Tanuljunk PHP-ül kezdőknek

Fábián Zoltán

2005

v0.81

Tartalomjegyzék

1 Bevezetés

1.1 Mire jó ez a jegyzet?

A jegyzetnek az a célja, hogy bevezesse az olvasót a ma robbanásszer űen terjedő PHP nyelv világába, megtanítsa az alapvető eljárásokat, fogalmakat és képessé tegye az olvasót saját PHP scriptek írására. Használjuk továbbá a Szily Kálmán Műszaki Középiskolában.

Ez nem teljes PHP dokumentáció! Teljes dokumentációt az Internetr ől lehet beszerezni, részben magyar nyelven az alábbi címekről:

http://www.php.net dokumentáció angolul

http://hu.php.net/docs.php oldal magyarul.

A jegyzetben található példaprogramokat, ötleteket, részben az alábbi forrásokból szemezgettem és ezúton köszönetet mondok nekik :

http://hu.php.net A PHP site magyar oldalai

php-lista@Gimli.externet.hu Magyar PHP lista

http://php4.x3.hu, Korsós István (KI),

kefa@mail.datanet.hu PHP-s cikksorozata a PC-World-ből, resource-ok PHP és egyéb témákban

http://phpbuilder.com/columns/ying20000602.php3?page=1 sessionok, Ying Zhang cikke

http://phpmailer.sourceforge.net Emailküldés

http://php.weblogs.com/ADODB ADODB – adatbáziskezelés

http://adodb.sourceforge.net

http://pear.php.net Pear programcsomag letöltése

http://smarty.php.net Smarty template rendszer

http://www.phpclass.org PHP alkalmazások és class-ok feltalálási helye

1.2 A PHP rövid története

A PHP története 1994 őszére nyúlik vissza, amikor a munkát keres ő Rasmus Lerdorf egy Perl CGI szkriptet használt a Web odalát felkeres ők regisztrálására. A látogatókat naplózó kódot "PHP-tools for Personal Home Page"-nek nevezte el. Az els ő nyilvános változat úgy 1995 táján látott napvilágot. Ez még csak néhány egyszerűbb feladatra volt használható, többek között számlálót, vendégkönyvet tartalmazott.

A PHP fejlesztése a Torontói Egyetemen folytatódott, ahol Rasmus Lerdorf olyan interfészt fejlesztett ki, aminek segítségével a HTML kódba ágyazott speciális utasítások közvetlenül érték el az egyetemi adatbázisokat. A rendszert Rasmus "Form Interpreter"-nek, FI-nek nevezte el. Az FI-ben használt elv már megegyezett a PHP alapelvével, miszerint a HTML kódba beágyazott utasításokat értelmezte és hajtotta végre az FI értelmez ője. Később a PHP és az FI összeházasításából született meg az els ő széles körben használt parancsértelmez ő a PHP/FI. Ez tartalmazta a PHP és az FI addigi szolgáltatásait, s őt az mSQL adatbázisok elérését is támogatta. Rasmus eleinte eljátszadozott a gondolattal, hogy a PHP-t kereskedelmi termékké teszi, de olyan komoly mennyiségű visszajelzést kapott más programozóktól, különböz ő kiegészítéseket és hibajavításokat küldve a PHP-hez, hogy letett ebbéli szándékáról. A PHP fejl ődéséhez és sokrétűségéhez nagymértékben hozzájárult külső programozók szabad és ingyenes részvétele a rendszer fejlesztésében. A PHP a mai napig is ingyenes termék, és ez valóban nagyon jó dolog.

Az els ő verzió megjelenését ől kezdve a PHP felhasználói tábora töretlenül növekedett. 1996-ban közel 15.000 Web oldalon használták a PHP/FI-t, 1997-ben már több mint 50.000 Web oldalon. Ebben az évben kezd ődött el a PHP sokkal jobban szervezett továbbfejlesztése. A PHP/FI-t értelmez őjét szinte az alapoktól kezdve újraírták, átemelve a PHP/FI-ben alkalmazott technikákat és kódot, de számos újat is hozzátéve. Így alakult ki a PHP 3-as változata, ami gyakorlatilag rendelkezett mindazokkal a képességekkel, amik a PHP népszerűségét megalapozták. A PHP fejl ődése azonban nem áll meg. Jelenleg az 5.1.x változatnál tart a fejlesztés, de köszönhet ően a "szabad szoftver" filozófiának nem valószínű, hogy itt megreked.

1.3 Mi a PHP?

Egy majdnem általános célú programozási nyelv, amely mára sokféle területen alkalmazható, amit a Weben keresztül meg lehet oldani. Dinamikus oldalak, adatbázis-kezelés, akár ügyviteli alkalmazások, levelezés, portálok, grafikai alkalmazások, file-kezelés, távoli adminisztráció, stb.

Mielőtt a PHP működését részleteznénk, meg kell ismerkedni egy kicsit a Web-en lévő alkalmazások lelkivilágával.

Amikor a böngészőben beírjuk egy olyan oldal nevét, és letöltjük azt, akkor az Interneten lévő WEB szerver küldi el a böngészőnknek a kívánt oldalt. Ez az oldal egy HTML oldal, egy ASCII szöveg állomány, amely megérkezve a böngészőnkre azt a böngésző értelmezi, és a tartalmát megjeleníti.

Ezek statikus oldalak, mivel a következő és az utána következő kérésekkor mindig pontosan ugyanazt az oldalt fogjuk újra meg újra megkapni, hiszen a szerveren ez egy file. Abban az esetben, ha azt szeretnénk, hogy az oldal tartalma változzon, azaz dinamikus oldalt szeretnénk, valamilyen módon az oldalt a szerveren létre kell hozni, módosítani kell. Ennek érdekében ki kell egészíteni a WEB-szervert olyan alkalmazásokkal, amelyek az oldal kérésekor futás közben állítják elő az oldal tartalmát, majd odaadják a szervernek, amely az eredményt továbbítja a böngésző felé. Az ilyen programokat összefoglaló néven **CGI** programoknak hívjuk, ami a Common Gateway Interface kifejezés rövidítése. Ezek a programok tehát új funkcionalitással bővítik ki a WEB szervereket. A legfontosabb ebben az, hogy ez által a böngészők interaktív módon tudnak kapcsolatot teremteni a WEB szerverrel, adatokat tudnak bevinni neki, a bevitt információ alapján változik a visszaadott érték. Az ilyen rendszer működése olyan, hogy a böngészőben beírt információt megkapja a WEB szerver, átadja a CGI programnak, ami az információ birtokában feldolgozza és visszaküldi a WEB szervernek, amely továbbítja a böngésző felé HTML kód formájában. A CGI programok vagy a szerver operációs rendszerén futni képes programok, amelyeket valamilyen nyelven, pl. C-ben írtak meg és fordítottak le, vagy úgynevezett scriptek, amelyek egy értelmező program közreműködésével futnak.

A **PHP programokat egy ilyen értelmező** futtatja a szerveren akkor, ha a böngésző **PHP, PHP3, PHP4**, vagy hasonló kiterjesztésű file-okra hivatkozik. Ekkor a WEB szerver meghívja a PHP értelmező programot (interpretert), amely értelmezi a kérdéses oldalt, majd az eredményt Web oldal formájában visszaadja a WEB szervernek, amely továbbítja a böngészőnek.

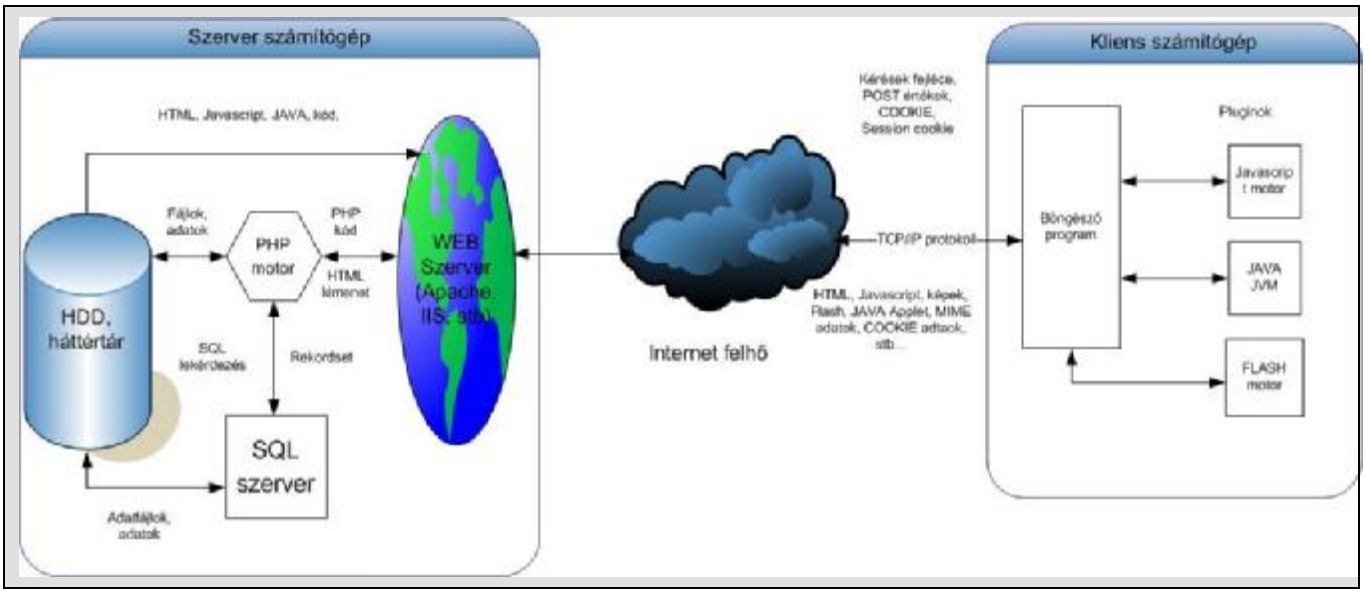
Tehát a PHP program egy WEB szerveren futó script, amelynek az eredménye egy WEB böngészőn keresztül jelenik meg.

A Javascriptek, a Vbscriptek JAVA appletek illetve Flash movie-k a böngészőn hajtódnak végre, a PHP scriptek a szerveren. A PHP script a böngészőn soha nem jelenik meg (ha megjelenik, akkor baj **J**), csak az a HTML vagy egyéb kód, amelyet előállít.

A böngésző és a szerver közötti kommunikáció folyamata:

- A böngészőbe a felhasználó beír egy URL-t, rákattint egy linkre, ekkor ugynevezett GET metódusú kérést küld el a böngésző a szervernek, vagy egy űrlapon rákattint az elküldés gombra, ekkor POST metódusú kérést küld a szervernek a böngésző. A kérés tulajdonképpen egy adatcsomag. A kérés részei:
    - A kérésben elmegy az URL, a metódus fajtája (POST vagy GET), a használandó protokol (HTTP 1.0 vagy HTTP 1.1)
    - POST metódus esetén a kérésben elmennek az űrlapban kitöltött adatmezők adatai
    - A böngésző neve, és speciális tulajdonságai
    - A böngésző által lekezelni képes speciális adatfajták szabványos nevei
    - A böngészőben tárolt és az adott domainhez tartozó cookie-k adatai
- Ha az URL a szerveren .html, .htm ,.gif, .jpg, .AVI, .mpg, stb… file-t jelöl meg, akkor a WEBszerver kikeresi a saját háttértárán a megfelelő fájlt és elküldi a böngésznek különböző plusz, úgynevezett fejlécadatok (header)kíséretében. A headerben lévő adatok az alábbiak lehetnek:
    - A kódolás nyelve, a cache-ek működésére vonatkozó parancsok,
    - Az esetlegesen elküldendő cookie-k
    - A SESSION-ökhöz tartozó un. session cookie-k.

- Ha az URL .php kiterjesztésű file-t jelöl, akkor a webszerver átadja az URL-t a PHP motornak.
    - A PHP motor megkeresi a háttértáron a megadott file-t
    - Betölti szintaktikailag ellen őrzi a betöltött php file tartalmát, majd elkezdi értelmezni és végrehajtani azt.
    - Ha szükséges a PHP-hez konfigurált egyéb modulokat elindít, kommunikál velük, mint pl. egy SQL szerver
    - A PHP kód segítségével a kimenetre lehet küldeni header parancsokat, cookie-k és session-ök adatait, mielőtt bármi HTML kódot kiküldene a PHP motor.
    - A végrehajtás során a PHP oldalakon keverni lehet a HTML és a PHP kódot. Ha az értelmez ő HTML kód részt talál, akkor azt a kódot kiküldi a kimenetre változatlanul.
    - A program futása során kiíró utasításokkal előállítja a HTML kimenetet és kiküldi a kimenetre.
    - A PHP motor által előállított HTML kimenetet a webszerver kapja meg.
    - A WEbszerver a tartalmat kiküldi a böngészőnek.
- A böngésző a megkapott információt betölti és értelmezi.
    - Ha a kódban javascript, flash, java vagy egyéb nem kifejezetten HTMl kód található, akkor a böngésző elindítja a kliens számítógépre telepített megfelelő feldolgozó modult.



2 Milyen rendszeren használható a PHP?

2.1.1 Milyen operációs rendszer?

Létezik az összes elterjedt operációs rendszeren, Win32, Linux, Solaris, BSD, Netware. A PHP hazsnálatához általában szükség van egy WEB szerverre is. Gondolva a tanulásra és az általában meglév ő számítástechnikai környezetre én Linux, W2000 vagy Windows XP rendszert javaslok.

2.1.2 Milyen WEB szervert?

A PHP telepítése el őtt mindenképpen szükségünk lesz egy m űködő WEB szerverre. Ehhez a Windows 2000 és Windows XP alatt használhatjuk a Microsoft Internet Information Server nev ű szoftverét, amely a rendszer része, vagy az internetről letöltött Apache WEB szervert.

Mi az Apache WEB szervert ajánljuk, annak egyszer ű telepítése, egyszer ű beállítása miatt. Megjegyezzük, hogy az Apache beállítása majdnem ugyanaz LINUX alatt és Win32 alatt is, tehát ennek a rendszernek a használatakor nagyon kis különbségek kellenek a LINUX-os verzióra való átálláshoz.

Én Windows XP-t Apache 2.0.xx rendszert használok. Léteznek el őre összeállított és nagyon egyszer űen telepíthető csomagok windows-ra. Ilyen pl. a BigApache (http://www.bigapache.org), illetve a továbbiakban felsorolok még ilyen rendszert.

PHPTriad: http://Sourceforge.net/phptriad

Uniform WEB server http://www.uniform.server.com

Más WEB szerverek. Amelyek tudnak futtatni PHP-t:

AbysWS Personal WEB szerver: http://www.aprelium.com

3 A PHP4 telepítése (Win32 változat)

A PHP telepítése operációs rendszerenként és WEB szerverenként más és más. A telepítésr ől további információ a http://hu.php.net és a http://www.php4win.de oldalakon található.

Ha a korábban leírt rendszert akarunk telepíteni, akkor általában egy egyszer ű windowsos program telwepítésén kell végigmennünk. A telepítés után lesz valahol egy PHP nevű könyvtárunk.

Ha az Apache és a PHP további konfigurálására van szükség, akkor egy PHP.INI nev ű file-ban kell ezt megtenni, amelynek a Windows könyvtárban a helye.

Az Apache konfigurálásához az apache/conf könyvtárban a httpd.conf file-t kell állítani. Minden állítás után az Apache szervert újra kell indítani.

A PHP4 esetén a php4ts.dll-t a C:\WINDOWS\SYSTEM32 vagy a C:\WINNT\SYSTEM32 könyvtárba kell tenni.

3.1.1 Konfigurálás Apache WEB szerver 1.3.xx vagy 2.0.xx esetén

1. Az Apache szerver konfigurációs állománya például az C:\Apache\conf\httpd.conf file. Szerkeszd meg ezt a file-t, az alábbi módon:

   Apache modul esetén:

```
# Apache modul esetén
LoadModule php4_module c:/php/sapi/php4apache.dll
AddType application/x-httpd-php .php4
```

   Másold be a \Winnt\system32 vagy a \windows\system könyvtárba a php4ts.dll file-t

   CGI modul esetén:

```
#for the cgi bináris esetben ScriptAlias /php4/ "C:/php/"
Action application/x-httpd-php4 "/php4/php.exe"
AddType application/x-httpd-php4 .php
```

Sajnos jelenleg az Apache modul nem fut a 2.0.xx-es verzióban.

2. Keresd meg a `DirectoryIndex` parancsot a konfigurációs állományban és egészítsd ki az alábbi módon

```
DirectoryIndex index.html index.php index.php3 index.php4
```

3. Indítsd újra az Apache szervert,

4. Írd meg a kedvenc ASCII editorod segítségével az alábbi tartalmú scriptet, mentsd el a szervered gyökérkönyvtárába, majd hívd meg a böngészőben az alábbi módon:

```
<?php
   echo phpinfo();
?>
```

5. Próbáld ki az alábbi programocskát a Böngésződdel a szervered root könyvtárából:

   `http://localhost/proba.php`

Ennek hatására lefut a script és kiírja a böngésződbe az éppen használt php rendszer rengeteg paraméterét.

4 Eszközök a PHP használatához

Felmerül a kérdés, hogy mi kellhet a PHP használatához a fentieken kívül. A várakozással ellentétben nem sok, de azért itt összefoglalom a lehetőségeket:

Kell **egy ASCII szövegszerkeszt ő**, kezdetben jó a NOTEPAD.EXE a Windowsból is. Ha ennél komolyabbra vágysz, akkor válaszd mondjuk az EditPlus nev ű editort, amely a http://www.editplus.com címr ől tölthet ő le. Számozza az oldalakat, és még a szintaktikát is színezi.

Ha az oldalak bonyolultak, akkor tudom ajánlani a ZEND alkalmazáscsomagot, amely fizet ős, de nagyon jó (http://ww.zend.com) vagy a phpEdit nevű fejlesztőeszközt (http://www.phpedit.com).

Ezen kívül sok más hasonló editor van forgalomban. A nagyok közül tudnám ajánlani a Macromedia Dreamweaver Ultradev 4 vagy a Dreamweaver MX csomagot is.

Nem árt, ha van egy jó HTML editorod, mert a php oldalak jelent ős részben azért HTML kódból is állnak. A freeware vagy shareware programok között nagy a választék van, pl. CoffeCup.

Kell egy böngész ő, ami adott a Windows-okban Internet Explorer 4/5/6-ot. Az IE azonban nem teljesen szabványos, ezért alternatívaként használható a Firefox-ot (www.mozilla.org), a Mozilla (www.mozilla.org) illetve az Opera, illetve bármilyen kedvenc böngésző. Az opera letölthető a www.opera.com -ról.

Szükséged lesz egy jó Help-re. Én a http://hu.php.net/docs.php oldaláról a CHM (Windows-os Help file) változatot javaslom, de használhatod a HTML verziókat is, és ha gyors Internet kapcsolatod van, akkor a Internetről is lehet online módon használni őket. A HTML verzió sokkal több példát tartalmaz a hozzászólók nagy száma miatt.

Nem árt, ha a HTTP-r ől is van egy jó Help-ed. A http://www.htmlhelp.com/ -ról letölthet ő többféle formában, köztük Windows-os Help formájában a HTML nyelv szintaktikája.

Elkerülhetetlen, hogy a PHP fejleszt ők megismerkedjenek a Javascript szintaktikájával is. A Javascriptek használatához Javascript helpre is szükség van. Erre a célra több forrás létezik, de sajnos magyar nyelven és Windows-os Help formájában nem tudok róla: Talán célszer ű a következ őt Web oldalt használni http://weblabor.hu/leiras/javascr/ vagy megnézni az alábbi oldalt:

Szükség lehet egy jó HTML / Javascript könyvre, például

Bócz Péter – Szász Péter: A világháló lehetőségei

És a CSS szabványok ismeretére, mert a formázás manapság korszer ű és elterjedt módja CSS-en alapul. A http://www.weblabor.hu websiteról kiindulva találunkmegfelelő CSS leírást is.

Ha adatbázisokat akarsz használni a PHP oldalaidon, akkor windowsos környezetben ajánlom a Microsoft Access használatát un. ODBC drivereken keresztül, vagy az SQLite (www.sqlite.org) adatbáziskezel őt, amely kicsi, gyors, ingyenes, és a PHP5 része, illetve a MySQL-t, ha (www.mysql.org) ha PHP4-et vagy a MySQLi-t, ha PHP5-öt használsz.

A hatékony fejlesztéshez a kés őbbiekben célszer ű megismerkedni a Pear (http://pear.php.net) oldalon található rutinkönyvtárral, az ADODB adatbázis absztrakciós réteggel (http://adodb.sourceforge.net), a Smarty sablonrendszerrel (http://smarty.php.net).

5 Egy HTML és egy PHP oldal szerkezete

Mielőtt továbbmennénk nézzük, mit akarunk programozni? Hogyan is néz ki egy HTML oldal:

```
<HTML>
    <head>
        <TITLE>Próba web oldal</TITLE>
    </head>
    <BODY>
        <P>
            Hello World!!!
        </P>
    </BODY>
</HTML>
```

Na ez nagy durranás volt. A kisbet ű-nagybetű kérdés nem számít. A bekezdéses írásmódot az áttekinthet őségért használok, de mint látjuk kés őbb, ez rendkívül fontos! A böngész ők a sor vége jelet, a szóközöket és a tabulátor jeleket kihagyják, ezért akár az egészet egy sorba is írhatnánk. A fenti HTML oldalt az alábbi php scripttel tudjuk előállíttatni:

```
<HTML>
    <head>
        <TITLE>Proba web oldal</TITLE>
    </head>
    <BODY>
        <P>
      <?php
      echo ”Hello World!!!”;
  ?>
        </P>
    </BODY>
</HTML>
```

Tisztán látszik, hogy a program szerkezete nagy vonalakban ugyanaz, de azért vannak különbségek. A PHP értelmező a kód első sorait változatlanul továbbítja a Web szervernek, de amikor a **<?php** taghez ér, értelmezi az ott lévő sorokat, és az **echo** paranccsal kiírja a Web szerver felé az **echo** utáni szöveget.

