -1	wait for any child process whose process group ID is equal to the absolute value of *pid*.
-1	wait for any child process; this is the same behaviour that the wait function exhibits.
0	wait for any child process whose process group ID is equal to that of the calling process.
> 0	wait for the child whose process ID is equal to the value of *pid*.

pcntl_waitpid() will store status information in the *status* parameter which can be evaluated using the following functions: pcntl_wifexited(), pcntl_wifstopped(), pcntl_wifsignaled(), pcntl_wexitstatus(), pcntl_wtermsig() and pcntl_wstopsig().

The value of *options* is the value of zero or more of the following two global constants OR'ed together:

Táblázat 2. possible values for *options*

WNOHANG	return immediately if no child has exited.
WUNTRACED	return for children which are stopped, and whose status has not been reported.

See also pcntl_fork(), pcntl_signal(), pcntl_wifexited(), pcntl_wifstopped(), pcntl_wifsignaled(), pcntl_wexitstatus(), pcntl_wtermsig() and pcntl_wstopsig().

pcntl_wexitstatus (PHP 4 >= 4.1.0)

Returns the return code of a terminated child

int **pcntl_wexitstatus** (int status) \linebreak

Returns the return code of a terminated child. This function is only useful if pcntl_wifexited() returned `TRUE`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wifexited().

pcntl_wifexited (PHP 4 >= 4.1.0)

Returns `TRUE` if status code represents a successful exit

int **pcntl_wifexited** (int status) \linebreak

Returns `TRUE` if the child status code represents a successful exit.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wexitstatus().

pcntl_wifsignaled (PHP 4 >= 4.1.0)

Returns `TRUE` if status code represents a termination due to a signal

int **pcntl_wifsignaled** (int status) \linebreak

Returns `TRUE` if the child process exited because of a signal which was not caught.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_signal().

pcntl_wifstopped (PHP 4 >= 4.1.0)

Returns `TRUE` if child process is currently stopped

int **pcntl_wifstopped** (int status) \linebreak

Returns `TRUE` if the child process which caused the return is currently stopped; this is only possible if the call to pcntl_waitpid() was done using the option `WUNTRACED`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid().

pcntl_wstopsig (PHP 4 >= 4.1.0)

Returns the signal which caused the child to stop

int **pcntl_wstopsig** (int status) \linebreak

Returns the number of the signal which caused the child to stop. This function is only useful if pcntl_wifstopped() returned `TRUE`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wifstopped().

pcntl_wtermsig (PHP 4 >= 4.1.0)

Returns the signal which caused the child to terminate

int **pcntl_wtermsig** (int status) \linebreak

Returns the number of the signal that caused the child process to terminate. This function is only useful if pcntl_wifsignaled() returned `TRUE`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid(), pcntl_signal() and pcntl_wifsignaled().

LXXXII. Programfuttató függvények

Ezek a függvények különböző külső program futtatására adnak lehetőséget, vagy segítik biztonságosabbá tenni ezeket a hívásokat. Érdemes megnézned a végrehajtó operátort is.

escapeshellarg (PHP 4 >= 4.0.3)

Shell paraméterként átadandó karakterláncot állít elő

string **escapeshellarg** (string arg) \linebreak

Az **escapeshellarg()** aposztrófok közé fogja az *arg* paraméterben átadott szöveget és megkettőzi a benne lévő aposztrófokat, így egy parancsértelmező függvények önálló paramétereként használható karaktersorozatot ad vissza. Ez a függvény használatos a felhasználótól érkező egyedi parancsértelmező függvények paramétereinek biztonságossá alakítására. Ezek a fügvények : az exec(), a system() és a végrehajtó operátor. Jellemző használata:

```
system("ls ".escapeshellarg($dir));
```

Lásd még: exec(), popen(), system() és a végrehajtó operátor!

escapeshellcmd (PHP 3, PHP 4)

Shell metakaraktereket alakít át

string **escapeshellcmd** (string command) \linebreak

Az **escapeshellcmd()** escape-et alkalmaz minden speciális karakterre, ami nem kívánt parancsok futtatását okozná. Ez a függvény arra használható, hogy biztosítsd a felhasználótól érkező adat tisztaságát egy exec() vagy egy system() függényhívásnál, vagy egy végrehajtó operátorban használnád. Általános használati forma:

```
$e = escapeshellcmd($parancs);
system("echo $e"); // itt mindegy, hogy az $e tartalmaz-e szóközt
$f = escapeshellcmd($filenev);
system("touch \"/tmp/$f\"; ls -l \"/tmp/$f\""); // itt viszont nem, ezért van idézőjelber
```

Lásd még: escapeshellarg(), exec(), popen(), system() és a végrehajtó operátor!

exec (PHP 3, PHP 4)

Külső programot futtat

string **exec** (string command [, string array [, int return_var]]) \linebreak

Az **exec()** lefuttatja a *command*-ban megadott parancsot, amely így nem ír ki semmit, csak egyszerűen visszaadja a legutolsó sort, amit a parancs generált. A passthru() függvény használható arra, hogy a parancs eredménye minden adatfeldolgozás nélkül kiírása kerüljön.

Az *array* paraméter megadásával a program kimenetét a PHP soronként a megadott tömb végéhez fűzi. Ha az átadott tömb már tartalmaz néhány elemet, akkor az **exec()** a benne lévő adatok után fűzi a sorokat. Ha nincs szükség a régi elemekre, akkor az unset() függvénnyel lehet törölni az egész tömböt, az **exec()** meghívása előtt.

Az *array* paraméterrel együtt a *return_var* a futtatott parancs visszatérési állapotát tárolja el.

Figyelem

Ha felhasználótól származó adatot kell átadni ennek a függvénynek, akkor előtte használd az escapeshellcmd() függvényt, annak biztosítására, hogy nem tudnak a felhasználók tetszőleges parancsot futtatni.

Megjegyzés: Ezzel a függvénnyel elindított de háttérben futó program kimenetét mindenképpen fájlba vagy valamilyen más kimeneti csatornába kell átirányítani. Ellenkező esetben a PHP várakozni fog addig, amíg a program futása be nem fejeződik.

Lásd még: system(), passthru(), popen(), escapeshellcmd() és a végrehajtó operátor!

passthru (PHP 3, PHP 4)

Külső programot futtat, és a kimenetet feldolgozás nélküli jeleníti meg

void **passthru** (string command [, int return_var]) \linebreak

A **passthru()** függvény hasonló az exec()-hez, ugyanis a *command* paraméterben megadott parancsot futtatja. A *return_var* paraméterben a parancs visszatérési állapota kerül. Ez a függvény használható az exec() vagy a system() helyett, ha a parancs kimenete bináris adat, amit közvetlenül a böngészőnek kell visszaküldeni. Tipikusan például a pbmplus eszközök futtatására használható, amelyek képesek közvetlenül kép visszaadására. A `Content-type` fejlécet *image/gif*-re állítva és a pbmplus programot meghívva, az visszaad egy képet, és így olyan PHP programot írhatók, amelyek közvetlenül képeket adnak vissza.

Figyelem

Ha felhasználótól származó adatot kell átadni ennek a függvénynek, akkor előtte használd az escapeshellcmd() függvényt, annak biztosítására, hogy nem tudnak a felhasználók tetszőleges parancsot futtatni.

Megjegyzés: Ezzel a függvénnyel elindított de háttérben futó program kimenetét mindenképpen fájlba vagy valamilyen más kimeneti csatornába kell átirányítani. Ellenkező esetben a PHP várakozni fog addig, amíg a program futása be nem fejeződik.

Lásd még: exec(), system(), popen(), escapeshellcmd() és a végrehajtó operátor!

proc_close (PHP 4 CVS only)

Close a process opened by proc_open and return the exit code of that process.

int **proc_close** (resource process) \linebreak

proc_close() is similar to popen() except that it only works on processes opened by proc_open(). **proc_close()** waits for the process to terminate, and returns it's exit code. If you have open pipes to that process, you should fclose() them prior to calling this function in order to avoid a deadlock - the child process may not be able to exit while the pipes are open.

proc_open (PHP 4 CVS only)

Execute a command and open file pointers for input/output

resource **proc_open** (string cmd, array descriptorspec, array pipes) \linebreak

proc_open() is similar to popen() but provides a much greater degree of control over the program execution. *`cmd`* is the command to be executed by the shell. *`descriptorspec`* is an indexed array where the key represents the descriptor number and the value represents how PHP will pass that descriptor to the child process. *`pipes`* will be set to an indexed array of file pointers that correspond to PHP's end of any pipes that are created. The return value is a resource representing the process; you should free it using proc_close() when you are finished with it.

```
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin is a pipe that the child will read from
   1 => array("pipe", "w"),  // stdout is a pipe that the child will write to
   2 => array("file", "/tmp/error-output.txt", "a"), // stderr is a file to write to
);
$process = proc_open("php", $descriptorspec, $pipes);
if (is_resource($process)) {
    // $pipes now looks like this:
    // 0 => writeable handle connected to child stdin
    // 1 => readable handle connected to child stdout
    // Any error output will be appended to /tmp/error-output.txt

    fwrite($pipes[0], "<?php echo \"Hello World!\"; ?>");
    fclose($pipes[0]);

    while(!feof($pipes[1])) {
        echo fgets($pipes[1], 1024);
    }
    fclose($pipes[1]);
    // It is important that you close any pipes before calling
    // proc_close in order to avoid a deadlock
    $return_value = proc_close($process);

    echo "command returned $return_value\n";
}
```

The file descriptor numbers in *descriptorspec* are not limited to 0, 1 and 2 - you may specify any valid file descriptor number and it will be passed to the child process. This allows your script to interoperate with other scripts that run as "co-processes". In particular, this is useful for passing passphrases to programs like PGP, GPG and openssl in a more secure manner. It is also useful for reading status information provided by those programs on auxillary file descriptors.

Megjegyzés: Windows compatibility: Descriptors beyond 2 (stderr) are made available to the child process as inheritable handles, but since the Windows architecture does not associate file descriptor numbers with low-level handles, the child process does not (yet) have a means of accessing those handles. Stdin, stdout and stderr work as expected.

Megjegyzés: This function was introduced in PHP 4.3.0.

Megjegyzés: If you only need a uni-directional (one-way) process pipe, use popen() instead, as it is much easier to use.

See also exec(), system(), passthru(), popen(), escapeshellcmd(), and the backtick operator.

shell_exec (PHP 4)

lefuttat parancsértelmezőben egy parancsot és a teljes kimenetet visszaadja

string **shell_exec** (string cmd) \linebreak

Ez a függvény azonos a végrehajtó operátorral.

system (PHP 3, PHP 4)

Külső programot futtat, és a kimenetet megjeleníti

string **system** (string command [, int return_var]) \linebreak

A **system()** függvény olyan, mint C-beli társa, azaz a *command* paraméterben megadott parancsot futtaja, és kiírja a kimenetet. A *return_var* paraméterben a parancs visszatérési állapota kerülhet, ha szerepel.

Figyelem

Ha felhasználótól származó adatot kell átadni ennek a függvénynek, akkor előtte használd az escapeshellcmd() függvényt, annak biztosítására, hogy nem tudnak a felhasználók tetszőleges parancsot futtatni.

> **Megjegyzés:** Ezzel a függvénnyel elindított de háttérben futó program kimenetét mindenképpen fájlba vagy valamilyen más kimeneti csatornába kell átirányítani. Ellenkező esetben a PHP várakozni fog addig, amíg a program futása be nem fejeződik.

A **system()** függvény automatikusan üríteni próbálja a szerver kimeneti pufferét minden kiadott sor után, ha a PHP szerver modulként fut.

Visszaadja a parancs kimenetének utolsó sorát siker esetén, vagy FALSE értéket ha hiba történik.

A passthru() függvény használható arra, hogy a parancs eredménye minden adatfeldolgozás nélkül kiírása kerüljön.

Lásd még: exec(), passthru(), popen(), escapeshellcmd() és a végrehajtó operátor!

LXXXIII. Printer functions

These functions are only available under Windows 9.x, ME, NT4 and 2000. They have been added in PHP 4 (4.0.4).

printer_abort (unknown)

Deletes the printer’s spool file

void **printer_abort** (resource handle) \linebreak

This function deletes the printers spool file.

handle must be a valid handle to a printer.

Példa 1. printer_abort() example

```
$handle = printer_open();
printer_abort($handle);
printer_close($handle);
```

printer_close (unknown)

Close an open printer connection

void **printer_close** (resource handle) \linebreak

This function closes the printer connection. **printer_close()** also closes the active device context.

handle must be a valid handle to a printer.

Példa 1. printer_close() example

```
$handle = printer_open();
printer_close($handle);
```

printer_create_brush (unknown)

Create a new brush

mixed **printer_create_brush** (int style, string color) \linebreak

The function creates a new brush and returns a handle to it. A brush is used to fill shapes. For an example see printer_select_brush(). *color* must be a color in RGB hex format, i.e. "000000" for black, *style* must be one of the following constants:

- *PRINTER_BRUSH_SOLID*: creates a brush with a solid color.
- *PRINTER_BRUSH_DIAGONAL*: creates a brush with a 45-degree upward left-to-right hatch (/).
- *PRINTER_BRUSH_CROSS*: creates a brush with a cross hatch (+).

- *PRINTER_BRUSH_DIAGCROSS*: creates a brush with a 45 cross hatch (x).
- *PRINTER_BRUSH_FDIAGONAL*: creates a brush with a 45-degree downward left-to-right hatch (\ ).
- *PRINTER_BRUSH_HORIZONTAL*: creates a brush with a horizontal hatch (-).
- *PRINTER_BRUSH_VERTICAL*: creates a brush with a vertical hatch (|).
- *PRINTER_BRUSH_CUSTOM*: creates a custom brush from an BMP file. The second parameter is used to specify the BMP instead of the RGB color code.

printer_create_dc (unknown)

Create a new device context

void **printer_create_dc** (resource handle) \linebreak

The function creates a new device context. A device context is used to customize the graphic objects of the document. *handle* must be a valid handle to a printer.

Példa 1. printer_create_dc() example

```
$handle = printer_open();
printer_start_doc($handle);
printer_start_page($handle);

printer_create_dc($handle);
/* do some stuff with the dc */
printer_set_option($handle, PRINTER_TEXT_COLOR, "333333");
printer_draw_text($handle, 1, 1, "text");
printer_delete_dc($handle);

/* create another dc */
printer_create_dc($handle);
printer_set_option($handle, PRINTER_TEXT_COLOR, "000000");
printer_draw_text($handle, 1, 1, "text");
/* do some stuff with the dc */

printer_delete_dc($handle);

printer_endpage($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_create_font (unknown)

Create a new font

mixed **printer_create_font** (string face, int height, int width, int font_weight, bool italic, bool underline, bool strikeout, int orientaton) \linebreak

The function creates a new font and returns a handle to it. A font is used to draw text. For an example see printer_select_font(). *face* must be a string specifying the font face. *height* specifies the font height, and *width* the font width. The *font_weight* specifies the font weight (400 is normal), and can be one of the following predefined constants.

- *PRINTER_FW_THIN*: sets the font weight to thin (100).
- *PRINTER_FW_ULTRALIGHT*: sets the font weight to ultra light (200).
- *PRINTER_FW_LIGHT*: sets the font weight to light (300).
- *PRINTER_FW_NORMAL*: sets the font weight to normal (400).
- *PRINTER_FW_MEDIUM*: sets the font weight to medium (500).
- *PRINTER_FW_BOLD*: sets the font weight to bold (700).
- *PRINTER_FW_ULTRABOLD*: sets the font weight to ultra bold (800).
- *PRINTER_FW_HEAVY*: sets the font weight to heavy (900).

italic can be TRUE or FALSE, and sets whether the font should be italic.

underline can be TRUE or FALSE, and sets whether the font should be underlined.

strikeout can be TRUE or FALSE, and sets whether the font should be striked out.

orientation specifies a rotation. For an example see printer_select_font().

printer_create_pen (unknown)

Create a new pen

mixed **printer_create_pen** (int style, int width, string color) \linebreak

The function creates a new pen and returns a handle to it. A pen is used to draw lines and curves. For an example see printer_select_pen(). *color* must be a color in RGB hex format, i.e. "000000" for black, *width* specifies the width of the pen whereas *style* must be one of the following constants:

- *PRINTER_PEN_SOLID*: creates a solid pen.
- *PRINTER_PEN_DASH*: creates a dashed pen.
- *PRINTER_PEN_DOT*: creates a dotted pen.
- *PRINTER_PEN_DASHDOT*: creates a pen with dashes and dots.
- *PRINTER_PEN_DASHDOTDOT*: creates a pen with dashes and double dots.
- *PRINTER_PEN_INVISIBLE*: creates an invisible pen.

printer_delete_brush (unknown)

Delete a brush

bool **printer_delete_brush** (resource handle) \linebreak

The function deletes the selected brush. For an example see printer_select_brush(). It returns `TRUE` on success, or `FALSE` otherwise. *`handle`* must be a valid handle to a brush.

printer_delete_dc (unknown)

Delete a device context

bool **printer_delete_dc** (resource handle) \linebreak

The function deletes the device context and returns `TRUE` on success, or `FALSE` if an error occurred. For an example see printer_create_dc(). *`handle`* must be a valid handle to a printer.

printer_delete_font (unknown)

Delete a font

bool **printer_delete_font** (resource handle) \linebreak

The function deletes the selected font. For an example see printer_select_font(). It returns `TRUE` on success, or `FALSE` otherwise. *`handle`* must be a valid handle to a font.

printer_delete_pen (unknown)

Delete a pen

bool **printer_delete_pen** (resource handle) \linebreak

The function deletes the selected pen. For an example see printer_select_pen(). It returns `TRUE` on success, or `FALSE` otherwise. *`handle`* must be a valid handle to a pen.

printer_draw_bmp (unknown)

Draw a bmp

void **printer_draw_bmp** (resource handle, string filename, int x, int y) \linebreak

The function simply draws an bmp the bitmap *`filename`* at position *`x`*, *`y`*. *`handle`* must be a valid handle to a printer.

The function returns `TRUE` on success, or otherwise `FALSE`.

Példa 1. printer_draw_bmp() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_draw_bmp($handle, "c:\\image.bmp", 1, 1);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_chord (unknown)

Draw a chord

void **printer_draw_chord** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad_x, int rad_y, int rad_x1, int rad_y1) \linebreak

The function simply draws an chord. *`handle`* must be a valid handle to a printer.

`rec_x` is the upper left x coordinate of the bounding rectangle.

`rec_y` is the upper left y coordinate of the bounding rectangle.

`rec_x1` is the lower right x coordinate of the bounding rectangle.

`rec_y1` is the lower right y coordinate of the bounding rectangle.

`rad_x` is x coordinate of the radial defining the beginning of the chord.

`rad_y` is y coordinate of the radial defining the beginning of the chord.

`rad_x1` is x coordinate of the radial defining the end of the chord.

`rad_y1` is y coordinate of the radial defining the end of the chord.

Példa 1. printer_draw_chord() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_chord($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);
```

```
printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_elipse (unknown)

Draw an ellipse

void **printer_draw_elipse** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y) \linebreak

The function simply draws an ellipse. *handle* must be a valid handle to a printer.

ul_x is the upper left x coordinate of the ellipse.

ul_y is the upper left y coordinate of the ellipse.

lr_x is the lower right x coordinate of the ellipse.

lr_y is the lower right y coordinate of the ellipse.

Példa 1. printer_draw_elipse() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_elipse($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_line (unknown)

Draw a line

void **printer_draw_line** (resource printer_handle, int from_x, int from_y, int to_x, int to_y) \linebreak

The function simply draws a line from position *`from_x`*, *`from_y`* to position *`to_x`*, *`to_y`* using the selected pen. *`printer_handle`* must be a valid handle to a printer.

Példa 1. printer_draw_line() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "000000");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 10, 1000, 10);
printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_pie (unknown)

Draw a pie

void **printer_draw_pie** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad1_x, int rad1_y, int rad2_x, int rad2_y) \linebreak

The function simply draws an pie. *`handle`* must be a valid handle to a printer.

`rec_x` is the upper left x coordinate of the bounding rectangle.

`rec_y` is the upper left y coordinate of the bounding rectangle.

`rec_x1` is the lower right x coordinate of the bounding rectangle.

`rec_y1` is the lower right y coordinate of the bounding rectangle.

`rad1_x` is x coordinate of the first radial's ending.

`rad1_y` is y coordinate of the first radial's ending.

`rad2_x` is x coordinate of the second radial's ending.

`rad2_y` is y coordinate of the second radial's ending.

Példa 1. printer_draw_pie() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
```

```
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_pie($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_rectangle (unknown)

Draw a rectangle

void **printer_draw_rectangle** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y) \linebreak

The function simply draws a rectangle.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

Példa 1. printer_draw_rectangle() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_roundrect (unknown)

Draw a rectangle with rounded corners

void **printer_draw_roundrect** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y, int width, int height) \linebreak

The function simply draws a rectangle with rounded corners.

`handle` must be a valid handle to a printer.

`ul_x` is the upper left x coordinate of the rectangle.

`ul_y` is the upper left y coordinate of the rectangle.

`lr_x` is the lower right x coordinate of the rectangle.

`lr_y` is the lower right y coordinate of the rectangle.

`width` is the width of the ellipse.

`height` is the height of the ellipse.

Példa 1. printer_draw_roundrect() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_roundrect($handle, 1, 1, 500, 500, 200, 200);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_text (unknown)

Draw text

void **printer_draw_text** (resource printer_handle, string text, int x, int y) \linebreak

The function simply draws *`text`* at position *`x`*, *`y`* using the selected font. *`printer_handle`* must be a valid handle to a printer.

Példa 1. printer_draw_text() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial",72,48,400,false,false,false,0);
printer_select_font($handle, $font);
printer_draw_text($handle, "test", 10, 10);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_end_doc (unknown)

Close document

bool **printer_end_doc** (resource handle) \linebreak

Closes a new document in the printer spooler. The document is now ready for printing. For an example see printer_start_doc(). *`handle`* must be a valid handle to a printer.

printer_end_page (unknown)

Close active page

bool **printer_end_page** (resource handle) \linebreak

The function closes the active page in the active document. For an example see printer_start_doc(). *`handle`* must be a valid handle to a printer.

printer_get_option (unknown)

Retrieve printer configuration data

mixed **printer_get_option** (resource handle, string option) \linebreak

The function retrieves the configuration setting of *`option`*. *`handle`* must be a valid handle to a printer. Take a look at printer_set_option() for the settings that can be retrieved, additionally the following settings can be retrieved:

- *PRINTER_DEVICENAME* returns the devicename of the printer.
- *PRINTER_DRIVERVERSION* returns the printer driver version.

Példa 1. printer_get_option() example

```
$handle = printer_open();
print printer_get_option($handle, PRINTER_DRIVERVERSION);
printer_close($handle);
```

printer_list (unknown)

Return an array of printers attached to the server

array **printer_list** (int enumtype [, string name [, int level]]) \linebreak

The function enumerates available printers and their capabilities. *level* sets the level of information request. Can be 1,2,4 or 5. *enumtype* must be one of the following predefined constants:

- *PRINTER_ENUM_LOCAL*: enumerates the locally installed printers.
- *PRINTER_ENUM_NAME*: enumerates the printer of *name*, can be a server, domain or print provider.
- *PRINTER_ENUM_SHARED*: this parameter can't be used alone, it has to be OR'ed with other parameters, i.e. PRINTER_ENUM_LOCAL to detect the locally shared printers.
- *PRINTER_ENUM_DEFAULT*: (Win9.x only) enumerates the default printer.
- *PRINTER_ENUM_CONNECTIONS*: (WinNT/2000 only) enumerates the printers to which the user has made connections.
- *PRINTER_ENUM_NETWORK*: (WinNT/2000 only) enumerates network printers in the computer's domain. Only valid if *level* is 1.
- *PRINTER_ENUM_REMOTE*: (WinNT/2000 only) enumerates network printers and print servers in the computer's domain. Only valid if *level* is 1.

Példa 1. printer_list() example

```
/* detect locally shared printer */
var_dump( printer_list(PRINTER_ENUM_LOCAL | PRINTER_ENUM_SHARED) );
```

printer_logical_fontheight (unknown)

Get logical font height

int **printer_logical_fontheight** (resource handle, int height) \linebreak

The function calculates the logical font height of *`height`*. *`handle`* must be a valid handle to a printer.

Példa 1. printer_logical_fontheight() example

```
$handle = printer_open();
print printer_logical_fontheight($handle, 72);
printer_close($handle);
```

printer_open (unknown)

Open connection to a printer

mixed **printer_open** ([string devicename]) \linebreak

This function tries to open a connection to the printer *`devicename`*, and returns a handle on success or FALSE on failure.

If no parameter was given it tries to open a connection to the default printer (if not specified in `php.ini` as `printer.default_printer`, php tries to detect it).

printer_open() also starts a device context.

Példa 1. printer_open() example

```
$handle = printer_open("HP Deskjet 930c");
$handle = printer_open();
```

printer_select_brush (unknown)

Select a brush

void **printer_select_brush** (resource printer_handle, resource brush_handle) \linebreak

The function selects a brush as the active drawing object of the actual device context. A brush is used to fill shapes. If you draw an rectangle the brush is used to draw the shapes, while the pen is used to draw the border. If you haven't selected a brush before drawing shapes, the shape won't be filled. *`printer_handle`* must be a valid handle to a printer. *`brush_handle`* must be a valid handle to a brush.

Példa 1. printer_select_brush() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);
$brush = printer_create_brush(PRINTER_BRUSH_CUSTOM, "c:\\brush.bmp");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1,1,500,500);

printer_delete_brush($brush);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "000000");
printer_select_brush($handle, $brush);
printer_draw_rectangle($handle, 1,501,500,1001);
printer_delete_brush($brush);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_select_font (unknown)

Select a font

void **printer_select_font** (resource printer_handle, resource font_handle) \linebreak

The function selects a font to draw text. *`printer_handle`* must be a valid handle to a printer. *`font_handle`* must be a valid handle to a font.

Példa 1. printer_select_font() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial", 148, 76, PRINTER_FW_MEDIUM, false, false, false, -
50);
printer_select_font($handle, $font);
printer_draw_text($handle, "PHP is simply cool", 40, 40);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_select_pen (unknown)

Select a pen

void **printer_select_pen** (resource printer_handle, resource pen_handle) \linebreak

The function selects a pen as the active drawing object of the actual device context. A pen is used to draw lines and curves. I.e. if you draw a single line the pen is used. If you draw an rectangle the pen is used to draw the borders, while the brush is used to fill the shape. If you haven't selected a pen before drawing shapes, the shape won't be outlined. *`printer_handle`* must be a valid handle to a printer. *`pen_handle`* must be a valid handle to a pen.

Példa 1. printer_select_pen() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "2222FF");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_set_option (unknown)

Configure the printer connection

bool **printer_set_option** (resource handle, int option, mixed value) \linebreak

The function sets the following options for the current connection. *`handle`* must be a valid handle to a printer. For *`option`* can be one of the following constants:

- *`PRINTER_COPIES`*: sets how many copies should be printed, *`value`* must be an integer.
- *`PRINTER_MODE`*: specifies the type of data (text, raw or emf), *`value`* must be a string.
- *`PRINTER_TITLE`*: specifies the name of the document, *`value`* must be a string.
- *`PRINTER_ORIENTATION`*: specifies the orientation of the paper, *`value`* can be either PRINTER_ORIENTATION_PORTRAIT or PRINTER_ORIENTATION_LANDSCAPE

- *PRINTER_RESOLUTION_Y*: specifies the y-resolution in DPI, *value* must be an integer.
- *PRINTER_RESOLUTION_X*: specifies the x-resolution in DPI, *value* must be an integer.
- *PRINTER_PAPER_FORMAT*: specifies the a predefined paper format, set *value* to PRINTER_FORMAT_CUSTOM if you want to specify a custom format with PRINTER_PAPER_WIDTH and PRINTER_PAPER_LENGTH. *value* can be one of the following constants.
  - *PRINTER_FORMAT_CUSTOM*: let's you specify a custom paper format.
  - *PRINTER_FORMAT_LETTER*: specifies standard letter format (8 1/2- by 11-inches).
  - *PRINTER_FORMAT_LETTER*: specifies standard legal format (8 1/2- by 14-inches).
  - *PRINTER_FORMAT_A3*: specifies standard A3 format (297- by 420-millimeters).
  - *PRINTER_FORMAT_A4*: specifies standard A4 format (210- by 297-millimeters).
  - *PRINTER_FORMAT_A5*: specifies standard A5 format (148- by 210-millimeters).
  - *PRINTER_FORMAT_B4*: specifies standard B4 format (250- by 354-millimeters).
  - *PRINTER_FORMAT_B5*: specifies standard B5 format (182- by 257-millimeter).
  - *PRINTER_FORMAT_FOLIO*: specifies standard FOLIO format (8 1/2- by 13-inch).

- *PRINTER_PAPER_LENGTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_LENGTH specifies a custom paper length in mm, *value* must be an integer.
- *PRINTER_PAPER_WIDTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_WIDTH specifies a custom paper width in mm, *value* must be an integer.
- *PRINTER_SCALE*: specifies the factor by which the printed output is to be scaled. the page size is scaled from the physical page size by a factor of scale/100. for example if you set the scale to 50, the output would be half of it's original size. *value* must be an integer.
- *PRINTER_BACKGROUND_COLOR*: specifies the background color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_COLOR*: specifies the text color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_ALIGN*: specifies the text alignment for the actual device context, *value* can be combined through OR'ing the following constants:
  - *PRINTER_TA_BASELINE*: text will be aligned at the base line.
  - *PRINTER_TA_BOTTOM*: text will be aligned at the bottom.
  - *PRINTER_TA_TOP*: text will be aligned at the top.
  - *PRINTER_TA_CENTER*: text will be aligned at the center.
  - *PRINTER_TA_LEFT*: text will be aligned at the left.
  - *PRINTER_TA_RIGHT*: text will be aligned at the right.

Példa 1. printer_set_option() example

```
$handle = printer_open();
printer_set_option($handle, PRINTER_SCALE, 75);
printer_set_option($handle, PRINTER_TEXT_ALIGN, PRINTER_TA_LEFT);
printer_close($handle);
```

printer_start_doc (unknown)

Start a new document

bool **printer_start_doc** (resource handle [, string document]) \linebreak

The function creates a new document in the printer spooler. A document can contain multiple pages, it's used to schedule the print job in the spooler. *`handle`* must be a valid handle to a printer. The optional parameter *`document`* can be used to set an alternative document name.

Példa 1. printer_start_doc() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_start_page (unknown)

Start a new page

bool **printer_start_page** (resource handle) \linebreak

The function creates a new page in the active document. For an example see printer_start_doc(). *`handle`* must be a valid handle to a printer.

printer_write (unknown)

Write data to the printer

bool **printer_write** (resource handle, string content) \linebreak

Writes *`content`* directly to the printer, and returns `TRUE` on success or `FALSE` if it failed.

`handle` must be a valid handle to a printer.

Példa 1. printer_write() example

```
$handle = printer_open();
printer_write($handle, "Text to print");
printer_close($handle);
```

LXXXIV. Pspell Functions

These functions allow you to check the spelling of a word and offer suggestions.

You need the aspell and pspell libraries, available from http://aspell.sourceforge.net/ and http://aspell.net/ respectively, and add the `--with-pspell[=dir]` option when compiling php.

pspell_add_to_personal (PHP 4 >= 4.0.2)

Add the word to a personal wordlist

int **pspell_add_to_personal** (int dictionary_link, string word) \linebreak

pspell_add_to_personal() adds a word to the personal wordlist. If you used pspell_new_config() with pspell_config_personal() to open the dictionary, you can save the wordlist later with pspell_save_wordlist(). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Példa 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_add_to_session (PHP 4 >= 4.0.2)

Add the word to the wordlist in the current session

int **pspell_add_to_session** (int dictionary_link, string word) \linebreak

pspell_add_to_session() adds a word to the wordlist associated with the current session. It is very similar to pspell_add_to_personal()

pspell_check (PHP 4 >= 4.0.2)

Check a word

bool **pspell_check** (int dictionary_link, string word) \linebreak

pspell_check() checks the spelling of a word and returns `TRUE` if the spelling is correct, `FALSE` if not.

Példa 1. pspell_check()

```
$pspell_link = pspell_new ("en");

if (pspell_check ($pspell_link, "testt")) {
    echo "This is a valid spelling";
```

```
} else {
    echo "Sorry, wrong spelling";
}
```

pspell_clear_session (PHP 4 >= 4.0.2)

Clear the current session

int **pspell_clear_session** (int dictionary_link) \linebreak

pspell_clear_session() clears the current session. The current wordlist becomes blank, and, for example, if you try to save it with pspell_save_wordlist(), nothing happens.

Példa 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_clear_session ($pspell_link);
pspell_save_wordlist ($pspell_link);    //"Vlad" will not be saved
```

pspell_config_create (PHP 4 >= 4.0.2)

Create a config used to open a dictionary

int **pspell_config_create** (string language [, string spelling [, string jargon [, string encoding]]]) \linebreak

pspell_config_create() has a very similar syntax to pspell_new(). In fact, using **pspell_config_create()** immediatelly followed by pspell_new_config() will produce the exact same result. However, after creating a new config, you can also use **pspell_config_*()** functions before calling pspell_new_config() to take advantage of some advanced functionality.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are ’utf-8’, ’iso8859-*’, ’koi8-r’, ’viscii’, ’cp1252’, ’machine unsigned 16’, ’machine unsigned 32’. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Példa 1. pspell_config_create()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_personal ($pspell_config, "en");
```

pspell_config_ignore (PHP 4 >= 4.0.2)

Ignore words less than N characters long

int **pspell_config_ignore** (int dictionary_link, int n) \linebreak

pspell_config_ignore() should be used on a config before calling pspell_new_config(). This function allows short words to be skipped by the spellchecker. Words less then n characters will be skipped.

Példa 1. pspell_config_ignore()

```
$pspell_config = pspell_config_create ("en");
pspell_config_ignore($pspell_config, 5);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "abcd");    //will not result in an error
```

pspell_config_mode (PHP 4 >= 4.0.2)

Change the mode number of suggestions returned

int **pspell_config_mode** (int dictionary_link, int mode) \linebreak

pspell_config_mode() should be used on a config before calling pspell_new_config(). This function determines how many suggestions will be returned by pspell_suggest().

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)

Példa 1. pspell_config_mode()

```
$pspell_config = pspell_config_create ("en");
pspell_config_mode($pspell_config, PSPELL_FAST);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
```

pspell_config_personal (PHP 4 >= 4.0.2)

Set a file that contains personal wordlist

int **pspell_config_personal** (int dictionary_link, string file) \linebreak

pspell_config_personal() should be used on a config before calling pspell_new_config(). The personal wordlist will be loaded and used in addition to the standard one after you call pspell_new_config(). If the file does not exist, it will be created. The file is also the file where pspell_save_wordlist() will save personal wordlist to. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Példa 1. pspell_config_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_repl (PHP 4 >= 4.0.2)

Set a file that contains replacement pairs

int **pspell_config_repl** (int dictionary_link, string file) \linebreak

pspell_config_repl() should be used on a config before calling pspell_new_config(). The replacement pairs improve the quality of the spellchecker. When a word is misspelled, and a proper suggestion was not found in the list, pspell_store_replacement() can be used to store a replacement pair and then pspell_save_wordlist() to save the wordlist along with the replacement pairs. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Példa 1. pspell_config_repl()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_runtogether (PHP 4 >= 4.0.2)

Consider run-together words as valid compounds

int **pspell_config_runtogether** (int dictionary_link, bool flag) \linebreak

pspell_config_runtogether() should be used on a config before calling pspell_new_config(). This function determines whether run-together words will be treated as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Példa 1. pspell_config_runtogether()

```
$pspell_config = pspell_config_create ("en");
pspell_config_runtogether ($pspell_config, true);
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_save_repl (PHP 4 >= 4.0.2)

Determine whether to save a replacement pairs list along with the wordlist

int **pspell_config_save_repl** (int dictionary_link, bool flag) \linebreak

pspell_config_save_repl() should be used on a config before calling pspell_new_config(). It determines whether pspell_save_wordlist() will save the replacement pairs along with the wordlist. Usually there is no need to use this function because if pspell_config_repl() is used, the replacement pairs will be saved by pspell_save_wordlist() anyway, and if it is not, the replacement pairs will not be saved. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

pspell_new_config (PHP 4 >= 4.0.2)

Load a new dictionary with settings based on a given config

int **pspell_new_config** (int config) \linebreak

pspell_new_config() opens up a new dictionary with settings specified in a config, created with with pspell_config_create() and modified with **pspell_config_*()** functions. This method provides you with the most flexibility and has all the functionality provided by pspell_new() and pspell_new_personal().

The config parameter is the one returned by pspell_config_create() when the config was created.

Példa 1. pspell_new_config()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
```

pspell_new_personal (PHP 4 >= 4.0.2)

Load a new dictionary with personal wordlist

int **pspell_new_personal** (string personal, string language [, string spelling [, string jargon [, string encoding [, int mode]]]]) \linebreak

pspell_new_personal() opens up a new dictionary with a personal wordlist and returns the dictionary link identifier for use in other pspell functions. The wordlist can be modified and saved with pspell_save_wordlist(), if desired. However, the replacement pairs are not saved. In order to save replacement pairs, you should create a config using pspell_config_create(), set the personal wordlist file with pspell_config_personal(), set the file for replacement pairs with pspell_config_repl(), and open a new dictionary with pspell_new_config().

The personal parameter specifies the file where words added to the personal list will be stored. It should be an absolute filename beginning with ’/’ because otherwise it will be relative to $HOME, which is "/root" for most systems, and is probably not what you want.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are ’american’, ’british’, and ’canadian’.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are ’utf-8’, ’iso8859-*’, ’koi8-r’, ’viscii’, ’cp1252’, ’machine unsigned 16’, ’machine unsigned 32’. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)
- `PSPELL_RUN_TOGETHER` - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, `PSPELL_FAST`, `PSPELL_NORMAL` and `PSPELL_BAD_SPELLERS` are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Példa 1. pspell_new_personal()

```
$pspell_link = pspell_new_personal ("/var/dictionaries/custom.pws",
        "en", "", "", "", PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_new (PHP 4 >= 4.0.2)

Load a new dictionary

int **pspell_new** (string language [, string spelling [, string jargon [, string encoding [, int mode]]]]) \linebreak

pspell_new() opens up a new dictionary and returns the dictionary link identifier for use in other pspell functions.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)
- `PSPELL_RUN_TOGETHER` - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, `PSPELL_FAST`, `PSPELL_NORMAL` and `PSPELL_BAD_SPELLERS` are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Példa 1. pspell_new()

```
$pspell_link = pspell_new ("en", "", "", "",
                           (PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_save_wordlist (PHP 4 >= 4.0.2)

Save the personal wordlist to a file

int **pspell_save_wordlist** (int dictionary_link) \linebreak

pspell_save_wordlist() saves the personal wordlist from the current session. The dictionary has to be open with pspell_new_personal(), and the location of files to be saved specified with pspell_config_personal() and (optionally) pspell_config_repl(). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Példa 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/tmp/dicts/newdict");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_store_replacement (PHP 4 >= 4.0.2)

Store a replacement pair for a word

int **pspell_store_replacement** (int dictionary_link, string misspelled, string correct) \linebreak

pspell_store_replacement() stores a replacement pair for a word, so that replacement can be returned by pspell_suggest() later. In order to be able to take advantage of this function, you have to use pspell_new_personal() to open the dictionary. In order to permanently save the replacement pair, you have to use pspell_config_personal() and pspell_config_repl() to set the path where to save your custom wordlists, and then use pspell_save_wordlist() for the changes to be written to disk. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Példa 1. pspell_store_replacement()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);

pspell_store_replacement ($pspell_link, $misspelled, $correct);
pspell_save_wordlist ($pspell_link);
```

pspell_suggest (PHP 4 >= 4.0.2)

Suggest spellings of a word

array **pspell_suggest** (int dictionary_link, string word) \linebreak

pspell_suggest() returns an array of possible spellings for the given word.

Példa 1. pspell_suggest()

```
$pspell_link = pspell_new ("en");

if (!pspell_check ($pspell_link, "testt")) {
    $suggestions = pspell_suggest ($pspell_link, "testt");

    foreach ($suggestions as $suggestion) {
        echo "Possible spelling: $suggestion<br>";
    }
}
```

LXXXV. GNU Readline

A readline() függvények a GNU Readline könyvtár használatát teszik lehetővé PHP-ből. Ezek a függvények egy szerkeszthető parancssort nyújtanak. Például ilyen a Bash, ami a kurzormozgató gombok használatával lehetőséget ad karakterek beszúrására, és a korábban kiadott parancsok visszakérésére. Ezen könyvtár interaktív természetéből fakadóan kis hasznnal bír Web alkalmazások tervezésekor, de hasznos lehet parancssorból futó programok írásakor.

A GNU Readline projekt honlapja a http://cnswww.cns.cwru.edu/~chet/readline/rltop.html címen található. A honlapot Chet Ramey tartja fent, aki a Bash szerzője.

readline_add_history (PHP 4)

Egy sort ad a ’history’-hoz

void **readline_add_history** (string line) \linebreak

Ez a függvény egy sort ad a parancssor ’history’-hoz.

readline_clear_history (PHP 4)

Törli a ’history’-t

boolean **readline_clear_history** (void) \linebreak

Ez a függvény törli a teljes ’history’-t.

readline_completion_function (PHP 4)

Regisztrál egy kiegészítést végző függvényt

boolean **readline_completion_function** (string line) \linebreak

Ez a függvény regisztrál egy kiegészítést végző függvényt. Egy létező függvény nevét kell megadnod, ami egy részparancsot vár és a lehetséges kiegészítések tömbjét adja vissza. Ez az a funkció, amit a tab billenytű leütésével kapsz ha Bash-t használsz.

readline_info (PHP 4)

Lekér/beállít egy belső readline változót

mixed **readline_info** ([string varname [, string newvalue]]) \linebreak

Ha paraméter nélkül hívod, visszaadja az összes readline beállítás tömbjét. Az elemek a következő indexeket kapják: done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name, és terminal_name.

Ha egy paramétert adsz át, a megadott beállítás értékét adja vissza. Ha két paramétert adsz át, a beállítás a megadott értékre változik.

readline_list_history (PHP 4)

A ’history’ listázása

array **readline_list_history** (void) \linebreak

Ez a függvény a teljes ’history’ tömbjével tér vissza. Az elemek `NULL`ától kezdve pozitív egész számok.

readline_read_history (PHP 4)

Beolvassa a ’history’-t

boolean **readline_read_history** (string filename) \linebreak

Beolvassa a ’history’-t egy fájlból.

readline_write_history (PHP 4)

Kimenti a ’history’-t

boolean **readline_write_history** (string filename) \linebreak

Kimenti a ’history’-t egy fájlba.

readline (PHP 4)

Beolvas egy sort

string **readline** ([string prompt]) \linebreak

Ez a függvény egy a felhasználótól érkező stringet ad vissza. Megadhatsz egy stringet, ami prompt-ként megjelenik a felhasználónak. A visszaadott sor végéről a soremelést a függvény törli. Ezt magadnak kell behelyezned a ’history’ listába a readline_add_history() függvénnyel.

Példa 1. readline() példa

```
// Három parancs bekérése a felhasználótól
for ($i=0; $i < 3; $i++) {
        $parancs = readline ("Parancs: ");
        readline_add_history ($parancs);
}

// A ’history’ kilistázása
print_r (readline_list_history());

// Változók listázása
print_r (readline_info());
```

LXXXVI. GNU Recode függvények

Ez a modul egy felületet biztosít a GNU Recode könyvtár 3.5-ös verziójához. Ahhoz, hogy használhasd az itt bemutatott függvényeket, a PHP-t a --with-recode paraméterrel kell fordítanod. Ahhoz, hogy ezt megtehesd a rendszereden rendelkezésre kell, hogy álljon a GNU Recode 3.5 vagy újabb verzió.

A GNU Recode könyvtár fájlokat alakít át különböző fájl és felületi kódolások figyelembevételével. Amikor ez nem végezhető el tökéletesen, kitüörölheti a problémás karaktereket, vagy közelítéseket alkalmazhat. A könyvtár felismer illetve képes előállítani 150 különböző karakterkódolást, és képes majdnem minden két kódolás között átalakítást végezni. A legtöbb Most RFC 1345 karakterkészlet támogatott.

recode_file (PHP 3>= 3.0.13, PHP 4)

Fájlból fájlba kódol egy recode kérésnek megfelelően

boolean **recode_file** (string request, resource input, resource output) \linebreak

Recode-olja az *input* fájl azonosítóval megadott fájlt az *output* fájl azonosító által megadott fájlba a *request* recode kérésnek megfelelően. Ha sikertelen a művelet, FALSE-al tér vissza, egyébként TRUE-val.

Egy a függvény még nem kezeli azokat a fájl hivatkozásokat, amik távoli fájlokra vonatkoznak (URLekre). Mindkét fájl azonosító helyi fájlra kell, hogy mutasson.

Példa 1. Alapvető recode_file() példa

```
$bemenet = fopen (’bemenet.txt’, ’r’);
$kimenet = fopen (’kimenet.txt’, ’w’);
recode_file ("us..flat", $bemenet, $kimenet);
```

recode_string (PHP 3>= 3.0.13, PHP 4)

Recode-ol egy stringet egy recode kéréssnek meggelelően

string **recode_string** (string request, string string) \linebreak

A *string* stringet recode-olja a *request* kérésnek megfelelően. A recode-olt string-el tér vissza vagy FALSE értékkel, ha képtelen végrehajtani a kérést.

Egy egyszerű recode kérés lehet a "lat1..iso646-de" string. Lásd még a GNU Recode dokumentációt további részletekért a recode kéréseket illetően.

Példa 1. Alapvető recode_string() példa:

```
print recode_string ("us..flat", "A kovetkezo betu nem az angol abc resze: &aacute;");
```

recode (PHP 4)

Recode-ol egy stringet egy recode kéréssnek meggelelően

string **recode_string** (string request, string string) \linebreak

Megjegyzés: Ez a függvény egy alias a recode_string() függvényre. A PHP 4-es verzióban került a nyelvbe.

LXXXVII. Reguláris kifejezések függvényei (Perl kompatibilis)

Ezekben a függvényekben használatos minták szintaktikája nagyon hasonlít a PERL-ben megismerthez. A kifejezéseket határolójelek közé kell rakni, például perjelek / közé. Az alfanumerikus karakterektől és a visszaperjeltől (\) eltekintve bármi használható határolójelként. Ha a határolójel magában a kifejezésben is szerepel, akkor egy visszaperjelet (\) kell eléírni. A PHP 4.0.4-s verziójától kezdve lehetőség van a (), {}, [], és <> párban álló határolók használatára, ahogy Perl-ben is.

A lezáró határolójelet különféle módosítók követhetik, amelyek a mintailleszkedést befolyásolják. Bővebben: Minta módosítók fejezet.

PHP a kiterjesztett POSIX reguláris kifejezéseket is támogatja.

Előfeltételek

A reguláris kifejezések támogatását a PCRE könyvtár biztosítja. Ez nyílt forráskódú szoftver, amit Philip Hazel írt, és a szerzői jogok az angliai Cambridge Egyetem tulajdona. A csomag hozzáférhető a ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/ címen.

Telepítés

PHP 4.2.0 -től kezdve ezek függvények alapértelmezésben telepítve lesznek. Régebbi verziók esetén a PHP-t a `--with-pcre-regex[=DIR]` opcióval kellett fordítani. A függvényeket le lehet tiltani fordítás idején a `--without-pcre-regex` opció megadásával.

Futásidejű beállítások

Ez a kiterjesztés semmilyen konfigurációs beállításokat nem definiál.

Erőforrás típusok

Ez a kiterjesztés semmilyen erőforrás típust nem definiál.

Előredefiniált állandók

`PREG_PATTERN_ORDER` `PREG_SET_ORDER` `PREG_SPLIT_NO_EMPTY`

`PREG_SPLIT_DELIM_CAPTURE`

Példák

Példa 1. Példák érvényes mintákra

- /<\/\w+>/
- |(\d{3})-\d+|Sm
- /^(?i)php[34]/
- {^\s+(\s+)?$}

Példa 2. Példák helytelen mintákra

- /href=’(.*)’ - hiányzó lezáró határolókarakter
- /\w+\s*\w+/J - ismeretlen módosító ’J’
- 1-\d3-\d3-\d4| - hiányzó kezdő határolókarakter

Minta módosítók (unknown)

A lehetséges, PCRE reguláris kifejezések kiértékelését befolyásoló módosítók felsorolása

Itt látható a pillanatnyilag használható PCRE módosítók listája. A zárójelekben közölt nevek a módosítók PCRE-ben használt belső nevei.

i (PCRE_CASELESS)

Ha ez a módosító be van kapcsolva, akkor a minta a nagy- és kisbetűk különbözőségére érzéketlen, a mintában előforduló betűk mind a kis- mind a nagybetűkre illeszkednek.

m (PCRE_MULTILINE)

Alapértelmezés szerint PCRE úgy tekinti a tárgyszöveget, mintha az egyetlen sorból állna (mégha történetesen tartalmaz is néhány újsor karaktert). A "sor eleje" metakarakter (^) csakis a tárgyszöveg elejére, míg a "sor vége" metakarakter ($) csakis a szöveg végére vagy a lezáró újsor-karakterre engedi illeszteni a mintát - kivéve ha *D* módosító be van kapcsolva. Ez ugyanúgy működik, mint Perl-ben.

Amikor ez a módosító be van kapcsolva, akkor "sor eleje" illetve "sor vége" szerkezetek közvetlenül a következő illetve közvetlenül a megelőző újsor karakterekre, és a szöveg legelejére illetve legvégére illeszkednek. Ez egyenértékű Perl /m módosítójával. Ha nincs a tárgyszövegben "\n" karakter vagy a mintában nincs ^ illetve $, akkor ennek a módosítónak nincs hatása.

s (PCRE_DOTALL)

Ha ez a módosító be van kapcsolva, a mintában előforduló . metakarakter minden karakterre - beleértve az újsor karaktert is - illeszkedik, míg enélkül az újsor karakterre nem illeszkedne. Ez egyenértékű a Perl /s módosítójával. A tagadó karakterosztályok (mint például [^a]) mindig illeszkednek az újsor karakterre függetlenül ennek a módósítónak a beállításától.

x (PCRE_EXTENDED)

Ha ez a módosító be van kapcsolva, akkor a mintában szereplő térközök - kivéve visszaperjelet elétéve vagy a karakterosztályokon belülieket - nem lesznek részei a mintának, valamint minden karakter, amely karakterosztályon kívüli literális # és a rákövetkező újsorkarakter közé esik, figyelmen kívül hagy a PCRE. Ez egyenértékű a Perl /x módosítójával, ami lehetővé teszi megjegyzések elhelyezését a bonyolultabb mintákban. Megjegyzendő, hogy ez csak bizonyos poziciókra vonatkozik, térközkarakterek soha nem szerepelhetnek speciális karaktersorozatokban, mint például (?(-ban, ami egy feltételes részminta kezdetét jelöli.

e

Ha ez a módosító be van kapcsolva, akkor **preg_replace ()** függvény a helyettesítési paraméterként kapott sztringet - a hivatkozások feloldása után - PHP kódként értelmezi, és ennek a kódnak az eredményét helyettesíti be a keresett szöveg helyére.

Csak a preg_replace() használja ezt a módosítót, a többi PCRE függvény figyelmen kívül hagyja.

A (PCRE_ANCHORED)

Ha ez a módosító be van kapcsolva, akkor a keresett mintát "lerögzíti", ami annyit jelent, hogy a minta illesztése korlátozva van a tárgyszöveg elejére. Ezt a hatást elérhetjük ha magát a mintát megfelelően alakítjuk ki, miként Perl-ben ennek ez az egyetlen módja.

D (PCRE_DOLLAR_ENDONLY)

Ha ez a módosító be van kapcsolva, akkor dollár metakarakter ($) a mintán belül csak a tárgyszöveg végére illeszkedik. Enélkül a módosító nélkül a $ a legutolsó újsorkarakter előtti pozicióra is illeszkedik, de semmilyen más poziciójú újsorkarakter előtt nem. Ezt a módosítót figyelmen kívül hagyja a PCRE, ha a *m* be van kapcsolva. Nincs megfelelője Perl-ben.

S

Ha egy mintát többször használunk, akkor érdemes több időt hagyni a kiértékelésére, hogy az illesztésre fordított idő lerövidülhessen. Ennek a módosítónak a bekapcsolásával ezt biztosítjuk. Jelenleg, csak olyan nem "rögzített" minták esetén hasznos, amelyek nem egy egyszerű, rögzített karakterrel kezdődnek.

U (PCRE_UNGREEDY)

Ez a módosító megfordítja a kvantorok (sokszorozók) mohóságát, azaz alapértelmezés szerint nem lesznek mohók, csak ha egy ? követi azokat. Ez nem Perl-kompatibilis. A mintában is beállítható (?U) sorozat megadásával.

X (PCRE_EXTRA)

Ez a módosító PCRE további lehetőségeit teszi elérhetővé, amelyek nem Perl-kompatibilisak. Valamennyi visszaperjel a mintában, amit olyan karakter követ, aminek nincs speciális jelentése, hibát okoz, és ilyenformán lefoglalja ezeket a karakterkombinációkat a jövőbeni kiterjesztések előtt. Alapértelmezés szerint az ilyen különleges jelentés nélküli karakter előtt álló visszaperjelet a Perl betű szerint veszi. Jelenleg nincs más jellegzetesség, amit ez a módosító befolyásolna.

u (PCRE_UTF8)

Ez a módosító is PCRE olyan további lehetőségét teszi elérhetővé, ami nem Perl-kompatibilis. A mintát UTF-8 kódolású szövegnek tekinti. Ez a módosító PHP 4.1.0-től kezdve érhető el.

Regurális kifejezések szintaxisa (unknown)

PCRE regurális kifejezések nyelvtanának leírása

A PCRE könyvtár függvények sorát tartalmazza, amelyek reguláris kifejezések mintaillesztését végzik el ugyanazt a szintaxist és szemantikát használva - néhány eltérést nem számítva -, mint a Perl 5. (lásd alább). A jelenlegi megvalósítás Perl 5.005-höz hasonló.

Különbségek Perl 5.005-höz képest:

1. 1. Alapértelmezés szerint térköznek tekintendő minden olyan karakter, amit a C könyvtár isspace() függvénye felismer, bár lehetőség van arra, hogy PCRE-t eltérő karakterkódtáblázattal fordítsuk le. Az isspace() rendszerint a szóköz, lapdobás, "kocsi-vissza", újsor és vízszintes illetve függőleges tabulátorokat tekinti térköznek. Perl 5-ben ez utóbbi már nem szerepel a listában. A \v szekvencia nagyon sokáig benne volt a Perl dokumentációjában, valójában soha nem ismerte fel mintaillesztés közben. Legalább az 5.002 verzióig ezt a karaktert is térköznek tekintették, de a 5.004 és 5.005 verziókban már nem illeszkedik a \s-re.
2. 2. PCRE nem engedi meg az előretekintő tesztekben (lookahead assertion) a kvantorok (sokszorozók) használatát. Perl-ben ez megengedett, de nem a várt eredményt adja. Például (?!a){3}, nem azt jelenti, hogy teszteli, vajon a következő három karakter nem "a", hanem azt, hogy háromszor "jelenti", ha a következő karakter nem "a".

3. Tagadó előretekintő tesztekben (negated lookahead assertion) szereplő "begyűjtő" részminták (capturing subpatterns) számon vannak tartva, de a hozzájuk tartozó numerikus változók soha nem kerülnek feltöltésre. Perl feltölti ezeket a numerikus változókat valamilyen, a sikertelen teszt előtti mintára illeszkedő részt követő karakterekkel, de csak akkor, ha a tagadó előretekintő tesztnek csak egy ága volt.
4. A bináris nullkarakter használata megengedett a tárgyszövegben, de a mintában nem, mert a mintául szolgáló sztring nullvégű C sztringként adódik át. A "\0" vezérlőszekvencia használható a mintában bináris nullkarakter megjelenítésére.
5. Nem támogatja az alábbi Perl vezérlőszekvenciákat: \l, \u, \L, \U, \E, \Q. Valójában ezek a funkciók a Perl sztringkezelő részében vannak megvalósítva, és nem tartoznak annak mintaillesztő motorjához.
6. A Perl \G tesztjét nem támogatja, mivel nem fontos az egyszeres minta illesztésekben.
7. Magától érthetődően PCRE nem támogatja a (?{kód}) szerkezetet.
8. A kézikönyv írása során néhány furcsaság még van a Perl 5.002_2-ben a begyűjtött sztringek feltöltésével kapcsolatban, ha a mintarészek ismétlődhetnek. Például, "aba"-ra illesztve a /^(a(b)?)+$/ mintát $2 numerikus változó értéke "b" lesz, míg a "aabbaa"-ra illesztve /^(aa(bb)?)+$/ -t $2 üres marad. Mindamellett /^(aa(b(b))?)+$/ -ra módosítva a mintát $2 és $3 is beállításra kerül. Perl 5.004-ben $2 mindkét esetben beállításra kerül, mint ahogy ez a PCRE-re is igaz. Ha a jövőben a Perl-ben ezt egységesíteni fogják, akkor PCRE követni fogja ezt.
9. Másik mindeddig megoldatlan eltérés, hogy pl. /^(a)?(?(1)a|b)+$/ minta Perl 5.002_2-ben illeszkedik a "a"-ra, míg PCRE-ben nem. Ráadásul mind Perl-ben mind PCRE-ben /^(a)?a/ minta "a"-ra illesztése során a $1 változó üresen marad.
10. PCRE néhény bővítést is nyújt a Perl reguláris kifejezéseinek lehetőségeihez:
    a. Jóllehet a hátratekintő tesztek csak rögzített hosszúságú sztringekre illeszkedhetnek, az alternatív ágak mindegyikében különböző hosszúságú sztringre illeszkedő minta adható meg. Perl 5.005-ben minden ágban ugyanakkorának kell lennie ennek a hossznak.
    b. Ha a PCRE_DOLLAR_ENDONLY be van kapcsolva, és a PCRE_MULTILINE nincs bekapcsolva, akkor $ metakarakter csakis a sztring legvégére illeszkedik.
    c. Ha PCRE_EXTRA be van kapcsolva, akkor bármilyen külön jelentéssel nem bíró karakter előtti visszaperjel hibának minősül.
    d. Ha PCRE_UNGREEDY be van kapcsolva, akkor a kvantorok (sokszorozók) mohóságát fordítva értelmezi, azaz, alapértelmezés szerint nem mohók, csak ha egy kérdőjel ? áll utánuk.

Bevezetés

Az alábbiakban a PCRE által megvalósított reguláris kifejezések szintaktikáját és szemantikáját adjuk meg. A reguláris kifejezések leírása a Perl dokumentációnak része is, és néhány könyv is foglalkozik vele, némelyik rengeteg példával. Jeffrey Friedl "Mastering Regular Expression" c. könyve, amely az O’Reilly gondozásában jelent meg (ISBN 1-56592-257-3) igen nagy mélységben tárgyalja a reguláris kifejezéseket. Az itteni leírást referenciának szánjuk.

A reguláris kifejezés maga egy minta, amit a tárgyszövegre kell illeszteni balról jobbra. A karakterek többsége maga helyett áll a mintában, azaz csak saját magára illeszkedik a tárgyszövegben. Egy

egyszerű példán bemutatva: az `a gyors vörös róka` minta a tárgyszöveg azon részére illeszkedik, ami megegyezik vele (azaz ahol az `a gyors vörös róka` előfordul).

Metakarakterek

A reguláris kifejezések ereje abban rejlik, hogy lehetőségünk van vagylagosságot (alternatívákat) és ismétléseket megadni a mintában. Ezeket a *metakarakterek* használatával lehet megadni, amelyek nem saját maguk helyett állnak a mintában, hanem speciális jelentéssel bírnak.

Két különböző típusú metakarakter létezik: az egyik bárhol előfordulhat a mintában, kivéve szögeletes zárójelek [] között, és a másik, ami csak szögeletes zárójelek között azonosítható metakarakterként. A szögletes zárójeleken kívüli metakarakterek:

\

: több karakterből álló vezérlőszekvencia kezdetét jelzi

^

: tárgyszöveg elejét (vagy sor elejét többsoros módban) jelenti

$

: tárgyszöveg végét (vagy sor végét többsoros módban) jelenti

.

: bármilyen karakterre illeszkedik, kivéve az újsor karaktert (alapértelmezés szerint)

[

: karakterosztály definició kezdetét jelzi

]

: karakterosztály definició végét jelzi

|

: elágazás kezdetét jelzi

(

: részminta kezdetét jelzi

)

: részminta végét jelzi

?

: opcionálitást jelez (0 vagy 1 előfordulást engedélyez), vagy kibővíti az előtte álló (jelentését, és a kvantorok (sokszorozók) mohóságát befolyásolja

*

: "0 vagy több" kvantor (sokszorozó)

+

: "1 vagy több" kvantor (sokszorozó)

{

"min/max" kvantor (sokszorozó) kezdetét jelzi

}

"min/max" kvantor (sokszorozó) végét jelzi

A minta szögletes zárójelek [] közötti részét karakterosztálynak hívjuk. A karakterosztályokban engedélyezett metakarakterek a következők:

\

több karakterből álló vezérlőszekvencia kezdetét jelzi

^

negálja az osztályt, kizáró / tagadó osztályt definiál, de csak ha ez az első karakter az osztályon belül

-

karaktertartományt jelez

]

lezárja a karakterosztályt

A következő fejezetek a metakarakterek használatát írják le.

visszaperjel \

A visszaperjel \ karakternek számos alkalmazása van. Először is, ha nem alfanumerikus karakter követi, akkor annak bármilyen különleges jelentését (metakarakterek) figyelmen kívül hagyja, és azt mint maga helyett álló karaktert értelmezi. Ebben a minőségében mind karakterosztályon belül mind kívül használható.

Például, ha "*" karaktert kell illeszteni, akkor "*"-t kell írni a mintában. Akármikor használható ez a szerkezet, akár a következő karakter egyébként metakarakterként lenne értelmezve, akár nem. (Egy kivétel van lásd a] karakternél - a fordító.) Ezért biztos módszer előírni azt, hogy minden nem alfanumerikus karakter maga helyett álljon, úgy, hogy visszaperjelet (\) írunk elé. Különben ha a visszaperjelre szeretnénk illeszteni, akkor "\\"-t kell írni.

Ha a minta PCRE_EXTENDED módosítóval van megadva, akkor minden olyan térközkaraktert figyelmen kívül hagy, amely karakterosztályon kívül vagy karakterosztályon kívüli "#" és azt követő újsor-karakter között szerepel. Visszaperjellel bevezetett térköz- vagy "#" karakterek a minta részét képezik.

A visszaperjel másik alkalmazása a nem nyomtatható karakterek kódolását teszi láthatóvá a mintában. Nincs megszorítás a nem nyomtatható karakterek megjelenésére, kivéve a mintát lezáró nullkaraktert. Ha a minta szövegszerkesztés folyományaként áll elő, akkor általában könnyebb valamelyik alábbi vezérlőszekvenciát használni, mint az általuk reprezentált karakter bináris alakját:

\a

csengő karakter (hexadecimálisan 07)

\cx

"control-x", ahol x bármilyen karakter lehet

\e

"escape" (hexadecimálisan 1B)

\f

lapdobás (hexadecimálisan 0C)

\n

újsor (hexadecimálisan 0A)

\r

kocsivissza (hexadecimálisan 0D)

\t

tabulátor (hexadecimálisan 09)

\xhh

hh hexadecimális kódú karakter

\ddd

ddd oktális kódú karakter, vagy "hivatkozás"

A "`\cx`" feloldása precízen megfogalmazva a következő: ha "`x`" kisbetű, akkor nagybetűvé konvertálódik. Ezután a karakter 6. bitjét (hexadecimálisan 40) invertálja. Ilyenformán "`\cz`"-ból hexadecimális 1A lesz, de "`\c{`"-ból hexadecimális 3B lesz, míg "`\c;`"-ból hexadecimális 7B.

A "`\x`" után két hexadecimális számjegy kerül beolvasásra (a betűk lehetnek kis és nagybetűk is).

A "`\0`" után két további oktális számjegy kerül beolvasásra. Minden esetben - még ha kevesebb, mint két számjegy került is megadásra - a meglevő számjegyek lesznek felhasználva. Ekképpen "`\0\x\07`" egy olyan sorozatot definiál, ahol két bináris nullkaraktert követ egy csengőkarakter. Bizonyosodj meg affelől, hogy mindig van két számjegy a kezdő 0 után, ha a következő karakter oktális számjegy!

A nem 0-val folytatódó visszaperjel kezelése bonyolult. Karakterosztályon kívül PCRE ezt és a soron következő számjegyeket tízes számrendszerbeli számként értelmezi. Ha szám 10-nél kisebb vagy legalább ezzel a számmal egyező "gyűjtő" nyitó zárójel előzi meg a kifejezésben, akkor az egész sorozatot *hivatkozás*ként értelmezi. Később a zárójelezett részminták fejezetében részletesen tárgyaljuk, hogyan használható is ez.

Karakterosztályon belül vagy ha a tízes számrendszerbeli szám 9-nél nagyobb, és nincs ennyi számú "gyűjtő" részminta, akkor PCRE újraolvassa a visszaperjelet követő három oktális karaktert, és egy bájtra váltja a 8 legkisebb helyiértékű bitet. Bármely rákövetkező számjegy maga helyett áll, magára illeszkedik.

\040

egy másik módja a szóköz megadásának

\40

ugyanaz mint előbb, ha biztosított, hogy 40-nél kevesebb "gyűjtő" részminta van előtte

\7

mindig hivatkozásként értelmezi

\11

hivatkozás is lehet, vagy a tabulátort írhatjuk így

\011

mindig tabulátort jelent

\0113

tabulátor és utána egy "3"

\113

113-as oktális kódú karakter ("K"), mivel hivatkozások száma nem lehet több, mint 99

\377

egy olyan bájt, ami teljes egészében 1 értékű biteket tartalmaz

\81

akár hivatkozás is lehet, vagy egy bináris nullkarakter és utána két karakter egy "8" és egy "1"

Figyelj arra, hogy 100-nál nagyobb oktális értékek elé nem szabad bevezető 0-t írni, mert mindig csak három oktális kerül beolvasásra.

Minden sorozat, ami egy egyszerű bájtot határoz meg, karakterosztályon belül és kívül is használható. Ráadásul karakterosztályokon belül a "\b" szekvencia "visszatörlés" karakterként (backspace) (hexadecimálisan 08) jelent. Ennek karakterosztályokon kívül más jelentése van (lásd alább).

A visszaperjel harmadik alkalmazási módja általános karakterfajták megadása:

\d

bármilyen tízes számrendszerbeli számjegy (decimális)

\D

bármilyen karakter, ami nem tízes számrendszerbeli számjegy

\s

bármilyen térközkarakter

\S

bármilyen nem térközkarakter

\w

bármilyen "szóépítő" karakter

\W

bármilyen nem "szóépítő" karakter

Mindegyik szekvenciapár két diszjunkt (egymással nem átfedő) részre osztja a karakterek teljes halmazát. Minden karakter csakis a párok egyik felébe tartozik.

"Szóépítő" karakter minden betű vagy számjegy vagy aláhúzás, azaz minden olyan karakter, amit egy Perl-típusú "`szónak`" része lehet. A betűk és számjegyek értelmezése a PCRE karaktertáblázataitól függ, amelyek változhatnak ha nyelvi beállítástól függően. Például, "fr" (vagy "hu" - a fordító) nyelvi beálllítás esetén 128-nál magasabb kódú karakterek használatosak az ékezetes betűkhöz, amelyekre illeszkedik a `\w`.

Ezek a karakterfajták mind karekterosztályokon belül és kívül is alkalmazhatók. Mindegyik csak egy, a neki megfelelő karakterre illeszkedik. Ha az aktuális illeszkedési pont a tárgyszöveg vége, akkor mindegyik sikertelen lesz, mert egyáltalán nincs karakter, amire illeszteni lehetne.

A visszaperjelet végül de nem utolsósorban néhány egyszerű teszt megadására használhatjuk. A tesztek (assertion) olyan feltételek, amelyeknek az illesztés adott pontján kell teljesülnie anélkül, hogy a tárgyszöbegből karaktereket használnának el - és így karaktereket vennének el az illesztéstől. Később megadjuk, hogy hogyan használhatók a részminták összetettebb tesztek megadásához. A "visszaperjeles" tesztek:

\b

szóhatár

\B

nem szóhatár

\A

tárgyszöveg kezdete (függetlenül többsoros üzemmódtól)

\Z

tárgyszöveg vége vagy újsor karakter a végén (függetlenül többsoros üzemmódtól)

\z

tárgyszöveg vége (függetlenül többsoros üzemmódtól)

Ezek a tesztek karakterosztályokban nem használhatók. (Figyeld meg, hogy "`\b`"-nek eltérő jelentése van karakteroszályokon belül, nevezetesen itt a "visszatörlés" karakter helyett áll.)

A szóhatár a tárgyszöveg azon poziciója, ahol az aktuális és az azt megelőző karakterek fajtája `\w` illetve `\W` vagy fordítva (az egyik `\w`-re illeszkedik, míg a másik `\W`-re), vagy a tárgyszöveg eleje illetve vége, ha az első illetve utolsó karakter a `\w`-re illeszkedik.