A következő sorban lévő **?>** tag jelzi az értelmezőnek a PHP script végét.

Egy oldalon többször is megnyithatjuk a scriptet és bezárhatjuk, azaz keverhetjük a php és a HTML kódot, s őt ha van merszünk, akkor beiktathatunk javascriptes részeket is.

Az alábbi lehetőségeink vannak arra, hogy php scriptet helyezzünk el egy oldalon:

```
<?php ..............php kód ..............?>
<script language=”PHP”> ..... php kód ...</script>
```

Ha beírjuk a PHP.INI-ben, a short_open_tag= on sort, akkor ez a megoldás is lehetséges:

```
<?..... php kód ....?>
```

Ha beírjuk a PHP.INI-be az asp_tags = on sort, akkor használható az alábbi szintaktika is:

```
<% ..... php kód ........%>
```

Speciális lehet őség. Ha egy változó értékét szeretnénk csak kiiratni és a short_open_tag = on sor be van írva, akkor egy változó értékét így adhatjuk át a web oldalnak legegyszer űbben:

```
<?=$valtozo ?>
```

6 A PHP nyelvi szabályai

A PHP nyelv szintaktikája nagyon hasonlít a C nyelvére.

A php scriptben lévő utasítások mindegyike után kötelezően ki kell tenni a **;** elválasztójelet.

Megjegyzés:

Bár a dokumentáció azt mondja, hogy egy PHP blokk záró tagja előtt nem kell kitenni, azért azt tanácsolom, hogy megszokás céljából mindenhol alkalmazzuk

A Megjegyzéseinket szintén C jellegű szintaktikával kétféleképpen tehetjük ki:

// Az egysoros kommentek, a sor végéig tarthatnak, mint a C-ben

jellel, mint a Unixos shell programoknál

/*.....*/ A több soros kommentek

```
<?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
?>
```

Amint látjuk a fentiek alapján a kiírandó szöveget **”…”** jelek közé tehetjük, de a jel pár lehet **’...’** is. A továbbiakban nézzük meg, hogy milyen típusú értékekkel dolgozhatunk.

7 Változók, adattípusok

A PHP-ben használhatunk változókat is. A változókban értékeket tárolhatunk. A változó neve el őtt mindig $ jel található. A PHP félig típusos nyelv.

7.1 A változó neve

A változó neve bet űvel vagy aláhúzás jellel kezd ődik és bármilyen alfanumerikus karakterrel, illetve 127...255 ASCII kódú karakterrel folytatódhat. A kis és nagybetűk különböznek!

7.2 A változó értéke

Amikor először adunk értéket egy változónak, akkor jön létre a változó.

```
$a = 5;
$todo = ”szöveg”;
```

Ha egy változó értékét azel őtt vizsgáljuk meg, miel őtt értéket adunk neki, az értéke NULL lesz és a PHP egy NOTICE üzenetet küld nekünk.

Egy változó típusát többféleképpen lehet meghatározni.

- Értéket adunk neki, az érték alapján létrejött változónak a típusa is egyértelművé válik.

Beállítjuk a típusát a settype (“változónév”, “típus”) paranccsal, ahol a típus az alábbiak közül választható: "integer", "double", "string", "array", "object"

- Típuskonverziót alkalmazunk. Az alkalmazható típusok ugyanazok, mint a fenti példában, hozzávéve, hogy az ”integer” lehet ”int” is, a ”double” lehet ”float” és ”real” is.

```
$a = (int) $b;
$c = (string) $d;
```

A változók értékadásakor használhatunk hasonló trükköket, mint a C-ben:

```
$a = $b = 5;
```

A fenti kifejezést úgy kell értelmezni, mintha az alábbiakat írtuk volna le:

```
$b = 5;
$a = b;
```

A későbbiekben foglalkozunk részletesen az operátorokkal, amelyek egy változó értékét meghatározzák.

A változónak bármilyen módon értéket adtunk, akkor a típusát is meghatároztuk.

7.3 A változók és kifejezések típusai

A különböző programozási nyelveken megszokott típusok a PHP-ban is megtalálhatók:

Numerikus típusok:

int, integer - egész típus – értéke –32768-32767-ig tart

Float, double, real – Lebegőpontos

String A C-ben használt string fogalomhoz hasonló, ”...” vagy ’....’ jelek közé írt ASCII karaktersorozat.

Array Tömb típus. A tömb elemi tetsz őlegesen vegyesek lehetnek

Egy tömb lehet egy, két és több dimenziós tömb is. A tömb indexelése történhet hagyományosan 0-tól kezd ődő indexeléssel, vagy lehet úgynevezett **asszociatív tömböt** is létrehozni, amikor a tömbindex valamiféle string, vagy egyéb érték.

Ebben a példában egy dimenziós tömböt hozunk létre.

```
$array = array (1, "hello", 1, "world", "hello");
```

Az alábbi példában kétdimenziós asszociatív hozunk létre.

```
$fruits = array (
    "fruits"  => array ("a"=>"orange", "b"=>"banana", "c"=>"apple"),
    "numbers" => array (1, 2, 3, 4, 5, 6),
    "holes"   => array ("first", 5 => "second", "third")
);
```

A hagyományos tömbök feldolgozásához for vagy pedig while cklust hazsnálunk, az asszociatítv tömbökéhez foreach ciklust. A ciklusokról később bővebben szólunk.

Object - Objektum. A típus hasonlít a C++ objektumaira, de vannak lényeges különbségek is.

Egy objektum típusú változó létrehozásához el ő ször definiálni kell magát az osztályt a `class` kulcsszóval, hasonlóképpen, mint C++ -ban, majd a definíció után a `new()` operátorral létre lehet hozni a megfelel ő változót. A class részét képezik változók és a classhoz tartozó függvények is. A kés őbbiekben részletesen is szólunk az objektumokról. Az alábbiakban egy példát látunk.

```
<?php
class Kosar {
    var $dolgok;       // A kosárban lev      ő dolgok

    function berak ($sorsz, $db) {     // berak a kosárba $db darabot az $sorsz indexű dologból
        $this->dolgok[$sorsz] += $db;
    }
    function kivesz ($sorsz, $db) {   // kivesz a kosárból $db darabot az $sorsz indexű dologból
        if ($this->items[$sorsz] > $db) {
            $this->items[$sorsz] -= $db;
            return true;
        } else {
            return false;
        }
    }
}
$a = new Kosar;
?>
```

boolean – Logikai értékek. Két érték előre definiált, a TRUE és a FALSE

null - Olyan változók, amelyeknek nincsen beállított értékük

Resource – Er őforrások, mint például file vagy adatbáziskezel ő mutatója. Valójában ezek a memória egyes helyére mutató pointerek!

Unknown type - Ismeretlen típusok

Változók és kifejezések visszatérési értékének típusát le lehet kérdezni az alábbi függvényekkel:

`int empty ($a)`	Megadja, hogy a változó üres volt-e vagy nem.
`string gettype ($a)`	Visszaadja a változó típusát. A lehetséges típusok az alábbiak: "integer", "double", "string", "array", "object", "unknown type"
`void settype($var,$type)`	Beállítja egy változó típusát. A lehetséges típusok ($type) lehet: "integer", "double", "string", "array", "object"
`int intval($a,[$alap])`	Visszaadja a változó értékét egésszé konvertálva a megadott alapú számrendszerben. Az alap default értéke 10.
`int is_array($a)`	Megadja, hogy a változó tömb vagy sem
`int is_bool($a)`	Megadja, hogy a változó logikai típusú vagy sem

`int is_float ($a)`
`int is_double ($a)`
`int is_real ($a)`
	Megadja, hogy a változó lebegőpontos-e, vagy sem.

`int is_long ($a)` `int is_integer ($a)`	Megadja, hogy a változó egész vagy sem.

`int is_int ($a)`	
`int is_numeric ($a)`	Megadja, hogy a változó numerikus, numerikus szöveg vagy sem.
`int is_object ($a)`	Megadja, hogy a változó objektum vagy sem
`int is_resource ($a)`	Megadja, hogy a változó erőforrás azonosító vagy sem (pl. file handler)
`int is_string ($a)`	Megadja, hogy a változó string vagy sem.
`int isset (változó)`	Megadja, hogy be van-e állítva a változó. A visszatérési értéke hamis, ha nem és Igaz, ha van értéke a változónak
`void print_r(kifejezés)`	Ember számára olvasható információt ad egy változóról.

```
<?php
$a = array (1, 2, array ("a", "b", "c"));
print_r ($a);
?>
```

7.4 Előre definiált változók

A változók között vannak olyanok, amelyek a rendszerben el őre definiáltak. Ezeknek a változóknak a nevét és pillanatnyi értékét a phpinfo() függvény segítségével lehet kiíratni. A változókra hivatkozva természetesen az értékeket fel tudjuk használni, és azt tudjuk manipulálni.

Ezen változóknak az értékét a PHP.INI file-ban lehet beállítani, néhány érték a használt WEB szervert ől függ, további értékek pedig a futtató operációs rendszertől függnek.

A változók másik nagy csoportjai a rendszerben alkalmazott modulok értékei. Néhány fontosabb változó csoport

7.4.1 Environment

Az operációs rendszerr ől, a környezeti beállításokról, a WEB szerverr ől, a böngész őről, az aktuálisan meghívott lapról és a kliensről minden lényeges adatot felsorol, beleértve az IP címeket is.

7.4.2 PHP Variables - A PHP változók listája

További gyakran használt változók tömbjei

a szerver által szolgáltatott tömb és a

```
$_SERVER[”PATH”]
```

A környezeti változók

```
$_ENV[”valtnev”]
```

Formok kezelésénél használható változók listája

a GET metódussal elküldött változók listája és értékei

```
$_GET[”valtnev”]
```

POST metódussal elküldött változók listája és értékei

```
$_POST[”valtnev”]
```

Sessionokben használt változók listája és értékei

```
$_SESSION[”valtnev”]
```

A cookiek adatait tartalmazó változók.

```
$_COOKIE[”valtnev”]
```

A HTTP protokollban definiált további változók. Ezek közül a változók közül néhány megtalálható a környezeti változók között is.

A szerver IP címe

```
$_SERVER[”REMOTE_ADDR”]
```

a szerver neve

```
$_SERVER[”REMOTE_HOST”]
```

a lekért oldal azonosítója

```
$_SERVER[”HTTP_REFERER”]
```

a szerverhez kapcsolódó böngésző fajtája, az operációs rendszer fajtája

```
$_SERVER[”HTTP_USER_AGENT”]
```

a WEB oldalnak átadott paramétersztring (a.php?A=proba)

```
$_SERVER[”QUERY_STRING”]
```

Az aktuálisan futtatott PHP oldal azonosítója:

```
$_SERVER["PATH_TRANSLATED"]
```

A szerveren lévő WEB szerver gyökérkönyvtára

```
$_SERVER["DOCUMENT_ROOT"]
```

Az alábbiakban egy példát mutatok be a fenti változók használatából:

A pár sort beszúrva egy script elejére el őűllítja konstans formában az aktuálisan futó script elérési útvonalát, továbbá a WEB szerver dokumentumainak elérési útvonalát.

```
$path=dirname($_SERVER["PATH_TRANSLATED"]);
DEFINE("PATH",$path);
DEFINE("DOCROOT", $_SERVER["DOCUMENT_ROOT"]);
```

7.5 Konstansok

A PHP-ben vannak előre definiált konstansok, továbbá mi is definiálhatunk a

```
define(nev, érték) parancs segítségével.
```

Az előre definiált konstansok közül néhány:

`__FILE__` az éppen futtatott file neve

`__LINE__` az éppen futó programsor

`PHP_VERSION` a futtatott rendszer verziószáma

`PHP_OS` a futtató operációs rendszer

`TRUE, FALSE` logikai értékek

8 Alapvet ő utasítások

Aki a Pascal nyelven nevelkedett megszokta, hogy a definiciós és a végrehajtható utasításoknak kötött sorrendje van. A C nyelv és többek között a PHP tartalmaz el őírásokat az utasítások sorrendjére, de azok nem annyira kötöttek, ezért aztán nagyon könny ű áttekinthetetlen és rosszul m űködő programot írni bennük. Ahhoz, hogy gyorsan tudjunk egy programot írni, ismertetünk néhány utasítást.

8.1 Echo

A PHP egyik leggyakrabban használt utasítása az **echo**. Segítségével a böngész őbe ki lehet írni a program futásának eredményét. A kiíratás során gondolnunk kell arra, hogy az eredmény egy HTML kód mindig, amit majd a böngésző tovább értelmez.

Az egymás után ír echo parancsok az eredményt egymás után írják ki a kimenetre, pozicionálás, visszalépés a szövegben nem lehetséges. Ha a kiírás után új sorban szeretnénk kezdeni a kiírást, akkor a HTML szerint egy
 tag-et (soremelés) vagy kell kiírni. Az alábbiakban több soros kiírást alkalmazunk:

```
<?PHP
 echo ”Szevasz tavasz<BR>”;
 echo ”Mit sütsz kis szücs?<BR>”;
?>
```

A numerikus eredmények kiírására is használhatjuk ezt az utasítást, azonban formátumozni nem tudjuk a kimenetet így.

```
<?PHP
 $a = 5;
 $b = 6;
 echo $a + $b;
 echo ”Mit sütsz kis szücs?<BR>”;
?>
```

További probléma, hogy mi van akkor, ha numerikus információt és string-et akarunk egy szövegben kiírni. A feladat megoldáshoz ugyanazt kell használnunk, mint amikor két string-et akarunk összefűzve kiírni.

```
<?PHP
 $a = 5;
 $b = 6;
 echo ”Az eredmény: ”.$a + $b;
 $sz = ”Tán sós húst sütsz kis sz    űcs?”;
 echo ”Mit sütsz kis szücs?”.$sz.”<BR>”;
?>
```

A fenti példában az echo parancs az összeadás m űveletét string-gé konvertálta és így íratta ki. A konverzió teljesen automatikus.

További érdekes lehetőség, amikor egy string-ben szeretnénk kiíratni egy változó értékét:

```
<?php
 $o = 5 + 6;
 Echo ”Az eredmény: $o<BR>”;
?>
```

Amint látjuk, a korábbi megfontolások alapján a soremelést a HTML szerint kell használnunk. Az alábbiakban néhány gyakran használt tag-et írunk le. A használható tag-ek tárát egy HTML kódolással foglalkozó könyvb ől, jegyzetből vagy Internet helyről meg lehet tudni.

A képernyőn való soremelésre a **
** tag szolgál.

Vízszintes vonal írására **<HR>**

Paragrafus eleje, vége: **<P> ....</P>**

Vastag betű **<B> ....</B>**

Dőlt betű **<I> ....</I>**

Táblázat létrehozására az alábbi példa egy soros, két oszlopos táblázatot hoz létre)

```
<table>
  <tr>
    <td>  <P> első oszlop</p> </td>
    <td>  <P> masodik oszlop</p> </td>
  </tr>
</table>
```

Mivel a fenti TAG-ek sztringek, ezért célszerűen az echo paranccsal kell kiiratnunk őket.

Az echo egy nyelvi elem és nem függvény, azaz valami olyasmi, mintha C-ben egy függvénymakrót hoznánk létre. **Éppen ezért bonyolultabb kifejezések kiiratásához alkalmatlan!**

Ha az adatokat formázottan szeretnénk kiírni, akkor a **printf()** függvényt kell használni

8.2 Formázott kiírás

`print()` – Ugyanaz, mint az Echo, csak a szintaktika kissé más.

`printf()` – kiiratás formázottan

`sprintf()` – formázott sztringet ad vissza az alábbi szintaktika szerint:

```
string sprintf (string formátum [, mixed paraméterek...])
```

A formátum szerint megadott karaktersorozattal tér vissza. A formátumkarakterek lényegében a C-ben megszokott formátumkarakterek.

A formátumstring több direktívát tartalmazhat. A % string vezeti be a direktívákat, majd utána következnek a formázó karakterek. Ezeken kívül minden karakter megjelenik a kimeneten. A kimenet konverziós parancsait a printf() és az sprintf() parancsban ugyanúgy lehet használni.

A konverziós parancssorozat az alábbi parancsokat tartalmazza:

Opcionális kitöltő karakter. Ezzel lehet a stringet megfelelő méretre kitölteni. Default értéke a szóköz. Ezen kívül lehet a 0, vagy egyéb karakter.

Igazítás karakter. Az eredmény balra vagy jobbra igazított lesz. default jobbraigazítás; a – karakter igazítja balra.

Szélesség meghatározó. Megmondja, hogy minimum hány karakter legyen az eredményben.

A tizedes jegyek száma. Csak a double formátum esetén hatásos. (A számformátumokat **number_format()** függvénnyel tudjuk még jól kezelni.)

A *típusmeghatározó* megmondja, hogy milyen típusú adatokat kell kezelnie. Lehetőségek:

`%` - a % jel.

`b` – az argmentum integer, és bináris számként jelenítjük meg

`c` – az argumentum integer, ASCII kódként jelenítjük meg.

`d` – az argumentum integer, decimális számként jelenítjük meg.

`f` - the argumentum double és lebegőpontos számként jelenítjük meg.

`o` - az argumentum integer, és oktális számként jelenítjük meg.

`s` – az argumentum string és így is jelenítjük meg.

`x` - az argumentum integer és hexadecimális számként jelenítjük meg (kisbetűvel)

`X` - az argumentum integer és hexadecimális számként jelenítjük meg (nagybetűvel)

```
$isodate = sprintf ("%04d-%02d-%02d", $ev, $ho, $nap);
$money1 = 68.75;
```

```
$money2 = 54.35;
$money = $money1 + $money2;              // Az echo $money kimenete "123.1" lesz
$formatted = sprintf ("%01.2f", $money); // Az echo $formatted kimenete "123.10"
echo $money;
echo $formatted;
```

9 Operátorok (m űveletek)

A következőkben megismerjük, hogy milyen műveleteket végezhetünk a különböző adattípusokkal.

9.1 Stringek közötti műveletek

Stringek összefűzése: **.**

```
$a = ”alma”.”körte” ;
echo $a;  //eredménye ”almakörte” lesz
```

Stringek hozzáadása meglévő sztringhez: **.=**

```
$a = ”meleg”;
$a .= ”víz”;
$echo $a; // eredmény: melegvíz
```

9.2 Aritmetikai műveletek

A numerikus értékek összeadására ugyanolyan operátorokat használunk, mint más nyelvekben.

```
echo $a + $b; //$a és $b összege
echo $a - $b; //$a és $b különbsége
echo $a * $b; //$a és $b szorzata
echo $a / $b; //$a és $b hányadosa (egész, ha $a és $b egészek és a hányados egész)
echo $a % $b;  //Modulus $a / $b maradéka
```

9.3 Hozzárendelés, értékadás

Az operátor az "=". Ez ugyanazt jelenti, mint Pascalban a := vagy C-ben az =. A bal oldal értéke legyen az, ami a jobb oldalé. A hozzárendelő kifejezésnek az értéke a bal oldalhoz rendelt érték.

```
$a = ($b = 4) + 5; // $a most 9, és $b 4
```

9.4 Növelő/csökkentő operátorok

A PHP támogatja a C-ben megismert inkrementáló és dekrementáló operátorokat. Az alábbiakban megismerjük azokat, majd példát látunk rájuk: Amikor az operátor a változó el őtt van, akkor a kiértékelés során el őször növekszik a változó értéke, majd értékeli ki a rendszer, míg a változó mögötti operátor esetén el őször kiértékeli a változót a rendszer, majd növeli vagy csökkenti az értékét!

```
<?php
$a = 33;
echo ++$a;  // Növeli $a-t eggyel, majd visszaadja $a értékét
echo $a++;  // Visszaadja $a értékét, majd növeli $a-t eggyel
echo --$a; // Csökkenti $a-t eggyel, majd visszaadja $a értékét
echo $a--; // Visszaadja $a értékét, majd csökkenti $a-t eggyel
?>
```

Itt egy másik példaprogram:

```
<?php
echo "<h3>Postinkrementálás</h3>";
$a = 5;
echo "5-nek kell lennie: " . $a++ . "<br>\n";
echo "6-nak kell lennie: " . $a . "<br>\n";

echo "<h3>Preinkrementálás</h3>";
$a = 5;
echo "6-nak kell lennie: " . ++$a . "<br>\n";
echo "6-nak kell lennie: " . $a . "<br>\n";

echo "<h3>Postdekrementálás</h3>";
$a = 5;
echo "5-nek kell lennie: " . $a-- . "<br>\n";
echo "4-nek kell lennie: " . $a . "<br>\n";
```

```
echo "<h3>Predekrementálás</h3>";
$a = 5;
echo "4-nek kell lennie: " . --$a . "<br>\n";
echo "4-nek kell lennie: " . $a . "<br>\n";
?>
```

9.5 Logikai operátorok

A logikai műveletek minden programozási nyelvben hasonlóan néznek ki. Az alábbi lehetőségek vannak. A PHP-ben az igaz értéket mindig az 1 és a hamis értéket a 0 hordozza. Ennek a tudásnak a birtokában ugyanakkor nem célszerű a 0 és 1 értékeket numerikusan használni.

```
<?php
$a = TRUE;
$b = FALSE;
echo $a and $b;  //És   Csak akkor igaz, ha mind $a mind $b igazak
echo $a or $b;  //Vagy  Akkor igaz, ha $a és $b között van igaz
echo $a xor $b;  //Kizáró vagy Akkor igaz, ha $a és $b közül pontosan egy igaz
echo ! $a;  //Tagadás  Igaz, ha $a nem igaz
echo $a && $b;  //És   Csak akkor igaz, ha mind $a mind $b igazak
echo $a || $b;  //Vagy  Akkor igaz, ha $a és $b között van igaz
?>
```

9.6 Összehasonlító operátorok

Az összehasonlító operátorok, mint nevük is sugallja, két érték összehasonlítására szolgálnak. Az eredmény igaz, vagy hamis lehet!

```
Echo $a == $b; //Egyenlő Igaz, ha $a és $b értéke egyenlő
<?php
$a = ”alma”;
$b = ”körte”;
echo $a === $b; //Azonos     Igaz, ha $a és $b értéke egyenlő, és azonos típusúak
   //     (csak PHP 4)
echo $a != $b; //Nem egyenlő   Igaz, ha $a és $b értékei különbözők
echo $a !== $b; //Nem azonos   Igaz, ha $a és $b értékei vagy típusai különbözők
   //     (csak PHP 4)
echo $a < $b; //Kisebb mint   Igaz, ha $a szigorúan kisebb, mint $b
echo $a > $b; //Nagyobb mint   Igaz, ha $a szigorúan nagyobb, mint $b
echo $a <= $b; //Kisebb, vagy egyenlő Igaz, ha $a kisebb, vagy egyenlő, mint $b
echo $a >= $b; //Nagyobb, vagy egyenlő Igaz, ha $a nagyobb, vagy egyenlő, mint $b
?>
```

Feltételes operátor a "?:", ami úgy működik, mint a C-ben és sok más nyelvben.

```
(kif1) ? (kif2) : (kif3);
```

A kifejezés *kif2*-t értékeli ki, ha *kif1* igaznak bizonyul, és *kif3*-at, ha *kif1* hamis.