A `\A`, `\Z` és `\z` tesztek különböznek a hagyományos kalap(os ékezet) "^" és dollárjel "$" szimbólumoktól, mert az előbbiek minden esetben csakis a tárgyszöveg legelejére illetve legvégére illeszkednek. Nem befolyásolja a viselkedésüket sem PCRE_NOTBOL sem PCRE_NOTEOL módosító. A `\Z` és `\z` közti különbség annyi, hogy a `\z` csak a tárgyszöveg végére illeszkedik, míg `\Z` az esetlegesen a tárgyszöveg végén lévő újsor karakter előtt is.

kalap(os ékezet) ^ és dollárjel $

Karakterosztályokon kívül az alapértelmezett illesztési üzemmódban a kalap(os ékezet) egy olyan tesztet jelent,amely csak akkor és csak is akkor igaz (TRUE), ha a pillanatnyi illesztési pont a tárgyszöveg kezdete. Karakterosztályokon belül teljesen más jelentése van (lásd alább).

A kalapnak akkor nem szükséges a minta legelején állnia, ha több választási lehetőség is fennáll (lásd: retesz pipa csővezeték |), de minden választható ágban a legelsőnek kell lennie, ha azt akarod, hogy a minta valaha is illeszkedjen. Ha mindegyik lehetséges ág "kalappal" kezdődik, azaz ha a minta mindenesetben csak a tárgyszöveg elejétől kezdve illeszkedhet, akkor "(le)rögzített" mintáról beszélünk. (Más módja is van, hogy egy mintát "rögzíts".)

A dollárjel olyan tesztet jelent, amely csak akkor és csakis akkor igaz (TRUE), ha a pillanatnyi illesztési pont a tárgyszöveg vége vagy egy újsor karakter előtti pozició - az alapértelmezés szerint. A dollárjelnek akkor nem szükséges a minta legvégén állnia, ha több választási lehetőség is fennáll (lásd: retesz pipa csővezeték |),de minden választható ágban - ahol szerepel - a legutolsónak kell lennie. A dollárjelnek nincs külön jelentése karakterosztályokon belül.

A dollárjel jelentése megváltozik és csak a tárgyszöveg végére illeszkedik, ha a PCRE_DOLLAR_ENDONLY módosító be van kapcsolva a minta fordításakor / kiértékelésekor vagy az illesztés pillanatában. Ez nem érinti a \Z tesztet.

A kalap(os ékezet) és a dollárjel jelentését megváltoztatja PCRE_MULTILINE módosító bekapcsolása. Ebben az esetben a közbenső "\n" újsorkarakterek közvetlen előtti illetve utáni pozicióira is illeszkednek a tárgyszöveg eleji és végi pozición felül. Például, így a /^abc$/ minta többsoros üzemmódban illeszkedik a "def\nabc" szövegre, de egyébként nem. Következésképpen az egysoros üzemmódban "rögzítettnek" tekintett minták, nem "rögzítettek" többsoros üzemmódban, mert ekkor "^"-kal kezdődő vagylagos ágak nem tekinthetők "rögzítettnek". A PCRE_DOLLAR_ENDONLY módosítót figyelmen kívül hagyja, ha a PCRE_MULTILINE, módosító be van kapcsolva.

Figyelj arra, hogy a \A, \Z és \z szekvenciák mindkét üzemmódban használhatók a tárgyszöveg elejére illetve végére illesztésben, és ha minden vagylagos ág \A -val kezdődik, akkor a minta mindig "rögzített" lesz akármi is a PCRE_MULTILINE módosító állapota.

pont .

Karakterosztályokon kívül a pont bármelyik karakterre illeszkedik beleértve minden nem nyomtatható karaktert is, kivéve az újsort - alapértelmezés szerint. Ha a PCRE_DOTALL be van kapcsolva, akkor az újsor karakterre is illeszkedik. A pont kezelése teljesen független a kalap(os ékezet) ^ és a dollárjel $ kezelésétől, az egyetlen közös hármukban, hogy mindegyiknek köze van az újsor karakterhez. A pontnak nincs speciális jelentése karakterosztályokon belül.

Szögeletes zárójel []

A nyitó szögeletes zárójel egy karakterosztály kezdetét, míg a záró párja annak végét jelzi. A bezáró zárójel önmagában nem hordoz speciális jelentést. Ha egy] karaktert kell a karakterosztályon belül megadni, akkor annak vagy legelöl kell állnia - az esetleges kalap(os ékezet) után -, vagy egy visszaperjelet kell eléírni.

A karakterosztályok csak a tárgyszöveg egyetlen karakterére illeszkednek. Ha a karakterosztály legelső karaktere nem a kalap(os ékezet) ^,akkor ez csak a karakterosztályba tartozó karakterek valamelyikére illeszkedik, ha a legelső karakter kalap(os ékezet) ^ , akkor pedig csak azokra, amelyek nem elemei az osztálynak. (Ez utóbbit hívjuk tagadó vagy kizáró karakterosztálynak - a fordító.) Ha a kalap(os ékezet) ^ karaktert kell megadni egy karakterosztály elemeként, akkor az nem lehet a legelső pozicióban, vagy visszaperjelet kell eléírni.

Az [aeiou] karakterosztály például minden kisbetűs ékezet nélküli magánhangzóra illeszkedik, míg az [^aeiou] pont ezekre nem. A kalap(os ékezet) kényelmes jelölési forma olyan karakterosztályok megadására, ahol a kizárandó elemeket könnyebb felsorolni. Ez nem teszt, ez karaktereket használ el a tárgyszövegből, ezért a tárgyszöveg végén az illesztés sikertelen lesz.

Amikor a kis- és nagybetűk különbözősége nem számít, akkor a betűk a kis- és nagybetűs alakjukat is képviselik, tehát ilyen esetben például [aeiou] illeszkedik az "A"-ra és az "a"-ra is, és az [^aeiou] nem illeszkedik a "A"-ra, csak ha a kis- és nagybetű különbözősége számít.

Karakterosztályokban az újsort soha nem kezelik különleges módon, bármi is a PCRE_DOTALL és a PCRE_MULTILINE módosító értéke. A kizáró karakterosztályok - mint pl.: [^a] - pedig mindig illeszkednek az újsor karakterre.

Az "-" elválasztójellel karaktertartományokat lehet megadni a karakterosztályon belül. A [d-m], például, a d és m közti bármelyik karakterre illeszkedik - d-t és m-et is beleértve. Ha az elválasztójelet kell a karakterosztályon belül megadni, akkor vagy egy visszaperjelet kell eléírni, vagy olyan pozicióban kell megadni, ahol nem lehet tartományképzőként értelmezni, jellemzően a karakterosztály első vagy utolsó tagjaként.

A "]" karaktert ebben a literális formában nem lehet egy tartomány záró tagjaként használni. A [W-]46] minta például azt jelenti, hogy a "W" és "-" karakterek valamelyikét kövesse a "46]" sztring, ezért a "W46]" vagy a "-46]" illeszkedik a mintára. Ha azonban visszaperjelet írunk elé, akkor a visszaperjel lesz a tartomány felső határa, és ekkor "[W-\]46]" mintát egy egyszerű osztályként értelmezi, amit három különálló karakter követ. Oktális vagy hexadecimális ábrázolásban lehet megadni a "]"-t mint tartományt lezáró karaktert.

A tartományok meghatározásakor az ASCII karaktertáblázat karakterkódjai számítanak. A karakterek helyett a numerikus kódjaik is használhatók, például [\000-\037]. Ha kis- és nagybetűkre nem érzékeny az illesztés és a tartomány betűket is felölel, akkor azok mind a kis- mind a nagybetűs alakjuknak megfelelő karaktereket egyaránt helyettesítik, például a [W-c] ugyanaz, mint a [][\^_‘wxyzabc] , és ha a "fr" nyelvi beállítást használjuk, akkor a [\xc8-\xcb] minta illeszkedik minden ékezetes e illetve E karakterre. (franciában van pár ilyen :)

A karakterosztályokhoz karakterfajtákat - \d, \D, \s, \S, \w és \W -t - is hozzáadhatunk, és így a fajtába tartozó karaterekkel bővíthejük. A [\dABCDEF] - például - minden hexadecimális számjegyre illeszkedik . A kalap(os ékezet)-t a nagybetűs (tagadó) fajtákkal együtt használva kényelmesen megadható a kisbetűs párjuknál korlátozotabb halmaz.A [^\W_] karakterosztály minden betűre és számjegyre illeszkedik, de az aláhúzásra nem (ami ugyancsak eleme a \w-nek).

A \, -, a kezdő ^ és a záró] karakterektől eltekintve a nem alfanumerikus karaktereknek nincs különleges jelentésük a karakterosztályokon belül, de nem okoz gondot, ha esetleg visszperjel előzi meg ezeket.

Retesz, pipa, csővezeték |

A "|" karakterrel vagylagos - alternatív - mintákat lehet megadni, pontosabban azokat választja el egymástól. Például

gyula|sacika

minta illeszkedik a "gyula" és a "sacika" szövegre is. Bármennyi vagylagos mintát fel lehet sorolni, köztük az üres mintát is, ami az üres sztringre illeszkedik. Az illesztés a mintákat balról jobbra, egymás után sorjában veszi, és a legelső sikeresen illesztett lesz használva. A részmintákban használt vagylagos ágak esetében a "sikeresen" azt jelenti, a minta fennmaradó részét is sikerül illeszteni.

Módosítók mintán belüli beállításai

A PCRE_CASELESS , PCRE_MULTILINE , PCRE_DOTALL és PCRE_EXTENDED módosítókat a minta belsejében is átállíthatjuk "(?" és ")" közé írt Perl-beli betűjelzéseikkel. Ezek a betűk:

i = PCRE_CASELESS
m = PCRE_MULTILINE
s = PCRE_DOTALL
x = PCRE_EXTENDED

A "(?im)" bekapcsolja a többsoros és a "kis-nagybetű mindegy" üzemmódot. Kikapcsolni az egyes beállításokat az eléjük írt kötőjellel lehet. Ezenkívül a módosítók együttes állítása is megengedett, például a (?im-sx) bekapcsolja a PCRE_CASELESS-t és a PCRE_MULTILINE-t valamint kikapcsolja PCRE_DOTALL-t és PCRE_EXTENDED-t. Ha egy betűjelzés a kötőjel mindkét oldalán szerepel, akkor a módosító ki lesz kapcsolva.

Ezeknek a beállításoknak az érvényességi köre attól függ, hogy a mintán belül hol fordulnak elő. Minden részmintán kívüli beállítás olyan, mintha az illesztés kezdetén lett volna megadva, ezért a következő minták tökéletesen ugyanúgy működnek:

(?i)abc
a(?i)bc
ab(?i)c
abc(?i)

, ami ugyanaz, mintha az "abc" mintát PCRE_CASELESS módosítóval adtuk volna meg. Másszóval a legfelső szintű beálllítások az egész mintára érvényesek, hacsak egy részmintában felül nem bírálod azokat. Ha a módosítónak több beállítása is szerepel a legfelső szinten, akkor a jobbról legutolsó fog érvényre

jutni.

Ha részmintán belül történik módosítóváltoztatás, akkor a hatás különböző (Perl 5.005 viselkedésbeli változása). A részmintán belüli módosító-állítás csak a részminta hátramaradó részét érinti, ezért

(a(?i)b)c

minta csak az "abc" és "aBc" szövegekre illeszkedik feltéve, hogy PCRE_CASELESS módosítót nem adtuk meg. Ez azt jelenti, hogy a minta különböző helyein a módosítók más és más értékeit lehet beállítani. Bármilyen változtatás egy vagylagos (választható) ágban a következő ágakra is hatással van ugyanabban a részmintában, például

(a(?i)b|c)

minta illeszkedik az "ab", "aB", "c" és "C" szövegekre még akkor is, ha "C"-re illeszkedve az nem is az első ágra esett a választás, ahol a módosító be lett állítva. Ennek az az oka, hogy a módosító beállítások hatásai már a kifejezés fordításakor / kiértékelésekor rögzülnek, máskülönben néha nagyon furcsa viselkedést tapasztalhatnánk.

A PCRE-specifikus PCRE_UNGREEDY és PCRE_EXTRA módosítókat a Perl-kompatibilisekhez hasonló módon állíthatók, az U és X karakterek használatával.A (?X) beállítás különleges, mert - még a legfelső szinten is - mindig hamarabb kell szerepelnie a mintában, mint a további kapcsolóknak. Legjobb a minta legelejére rakni.

Részminták

A részminták egymásba ágyazhatók, és kerek zárójel fogja közre őket. A minta egy darabját részmintaként szerepeltetni két dolgot jelent:

1. Elágazási lehetőségek helyét jelöli ki (csoportosítás):

szél(toló|kakas|)

minta illeszkedik a "szél", "széltoló" vagy "szélkakas" szavak egyikére. Zárójel nélkül a "széltoló"-ra, a "kakas"-ra vagy az üres sztringre illeszkedne.

2. A részmintát "gyűjtő" részmintává változtatja. Ha a teljes mintát sikerült illeszteni, akkor a tárgyszöveg azon részei, amelyek a "gyűjtő" részmintákra illeszkedtek,

a **pcre_exec()** függvény *ovector* paraméterében a hívó függvénynek lesznek átadva.
A kerek nyitó zárójelek számozása balról jobbra 1-től kezdve történik, és "gyűjtő" részminták (sor)számát adják.

Például, ha az "a vörös király" szöveget illesztjük

a ((vörös|sápadt) (király|királynő))

mintára, akkor a megtalált / begyűjtött szövegrészek :
1. "vörös király", 2. "vörös" és 3. "király".

Az a tény, hogy a kerek zárójelnek kettős szerepe van, nem minden esetben hasznos. Sokszor előfordulhat, hogy csak a csoportosító funkcióra van szükségünk "begyűjtés" nélkül. Erre szolgál a nyitó zárójel után irható "?:" karaktersorozat , mert ekkor a részminta nem gyűjti be / tárolja el a ráilleszkedő szövegrészt, és nem is számít bele a következő "gyűjtő" részminták sorszámozásába. Az előző példánál maradva "a sápadt királynő"

a ((?:vörös|sápadt) (király|királynő))

mintára illesztve a megtalált / begyűjtött részminták :
1. "sápadt királynő" és a 2. "királynő".
A "gyűjtő" részminták száma legfeljebb 99 lehet,és az összes részminta "gyűjtő" és "nem-gyűjtő" együtt legfeljebb 200.

Ha egy nem-gyűjtő részminta elején módosítókat kell állítani, akkor kényelmes rövidítésként használható, hogy a módosítók betűjelei a "?" és a ":" közé írhatók. Ebből adódóan az alábbi két minta

(?i:szombat|szerda)
(?:(?i)szombat|szerda)

pontosan ugyanazokra a sztringekre illeszkedik.
Mivel a vagylagos / választható ágak próbálgatása balról jobbra történik, és a módosító hatása addig él, amíg a részminta le nem zárul, az egyik ágban beállított módosító hatással van a következő ágra is, ezért a fenti minta illeszkedik a "SZERDA" és "Szombat" szavakra is.

ISMÉTLÉSEK

Az ismétléseket kvantorokkal (sokszorozókkal) lehet megadni a következő poziciók után:

egyszerű karakter (esetleg visszaperjellel bevezetve),

. metakarakter,
karakterosztály,
hivatkozás (ld. következő szakasz) vagy
zárójeles részminta (nem teszt)

után.

Az általános ismétlő kvantor (sokszorozó) - kapcsos zárójelben vesszővel elválasztva - írja elő azt minimális és maximális előfordulási számot, ahányszor a mintát illeszteni kell illetve lehet. Ezeknek 65536-nél kisebbnek kell lenniük, és az első legfeljebb annyi lehet, mint a második. Például a

```
z{2,4}
```

minta illeszkedik a "zz", "zzz" vagy "zzzz" szövegekre. Az egymagában álló záró kapcsos zárójel nem bír speciális jelentéssel. Ha a második számot nem adjuk meg, de a vessző szerepel, akkor az illesztésnek nincs felső határa, de ha a vesszőt is elhagyjuk, akkor a kvantor (sokszorozó) pontosan a megadott számú illesztést engedi csak. Ekképpen:

```
[aáeéiíoóöőuúüű]{3,}
```

legalább 3 egymást követő magánhangzóra illeszkedik, de akár többre is, míg a

```
\d{8}
```

csakis 8 darab számjegyre. Ha a nyitó kapcsos zárójel olyan helyen szerepel, ahol kvantor nem szerepelhet, akkor betű szerint lesz figyelembe véve, maga helyett áll a mintában. A {,6} például nem kvantor (sokszorozó), hanem egy négy karakteres minta betű szerint értelmezve.

A {0} kvantor (sokszorozó) megengedett, és hatása olyan, mintha az előtte álló mintaelem és maga a kvantor nem is szerepelne.

Kényelmi és történelmi okokból három, gyakori kvantornak létezik rövidítése is:

```
*    egyenlő a {0,} - val
+    egyenlő a {1,} - val
?    egyenlő a {0,1} - val
```

Végtelen ciklusokat is lehet definiálni úgy, ha egy olyan részmintát sokszorozzunk felsőhatár nélkül, amelynek nem muszáj illeszkednie egyetlen karakterre sem, például:

```
(a?)*
```

Perl és PCRE korábbi verziói hibát jeleztek az ilyen minták kiértékelésekor / fordításakor. Azonban vannak esetek,amikor ezek a minták is hasznosak lehetnek, ezért ezeket a mintákat most már elfogadják, de ha a részminták ismételten nem illeszkednek egy karakterre sem, a ciklust erőszakkal megszakítják.

Alapértelmezés szerint a kvantorok (sokszorozók) "mohók" , azaz amennyi karakterre csak illeszteni tudnak, azt mind elhasználják a tárgyszövegből a felső határ figyelembe vétele mellett és anélkül, hogy a minta többi része sikertelenül illeszkedne.
A klasszikus példa, ahol ez problémát okoz, a C forráskódok megjegyzéseire illeszkedő kifejezés. Ezek a megjegyzések nem egymásba ágyazhatók, /* és */ közé írt tetszőleges szövegek, amelyekben szerepelhet a * és / jel is (nem egymás mellett).

```
/\*.*\*/
```

mintát a

```
/* első megjegyzés */  c_code();  /* második megjegyzés */
```

C forrásrészletre illesztve kudarcot vallunk - köszönhetően a .* elem mohóságának. (Az illesztés sikeres, csak nem arra való, amire szerettük volna - a fordító).

Ha azonban a kvantort (sokszorozót) kérdőjel ? követi, akkor ez megszünteti annak mohóságát, és csak a lehető legkevesebb illesztést engedi, ezért a

```
/\*.*?\*/
```

minta az, ami C forráskódú megjegyzésekre való. A különböző kvantorok (sokszorozók) jelentése egyébként nem változik ettől, csak az előnyben részesített illeszkedési számuk. A kérdőjelnek ez a szerepe nem összetévesztendő a feltételes kvantorral (sokszorozóval), mert a kérdőjelet kétféleképpen és néha megkettőzve használjuk, például:

```
\d??\d
```

egy számjegyre illeszkedik leginkább, és csak akkor kettőre, ha a minta maradék része másképp nem illeszkedne.

Ha a PCRE_UNGREEDY módosító be van kapcsolva, akkor a kvantorok (sokszorozók) alapértelmezés szerint nem mohók (szerények :), csak ha egykérdőjel követi őket. Másszóval, a kérdőjel átállítja a kvantorok alapértelmezés szerinti viselkedését.

Amikor egy zárójelezett részminta egynél többször vagy

legfeljebb meghatározott számban ismétlődhet, akkor a kiértékelt minta több memóriát foglal el a minimális és maximális előfordulással arányosan.

Ha a minta .* -val vagy .{0,} -val kezdődik és a PCRE_DOTALL módosító (Perl-beli /s-nek megfelelője) be van kapcsolva, és így a . illeszkedik az újsor karakterre is, akkor a minta hallgatólagosan "rögzített" lesz, mert bármi is következik utána, az a tárgyszöveg minden karakterpozíciójára "rá lesz próbálva", így nincs az elsőn kívül más karakterpozíció, ahonnan az egész illesztés újrakezdődhetne. PCRE az ilyen mintákat úgy tekinti, mintha \A-val kezdődne. Azokban az esetekben, amikor tudjuk, hogy a tárgyszöveg nem tartalmaz újsorkaraktereket és a minta .* -gal kezdődik, érdemes beállítani a PCRE_DOTALL módosítót vagy ^ -t használni a "rögzítés" eléréséhez, hogy ezt az optimalizálást kieszközöljük.

Ha a "(be)gyűjtő" részminta az illesztés során ismételten felhasználásra kerül, akkor a begyűjtött / megtalált szövegrész az utolsó ismétlésben illeszkedő rész lesz. Például a

```
(csori[keat]{3}\s*)+
```

minta a "csorikea csoriaet" történő illesztése után a begyűjtött / megtalált szövegrész "csoriaet" lesz. Egymásba ágyazott részminták esetén a begyűjtött érték lehet, hogy az előző lépésben lett beállítva, például:

```
/(a|(b))+/
```

illesztve a "aba" -ra a második begyűjtött szövegrész a "b".

HIVATKOZÁSOK

Karakterosztályon kívül visszaperjelet követő számjegy(ek) a mintában korábban előforduló "gyűjtő" részmintára történő hivatkozásnak minősül(nek), ha a számnak megfelelő "gyűjtő" részminta nyitó kerek-zárójele (előfordult már a mintában.

Ha azonban ez a tízes számrendszerbeli szám 10-nél kisebb, akkor mindig hivatkozásnak minősül, s csak akkor okoz hibát, ha nincs ennyi "gyűjtő" részminta megnyitva az egész mintán belül. Magyarán a 10-nél kisebb sorszámú kerek nyitó zárójeleknek nem kell a hivatkozás előtt, a mintában ettől balra állniuk. Olvasd el a "visszaperjel \" szakaszt, ahol részletesen olvashatsz arról, hogy miképpen kezeli a PCRE a visszaperjel után álló számjegyeket.

A hivatkozások az általuk megcímzett "gyűjtő" részmintákra ténylegesen illeszkedő szövegrészeket testesítik meg, és nem azokat, amelyre illeszkedhettek volna egyébként. Mindjárt megvilágítjuk egy példán keresztül:

```
(ért|kegy) és \1elem
```

illeszkedik a "kegy és kegyelem"-re és "ért és értelem"-re, de nem illeszkedik az "ért és kegyelem"-re.
Ha a kis- és nagybetűk különbsége érvényben van a hivatkozás pillanatában, akkor számít a betűalakok különbsége. Például

```
((?i)rah)\s+\1
```

minta illeszkedik a "rah rah" és "RAH RAH" szövegekre, de a "RAH rah" -ra még akkor sem, ha az eredeti "gyűjtő" részmintában a kis- és nagybetűk különbsége nem számított.

Egy részmintára több hivatkozás is utalhat. Ha a részminta ténylegesen nem lett felhasználva az illesztés során, akkor bármilyen hivatkozás rá hibát jelent. Például

```
(a|(bc))\2
```

minta mindig hibázik, ha az illesztés "a"-val kezdődött és nem "bc"-vel.Mivel legfeljebb 99 hivatkozás lehet egy mintán belül, minden visszaperjel után álló számjegy egy lehetséges hivatkozás részét képezheti. Ha a mintát számjeggyel kellene folytatni, akkor a hivatkozás végét valamilyen elválasztókarakterrel kell jelezni. Ha PCRE_EXTENDED módosító be van kapcsolva, akkor ez bármilyen térközkarakter lehet, egyébként egy üres megjegyzés is megfelel.

Azok a hivatkozások, amelyek magára a hivatkozást tartalmazó részmintára utalnak,a részminta első illesztésekor hibáznak, ezért például (a\1) soha nem illeszkedik semmire. Azonban az ilyen hivatkozások is értelmet nyerhetnek, például az alábbi minta

```
(a|b\1)+
```

illeszkedik a csupa "a"-ból álló és az "aba", "ababaa" s ehhez hasonló sorozatokra. A részminta illesztés minden iterációjában a hivatkozás az előző iterációnak megfelelő karaktersorozatra utal. Azért, hogy ez működjön, a mintának olyannak kell lennie, hogy az első lépésben nem kelljen a hivatkozásnak illeszkednie.Ez vagylagos ágak vagy elhagyható - minimálisan 0 ismétlést előíró - kvantor (sokszorozó) használatával megoldható.

TESZTEK - FELTÉTELEK

Angolul "assertion"-nek nevezett szerkezet olyan tesztet vagy feltételt jelent, ami a pillanatnyi illesztési pozició előtti vagy utáni karakterekre vonatkozik, de ténylegesen nem használ el karaktereket a tárgyszövegből az illesztés elől, csak ellenőrzi egy feltétel fennállását. Az egyszerű teszteket - \b, \B, \A, \Z, \z, ^ és $ - már tárgyaltuk. Az ennél bonyolultabbakat részmintaként kell megadni.Kétféle teszt van: az előretekintő, ami a pillanatnyi illesztési pozició utáni karaktereket vizsgálja, és a hátratekintő, ami az az előttieket.

A teszt részmintája a szokásos módon illeszkedik, attól eltekintve, hogy a pillanatnyi illesztési pozíciót nem változtatja meg. Az előretekintő tesztek "(?=" vagy tagadó feltétel esetén "(?!" sorozattal kezdődnek, például

```
\w+(?=;)
```

illeszkedik minden szóra, amelyet pontosvessző követ, de a pontosvessző nem szerepel az illeszkedésben, és az

```
ize(?!bigyo)
```

illeszkedik a "ize" minden olyan előfordulására, amit nem a "bigyo" követ. Nyilvánvaló, hogy egy hasonló minta

```
(?!ize)bigyo
```

nem azokat a "bigyo"-kat találja meg, amelyek előtt nem "ize" van; hanem minden "bigyo"-t megtalál, mert a (?!ize) teszt mindig igaz, hiszen az utána jövő három karakter tényleg nem "ize", hanem "big" (tesztként nem változik a pillanatnyi illesztési pozició - a fordító). Ahhoz, hogy a kívánt hatást érjük el a hátratekintő tesztre van szükségünk.

A hátratekintő tesztek (?<= vagy tagadó feltétel esetén a (?<! sorozattal kezdődnek, például

```
(?<!ize)bigyo
```

azokat a "bigyo"-kat találja meg, amelyek előtt nincs "ize". A hátratekintő tesztek csak rögzített hosszúságúak lehetnek. Ha azonban több alternatívát adunk meg a teszten belül, a különböző ágaknak lehet eltérő a hossza, ilyenformán a

```
(?<=tulok|szamár)
```

megengedett, ellentétben a

```
(?<!kutyus(ok)?|macsk(ák)?)
```

mintával, amely kiértékelés / fordítás során hibát okoz.
Az eltérő hosszúságú ágtesztek a hátratekintő tesztek legfelső szintjén megengedettek. Ez PCRE bővítés a Perl 5.005-höz képest,ahol minden ágtesztnek azonos hosszúságúnak kell lennie. A

(?<=ab(c|de))

mintához hasonlók nem megengedettek, mert a legfelső szinten levő egyetlen ág két különböző hosszúságú szövegre is illeszkedhet, de átírható, hogy a vagylagosságot a felső szintre emeljük:

(?<=abc|abde)

A hátratekintő tesztek megvalósítása a következő: minden ágra a pillanatnyi illesztési pozició átmenetileg hátrébb kerül az ágtesztnek megfelelő számú karakterrel és ezután illeszteni próbál. Ha nincs elég karakter a visszalépéshez, az illesztés sikertelen lesz. A hátratekintő tesztek az egyszeri részmintákkal együtt különösen hasznosak lehetnek a szöveg végi illeszkedéseknél, az "Egyszeri részminták" c. szakasz végén található erre példa.

Számos különböző fajtájú teszt állhat egymás után, például:

(?<=\d{3})(?<!999)ize

minta azokra az "ize"-kre illeszkedik, ami előtt 3 számjegy áll, amely nem a "999". Figyelj arra, hogy minden teszt egymástól függetlenül, ugyanabban az illesztési pozicóban kerül kiértékelésre. Először azt ellenőrzi, hogy a megelőző három karakter számjegy-e és utána azt, hogy ezek nem a "999"-t adják.Ez a minta nem azokra az "ize"-kre illeszkedik, amelyek előtt hat karakter áll: három számjegy és másik három, ami nem a "999", példaképpen nem illeszkedik a "123abcize" szövegre. Az a minta, ami ezt tudja a következő:

(?<=\d{3}...)(?<!999)ize

Ebben az esetben az első teszt a megelőző hat karaktert nézi, és azt vizsgálja, hogy az első három számjegy-e, ezután a második feltétel a maradék három karaktert nézi, hogy azok nem a "999" sorozat. A tesztek egymásba ágyazhatók bármilyen komibinációban. Például a

(?<=(?<!ize)bigyo)figyula

arra a "figyula"-ra illeszkedik, amit egy "bigyo" előz meg, ami előtt nem áll "ize". Miközben

```
(?<=\d{3}(?!999)...)ize
```

egy másik minta az olyan "ize"-kre, amelyek előtt három számjegy s ezután három olyan karakter áll, amely nem "999".

A teszt-minták nem "gyűjtő" részminták, és nem ismételhetők, hiszen nincs sok értelme ugyanazt többször tesztelni. Ha bármilyen teszt "gyűjtő" részmintát tartalmaz, akkor ezek a részminták is beleszámítanak a "gyűjtő" részminták számozásába, azonban csak pozitív tesztekre lesz végrehajtva a megtalált szövegrész "begyűjtése", mert tagadó tesztek esetén ennek nincs értelme.

A tesztek beleszámítanak a zárójelezett részmintákra vonatkozó 200-as felső korlátba.

Egyszeri részminták

A mind a maximalizáló (mohó) mind a minimalizáló (szerény) ismétlés esetén a további részek sikertelen illesztése a általában az ismételt elem újbóli kiértékelését eredményezi, hogy eltérő ismétlésszámmal a minta maradéka illeszkedik-e. Néhány esetben hasznos ezt megelőzni akár a mintaillesztés viselkedésének megváltoztatásával, vagy hagyni, hogy az illesztés hamarabb elbukjon - mint egyébként -, ha a minta készítője tudja, hogy nincs értelme folytatni.

Nézzük a következő példát, ahol a \d+ize mintát a

```
123456bigyo
```

szövegre illesztjük.
Miután 6 számjeggyel végzett,az illesztés elbukik az "ize"-n. Ekkor a szokásos lépés, hogy megpróbálja újból 5 majd 4 és - és így tovább sorjában - eggyel kevesebb számjegyet illeszteni a "\d+"-re , amíg végül minden lehetőséget kimerítve az illesztés elbukik. Az egyszeri részminták azt a lehetőséget biztosítják nekünk, hogy ha egyszer a minta egy darabja illeszkedett, akkor az ne legyen újra és újra kiértékelve a fenti módon, hanem az illesztés fejeződjön be azonnal, ha az "ize"-t nem tudja illeszteni mar az első alkalommal. Ennek a jelölésére egy speciális zárójelezés szolgál (?>)

```
(?>\d+)bigyo
```

Ez a zárójelféle lezárja a közrefogott részmintát, ha az egyszer már illeszkedett,és az illesztés későbbi sikertelen szakaszában megakadályozza, hogy az illesztés visszalépjen ebbe a részbe. Emellett azonban az előző elemekre a

visszalépegetés a szokásos módon történik.

Másfajta leírása, hogy ez a fajta részminta úgy illeszkedik a szövegre, mint ahogy egy, a pillanatnyi illesztési ponthoz rögzített, de az eredetivel teljesen megegyező, önálló minta lenne.

Az egyszeri részminták nem "gyűjtő" részminták. Egyszerű esetekben egy maximalizált ismétlésnek tekinthető, amely minden lehetőt elnyel - ahogy a fenti példában. Miközben tehát a \d+ és a \d+? is annyi karaktert készül elnyelni, hogy a minta maradék része is illeszkedni tudjon, addig a (?>\d+) csakis a teljes számjegy sorozatra illeszkedik.

Ez a szerkezet egymásba ágyazható, és természetesen tartalmazhat tetszőlegesen bonyolult részmintákat is.

Az egyszeri részminták a hátratekintő tesztekkel közösen hatékony a szöveg végére illeszkedő mintát képesek megadni. Nézzük a következő mintát

```
abcd$
```

egy hosszú, nem illeszkedő szövegre alkalmazva. Mivel az illesztés balról jobbra történik, PCRE a tárgyszöveg minden "a" betűjét végignézi, és megvizsgálja, hogy onnantól a minta többi része illeszkedik-e. Ha a mintát így adjuk meg:

```
^.*abcd$
```

akkor a kezdő .* először a teljes szövegre illeszkedik, de mikor a minta maradéka elbukik - hiszen nem következik semmilyen "a" betű -, az algoritmus visszalép, hogy az utolsó karakter kivételével mindent elnyeljen, majd az utolsó két karakter kivételével, stb. Az "a" megkeresése mégegyszer lefedi az egész szöveget - most jobbról balról -, ez sem jobb. Ha azonban a mintát az alábbi alakra hozzuk

```
^(?>.*)(?<=abcd)
```

az eljárás nem tud visszalépni .* elemre, csak az egész szövegre tud illeszkedni a minta. A soron következő hátratekintő teszt pedig az utolsó négy karaktert ellenőrzi. Ha a teszt sikertelen, akkor az illesztés nyomban elbukik. Hosszú szövegek esetén ez a megközelítés jelentős - mérhető - különbséget jelent a feldolgozási időben.

A korlátlan ismétlésű részmintákat tartalmazó mintákban, amelyek maguk is korlátlan számban ismétlődhetnek, az egyszeri részmintával akadályozható meg egyedül az, hogy néhány sikertelen illesztés valóban hosszú ideig tartson. A

```
(\D+|<\d+>)*[!?]
```

minta azokra a szövegekre illeszkedik, amelyek korlátlan számú nem számjegy karakterek vagy <> közé írt számjegyek sorozatából és egy !-ből vagy ?-ből állnak. Ha illeszkedik, az gyorsan kiderül. Ám ha az

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

szövegre próbáljuk illeszteni, akkor sok időt vesz igénybe, mire "értesülünk a kudarcról". Ez azért van, mert a szöveget nagyon sokféleképpen lehet feldarabolni két ismétlés között és mindet végig kell próbálni. (A példában azért szerepelt [!?] a minta végén egy egyszerű karakter helyett, mert Perl és PCRE is rendelkezik egy beépített optimalizálással, amely lehetővé teszi a sikertelenség korai jelzését, mert megjegyzik az utolsó illesztendő karaktert és korán jeleznek, ha az nincs benne a tárgyszövegben.)
Ha a mintát a következőképp módosítjuk:

```
((?>\D+)|<\d+>)*[!?]
```

akkor a nem számjegyekből álló sorozatot nem lehet széttörni és hamar kiderül, hogy az illesztés nem sikerült.

Feltételes részminták

Lehetőség van arra, hogy egy teszt (assertion) eredményétől vagy egy előző "gyűjtő" részminta illeszkedésétől függően más-más részmintával folytatódjon az illesztés. A feltételes részminták két alakja:

```
(?(feltétel)igen-minta)
(?(feltétel)igen-minta|nem-minta)
```

Ha a feltétel teljesül, az "igen-minta" lesz illesztve, egyébként a "nem-minta", ha megadtunk ilyet. Ha 2-nél több választható részmintát adtunk meg, akkor az a kiértkelés - fordítás során hibát okoz.

Kétfajta feltételt lehet megadni.Ha a kerek zárójelek között csak számjegyek szerepelnek, akkor ezt egy hivatkozásként értelmezve a feltétel abban az esetben teljesül, ha a hivatkozott "gyűjtő" részminta előzőleg már illeszkedett a szöveg valamely részére. Nézzük a következő mintát, amelyben a térközök nem számítanak, csak az olvashatóságot és a könnyebb tárgyalást szolgálják. (Feltételezzük, hogy a PCRE_EXTENDED be van kapcsolva):

```
( \( )?    [^()]+    (?(1) \) )
```

A minta első része egy nem kötelező kerek nyitó zárójelre illeszkedik, és ha ez a karakter szerepel, akkor az első begyűjtött - megtalált szövegrész (\1 vagy $1 numerikus változó) ezt fogja tartlamazni. A minta második része a nem kerek zárójelekre illeszkedik. A harmadik részben található a feltételes részminta, ami azt teszteli, hogy az első részben a nyitózárójel illeszkedett-e vagy sem. Ha igen, azaz ha a tárgyszöveg nyitó zárójellel kezdődött, akkor az "igen-részminta" érvényre jut, és a bezáró kerek zárójel már kötelező lesz, egyébként - mivel nincs megadva "nem-minta" - semminek nem kell illeszkednie. Magyarán ez a minta az elhagyható zárójelek közé fogott zárójeleket nem tartalmazó karaktersorozatra illeszkedik.

Ha a feltételrészben nem csak számjegyek vannak, akkor annak kötelezően egy tesztnek (assertion) kell lennie. Ez lehet akár előre- vagy hátratekintő, állító vagy tagadó felétel is. Tekintsünk most egy, két választható mintából álló példát, ahol a térközök szintén nem számítanak:

```
(?(?=[^a-z]*[a-z])
\d{2}-[a-z]{3}-\d{2} | \d{2}-\d{2}-\d{2} )
```

```
(Az első sorban a tesztfeltétel,
 a másodikban a két választható minta.)
```

A feltétel olyan előretekintő teszt, ami azt vizsgálja, hogy egy betű előtt elhagyható, nem betűkből álló karaktersorozat áll, azaz legalább egy betű jelenlétét vizsgálja. Ha ez a betű jelen van a szövegben, akkor a tárgyszöveg az első mintára illeszkedik, egyébként a másodikra. Ez a minta vagy a "dd-aaa-dd" vagy a "dd-dd-dd" alakú szövegekre illeszkedik, ahol "aaa" három betűt, "dd" pedig két számjegyet jelez.

Megjegyzések

A (?# karartersorozat megjegyzések kezdetét jelzi a mintán belül, amely a következő bezáró kerek zárójelig tart. A megjegyzéseket nem lehet egymásba ágyazni, és a bennük előforduló karakterek nem játszanak semmilyen szerepet a mintaillesztés során.

Ha a PCRE_EXTENDED módosító be van kapcsolva, akkor karakterosztályon kívüli visszaperjel nélküli # karakter "egy soros" megjegyzést vezet be, amely a legközelebbi újsor karakterig tart.

Rekurzív (önhivatkozó) minták

Nézzük az egymásba ágyazható zárójeles kifejezésekre illeszkedő minta problémáját. Rekurzió használata nélkül a legtöbb, amit tehetünk, hogy rögzített számú egymásba ágyazhatóságot kezelő mintát használunk. Nem megoldható a tetszőleges mélységű egymásbaágyazhatóság.
Perl 5.6 kísérleti jelleggel biztosítja a lehetőséget a reguláris kifejezések rekurzív bejárására (több más mellett). A különleges (?R) elem jelenti a meghatározott önhivatkozási pontot.
A PCRE minta, amely megoldja a zárójelezett szövegek problémáját (feltéve, hogy a PCRE_EXTENDED módosító be van kapcsolva, tehát a térközök nem részei a mintának) a következő:

```
\( ( (?>[^()]+) | (?R) )* \)
```

Először egy nyitó zárójelre illeszkedik, ezután tetszőleges hosszúságú karakterresorozatra, amely vagy nem tartalmaz zárójeleket, vagy rekurzívan az egész mintára illeszkedik (a zárójelezett szövegekre). Végül a bezáró zárójel következik.

Ez a sajátságos minta korlátlan számú, egymásba ágyazott ismétlést takar, ezért a zárójel nélküli szövegrészre illesztésnél az egyszeri részminta használata nagyon fontos, kiváltképp ha nem illeszkedő tárgyszövegre alkalmazzuk. Ha például a

```
(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa()
```

szövegre próbáljuk a mintát, akkor nagyon hamar megkapjuk az eredményt. Ha nem használnánk az egyszeri részmintát, akkor az illesztés nagyon sokáig tartana, mert rengeteg különböző módon lehet a tárgyszöveget felszeletelni a * és + kvantorok (sokszorozók) között, és minden lehetőséget ki kell próbálni, mire az illesztést végleg sikertelennek lehet minősíteni.

A "gyűjtő" részminták által begyűjtött szövegek a rekurzió során a legutoljára illesztett szövegrészek.A fenti mintát a

```
(ab(cd)ef)
```

szövegre alkalmazva a "begyűjtött" / megtalált rész az "ef" lesz, amely a felső szinten az utolsó felvett érték. Ha további zárójeleket helyezünk el a mintában ekképpen:

```
\( ( ( (?>[^()]+) | (?R) )* ) \)
  ^                   ^
  ^                   ^
```

, akkor a "ab(cd)ef" szöveg lesz legkülső "gyűjtő" zárójel értéke. Ha 15-nél több "gyűjtő" zárójel van a mintában, akkor PCRE tartalék memóriát foglal le, hogy az adatokat itt tárolja a rekurzió alatt (pcre_malloc-kal foglalja le, és a pcre_free-vel szabadítja fel azután.) Ha nem lehet több memóriához hozzájutni, akkor csak az első 15 "gyűjtő" zárójelnek menti el az adatait, minthogy lehetetlen "nincs elég memória" hibaüzenetet kiadni a rekurzión belül.

Teljesítmény

Bizonyos minták sokkal hatékonyabbak, mint mások, például [aáeéiíoóöőuúüű] karakterosztály hatékonyabb, mint ugyanezt választható mintákkal (a|á|e|é|i|í|o|ó|ö|ő|u|ú|ü|ű) megoldani. Általában a legegyszerűbb szerkezet, amely a kívánt viselkedést biztosítja rendszerint a leghatékonyabb. Jeffrey Friedl könyve rengeteg példát tartalmaz, hogyan kell reguláris kifejezéseket optimalizálni a hatékony végrehajtáshoz.

Ha a minta .* -gal kezdődik, és a PCRE_DOTALL módosító be van kapcsolva, akkor PCRE a mintát hallagtólagosan "rögzíti", mivel az csak a tárgyszöveg elejére illeszkedhet. Ha ez a módosító nincs beállítva, akkor nem élhet ilyen optimalizálással, mivel a . metakarakter nem illeszkedik az újsorkarakterre, ezért előfordulhat, hogy a mintát minden egyes újsortól illeszteni kell, nem csak a szöveg elejére. A

(.*) második

minta úgy illeszkedik az "első\nés második" szövegre (ahol a \n az újsort testesíti meg), hogy az első "begyűjtött" szövegrész az "és" lesz. Ehhez a PCRE-nek minden újsor után újból meg kell próbálni az illesztést.

Ha az ilyesfajta mintákat olyan szövegekre alkalmazod, amelyekben nincsenek sortörések (újsorkarakterek), akkor a legjobb teljesítmény a PCRE_DOTALL módosító beállításával vagy a "rögzítés" egyértelmű megadásával lehetséges: ^.*. Ez megkíméli a PCRE-t, hogy végignézze a tárgyszöveget újsorkaraktert keresve, ahonnan az illesztést újrakezdheti.

Vigyázz azokkal a mintákkal, amelyek egymásba ágyazott határozatlan számú ismétlést tartalmaznak. Ezek a minták sok időt vesznek el, ha nem illeszkedő szövegre kell alkalmazni őket. Nézzük a

(a+)*

mintát! Ez 33 féleképpen illeszkedik az "aaaa" szövegre, és

ez a szám rohamosan növekszik, ahogy a szöveg egyre hosszabb lesz. (A * illeszkedhet 0-szor, 1-szer, 2-szer, 3-szor vagy 4-szer, és - 0-t kivéve - minden esetben a + is különböző számúszor.) Amikor a minta fennmaradó része ilyen és az illesztés egésze sikertelenül érne vége, PCRE elvben végigpróbálja az összes lehetőséget, ami rendkívül sok időt vehet igénybe.

Néhány egyszerűbb minta esetén optimalizálás történik, mint például

(a+)*b

, ahol literális (önmagára illeszkedő) karakter követi a vizsgált részt. Mielőtt a hagyományos illesztési eljárás elkezdődik, PCRE megvizsgálja, hogy előfordul-e később a mintában "b", és ha nem, akkor azonnal sikertelennek minősíti az illesztést. Amikor azonban nem literális követi a mintarészt, akkor ez az optimalizáció nem használható. A különbséget megvizsgálhatod a

(a+)*\d

minta viselkedésével összehasonlítva. Az előbbi szinte azonnal véget ér, ha csupa "a"-ból álló sorozatra illesztik, míg az utóbbi kipróbálása jelentős időt vesz el 20 karakternél hosszabb tárgyszövegekre.

preg_grep (PHP 4)

Visszaadja azokat a tömbelemeket, amelyek illeszkednek a mintára

array **preg_grep** (string pattern, array input) \linebreak

A **preg_grep()** függvény egy olyan tömbbel tér vissza, amelyben az *input* tömb azon elemei szerepelnek, amelyek a *pattern* mintára illeszkedtek.

PHP 4.0.4-től kezdve a **preg_grep()** függvény eredményeként visszaadott tömb az *input* indexeit használja. Ha ez nem előnyös, akkor az array_values() függvénnyel lehet a **preg_grep()** által visszaadott tömböt újraindexelni.

Példa 1. preg_grep() példa

```
// visszad minden olyan $tomb-beli elemet, ami lebegőpontos számot tartalmaz
$lebegopontos_tomb = preg_grep ("/^(\d+)?\.\d+$/", $tomb);
```

preg_match_all (PHP 3>= 3.0.9, PHP 4)

Reguláris kifejezés teljeskörű illesztését végzi

int **preg_match_all** (string pattern, string subject, array matches [, int order]) \linebreak

A *`subject`* szövegben megkeresi az összes, a *`pattern`* mintára illeszkedő részt, és a *`matches`* tömbbe menti azokat az *`order`* paraméterben meghatározott sorrendben.

Amint az első illeszkedés megtalálta, a soron következő illesztést az utolsó illeszkedés végétől folytatja.

Az *`order`* paraméter két értéket vehet fel:

PREG_PATTERN_ORDER

Azt eredményezi, hogy a `$matches[0]` olyan tömb lesz, ahol az összes a teljes mintára illeszkedő szövegrész lesz egymás után, a `$matches[1]` tömbben pedig azok a szövegrészek, amelyek első "gyűjtő" részmintára illeszkednek, és így tovább.

```
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>példa: </b><div align=left>ez itt a teszt szöveg</div>",
    $out, PREG_PATTERN_ORDER);
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n"
```

Ez a példa az alábbi eredményt adja:

```
<b>példa: </b>, <div align=left>ez itt a teszt szöveg</div>
példa: , ez itt a teszt szöveg
```

Tehát, az `$out[0]` egy tömb az összes, teljes mintára illeszkedő szöveggel, és `$out[1]` egy tömb az összes HTML-elem közti szöveggel.

PREG_SET_ORDER

Azt eredményezi, hogy $matches[0]-ben az első illesztés eredményeit, a $matches[1] a másodikét fogja tartalmazni, és így tovább.

```
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>példa: </b><div align=left>ez itt a teszt szöveg</div>",
    $out, PREG_SET_ORDER);
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n"
```

Ez a példa az alábbi eredményt adja:

```
<b>példa: </b>, példa:
<div align=left>ez itt a teszt szöveg</div>, ez itt a teszt szöveg
```

Ebben az esetben `$matches[0]` az első illesztés során megtalált szövegrészeket tartalmazza, a `$matches[0][0]`-ban a teljes mintára illeszkedő szöveggel, `$matches[0][1]`-ban az első "gyűjtő" részmintára illeszkedő résszel, és így tovább. Ehhez hasonlóan `$matches[1]` a második illesztés során megtalált szövegeket tartalmazza, stb.

Ha nincs megadva az *`order`* paraméter, akkor az alapértelmezés a PREG_PATTERN_ORDER.

Ez a függvény az összes illeszkedés számával tér vissza, vagy `FALSE`-szal sikertelen illesztés vagy hiba esetén.

Példa 1. Az összes telefonszám kigyűjtése egy adott szövegből.

```
preg_match_all ("/\(?  (\d{3})?  \)?  (?(1)  [\-\s] ) \d{3}-\d{4}/x",
                "Hívd a 555-1212 vagy 1-800-555-1212 számot", $phones);
```

Példa 2. HTML elemek keresése (mohón)

```
// A \\2 példa a "gyűjtő" részmintára történő hivatkozásra. Ez mondja meg
// PCRE-nek, hogy a másodikként megadott kerek zárójelre - ami ebben az
// esetben ([\w]+) - illesztett szövegnek kell szerepelnie a hivatkozás
// helyén is. A plusz visszaperjelre azért van szükség, mert a minta
// macskakörmök ("") közé van fogva.
$html = "<b>félkövér szöveg</b><a href=howdy.html>kattints ide</a>

preg_match_all ("/(<([\w]+)[^<]*>)(.*)(<\/\\2>)/", $html,
$matches);

for ($i=0; $i < count($matches[0]); $i++) {
echo "passzolt: ".$matches[0][$i]."\n";
echo "1. rész: ".$matches[1][$i]."\n";
echo "2. rész: ".$matches[3][$i]."\n";
echo "3. rész: ".$matches[4][$i]."\n\n";
}
```

A példa ezt eredményezi:

```
passzolt: <b>félkövér szöveg</b>
1. rész: <b>
2. rész: félkövér szöveg
3. rész: </b>
```

```
passzolt: <a href=howdy.html>kattints ide</a>
1. rész: <a href=howdy.html>
2. rész: kattints ide
3. rész: </a>
```

Lásd még: preg_match(), preg_replace(), és preg_split()!

preg_match (PHP 3>= 3.0.9, PHP 4)

Reguláris kifejezést illeszt a megadott szövegre

int **preg_match** (string pattern, string subject [, array matches]) \linebreak

A *subject* szövegben egyezést keres a *pattern* mintában megadott reguláris kifejezéssel.

A *matches* paraméterrel meghívva a keresés végeredményét ebbe a tömbbe tárolja el a függvény. A $matches[0] tartalmazza azt a szövegrészt, ami a teljes mintára illeszkedett, és a további elemek ($matches[1] és így tovább) azokat, amelyek a tömbindex szerinti "gyűjtő" részmintákra illeszkedtek.

Ez a függvény TRUE-val tér vissza, ha a *pattern* mintát sikerült illesztenie a *subject* szövegre, egyébként - sikertelen illesztés vagy hiba esetén - FALSE-szal.

Példa 1. A "php" sztring keresése

```
// az "i" a minta határolójele után azt jelzi,
// hogy a kis- és nagybetűkre közömbös a keresés.
if (preg_match ("/php/i", "PHP a web-scripting nyelvek egyike.")) {
    print "A minta passzol.";
} else {
    print "A minta nem passzol.";
}
```

Példa 2. A "web" szó keresése

```
// a mintában a \b szóhatárt jelöl, ezért csak a különálló
// "web" szavakra fog illeszkedni, és szórészletekre
// úgymint "webbing" vagy "cobweb" pedig nem.
if (preg_match ("/\bweb\b/i", "PHP a web scripting nyelvek egyike."))
{
    print "A minta passzol.";
} else {
    print "A match nem passzol.";
}
if (preg_match ("/\bweb\b/i", "PHP a website scripting nyelvek egyike.")) {
```

```
    print "A minta passzol.";
} else {
    print "A match nem passzol.";
}
```

Példa 3. domain-név kinyerése URL-ből

```
// gazdagép neve az URL-ből
preg_match("/^(http:\/\/)?([^\/]+)/i",
"http://www.php.net/index.html", $matches);
$host = $matches[2];
// a gazdagépnév utolsó két része
preg_match("/[^\.\/]+\.[^\.\/]+$/",$host,$matches);
echo "A domain név : ".$matches[0]."\n";
```

Ez a példa a következőt eredményezi:

```
A domain név : php.net
```

Lásd még: preg_match_all(), preg_replace() és preg_split()!

preg_quote (PHP 3>= 3.0.9, PHP 4)

Reguláris kifejezés metakaraktereit literálissá változtatja

string **preg_quote** (string str [, string delimiter]) \linebreak

A **preg_quote()** függvény az *`str`* paraméterben kapott szövegben minden olyan karakter elé egy visszaperjelet helyez el, amely egy reguláris kifejezésben speciális jelentéssel bírhat. Ez akkor hasznos, ha olyan futási idő alatt összeállított sztringet kell egy mintában felhasználni, amelyben előfordulhatnak speciális jelentéssel bíró reguláris karakterek.

Ha a nem kötelező *`delimiter`*-t is megadod, akkor ez a karakter is visszaperjellel lesz bevezetve. Ez pedig azért hasznos, mert PCRE által megkövetelt határolójeleket is le lehet kezelni így. A / a leggyakrabban használt határolójel.

A reguláris kifejezésekben előforduló speciális karakterek:

```
. \ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

Példa 1.

```
$kulcsszavak = "$40 egy g3/400-ért";
$kulcsszavak = preg_quote ($keywords, "/");
echo $kulcsszavak; // \$40 egy g3\/400-ért adja vissza
```

Példa 2. Adott szöveg dőlt betűssé alakítása HTML-ben

```
// Ebben a példában preg_quote($szo)-t használjuk arra, hogy
// a csillagokat megfosszuk a reguláris kifejezésekben használt
// speciális jelentésüktől.

$szoveg = "Ezt a könyvet *nagyon* nehéz megtalálni.";
$szo    = "*nagyon*";
$szoveg = preg_replace ("/".preg_quote($szo)."/",
                        "<i>".$szo."</i>;",
                        $szoveg);
```

preg_replace_callback (PHP 4 >= 4.0.5)

Reguláris kifejezés alapján keres és a megadott függvényt meghívva végzi a cserét

mixed **preg_replace_callback** (mixed pattern, mixed callback, mixed subject [, int limit]) \linebreak

Ennek a függvénynek a működése szinte azonos a preg_replace() függvényével azt leszámítva, hogy *replacement* paraméterként egy *callback* függvényt kell megadni, amely minden sikeres illesztésnél meghívásra kerül. Az illeszkedő szövegeket egy tömbben adja át a PHP a 'meghívott' függvénynek, és annak a helyettesítési szöveggel kell visszatérnie. Ez a függvény a PHP 4.0.5-től elérhető.

Lásd még: preg_replace()!

preg_replace (PHP 3>= 3.0.9, PHP 4)

Reguláris kifejezés alapján keres és cserél

mixed **preg_replace** (mixed pattern, mixed replacement, mixed subject [, int limit]) \linebreak

A *subject*-ben megadott szövegben keres a *pattern* paraméterben megadott mintára illeszkedő részeket és azokat lecseréli a *replacement*-ben megadott kifejezésre. Ha a *limit* is szerepel, akkor csak az első *limit* számú illeszkedő részt cseréli le. Ha *limit* hiányzik vagy értéke -1, akkor minden illeszkedő részt cserél.

A *replacement*-ben hivatkozásokat is el lehet helyezni *n* vagy - PHP 4.0.4-től kezdve - *$n* alakban. (Ez utóbbit részesítsd előnyben.) Minden ilyen hivatkozás az *n*. zárójelezett részminta által megtalált szöveggel lesz helyettesítve. Az *n* értéke 0-tól 99-ig terjedhet, ahol is a \\0 vagy $0 hivatkozás a teljes mintára illeszkedő szöveget jelenti. A kerek nyitó zárójelek 1-től kezdve balról jobbra vannak számozva és az általuk bevezetett "gyűjtő" részminták sorszámát adják.

Ha volt egyezés, akkor az új *subject*-tel tér vissza a függvény, egyébként a változatlanul hagyott, eredeti *subject*-tel.

A **preg_replace()**-nek átadott minden paraméter lehet tömb is.

Ha *subject* tömb típusú, akkor a keresés és a helyettesítés a tömb minden elemén végrehajtódik, és a visszatérési érték is tömb lesz.

Ha a *pattern* és *replacement* paraméterek tömb típusúak, akkor a **preg_replace()** veszi a tömbök elemeit és mindegyikkel páronként elvégzi a keresést és a helyettesítést a *subject* szövegen. Ha a *replacement* tömbnek a *pattern* tömbnél a kevesebb eleme van, akkor a pár nélküli mintákat üres sztringgel fogja helyettesíteni. Ha a *pattern* tömb és *replacement* sztring típusú, akkor ezt a helyettesítési szöveget használja minden *pattern*-beli mintához. A fordított esetnek nem lenne sok értelme.

Az /e módosító hatására a **preg_replace()** a *replacement* paramétert PHP kódként értelmezi, miután a hivatkozások behelyettesítését elvégezte. Jó tanácsként: érdemes megbizonyosodni arról, hogy *replacement* érvényes PHP kódot tartalmaz (sztringben megadva), másképp a PHP szintaktikai hibát (Parse Error) fog jelezni abban a sorban, ahol a **preg_replace()** függvényt meghívtad.

Példa 1. Néhány érték cseréje

```
$mintak = array ("/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/",
                 "/^\s*{(\w+)}\s*=/");
$csere  = array ("\\3/\\4/\\1\\2", "$\\1 =");
print preg_replace ($mintak, $csere, "{kezdoDatum} = 1999-5-27");
```

Ennek a példának a végeredménye:

```
$kezdoDatum = 5/27/1999
```

Példa 2. Az /e módosítót használata:

```
preg_replace ("/(<\/?)(\w+)([^>]*>)/e",
              "'\\1'.strtoupper('\\2').'\\3'",
              $html_body);
```

Ez az összes HTML elemet nagybetűsre cseréli a bemeneti szövegben.

Példa 3. HTML-ből sima szöveggé konvertálás

```
// A $dokumentum HTML formátumú.
// Minden HTML elemet, javascript szakaszt és térközt eltávolít a
// dokumentumból. Ezenkívül néhány megszokott HTML entitást is a
// sima, karakteres megfelelőikre konvertál.

$mit = array ("'<script[^<]*?>.*?</script>'si", // javascript eltüntetése
              "'<[\/\!]*?[^<>]*?>'si",  // HTML elemek eltüntetése
              "'([\r\n])[\s]+'",  // térközök
              "'&(quot|#34);'i",  // HTML entitások
              "'&(amp|#38);'i",
              "'&(lt|#60);'i",
              "'&(gt|#62);'i",
              "'&(nbsp|#160);'i",
              "'&(iexcl|#161);'i",
              "'&(cent|#162);'i",
              "'&(pound|#163);'i",
              "'&(copy|#169);'i",
              "'&#(\d+);'e");  // PHP kódként értelmezze

$mire = array ("",
               "",
               "\\1",
               "\"",
               "&",
               "<",
               ">",
               " ",
               chr(161),
               chr(162),
               chr(163),
               chr(169),
               "chr(\\1)");

$szoveg = preg_replace ($mit, $mire, $dokumentum);
```

Megjegyzés: A *`limit`* paraméter a PHP 4.0.1pl2 után került a nyelvbe.

Lásd még: preg_match(), preg_match_all() és preg_split()!

preg_split (PHP 3>= 3.0.9, PHP 4)

Sztringet darabol fel reguláris kifejezésre illeszkedő részei mentén

array **preg_split** (string pattern, string subject [, int limit [, int flags]]) \linebreak

Megjegyzés: A *`flags`* paraméterrel a PHP 4 Beta 3-ban lett kiegészítve.

Ez a függvény a *`pattern`* minta illesztésének határai mentén darabolja fel a *`subject`* szöveget, és a szövegdarabokat tömbbe összefogva adja vissza.

Ha a *`limit`* meg van adva, akkor csak *`limit`* számú elemmel tér vissza. A *`limit`* értéke lehet -1 (jelentése: nincs korlát), és ez akkor hasznos, ha a *`flags`* paramétert egy konkrét korlát megadása nélkül kell beállítani.

A *`flags`* a következő jelzőknek bármilyen kombinációja lehet a (bitszintű | operátorral):

PREG_SPLIT_NO_EMPTY

Ekkor csak a nem üres szövegrészeket adja vissza a **preg_split()**.

PREG_SPLIT_DELIM_CAPTURE

Ekkor a kerekzárójelek közé fogott ("gyűjtő") részmintára illeszkedő szövegrészeket is visszadja a függvény. Ez a jelző 4.0.5 verziótól használható.

Példa 1. preg_split() példa : a keresett szöveg kucslszavait adja vissza.

```
// Feldarabolja a kifejezést vesszők és térközkarakterek mentén,
// amelyek a következők " ", \r, \t, \n és \f .
$kulcsszavak = preg_split ("/[\s,]+/", "hypertext language, programming");
```

Példa 2. Sztring karakterekre darabolása

```
$sztring = 'karakterfűzér';
$karakterek = preg_split('//', $sztring, -1, PREG_SPLIT_NO_EMPTY);
print_r($karakterek);
```

Lásd még: spliti(), split(), implode(), preg_match(), preg_match_all() és preg_replace()!

LXXXVIII. qtdom functions

qdom_error (PHP 4 >= 4.0.5)

Returns the error string from the last QDOM operation or FALSE if no errors occured

string **qdom_error** (void) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

qdom_tree (PHP 4 >= 4.0.4)

creates a tree of an xml string

object **qdom_tree** (string) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

LXXXIX. Regular Expression Functions (POSIX Extended)

> **Megjegyzés:** PHP also supports regular expressions using a Perl-compatible syntax using the PCRE functions. Those functions support non-greedy matching, assertions, conditional subpatterns, and a number of other features not supported by the POSIX-extended regular expression syntax.

> **Figyelem**
>
> These regular expression functions are not binary-safe. The PCRE functions are.

Regular expressions are used for complex string manipulation in PHP. The functions that support regular expressions are:

- ereg()
- ereg_replace()
- eregi()
- eregi_replace()
- split()
- spliti()

These functions all take a regular expression string as their first argument. PHP uses the POSIX extended regular expressions as defined by POSIX 1003.2. For a full description of POSIX regular expressions see the regex man pages included in the regex directory in the PHP distribution. It's in manpage format, so you'll want to do something along the lines of **man /usr/local/src/regex/regex.7** in order to read it.

Példa 1. Regular Expression Examples

```
ereg ("abc", $string);
/* Returns true if"abc"
   is found anywhere in $string. */

ereg ("^abc", $string);
/* Returns true if "abc";
   is found at the beginning of $string. */

ereg ("abc$", $string);
/* Returns true if "abc"
   is found at the end of $string. */

eregi ("(ozilla.[23]|MSIE.3)", $HTTP_USER_AGENT);
/* Returns true if client browser
   is Netscape 2, 3 or MSIE 3. */
```

```
ereg ("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)", $string,$regs);
/* Places three space separated words
   into $regs[1], $regs[2] and $regs[3]. */

$string = ereg_replace ("^", "<br />", $string);
/* Put a <br /> tag at the beginning of $string. */

$string = ereg_replace ("$", "<br />", $string);
/* Put a <br />; tag at the end of $string. */

$string = ereg_replace ("\n", "", $string);
/* Get rid of any newline
   characters in $string. */
```

ereg_replace (PHP 3, PHP 4)

Replace regular expression

string **ereg_replace** (string pattern, string replacement, string string) \linebreak

> **Megjegyzés:** preg_replace(), which uses a Perl-compatible regular expression syntax, is often a faster alternative to **ereg_replace()**.

This function scans *`string`* for matches to *`pattern`*, then replaces the matched text with *`replacement`*.

The modified string is returned. (Which may mean that the original string is returned if there are no matches to be replaced.)

If *`pattern`* contains parenthesized substrings, *`replacement`* may contain substrings of the form `\\`*`digit`*, which will be replaced by the text matching the digit'th parenthesized substring; `\\0` will produce the entire contents of string. Up to nine substrings may be used. Parentheses may be nested, in which case they are counted by the opening parenthesis.

If no matches are found in *`string`*, then *`string`* will be returned unchanged.

For example, the following code snippet prints "This was a test" three times:

Példa 1. ereg_replace() Example

```
$string = "This is a test";
echo ereg_replace (" is", " was", $string);
echo ereg_replace ("( )is", "\\1was", $string);
echo ereg_replace ("(( )is)", "\\2was", $string);
```

One thing to take note of is that if you use an integer value as the *`replacement`* parameter, you may not get the results you expect. This is because **ereg_replace()** will interpret the number as the ordinal value of a character, and apply that. For instance:

Példa 2. ereg_replace() Example

```
<?php
/* This will not work as expected. */
$num = 4;
$string = "This string has four words.";
$string = ereg_replace('four', $num, $string);
echo $string;   /* Output: 'This string has   words.' */

/* This will work. */
$num = '4';
$string = "This string has four words.";
$string = ereg_replace('four', $num, $string);
echo $string;   /* Output: 'This string has 4 words.' */
?>
```

Példa 3. Replace URLs with links

```
$text = ereg_replace("[[:alpha:]]+://[^<>[:space:]]+[[:alnum:]/]",
                     "<a href=\"\\0\">\\0</a>", $text);
```

See also ereg(), eregi(), eregi_replace(), str_replace(), and preg_match().

ereg (PHP 3, PHP 4)

Regular expression match

int **ereg** (string pattern, string string [, array regs]) \linebreak

> **Megjegyzés:** preg_match(), which uses a Perl-compatible regular expression syntax, is often a faster alternative to **ereg()**.

Searches a *`string`* for matches to the regular expression given in *`pattern`*.

If matches are found for parenthesized substrings of *`pattern`* and the function is called with the third argument *`regs`*, the matches will be stored in the elements of the array *`regs`*. $regs[1] will contain the substring which starts at the first left parenthesis; $regs[2] will contain the substring starting at the second, and so on. $regs[0] will contain a copy of the complete string matched.

> **Megjegyzés:** Up to (and including) PHP 4.1.0 `$regs` will be filled with exactly ten elements, even though more or fewer than ten parenthesized substrings may actually have matched. This has no effect on **ereg()**'s ability to match more substrings. If no matches are found, `$regs` will not be altered by **ereg()**.

Searching is case sensitive.

Returns TRUE if a match for *`pattern`* was found in *`string`*, or FALSE if no matches were found or an error occurred.

The following code snippet takes a date in ISO format (YYYY-MM-DD) and prints it in DD.MM.YYYY format:

Példa 1. ereg() Example

```
if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs)) {
    echo "$regs[3].$regs[2].$regs[1]";
```

```
} else {
    echo "Invalid date format: $date";
}
```

See also eregi(), ereg_replace(), eregi_replace(), and preg_match().

eregi_replace (PHP 3, PHP 4)

replace regular expression case insensitive

string **eregi_replace** (string pattern, string replacement, string string) \linebreak

This function is identical to ereg_replace() except that this ignores case distinction when matching alphabetic characters.

See also ereg(), eregi(), and ereg_replace().

eregi (PHP 3, PHP 4)

case insensitive regular expression match

int **eregi** (string pattern, string string [, array regs]) \linebreak

This function is identical to ereg() except that this ignores case distinction when matching alphabetic characters.

Példa 1. eregi() example

```
if (eregi("z", $string)) {
    echo "'$string' contains a 'z' or 'Z'!";
}
```

See also ereg(), ereg_replace(), and eregi_replace().

split (PHP 3, PHP 4)

split string into array by regular expression

array **split** (string pattern, string string [, int limit]) \linebreak

> **Megjegyzés:** preg_split(), which uses a Perl-compatible regular expression syntax, is often a faster alternative to **split()**.

Returns an array of strings, each of which is a substring of *`string`* formed by splitting it on boundaries formed by the regular expression *`pattern`*. If *`limit`* is set, the returned array will contain a maximum of *`limit`* elements with the last element containing the whole rest of *`string`*. If an error occurs, **split()** returns `FALSE`.

To split off the first four fields from a line from `/etc/passwd`:

Példa 1. split() Example

```
list($user,$pass,$uid,$gid,$extra)= split (":", $passwd_line, 5);
```

> **Megjegyzés:** If there are *n* occurrences of *`pattern`*, the returned array will contain *n*`+1` items. For example, if there is no occurrence of *`pattern`*, an array with only one element will be returned. Of course, this is also true if *`string`* is empty.

To parse a date which may be delimited with slashes, dots, or hyphens:

Példa 2. split() Example

```
$date = "04/30/1973";  // Delimiters may be slash, dot, or hyphen
list ($month, $day, $year) = split ('[/.-]', $date);
echo "Month: $month; Day: $day; Year: $year<br>\n";
```

Note that *`pattern`* is case-sensitive.

Note that if you don't require the power of regular expressions, it is faster to use explode(), which doesn't incur the overhead of the regular expression engine.

For users looking for a way to emulate Perl's **@chars = split(", $str)** behaviour, please see the examples for preg_split().

Please note that *`pattern`* is a regular expression. If you want to split on any of the characters which are considered special by regular expressions, you'll need to escape them first. If you think **split()** (or any other regex function, for that matter) is doing something weird, please read the file `regex.7`, included in the `regex/` subdirectory of the PHP distribution. It's in manpage format, so you'll want to do something along the lines of **man /usr/local/src/regex/regex.7** in order to read it.

See also: preg_split(), spliti(), explode(), implode(), chunk_split(), and wordwrap().

spliti (PHP 4 >= 4.0.1)

Split string into array by regular expression case insensitive

array **spliti** (string pattern, string string [, int limit]) \linebreak

This function is identical to split() except that this ignores case distinction when matching alphabetic characters.

See also **preg_spliti()**, split(), explode(), and implode().

sql_regcase (PHP 3, PHP 4)

Make regular expression for case insensitive match

string **sql_regcase** (string string) \linebreak

Returns a valid regular expression which will match *`string`*, ignoring case. This expression is *`string`* with each character converted to a bracket expression; this bracket expression contains that character's uppercase and lowercase form if applicable, otherwise it contains the original character twice.

Példa 1. sql_regcase() Example

```
echo sql_regcase ("Foo bar");
```

prints

```
[Ff][Oo][Oo] [Bb][Aa][Rr]
```

.

This can be used to achieve case insensitive pattern matching in products which support only case sensitive regular expressions.

XC. Semaphore, Shared Memory and IPC Functions

This module provides wrappers for the System V IPC family of functions. It includes semaphores, shared memory and inter-process messaging (IPC).

Semaphores may be used to provide exclusive access to resources on the current machine, or to limit the number of processes that may simultaneously use a resource.

This module provides also shared memory functions using System V shared memory. Shared memory may be used to provide access to global variables. Different httpd-daemons and even other programs (such as Perl, C, ...) are able to access this data to provide a global data-exchange. Remember, that shared memory is NOT safe against simultaneous access. Use semaphores for synchronization.

Táblázat 1. Limits of Shared Memory by the Unix OS

SHMMAX	max size of shared memory, normally 131072 bytes
SHMMIN	minimum size of shared memory, normally 1 byte
SHMMNI	max amount of shared memory segments on a system, normally 100
SHMSEG	max amount of shared memory segments per process, normally 6

The messaging functions may be used to send and receive messages to/from other processes. They provide a simple and effective means of exchanging data between processes, without the need for setting up an alternative using unix domain sockets.

Megjegyzés: These functions do not work on Windows systems.

ftok (PHP 4 >= 4.2.0)

Convert a pathname and a project identifier to a System V IPC key

int **ftok** (string pathname, string proj) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

msg_get_queue (PHP 4 CVS only)

Create or attach to a message queue

int **msg_get_queue** (int key [, int perms]) \linebreak

msg_get_queue() returns an id that can be used to access the System V message queue with the given *key*. The first call creates the message queue with the optional *perms* (default: 0666). A second call to **msg_get_queue()** for the same *key* will return a different message queue identifier, but both identifiers access the same underlying message queue. If the message queue already exists, the *perms* will be ignored.

See also: msg_remove_queue(), msg_receive(), msg_send(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_receive (PHP 4 CVS only)

Receive a message from a message queue

bool **msg_receive** (int queue, int desiredmsgtype, int msgtype, int maxsize, mixed message [, bool unserialize [, int flags [, int errorcode]]]) \linebreak

msg_receive() will receive the first message from the specified *queue* of the type specified by *desiredmsgtype*. The type of the message that was received will be stored in *msgtype*. The maximum size of message to be accepted is specified by the *maxsize*; if the message in the queue is larger than this size the function will fail (unless you set *flags* as described below). The received message will be stored in *message*, unless there were errors receiving the message, in which case the optional *errorcode* will be set to the value of the system errno variable to help you identify the cause.

If *desiredmsgtype* is 0, the message from the front of the queue is returned. If *desiredmsgtype* is greater than 0, then the first message of that type is returned. If *desiredmsgtype* is less than 0, the first message on the queue with the lowest type less than or equal to the absolute value of *desiredmsgtype* will be read. If no messages match the criteria,

your script will wait until a suitable message arrives on the queue. You can prevent the script from blocking by specifying MSG_IPC_NOWAIT in the *flags* parameter.

unserialize defaults to TRUE; if it is set to TRUE, the message is treated as though it was serialized using the same mechanism as the session module. The message will be unserialized and then returned to your script. This allows you to easily receive arrays or complex object structures from other PHP scripts, or if you are using the WDDX serializer, from any WDDX compatible source. If *unserialize* is FALSE, the message will be returned as a binary-safe string.

The optional *flags* allows you to pass flags to the low-level msgrcv system call. It defaults to 0, but you may specify one or more of the following values (by adding or ORing them together).

Táblázat 1. Flag values for msg_receive

MSG_IPC_NOWAIT	If there are no messages of the *desiredmsgtype*, return immediately and do not wait. The function will fail and return an integer value corresponding to ENOMSG.
MSG_EXCEPT	Using this flag in combination with a *desiredmsgtype* greater than 0 will cause the function to receive the first message that is not equal to *desiredmsgtype*.
MSG_NOERROR	If the message is longer than *maxsize*, setting this flag will truncate the message to *maxsize* and will not signal an error.

Upon successful completion the message queue data structure is updated as follows: *msg_lrpid* is set to the process-ID of the calling process, *msg_qnum* is decremented by 1 and *msg_rtime* is set to the current time.

msg_receive() returns TRUE on success or FALSE on failure. If the function fails, the optional *errorcode* will be set to the value of the system errno variable.

See also: msg_remove_queue(), msg_send(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_remove_queue (PHP 4 CVS only)

Destroy a message queue

bool **msg_remove_queue** (int queue) \linebreak

msg_remove_queue() destroys the message queue specified by the *queue*. Only use this function when all processes have finished working with the message queue and you need to release the system resources held by it.

See also: **msg_remove_queue()**, msg_receive(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_send (PHP 4 CVS only)

Send a message to a message queue

bool **msg_send** (int queue, int msgtype, mixed message [, bool serialize [, bool blocking [, int errorcode]]]) \linebreak

msg_send() sends a *message* of type *msgtype* (which MUST be greater than 0) to a the message queue specified by *queue*.

If the message is too large to fit in the queue, your script will wait until another process reads messages from the queue and frees enough space for your message to be sent. This is called blocking; you can prevent blocking by setting the optional *blocking* parameter to FALSE, in which case **msg_send()** will immediately return FALSE if the message is too big for the queue, and set the optional *errorcode* to EAGAIN, indicating that you should try to send your message again a little later on.

The optional *serialize* controls how the *message* is sent. *serialize* defaults to TRUE which means that the *message* is serialized using the same mechanism as the session module before being sent to the queue. This allows complex arrays and objects to be sent to other PHP scripts, or if you are using the WDDX serializer, to any WDDX compatible client.

Upon successful completion the message queue data structure is updated as follows: *msg_lspid* is set to the process-ID of the calling process, *msg_qnum* is incremented by 1 and *msg_stime* is set to the current time.

See also: msg_remove_queue(), msg_receive(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_set_queue (PHP 4 CVS only)

Set information in the message queue data structure

bool **msg_set_queue** (int queue, array data) \linebreak

msg_set_queue() allows you to change the values of the msg_perm.uid, msg_perm.gid, msg_perm.mode and msg_qbytes fields of the underlying message queue data structure. You specify the values you require by setting the value of the keys that you require in the *data* array.

Changing the data structure will require that PHP be running as the same user that created the the queue, owns the queue (as determined by the existing msg_perm.xxx fields), or be running with root privileges. root privileges are required to raise the msg_qbytes values above the system defined limit.

See also: msg_remove_queue(), msg_receive(), msg_stat_queue() and **msg_set_queue()**.

This function was introduced in PHP 4.3.0 (not yet released).

msg_stat_queue (PHP 4 CVS only)

Returns information from the message queue data structure

array **msg_stat_queue** (int queue) \linebreak

msg_stat_queue() returns the message queue meta data for the message queue specified by the *queue*. This is useful, for example, to determine which process sent the message that was just received.

The return value is an array whose keys and values have the following meanings:

Táblázat 1. Array structure for msg_stat_queue

msg_perm.uid	The uid of the owner of the queue.
msg_perm.gid	The gid of the owner of the queue.
msg_perm.mode	The file access mode of the queue.
msg_stime	The time that the last message was sent to the queue.
msg_rtime	The time that the last message was received from the queue.
msg_ctime	The time that the queue was last changed.
msg_qnum	The number of messages waiting to be read from the queue.
msg_qbytes	The number of bytes of space currently available in the queue to hold sent messages until they are received.
msg_lspid	The pid of the process that sent the last message to the queue.
msg_lrpid	The pid of the process that received the last message from the queue.

See also: msg_remove_queue(), msg_receive(), **msg_stat_queue()** and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

sem_acquire (PHP 3>= 3.0.6, PHP 4)

Acquire a semaphore

bool **sem_acquire** (int sem_identifier) \linebreak

Returns: TRUE on success, FALSE on error.

sem_acquire() blocks (if necessary) until the semaphore can be acquired. A process attempting to acquire a semaphore which it has already acquired will block forever if acquiring the semaphore would cause its max_acquire value to be exceeded.

After processing a request, any semaphores acquired by the process but not explicitly released will be released automatically and a warning will be generated.

See also: sem_get() and sem_release().

sem_get (PHP 3>= 3.0.6, PHP 4)

Get a semaphore id

int **sem_get** (int key [, int max_acquire [, int perm]]) \linebreak

Returns: A positive semaphore identifier on success, or FALSE on error.

sem_get() returns an id that can be used to access the System V semaphore with the given key. The semaphore is created if necessary using the permission bits specified in perm (defaults to 0666). The number of processes that can acquire the semaphore simultaneously is set to max_acquire (defaults to 1). Actually this value is set only if the process finds it is the only process currently attached to the semaphore.

A second call to **sem_get()** for the same key will return a different semaphore identifier, but both identifiers access the same underlying semaphore.

See also: sem_acquire(), sem_release() and ftok().

Megjegyzés: This function does not work on Windows systems.

sem_release (PHP 3>= 3.0.6, PHP 4)

Release a semaphore

bool **sem_release** (int sem_identifier) \linebreak

Returns: TRUE on success, FALSE on error.

sem_release() releases the semaphore if it is currently acquired by the calling process, otherwise a warning is generated.

After releasing the semaphore, sem_acquire() may be called to re-acquire it.

See also: sem_get() and sem_acquire().

Megjegyzés: This function does not work on Windows systems.

sem_remove (PHP 4 >= 4.1.0)

Remove a semaphore

bool **sem_remove** (int sem_identifier) \linebreak

Returns: TRUE on success, FALSE on error.

sem_remove() removes the semaphore *`sem_identifier`* if it has been created by sem_get(), otherwise generates a warning.

After removing the semaphore, it is no more accessible.

See also: sem_get(), sem_release() and sem_acquire().

Megjegyzés: This function does not work on Windows systems. It was added on PHP 4.1.0.

shm_attach (PHP 3>= 3.0.6, PHP 4)

Creates or open a shared memory segment

int **shm_attach** (int key [, int memsize [, int perm]]) \linebreak

shm_attach() returns an id that that can be used to access the System V shared memory with the given key, the first call creates the shared memory segment with mem_size (default: sysvshm.init_mem in the configuration file, otherwise 10000 bytes) and the optional perm-bits (default: 0666).

A second call to **shm_attach()** for the same *`key`* will return a different shared memory identifier, but both identifiers access the same underlying shared memory. *`Memsize`* and *`perm`* will be ignored.

See also: ftok().

Megjegyzés: This function does not work on Windows systems.

shm_detach (PHP 3>= 3.0.6, PHP 4)

Disconnects from shared memory segment

int **shm_detach** (int shm_identifier) \linebreak

shm_detach() disconnects from the shared memory given by the *`shm_identifier`* created by shm_attach(). Remember, that shared memory still exist in the Unix system and the data is still present.

shm_get_var (PHP 3>= 3.0.6, PHP 4)

Returns a variable from shared memory

mixed **shm_get_var** (int id, int variable_key) \linebreak

shm_get_var() returns the variable with a given *`variable_key`*. The variable is still present in the shared memory.

Megjegyzés: This function does not work on Windows systems.

shm_put_var (PHP 3>= 3.0.6, PHP 4)

Inserts or updates a variable in shared memory

int **shm_put_var** (int shm_identifier, int variable_key, mixed variable) \linebreak

Inserts or updates a *`variable`* with a given *`variable_key`*. All variable-types are supported.

Megjegyzés: This function does not work on Windows systems.

shm_remove_var (PHP 3>= 3.0.6, PHP 4)

Removes a variable from shared memory

int **shm_remove_var** (int id, int variable_key) \linebreak

Removes a variable with a given *`variable_key`* and frees the occupied memory.

Megjegyzés: This function does not work on Windows systems.

shm_remove (PHP 3>= 3.0.6, PHP 4)

Removes shared memory from Unix systems

int **shm_remove** (int shm_identifier) \linebreak

Removes shared memory from Unix systems. All data will be destroyed.

Megjegyzés: This function does not work on Windows systems.

XCI. SESAM database functions

SESAM/SQL-Server is a mainframe database system, developed by Fujitsu Siemens Computers, Germany. It runs on high-end mainframe servers using the operating system BS2000/OSD.

In numerous productive BS2000 installations, SESAM/SQL-Server has proven ...

- the ease of use of Java-, Web- and client/server connectivity,
- the capability to work with an availability of more than 99.99%,
- the ability to manage tens and even hundreds of thousands of users.

Now there is a PHP3 SESAM interface available which allows database operations via PHP-scripts.

Configuration notes: There is no standalone support for the PHP SESAM interface, it works only as an integrated Apache module. In the Apache PHP module, this SESAM interface is configured using Apache directives.

Táblázat 1. SESAM Configuration directives

Directive	Meaning
`php3_sesam_oml`	Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. Example: `php3_sesam_oml $.SYSLNK.SESAM-SQL.030`

`php3_sesam_configfile`	Name of SESAM application configuration file. Required for using SESAM functions. Example:
	
	`php3_sesam_configfile $SESAM.SESAM.CONF.AW`
	
	It will usually contain a configuration like (see SESAM reference manual):
	
	`CNF=B`
	`NAM=K`
	`NOTYPE`

`php3_sesam_messagecatalog`	Name of SESAM message catalog file. In most cases, this directive is not neccessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive. Example: `php3_sesam_messagecatalog $.SYSMES.SESAM-SQL.030`

In addition to the configuration of the PHP/SESAM interface, you have to configure the SESAM-Database server itself on your mainframe as usual. That means:

- starting the SESAM database handler (DBH), and
- connecting the databases with the SESAM database handler

To get a connection between a PHP script and the database handler, the `CNF` and `NAM` parameters of the selected SESAM configuration file must match the id of the started database handler.

In case of distributed databases you have to start a SESAM/SQL-DCN agent with the distribution table including the host and database names.

The communication between PHP (running in the POSIX subsystem) and the database handler (running outside the POSIX subsystem) is realized by a special driver module called SQLSCI and SESAM connection modules using common memory. Because of the common memory access, and because PHP is a static part of the web server, database accesses are very fast, as they do not require remote accesses via ODBC, JDBC or UTM.

Only a small stub loader (SESMOD) is linked with PHP, and the SESAM connection modules are pulled in from SESAM's OML PLAM library. In the configuration, you must tell PHP the name of this PLAM library, and the file link to use for the SESAM configuration file (As of SESAM V3.0, SQLSCI is available in the SESAM Tool Library, which is part of the standard distribution).

Because the SQL command quoting for single quotes uses duplicated single quotes (as opposed to a single quote preceded by a backslash, used in some other databases), it is advisable to set the PHP configuration directives `php3_magic_quotes_gpc` and `php3_magic_quotes_sybase` to `On` for all PHP scripts using the SESAM interface.

Runtime considerations: Because of limitations of the BS2000 process model, the driver can be loaded only after the Apache server has forked off its server child processes. This will slightly slow down the initial SESAM request of each child, but subsequent accesses will respond at full speed.

When explicitly defining a Message Catalog for SESAM, that catalog will be loaded each time the driver is loaded (i.e., at the initial SESAM request). The BS2000 operating system prints a message after successful load of the message catalog, which will be sent to Apache's error_log file. BS2000 currently does not allow suppression of this message, it will slowly fill up the log.

Make sure that the SESAM OML PLAM library and SESAM configuration file are readable by the user id running the web server. Otherwise, the server will be unable to load the driver, and will not allow to call any SESAM functions. Also, access to the database must be granted to the user id under which the Apache server is running. Otherwise, connections to the SESAM database handler will fail.

Cursor Types: The result cursors which are allocated for SQL "select type" queries can be either "sequential" or "scrollable". Because of the larger memory overhead needed by "scrollable" cursors, the default is "sequential".

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: `SESAM_SEEK_NEXT`) and the scrolling offset which can either be set once by sesam_seek_row() or each time when fetching a row using sesam_fetch_row(). When fetching a row using a

"scrollable" cursor, the following post-processing is done for the global default values for the scrolling type and scrolling offset:

Táblázat 2. Scrolled Cursor Post-Processing

Scroll Type	Action
SESAM_SEEK_NEXT	none
SESAM_SEEK_PRIOR	none
SESAM_SEEK_FIRST	set scroll type to SESAM_SEEK_NEXT
SESAM_SEEK_LAST	set scroll type to SESAM_SEEK_PRIOR
SESAM_SEEK_ABSOLUTE	Auto-Increment internal offset value
SESAM_SEEK_RELATIVE	none. (maintain global default *offset* value, which allows for, e.g., fetching each 10th row backwards)

Porting note: Because in the PHP world it is natural to start indexes at zero (rather than 1), some adaptions have been made to the SESAM interface: whenever an indexed array is starting with index 1 in the native SESAM interface, the PHP interface uses index 0 as a starting point. E.g., when retrieving columns with sesam_fetch_row(), the first column has the index 0, and the subsequent columns have indexes up to (but not including) the column count ($array["count"]). When porting SESAM applications from other high level languages to PHP, be aware of this changed interface. Where appropriate, the description of the respective php sesam functions include a note that the index is zero based.

Security concerns: When allowing access to the SESAM databases, the web server user should only have as little privileges as possible. For most databases, only read access privilege should be granted. Depending on your usage scenario, add more access rights as you see fit. Never allow full control to any database for any user from the 'net! Restrict access to php scripts which must administer the database by using password control and/or SSL security.

Migration from other SQL databases: No two SQL dialects are ever 100% compatible. When porting SQL applications from other database interfaces to SESAM, some adaption may be required. The following typical differences should be noted:

- Vendor specific data types

  Some vendor specific data types may have to be replaced by standard SQL data types (e.g., TEXT could be replaced by VARCHAR(max. size)).

- Keywords as SQL identifiers

  In SESAM (as in standard SQL), such identifiers must be enclosed in double quotes (or

renamed).

- Display length in data types

  SESAM data types have a precision, not a display length. Instead of `int(4)` (intended use: integers up to '9999'), SESAM requires simply `int` for an implied size of 31 bits. Also, the only datetime data types available in SESAM are: `DATE`, `TIME(3)` and `TIMESTAMP(3)`.

- SQL types with vendor-specific `unsigned`, `zerofill`, or `auto_increment` attributes

  `Unsigned` and `zerofill` are not supported. `Auto_increment` is automatic (use `"INSERT ... VALUES(*, ...)"` instead of `"... VALUES(0, ...)"` to take advantage of SESAM-implied auto-increment.

- **int ... DEFAULT '0000'**

  Numeric variables must not be initialized with string constants. Use **DEFAULT 0** instead. To initialize variables of the datetime SQL data types, the initialization string must be prefixed with the respective type keyword, as in: `CREATE TABLE exmpl ( xtime timestamp(3) DEFAULT TIMESTAMP '1970-01-01 00:00:00.000' NOT NULL );`

- **$count = xxxx_num_rows();**

  Some databases promise to guess/estimate the number of the rows in a query result, even though the returned value is grossly incorrect. SESAM does not know the number of rows in a query result before actually fetching them. If you REALLY need the count, try **SELECT COUNT(...) WHERE ...**, it will tell you the number of hits. A second query will (hopefully) return the results.

- **DROP TABLE thename;**

  In SESAM, in the **DROP TABLE** command, the table name must be either followed by the keyword `RESTRICT` or `CASCADE`. When specifying `RESTRICT`, an error is returned if there are dependent objects (e.g., VIEWs), while with `CASCADE`, dependent objects will be deleted along with the specified table.

Notes on the use of various SQL types: SESAM does not currently support the BLOB type. A future version of SESAM will have support for BLOB.

At the PHP interface, the following type conversions are automatically applied when retrieving SQL fields:

Táblázat 3. SQL to PHP Type Conversions

SQL Type	PHP Type
SMALLINT, INTEGER	integer
NUMERIC, DECIMAL, FLOAT, REAL, DOUBLE	float
DATE, TIME, TIMESTAMP	string
VARCHAR, CHARACTER	string

When retrieving a complete row, the result is returned as an array. Empty fields are not filled in, so you will have to check for the existence of the individual fields yourself (use isset() or empty() to test for empty fields). That allows more user control over the appearance of empty fields (than in the case of an empty string as the representation of an empty field).

Support of SESAM's "multiple fields" feature: The special "multiple fields" feature of SESAM allows a column to consist of an array of fields. Such a "multiple field" column can be created like this:

Példa 1. Creating a "multiple field" column

```
CREATE TABLE multi_field_test (
    pkey CHAR(20) PRIMARY KEY,
    multi(3) CHAR(12)
)
```

and can be filled in using:

Példa 2. Filling a "multiple field" column

```
INSERT INTO multi_field_test (pkey, multi(2..3) )
    VALUES ('Second', <'first_val', 'second_val'>)
```

Note that (like in this case) leading empty sub-fields are ignored, and the filled-in values are collapsed, so that in the above example the result will appear as multi(1..2) instead of multi(2..3).

When retrieving a result row, "multiple columns" are accessed like "inlined" additional columns. In the example above, "pkey" will have the index 0, and the three "multi(1..3)" columns will be accessible as indices 1 through 3.

For specific SESAM details, please refer to the SESAM/SQL-Server documentation (english) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_en.htm) or the SESAM/SQL-Server documentation (german) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_gr.htm), both available online, or use the respective manuals.

sesam_affected_rows (PHP 3 CVS only)

Get number of rows affected by an immediate query

int **sesam_affected_rows** (string result_id) \linebreak

result_id is a valid result id returned by sesam_query().

Returns the number of rows affected by a query associated with *result_id*.

The **sesam_affected_rows()** function can only return useful values when used in combination with "immediate" SQL statements (updating operations like INSERT, UPDATE and DELETE) because SESAM does not deliver any "affected rows" information for "select type" queries. The number returned is the number of affected rows.

See also: sesam_query() and sesam_execimm()

```
$result = sesam_execimm ("DELETE FROM PHONE WHERE LASTNAME = '".strtoupper ($name)."'");
if (!$result) {
    ... error ...
}
print sesam_affected_rows ($result).
    " entries with last name ".$name." deleted.\n"
```

sesam_commit (PHP 3 CVS only)

Commit pending updates to the SESAM database

bool **sesam_commit** (void) \linebreak

Returns: TRUE on success, FALSE on errors

sesam_commit() commits any pending updates to the database.

Note that there is no "auto-commit" feature as in other databases, as it could lead to accidental data loss. Uncommitted data at the end of the current script (or when calling sesam_disconnect()) will be discarded by an implied sesam_rollback() call.

See also: sesam_rollback().

Példa 1. Committing an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (!sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)"))
        die("insert failed");
    if (!sesam_commit())
        die("commit failed");
}
?>
```

sesam_connect (PHP 3 CVS only)

Open SESAM database connection

bool **sesam_connect** (string catalog, string schema, string user) \linebreak

Returns TRUE if a connection to the SESAM database was made, or FALSE on error.

sesam_connect() establishes a connection to an SESAM database handler task. The connection is always "persistent" in the sense that only the very first invocation will actually load the driver from the configured SESAM OML PLAM library. Subsequent calls will reuse the driver and will immediately use the given catalog, schema, and user.

When creating a database, the *"catalog"* name is specified in the SESAM configuration directive **//ADD-SQL-DATABASE-CATALOG-LIST ENTRY-1 = *CATALOG(CATALOG-NAME = catalogname,...)**

The *"schema"* references the desired database schema (see SESAM handbook).

The *"user"* argument references one of the users which are allowed to access this *"catalog"* / *"schema"* combination. Note that *"user"* is completely independent from both the system's user id's and from HTTP user/password protection. It appears in the SESAM configuration only.

See also sesam_disconnect().

Példa 1. Connect to a SESAM database

```
<?php
if (!sesam_connect ("mycatalog", "myschema", "otto")
    die("Unable to connect to SESAM");
?>
```

sesam_diagnostic (PHP 3 CVS only)

Return status information for last SESAM call

array **sesam_diagnostic** (void) \linebreak

Returns an associative array of status and return codes for the last SQL query/statement/command. Elements of the array are:

Element	Contents

Táblázat 1. Status information returned by sesam_diagnostic()

Element	Contents
$array["sqlstate"]	5 digit SQL return code (see the SESAM manual for the description of the possible values of SQLSTATE)
$array["rowcount"]	number of affected rows in last update/insert/delete (set after "immediate" statements only)
$array["errmsg"]	"human readable" error message string (set after errors only)
$array["errcol"]	error column number of previous error (0-based; or -1 if undefined. Set after errors only)
$array["errlin"]	error line number of previous error (0-based; or -1 if undefined. Set after errors only)

In the following example, a syntax error (E SEW42AE ILLEGAL CHARACTER) is displayed by including the offending SQL statement and pointing to the error location:

Példa 1. Displaying SESAM error messages with error position

```
<?php
// Function which prints a formatted error message,
// displaying a pointer to the syntax error in the
// SQL statement
function PrintReturncode ($exec_str) {
    $err = Sesam_Diagnostic();
    $colspan=4; // 4 cols for: sqlstate, errlin, errcol, rowcount
    if ($err["errlin"] == -1)
        --$colspan;
    if ($err["errcol"] == -1)
        --$colspan;
    if ($err["rowcount"] == 0)
        --$colspan;
    echo "<TABLE BORDER>\n";
    echo "<TR><TH COLSPAN=".$colspan."><FONT COLOR=red>ERROR:</FONT> ".
   htmlspecialchars($err["errmsg"])."</TH></TR>\n";
    if ($err["errcol"] >= 0) {
        echo "<TR><TD COLSPAN=".$colspan."><PRE>\n";
        $errstmt = $exec_str."\n";
        for ($lin=0; $errstmt != ""; ++$lin) {
            if ($lin != $err["errlin"]) { // $lin is less or greater than errlin
                if (!($i = strchr ($errstmt, "\n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                if ($line != "\n")
                    print htmlspecialchars ($line);
            } else {
                if (! ($i = strchr ($errstmt, "\n")))
```

```
                    $i = "";
                $line = substr ($errstmt, 0, strlen ($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                for ($col=0; $col < $err["errcol"]; ++$col)
                    echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
                echo "<FONT COLOR=RED><BLINK>\\</BLINK></FONT>\n";
                print "<FONT COLOR=\"#880000\">".htmlspecialchars($line)."</FONT>";
                for ($col=0; $col < $err["errcol"]; ++$col)
                    echo (substr ($line, $col, 1) == "\t") ? "\t" : ".";
                echo "<FONT COLOR=RED><BLINK>/</BLINK></FONT>\n";
            }
        }
        echo "</PRE></TD></TR>\n";
    }
    echo "<TR>\n";
    echo " <TD>sqlstate=" . $err["sqlstate"] . "</TD>\n";
    if ($err["errlin"] != -1)
        echo " <TD>errlin=" . $err["errlin"] . "</TD>\n";
    if ($err["errcol"] != -1)
        echo " <TD>errcol=" . $err["errcol"] . "</TD>\n";
    if ($err["rowcount"] != 0)
         echo " <TD>rowcount=" . $err["rowcount"] . "</TD>\n";
    echo "</TR>\n";
    echo "</TABLE>\n";
}

if (!sesam_connect ("mycatalog", "phoneno", "otto"))
  die ("cannot connect");

$stmt = "SELECT * FROM phone\n".
        " WHERE@ LASTNAME='KRAEMER'\n".
        " ORDER BY FIRSTNAME";
if (!($result = sesam_query ($stmt)))
    PrintReturncode ($stmt);
?>
```

See also: sesam_errormsg() for simple access to the error string only

sesam_disconnect (PHP 3 CVS only)

Detach from SESAM connection

bool **sesam_disconnect** (void) \linebreak

Returns: always `TRUE`.

sesam_disconnect() closes the logical link to a SESAM database (without actually disconnecting and unloading the driver).

Note that this isn't usually necessary, as the open connection is automatically closed at the end of the script's execution. Uncommitted data will be discarded, because an implicit sesam_rollback() is executed.

sesam_disconnect() will not close the persistent link, it will only invalidate the currently defined *"catalog"*, *"schema"* and *"user"* triple, so that any sesam function called after **sesam_disconnect()** will fail.

See also: sesam_connect().

Példa 1. Closing a SESAM connection

```
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    ... some queries and stuff ...
    sesam_disconnect();
}
```

sesam_errormsg (PHP 3 CVS only)

Returns error message of last SESAM call

string **sesam_errormsg** (void) \linebreak

Returns the SESAM error message associated with the most recent SESAM error.

```
if (!sesam_execimm ($stmt))
  printf ("%s<br>\n", sesam_errormsg());
```

See also: sesam_diagnostic() for the full set of SESAM SQL status information

sesam_execimm (PHP 3 CVS only)

Execute an "immediate" SQL-statement

string **sesam_execimm** (string query) \linebreak

Returns: A SESAM "result identifier" on success, or FALSE on error.

sesam_execimm() executes an "immediate" statement (i.e., a statement like UPDATE, INSERT or DELETE which returns no result, and has no INPUT or OUTPUT variables). "select type" queries can not be used with **sesam_execimm()**. Sets the *affected_rows* value for retrieval by the sesam_affected_rows() function.

Note that sesam_query() can handle both "immediate" and "select-type" queries. Use **sesam_execimm()** only if you know beforehand what type of statement will be executed. An attempt to use SELECT type queries with **sesam_execimm()** will return `$err["sqlstate"] == "42SBW"`.

The returned "result identifier" can not be used for retrieving anything but the sesam_affected_rows(); it is only returned for symmetry with the sesam_query() function.

```
$stmt = "INSERT INTO mytable VALUES (’one’, ’two’)";
$result = sesam_execimm ($stmt);
$err = sesam_diagnostic();
print ("sqlstate = ".$err["sqlstate"]."\n".
       "Affected rows = ".$err["rowcount"]." == ".
       sesam_affected_rows($result)."\n");
```

See also: sesam_query() and sesam_affected_rows().

sesam_fetch_array (PHP 3 CVS only)

Fetch one row as an associative array

array **sesam_fetch_array** (string result_id [, int whence [, int offset]]) \linebreak

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

sesam_fetch_array() is an alternative version of sesam_fetch_row(). Instead of storing the data in the numeric indices of the result array, it stores the data in associative indices, using the field names as keys.

result_id is a valid result id returned by sesam_query() (select type queries only!).

For the valid values of the optional *whence* and *offset* parameters, see the sesam_fetch_row() function for details.

sesam_fetch_array() fetches one row of data from the result associated with the specified result identifier. The row is returned as an associative array. Each result column is stored with an associative index equal to its column (aka. field) name. The column names are converted to lower case.

Columns without a field name (e.g., results of arithmetic operations) and empty fields are not stored in the array. Also, if two or more columns of the result have the same column names, the later column will take precedence. In this situation, either call sesam_fetch_row() or make an alias for the column.

```
SELECT TBL1.COL AS FOO, TBL2.COL AS BAR FROM TBL1, TBL2
```

A special handling allows fetching "multiple field" columns (which would otherwise all have the same column names). For each column of a "multiple field", the index name is constructed by appending the string "(n)" where n is the sub-index of the multiple field column, ranging from 1 to its declared repetition factor. The indices are NOT zero based, in order to match the nomenclature used in the respective query syntax. For a column declared as:

```
CREATE TABLE ... ( ... MULTI(3) INT )
```

the associative indices used for the individual "multiple field" columns would be `"multi(1)"`, `"multi(2)"`, and `"multi(3)"` respectively.

Subsequent calls to **sesam_fetch_array()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or `FALSE` if there are no more rows.

Példa 1. SESAM fetch array

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table:
print "<TABLE BORDER>\n";
while (($row = sesam_fetch_array ($result)) && count ($row) > 0) {
    print " <TR>\n";
    print " <TD>".htmlspecialchars ($row["firstname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["lastname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["phoneno"])."</TD>\n";
    print " </TR>\n";
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: sesam_fetch_row() which returns an indexed array.

sesam_fetch_result (PHP 3 CVS only)

Return all or part of a query result

mixed **sesam_fetch_result** (string result_id [, int max_rows]) \linebreak

Returns a mixed array with the query result entries, optionally limited to a maximum of *`max_rows`* rows. Note that both row and column indexes are zero-based.

Táblázat 1. Mixed result set returned by sesam_fetch_result()

Array Element	Contents
int $arr["count"]	number of columns in result set (or zero if this was an "immediate" query)
int $arr["rows"]	number of rows in result set (between zero and *`max_rows`*)

Array Element	Contents
bool $arr["truncated"]	TRUE if the number of rows was at least *max_rows*, FALSE otherwise. Note that even when this is TRUE, the next **sesam_fetch_result()** call may return zero rows because there are no more result entries.
mixed $arr[col][row]	result data for all the fields at row(row) and column(col), (where the integer index row is between 0 and $arr["rows"]-1, and col is between 0 and $arr["count"]-1). Fields may be empty, so you must check for the existence of a field by using the php isset() function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Note that the amount of memory used up by a large query may be gigantic. Use the *max_rows* parameter to limit the maximum number of rows returned, unless you are absolutely sure that your result will not use up all available memory.

See also: sesam_fetch_row(), and sesam_field_array() to check for "multiple fields". See the description of the sesam_query() function for a complete example using **sesam_fetch_result()**.

sesam_fetch_row (PHP 3 CVS only)

Fetch one row as an array

array **sesam_fetch_row** (string result_id [, int whence [, int offset]]) \linebreak

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

The number of columns in the result set is returned in an associative array element $array["count"]. Because some of the result columns may be empty, the count() function can not be used on the result row returned by **sesam_fetch_row()**.

result_id is a valid result id returned by sesam_query() (select type queries only!).

whence is an optional parameter for a fetch operation on "scrollable" cursors, which can be set to the following predefined constants:

Táblázat 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially (after fetch, the internal default is set to SESAM_SEEK_NEXT)
1	SESAM_SEEK_PRIOR	read sequentially backwards (after fetch, the internal default is set to SESAM_SEEK_PRIOR)

Value	Constant	Meaning
2	SESAM_SEEK_FIRST	rewind to first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	seek to last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	seek to absolute row number given as *offset* (Zero-based. After fetch, the internal default is set to SESAM_SEEK_ABSOLUTE, and the internal offset value is auto-incremented)
5	SESAM_SEEK_RELATIVE	seek relative to current scroll position, where *offset* can be a positive or negative offset value.

This parameter is only valid for "scrollable" cursors.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. If the *whence* parameter is omitted, the global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT, and settable by sesam_seek_row()) are used. If *whence* is supplied, its value replaces the global default.

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE. This parameter is only valid for "scrollable" cursors.

sesam_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array (indexed by values between 0 and $array["count"]-1). Fields may be empty, so you must check for the existence of a field by using the php isset() function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Subsequent calls to **sesam_fetch_row()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or FALSE if there are no more rows.

Példa 1. SESAM fetch rows

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table in backward order
print "<TABLE BORDER>\n";
$row = sesam_fetch_row ($result, SESAM_SEEK_LAST);
while (is_array ($row)) {
    print " <TR>\n";
    for ($col = 0; $col < $row["count"]; ++$col) {
        print "  <TD>".htmlspecialchars ($row[$col])."</TD>\n";
```

```
    }
    print " </TR>\n";
    // use implied SESAM_SEEK_PRIOR
    $row = sesam_fetch_row ($result);
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: sesam_fetch_array() which returns an associative array, and sesam_fetch_result() which returns many rows per invocation.

sesam_field_array (PHP 3 CVS only)

Return meta information about individual columns in a result

array **sesam_field_array** (string result_id) \linebreak

result_id is a valid result id returned by sesam_query().

Returns a mixed associative/indexed array with meta information (column name, type, precision, ...) about individual columns of the result after the query associated with *result_id*.

Táblázat 1. Mixed result set returned by sesam_field_array()

Array Element	Contents
int $arr["count"]	Total number of columns in result set (or zero if this was an "immediate" query). SESAM "multiple fields" are "inlined" and treated like the respective number of columns.
string $arr[col]["name"]	column name for column(col), where col is between 0 and $arr["count"]-1. The returned value can be the empty string (for dynamically computed columns). SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same column name.
string $arr[col]["count"]	The "count" attribute describes the repetition factor when the column has been declared as a "multiple field". Usually, the "count" attribute is 1. The first column of a "multiple field" column however contains the number of repetitions (the second and following column of the "multiple field" contain a "count" attribute of 1). This can be used to detect "multiple fields" in the result set. See the example shown in the sesam_query() description for a sample use of the "count" attribute.

Array Element	Contents
string $arr[col]["type"]	php variable type of the data for column(`col`), where `col` is between 0 and `$arr["count"]-1`. The returned value can be one of • integer • float • string depending on the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same php type.
string $arr[col]["sqltype"]	SQL variable type of the column data for column(`col`), where `col` is between 0 and `$arr["count"]-1`. The returned value can be one of • "CHARACTER" • "VARCHAR" • "NUMERIC" • "DECIMAL" • "INTEGER" • "SMALLINT" • "FLOAT" • "REAL" • "DOUBLE" • "DATE" • "TIME" • "TIMESTAMP" describing the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same SQL type.
string $arr[col]["length"]	The SQL "length" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "length" attribute is used with "CHARACTER" and "VARCHAR" SQL types to specify the (maximum) length of the string variable. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same length attribute.

Array Element	Contents
string $arr[col]["precision"]	The "precision" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "precision" attribute is used with numeric and time data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same precision attribute.
string $arr[col]["scale"]	The "scale" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "scale" attribute is used with numeric data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same scale attribute.

See the sesam_query() function for an example of the **sesam_field_array()** use.

sesam_field_name (PHP 3 CVS only)

Return one column name of the result set

int **sesam_field_name** (string result_id, int index) \linebreak

Returns the name of a field (i.e., the column name) in the result set, or FALSE on error.

For "immediate" queries, or for dynamic columns, an empty string is returned.

Megjegyzés: The column index is zero-based, not one-based as in SESAM.

See also: sesam_field_array(). It provides an easier interface to access the column names and types, and allows for detection of "multiple fields".

sesam_free_result (PHP 3 CVS only)

Releases resources for the query

int **sesam_free_result** (string result_id) \linebreak

Releases resources for the query associated with *`result_id`*. Returns FALSE on error.

sesam_num_fields (PHP 3 CVS only)

Return the number of fields/columns in a result set

int **sesam_num_fields** (string result_id) \linebreak

After calling sesam_query() with a "select type" query, this function gives you the number of columns in the result. Returns an integer describing the total number of columns (aka. fields) in the current *`result_id`* result set or `FALSE` on error.

For "immediate" statements, the value zero is returned. The SESAM "multiple field" columns count as their respective dimension, i.e., a three-column "multiple field" counts as three columns.

See also: sesam_query() and sesam_field_array() for a way to distinguish between "multiple field" columns and regular columns.

sesam_query (PHP 3 CVS only)

Perform a SESAM SQL query and prepare the result

string **sesam_query** (string query [, bool scrollable]) \linebreak

Returns: A SESAM "result identifier" on success, or `FALSE` on error.

A "result_id" resource is used by other functions to retrieve the query results.

sesam_query() sends a query to the currently active database on the server. It can execute both "immediate" SQL statements and "select type" queries. If an "immediate" statement is executed, then no cursor is allocated, and any subsequent sesam_fetch_row() or sesam_fetch_result() call will return an empty result (zero columns, indicating end-of-result). For "select type" statements, a result descriptor and a (scrollable or sequential, depending on the optional boolean *`scrollable`* parameter) cursor will be allocated. If *`scrollable`* is omitted, the cursor will be sequential.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: `SESAM_SEEK_NEXT`) and the scrolling offset which can either be set once by sesam_seek_row() or each time when fetching a row using sesam_fetch_row().

For "immediate" statements, the number of affected rows is saved for retrieval by the sesam_affected_rows() function.

See also: sesam_fetch_row() and sesam_fetch_result().

Példa 1. Show all rows of the "phone" table as a html table

```
<?php
if (!sesam_connect ("phonedb", "demo", "otto"))
    die ("cannot connect");
$result = sesam_query ("select * from phone");
if (!$result) {
    $err = sesam_diagnostic();
    die ($err["errmsg"]);
}
echo "<TABLE BORDER>\n";
// Add title header with column names above the result:
if ($cols = sesam_field_array ($result)) {
    echo " <TR><TH COLSPAN=".$cols["count"].">Result:</TH></TR>\n";
    echo " <TR>\n";
    for ($col = 0; $col < $cols["count"]; ++$col) {
        $colattr = $cols[$col];
```

```
            /* Span the table head over SESAM's "Multiple Fields": */
            if ($colattr["count"] > 1) {
                echo "  <TH COLSPAN=".$colattr["count"].">".$colattr["name"].
                    "(1..".$colattr["count"].")</TH>\n";
                $col += $colattr["count"] - 1;
            } else
                echo "  <TH>" . $colattr["name"] . "</TH>\n";
        }
        echo " </TR>\n";
    }

    do {
        // Fetch the result in chunks of 100 rows max.
        $ok = sesam_fetch_result ($result, 100);
        for ($row=0; $row < $ok["rows"]; ++$row) {
            echo " <TR>\n";
            for ($col = 0; $col < $ok["cols"]; ++$col) {
                if (isset($ok[$col][$row]))
                    echo "  <TD>" . $ok[$col][$row] . "</TD>\n";
                } else {
                    echo "  <TD>-empty-</TD>\n";
                }
            }
            echo " </TR>\n";
        }
    }
    while ($ok["truncated"]) { // while there may be more data
        echo "</TABLE>\n";
    }
    // free result id
    sesam_free_result($result);
    ?>
```

sesam_rollback (PHP 3 CVS only)

Discard any pending updates to the SESAM database

bool **sesam_rollback** (void) \linebreak

Returns: TRUE on success, FALSE on errors

sesam_rollback() discards any pending updates to the database. Also affected are result cursors and result descriptors.

At the end of each script, and as part of the sesam_disconnect() function, an implied **sesam_rollback()** is executed, discarding any pending changes to the database.

See also: sesam_commit().

Példa 1. Discarding an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)")
        && sesam_execimm ("INSERT INTO othertable VALUES (*, 'Another Test', 1)"))
        sesam_commit();
    else
        sesam_rollback();
}
?>
```

sesam_seek_row (PHP 3 CVS only)

Set scrollable cursor mode for subsequent fetches

bool **sesam_seek_row** (string result_id, int whence [, int offset]) \linebreak

result_id is a valid result id (select type queries only, and only if a "scrollable" cursor was requested when calling sesam_query()).

whence sets the global default value for the scrolling type, it specifies the scroll type to use in subsequent fetch operations on "scrollable" cursors, which can be set to the following predefined constants:

Táblázat 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially
1	SESAM_SEEK_PRIOR	read sequentially backwards
2	SESAM_SEEK_FIRST	fetch first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	fetch last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	fetch absolute row number given as *offset* (Zero-based. After fetch, the default is set to SESAM_SEEK_ABSOLUTE, and the offset value is auto-incremented)

Value	Constant	Meaning
5	SESAM_SEEK_RELATIVE	fetch relative to current scroll position, where *offset* can be a positive or negative offset value (this also sets the default "offset" value for subsequent fetches).

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE.

sesam_settransaction (PHP 3 CVS only)

Set SESAM transaction parameters

bool **sesam_settransaction** (int isolation_level, int read_only) \linebreak

Returns: TRUE if the values are valid, and the **settransaction()** operation was successful, FALSE otherwise.

sesam_settransaction() overrides the default values for the "isolation level" and "read-only" transaction parameters (which are set in the SESAM configuration file), in order to optimize subsequent queries and guarantee database consistency. The overridden values are used for the next transaction only.

sesam_settransaction() can only be called before starting a transaction, not after the transaction has been started already.

To simplify the use in php scripts, the following constants have been predefined in php (see SESAM handbook for detailed explanation of the semantics):

Táblázat 1. Valid values for *"Isolation_Level"* parameter

Value	Constant	Meaning
1	SESAM_TXISOL_READ_UNCOMMITTED	Read Uncommitted
2	SESAM_TXISOL_READ_COMMITTED	Read Committed
3	SESAM_TXISOL_REPEATABLE_READ	Repeatable Read
4	SESAM_TXISOL_SERIALIZABLE	Serializable

Táblázat 2. Valid values for *"Read_Only"* parameter

Value	Constant	Meaning
0	SESAM_TXREAD_READWRITE	Read/Write
1	SESAM_TXREAD_READONLY	Read-Only

The values set by **sesam_settransaction()** will override the default setting specified in the SESAM configuration file.

Példa 1. Setting SESAM transaction parameters

```
<?php
sesam_settransaction (SESAM_TXISOL_REPEATABLE_READ,
                      SESAM_TXREAD_READONLY);
?>
```

XCII. Munkamenet kezelő függvények

A PHP munkamenet kezelése lehetővé teszi adatok megőrzését az egymást követő oldal lekérdezések között. Ez képessé tesz még testreszabhatóbb oldalak készítésére.

Ha ismered a PHPLIB munkamenet kezelését, sok hasonlóságot fogsz felfedezni a PHP munkamenet kezelő függvényeit illetően.

Minden webhelyedre látogató egy-egy egyedi azonosítót kap, az úgynevezett munkamenet azonosítót (session azonosítót). Ez vagy egy sütiben (cookie) tárolódik a látogató gépén, vagy az URL-ben közlekedik oldalról oldalra.

A munkamenet támogatás lehetővé teszi tetszőleges számú változó megőrzését a PHP oldal lekérdezések között. Ha egy látogató érkezik webhelyedre, háromféleképpen kezdődhet el, vagy folytatódhat a munkamenete. Mindhárom esetben a PHP egy munkamenet azonosító érkezését várja. Ha nem érkezik, új munkamenetet indít.

Ha a session.auto_start be van kapcsolva, a PHP automatikusan megkezdni / folytatja a munkamenetet. Ha ez nincs bekapcsolva, akkor közvetlenül a session_start() függvénnyel, vagy közvetve a session_register() függvénnyel tudod a munkamenetet folytatni / megkezdeni. Ha egy érvényes azonosító érkezett, a korábban beállított munkamenet környezet visszaállításra kerül.

Minden a munkamenethez rendelt változó szerializálódik a PHP oldal futásának befejeződésekor. A nem definiált, de munkamenethez rendelt változók a későbbi folytatásokban nem jönnek újból létre.

A `track_vars` és `register_globals` ini beállítások befolyásolják a munkamenet változók tárolásának és visszaállításának módját.

> **Megjegyzés:** A PHP 4.0.3 verziótól kezdve a `track_vars` mindig be van kapcsolva, kikapcsolására nincs mód.

> **Megjegyzés:** A 4.1.0 verziótól kezdve a `$_SESSION` szuperglobális változó rendelkezésre áll, csakúgy mint a `$_POST`, `$_GET`, `$_REQUEST` és így tovább. A `$HTTP_SESSION_VARS` változótól eltérően a `$_SESSION` mindig elérhető (szuperglobális). Ezért a `global` kulcsszóval nem szabad együtt használni a `$_SESSION` változót.

Ha a `track_vars` be van kapcsolva, és a `register_globals` ki van kapcsolva, csak a `$HTTP_SESSION_VARS` asszociatív tömb elemei a munkamenet változók. A visszaállított munkamenet változók is csak a `$HTTP_SESSION_VARS` tömbben lesznek megtalálhatóak.

Példa 1. Változó bejegyzése a munkamenetbe a `track_vars` bekapcsolt állapotában.

```
<?php
if (isset($HTTP_SESSION_VARS[’szamlalo’])) {
   $HTTP_SESSION_VARS[’szamlalo’]++;
}
else {
   $HTTP_SESSION_VARS[’szamlalo’] = 0;
}
?>
```

A `$_SESSION` változó használata (vagy a `$HTTP_SESSION_VARS` változó használata PHP 4.0.6 vagy régebbi verzió esetén) javasolt biztonsági és olvashatósági szempontok miatt. A `$_SESSION` vagy `$HTTP_SESSION_VARS` használatával nincs szükség a session_register()/session_unregister()/session_is_registered() függvényekre. A programozók éppen úgy dolgozhatnak a munkamenet változókkal, mint a "normális" változókkal.

Példa 2. Változó bejegyzése a munkamenetbe a $_SESSION tömbbel.

```
<?php
// Használd a $HTTP_SESSION_VARS tömböt PHP 4.0.6 vagy régebbi verzióban
if (!isset($_SESSION['szamlalo'])) {
    $_SESSION['szamlalo'] = 0;
} else {
    $_SESSION['szamlalo']++;
}
?>
```

Példa 3. Változó bejegyzés törlése a $_SESSION tömbbel.

```
<?php
// Használd a $HTTP_SESSION_VARS tömböt PHP 4.0.6 vagy régebbi verzióban
unset($_SESSION['szamlalo']);

?>
```

Ha a `register_globals` be van kapcsolva, akkor minden globális változó lehet munkamenet változóként regisztrálva, és a munkamenet későbbi folytatásaiban is létre fognak jönni globális változókként a bejegyzett változók. Mivel ebben az esetben a PHP-nek tudnia kell, hogy mely globális változók bejegyzett munkamenet változók is egyben, a programozónak a session_register() függvényt kell használnia. Emlékeztetőként a `$HTTP_SESSION_VARS`/`$_SESSION` tömbök

használatakor nincs szükség a session_register() függvényre.

> **Figyelem**
>
> Ha a `$HTTP_SESSION_VARS`/`$_SESSION` tömböt használod és kikapcsolod a `register_globals` beállítást, ne használd a session_register(), session_is_registered() és session_unregister() függvényeket.
>
> Ha bekapcsolod a `register_globals` beállítást, a session_unregister() függvényt kell használnod, mivel a munkamenet változók globális változókban jelennek meg. A `register_globals` kikapcsolása javasolt mind biztonsági, mind teljesítmény szempontokból.

Példa 4. Változó bejegyzése a `register_globals` bekapcsolt állapota mellett.

```
<?php
if (!session_is_registered(’szamlalo’)) {
    session_register("szamlalo");
    $szamlalo = 0;
}
else {
    $szamlalo++;
}
?>
```

Ha mind a `track_vars`, mind a `register_globals` beállítások be vannak kapcsolva, a globális változók és a `$HTTP_SESSION_VARS`/`$_SESSION` tömbök elemei ugynazokra az értékekre fognak mutatni a már bejegyzett változókat illetően.

Ha a programozó a session_register() függvényt használja munkamenet változók bejegyzésére, akkor ebben az esetben a `$HTTP_SESSION_VARS`/`$_SESSION` tömbökben nem fog megjelenni ez a változó a következő munkamenet folytatása előtt (azaz a következő PHP program futás előtt).

Kétféleképpen "közlekedhet" a munkamenet azonosító:

- Süti formájában
- URL / űrlap paraméterben

A munkamenetkezelő modul mindkét formát támogatja. A sütik optimálisak, de sajnos nem megbízhatóak (sokan nem fogadják a sütiket), ezért nem lehet rájuk biztonsággal építeni. A második módszer az azonosítót közvetlenül az URL-ekbe és űrlapokba helyezi.

A PHP képes ezutóbbi azonosító terjesztést átlátszóan végezni, ha az `--enable-trans-sid` opcióval fodítottad. Ha ezt az opciót bekapcsolod, a reltív URL-ekhez és űrlapokhoz automatikusan hozzáteszi a PHP a munkamenet azonosítót. Alternatív módszerként használhatod a `SID` konstanst, ami definiált, ha a kliens nem küldte vissza a megfelelő sütit. A `SID` vagy

`session_name=session_id` (munkamenet név, munkamenet azonosító) formátumú, vagy egy üres karektersorozat.

A következő példa egy változó bejegyzést, és egy következő oldalra mutató link készítését mutatja,

Példa 5. Egy látogató által lehívott oldalak számlálása

```
<?php
if (!session_is_registered('szamlalo')) {
    session_register('szamlalo');
    $szamlalo = 1;
}
else {
    $szamlalo++;
}
?>

Üdv látogató, eddig <?php echo $szamlalo; ?> alkalommal
láttad ezt az oldalt.<p>

<?php
# Az <?php echo SID?> szükséges, hogy a munkamenet azonosítót
# megőrizzük abban az esetben is, ha a látogató nem fogadja
# a sütit. (<?=SID?> használható, ha a rövid tagek engedélyezettek)
?>

A továbblépéshez <A HREF="kovoldal.php?<?php echo SID?>">kattints ide</A>.
```

A `<?=SID?>` nem szükséges, ha az `--enable-trans-sid` opcióval fordítottad a PHP-t.

> **Megjegyzés:** A nem relatív hivatkozások feltételezetten külső oldalakra mutatnak, ezért nem egészülnek ki automatikusan a SID-el. Komoly biztonsági problémákat okozna, ha a SID-ek elkerülnének egy másik kiszolgálóhoz.

A munkamenethez tartozó adatok állományokban tárolódnak. Ha adatbázisban, vagy más tároló eszközön szeretnéd rögzíteni a munkamenethez tartozó adatokat, a session_set_save_handler() függvényt kell használnod a saját tároló függvényeid bejegyzéséhez.

A munkamenet kezelő függvények számos beállítási lehetőséget biztosítanak, amiket a `php.ini` állományban helyezhetsz el. Egy rövid áttekintés következik.

- A `session.save_handler` beállítás adja meg a használt tárolás kezelőt, ami a munkamenet adatok elmentésére, és visszakeresésére szolgál. Az alapértéke `files`.
- A `session.save_path` beállítás az a paraméter, amit a tárolás kezelő megkap. Ha az alapértelmezett állomány kezelőt használod, ez a munkamenetek tárolására használt ideiglenes

állományok könytárának elérési útja. Alapértelmezése `/tmp`. Ha a `session.save_path` mélysége több mint kettő, az automatikus szemét takarítás nem fog működni.

> **Figyelem**
>
> Ha ezt az értéket egy mindenki által olvasható könyvtárra állítod (mint az alapbeállítású `/tmp`), akkor a kiszolgálón lévő többi felhasználó el tudja kapni a munkameneteidet a könyvtárban lévő állományok neve és tartalma alapján.

- A `session.name` a munkamenet neve, amit a PHP a süti beállításakor, az URL / űrlap paraméterek felvételekor és a munkamenet folytatásakor használ. Csak alfanumerikus karaktereket tartalmazhat. Alapértéke `PHPSESSID`.
- A `session.auto_start` beállítás adja meg, hogy a PHP kezdjen-e / folytasson-e minden PHP oldal futattásnál automatikusan munkamenetet. Alapértéke `0` (kikapcsolt).
- A `session.cookie_lifetime` a beállított süti élettartalmát szabályozza másodpercekben megadva. A nulla érték azt jelenti, hogy "amíg a böngészőt be nem zárják". Alapértéke éppen `0`.
- A `session.serialize_handler` a szerializáláshoz használt kezelő nevét adja meg. Jelenleg egy beépített PHP formátum (`php` néven) és egy WDDX formátum támogatott (`wddx` néven). A WDDX formátumhoz szükséges, hogy a WDDX támogatás a PHP-be legyen fordítva. Lásd WDDX. Alapértéke a beépített `php` kezelő.
- A `session.gc_probability` megadja, hogy mekkora valószínűséggel indul el a gc (garbage collection - szemétgyűjtés) az egyes PHP lekérdezésekkor. Az értéket százalékban kell megadni. Alapértéke `1`.
- A `session.gc_maxlifetime` megadja, hogy hány másodperc elteltével tekinti a szemétgyűtjtő szemétnek, és eltakarítandónak a munkamenet adatokat.
- A `session.referer_check` azt a rész-karaktersorozatot tartalmazza, amit minden HTTP Referer fejlécben keresni szeretnél a munkamenetek még biztonságosabb azonosítása céljából. Ha a megadott rész-karaktersorozatot nem találja meg a PHP a Refererben, a küldött munkamenet azonosítót nem fogja elfogadni. Alapértéke az üres karaktersorozat.
- A `session.entropy_file` egy elérési utat határoz meg ahhoz az "állományhoz", amit a PHP külső entrópia forrásnak használ a munkamenet azonosító előállításakor. Ez lehet például `/dev/random` vagy `/dev/urandom`. Ezek sok Unix rendszeren elérhetőek.
- A `session.entropy_length` az olvasott byte-ok számát adja meg, amennyit a fent említett állományból olvas a PHP. Alapértéke `0` (kikapcsolt).
- A `session.use_cookies` határozza meg, hogy a PHP használ-e sütiket a munkamenet azonosító tárolására a kliens oldalon. Alapértéke `1` (engedélyezett).
- A `session.cookie_path` azt az elérési utat határozza meg, amit a munkamenet sütiben beállít a PHP. Alapértéke `/`.
- A `session.cookie_domain` azt a domain értéket határozza meg, amit a PHP a munkamenet sütiben beállít. Alapértéke semmi.
- A `session.cache_limiter` azt a cache metódust határozza meg, amit a munkamenet oldalakra alkalmazni kell. Lehetséges értékei: none, nocache, private, private_no_expire és public.

Alapértéke `nocache`.

- A `session.cache_expire` megadja, hogy hány percig legyen aktuális a cache-ben egy munkamenetet használó oldal. Ennek nincs hatása, ha nocache módot választasz. Alapértéke `180`.
- A `session.use_trans_sid` szabályozza, hogy az átlátszó SID hozzáadás be van-e kapcsolva vagy nincs. Ehhez szükséges, hogy a `--enable-trans-sid` opcióval fordítsd korábban a PHP-t. Alapértéke `1` (engedélyezett).
- Az `url_rewriter.tags` adja meg, hogy mely HTML elemeket kell átírni, ha a munkamenet azonosítók átlátszó elhelyezése be van kapcsolva. Alapértéke `a=href,area=href,frame=src,input=src,form=fakeentry`

Megjegyzés: A munkamenetek támogatása a PHP 4.0 verzióban jelent meg.

session_cache_expire (PHP 4 >= 4.2.0)

Az aktuális cache lejárat lekérdezése

int **session_cache_expire** ([int new_cache_expire]) \linebreak

A **session_cache_expire()** visszatér az aktuális cache lejárati idővel. Ha a *`new_cache_expire`* paramétert is megadod, az érték a *`new_cache_expire`*-ben megadottra változik.

session_cache_limiter (PHP 4 >= 4.0.3)

Az aktuális munkamenet cache korlátozás lekérdezése / beállítása

string **session_cache_limiter** ([string cache_limiter]) \linebreak

A **session_cache_limiter()** visszatér az aktuális munkamenet cache korlátozás értékével. Ha a *`cache_limiter`* paraméter is meg van adva, a cache korlátozás a megadott értékre változik.

A cache korlátozás szabályozza a cache befolyásoló HTTP fejléceket, amiket a PHP a böngészőnek küld. Ezek a fejlécek szabályozzák, hogy a munkameneteket használó oldalak hogyan jelenjenek meg a gyorsítótárakban. Ha a `nocache` értéket állítod be, minden kliens oldali cache-elést letiltasz. A `public` értékkel éppen ellenkezőleg engedélyezed a cache-elést. Ezen kívül még a `private` értéket is meg tudod adni, ami valamivel korlátozottabb, mint a `public`.

`private` módban az Expire fejléc, amit a böngésző kap gondot okozhat a feldolgozásban néhány böngészőnél, mint pl. a Mozilla. Ezt a problémát a `private_no_expire` mód bekapcsolásával tudod orvosolni. Az Expire fejlécet ebben a módban nem küldi el a PHP a kliensnek.

Megjegyzés: A `private_no_expire` a 4.2.0dev verzióban került a lehetőségek közé.

A korlátozó érték a `php.ini` `session.cache_limiter` beállítására áll vissza a PHP program kérések végrehajtásakor. Ezért minden programban, ahol az alapértelmezett korlátozás nem megfelelő, meg kell hívnod a **session_cache_limiter()** függvényt a session_start() hívása előtt.

Példa 1. session_cache_limiter() példák

```
<?php

# A korlátozó ’private’ értékre állítása

session_cache_limiter(’private’);
$cache_limiter = session_cache_limiter();

echo "A cache korlátozás értéke $cache_limiter<p>";
?>
```

session_decode (PHP 4)

Visszakódolja a munkamenet adatokat egy karaktersorozatból

bool **session_decode** (string data) \linebreak

A **session_decode()** visszakódolja a munkamenet adatokat a *data* karaktersorozatból, beállítva a munkamenetben lévő változókat.

session_destroy (PHP 4)

Adott munkamenethez tartozó minden adat törlése

bool **session_destroy** (void) \linebreak

A **session_destroy()** az aktuális munkamenethez tartozó minden adatot töröl. Nem törli azonban a globális változókat, vagy a munkamenethez beállított sütit.

Ez a függvény TRUE értékkel tér vissza siker esetén, FALSE értékkel tér vissza, ha nem tudja a munkamenet adatait törölni.

Példa 1. Munkamenet törlése

```
<?php

// Munkamenet megkezdése / folytatása.
// Ha használsz session_name("valami") hívást, ne feledd itt!
session_start();
// Minden munkamenet változó törlése.
session_unset();
// A munkamenet törlése.
session_destroy();

?>
```

Példa 2. Munkamenet törlése a $_SESSION tömbbel

```
<?php

// Munkamenet megkezdése / folytatása.
// Ha használsz session_name("valami") hívást, ne feledd itt!
session_start();
// Minden munkamenet változó törlése.
$_SESSION = array();
// A munkamenet törlése.
session_destroy();
```

```
?>
```

session_encode (PHP 4)

Elkódolja a munkamenet adatokat egy karaktersorozatba

string **session_encode** (void) \linebreak

A **session_encode()** visszatér az aktuális munkamenet adatainak egy karaktersorozatba kódolt változatával.

session_get_cookie_params (PHP 4)

Lekérdezi a munkamenet süti paramétereit

array **session_get_cookie_params** (void) \linebreak

A **session_get_cookie_params()** visszaadja a munkamanet süti paramétereit egy asszociatív tömbben. Ez a tömb a következő elemeket tartalmazza:

- "lifetime" - A süti élettartalma.
- "path" - A süti elérési út korlátozása.
- "domain" - A süti domain név korlátozása.
- "secure" - A süti csak biztonságos kapcsolaton keresztül közlekedhet. (Ez a PHP 4.0.4 verzió óta lehetséges.)

session_id (PHP 4)

Az aktuális munkamenet azonosító lekérdezése / beállítása

string **session_id** ([string id]) \linebreak

A **session_id()** visszatér az aktuális munkamenet azonosítójával. Ha az *id* paramétert is megadod, akkor az az érték lesz az új munkamenet azonosító.

A SID konstans is használható az aktuális munkemenet név és azonosító lekérdezésére, ha URL-be szeretnéd tenni az azonosítót.

session_is_registered (PHP 4)

Egy változó bejegyzettségét ellenőrzi

bool **session_is_registered** (string name) \linebreak

A **session_is_registered()** `TRUE` értékkel tér vissza, ha a *`name`* nevű változó be van jegyezve az aktuális munkamenet változójaként.

Megjegyzés: Ha a `$_SESSION` (vagy `$HTTP_SESSION_VARS` PHP 4.0.6 vagy korábbi verziókban) tömböket használod, az isset() függvény hívásával derítheted ki, hogy a változó be van-e jegyezve a munkamenetbe.

Figyelem

A `$HTTP_SESSION_VARS` vagy `$_SESSION` tömbök használata esetén ne hívd meg a session_register(), **session_is_registered()** vagy session_unregister() függvényeket.

session_module_name (PHP 4)

Az aktuális munkamenet modul lekérdezése / beállítása

string **session_module_name** ([string module]) \linebreak

A **session_module_name()** visszatér az aktuális munkamenet modul nevével. Ha a *`module`* paramétert is megadod, a modult a megadottra állítja át.

session_name (PHP 4)

Az aktuális munkamenet név lekérdezése / beállítása

string **session_name** ([string name]) \linebreak

A **session_name()** visszaadja az aktuális munkamenet nevét. Ha a *`name`* paraméter meg van adva, az aktuális munkamenet neve erre változik.

A munkamenet neve azonosítja a munkamenetet a sütikben, URL-ekben és űrlapokban. Csak alfanumerikus karaktereket tartalmazhat, rövidnek és érthetőnek kell lennie (azon látogatók számára, akik például bekapcsolt süti figyelmeztetők olvasása alapján döntik el a süti engedélyezését). A munkamenet neve visszatér a `php.ini` `session.name` beállítására a PHP program futása kezdetén. Ezért mindig meg kell hívni a **session_name()** függvényt a kívánt névvel a session_start() és/vagy session_register() hívások előtt, ha nem az alapértelmezett munkamenet nevet szeretnéd használni.

Példa 1. session_name() példa

```
<?php

// a munkamenet nevét webhelyneve-re állíjtuk

$elozo_nev = session_name("webhelyneve");

echo "A megelőző munkamenet név $elozo_nev volt.<p>";
?>
```

session_readonly (unknown)

Begin session - reinitializes freezed variables, but no writeback on request end

void **session_readonly** (void) \linebreak

Read in session data without locking the session data. Changing session data is not possible, but frameset performance will be improved.

session_register (PHP 4)

Egy vagy több változó bejegyzése munkamenet változókként

bool **session_register** (mixed name [, mixed ...]) \linebreak

A **session_register()** függvényt tetszőleges számú paraméterrel hívhatod. Bármely paraméter lehet egy karaktersorozat, ami egy változó neve, vagy egy tömb, ami változóneveket vagy újabb tömböket tartalmaz. A **session_register()** minden megadott nevű globális változót bejegyzi az aktuális munkamenet változójaként.

Figyelem

Ez a függvény *globális* változók regisztrálására használható. Ha egy függvényen belül szeretnél munkamenet változót bejegyezni, előbb globálissá kell tenned a **global()** kulcsszó használatával, vagy a munkamenet tömböket kell használnod, ahogy lentebb leírjuk.

Figyelem

Ha a `$HTTP_SESSION_VARS` vagy `$_SESSION` tömböket használod, ne használd a **session_register()**, session_is_registered() és session_unregister() függvényeket változók bejegyzésére!

Ez a függvény `TRUE` értékkel tér vissza, ha minden megadott nevű változó sikeresen bejegyzésre került a munkamenetben.

Ha a session_start() függvényt nem hívtad meg ennek a függvénynek a hívása előtt, a **session_register()** meghívásával a PHP közvetve automatikusan meghívja a session_start() függvényt paraméterek nélkül.

Egyszerűbben jegyezheted be a munkamenet változókat, ha a `$HTTP_SESSION_VARS` vagy `$_SESSION` (PHP >= 4.1.0) asszociatív tömbök egyikében hozol létre egy új kulcsot és értéket.