9.7 Bitorientált operátorok

A bitorientált operátorok teszik lehetővé, hogy egy egész érték bizonyos bitjeit beállítsuk, vagy kimaszkoljuk.

```
<?php
$a = 126;
$b = 3 ;
echo $a & $b; //És Azon helyeken, ahol mind $a-ban, mind $b-ben '1' volt, az eredményben
  // '1' lesz, egyébként '0'.
echo $a | $b; //Vagy Ott lesz '1' az eredmény, ahol vagy $a-ban, vagy $b-ben '1' állt.
echo $a ^ $b; //Kizáró vagy Ott lesz '1', ahol vagy $a-ban, vagy $b-ben '1' áll, de csak az
  // egyikben.
echo ~ $a; //Nem $a összes bitjét invertálja
echo $a << $b; //Eltolás balra  $a bitjeit $b-vel balra tolja (minden tolás 2-vel való szorzást
  // jelent [amíg el nem fogynak a bitek. A legfelső helyiérték az
  // elő jelbit.)
echo $a >> $b //Eltolás jobbra  $a bitjeit $b-vel jobbra tolja (minden tolás 2-vel való
  // [egész!]osztást jelent. Mivel a legfelső bit az előjelbit, negatív szám
  // jobbra tolása fura eredményre vezet!)
?>
```

9.8 Hibakezelő operátorok

A PHP egy hibakezelő operátort támogat, az at (kukac) jelet (@). Ha egy PHP kifejezés elé írod, 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 ellen őrizd minél hamarabb a kifejezést követően ha használható információt szeretnél kapni.

```
<?php
 /* Szándékos SQL hiba (plusz idéz    őjel a táblanévnél): */
 $res = @mysql_query ("select nev, kod from 'nevlista") or
 die ("A lekérés sikertelen volt. A hiba: $php_errormsg");
?>
```

9.9 Végrehajtó operátorok

A PHP-segítségével utasításokat hajthatok végre az operációs rendszeren. A jel a visszaidéz   őjel ``. Ha közéjük írok egy parancsot az operációs rendszer részére, akkor az megpróbálja végrehajtani és egy változónak átadni az eredményt.

[Az alábbi kis példa az aktuális könyvtár tartalmát (hosszú lista, rejtett fájlok is) formázva írja ki (illetve fix szélességű betűket használva, entereket tiszteletben tartva)]

```
<?php
 $output = `dir C:\`;
 echo "<pre>$output</pre>";
?>
```

Ebben a témában az alábbi függvényeket érdemes még megtekinteni a doksiból: **system()**, **passthru()**, **exec()**, **popen()**, és **escapeshellcmd()**.

10 Vezérlési szerkezetek

A PHP-ben ugyanúgy, mint más programozási nyelvekben az utasítások végrehajtásának sorrendje alapvet ően fentről lefelé. Ily módon csak szekvenciális programokat lehet írni, azonban rövid tanukás után szükségessé válik elágazásokat és ciklusokat tartalmazó programok írása is. A vezérlési szerkezetek, mint mindig itt is a C-hez hasonlóak.

A program a különböz ő irányokba való továbbhaladását általában egy kifejezés határozza meg. Ennek a kifejezésnek az értéke logikai alapvet ően, azonban a PHP hasonlóképpen, mint a C nem különböztet meg külön logikai értékeket, hanem a 0 és a nem 0 értékek jelentik azt. Ily módon, ha egy numerikus kifejezés 0, akkor hamis, és ha nem 0, akkor igaznak tekinthető. Ha egy string kifejezés üres, akkor hamis, ha van értéke, akkor igaz. Az olyan változók, amelyeket még azel őtt értékelünk ki, hogy értéket kaptak volna (ez nem helyes), a NULL értékkel bírnak.

10.1 Elágazások

Az első fontos lehetőség a feltételhez kötött végrehajtás. Ha bármelyik ágon több utasítást akarunk végrehajtatni, akkor szintén a C szintaktika szerint { ...} jelpárost kell használnunk

If(utasítás) ...

```
if(kifejezés) utasítás;
```

```
if(kifejezés) {
 utasítás1;
 utasitas2;
 .....
}
```

If .... else....

Ha a kifejezés igaz, akkor az utasítás1 különben az utsítás2 hajtódik végre.

```
if( kifejezés )  utasítás1;
else    utasítás2;
```

Ha a kifejezés igaz, akkor az utasítás1 ág hajtódik végre, különben a másik

```
if( kifejezés ) {
 utasítás1;
 utasitas2;
 .....
}else{
 utasítás3;
 utasitas4;
 .....
}
```

If ... elseif ... else ...

Ha kettőnél több elégazást szeretnénk, akkor az alábbi szintaktikával tudjuk a kérdést megoldani:

```
if( kifejezés1 )  utasítás1;
elseif(kifejezés2) utasítás2;
else               utasítás3;
```

vagy

```
if( kifejezés1 ) {
 utasítás1;
 utasitas2;
.....
}elseif( kifejezés2 ){
 utasítás3;
```

```
 utasitas4;
 .....
}else {
 utasítás5;
 utasítás6;
 ......
}
```

Switch(kifejezés)

A fenti esetben az if és az elseif utasításnál lév ő kifejezések tetsz őlegesek lehetnek, és az így felírt vezérlési szerkezettel meglehetősen bonyolult elágazásokat lehet létrehozni. Ezzel szemben általánosabb eset, amikor egy változ értékét ől függ ően akarunk több féle m űveletet is végrehajtani. Erre a célra alkalmas a C-b ől jól ismert switch() függvény, amely több irányú elágazást hajt végre. Az ágak meghatározásakor csak konstansokat használhatunk, és az ágra akkor adódik át a vezérlés, ha a switch függvényben lév ő változó értéke pontosan a konstans értékével egyezik meg. Amennyiben egy ágra ráadtuk vezérlést, majd végrehajtottuk az ott definiált utasításokat, a switch szerkezet végére kell ugranunk a `break` utasítás segítségével, mivel különben rácsorognánk a következő case feltételvizsgálatokra.

Nézzük a szintaktikáját:

```
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 fenti példában az $i változó értékét ől függően léptünk valamelyik irányba. Ha a változó értéke nem vette fel sem a 0, 1 vagy 2 értékeket, akkor a default utáni parancs hajtódik végre.

Gyakori eset, hogy amikor több különböz ő eseményt egyetlen változó különböz ő értékei alapján akarsz végrehajtatni, akkor használod a switch utasítást. El őfordulhat, hogy ugyanazt az eseményt több érték is jelölheti, illetve több különböző esemény van. Az alábbi példában a $jelz ő változó 0, 1, 2 értéke esetén az fv1() függvény fut le, a 3-as értékre az fv2() függvény, egyéb esetekben az fv3().

```
switch ($i) {
    case 0:
    case 1:
    case 2:
        fv1($jelzo);
        break;
    case 3:
        fv2($jelzo);
        break;
    default:
 fv3($jelzo);
}
```

A fenti példában az fv1, fv2, fv3 függvények lehetnek akármik!

10.2 Ciklusok

Mint a C-ben, itt is hasonló ciklusszerkezetek léteznek.

While (kifejezés)

Itt a kifejezés a ciklus előtt értékelődik ki, azaz a ciklus magja nem biztos, hogy végrehajtódik.

```
While (kifejezés) utasítás; While (kifejezés) {
                                    Utasítás1;
                                    Utasítás2;
                                    ...
                                }
```

Példa:

```
<?php
 $i = 1;
 while ($i <= 10) {
      print $i++;   // a kiírt érték $i, a kiírás után n      ő $i értéke
 }
 $i = 1;    // Az el             őző ciklus másképpen, de ugyanazzal az eremdénnyel
 while ($i <= 10) {
      print $i;   // a kiírt érték $i, a kiírás után n       ő $i értéke
      $i++;
  }
?>
```

Do ..... While(kifejezés)

A kifejezés a ciklus végén értékelődik ki, azaz a ciklus magja egyszer mindenképpen lefut.

```
Do                                  Do{
 utasítás                                Utasítás1;
while(kifejezés);                        Utasítás2;
                                         ...
                                    }while(kifejezés);
```

Példa:

```
<?php
$i = 0;
do {
   print $i;
} while ($i>0);
?>
```

For(inicializáló kifejezés; Benntmaradás kifejezése; iteráló kifejezés)

A ciklus elején egyszer fut le az `inicializáló kifejezés`. Általában ez egy változónak ad egy kezd őértéket. Az ilyen változót ciklusváltozónak hívjuk.

A `Benntmaradás kifejezése` mindannyiszor a ciklus magjának lefutása el őtt értékeli ki a rendszer. Ha a kifejezés igaz értéket ad vissza, akkor lefut a ciklus magja, ha hamis értéket, akkor a ciklus utáni els ő utasításon folytatódik a futás. Ha a kifejezés helyére üres értéket írunk, akkor végtelen ciklusba kerülhetünk, hacsak a ciklus belsejéből nem ugrunk ki a break utasítással.

Az `Iteráló kifejezés` felelős azért, hogy a ciklus valamikor befejez ődjön. Általában a ciklusváltozó értékét növeljük vagy csökkentjük eggyel.

```
For (init;bentmaradás;iterácio)     For (init;bentmaradás;iterácio){
utasítás;                            Utasítás1;
                                     Utasítás2
                                     ......
                                    }
```

Példa:

```
<?php
// Írassuk ki az egész számokat visszafelé 100-tól 1-ig
```

```
 for ($i = 100; $i >0; $i--) {
     print $i;
 }

//Írassuk ki egy két dimenziós tömb elemit soronként
$a= array( array(3,5,4,6),array(12,234,12,12),array(3,6,5,77));
for ($i =1; $i<= 0; $i--) {
 for (j=1;j<=4;j++){
  print $a[$i][$j].” ”;
 }
 print ”<BR>”;
 }
?>
```

Foreach (tömb_kifejezés as $érték) vagy Foreach (tömb_kifejezés as $kulcs => $érték)

Ez a fajta ciklus a Perl nyelvből került át a PHP-be.

Ezt a ciklust arra használhatjuk, hogy egy ciklussal végigmenjünk egy tömb minden egyes elemén. Rendkívül jól használható adatbázis lekérdezések vagy egyéb tömbben visszaadott adatok feldolgozása során. Kétféle

```
Foreach ($tömbváltozó as $ertek)
utasítás;
```

```
Foreach ($tömbváltozó as $ertek){
      Utasítás1;
      Utasítás2
      ...
}
```

Példa:

```
$tomb = array (1, 2, 3, 17);

foreach ($tomb as $ertek) {
   print "Az aktuális értéke \$tomb-nek: $ertek.\n";
}
```

A második formában a $kulcs változó megkapja a tömb éppen aktuális indexét, ezáltal egyes esetekben könnyebb a feldolgozása

```
Foreach($tömbváltozó as $kulcs =>
$ertek) utasítás;
```

```
Foreach ($tömbváltozó as $kulcs => $ertek){
      Utasítás1;
      Utasítás2
}
```

Példa:

```
* harmadik foreach példa: kulcs és érték */

$tomb = array ( "egy" => 1, "kettő" => 2, "három" => 3, "tizenhét" => 17 );

foreach ($tomb as $kulcs => $ertek) {
    print "\$tomb[$kulcs] => $ertek.\n";
}
```

break vagy break n

A break utasítás arra szolgál, hogy segítségével egy struktúrából az adott helyen ki tudjunk ugrani. Ha utána írunk egy számot, akkor annak alapján több egymásba ágyazott struktúrából is ki tud ugrani:

Példának okáért korábban a switch utasításnál mutattunk egy példát a break használatára, az opcionális módra pedig itt van egy példa:

```
$i = 0;
while ($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 vagy continue n

Bár programozási módszertanok kerülik a ciklusokból való kiugrálást és a ciklusmagon belüli iterációt, azért minden általános nyelvben benne van a lehetőség, beleértve a Pascalt és a C-t is. Ez az utasítás akkor használható, ha a ciklus belsejében már eld őlnek a további iterációra vonatkozó feltételek és nem akarjuk, hogy a ciklus magjának többi részét feleslegesen futtassuk. Ha a continue n formát használjuk, akkor több egymásba ágyazott struktúrát tudunk folytatni. Az alábbi példában 100 db véletlenszámot hozunk létre a 0..4 egész tartományból és az eloszlásukat vizsgáljuk. A continue utasítások hatására ha megvan az érték, további feltételeket nem értékel ki a program, hanem rögtön iterálja a for ciklust.

```
<?php
$n = 100;
srand (double) microtime() * 1000000);
$a1=$a2=$a3=$a4=$a5=0;
for($i=1;$i < $n;$i++){
 $veletlen= rand(0,4);
 switch($veletlen){
  case 0: $a1++;
   continue;
  case 1: $a2++;
   continue;
  case 2: $a3++;
   continue;
  case 3: $a4++;
   continue;
  default:
   $a5++;
 }
}
?>
```

10.3 Elágazások és ciklusok használata HTML kóddal keverve

Gyakori, hogy a HTML oldalon két különböző kódot szeretnénk megjeleníteni, attól függően, hogy éppen melyik feltétel igaz, ugyanakkor a kiírandó HTML részt túl bonyolultan állíthatjuk csak el ő PHP-ben. Ebben az esetben alkalmazni lehet az alábbi példához hasonlót:

```
<?php if ( kifejezés) { echo ”elso ág”; ?>
 <table>
  <tr>
       <td>  <P> Szevasz tavasz, a kifejezés igaz</p> </td>
       <td>  <P> valamit visz a víz</p> </td>
  </tr>
  .......
 </table>
<?php } else {  echo ”Másik oldal”; ?>
 <p> Ez a rész itt egyszer    űbb lett!</p>
<?php
}
?>
```

Az igaz kifejezés esetén az igaz ágban létrejövő egy soros táblának két oszlopa lesz, míg a hamis érték esetén csak a második szakaszban lév ő kiírás jelenik meg, mivel a PHP amikor megszakad a PHP kód – itt többször is megszakad -, akkor a szervernek változatlanul adja vissza a HTML kódot.

Sajnos az így megírt program kissé áttekinthetetlen.

10.4 PHP lapok beszúrása, ”makro”-k használata.

Gyakran megfogalmazott feladat, hogy egy több lapból álló PHP programban minden PHP oldal elején fussanak le ugyanazok a beállítások, ugyanazok az inicializáló paraméterek, illetve ugyanazokat a függvényeket és lehetőségeket használjuk minden PHP scriptben. Valami olyasmire gondolok itt, mint Pascal-ban a unit fogalma vagy C-ben a header file-ok fogalma.

Erre a célra két parancs áll rendelkezésre, amelyeknek a működése nem pontosan ugyanaz, de nagyon hasonló. Mind a két esetben arról van szó, hogy a PHP script egy adott pontján behívunk egy file-t, amelynek a kódja futásidőben bekerül a PHP scriptbe, a szerver kiértékeli, és annak megfelelően jár el. A file tartalma tehát a futás idejére úgy válik a PHP script részévé, mintha eleve abba beírtuk volna.

Mivel a Webszerver kezdi a kérdéses include file-t feldolgozni, ezért ilyenkor HTML módba kerül a szerver, így a PHP kód feldolgozásához az include file elején PHP nyitó tag-et kell elhelyezni és a végén zárótag-et. (<?php .....?>)

Include ()

Az include használata esetén a megadott file mindannyiszor kiértékelődik, ahányszor a vezérlés az include-ra kerül. Ennek eredményeként, ha egy ciklus belsejében megfelelően helyezzük el a file-t, akkor a ciklus értékétől függően mindig más és más file-t szerkesztünk be, mint az alábbi példában láthatjuk:

```
$fileok = array ('elso.php', 'masodik.php', 'harmadik.php');
for ($i = 0; $i < count($fileok); $i++) {
    include ($files[$i]);
}
```

require()

A require parancs akkor is beolvassa a célfile-t, ha soha nem hajtódik végre, és ha hiányzik a file, akkor a program futása megszakad.

A fentiek miatt require parancsot csak konstans file-nevekkel szabad használni.

```
<?php
 require ("file.php");
?>
```

include_once()

Az utasítás hasonlít az include() utasításra, azzal a különbséggel, hogy csak egyszer kerül bele a futáskor a kérdéses file. Ez azért fontos, mert ha az include() utasítással többször is meghívjuk futás közben ugyanazt a file-t, akkor a benne lévő globális változók és függvények többször kerülnek a kódba, ami miatt a futás hibaüzenettel leáll, hiszen ugyanaz az azonosító más és mást jelöl. Az include_once() segítségével ezt a hibaforrást kiküszöbölhetjük.

require_once()

Az utasítás hasonlóan az include_once() utasításhoz csak egyszer hívja be a kérdéses file-t futás közben, így a függvények és globálisváltozók újradefiniálása körüli problémák kiküszöbölhetőek. Egyébként a require() utasítással megegyező a szerepe és működése.

10.5 Távoli file-ok hívása

A fenti két utasítás – include és require – alkalmas arra, hogy a PHP megfelelő konfigurálása esetén, akár másik Web szerveren is elérjünk távoli file-okat. Ehhez az "URL fopen wrapper"-eket kell bekapcsolni, a PHP4.0.3-tól kezdődően az *allow_url_fopen* php.ini beállításával. Ez a lehetőség Windows rendszereken nem működik.

Ennek a tulajdonságnak persze vannak veszélyei is. Ha egy ilyen távoli hívás során olyan lapot hívunk meg, aminek a tartalmára nincsen hatásunk, akkor az include-olt file tartalma lehet olyan, hogy a mi rendszerünkre veszélyeket hordoz. A távoli helyen megfelelően elkészített lap email-ben elküldheti az azonosítókat és egyéb olyan adatokat, amelyeket nem szeretnénk nyilvánosságra hozni, ezért ezzel a lehetőséggel óvatosan kell bánni.

11 Saját függvények, változók élettartama és láthatósága

Eddig sok szó esett a változókról és esetenként a függvényekr ől is, azonban az eddigiek alapján azt gondolná az ember, hogy csak ilyen egyszerű programok írhatók PHP-ben. Az igazság az, hogy PHP-ben csak a futási idő és a futtató rendszer memóriája szab határt az alkalmazott program bonyolultságának.

11.1 Függvények

A PHP-ban is használhatunk programstruktúrákat, programszegmenseket. Már Pascal-ból vagy C-b ől is jól ismerhetjük az alapvető két eljárástípust, amit Pascalban Procedure illetve Function névvel illetünk, C-ben pedig típus nélküli illetve típusos function-nak mondunk. Mind a két nyelven a különbség az, hogy ad-e vissza a kérdéses eljárástípus értéket, vagy nem.

A PHP-ben csakúgy, mint a C-ben kétféle eljárástípust használhatunk. A szintaktika a következő:

Ez az eljárástípus nem ad vissza értéket.

```
function függvény_név(paraméterlista) {
 A függvény törzse;
}
```

Ez az eljárástípus a definiáltnak megfelelő típusú értéket ad vissza:

```
function függvény_név(paraméterlista) {
     A függvény törzse;
     return érték;
}
```

Látható, hogy a visszatérési érték léte vagy nem léte a programozótól függ, ezért egyes esetekben célszer ű a visszatérési érték típusát megállapítani ahhoz, hogy van-e egyáltalán visszatérési érték.

Rekurzió lehetséges a függvényhívásoknál, de természetesen arra kell vigyázni, hogy a rekurzió véget érjen. Az alábbi kis példa az N faktoriális kiszámítását végzi rekurzív módon.

```
<?php
function nfakt($n){
    if ($n>1) $nf = $n*nfakt($n-1);
    else $nf = 1;

    echo $n." => ".$nf."<BR>";
    return $nf;
}
nfakt(200);
?>
```

11.2 Paraméterátadás

Egy függvény definiálásakor meg kell határozni, hogy milyen paramétereket vegyen át. Ezt a formális paraméterlistával tudjuk megtenni, amelyben az átvevő változók neveit vesszővel elválasztva fel kell sorolni.

A híváskor nem kell minden paramétert átadni. Ebben az esetben a kérdéses paramétereknek a függvényen belül nem lesz értéke. Az empty() függvénnyel lehet megvizsgálni, hogy egy paraméter kapott-e értéket vagy sem vagy a függvény formális paraméterlistájában default értéket kell adni a paraméternek.

A PHP-ben kétféle paraméterátadás, létezik

Érték szerinti

A hívó kifejezés értéke behelyettesít ődik a meghívott függvény paraméterlistájában szerepl ő változóba és a függvényen belül a paraméter használható. Ez az alapértelmezés.

Cím szerinti

Ez azt jelenti, hogy a változó memóriacímét adjuk át a függvénynek, amely a cím ismeretében tudja azt módosítani és a függvényből való visszatéréskor a változások megmaradnak.