```
$barney = "Egy bíborszínű dinoszaurusz.";
session_register("barney");

$HTTP_SESSION_VARS["zim"] = "Támadó egy másik bolygóról.";

# A szuperglobális $_SESSION tömb a PHP 4.1.0 verzió óta használható
$_SESSION["spongebob"] = "Szögletes nadrágja van.";
```

> **Megjegyzés:** Jelenleg nem lehetséges erőforrások bejegyzése munkamenetekben. Nem lehetséges például egy adatbázis kapcsolat létrehozása, regisztrálása a munkamenetben, és a következő munkamenet folytatáskor az adatbázis kapcsolat használata. Az erőforrásokkal visszatérő függvények arról ismerhetőek fel, hogy a leírásukban `resource` a megadott visszatérési típus. Az erőforrásokkal visszatérő függvények listája megtalálható az erőforrás típusok függelékben.
>
> A `$_SESSION` (vagy `$HTTP_SESSION_VARS` PHP 4.0.6 vagy korábbi verziókban) használatakor új asszociatív indexet kell létrehozni a `$_SESSION` tömbben. Például $_SESSION['valtozo'] = 'ABC';

Lásd még session_is_registered() és session_unregister().

session_save_path (PHP 4)

Az aktuális munkamenet mentési könyvtár lekérdezése / beállítása

string **session_save_path** ([string path]) \linebreak

A **session_save_path()** visszatér az aktuális munkamenet mentési könyvtárral, ahol a PHP a munkamenet adatokat tárolja. Ha a *`path`* paraméter meg van adva, a mentési könyvtár megváltozik.

> **Megjegyzés:** Néhány operációs renszeren lehetőség van arra, hogy olyan elérési utat adjál meg, ami egy kis állományokat hatékonyabban kezelő állományrendszeren van. Linuxon például a reiserfs állományrendszer jobb teljesítménnyel kezeli a PHP munkamenet állományait, mint az ext2fs.

session_set_cookie_params (PHP 4)

A munkamenet süti paramétereit állítja be

void **session_set_cookie_params** (int lifetime [, string path [, string domain [, bool secure]]]) \linebreak

A `php.ini` állományban definiált süti paraméterek felülírása. Ezen függvény hatása csak az aktuális program futásának végéig tart.

Megjegyzés: A *`secure`* paraméter a PHP 4.0.4 verziója óta használható.

session_set_save_handler (PHP 4)

Felhasználói szintű munkamenet tároló függvényeket állít be

void **session_set_save_handler** (string open, string close, string read, string write, string destroy, string gc) \linebreak

A **session_set_save_handler()** beállítja a programozó által PHP-ben megírt munkamenet tároló függvényeket. Ezeket használja majd a PHP a munkamenet adatainak tárolására és lekérdezésére. Ez a lehetőség akkor hasznos, ha a PHP által kínált módszerek nem megfelelőek. Másfajta tárolás lehetséges az adatok lokális adatbázisban történő rögzítésével.

Megjegyzés: A `php.ini` *`session.save_handler`* beállítását mindenképpen *`user`* értékre kell állítanod, hogy a **session_set_save_handler()** függvény hívásának legyen hatása.

Megjegyzés: Az írás kezelő csak a PHP program kimenetének elküldése után hívódik meg. Ezért az írás kezelőben a böngésző számára kiírt, tipikusan hibakeresést szolgáló adatok nem lesznek sohasem láthatóak a böngészőben. Ha a hibakereséshez adatok kiírása szükséges, javasolt napló állomány használata erre a célra.

Az alább látható példa állmoány alapú munkamenet adat tárolást valósít meg, hasonlóan a PHP alapértelmezésű *`files`* kezelőjéhez. Ez a példa könnyen kiterjeszthető egy adatbázis tárolást alkalmazó kezelővé, felhasználva a kedvenc adatbáziskezelőd függvényeit.

Az olvasó függvénynek mindig karaktersorozattal kell visszatérnie, hogy a mentést végző kezelő helyesen működjön. Üres karaktersorozattal kell visszatérni, ha nincs adat, amit olvasni lehetne. Más kezelők visszatérési értékei logikai típusúvá alakulnak. Igaz értéket kell visszaadni siker estén, hamisat hiba esetén.

Példa 1. session_set_save_handler() példa

```
<?php
function open ($save_path, $session_name) {
  global $sess_save_path, $sess_session_name;
```

```
  $sess_save_path = $save_path;
  $sess_session_name = $session_name;
  return(true);
}

function close() {
  return(true);
}

function read ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "r")) {
    $sess_data = fread($fp, filesize($sess_file));
    return($sess_data);
  } else {
    return(""); // Itt mindenképpen "" a visszatérési érték.
  }

}

function write ($id, $sess_data) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "w")) {
    return(fwrite($fp, $sess_data));
  } else {
    return(false);
  }

}

function destroy ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  return(@unlink($sess_file));
}

/*****************************************
 * FIGYELEM - Itt mindenképpen meg kell      *
 * valósítani valamiféle szemétgyűjtést.     *
 *****************************************/
function gc ($maxlifetime) {
  return true;
}

session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");

session_start();

// Ezután hagyományosan használható a munkamenet

?>
```

session_start (PHP 4)

Munkamenet megkezdése / folytatása

bool **session_start** (void) \linebreak

A **session_start()** elkezd egy új munkamenetet, vagy folytat egy korábbit, ha a PHP érvényes munkamenet azonosító érkezését érzékelte GET/POST paraméterben vagy sütiben.

Ha az alapbeállítású névtől eltérő nevet szeretnél használni a munkamenet számára, használd a session_name() függvényt a **session_start()** meghívása *előtt*.

Ez a függvény mindig `TRUE` értékkel tér vissza.

> **Megjegyzés:** Ha süti alapú munkameneteket használsz, a **session_start()** hívásnak minden kimenetet meg kell előznie.

A **session_start()** egy belső kimenet kezelőt regisztrál az URL-ek és űrlapok átírására, ha a `trans-sid` funkció be van kapcsolva. Ha az `ob_gzhandler`-t vagy más hasonló kimenet kezelőket használsz az ob_start() függvénnyel, a kimenet kezelők sorrendje fontos a megfelelő kimenet érdekében. Ezért az `ob_gzhandler`-t a munkamenet megkezdése előtt kell regisztrálni.

> **Megjegyzés:** A `zlib.output_compression` használata javasolt az `ob_gzhandler` helyett.

session_unregister (PHP 4)

Munkamenet változó törlése a munkamenetből

bool **session_unregister** (string name) \linebreak

A **session_unregister()** törli (elfelejti) a *`name`* paraméterben megadott nevű globális munkamenet változót, az aktuális munkamenetből.

Ez a függvény `TRUE` értékkel tér vissza, ha a változó törlése sikeresen megtörtént az aktuális munkamenetet érintően.

> **Megjegyzés:** A `$_SESSION` (vagy `$HTTP_SESSION_VARS` PHP 4.0.6 vagy korábbi verziókban) használatakor az unset() függvényt kell használni munkamenet változó törlésére, paraméterként átadva a tömb változóhoz tartozó elemét.

> **Figyelem**
>
> Ez a függvény nem törli a munkamenet változóhoz tartozó globális *name* nevű változót, csak megszünteti a változó és a munkamenet kapcsolatát. Meg kell hívnod az unset() függvényt a változóra, ha törölni szeretnéd a globális változót is.

> **Figyelem**
>
> A `$HTTP_SESSION_VARS` vagy `$_SESSION` tömbök használata esetén ne hívd meg a session_register(), session_is_registered() vagy **session_unregister()** függvényeket.

session_unset (PHP 4)

Minden munkamenet változó felszabadítása

void **session_unset** (void) \linebreak

A **session_unset()** függvény minden éppen bejegyzett munkamenet változót felszabadít.

> **Megjegyzés:** Ha a `$_SESSION` (vagy `$HTTP_SESSION_VARS` PHP 4.0.6 vagy korábbi verziókban) tömböket használod, az unset() függvénnyel tudsz egyes változókat törlni. A teljes törléshez használd a $_SESSION = array(); sort.

session_write_close (PHP 4 >= 4.0.4)

A munkamenet adatok kiírása és a munkamenet lezárása

void **session_write_close** (void) \linebreak

Az aktuális munkamenet befejezése, és a munkamenet adatok tárolása.

A munkamenet adatok elmentésre kerülnek a programod futásának végeztével, anélkül, hogy meghívnád a **session_write_close()** függvényt, de mivel a munkamenet adatok zároltak az egyidőben írások elkerülésére, egyszerre csak egy program tud dolgozni a munkamenettel. Ha kereteket használsz a weboldaladon a munkamenetekkel együtt, észre fogod venni, hogy a keretek egyesével töltődnek le emiatt a zárolás miatt. A keretek betöltéséhez szükséges időt lerövidítheted, ha rögtön meghívod ezt a függvényt a programodban, amikor már biztos, hogy minden munkamenet változtatást elvégeztél.

XCIII. Shared Memory Functions

Shmop is an easy to use set of functions that allows php to read, write, create and delete UNIX shared memory segments. The functions will not work on windows, as it does not support shared memory. To use shmop you will need to compile php with the --enable-shmop parameter in your configure line.

> **Megjegyzés:** In PHP 4.0.3, there functions were prefixed by `shm` rather than `shmop`.

Példa 1. Shared Memory Operations Overview

```
<?php

// Create 100 byte shared memory block with system id if 0xff3
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if(!$shm_id) {
echo "Couldn’t create shared memory segment\n";
}

// Get shared memory block’s size
$shm_size = shmop_size($shm_id);
echo "SHM Block Size: ".$shm_size. " has been created.\n";

// Lets write a test string into shared memory
$shm_bytes_written = shmop_write($shm_id, "my shared memory block", 0);
if($shm_bytes_written != strlen("my shared memory block")) {
echo "Couldn’t write the entire length of data\n";
}

// Now lets read the string back
$my_string = shmop_read($shm_id, 0, $shm_size);
if(!$my_string) {
echo "Couldn’t read from shared memory block\n";
}
echo "The data inside shared memory was: ".$my_string."\n";

//Now lets delete the block and close the shared memory segment
if(!shmop_delete($shm_id)) {
echo "Couldn’t mark shared memory block for deletion.";
}
shmop_close($shm_id);

?>
```

shmop_close (PHP 4 >= 4.0.4)

Close shared memory block

int **shmop_close** (int shmid) \linebreak

shmop_close() is used to close a shared memory block.

shmop_close() takes the shmid, which is the shared memory block identifier created by shmop_open().

Példa 1. Closing shared memory block

```
<?php
shmop_close($shm_id);
?>
```

This example will close shared memory block identified by `$shm_id`.

shmop_delete (PHP 4 >= 4.0.4)

Delete shared memory block

int **shmop_delete** (int shmid) \linebreak

shmop_delete() is used to delete a shared memory block.

shmop_delete() takes the shmid, which is the shared memory block identifier created by shmop_open(). On success 1 is returned, on failure 0 is returned.

Példa 1. Deleting shared memory block

```
<?php
shmop_delete($shm_id);
?>
```

This example will delete shared memory block identified by `$shm_id`.

shmop_open (PHP 4 >= 4.0.4)

Create or open shared memory block

int **shmop_open** (int key, string flags, int mode, int size) \linebreak

shmop_open() can create or open a shared memory block.

shmop_open() takes 4 parameters: key, which is the system's id for the shared memory block, this parameter can be passed as a decimal or hex. The second parameter are the flags that you can use:

- "a" for access (sets SHM_RDONLY for shmat) use this flag when you need to open an existing shared memory segment for read only
- "c" for create (sets IPC_CREATE) use this flag when you need to create a new shared memory segment or if a segment with the same key exists, try to open it for read and write
- "w" for read & write access use this flag when you need to read and write to a shared memory segment, use this flag in most cases.
- "n" create a new memory segment (sets IPC_CREATE|IPC_EXCL) use this flag when you want to create a new shared memory segment but if one already exists with the same flag, fail. This is useful for security purposes, using this you can prevent race condition exploits.

The third parameter is the mode, which are the permissions that you wish to assign to your memory segment, those are the same as permission for a file. Permissions need to be passed in octal form ex. 0644. The last parameter is size of the shared memory block you wish to create in bytes.

Megjegyzés: Note: the 3rd and 4th should be entered as 0 if you are opening an existing memory segment. On success **shmop_open()** will return an id that you can use to access the shared memory segment you've created.

Példa 1. Create a new shared memory block

```
<?php
$shm_id = shmop_open(0x0fff, "c", 0644, 100);
?>
```

This example opened a shared memory block with a system id of 0x0fff.

shmop_read (PHP 4 >= 4.0.4)

Read data from shared memory block

string **shmop_read** (int shmid, int start, int count) \linebreak

shmop_read() will read a string from shared memory block.

shmop_read() takes 3 parameters: shmid, which is the shared memory block identifier created by shmop_open(), offset from which to start reading and count on the number of bytes to read.

Példa 1. Reading shared memory block

```
<?php
$shm_data = shmop_read($shm_id, 0, 50);
?>
```

This example will read 50 bytes from shared memory block and place the data inside `$shm_data`.

shmop_size (PHP 4 >= 4.0.4)

Get size of shared memory block

int **shmop_size** (int shmid) \linebreak

shmop_size() is used to get the size, in bytes of the shared memory block.

shmop_size() takes the shmid, which is the shared memory block identifier created by shmop_open(), the function will return and int, which represents the number of bytes the shared memory block occupies.

Példa 1. Getting the size of the shared memory block

```
<?php
$shm_size = shmop_size($shm_id);
?>
```

This example will put the size of shared memory block identified by `$shm_id` into `$shm_size`.

shmop_write (PHP 4 >= 4.0.4)

Write data into shared memory block

int **shmop_write** (int shmid, string data, int offset) \linebreak

shmop_write() will write a string into shared memory block.

shmop_write() takes 3 parameters: shmid, which is the shared memory block identifier created by shmop_open(), data, a string that you want to write into shared memory block and offset, which specifies where to start writing data inside the shared memory segment.

Példa 1. Writing to shared memory block

```
<?php
$shm_bytes_written = shmop_write($shm_id, $my_string, 0);
?>
```

This example will write data inside `$my_string` into shared memory block, `$shm_bytes_written` will contain the number of bytes written.

XCIV. Shockwave Flash functions

PHP offers the ability to create Shockwave Flash files via Paul Haeberli's libswf module. You can download libswf at ftp://ftp.sgi.com/sgi/graphics/grafica/flash. Once you have libswf all you need to do is to configure `--with-swf[=DIR]` where DIR is a location containing the directories include and lib. The include directory has to contain the swf.h file and the lib directory has to contain the libswf.a file. If you unpack the libswf distribution the two files will be in one directory. Consequently you will have to copy the files to the proper location manually.

Once you've successfully installed PHP with Shockwave Flash support you can then go about creating Shockwave files from PHP. You would be surprised at what you can do, take the following code:

Példa 1. SWF example

```
<?php
swf_openfile ("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2 (-100, 100, -100, 100);
swf_defineline (1, -70, 0, 70, 0, .2);
swf_definerect (4, 60, -10, 70, 0, 0);
swf_definerect (5, -60, 0, -70, 10, 0);
swf_addcolor (0, 0, 0, 0);

swf_definefont (10, "Mod");
swf_fontsize (5);
swf_fontslant (10);
swf_definetext (11, "This be Flash wit PHP!", 1);

swf_pushmatrix ();
swf_translate (-50, 80, 0);
swf_placeobject (11, 60);
swf_popmatrix ();

for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix ();
    swf_scale (1-($p*.9), 1, 1);
    swf_rotate (60*$p,  'z');
    swf_translate (20+20*$p, $p/1.5, 0);
    swf_rotate (270*$p,  'z');
    swf_addcolor ($p, 0, $p/1.2, -$p);
    swf_placeobject (1, 50);
    swf_placeobject (4, 50);
    swf_placeobject (5, 50);
    swf_popmatrix ();
    swf_showframe ();
}

for ($i = 0; $i < 30; $i++) {
    swf_removeobject (50);
    if (($i%4) == 0) {
        swf_showframe ();
    }
}
```

```
swf_startdoaction ();
swf_actionstop ();
swf_enddoaction ();

swf_closefile ();
?>
```

Megjegyzés: SWF support was added in PHP 4 RC2.

The libswf does not have support for Windows. The development of that library has been stopped, and the source is not available to port it to another systems.

For up to date SWF support take a look at the MING functions.

swf_actiongeturl (PHP 4)

Get a URL from a Shockwave Flash movie

void **swf_actiongeturl** (string url, string target) \linebreak

The **swf_actionGetUrl()** function gets the URL specified by the parameter *`url`* with the target *`target`*.

swf_actiongotoframe (PHP 4)

Play a frame and then stop

void **swf_actiongotoframe** (int framenumber) \linebreak

The **swf_actionGotoFrame()** function will go to the frame specified by *`framenumber`*, play it, and then stop.

swf_actiongotolabel (PHP 4)

Display a frame with the specified label

void **swf_actiongotolabel** (string label) \linebreak

The **swf_actionGotoLabel()** function displays the frame with the label given by the *`label`* parameter and then stops.

swf_actionnextframe (PHP 4)

Go foward one frame

void **swf_actionnextframe** (void) \linebreak

Go foward one frame.

swf_actionplay (PHP 4)

Start playing the flash movie from the current frame

void **swf_actionplay** (void) \linebreak

Start playing the flash movie from the current frame.

swf_actionprevframe (PHP 4)

Go backwards one frame

void **swf_actionprevframe** (void) \linebreak

swf_actionsettarget (PHP 4)

Set the context for actions

void **swf_actionsettarget** (string target) \linebreak

The **swf_actionSetTarget()** function sets the context for all actions. You can use this to control other flash movies that are currently playing.

swf_actionstop (PHP 4)

Stop playing the flash movie at the current frame

void **swf_actionstop** (void) \linebreak

Stop playing the flash movie at the current frame.

swf_actiontogglequality (PHP 4)

Toggle between low and high quality

void **swf_actiontogglequality** (void) \linebreak

Toggle the flash movie between high and low quality.

swf_actionwaitforframe (PHP 4)

Skip actions if a frame has not been loaded

void **swf_actionwaitforframe** (int framenumber, int skipcount) \linebreak

The **swf_actionWaitForFrame()** function will check to see if the frame, specified by the *`framenumber`* parameter has been loaded, if not it will skip the number of actions specified by the *`skipcount`* parameter. This can be useful for "Loading..." type animations.

swf_addbuttonrecord (PHP 4)

Controls location, appearance and active area of the current button

void **swf_addbuttonrecord** (int states, int shapeid, int depth) \linebreak

The **swf_addbuttonrecord()** function allows you to define the specifics of using a button. The first parameter, *states*, defines what states the button can have, these can be any or all of the following constants: BSHitTest, BSDown, BSOver or BSUp. The second parameter, the *shapeid* is the look of the button, this is usually the object id of the shape of the button. The *depth* parameter is the placement of the button in the current frame.

Példa 1. swf_addbuttonrecord() function example

```
swf_startButton ($objid, TYPE_MENUBUTTON);
    swf_addButtonRecord (BSDown|BSOver, $buttonImageId, 340);
    swf_onCondition (MenuEnter);
        swf_actionGetUrl ("http://www.designmultimedia.com", "_level1");
    swf_onCondition (MenuExit);
        swf_actionGetUrl ("", "_level1");
swf_endButton ();
```

swf_addcolor (PHP 4)

Set the global add color to the rgba value specified

void **swf_addcolor** (float r, float g, float b, float a) \linebreak

The **swf_addcolor()** function sets the global add color to the *rgba* color specified. This color is then used (implicitly) by the swf_placeobject(), swf_modifyobject() and the swf_addbuttonrecord() functions. The color of the object will be add by the *rgba* values when the object is written to the screen.

Megjegyzés: The *rgba* values can be either positive or negative.

swf_closefile (PHP 4)

Close the current Shockwave Flash file

void **swf_closefile** ([int return_file]) \linebreak

Close a file that was opened by the swf_openfile() function. If the *return_file* parameter is set then the contents of the SWF file are returned from the function.

Példa 1. Creating a simple flash file based on user input and outputting it and saving it in a database

```
<?php

// The $text variable is submitted by the
// user

// Global variables for database
// access (used in the swf_savedata() function)
$DBHOST = "localhost";
$DBUSER = "sterling";
$DBPASS = "secret";

swf_openfile ("php://stdout", 256, 256, 30, 1, 1, 1);

    swf_definefont (10, "Ligon-Bold");
        swf_fontsize (12);
        swf_fontslant (10);

    swf_definetext (11, $text, 1);

    swf_pushmatrix ();
        swf_translate (-50, 80, 0);
        swf_placeobject (11, 60);
    swf_popmatrix ();

    swf_showframe ();

    swf_startdoaction ();
        swf_actionstop ();
    swf_enddoaction ();

$data = swf_closefile (1);

$data ?
  swf_savedata ($data) :
  die ("Error could not save SWF file");

// void swf_savedata (string data)
// Save the generated file a database
// for later retrieval
function swf_savedata ($data)
{
    global $DBHOST,
           $DBUSER,
           $DBPASS;

    $dbh = @mysql_connect ($DBHOST, $DBUSER, $DBPASS);

    if (!$dbh) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }

    $stmt = "INSERT INTO swf_files (file) VALUES ('$data')";
```

```
    $sth = @mysql_query ($stmt, $dbh);

    if (!$sth) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }

    @mysql_free_result ($sth);
    @mysql_close ($dbh);
}
?>
```

swf_definebitmap (PHP 4)

Define a bitmap

void **swf_definebitmap** (int objid, string image_name) \linebreak

The **swf_definebitmap()** function defines a bitmap given a GIF, JPEG, RGB or FI image. The image will be converted into a Flash JPEG or Flash color map format.

swf_definefont (PHP 4)

Defines a font

void **swf_definefont** (int fontid, string fontname) \linebreak

The **swf_definefont()** function defines a font given by the *fontname* parameter and gives it the id specified by the *fontid* parameter. It then sets the font given by *fontname* to the current font.

swf_defineline (PHP 4)

Define a line

void **swf_defineline** (int objid, float x1, float y1, float x2, float y2, float width) \linebreak

The **swf_defineline()** defines a line starting from the x coordinate given by *x1* and the y coordinate given by *y1* parameter. Up to the x coordinate given by the *x2* parameter and the y coordinate given by the *y2* parameter. It will have a width defined by the *width* parameter.

swf_definepoly (PHP 4)

Define a polygon

void **swf_definepoly** (int objid, array coords, int npoints, float width) \linebreak

The **swf_definepoly()** function defines a polygon given an array of x, y coordinates (the coordinates are defined in the parameter *coords*). The parameter *npoints* is the number of overall points that are contained in the array given by *coords*. The *width* is the width of the polygon's border, if set to 0.0 the polygon is filled.

swf_definerect (PHP 4)

Define a rectangle

void **swf_definerect** (int objid, float x1, float y1, float x2, float y2, float width) \linebreak

The **swf_definerect()** defines a rectangle with an upper left hand coordinate given by the x, *x1*, and the y, *y1*. And a lower right hand coordinate given by the x coordinate, *x2*, and the y coordinate, *y2* . Width of the rectangles border is given by the *width* parameter, if the width is 0.0 then the rectangle is filled.

swf_definetext (PHP 4)

Define a text string

void **swf_definetext** (int objid, string str, int docenter) \linebreak

Define a text string (the *str* parameter) using the current font and font size. The *docenter* is where the word is centered, if *docenter* is 1, then the word is centered in x.

swf_endbutton (PHP 4)

End the definition of the current button

void **swf_endbutton** (void) \linebreak

The **swf_endButton()** function ends the definition of the current button.

swf_enddoaction (PHP 4)

End the current action

void **swf_enddoaction** (void) \linebreak

Ends the current action started by the swf_startdoaction() function.

swf_endshape (PHP 4)

Completes the definition of the current shape

void **swf_endshape** (void) \linebreak

The **swf_endshape()** completes the definition of the current shape.

swf_endsymbol (PHP 4)

End the definition of a symbol

void **swf_endsymbol** (void) \linebreak

The **swf_endsymbol()** function ends the definition of a symbol that was started by the swf_startsymbol() function.

swf_fontsize (PHP 4)

Change the font size

void **swf_fontsize** (float size) \linebreak

The **swf_fontsize()** function changes the font size to the value given by the *size* parameter.

swf_fontslant (PHP 4)

Set the font slant

void **swf_fontslant** (float slant) \linebreak

Set the current font slant to the angle indicated by the *slant* parameter. Positive values create a foward slant, negative values create a negative slant.

swf_fonttracking (PHP 4)

Set the current font tracking

void **swf_fonttracking** (float tracking) \linebreak

Set the font tracking to the value specified by the *tracking* parameter. This function is used to increase the spacing between letters and text, positive values increase the space and negative values decrease the space between letters.

swf_getbitmapinfo (PHP 4)

Get information about a bitmap

array **swf_getbitmapinfo** (int bitmapid) \linebreak

The **swf_getbitmapinfo()** function returns an array of information about a bitmap given by the *bitmapid* parameter. The returned array has the following elements:

- "size" - The size in bytes of the bitmap.
- "width" - The width in pixels of the bitmap.
- "height" - The height in pixels of the bitmap.

swf_getfontinfo (PHP 4)

The height in pixels of a capital A and a lowercase x

array **swf_getfontinfo** (void) \linebreak

The **swf_getfontinfo()** function returns an associative array with the following parameters:

- Aheight - The height in pixels of a capital A.
- xheight - The height in pixels of a lowercase x.

swf_getframe (PHP 4)

Get the frame number of the current frame

int **swf_getframe** (void) \linebreak

The **swf_getframe()** function gets the number of the current frame.

swf_labelframe (PHP 4)

Label the current frame

void **swf_labelframe** (string name) \linebreak

Label the current frame with the name given by the *name* parameter.

swf_lookat (PHP 4)

Define a viewing transformation

void **swf_lookat** (float view_x, float view_y, float view_z, float reference_x, float reference_y, float reference_z, float twist) \linebreak

The **swf_lookat()** function defines a viewing transformation by giving the viewing position (the parameters *view_x*, *view_y*, and *view_z*) and the coordinates of a reference point in the scene, the reference point is defined by the *reference_x*, *reference_y* , and *reference_z* parameters. The *twist* controls the rotation along with viewer's z axis.

swf_modifyobject (PHP 4)

Modify an object

void **swf_modifyobject** (int depth, int how) \linebreak

Updates the position and/or color of the object at the specified depth, *depth*. The parameter *how* determines what is updated. *how* can either be the constant MOD_MATRIX or MOD_COLOR or it can be a combination of both (MOD_MATRIX|MOD_COLOR).

MOD_COLOR uses the current mulcolor (specified by the function swf_mulcolor()) and addcolor (specified by the function swf_addcolor()) to color the object. MOD_MATRIX uses the current matrix to position the object.

swf_mulcolor (PHP 4)

Sets the global multiply color to the rgba value specified

void **swf_mulcolor** (float r, float g, float b, float a) \linebreak

The **swf_mulcolor()** function sets the global multiply color to the *rgba* color specified. This color is then used (implicitly) by the swf_placeobject(), swf_modifyobject() and the swf_addbuttonrecord() functions. The color of the object will be multiplied by the *rgba* values when the object is written to the screen.

Megjegyzés: The *rgba* values can be either positive or negative.

swf_nextid (PHP 4)

Returns the next free object id

int **swf_nextid** (void) \linebreak

The **swf_nextid()** function returns the next available object id.

swf_oncondition (PHP 4)

Describe a transition used to trigger an action list

void **swf_oncondition** (int transition) \linebreak

The **swf_onCondition()** function describes a transition that will trigger an action list. There are several types of possible transitions, the following are for buttons defined as TYPE_MENUBUTTON:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- IdletoOverDown
- OutDowntoIdle
- MenuEnter (IdletoOverUp|IdletoOverDown)
- MenuExit (OverUptoIdle|OverDowntoIdle)

For TYPE_PUSHBUTTON there are the following options:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- OverDowntoOutDown
- OutDowntoOverDown
- OutDowntoIdle
- ButtonEnter (IdletoOverUp|OutDowntoOverDown)
- ButtonExit (OverUptoIdle|OverDowntoOutDown)

swf_openfile (PHP 4)

Open a new Shockwave Flash file

void **swf_openfile** (string filename, float width, float height, float framerate, float r, float g, float b) \linebreak

The **swf_openfile()** function opens a new file named *`filename`* with a width of *`width`* and a height of *`height`* a frame rate of *`framerate`* and background with a red color of *`r`* a green color of *`g`* and a blue color of *`b`*.

The **swf_openfile()** must be the first function you call, otherwise your script will cause a segfault. If you want to send your output to the screen make the filename: "php://stdout" (support for this is in 4.0.1 and up).

swf_ortho2 (PHP 4)

Defines 2D orthographic mapping of user coordinates onto the current viewport

void **swf_ortho2** (float xmin, float xmax, float ymin, float ymax) \linebreak

The **swf_ortho2()** function defines a two dimensional orthographic mapping of user coordinates onto the current viewport, this defaults to one to one mapping of the area of the Flash movie. If a perspective transformation is desired, the **swf_perspective ()** function can be used.

swf_ortho (PHP 4 >= 4.0.1)

Defines an orthographic mapping of user coordinates onto the current viewport

void **swf_ortho** (float xmin, float xmax, float ymin, float ymax, float zmin, float zmax) \linebreak

The **swf_ortho()** function defines a orthographic mapping of user coordinates onto the current viewport.

swf_perspective (PHP 4)

Define a perspective projection transformation

void **swf_perspective** (float fovy, float aspect, float near, float far) \linebreak

The **swf_perspective()** function defines a perspective projection transformation. The *fovy* parameter is field-of-view angle in the y direction. The *aspect* parameter should be set to the aspect ratio of the viewport that is being drawn onto. The *near* parameter is the near clipping plane and the *far* parameter is the far clipping plane.

Megjegyzés: Various distortion artifacts may appear when performing a perspective projection, this is because Flash players only have a two dimensional matrix. Some are not to pretty.

swf_placeobject (PHP 4)

Place an object onto the screen

void **swf_placeobject** (int objid, int depth) \linebreak

Places the object specified by *objid* in the current frame at a depth of *depth*. The *objid* parameter and the *depth* must be between 1 and 65535.

This uses the current mulcolor (specified by swf_mulcolor()) and the current addcolor (specified by swf_addcolor()) to color the object and it uses the current matrix to position the object.

Megjegyzés: Full RGBA colors are supported.

swf_polarview (PHP 4)

Define the viewer's position with polar coordinates

void **swf_polarview** (float dist, float azimuth, float incidence, float twist) \linebreak

The **swf_polarview()** function defines the viewer's position in polar coordinates. The *dist* parameter gives the distance between the viewpoint to the world space origin. The *azimuth* parameter defines the azimuthal angle in the x,y coordinate plane, measured in distance from the y axis. The *incidence* parameter defines the angle of incidence in the y,z plane, measured in distance from the z axis. The incidence angle is defined as the angle of the viewport relative to the z axis. Finally the *twist* specifies the amount that the viewpoint is to be rotated about the line of sight using the right hand rule.

swf_popmatrix (PHP 4)

Restore a previous transformation matrix

void **swf_popmatrix** (void) \linebreak

The **swf_popmatrix()** function pushes the current transformation matrix back onto the stack.

swf_posround (PHP 4)

Enables or Disables the rounding of the translation when objects are placed or moved

void **swf_posround** (int round) \linebreak

The **swf_posround()** function enables or disables the rounding of the translation when objects are placed or moved, there are times when text becomes more readable because rounding has been enabled. The *round* is whether to enable rounding or not, if set to the value of 1, then rounding is enabled, if set to 0 then rounding is disabled.

swf_pushmatrix (PHP 4)

Push the current transformation matrix back unto the stack

void **swf_pushmatrix** (void) \linebreak

The **swf_pushmatrix()** function pushes the current transformation matrix back onto the stack.

swf_removeobject (PHP 4)

Remove an object

void **swf_removeobject** (int depth) \linebreak

Removes the object at the depth specified by *depth*.

swf_rotate (PHP 4)

Rotate the current transformation

void **swf_rotate** (float angle, string axis) \linebreak

The **swf_rotate()** rotates the current transformation by the angle given by the *angle* parameter around the axis given by the *axis* parameter. Valid values for the axis are 'x' (the x axis), 'y' (the y axis) or 'z' (the z axis).

swf_scale (PHP 4)

Scale the current transformation

void **swf_scale** (float x, float y, float z) \linebreak

The **swf_scale()** scales the x coordinate of the curve by the value of the *x* parameter, the y coordinate of the curve by the value of the *y* parameter, and the z coordinate of the curve by the value of the *z* parameter.

swf_setfont (PHP 4)

Change the current font

void **swf_setfont** (int fontid) \linebreak

The **swf_setfont()** sets the current font to the value given by the *fontid* parameter.

swf_setframe (PHP 4)

Switch to a specified frame

void **swf_setframe** (int framenumber) \linebreak

The **swf_setframe()** changes the active frame to the frame specified by *framenumber*.

swf_shapearc (PHP 4)

Draw a circular arc

void **swf_shapearc** (float x, float y, float r, float ang1, float ang2) \linebreak

The **swf_shapeArc()** function draws a circular arc from angle A given by the *ang1* parameter to angle B given by the *ang2* parameter. The center of the circle has an x coordinate given by the *x* parameter and a y coordinate given by the *y*, the radius of the circle is given by the *r* parameter.

swf_shapecurveto3 (PHP 4)

Draw a cubic bezier curve

void **swf_shapecurveto3** (float x1, float y1, float x2, float y2, float x3, float y3) \linebreak

Draw a cubic bezier curve using the x,y coordinate pairs *x1*, *y1* and *x2*,*y2* as off curve control points and the x,y coordinate *x3*, *y3* as an endpoint. The current position is then set to the x,y coordinate pair given by *x3*,*y3*.

swf_shapecurveto (PHP 4)

Draw a quadratic bezier curve between two points

void **swf_shapecurveto** (float x1, float y1, float x2, float y2) \linebreak

The **swf_shapecurveto()** function draws a quadratic bezier curve from the current location, though the x coordinate given by *x1* and the y coordinate given by *y1* to the x coordinate given by *x2* and the y coordinate given by *y2*. The current position is then set to the x,y coordinates given by the *x2* and *y2* parameters

swf_shapefillbitmapclip (PHP 4)

Set current fill mode to clipped bitmap

void **swf_shapefillbitmapclip** (int bitmapid) \linebreak

Sets the fill to bitmap clipped, empty spaces will be filled by the bitmap given by the *bitmapid* parameter.

swf_shapefillbitmaptile (PHP 4)

Set current fill mode to tiled bitmap

void **swf_shapefillbitmaptile** (int bitmapid) \linebreak

Sets the fill to bitmap tile, empty spaces will be filled by the bitmap given by the *bitmapid* parameter (tiled).

swf_shapefilloff (PHP 4)

Turns off filling

void **swf_shapefilloff** (void) \linebreak

The **swf_shapeFillOff()** function turns off filling for the current shape.

swf_shapefillsolid (PHP 4)

Set the current fill style to the specified color

void **swf_shapefillsolid** (float r, float g, float b, float a) \linebreak

The **swf_shapeFillSolid()** function sets the current fill style to solid, and then sets the fill color to the values of the *rgba* parameters.

swf_shapelinesolid (PHP 4)

Set the current line style

void **swf_shapelinesolid** (float r, float g, float b, float a, float width) \linebreak

The **swf_shapeLineSolid()** function sets the current line style to the color of the *rgba* parameters and width to the *width* parameter. If 0.0 is given as a width then no lines are drawn.

swf_shapelineto (PHP 4)

Draw a line

void **swf_shapelineto** (float x, float y) \linebreak

The **swf_shapeLineTo()** draws a line to the x,y coordinates given by the *x* parameter & the *y* parameter. The current position is then set to the x,y parameters.

swf_shapemoveto (PHP 4)

Move the current position

void **swf_shapemoveto** (float x, float y) \linebreak

The **swf_shapeMoveTo()** function moves the current position to the x coordinate given by the *x* parameter and the y position given by the *y* parameter.

swf_showframe (PHP 4)

Display the current frame

void **swf_showframe** (void) \linebreak

The swf_showframe function will output the current frame.

swf_startbutton (PHP 4)

Start the definition of a button

void **swf_startbutton** (int objid, int type) \linebreak

The **swf_startbutton()** function starts off the definition of a button. The *type* parameter can either be TYPE_MENUBUTTON or TYPE_PUSHBUTTON. The TYPE_MENUBUTTON constant allows the focus to travel from the button when the mouse is down, TYPE_PUSHBUTTON does not allow the focus to travel when the mouse is down.

swf_startdoaction (PHP 4)

Start a description of an action list for the current frame

void **swf_startdoaction** (void) \linebreak

The **swf_startdoaction()** function starts the description of an action list for the current frame. This must be called before actions are defined for the current frame.

swf_startshape (PHP 4)

Start a complex shape

void **swf_startshape** (int objid) \linebreak

The **swf_startshape()** function starts a complex shape, with an object id given by the *objid* parameter.

swf_startsymbol (PHP 4)

Define a symbol

void **swf_startsymbol** (int objid) \linebreak

Define an object id as a symbol. Symbols are tiny flash movies that can be played simultaneously. The *objid* parameter is the object id you want to define as a symbol.

swf_textwidth (PHP 4)

Get the width of a string

float **swf_textwidth** (string str) \linebreak

The **swf_textwidth()** function gives the width of the string, *str*, in pixels, using the current font and font size.

swf_translate (PHP 4)

Translate the current transformations

void **swf_translate** (float x, float y, float z) \linebreak

The **swf_translate()** function translates the current transformation by the *x*, *y*, and *z* values given.

swf_viewport (PHP 4)

Select an area for future drawing

void **swf_viewport** (float xmin, float xmax, float ymin, float ymax) \linebreak

The **swf_viewport()** function selects an area for future drawing for *xmin* to *xmax* and *ymin* to *ymax*, if this function is not called the area defaults to the size of the screen.

XCV. SNMP functions

In order to use the SNMP functions on Unix you need to install the UCD SNMP (http://net-snmp.sourceforge.net/) package. On Windows these functions are only available on NT and not on Win95/98.

Important: In order to use the UCD SNMP package, you need to define NO_ZEROLENGTH_COMMUNITY to 1 before compiling it. After configuring UCD SNMP, edit config.h and search for NO_ZEROLENGTH_COMMUNITY. Uncomment the #define line. It should look like this afterwards:

```
#define NO_ZEROLENGTH_COMMUNITY 1
```

If you see strange segmentation faults in combination with SNMP commands, you did not follow the above instructions. If you do not want to recompile UCD SNMP, you can compile PHP with the --enable-ucd-snmp-hack switch which will work around the misfeature.

snmp_get_quick_print (PHP 3>= 3.0.8, PHP 4)

Fetch the current value of the UCD library's quick_print setting

bool **snmp_get_quick_print** (void) \linebreak

Returns the current value stored in the UCD Library for quick_print. quick_print is off by default.

```
$quickprint = snmp_get_quick_print();
```

Above function call would return `FALSE` if quick_print is off, and `TRUE` if quick_print is on.

snmp_get_quick_print() is only available when using the UCD SNMP library. This function is not available when using the Windows SNMP library.

See: snmp_set_quick_print() for a full description of what quick_print does.

snmp_set_quick_print (PHP 3>= 3.0.8, PHP 4)

Set the value of quick_print within the UCD SNMP library

void **snmp_set_quick_print** (bool quick_print) \linebreak

Sets the value of quick_print within the UCD SNMP library. When this is set (1), the SNMP library will return 'quick printed' values. This means that just the value will be printed. When quick_print is not enabled (default) the UCD SNMP library prints extra information including the type of the value (i.e. IpAddress or OID). Additionally, if quick_print is not enabled, the library prints additional hex values for all strings of three characters or less.

Setting quick_print is often used when using the information returned rather then displaying it.

```
snmp_set_quick_print(0);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
snmp_set_quick_print(1);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
```

The first value printed might be: 'Timeticks: (0) 0:00:00.00', whereas with quick_print enabled, just '0:00:00.00' would be printed.

By default the UCD SNMP library returns verbose values, quick_print is used to return only the value.

Currently strings are still returned with extra quotes, this will be corrected in a later release.

snmp_set_quick_print() is only available when using the UCD SNMP library. This function is not available when using the Windows SNMP library.

snmpget (PHP 3, PHP 4)

Fetch an SNMP object

string **snmpget** (string hostname, string community, string object_id [, int timeout [, int retries]]) \linebreak

Returns SNMP object value on success and FALSE on error.

The **snmpget()** function is used to read the value of an SNMP object specified by the *object_id*. SNMP agent is specified by the *hostname* and the read community is specified by the *community* parameter.

```
$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0");
```

snmprealwalk (PHP 3>= 3.0.8, PHP 4)

Return all objects including their respective object ID within the specified one

array **snmprealwalk** (string host, string community, string object_id [, int timeout [, int retries]]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

snmpset (PHP 3>= 3.0.12, PHP 4)

Set an SNMP object

bool **snmpset** (string hostname, string community, string object_id, string type, mixed value [, int timeout [, int retries]]) \linebreak

Sets the specified SNMP object value, returning TRUE on success and FALSE on error.

The **snmpset()** function is used to set the value of an SNMP object specified by the *object_id*. SNMP agent is specified by the *hostname* and the read community is specified by the *community* parameter.

snmpwalk (PHP 3, PHP 4)

Fetch all the SNMP objects from an agent

array **snmpwalk** (string hostname, string community, string object_id [, int timeout [, int retries]]) \linebreak

Returns an array of SNMP object values starting from the *object_id* as root and FALSE on error.

snmpwalk() function is used to read all the values from an SNMP agent specified by the *hostname*. *Community* specifies the read community for that agent. A NULL *object_id* is taken as the root of the SNMP objects tree and all objects under that tree are returned as an array. If *object_id* is specified, all the SNMP objects below that *object_id* are returned.

```
$a = snmpwalk("127.0.0.1", "public", "");
```

Above function call would return all the SNMP objects from the SNMP agent running on localhost. One can step through the values with a loop

```
for ($i=0; $i < count($a); $i++) {
    echo $a[$i];
}
```

snmpwalkoid (PHP 3>= 3.0.8, PHP 4)

Query for a tree of information about a network entity

array **snmpwalkoid** (string hostname, string community, string object_id [, int timeout [, int retries]]) \linebreak

Returns an associative array with object ids and their respective object value starting from the *object_id* as root and FALSE on error.

snmpwalkoid() function is used to read all object ids and their respective values from an SNMP agent specified by the *hostname*. Community specifies the read *community* for that agent. A NULL *object_id* is taken as the root of the SNMP objects tree and all objects under that tree are returned as an array. If *object_id* is specified, all the SNMP objects below that *object_id* are returned.

The existence of **snmpwalkoid()** and snmpwalk() has historical reasons. Both functions are provided for backward compatibility.

```
$a = snmpwalkoid("127.0.0.1", "public", "");
```

Above function call would return all the SNMP objects from the SNMP agent running on localhost. One can step through the values with a loop

```
for (reset($a); $i = key($a); next($a)) {
    echo "$i: $a[$i]<br>\n";
}
```

XCVI. Socket functions

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

The socket extension implements a low-level interface to the socket communication functions based on the popular BSD sockets, providing the possibility to act as a socket server as well as a client.

The socket functions described here are part of an extension to PHP which must be enabled at compile time by giving the `--enable-sockets` option to **configure**.

For a more generic client-side socket interface, see fsockopen() and pfsockopen().

When using these functions, it is important to remember that while many of them have identical names to their C counterparts, they often have different declarations. Please be sure to read the descriptions to avoid confusion.

The socket extension was written to provide a useable interface to the powerful BSD sockets. Care has been taken that the functions work equally well on Win32 and Unix implementations. Almost all of the sockets functions may fail under certain conditions and therefore emit an `E_WARNING` message describing the error. Sometimes this doesn't happen to the desire of the developer. For example the function socket_read() may suddenly emit an `E_WARNING` message because the connection broke unexpectedly. It's common to suppress the warning with the `@`-operator and catch the error code within the application with the socket_last_error() function. You may call the socket_strerror() function with this error code to retrieve a string describing the error. See their description for more information.

> **Megjegyzés:** The `E_WARNING` messages generated by the socket extension are in english though the retrieved error message will appear depending on the current locale (`LC_MESSAGES`):

```
Warning - socket_bind() unable to bind address [98]: Die Adresse wird bereits verwendet
```

That said, those unfamiliar with socket programming can still find a lot of useful material in the appropriate Unix man pages, and there is a great deal of tutorial information on socket programming in C on the web, much of which can be applied, with slight modifications, to socket programming in PHP. The UNIX Socket FAQ (http://www.developerweb.net/sock-faq/) might be a good start.

Példa 1. Socket example: Simple TCP/IP server

This example shows a simple talkback server. Change the `address` and `port` variables to suit your setup and execute. You may then connect to the server with a command similar to: **telnet**

192.168.1.53 10000 (where the address and port match your setup). Anything you type will then be output on the server side, and echoed back to you. To disconnect, enter 'quit'.

```
#!/usr/local/bin/php -q
<?php
error_reporting (E_ALL);

/* Allow the script to hang around waiting for connections. */
set_time_limit (0);

/* Turn on implicit output flushing so we see what we're getting
 * as it comes in. */
ob_implicit_flush ();

$address = '192.168.1.53';
$port = 10000;

if (($sock = socket_create (AF_INET, SOCK_STREAM, 0)) < 0) {
    echo "socket_create() failed: reason: " . socket_strerror ($sock) . "\n";
}

if (($ret = socket_bind ($sock, $address, $port)) < 0) {
    echo "socket_bind() failed: reason: " . socket_strerror ($ret) . "\n";
}

if (($ret = socket_listen ($sock, 5)) < 0) {
    echo "socket_listen() failed: reason: " . socket_strerror ($ret) . "\n";
}

do {
    if (($msgsock = socket_accept($sock)) < 0) {
        echo "socket_accept() failed: reason: " . socket_strerror ($msgsock) . "\n";
        break;
    }
    /* Send instructions. */
    $msg = "\nWelcome to the PHP Test Server. \n" .
        "To quit, type 'quit'. To shut down the server type 'shutdown'.\n";
    socket_write($msgsock, $msg, strlen($msg));

    do {
        if (FALSE === ($buf = socket_read ($msgsock, 2048))) {
            echo "socket_read() failed: reason: " . socket_strerror ($ret) . "\n";
            break 2;
        }
        if (!$buf = trim ($buf)) {
            continue;
        }
        if ($buf == 'quit') {
            break;
        }
        if ($buf == 'shutdown') {
            socket_close ($msgsock);
            break 2;
        }
        $talkback = "PHP: You said '$buf'.\n";
        socket_write ($msgsock, $talkback, strlen ($talkback));
        echo "$buf\n";
```

```
    } while (true);
    socket_close ($msgsock);
} while (true);

socket_close ($sock);
?>
```

Példa 2. Socket example: Simple TCP/IP client

This example shows a simple, one-shot HTTP client. It simply connects to a page, submits a HEAD request, echoes the reply, and exits.

```
<?php
error_reporting (E_ALL);

echo "<h2>TCP/IP Connection</h2>\n";

/* Get the port for the WWW service. */
$service_port = getservbyname (’www’, ’tcp’);

/* Get the IP address for the target host. */
$address = gethostbyname (’www.example.com’);

/* Create a TCP/IP socket. */
$socket = socket_create (AF_INET, SOCK_STREAM, 0);
if ($socket < 0) {
    echo "socket_create() failed: reason: " . socket_strerror ($socket) . "\n";
} else {
    echo "OK.\n";
}

echo "Attempting to connect to ’$address’ on port ’$service_port’...";
$result = socket_connect ($socket, $address, $service_port);
if ($result < 0) {
    echo "socket_connect() failed.\nReason: ($result) " . socket_strerror($result) . "\n'
} else {
    echo "OK.\n";
}

$in = "HEAD / HTTP/1.0\r\n\r\n";
$out = ”;

echo "Sending HTTP HEAD request...";
socket_write ($socket, $in, strlen ($in));
echo "OK.\n";

echo "Reading response:\n\n";
while ($out = socket_read ($socket, 2048)) {
    echo $out;
}

echo "Closing socket...";
```

```
socket_close ($socket);
echo "OK.\n\n";
?>
```

socket_accept (PHP 4 >= 4.1.0)

Accepts a connection on a socket

resource **socket_accept** (resource socket) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

After the socket *socket* has been created using socket_create(), bound to a name with socket_bind(), and told to listen for connections with socket_listen(), this function will accept incoming connections on that socket. Once a successful connection is made, a new socket resource is returned, which may be used for communication. If there are multiple connections queued on the socket, the first will be used. If there are no pending connections, **socket_accept()** will block until a connection becomes present. If *socket* has been made non-blocking using socket_set_blocking() or socket_set_nonblock(), FALSE will be returned.

The socket resource returned by **socket_accept()** may not be used to accept new connections. The original listening socket *socket*, however, remains open and may be reused.

Returns a new socket resource on success, or FALSE on error. The actual error code can be retrieved by calling socket_last_error(). This error code may be passed to socket_strerror() to get a textual explanation of the error.

See also socket_bind(), socket_connect(), socket_listen(), socket_create(), and socket_strerror().

socket_bind (PHP 4 >= 4.1.0)

Binds a name to a socket

bool **socket_bind** (resource socket, string address [, int port]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

socket_bind() binds the name given in *address* to the socket described by *socket*, which must be a valid socket resource created with socket_create().

The *address* parameter is either an IP address in dotted-quad notation (e.g. `127.0.0.1`), if the socket is of the `AF_INET` family; or the pathname of a Unix-domain socket, if the socket family is `AF_UNIX`.

The *port* parameter is only used when connecting to an `AF_INET` socket, and designates the port on the remote host to which a connection should be made.

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

See also socket_connect(), socket_listen(), socket_create(), socket_last_error() and socket_strerror().

socket_clear_error (PHP 4 >= 4.2.0)

Clears the error on the socket or the last error code

void **socket_clear_error** ([resource socket]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

This function clears the error code on the given socket or the global last socket error.

This function allows explicitely resetting the error code value either of a socket or of the extension global last error code. This may be useful to detect within a part of the application if an error occured or not.

See also socket_last_error() and socket_strerror().

socket_close (PHP 4 >= 4.1.0)

Closes a socket resource

void **socket_close** (resource socket) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

socket_close() closes the socket resource given by *`socket`*.

> **Megjegyzés: socket_close()** can't be used on PHP file resources created with fopen(), popen(), fsockopen(), or **psockopen()**; it is meant for sockets created with socket_create() or socket_accept().

See also socket_bind(), socket_listen(), socket_create() and socket_strerror().

socket_connect (PHP 4 >= 4.1.0)

Initiates a connection on a socket

bool **socket_connect** (resource socket, string address [, int port]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Initiates a connection using the socket resource *`socket`*, which must be a valid socket resource created with socket_create().

The *`address`* parameter is either an IP address in dotted-quad notation (e.g. `127.0.0.1`), if the socket is of the `AF_INET` family; or the pathname of a Unix-domain socket, if the socket family is `AF_UNIX`.

The *`port`* parameter is only used when connecting to an `AF_INET` socket, and designates the port on the remote host to which a connection should be made.

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

See also socket_bind(), socket_listen(), socket_create(), socket_last_error() and socket_strerror().

socket_create_listen (PHP 4 >= 4.1.0)

Opens a socket on port to accept connections

resource **socket_create_listen** (int port [, int backlog]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

This function is meant to ease the task of creating a new socket which only listens to accept new connections.

socket_create_listen() create a new socket resource of type `AF_INET` listening on *all* local interfaces on the given port waiting for new connections.

The *`backlog`* parameter defines the maximum length the queue of pending connections may grow to. `SOMAXCONN` may be passed as *`backlog`* parameter, see socket_listen() for more information.

socket_create_listen() returns a new socket resource on success or `FALSE` on error. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

Megjegyzés: If you want to create a socket which only listens on a certain interfaces you need to use socket_create(), socket_bind() and socket_listen().

See also socket_create(), socket_bind(), socket_listen(), socket_last_error() and socket_strerror().

socket_create_pair (PHP 4 >= 4.1.0)

Creates a pair of indistinguishable sockets and stores them in fds.

bool **socket_create_pair** (int domain, int type, int protocol, array &fd) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_create (PHP 4 >= 4.1.0)

Create a socket (endpoint for communication)

resource **socket_create** (int domain, int type, int protocol) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Creates a communication endpoint (a socket), and returns a socket resource.

The *`domain`* parameter sets the domain (protocol family) to be used for communication. Currently, `AF_INET` and `AF_UNIX` are understood. `AF_INET` is typical used for internet based communication. `AF_UNIX` uses pathnames to identify sockets and can therefore only be used for local communication (which is faster, on the other hand).

The *type* parameter selects the socket type. This is one of SOCK_STREAM, SOCK_DGRAM, SOCK_SEQPACKET, SOCK_RAW, SOCK_RDM, or SOCK_PACKET. The two most common types are SOCK_DGRAM for UDP (connectionless) communication and SOCK_STREAM for TCP communication.

protocol sets the protocol which is either SOL_UDP or SOL_TCP.

Returns a socket resource on success, or FALSE on error. The actual error code can be retrieved by calling socket_last_error(). This error code may be passed to socket_strerror() to get a textual explanation of the error.

For more information on the usage of **socket_create()**, as well as on the meanings of the various parameters, see the Unix man page socket (2).

Megjegyzés: If an invalid *domain* or *type* is given, **socket_create()** defaults to AF_INET and SOCK_STREAM respectively and additionally emits an E_WARNING message.

See also socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_last_error(), and socket_strerror().

socket_get_option (PHP 4 CVS only)

Gets socket options for the socket

mixed **socket_get_option** (resource socket, int level, int optname) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

Megjegyzés: This function used to be called socket_getopt() prior to PHP 4.3.0

socket_getpeername (PHP 4 >= 4.1.0)

Queries the remote side of the given socket which may either result in host/port or in a UNIX filesystem path, dependent on its type.

bool **socket_getpeername** (resource socket, string &addr [, int &port]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

If the given socket is of type `AF_INET`, **socket_getpeername()** will return the peers (remote) *IP address* in dotted-quad notation (e.g. `127.0.0.1`) in the *`address`* parameter and, if the optional *`port`* parameter is present, also the associated port.

If the given socket is of type `AF_UNIX`, **socket_getpeername()** will return the UNIX filesystem path (e.g. `/var/run/daemon.sock`) in the *`address`* parameter.

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad. **socket_getpeername()** may also return `FALSE` if the socket type is not any of `AF_INET` or `AF_UNIX`, in which case the last socket error code is *not* updated.

See also **socket_getpeername()**, socket_last_error() and socket_strerror().

socket_getsockname (PHP 4 >= 4.1.0)

Queries the local side of the given socket which may either result in host/port or in a UNIX filesystem path, dependent on its type.

bool **socket_getsockname** (resource socket, string &addr [, int &port]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

If the given socket is of type `AF_INET`, **socket_getsockname()** will return the local *IP address* in dotted-quad notation (e.g. `127.0.0.1`) in the *`address`* parameter and, if the optional *`port`* parameter is present, also the associated port.

If the given socket is of type `AF_UNIX`, **socket_getsockname()** will return the UNIX filesystem path (e.g. `/var/run/daemon.sock`) in the *`address`* parameter.

Siker esetén `TRUE` értékkel tér vissza, ellenkező esetben `FALSE` értéket ad. **socket_getsockname()** may also return `FALSE` if the socket type is not any of `AF_INET` or `AF_UNIX`, in which case the last socket error code is *not* updated.

See also socket_getpeername(), socket_last_error() and socket_strerror().

socket_iovec_add (PHP 4 >= 4.1.0)

Adds a new vector to the scatter/gather array

bool **socket_iovec_add** (resource iovec, int iov_len) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_iovec_alloc (PHP 4 >= 4.1.0)

...]) Builds a 'struct iovec' for use with sendmsg, recvmsg, writev, and readv

resource **socket_iovec_alloc** (int num_vectors [, int]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_iovec_delete (PHP 4 >= 4.1.0)

Deletes a vector from an array of vectors

bool **socket_iovec_delete** (resource iovec, int iov_pos) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_iovec_fetch (PHP 4 >= 4.1.0)

Returns the data held in the iovec specified by iovec_id[iovec_position]

string **socket_iovec_fetch** (resource iovec, int iovec_position) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_iovec_free (PHP 4 >= 4.1.0)

Frees the iovec specified by iovec_id

bool **socket_iovec_free** (resource iovec) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_iovec_set (PHP 4 >= 4.1.0)

Sets the data held in iovec_id[iovec_position] to new_val

bool **socket_iovec_set** (resource iovec, int iovec_position, string new_val) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_last_error (PHP 4 >= 4.1.0)

Returns the last error on the socket

int **socket_last_error** ([resource socket]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

This function returns a socket error code.

If a socket resource is passed to this function, the last error which occured on this particular socket is returned. If the socket resource is ommited, the error code of the last failed socket function is returned. The latter is in particular helpful for functions like socket_create() which don't return a socket on failure and socket_select() which can fail for reasons not directly tied to a particular socket. The error code is suitable to be fed to socket_strerror() which returns a string describing the given error code.

```
if (false == ($socket = @socket_create(AF_INET, SOCK_STREAM, SOL_TCP))) {
    die("Couldn't create socket, error code is: " . socket_last_error() .
        ",error message is: " . socket_strerror(socket_last_error()));
}
```

Megjegyzés: socket_last_error() does not clear the error code, use socket_clear_error() for this purpose.

socket_listen (PHP 4 >= 4.1.0)

Listens for a connection on a socket

bool **socket_listen** (resource socket [, int backlog]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

After the socket *`socket`* has been created using socket_create() and bound to a name with socket_bind(), it may be told to listen for incoming connections on *`socket`*.

A maximum of *`backlog`* incoming connections will be queued for processing. If a connection request arrives with the queue full the client may receive an error with an indication of `ECONNREFUSED`, or, if the underlying protocol supports retransmission, the request may be ignored so that retries may succeed.

> **Megjegyzés:** The maximum number passed to the *backlog* parameter highly depends on the underlying platform. On linux, it is silently truncated to SOMAXCONN. On win32, if passed SOMAXCONN, the underlying service provider responsible for the socket will set the backlog to a maximum *reasonable* value. There is no standard provision to find out the actual backlog value on this platform.

socket_listen() is applicable only to sockets of type SOCK_STREAM or SOCK_SEQPACKET.

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

See also socket_accept(), socket_bind(), socket_connect(), socket_create() and socket_strerror().

socket_read (PHP 4 >= 4.1.0)

Reads a maximum of length bytes from a socket

string **socket_read** (resource socket, int length [, int type]) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

The function **socket_read()** reads from the socket resource *socket* created by the socket_create() or socket_accept() functions. The maximum number of bytes read is specified by the *length* parameter. Otherwise you can use \r, \n, or \0 to end reading (depending on the *type* parameter, see below).

socket_read() returns the data as a string on success, or FALSE on error. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual representation of the error.

> **Megjegyzés: socket_read()** may return a zero length string ("") indicating the end of communication (i.e. the remote end point has closed the connection).

Optional *type* parameter is a named constant:

- PHP_BINARY_READ - use the system read() function. Safe for reading binary data. (Default in PHP >= 4.1.0)
- PHP_NORMAL_READ - reading stops at \n or \r. (Default in PHP <= 4.0.6)

See also socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_last_error(), socket_strerror() and socket_write().

socket_readv (PHP 4 >= 4.1.0)

Reads from an fd, using the scatter-gather array defined by iovec_id

bool **socket_readv** (resource socket, resource iovec_id) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_recv (PHP 4 >= 4.1.0)

Receives data from a connected socket

string **socket_recv** (resource socket, int len, int flags) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_recvfrom (PHP 4 >= 4.1.0)

Receives data from a socket, connected or not

int **socket_recvfrom** (resource socket, string &buf, int len, int flags, string &name [, int &port]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_recvmsg (PHP 4 >= 4.1.0)

Used to receive messages on a socket, whether connection-oriented or not

bool **socket_recvmsg** (resource socket, resource iovec, array &control, int &controllen, int &flags, string &addr [, int &port]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_select (PHP 4 >= 4.1.0)

Runs the select() system call on the given arrays of sockets with a timeout specified by tv_sec and tv_usec

int **socket_select** (resource &read, resource &write, resource &except, int tv_sec [, int tv_usec]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

The **socket_select()** accepts arrays of sockets and waits for them to change status. Those coming with BSD sockets background will recognize that those socket resource arrays are in fact the so-called file descriptor sets. Three independent arrays of socket resources are watched.

The sockets listed in the *read* array will be watched to see if characters become available for reading (more precisely, to see if a read will not block - in particular, a socket resource is also ready on end-of-file, in which case a socket_read() will return a zero length string).

The sockets listed in the *write* array will be watched to see if a write will not block.

The sockets listed in the *except* array will be watched for exceptions.

Figyelem

On exit, the arrays are modified to indicate which socket resource actually changed status.

You do not need to pass every array to **socket_select()**. You can leave it out and use an empty array or NULL instead. Also do not forget that those arrays are passed *by reference* and will be modified after **socket_select()** returns.

Example:

```
/* Prepare the read array */
$read = array($socket1, $socket2);

if (false === ($num_changed_sockets = socket_select($read, $write = NULL, $ex-
cept = NULL, 0))) {
    /* Error handling */
else if ($num_changed_sockets > 0) {
    /* At least at one of the sockets something interesting happened */
}
```

Megjegyzés: Due a limitation in the current Zend Engine it is not possible to pass a constant modifier like NULL directly as a parameter to a function which expects this parameter to be passed by reference. Instead use a temporary variable or an expression with the leftmost member being a temporary variable:

```
socket_select($r, $w, $e = NULL, 0);
```

The *tv_sec* and *tv_usec* together form the *timeout* parameter. The *timeout* is an upper bound on the amount of time elapsed before **socket_select()** return. *tv_sec* may be zero , causing **socket_select()** to return immediately. This is useful for polling. If *tv_sec* is NULL (no timeout), **socket_select()** can block indefinitely.

On success **socket_select()** returns the number of socket resorces contained in the modified arrays, which may be zero if the timeout expires before anything interesting happens. On error FALSE is returned. The error code can be retrieved with socket_last_error().

Megjegyzés: Be sure to use the === operator when checking for an error. Since the **socket_select()** may return 0 the comparison with == would evaluate to TRUE:

```
if (false === socket_select($r, $w, $e = NULL, 0)) {
    echo "socket_select() failed, reason: " . socket_strerror(socket_last_error()) . "\n";
}
```

Megjegyzés: Be aware that some socket implementations need to be handled very carefully. A few basic rules:

- You should always try to use **socket_select()** without timeout. Your program should have nothing to do if there is no data available. Code that depends on timeouts is not usually portable and difficult to debug.
- No socket resource must be added to any set if you do not intend to check its result after the **socket_select()** call, and respond appropriately. After **socket_select()** returns, all socket resources in all arrays must be checked. Any socket resource that is available for writing must be written to, and any socket resource available for reading must be read from.
- If you read/write to a socket returns in the arrays be aware that they do not necessarily read/write the full amount of data you have requested. Be prepared to even only be able to read/write a single byte.
- It's common to most socket implementations that the only exception caught with the *except* array is out-of-bound data received on a socket.

See also socket_read(), socket_write(), socket_last_error() and socket_strerror().

socket_send (PHP 4 >= 4.1.0)

Sends data to a connected socket

int **socket_send** (resource socket, string buf, int len, int flags) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_sendmsg (PHP 4 >= 4.1.0)

Sends a message to a socket, regardless of whether it is connection-oriented or not

bool **socket_sendmsg** (resource socket, resource iovec, int flags, string addr [, int port]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_sendto (PHP 4 >= 4.1.0)

Sends a message to a socket, whether it is connected or not

int **socket_sendto** (resource socket, string buf, int len, int flags, string addr [, int port]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_set_nonblock (PHP 4 >= 4.1.0)

Sets nonblocking mode for file descriptor fd

bool **socket_set_nonblock** (resource socket) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_set_option (PHP 4 CVS only)

Sets socket options for the socket

bool **socket_set_option** (resource socket, int level, int optname, int) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

Megjegyzés: This function used to be called `socket_setopt()` prior to PHP 4.3.0

socket_shutdown (PHP 4 >= 4.1.0)

Shuts down a socket for receiving, sending, or both.

bool **socket_shutdown** (resource socket [, int how]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

socket_strerror (PHP 4 >= 4.1.0)

Return a string describing a socket error

string **socket_strerror** (int errno) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

socket_strerror() takes as its *errno* parameter a socket error code as returned by socket_last_error() and returns the corresponding explanatory text. This makes it a bit more pleasant to figure out why something didn't work; for instance, instead of having to track down a system include file to find out what '-111' means, you just pass it to **socket_strerror()**, and it tells you what happened.

Példa 1. socket_strerror() example

```
<?php
if (false == ($socket = @socket_create(AF_INET, SOCK_STREAM, 0))) {
   echo "socket_create() failed: reason: " . socket_strerror(socket_last_error()) . "\n";
}

if (false == (@socket_bind($socket, '127.0.0.1', 80))) {
   echo "socket_bind() failed: reason: " . socket_strerror(socket_last_error($socket)) .
}
?>
```

The expected output from the above example (assuming the script is not run with root privileges):

```
socket_bind() failed: reason: Permission denied
```

See also socket_accept(), socket_bind(), socket_connect(), socket_listen(), and socket_create().

socket_write (PHP 4 >= 4.1.0)

Write to a socket

int **socket_write** (resource socket, string buffer [, int length]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

The function **socket_write()** writes to the socket *socket* from *buffer*.

The optional parameter *length* can specify an alternate length of bytes written to the socket. If this length is greater then the buffer length, it is silently truncated to the length of the buffer.

Returns the number of bytes successfully written to the socket or FALSE one error. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

> **Megjegyzés: socket_write()** does not necessarily write all bytes from the given buffer. It's valid that, depending on the network buffers etc., only a certain amount of data, even one byte, is written though your buffer is greater. You have to watch out so you don't unintentionally forget to transmit the rest of your data.

> **Megjegyzés:** It is perfectly valid for **socket_write()** to return zero which means no bytes have been written. Be sure to use the === operator to check for FALSE in case of an error.

See also socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_read() and socket_strerror().

socket_writev (PHP 4 >= 4.1.0)

Writes to a file descriptor, fd, using the scatter-gather array defined by iovec_id

bool **socket_writev** (resource socket, resource iovec_id) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

XCVII. String functions

These functions all manipulate strings in various ways. Some more specialized sections can be found in the regular expression and URL handling sections.

For information on how strings behave, especially with regard to usage of single quotes, double quotes, and escape sequences, see the Strings entry in the Types section of the manual.

For even more powerful string handling and manipulating functions take a look at the POSIX regular expression functions and the Perl compatible regular expression functions.

addcslashes (PHP 4)

Quote string with slashes in a C style

string **addcslashes** (string str, string charlist) \linebreak

Returns a string with backslashes before characters that are listed in *charlist* parameter. It escapes \n, \r etc. in C-like style, characters with ASCII code lower than 32 and higher than 126 are converted to octal representation.

Be careful if you choose to escape characters 0, a, b, f, n, r, t and v. They will be converted to \0, \a, \b, \f, \n, \r, \t and \v. In PHP \0 (NULL), \r (carriage return), \n (newline) and \t (tab) are predefined escape sequences, while in C all of these are predefined escape sequences.

charlist like "\0..\37", which would escape all characters with ASCII code between 0 and 31.

Példa 1. addcslashes() example

```
$escaped = addcslashes($not_escaped, "\0..\37!@\177..\377");
```

When you define a sequence of characters in the charlist argument make sure that you know what characters come between the characters that you set as the start and end of the range.

```
echo addcslashes('foo[ ]', 'A..z');
// output:  \f\o\o\[ \]
// All upper and lower-case letters will be escaped
// ... but so will the [\]^_‘ and any tabs, line
// feeds, carriage returns, etc.
```

Also, if the first character in a range has a lower ASCII value than the second character in the range, no range will be constructed. Only the start, end and period characters will be escaped. Use the ord() function to find the ASCII value for a character.

```
echo addcslashes("zoo['.']", 'z..A');
// output:  \zoo['\.']
```

See also stripcslashes(), stripslashes(), htmlspecialchars(), and quotemeta().

addslashes (PHP 3, PHP 4)

Quote string with slashes

string **addslashes** (string str) \linebreak

Returns a string with backslashes before characters that need to be quoted in database queries etc. These characters are single quote (`'`), double quote (`"`), backslash (\) and NUL (the `NULL` byte).

Megjegyzés: magic_quotes_gpc is ON by default.

See also stripslashes(), htmlspecialchars(), and quotemeta().

bin2hex (PHP 3>= 3.0.9, PHP 4)

Convert binary data into hexadecimal representation

string **bin2hex** (string str) \linebreak

Returns an ASCII string containing the hexadecimal representation of *`str`*. The conversion is done byte-wise with the high-nibble first.

See also pack() and unpack().

chop (PHP 3, PHP 4)

Alias of rtrim()

This function is an alias of rtrim().

Megjegyzés: chop() is different than the Perl `chop()` function, which removes the last character in the string.

chr (PHP 3, PHP 4)

Return a specific character

string **chr** (int ascii) \linebreak

Returns a one-character string containing the character specified by *`ascii`*.

Példa 1. chr() example

```
$str .= chr(27); /* add an escape character at the end of $str */

/* Often this is more useful */

$str = sprintf("The string ends in escape: %c", 27);
```

You can find an ASCII-table over here:
http://www.mindspring.com/~jc1/serial/Resources/ASCII.html.

This function complements ord(). See also sprintf() with a format string of `%c`.

chunk_split (PHP 3>= 3.0.6, PHP 4)

Split a string into smaller chunks

string **chunk_split** (string body [, int chunklen [, string end]]) \linebreak

Can be used to split a string into smaller chunks which is useful for e.g. converting base64_encode output to match RFC 2045 semantics. It inserts *`end`* (defaults to "\r\n") every *`chunklen`* characters (defaults to 76). It returns the new string leaving the original string untouched.

Példa 1. chunk_split() example

```
# format $data using RFC 2045 semantics

$new_string = chunk_split(base64_encode($data));
```

See also explode(), split() and wordwrap().

convert_cyr_string (PHP 3>= 3.0.6, PHP 4)

Convert from one Cyrillic character set to another

string **convert_cyr_string** (string str, string from, string to) \linebreak

This function returns the given string converted from one Cyrillic character set to another. The *`from`* and *`to`* arguments are single characters that represent the source and target Cyrillic character sets. The supported types are:

- k - koi8-r
- w - windows-1251

- i - iso8859-5
- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

count_chars (PHP 4)

Return information about characters used in a string

mixed **count_chars** (string string [, int mode]) \linebreak

Counts the number of occurrences of every byte-value (0..255) in *`string`* and returns it in various ways. The optional parameter *`Mode`* default to 0. Depending on *`mode`* **count_chars()** returns one of the following:

- 0 - an array with the byte-value as key and the frequency of every byte as value.
- 1 - same as 0 but only byte-values with a frequency greater than zero are listed.
- 2 - same as 0 but only byte-values with a frequency equal to zero are listed.
- 3 - a string containing all used byte-values is returned.
- 4 - a string containing all not used byte-values is returned.

crc32 (PHP 4 >= 4.0.1)

Calculates the crc32 polynomial of a string

int **crc32** (string str) \linebreak

Generates the cyclic redundancy checksum polynomial of 32-bit lengths of the *`str`*. This is usually used to validate the integrity of data being transmitted.

See also: md5()

crypt (PHP 3, PHP 4)

One-way string encryption (hashing)

string **crypt** (string str [, string salt]) \linebreak

crypt() will return an encrypted string using the standard Unix DES-based encryption algorithm or alternative algorithms that may be available on the system. Arguments are a string to be encrypted and an optional salt string to base the encryption on. See the Unix man page for your crypt function for more information.

If the salt argument is not provided, one will be randomly generated by PHP.

Some operating systems support more than one type of encryption. In fact, sometimes the standard DES-based encryption is replaced by an MD5-based encryption algorithm. The encryption type is triggered by the salt argument. At install time, PHP determines the capabilities of the crypt function and will accept salts for other encryption types. If no salt is provided, PHP will auto-generate a standard two character salt by default, unless the default encryption type on the system is MD5, in which case a random MD5-compatible salt is generated. PHP sets a constant named CRYPT_SALT_LENGTH which tells you whether a regular two character salt applies to your system or the longer twelve character salt is applicable.

If you are using the supplied salt, you should be aware that the salt is generated once. If you are calling this function recursively, this may impact both appearance and security.

The standard DES-based encryption **crypt()** returns the salt as the first two characters of the output. It also only uses the first eight characters of *`str`*, so longer strings that start with the same eight characters will generate the same result (when the same salt is used).

On systems where the crypt() function supports multiple encryption types, the following constants are set to 0 or 1 depending on whether the given type is available:

- CRYPT_STD_DES - Standard DES-based encryption with a two character salt
- CRYPT_EXT_DES - Extended DES-based encryption with a nine character salt
- CRYPT_MD5 - MD5 encryption with a twelve character salt starting with 1
- CRYPT_BLOWFISH - Blowfish encryption with a sixteen character salt starting with 2

Megjegyzés: There is no decrypt function, since **crypt()** uses a one-way algorithm.

Példa 1. crypt() examples

```
<?php
$password = crypt("My1sTpassword"); # let salt be generated

# You should pass the entire results of crypt() as the salt for comparing a
# password, to avoid problems when different hashing algorithms are used. (As
# it says above, standard DES-based password hashing uses a 2-character salt,
# but MD5-based hashing uses 12.)
if (crypt($user_input,$password) == $password) {
   echo "Password verified!";
}
?>
```

See also md5() and the Mcrypt extension.

echo (unknown)

Output one or more strings

echo (string arg1 [, string argn...]) \linebreak

Outputs all parameters.

echo() is not actually a function (it is a language construct) so you are not required to use parentheses with it. In fact, if you want to pass more than one parameter to echo, you must not enclose the parameters within parentheses. It is not possible to use **echo()** in a variable function context, but you can use print() instead.

Példa 1. echo() examples

```
<?php
echo "Hello World";

echo "This spans
multiple lines. The newlines will be
output as well";

echo "This spans\nmultiple lines. The newlines will be\noutput as well.";

echo "Escaping characters is done \"Like this\".";

//You can use variables inside of an echo statement
$foo = "foobar";
$bar = "barbaz";

echo "foo is $foo"; // foo is foobar

// Using single quotes will print the variable name, not the value
echo 'foo is $foo'; // foo is $foo

// If you are not using any other characters, you can just echo variables
echo $foo;          // foobar
echo $foo,$bar;     // foobarbarbaz

echo <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon no extra whitespace!
END;

// Because echo is not a function, following code is invalid.
($some_var) ? echo('true'): echo('false');

// However, the following examples will work:
($some_var) ? print('true'): print('false'); // print is a function
echo $some_var ? 'true': 'false'; // changing the statement around
?>
```

echo() also has a shortcut syntax, where you can immediately follow the opening tag with an equals sign.

```
I have <?=$foo?> foo.
```

Megjegyzés: This short syntax only works with the short_open_tag configuration setting enabled.

See also: print(), printf(), and flush().

explode (PHP 3, PHP 4)

Split a string by string

array **explode** (string separator, string string [, int limit]) \linebreak

Returns an array of strings, each of which is a substring of *`string`* formed by splitting it on boundaries formed by the string *`separator`*. If *`limit`* is set, the returned array will contain a maximum of *`limit`* elements with the last element containing the rest of *`string`*.

If *`separator`* is an empty string (""), **explode()** will return FALSE. If *`separator`* contains a value that is not contained in *`string`*, then **explode()** will return an array containing *`string`*.

Megjegyzés: The *`limit`* parameter was added in PHP 4.0.1

Példa 1. explode() examples

```
$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);

$data = "foo:*:1023:1000::/home/foo:/bin/sh";
list($user,$pass,$uid,$gid,$gecos,$home,$shell) = explode(":",$data);
```

Megjegyzés: Although implode() can for historical reasons accept its parameters in either order, **explode()** cannot. You must ensure that the *`separator`* argument comes before the *`string`* argument.

See also preg_split(), spliti(), split(), and implode().

get_html_translation_table (PHP 4)

Returns the translation table used by htmlspecialchars() and htmlentities()

string **get_html_translation_table** (int table [, int quote_style]) \linebreak

get_html_translation_table() will return the translation table that is used internally for htmlspecialchars() and htmlentities(). There are two new defines (*HTML_ENTITIES*, *HTML_SPECIALCHARS*) that allow you to specify the table you want. And as in the htmlspecialchars() and htmlentities() functions you can optionally specify the quote_style you are working with. The default is ENT_COMPAT mode. See the description of these modes in htmlspecialchars().

Példa 1. Translation Table Example

```
$trans = get_html_translation_table(HTML_ENTITIES);
$str = "Hallo & <Frau> & Krämer";
$encoded = strtr($str, $trans);
```

The `$encoded` variable will now contain: `"Hallo & <Frau> & Kr&auml;mer"`.

The cool thing is using array_flip() to change the direction of the translation.

```
$trans = array_flip($trans);
$original = strtr($encoded, $trans);
```

The content of `$original` would be: "Hallo & <Frau> & Krämer".

See also: htmlspecialchars(), htmlentities(), strtr(), and array_flip().

get_meta_tags (PHP 3>= 3.0.4, PHP 4)

Extracts all meta tag content attributes from a file and returns an array

array **get_meta_tags** (string filename [, int use_include_path]) \linebreak

Opens *filename* and parses it line by line for <meta> tags of the form

Példa 1. Meta Tags Example

```
<meta name="author" content="name">
<meta name="tags" content="php3 documentation">
</head> <!-- parsing stops here -->
```

(pay attention to line endings - PHP uses a native function to parse the input, so a Mac file won't work on Unix).

The value of the name property becomes the key, the value of the content property becomes the value of the returned array, so you can easily use standard array functions to traverse it or access single values. Special characters in the value of the name property are substituted with '_', the rest is converted to lower case.

Setting *use_include_path* to 1 will result in PHP trying to open the file along the standard include path.

hebrev (PHP 3, PHP 4)

Convert logical Hebrew text to visual text

string **hebrev** (string hebrew_text [, int max_chars_per_line]) \linebreak

The optional parameter *max_chars_per_line* indicates maximum number of characters per line will be output. The function tries to avoid breaking words.

See also hebrevc()

hebrevc (PHP 3, PHP 4)

Convert logical Hebrew text to visual text with newline conversion

string **hebrevc** (string hebrew_text [, int max_chars_per_line]) \linebreak

This function is similar to hebrev() with the difference that it converts newlines (\n) to "
\n". The optional parameter *max_chars_per_line* indicates maximum number of characters per line will be output. The function tries to avoid breaking words.

See also hebrev()

htmlentities (PHP 3, PHP 4)

Convert all applicable characters to HTML entities

string **htmlentities** (string string [, int quote_style [, string charset]]) \linebreak

This function is identical to htmlspecialchars() in all ways, except that all characters which have HTML character entity equivalents are translated into these entities. Like htmlspecialchars(), it takes an optional second argument which indicates what should be done with single and double quotes. `ENT_COMPAT` (the default) will only convert double-quotes and leave single-quotes alone. `ENT_QUOTES` will convert both double and single quotes, and `ENT_NOQUOTES` will leave both double and single quotes unconverted.

At present, the ISO-8859-1 character set is used as default. Support for the optional second argument was added in PHP 3.0.17 and PHP 4.0.3.

Like htmlspecialchars(), it takes an optional third argument which defines character set used in conversion. Support for this argument was added in PHP 4.1.0.

There is no reverse of this function. However, you can create one on your own. Here is an example of how to do this.

Példa 1. Reverse of htmlentities()

```
<?php
function unhtmlentities ($string)
{
$trans_tbl = get_html_translation_table (HTML_ENTITIES);
$trans_tbl = array_flip ($trans_tbl);
return strtr ($string, $trans_tbl);
}
?>
```

See also htmlspecialchars() and nl2br().

htmlspecialchars (PHP 3, PHP 4)

Convert special characters to HTML entities

string **htmlspecialchars** (string string [, int quote_style [, string charset]]) \linebreak

Certain characters have special significance in HTML, and should be represented by HTML entities if they are to preserve their meanings. This function returns a string with some of these conversions made; the translations made are those most useful for everyday web programming. If you require all HTML character entities to be translated, use htmlentities() instead.

This function is useful in preventing user-supplied text from containing HTML markup, such as in a message board or guest book application. The optional second argument, quote_style, tells the function what to do with single and double quote characters. The default mode, ENT_COMPAT, is the backwards compatible mode which only translates the double-quote character and leaves the single-quote untranslated. If ENT_QUOTES is set, both single and double quotes are translated and if ENT_NOQUOTES is set neither single nor double quotes are translated.

The translations performed are:

- '&' (ampersand) becomes '&'
- '"' (double quote) becomes '"' when ENT_NOQUOTES is not set.
- "'" (single quote) becomes ''' only when ENT_QUOTES is set.
- '<' (less than) becomes '<'
- '>' (greater than) becomes '>'

Példa 1. htmlspecialchars() example

```
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
```

Note that this function does not translate anything beyond what is listed above. For full entity translation, see htmlentities(). Support for the optional second argument was added in PHP 3.0.17 and PHP 4.0.3.

The third argument defines character set used in conversion. The default character set is ISO-8859-1. Support for this third argument was added in PHP 4.1.0.

See also htmlentities() and nl2br().

implode (PHP 3, PHP 4)

Join array elements with a string

string **implode** (string glue, array pieces) \linebreak

Returns a string containing a string representation of all the array elements in the same order, with the glue string between each element.

Példa 1. implode() example

```
<?php

$array = array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);

print $comma_separated; // lastname,email,phone

?>
```

> **Megjegyzés: implode()** can, for historical reasons, accept its parameters in either order. For consistency with explode(), however, it may be less confusing to use the documented order of arguments.

See also explode(), and split().

join (PHP 3, PHP 4)

Join array elements with a string

string **join** (string glue, array pieces) \linebreak

join() is an alias to implode(), and is identical in every way.

See also explode(), implode(), and split().

levenshtein (PHP 3>= 3.0.17, PHP 4 >= 4.0.1)

Calculate Levenshtein distance between two strings

int **levenshtein** (string str1, string str2) \linebreak int **levenshtein** (string str1, string str2, int cost_ins, int cost_rep, int cost_del) \linebreak int **levenshtein** (string str1, string str2, function cost) \linebreak

This function returns the Levenshtein-Distance between the two argument strings or -1, if one of the argument strings is longer than the limit of 255 characters (255 should be more than enough for name or dictionary comparison, and nobody serious would be doing genetic analysis with PHP).

The Levenshtein distance is defined as the minimal number of characters you have to replace, insert or delete to transform *`str1`* into *`str2`*. The complexity of the algorithm is `O(m*n)`, where `n` and `m` are the length of *`str1`* and *`str2`* (rather good when compared to similar_text(), which is O(max(n,m)**3), but still expensive).

In its simplest form the function will take only the two strings as parameter and will calculate just the number of insert, replace and delete operations needed to transform *`str1`* into *`str2`*.

A second variant will take three additional parameters that define the cost of insert, replace and delete operations. This is more general and adaptive than variant one, but not as efficient.

The third variant (which is not implemented yet) will be the most general and adaptive, but also the slowest alternative. It will call a user-supplied function that will determine the cost for every possible operation.

The user-supplied function will be called with the following arguments:

- operation to apply: ’I’, ’R’ or ’D’
- actual character in string 1
- actual character in string 2
- position in string 1
- position in string 2
- remaining characters in string 1
- remaining characters in string 2

The user-supplied function has to return a positive integer describing the cost for this particular operation, but it may decide to use only some of the supplied arguments.

The user-supplied function approach offers the possibility to take into account the relevance of and/or difference between certain symbols (characters) or even the context those symbols appear in to determine the cost of insert, replace and delete operations, but at the cost of losing all

optimizations done regarding cpu register utilization and cache misses that have been worked into the other two variants.

See also soundex(), similar_text(), and metaphone().

localeconv (PHP 4 >= 4.0.5)

Get numeric formatting information

array **localeconv** (void) \linebreak

Returns an associative array containing localized numeric and monetary formatting information.

localeconv() returns data based upon the current locale as set by setlocale(). The associative array that is returned contains the following fields:

Array element	Description
decimal_point	Decimal point character
thousands_sep	Thousands separator
grouping	Array containing numeric groupings
int_curr_symbol	International currency symbol (i.e. USD)
currency_symbol	Local currency symbol (i.e. $)
mon_decimal_point	Monetary decimal point character
mon_thousands_sep	Monetary thousands separator
mon_grouping	Array containing monetary groupings
positive_sign	Sign for positive values
negative_sign	Sign for negative values
int_frac_digits	International fractional digits
frac_digits	Local fractional digits
p_cs_precedes	TRUE if currency_symbol precedes a positive value, FALSE if it succeeds one
p_sep_by_space	TRUE if a space separates currency_symbol from a positive value, FALSE otherwise
n_cs_precedes	TRUE if currency_symbol precedes a negative value, FALSE if it succeeds one
n_sep_by_space	TRUE if a space separates currency_symbol from a negative value, FALSE otherwise

Array element	Description
p_sign_posn	0 Parentheses surround the quantity and currency_symbol 1 The sign string precedes the quantity and currency_symbol 2 The sign string succeeds the quantity and currency_symbol 3 The sign string immediately precedes the currency_symbol 4 The sign string immediately succeeds the currency_symbol
n_sign_posn	0 Parentheses surround the quantity and currency_symbol 1 The sign string precedes the quantity and currency_symbol 2 The sign string succeeds the quantity and currency_symbol 3 The sign string immediately precedes the currency_symbol 4 The sign string immediately succeeds the currency_symbol

The grouping fields contain arrays that define the way numbers should be grouped. For example, the grouping field for the en_US locale, would contain a 2 item array with the values 3 and 3. The higher the index in the array, the farther left the grouping is. If an array element is equal to CHAR_MAX, no further grouping is done. If an array element is equal to 0, the previous element should be used.

Példa 1. localeconv() example

```
setlocale(LC_ALL, "en_US");

$locale_info = localeconv();

echo "<PRE>\n";
echo "-------------------------------------------\n";
echo "  Monetary information for current locale:  \n";
echo "-------------------------------------------\n\n";

echo "int_curr_symbol:   {$locale_info["int_curr_symbol"]}\n";
echo "currency_symbol:   {$locale_info["currency_symbol"]}\n";
```

```
echo "mon_decimal_point: {$locale_info["mon_decimal_point"]}\n";
echo "mon_thousands_sep: {$locale_info["mon_thousands_sep"]}\n";
echo "positive_sign:     {$locale_info["positive_sign"]}\n";
echo "negative_sign:     {$locale_info["negative_sign"]}\n";
echo "int_frac_digits:   {$locale_info["int_frac_digits"]}\n";
echo "frac_digits:       {$locale_info["frac_digits"]}\n";
echo "p_cs_precedes:     {$locale_info["p_cs_precedes"]}\n";
echo "p_sep_by_space:    {$locale_info["p_sep_by_space"]}\n";
echo "n_cs_precedes:     {$locale_info["n_cs_precedes"]}\n";
echo "n_sep_by_space:    {$locale_info["n_sep_by_space"]}\n";
echo "p_sign_posn:       {$locale_info["p_sign_posn"]}\n";
echo "n_sign_posn:       {$locale_info["n_sign_posn"]}\n";
echo "</PRE>\n";
```

The constant CHAR_MAX is also defined for the use mentioned above.

See also: setlocale().

ltrim (PHP 3, PHP 4)

Strip whitespace from the beginning of a string

string **ltrim** (string str [, string charlist]) \linebreak

> **Megjegyzés:** The second parameter was added in PHP 4.1.0

This function returns a string with whitespace stripped from the beginning of *`str`*. Without the second parameter, **ltrim()** will strip these characters:

- " " (ASCII `32` (`0x20`)), an ordinary space.
- "\t" (ASCII `9` (`0x09`)), a tab.
- "\n" (ASCII `10` (`0x0A`)), a new line (line feed).
- "\r" (ASCII `13` (`0x0D`)), a carriage return.
- "\0" (ASCII `0` (`0x00`)), the `NUL`-byte.
- "\x0B" (ASCII `11` (`0x0B`)), a vertical tab.

You can also specify the characters you want to strip, by means of the *`charlist`* parameter. Simply list all characters that you want to be stripped. With `..` you can specify a range of characters.

Példa 1. Usage example of ltrim()

```
<?php

$text = "\t\tThese are a few words :) ...  ";
```

```
$trimmed = ltrim($text);
// $trimmed = "These are a few words :) ...  "
$trimmed = ltrim($text," \t.");
// $trimmed = "These are a few words :) ...  "
$clean = ltrim($binary,"\0x00..\0x1F");
// trim the ASCII control characters at the beginning of $binary
// (from 0 to 31 inclusive)

?>
```

See also trim() and rtrim().

md5_file (PHP 4 >= 4.2.0)

Calculates the md5 hash of a given filename

string **md5_file** (string filename) \linebreak

Calculates the MD5 hash of the specified *`filename`* using the RSA Data Security, Inc. MD5 Message-Digest Algorithm (http://www.faqs.org/rfcs/rfc1321.html), and returns that hash.

This function has the same purpose of the command line utility md5sum.

See also: md5() and crc32()

md5 (PHP 3, PHP 4)

Calculate the md5 hash of a string

string **md5** (string str) \linebreak

Calculates the MD5 hash of *`str`* using the RSA Data Security, Inc. MD5 Message-Digest Algorithm (http://www.faqs.org/rfcs/rfc1321.html), and returns that hash. The hash is a 32-character hexadecimal number.

See also: crc32() and md5_file()

metaphone (PHP 4)

Calculate the metaphone key of a string

string **metaphone** (string str) \linebreak

Calculates the metaphone key of *`str`*.

Similar to soundex() metaphone creates the same key for similar sounding words. It's more accurate than soundex() as it knows the basic rules of English pronunciation. The metaphone generated keys are of variable length.

Metaphone was developed by Lawrence Philips <lphilips@verity.com>. It is described in ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995].

nl_langinfo (PHP 4 >= 4.1.0)

Query language and locale information

string **nl_langinfo** (int item) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

nl2br (PHP 3, PHP 4)

Inserts HTML line breaks before all newlines in a string

string **nl2br** (string string) \linebreak

Returns *`string`* with '<br />' inserted before all newlines.

> **Megjegyzés:** Starting with PHP 4.0.5, **nl2br()** is now XHTML compliant. All versions before 4.0.5 will return *`string`* with '<br>' inserted before newlines instead of '<br />'.

See also htmlspecialchars(), htmlentities() and wordwrap().

ord (PHP 3, PHP 4)

Return ASCII value of character

int **ord** (string string) \linebreak

Returns the ASCII value of the first character of *`string`*. This function complements chr().

Példa 1. ord() example

```
if (ord($str) == 10) {
    echo "The first character of \$str is a line feed.\n";
}
```

You can find an ASCII-table over here:
http://www.mindspring.com/~jc1/serial/Resources/ASCII.html.

See also chr().

parse_str (PHP 3, PHP 4)

Parses the string into variables

void **parse_str** (string str [, array arr]) \linebreak

Parses *str* as if it were the query string passed via an URL and sets variables in the current scope. If the second parameter *arr* is present, variables are stored in this variable as an array elements instead.

Megjegyzés: Support for the optional second parameter was added in PHP 4.0.3.

Példa 1. Using parse_str()

```
$str = "first=value&second[]=this+works&second[]=another";
parse_str($str);
echo $first;     /* prints "value" */
echo $second[0]; /* prints "this works" */
echo $second[1]; /* prints "another" */
```

See also set_magic_quotes_runtime() and urldecode().

print (unknown)

Output a string

print (string arg) \linebreak

Outputs *arg*. Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

print() is not actually a real function (it is a language construct) so you are not required to use parentheses with it.

Példa 1. print() examples

```
<?php
print("Hello World");

print "print() also works without parentheses.";

print "This spans
multiple lines. The newlines will be
output as well";

print "This spans\nmultiple lines. The newlines will be\noutput as well.";

print "escaping characters is done \"Like this\".";

// You can use variables inside of an print statement
$foo = "foobar";
$bar = "barbaz";

print "foo is $foo"; // foo is foobar

// Using single quotes will print the variable name, not the value
print 'foo is $foo'; // foo is $foo

// If you are not using any other characters, you can just print variables
print $foo;          // foobar

print <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon no extra whitespace!
END;
?>
```

See also echo(), printf(), and flush().

printf (PHP 3, PHP 4)

Output a formatted string

void **printf** (string format [, mixed args]) \linebreak

Produces output according to *format*, which is described in the documentation for sprintf().

See also: print(), sprintf(), sscanf(), fscanf(), and flush().

quoted_printable_decode (PHP 3>= 3.0.6, PHP 4)

Convert a quoted-printable string to an 8 bit string

string **quoted_printable_decode** (string str) \linebreak

This function returns an 8-bit binary string corresponding to the decoded quoted printable string. This function is similar to imap_qprint(), except this one does not require the IMAP module to work.

quotemeta (PHP 3, PHP 4)

Quote meta characters

string **quotemeta** (string str) \linebreak

Returns a version of str with a backslash character (\) before every character that is among these:

```
. \\ + * ? [ ^ ] ( $ )
```

See also addslashes(), htmlentities(), htmlspecialchars(), nl2br(), and stripslashes().

rtrim (PHP 3, PHP 4)

Strip whitespace from the end of a string

string **rtrim** (string str [, string charlist]) \linebreak

Megjegyzés: The second parameter was added in PHP 4.1.0

This function returns a string with whitespace stripped from the end of *`str`*. Without the second parameter, **rtrim()** will strip these characters:

- " " (ASCII `32` (`0x20`)), an ordinary space.
- "\t" (ASCII `9` (`0x09`)), a tab.
- "\n" (ASCII `10` (`0x0A`)), a new line (line feed).
- "\r" (ASCII `13` (`0x0D`)), a carriage return.
- "\0" (ASCII `0` (`0x00`)), the `NUL`-byte.
- "\x0B" (ASCII `11` (`0x0B`)), a vertical tab.

You can also specify the characters you want to strip, by means of the *`charlist`* parameter. Simply list all characters that you want to be stripped. With `..` you can specify a range of characters.

Példa 1. Usage example of rtrim()

```
<?php

$text = "\t\tThese are a few words :) ...  ";
$trimmed = rtrim($text);
// $trimmed = "\t\tThese are a few words :) ..."
$trimmed = rtrim($text," \t.");
// $trimmed = "\t\tThese are a few words :)"
$clean = rtrim($binary,"\0x00..\0x1F");
// trim the ASCII control characters at the end of $binary
// (from 0 to 31 inclusive)

?>
```

See also trim() and ltrim().

setlocale (PHP 3, PHP 4)

Set locale information

string **setlocale** (mixed category, string locale) \linebreak

`Category` is a named constant (or string) specifying the category of the functions affected by the locale setting:

- LC_ALL for all of the below
- LC_COLLATE for string comparison, see strcoll()
- LC_CTYPE for character classification and conversion, for example strtoupper()
- LC_MONETARY for localeconv()
- LC_NUMERIC for decimal separator (See also: localeconv())
- LC_TIME for date and time formatting with strftime()

If *`locale`* is the empty string `""`, the locale names will be set from the values of environment variables with the same names as the above categories, or from "LANG".

If locale is zero or `"0"`, the locale setting is not affected, only the current setting is returned.

Setlocale returns the new current locale, or `FALSE` if the locale functionality is not implemented in the platform, the specified locale does not exist or the category name is invalid. An invalid category name also causes a warning message.

Példa 1. setlocale() Examples

```
<?php
    /* Set locale to Dutch */
    setlocale (LC_ALL, 'nl_NL');

    /* Output: vrijdag 22 december 1978 */
    echo strftime ("%A %e %B %Y", mktime (0, 0, 0, 12, 22, 1978));
?>
```

similar_text (PHP 3>= 3.0.7, PHP 4)

Calculate the similarity between two strings

int **similar_text** (string first, string second [, float percent]) \linebreak

This calculates the similarity between two strings as described in Oliver [1993]. Note that this implementation does not use a stack as in Oliver's pseudo code, but recursive calls which may or may not speed up the whole process. Note also that the complexity of this algorithm is O(N**3) where N is the length of the longest string.

By passing a reference as third argument, **similar_text()** will calculate the similarity in percent for you. It returns the number of matching chars in both strings.

soundex (PHP 3, PHP 4)

Calculate the soundex key of a string

string **soundex** (string str) \linebreak

Calculates the soundex key of *`str`*.

Soundex keys have the property that words pronounced similarly produce the same soundex key, and can thus be used to simplify searches in databases where you know the pronunciation but not the spelling. This soundex function returns a string 4 characters long, starting with a letter.

This particular soundex function is one described by Donald Knuth in "The Art Of Computer Programming, vol. 3: Sorting And Searching", Addison-Wesley (1973), pp. 391-392.

Példa 1. Soundex Examples

```
soundex("Euler") == soundex("Ellery") == 'E460';
soundex("Gauss") == soundex("Ghosh") == 'G200';
soundex("Hilbert") == soundex("Heilbronn") == 'H416';
soundex("Knuth") == soundex("Kant") == 'K530';
soundex("Lloyd") == soundex("Ladd") == 'L300';
```

```
soundex("Lukasiewicz") == soundex("Lissajous") == 'L222';
```

See also levenshtein(), metaphone(), and similar_text().

sprintf (PHP 3, PHP 4)

Return a formatted string

string **sprintf** (string format [, mixed args]) \linebreak

Returns a string produced according to the formatting string *`format`*.

The format string is composed of zero or more directives: ordinary characters (excluding `%`) that are copied directly to the result, and *conversion specifications*, each of which results in fetching its own parameter. This applies to both **sprintf()** and printf().

Each conversion specification consists of a percent sign (`%`), followed by one or more of these elements, in order:

1. An optional *padding specifier* that says what character will be used for padding the results to the right string size. This may be a space character or a `0` (zero character). The default is to pad with spaces. An alternate padding character can be specified by prefixing it with a single quote (`'`). See the examples below.
2. An optional *alignment specifier* that says if the result should be left-justified or right-justified. The default is right-justified; a `-` character here will make it left-justified.
3. An optional number, a *width specifier* that says how many characters (minimum) this conversion should result in.
4. An optional *precision specifier* that says how many decimal digits should be displayed for floating-point numbers. This option has no effect for other types than float. (Another function useful for formatting numbers is number_format().)
5. A *type specifier* that says what type the argument data should be treated as. Possible types:
   `%` - a literal percent character. No argument is required.
   `b` - the argument is treated as an integer, and presented as a binary number.
   `c` - the argument is treated as an integer, and presented as the character with that ASCII value.
   `d` - the argument is treated as an integer, and presented as a (signed) decimal number.
   `u` - the argument is treated as an integer, and presented as an unsigned decimal number.
   `f` - the argument is treated as a float, and presented as a floating-point number.
   `o` - the argument is treated as an integer, and presented as an octal number.
   `s` - the argument is treated as and presented as a string.
   `x` - the argument is treated as an integer and presented as a hexadecimal number (with lowercase letters).
   `X` - the argument is treated as an integer and presented as a hexadecimal number (with uppercase letters).

As of PHP version 4.0.6 the format string supports argument numbering/swapping. Here is an example:

Példa 1. Argument swapping

```
$format = "There are %d monkeys in the %s";
printf($format,$num,$location);
```

This might output, "There are 5 monkeys in the tree". But imagine we are creating a format string in a separate file, commonly because we would like to internationalize it and we rewrite it as:

Példa 2. Argument swapping

```
$format = "The %s contains %d monkeys";
printf($format,$num,$location);
```

We now have a problem. The order of the placeholders in the format string does not match the order of the arguments in the code. We would like to leave the code as is and simply indicate in the format string which arguments the placeholders refer to. We would write the format string like this instead:

Példa 3. Argument swapping

```
$format = "The %2\$s contains %1\$d monkeys";
printf($format,$num,$location);
```

An added benefit here is that you can repeat the placeholders without adding more arguments in the code. For example:

Példa 4. Argument swapping

```
$format = "The %2\$s contains %1\$d monkeys.
           That’s a nice %2\$s full of %1\$d monkeys.";
printf($format, $num, $location);
```

See also: printf(), sscanf(), fscanf(), and number_format().

Példa 5. sprintf(): zero-padded integers

```
$isodate = sprintf("%04d-%02d-%02d", $year, $month, $day);
```

Példa 6. sprintf(): formatting currency

```
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money will output "123.1";
$formatted = sprintf("%01.2f", $money);
// echo $formatted will output "123.10"
```

sscanf (PHP 4 >= 4.0.1)

Parses input from a string according to a format

mixed **sscanf** (string str, string format [, string var1]) \linebreak

The function **sscanf()** is the input analog of printf(). **sscanf()** reads from the string *`str`* and interprets it according to the specified *`format`*. If only two parameters were passed to this function, the values parsed will be returned as an array.

Any whitespace in the format string matches any whitespace in the input string. This means that even a tab \n in the format string can match a single space character in the input string.

Példa 1. sscanf() Example

```
// getting the serial number
$serial = sscanf("SN/2350001","SN/%d");
// and the date of manufacturing
$mandate = "January 01 2000";
list($month, $day, $year) = sscanf($mandate,"%s %d %d");
echo "Item $serial was manufactured on: $year-".substr($month,0,3)."-$day\n";
```

If optional parameters are passed, the function will return the number of assigned values. The optional parameters must be passed by reference.

Példa 2. sscanf() - using optional parameters

```
// get author info and generate DocBook entry
$auth = "24\tLewis Carroll";
$n = sscanf($auth,"%d\t%s %s", &$id, &$first, &$last);
echo "<author id='$id'>
    <firstname>$first</firstname>
    <surname>$last</surname>
</author>\n";
```

See also: fscanf(), printf(), and sprintf().

str_pad (PHP 4 >= 4.0.1)

Pad a string to a certain length with another string

string **str_pad** (string input, int pad_length [, string pad_string [, int pad_type]]) \linebreak

This functions returns the *input* string padded on the left, the right, or both sides to the specified padding length. If the optional argument *pad_string* is not supplied, the *input* is padded with spaces, otherwise it is padded with characters from *pad_string* up to the limit.

Optional argument *pad_type* can be STR_PAD_RIGHT, STR_PAD_LEFT, or STR_PAD_BOTH. If *pad_type* is not specified it is assumed to be STR_PAD_RIGHT.

If the value of *pad_ length* is negative or less than the length of the input string, no padding takes place.

Példa 1. str_pad() example

```
$input = "Alien";
print str_pad($input, 10);                      // produces "Alien     "
print str_pad($input, 10, "-=", STR_PAD_LEFT);  // produces "-=-=-Alien"
print str_pad($input, 10, "_", STR_PAD_BOTH);   // produces "__Alien___"
```

str_repeat (PHP 4)

Repeat a string

string **str_repeat** (string input, int multiplier) \linebreak

Returns *input_str* repeated *multiplier* times. *multiplier* has to be greater than or equal to 0. If the *multiplier* is set to 0, the function will return an empty string.

Példa 1. str_repeat() example

```
echo str_repeat("-=", 10);
```

This will output "-=-=-=-=-=-=-=-=-=-=".

See also: for, str_pad(), and substr_count().

str_replace (PHP 3>= 3.0.6, PHP 4)

Replace all occurrences of the search string with the replacement string

mixed **str_replace** (mixed search, mixed replace, mixed subject) \linebreak

This function returns a string or an array with all occurences of *search* in *subject* replaced with the given *replace* value. If you don't need fancy replacing rules, you should always use this function instead of ereg_replace() or preg_replace().

In PHP 4.0.5 and later, every parameter to **str_replace()** can be an array.

If *subject* is an array, then the search and replace is performed with every entry of *subject*, and the return value is an array as well.

If *search* and *replace* are arrays, then **str_replace()** takes a value from each array and uses them to do search and replace on *subject*. If *replace* has fewer values than *search*, then an empty string is used for the rest of replacement values. If *search* is an array and *replace* is a string; then this replacement string is used for every value of *search*.

Példa 1. str_replace() example

```
$bodytag = str_replace("%body%", "black", "<body text=%body%>");
```

This function is binary safe.

Megjegyzés: str_replace() was added in PHP 3.0.6, but was buggy up until PHP 3.0.8.

See also ereg_replace(), preg_replace(), and strtr().

str_rot13 (PHP 4 >= 4.2.0)

Perform the rot13 transform on a string

string **str_rot13** (string str) \linebreak

This function performs the ROT13 encoding on the *str* argument and returns the resulting string. The ROT13 encoding simply shifts every letter by 13 places in the alphabet while leaving non-alpha characters untouched. Encoding and decoding are done by the same function, passing an encoded string as argument will return the original version.

strcasecmp (PHP 3>= 3.0.2, PHP 4)

Binary safe case-insensitive string comparison

int **strcasecmp** (string str1, string str2) \linebreak

Returns < 0 if *`str1`* is less than *`str2`*; > 0 if *`str1`* is greater than *`str2`*, and 0 if they are equal.

Példa 1. strcasecmp() example

```
$var1 = "Hello";
$var2 = "hello";
if (!strcasecmp($var1, $var2)) {
    echo '$var1 is equal to $var2 in a case-insensitive string comparison';
}
```

See also ereg(), strcmp(), substr(), stristr(), strncasecmp(), and strstr().

strchr (PHP 3, PHP 4)

Find the first occurrence of a character

string **strchr** (string haystack, string needle) \linebreak

This function is an alias for strstr(), and is identical in every way.

strcmp (PHP 3, PHP 4)

Binary safe string comparison

int **strcmp** (string str1, string str2) \linebreak

Returns < 0 if *`str1`* is less than *`str2`*; > 0 if *`str1`* is greater than *`str2`*, and 0 if they are equal.

Note that this comparison is case sensitive.

See also ereg(), strcasecmp(), substr(), stristr(), strncasecmp(), strncmp(), and strstr().

strcoll (PHP 4 >= 4.0.5)

Locale based string comparison

int **strcoll** (string str1, string str2) \linebreak

Returns < 0 if *`str1`* is less than *`str2`*; > 0 if *`str1`* is greater than *`str2`*, and 0 if they are equal. **strcoll()** uses the current locale for doing the comparisons. If the current locale is C or POSIX, this function is equivalent to strcmp().

Note that this comparison is case sensitive, and unlike strcmp() this function is not binary safe.

See also ereg(), strcmp(), strcasecmp(), substr(), stristr(), strncasecmp(), strncmp(), strstr(), and setlocale().

strcspn (PHP 3>= 3.0.3, PHP 4)

Find length of initial segment not matching mask

int **strcspn** (string str1, string str2) \linebreak

Returns the length of the initial segment of *`str1`* which does *not* contain any of the characters in *`str2`*.

See also strspn().

strip_tags (PHP 3>= 3.0.8, PHP 4)

Strip HTML and PHP tags from a string

string **strip_tags** (string str [, string allowable_tags]) \linebreak

This function tries to return a string with all HTML and PHP tags stripped from a given *`str`*. It errors on the side of caution in case of incomplete or bogus tags. It uses the same tag stripping state machine as the fgetss() function.

You can use the optional second parameter to specify tags which should not be stripped.

Megjegyzés: *`allowable_tags`* was added in PHP 3.0.13 and PHP 4.0b3.

Példa 1. strip_tags() example

```
$string = strip_tags($string, '<a><b><i><u>');
```

Figyelem

This function does not modify any attributes on the tags that you allow using *`allowable_tags`*, including the `style` and `onmouseover` attributes that a mischievous user may abuse when posting text that will be shown to other users.

stripcslashes (PHP 4)

Un-quote string quoted with addcslashes()

string **stripcslashes** (string str) \linebreak

Returns a string with backslashes stripped off. Recognizes C-like `\n`, `\r` ..., octal and hexadecimal representation.

See also addcslashes().

stripslashes (PHP 3, PHP 4)

Un-quote string quoted with addslashes()

string **stripslashes** (string str) \linebreak

Returns a string with backslashes stripped off. (`\'` becomes `'` and so on.) Double backslashes are made into a single backslash.

See also addslashes().

stristr (PHP 3>= 3.0.6, PHP 4)

Case-insensitive strstr()

string **stristr** (string haystack, string needle) \linebreak

Returns all of *`haystack`* from the first occurrence of *`needle`* to the end. *`needle`* and *`haystack`* are examined in a case-insensitive manner.

If *`needle`* is not found, returns FALSE.

If *`needle`* is not a string, it is converted to an integer and applied as the ordinal value of a character.

See also strchr(), strrchr(), substr(), and ereg().

strlen (PHP 3, PHP 4)

Get string length

int **strlen** (string str) \linebreak

Returns the length of *`string`*.

strnatcasecmp (PHP 4)

Case insensitive string comparisons using a "natural order" algorithm

int **strnatcasecmp** (string str1, string str2) \linebreak

This function implements a comparison algorithm that orders alphanumeric strings in the way a human being would. The behaviour of this function is similar to strnatcmp(), except that the comparison is not case sensitive. For more infomation see: Martin Pool's Natural Order String Comparison (http://naturalordersort.org/) page.

Similar to other string comparison functions, this one returns < 0 if *`str1`* is less than *`str2`*; > 0 if *`str1`* is greater than *`str2`*, and 0 if they are equal.

See also ereg(), strcasecmp(), substr(), stristr(), strcmp(), strncmp(), strncasecmp(), strnatcmp(), and strstr().

strnatcmp (PHP 4)

String comparisons using a "natural order" algorithm

int **strnatcmp** (string str1, string str2) \linebreak

This function implements a comparison algorithm that orders alphanumeric strings in the way a human being would, this is described as a "natural ordering". An example of the difference between this algorithm and the regular computer string sorting algorithms (used in strcmp()) can be seen below:

```
$arr1 = $arr2 = array("img12.png","img10.png","img2.png","img1.png");
echo "Standard string comparison\n";
usort($arr1,"strcmp");
print_r($arr1);
echo "\nNatural order string comparison\n";
usort($arr2,"strnatcmp");
print_r($arr2);
```

The code above will generate the following output:

```
Standard string comparison
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Natural order string comparison
Array
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

For more information see: Martin Pool's Natural Order String Comparison (http://naturalordersort.org/) page.

Similar to other string comparison functions, this one returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

Note that this comparison is case sensitive.

See also ereg(), strcasecmp(), substr(), stristr(), strcmp(), strncmp(), strncasecmp(), strnatcasecmp(), strstr(), natsort() and natcasesort().

strncasecmp (PHP 4 >= 4.0.2)

Binary safe case-insensitive string comparison of the first n characters

int **strncasecmp** (string str1, string str2, int len) \linebreak

This function is similar to strcasecmp(), with the difference that you can specify the (upper limit of the) number of characters (*len*) from each string to be used in the comparison. If any of the strings is shorter than *len*, then the length of that string will be used for the comparison.

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

See also ereg(), strcasecmp(), strcmp(), substr(), stristr(), and strstr().

strncmp (PHP 4)

Binary safe string comparison of the first n characters

int **strncmp** (string str1, string str2, int len) \linebreak

This function is similar to strcmp(), with the difference that you can specify the (upper limit of the) number of characters (*len*) from each string to be used in the comparison. If any of the strings is shorter than *len*, then the length of that string will be used for the comparison.

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

Note that this comparison is case sensitive.

See also ereg(), strncasecmp(), strcasecmp(), substr(), stristr(), strcmp(), and strstr().

strpos (PHP 3, PHP 4)

Find position of first occurrence of a string

int **strpos** (string haystack, string needle [, int offset]) \linebreak

Returns the numeric position of the first occurrence of *needle* in the *haystack* string. Unlike the strrpos(), this function can take a full string as the *needle* parameter and the entire string will be used.

If *needle* is not found, returns FALSE.

> **Megjegyzés:** It is easy to mistake the return values for "character found at position 0" and "character not found". Here's how to detect the difference:

```
// in PHP 4.0b3 and newer:
$pos = strpos($mystring, "b");
if ($pos === false) { // note: three equal signs
    // not found...
}

// in versions older than 4.0b3:
$pos = strpos($mystring, "b");
if (!is_integer($pos)) {
    // not found...
}
```

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

The optional *offset* parameter allows you to specify which character in *haystack* to start searching. The position returned is still relative to the the beginning of *haystack*.

See also strrpos(), strrchr(), substr(), stristr(), and strstr().

strrchr (PHP 3, PHP 4)

Find the last occurrence of a character in a string

string **strrchr** (string haystack, string needle) \linebreak

This function returns the portion of *haystack* which starts at the last occurrence of *needle* and goes until the end of *haystack*.

Returns FALSE if *needle* is not found.

If *needle* contains more than one character, the first is used.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

Példa 1. strrchr() example

```
// get last directory in $PATH
$dir = substr(strrchr($PATH, ":"), 1);
```

```
// get everything after last newline
$text = "Line 1\nLine 2\nLine 3";
$last = substr(strrchr($text, 10), 1 );
```

See also strchr(), substr(), stristr(), and strstr().

strrev (PHP 3, PHP 4)

Reverse a string

string **strrev** (string string) \linebreak

Returns *`string`*, reversed.

Példa 1. Reversing a string with strrev()

```
<php
echo strrev("Hello world!"); // outputs "!dlrow olleH"
?>
```

strrpos (PHP 3, PHP 4)

Find position of last occurrence of a char in a string

int **strrpos** (string haystack, char needle) \linebreak

Returns the numeric position of the last occurrence of *`needle`* in the *`haystack`* string. Note that the needle in this case can only be a single character. If a string is passed as the needle, then only the first character of that string will be used.

If *`needle`* is not found, returns `FALSE`.

> **Megjegyzés:** It is easy to mistake the return values for "character found at position 0" and "character not found". Here's how to detect the difference:

```
// in PHP 4.0b3 and newer:
$pos = strrpos($mystring, "b");
if ($pos === false) { // note: three equal signs
    // not found...
}

// in versions older than 4.0b3:
$pos = strrpos($mystring, "b");
if (is_string($pos) && !$pos) {
```

```
    // not found...
}
```

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

See also strpos(), strrchr(), substr(), stristr(), and strstr().

strspn (PHP 3>= 3.0.3, PHP 4)

Find length of initial segment matching mask

int **strspn** (string str1, string str2) \linebreak

Returns the length of the initial segment of *str1* which consists entirely of characters in *str2*.

The line of code:

```
$var = strspn("42 is the answer, what is the question ...", "1234567890");
```

will assign 2 to $var, because the string "42" will be the longest segment containing characters from "1234567890".

See also strcspn().

strstr (PHP 3, PHP 4)

Find first occurrence of a string

string **strstr** (string haystack, string needle) \linebreak

Returns part of *haystack* string from the first occurrence of *needle* to the end of *haystack*.

If *needle* is not found, returns FALSE.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

Megjegyzés: This function is case-sensitive. For case-insensitive searches, use stristr().

Példa 1. strstr() example

```
$email = 'user@example.com';
$domain = strstr($email, '@');
print $domain; // prints @example.com
```

See also ereg(), preg_match(), strchr(), stristr(), strpos(), strrchr(), and substr().

strtok (PHP 3, PHP 4)

Tokenize string

string **strtok** (string arg1, string arg2) \linebreak

strtok() splits a string (*arg1*) into smaller strings (tokens), with each token being delimited by any character from *arg2*. That is, if you have a string like "This is an example string" you could tokenize this string into its individual words by using the space character as the token.

Példa 1. strtok() example

```
$string = "This is\tan example\nstring";
/* Use tab and newline as tokenizing characters as well  */
$tok = strtok($string," \n\t");
while ($tok) {
    echo "Word=$tok<br>";
    $tok = strtok(" \n\t");
}
```

Note that only the first call to strtok uses the string argument. Every subsequent call to strtok only needs the token to use, as it keeps track of where it is in the current string. To start over, or to tokenize a new string you simply call strtok with the string argument again to initialize it. Note that you may put multiple tokens in the token parameter. The string will be tokenized when any one of the characters in the argument are found.

The behavior when an empty part was found changed with PHP 4.1.0. The old behavior returned an empty string, while the new, correct, behavior simply skips the part of the string:

Példa 2. Old strtok() behavior

```
    $first_token  = strtok('/something', '/');
    $second_token = strtok('/');
    var_dump ($first_token, $second_token);

/* Output:
```

```
    string(0) ""
    string(9) "something"
*/
```

Példa 3. New strtok() behavior

```
    $first_token  = strtok('/something', '/');
    $second_token = strtok('/');
    var_dump ($first_token, $second_token);

/* Output:
    string(9) "something"
    bool(false)
*/
```

Also be careful that your tokens may be equal to "0". This evaluates to `FALSE` in conditional expressions.

See also split() and explode().

strtolower (PHP 3, PHP 4)

Make a string lowercase

string **strtolower** (string str) \linebreak

Returns *`string`* with all alphabetic characters converted to lowercase.

Note that 'alphabetic' is determined by the current locale. This means that in i.e. the default "C" locale, characters such as umlaut-A (Ä) will not be converted.

Példa 1. strtolower() example

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtolower($str);
print $str; # Prints mary had a little lamb and she loved it so
```

See also strtoupper(), ucfirst(), and ucwords().

strtoupper (PHP 3, PHP 4)

Make a string uppercase

string **strtoupper** (string string) \linebreak

Returns *`string`* with all alphabetic characters converted to uppercase.

Note that 'alphabetic' is determined by the current locale. For instance, in the default "C" locale characters such as umlaut-a (ä) will not be converted.

Példa 1. strtoupper() example

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtoupper($str);
print $str; # Prints MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
```

See also strtolower(), ucfirst(), and ucwords().

strtr (PHP 3, PHP 4)

Translate certain characters

string **strtr** (string str, string from, string to) \linebreak string **strtr** (string str, array replace_pairs) \linebreak

This function returns a copy of *`str`*, translating all occurrences of each character in *`from`* to the corresponding character in *`to`* and returning the result.

If *`from`* and *`to`* are different lengths, the extra characters in the longer of the two are ignored.

Példa 1. strtr() example

```
$addr = strtr($addr, "äĺö", "aao");
```

strtr() can be called with only two arguments. If called with two arguments it behaves in a new way: *`from`* then has to be an array that contains string -> string pairs that will be replaced in the source string. **strtr()** will always look for the longest possible match first and will *NOT* try to replace stuff that it has already worked on.

Examples:

```
$trans = array("hello" => "hi", "hi" => "hello");
echo strtr("hi all, I said hello", $trans) . "\n";
```

This will show: "hello all, I said hi",

> **Megjegyzés:** This optional *to* and *from* parameters were added in PHP 4.0.0

See also ereg_replace().

substr_count (PHP 4)

Count the number of substring occurrences

int **substr_count** (string haystack, string needle) \linebreak

substr_count() returns the number of times the *needle* substring occurs in the *haystack* string.

Példa 1. substr_count() example

```
print substr_count("This is a test", "is"); // prints out 2
```

substr_replace (PHP 4)

Replace text within a portion of a string

string **substr_replace** (string string, string replacement, int start [, int length]) \linebreak

substr_replace() replaces a copy of *string* delimited by the *start* and (optionally) *length* parameters with the string given in *replacement*. The result is returned.

If *start* is positive, the replacing will begin at the *start*'th offset into *string*.

If *start* is negative, the replacing will begin at the *start*'th character from the end of *string*.

If *length* is given and is positive, it represents the length of the portion of *string* which is to be replaced. If it is negative, it represents the number of characters from the end of *string* at which to stop replacing. If it is not given, then it will default to strlen(*string*); i.e. end the replacing at the end of *string*.

Példa 1. substr_replace() example

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr>\n";

/* These two examples replace all of $var with 'bob'. */
echo substr_replace($var, 'bob', 0) . "<br>\n";
```

```
echo substr_replace($var, 'bob', 0, strlen($var)) . "<br>\n";

/* Insert 'bob' right at the beginning of $var. */
echo substr_replace($var, 'bob', 0, 0) . "<br>\n";

/* These next two replace 'MNRPQR' in $var with 'bob'. */
echo substr_replace($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace($var, 'bob', -7, -1) . "<br>\n";

/* Delete 'MNRPQR' from $var. */
echo substr_replace($var, '', 10, -1) . "<br>\n";
?>
```

See also str_replace() and substr().

substr (PHP 3, PHP 4)

Return part of a string

string **substr** (string string, int start [, int length]) \linebreak

Substr returns the portion of *`string`* specified by the *`start`* and *`length`* parameters.

If *`start`* is positive, the returned string will start at the *`start`*'th position in *`string`*, counting from zero. For instance, in the string '`abcdef`', the character at position `0` is '`a`', the character at position `2` is '`c`', and so forth.

Példa 1. Basic substr() usage

```
<?php
$rest = substr("abcdef", 1);    // returns "bcdef"
$rest = substr("abcdef", 1, 3); // returns "bcd"
$rest = substr("abcdef", 0, 4); // returns "abcd"
$rest = substr("abcdef", 0, 8); // returns "abcdef"

// Accessing via curly braces is another option
$string = 'abcdef';
echo $string{0};                // returns a
echo $string{3};                // returns d
?>
```

If *`start`* is negative, the returned string will start at the *`start`*'th character from the end of *`string`*.

Példa 2. Using a negative *start*

```
<?php
$rest = substr("abcdef", -1);    // returns "f"
$rest = substr("abcdef", -2);    // returns "ef"
$rest = substr("abcdef", -3, 1); // returns "d"
?>
```

If *length* is given and is positive, the string returned will contain at most *length* characters beginning from *start* (depending on the length of *string*. If *string* is less than *start* characters long, FALSE will be returned.

If *length* is given and is negative, then that many characters will be omitted from the end of *string* (after the start position has been calculated when a *start* is negative). If *start* denotes a position beyond this truncation, an empty string will be returned.

Példa 3. Using a negative *length*

```
<?php
$rest = substr("abcdef", 0, -1);  // returns "abcde"
$rest = substr("abcdef", 2, -1);  // returns "cde"
$rest = substr("abcdef", 4, -4);  // returns ""
$rest = substr("abcdef", -3, -1); // returns "de"
?>
```

See also strrchr() and ereg().

trim (PHP 3, PHP 4)

Strip whitespace from the beginning and end of a string

string **trim** (string str [, string charlist]) \linebreak

> **Megjegyzés:** The optional *charlist* parameter was added in PHP 4.1.0

This function returns a string with whitespace stripped from the beginning and end of *str*. Without the second parameter, **trim()** will strip these characters:

- " " (ASCII 32 (0x20)), an ordinary space.
- "\t" (ASCII 9 (0x09)), a tab.
- "\n" (ASCII 10 (0x0A)), a new line (line feed).
- "\r" (ASCII 13 (0x0D)), a carriage return.
- "\0" (ASCII 0 (0x00)), the NUL-byte.

- "\x0B" (ASCII `11` (`0x0B`)), a vertical tab.

You can also specify the characters you want to strip, by means of the *`charlist`* parameter. Simply list all characters that you want to be stripped. With `..` you can specify a range of characters.

Példa 1. Usage example of trim()

```
<?php

$text = "\t\tThese are a few words :) ...  ";
$trimmed = trim($text);
// $trimmed = "These are a few words :) ..."
$trimmed = trim($text," \t.");
// $trimmed = "These are a few words :)"
$clean = trim($binary,"\0x00..\0x1F");
// trim the ASCII control characters at the beginning and end of $binary
// (from 0 to 31 inclusive)

?>
```

See also ltrim() and rtrim().

ucfirst (PHP 3, PHP 4)

Make a string’s first character uppercase

string **ucfirst** (string str) \linebreak

Returns a string with the first character of *`str`* capitalized, if that character is alphabetic.

Note that ’alphabetic’ is determined by the current locale. For instance, in the default "C" locale characters such as umlaut-a (ä) will not be converted.

Példa 1. ucfirst() example

```
$foo = ’hello world!’;
$foo = ucfirst($foo);             // Hello world!

$bar = ’HELLO WORLD!’;
$bar = ucfirst($bar);             // HELLO WORLD!
$bar = ucfirst(strtolower($bar)); // Hello world!
```

See also strtolower(), strtoupper(), and ucwords().

ucwords (PHP 3>= 3.0.3, PHP 4)

Uppercase the first character of each word in a string

string **ucwords** (string str) \linebreak

Returns a string with the first character of each word in *str* capitalized, if that character is alphabetic.

Példa 1. ucwords() example

```
$foo = 'hello world!';
$foo = ucwords($foo);             // Hello World!

$bar = 'HELLO WORLD!';
$bar = ucwords($bar);             // HELLO WORLD!
$bar = ucwords(strtolower($bar)); // Hello World!
```

Megjegyzés: The definition of a word is any string of characters that is immediately after a whitespace (These are: space, form-feed, newline, carriage return, horizontal tab, and vertical tab).

See also strtoupper(), strtolower() and ucfirst().

vprintf (PHP 4 >= 4.1.0)

Output a formatted string

void **vprintf** (string format, array args) \linebreak

Display array values as a formatted string according to *format* (which is described in the documentation for sprintf()).

Operates as printf() but accepts an array of arguments, rather than a variable number of arguments.

See also: printf(), sprintf(), vsprintf()

vsprintf (PHP 4 >= 4.1.0)

Return a formatted string

string **vsprintf** (string format, array args) \linebreak

Return array values as a formatted string according to *format* (which is described in the documentation for sprintf()).

Operates as sprintf() but accepts an array of arguments, rather than a variable number of arguments.

See also: sprintf(), **vsprintf()**, vprintf()

wordwrap (PHP 4 >= 4.0.2)

Wraps a string to a given number of characters using a string break character.

string **wordwrap** (string str [, int width [, string break [, int cut]]]) \linebreak

Returns a string with *str* wrapped at the column number specified by the (optional) *width* parameter. The line is broken using the (optional) *break* parameter.

wordwrap() will automatically wrap at column 75 and break using '\n' (newline) if *width* or *break* are not given.

If the *cut* is set to 1, the string is always wrapped at the specified width. So if you have a word that is larger than the given width, it is broken apart. (See second example).

> **Megjegyzés:** The optional *cut* parameter was added in PHP 4.0.3

Példa 1. wordwrap() example

```
$text = "The quick brown fox jumped over the lazy dog.";
$newtext = wordwrap( $text, 20 );

echo "$newtext\n";
```

This example would display:

```
The quick brown fox
jumped over the lazy dog.
```

Példa 2. wordwrap() example

```
$text = "A very long woooooooooooord.";
$newtext = wordwrap( $text, 8, "\n", 1);

echo "$newtext\n";
```

This example would display:

```
A very
long
woooooooo
oooooord.
```

See also nl2br().

XCVIII. Sybase functions

sybase_affected_rows (PHP 3>= 3.0.6, PHP 4)

get number of affected rows in last query

int **sybase_affected_rows** ([int link_identifier]) \linebreak

Returns: The number of affected rows by the last query.

sybase_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query on the server associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

This command is not effective for SELECT statements, only on statements which modify records. To retrieve the number of rows returned from a SELECT, use sybase_num_rows().

Megjegyzés: This function is only available using the CT library interface to Sybase, and not the DB library.

sybase_close (PHP 3, PHP 4)

close Sybase connection

bool **sybase_close** (int link_identifier) \linebreak

Returns: TRUE on success, FALSE on error

sybase_close() closes the link to a Sybase database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

sybase_close() will not close persistent links generated by sybase_pconnect().

See also: sybase_connect(), sybase_pconnect().

sybase_connect (PHP 3, PHP 4)

open Sybase server connection

int **sybase_connect** (string servername, string username, string password [, string charset]) \linebreak

Returns: A positive Sybase link identifier on success, or FALSE on error.

sybase_connect() establishes a connection to a Sybase server. The servername argument has to be a valid servername that is defined in the 'interfaces' file.

In case a second call is made to **sybase_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling sybase_close().

See also sybase_pconnect(), sybase_close().

sybase_data_seek (PHP 3, PHP 4)

move internal row pointer

bool **sybase_data_seek** (int result_identifier, int row_number) \linebreak

Returns: TRUE on success, FALSE on failure

sybase_data_seek() moves the internal row pointer of the Sybase result associated with the specified result identifier to pointer to the specifyed row number. The next call to sybase_fetch_row() would return that row.

See also: **sybase_data_seek()**.

sybase_fetch_array (PHP 3, PHP 4)

fetch row as array

array **sybase_fetch_array** (int result) \linebreak

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

sybase_fetch_array() is an extended version of sybase_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using **sybase_fetch_array()** is NOT significantly slower than using sybase_fetch_row(), while it provides a significant added value.

For further details, also see sybase_fetch_row().

sybase_fetch_field (PHP 3, PHP 4)

get field information

object **sybase_fetch_field** (int result [, int field_offset]) \linebreak

Returns an object containing field information.

sybase_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by **sybase_fetch_field()** is retreived.

The properties of the object are:

- name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.

- column_source - the table from which the column was taken
- max_length - maximum length of the column
- numeric - 1 if the column is numeric
- type - datatype of the column

See also sybase_field_seek()

sybase_fetch_object (PHP 3, PHP 4)

fetch row as object

int **sybase_fetch_object** (int result) \linebreak

Returns: An object with properties that correspond to the fetched row, or FALSE if there are no more rows.

sybase_fetch_object() is similar to sybase_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to sybase_fetch_array(), and almost as quick as sybase_fetch_row() (the difference is insignificant).

See also: sybase_fetch_array() and sybase_fetch_row().

sybase_fetch_row (PHP 3, PHP 4)

get row as enumerated array

array **sybase_fetch_row** (int result) \linebreak

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

sybase_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **sybase_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

See also: sybase_fetch_array(), sybase_fetch_object(), sybase_data_seek(), **sybase_fetch_lengths()**, and sybase_result().

sybase_field_seek (PHP 3, PHP 4)

set field offset

int **sybase_field_seek** (int result, int field_offset) \linebreak

Seeks to the specified field offset. If the next call to sybase_fetch_field() won't include a field offset, this field would be returned.

See also: sybase_fetch_field().

sybase_free_result (PHP 3, PHP 4)

free result memory

bool **sybase_free_result** (int result) \linebreak

sybase_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script ends. You may call **sybase_free_result()** with the result identifier as an argument and the associated result memory will be freed.

sybase_get_last_message (PHP 3, PHP 4)

Returns the last message from the server

string **sybase_get_last_message** (void) \linebreak

sybase_get_last_message() returns the last message reported by the server.

sybase_min_client_severity (PHP 3, PHP 4)

Sets minimum client severity

void **sybase_min_client_severity** (int severity) \linebreak

sybase_min_client_severity() sets the minimum client severity level.

> **Megjegyzés:** This function is only available using the CT library interface to Sybase, and not the DB library.

See also: sybase_min_server_severity().

sybase_min_error_severity (PHP 3, PHP 4)

Sets minimum error severity

void **sybase_min_error_severity** (int severity) \linebreak

sybase_min_error_severity() sets the minimum error severity level.

See also: sybase_min_message_severity().

sybase_min_message_severity (PHP 3, PHP 4)

Sets minimum message severity

void **sybase_min_message_severity** (int severity) \linebreak

sybase_min_message_severity() sets the minimum message severity level.

See also: sybase_min_error_severity().

sybase_min_server_severity (PHP 3, PHP 4)

Sets minimum server severity

void **sybase_min_server_severity** (int severity) \linebreak

sybase_min_server_severity() sets the minimum server severity level.

> **Megjegyzés:** This function is only available using the CT library interface to Sybase, and not the DB library.

See also: sybase_min_client_severity().

sybase_num_fields (PHP 3, PHP 4)

get number of fields in result

int **sybase_num_fields** (int result) \linebreak

sybase_num_fields() returns the number of fields in a result set.

See also: **sybase_db_query()**, sybase_query(), sybase_fetch_field(), sybase_num_rows().

sybase_num_rows (PHP 3, PHP 4)

get number of rows in result

int **sybase_num_rows** (int result) \linebreak

sybase_num_rows() returns the number of rows in a result set.

See also: **sybase_db_query()**, sybase_query() and, sybase_fetch_row().

sybase_pconnect (PHP 3, PHP 4)

open persistent Sybase connection

int **sybase_pconnect** (string servername, string username, string password [, string charset]) \linebreak

Returns: A positive Sybase persistent link identifier on success, or FALSE on error

sybase_pconnect() acts very much like sybase_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (sybase_close() will not close links established by **sybase_pconnect()()**).

This type of links is therefore called 'persistent'.

sybase_query (PHP 3, PHP 4)

send Sybase query

int **sybase_query** (string query, int link_identifier) \linebreak

Returns: A positive Sybase result identifier on success, or FALSE on error.

sybase_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if sybase_connect() was called, and use it.

See also: **sybase_db_query()**, sybase_select_db(), and sybase_connect().

sybase_result (PHP 3, PHP 4)

get result data

string **sybase_result** (int result, int row, mixed field) \linebreak

Returns: The contents of the cell at the row and offset in the specified Sybase result set.

sybase_result() returns the contents of one cell from a Sybase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than sybase_result(). Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: sybase_fetch_row(), sybase_fetch_array(), and sybase_fetch_object().

sybase_select_db (PHP 3, PHP 4)

select Sybase database

bool **sybase_select_db** (string database_name, int link_identifier) \linebreak

Returns: TRUE on success, FALSE on error

sybase_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if sybase_connect() was called, and use it.

Every subsequent call to sybase_query() will be made on the active database.

See also: sybase_connect(), sybase_pconnect(), and sybase_query()

XCIX. Tokenizer functions

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

token_get_all (PHP 4 >= 4.2.0)

Split given source in tokens

array **token_get_all** (string source) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

token_name (PHP 4 >= 4.2.0)

Get the name of a given token

string **token_name** (int type) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

C. URL függvények

base64_decode (PHP 3, PHP 4)

MIME base64-el kódolt adatot dekódol

string **base64_decode** (string encoded_data) \linebreak

A **base64_decode()** dekódolja az *encoded_data* paraméterben megadott adatot, és visszaadja az eredeti adatot. A visszaadott adat lehet bináris.

Lásd még: base64_encode(), RFC-2045 6.8. rész.

base64_encode (PHP 3, PHP 4)

Adatot kódol MIME base64 kódolással

string **base64_encode** (string data) \linebreak

A **base64_encode()** visszaadja a *data* paraméterben megadott adat base64 kódolt alakját. A kódolást úgy tervezték, hogy a bináris adatok is túléljék az olyan közvetítő csatornákat, amelyek nem képesek 8 bites adattovábbításra (pl. mail üzenetek).

A base64-kódolt adat megközelítőleg 33%-kal nagyobb az eredeti méreténél.

Lásd még: base64_decode(), chunk_split(), RFC-2045 6.8 rész.

parse_url (PHP 3, PHP 4)

Feldolgoz egy URL-t, és visszaadja a komponenseit

array **parse_url** (string url) \linebreak

Ez a függvény olyan asszociatív tömböt ad vissza, amely tartalmazza az alábbi elemek közül azokat, amelyek az *url* részei:

- *scheme* - pl.: http://
- *host*
- *port*
- *user*
- *pass*
- *path*
- *query* - ? utáni rész
- *fragment* - # utáni rész

Lásd még: pathinfo()!

rawurldecode (PHP 3, PHP 4)

URL-kódolt karakterláncot kódol vissza

string **rawurldecode** (string str) \linebreak

A *str*-ben megadott sztringet visszakódolja úgy, hogy minden százalékjelet (%) és utána álló két hexadecimális számjegyet a megjelölt sorszámú karakterrel cserél fel. Például a

```
foo%20bar%40baz
```

karaktersorozatból

```
foo bar@baz
```

lesz.

> **Megjegyzés: rawurldecode()** nem cseréli le a plusz jelet ('+') szóközzé, amint azt az urldecode() teszi.

Lásd még: rawurlencode(), urldecode() és urlencode()!

rawurlencode (PHP 3, PHP 4)

URL-kódolást végez az RFC1738 szerint

string **rawurlencode** (string str) \linebreak

Olyan stringgel tér vissza, amelyben az *str* nem alfanumerikus karakterei - kivéve a

```
-_.
```

karaktereket - egy szézalékjelre (%) és azt követő két hexadecimális számjegyre lesz lecserélve. Ez az RFC1738-ben leírt kódolásnak megfelelő, amely feloldja a speciális karakterek az URL-ekben történő használatát nehezítő problémát, illetve a közvetítő média által végrehajtott kódolás okozta hibákat kiküszöböli. Ha egy jelszót kell például elhelyezni egy FTP URL-ben:

Példa 1. Első rawurlencode() példa

```
echo '<a href="ftp://user:', rawurlencode ('foo @+%/'),
     '@ftp.domain.hu/titkos.txt">';
```

Előfordulhat azonban, hogy az URL PATH_INFO komponensében kell adatot átadni:

Példa 2. Második rawurlencode() példa

```
echo ’<a href="http://domain.hu/kirendeltsegek_listazo/’,
    rawurlencode (’kereskedelmi és marketing/Pécs’), ’">’;
```

Lásd még: rawurldecode(), urlencode() és urldecode()!

urldecode (PHP 3, PHP 4)

Dekódol egy URL-kódolt karakterláncot

string **urldecode** (string str) \linebreak

Dekódol bármilyen %## kódolást a megadott karakterláncban. A dekódolt szöveget adja vissza.

Példa 1. urldecode() példa

```
$a = split(’&’, $querystring);
$i = 0;
while ($i < count($a)) {
    $b = split(’=’, $a[$i]);
    echo ’A(z) ’, htmlspecialchars(urldecode($b[0])),
         ’ paraméter értéke ’, htmlspecialchars(urldecode($b[1])), "<br>";
    $i++;
}
```

Lásd még: urlencode()!

urlencode (PHP 3, PHP 4)

URL-kódol egy karakterláncot

string **urlencode** (string str) \linebreak

Visszaad egy karakterláncot, amiben minden nem alfanumerikus karakter, kivéve a `-_.` karaktereket, százalék jelre (%) és két hexa számra cserélt, és a szóközök helyén plusz jelek (+) állnak. Ez az a kódolási forma, amivel a kérdőívekkel küldött adatok a weben közlekednek, azaz az `application/x-www-form-urlencoded` média típus. Ez annyiban különbözik az RFC1738 kódolástól (rawurlencode()), hogy történelmi okok miatt a szóközök itt plusz (+) jelekké alakulnak. Ez a függvény kényelmes lehetőséget ad arra, hogy az URL kérdőjel utáni részébe írjunk, változókat átadva a következő oldalnak:

Példa 1. urlencode() példa

```
echo ’<a href="ujoldal.php?ize=’, urlencode($valtozo), ’">’;
```

Figyelem: Vigyázz az olyan változókkal, amelyek neve HTML entitások nevével kezdődik. Az olyan karaktersorozatokat, mint &, &copy és &pound a böngészők lecserélik a megfelelő entitásra, és nem a várt eredményt kapod. Ez egy nyilvánvaló probléma, amire a W3C már évekkel ezelőtt felhívta a figyelmet. A referencia itt olvasható: http://www.w3.org/TR/html4/appendix/notes.html#h-B.2.2. A PHP támogatja a paraméter-elválasztó karakter megváltoztatását, a W3C által javasolt pontosvesszőre, az arg_separator nevű .ini beállítással. Sajnálatosan a legtöbb böngésző nem támogatja ezt az elválasztókaraktert. Egy hordozhatóbb módszer az & használata az & karakter helyett elválasztóként. Így nem kell megváltoztatnod a PHP arg_separator beállítását sem. Hagyd ezt & értéken, de használd az URL kódoláshoz a htmlentities()(urlencode($adat)) módszert.

Példa 2. urlencode/htmlentities() példa

```
echo ’<a href="ujoldal.php?ize=’, htmlentities(urlencode($valtozo)), ’">’;
```

Lásd még: urldecode(), htmlentities(), rawurldecode() és rawurlencode()!

CI. Változókkal kapcsolatos függvények

A változók viselkedéséről további információkat kaphatsz a változók témakörben a nyelv alapjai című részben.

doubleval (PHP 3, PHP 4)

A floatval() másik neve

Ez a függvény a floatval() függvény másik neve.

> **Megjegyzés:** Ez a függvény egy átnevezés maradványa. Régebbi PHP verziókban ezt a nevet kell használnod, ha az új floatval() név még nem elérhető.

empty (unknown)

Változó ürességének tesztelése

boolean **empty** (mixed var) \linebreak

> **Megjegyzés:** Az **empty()** egy nyelvi szerkezet.

Ez a `(boolean)` *`var`*, ellentetje, kivéve hogy nem kapsz figyelmeztetést, ha a változó nincs beállítva. Lásd még a logikai értékké alakítás című részt.

```
$valtozo = 0;

if (empty($valtozo)) { // igazat ad vissza
    print '$valtozo nulla vagy nincs beállítva';
}

if (!isset($valtozo)) { // hamisat ad vissza
    print '$valtozo egyáltalán nincs beállítva';
}
```

Értelmetlen kifejezésekre használni ezt a függvényt, mint pl. **empty (addslashes ($nev))**, mivel nincs jelentése annak, hogy valami, ami nem is változó, változó-e üres értékkel...

Lásd még isset() és unset().

floatval (PHP 4 >= 4.2.0)

Változó lebegőpontos értékének visszaadása

float **floatval** (mixed var) \linebreak

Visszaddja *`var`* float típusúvá alakított értékét.

A *var* lehet bármilye skaláris típus. Nem használhatod azonban a **floatval()** függvényt tömbökön vagy objektumokon.

```
$valtozo = ’122.34343szöveg’;
$lebegopontos_ertek = floatval($valtozo);
print $lebegopontos_ertek; // kiírja, hogy 122.34343
```

Lásd még intval(), strval(), settype() és Buvészkedés a típusokkal.

get_defined_vars (PHP 4 >= 4.0.4)

Minden definiált változó lekérdezése

array **get_defined_vars** (void) \linebreak

Ez a függvény egy többdimenziós tömbbel tér vissza az összes definiált változóval, legyenek azok akár környezeti, szerver vagy felhasználó által létrehozott változók.