Ez utóbbi esetben, ha cím szerint akarunk átvenni értéket egy függvénnyel, akkor használnunk kell a **&** operátort.

```
<?php
function fgv(&$n) {
    $n *= 2;
```

```
}

$n=100;
echo ”$n<BR>”;
fgv($n);
echo ”$n<BR>”;
?>
```

A függvények paramétereinek átvétele még egy módon megtörténhet. A PHP támogatja a paraméterek kezdőértékadását. Ennek akkor van értelme, ha a függvény hívásakor nem adtunk át értéket a függvénynek. Ilyenkor alapértelmezett értéket kap az a paraméter, amelyet a hívó nem adott át. Természetesen az át nem adott paramétereknek a helye az átadottak után helyezkedjen el!

```
<?php
function joghurtot_keszit ($flavour, $type = "acidophilus") {
    return "Készítek egy köcsög $flavour ízű $type-ot.\n";
}

echo joghurtot_keszit ("eper");
?>
```

Ha azt szeretnénk, hogy egy függvény változó számú paramétert vegyen át és a függvényben meg akarjuk állapítani, hogy éppen most hány paraméter van, akkor a következő függvényeket használni:

func_get_args() - Egy tömbben visszaadja az átadott paraméterek listáját.

func_num_arg() - Megadja a kapott függvények számát

func_get_arg(sorszám) - Visszaadja a sorszám paraméterrel megadott paramétert. Ha a sorszám nagyobb, mint a paraméterlista utolsó elemének indexe, akkor hibaüzenet jön (warning). A paraméterlista indexe 0-val kezdődik.

Az alábbi példa ezeket példázza.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "paraméterek száma: $numargs<br>\n";
    if ($numargs >= 2) {
      echo "A második paraméter: " . func_get_arg (1) . "<br>\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $numargs; $i++) {
     echo "$i-ik paraméter: " . $arg_list[$i] . "<br>\n";
    }
}
foo (1, 2, 3);
?>
```

11.3 Függvények visszatérési értéke

Egy függvény tetsz őleges típust, még tömböt, vagy változóreferenciát is vissza tud adni. Több érték visszaadására a tömböt használhatjuk.

11.4 Változók élettartalma és láthatósága

A PHP-ban nagyon egyszerű szabályok vannak a változók láthatóságára és élettartamára vonatkozólag.

Élettartam

A változó akkor jön létre, amikor létrehozzuk, értéket adunk neki.

Egy változó megsemmisül, ha az unset(változónév) paranccsal felszabadítjuk a változó nevét (memóriaterületét is), vagy véget ér az adott PHP script oldal. Ez alól kivétel, ha a változókat átadjuk egy másik PHP oldalnak a POST, GET metódussal, a cookie-k vagy sessionok használatával.

Egy PHP minden részén használhatók a $_POST, $_GET, $_SESSION, $_COOKIE, $_ENV, $_SERVER, $GLOBALS tömb elemei. Ezek a szuperglobális változók.

Létezik-e a változó

Azt, hogy egy változó létezik-e az isset() függvénnyel kérdezhetjük le. Igazat ad vissza, ha a változó létezik, az empty() függvény pedig igazat ad vissza, ha a változó értéke 0, üres, vagy nem létezik!

```
<?PHP
 if (!isset($_SESSION[”logged_in”])) die(”Jelentkezz be!”);
 print(”Beléptél!”);
?>
```

Láthatóság

A PHP oldalon létrejött és a különböz ő módokon átvett változók globálisak, azaz attól a helyt ől kezdve láthatók mindenhonnan, azonban ha meghívunk egy függvényt, akkor abban a függvényben csak azok a változók láthatók, amelyeket a függvényben hoztunk létre, vagy paraméterként adtunk át. Ha a függvényb ől kilépünk, akkor ezek a változók megsemmisülnek kivéve, ha nem cím szerinti paraméterátadás során jöttek létre. Azaz ezek a változók lokálisak lesznek a függvényre nézve.

Ha egy függvényből új függvényt hívunk, akkor abban nem lehet látni a hívó függvény változóit.

A fentiek alól az egyetlen kivétel, ha a függvényben használjuk a global parancsot, aminek segítségével importálhatjuk a script globális változóit a függvénybe.

```
<?php
 $a = 1;
 $b = 2;

 Function Osszead () {
      global $a, $b;
  $b = $a + $b;
 }
 Ossszead ();
 echo $b;
?>
```

A másik lehet őség az, hogy használjuk a `$GLOBALS[]` asszociatív tömb értékeit, amelyben minden bejegyzett globális változó megtalálható.

```
<?php
 $b = 1;
 $a = 10;
 $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
 echo $b;
?>
```

Létrehozhatunk statikus változókat is.

Statikus változó egy függvényben jöhet létre. Amikor kilépünk a függvényb ől már nem használhatjuk ennek a változónak az értékét, de ha újra meghívjuk a függvényt, nini még megvan az el őző értéke. **J** ott folytathatjuk, ahol abbahagytuk. Ennek néha van értelme. Mindenesetre a használatához a függvényben a **static** kulcsszót kell használni. Az alábbi kis szösszenet egy ciklusból meghívja újra meg újra a Test() függvényt és ennek során az $a változó tartalma folyamatosan nő, noha mindig kilépünk a függvényből. Na ja, statikusnak deklaráltuk! A statikus változónak kezdőértéket adva, az csak egyszer fut le, amikor először meghívjuk a függvényt.

```
Function Test () {
    static $a = 0;
    echo $a;
    $a++;
}
for ($i =0; $i< 10;$i++) {
 Test();
 Echo ”blablabla”
}
```

11.5 Változók átadása lapok között

Gyakori kérdés a PHP-ben programozók között, hogyan tudnak értékeket átadni a PHP lapok között, hiszen ha egy lap lefut, akkor eddig úgy tudtuk, hogy a lapon keletkezet változók is megsz űnnek. Amikor el őször

szembekerültem a problémával, akkor azt hittem, hogy a globális változók oldják meg a problémát. Sajnos a dolog nem ennyire egyszerű, de nem is túlságosan bonyolult. Átadhatunk egyedi változókat és egy dimenziós tömböket is. Négy lehetőségünk van erre.

11.5.1 Header utasítás

A **Header** utasítást csak akkor használhatjuk, ha az adott oldalon még semmiféle képernyőre írás nem volt, azaz a keletkező HTML oldal tartalmi részét még nem kezdtük írni. A header segítségével bármilyen header-t elküldhetünk. Az alábbi példában egy teljes header sorozatot írunk ki a HTML oldalra

```
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-cache, must-revalidate");  // HTTP/1.1
header ("Pragma: no-cache");                          // HTTP/1.0

header ("HTTP/1.0 404 Not Found");
```

Itt egy hibakezelést írunk ki, az URL nem található. Akkor lehet ilyet tenni, ha például az Apache szerverünk hibaüzeneteit átirányítjuk a saját oldalainkra.

```
header ("HTTP/1.0 404 Not Found");
```

Böngésző átirányítása. Itt adhatjuk meg az új oldalt. Ez a parancs nem csak a böngészőt vágja át az új oldalra, hanem a szervernek is visszaküld egy átirányítás státusz üzenetet is.

```
header ("Location: http://www.php.net"); /* Átirányítja a böngészőt a PHP web oldalra */
exit;                /* Ha nem megy az átirányítás, akkor az exit parancs biztosan kilép */
```

11.5.2 GET metódus

A GET metódust úgy használhatjuk, hogy meghívunk egy lapot az oldalunkról egy másik lapot és az URL végére paraméterként átadjuk a változókat, valahogy így, ahogy a következő példákban látjuk:

Az első példában igazából nem is PHP a megoldás, egyszerűen a <BODY> TAG-ben megadjuk, hogy melyik oldalt és menyi idő múlva hívja meg az oldal. Ennek a megoldásnak hibája, hogy tulajdonképpen itt egy Javascriptet használunk. A példában 3 másodpercig vár a betöltődés után a böngésző, majd a szerver átdobja az új oldallal és meghívja a lapot a user, pwd és a level változókkal.

```
<BODY OnLoad=timerID=setTimeout('location="index.php?user=anonym&pwd=anonymous&level=1"',30000)>
```

A következő példában hasonlót teszünk, de itt a HTML oldal fejlécében dolgozunk. Felhasználjuk a HTML meta tag-ját. Itt is 3 másodperc múlve hívja be a következő oldalt és az előző oldalról átadjuk az előző példában látott 3 változó pillanatnyi értékét.

```
<meta http-equiv="refresh" content="3;
  URL=<index.php?<?php echo 'user=$user&pwd=$pwd&level=$level' ?>">
```

A harmadik példában a PHP **header** utasítását használjuk. A példában egy POST metódussal egy űrlapon bevitt adatokat vizsgálunk meg, és amennyiben hiányzik az adat, akkor egy hibakezelő függvénybe irányítom át, ahol a header segítségével átirányítom egy másik oldalra, átadva neki a megfelelő változókat.

```
function sorry($msg,$from=1,$glob="")
{
 header("Location: sorryuser.php?from=$from&msg=$msg&glob=$glob");
}

if(empty($name)) sorry("Hiányzik a név adat! Kötelező kitölteni",1);
if(empty($loginname)) sorry("Hiányzó login név! Kötelező kitölteni",1);
if(empty($email)) sorry("Hiányó E-mail cím! Kötelező kitölteni. Itt kapod meg a jelszót!",1);
```

A fenti három lehetőség közös hibája az, hogy az átirányított lapok URL-je megjelenik a böngészőben, azaz titkos információt nem tudunk átadni, továbbá azok a böngészők, amelyek nem tűrik az átirányítást, nem fognak továbbmenni.

11.5.3 POST metódus

A POST adatátviteli metódust az űrlapokkal kapcsolatban használhatjuk legtermészetesebben. Itt egyel őre csak annyit mondunk, hogy az ürlapok olyan HTML kódok, amelyen keresztül a böngésző előtt ülő felhasználó beírhat adatokat a HTML oldalon, az űrlap SUBMIT gombjának megnyomására pedig az űrlapon definiált mez ők tartalmát, mint változóneveket és változó tartalmakat elküldi a cél oldalnak a böngész ő. Az űrlap fejlécében meg kell adni a cél oldalt (kinek küldjük) és a megfdelel ő oldal, ha az olyan oldal, amit a szerver meg tud jeleníteni betöltődik a böngészőbe.

A PHP esetén a módszer az, hogy az űrlap kitöltése után a submit gomb megnyomásával elküldjük az eredményeket egy PHP oldalnak, amely betölt ődéskor megkapja az elküldött változókat, esetleg elvégzi azokat a ferladatokat, amelyekre rendeltetett, majd megjelenít valami választ.

Ennél a módszernél, az elküldött értékek nem látható módon kerülnek el a meghívott oldalhoz, tehát ezzel a módszerrel viszonylag könnyű változóértékeket átadni.

Vigyázni kell azonban arra, hogy az adatbevitel alapvet ően string és ha nem úgy használjuk fel azokat a bevitt stringeket, hogy előtte kiszűrjük a ../../etc/ ... stb jelleg ű adatrokatés nem figyelünk arra, hogy az eredményeket a lehető legtöbb szempont szerint ellenőrizzük, akkor a web site-unk feltörhető lehet.

Az alábbi példában egy olyan HTML oldalt mutatok be, amely egy űrlapot tartalmaz, a submit gomb megnyomásának hatására az oldal önmagának (!) küldi el a változókat, majd a submit változó értéke alapján egy elágazásra kerül a végrehajtás és az eredményt elküldi e-mailben egy megadott címre. A lapon van egy kis Javascript betét is, amely az aktuális id őpont beszúrására szolgál. Az űrlapon található olyan mező is, amelynek a tartalma hidden, azaz az űrlapon nem jelenik meg.

```
<html>
<head>
<meta http-equiv="Content-Language" content="hu">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
<title>Munkalap</title>
<script language="JavaScript">

function Kitolt()
{
var x=0;
for (x=0;x<document.munkalap.lista.length;x++)
 if (document.munkalap.lista.options[x].selected)
  document.munkalap.ceg.value = document.munkalap.lista.options[x].value;
 Alert (document.munkalap.lista.options[x].value);
}

</script>
</head>
<body bgcolor="#efefef" text="#00000000" link="#6666CC" vlink="#FF9900">

<script language="php">
  $datum = date('Y.M.d H:i');  //aktuális dátum
  $mikor= date ('Y.M.d');      //
  if (!empty($pswd))   // egyszer        ű (primitiv) password ellenőrzés
     $jls = "_".$pswd."_"; // Lehetne biztonsáégosabban is, de itt most ez nem szempont
  else $jls = "__";

  // A munkalap elküldéséhez ki kell tölteni a partnercég nevét is.
  if(!empty($ceg))
  {
 $jel= (strpos($jls, "xxxx") == 0) | empty($munkavegzo);
 if(!$jel)
 {
 // Itt állítjuk össze az Email-t az átküldött változók értékéb    ől.
      $uze ="";
      $uze= $uze ."Cég                   $ceg\n";
      $uze= $uze ."Bejelentő             $bejelento\n";
      $uze= $uze ."Bejelentés időpontja  $mikor\n";
      $uze= $uze ."Hibajelenség          $hibajelenség\n";
      $uze= $uze ."A hiba oka            $hibaok\n";
      $uze= $uze ."Az elvégzett munka    $elvegzett_munka\n";
      $uze= $uze ."A munkavégzés alapja\n";
      $uze= $uze ."- Garanciális         $granciális\n";
      $uze= $uze ."- Rendszergazdai      $rendszergazda\n";
      $uze= $uze ."- Fizetős             $fizetos\n";
      $uze= $uze ."- Kiszállás           $kiszallas\n";
```

```
      $uze= $uze ."- Műhelyben           $muhely\n";
      $uze= $uze ."- Rendszergazdai      $rendszergazda\n\n";
      $uze= $uze ."A szükséges munkaidő  $munkaido\n";
   if (!empty($munkadij)) {
    $uze= $uze ."Számlázott munkadíj   $munkadij Ft + 25% ÁFA\n\n";
   }
      if (!empty($alkatreszek)) {
  $uze= $uze ."Beépített alkatrészek $alkatreszek\n";
     $uze= $uze ."Alkatrészek ára       $alkatreszar Ft + 25% ÁFA\n\n";
   }
      $uze= $uze ."\n";
      $uze= $uze ."Munkavégző            $munkavegzo\n";
      $uze= $uze ."Dátum                 $datum\n";
      $uze= $uze ."Igazolás              $igazolas\n\n";
      $uze= $uze ."A munkalapot küldő gép adatai\n";
      $uze= $uze ."A gép IP címe         ".$HTTP_ENV_VARS['HTTP_HOST']."\n";
      $uze= $uze ."A gép neve            ".$HTTP_ENV_VARS['REMOTE_HOST']."\n";
      $uze= $uze ."A gépen futó böngésző ".$HTTP_USER_AGENT."\n";

      mail("europr@matavnet.hu","munkalap",$uze);

   if(  Die("Az munkalapot elküldtük!"));
 }
 if ($jel)
  </script>
  <script language="javascript"> Alert ("Hiányosan töltötte ki a munkalapot"); </script>
 <script language="php">
   }
 </script>

<table width="77%" border="0">
  <tr>
    <td><font size="6" color="#6666CC"><b><img src="../jozsi_logo.png" width="100"
height="93"></b></font></td>
    <td><font size="6" color="#6666CC"><b>Józsi Cégének munkalapja</b></font></td>
  </tr>
</table>
<form name="munkalap" method="post" enctype="multipart/form-data" action="index.php">
  <table width="76%" border="0">
    <tr>
      <td width="21%"  valign="top">A partnercég:</td>
      <td width="79%"  valign="top">
        <input type="text" name="ceg" size="60">
        <select name="lista" onclick="Kitolt()">
          <option value="Nincs a listában"> </option>
          <option value="Emulátor KFT ">EKFT</option>
          <option value="Géza KFT">GKFT</option>
          <option value="Magyar-Uránusz Ujságírók Baráti Társasága">MUÚBNT</option>
          <option value="Magyar Tudományos Akadémia Ördögűző Intézete">MTAÖI</option>
          <option value="Alladin BT">ABT</option>
          <option value="Balogh Aladár SZKI">BA..KI</option>
          <option value="Kiss Piroska Irodalmi Múzeum">KPIM</option>
        </select>
       </td>
    </tr>
    <tr>
      <td width="21%" valign="top">A bejelentő neve:</td>
      <td width="79%"  valign="top">
        <table border="0">
          <tr>
            <td width="131">
              <input type="text"  name="bejelento"  cols ="60% ">
            </td>
            <td width="310">
   A bejelentés id            őpontja:
   <input type="text" name="mikor" value="<?php echo $mikor; ?>" >
            </td>
          </tr>
        </table>
      </td>
    </tr>
    <tr>
      <td width="21%"  valign="top">A hibajelenség:</td>
      <td width="79%"  valign="top">
        <textarea name="hibajelenseg" rows=4 cols =60 ></textarea>
```

```
      </td>
    </tr>
    <tr>
      <td width="21%" valign="top">A megállapított
        hiba</td>
      <td width="79%"  valign="top">
        <textarea name="hibaok"  rows=5 cols =60 ></textarea>
      </td>
    </tr>
    <tr>
      <td width="21%"  valign="top">Az elvégzett
        munka leírása</td>
      <td width="79%"  valign="top">
        <textarea name="elvegzett_munka"  rows="6" cols =60 >
      </textarea>
    </td>
    </tr>
    <tr>
      <td width="21%"  valign="top">
        <table width="75%" border="0">
          <tr>
            <td>Garanciális?</td>
            <td>
              <input type="checkbox" name="garancialis" >
              </td>
          </tr>
          <tr>
            <td>Rendszergazdai?</td>
            <td>
              <input type="checkbox" name="rendszergazda" >
             </td>
          </tr>
          <tr>
            <td>Fizetős?</td>
            <td>
              <input type="checkbox" name="fizetos" >
              </td>
          </tr>
          <tr>
            <td>Kiszállás</td>
            <td> <b>
              <input type="checkbox" name="kiszallas" >
              </b></td>
          </tr>
          <tr>
            <td>Műhely</td>
            <td>
              <input type="checkbox" name="muhely" >
             </td>
          </tr>
        </table>
      </td>
      <td width="79%">
        <p align="left">Beépített alkatrészek<br>
          <textarea name="alkatreszek" rows=2 cols =60 ></textarea>
        </p>
        <p align="left">Alkatrészek ára (nettó)
          <input type="text" name="alkatreszar" size="40">
          +25% ÁFA</p>
      </td>
    </tr>
    <tr>
      <td width="21%"  valign="top"><br>
        A munkát végző(k): <input type="text"  name="munkavegzo" size="30">
      </td>
      <td width="79%">Munkaórák <input type="text" name="munkaido" >
  <br>Munkadíj (netto) <input type="text" name="munkadij">+25% ÁFA
   </td>
    </tr>
    <tr>
      <td width="21%"  valign="top">
        <p>Dátum:
          <?php echo "$datum";  ?>
        </p>
      </td>
```

```
      <td width="79%"> Digitális aláírás:
        <input type="password" name="pswd" value="titok" size=20>
      </td>
    </tr>
    <tr>
      <td width="21%">
        <p>
          <input type="submit" name="Submit" value="Elküldés">
          <input type="submit" name="Reset" value="Mégsem">
         </p>
      </td>
      <td width="79%"> <font face="Times New Roman, Times, serif">Igazolás:<br>
        <textarea  name="Igazolas"  rows=2 cols =60 ></textarea>
        </font></td>
    </tr>
  </table>
</form>

<p><A HREF="index.html" onMouseOver="document.vissza.src='../visszaanim.gif';"
onMouseOut="document.vissza.src='../vissza.gif'">
  <IMG SRC="../vissza.gif" NAME="vissza" ALT="Vissza a főoldalra" BORDER=0 height="25">
  </A> </p>
<p>Utolsó módosítás: 2011. március 12.</p>
</body>
</html>
```

11.5.4 $_SESSION változók

A session változók olyan változók, amelyek megtartják értékeiket miközben a felhasználó egyik oldalról átlép a másikra anélkül, hogy a korábban ismertetett módszerek valamelyikével direkt át kellene adnunk az értékeket a lapok között. Ez a lehet őség igazi globális változókat enged meg és sokkal összetettebb WEB-es programok készítését teszi lehetővé. Több lapból álló site fejlesztése gyakorlatilag session változók nélkül nem megy.

Amikor egy felhasználó belép egy WEB oldalra, akkor egy egyedi azonosító keletkezik, az úgynevezett session id (SID), amelyet vagy a böngészőben tárolunk úgynevezett cookie (süti) formájában, vagy a szerver oldalon tartunk nyilván. A sessionok támogatják korlátlan mennyiségű változó regisztrálását és a tartalmuk megtartását. Amikor a felhasználó eléri a web oldalt, akkor a PHP automatikusan leellen őrzi, hogy a megfelel ő session id vajon már létezik-e a szerveren. Ha létezik a session id, akkor a session-höz tartozó elmentett értékeket hozzárendeli a lekért oldalhoz.

Minden oldal elején használjuk a session_start() függvényt, vagy implicit módon a session_register() függvényt.

Amikor a látogató elindít egy PHP-s lekérést, a PHP motor megnézi, hogy a fenti esetekben van-e a kéréshez hozzárendelve egy session id. Ha van, akkor a korábban elmentett környezetet hozzárendeli ehhez a kéréshez, azaz visszaállítja a megfelel ő változókat. Minden regisztrált változót elment a rendszer a kérés befejez ődésekor. Azok a regisztrált változók, amelyek nem kaptak értéket, azaz nem definiáltuk őket, a nem definiáltak közé kerülnek. Ezek a változók csak akkor kerülnek a definiáltak közé később is, ha a user értéket ad neki.

Ha a `register_globals` engedélyezett, akkor minden globális változót session változónak tudunk elmenteni, és a session változók a következő kérés során automatikusan globális változókká válnak.

Hogyan kezelhetjük a session id-ket?

Cookie - sütikkel

URL parameterekkel

A session modul mind a két változatot támogatja. A cookie-k az optimálisak, viszont vannak olyan kliensek, akik nem támogatják a cookie-k elhelyezését a gépükön biztonsági okokból, ráadásul ilyenkor a böngész ő és a szerver között vándorolnak adatok is. Ez biztonsági problémákat vet fel. A második módszer esetén a session id az URL része.

A PHP képes hajlékonyan kezelni a kérdést, ha megfelel ően fordítottuk. Ebben az esetben a relatív URI-k megváltoznak automatikusan és tartalmazni fogják a session ID-t (=SID). Más esetben használhatjuk a SID konstanst, amely a session_name=session_ID vagy egy üres stringet tartalmaz