```
$b = array(1,1,2,3,5,8);

$tomb = get_defined_vars();

// $b kiírása
print_r($tomb["b"]);

// A PHP feldolgozó elérési útja (ha CGI-ként használod)
// például /usr/local/bin/php
echo $tomb["_"];

// A parancssori paraméterek kiírása (ha vannak)
print_r($tomb["argv"]);

// Minden szerver változó kiírása
print_r($tomb["HTTP_SERVER_VARS"]);

// A változók tömbjéből minden kulcs kiírása
print_r(array_keys(get_defined_vars()));
```

Lásd még get_defined_functions() és get_defined_constants().

get_resource_type (PHP 4 >= 4.0.2)

Erőforrás típus lekérdezése

string **get_resource_type** (resource handle) \linebreak

Ez a függvény az erőforrás típust (resource) reprezentáló karaktersorozattal tér vissza. Ha a paraméter nem érvényes erőforrás, hibát kapsz.

```
$kapcs = mysql_connect();
echo get_resource_type($kapcs)."\n";
// kiírja: mysql link

$ize = fopen("ize","w");
echo get_resource_type($ize)."\n";
// kiírja: file

$dok = new_xmldoc("1.0");
echo get_resource_type($dok->doc)."\n";
// kiírja: domxml document
```

gettype (PHP 3, PHP 4)

Változó típusának visszaadása

string **gettype** (mixed var) \linebreak

Visszaadja a *var* paraméterben megadott PHP változó típusát.

> **Figyelem**
>
> Soha ne használd a **gettype()** függvényt egy bizonyos típusba tartozás vizsgálatához, mivel a visszaadott karaktersorozat megváltozhat későbbi PHP verziókban. Ráadásul lassabb, mint egy bizonyos típus vizsgálata, mivel karaktersorozat általakítást használ.
>
> Ehelyett használd az `is_*` függvényeket.

A lehetséges visszatérési értékek:

- "boolean" - logikai típus (a PHP 4-esben)
- "integer" - egész típus
- "double" (történeti okokból "double" a visszatérési érték a float lebegőpontos típusra, és nem "float")
- "string" - karaktersorozat

- "array" - tömb
- "object" - objektum
- "resource" - erőforrás (a PHP 4-esben)
- "NULL" - NULL (a PHP 4-esben)
- "user function" - felhasználói függvény (csak a PHP 3-asban!)
- "unknown type" - ismeretlen típus

A PHP 4-esben, a function_exists() és method_exists() függvényeket kell használnod, a korábbi "user function" visszatérési értéket tesztelő PHP 3-as **gettype()** hívások helyett.

Lásd még settype(), is_array(), is_bool(), is_float(), is_integer(), is_null(), is_numeric(), is_object(), is_resource(), is_scalar() és is_string().

import_request_variables (PHP 4 >= 4.1.0)

A GET/POST/Cookie változók betöltése a globális környezetbe

bool **import_request_variables** (string types [, string prefix]) \linebreak

Ez a függvény a GET/POST/Cookie változókat betölti a globális környezetbe. Hasznos lehet, ha kikapcsoltad a register_globals beállítást, de szeretnél látni néhány változót a globális környezetben.

A *types* paraméter használatával meg tudod adni, hogy milyen típusú változókat importáljon a PHP. Használhatod a 'G', 'P' és 'C' karaktereket a GET, POST és Cookie változókhoz értelemszerűen. Ezek a karakterek nem érzékenyek a kis- és nagybetűs írásra, ezért használhatod a 'g', 'p' és 'c' karaktereket is. A POST típus magában foglalja a feltöltött állomány információkat is. Fontos, hogy a betűk sorrendje számít, mivel például "gp" használatakor az azonos nevű POST változók felül fogják írni a GET változókat. A GPC-n kívüli betűket nem veszi figyelembe a függvény.

> **Megjegyzés:** Habár a *prefix* paraméter opcionális, egy megjegyzés szintű hibaüzenetet fogsz kapni, ha nem adod meg, vagy egy üres karaktersorozatot adsz meg. Ez egy lehetséges biztonsági rés ugyanis. A megjegyzés szintű hibák nem kerülnek kiírásra alapbeállításban.

```
// Importáljuk a GET és POST változókat az
// "rvar_" előtagot odatéve minden név elé
import_request_variables("gP", "rvar_");
```

Lásd még register_globals és track_vars.

intval (PHP 3, PHP 4)

Változó egész értékének visszaadása

int **intval** (mixed var [, int base]) \linebreak

Visszadja a *var* változó értékét egész típussá alakítva, felhasználva a *base*-ben megadott számrendszert (ez alapbeállításban 10).

A *var* bármilyen skalár típus lehet. Nem alkalmazhatod azonban az **intval()**-t tömbökre vagy objektumokra.

Megjegyzés: A *base* paraméternek csak akkor van hatása, ha a *var* karaktersorozat.

Lásd még floatval(), strval(), settype() és Buvészkedés a típusokkal.

is_array (PHP 3, PHP 4)

Változó tömb típusba tartozásának ellenőrzése

bool **is_array** (mixed var) \linebreak

TRUE értéket ad vissza, ha a *var* tömb, FALSE értéket ad egyébként.

Lásd még is_float(), is_int(), is_integer(), is_string() és is_object().

is_bool (PHP 4)

Változó logikai típusba tartozásának ellenőrzése

bool **is_bool** (mixed var) \linebreak

TRUE értéket ad vissza, ha a *var* logikai típusú, FALSE értéket ad egyébként.

Lásd még is_array(), is_float(), is_int(), is_integer(), is_string() és is_object().

is_callable (PHP 4 >= 4.0.6)

Hívható tulajdonság ellenőrzése

bool **is_callable** (mixed var [, bool syntax_only [, string callable_name]]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

is_double (PHP 3, PHP 4)

Az is_float() másik neve

Az is_float() másik neve.

is_float (PHP 3, PHP 4)

Változó lebegőpontos típusba tartozásának ellenőrzése

bool **is_float** (mixed var) \linebreak

TRUE értéket ad vissza, ha a *var* lebegőpontos, FALSE értéket ad egyébként.

> **Megjegyzés:** Ha azt szeretnéd ellenőrizni, hogy egy változó szám, vagy egy számot tartalmazó karaktersorozat (mint például egy űrlap érték, ami mindig karaktersorozat), használd az is_numeric() függvényt.

Lásd még is_bool(), is_int(), is_integer(), is_numeric(), is_string(), is_array() és is_object().

is_int (PHP 3, PHP 4)

Változó egész szám típusba tartozásának ellenőrzése

bool **is_int** (mixed var) \linebreak

TRUE értéket ad vissza, ha a *var* egész szám, FALSE értéket ad egyébként.

> **Megjegyzés:** Ha azt szeretnéd ellenőrizni, hogy egy változó szám, vagy egy számot tartalmazó karaktersorozat (mint például egy űrlap érték, ami mindig karaktersorozat), használd az is_numeric() függvényt.

Lásd még: is_bool(), is_float(), is_integer(), is_numeric(), is_string(), is_array() és is_object().

is_integer (PHP 3, PHP 4)

A is_int() másik neve

Ez a függvény az is_int() másik neve.

is_long (PHP 3, PHP 4)

A is_int() másik neve

Ez a függvény az is_int() másik neve.

is_null (PHP 4 >= 4.0.4)

Változó NULL típusba tartozásának ellenőrzése

bool **is_null** (mixed var) \linebreak

TRUE értéket ad vissza, ha a *var* NULL, FALSE értéket ad egyébként.

Lásd még is_bool(), is_numeric(), is_float(), is_int(), is_string(), is_object(), is_array(), is_integer() és is_real()

is_numeric (PHP 4)

Változó numerikus vagy numerikus szöveg típusba tartozásának ellenőrzése

bool **is_numeric** (mixed var) \linebreak

TRUE értéket ad vissza, ha a *var* numerikus vagy numerikus szöveg, FALSE értéket ad egyébként.

Lásd még is_bool(), is_float(), is_int(), is_string(), is_object(), is_array() és is_integer().

is_object (PHP 3, PHP 4)

Változó objektum típusba tartozásának ellenőrzése

bool **is_object** (mixed var) \linebreak

TRUE értéket ad vissza, ha a *var* objektum, FALSE értéket ad egyébként.

Lásd még is_bool(), is_int(), is_integer(), is_float(), is_string() és is_array().

is_real (PHP 3, PHP 4)

Az is_float() másik neve

Ez a függvény az is_float() másik neve.

is_resource (PHP 4)

Változó erőforrás típusba tartozásának ellenőrzése

bool **is_resource** (mixed var) \linebreak

Az **is_resource()** TRUE értéket ad vissza, ha a *var* paraméter által megadott változó erőforrásazonosító, FALSE értéket ad egyébként.

Lásd a resource típus dokumentációját további információkért.

is_scalar (PHP 4 >= 4.0.5)

Változó skalár típusba tartozásának ellenőrzése

bool **is_scalar** (mixed var) \linebreak

Az **is_scalar()** TRUE értéket ad, ha a *var* változó skalár, egyébként FALSE értéket ad.

A skaláris változók csoportja az integer, float, string és boolean típusokat tartalmazza. Az array, object és resource típusok nem skalárok.

```
function valtozo_kiiras($valtozo) {
    if (is_scalar($valtozo)) {
        echo $valtozo;
    } else {
        var_dump($valtozo);
    }
}
$pi = 3.1416;
$proteinek = array("hemoglobin", "cytochrome c oxidase", "ferredoxin");

valtozo_kiiras($pi);
// kiírja: 3.1416

változo_kiiras($proteinek)
// kiírja:
// array(3) {
//   [0]=>
//   string(10) "hemoglobin"
//   [1]=>
//   string(20) "cytochrome c oxidase"
//   [2]=>
//   string(10) "ferredoxin"
// }
```

Megjegyzés: Az **is_scalar()** nem tekinti a resource (erőforrás) típusú értékeket skalároknak, mivel az erőforrás absztrakt adattípus, jelenleg az egész számokra építve. Ez a megvalósítási részlet azonban nem vehető figyelembe, mivel később megváltozhat.

Lásd még is_bool(), is_numeric(), is_float(), is_int(), is_real(), is_string(), is_object(), is_array() és is_integer().

is_string (PHP 3, PHP 4)

Változó karaktersorozat típusba tartozásának ellenőrzése

bool **is_string** (mixed var) \linebreak

`TRUE` értéket ad vissza, ha a *`var`* karaktersorozat, `FALSE` értéket ad egyébként.

Lásd még is_bool(), is_int(), is_integer(), is_float(), is_real(), is_object() és is_array().

isset (unknown)

Változó beállítottságának ellenőrzése

boolean **isset** (mixed var [, mixed var [, ...]]) \linebreak

> **Megjegyzés:** Az **isset()** egy nyelvi szerkezet.

`TRUE` értéket ad vissza, ha *`var`* létezik, `FALSE` értéket ad egyébként.

Ha egy változót megszüntettél unset()-el, az **isset()** hamisat fog adni. Az **isset()** továbbá `FALSE` érétket fog adni, ha az ellenőrzött változó értéke `NULL`. Szintén fontos, hogy a `NULL` bájt (`"\0"`) nem egyezik meg a PHP `NULL` konstans értékével.

```
$a = "próba";
$b = "még egy próba";

echo isset ($a);     // TRUE
echo isset ($a, $b)  // TRUE

unset ($a);
echo isset ($a);     // FALSE
echo isset ($a, $b); // FALSE

$ize = NULL;
print isset ($ize);  // FALSE
```

Lásd még empty() és unset().

print_r (PHP 4)

Ember számára olvasható információ egy változóról

void **print_r** (mixed expression) \linebreak

Ez a függvény egy ember számára olvasható formában ad vissza információt a megadott változóról. Ha egy string, integer vagy float típusú értéket kap, magát az értéket írja ki. Ha tömböt kap, az elemek kulcs-érték párjait írja ki. Hasonlóan viselkedik objektumok esetében is.

A **print_r()** a tömb belső mutatóját a tömb végére mozgatja. Használd a reset() függvényt, ha előre kell mozgatnod a mutatót.

> **Tipp:** Mint bármilyen más esetben, amikor a kimenet közvetlenül a böngészőhöz kerül, használhatod az kimenet szabályozó függvényeket, hogy a függvény kimenetét "elkapd", és elmentsd például egy string-ben.

```
<?php
    $tomb = array (’a’ => ’alma’, ’b’ => ’banán’, ’c’ => array (’x’,’y’,’z’));
    print_r($tomb);
?>
```

Kimenete:

```
<pre>
Array
(
    [a] => alma
    [b] => banán
    [c] => Array
        (
            [0] => x
            [1] => y
            [2] => z
        )
)
</pre>
```

> **Megjegyzés:** Ez a függvény a PHP 4.0.4 verzió előtt a ’végtelenségig’ futott, ha egy direkt, vagy indirekt magára mutató tömböt vagy objektumot adtál át paraméterül. Egy példa erre a `print_r($GLOBALS)`, hívás, mivel a `$GLOBALS` magában egy globális változó, és így magára is tartalmaz referenciát.

Lásd még ob_start(), var_dump(), és var_export().

serialize (PHP 3>= 3.0.5, PHP 4)

Karaktersorozat előállítása egy értékből

string **serialize** (mixed value) \linebreak

A **serialize()** egy karaktersorozattal tér vissza, ami a *value* paraméterben megadott érték karaktersorozatba kódolt megfelelője. Így bárhol tárolható.

Ez a függvény hasznos PHP értékek tárolására és továbbítására anélkül, hogy a szerkezetük és/vagy típusuk elveszne.

Ha ismét PHP értékbe szeretnéd tenni a "szerializált" karaktersorozatot, használd az unserialize() függvényt. A **serialize()** minden típust kezel a resource típus kivételével. Szerializálhatsz akár olyan tömböket is, amik magukra tartalmaznak referenciát. A referenciák, amiket a tömbben/objektumban tárolsz, szintén bekerülnek a kódolt karaktersorozatba.

> **Megjegyzés:** A PHP 3-ban az objektum tulajdonságok szerializálódnak, de a metódusok elvesznek. A PHP 4-esben ilyen probléma nem lép fel. Lásd az objektumok szerializációja részt a Osztályok, objektumok című fejezetben.

Példa 1. serialize() példa

```
// A $munkamenet_adat egy többdimenziós tömb, az aktuális
// felhasználó munkamenet adataival. A serialize()-t használjuk
// az adatok adatbázisban történő tárolására a kérés végén.

$conn = odbc_connect ("webdb", "php", "csirke");
$stmt = odbc_prepare ($conn,
      "UPDATE munkamenetek SET adat = ? WHERE id = ?");
$sqladat = array (serialize($munkamenet_adat), $PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqladat)) {
    $stmt = odbc_prepare($conn,
     "INSERT INTO munkamenetek (id, adat) VALUES(?, ?)");
    if (!odbc_execute($stmt, &$sqladat)) {
    /* Valami nem jött össze, hibakezelés */
    }
}
```

Lásd még unserialize().

settype (PHP 3, PHP 4)

Változó típusának beállítása

bool **settype** (mixed var, string type) \linebreak

Beállítja a *var* változó típusát *type* típusra.

Lehetséges *type* értékek:

- "boolean" (vagy a PHP 4.2.0 óta "bool")
- "integer" (vagy a PHP 4.2.0 óta "int")
- "float" (csak a PHP 4.2.0 óta, korábban "double")
- "string"
- "array"
- "object"
- "null" (a PHP 4.0.8 óta)

TRUE értéket ad siker esetén, FALSE értéket egyébként.

Példa 1. settype() példa

```
$ize    = "5valami"; // string
$valami = true;      // boolean

settype($ize, "integer"); // $ize most 5 (integer)
settype($valami, "string");  // $valami most "1" (string)
```

Lásd még gettype(), típus átalakítás és bűvészkedés a típusokkal.

strval (PHP 3, PHP 4)

Egy változó karaktersorozat értékét adja vissza

string **strval** (mixed var) \linebreak

Visszaadja a karaktersorozattá alakított értékét a *var* paraméterben megadott változónak. Lásd a string típus dokumentációját az átalakítás leírásához.

A *var* bármilyen skalár típus lehet. Nem alkalmazhatod azonban az **strval()** függvényt tömbökön vagy objektumokon.

Lásd még floatval(), intval(), settype() és bűvészkedés a típusokkal.

unserialize (PHP 3>= 3.0.5, PHP 4)

PHP értéket állít elő egy kódolt karaktersorozatból

mixed **unserialize** (string str) \linebreak

Az **unserialize()** egy szerializált értéket vár, (lásd a serialize() függvényt) és visszaalakítja PHP értékké azt. Az átalakított értéket adja vissza, ami lehet integer, float, string, array vagy object típusú.

> **Megjegyzés:** Definiálhatsz egy callback függvényt, amit a PHP akkor hív meg, ha egy nemlétező osztályból kell példányt létrehoznia a deszerializáció során. Erre akkor lehet szükség, ha nem szeretnél egy tökéletlen "__PHP_Incomplete_Class" objektumot kapni. Használd a `php.ini`, ini_set() vagy .htaccess segítségével beállítható 'unserialize_callback_func' értéket a függvény beállítására. Minden alkalommal, amikor egy nem definiált osztályból kell példányt létrehozni, ezt a függvényt fogja a PHP meghívni. Ha ki szeretnéd kapcsolni ezt a szolgáltatást, állítsd üresre ezt a beállítást.

Példa 1. unserialize_callback_func példa

```
$szerializalt_obj='O:1:"a":1:{s:5:"value";s:3:"100";}';

ini_set('unserialize_callback_func','mycallback'); // saját callback függvény

function mycallback($osztaly_neve) {
    // Itt be tudod tölteni azt a külső állományt, ami az
    // osztály definícióját tartalmazza. Az osztály nevét
    // kapod paraméterül, az alapján dönthetsz.
}
```

> **Megjegyzés:** A PHP 3-ban az objektum tulajdonságok szerializálódnak, de a metódusok elvesznek. A PHP 4-esben ilyen probléma nem lép fel. Lásd az objektumok szerializációja részt a Osztályok, objektumok című fejezetben.

Példa 2. unserialize() példa

```
// Itt az unserialize() függvényt használjuk, miután az adatbázisból
// lekérdeztük a munkamenet adatot a $munkamenet_adat változóba.
// Ez a példa kiegészíti a serialize() példáját.

$conn = odbc_connect ("webdb", "php", "csirke");
$stmt = odbc_prepare ($conn, "SELECT adat FROM munkamenetek WHERE id = ?");
$sqladat = array ($PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqladat) || !odbc_fetch_into ($stmt, &$tmp)) {
    // Ha az execute() vagy a fetch() hibás, üres tömböt veszünk
```

```
    $munkamenet_adat = array();
} else {
    // Most már a deszerializált adatokat a $tmp[0]-ban találjuk
    $munkamenet_adat = unserialize($tmp[0]);
    if (!is_array($munkamenet_adat)) {
        // Valami rosszul ment, üres tömböt veszünk
        $munkamenet_adat = array();
    }
}
```

Lásd még serialize().

unset (unknown)

Adott változó felszabadítása

void **unset** (mixed var [, mixed var [, ...]]) \linebreak

Megjegyzés: Az **unset()** egy nyelvi szerkezet.

Az **unset()** megszünteti az adott változókat. A PHP 3-ban mindig `TRUE` értékkel tér vissza (egészen pontosan az 1 értékű egész számmal). A PHP 4-esben az **unset()** többé már nem függvény, hanem parancs. Ezért nincs visszatérési érték, és a visszatérési érték használata feldolgozási hibát jelent.

Példa 1. unset() példa

```
// Egy változó törlése
unset ($ize);

// Egy tömbelem törlése (nem az egész tömb törlése!)
unset ($valami['barmi']);

// Több mint egy változó törlése
unset ($ize1, $ize2, $ize3);
```

Az **unset()** hatása egy függvényen belül attól függ, hogy milyen típusú változót próbálsz meg törölni.

Ha egy globális környezetből betöltött változót próbálsz meg törölni a függvényben, csak a lokális változót törlöd. A hívó környezetben lévő változó megtartja **unset()** hívás előtti értékét.

```
function ize_torlese() {
    global $ize;
    unset($ize);
```

```
}

$ize = ’valami’;
ize_torlese();
echo $ize;
```

A fenti példa a következőt írja ki:

```
valami
```

Ha egy olyan változót törölsz egy függvényben, amit referenciaképzéssel adtál át, csak a helyi változót törlöd. A hívó környezetben lévő változó megtartja **unset()** hívás előtti értékét.

```
function ize(&$valami) {
    unset($valami);
    $valami = "másmilyen szöveg";
}

$valami = ’valamilyen szöveg’;
echo "$valami\n";

foo($valami);
echo "$valami\n";
```

A fenti példa kimenete:

```
valamilyen szöveg
valamilyen szöveg
```

Ha egy statikus változót törölsz egy függvényben, az **unset()** törli a változót, és minden rá mutató referenciát.

```
function ize() {
    static $a;
    $a++;
    echo "$a\n";
    unset($a);
}

ize();
```

```
ize();
ize();
```

A fenti példa kimenete:

```
1
2
3
```

Ha egy globális változót szeretnél törölni egy függvényen belül, használd a `$GLOBALS` tömböt erre a célra:

```
function ize() {
    unset($GLOBALS[’valami’]);
}

$valami = "bármi";
ize();
```

Lásd még isset() és empty().

var_dump (PHP 3>= 3.0.5, PHP 4)

Információ egy változóról

void **var_dump** (mixed expression [, mixed expression [, ...]]) \linebreak

Ez a függvény strukturált információt ad egy vagy több kifejezésről, az értékek és típusok kiírásával. A tömbök rekurzívan kerülnek feldolgozásra, az értékek bekezdésével, hogy lásd a struktúrát.

> **Tipp:** Mint bármilyen más esetben, amikor a kimenet közvetlenül a böngészőhöz kerül, használhatod az kimenet szabályozó függvényeket, hogy a függvény kimenetét "elkapd", és elmentsd például egy string-ben.

Hasonlítsd össze a **var_dump()** függvényt és a print_r() függvényt.

```
<pre>
<?php
$a = array (1, 2, array ("a", "b", "c"));
```

```
var_dump ($a);

/* kimenete:
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}

*/

$b = 3.1;
$c = TRUE;
var_dump($b,$c);

/* kimenete:
float(3.1)
bool(true)

*/
?>
</pre>
```

var_export (PHP 4 >= 4.2.0)

Érvényes PHP kód reprezentációt ad egy változóról

mixed **var_export** (mixed expression [, bool return]) \linebreak

Ez a függvény a paraméterben átadott érték struktúrált karaktersorozat reprezentációját adja vissza. Hasonlít a var_dump() függvényhez, azzal a kivétellel, hogy a **var_export()** előállított értéke érvényes PHP kód.

Az előállított érték alapesetben kiíródik, de ha a második paraméterben a `TRUE` értéket adod meg a függvény visszatér az előállított értékkel.

Hasonlítsd össze a **var_export()** és a var_dump() függvényeket.

```
<pre>
<?php
```

```
$a = array (1, 2, array ("a", "b", "c"));
var_export ($a);

/* kimenete:
array (
  0 => 1,
  1 => 2,
  2 =>
  array (
    0 => 'a',
    1 => 'b',
    2 => 'c',
  ),
)
*/

$b = 3.1;
$v = var_export($b, TRUE);
echo $v;

/* kimenete:
3.1
*/
?>
</pre>
```

CII. vpopmail functions

vpopmail_add_alias_domain_ex (PHP 4 >= 4.0.5)

Add alias to an existing virtual domain

bool **vpopmail_add_alias_domain_ex** (string olddomain, string newdomain) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_add_alias_domain (PHP 4 >= 4.0.5)

Add an alias for a virtual domain

bool **vpopmail_add_alias_domain** (string domain, string aliasdomain) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_add_domain_ex (PHP 4 >= 4.0.5)

Add a new virtual domain

bool **vpopmail_add_domain_ex** (string domain, string passwd [, string quota [, string bounce [, bool apop]]]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_add_domain (PHP 4 >= 4.0.5)

Add a new virtual domain

bool **vpopmail_add_domain** (string domain, string dir, int uid, int gid) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_add_user (PHP 4 >= 4.0.5)

Add a new user to the specified virtual domain

bool **vpopmail_add_user** (string user, string domain, string password [, string gecos [, bool apop]]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_alias_add (PHP 4 >= 4.1.0)

insert a virtual alias

bool **vpopmail_alias_add** (string user, string domain, string alias) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_alias_del_domain (PHP 4 >= 4.1.0)

deletes all virtual aliases of a domain

bool **vpopmail_alias_del_domain** (string domain) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_alias_del (PHP 4 >= 4.1.0)

deletes all virtual aliases of a user

bool **vpopmail_alias_del** (string user, string domain) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_alias_get_all (PHP 4 >= 4.1.0)

get all lines of an alias for a domain

array **vpopmail_alias_get_all** (string domain) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_alias_get (PHP 4 >= 4.1.0)

get all lines of an alias for a domain

array **vpopmail_alias_get** (string alias, string domain) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_auth_user (PHP 4 >= 4.0.5)

Attempt to validate a username/domain/password. Returns true/false

bool **vpopmail_auth_user** (string user, string domain, string password [, string apop]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_del_domain_ex (PHP 4 >= 4.0.5)

Delete a virtual domain

bool **vpopmail_del_domain_ex** (string domain) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_del_domain (PHP 4 >= 4.0.5)

Delete a virtual domain

bool **vpopmail_del_domain** (string domain) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_del_user (PHP 4 >= 4.0.5)

Delete a user from a virtual domain

bool **vpopmail_del_user** (string user, string domain) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_error (PHP 4 >= 4.0.5)

Get text message for last vpopmail error. Returns string

string **vpopmail_error** (void) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_passwd (PHP 4 >= 4.0.5)

Change a virtual user’s password

bool **vpopmail_passwd** (string user, string domain, string password) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

vpopmail_set_user_quota (PHP 4 >= 4.0.5)

Sets a virtual user’s quota

bool **vpopmail_set_user_quota** (string user, string domain, string quota) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

CIII. W32api functions

w32api_deftype (PHP 4 >= 4.2.0)

Defines a type for use with other w32api_functions

int **w32api_deftype** (string typename, string member1_type, string member1_name) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

w32api_init_dtype (PHP 4 >= 4.2.0)

Creates an instance to the data type typename and fills it with the values val1, val2, the function then returns a DYNAPARM which can be passed when invoking a function as a parameter

resource **w32api_init_dtype** (string typename, mixed val1, mixed val2) \linebreak

> **Figyelem**
>
> Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

w32api_invoke_function (PHP 4 >= 4.2.0)

Invokes function funcname with the arguments passed after the function name

mixed **w32api_invoke_function** (string funcname) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

w32api_register_function (PHP 4 >= 4.2.0)

Registers function function_name from library with PHP

bool **w32api_register_function** (string libary, string function_name) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

w32api_set_call_method (PHP 4 >= 4.2.0)

Sets the calling method used

void **w32api_set_call_method** (int method) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK* . A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

CIV. WDDX függvények

Ezek a függvények a WDDX (http://www.openwddx.org/)-szel dolgoznak.

Ahhoz, hogy használhasd a WDDX-et, telepítened kell az expat programkönyvtárat (ami az Apache 1.3.7 vagy későbbi verziók része) és újra kell fordítanod a PHP-t a `--with-xml` és `--enable-wddx` paraméterekkel.

> **Megjegyzés:** Ha nemcsak ASCII karaktereket kell használni, akkor be kell állítani a megfelelő nyelvi környezetet a függvények használata előtt, lásd: setlocale().

Figyelj arra, hogy minden változókat stringgé alakító függvény az átadott tömb első elemét ellenőrzi, hogy megállapítsa, hogy egy tömböt vagy egy struktúrát állítson-e elő. Ha az első elem kulcsa string, akkor struktúrát állít elő, egyébként tömböt.

Példa 1. Egyszerű értékek átalakítása

```
<?php
print wddx_serialize_value("PHP to WDDX packet example", "PHP packet");
?>
```

Ez a példa az alábbi eredményt adja:

```
<wddxPacket version='1.0'><header comment='PHP packet'/><data>
<string>PHP to WDDX packet example</string></data></wddxPacket>
```

Példa 2. Bővíthető csomagok

```
<?php
$pi = 3.1415926;
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "pi");

/* A $cities képzeletben egy adatbázisból származik */
$cities = array("Austin", "Novato", "Seattle");
wddx_add_vars($packet_id, "cities");

$packet = wddx_packet_end($packet_id);
print $packet;
?>
```

Ez a példa az alábbi eredményt adja:

```
<wddxPacket version='1.0'><header comment='PHP'/><data><struct>
<var name='pi'><number>3.1415926</number></var><var name='cities'>
<array length='3'><string>Austin</string><string>Novato</string>
<string>Seattle</string></array></var></struct></data></wddxPacket>
```

wddx_add_vars (PHP 3>= 3.0.7, PHP 4)

Hozzáad a megadott WDDX csomaghoz változókat

wddx_add_vars (int packet_id, mixed name_var [, mixed ...]) \linebreak

A **wddx_add_vars()** az átadott változók *packet_id* által hivatkozott csomagba helyezésére szolgál. A változókat pontosan úgy kell megadni, mint a wddx_serialize_vars() függvény esetén.

wddx_deserialize (PHP 3>= 3.0.7, PHP 4)

Kibont egy WDDX csomagot

mixed **wddx_deserialize** (string packet) \linebreak

A **wddx_deserialized()** a kapott *packet* csomagot felbontja. Visszaadja az eredményt, ami egy string, szám vagy tömb. A struktúrák asszociatív tömbökké alakulnak!

wddx_packet_end (PHP 3>= 3.0.7, PHP 4)

Befejezi a megadott WDDX csomagot

string **wddx_packet_end** (int packet_id) \linebreak

A **wddx_packet_end()** befejezi a *packet_id* paraméterrel megadott WDDX csomagot és visszatér a stringgel, ami a csomagot tartalmazza.

wddx_packet_start (PHP 3>= 3.0.7, PHP 4)

Egy új WDDX csomagot kezd egy struktúrával

int **wddx_packet_start** ([string comment]) \linebreak

A **wddx_packet_start()** egy új WDDX csomagot kezd, amibe később újabb változók helyezhetőek. Van egy elhagyható *comment* string paramétere, és későbbi használat céljára egy csomag azonosítóval tér vissza. Automatikusan egy struktúra definíciót kezd a csomagban.

wddx_serialize_value (PHP 3>= 3.0.7, PHP 4)

Egy egyedülálló értékből WDDX csomagot állít elő

string **wddx_serialize_value** (mixed var [, string comment]) \linebreak

A **wddx_serialize_value()** egy addott értékből állít elő egy WDDX csomagot. Veszi a *var* értékét, és az elhagyható *comment* stringet ami a csomag fejlécében jelenik meg, és visszaadja a WDDX csomagot.

wddx_serialize_vars (PHP 3>= 3.0.7, PHP 4)

Változókat csomagol egy WDDX csomagba

string **wddx_serialize_vars** (mixed var_name [, mixed ...]) \linebreak

A **wddx_serialize_vars()** az átadott változók WDDX csomaggá alakítására használható.

A **wddx_serialize_vars()** tetszőleges számú paramétert elfogad, amelyek közül bármelyik lehet egy string, ami egy változó nevét adja, vagy egy tömb, ami változónevek tömbjét adja.

Példa 1. wddx_serialize_vars() példa

```
<?php
$a = 1;
$b = 5.5;
$c = array("blue", "orange", "violet");
$d = "colors";

$clvars = array("c", "d");
print wddx_serialize_vars("a", "b", $clvars);
?>
```

Ez a példa az alábbi eredményt adja:

```
<wddxPacket version='1.0'><header/><data><struct><var name='a'><number>1</number></var>
<var name='b'><number>5.5</number></var><var name='c'><array length='3'>
<string>blue</string><string>orange</string><string>violet</string></array></var>
<var name='d'><string>colors</string></var></struct></data></wddxPacket>
```

CV. XML értelmező függvények

Bemutatás

Az XML-ről néhány szóban

Az XML (eXtensible Markup Language) olyan adatformátum, amit a Weben keresztüli adatcserére terveztek. Ezt az ajánlást a World Wide Web Consortium (W3C) dolgozta ki és gondozza. További információk az XML-ről és a kapcsolódó technológiákról a http://www.w3.org/XML/ címen található.

Installálás

Ez a kiterjesztés az expat könyvtárat használja, amely a http://www.jclark.com/xml/ címen beszerezhető. A Makefile, amely ezzel a csomaggal jön alapértelmezés szerint nem telepíti a nekünk szükséges eljáráskönyvtárat, ehhez a következő sorokat kell beleírni:

```
libexpat.a: $(OBJS)
    ar -rc $@ $(OBJS)
    ranlib $@
```

Elérhető forrás RPM csomag is a következő helyen: http://sourceforge.net/projects/expat/, ha nem akarsz kínlódni a fordítással ;)

Az Apache 1.3.7 vagy későbbi verziójával már együtt jár az expat könyvtár. Ekkor a PHP-t csak egyszerűen a `--with-xml` opcióval kell fordítani - minden kiegészítő elérési útvonal nélkül - és ekkor az Apacheüba épített expat könyvtárat használja majd.

Az UNIX rendszereken a **configure** parancsot kell a `--with-xml` kapcsolóval. Az expat könyvtárnak olyan helyen kell installálva lennie, ahol a fordító el tudja érni. Ha a PHP-t modulként Apache 1.3.9 (ennek vagy későbbi verziójú) alá installálod, akkor a PHP automatikusan az Apache beépített expat eljáráskönyvtárat fogja használni. Szükség lehet CPPFLAGS és a LDFLAGS környezeti változók beállítására a rendszeren, mielőtt elindítod a configure parancsot, ha az expat-ot valamilyen egzotikus helyre installáltad.

A PHP telepítése.... *Tada!* Ezt kell(ene) látnod!

Erről a bővítményről

Ez a PHP bővítmény támogatást nyújt a James Clark által írt expat-hoz a PHP-ben. Ezzel az eszkökézslettel elemezhetsz, de nem érvényesíthetsz XML dokumentumokat. Háromféle forrás

karakter kódolást támogat, amit a PHP is: `US-ASCII`, `ISO-8859-1` és az`UTF-8`. Az `UTF-16` nem támogatott.

Ezzel a bővítménnyel létrehozhatsz XML elemzőket madj definiálhatsz *"handler"-eket* különféle XML feladatokhoz. Mindegyik XML elemzőnek van néhány paramétere amit beállíthasz.

Az XML eseménykezelők:

Táblázat 1. Támogatott XML kezelők (handler-ek)

PHP függvény a handler beállításához	Event description
xml_set_element_handler()	Element events are issued whenever the XML parser encounters start or end tags. There are separate handlers for start tags and end tags.
xml_set_character_data_handler()	Character data is roughly all the non-markup contents of XML documents, including whitespace between tags. Note that the XML parser does not add or remove any whitespace, it is up to the application (you) to decide whether whitespace is significant.
xml_set_processing_instruction_handler()	PHP programmers should be familiar with processing instructions (PIs) already. <?php ?> is a processing instruction, where *php* is called the "PI target". The handling of these are application-specific, except that all PI targets starting with "XML" are reserved.
xml_set_default_handler()	What goes not to another handler goes to the default handler. You will get things like the XML and document type declarations in the default handler.
xml_set_unparsed_entity_decl_handler()	This handler will be called for declaration of an unparsed (NDATA) entity.
xml_set_notation_decl_handler()	This handler is called for declaration of a notation.
xml_set_external_entity_ref_handler()	This handler is called when the XML parser finds a reference to an external parsed general entity. This can be a reference to a file or URL, for example. See the external entity example for a demonstration.

Case Folding

The element handler functions may get their element names *case-folded*. Case-folding is defined by the XML standard as "a process applied to a sequence of characters, in which those identified as non-uppercase are replaced by their uppercase equivalents". In other words, when it comes to XML, case-folding simply means uppercasing.

By default, all the element names that are passed to the handler functions are case-folded. This behaviour can be queried and controlled per XML parser with the xml_parser_get_option() and

xml_parser_set_option() functions, respectively.

Error Codes

The following constants are defined for XML error codes (as returned by xml_parse()):

XML_ERROR_NONE
XML_ERROR_NO_MEMORY
XML_ERROR_SYNTAX
XML_ERROR_NO_ELEMENTS
XML_ERROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
XML_ERROR_PARAM_ENTITY_REF
XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
XML_ERROR_MISPLACED_XML_PI
XML_ERROR_UNKNOWN_ENCODING
XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION
XML_ERROR_EXTERNAL_ENTITY_HANDLING

Character Encoding

PHP's XML extension supports the Unicode (http://www.unicode.org/) character set through different *character encoding*s. There are two types of character encodings, *source encoding* and *target encoding*. PHP's internal representation of the document is always encoded with `UTF-8`.

Source encoding is done when an XML document is parsed. Upon creating an XML parser, a source encoding can be specified (this encoding can not be changed later in the XML parser's lifetime). The supported source encodings are `ISO-8859-1`, `US-ASCII` and `UTF-8`. The former two are single-byte encodings, which means that each character is represented by a single byte. `UTF-8` can encode characters composed by a variable number of bits (up to 21) in one to four bytes. The default source encoding used by PHP is `ISO-8859-1`.

Target encoding is done when PHP passes data to XML handler functions. When an XML parser is created, the target encoding is set to the same as the source encoding, but this may be changed at any point. The target encoding will affect character data as well as tag names and processing instruction targets.

If the XML parser encounters characters outside the range that its source encoding is capable of representing, it will return an error.

If PHP encounters characters in the parsed XML document that can not be represented in the chosen target encoding, the problem characters will be "demoted". Currently, this means that such characters

are replaced by a question mark.

Some Examples

Here are some example PHP scripts parsing XML documents.

XML Element Structure Example

This first example displays the stucture of the start elements in a document with indentation.

Példa 1. Show XML Element Structure

```
$file = "data.xml";
$depth = array();

function startElement($parser, $name, $attrs) {
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "  ";
    }
    print "$name\n";
    $depth[$parser]++;
}

function endElement($parser, $name) {
    global $depth;
    $depth[$parser]--;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

XML Tag Mapping Example

Példa 2. Map XML to HTML

This example maps tags in an XML document directly to HTML tags. Elements not found in the "map array" are ignored. Of course, this example will only work with a specific XML document type.

```
$file = "data.xml";
$map_array = array(
    "BOLD"     => "B",
    "EMPHASIS" => "I",
    "LITERAL"  => "TT"
);

function startElement($parser, $name, $attrs) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "<$htmltag>";
    }
}

function endElement($parser, $name) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "</$htmltag>";
    }
}

function characterData($parser, $data) {
    print $data;
}

$xml_parser = xml_parser_create();
// use case-folding so we are sure to find the tag in $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

XML External Entity Example

This example highlights XML code. It illustrates how to use an external entity reference handler to include and parse other documents, as well as how PIs can be processed, and a way of determining "trust" for PIs containing code.

XML documents that can be used for this example are found below the example (`xmltest.xml` and `xmltest2.xml`.)

Példa 3. External Entity Example

```
$file = "xmltest.xml";

function trustedFile($file) {
    // only trust local files owned by ourselves
    if (!eregi("^([a-z]+)://", $file)
        && fileowner($file) == getmyuid()) {
            return true;
    }
    return false;
}

function startElement($parser, $name, $attribs) {
    print "&lt;<font color=\"#0000cc\">$name</font>";
    if (sizeof($attribs)) {
        while (list($k, $v) = each($attribs)) {
            print " <font color=\"#009900\">$k</font>=\"<font
                   color=\"#990000\">$v</font>\"";
        }
    }
    print "&gt;";
}

function endElement($parser, $name) {
    print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}

function characterData($parser, $data) {
    print "<b>$data</b>";
}

function PIHandler($parser, $target, $data) {
    switch (strtolower($target)) {
        case "php":
            global $parser_file;
            // If the parsed document is "trusted", we say it is safe
            // to execute PHP code inside it.  If not, display the code
            // instead.
            if (trustedFile($parser_file[$parser])) {
```

```
                eval($data);
            } else {
                printf("Untrusted PHP code: <i>%s</i>",
                        htmlspecialchars($data));
            }
            break;
    }
}

function defaultHandler($parser, $data) {
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>',
                htmlspecialchars($data));
    } else {
        printf('<font size="-1">%s</font>',
                htmlspecialchars($data));
    }
}

function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                  $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Could not open entity %s at %s\n", $openEntityNames,
                   $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML error: %s at line %d while parsing entity %s\n",
                       xml_error_string(xml_get_error_code($parser)),
                       xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return false;
            }
        }
        xml_parser_free($parser);
        return true;
    }
    return false;
}

function new_xml_parser($file) {
    global $parser_file;

    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");

    if (!($fp = @fopen($file, "r"))) {
        return false;
    }
    if (!is_array($parser_file)) {
```

```
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("could not open XML input");
}

print "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
print "</pre>";
print "parse complete\n";
xml_parser_free($xml_parser);

?>
```

Példa 4. xmltest.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
 <TITLE>Title &plainEntity;</TITLE>
 <para>
  <informaltable>
   <tgroup cols="3">
    <tbody>
     <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <section id="about">
  <title>About this Document</title>
  <para>
   <!-- this is a comment -->
   <?php print 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
 </section>
```

```
</chapter>
```

This file is included from `xmltest.xml`:

Példa 5. xmltest2.xml

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
   <element attrib="value"/>
   &testEnt;
   <?php print "This is some more PHP code being executed."; ?>
</foo>
```

utf8_decode (PHP 3>= 3.0.6, PHP 4)

Converts a string with ISO-8859-1 characters encoded with UTF-8 to single-byte ISO-8859-1.

string **utf8_decode** (string data) \linebreak

This function decodes *data*, assumed to be UTF-8 encoded, to ISO-8859-1.

See utf8_encode() for an explaination of UTF-8 encoding.

utf8_encode (PHP 3>= 3.0.6, PHP 4)

encodes an ISO-8859-1 string to UTF-8

string **utf8_encode** (string data) \linebreak

This function encodes the string *data* to UTF-8, and returns the encoded version. UTF-8 is a standard mechanism used by Unicodefor encoding *wide character* values into a byte stream. UTF-8 is transparent to plain ASCII characters, is self-synchronized (meaning it is possible for a program to figure out where in the bytestream characters start) and can be used with normal string comparison functions for sorting and such. PHP encodes UTF-8 characters in up to four bytes, like this:

Táblázat 1. UTF-8 encoding

bytes	bits	representation
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

Each *b* represents a bit that can be used to store character data.

xml_error_string (PHP 3>= 3.0.6, PHP 4)

get XML parser error string

string **xml_error_string** (int code) \linebreak

code

An error code from xml_get_error_code().

Returns a string with a textual description of the error code *code*, or FALSE if no description was found.

xml_get_current_byte_index (PHP 3>= 3.0.6, PHP 4)

get current byte index for an XML parser

int **xml_get_current_byte_index** (int parser) \linebreak

parser

A reference to the XML parser to get byte index from.

This function returns FALSE if *parser* does not refer to a valid parser, or else it returns which byte index the parser is currently at in its data buffer (starting at 0).

xml_get_current_column_number (PHP 3>= 3.0.6, PHP 4)

Get current column number for an XML parser

int **xml_get_current_column_number** (int parser) \linebreak

parser

A reference to the XML parser to get column number from.

This function returns FALSE if *parser* does not refer to a valid parser, or else it returns which column on the current line (as given by xml_get_current_line_number()) the parser is currently at.

xml_get_current_line_number (PHP 3>= 3.0.6, PHP 4)

get current line number for an XML parser

int **xml_get_current_line_number** (int parser) \linebreak

parser

A reference to the XML parser to get line number from.

This function returns FALSE if *parser* does not refer to a valid parser, or else it returns which line the parser is currently at in its data buffer.

xml_get_error_code (PHP 3>= 3.0.6, PHP 4)

get XML parser error code

int **xml_get_error_code** (int parser) \linebreak

parser

A reference to the XML parser to get error code from.

This function returns FALSE if *parser* does not refer to a valid parser, or else it returns one of the error codes listed in the error codes section.

xml_parse_into_struct (PHP 3>= 3.0.8, PHP 4)

Parse XML data into an array structure

int **xml_parse_into_struct** (int parser, string data, array &values, array &index) \linebreak

This function parses an XML file into 2 parallel array structures, one (*index*) containing pointers to the location of the appropriate values in the *values* array. These last two parameters must be passed by reference.

Below is an example that illustrates the internal structure of the arrays being generated by the function. We use a simple note tag embeded inside a para tag, and then we parse this an print out the structures generated:

```
$simple = "<para><note>simple note</note></para>";
$p = xml_parser_create();
xml_parse_into_struct($p,$simple,$vals,$index);
xml_parser_free($p);
echo "Index array\n";
print_r($index);
echo "\nVals array\n";
print_r($vals);
```

When we run that code, the output will be:

```
Index array
Array
(
    [PARA] => Array
        (
            [0] => 0
            [1] => 2
        )

    [NOTE] => Array
        (
            [0] => 1
        )
```

```
)

Vals array
Array
(
    [0] => Array
        (
            [tag] => PARA
            [type] => open
            [level] => 1
        )

    [1] => Array
        (
            [tag] => NOTE
            [type] => complete
            [level] => 2
            [value] => simple note
        )

    [2] => Array
        (
            [tag] => PARA
            [type] => close
            [level] => 1
        )

)
```

Event-driven parsing (based on the expat library) can get complicated when you have an XML document that is complex. This function does not produce a DOM style object, but it generates structures amenable of being transversed in a tree fashion. Thus, we can create objects representing the data in the XML file easily. Let’s consider the following XML file representing a small database of aminoacids information:

Példa 1. moldb.xml - small database of molecular information

```
<?xml version="1.0"?>
<moldb>

    <molecule>
        <name>Alanine</name>
        <symbol>ala</symbol>
        <code>A</code>
        <type>hydrophobic</type>
    </molecule>

    <molecule>
        <name>Lysine</name>
        <symbol>lys</symbol>
        <code>K</code>
        <type>charged</type>
```

```
    </molecule>

</moldb>
```

And some code to parse the document and generate the appropriate objects:

Példa 2. parsemoldb.php - parses moldb.xml into and array of molecular objects

```
<?php

class AminoAcid {
    var $name;  // aa name
    var $symbol;    // three letter symbol
    var $code;  // one letter code
    var $type;  // hydrophobic, charged or neutral

    function AminoAcid ($aa) {
        foreach ($aa as $k=>$v)
            $this->$k = $aa[$k];
    }
}

function readDatabase($filename) {
    // read the xml database of aminoacids
    $data = implode("",file($filename));
    $parser = xml_parser_create();
    xml_parser_set_option($parser,XML_OPTION_CASE_FOLDING,0);
    xml_parser_set_option($parser,XML_OPTION_SKIP_WHITE,1);
    xml_parse_into_struct($parser,$data,$values,$tags);
    xml_parser_free($parser);

    // loop through the structures
    foreach ($tags as $key=>$val) {
        if ($key == "molecule") {
            $molranges = $val;
            // each contiguous pair of array entries are the
            // lower and upper range for each molecule definition
            for ($i=0; $i < count($molranges); $i+=2) {
                    $offset = $molranges[$i] + 1;
                $len = $molranges[$i + 1] - $offset;
                $tdb[] = parseMol(array_slice($values, $offset, $len));
            }
        } else {
            continue;
        }
    }
    return $tdb;
}

function parseMol($mvalues) {
    for ($i=0; $i < count($mvalues); $i++)
        $mol[$mvalues[$i]["tag"]] = $mvalues[$i]["value"];
    return new AminoAcid($mol);
}
```

```
$db = readDatabase("moldb.xml");
echo "** Database of AminoAcid objects:\n";
print_r($db);

?>
```

After executing `parsemoldb.php`, the variable `$db` contains an array of `AminoAcid` objects, and the output of the script confirms that:

```
** Database of AminoAcid objects:
Array
(
    [0] => aminoacid Object
        (
            [name] => Alanine
            [symbol] => ala
            [code] => A
            [type] => hydrophobic
        )

    [1] => aminoacid Object
        (
            [name] => Lysine
            [symbol] => lys
            [code] => K
            [type] => charged
        )

)
```

xml_parse (PHP 3>= 3.0.6, PHP 4)

start parsing an XML document

int **xml_parse** (int parser, string data [, int isFinal]) \linebreak

parser

A reference to the XML parser to use.

data

Chunk of data to parse. A document may be parsed piece-wise by calling **xml_parse()** several times with new data, as long as the *isFinal* parameter is set and `TRUE` when the last data is parsed.

isFinal (optional)

If set and TRUE, *data* is the last piece of data sent in this parse.

When the XML document is parsed, the handlers for the configured events are called as many times as necessary, after which this function returns TRUE or FALSE.

TRUE is returned if the parse was successful, FALSE if it was not successful, or if *parser* does not refer to a valid parser. For unsuccessful parses, error information can be retrieved with xml_get_error_code(), xml_error_string(), xml_get_current_line_number(), xml_get_current_column_number() and xml_get_current_byte_index().

xml_parser_create_ns (PHP 4 >= 4.0.5)

Create an XML parser

int **xml_parser_create_ns** ([string encoding [, string sep]]) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xml_parser_create (PHP 3>= 3.0.6, PHP 4)

create an XML parser

int **xml_parser_create** ([string encoding]) \linebreak

encoding (optional)

Which character encoding the parser should use. The following character encodings are supported:
ISO-8859-1 (default)
US-ASCII
UTF-8

This function creates an XML parser and returns a handle for use by other XML functions. Returns FALSE on failure.

xml_parser_free (PHP 3>= 3.0.6, PHP 4)

Free an XML parser

string **xml_parser_free** (int parser) \linebreak

parser

A reference to the XML parser to free.

This function returns FALSE if *parser* does not refer to a valid parser, or else it frees the parser and returns TRUE.

xml_parser_get_option (PHP 3>= 3.0.6, PHP 4)

get options from an XML parser

mixed **xml_parser_get_option** (int parser, int option) \linebreak

parser

A reference to the XML parser to get an option from.

option

Which option to fetch. See xml_parser_set_option() for a list of options.

This function returns FALSE if *parser* does not refer to a valid parser, or if the option could not be set. Else the option's value is returned.

See xml_parser_set_option() for the list of options.

xml_parser_set_option (PHP 3>= 3.0.6, PHP 4)

set options in an XML parser

int **xml_parser_set_option** (int parser, int option, mixed value) \linebreak

parser

A reference to the XML parser to set an option in.

option

Which option to set. See below.

value

The option's new value.

This function returns FALSE if *parser* does not refer to a valid parser, or if the option could not be set. Else the option is set and TRUE is returned.

The following options are available:

Táblázat 1. XML parser options

Option constant	Data type	Description
XML_OPTION_CASE_FOLDING	integer	Controls whether case-folding is enabled for this XML parser. Enabled by default.
XML_OPTION_TARGET_ENCODING	string	Sets which target encoding to use in this XML parser. By default, it is set to the same as the source encoding used by xml_parser_create(). Supported target encodings are `ISO-8859-1`, `US-ASCII` and `UTF-8`.

xml_set_character_data_handler (PHP 3>= 3.0.6, PHP 4)

set up character data handler

int **xml_set_character_data_handler** (int parser, string handler) \linebreak

Sets the character data handler function for the XML parser *`parser`*. *`handler`* is a string containing the name of a function that must exist when xml_parse() is called for *`parser`*.

The function named by *`handler`* must accept two parameters: ***`handler`*** (int parser, string data) \linebreak

`parser`

The first parameter, *`parser`*, is a reference to the XML parser calling the handler.

`data`

The second parameter, *`data`*, contains the character data as a string.

If a handler function is set to an empty string, or `FALSE`, the handler in question is disabled.

`TRUE` is returned if the handler is set up, `FALSE` if *`parser`* is not a parser.

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

xml_set_default_handler (PHP 3>= 3.0.6, PHP 4)

set up default handler

int **xml_set_default_handler** (int parser, string handler) \linebreak

Sets the default handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

The function named by *handler* must accept two parameters: ***handler*** (int parser, string data) \linebreak

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

data

The second parameter, *data*, contains the character data. This may be the XML declaration, document type declaration, entities or other data for which no other handler exists.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handler is set up, FALSE if *parser* is not a parser.

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

xml_set_element_handler (PHP 3>= 3.0.6, PHP 4)

set up start and end element handlers

int **xml_set_element_handler** (int parser, string startElementHandler, string endElementHandler) \linebreak

Sets the element handler functions for the XML parser *parser*. *startElementHandler* and *endElementHandler* are strings containing the names of functions that must exist when xml_parse() is called for *parser*.

The function named by *startElementHandler* must accept three parameters: ***startElementHandler*** (int parser, string name, array attribs) \linebreak

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

name

The second parameter, *name*, contains the name of the element for which this handler is called. If case-folding is in effect for this parser, the element name will be in uppercase letters.

attribs

The third parameter, *attribs*, contains an associative array with the element's attributes (if any). The keys of this array are the attribute names, the values are the attribute values. Attribute names are case-folded on the same criteria as element names. Attribute values are *not* case-folded.

The original order of the attributes can be retrieved by walking through `attribs` the normal way, using each(). The first key in the array was the first attribute, and so on.

The function named by `endElementHandler` must accept two parameters:
endElementHandler (int parser, string name) \linebreak

parser

The first parameter, `parser`, is a reference to the XML parser calling the handler.

name

The second parameter, `name`, contains the name of the element for which this handler is called. If case-folding is in effect for this parser, the element name will be in uppercase letters.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handlers are set up, FALSE if `parser` is not a parser.

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

xml_set_end_namespace_decl_handler (PHP 4 >= 4.0.5)

Set up character data handler

int **xml_set_end_namespace_decl_handler** (int pind, string hdl) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

xml_set_external_entity_ref_handler (PHP 3>= 3.0.6, PHP 4)

set up external entity reference handler

int **xml_set_external_entity_ref_handler** (int parser, string handler) \linebreak

Sets the notation declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

The function named by *handler* must accept five parameters, and should return an integer value. If the value returned from the handler is FALSE (which it will be if no value is returned), the XML parser will stop parsing and xml_get_error_code() will return XML_ERROR_EXTERNAL_ENTITY_HANDLING. int ***handler*** (int parser, string openEntityNames, string base, string systemId, string publicId) \linebreak

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

openEntityNames

The second parameter, *openEntityNames*, is a space-separated list of the names of the entities that are open for the parse of this entity (including the name of the referenced entity).

base

This is the base for resolving the system identifier (*systemid*) of the external entity. Currently this parameter will always be set to an empty string.

systemId

The fourth parameter, *systemId*, is the system identifier as specified in the entity declaration.

publicId

The fifth parameter, *publicId*, is the public identifier as specified in the entity declaration, or an empty string if none was specified; the whitespace in the public identifier will have been normalized as required by the XML spec.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handler is set up, FALSE if *parser* is not a parser.

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

xml_set_notation_decl_handler (PHP 3>= 3.0.6, PHP 4)

set up notation declaration handler

int **xml_set_notation_decl_handler** (int parser, string handler) \linebreak

Sets the notation declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

A notation declaration is part of the document's DTD and has the following format:

```
<!NOTATION
     name {systemId |
     publicId}>
```

See section 4.7 of the XML 1.0 spec (http://www.w3.org/TR/1998/REC-xml-19980210#Notations) for the definition of notation declarations.

The function named by *handler* must accept five parameters: ***handler*** (int parser, string notationName, string base, string systemId, string publicId) \linebreak

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

notationName

This is the notation's *name*, as per the notation format described above.

base

This is the base for resolving the system identifier (*systemId*) of the notation declaration. Currently this parameter will always be set to an empty string.

systemId

System identifier of the external notation declaration.

publicId

Public identifier of the external notation declaration.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handler is set up, FALSE if *parser* is not a parser.

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

xml_set_object (PHP 4)

Use XML Parser within an object

void **xml_set_object** (int parser, object &object) \linebreak

This function allows to use *parser* inside *object*. All callback functions could be set with xml_set_element_handler() etc and assumed to be methods of *object*.

```
<?php
class xml  {
var $parser;

function xml() {
    $this->parser = xml_parser_create();
    xml_set_object($this->parser,&$this);
    xml_set_element_handler($this->parser,"tag_open","tag_close");
    xml_set_character_data_handler($this->parser,"cdata");
}
```

```
function parse($data) {
    xml_parse($this->parser,$data);
}

function tag_open($parser,$tag,$attributes) {
    var_dump($parser,$tag,$attributes);
}

function cdata($parser,$cdata) {
    var_dump($parser,$cdata);
}

function tag_close($parser,$tag) {
    var_dump($parser,$tag);
}

} // end of class xml

$xml_parser = new xml();
$xml_parser->parse("<A ID=\"hallo\">PHP</A>");
?>
```

xml_set_processing_instruction_handler (PHP 3>= 3.0.6, PHP 4)

Set up processing instruction (PI) handler

int **xml_set_processing_instruction_handler** (int parser, string handler) \linebreak

Sets the processing instruction (PI) handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

A processing instruction has the following format:

```
<?
       target
       data?>
```

You can put PHP code into such a tag, but be aware of one limitation: in an XML PI, the PI end tag (?>) can not be quoted, so this character sequence should not appear in the PHP code you embed with PIs in XML documents. If it does, the rest of the PHP code, as well as the "real" PI end tag, will be treated as character data.

The function named by *handler* must accept three parameters: ***handler*** (int parser, string target, string data) \linebreak

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

target

The second parameter, *target*, contains the PI target.

data

The third parameter, *data*, contains the PI data.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handler is set up, FALSE if *parser* is not a parser.

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

xml_set_start_namespace_decl_handler (PHP 4 >= 4.0.5)

Set up character data handler

int **xml_set_start_namespace_decl_handler** (int pind, string hdl) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

xml_set_unparsed_entity_decl_handler (PHP 3>= 3.0.6, PHP 4)

Set up unparsed entity declaration handler

int **xml_set_unparsed_entity_decl_handler** (int parser, string handler) \linebreak

Sets the unparsed entity declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

This handler will be called if the XML parser encounters an external entity declaration with an NDATA declaration, like the following:

```
<!ENTITY name {publicId | systemId}
        NDATA notationName>
```

See section 4.2.2 of the XML 1.0 spec (http://www.w3.org/TR/1998/REC-xml-19980210#sec-external-ent) for the definition of notation declared external entities.

The function named by *`handler`* must accept six parameters: ***`handler`*** (int parser, string entityName, string base, string systemId, string publicId, string notationName) \linebreak

`parser`

The first parameter, *`parser`*, is a reference to the XML parser calling the handler.

`entityName`

The name of the entity that is about to be defined.

`base`

This is the base for resolving the system identifier (*`systemId`*) of the external entity. Currently this parameter will always be set to an empty string.

`systemId`

System identifier for the external entity.

`publicId`

Public identifier for the external entity.

`notationName`

Name of the notation of this entity (see xml_set_notation_decl_handler()).

If a handler function is set to an empty string, or `FALSE`, the handler in question is disabled.

`TRUE` is returned if the handler is set up, `FALSE` if *`parser`* is not a parser.

Megjegyzés: A függvény neve helyett egy tömböt is átadhatsz, ami egy objektum referenciát és egy metódus nevet kell tartalmazzon.

CVI. XMLRPC functions

Figyelem

Ez a kiterjesztés *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. Ez azt jelenti, hogy minden itt dokumentált működés, beleértve a függvények nevét, működését vagy bármi más, amit a kiterjesztés kapcsán leírtunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a kiterjesztést csak a saját felelősségedre használd!

xmlrpc_decode_request (PHP 4 >= 4.1.0)

Decodes XML into native PHP types

array **xmlrpc_decode_request** (string xml, string method [, string encoding]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_decode (PHP 4 >= 4.1.0)

Decodes XML into native PHP types

array **xmlrpc_decode** (string xml [, string encoding]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_encode_request (PHP 4 >= 4.1.0)

Generates XML for a method request

string **xmlrpc_encode_request** (string method, mixed params) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_encode (PHP 4 >= 4.1.0)

Generates XML for a PHP value

string **xmlrpc_encode** (mixed value) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_get_type (PHP 4 >= 4.1.0)

Gets xmlrpc type for a PHP value. Especially useful for base64 and datetime strings

string **xmlrpc_get_type** (mixed value) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_parse_method_descriptions (PHP 4 >= 4.1.0)

Decodes XML into a list of method descriptions

array **xmlrpc_parse_method_descriptions** (string xml) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_server_add_introspection_data (PHP 4 >= 4.1.0)

Adds introspection documentation

int **xmlrpc_server_add_introspection_data** (resource server, array desc) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_server_call_method (PHP 4 >= 4.1.0)

Parses XML requests and call methods

mixed **xmlrpc_server_call_method** (resource server, string xml, mixed user_data [, array output_options]) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_server_create (PHP 4 >= 4.1.0)

Creates an xmlrpc server

resource **xmlrpc_server_create** (void) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_server_destroy (PHP 4 >= 4.1.0)

Destroys server resources

void **xmlrpc_server_destroy** (resource server) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_server_register_introspection_callback (PHP 4 >= 4.1.0)

Register a PHP function to generate documentation

bool **xmlrpc_server_register_introspection_callback** (resource server, string function) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_server_register_method (PHP 4 >= 4.1.0)

Register a PHP function to resource method matching method_name

bool **xmlrpc_server_register_method** (resource server, string method_name, string function) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xmlrpc_set_type (PHP 4 >= 4.1.0)

Sets xmlrpc type, base64 or datetime, for a PHP string value

bool **xmlrpc_set_type** (string value, string type) \linebreak

Figyelem

Ez a függvény *KÍSÉRLETI JELLEGGEL MŰKÖDIK*. A függvény működése, neve, bármi amit a függévénnyel kapcsolatban dokumentáltunk megváltozhat egy későbbi PHP kiadásban minden figyelmeztetés nélkül. Ezt a függvényt csak a saját felelősségedre használd!

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

CVII. XSLT függvények

Bevezetés

Az XSLT és a Sablotron

Az XSLT (Extensible Stylesheet Language (XSL) Transformations) XML dokumentumok más szerkezetű XML dokumentumokká alakítását végző nyelv. A World Wide Web consortium (W3C) igazgatja ezt a szabványt. Az XSLT-ről és a kapcsolodó technológiákról a http://www.w3.org/TR/xslt címen találsz több olvasnivalót.

Telepítés

Ez a kiterjesztés a Sablotron és expat csomagokat használja, amelyeknek forrásai és fordított változata is megtalálhatók a http://www.gingerall.com/ címen.

UNIX alatt, a **configure** parancsot az `--enable-xslt --with-xslt-sablot` opciókkal kell használni. A Sablotron csomagot olyan helyre kell telepíteni, ahol a fordító megtalálja.

Erről a kiterjesztésről

Ez a PHP kiterjesztés a használt motortól független API (programozási felületet) biztosít XSLT transzformációk elvégzéhez, ám jelenleg csak a Ginger Alliance Sablotron motorját támogatja. Más XSLT könyvtárak támogatása is tervbe van véve, mint például Xalan vagy libxslt.

> **Megjegyzés:** Ez a kiterjesztés különbözik a PHP 4.1.x előtti verziókban meglevőtől, és az új kiterjesztés csak a PHP 4.1.x verzióiban használható. A régi kiterjesztéssel kapcsolatos kérdéseket a php-general@lists.php.net levelező listán teheted fel.

xslt_create (PHP 4 >= 4.0.3)

új XSLT feldolgozó indítása

resource **xslt_create** (void) \linebreak

Létrehoz az XSLT feldolgozó erőforrást, amellyel a többi XSLT függvény dolgozik. Ezt az erőforrást adja vissza.

xslt_errno (PHP 4 >= 4.0.3)

visszaadja a hibakódot

int **xslt_errno** (resource xh) \linebreak

Visszaadja az *xh* XSLT feldolgozó kibocsátotta legutolsó hiba kódját.

xslt_error (PHP 4 >= 4.0.3)

visszaadja a hibaüzenetet

mixed **xslt_error** (resource xh) \linebreak

Visszaadja az *xh* XSLT feldolgozó kibocsátotta legutolsó hiba szövegét.

Példa 1. Hibakezelés az xslt_error() és xslt_errno() függvényekkel.