```
(pl. PHPSESSID=8e1f5ff69434aea7ecab51da33314b53&PHPSESSID=8e1f5ff69434aea7ecab51da33314b53 )
```

Az alábbi példában bemutatjuk, hogyan lehet regisztrálni egy változót és egy URI-hoz hozzárendelni a session ID-t, felhasználva a SID-et.

Példa 3. Egy user bejelentkezéseit számolja le ez a példa

```
<?php
session_start();
$_SESSION[”count”]++;
$count = $_SESSION[”count”];
?>

Hello visitor, you have seen this page <? echo $count; ?> times.<p>

<php?
# the <?=SID?> is necessary to preserve the session id
# in the case that the user has disabled cookies
?>

To continue, <A HREF="nextpage.php?<?=SID?>">click here</A>
```

Ha a fenti kódot lefuttatjuk és megnézzük a PHP.INI-ben megadott könyvtárban lévő file-okat, akkor látni fogjuk, hogy a session indulása után létrejön egy file (pl. C:\temp-ben) valami hasonló névvel, sess_8e1f5ff69434aea7ecab51da33314b53. Ez tartalmazza a session változók nevét és értékét. Ez felveti azt a problémát, hogy az ilyen típusú file-ok a szerveren lév ő temp könyvtárban csak gy űlnek és korrekt lekezelésük idővel nagyon nehézzé válik. Azt is figyelembe kell venni, hogy egyes sessionok elévülnek, másokat nem lehet még törölni, mert éppen futó alkalmazás használja.

A PHP.INI session részében vannak azok a beállítások, amelyek a session file-ok elévülését, a szemétszedést és egyebeket szabályoznak. A szerver automatikusan gondoskodik egy idő múlva a session file-ok törléséről.

Másfelől a fejlesztőket gondoltak arra is, hogy a programozók a saját kezükbe akarják venni a session kezelésének lehetőségét. Mivel ez nem a kezdők szintjén elérhető, ezért itt nem foglalkozunk vele.

A ”sess_” kezdet ű filenevek session file-okat takarnak, azoknak megnézzük és a jelenlegi id őt és az utolsó hozzáférésének idejét kivonva egymásból megnézzük, hogy lejárt-e az élete.

11.5.5 COOKIE-k (sütik)

A cookiek használata a PHP környezetben lehetséges, de nem igazán ajánlott. A PHP alkalmazások a szerveren futnak és általában valamiféle user azonosításhoz kötöttek. A cookike-k használata esetén a cookie-ban tárolt adatok átkerülnek a böngész őt futtastó számítógépre és ott egy textfile-ban tárolódnak, aminek a visszafejtése csak idő kérdése, éppen ezért fontos vagy titkolni való adatot a cookie-kba sohas ne tároljunk, inkább használjunk sessionöket.

bool **setcookie** (string nev, string ertek, int lejarat, string utvonal, sting domain, int titkos)

A fenti függvény minden paramétere az els őt kivéve elhagyható. A függvényt a html oldal headerében kell elküldeni, mielőtt az oldalra bármit kiírnánk! Az alábbi példában elküldünk egy egyszerű értéket:

Param	**Leírás**	**Példa**
Nev	A süti neve	'teszt’ nevű változót hozzuk létre `$_COOKIE['teszt']`
Ertek	Ez az érték tárolódik a kliens oldalon	Ezt az értéket tároljuk a 'teszt' nev ű változóban. $ertek =`$_COOKIE['teszt']`
Lejarat	A süti lejárati ideje másod-percekben megadva . Beállítása time() + lejárati idő.	`time()+60*60*24*2` két napot állítunk be lejáratnak. Ha nincs beállítva, akkor aböngésző bezárásáig érvényes a süti.
Utvonal	Hol tároljuk a sütiket.	Minden böngészőrendelkezik egy alapértelmezett süti tárhellyel. Beállíthatjuk, hogy ehhez képest hol tárolja a sütiket. Például a /fz/ az fz alkönyvtárba teszi a sütiket.
Domain	Az a domain, amire érvényes a süti	Itt adhatjuk meg, hogy melyik aldomainre legyen érvényes a süti. A `www.fz.ini.hu` esetén csak erre a domainre érvényes.

Param	Leírás	Példa
Titkos	Ha az érték 1, akkor c sak HTTPS esetén küldi sütit.	0 vagy 1 , alapérterlmezés 0

```
<?php
$ertek = 'Ez itt a példaszöveg';
setcookie ("teszt", $ertek);
setcookie ("teszt", $ertek,time()+3600);  /* egy óra múlva jár le a süti */
setcookie ("teszt", $ertek,time()+3600, "/fz/", ".fz.ini.hu", 1);
/* Az /fz alkönyvtárban, a www.fz.ini.hu domain és https protokoll esetén */
?>
```

Tömböket is tárolhatunk sütikben.

```
<?php
// A sütik beállítása
setcookie ("cookie[three]", "cookiethree");
setcookie ("cookie[two]", "cookietwo");
setcookie ("cookie[one]", "cookieone");

// A következő oldalon betöltve az alábbi kóddal irathatjuk ki az adatokat:
if (isset($_COOKIE['cookie'])) {
    foreach ($_COOKIE['cookie'] as $name => $value) {
        echo "$name : $value <br />\n";
    }
}

/* Ez lesz az eredmény
three : cookiethree
two : cookietwo
one : cookieone
*/
?>
```

A következő oldalt betöltve a böngész őbe a sütik automatikusan megjelennek a $_COOKIE tömbben, és azokat az értékeket lehet használni. Ha a register_globals paramétert bekapcsoljuk a php.ini-ben, akkor automatikusan létrejönnek a megfelel ő változók, de korábban említettük, hogy ennek a paraméternek a bekapcsolása nem javallott.

Ha a $_COOKIE tömb értékeit debuggolás céljából ki akarjuk iratni, akkor használjuk a következő utasítást:

```
<?php
Print_r($_COOKIE)
?>
```

12 Konverzió Adattípusok között

A PHP automatikusan, és meglehet ősen szabadon kezeli a típusokat. Néha szükség lehet egy bizonyos adattípus alkalmazására. Ekkor használhatjuk az alábbi módszereket:

Előírjuk a típust, mint a C-ben:

```
$a = (float) $b;
$c = (int) $c;
```

Használhatjuk a *bool* **settype** ($valtozo, ”tipus”) függvényt. Ekkor a tetszőleges típusú $változó-t átkonvertáljuk az adott típusra. A konverzió sikerességéről *bool* eredményt ad vissza a függvény.

Lehetséges *tipus* 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.

```
$size    = "5valami"; // string
$valami = true;      // boolean

settype($size, "integer"); // $size most 5 (integer)
settype($valami, "string");  // $valami most "1" (string)
```

13 Tömbök

A tömbök, azok különböző lehetőségei, és az azok köré felsorakoztatott függvények a PHP programozás egyik legerőteljesebb eszközrendszerét alkotják, ugyanakkor rendkívül egyszerűen és könnyedén használhatók. A tömb változók halmaza, melyeket a tömbön belül sorban tárolhatunk és a teljes adathalmazt egyszerre is kezelhetjük, ugyanakkor a tömb elemeihez külön-külön is hozzáférhetünk. Fontos tulajdonsága a tömböknek, hogy egy tömbön belül az elemek típusa különböző lehet. Egy tömb elemeit legegyszerűbben explicit módon, elemenként tölthetjük fel.

```
$tomb[1] = "dBase";
$tomb[2] = "FoxPro";
$tomb[4] = "Clipper";
$tomb[5] = 42;
```

Látható, hogy a tömb elemeinek megadásakor nem szükséges a sorrendiséget szigorúan betartani. Egy tömb elemeihez a fentieknél egyszerűbben is, a tömbindex használata nélkül is lehet elemeket adni:

```
$tomb[] = "Basic";
$tomb[] = "FoxPro";
```

Ily módon a tömb végéhez kapcsolódnak az új elemek, az index értéke pedig az legutolsó indexelemnél eggyel magasabb lesz. Hasonlóan működik az **array_push()** függvény, azzal a különbséggel, hogy egy utasításon belül több értéket is hozzáfűzhetünk a tömbhöz:

```
array_push($tomb, "Cobol", "Fortran");
```

Szép lassan dagadó tömbünk a fenti utasításokat követően már így néz ki:

```
Array
(
    [1] => dBase
    [2] => FoxPro
    [4] => Clipper
    [5] => 42
    [6] => Basic
    [7] => FoxPro
    [8] => Cobol
    [9] => Fortran
)
```

Természetesen a tömbök értékeinek megadásához hasonlóan férhetünk hozzá a tömbelemekhez, azonban a fent említett **array_push()** függvény párja, az **array_pop()** függvény is rendelkezésünkre áll, mely azonban nemcsak egyszerűen a tömb utolsó elemét adja vissza értékül, hanem a tömb elemeinek számát is csökkenti az utolsó elemmel:

```
$nyelv1 = $tomb[1];  // $nyelv1 értéke "dBase"
$nyelv2 = $tomb[4];  // $nyelv2 értéke "FoxPro"
$nyelv9 = array_pop($tomb);// $nyelv9 értéke "Fortran" és a tömb nyolc elemű lesz…
```

Bonyolítsuk egy kicsit a dolgokat. Ezidáig a tömbünk egy dimenziós volt, azonban a PHP nyelvben a tömbök kettő vagy akár több dimenziósak is lehetnek. Az értékadás legegyszerűbb módja ilyen esetben is az explicit értékadás:

```
$auto[1][1] = "Maserati";
$auto[1][2] = "olasz";
$auto[2][1] = "Renault";
$auto[2][2] = "francia";
$auto[3][1] = "Mercedes";
$auto[3][2] = "német";
```

a tömb valahogyan így fog kinézni:

```
Array
(
    [1] => Array
        (
            [1] => Maserati
            [2] => olasz
        )
    [2] => Array
        (
```

```
            [1] => Renault
            [2] => francia
        )
    [3] => Array
        (
            [1] => Mercedes
            [2] => német
        )
)
```

Ilyen és ehhez hasonló tömbök létrehozására, azonban sokkal tömörebb és olvashatóbb módszer az array() függvény használata. Ez a függvény a paraméterként megadott értékeket tömb formában adja vissza. Így a fenti értékadással pontosan megegyező eredményt ad a következő:

```
$auto[1] = array( "Maserati" , "olasz" );
$auto[2] = array( "Renault" , "francia" );
$auto[3] = array( "Mercedes" , "német" );
```

Ahogyan azonban a tömbelemek típusaira vonatkozóan nincsenek túl szigorú megkötései a PHP nyelvnek, ugyanúgy nem kezeli szigorúan a többdimenziós tömbök elemszámait sem a PHP. Az alábbi értékadás teljesen helyes eredményt ad:

```
$auto[1] = array( "Maserati" , "olasz" );
$auto[2] = array( "Renault" , "francia" , "406", "206" );
$auto[3] = array( "Mercedes" , "német" , "E320",
                  "Vito" , "Sprinter kisteherautó" );
```

Természetesen az **array_pop()** és az **array_push()** függvények az **array()** függvénnyel ötvözve több dimenziós tömbök esetén is használhatók.

```
array_push( $auto, array("Citroen" , "francia" , "ZX" , "Xsara");
```

A fenti esetekben a tömb elemei azok sorszámaival voltak azonosítva. A PHP ismeri az **asszociatív tömbök fogalmát** is. Az asszociatív tömbök rendkívül hasznos és sokoldalú elemei a PHP nyelvnek. A PERL nyelvben használt hash típusú tömbökhöz hasonlóan működnek. A tömbelemekre való hivatkozás ilyen esetben nem sorszámmal, hanem egy indexelem (kulcs) segítségével történik, egyszerűen úgy, hogy a sorszám helyére, az indexelemet helyezzük.

```
$tomb["első"] = "Kis Gedeon";
$tomb["második"] = "Nagy Elemér";
```

Függetlenül attól, hogy a tömb elemeinek milyen sorrendben adtunk értéket, az elemeket az indexkulcs segítségével érhetjük el, és ez nem függ attól, ha a tömbhöz hozzáfűzünk, vagy attól elveszünk egy elemet. Új elem bármikor hozzáfűzhető a tömbhöz:

```
$tomb["harmadik"] = "Kukonya Berkó";
```

Az asszociatív tömbök lehetnek egydimenziósak, mint a fenti példában, de lehetnek több dimenziósak is. A fenti példát kibővíthetjük több dimenzióssá:

```
$tomb["első"]["neve"] = "Kis Gedeon";
$tomb["első"]["kora"] = 27;
$tomb["második"]["neve"] = "Nagy Elemér";
$tomb["második"]["kora"] = 22;
```

Ha a "Nagy Elemér" értéket elemet a **$tomb["második"]["neve"]** hivatkozással tudjuk elérni, de ha **$sorszam** értéke "második" akkor akár **$tomb[$sorszam]["neve"]** hivatkozással is elérhetjük a keresett elemet.
A normál és az asszociatív tömbök létrehozására egyaránt használható az **array()** függvény, amit leginkább tömbök kezdőértékfeltöltése során használhatunk, egy értékadással kiküszöbölve többet. A fenti példákkal megegyezőek az alábbi értékadások:

```
$tomb = array ( "első"    => "Kis Gedeon",
                "második" => "Nagy Elemér");

$tomb = array ("első"    => array ("neve" => "Kis Gedeon",
                                   "kora" => 27),
```

```
                "második" => array ("neve" => "Nagy Elemér",
                                    "kora" => 22) );
```

Mint az alábbi példa is mutatja, az értékadás esetén az index értékét nemcsak konkrétan, hanem változóval is megadhatjuk, így már meglehet ősen rugalmasan töthetjük fel tömbjeinket adatainkkal. A következ ő példa megmutatja a **print_r()** függvény használatát is, amit tetszőleges változó értékének kiíratásához használhatunk, de mivel tömbváltozó esetében a komplett tömbstruktúrát is megjeleníti leginkább tesztelési célokra használható nagyon jól.

```
<?php

    $nick1 = "Tabbi";
    $nick2 = "Chris";
    $tomb = array    (
        $nick1 => array("nev" => "Tabi Imre",
                        "email" => "tabbi@freemail.hu"),
        $nick2 => array("nev" => "Nagy Krisztián",
                        "email" => "chris@nomail.hu")
        );
    echo("<PRE><b>");
    print_r($tomb);
    echo("<HR>");
    print_r($tomb["Tabbi"]["nev"]);

    echo("</b></PRE>");
?>

A program kimenete a következő lesz:
Array
(
    [Tabbi] => Array
        (
            [nev] => Tabi Imre
            [email] => tabbi@freemail.hu
        )

    [Chris] => Array
        (
            [nev] => Nagy Krisztián
            [email] => chris@nomail.hu
        )
)
```

Asszociatív tömbök esetében azonban figyelemmel kell lenni arra, hogy ilyen tömb elemeit kizárólag a meghatározott indexértékkel érhetjük el, a tömb sorszámával nem. Ennek rendkívül egyszerű az oka. Az egyszerű sorszámozott tömb is asszociatív tömb, ahol a tömbindex maga a sorszám. S őt egy tömbön belül keverhetjük is a sorszámozott és az indexelt elemeket, de azért ezt kerüljük, csak gondot okozunk magunknak. A normál és az asszociatív típusú tömbök a PHP programozás során rendkívül változatosan és hatékonyan használhatók, f őleg akkor, ha tudjuk azt, hogy a PHP a tömbök elemeire, az elemszámokra és a tömbelemek típusaira vonatkozóan rendkívül szabad kezet ad nekünk:

- többdimenziós tömbön belül az egyik index lehet asszociatív, a másik normál
- többdimenziós tömb esetében a tömbelem tömböknek nem kell feltétlenül azonos elemszámúaknak lenni, vagyis **$tomb[1]** lehet öt elemű, míg **$tomb[2]** lehet akár 8 elemű is.
- egydimenziós tömbök esetében a tömbelemek lehetnek különböz ű típusú adatok, de még többdimenziós tömbök esetében sem kell a tömbelem tömbök adatszerkezetének megegyeznie.

Vagyis elég nagy szabadsággal használhatjuk a tömbváltozókat, mégis érdemes szem el őtt tartani, hogy ha lehet, járjunk el következetesen a változók értékadásával és azok használatával.

A PHP nyelvben a tömbváltozókhoz is egy egész sor függvény és utasítás kapcsolódik. Ezek a http://weblabor.hu/php/ref.array.php címen találhatók meg. Ezek a függvények egy egész sor feladatot láthatnak el kezdve a tömbök definíciójától az értékfeltöltésen és a tömbben való mozgáson keresztül a tömbelemek legváltozatosabb módú sorbarendezéséig. Mivel a tömbök a PHP programozás során igen kiterjedten használatosak, és a tömbökhöz kapcsolódó függvények fontossága kiemelked ő, így ezen függvényekb ől a fontosabbakat a következő részekben részletesebben is tárgyaljuk majd.

14 Sztringek, szövegek manipulációja

A továbbiakban néhány gyakran el őforduló szövegmanipulációs feladat megoldását tekintjük meg PHP-ban. Természetesen sok olyan feladat van, amit máshol, máshogy már vagy még érintünk, illetve sok egyéb, itt nem megemlített függvény található a stringkezelésnél. A példák a php manualból valók!

Gyakori, hogy egy string hosszát meg szeretnénk tudni: **strlen()**

Ha egy string valahány karakterét ki akarjuk venni a szövegből, akkor a használandó függvény: **substr()**

A 2. és harmadik paraméter lehet negatív is. Ekkor a jelentésük a szöveg végéről értendő.

```
<?php
$l=strlen(”szevasz tavasz!”);
$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”
$string = 'abcdef';
echo $string{0};                // returns a
echo $string{3};               // returns d
$rest = substr("abcdef", -1);    // returns "f"
$rest = substr("abcdef", -2);    // returns "ef"
$rest = substr("abcdef", -3, 1); // returns "d"
$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"
?>
```

Ha ki akarom cserélni a szöveg egy részét más részre, akkor: **substr_replace()** .

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Eredeti: $var<hr>\n";

// Beszúrjuk a 'bob' szót a $var elejére
echo substr_replace($var, 'bob', 0, 0) . "<br>\n";

// Ha a második paraméter negatív, akkor a szöveg végéről indul a csere.
echo substr_replace($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace($var, 'bob', -7, -1) . "<br>\n";

// Töröljük a szöveg elejéről
echo substr_replace($var, '', 10, -1) . "<br>\n";
?>
```

A szövegben egyes karakterek, stringek előfordulásait kicseréljük másik stringekre. **Str_replace()**

```
<?php
// Eredmény: <body text='black'>
$bodytag = str_replace("%body%", "black", "<body text='%body%'>");

// Eredmény: Hll Wrld f PHP
$vowels = array("a", "e", "i", "o", "u", "A", "E", "I", "O", "U");
$onlyconstants = str_replace($vowels, "", "Hello World of PHP");

// Provides: You should eat pizza, beer, and ice cream every day
$phrase  = "You should eat fruits, vegetables, and fiber every day.";
$healthy = array("fruits", "vegetables", "fiber");
$yummy   = array("pizza", "beer", "ice cream");

$newphrase = str_replace($healthy, $yummy, $phrase);
?>
```

Ha egy string helyét akarom megtudni egy másik stringben, akkor: **strpos();**

Ugyanez a végéről: **strrpos()**

Ugyanezek kis/nagybetű érzéketlen módon: **stripos(), strripos()**

```
<?php
$findme    = 'a';
$mystring1 = 'xyz';
$mystring2 = 'ABC';
$pos0 = strpos ($mystring1, ”x”);
$pos1 = stripos($mystring1, $findme);
$pos2 = stripos($mystring2, $findme);

if ($pos0 === false) {
    echo "A '$findme' stringet nem találtam a '$mystring1' stringben";
}else{
    echo "Az 'x' karakter helye: $pos0";
}
// 'a' nincsen 'xyz'-ben
if ($pos1 === false) {
    echo "A '$findme' stringet nem találtam a '$mystring1' stringben";
}

if ($pos2 !== false) {
    echo "Megtaláltuk a '$findme' stringet '$mystring2' stringben ezen a helyen: $pos2";
}
?>
```

Egy string kiegészítése karakterekkel: **str_pad()**

```
<?php
$input = "Alien";
print str_pad($input, 10);                      // produces "Alien     "
print str_pad($input, 10, "-=", STR_PAD_LEFT);  // produces "-=-=-Alien"
print str_pad($input, 10, "_", STR_PAD_BOTH);   // produces "__Alien___"
print str_pad($input, 6 , "___");               // produces "Alien_"
?>
```

Ha egy szöveg elejéről és végéről le akarom vágni a bevezető és a záró szóközt, tab-ot, soremelés karaktert, akkor a **trim()** függvényt használhatom. Ha csak a szöveg elejér ől akarom levágni a fent említett karaktereket, akkor **ltrim()**, ha a végér ől, akkor **rtrim()**. Alapesetben a levágandó karakterekhez további kiegészítéseket is rendelhetek.

```
<?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
?>
```

Érdekes lehet őség bizonyos speciális karakterek, tag-ek kihagyása a stringekb ől, illetve bizonyos speciális karakterek beszúrása:

strip_tags(), htmlentities(), htmlspecialchars(),stripslashes(),

```
<?php
$str = "Is your name O\'reilly?";

// Kivettük az idézőjel karaktert. Eredmény: Is your name O'reilly?
echo stripslashes($str);
echo strip_tags($string, '<a><b><i><u><span><body>');                                                    //Kive

$str = "A 'quote' is <b>bold</b>";

// Kimenet: A 'quote' is &lt;b&gt;bold&lt;/b&gt;
echo htmlentities($str);

$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
echo $new; // &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;
?>
```

Gyakori feladat, hogy szét kell vágni egy stringet valailyen karakter mentén darabokra, például szavakra, ahol a szavakat határoló karakter többféle is lehet. Az eredmény egy stringekből álló tömb lesz: **Explode()**

```
<?php
// Example 1
$pizza  = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
print $pieces[0]; // piece1
print $pieces[1]; // piece2

// Example 2
$data = "foo:*:1023:1000::/home/foo:/bin/sh";
list($user,$pass,$uid,$gid,$gecos,$home,$shell) = explode(":",$data);
print $user; // foo
print $pass; // *

?>
```

Az alábbi példa egy PHP oldalról keresztreferenciát készít és kiírja a beinclude-olt file-okat.

```
<?php
/*************************************
* Crossreference v1.0
* It makes a crossreference list about the
* variables of current script
* with command line PHP interpreter.
*
* Copyright by Fabian Zoltan 2004
*
*************************************/
$lines = array();
$vars = array();

$terminal = array("  ","<",">","=",";",")","(",")","\"","\'","!","+","-
","\t","*","/","?","&",":",".","#","@","{","}","[","]","|","%",",");
$ar=str_replace("\\","/",$argv); //Az összes \\ jelet /-re cserélem

$text=file($ar[1]); //Megnyitom a file-t

function includes($f) {
 $terminal = array("  ",";","!","+","-","*","/","=","\n"); //A szöveghatároló jelek listája
 $fileterm = array("\"","'");
 $inc =array(); //includes calling
 $lines = array();
 $out = "";
 $v   = "";

 $token = array("include","include_once","require"," require_once");
 $text=file($f);
 while (list ($key, $val) = each ($text)) {
  $lin = str_replace($terminal," ",$val);
  if( (strpos($lin,$token[0])>0) or
   (strpos($lin,$token[1])>0) or
   (strpos($lin,$token[2])>0) or
   (strpos($lin,$token[3])>0)
  ) {
   $lines[]=str_replace($terminal," ",$val);
  }
 }
 $i=0;
 /*while ($i<$sizeof($lines)){
  if( ) {
   $out .= includes($f);
  }
  $i++;
 } */
 print_r($lines);
 Return $out;
}
```

```
// Variables crossreference
function variables($text) {
global $terminal,$lines,$vars;
//change Terminator characters
 while (list ($key, $val) = each ($text)) {
  $lines[]=str_replace($terminal," ",$val);
 }

 $splitted =array();
 $line=0;
 while(list($key,$row)= each($lines)) {
  $splitted = explode(" ",$row); //Szétvágom szavakra a sort. Az eredmény egy tömb
  for($i=0; $i<sizeof($splitted); $i++) {
      $st = $splitted[$i];
   if(substr($st,0,1)== "$"){
    if(!isset($vars[$st])) $vars[$st] = "->".sprintf("%4d",$line).", ";
    else       $vars[$st].= sprintf("%6d",$line).", ";
   }
  }
  $line++;
 }
 ksort($vars);
 $out = "--------- Variables ---------\n";
 while (list ($key, $val) = each ($vars)) {
  $out .=str_pad($key,10).$val."\n";
 }
 Return $out;
}

printf(variables($text));
printf(includes($ar[1]));
?>
```

15 Formok /űrlapok – Interaktív programok írása

A PHP és általában a WEB-es programozás egyik sarokköve volt az interaktivitás megjelenése. Ehhez arra volt szükség, hogy a böngészőnkön beírt adatokat vissza tudjuk küldeni a szervernek, amely azt feldolgozza.

A HTML-ben lehet űrlapokat létrehozni az alábbi szintaktikával:

```
<form name='Urlap' action='index.php' method='xxxx'>
 <INPUT ......>
</form>
```

A fenti xxxx= vagy GET vagy POST metódus lehet.

Az űrlapok belsejében minden html elemet használhatunk, és itt használhatunk olyan beviteli mez őket, amelyek változóknak adnak értéket. A POST és a GET metódus segítségével a változók neve és értéke eljut az action-nal megjelölt laphoz, amely azt fel tudja dolgozni. Az űrlapon belül az alábbi adatbeviteli lheetőségek vannak:

```
<INPUT TYPE='TEXT' Name="text" VALUE="Kezdőszöveg" WIDTH="60">
```

Egy soros szöveg bevitelére szolgál

```
<INPUT TYPE='PASSWORD' Name="text" VALUE="Kezdőszöveg" WIDTH="60">
```

- Egy soros password, beviteléhez kell.

15.1.1 Önmagukat meghívó űrlapok

Gyakori feladat, hogy egy Űrlapot meghívunk, leellenőrizzük és a következő oldalon csak akkor küldjük tovább a böngészőt, ha az oldal hibátlanul ki van töltve. Ehhez az alábbi dolgoknak kell teljesülnie:

Az űrlapot saját magának küldi el a PHP oldal

Amikor beérkezik a kérés a szerver oldalon ellenőrizzük a megfelelő értékek meglétét.

Ha az értékek megvannak, akkor feldolgozzuk őket

Ha nincsenek meg az értékek, akkor újra meghívjuk a feldolgozandó adatot:

Az alábbi példában két rértéket küldünk el egy feldolgozandó oldalnak. Ha nincs kitöltve mind a két érték, akkor újrahívjuk a feldolgozandó oldalt.

A feldolgozás részen a feldolgozott értékek valamiféle lekérdezésbe torkollnak vagy akármilyen egyéb m űveletet végezhetünk vele.

Javasolt egyéb műveletek:

Az átadott adatok típus szerinti ellenőrzése. Erre használhatók az alábbi függvények:

is_bool() – logikai-?
is_int(), is_integer() – egész-e?
is_float(), is_real(),– lebegőpontos-e?
is_object() – Objektum-e?
is_array(). – Tömb-e?
Is_string() – string-e?

Az átadott adatokból célszerű kivenni a HTML kódokat, mivel biztonsági problémák merülhetnek fel

```
<?php
 $string = strip_tags($string);
?>
```

Az átadott értékekből célszerű kiszedni a [{()}] jeleket.

Ha az így bevitt értékek közvetlenül SQL lekérdezések összeállítására használatosak, akkor célszer ű még egyéb speciális jelek kivétele is, például = <> OR, AND ! stb...

```
$a =array("[","{","(",")","}","]","or","and","!");
$str = str_ireplace($a, " ", $str);
```

```
<?php
$OK='';
```

```
$nev =’’;
$pwd=’’;
$uz =’’;

if (isset($_POST[’OK’]) $OK =$_POST[’OK’];
if (isset($_POST[’nev’]) $nev =$_POST[’nev’];
if (isset($_POST[’pwd’]) $pwd =$_POST[’pwd’];

if ($OK =”OK”) {
 $nev =$_POST[„nev”];
 $pwd =$_POST[”pwd”];

 //HTML tag-ek kiszedése
 $nev = strip_tags($nev);
 $pwd = strip_tags($pwd);

 //speciális jelek kivétele
 $a =array(”[”,”{”,”(”,”)”,”}”,”]”,”or”,”and”,”!”);
 $str = str_ireplace($a, " ", $str);

 if (!isset($nev) or empty($nev))  $uz =’Töltsd ki a név beviteli mez   őt”;
 if (!isset($pwd) or empty($pwd))  $uz .=’Töltsd ki a jelszó beviteli mez   őt”;
}

if (!empty($uz)){
 //Form kiiratása
 print(”<P>”.$uz.”</P>”);
 print(’<FORM METHOD=”POST”>’);
 print(”<Table>”);
 print(”<TR><TD>Név:</TD><TD>
 print(”<input type=’text’name=’nev’ value=’”.$nev.”’>”);
 print(”</TD></TR>”);

 print(”<TR><TD>Jelszó:</TD><TD>”);
 print(”<input type=’password’name=’pwd’ value=’”.$pwd.”’></TD></TR>”);
 print(”<TR><TD> </TD><TD>”);
 print(”<input type=’submit’ name=’OK’ value=’OK’>”);
 print(”</TD></TR></TABLE>”);
 print(’</FORM>’);
}else{
 /* Feldolgozás*/
}
?>
```

16 Formok adatainak feldolgozása – szerver- és kliens oldalon

A PHP-ben a HTML űrlapok (FORM-ok) felhasználásával történik meg az interakció a felhasználó és programja között. Az űrlapok használatának gyakori formája, hogy az űrlap a saját magát tartalmazó oldalt hívja meg és az űrlap kitöltöttségét olyan kóddal ellen őrizzük, amely ugyanazon az oldalon van. Ez célszer űen azért lehet így, mert a beviteli formátum és az ellen őrzés is egy helyen található. A példában egy egyész típusú, 1000-nél nagyobb adatot és egy folyószámla nevet vár a beviteli oldalon a program.

```
<?php
$message = ’’;
$adat =’’;
$ok = True;
if  (isset($_POST(”OK”) && $_POST[”OK”] ==”Elküld”){
 // Ellen   őrizzük az adat léteznik-e és megfelelő formátumúak?
 $adat = -1;
 if (isset($_POST[”adat”])){
  $adat= $_POST[”adat”];
  if (is_int($adat)){
   if (!($adat>1000)){
    $ok = $ok && False; //Ezernél nagyobb értéket ellen                őrzök éppen
    $mess_adat = ’<FONT color=”#FF0000”>Túl kicsi érték!</FONT><BR>’;
   }
  }else{
   $mess_adat = ’<FONT color=”#FF0000”>Hibás adattípus!</FONT><BR>’;
  }
 }else{
  $mess_adat = ’<FONT color=”#FF0000”>Hiányzó adat!</FONT><BR>’;
 }
 //Ellen   őrizzük, hogy a folyószámla neve megfelel-e?
 $fszamla = -1;
 if (isset($_POST[”fszamla”])){

  $fszamla= $_POST[”fszamla”];
  if (is_string($fszamla)){
   $fszamla = $urldecode($fszamla); //Esetleges URL kódolás dekódolja
   $fszamla = strip_tags ( $fszamla); //kiveszi a HTML és PHP tageket
   $fszamla = stripcslashes ( $fszamla); // kiveszi a \ jeleket

   //SQL injekció kisz           űrése!!!
   $keres = array (”WHERE”,”LIKE”,”(”,”)”);
   $csere = array (” ”,” ”,” ”,” ”);
   $fszamla = str_replace($keres,$csere,$fszamla);

   if (van_e_szamla($fszamla)){
    $ok = $ok && False; //Ezernél nagyobb értéket ellen                őrzök éppen
    $mess_fszamla = ’<FONT color=”#FF0000”>Nincs ilyenfolyószámla!</FONT><BR>’;
   }
  }else{
   $mess_fszamla = ’<FONT color=”#FF0000”>Hibás adattípus!</FONT><BR>’;
  }
 }else{
  $mess_fszamla = ’<FONT color=”#FF0000”>Hiányzó adat!</FONT><BR>’;
 }

 .... itt ellen   őrizzük a többi szükséges mező értékét és létezését ...
}

if(!$ok){
 print(’<FORM name=”pelda” action=”urlap.php” method=”POST”>’);
 print($mess_adat);
 print(<’INPUT type=”text” name=”adat” value=<% = $adat %> > ’);
 ... további beviteli mez     ők ...
 print(’<INPUT type=”submit” name =”OK” value=”Elküld”>’);
 print(’</FORM>’);
}else{
 .... adatok feldolgozása ...
 header("Location:    http://www.example.com/"); //A böngész   őt átdobom a következő oldalra
 exit;                    // a kód többi része ne fusson le
}
?>
```

A párbeszédek alkalmazásánál fontos, hogy ellen őrizzük a bevitt adatokat, mivel véletlenül, akarva vagy akaratlanul olyan adatok kerülhetnek a meghívott oldalra, aminek típusa, formátuma nem felel meg az elvárásoknak ennek érdekében az alábbi ellenőrzéseket célszerű megtenni:

- Egy oldalon minden mező ki van-e töltve
- A megfelelő típusú, formátumú adat került-e bele
- Nincs-e benne HTML vagy PHP, vagy SQL utasítás kód (ezek sanda szándékok esetén szoktak bekerülni)

Az ellen őrzést két lépcs őben célszer ű megtenni. A kliens oldalon megfelel ő Javascript kód segítségével és a szerveroldalon a PHP kód segítségével, mint fent is látszik.

Bár a Javascript ennek a jegyzetnek nem témája, azért az ellen őrzésre adunk példát itt. A megoldás alapja, hogy minden beviteli mező vagy a Formon lév ő ”Elküld” gomb megnyomására lefut egy ellen őrző script a böngész őn, ami ellenőrzi, hogy ki vannak-e töltve a megfelelő mezők.

A Javascript ellenőrzés ugyanakkor nem helyettesíti a PHP oldali ellen őrzést, mert a böngész ő oldalon megfelelő technikákkal el lehet kerülni az ellenőrzést.

17 Levélküldés, plain text, html levél, attachement

A PHP-ban van lehet őség arra, hogy leveleket küldjünk el megadott címre, megadott tartalomal. Az üzenetküldéshez először a PHP.INI-ben be kell lenni állítani az alábbi szakasz értékeit. Az alábbi sorok a Win32-es rendszeren beállítandókat tartalmazza. Be kell állítani annak az SMTP szervernek a címét kell írni, amelyik elküldi a levelünket és annak a usernek a nevét, akivel defaultban küldünk levelet.

```
[mail function]
SMTP     = mail.szily.sulinet.hu  ;for win32 only
sendmail_from  =    fz@mail.szily.sulinet.hu ;for win32 only
```

A levél elküldése a mail paranccsal történik, de miel őtt elküldenénk, létre kell hozni azokat a stringeket, amelyek segítségével a levél különböző szakaszai létrejönnek.

```
$cimzett = fz@mail.szily.sulinet.hu;
$tema = ”drágám”;
$uzenet = ”A kölykök összementek. Mind a kettőt hiába keresem. Géza”
$fejlecek .= "From: Tőlem <geza@kekazeg.to>\n";
$fejlecek .= "X-Sender: <birthday@php.net>\n";
$fejlecek .= "X-Mailer: PHP\n";      // Levelez           őprogram
$fejlecek .= "X-Priority: 1\n";      // Sürg           ős üzenet!
$fejlecek .= "Return-Path: <geza@kekazeg.to>\n"; //Hiba esetén ide jön levél

mail($cimzett, $tema, $uzenet, $fejlecek);
```

A levél szövegét és a fejlécet tovább cifrázhatjuk. A fenti mez őkön kívül használhatjuk például a cc: bc: Reply-To: és a hasonló fejléc mezőket is. Hogy milyen mezőket lehet használni, bámelyik leveled fejlécéből kinézheted.

Sajnos az attachmentek kezelése nem tartozik ele a php által támogatott dolgok körébe, ezért küls ő alkalmazás nélkül nem tudunk attachmentet küldeni php-ben.

Egy lehet őség adódik komolyabb levelezési funkciók használatára. Meg kell hívni egy küls ő, parancssori levelezőszoftvert, amelynek a megfelel ő paraméterezésével tetsz őleges levelezési funkcókat el lehet érni. Ilyen szoftver például a POSTIE.EXE Windows alatt. Ez a program és más hasonló programok az Internetr ől letölthetők.

A levelezés kérdéskörében fontos, hogy formázott leveleket is tudjunk küldeni. Erre alkalmasak a HTML levelek. A HTML levelek küldéséhez elég sokmindent el kell végezni, alapesetben a PHP nem támogatja a HTML levelek küldését, de a NET-en találhatók egyszer ű objektumcsomagok, amelyek ebben az esetben segítenek (http://phpmailer.sourceforge.net) Ennek a csomagnak a használatára álljon itt egy példa:

```
<?php
require("class.phpmailer.php");

$mail = new phpmailer();

$mail->IsSMTP();                    // set mailer to use SMTP
$mail->Host = "smtp.mydomain.com";  // specify main and backup server
$mail->SMTPAuth = true       // turn on SMTP authentication
$mail->Username = "fz"     // SMTP username
$mail->Password = "titok"   // SMTP password

$mail->From = "fz@mail.szily.sulinet.hu”;
$mail->FromName = "Levelező";
$mail->AddAddress("XY@mail.com", "X Y");
$mail->AddReplyTo("info@mail.com", "Information");

$mail->WordWrap = 50;     // set word wrap to 50 characters
$mail->AddAttachment("C:\temp\text.zip");  // add attachments
$mail->IsHTML(true);      // set email format to HTML

$mail->Subject = "Ez itt a targy";
$mail->Body    = "Ez a HTML szöveg rész <b>vastagon!</b>";
$mail->AltBody = "Ez a rész a nem HTMl szöveg";

if(!$mail->Send())
{
```

```
   echo "A szöveg nem ment el. <p>";
   echo "Levelező hiba " . $mail->ErrorInfo;
   exit;
}

echo "A levél elment";
?>
```

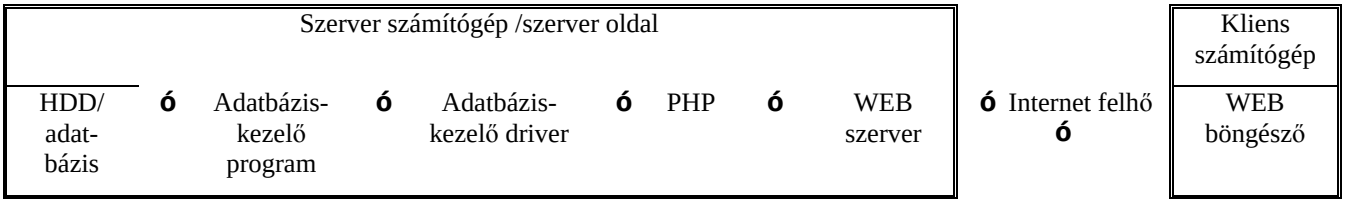
A fenti csomag telepítésekor gondoskodni kell arról, hogy a PHP.INI-ben be legyen állítva az include_path a class.phpmailer.php file-ra, amit az alábbi módon tehetünk meg futás közben:

```
$incl_path = ini_get("include_path");
$incl_path .= ”;./phpmailer”;
ini_set("include_path", $incl_path);
```

18 Adatbázisok

A PHP programozási nyelv úgy kezeli az adatbázisokat, hogy a nyelvbe beépítették az elterjedt és támogatott adatbázis-kezelők támogatását. Az adatbáziskezel ők általában SQL rendszerek, és kliens-szerver minta szerint működnek együtt a PHP-val.

Egy PHP-val meghajtott adatbázis-kezelő rendszernek az alábbiakban lehet felrajzolni a működési sémáját:



Egyes adatbázis-kezel őket a PHP alapból támogatja, míg másokhoz be kell tölteni a PHP.INI-ben a megfelel ő sorok segítségével a támogatást.

Természetesen minden esetben kell telepíteni egy adatbázis-szerver szoftvert, amely nem mindig ingyenes! Magyarországon elterjedt legfontosabb adatbáziskezelők az alábbiak

Adatbázis-kezelő	**Támogatás módja**	**Az adatbázis-kezelő web címe**	**Tulajdonságai**
MySQL	Beépítve	www.mysql.com	Free, gyors, Win és LINUX verzió létezik
Interbase	php_interbase.dll	www.interbase.com	Teljes kör ű, jó adatbázis-kezel ő, ajánlható, az InterBase 6.01 free!
Microsoft SQL7.0/ 2000	php_mssql70.dll php_mssql.dll	www.microsoft.com	Régebbi verzió, létezik un. Personal és Evaulation Edition. Minden MS fejlesztőtermék része a Personal Edition változat.
Oracle8	php_oci8.dll	www.oracle.com	Nem free, teljes körű, nagy tudású SQL szerver
PostGres SQL	php_pgsql.dll	www.postgresql.org.	LINUX és Windows verzió is van, free, teljes SQL rendszer
ODBC	Beépítve		Az ODBC = Open DataBaseConnectivity – A Microsoft által létrehozott, elvileg platform és adatbázis-kezel ő független felület, minden elterjedt adatforrásnak van ODBC drivere. Win és Linux alatt is léteznek ODBC driverek

18.1 MySQL

A fenti adatbázis-kezel őkről annyit, hogy jelenleg a legelterjedtebb ilyen alkalmazás a Linuxon, Netware-en és Windows rendszereken is futó MySQL. Ez a rendszer talán a leggyorsabb az összes elterjedt adatbázis-kezel ő között, de nem valósítja meg az SQL összes lehet őségét. A MySQL nagyon el őnyös tulajdonsága, hogy GNU liszensszel lehet használni, azaz akkor is free, ha eladási céllal használjuk, de magát a MySQL adatbázis-kezel ő programot nem adjuk el (**J**). A MySQL letölthető az alábbi címről: http://www.mysql.com/downloads/

Telepítése egyszer ű Windowson el kell indítani a ZIP-b ől való kicsomagolás után a SETUP.EXE programot, a többit elkészíti ő. Ha firewall van a gépünkön, akkor a TCP/IP 3306-os portját kell szabaddá tennünk. Ha a szerver ugyanazon a gépen fut, mint amelyiken a WEB szervert és a PHP-t futtatjuk, akkor a Firewallon nem szabad (!) engedélyezni más gépről ehhez a porhoz való hozzáférést.

Ha fejlesztés közben az adatbázisunkat módosítani akarjuk, különböz ő dolgokat akarunk elvégezni, akkor ajánlhatjuk a free phpMyAdmin nev ű csomagot, illetve a professzonális, de pénzes EMS MySQL Manager programcsomagot. Ha feltesszük a MySQL ODBC driverét, akkor viszont tetsz őleges ODBC kompatibilis alkalmazásból el tudjuk végezni az adatok módosítását, feltöltését stb... Akár ez lehet egy Microsoft Access is.