```
<?php

$xh = xslt_create();
$result = xslt_process($xh, 'kutya.xml', 'kedvencek.xsl');
if (!$result) {
    die(sprintf("Nem lehet az XSLT dokumentumot feldolgozni [%d]: %s",
                xslt_errno($xh), xslt_error($xh)));
}

print($result);

xslt_free($xh);
?>
```

xslt_free (PHP 4 >= 4.0.3)

megszünteti az XSLT feldolgozó erőforrás

void **xslt_free** (resource xh) \linebreak

Az *xh* paraméterben megadott XSLT feldolgozó erőforrást felszabadítja.

xslt_process (PHP 4 >= 4.0.3)

végrehajtja az XSLT transzformációt

mixed **xslt_process** (resource xh, string xml, string xsl [, string result [, array arguments [, array parameters]]]) \linebreak

Az új XSLT kiterjesztés magva a **xslt_process()** függvény. Ezzel lehet XSLT transzformációkat végezni szinte mindenféle forrású adaton - az argumentum puffer koncepiójára építve. Az argumentum puffer elképzelése a Sablotron XSLT feldolgozótól származik, amely jelenleg az egyetlen támogatott XSLT feldolgozó ehhez a kiterjesztéshez.

A legegyszerűbb transzformáció, ha az **xslt_process()** fájlokon dolgozik: a megadott XML fájlt egy XSLT fájl alapján transzformál és a végeredményt egy harmadikba írja. Ez Sablotronnal tényleg elég egyszerű...

Példa 1. Az xslt_process() használata XML fájl XSLT transzformációjára. A végeredmény is fájlba kerül

```
<?php

// új XSLT feldolgozó lefoglalása
$xh = xslt_create();

// dokumentum feldolgozása
if (xslt_process($xh, 'minta.xml', 'minta.xsl', 'eredmeny.xml')) {
    print "SIKER, a minta.xml a minta.xsl alapján feldolgozásra került,";
    print " az eredmeny.xml tartalmazza a muvelet vegeredmenyet, ami:<br />\n";
    print "<pre>\n";
    readfile('eredmeny.xml');
    print "</pre>\n";
}
else {
    print "Sajnálom, a minta.xml-t nem lehet a minta.xsl alapján feldolgozni,";
    print "amelynek az oka: " . xslt_error($xh);
    print " és a hibakódja: " . xslt_errno($xh);
}

xslt_free($xh);

?>
```

Bár nagyon hasznos ez a müködési elv, sokszor, főképp webes környezetben a kimenetet közvetlenül ki kell iratni, nem pedig fájlba menteni; ezért a harmadik paraméter elhagyása vagy `NULL` értéke esetén, az **xslt_process()** automatikusan az XSLT transzformáció eredményével tér vissza.

Példa 2. Az xslt_process() XML fájl XSL fájl alapú transzformációjának eredménye változóba kerül elmentésre

```
<?php

// új XSLT feldolgozó lefoglalása
$xh = xslt_create();

// dokumentum feldolgozása
$result = xslt_process($xh, 'minta.xml', 'minta.xsl');
if ($result) {
    print "SIKER, a minta.xml a minta.xsl alapján feldolgozásra került,";
    print " amelynek az eredménye a \$result változóba került: <br />\n";
    print "<pre>\n";
    print $result;
    print "</pre>\n";
}
else {
    print "Sajnálom, a minta.xml-t nem lehet a minta.xsl alapján feldolgozni,";
    print "amelynek az oka: " . xslt_error($xh);
    print " és a hibakódja: " . xslt_errno($xh);
}

xslt_free($xh);

?>
```

A két fenti eset a XSLT transzformáció két legegyszerűbb esete, ám merem állítani a leggyakoribb módjai is ennek. Vannak olyan helyzetek azonban, amikor az XML és az XSLT fájlok nem állnak rendelkezésre, hanem azok külső forrásból - pl. adatbázisból vagy hálózaton keresztül érkeznek. Ekkor az XML és XSLT adatok csak változók formájában hozzáférhetők, és piaci alkalmazások esetén a fájlba másolás okozta veszteség nem megengedhető. Ebben segít az XSLT "argument" szintaxisa. XML és XSLT fájlok helyett ún. "argumentum helyfoglalókat" lehet definiálni, amelyek helyett az *`arguments`* (5.) paraméterben megadott tartalmat fogja használni az XSLT feldolgozó. A következő példa egyetlen fájlt sem használ.

Példa 3. Using the xslt_process() to transform a variable containing XML data and a variable containing XSL data into a variable containing the resulting XML data

```
<?php
// $xml és $xsl tartalmazza az XML és XSL adatokat

$arguments = array(
     '/_xml' => $xml,
     '/_xsl' => $xsl
```

```
);

// új XSLT feldolgozó allokálása
$xh = xslt_create();

// a dokumentum feldolgozása
$result = xslt_process($xh, 'arg:/_xml', 'arg:/_xsl', NULL, $arguments);
if ($result) {
    print "SIKER, a minta.xml a minta.xsl alapján feldolgozásra került,";
    print " amelynek az eredménye a \$result változóba került: <br />\n";
    print "<pre>\n";
    print $result;
    print "</pre>\n";
}
else {
    print "Sajnálom, a minta.xml-t nem lehet a minta.xsl alapján feldolgozni,";
    print "amelynek az oka: " . xslt_error($xh);
    print " és a hibakódja: " . xslt_errno($xh);
}

xslt_free($xh);
?>
```

Végezetül, az **xslt_process()** utolsó, *parameters* paraméterében bármlyen adat átadható az XSLT feldolgozónak az XSL dokumentumhoz. Ezeket a paramétereket az XSL-en belül `<xsl:param name="parameter_name">` utasítással érhetők el.

xslt_set_base (PHP 4 >= 4.0.5)

beállítja az alap URI-t mindegyik XSLT feldolgozáshoz

void **xslt_set_base** (resource xh, string uri) \linebreak

Beállítja azt az alap URI-t mindegyik XSLT feldolgozáshoz, amely az Xpath kifejezésekben előforduló `document()` és egyéb külső forrást használó parancsok kezeléséhez szükséges.

xslt_set_encoding (PHP 4 >= 4.0.5)

beállítja az XML dokumentumok kódolását

void **xslt_set_encoding** (resource xh, string encoding) \linebreak

Beállítja az XSLT transzformáció kimenetén használt kódolást. Csak akkor használható, ha a Sablotron kiszolgáló `encoding` támogatással lett fordítva.

xslt_set_error_handler (PHP 4 >= 4.0.4)

beállítja az XSLT feldolgozó hibakezelőjét

void **xslt_set_error_handler** (resource xh, mixed handler) \linebreak

Beállítja az *xh*-ban megadott XSLT feldolgozó hibakezelőjét. Ez a hibakezelő kerül meghívásra mindannyiszor, ha hiba történik az XSLT transzformáció során. Ugyanez a függvény kezeli a megjegyzéseket (notice).

xslt_set_log (PHP 4 >= 4.0.6)

beállítja a naplófájl nevét

void **xslt_set_log** (resource xh, mixed log) \linebreak

xh

Hivatkozás az XSLT elemzőre.

log

Ez a paraméter lehet egy logikai érték, ami ki-bekapcsolja a naplózást, vagy lehet egy sztring is, amely a naplófájl nevét tartalmazza.

Ezzel a függvénnyel beállítható, hogy melyik fájlba kerüljenek az XSLT üzenetek. Ezek az üzenetek abban különböznek a hibaüzenetektől, hogy nem tényleges hibákról értesítnek, hanem inkább az XSLT feldolgozó állapotáról, ezért hibakereséskor hasznosak, ha valami nem jól működik az XSLT körül.

Alapértelmezés szerint a naplózás inaktív, így az engedélyezéshez először logikai paraméterrel kell meghívni az **xslt_set_log()**-ot, és ha a naplózást fájlba szeretnéd átirányítani, akkor egy sztring paraméterrel is meg kell hívni.

Példa 1. XSLT naplózás használata

```
<?php

$xh = xslt_create();
xslt_set_log($xh, TRUE);
xslt_set_log($xh, getcwd() . 'xslt.log');

$result = xslt_process($xh, 'kutya.xml', 'kedvencek.xsl');
print($result);

xslt_free($xh);
?>
```

xslt_set_sax_handler (4.0.3 - 4.0.6 only)

beállítja az XSLT feldolgozó SAX kezelőjét

void **xslt_set_sax_handler** (resource xh, array handlers) \linebreak

Beállítja a SAX kezelőt az *xh*-ban megadott XSLT feldolgozó számára. A SAX kezelőket a *handlers* kétdimenziós tömbben kell megadni a következők szerint (bármelyik legfelső szintű elem elhagyható):

```
array(
[document] =>
    array(
        start document handler,
        end document handler
    ),
[element] =>
    array(
        start element handler,
        end element handler
    ),
[namespace] =>
    array(
        start namespace handler,
        end namespace handler
    ),
[comment] => comment handler,
[pi] => processing instruction handler,
[character] => character data handler
)
```

xslt_set_sax_handlers (PHP 4 >= 4.0.6)

beállítja az XML dokumentumok feldolgozásakor meghívandó SAX kezelőt

void **xslt_set_sax_handlers** (resource processor, array handlers) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

xslt_set_scheme_handler (4.0.5 - 4.0.6 only)

beállítja az XSLT feldolgozó scheme kezelőjét

void **xslt_set_scheme_handler** (resource xh, array handlers) \linebreak

Beállítja az *xh*-ban megadott XSLT feldolgozó `scheme` kezelőjét. A `scheme` kezelőket a *`handlers`* tömbben kell megadni a következők szerint (bármelyik elem elhagyható):

```
array(
[get_all] => get all handler,
[open] => open handler,
[get] => get handler,
[put] => put handler,
[close] => close handler
)
```

xslt_set_scheme_handlers (PHP 4 >= 4.0.6)

beállítja az XSLT feldolgozó scheme kezelőit

void **xslt_set_scheme_handlers** (resource processor, array handlers) \linebreak

Figyelem

Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

CVIII. YAZ functions

Introduction

This extension offers a PHP interface to the YAZ toolkit that implements the Z39.50 protocol for information retrieval. With this extension you can easily implement a Z39.50 origin (client) that searches or scans Z39.50 targets (servers) in parallel.

YAZ is available at http://www.indexdata.dk/yaz/. You can find news information, example scripts, etc. for this extension at http://www.indexdata.dk/phpyaz/.

The module hides most of the complexity of Z39.50 so it should be fairly easy to use. It supports persistent stateless connections very similar to those offered by the various SQL APIs that are available for PHP. This means that sessions are stateless but shared amongst users, thus saving the connect and initialize phase steps in most cases.

Installation

Compile YAZ and install it. Build PHP with your favourite modules and add option `--with-yaz`. Your task is roughly the following:

```
gunzip -c yaz-1.6.tar.gz|tar xf -
gunzip -c php-4.0.X.tar.gz|tar xf -
cd yaz-1.6
./configure --prefix=/usr
make
make install
cd ../php-4.0.X
./configure --with-yaz=/usr/bin
make
make install
```

Example

PHP/YAZ keeps track of connections with targets (Z-Associations). A positive integer represents the ID of a particular association.

Példa 1. Parallel searching using YAZ()

The script below demonstrates the parallel searching feature of the API. When invoked with no arguments it prints a query form; else (arguments are supplied) it searches the targets as given in in array host.

```
$num_hosts = count ($host);
if (empty($term) || count($host) == 0) {
    echo '<form method="get">
    <input type="checkbox"
    name="host[]" value="bagel.indexdata.dk/gils">
        GILS test
    <input type="checkbox"
    name="host[]" value="localhost:9999/Default">
        local test
    <input type="checkbox" checked="1"
    name="host[]" value="z3950.bell-labs.com/books">
        BELL Labs Library
    <br>
    RPN Query:
    <input type="text" size="30" name="term">
    <input type="submit" name="action" value="Search">
    ';
} else {
    echo 'You searced for ' . htmlspecialchars($term) . '<br>';
    for ($i = 0; $i < $num_hosts; $i++) {
        $id[] = yaz_connect($host[$i]);
        yaz_syntax($id[$i],"sutrs");
        yaz_search($id[$i],"rpn",$term);
    }
    yaz_wait();
    for ($i = 0; $i < $num_hosts; $i++) {
        echo '<hr>' . $host[$i] . ":";
        $error = yaz_error($id[$i]);
        if (!empty($error)) {
            echo "Error: $error";
        } else {
            $hits = yaz_hits($id[$i]);
            echo "Result Count $hits";
        }
        echo '<dl>';
        for ($p = 1; $p <= 10; $p++) {
            $rec = yaz_record($id[$i],$p,"string");
            if (empty($rec)) continue;
            echo "<dt><b>$p</b></dt><dd>";
            echo ereg_replace("\n", "<br>\n",$rec);
            echo "</dd>";
        }
        echo '</dl>';
    }
}
```

yaz_addinfo (PHP 4 >= 4.0.1)

Returns additional error information

int **yaz_addinfo** (int id) \linebreak

Returns additional error message for target (last request). An empty string is returned if last operation was a success or if no additional information was provided by the target.

yaz_ccl_conf (PHP 4 >= 4.0.5)

Configure CCL parser

int **yaz_ccl_conf** (int id, array config) \linebreak

This function configures the CCL query parser for a target with definitions of access points (CCL qualifiers) and their mapping to RPN. To map a specific CCL query to RPN afterwards call the yaz_ccl_parse() function. Each index of the array *`config`* is the name of a CCL field and the corresponding value holds a string that specifies a mapping to RPN. The mapping is a sequence of attribute-type, attribute-value pairs. Attribute-type and attribute-value is separated by an equal sign (=). Each pair is separated by white space.

Példa 1. CCL configuration

In the example below, the CCL parser is configured to support three CCL fields: `ti`, `au` and `isbn`. Each field is mapped to their BIB-1 equivalent. It is assumed that variable `$id` is a target ID.

```
$field["ti"] = "1=4";
$field["au"] = "1=1";
$field["isbn"] = "1=7";
yaz_ccl_conf($id,$field);
```

yaz_ccl_parse (PHP 4 >= 4.0.5)

Invoke CCL Parser

int **yaz_ccl_parse** (int id, string query, array & result) \linebreak

This function invokes the CCL parser. It converts a given CCL FIND query to an RPN query which may be passed to the yaz_search() function to perform a search. To define a set of valid CCL fields call yaz_ccl_conf() prior to this function. If the supplied *`query`* was successfully converted to RPN, this function returns `TRUE`, and the index `rpn` of the supplied array *`result`* holds a valid RPN query. If the query could not be converted (because of invalid syntax, unknown field, etc.) this function returns `FALSE` and three indexes are set in the resulting array to indicate the cause of

failure: `errorcode`CCL error code (integer), `errorstring`CCL error string, and `errorpos`approximate position in query of failure (integer is character position).

yaz_close (PHP 4 >= 4.0.1)

Closes a YAZ connection

int **yaz_close** (int id) \linebreak

Closes the Z-association given by *`id`*. The *`id`* is a target ID as returned by a previous yaz_connect() command.

yaz_connect (PHP 4 >= 4.0.1)

Prepares for a connection and Z-association to a Z39.50 target.

int **yaz_connect** (string zurl [, mixed options]) \linebreak

This function returns a positive ID on success; zero on failure.

yaz_connect() prepares for a connection to a Z39.50 target. The zurl argument takes the form host[:port][/database]. If port is omitted 210 is used. If database is omitted Default is used. This function is non-blocking and doesn't attempt to establish a socket - it merely prepares a connect to be performed later when yaz_wait() is called.

If the second argument, *`options`*, is given as a string it is treated as the Z39.50 V2 authentication string (OpenAuth).

If *`options`* is given as an array the contents of the array serves as options. Note that array options are only supported for PHP 4.1.0 and later.

yaz_connect() options

user

Username for authentication.

group

Group for authentication.

password

Password for authentication.

cookie

Cookie for session (YAZ proxy).

proxy

Proxy for connection (YAZ proxy).

persistent

A boolean. If TRUE the connection is persistent; If FALSE the connection is not persistent. By default connections are persistent.

piggyback

A boolean. If TRUE piggyback is enabled for searches; If FALSE piggyback is disabled. By default piggyback is enabled. Enabling piggyback is more efficient and usually saves a network-round-trip for first time fetches of records. However, a few Z39.50 targets doesn't support piggyback or they ignore element set names. For those, piggyback should be disabled.

yaz_database (PHP 4 >= 4.0.6)

Specifies the databases within a session

int **yaz_database** (int id, string databases) \linebreak

This function specifies one or more databases to be used in search, retrieval, etc. - overriding databases specified in call to yaz_connect(). Multiple databases are separated by a plus sign +.

This function allows you to use different sets of databases within a session.

Returns TRUE on success; FALSE on error.

yaz_element (PHP 4 >= 4.0.1)

Specifies Element-Set Name for retrieval

int **yaz_element** (int id, string elementset) \linebreak

This function is used in conjunction with yaz_search() and yaz_present() to specify the element set name for records to be retrieved. Most servers support F (full) and B (brief).

Returns TRUE on success; FALSE on error.

yaz_errno (PHP 4 >= 4.0.1)

Returns error number

int **yaz_errno** (int id) \linebreak

Returns error for target (last request). A positive value is returned if the target returned a diagnostic code; a value of zero is returned if no errors occurred (success); negative value is returned for other errors (such as when the target closed connection, etc).

yaz_errno() should be called after network activity for each target - (after yaz_wait() returns) to determine the success or failure of the last operation (e.g. search).

yaz_error (PHP 4 >= 4.0.1)

Returns error description

string **yaz_error** (int id) \linebreak

Returns error message for target (last request). An empty string is returned if last operation was a success.

yaz_error() returns an english text message corresponding to the last error number as returned by yaz_errno().

yaz_hits (PHP 4 >= 4.0.1)

Returns number of hits for last search

int **yaz_hits** (int id) \linebreak

yaz_hits() returns number of hits for last search.

yaz_itemorder (PHP 4 >= 4.0.5)

Prepares for Z39.50 Item Order with an ILL-Request package

int **yaz_itemorder** (array args) \linebreak

This function prepares for an Extended Services request using the Profile for the Use of Z39.50 Item Order Extended Service to Transport ILL (Profile/1). See this (http://www.nlc-bnc.ca/iso/ill/stanprf.htm) and the specification (http://www.nlc-bnc.ca/iso/ill/document/standard/z-ill-1a.pdf). The args parameter must be a hash array with information about the Item Order request to be sent. The key of the hash is the name of the corresponding ASN.1 tag path. For example, the ISBN below the Item-ID has the key item-id,ISBN.

The ILL-Request parameters are:

protocol-version-num
transaction-id,initial-requester-id,person-or-institution-symbol,person
transaction-id,initial-requester-id,person-or-institution-symbol,institution
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-person
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-institution
transaction-id,transaction-group-qualifier
transaction-id,transaction-qualifier
transaction-id,sub-transaction-qualifier
service-date-time,this,date
service-date-time,this,time
service-date-time,original,date
service-date-time,original,time
requester-id,person-or-institution-symbol,person
requester-id,person-or-institution-symbol,institution
requester-id,name-of-person-or-institution,name-of-person
requester-id,name-of-person-or-institution,name-of-institution

responder-id,person-or-institution-symbol,person
responder-id,person-or-institution-symbol,institution
responder-id,name-of-person-or-institution,name-of-person
responder-id,name-of-person-or-institution,name-of-institution
transaction-type
delivery-address,postal-address,name-of-person-or-institution,name-of-person
delivery-address,postal-address,name-of-person-or-institution,name-of-institution
delivery-address,postal-address,extended-postal-delivery-address
delivery-address,postal-address,street-and-number
delivery-address,postal-address,post-office-box
delivery-address,postal-address,city
delivery-address,postal-address,region
delivery-address,postal-address,country
delivery-address,postal-address,postal-code
delivery-address,electronic-address,telecom-service-identifier
delivery-address,electronic-address,telecom-service-addreess
billing-address,postal-address,name-of-person-or-institution,name-of-person
billing-address,postal-address,name-of-person-or-institution,name-of-institution
billing-address,postal-address,extended-postal-delivery-address
billing-address,postal-address,street-and-number
billing-address,postal-address,post-office-box
billing-address,postal-address,city
billing-address,postal-address,region
billing-address,postal-address,country
billing-address,postal-address,postal-code
billing-address,electronic-address,telecom-service-identifier
billing-address,electronic-address,telecom-service-addreess
ill-service-type
requester-optional-messages,can-send-RECEIVED
requester-optional-messages,can-send-RETURNED
requester-optional-messages,requester-SHIPPED
requester-optional-messages,requester-CHECKED-IN
search-type,level-of-service
search-type,need-before-date
search-type,expiry-date
search-type,expiry-flag
place-on-hold
client-id,client-name
client-id,client-status
client-id,client-identifier
item-id,item-type
item-id,call-number
item-id,author
item-id,title
item-id,sub-title
item-id,sponsoring-body
item-id,place-of-publication
item-id,publisher
item-id,series-title-number
item-id,volume-issue
item-id,edition
item-id,publication-date

item-id,publication-date-of-component
item-id,author-of-article
item-id,title-of-article
item-id,pagination
item-id,ISBN
item-id,ISSN
item-id,additional-no-letters
item-id,verification-reference-source
copyright-complicance
retry-flag
forward-flag
requester-note
forward-note

There are also a few parameters that are part of the Extended Services Request package and the ItemOrder package:

package-name
user-id
contact-name
contact-phone
contact-email
itemorder-item

yaz_present (PHP 4 >= 4.0.5)

Prepares for retrieval (Z39.50 present).

int **yaz_present** (void) \linebreak

This function prepares for retrieval of records after a successful search. The yaz_range() should be called prior to this function to specify the range of records to be retrieved.

yaz_range (PHP 4 >= 4.0.1)

Specifies the maximum number of records to retrieve

int **yaz_range** (int id, int start, int number) \linebreak

This function is used in conjunction with yaz_search() to specify the maximum number of records to retrieve (number) and the first record position (start). If this function is not invoked (only yaz_search()) start is set to 1 and number is set to 10.

Returns TRUE on success; FALSE on error.

yaz_record (PHP 4 >= 4.0.1)

Returns a record

int **yaz_record** (int id, int pos, string type) \linebreak

Returns record at position or empty string if no record exists at given position.

The **yaz_record()** function inspects a record in the current result set at the position specified. If no database record exists at the given position an empty string is returned. The argument, type, specifies the form of the returned record. If type is "string" the record is returned in a string representation suitable for printing (for XML and SUTRS). If type is "array" the record is returned as an array representation (for structured records).

yaz_scan_result (PHP 4 >= 4.0.5)

Returns Scan Response result

array **yaz_scan_result** (int id [, array & result]) \linebreak

Given a target ID this function returns and array with terms as received from the target in the last Scan Response. This function returns an array (0..n-1) where n is the number of terms returned. Each value is a pair where first item is term, second item is result-count. If the *`result`* is given it will be modified to hold additional information taken from the Scan Response: `number` (number of entries returned), `stepsize` (Step-size), `position` (position of term), `status` (Scan Status).

yaz_scan (PHP 4 >= 4.0.5)

Prepares for a scan

int **yaz_scan** (int id, string type, string startterm [, array flags]) \linebreak

This function prepares for a Z39.50 Scan Request. Argument *`id`* specifies target ID. Starting term point for the scan is given by *`startterm`*. The form in which is the starting term is specified is given by *`type`*. Currently type `rpn` is supported. The optional *`flags`* specifies additional information to control the behaviour of the scan request. Three indexes are currently read from the flags: `number` (number of terms requested), `position` (preferred position of term) and `stepSize` (preferred step size). To actually tranfer the Scan Request to the target and receive the Scan Response, yaz_wait() must be called. Upon completion of yaz_wait() call yaz_error() and yaz_scan_result() to handle the response.

The syntax of *`startterm`* is similar to the RPN query as described in yaz_search(). The startterm consists of zero or more `@attr`-operator specifications, then followed by exactly one token.

Példa 1. PHP function that scans titles

```
function scan_titles($id, $starterm) {
  yaz_scan($id,"rpn", "@attr 1=4 " . $starterm);
  yaz_wait();
```

```
  $errno = yaz_errno($id);
  if ($errno == 0) {
    $ar = yaz_scan_result($id,&$options);
    echo 'Scan ok; ';
    $ar = yaz_scan_result($id, &$options);
    while(list($key,$val)=each($options)) {
       echo "$key = $val  ";
    }
    echo '<br><table><tr><td>';
    while(list($key,list($k, $term, $tcount))=each($ar)) {
      if (empty($k)) continue;
      echo "<tr><td>$term</td><td>";
      echo $tcount;
      echo "</td></tr>";
    }
    echo '</table>';
  } else {
    echo "Scan failed. Error: " . yaz_error($id) . "<br>";
  }
}
```

yaz_search (PHP 4 >= 4.0.1)

Prepares for a search

int **yaz_search** (int id, string type, string query) \linebreak

yaz_search() prepares for a search on the target with given id. The type represents the query type - only "rpn" is supported now in which case the third argument specifies a Type-1 query (RPN). Like yaz_connect() this function is non-blocking and only prepares for a search to be executed later when yaz_wait() is called.

The RPN query is a textual represenation of the Type-1 query as defined by the Z39.50 standard. However, in the text representation as used by YAZ a prefix notation is used, that is the operater precedes the operands. The query string is a sequence of tokens where white space is ignored is ignored unless surrounded by double quotes. Tokens beginning with an at-character (@) are considered operators, otherwise they are treated as search terms.

Táblázat 1. RPN Operators

Syntax	Description
`@and query1 query2`	intersection of query1 and query2
`@or query1 query2`	union of query1 and query2
`@not query1 query2`	query1 and not query2
`@set name`	result set reference

Syntax	Description
`@attrset set query`	specifies attribute-set for query. This construction is only allowed once - in the beginning of the whole query
`@attr set type=value query`	applies attribute to query. The type and value are integers specifying the attribute-type and attribute-value respectively. The set, if given, specifies the attribute-set.

Példa 1. Query Examples

Query

`computer`
matches documents where "computer" occur. No attributes are specified.

The Query

`"donald knuth"`
matches documents where "donald knuth" occur.

For the query

`@attr 1=4 art`
attribute type is 1 (Bib-1 use), attribute value is 4 Title), so this should match documents where `art` occur in the title.

Another more complex one:

`@attrset gils @and @attr 1=4 art @attr 1=1003 "donald knuth"`
The query as a whole uses the GILS attributeset. The query matches documents where `art` occur in the title and in which `donald knuth` occur in the author.

yaz_sort (PHP 4 >= 4.1.0)

Sets sorting criteria

int **yaz_sort** (int id, string criteria) \linebreak

This function sets sorting criteria and enables Z39.50 Sort. Use this function together with yaz_search() or yaz_present(). Using this function alone doesn't have any effect. If used in conjunction with yaz_search() a Z39.50 Sort will be sent after a search response has been received and before any records are retrieved with Z39.50 Present. The *`criteria`* takes the form

`field1 flags1 field2 flags2` ...

where field1 specifies primary attributes for sort, field2 seconds, etc.. The field specifies either numerical attribute combinations consisting of type=value pairs separated by comma (e.g. `1=4,2=1`) ; or the field may specify a plain string criteria (e.g. `title`. The flags is a sequnce of the following characters which may not be separated by any white space.

Sort Flags

`a`

Sort ascending

`d`

Sort descending

`i`

Case insensitive sorting

`s`

Case sensitive sorting

Példa 1. Sort Criterias

To sort on Bib1 attribute title, case insensitive, and ascending you'd use the following sort criteria:

```
1=4 ia
```

If the secondary sorting criteria should be author, case sensitive and ascending you'd use:

```
1=4 ia 1=1003 sa
```

yaz_syntax (PHP 4 >= 4.0.1)

Specifies the preferred record syntax for retrieval.

int **yaz_syntax** (int id, string syntax) \linebreak

The syntax is specified as an OID (Object Identifier) in a raw dot-notation (like `1.2.840.10003.5.10`) or as one of the known registered record syntaxes (sutrs, usmarc, grs1, xml, etc.). This function is used in conjunction with yaz_search() and yaz_present() to specify the preferred record syntax for retrieval.

yaz_wait (PHP 4 >= 4.0.1)

Wait for Z39.50 requests to complete

int **yaz_wait** ([array options]) \linebreak

This function carries out networked (blocked) activity for outstanding requests which have been prepared by the functions yaz_connect(), yaz_search(), yaz_present(), yaz_scan() and yaz_itemorder(). **yaz_wait()** returns when all targets have either completed all requests or aborted (in case of errors).

If the *`options`* array is given that holds options that change the behaviour of **yaz_wait()**.

`timeout`

Sets timeout in seconds. If a target hasn't responded within the timeout it is considered dead and **yaz_wait()** returns. The default value for timeout is 15 seconds.

CIX. YP/NIS Functions

NIS (formerly called Yellow Pages) allows network management of important administrative files (e.g. the password file). For more information refer to the NIS manpage and Introduction to YP/NIS (http://www.desy.de/~sieversm/ypdoku/ypdoku/ypdoku.html). There is also a book called Managing NFS and NIS (http://www.oreilly.com/catalog/nfs/noframes.html) by Hal Stern.

To get these functions to work, you have to configure PHP with `--with-yp`(PHP 3) or `--enable-yp`(PHP 4).

yp_all (PHP 4 >= 4.0.6)

Traverse the map and call a function on each entry

void **yp_all** (string domain, string map, string callback) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

yp_cat (PHP 4 >= 4.0.6)

Return an array containing the entire map

array **yp_cat** (string domain, string map) \linebreak

> **Figyelem**
>
> Ez a függvény jelenleg nincs dokumentálva, csak a paraméterek listája található itt.

yp_err_string (PHP 4 >= 4.0.6)

Returns the error string associated with the previous operation

string **yp_err_string** (void) \linebreak

yp_err_string() returns the error message associated with the previous operation. Useful to indicate what exactly went wrong.

Példa 1. Example for NIS errors

```
<?php
    echo "Error: " . yp_err_string();
?>
```

See also yp_errno().

yp_errno (PHP 4 >= 4.0.6)

Returns the error code of the previous operation

int **yp_errno** (void) \linebreak

yp_errno() returns the error code of the previous operation.

Possible errors are:

1 args to function are bad
2 RPC failure - domain has been unbound
3 can’t bind to server on this domain
4 no such map in server’s domain
5 no such key in map
6 internal yp server or client error
7 resource allocation failure
8 no more records in map database
9 can’t communicate with portmapper
10 can’t communicate with ypbind
11 can’t communicate with ypserv
12 local domain name not set
13 yp database is bad
14 yp version mismatch
15 access violation
16 database busy

See also yp_err_string().

yp_first (PHP 3>= 3.0.7, PHP 4)

Returns the first key-value pair from the named map

array **yp_first** (string domain, string map) \linebreak

yp_first() returns the first key-value pair from the named map in the named domain, otherwise FALSE.

Példa 1. Example for the NIS first

```
<?php
$entry = yp_first($domain, "passwd.byname");
$key = $entry ["key"];
$value = $entry ["value"];
echo "First entry in this map has key " . $key . " and value " . $value;
?>
```

See also yp-get-default-domain()

yp_get_default_domain (PHP 3>= 3.0.7, PHP 4)

Fetches the machine’s default NIS domain

int **yp_get_default_domain** (void) \linebreak

yp_get_default_domain() returns the default domain of the node or FALSE. Can be used as the domain parameter for successive NIS calls.

A NIS domain can be described a group of NIS maps. Every host that needs to look up information binds itself to a certain domain. Refer to the documents mentioned at the beginning for more detailed information.

Példa 1. Example for the default domain

```
<?php
$domain = yp_get_default_domain();
echo "Default NIS domain is: " . $domain;
?>
```

yp_master (PHP 3>= 3.0.7, PHP 4)

Returns the machine name of the master NIS server for a map

string **yp_master** (string domain, string map) \linebreak

yp_master() returns the machine name of the master NIS server for a map.

Példa 1. Example for the NIS master

```
<?php
$number = yp_master ($domain, $mapname);
echo "Master for this map is: " . $master;
?>
```

See also yp-get-default-domain().

yp_match (PHP 3>= 3.0.7, PHP 4)

Returns the matched line

string **yp_match** (string domain, string map, string key) \linebreak

yp_match() returns the value associated with the passed key out of the specified map or FALSE. This key must be exact.

Példa 1. Example for NIS match

```
<?php
$entry = yp_match ($domain, "passwd.byname", "joe");
echo "Matched entry is: " . $entry;
?>
```

In this case this could be: joe:##joe:11111:100:Joe User:/home/j/joe:/usr/local/bin/bash

See also yp-get-default-domain()

yp_next (PHP 3>= 3.0.7, PHP 4)

Returns the next key-value pair in the named map.

array **yp_next** (string domain, string map, string key) \linebreak

yp_next() returns the next key-value pair in the named map after the specified key or FALSE.

Példa 1. Example for NIS next

```
<?php
$entry = yp_next ($domain, "passwd.byname", "joe");

if (!$entry) {
echo "No more entries found\n";
    <!-- echo yp_errno() . ": " . yp_err_string(); -->
}

$key = key ($entry);

echo "The next entry after joe has key " . $key
      . " and value " . $entry[$key];
?>
```

See also yp-get-default-domain().

yp_order (PHP 3>= 3.0.7, PHP 4)

Returns the order number for a map

int **yp_order** (string domain, string map) \linebreak

yp_order() returns the order number for a map or FALSE.

Példa 1. Example for the NIS order

```
<?php
    $number = yp_order($domain,$mapname);
    echo "Order number for this map is: " . $number;
?>
```

See also yp-get-default-domain().

CX. Zip Fájl függvények (csak olvasáshoz)

Ez a modul a ZZIPlib (http://zziplib.sourceforge.net/) könyvtár függvényeit használja, a zip eljárással tömörített állományok és a bennük levő fájlok olvasásához. A ZZIPlib-et Guido Draheimnek köszönhetjük.

Vedd figyelembe, hogy a ZZIPlib csak a teljes zip imlementációnak megfelelő tömörített zip fájlokat kezeli. A zip programkészletre is szükség van ahhoz, hogy zip fájlokat olvasson a modul.

A zip támogatás nincs beépítve a PHP-ben, a --with-zip opcióval kell fordítanod, hogy használhasd. Szükség van még a ZZIPlib 0.10.6 vagy későbbi kiadására is.

> **Megjegyzés:** A zip támogatás a PHP 4.1.0-nál régebbi verziókban csak kísérleti jellegű, tehát ezekben nem működik megbízhatóan. Ez a dokumentum a 4.1.0 és az újabb PHP verziók zip modulját mutatja be.

Példa

Az alábbi példa megnyit egy zip állományt, beolvassa a benne található fájlokat és kinyomtatja tartalmukat. A példában felhasznált `test2.zip` állomány a ZZIPlib csomag egyik "beépített" próba állománya.

Példa 1. Példaprogram

```
<?php

$zip = zip_open("/tmp/test2.zip");

if ($zip) {

    while ($zip_entry = zip_read($zip)) {
        echo "Név:               " . zip_entry_name($zip_entry) . "\n";
        echo "Tömörítetlen méret: " . zip_entry_filesize($zip_entry) . "\n";
        echo "Tömörített méret:   " . zip_entry_compressedsize($zip_entry) . "\n";
        echo "Tömörítési metódus: " . zip_entry_compressionmethod($zip_entry) . "\n";

        if (zip_entry_open($zip, $zip_entry, "r")) {
            echo "A fájl tartalma:\n";
            $buf = zip_entry_read($zip_entry, zip_entry_filesize($zip_entry));
            echo "$buf\n";
            zip_entry_close($zip_entry);
        }
        echo "\n";

    }

    zip_close($zip);

}

?>
```

A program először megnyitja a megadott útvonalon található `test2.zip` fájlt. A ciklust addig végzi, amíg a a zip_read() egy érvényes `$zip_entry` (vagyis a zip állományban létező fájl vagy könyvtár) értékkel tér vissza. A zip_read() tulajdonképpen egy tömb eléréseként képzelhető, amely tömbbe a zip_open() betölti a megnyitott zip állományban lévő fájlok neveit. Minden egyes ciklusnál a program automatikusan előre lépteti a tömböt eggyel.

zip_close (PHP 4 >= 4.1.0)

Lezár egy megnyitott ZIP állományt

void **zip_close** (resource zip) \linebreak

Lezár egy megnyitott ZIP állományt. Ez csak akkor működik, ha a `zip` forrásfájlt előzőleg a zip_open() függvénnyel nyitottuk meg.

Ennek a függvénynek nincs visszatérési értéke.

Lásd még: zip_open() és zip_read().

zip_entry_close (PHP 4 >= 4.1.0)

Lezár egy ZIP állományban található objektumot

void **zip_entry_close** (resource zip_entry) \linebreak

Lezárja a zip állományban lévő `zip_entry` objektumot. A `zip_entry` paraméter egy létező objektum a zip fájlban, amit a zip_entry_open() függvénnyel nyitottunk meg.

Ennek a függvénynek nincs visszatérési értéke.

Lásd még: zip_entry_open() és zip_entry_read().

zip_entry_compressedsize (PHP 4 >= 4.1.0)

Egy zip-ben lévő objektum tömörített méretét adja vissza

int **zip_entry_compressedsize** (resource zip_entry) \linebreak

Az adott `zip_entry` objektum tömörített méretével tér vissza. A `zip_entry` egy létező zip állománybeli objektumra mutat, amihez a zip_read() függvény segítségével jutunk.

Lásd még: zip_open() és zip_read().

zip_entry_compressionmethod (PHP 4 >= 4.1.0)

A megadott zip objektum tömörítési típusával tér vissza

string **zip_entry_compressionmethod** (resource zip_entry) \linebreak

A megadott `zip_entry` objektum tömörítési metódusának típusával tér vissza. A paraméter egy létező zip állománybeli objektumra mutat, amihez a zip_read() függvény segítségével jutottunk.

Lásd még: zip_open() és zip_read().

zip_entry_filesize (PHP 4 >= 4.1.0)

Az adott zip objektum tömörítetlen fájlméretét adja vissza

int **zip_entry_filesize** (resource zip_entry) \linebreak

Visszatér az adott *`zip_entry`* objektum tömörítetlen fájlméretének értékével. Ez a *`zip_entry`* paraméter egy létező zip objektum ami a zip_read() függvény visszatérési értéke.

Lásd még: zip_open() és zip_read().

zip_entry_name (PHP 4 >= 4.1.0)

A zip objektum nevével tér vissza

string **zip_entry_name** (resource zip_entry) \linebreak

A *`zip_entry`* paraméterrel megadott zip objektum nevével tér vissza. A *`zip_entry`* paraméter egy létező zip objektum, amit a zip_read() függvénnyel olvashatunk ki a zip állományból.

Lásd még: zip_open() és zip_read().

zip_entry_open (PHP 4 >= 4.1.0)

Olvasásra megnyit egy zip objektumot

bool **zip_entry_open** (resource zip, resource zip_entry [, string mode]) \linebreak

Olvasásra megnyit egy létező zip objektumot egy zip állományból. A *`zip`* paraméter egy létező zip fájl, amit a zip_open() függvénnyel nyitottunk meg. Az olvasni kívánt objektumot a *`zip_entry`* definiálja, ami a zip_read() függvény visszatérési értéke. A *`mode`* nem kötelező paraméter, lásd az fopen() dokumentációját a lehetséges értékekért.

> **Megjegyzés:** Jelenleg a *`mode`* paramétert nem veszi figyelembe a PHP, csak az `"rb"` alapértelmezett értékkel dolgozik. Emiatt a PHP-ben csak olvasási hozzáférést kapunk a zip állományokhoz. Az fopen() dokumentációja részletesen magyarázza a különböző fájl kapcsolókat, köztük az `"rb"`-t is.

Ez a függvény siker esetén logikai IGAZ (`TRUE`), kudarcnál logikai HAMIS (`FALSE`) értékeket vesz fel.

> **Megjegyzés:** Ellentétben az fopen() és más hasonló függvényekkel, a **zip_entry_open()** függvény visszatérési értéke csak a művelet eredményességét mutatja és nincs szükség a vizsgált zip objektum olvasására vagy lezárására.

Lásd még: zip_entry_read() és zip_entry_close().

zip_entry_read (PHP 4 >= 4.1.0)

Olvas egy megnyitott zip objektumból

string **zip_entry_read** (resource zip_entry [, int length]) \linebreak

A *`length`* paraméterben megadott bájt hosszúságú részt olvas be egy megnyitott zip objektumból. Ha a *`length`* paraméter nincs megadva, az alapértelmezés 1024 bájt. A *`zip_entry`* paraméter egy létező zip objektum, ami a zip_read() függvény visszatérési értéke.

Megjegyzés: A *`length`* paramétert tömörítetlen méretben kell érteni!

Az olvasott adattal vagy (pl. hibás adatok esetén) logikai HAMIS (FALSE) értékkel tér vissza, ha elérte a fájl végét.

Lásd még: zip_entry_open(), zip_entry_close() és zip_entry_filesize().

zip_open (PHP 4 >= 4.1.0)

Megnyit egy zip állományt

resource **zip_open** (string filename) \linebreak

Olvasásra megnyit egy zip állományt. A *`filename`* az olvasni kívánt zip állomány neve.

Ezzel a függvénnyel nyitjuk meg a zip állományokat további feldolgozásra. Miután megnyitottuk, a zip_read() függvénnyel olvashatjuk ill. a zip_close() függvénnyel zárhatjuk le az állományt. A **zip_open()** logikai HAMIS (FALSE) értékkel tér vissza, ha a *`filename`* paraméterben megadott fájl nem létezik.

Lásd még: zip_read() és zip_close().

zip_read (PHP 4 >= 4.1.0)

Beolvassa a soron következő zip objektumot egy zip állományban

resource **zip_read** (resource zip) \linebreak

Beolvassa a soron következő zip objektumot egy zip állományból. A *`zip`* paraméter egy létező zip állomány, amit előzőleg megnyitottunk a zip_open() függvénnyel.

Ez a függvény az aktuális zip objektumot feldolgozhatóvá teszi a **zip_entry_...()** függvények számára.

Lásd még: zip_open(), zip_close(), zip_entry_open(), és zip_entry_read().

CXI. Zlib Compression Functions

This module uses the functions of zlib (http://www.gzip.org/zlib/) by Jean-loup Gailly and Mark Adler to transparently read and write gzip (.gz) compressed files. You have to use a zlib version >= 1.0.9 with this module.

This module contains versions of most of the filesystem functions which work with gzip-compressed files (and uncompressed files, too, but not with sockets).

> **Megjegyzés:** Version 4.0.4 introduces a fopen-wrapper for .gz-files, so that you can use a special 'zlib:' URL to access compressed files transparently using the normal f*() file access functions if you prepend the filename or path with a 'zlib:' prefix when calling fopen().
>
> In version 4.3.0, this special prefix has been changed to 'zlib://' to prevent ambiguities with filenames containing ':'.
>
> This feature requires a C runtime library that provides the `fopencookie()` function. To my current knowledge the GNU libc is the only library that provides this feature.

Small code example

Opens a temporary file and writes a test string to it, then it prints out the content of this file twice.

Példa 1. Small Zlib Example

```
<?php

$filename = tempnam ('/tmp', 'zlibtest').'.gz';
print "<html>\n<head></head>\n<body>\n<pre>\n";
$s = "Only a test, test, test, test, test, test, test, test!\n";

// open file for writing with maximum compression
$zp = gzopen($filename, "w9");

// write string to file
gzwrite($zp, $s);

// close file
gzclose($zp);

// open file for reading
$zp = gzopen($filename, "r");

// read 3 char
print gzread($zp, 3);

// output until end of the file and close it.
gzpassthru($zp);

print "\n";

// open file and print content (the 2nd time).
if (readgzfile($filename) != strlen($s)) {
```

```
        echo "Error with zlib functions!";
}
unlink($filename);
print "</pre>\n</h1></body>\n</html>\n";

?>
```

gzclose (PHP 3, PHP 4)

Close an open gz-file pointer

int **gzclose** (int zp) \linebreak

The gz-file pointed to by zp is closed.

Returns TRUE on success and FALSE on failure.

The gz-file pointer must be valid, and must point to a file successfully opened by gzopen().

gzcompress (PHP 4 >= 4.0.1)

Compress a string

string **gzcompress** (string data [, int level]) \linebreak

This function returns a compressed version of the input *`data`* using the ZLIB data format, or FALSE if an error is encountered. The optional parameter *`level`* can be given as 0 for no compression up to 9 for maximum compression.

For details on the ZLIB compression algorithm see the document "ZLIB Compressed Data Format Specification version 3.3 (http://www.ietf.org/rfc/rfc1950.txt)" (RFC 1950).

> **Megjegyzés:** This is *not* the same as gzip compression, which includes some header data. See gzencode() for gzip compression.

See also gzdeflate(), gzinflate(), gzuncompress(), gzencode().

gzdeflate (PHP 4 >= 4.0.4)

Deflate a string

string **gzdeflate** (string data [, int level]) \linebreak

This function returns a compressed version of the input *`data`* using the DEFLATE data format, or FALSE if an error is encountered. The optional parameter *`level`* can be given as 0 for no compression up to 9 for maximum compression.

For details on the DEFLATE compression algorithm see the document "DEFLATE Compressed Data Format Specification version 1.3 (http://www.ietf.org/rfc/rfc1951.txt)" (RFC 1951).

See also gzinflate(), gzcompress(), gzuncompress(), gzencode().

gzencode (PHP 4 >= 4.0.4)

Create a gzip compressed string

string **gzencode** (string data [, int level [, int encoding_mode]]) \linebreak

This function returns a compressed version of the input *`data`* compatible with the output of the **gzip** program, or `FALSE` if an error is encountered. The optional parameter *`level`* can be given as 0 for no compression up to 9 for maximum compression, if not given the default compression level will be the default compression level of the zlib library.

You can also give `FORCE_GZIP` (the default) or `FORCE_DEFLATE` as optional third paramter *`encoding_mode`*. If you use `FORCE_DEFLATE`, you get a standard zlib deflated string (inclusive zlib headers) after the gzip file header but without the trailing crc32 checksum.

Megjegyzés: *`level`* was added in PHP 4.2, before PHP 4.2 **gzencode()** only had the *`data`* and (optional) *`encoding_mode`* parameters..

The resulting data contains the appropriate headers and data structure to make a standard .gz file, e.g.:

Példa 1. Creating a gzip file

```
<?php
    $data = implode("", file("bigfile.txt"));
    $gzdata = gzencode($data, 9);
    $fp = fopen("bigfile.txt.gz", "w");
    fwrite($fp, $gzdata);
    fclose($fp);
?>
```

For more information on the GZIP file format, see the document: GZIP file format specification version 4.3 (http://www.ietf.org/rfc/rfc1952.txt) (RFC 1952).

See also gzcompress(). gzuncompress(), gzdeflate(), gzinflate().

gzeof (PHP 3, PHP 4)

Test for end-of-file on a gz-file pointer

int **gzeof** (int zp) \linebreak

Returns `TRUE` if the gz-file pointer is at EOF or an error occurs; otherwise returns `FALSE`.

The gz-file pointer must be valid, and must point to a file successfully opened by gzopen().

gzfile (PHP 3, PHP 4)

Read entire gz-file into an array

array **gzfile** (string filename [, int use_include_path]) \linebreak

Identical to readgzfile(), except that **gzfile()** returns the file in an array.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also readgzfile(), and gzopen().

gzgetc (PHP 3, PHP 4)

Get character from gz-file pointer

string **gzgetc** (int zp) \linebreak

Returns a string containing a single (uncompressed) character read from the file pointed to by zp. Returns FALSE on EOF (as does gzeof()).

The gz-file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzopen(), and gzgets().

gzgets (PHP 3, PHP 4)

Get line from file pointer

string **gzgets** (int zp, int length) \linebreak

Returns a (uncompressed) string of up to length - 1 bytes read from the file pointed to by fp. Reading ends when length - 1 bytes have been read, on a newline, or on EOF (whichever comes first).

If an error occurs, returns FALSE.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzopen(), gzgetc(), and fgets().

gzgetss (PHP 3, PHP 4)

Get line from gz-file pointer and strip HTML tags

string **gzgetss** (int zp, int length [, string allowable_tags]) \linebreak

Identical to gzgets(), except that **gzgetss()** attempts to strip any HTML and PHP tags from the text it reads.

You can use the optional third parameter to specify tags which should not be stripped.

Megjegyzés: *Allowable_tags* was added in PHP 3.0.13, PHP4B3.

See also gzgets(), gzopen(), and strip_tags().

gzinflate (PHP 4 >= 4.0.4)

Inflate a deflated string

string **gzinflate** (string data [, int length]) \linebreak

This function takes *data* compressed by gzdeflate() and returns the original uncompressed data or FALSE on error. The function will return an error if the uncompressed data is more than 256 times the length of the compressed input *data* or more than the optional parameter *length*.

See also gzcompress(). gzuncompress(), gzdeflate(), gzencode().

gzopen (PHP 3, PHP 4)

Open gz-file

int **gzopen** (string filename, string mode [, int use_include_path]) \linebreak

Opens a gzip (.gz) file for reading or writing. The mode parameter is as in fopen() ("rb" or "wb") but can also include a compression level ("wb9") or a strategy: ’f’ for filtered data as in "wb6f", ’h’ for Huffman only compression as in "wb1h". (See the description of deflateInit2 in zlib.h for more information about the strategy parameter.)

gzopen() can be used to read a file which is not in gzip format; in this case gzread() will directly read from the file without decompression.

gzopen() returns a file pointer to the file opened, after that, everything you read from this file descriptor will be transparently decompressed and what you write gets compressed.

If the open fails, the function returns FALSE.

You can use the optional third parameter and set it to "1", if you want to search for the file in the include_path, too.

Példa 1. gzopen() Example

```
$fp = gzopen ("/tmp/file.gz", "r");
```

See also gzclose().

gzpassthru (PHP 3, PHP 4)

Output all remaining data on a gz-file pointer

int **gzpassthru** (int zp) \linebreak

Reads to EOF on the given gz-file pointer and writes the (uncompressed) results to standard output.

If an error occurs, returns FALSE.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

The gz-file is closed when **gzpassthru()** is done reading it (leaving *zp* useless).

gzputs (PHP 3, PHP 4)

Write to a gz-file pointer

int **gzputs** (int zp, string str [, int length]) \linebreak

gzputs() is an alias to gzwrite(), and is identical in every way.

gzread (PHP 3, PHP 4)

Binary-safe gz-file read

string **gzread** (int zp, int length) \linebreak

gzread() reads up to *length* bytes from the gz-file pointer referenced by *zp*. Reading stops when *length* (uncompressed) bytes have been read or EOF is reached, whichever comes first.

```
// get contents of a gz-file into a string
$filename = "/usr/local/something.txt.gz";
$zd = gzopen ($filename, "r");
$contents = gzread ($zd, 10000);
gzclose ($zd);
```

See also gzwrite(), gzopen(), gzgets(), gzgetss(), gzfile(), and gzpassthru().

gzrewind (PHP 3, PHP 4)

Rewind the position of a gz-file pointer

int **gzrewind** (int zp) \linebreak

Sets the file position indicator for zp to the beginning of the file stream.

If an error occurs, returns 0.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzseek() and gztell().

gzseek (PHP 3, PHP 4)

Seek on a gz-file pointer

int **gzseek** (int zp, int offset) \linebreak

Sets the file position indicator for the file referenced by zp to offset bytes into the file stream. Equivalent to calling (in C) `gzseek( zp, offset, SEEK_SET )`.

If the file is opened for reading, this function is emulated but can be extremely slow. If the file is opened for writing, only forward seeks are supported; gzseek then compresses a sequence of zeroes up to the new starting position.

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

See also gztell() and gzrewind().

gztell (PHP 3, PHP 4)

Tell gz-file pointer read/write position

int **gztell** (int zp) \linebreak

Returns the position of the file pointer referenced by *`zp`*; i.e., its offset into the file stream.

If an error occurs, returns `FALSE`.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzopen(), gzseek() and gzrewind().

gzuncompress (PHP 4 >= 4.0.1)

Uncompress a deflated string

string **gzuncompress** (string data [, int length]) \linebreak

This function takes *`data`* compressed by gzcompress() and returns the original uncompressed data or `FALSE` on error. The function will return an error if the uncompressed data is more than 256 times the length of the compressed input *`data`* or more than the optional parameter *`length`*.

See also gzdeflate(), gzinflate(), gzcompress(), gzencode().

gzwrite (PHP 3, PHP 4)

Binary-safe gz-file write

int **gzwrite** (int zp, string string [, int length]) \linebreak

gzwrite() writes the contents of *`string`* to the gz-file stream pointed to by *`zp`*. If the *`length`* argument is given, writing will stop after *`length`* (uncompressed) bytes have been written or the end of *`string`* is reached, whichever comes first.

Note that if the *length* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *string*.

See also gzread(), gzopen(), and gzputs().

readgzfile (PHP 3, PHP 4)

Output a gz-file

int **readgzfile** (string filename [, int use_include_path]) \linebreak

Reads a file, decompresses it and writes it to standard output.

readgzfile() can be used to read a file which is not in gzip format; in this case **readgzfile()** will directly read from the file without decompression.

Returns the number of (uncompressed) bytes read from the file. If an error occurs, FALSE is returned and unless the function was called as @readgzfile, an error message is printed.

The file *filename* will be opened from the filesystem and its contents written to standard output.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also gzpassthru(), gzfile(), and gzopen().

Rész V. Extending PHP 4.0

Fejezet 25. Overview

"Extending PHP" is easier said than done. PHP has evolved to a full-fledged tool consisting of a few megabytes of source code, and to hack a system like this quite a few things have to be learned and considered. When structuring this chapter, we finally decided on the "learn by doing" approach. This is not the most scientific and professional approach, but the method that's the most fun and gives the best end results. In the following sections, you'll learn quickly how to get the most basic extensions to work almost instantly. After that, you'll learn about Zend's advanced API functionality. The alternative would have been to try to impart the functionality, design, tips, tricks, etc. as a whole, all at once, thus giving a complete look at the big picture before doing anything practical. Although this is the "better" method, as no dirty hacks have to be made, it can be very frustrating as well as energy- and time-consuming, which is why we've decided on the direct approach.

Note that even though this chapter tries to impart as much knowledge as possible about the inner workings of PHP, it's impossible to really give a complete guide to extending PHP that works 100% of the time in all cases. PHP is such a huge and complex package that its inner workings can only be understood if you make yourself familiar with it by practicing, so we encourage you to work with the source.

What Is Zend? and What Is PHP?

The name *Zend* refers to the language engine, PHP's core. The term *PHP* refers to the complete system as it appears from the outside. This might sound a bit confusing at first, but it's not that complicated (see 25-1 Ábra). To implement a Web script interpreter, you need three parts:

1. The *interpreter* part analyzes the input code, translates it, and executes it.
2. The *functionality* part implements the functionality of the language (its functions, etc.).
3. The *interface* part talks to the Web server, etc.

Zend takes part 1 completely and a bit of part 2; PHP takes parts 2 and 3. Together they form the complete PHP package. Zend itself really forms only the language core, implementing PHP at its very basics with some predefined functions. PHP contains all the modules that actually create the language's outstanding capabilities.

Ábra 25-1. The internal structure of PHP.

The following sections discuss where PHP can be extended and how it's done.

Fejezet 26. Extension Possibilities

As shown in 25-1 Ábra above, PHP can be extended primarily at three points: external modules, built-in modules, and the Zend engine. The following sections discuss these options.

External Modules

External modules can be loaded at script runtime using the function dl(). This function loads a shared object from disk and makes its functionality available to the script to which it's being bound. After the script is terminated, the external module is discarded from memory. This method has both advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
External modules don't require recompiling of PHP.	The shared objects need to be loaded every time a script is being executed (every hit), which is very slow.
The size of PHP remains small by "outsourcing" certain functionality.	External additional files clutter up the disk.
	Every script that wants to use an external module's functionality has to specifically include a call to dl(), or the `extension` tag in `php.ini` needs to be modified (which is not always a suitable solution).

To sum up, external modules are great for third-party products, small additions to PHP that are rarely used, or just for testing purposes. To develop additional functionality quickly, external modules provide the best results. For frequent usage, larger implementations, and complex code, the disadvantages outweigh the advantages.

Third parties might consider using the `extension` tag in `php.ini` to create additional external modules to PHP. These external modules are completely detached from the main package, which is a very handy feature in commercial environments. Commercial distributors can simply ship disks or archives containing only their additional modules, without the need to create fixed and solid PHP binaries that don't allow other modules to be bound to them.

Built-in Modules

Built-in modules are compiled directly into PHP and carried around with every PHP process; their functionality is instantly available to every script that's being run. Like external modules, built-in modules have advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
No need to load the module specifically; the functionality is instantly available.	Changes to built-in modules require recompiling of PHP.
No external files clutter up the disk; everything resides in the PHP binary.	The PHP binary grows and consumes more memory.

Built-in modules are best when you have a solid library of functions that remains relatively unchanged, requires better than poor-to-average performance, or is used frequently by many scripts

on your site. The need to recompile PHP is quickly compensated by the benefit in speed and ease of use. However, built-in modules are not ideal when rapid development of small additions is required.

The Zend Engine

Of course, extensions can also be implemented directly in the Zend engine. This strategy is good if you need a change in the language behavior or require special functions to be built directly into the language core. In general, however, modifications to the Zend engine should be avoided. Changes here result in incompatibilities with the rest of the world, and hardly anyone will ever adapt to specially patched Zend engines. Modifications can’t be detached from the main PHP sources and are overridden with the next update using the "official" source repositories. Therefore, this method is generally considered bad practice and, due to its rarity, is not covered in this book.

Fejezet 27. Source Layout

> **Megjegyzés:** Prior to working through the rest of this chapter, you should retrieve clean, unmodified source trees of your favorite Web server. We're working with Apache (available at http://www.apache.org/) and, of course, with PHP (available at http://www.php.net/ - does it need to be said?).
>
> Make sure that you can compile a working PHP environment by yourself! We won't go into this issue here, however, as you should already have this most basic ability when studying this chapter.

Before we start discussing code issues, you should familiarize yourself with the source tree to be able to quickly navigate through PHP's files. This is a must-have ability to implement and debug extensions.

After extracting the PHP archive, you'll see a directory layout similar to that in 27-1 Ábra.

Ábra 27-1. Main directory layout of the PHP source tree.

The following table describes the contents of the major directories.

Directory	Contents
`php-4`	Main PHP source files and main header files; here you'll find all of PHP's API definitions, macros, etc. (imp
`ext`	Repository for dynamic and built-in modules; by default, these are the "official" PHP modules that have bee
`pear`	Directory for the PHP class repository. At the time of this writing, this is still in the design phase, but it's bei
`sapi`	Contains the code for the different server abstraction layers.
`TSRM`	Location of the "Thread Safe Resource Manager" (TSRM) for Zend and PHP.
`Zend`	Location of Zend's file; here you'll find all of Zend's API definitions, macros, etc. (important).

Discussing all the files included in the PHP package is beyond the scope of this chapter. However, you should take a close look at the following files:

- `php.h`, located in the main PHP directory. This file contains most of PHP's macro and API definitions.
- `zend.h`, located in the main Zend directory. This file contains most of Zend's macros and definitions.
- `zend_API.h`, also located in the Zend directory, which defines Zend's API.

You should also follow some sub-inclusions from these files; for example, the ones relating to the Zend executor, the PHP initialization file support, and such. After reading these files, take the time to navigate around the package a little to see the interdependencies of all files and modules - how they

relate to each other and especially how they make use of each other. This also helps you to adapt to the coding style in which PHP is authored. To extend PHP, you should quickly adapt to this style.

Extension Conventions

Zend is built using certain conventions; to avoid breaking its standards, you should follow the rules described in the following sections.

Macros

For almost every important task, Zend ships predefined macros that are extremely handy. The tables and figures in the following sections describe most of the basic functions, structures, and macros. The macro definitions can be found mainly in `zend.h` and `zend_API.h`. We suggest that you take a close look at these files after having studied this chapter. (Although you can go ahead and read them now, not everything will make sense to you yet.)

Memory Management

Resource management is a crucial issue, especially in server software. One of the most valuable resources is memory, and memory management should be handled with extreme care. Memory management has been partially abstracted in Zend, and you should stick to this abstraction for obvious reasons: Due to the abstraction, Zend gets full control over all memory allocations. Zend is able to determine whether a block is in use, automatically freeing unused blocks and blocks with lost references, and thus prevent memory leaks. The functions to be used are described in the following table:

Function	Description
emalloc()	Serves as replacement for **malloc()**.
efree()	Serves as replacement for **free()**.
estrdup()	Serves as replacement for **strdup()**.
estrndup()	Serves as replacement for **strndup()**. Faster than **estrdup()** and binary-safe. This is the recommended funct
ecalloc()	Serves as replacement for **calloc()**.
erealloc()	Serves as replacement for **realloc()**.

emalloc(), **estrdup()**, **estrndup()**, **ecalloc()**, and **erealloc()** allocate internal memory; **efree()** frees these previously allocated blocks. Memory handled by the **e*()** functions is considered local to the current process and is discarded as soon as the script executed by this process is terminated.

> **Figyelem**
>
> To allocate resident memory that survives termination of the current script, you can use **malloc()** and **free()**. This should only be done with extreme care, however, and only in conjunction with demands of the Zend API; otherwise, you risk memory leaks.

Zend also features a thread-safe resource manager to provide better native support for multithreaded

Web servers. This requires you to allocate local structures for all of your global variables to allow concurrent threads to be run. Because the thread-safe mode of Zend was not finished back when this was written, it is not yet extensively covered here.

Directory and File Functions

The following directory and file functions should be used in Zend modules. They behave exactly like their C counterparts, but provide virtual working directory support on the thread level.

Zend Function	Regular C Function
V_GETCWD()	getcwd()
V_FOPEN()	fopen()
V_OPEN()	**open()**
V_CHDIR()	chdir()
V_GETWD()	**getwd()**
V_CHDIR_FILE()	Takes a file path as an argument and changes the current working directory to that file's directory.
V_STAT()	stat()
V_LSTAT()	lstat()

String Handling

Strings are handled a bit differently by the Zend engine than other values such as integers, Booleans, etc., which don't require additional memory allocation for storing their values. If you want to return a string from a function, introduce a new string variable to the symbol table, or do something similar, you have to make sure that the memory the string will be occupying has previously been allocated, using the aforementioned **e*()** functions for allocation. (This might not make much sense to you yet; just keep it somewhere in your head for now - we'll get back to it shortly.)

Complex Types

Complex types such as arrays and objects require different treatment. Zend features a single API for these types - they're stored using hash tables.

> **Megjegyzés:** To reduce complexity in the following source examples, we're only working with simple types such as integers at first. A discussion about creating more advanced types follows later in this chapter.

Fejezet 28. PHP’s Automatic Build System

PHP 4 features an automatic build system that's very flexible. All modules reside in a subdirectory of the `ext` directory. In addition to its own sources, each module consists of an M4 file (for example, see http://www.gnu.org/manual/m4/html_mono/m4.html) for configuration and a Makefile.in file, which is responsible for compilation (the results of autoconf and automake; for example, see http://sourceware.cygnus.com/autoconf/autoconf.html and http://sourceware.cygnus.com/automake/automake.html).

Both files are generated automatically, along with `.cvsignore`, by a little shell script named `ext_skel` that resides in the `ext` directory. As argument it takes the name of the module that you want to create. The shell script then creates a directory of the same name, along with the appropriate `config.m4` and `Makefile.in` files.

Step by step, the process looks like this:

```
root@dev:/usr/local/src/php4/ext > ./ext_skel my_module
Creating directory
Creating basic files: config.m4 Makefile.in .cvsignore [done].
To use your new extension, you will have to execute the following steps:
    $ cd ..
    $ ./buildconf
    $ ./configure # (your extension is automatically enabled)
    $ vi ext/my_module/my_module.c
    $ make
Repeat the last two steps as often as necessary.
```

This instruction creates the aforementioned files. To include the new module in the automatic configuration and build process, you have to run `buildconf`, which regenerates the `configure` script by searching through the `ext` directory and including all found `config.m4` files.

Finally, running `configure` parses all configuration options and generates a makefile based on those options and the options you specify in `Makefile.in`.

28-1 Példashows the previously generated `Makefile.in`:

Példa 28-1. The default `Makefile.in`.

```
# $Id: Extending_Zend_Build.xml,v 1.6 2002/03/25 08:13:46 hholzgra Exp $
LTLIBRARY_NAME        = libmy_module.la
LTLIBRARY_SOURCES     = my_module.c
LTLIBRARY_SHARED_NAME = my_module.la include
$(top_srcdir)/build/dynlib.mk
```

There's not much to tell about this one: It contains the names of the input and output files. You could also specify build instructions for other files if your module is built from multiple source files.

The default `config.m4` shown in 28-2 Példa'/> is a bit more complex:

Példa 28-2. The default `config.m4`.

```
dnl $Id: Extending_Zend_Build.xml,v 1.6 2002/03/25 08:13:46 hholzgra Exp $
dnl config.m4 for extension my_module
dnl don't forget to call PHP_EXTENSION(my_module)
dnl If your extension references something external, use with:
PHP_ARG_WITH(my_module, for my_module support,
dnl Make sure that the comment is aligned:
    [  --with-my_module             Include my_module support])
```

```
dnl Otherwise use enable:
PHP_ARG_ENABLE(my_module, whether to enable my_module support,
dnl Make sure that the comment is aligned:
    [  --enable-my_module           Enable my_module support])
if test "$PHP_MY_MODULE" != "no"; then
dnl Action..
PHP_EXTENSION(my_module, $ext_shared)
fi
```

If you're unfamiliar with M4 files (now is certainly a good time to get familiar), this might be a bit confusing at first; but it's actually quite easy.

Note: Everything prefixed with `dnl` is treated as a comment and is not parsed.

The `config.m4` file is responsible for parsing the command-line options passed to `configure` at configuration time. This means that it has to check for required external files and do similar configuration and setup tasks.

The default file creates two configuration directives in the `configure` script: `--with-my_module` and `--enable-my_module`. Use the first option when referring external files (such as the `--with-apache` directive that refers to the Apache directory). Use the second option when the user simply has to decide whether to enable your extension. Regardless of which option you use, you should uncomment the other, unnecessary one; that is, if you're using `--enable-my_module`, you should remove support for `--with-my_module`, and vice versa.

By default, the `config.m4` file created by `ext_skel` accepts both directives and automatically enables your extension. Enabling the extension is done by using the `PHP_EXTENSION` macro. To change the default behavior to include your module into the PHP binary when desired by the user (by explicitly specifying `--enable-my_module` or `--with-my_module`), change the test for `$PHP_MY_MODULE` to `== "yes"`:

```
if test "$PHP_MY_MODULE" == "yes"; then dnl
    Action.. PHP_EXTENSION(my_module, $ext_shared)
    fi
```

This would require you to use `--enable-my_module` each time when reconfiguring and recompiling PHP.

Note: Be sure to run `buildconf` every time you change `config.m4`!

We'll go into more details on the M4 macros available to your configuration scripts later in this chapter. For now, we'll simply use the default files. The sample sources on the CD-ROM all have working `config.m4` files. To include them into the PHP build process, simply copy the source directories to your PHP `ext` directory, run `buildconf`, and then include the sample modules you want by using the appropriate `--enable-*` directives with `configure`.

Fejezet 29. Creating Extensions

We'll start with the creation of a very simple extension at first, which basically does nothing more than implement a function that returns the integer it receives as parameter. 29-1 Példa shows the source.

Példa 29-1. A simple extension.

```
/* include standard header */
#include "php.h"

/* declaration of functions to be exported */
ZEND_FUNCTION(first_module);

/* compiled function list so Zend knows what's in this module */
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};

/* compiled module information */
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES
};

/* implement standard "stub" routine to introduce ourselves to Zend */
#if COMPILE_DL_FIRST_MODULE
ZEND_GET_MODULE(firstmod)
#endif

/* implement function that is meant to be made available to PHP */
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAIL-
URE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

This code contains a complete PHP module. We'll explain the source code in detail shortly, but first we'd like to discuss the build process. (This will allow the impatient to experiment before we dive into API discussions.)

Megjegyzés: The example source makes use of some features introduced with the Zend version used in PHP 4.1.0 and above, it won't compile with older PHP 4.0.x versions.

Compiling Modules

There are basically two ways to compile modules:

- Use the provided "make" mechanism in the `ext` directory, which also allows building of dynamic loadable modules.
- Compile the sources manually.

The first method should definitely be favored, since, as of PHP 4.0, this has been standardized into a sophisticated build process. The fact that it is so sophisticated is also its drawback, unfortunately - it's hard to understand at first. We'll provide a more detailed introduction to this later in the chapter, but first let's work with the default files.

The second method is good for those who (for some reason) don't have the full PHP source tree available, don't have access to all files, or just like to juggle with their keyboard. These cases should be extremely rare, but for the sake of completeness we'll also describe this method.

Compiling Using Make. To compile the sample sources using the standard mechanism, copy all their subdirectories to the `ext` directory of your PHP source tree. Then run `buildconf`, which will create an updated `configure` script containing appropriate options for the new extension. By default, all the sample sources are disabled, so you don't have to fear breaking your build process.

After you run `buildconf`, `configure --help` shows the following additional modules:

```
--enable-array_experiments   BOOK: Enables array experiments
--enable-call_userland       BOOK: Enables userland module
--enable-cross_conversion    BOOK: Enables cross-conversion module
--enable-first_module        BOOK: Enables first module
--enable-infoprint           BOOK: Enables infoprint module
--enable-reference_test      BOOK: Enables reference test module
--enable-resource_test       BOOK: Enables resource test module
--enable-variable_creation   BOOK: Enables variable-creation module
```

The module shown earlier in 29-1 Példa can be enabled with `--enable-first_module` or `--enable-first_module=yes`.

Compiling Manually. To compile your modules manually, you need the following commands:

Action	Command
Compiling	cc -fpic -DCOMPILE_DL=1 -I/usr/local/include -I. -I.. -I../Zend -c -o <your_object_file> <your_c_f
Linking	cc -shared -L/usr/local/lib -rdynamic -o <your_module_file> <your_object_file(s)>

The command to compile the module simply instructs the compiler to generate position-independent code (`-fpic` shouldn't be omitted) and additionally defines the constant `COMPILE_DL` to tell the module code that it's compiled as a dynamically loadable module (the test module above checks for this; we'll discuss it shortly). After these options, it specifies a number of standard include paths that should be used as the minimal set to compile the source files.

Note: All include paths in the example are relative to the directory `ext`. If you're compiling from another directory, change the pathnames accordingly. Required items are the PHP directory, the `Zend` directory, and (if necessary), the directory in which your module resides.

The link command is also a plain vanilla command instructing linkage as a dynamic module.

You can include optimization options in the compilation command, although these have been omitted in this example (but some are included in the makefile template described in an earlier section).

Note: Compiling and linking manually as a static module into the PHP binary involves very long instructions and thus is not discussed here. (It's not very efficient to type all those commands.)

Fejezet 30. Using Extensions

Depending on the build process you selected, you should either end up with a new PHP binary to be linked into your Web server (or run as CGI), or with an .so (shared object) file. If you compiled the example file `first_module.c` as a shared object, your result file should be `first_module.so`. To use it, you first have to copy it to a place from which it's accessible to PHP. For a simple test procedure, you can copy it to your `htdocs` directory and try it with the source in 30-1 Példa. If you compiled it into the PHP binary, omit the call to dl(), as the module's functionality is instantly available to your scripts.

> **Figyelem**
>
> For security reasons, you *should not* put your dynamic modules into publicly accessible directories. Even though it *can* be done and it simplifies testing, you should put them into a separate directory in production environments.

Példa 30-1. A test file for first_module.so.

```
<?php

// remove next comment if necessary
// dl("first_module.so");

$param = 2;
$return = first_module($param);

print("We sent '$param' and got '$return'");

?>
```

Calling this PHP file in your Web browser should give you the output shown in 30-1 Ábra.

Ábra 30-1. Output of first_module.php.

If required, the dynamic loadable module is loaded by calling the dl() function. This function looks for the specified shared object, loads it, and makes its functions available to PHP. The module exports the function **first_module()**, which accepts a single parameter, converts it to an integer, and returns the result of the conversion.

If you've gotten this far, congratulations! You just built your first extension to PHP.

Fejezet 31. Troubleshooting

Actually, not much troubleshooting can be done when compiling static or dynamic modules. The only problem that could arise is that the compiler will complain about missing definitions or something similar. In this case, make sure that all header files are available and that you specified their path correctly in the compilation command. To be sure that everything is located correctly, extract a clean PHP source tree and use the automatic build in the `ext` directory with the fresh files; this will guarantee a safe compilation environment. If this fails, try manual compilation.

PHP might also complain about missing functions in your module. (This shouldn’t happen with the sample sources if you didn’t modify them.) If the names of external functions you’re trying to access from your module are misspelled, they’ll remain as "unlinked symbols" in the symbol table. During dynamic loading and linkage by PHP, they won’t resolve because of the typing errors - there are no corresponding symbols in the main binary. Look for incorrect declarations in your module file or incorrectly written external references. Note that this problem is specific to dynamic loadable modules; it doesn’t occur with static modules. Errors in static modules show up at compile time.

Fejezet 32. Source Discussion

Now that you've got a safe build environment and you're able to include the modules into PHP files, it's time to discuss how everything works.

Module Structure

All PHP modules follow a common structure:

- Header file inclusions (to include all required macros, API definitions, etc.)
- C declaration of exported functions (required to declare the Zend function block)
- Declaration of the Zend function block
- Declaration of the Zend module block
- Implementation of **get_module()**
- Implementation of all exported functions

Header File Inclusions

The only header file you really have to include for your modules is `php.h`, located in the PHP directory. This file makes all macros and API definitions required to build new modules available to your code.

Tip: It's good practice to create a separate header file for your module that contains module-specific definitions. This header file should contain all the forward definitions for exported functions and include `php.h`. If you created your module using `ext_skel` you already have such a header file prepared.

Declaring Exported Functions

To declare functions that are to be exported (i.e., made available to PHP as new native functions), Zend provides a set of macros. A sample declaration looks like this:

```
ZEND_FUNCTION ( my_function );
```

`ZEND_FUNCTION` declares a new C function that complies with Zend's internal API. This means that the function is of type `void` and accepts `INTERNAL_FUNCTION_PARAMETERS` (another macro) as parameters. Additionally, it prefixes the function name with `zif`. The immediately expanded version of the above definitions would look like this:

```
void zif_my_function ( INTERNAL_FUNCTION_PARAMETERS );
```

Expanding `INTERNAL_FUNCTION_PARAMETERS` results in the following:

```
void zif_my_function( int ht
```

```
, zval * return_value
, zval * this_ptr
, int return_value_used
, zend_executor_globals * executor_globals
);
```

Since the interpreter and executor core have been separated from the main PHP package, a second API defining macros and function sets has evolved: the Zend API. As the Zend API now handles quite a few of the responsibilities that previously belonged to PHP, a lot of PHP functions have been reduced to macros aliasing to calls into the Zend API. The recommended practice is to use the Zend API wherever possible, as the old API is only preserved for compatibility reasons. For example, the types zval and pval are identical. zval is Zend's definition; pval is PHP's definition (actually, pval is an alias for zval now). As the macro `INTERNAL_FUNCTION_PARAMETERS` is a Zend macro, the above declaration contains zval. When writing code, you should always use zval to conform to the new Zend API.

The parameter list of this declaration is very important; you should keep these parameters in mind (see 32-1 Táblázat for descriptions).

Táblázat 32-1. Zend's Parameters to Functions Called from PHP

Parameter	Description
ht	The number of arguments passed to the Zend function. You should not touch this directly, but instea
return_value	This variable is used to pass any return values of your function back to PHP. Access to this variable
this_ptr	Using this variable, you can gain access to the object in which your function is contained, if it's use
return_value_used	This flag indicates whether an eventual return value from this function will actually be used by the c
executor_globals	This variable points to global settings of the Zend engine. You'll find this useful when creating new

Declaration of the Zend Function Block

Now that you have declared the functions to be exported, you also have to introduce them to Zend. Introducing the list of functions is done by using an array of zend_function_entry. This array consecutively contains all functions that are to be made available externally, with the function's name as it should appear in PHP and its name as defined in the C source. Internally, zend_function_entry is defined as shown in 32-1 Példa.

Példa 32-1. Internal declaration of zend_function_entry.

```
typedef struct _zend_function_entry {
    char *fname;
    void (*handler)(INTERNAL_FUNCTION_PARAMETERS);
    unsigned char *func_arg_types;
} zend_function_entry;
```

Entry	Description

fname	Denotes the function name as seen in PHP (for example, `fopen`, `mysql_connect`, or, in our example,
handler	Pointer to the C function responsible for handling calls to this function. For example, see the standard
func_arg_types	Allows you to mark certain parameters so that they're forced to be passed by reference. You usually sh

In the example above, the declaration looks like this:

```
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};
```

You can see that the last entry in the list always has to be `{NULL, NULL, NULL}`. This marker has to be set for Zend to know when the end of the list of exported functions is reached.

> **Megjegyzés:** You *cannot* use the predefined macros for the end marker, as these would try to refer to a function named "NULL"!

The macro `ZEND_FE` (short for 'Zend Function Entry') simply expands to a structure entry in zend_function_entry. Note that these macros introduce a special naming scheme to your functions - your C functions will be prefixed with `zif_`, meaning that `ZEND_FE(first_module)` will refer to a C function **zif_first_module()**. If you want to mix macro usage with hand-coded entries (not a good practice), keep this in mind.

Tip: Compilation errors that refer to functions named **zif_*()** relate to functions defined with `ZEND_FE`.

32-2 Táblázat shows a list of all the macros that you can use to define functions.

Táblázat 32-2. Macros for Defining Functions

Macro Name	Description
`ZEND_FE(name, arg_types)`	Defines a function entry of the name name in zend_functio
`ZEND_NAMED_FE(php_name, name, arg_types)`	Defines a function that will be available to PHP by the nam
`ZEND_FALIAS(name, alias, arg_types)`	Defines an alias named alias for name. arg_types needs to
`PHP_FE(name, arg_types)`	Old PHP API equivalent of `ZEND_FE`.
`PHP_NAMED_FE(runtime_name, name, arg_types)`	Old PHP API equivalent of `ZEND_NAMED_FE`.

Note: You can't use `ZEND_FE` in conjunction with `PHP_FUNCTION`, or `PHP_FE` in conjunction with `ZEND_FUNCTION`. However, it's perfectly legal to mix `ZEND_FE` and `ZEND_FUNCTION` with `PHP_FE` and `PHP_FUNCTION` when staying with the same macro set for each function to be declared. But mixing is *not* recommended; instead, you're advised to use the `ZEND_*` macros only.

Declaration of the Zend Module Block

This block is stored in the structure zend_module_entry and contains all necessary information to

describe the contents of this module to Zend. You can see the internal definition of this module in 32-2 Példa.

Példa 32-2. Internal declaration of zend_module_entry.

```
typedef struct _zend_module_entry zend_module_entry;

    struct _zend_module_entry {
    unsigned short size;
    unsigned int zend_api;
    unsigned char zend_debug;
    unsigned char zts;
    char *name;
    zend_function_entry *functions;
    int (*module_startup_func)(INIT_FUNC_ARGS);
    int (*module_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    int (*request_startup_func)(INIT_FUNC_ARGS);
    int (*request_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    void (*info_func)(ZEND_MODULE_INFO_FUNC_ARGS);
    char *version;
    int (*global_startup_func)(void);
    int (*global_shutdown_func)(void);

[ Rest of the structure is not interesting here ]

};
```

Entry	Description
size, zend_api, zend_debug and zts	Usually filled with the "STANDARD_MODULE_HEADER", which fills these four mer
name	Contains the module name (for example, "File functions", "Socket funct
functions	Points to the Zend function block, discussed in the preceding section.
module_startup_func	This function is called once upon module initialization and can be used to do one-
module_shutdown_func	This function is called once upon module shutdown and can be used to do one-tim
request_startup_func	This function is called once upon every page request and can be used to do one-ti
request_shutdown_func	This function is called once after every page request and works as counterpart to r
info_func	When phpinfo() is called in a script, Zend cycles through all loaded modules and
version	The version of the module. You can use NO_VERSION_YET if you don't want to g
Remaining structure elements	These are used internally and can be prefilled by using the macro STANDARD_MOD

In our example, this structure is implemented as follows:

```
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL, NULL, NULL, NULL, NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES,
};
```

This is basically the easiest and most minimal set of values you could ever use. The module name is set to `First Module`, then the function list is referenced, after which all startup and shutdown functions are marked as being unused.

For reference purposes, you can find a list of the macros involved in declared startup and shutdown functions in 32-3 Táblázat. These are not used in our basic example yet, but will be demonstrated later on. You should make use of these macros to declare your startup and shutdown functions, as these require special arguments to be passed (`INIT_FUNC_ARGS` and `SHUTDOWN_FUNC_ARGS`), which are automatically included into the function declaration when using the predefined macros. If you declare your functions manually and the PHP developers decide that a change in the argument list is necessary, you'll have to change your module sources to remain compatible.

Táblázat 32-3. Macros to Declare Startup and Shutdown Functions

Macro	Description
`ZEND_MINIT(module)`	Declares a function for module startup. The generated name will be `zend_minit_<mod`
`ZEND_MSHUTDOWN(module)`	Declares a function for module shutdown. The generated name will be `zend_mshutdow`
`ZEND_RINIT(module)`	Declares a function for request startup. The generated name will be `zend_rinit_<mod`
`ZEND_RSHUTDOWN(module)`	Declares a function for request shutdown. The generated name will be `zend_rshutdow`
`ZEND_MINFO(module)`	Declares a function for printing module information, used when phpinfo() is called. The

Creation of get_module()

This function is special to all dynamic loadable modules. Take a look at the creation via the `ZEND_GET_MODULE` macro first:

```
#if COMPILE_DL_FIRSTMOD
     ZEND_GET_MODULE(firstmod)
#endif
```

The function implementation is surrounded by a conditional compilation statement. This is needed since the function **get_module()** is only required if your module is built as a dynamic extension. By specifying a definition of `COMPILE_DL_FIRSTMOD` in the compiler command (see above for a discussion of the compilation instructions required to build a dynamic extension), you can instruct your module whether you intend to build it as a dynamic extension or as a built-in module. If you want a built-in module, the implementation of **get_module()** is simply left out.

get_module() is called by Zend at load time of the module. You can think of it as being invoked by the dl() call in your script. Its purpose is to pass the module information block back to Zend in order to inform the engine about the module contents.

If you don't implement a **get_module()** function in your dynamic loadable module, Zend will compliment you with an error message when trying to access it.

Implementation of All Exported Functions

Implementing the exported functions is the final step. The example function in `first_module`

looks like this:

```
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAIL-
URE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

The function declaration is done using `ZEND_FUNCTION`, which corresponds to `ZEND_FE` in the function entry table (discussed earlier).

After the declaration, code for checking and retrieving the function's arguments, argument conversion, and return value generation follows (more on this later).

Summary

That's it, basically - there's nothing more to implementing PHP modules. Built-in modules are structured similarly to dynamic modules, so, equipped with the information presented in the previous sections, you'll be able to fight the odds when encountering PHP module source files.

Now, in the following sections, read on about how to make use of PHP's internals to build powerful extensions.

Fejezet 33. Accepting Arguments

One of the most important issues for language extensions is accepting and dealing with data passed via arguments. Most extensions are built to deal with specific input data (or require parameters to perform their specific actions), and function arguments are the only real way to exchange data between the PHP level and the C level. Of course, there's also the possibility of exchanging data using predefined global values (which is also discussed later), but this should be avoided by all means, as it's extremely bad practice.

PHP doesn't make use of any formal function declarations; this is why call syntax is always completely dynamic and never checked for errors. Checking for correct call syntax is left to the user code. For example, it's possible to call a function using only one argument at one time and four arguments the next time - both invocations are syntactically absolutely correct.

Determining the Number of Arguments

Since PHP doesn't have formal function definitions with support for call syntax checking, and since PHP features variable arguments, sometimes you need to find out with how many arguments your function has been called. You can use the `ZEND_NUM_ARGS` macro in this case. In previous versions of PHP, this macro retrieved the number of arguments with which the function has been called based on the function's hash table entry, ht, which is passed in the `INTERNAL_FUNCTION_PARAMETERS` list. As ht itself now contains the number of arguments that have been passed to the function, `ZEND_NUM_ARGS` has been stripped down to a dummy macro (see its definition in `zend_API.h`). But it's still good practice to use it, to remain compatible with future changes in the call interface. *Note:* The old PHP equivalent of this macro is `ARG_COUNT`.

The following code checks for the correct number of arguments:

```
if(ZEND_NUM_ARGS() != 2) WRONG_PARAM_COUNT;
```

If the function is not called with two arguments, it exits with an error message. The code snippet above makes use of the tool macro `WRONG_PARAM_COUNT`, which can be used to generate a standard error message (see 33-1 Ábra).

Ábra 33-1. `WRONG_PARAM_COUNT` in action.

This macro prints a default error message and then returns to the caller. Its definition can also be found in `zend_API.h` and looks like this:

```
ZEND_API void wrong_param_count(void);

#define WRONG_PARAM_COUNT { wrong_param_count(); return; }
```

As you can see, it calls an internal function named **wrong_param_count()** that's responsible for printing the warning. For details on generating customized error messages, see the later section "Printing Information."

Retrieving Arguments

> **New parameter parsing API:** This chapter documents the new Zend parameter parsing API introduced by Andrei Zmievski. It was introduced in the development stage between PHP 4.0.6 and 4.1.0 .

Parsing parameters is a very common operation and it may get a bit tedious. It would also be nice to have standardized error checking and error messages. Since PHP 4.1.0, there is a way to do just that by using the new parameter parsing API. It greatly simplifies the process of receiving parameteres, but it has a drawback in that it can't be used for functions that expect variable number of parameters. But since the vast majority of functions do not fall into those categories, this parsing API is recommended as the new standard way.

The prototype for parameter parsing function looks like this:

```
int zend_parse_parameters(int num_args TSRMLS_DC, char *type_spec, ...);
```

The first argument to this function is supposed to be the number of actual parameters passed to your function, so `ZEND_NUM_ARGS()` can be used for that. The second parameter should always be `TSRMLS_CC` macro. The third argument is a string that specifies the number and types of arguments your function is expecting, similar to how printf format string specifies the number and format of the output values it should operate on. And finally the rest of the arguments are pointers to variables which should receive the values from the parameters.

zend_parse_parameters() also performs type conversions whenever possible, so that you always receive the data in the format you asked for. Any type of scalar can be converted to another one, but conversions between complex types (arrays, objects, and resources) and scalar types are not allowed.

If the parameters could be obtained successfully and there were no errors during type conversion, the function will return `SUCCESS`, otherwise it will return `FAILURE`. The function will output informative error messages, if the number of received parameters does not match the requested number, or if type conversion could not be performed.

Here are some sample error messages:

```
Warning - ini_get_all() requires at most 1 parameter, 2 given
Warning - wddx_deserialize() expects parameter 1 to be string, array given
```

Of course each error message is accompanied by the filename and line number on which it occurs.

Here is the full list of type specifiers:

- `l` - long
- `d` - double
- `s` - string (with possible null bytes) and its length
- `b` - boolean

- `r` - resource, stored in `zval*`
- `a` - array, stored in `zval*`
- `o` - object (of any class), stored in `zval*`
- `O` - object (of class specified by class entry), stored in `zval*`
- `z` - the actual `zval*`

The following characters also have a meaning in the specifier string:

- `|` - indicates that the remaining parameters are optional. The storage variables corresponding to these parameters should be initialized to default values by the extension, since they will not be touched by the parsing function if the parameters are not passed.
- `/` - the parsing function will call **SEPARATE_ZVAL_IF_NOT_REF()** on the parameter it follows, to provide a copy of the parameter, unless it's a reference.
- `!` - the parameter it follows can be of specified type or `NULL` (only applies to a, o, O, r, and z). If `NULL` value is passed by the user, the storage pointer will be set to `NULL`.

The best way to illustrate the usage of this function is through examples:

```
/* Gets a long, a string and its length, and a zval. */
long l;
char *s;
int s_len;
zval *param;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "lsz", &l, &s, &s_len, &param) == FAILURE) {
    return;
}

/* Gets an object of class specified by my_ce, and an optional double. */
zval *obj;
double d = 0.5;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "O|d", &obj, my_ce, &d) == FAILURE) {
    return;
}

/* Gets an object or null, and an array.
   If null is passed for object, obj will be set to NULL. */
zval *obj;
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "O!a", &obj, &arr) == FAIL-
URE) {
    return;
}

/* Gets a separated array. */
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "a/", &arr) == FAILURE) {
    return;
}

/* Get only the first three parameters (useful for varargs functions). */
```

```
zval *z;
zend_bool b;
zval *r;
if (zend_parse_parameters(3, "zbr!", &z, &b, &r) == FAILURE) {
    return;
}
```

Note that in the last example we pass 3 for the number of received parameters, instead of **ZEND_NUM_ARGS()**. What this lets us do is receive the least number of parameters if our function expects a variable number of them. Of course, if you want to operate on the rest of the parameters, you will have to use **zend_get_parameters_array_ex()** to obtain them.

The parsing function has an extended version that allows for an additional flags argument that controls its actions.

```
int zend_parse_parameters_ex(int flags, int num_args TSRMLS_DC, char *type_spec, ...);
```

The only flag you can pass currently is `ZEND_PARSE_PARAMS_QUIET`, which instructs the function to not output any error messages during its operation. This is useful for functions that expect several sets of completely different arguments, but you will have to output your own error messages.

For example, here is how you would get either a set of three longs or a string:

```
long l1, l2, l3;
char *s;
if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                             ZEND_NUM_ARGS() TSRMLS_CC,
                             "lll", &l1, &l2, &l3) == SUCCESS) {
    /* manipulate longs */
} else if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                                    ZEND_NUM_ARGS(), "s", &s, &s_len) == SUC-
CESS) {
    /* manipulate string */
} else {
    php_error(E_WARNING, "%s() takes either three long values or a string as argument",
              get_active_function_name(TSRMLS_C));
    return;
}
```

With all the abovementioned ways of receiving function parameters you should have a good handle on this process. For even more example, look through the source code for extensions that are shipped with PHP - they illustrate every conceivable situation.

Old way of retrieving arguments (deprecated)

> **Deprecated parameter parsing API:** This API is deprecated and superseded by the new ZEND parameter parsing API.

After having checked the number of arguments, you need to get access to the arguments themselves. This is done with the help of **zend_get_parameters_ex()**:

```
zval **parameter;

if(zend_get_parameters_ex(1, &parameter) != SUCCESS)
  WRONG_PARAM_COUNT;
```

All arguments are stored in a zval container, which needs to be pointed to *twice*. The snippet above tries to retrieve one argument and make it available to us via the parameter pointer.

zend_get_parameters_ex() accepts at least two arguments. The first argument is the number of arguments to retrieve (which should match the number of arguments with which the function has been called; this is why it's important to check for correct call syntax). The second argument (and all following arguments) are pointers to pointers to pointers to zvals. (Confusing, isn't it?) All these pointers are required because Zend works internally with **zval; to adjust a local **zval in our function,**zend_get_parameters_ex()** requires a pointer to it.

The return value of **zend_get_parameters_ex()** can either be `SUCCESS` or `FAILURE`, indicating (unsurprisingly) success or failure of the argument processing. A failure is most likely related to an incorrect number of arguments being specified, in which case you should exit with `WRONG_PARAM_COUNT`.

To retrieve more than one argument, you can use a similar snippet:

```
zval **param1, **param2, **param3, **param4;

if(zend_get_parameters_ex(4, &param1, &param2, &param3, &param4) != SUCCESS)
    WRONG_PARAM_COUNT;
```

zend_get_parameters_ex() only checks whether you're trying to retrieve too many parameters. If the function is called with five arguments, but you're only retrieving three of them with **zend_get_parameters_ex()**, you won't get an error but will get the first three parameters instead. Subsequent calls of **zend_get_parameters_ex()** won't retrieve the remaining arguments, but will get the same arguments again.

Dealing with a Variable Number of Arguments/Optional Parameters

If your function is meant to accept a variable number of arguments, the snippets just described are sometimes suboptimal solutions. You have to create a line calling **zend_get_parameters_ex()** for every possible number of arguments, which is often unsatisfying.

For this case, you can use the function **zend_get_parameters_array_ex()**, which accepts the number of parameters to retrieve and an array in which to store them:

```
zval **parameter_array[4];

/* get the number of arguments */
argument_count = ZEND_NUM_ARGS();

/* see if it satisfies our minimal request (2 arguments) */
/* and our maximal acceptance (4 arguments) */
if(argument_count < 2 || argument_count > 5)
    WRONG_PARAM_COUNT;

/* argument count is correct, now retrieve arguments */
if(zend_get_parameters_array_ex(argument_count, parameter_array) != SUCCESS)
    WRONG_PARAM_COUNT;
```

First, the number of arguments is checked to make sure that it's in the accepted range. After that, **zend_get_parameters_array_ex()** is used to fill parameter_array with valid pointers to the argument values.

A very clever implementation of this can be found in the code handling PHP's fsockopen() located in `ext/standard/fsock.c`, as shown in 33-1 Példa. Don't worry if you don't know all the functions used in this source yet; we'll get to them shortly.

Példa 33-1. PHP's implementation of variable arguments in fsockopen().

```
pval **args[5];
int *sock=emalloc(sizeof(int));
int *sockp;
int arg_count=ARG_COUNT(ht);
int socketd = -1;
unsigned char udp = 0;
struct timeval timeout = { 60, 0 };
unsigned short portno;
unsigned long conv;
char *key = NULL;
FLS_FETCH();

if (arg_count > 5 || arg_count < 2 || zend_get_parameters_array_ex(arg_count,args)==FAILU
    CLOSE_SOCK(1);
    WRONG_PARAM_COUNT;
}

switch(arg_count) {
```

```
    case 5:
        convert_to_double_ex(args[4]);
        conv = (unsigned long) (Z_DVAL_P(args[4]) * 1000000.0);
        timeout.tv_sec = conv / 1000000;
        timeout.tv_usec = conv % 1000000;
        /* fall-through */
    case 4:
        if (!PZVAL_IS_REF(*args[3])) {
            php_error(E_WARNING,"error string argument to fsockopen not passed by referen
        }
        pval_copy_constructor(*args[3]);
        ZVAL_EMPTY_STRING(*args[3]);
        /* fall-through */
    case 3:
        if (!PZVAL_IS_REF(*args[2])) {
            php_error(E_WARNING,"error argument to fsockopen not passed by reference");
            return;
        }
        ZVAL_LONG(*args[2], 0);
        break;
}

convert_to_string_ex(args[0]);
convert_to_long_ex(args[1]);
portno = (unsigned short) Z_LVAL_P(args[1]);

key = emalloc(Z_STRLEN_P(args[0]) + 10);
```

fsockopen() accepts two, three, four, or five parameters. After the obligatory variable declarations, the function checks for the correct range of arguments. Then it uses a fall-through mechanism in a `switch()` statement to deal with all arguments. The `switch()` statement starts with the maximum number of arguments being passed (five). After that, it automatically processes the case of four arguments being passed, then three, by omitting the otherwise obligatory `break` keyword in all stages. After having processed the last case, it exits the `switch()` statement and does the minimal argument processing needed if the function is invoked with only two arguments.

This multiple-stage type of processing, similar to a stairway, allows convenient processing of a variable number of arguments.

Accessing Arguments

To access arguments, it’s necessary for each argument to have a clearly defined type. Again, PHP’s extremely dynamic nature introduces some quirks. Because PHP never does any kind of type checking, it’s possible for a caller to pass any kind of data to your functions, whether you want it or not. If you expect an integer, for example, the caller might pass an array, and vice versa - PHP simply won’t notice.

To work around this, you have to use a set of API functions to force a type conversion on every argument that’s being passed (see 33-1 Táblázat).

Note: All conversion functions expect a **zval as parameter.

Táblázat 33-1. Argument Conversion Functions

Function	Description
convert_to_boolean_ex()	Forces conversion to a Boolean type. Boolean values remain untouched. Longs, d
convert_to_long_ex()	Forces conversion to a long, the default integer type. NULL values, Booleans, res
convert_to_double_ex()	Forces conversion to a double, the default floating-point type. NULL values, Bool
convert_to_string_ex()	Forces conversion to a string. Strings remain untouched. NULL values are conver
`convert_to_array_ex(value)`	Forces conversion to an array. Arrays remain untouched. Objects are converted to
`convert_to_object_ex(value)`	Forces conversion to an object. Objects remain untouched. NULL values are conv
`convert_to_null_ex(value)`	Forces the type to become a NULL value, meaning empty.

> **Megjegyzés:** You can find a demonstration of the behavior in `cross_conversion.php` on the accompanying CD-ROM. 33-2 Ábra shows the output.

Ábra 33-2. Cross-conversion behavior of PHP.

Using these functions on your arguments will ensure type safety for all data that's passed to you. If the supplied type doesn't match the required type, PHP forces dummy contents on the resulting value (empty strings, arrays, or objects, `0` for numeric values, `FALSE` for Booleans) to ensure a defined state.

Following is a quote from the sample module discussed previously, which makes use of the conversion functions:

```
zval **parameter;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &parameter) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

convert_to_long_ex(parameter);

RETURN_LONG(Z_LVAL_P(parameter));
```

After retrieving the parameter pointer, the parameter value is converted to a long (an integer), which also forms the return value of this function. Understanding access to the contents of the value requires a short discussion of the zval type, whose definition is shown in 33-2 Példa.

Példa 33-2. PHP/Zend zval type definition.

```
typedef pval zval;

typedef struct _zval_struct zval;

typedef union _zvalue_value {
long lval; /* long value */
double dval; /* double value */
struct {
char *val;
int len;
} str;
HashTable *ht; /* hash table value */
struct {
zend_class_entry *ce;
HashTable *properties;
} obj;
} zvalue_value;

struct _zval_struct {
/* Variable information */
zvalue_value value; /* value */
unsigned char type; /* active type */
unsigned char is_ref;
short refcount;
};
```

Actually, pval (defined in `php.h`) is only an alias of zval (defined in `zend.h`), which in turn refers to _zval_struct. This is a most interesting structure. _zval_struct is the "master" structure, containing the value structure, type, and reference information. The substructure zvalue_value is a union that contains the variable's contents. Depending on the variable's type, you'll have to access different members of this union. For a description of both structures, see 33-2 Táblázat, 33-3 Táblázat and 33-4 Táblázat.

Táblázat 33-2. Zend zval Structure

Entry	Description
value	Union containing this variable's contents. See 33-3 Táblázat for a description.
type	Contains this variable's type. For a list of available types, see 33-4 Táblázat.
is_ref	0 means that this variable is not a reference; 1 means that this variable is a reference to another variable.
refcount	The number of references that exist for this variable. For every new reference to the value stored in this variab

Táblázat 33-3. Zend zvalue_value Structure

Entry	Description
lval	Use this property if the variable is of the type `IS_LONG`, `IS_BOOLEAN`, or `IS_RESOURCE`.
dval	Use this property if the variable is of the type `IS_DOUBLE`.
str	This structure can be used to access variables of the type `IS_STRING`. The member len contains the string length

ht	This entry points to the variable's hash table entry if the variable is an array.
obj	Use this property if the variable is of the type `IS_OBJECT`.

Táblázat 33-4. Zend Variable Type Constants

Constant	Description
`IS_NULL`	Denotes a NULL (empty) value.
`IS_LONG`	A long (integer) value.
`IS_DOUBLE`	A double (floating point) value.
`IS_STRING`	A string.
`IS_ARRAY`	Denotes an array.
`IS_OBJECT`	An object.
`IS_BOOL`	A Boolean value.
`IS_RESOURCE`	A resource (for a discussion of resources, see the appropriate section below).
`IS_CONSTANT`	A constant (defined) value.

To access a long you access zval.value.lval, to access a double you use zval.value.dval, and so on. Because all values are stored in a union, trying to access data with incorrect union members results in meaningless output.

Accessing arrays and objects is a bit more complicated and is discussed later.

Dealing with Arguments Passed by Reference

If your function accepts arguments passed by reference that you intend to modify, you need to take some precautions.

What we didn't say yet is that under the circumstances presented so far, you don't have write access to any zval containers designating function parameters that have been passed to you. Of course, you can change any zval containers that you created within your function, but you mustn't change any zvals that refer to Zend-internal data!

We've only discussed the so-called ***_ex()** API so far. You may have noticed that the API functions we've used are called **zend_get_parameters_ex()** instead of **zend_get_parameters()**, **convert_to_long_ex()** instead of **convert_to_long()**, etc. The ***_ex()** functions form the so-called new "extended" Zend API. They give a minor speed increase over the old API, but as a tradeoff are only meant for providing read-only access.

Because Zend works internally with references, different variables may reference the same value. Write access to a zval container requires this container to contain an isolated value, meaning a value that's not referenced by any other containers. If a zval container were referenced by other containers and you changed the referenced zval, you would automatically change the contents of the other containers referencing this zval (because they'd simply point to the changed value and thus change their own value as well).

zend_get_parameters_ex() doesn't care about this situation, but simply returns a pointer to the desired zval containers, whether they consist of references or not. Its corresponding function in the traditional API, **zend_get_parameters()**, immediately checks for referenced values. If it finds a

reference, it creates a new, isolated zval container; copies the referenced data into this newly allocated space; and then returns a pointer to the new, isolated value.

This action is called *zval separation* (or pval separation). Because the ***_ex()** API doesn't perform zval separation, it's considerably faster, while at the same time disabling write access.

To change parameters, however, write access is required. Zend deals with this situation in a special way: Whenever a parameter to a function is passed by reference, it performs automatic zval separation. This means that whenever you're calling a function like this in PHP, Zend will automatically ensure that $parameter is being passed as an isolated value, rendering it to a write-safe state:

```
my_function(&$parameter);
```

But this *is not* the case with regular parameters! All other parameters that are not passed by reference are in a read-only state.

This requires you to make sure that you're really working with a reference - otherwise you might produce unwanted results. To check for a parameter being passed by reference, you can use the macro `PZVAL_IS_REF`. This macro accepts a `zval*` to check if it is a reference or not. Examples are given in in 33-3 Példa.

Példa 33-3. Testing for referenced parameter passing.

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
    return;

/* check for parameter being passed by reference */
if (!PZVAL_IS_REF(*parameter)) {
{
    zend_error(E_WARNING, "Parameter wasn't passed by reference");
    RETURN_NULL();
}

/* make changes to the parameter */
ZVAL_LONG(*parameter, 10);
```

Assuring Write Safety for Other Parameters

You might run into a situation in which you need write access to a parameter that's retrieved with **zend_get_parameters_ex()** but not passed by reference. For this case, you can use the macro SEPARATE_ZVAL, which does a zval separation on the provided container. The newly generated zval is detached from internal data and has only a local scope, meaning that it can be changed or destroyed without implying global changes in the script context:

```
zval **parameter;

/* retrieve parameter */
zend_get_parameters_ex(1, &parameter);

/* at this stage, <parameter> still is connected */
/* to Zend's internal data buffers */

/* make <parameter> write-safe */
SEPARATE_ZVAL(parameter);

/* now we can safely modify <parameter> */
/* without implying global changes */
```

SEPARATE_ZVAL uses **emalloc()** to allocate the new zval container, which means that even if you don't deallocate this memory yourself, it will be destroyed automatically upon script termination. However, doing a lot of calls to this macro without freeing the resulting containers will clutter up your RAM.

Note: As you can easily work around the lack of write access in the "traditional" API (with **zend_get_parameters()** and so on), this API seems to be obsolete, and is not discussed further in this chapter.

Fejezet 34. Creating Variables

When exchanging data from your own extensions with PHP scripts, one of the most important issues is the creation of variables. This section shows you how to deal with the variable types that PHP supports.

Overview

To create new variables that can be seen "from the outside" by the executing script, you need to allocate a new zval container, fill this container with meaningful values, and then introduce it to Zend's internal symbol table. This basic process is common to all variable creations:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol ta-
ble */
ZEND_SET_SYMBOL(EG(active_symbol_table), "new_variable_name", new_variable);

/* the variable is now accessible to the script by using $new_variable_name */
```

The macro `MAKE_STD_ZVAL` allocates a new zval container using `ALLOC_ZVAL` and initializes it using `INIT_ZVAL`. As implemented in Zend at the time of this writing, *initializing* means setting the reference count to `1` and clearing the is_ref flag, but this process could be extended later - this is why it's a good idea to keep using `MAKE_STD_ZVAL` instead of only using `ALLOC_ZVAL`. If you want to optimize for speed (and you don't have to explicitly initialize the zval container here), you can use `ALLOC_ZVAL`, but this isn't recommended because it doesn't ensure data integrity.

`ZEND_SET_SYMBOL` takes care of introducing the new variable to Zend's symbol table. This macro checks whether the value already exists in the symbol table and converts the new symbol to a reference if so (with automatic deallocation of the old zval container). This is the preferred method if speed is not a crucial issue and you'd like to keep memory usage low.

Note that `ZEND_SET_SYMBOL` makes use of the Zend executor globals via the macro `EG`. By specifying `EG(active_symbol_table)`, you get access to the currently active symbol table, dealing with the active, local scope. The local scope may differ depending on whether the function was invoked from within a function.

If you need to optimize for speed and don't care about optimal memory usage, you can omit the check for an existing variable with the same value and instead force insertion into the symbol table by using **zend_hash_update()**:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol ta-
ble */
zend_hash_update(
```

```
    EG(active_symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

This is actually the standard method used in most modules.

The variables generated with the snippet above will always be of local scope, so they reside in the context in which the function has been called. To create new variables in the global scope, use the same method but refer to another symbol table:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global sym-
bol table
ZEND_SET_SYMBOL(&EG(symbol_table), "new_variable_name", new_variable);
```

The macro `ZEND_SET_SYMBOL` is now being called with a reference to the main, global symbol table by referring `EG(symbol_table)`.

Note: The active_symbol_table variable is a pointer, but symbol_table is not. This is why you have to use `EG(active_symbol_table)` and `&EG(symbol_table)` as parameters to `ZEND_SET_SYMBOL` - it requires a pointer.

Similarly, to get a more efficient version, you can hardcode the symbol table update:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global sym-
bol table
zend_hash_update(
    &EG(symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

34-1 Példa shows a sample source that creates two variables - local_variable with a local scope and global_variable with a global scope (see Figure 9.7). The full example can be found on the CD-ROM.

Note: You can see that the global variable is actually not accessible from within the function. This is because it's not imported into the local scope using `global $global_variable;` in the PHP source.

Példa 34-1. Creating variables with different scopes.

```
ZEND_FUNCTION(variable_creation)
{
    zval *new_var1, *new_var2;

    MAKE_STD_ZVAL(new_var1);
    MAKE_STD_ZVAL(new_var2);

    ZVAL_LONG(new_var1, 10);
    ZVAL_LONG(new_var2, 5);

    ZEND_SET_SYMBOL(EG(active_symbol_table), "local_variable", new_var1);
    ZEND_SET_SYMBOL(&EG(symbol_table), "global_variable", new_var2);

    RETURN_NULL();

}
```

Longs (Integers)

Now let's get to the assignment of data to variables, starting with longs. Longs are PHP's integers and are very simple to store. Looking at the zval.value container structure discussed earlier in this chapter, you can see that the long data type is directly contained in the union, namely in the lval field. The corresponding type value for longs is `IS_LONG` (see 34-2 Példa).

Példa 34-2. Creation of a long.

```
zval *new_long;

MAKE_STD_ZVAL(new_long);

new_long->type = IS_LONG;
new_long->value.lval = 10;
```

Alternatively, you can use the macro `ZVAL_LONG`:

```
zval *new_long;

MAKE_STD_ZVAL(new_long);
ZVAL_LONG(new_long, 10);
```

Doubles (Floats)

Doubles are PHP's floats and are as easy to assign as longs, because their value is also contained directly in the union. The member in the zval.value container is dval; the corresponding type is `IS_DOUBLE`.

```
zval *new_double;

MAKE_STD_ZVAL(new_double);

new_double->type = IS_DOUBLE;
new_double->value.dval = 3.45;
```

Alternatively, you can use the macro `ZVAL_DOUBLE`:

```
zval *new_double;

MAKE_STD_ZVAL(new_double);
ZVAL_DOUBLE(new_double, 3.45);
```

Strings

Strings need slightly more effort. As mentioned earlier, all strings that will be associated with Zend's internal data structures need to be allocated using Zend's own memory-management functions. Referencing of static strings or strings allocated with standard routines is not allowed. To assign strings, you have to access the structure str in the zval.value container. The corresponding type is `IS_STRING`:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);

new_string->type = IS_STRING;
new_string->value.str.len = strlen(string_contents);
new_string->value.str.val = estrdup(string_contents);
```

Note the usage of Zend's **estrdup()** here. Of course, you can also use the predefined macro `ZVAL_STRING`:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);
ZVAL_STRING(new_string, string_contents, 1);
```

`ZVAL_STRING` accepts a third parameter that indicates whether the supplied string contents should be duplicated (using **estrdup()**). Setting this parameter to `1` causes the string to be duplicated; `0` simply uses the supplied pointer for the variable contents. This is most useful if you want to create a new variable referring to a string that's already allocated in Zend internal memory.

If you want to truncate the string at a certain position or you already know its length, you can use `ZVAL_STRINGL(zval, string, length, duplicate)`, which accepts an explicit string length to be set for the new string. This macro is faster than `ZVAL_STRING` and also binary-safe.

To create empty strings, set the string length to `0` and use `empty_string` as contents:

```
new_string->type = IS_STRING;
new_string->value.str.len = 0;
new_string->value.str.val = empty_string;
```

Of course, there's a macro for this as well (`ZVAL_EMPTY_STRING`):

```
MAKE_STD_ZVAL(new_string);
ZVAL_EMPTY_STRING(new_string);
```

Booleans

Booleans are created just like longs, but have the type `IS_BOOL`. Allowed values in lval are `0` and `1`:

```
zval *new_bool;

MAKE_STD_ZVAL(new_bool);

new_bool->type = IS_BOOL;
new_bool->value.lval = 1;
```

The corresponding macros for this type are `ZVAL_BOOL` (allowing specification of the value) as well as `ZVAL_TRUE` and `ZVAL_FALSE` (which explicitly set the value to `TRUE` and `FALSE`, respectively).

Arrays

Arrays are stored using Zend's internal hash tables, which can be accessed using the **zend_hash_*()** API. For every array that you want to create, you need a new hash table handle, which will be stored in the ht member of the zval.value container.

There's a whole API solely for the creation of arrays, which is extremely handy. To start a new array, you call **array_init()**.

```
zval *new_array;

MAKE_STD_ZVAL(new_array);

if(array_init(new_array) != SUCCESS)
{
    // do error handling here
}
```

If **array_init()** fails to create a new array, it returns `FAILURE`.

To add new elements to the array, you can use numerous functions, depending on what you want to do. 34-1 Táblázat, 34-2 Táblázat and 34-3 Táblázat describe these functions. All functions return `FAILURE` on failure and `SUCCESS` on success.

Táblázat 34-1. Zend's API for Associative Arrays

Function	Description
add_assoc_long(zval *array, char *key, long n);()	Adds an element of type `long`.
add_assoc_unset(zval *array, char *key);()	Adds an unset element.
add_assoc_bool(zval *array, char *key, int b);()	Adds a Boolean element.
add_assoc_resource(zval *array, char *key, int r);()	Adds a resource to the array.
add_assoc_double(zval *array, char *key, double d);()	Adds a floating-point value.
add_assoc_string(zval *array, char *key, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_assoc_stringl(zval *array, char *key, char *str, uint length, int duplicate); ()	Adds a string with the desired length length to the array. Otherwise, behaves like **add_assoc_string()**.

Táblázat 34-2. Zend's API for Indexed Arrays, Part 1

Function	Description
add_index_long(zval *array, uint idx, long n);()	Adds an element of type `long`.
add_index_unset(zval *array, uint idx);()	Adds an unset element.
add_index_bool(zval *array, uint idx, int b);()	Adds a Boolean element.
add_index_resource(zval *array, uint idx, int r);()	Adds a resource to the array.
add_index_double(zval *array, uint idx, double d);()	Adds a floating-point value.

add_index_string(zval *array, uint idx, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_index_stringl(zval *array, uint idx, char *str, uint length, int duplicate);()	Adds a string with the desired length length to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.

Táblázat 34-3. Zend's API for Indexed Arrays, Part 2

Function	Description
add_next_index_long(zval *array, long n);()	Adds an element of type `long`.
add_next_index_unset(zval *array);()	Adds an unset element.
add_next_index_bool(zval *array, int b);()	Adds a Boolean element.
add_next_index_resource(zval *array, int r);()	Adds a resource to the array.
add_next_index_double(zval *array, double d);()	Adds a floating-point value.
add_next_index_string(zval *array, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_next_index_stringl(zval *array, char *str, uint length, int duplicate);()	Adds a string with the desired length length to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.

All these functions provide a handy abstraction to Zend's internal hash API. Of course, you can also use the hash functions directly - for example, if you already have a zval container allocated that you want to insert into an array. This is done using **zend_hash_update()()** for associative arrays (see 34-3 Példa) and **zend_hash_index_update()** for indexed arrays (see 34-4 Példa):

Példa 34-3. Adding an element to an associative array.

```
zval *new_array, *new_element;
char *key = "element_key";

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

if(array_init(new_array) == FAILURE)
{
    // do error handling here
}

ZVAL_LONG(new_element, 10);

if(zend_hash_update(new_array->value.ht, key, strlen(key) + 1, (void *)&new_element, size
{
    // do error handling here
}
```

Példa 34-4. Adding an element to an indexed array.

```
zval *new_array, *new_element;
int key = 2;

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

if(array_init(new_array) == FAILURE)
{
    // do error handling here
}

ZVAL_LONG(new_element, 10);

if(zend_hash_index_update(new_array->value.ht, key, (void *)&new_element, sizeof(zval *),
{
    // do error handling here
}
```

To emulate the functionality of **add_next_index_*()**, you can use this:

```
zend_hash_next_index_insert(ht, zval **new_element, sizeof(zval *), NULL)
```

Note: To return arrays from a function, use **array_init()** and all following actions on the predefined variable return_value (given as argument to your exported function; see the earlier discussion of the call interface). You do not have to use `MAKE_STD_ZVAL` on this.

Tip: To avoid having to write `new_array->value.ht` every time, you can use `HASH_OF(new_array)`, which is also recommended for compatibility and style reasons.

Objects

Since objects can be converted to arrays (and vice versa), you might have already guessed that they have a lot of similarities to arrays in PHP. Objects are maintained with the same hash functions, but there's a different API for creating them.

To initialize an object, you use the function **object_init()**:

```
zval *new_object;

MAKE_STD_ZVAL(new_object);

if(object_init(new_object) != SUCCESS)
{
    // do error handling here
}
```

You can use the functions described in 34-4 Táblázat to add members to your object.

Táblázat 34-4. Zend's API for Object Creation

Function	Description
add_property_long(zval *object, char *key, long l);()	Adds a long to the object.
add_property_unset(zval *object, char *key);()	Adds an unset property to the
add_property_bool(zval *object, char *key, int b);()	Adds a Boolean to the object
add_property_resource(zval *object, char *key, long r);()	Adds a resource to the object
add_property_double(zval *object, char *key, double d);()	Adds a double to the object.
add_property_string(zval *object, char *key, char *str, int duplicate);()	Adds a string to the object.
add_property_stringl(zval *object, char *key, char *str, uint length, int duplicate);()	Adds a string of the specified
add_property_zval(zval *obect, char *key, zval *container):()	Adds a `zval` container to the

Resources

Resources are a special kind of data type in PHP. The term *resources* doesn't really refer to any special kind of data, but to an abstraction method for maintaining any kind of information. Resources are kept in a special resource list within Zend. Each entry in the list has a correspondending type definition that denotes the kind of resource to which it refers. Zend then internally manages all references to this resource. Access to a resource is never possible directly - only via a provided API. As soon as all references to a specific resource are lost, a corresponding shutdown function is called.

For example, resources are used to store database links and file descriptors. The *de facto* standard implementation can be found in the MySQL module, but other modules such as the Oracle module also make use of resources.

> **Megjegyzés:** In fact, a resource can be a pointer to anything you need to handle in your functions (e.g. pointer to a structure) and the user only has to pass a single resource variable to your function.

To create a new resource you need to register a resource destruction handler for it. Since you can store any kind of data as a resource, Zend needs to know how to free this resource if its not longer needed. This works by registering your own resource destruction handler to Zend which in turn gets called by Zend whenever your resource can be freed (whether manually or automatically). Registering your resource handler within Zend returns you the **resource type handle** for that resource. This handle is needed whenever you want to access a resource of this type later and is most of time stored in a global static variable within your extension. There is no need to worry about thread safety here because you only register your resource handler once during module initialization.

The Zend function to register your resource handler is defined as:

```
ZEND_API int zend_register_list_destructors_ex(rsrc_dtor_func_t ld, rsrc_dtor_func_t pld,
```

There are two different kinds of resource destruction handlers you can pass to this function: a handler for normal resources and a handler for persistent resources. Persistent resources are for example used for database connection. When registering a resource, either of these handlers must be given. For the other handler just pass `NULL`.

zend_register_list_destructors_ex() accepts the following parameters:

`ld`	Normal resource destruction handler callback
`pld`	Pesistent resource destruction handler callback
`type_name`	A string specifying the name of your resource. It's always a good thing to specify an unique name wi
`module_number`	The `module_number` is automatically available in your `PHP_MINIT_FUNCTION` function and therefo

The return value is an unique integer ID for your **resource type**.

The resource destruction handler (either normal or persistent resources) has the following prototype:

```
void resource_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC);
```

The passed `rsrc` is a pointer to the following structure:

```
typedef struct _zend_rsrc_list_entry {

    void *ptr;
    int type;
    int refcount;

} zend_rsrc_list_entry;
```

The member `void *ptr` is the actual pointer to your resource.

Now we know how to start things, we define our own resource we want register within Zend. It is only a simple structure with two integer members:

```
typedef struct {

    int resource_link;
    int resource_type;

} my_resource;
```

Our resource destruction handler is probably going to look something like this:

```
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    // You most likely cast the void pointer to your structure type

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;

    // Now do whatever needs to be done with you resource. Closing
    // Files, Sockets, freeing additional memory, etc.
    // Also, don't forget to actually free the memory for your resource too!

    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}
```

> **Megjegyzés:** One important thing to mention: If your resource is a rather complex structure which also contains pointers to memory you allocated during runtime you have to free them **before** freeing the resource itself!

Now that we have defined

1. what our resource is and
2. our resource destruction handler

we can go on and do the rest of the steps:

1. create a global variable within the extension holding the resource ID so it can be accessed from every function which needs it
2. define the resource name
3. write the resource destruction handler
4. and finally register the handler

```
// Somewhere in your extension, define the variable for your registered resources.
// If you wondered what 'le' stands for: it simply means 'list entry'.
static int le_myresource;

// It's nice to define your resource name somewhere
#define le_myresource_name  "My type of resource"

[...]

// Now actually define our resource destruction handler
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;
    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}

[...]

PHP_MINIT_FUNCTION(my_extension) {

    // Note that 'module_number' is already provided through the
    // PHP_MINIT_FUNCTION() function definition.

    le_myresource = zend_register_resource_destructors_ex(my_destruction_handler, NUL

    // You can register additional resources, initialize
    // your global vars, constants, whatever.
}
```

To actually register a new resource you use can either use the **zend_register_resource()** function or the **ZEND_REGISTER_RESOURE()** macro, both defined in zend_list.h . Although the arguments for both map 1:1 it's a good idea to always use macros to be upwards compatible:

```
int ZEND_REGISTER_RESOURCE(zval *rsrc_result, void *rsrc_pointer, int rsrc_type);
```

`rsrc_result`	This is an already initialized `zval *` container.
`rsrc_pointer`	Your resource pointer you want to store.
`rsrc_type`	The type which you received when you registered the resource destruction handler. If you followed the

The return value is an unique integer identifier for that resource.

What is really going on when you register a new resource is it gets inserted in an internal list in Zend and the result is just stored in the given `zval *` container:

```
rsrc_id = zend_list_insert(rsrc_pointer, rsrc_type);

if (rsrc_result) {
    rsrc_result->value.lval = rsrc_id;
    rsrc_result->type = IS_RESOURCE;
}

return rsrc_id;
```

The returned `rsrc_id` uniquly identifies the newly registered resource. You can use the macro `RETURN_RESOURE` to return it to the user:

```
RETURN_RESOURCE(rsrc_id)
```

Megjegyzés: It is common practice that if you want to return the resource immidiately to the user you specify the `return_value` as the `zval *` container.

Zend now keeps track of all references to this resource. As soon as all references to the resource are lost, the destructor that you previously registered for this resource is called. The nice thing about this setup is that you don't have to worry about memory leakages introduced by allocations in your module - just register all memory allocations that your calling script will refer to as resources. As soon as the script decides it doesn't need them anymore, Zend will find out and tell you.

Now that the user got his resource, at some point he is passing it back to one of your functions. The value.lval inside the `zval *` container contains the key to your resource and thus can be used to fetch the resource with the following macro: `ZEND_FETCH_RESOURCE`:

```
ZEND_FETCH_RESOURCE(rsrc, rsrc_type, rsrc_id, default_rsrc_id, resource_type_name, resour
```

`rsrc`	This is your pointer which will point to your previously registered resource.
`rsrc_type`	This is the typecast argument for your pointer, e.g. `myresource *`.
`rsrc_id`	This is the address of the `zval *`container the user passed to your function, e.g. `&z_resourc`
`default_rsrc_id`	This integer specifies the default resource `ID` if no resource could be fetched or -1.
`resource_type_name`	This is the name of the requested resource. It's a string and is used when the resource can't be
`resource_type`	The `resource_type` you got back when registering the resource destruction handler. In our e

This macro has no return value. It is for the developers convenience and takes care of TSRMLS arguments passing and also does check if the resource could be fetched. It throws a warning message and returns the current PHP function with `NULL` if there was a problem retrieving the resource.

To force removal of a resource from the list, use the function **zend_list_delete()**. You can also force the reference count to increase if you know that you're creating another reference for a previously allocated value (for example, if you're automatically reusing a default database link). For this case, use the function **zend_list_addref()**. To search for previously allocated resource entries, use **zend_list_find()**. The complete API can be found in `zend_list.h`.

Macros for Automatic Global Variable Creation

In addition to the macros discussed earlier, a few macros allow easy creation of simple global variables. These are nice to know in case you want to introduce global flags, for example. This is somewhat bad practice, but Table 34-5 Táblázat describes macros that do exactly this task. They don't need any zval allocation; you simply have to supply a variable name and value.

Táblázat 34-5. Macros for Global Variable Creation

Macro	Description
`SET_VAR_STRING(name, value)`	Creates a new string.
`SET_VAR_STRINGL(name, value, length)`	Creates a new string of the specified length. This macro is faster than `SET_VAR_STRING` and also binary-safe.
`SET_VAR_LONG(name, value)`	Creates a new long.
`SET_VAR_DOUBLE(name, value)`	Creates a new double.

Creating Constants

Zend supports the creation of true constants (as opposed to regular variables). Constants are accessed without the typical dollar sign (`$`) prefix and are available in all scopes. Examples include `TRUE` and `FALSE`, to name just two.

To create your own constants, you can use the macros in 34-6 Táblázat. All the macros create a constant with the specified name and value.

You can also specify flags for each constant:

- `CONST_CS` - This constant's name is to be treated as case sensitive.
- `CONST_PERSISTENT` - This constant is persistent and won't be "forgotten" when the current process carrying this constant shuts down.

To use the flags, combine them using a inary OR:

```
// register a new constant of type "long"
    REGISTER_LONG_CONSTANT("NEW_MEANINGFUL_CONSTANT", 324, CONST_CS |
    CONST_PERSISTENT);
```

There are two types of macros - `REGISTER_*_CONSTANT` and`REGISTER_MAIN_*_CONSTANT`. The first type creates constants bound to the current module. These constants are dumped from the symbol table as soon as the module that registered the constant is unloaded from memory. The second type creates constants that remain in the symbol table independently of the module.

Táblázat 34-6. Macros for Creating Constants

Macro	
`REGISTER_LONG_CONSTANT(name, value, flags)`	`REGISTER_MAIN_LONG_CONSTANT(name, value, flags`
`REGISTER_DOUBLE_CONSTANT(name, value, flags)`	`REGISTER_MAIN_DOUBLE_CONSTANT(name, value, f`
`REGISTER_STRING_CONSTANT(name, value, flags)`	`REGISTER_MAIN_STRING_CONSTANT(name, value, f`
`REGISTER_STRINGL_CONSTANT(name, value, length, flags)`	`REGISTER_MAIN_STRINGL_CONSTANT(name`

Fejezet 35. Duplicating Variable Contents: The Copy Constructor

Sooner or later, you may need to assign the contents of one zval container to another. This is easier said than done, since the zval container doesn't contain only type information, but also references to places in Zend's internal data. For example, depending on their size, arrays and objects may be nested with lots of hash table entries. By assigning one zval to another, you avoid duplicating the hash table entries, using only a reference to them (at most).

To copy this complex kind of data, use the *copy constructor*. Copy constructors are typically defined in languages that support operator overloading, with the express purpose of copying complex types. If you define an object in such a language, you have the possibility of overloading the "=" operator, which is usually responsible for assigning the contents of the lvalue (result of the evaluation of the left side of the operator) to the rvalue (same for the right side).

Overloading means assigning a different meaning to this operator, and is usually used to assign a function call to an operator. Whenever this operator would be used on such an object in a program, this function would be called with the lvalue and rvalue as parameters. Equipped with that information, it can perform the operation it intends the "=" operator to have (usually an extended form of copying).

This same form of "extended copying" is also necessary for PHP's zval containers. Again, in the case of an array, this extended copying would imply re-creation of all hash table entries relating to this array. For strings, proper memory allocation would have to be assured, and so on.

Zend ships with such a function, called **zend_copy_ctor()** (the previous PHP equivalent was **pval_copy_constructor()**).

A most useful demonstration is a function that accepts a complex type as argument, modifies it, and then returns the argument:

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
   return;
}

// do modifications to the parameter here

// now we want to return the modified container:
*return_value == *parameter;
zval_copy_ctor(return_value);
```

The first part of the function is plain-vanilla argument retrieval. After the (left out) modifications, however, it gets interesting: The container of parameter is assigned to the (predefined) return_value container. Now, in order to effectively duplicate its contents, the copy constructor is called. The copy constructor works directly with the supplied argument, and the standard return values are `FAILURE` on failure and `SUCCESS` on success.

If you omit the call to the copy constructor in this example, both parameter and return_value would point to the same internal data, meaning that return_value would be an illegal additional reference to the same data structures. Whenever changes occurred in the data that parameter points to, return_value might be affected. Thus, in order to create separate copies, the copy constructor must be used.

The copy constructor's counterpart in the Zend API, the destructor **zval_dtor()**, does the opposite of the constructor.

Fejezet 36. Returning Values

Returning values from your functions to PHP was described briefly in an earlier section; this section gives the details. Return values are passed via the return_value variable, which is passed to your functions as argument. The return_value argument consists of a zval container (see the earlier discussion of the call interface) that you can freely modify. The container itself is already allocated, so you don't have to run `MAKE_STD_ZVAL` on it. Instead, you can access its members directly.

To make returning values from functions easier and to prevent hassles with accessing the internal structures of the zval container, a set of predefined macros is available (as usual). These macros automatically set the correspondent type and value, as described in 36-1 Táblázat and 36-2 Táblázat.

> **Megjegyzés:** The macros in 36-1 Táblázat automatically *return* from your function, those in 36-2 Táblázat only *set* the return value; they don't return from your function.

Táblázat 36-1. Predefined Macros for Returning Values from a Function

Macro	Description
`RETURN_RESOURCE(resource)`	Returns a resource.
`RETURN_BOOL(bool)`	Returns a Boolean.
`RETURN_NULL()`	Returns nothing (a NULL value).
`RETURN_LONG(long)`	Returns a long.
`RETURN_DOUBLE(double)`	Returns a double.
`RETURN_STRING(string, duplicate)`	Returns a string. The duplicate flag indicates whether the string should be duplicated using **estrdup()**.
`RETURN_STRINGL(string, length, duplicate)`	Returns a string of the specified length; otherwise, behaves like `RETURN_STRING`. This macro is faster and binary-safe, however.
`RETURN_EMPTY_STRING()`	Returns an empty string.
`RETURN_FALSE`	Returns Boolean false.
`RETURN_TRUE`	Returns Boolean true.

Táblázat 36-2. Predefined Macros for Setting the Return Value of a Function

Macro	Description
`RETVAL_RESOURCE(resource)`	Sets the return value to the specified resource.
`RETVAL_BOOL(bool)`	Sets the return value to the specified Boolean value.
`RETVAL_NULL`	Sets the return value to NULL.
`RETVAL_LONG(long)`	Sets the return value to the specified long.
`RETVAL_DOUBLE(double)`	Sets the return value to the specified double.
`RETVAL_STRING(string, duplicate)`	Sets the return value to the specified string and duplicates it to Zend internal memory if desired (see also `RETURN_STRING`).

`RETVAL_STRINGL(string, length, duplicate)`	Sets the return value to the specified string and forces the length to become length (see also `RETVAL_STRING`). This macro is faster and binary-safe, and should be used whenever the string length is known.
`RETVAL_EMPTY_STRING`	Sets the return value to an empty string.
`RETVAL_FALSE`	Sets the return value to Boolean false.
`RETVAL_TRUE`	Sets the return value to Boolean true.

Complex types such as arrays and objects can be returned by using **array_init()** and **object_init()**, as well as the corresponding hash functions on return_value. Since these types cannot be constructed of trivial information, there are no predefined macros for them.

Fejezet 37. Printing Information

Often it's necessary to print messages to the output stream from your module, just as print() would be used within a script. PHP offers functions for most generic tasks, such as printing warning messages, generating output for phpinfo(), and so on. The following sections provide more details. Examples of these functions can be found on the CD-ROM.

zend_printf()

zend_printf() works like the standard printf(), except that it prints to Zend's output stream.

zend_error()

zend_error() can be used to generate error messages. This function accepts two arguments; the first is the error type (see `zend_errors.h`), and the second is the error message.

```
zend_error(E_WARNING, "This function has been called with empty arguments");
```

37-1 Táblázat shows a list of possible values (see 37-1 Ábra). These values are also referred to in `php.ini`. Depending on which error type you choose, your messages will be logged.

Táblázat 37-1. Zend's Predefined Error Messages.

Error	Description
`E_ERROR`	Signals an error and terminates execution of the script immediately .
`E_WARNING`	Signals a generic warning. Execution continues.
`E_PARSE`	Signals a parser error. Execution continues.
`E_NOTICE`	Signals a notice. Execution continues. Note that by default the display of this type of error mes
`E_CORE_ERROR`	Internal error by the core; shouldn't be used by user-written modules.
`E_COMPILE_ERROR`	Internal error by the compiler; shouldn't be used by user-written modules.
`E_COMPILE_WARNING`	Internal warning by the compiler; shouldn't be used by user-written modules.

Ábra 37-1. Display of warning messages in the browser.

Including Output in phpinfo()

After creating a real module, you'll want to show information about the module in phpinfo() (in addition to the module name, which appears in the module list by default). PHP allows you to create your own section in the phpinfo() output with the `ZEND_MINFO()` function. This function should be placed in the module descriptor block (discussed earlier) and is always called whenever a script calls phpinfo().

PHP automatically prints a section in phpinfo() for you if you specify the `ZEND_MINFO` function, including the module name in the heading. Everything else must be formatted and printed by you.

Typically, you can print an HTML table header using **php_info_print_table_start()** and then use the standard functions **php_info_print_table_header()** and **php_info_print_table_row()**. As arguments, both take the number of columns (as integers) and the column contents (as strings). 37-1 Példa shows a source example and its output. To print the table footer, use **php_info_print_table_end()**.

Példa 37-1. Source code and screenshot for output in phpinfo().

```
php_info_print_table_start();
php_info_print_table_header(2, "First column", "Second column");
php_info_print_table_row(2, "Entry in first row", "Another entry");
php_info_print_table_row(2, "Just to fill", "another row here");
php_info_print_table_end();
```

Execution Information

You can also print execution information, such as the current file being executed. The name of the function currently being executed can be retrieved using the function **get_active_function_name()**. This function returns a pointer to the function name and doesn't accept any arguments. To retrieve the name of the file currently being executed, use **zend_get_executed_filename()**. This function accesses the executor globals, which are passed to it using the `TSRMLS_C` macro. The executor globals are automatically available to every function that's called directly by Zend (they're part of the `INTERNAL_FUNCTION_PARAMETERS` described earlier in this chapter). If you want to access the executor globals in another function that doesn't have them available automatically, call the macro `TSRMLS_FETCH()` once in that function; this will introduce them to your local scope.

Finally, the line number currently being executed can be retrieved using the function **zend_get_executed_lineno()**. This function also requires the executor globals as arguments. For examples of these functions, see 37-2 Példa.

Példa 37-2. Printing execution information.

```
zend_printf("The name of the current function is %s<br>", get_active_function_name(TSRMLS
zend_printf("The file currently executed is %s<br>", zend_get_executed_filename(TSRMLS_C)
zend_printf("The current line being executed is %i<br>", zend_get_executed_lineno(TSRMLS_
```

Fejezet 38. Startup and Shutdown Functions

Startup and shutdown functions can be used for one-time initialization and deinitialization of your modules. As discussed earlier in this chapter (see the description of the Zend module descriptor block), there are global, module, and request startup and shutdown events.

The global startup functions are called once when PHP starts up; similarly, the global shutdown functions are called once when PHP shuts down. Please note that they’re really only called *once*, not when a new Apache process is being created!

The module startup and shutdown functions are called whenever a module is loaded and needs initialization; the request startup and shutdown functions are called every time a request is processed (meaning that a file is being executed).

For dynamic extensions, module and request startup/shutdown events happen at the same time.

Declaration and implementation of these functions can be done with macros; see the earlier section "Declaration of the Zend Module Block" for details.

Fejezet 39. Calling User Functions

You can call user functions from your own modules, which is very handy when implementing callbacks; for example, for array walking, searching, or simply for event-based programs.

User functions can be called with the function **call_user_function_ex()**. It requires a hash value for the function table you want to access, a pointer to an object (if you want to call a method), the function name, return value, number of arguments, argument array, and a flag indicating whether you want to perform zval separation.

```
ZEND_API int call_user_function_ex(HashTable *function_table, zval *object,
zval *function_name, zval **retval_ptr_ptr,
int param_count, zval **params[],
int no_separation);
```

Note that you don’t have to specify both function_table and object; either will do. If you want to call a method, you have to supply the object that contains this method, in which case **call_user_function()**automatically sets the function table to this object’s function table. Otherwise, you only need to specify function_table and can set object to `NULL`.

Usually, the default function table is the "root" function table containing all function entries. This function table is part of the compiler globals and can be accessed using the macro `CG`. To introduce the compiler globals to your function, call the macro `TSRMLS_FETCH` once.

The function name is specified in a zval container. This might be a bit surprising at first, but is quite a logical step, since most of the time you’ll accept function names as parameters from calling functions within your script, which in turn are contained in zval containers again. Thus, you only have to pass your arguments through to this function. This zval must be of type `IS_STRING`.

The next argument consists of a pointer to the return value. You don’t have to allocate memory for this container; the function will do so by itself. However, you have to destroy this container (using **zval_dtor()**) afterward!

Next is the parameter count as integer and an array containing all necessary parameters. The last argument specifies whether the function should perform zval separation - this should always be set to `0`. If set to `1`, the function consumes less memory but fails if any of the parameters need separation.

39-1 Példa shows a small demonstration of calling a user function. The code calls a function that’s supplied to it as argument and directly passes this function’s return value through as its own return value. Note the use of the constructor and destructor calls at the end - it might not be necessary to do it this way here (since they should be separate values, the assignment might be safe), but this is bulletproof.

Példa 39-1. Calling user functions.

```
zval **function_name;
zval *retval;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &function_name) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

if((*function_name)->type != IS_STRING)
{
    zend_error(E_ERROR, "Function requires string argument");
}

TSRMSLS_FETCH();
```

```
if(call_user_function_ex(CG(function_table), NULL, *function_name, &retval, 0, NULL, 0)
{
    zend_error(E_ERROR, "Function call failed");
}

zend_printf("We have %i as type<br>", retval->type);

*return_value = *retval;
zval_copy_ctor(return_value);
zval_ptr_dtor(&retval);



<?php

dl("call_userland.so");

function test_function()
{

    print("We are in the test function!<br>");

    return("hello");

}

$return_value = call_userland("test_function");

print("Return value: \"$return_value\"<br>");
?>
```

Fejezet 40. Initialization File Support

PHP 4 features a redesigned initialization file support. It's now possible to specify default initialization entries directly in your code, read and change these values at runtime, and create message handlers for change notifications.

To create an .ini section in your own module, use the macros `PHP_INI_BEGIN()` to mark the beginning of such a section and `PHP_INI_END()` to mark its end. In between you can use `PHP_INI_ENTRY()` to create entries.

```
PHP_INI_BEGIN()
PHP_INI_ENTRY("first_ini_entry",  "has_string_value", PHP_INI_ALL, NULL)
PHP_INI_ENTRY("second_ini_entry", "2",                PHP_INI_SYSTEM, OnChangeSecond)
PHP_INI_ENTRY("third_ini_entry",  "xyz",              PHP_INI_USER, NULL)
PHP_INI_END()
```

The `PHP_INI_ENTRY()` macro accepts four parameters: the entry name, the entry value, its change permissions, and a pointer to a change-notification handler. Both entry name and value must be specified as strings, regardless of whether they really are strings or integers.

The permissions are grouped into three sections:`PHP_INI_SYSTEM` allows a change only directly in the `php3.ini` file; `PHP_INI_USER` allows a change to be overridden by a user at runtime using additional configuration files, such as `.htaccess`; and `PHP_INI_ALL` allows changes to be made without restrictions. There's also a fourth level, `PHP_INI_PERDIR`, for which we couldn't verify its behavior yet.

The fourth parameter consists of a pointer to a change-notification handler. Whenever one of these initialization entries is changed, this handler is called. Such a handler can be declared using the `PHP_INI_MH` macro:

```
PHP_INI_MH(OnChangeSecond);             // handler for ini-entry "second_ini_entry"

// specify ini-entries here

PHP_INI_MH(OnChangeSecond)
{

    zend_printf("Message caught, our ini entry has been changed to %s<br>", new_value);

    return(SUCCESS);

}
```

The new value is given to the change handler as string in the variable new_value. When looking at the definition of `PHP_INI_MH`, you actually have a few parameters to use:

```
#define PHP_INI_MH(name) int name(php_ini_entry *entry, char *new_value,
                                  uint new_value_length, void *mh_arg1,
                                  void *mh_arg2, void *mh_arg3)
```

All these definitions can be found in `php_ini.h`. Your message handler will have access to a structure that contains the full entry, the new value, its length, and three optional arguments. These optional arguments can be specified with the additional macros `PHP_INI_ENTRY1` (allowing one additional argument), `PHP_INI_ENTRY2` (allowing two additional arguments), and `PHP_INI_ENTRY3` (allowing three additional arguments).

The change-notification handlers should be used to cache initialization entries locally for faster access or to perform certain tasks that are required if a value changes. For example, if a constant

connection to a certain host is required by a module and someone changes the hostname, automatically terminate the old connection and attempt a new one.

Access to initialization entries can also be handled with the macros shown in 40-1 Táblázat.

Táblázat 40-1. Macros to Access Initialization Entries in PHP

Macro	Description
`INI_INT(name)`	Returns the current value of entry `name` as integer (long).
`INI_FLT(name)`	Returns the current value of entry `name` as float (double).
`INI_STR(name)`	Returns the current value of entry `name` as string. *Note:* This string is not duplicated, but inst
`INI_BOOL(name)`	Returns the current value of entry `name` as Boolean (defined as zend_bool, which currently m
`INI_ORIG_INT(name)`	Returns the original value of entry `name` as integer (long).
`INI_ORIG_FLT(name)`	Returns the original value of entry `name` as float (double).
`INI_ORIG_STR(name)`	Returns the original value of entry `name` as string. Note: This string is not duplicated, but inst
`INI_ORIG_BOOL(name)`	Returns the original value of entry `name` as Boolean (defined as zend_bool, which currently m

Finally, you have to introduce your initialization entries to PHP. This can be done in the module startup and shutdown functions, using the macros `REGISTER_INI_ENTRIES()` and `UNREGISTER_INI_ENTRIES()`:

```
ZEND_MINIT_FUNCTION(mymodule)
{

    REGISTER_INI_ENTRIES();

}

ZEND_MSHUTDOWN_FUNCTION(mymodule)
{

    UNREGISTER_INI_ENTRIES();

}
```

Fejezet 41. Where to Go from Here

You’ve learned a lot about PHP. You now know how to create dynamic loadable modules and statically linked extensions. You’ve learned how PHP and Zend deal with internal storage of variables and how you can create and access these variables. You know quite a set of tool functions that do a lot of routine tasks such as printing informational texts, automatically introducing variables to the symbol table, and so on.

Even though this chapter often had a mostly "referential" character, we hope that it gave you insight on how to start writing your own extensions. For the sake of space, we had to leave out a lot; we suggest that you take the time to study the header files and some modules (especially the ones in the `ext/standard` directory and the MySQL module, as these implement commonly known functionality). This will give you an idea of how other people have used the API functions - particularly those that didn’t make it into this chapter.

Fejezet 42. Reference: Some Configuration Macros

config.m4

The file `config.m4` is processed by `buildconf` and must contain all the instructions to be executed during configuration. For example, these can include tests for required external files, such as header files, libraries, and so on. PHP defines a set of macros that can be used in this process, the most useful of which are described in 42-1 Táblázat.

Táblázat 42-1. M4 Macros for `config.m4`

Macro	Description
`AC_MSG_CHECKING(message)`	Prints a "checking <mess
`AC_MSG_RESULT(value)`	Gives the result to `AC_MS`
`AC_MSG_ERROR(message)`	Prints message as error m
`AC_DEFINE(name,value,description)`	Adds `#define` to `php_c`
`AC_ADD_INCLUDE(path)`	Adds a compiler include
`AC_ADD_LIBRARY_WITH_PATH(libraryname,librarypath)`	Specifies an additional lib
`AC_ARG_WITH(modulename,description,unconditionaltest,conditionaltest)`	Quite a powerful macro, a
`PHP_EXTENSION(modulename, [shared])`	This macro is a *must* to ca

Fejezet 43. API Macros

A set of macros was introduced into Zend's API that simplify access to zval containers (see 43-1 Táblázat).

Táblázat 43-1. API Macros for Accessing zval Containers

Macro	Refers to
`Z_LVAL(zval)`	(zval).value.lval
`Z_DVAL(zval)`	(zval).value.dval
`Z_STRVAL(zval)`	(zval).value.str.val
`Z_STRLEN(zval)`	(zval).value.str.len
`Z_ARRVAL(zval)`	(zval).value.ht
`Z_LVAL_P(zval)`	(*zval).value.lval
`Z_DVAL_P(zval)`	(*zval).value.dval
`Z_STRVAL_P(zval_p)`	(*zval).value.str.val
`Z_STRLEN_P(zval_p)`	(*zval).value.str.len
`Z_ARRVAL_P(zval_p)`	(*zval).value.ht
`Z_LVAL_PP(zval_pp)`	(**zval).value.lval
`Z_DVAL_PP(zval_pp)`	(**zval).value.dval
`Z_STRVAL_PP(zval_pp)`	(**zval).value.str.val
`Z_STRLEN_PP(zval_pp)`	(**zval).value.str.len
`Z_ARRVAL_PP(zval_pp)`	(**zval).value.ht