A MySQL használatához dokumentációt innen lehet letölteni:

http://www.mysql.com/get/Downloads/Manual/manual.chm-2002-10-07.zip/from/pick

18.2 PostGres SQL

A PostGres SQL egy kicsit komolyabb alkalmazás, mint a fent említett adatbázis-kezel ő. A telepítése egyszer ű, mert 2005-ben megjelent a Windowsos verzió is.

A PostgresSQL használatához dokumentációt az alábbi helyen lehet elérni:

http://www.postgres.org/docs/

Itt HTML és PDF formátumban vannak meg a megfelelő dokumentumok.

18.3 Adatbázis-kezelés natív módon

Az adatbázis-kezelők használatához a PHP.INI-ben meg kell adni a megfelelő beállítást (vagy, mint például a MySQL-nél, eleve a támogatás a rendszer része), és programunkban használhatjuk az adatbázisra vonatkozó utasításokat.

A PHP az adatbáziskezelésre általában az alábbi utasításfajtákat tartalmazza.

- Az adatbázis-kezelő szerverrel kapcsolatot teremtő, a kapcsolatot lezáró parancsok
- Az adatbázisokra vonatkozó lekérdező és adminisztrációs parancsok
- Egy megnyitott adatbázisra vonatkozó adminisztrációs és lekérdező parancsok
- Az SQL lekérdezéseket átküldő parancs
- A lekérdezés eredményét feldolgozó parancsok.

A továbbiakban a MySQL utasításai alapján mutatom be az adatbáziskezeléssel kapcsolatos utasításokat. Más adatbáziskezelő esetén a változás általában csak annyi, hogy a névben az előtag más. Például:

```
Mysql_connect() //MySQL adatbázis-kezelő esetén
pg_connect()   //PostgresSQL
// a PHP5 esetén a mysql függvények helyett mysqli függvényeket kell használni.
```

Kapcsolódás egy adatbáziskezelőhöz, amely vagy lokális gépen helyezkedik el, vagy egy távoli, IP címmel meghatározott gépen. A kapcsolat létrehozása meg kell adni a kapcsolódási helyet (localhost, vagy IP cím), a kapcsolódó user-t, és a user jelszavát.

```
$link = mysql_connect(”localhost”,$username,$password);
```

vagy

```
$link = mysql_pconnect(”localhost”,$username,$password);
```

A fentiekhez egy kis magyarázat. A connect és pconnect között az a különbség, hogy a connect-tel megnyitott kapcsolat lezárul a kérdéses php oldal lefutása után, míg a pconnect kapcsolat nyitva marad (Persistent connection), és újbóli kapcsolódás során csak akkor kell egy viszonylag sokáig tartó kapcsolódási procedúrán átmennie a rendszernek, ha a kapcsolat nincsen nyitva. Egyébként a két függvény hazsnálata ugyanaz

A visszaadott érték a kapcsolat sorszáma. Ennek alapján lehet a továbbiakban azonosítani több megnyitott kapcsolat esetén, hogy éppen melyik kapcsolatot használjuk.

A kapcsolat bezárása:

```
mysql_close($kapcsolat)
```

```
<?php
    $link = mysql_connect ("kraemer", "marliesle", "secret") or die ("Could not connect");
    print ("Connected successfully");
    mysql_close ($link);
?>
```

Az adatbáziskezelőn lévő adatbázisokra vonatkozó parancsok

```
int mysql_list_dbs ([int link_identifier])
int mysql_create_db (string database name [, int link_identifier])
int mysql_drop_db (string database_name [, int link_identifier])
<?php
    $link = mysql_pconnect ("kron", "jutta", "geheim")
        or die ("Could not connect");
    if (mysql_create_db ("my_db")) {
```

```
        print ("Database created successfully\n");
    } else {
        printf ("Error creating database: %s\n", mysql_error ());
    }
?>
```

```
int mysql_select_db (string database_name [, int link_identifier])
```

Egy konkrét adatbázis tábláinak kilistázása

Egy adatbázis (lekérdezés) oszlopneveinek listázása. Ugyanez az utasítás a lekérdezés eredményének tetsz őleges adatait is kilistázza ...

```
object mysql_fetch_field (int result [, int result_type])
```

Mezőnevek kiírása, és tárolása a mezonev tömbben

```
<?php
$result=mysql_db_query($querydb,$query);
$m=mysql_num_rows($result);
$n=mysql_num_fields($result);
for($i=0;$i<$n;$i++)
{
 //Oszlopnév/mez    őnév bemásolása tömbbe később felhasználásra
 $mezok=mysql_fetch_field($result);
 $meznev[]=$mezok->name;
 //Oszlop/Mez    ő adatainak kilistázása.
     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>";
}
mysql_close();
?>
```

Egy SQL parancs elküldése

```
resource mysql_query ( string query [, resource link_identifier])
```

Az Adatbáziskezel őnek adott SQL, vagy egyéb utasítás eredményének átvétele a pufferb ől. Az eredmény általában két dimenziós tömbbe érkezik, amelyet vagy a tömbkezel ő függvények segítségével lehet feldolgozni, vagy indexel és ciklussal végig kell szaladni rajta. A visszaadott paraméterr ől mindig meg tudjuk mondani, hogy hány sorból áll. Az alábbi függvény megadja az eredmény sorainak számát. Ha nincs eredmény sor, akkor az értéke 0.

```
int mysql_num_rows()
```

Ha Delete, Update utasítások eredményére vagyunk kíváncsiak, akkor az alábbi függvényt kell használnunk.

```
int mysql_affected_rows ( [resource link_identifier])
```

Egy sort ad vissza egy tömbbe az alábbi függvény. A tömb öt sorszámokkal lehet indexelni, a sorszámozás 0-val kezdődik. Akkor használhatjuk, ha egy lekérdezésről nem tudjuk előre, hogy hány sort ad majd vissza.

```
array mysql_fetch_row([resource link_identifier])
```

Ennél a függvénynél szükségünk lehet a visszaadott oszlopok számára is. Ezt az alábbi függvénnyel tudjuk meg:

```
int mysql_num_fields ( [resource result])
```

Ez a függvény a sort szintén egy tömbben adja vissza, de a tömbelemek sorszámozva és a mez őnévvel is, mint asszociatív tömbindexxel elérhetők.

```
array mysql_fetch_array([resource link_identifier)
```

Az alábbi példa elküld egy lekérdezést, az eredmény oszlopneveit lekérdezi, majd megjeleníti táblázatos formában.

```
int mysql_fetch_object([resource link_identifier])
```

Az alábbi példa elküld egy lekérdezést, az eredmény oszlopneveit lekérdezi, majd megjeleníti táblázatos formában.

```
<?php
......
$qry = ”SELECT * FROM tabla”;
$result=mysql_db_query($qry,$query);
$m=mysql_num_rows($result);
$n=mysql_num_fields($result);
echo ”<TABLE><TR>”;
for($i=0;$i<$n;$i++)
{
 $mezok=mysql_fetch_field($result);
 echo "<TD>".$mezok->name."</TD>";
 $meznev[]=$mezok->name;
}
echo"</TR>";
// mezok kiirasanak vege //

// mezokben levo adatok kiirasa //
for($i=0;$i<$m;$i++)
{
   echo"<TR>";
 $adat=mysql_fetch_array($result);
 for($j=0;$j<$n;$j++)
 echo "<TD>".$adat[$meznev[$j]]."</TD>";
   echo"</TR>";
}
echo ”</TABLE>”;
?>
```

18.4 Tipikus feladatok adatbázis-kezelésnél

Egy adatbázis-kezelő használata során az alábbi tipikusnak mondható feladatokat kell megoldani PHP-ban, (de hasonlóképpen más programozási nyelveken is).

- Kapcsolódunk a szerveren lévő adatbázishoz.(láttunk mintát hozzá)
- Elküldünk egy SQL lekérdezést az adatbázis-kezelő programnak (Láttunk mintát)
- Az elküldött lekérdezésre kapott válasz általában egy táblázattal, vagy két-dimenziós tömbbel reprezentálható. Ezt szokás rekordszet-nek hívni. Egy ilyen rekordszetet megjelenítünk, célszer űen táblázatos formában. Ha a rekordszet túl sok sorból áll, akkor vagy a lekérdezéskor adunk olyan feltételeket, amely kevesebb sort ada eredményül, vagy megalkotjuk annak a lehet őségét, hogy a táblázat eredményét görgetni, vagy lapozni lehessen. (később általános mintát láttunk, az előző fejezetben bizonyos speciális eseteket néztünk)
- A rekordszet egyes sorait „űrlap” formájában meg akarjuk jeleníteni (később látunk mintát a fejezetben)
- Az űrlap vagy a rekordset eredményét ki akarjuk nyomtatni
- Az adatbázisba új adatot akarunk bevinni.
    - Egy rekordszethez illeszkedő új sorral
    - Egy rekordszethez illszkedő új sor-űrlap formájában bevitt adatokkal.
    - Minden adatbevitelnél valamilyen tipizálható ellen őrzési feladatot illik elvégezni, vagy a bevitt adatokat szintaktikailag és/vagy szemantikailag ellenőrizni illik.

    - Adatok bevitelekor vagy módosításkakor, amikor csak lehet az adatokat egy felkínált listából kelljen kiválasztani – a tévedések elkerülése miatt.

- Adatok módosítása lehetőség szerint az új adat bevitelével azonos formátumban és módon történjen.
- Egyes adatok, adatsorok, rekordok törlése, ellenőrzött módon.
- Az adatok bevitele vagy módosítása során, hibás adatfelvitel miatti ismétléskor a korábban bevitt vagy meglévő adat jelenjen meg a beviteli űrlapon.

A fenti feladatokhoz az alábbi nem igazán PHP-ben megvalósított fogalmakat társíthatunk:

Az adatok táblázatos megjelenéséhez az alábbi séma szerint érdemes eljárni abban az esetben, ha két dimenziós táblázatban kapom meg az adatokat.

```
Print( ”<table>”);
For($i=0;i<$maxsor;$i++){
 Print(”<tr>”);
 For($j=0;$j<$maxoszlop; $j++);{
  Print(”<td>);
  Print(adat[$i][$j]);
  Print(”</td>);
 }
 Print(”</tr>”);
}
print(”</table>”);
```

Természetesen a táblázat formázását, a szegélyeket, stb… mindenki a saját ízlése szerint alkothatja meg.

Ha az adatokat soronként vezsem át az adatbáézis-kezel őtől, akkor a fenti algoritmus kicsit megváltozik. Azt is megmutatom a következ ő példában, ha nem az oszlopok száma adott a lekérdezett rekordszetben, bár ezt is mindig meg lehet tudni egy adott esetben, hanem a mez őnevekkel, asszociatív módon tudok hivatkozni egy sorra. Erre az előző fejezetben láttunk példát.

```
<?php
//egyszerű rekord megjelenítése
$result=mysql_db_query($querydb,$query);
$sor=mysql_num_rows($result);
$n=mysql_num_fields($result);
print(”<table>”);
for($i=0;$i<$n;$i++)
{
 print(”<tr>”);
 //Oszlopnév/mez    őnév bemásolása tömbbe későbbi felhasználásra
 $mezok=mysql_fetch_field($result);
 $meznev[]=$mezok->name;

 //Oszlop/Mez    ő adatainak kilistázása.
      print(”<td align=right>”);
   print($meznev[$i]);   //A mez                      ő neve
  print(”</td>”)
      print(”<td>”);
   print($sor[$meznev]);   //A mez                    ő értéke
  print(”</td>”)
 print(”</tr>”);
}
print(”</table>”);
mysql_close();
?>
```

A fenti megjelenítés még elemi módomn sem alkalmas új adatok bevitelére, hiszen új adatokat WEB böngész ő esetén csakis FORM-okon keresztül vihetünk fel. Mindezek mellett meg kell oldanunk a mez ők ellen őrzését is. Ennek megfelel ően módosítanunk kell a fenti megoldást kissé. A példáról annyit, hogy egy űrlap jelenik meg, amely a megjeleníti a mez őket egy INPUT HTML tag segítségével, amivel a z értéket rögtön módosítani is lehet. A bevitt adatokat minden mez ő esetén a böngész ő javascript kódja rögtön ellen őrzi is (Hogy egyáltalán ki van-e töltve). Ha nincsen kitöltve, akkor hibaüzenet érkezik.

```
<HTML>
<HEAD>
</HEAD>
```

```
<SCRIPT LANGUAGE="JavaScript1.2" type="text/JavaScript1.2">
 function elenoriz()
 {
  var k=document.form1.text.value;
  if (k.length<1){
   alert("Nincs kitöltve az adat!!");
   return false;
  } else{
   return true;
  }
 }
</SCRIPT>
<BODY>

<?php
//egyszerű rekord megjelenítése
$result=mysql_db_query($querydb,$query);
$sor=mysql_num_rows($result);
$n=mysql_num_fields($result);
print(”<FORM name=”form1” ACTION=urlap.php method="get" ">
print(”<table>”);
for($i=0;$i<$n;$i++)
{
 print(”<tr>”);
 //Oszlopnév/mez    őnév bemásolása tömbbe későbbi felhasználásra
 $mezok=mysql_fetch_field($result);
 $meznev[]=$mezok->name;

 //Oszlop/Mez    ő adatainak kilistázása.
      print(”<td align=right>”);
   print($meznev[$i]);   //A mez                        ő neve
  print(”</td>”);
      print(”<td>”);
  print(”<INPUT name=\””.$meznev[$i].”\” type=\"text\" value=\””.$sor[$meznev].”\”
                 onchange=\”javascript:ellenoriz(”.$meznev[$i].”)\”>”);

   print($sor[$meznev]);   //A mez                     ő értéke
  print(”</td>”)
 print(”</tr>”);
}
 print(”<tr>”);
  print(”<td>”);
  print(”<INPUT name=\"OK\"  type=\"submit\" value=\"Oké\">”);
  print(”<INPUT name=\"Törlés\"  type=\"reset\"  value=\"Mégse\">”);
  print(”</td>”)
  print(”<td>”);

  print(”</td>”);
 print(”</tr>”);
print(”</table>”);

mysql_close();
?>
</BODY>
</HTML>
```

18.5 Adatbázis hibakezelés

A MySQL szerver a lekérdezések során hibakódot és szöveges üzenetet is küld vissza a kliensnek. A hiábk kezelésére az alábbi függvényewk használhatók:

```
string mysql_error ([int link_identifier])
```

A MySQL szerver által visszaadott hiba szövegesen

```
int mysql_errno ([int link_identifier])
```

A MySQL szerver által visszaadott hiba kódja.

```
<?php
 mysql_connect("marliesle");
 echo mysql_errno().": ".mysql_error()."<BR>";

 mysql_select_db("nincsadatbazis");
 echo mysql_errno().": ".mysql_error()."<BR>";

 $conn = mysql_query("SELECT * FROM nincsadattabla");
 echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

A fenti lehetőségek csak a MySQL-re vonatkoznak, de hasonlóan lehet használni őket más adatbázisok esetén is.

18.6 Adatbáziskezelés másképpen – absztrakciós rétegek, ADODB, ODBC

Fontos kérdés, hogy mit tegyünk akkor, ha el őre nem tudjuk, hogy milyen adatbáziskezel őt fogunk használni az oldalunkon, vagy a fejlesztés nem ugyanazon az adatbáziskezelőn folyik, mint ahol a program fog futni?

Az egyes adatbáziskezelők az SQL szintakszis szerint működnek, tehát ha készítünk egy olyan programcsomagot, amely eltakarja el őlünk az adatbáziskezel ők közti különbségeket, akkor tudunk erre alapozva olyan alkalmazást írni, amelynek mindegy, hogy melyik adatbáziskezel őt használja. Ez az **adatbázis absztrakciós réteg alkalmazásának módszere.**

18.6.1 Az absztrakciós réteg

A konkrét adatbáziskezel őtől független olyan utility csomag, amely eltakarja az adatbázis-kezel ők közötti különbségeket, kiegészíti azokat plusz funkciókkal kényelmesebbé teszi az adatbáziskezeléssel kapcsolatos programozást.

PHP program
Adatbázis típusa, kapcs.azonosító, SQL ↓ ↑ A lekérdezés eredménye
Absztrakciós réteg (pl. ADODB)
↓↑ ↓↑ ↓↑ ↓↑

MySQL driver	**PQ SQL driver**	**MSSQL driver**	**ODBC driver**

Felmerül a kérdés, hogy vajon nem, lassítja-e le nagyon a programjaink futását egy ilyen réteg alkalmazása. Megnyugtathatjuk a kedves olvasót, hogy bármiféle absztrakciós réteg lassítja a rendszert, de csak olyan csekély mértékben, hogy a lassulás észrevehetetlen, csak extrém nagy szerver terhelés, mellett észrevehető, ugyanakkor az alkalmazásfejlesztés nagyságrendekkel meggyorsul.

Ilyen egyszerű absztrakciós réteget többet is lehet találni az interneten:

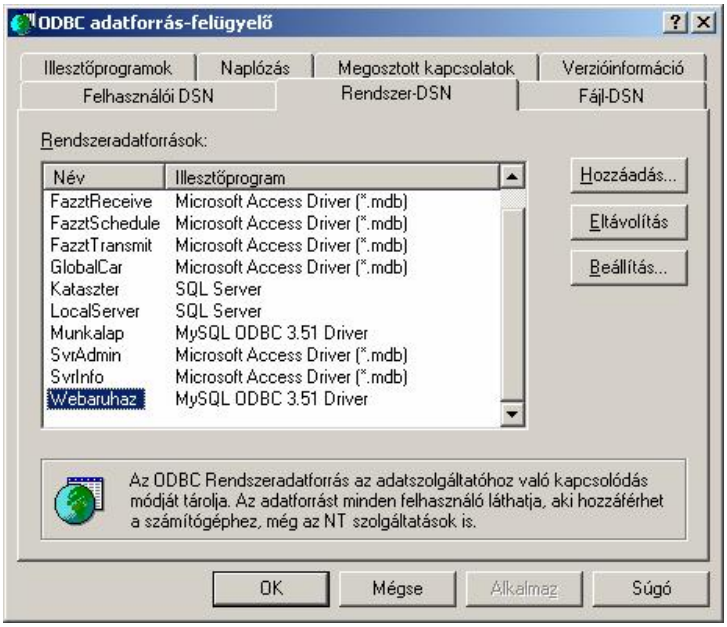
18.6.2 ODBC programcsomag

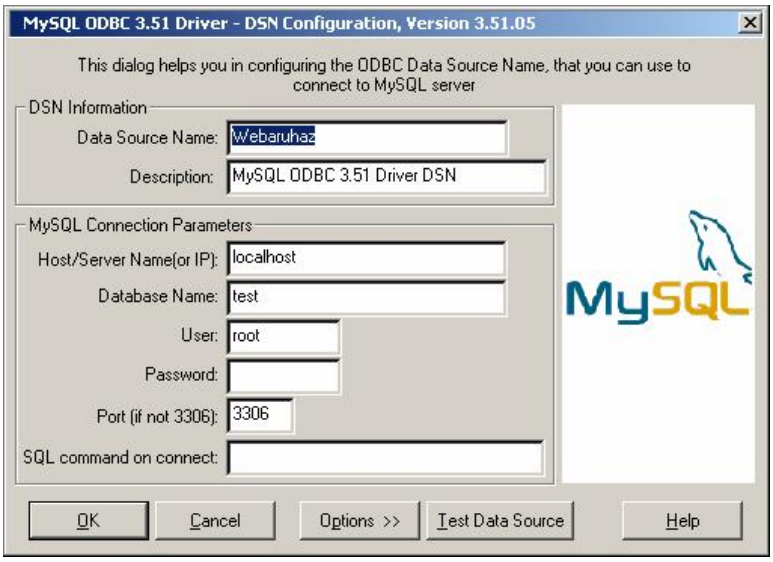
A Microsoft már korábban megvalósított egy adatbáziskezelés esetén jól használható absztrakciós eljárást, ez pedig az ODBC driverek csoportja.

Az ODBC driverek olyan meghajtók, amelyek a windows rendszerre telepített (szinte) tetsz őleges adatbáziskezelőt egységes módon érnek el. Ennek megfelelően az ilyen drivert használó alkalmazások a Windows rendszeren az SQL szerver módosításával is működnek, ODBC-n keresztül érik el őket.

Használatuk:

1. Létre kell hozni az adatbázist valamilyen adatbáziskezel övel: Egy windows rendszeren lehet, Access, MSSQL, MySQL, PostgresSQL, stb…, amelynek létezik ODBC drivere.
2. Létre kell hozni egy ODBC kapcsolatot. Ezt az ODBC kezel övel tehetjük meg. A különböz ő Windows verziók esetén ez máshol van, de alapvetően a Vezérlőpult környékén kell keresni „ODBC adatforrások” vagy hasonló néven. Az alábbiakban a Windows XP magyar verziójából vesszük a példát:





A fenti beállításokkal most egy MySQL adatforrást adtunk meg.

Ezek után az ODBC adatbáziskezel ő utasításokat kell használni a PHP-ban. Az ODBC csomag részletes leírását lásd az alábbi helyen: http://www.php.net/manual/hu/ref.odbc.php

18.6.3 Az ADODB réteg

Az ADODB eredetileg a Microsoft által kifejlesztett adatbáziskezelési programcsomag, amely egyesíti az SQL adatkezelés és a kliens oldali adatelérés el őnyeit. A lényege, hogy az adatbáziskezel őtől lekérdezésekkel kapott eredményeket – rekordszeteket – nem csak szekvenciálisan dolgozhatunk fel, hanem véletlen eléréssel bármelyik részét elérhetjük. Segítségével könnyen tudunk gördíthet ő menüket, listboxokat és ehhez hasonló objektumokat készíteni grafikus rndszerben.

Az php ADODB alkalmazáscsomag ehhez hasonló funkciókat valósít meg. Az ADOdb for PHP-hez legalább PHP 4.0.2 kell. Használatának előnyei:

- Könnyen használható a windowson programozóknak, mivel szintaktikája hasonlít a Microsoft féle ADODB-hez.
- A PHP natív adatbázis-kezelése felé egy réteget von, ami nem lassítja le lényegesen az adatbázis elérését, viszont tetszőleges adatbázison azonossá teszi az adatbáziskok kezelését.
- Nem csak lekérdezésekre, hanem Update és Insert utasításokra is jól használható.
- Támogatja a PHP4 sessionját.
- Saját magában definiált adattípusokkal dolgozik – ez is a különböző rendszerek kompatibilitását segíti elő.

Telepítése:

Az ADODB library-t letöltés után be kell tenni az adodb alkönyvtárba, vagy a PHP include library-jai közé. Ez utóbbi esetben a PHP.ini file-t meg kell szerkeszteni egy kicsit:

```
; Windows: "\path1;\path2"
include_path = ".;i:\phplibs;i:\phplibs\adodb;"
```

Ekkor az alábbi formát lehet használnunk, mivel a PHP eléri mindig az ADODB-t.

```
<?php
 include('adodb.inc.php');
 $db =    ADONewConnection('mysql');
 $db->debug = true;
 $db->   Connect($server, $user, $password, $database);
 $rs = $db->   Execute('select * from some_small_table');
 print_r($rs->   GetRows());
?>
```

Ha nem tudjuk library-be tenni, akkor egy alkönyvtárba tesszük az ADODB programcsomagot.

```
<?php
 include('adodb/adodb.inc.php');
 $db = &   ADONewConnection('mysql');
 $db->debug = true;
 $db->   Connect($server, $user, $password, $database);
 $rs = $db->   Execute('select * from some_small_table');
 print_r($rs->   GetRows());
?>
```

A scriptben az

```
include('adodb/adodb.inc.php');
```

utasítással mondjuk meg a scriptnek, hogy ADODB adatbázist fog kezelni a program. Az adatbázis típusát az

```
$conn = ADONewConnection('mysql') vagy a
$conn = &ADONewConnection(' mysql');
```

formát használhatjuk a kapcsolat létrehozására. Ennek hatására létrejön a $conn nev ű objektum, amely minden olyan paramétert és metódust tartalmaz, ami a későbbi kezeléshez szükséges. Amennyiben egy másik oldalon újra szükségünk van ennek az objektumnak az adataira, az objektumot célszerű lementeni sessionba, majd a következő oldal elején újra betölteni. Feltéve azt, hogy az objektum definició már lefusson, amikor a session elindul, tehát, ha van egy általános session kezelő rutincsomagunk, akkor a helyes includolási sorrend az alábbi:

```
include('adodb.inc.php');
include('session.php');
```

A továbbiakban nézzünk egy egyszerű példát, amely a Microsoft Acces-hez adott NorthWind adatbázisból egy lekérdezést hajt végre, végigmegy a rekordokon és ha dátum vagy idő típusú mezőt talál, akkor azt megfelelő formátumban írja ki.

```
<?php
include('adodb.inc.php');
$conn = &ADONewConnection('access'); # Kapcsolat létrehozása

$conn->PConnect('northwind');    # MS ACCESS-hez kapcsolódunk, northwind dsn-nel
$recordSet = &$conn->Execute('select CustomerID,OrderDate from Orders');
if (!$recordSet)
 print $conn->ErrorMsg();  # Hibaüzenet, ha nincsen eredménye a lekérdezésnek
else
while (!$recordSet->EOF) {
 $fld = $recordSet->FetchField(1); # Az első mező beolvasása
 $type = $recordSet->MetaType($fld->type);
 if ( $type == 'D' || $type == 'T')
  print $recordSet->fields[0].' '.
   $recordSet->UserDate($recordSet->fields[1],'m/d/Y').'<BR>';
 else
  print $recordSet->fields[0].' '.$recordSet->fields[1].'<BR>';

 $recordSet->MoveNext();
}
$recordSet->Close(); # optional
$conn->Close(); # optional
?>
```

A $conn ADO objektumhoz a

```
$conn->Pconnect('northwind')
```

metódus kapcsolja hozzá a Northwind adatbázist, ami természetesen elérhető a windowsos rendszerben.

Az adatok lekérdezését az

```
$recordSet = $conn->execute('select * from Orders') ;
```

metódus hajtja végre. Ez a $recordSet változóba visszatér egy ADO rekordszet objektummal. Ha valamilyen ok miatt nem jön létre a rekordszet, akkor azt le lehet kérdezni, és hibakezelést lehet végrehajtani. A rekordszetnek van kurzora. Ez a kurzor jelöli meg az aktuális rekordot. A kurzort

```
$recordSet->MoveNext()
```

metódussal tudom tovább mozgatni. A sorok közötti mozgást további parancsok is könnyítik:

`$recordSet->Move($n)` - az aktuális sorhoz képest n sorral ugrik tovább a kurzor

`$recordSet->MoveNext()` – következő sorra lép a kurzor

`$recordSet->MoveFirst()` – első sorra ugrik a kurzor

`$recordSet->MoveLast()` – utolsó sorra ugrik a kurzor

`$recordSet->AbsolutePosition (n)` – Egy abszolút módon megadott sorra ugrik

`$sorszám = $recordSet->CurrentRow ()` – Az aktuális sor számát adja vissza

A FetchField függvény teszteli le a mezők típusát. 3 elemű objektumot ad vissza.

- **name**: oszlop neve
- **type**: az eredeti mezőtípus
- **max_length**: a mező max. hossza. Néha nem ad vissza semmit (pl. MySQL)

A `MetaType`() lefordítja az eredeti típust általános ADODB típusra. Az alábbi *`generic`* típusok léteznek:

- **C**: szöveg, a <input type="text"> taggal jeleníthető meg
- **X**: TeXt, nagy szöveg, a <textarea> taggal jeleníthető meg
- **B**: Blobs, Binary Large Objects. Általában képek.
- **D**: Dátum

- **T**: Időbélyeg
- **L**: Logikai (boolean vagy bit mező)
- **I**: Egész
- **N**: Numerikus, illetve autoinkrementális.
- **R**: Sorozat típus. Lehet sorozat vagy, autoinkrement egész.

Dátum vagy Időbélyeg esetén a UserDate() függvény konvertálja és kiírja a megfelelő formátumú időt.

18.7 További absztrakció

Az adatbáziskezelés további egyszer űsítéseket igényelhet, különösen oktatási környezetben vagy egyszer űbb alkalmazások esetén. Valójában a legtöbb adatbázis művelet visszavezethető az alábbi adatbázis feladatokra:

1. Egy összetett lekérdezés, aminek az eredménye két dimenziós tömb
2. Új sor beszúrása egy meglévő táblába
3. Egy sor adatainak a módosítása egy táblában
4. Egy sor adatainak a törlése egy táblában
5. Egyéb SQL utasítások kiadása

Az 1. feladatkörnek a jellemző tulajdomnsága az, hogy két dimenziós tomb az eredménye.

A 2., 3., 4. utasítások eredménye egy szám. 0, ha nem sikerült, 1 vagy több, ha sikerült és a szám azt jelzi, hogy hány sorra hajtódott verge az utasítás.

A fenti feladatokat 2 függvénnyel meg lehet oldani. Az alábbiakra látunk egy példát azzal a kiegészítéssel, hogy az 1-es feladatnak a speciális esete az, hogy csak egy sort kapunk vissza eredményül, továbbá definiáltunk két példát arra, hogy hogyan kell egy tablet és egy tábla egy rekordját megjeleníteni.

```
<?php
/**************************
* ADODB interface és függvények
* Copyright by Fabian Zoltan
**************************/
require_once("adodb/adodb.inc.php");   //ADODB driver betöltése

  $ADO_CON =&ADONewConnection('access');
  $dsn = "Driver={Microsoft Access
Driver(*.mdb)};Dbq=D:/wwwroot/_tanitas/_lib/2002c.mdb;Uid=Admin;Pwd=;";
 $ADO_CON->PConnect($dsn);  //Adatbázishoz kapcsolódás
 $ADO_CON->debug = false;          //Az adatbázis lekérdezések debuggolása kikapcsolva

/**
 * @return Recordset
 * @param string $qrystr
 * @desc INSERT, UPDATE, DELETE kiadásakor érdemes használni
*/
function OpenRs(&$con,$qrystr){
 $rs = $con->Execute($qrystr);
   return $rs;
}

/**
 * @return unknown
 * @param ADOConnection $con
 * @param string $sql
 * @desc Egy lekérdezés teljes 2 dimenziós eredménytömbjét adja vissza
*/
Function ExecuteRSGetAll(&$con,$sql){
 $con->SetFetchMode(ADODB_FETCH_BOTH);

 $rs = $con->Execute($sql);
 $rsall = $rs->GetAll();
 if(!isset($rsall)){
  $rsall = array();
 }
 return $rsall;
}
```

```
/**
 * @return Megadott mezőt vagy a teljes sort adja vissza
 * @param Lekérdezés szövege $qry
 * @param keresett mező $sRet
 * @desc Végrehajt egy lekérdezést és egy sort vagy a sor egy oszlopát adja vissza.
*/
function ExecuteRSRow(&$con,$sql,$sRet=""){
 $row = array();
 $con->SetFetchMode(ADODB_FETCH_BOTH);
 $rs  = $con->Execute($sql);
 $row = $rs->FetchRow();
 $rs->close();
 if(empty($sRet)) return $row;
 else          return $row[$sRet];
}
/*
* Egy tömböt megjelenít táblázatos formában
*  @param $2D tömb
*/
Function 2dtable($t){
 $n = count($t);     //Sorok száma
 //
 print("<Table border='1'>\n");  //Tábla nyitó tag
 if($n>0){
  print("<tr>\n");    //egy sor nyitó html tag
  foreach($t[0] AS $j => $r){
   print("<td>".$j."</td>\n"); //Egy cella kiiratása
  }
  print("</tr>\n");   //Egy sor záró tag
 }

 for($i = 0; $i<$n;$i++){   //ciklus a sorokon végig
  $row = $t[$i];
  print("<tr>\n");    //egy sor nyitó html tag
  foreach($row AS $j => $r){
   print("<td>".$r."</td>\n"); //Egy cella kiiratása
  }
  print("</tr>\n");   //Egy sor záró tag
 }
 print("</table>\n");    //tábla záró tag
}
/**
* Egy rekord adatai oszlopos formában, űrlap formájában
* egy sor
* @paraméter 1 rekord tömb alakban
*/
Function egyrekord($r, $target="", $method="POST", $submit="Elküld"){
 print("<FROM method='".$method."' Name='".$method."' ".isset($target?$"ACTION='".$target."'":"").">");
 print("<TABLE>");
 foreach($r AS $i => $e){
   print("<TR>");
   print("<td>".$i."</TD>");
   print("<td><input type='text' name='".$i."' value='".$i."'></TD>");
   print("</TR>");
 }
 print("</TABLE>");
}
?>
```

-

18.8 Idő kezelése PHP-ben és MySQL-ben

Az időpontok kezelése a Unix hagyományokra épül. A Unix „időszámítás” kezdete 1970. január.01, 0 óra 0 perc. Az időt a Unix rendszerek innen számított másodpercekben mérik és tárolják. Ennek megfelelően a PHP és a MySQL is innen indul ki. A UNIX idő elnevezése TIMESTAMP. Az így kapott időpont azonban egy nagy egész szám, amit az adatok bevitelénél és kiiratásánál nemigen lehet használni, ezért léteznek konverziós függvények ilyen célokra.

A PHP esetén az aktuális időpontot a

string Date(formátumstring [,timestamp]) függvény állítja elő

A fromátumstringben az alábbi megjegyzések lehetnek:

Év – y két karakter (01), Y 4 karakter hosszú (2001)

Hónap - m – két karakter, sorszám 01, 02,…, M – a hónap angol neve (January, February, …)

Nap – d –két karakter, a hónapban a nap sorszáma, D – elnevezése angolul (Monday, Thuesday, …)

Óra – h – 12 órás ciklus 00-12, H – 24 órás ciklus két karakter 00 - 23

Perc – i – kt karakter 00-59

Az eredmény tehát string!

Az időpont átalakítása TIMESTAMP formátumra az alábbi függvénnyel lehetséges:

```
int mktime (int óra, int perc, int másodperc, int hónap, int nap, int év [, int is_dst])
```

Amennyiben nem akarunk kitölteni bizonyos értékeket, akkor 0-val kell feltölteni őket. Az utolsó paraméter a nappal értékét tartalmazza, általában elhagyható, sőt gyakorlatilag elhagyandó.

A MySQL esetén az időpont tárolása és formátuma hasonlóan kettős. Lehet tárolni időpontot TIMESTAMP formátumban. Ekkor a UNIX timestamp lesz a belső tárolási formátum, de a megjelenítésnél már karaktersorozatot kapunk vissza. Ahhoz, hogy a PHP-nek megfelelő Timestamp legyen egy lekérdezés eredménye, a MySQL TIMESTAMP()függvényét kell használni. Az alábbiakban egy ilyen megoldásra látunk példát.

```
Function koncertlista($datum)
{
 // A lekérdezés összeállítása
 $qry ="SELECT UNIX_TIMESTAMP(Mikor) AS Ido,Mi,Hol FROM esemeny WHERE Mikor > '".$datum."'";
 $result = @mysql_query($qry, $fc);
 $str ="<UL>";
 while ($a = mysql_fetch_object($result)) {
  $str .= "<LI>".date("Y.M.d, D H:i",$a->Ido).", ".$a->Mi."<BR>";
  $str .= $a->Hol."<BR>Belép        ő:".$a->Belepo." Ft</LI>";
 }
 $str .="</UL>";
echo $str;
return;
}

// Itt hívom meg az aktuális dátummal a függvényt. A formátum azért fontos, mert csak így
// hasonlítható össze a MySQL Timestamp-jével.
koncertlista(date("YmdHis"));

Az eredmény az alábbi lesz:
2002.Jan.12, Sat 09:12, Koncert
Uj várklub
Belépő:500 Ft
```

Az alkalmazott adattípus az SQL táblában a TIMESTAMP(14)

18.9 Sokáig futó programok

A PHP scriptek futásának a maximális ideje korlátozva van a PHP-ben, mert ha egy végtelen ciklust ír az ember, akkor lefagyaszthatja a szervert. A korlát default értéke 30 sec. Ha egy scriptünk mégis tovább futna, mert valamilyen oknál fogva a feldolgozás lassú, vagy esetleg egy adatbázisszervert ől vár adatokat, akkor célszer ű használnunk az alábbi függvényt a PHP oldalunk elején, mivel ez a kívánt ideig engedi futni a scriptet

```
Set_time_limit( másodperc)
```

Célszerűen csak akkora értéket írjunk be, hogy a script biztonságosan lefusson, ugyanakkor idejében ki is lépjen, ha kell. Ha időnek 0-t írunk, akkor soha nem jár le a script, és ha végtelen ciklust teszünk bele, akkor örökhajtós lesz a scriptünk.

19 File-ok, könyvtárak kezelése a szerveren és távoli URL-eken

Az alábbi fejezetben megnézzük, hogy melyek a leggyakrabban használt függvények és módszerek a PHP programokban. Amennyiben valaki úgy vélné, hogy más függvények gyakrabban használatosak az ő programjaiban, nagyon sajnálom, a help alapján kell megnézni a működésüket – sokszor én is ott nézem meg.

Az alábbi esetek fordulnak elő a filekezelésnél leggyakrabban a PHP programok esetén:

- Távoli szerverről szeretnénk olvasni, vagy távoli szerverre írni
- Helyi könyvtárszerkezetből szeretnénk olvasni, vagy ide szeretnénk írni. Az írásnál előfordulhat, hogy a kimenetet a helyi standard outputra szeretnénk tenni.
- A fenti eseményeket bináris vagy text file esetén is meg akarjuk valósítani.
- Helyi és távoli gépen is szeretnénk könyvtárakat létrehozni, vagy könyvtárakat törölni.
- Alkalmanként kell vizsgálnunk, hogy a kérdéses file a helyi filestruktúrában vagy távoli szerveren létezik-e?
- Egy megadott elérési útvonal file-ra, könyvtárra vagy link-re mutat-e?
- A fileok kezeléséhez jó, ha tudjuk az alábbiakat:
    - Az éppen aktuálisan futó alkalmazás lokális szerveren lévő elérési útvonalát a
        - $_SERVER[”PATH_TRANSLATED”] stringváltozó tartalmazza
    - A WEB szerver gyökérkönyvtárát a
        - $_SERVER[”SERVER_ROOT”] változó tartalmazza
    - Az operációs rendszer temporary könyvtárát
        - a $_ENV[”TEMP”] vagy hasonló nevű szuperglobális változó tartalmazza.

A file kezelésnél tundnunk kell, hogy az otthoni környezetben futtatott program eléri-e az éles szerveren is ugyanazt a könyvtárat. Ha egy web struktúrában lévő file-t kell elérnünk, akkor célszerű kiindulnunk az éppen futó alkalmazás könyvtárából. A fentiek alapján, ha az index.php alkalmazásunk az alábbi helyen fut:

C:\wwwroot\valami

És egy geza nevű alkönyvtárban lévő szoveg.txt file-ra szeretnénk hivatkozni, akkor az alábbi elérést kell alkalmazni:

```
$el = $_SERVER[”PATH_TRANSLATED”].”/geza/szoveg.txt”;
```

Az elérési útvonalak kezelése során gyakori, hogy szét kell választanom az elérési utat könyvtár, filenév illetve kiterjesztés részre. Az alábbi függvények a fenti elérési utat választják szét:

```
$path = ”C:/wwwroot/valami/geza/szoveg.txt”;

$file = basename ($path);        // $file értéke "szoveg.txt"
$file = basename ($path,".txt"); // $file értéke "szoveg", mivel a második paraméter hatására
       // a végér                          ől levágja a megadott stringet.
$dir  = dirname ($path);     // $d értéke ”C:\wwwroot\valami\geza" lesz.
```

Alternatív megoldás a fenti feladatra lehet a pathinfo függvény:

```
<?php
 $path_parts = pathinfo("/www/htdocs/index.html");
 echo $path_parts["dirname"] . "\n";
 echo $path_parts["basename"] . "\n";
 echo $path_parts["extension"] . "\n";
?>
```

Ha el akarjuk dönteni, hogy a filerendszer egy bejegyzése könyvtár, file, vagy egy link, akkor az alábbi függvényeket használhatjuk:

boolean is_dir($path);

boolean is_file($path);

boolean is_link($path);

Egy file további tulajdonságait – attributumait az alábbi függvényekkel kaphatjuk meg:

int is_executable($path)- Megmondja, hogy a bejegyzés futtatható-e

int is_readable($filename) – Megmondja, hogy a file olvasható-e

int is_writeable($filename) – Megmondja, hogy a file írható-e

int fileatime ($filename)- A file utolsó elérési ideje. A visszaadott idő UNIX időbélyeg formájában

int filemtime($filename) – A file utolsó módosításának ideje Unix időbélyeg formájában

int Fileowner($filename) – A file tulajdonosának ID-je. Csak lokális filerendszerben használható.

int fileperms($filename) – A filera vonatkozó jogokat adja vissza. Csak lokális filerendszerben használható.

int filesize($filename) - a file mérete

```
$filename = "C:/wwwroot/valami/geza/szoveg.txt";
echo filesize($filename);
```

A filekezelés meglehetősen hasonlít a C-ben megszokottakra:

Egy file olvasása előtt meg kell nyitnunk azt. Erre a célra az

$fp = fopen($filenev, mód)

függvényt használhatjuk, ahol a `$filenév` egy lokális filerendszer egy file-ja vagy egy URL-lel megadott file lehet. Ha windows rendszert használunk, akkor a filenévben az elérési útban szerepl ő könyvtárneveket a \\ jellel kell elválasztani.

```
$fp=fopen ("C:\\wwwroot\\index.htm","w");
```

Ha a $filenév a http:// vagy az ftp:// jelekkel kezdődik, akkor egy távoli szerveren lévő file-ról van szó.

```
$fp = fopen("http://www.szily.sulinet.hu/index.html","r");
```

A file megnyitásának módja az alábbi lehet:

mód	leírás	Példa
'r'	Olvasásra nyitom meg a file-t. File pointer a file elején áll	`$fp=fopen("C:\\autoexec.bat","r")`
'w'	Írásra nyitom meg a file-t. A file pointer a file elejére áll és ha volt ilyen file, akkor megsemmisül, zéró hosszúságúvá válik	`$fp=fopen("C:\\inetpub\\default.htm","w")`
'r+'	Írásra és olvasásra nyitom meg a file-t. A pointer a file elejére áll.	`$fp=fopen("C:\\inetpub\\adat.ini","r+")`
'w+'	Írásra és olvasásra nyitom meg a file-t. A pointer a file elejére áll. A file megnyitáskor 0 hosszúságúra csonkolódik	`$fp=fopen("C:\\inetpub\\adat.ini","w+")`
'a'	Írásra nyitom meg a file-t. A pointer a file végére áll.	`$fp=fopen("C:\\inetpub\\adat.ini","a")`
'a+'	Írásra és olvasásra nyitom meg a file-t. A pointer a file végére áll.	`$fp=fopen("C:\\inetpub\\adat.ini","a+")`
'b'	Ha biztosan bináris file-t akarok olvasni vagy írni, akkor a megnyitáskor a mód stringbe be kell írni ezt is.	`$fp=fopen("C:\\autoexec.bat","rb")`

A fenti utasítás hibaüzenetet ad, ha nem létezik az olvasásra megnyitni akart file. Ezért a hibakezeléshez az alábbiakat kell tenni:

```
    if (file_exists($File))
    // Létezik a file?
        {
        $fp = fopen($File,"r");
        // Megnyitjuk olvasásra
  .....
```

Mielőtt olvasni szeretnénk egy file-t, meg kell nézni, hogy létezik-e. A létezés ellenőrzésére a

Boolean file_exists($filename)

függvényt használhatjuk. A visszatérési értéke igaz, ha a file létezik, hamis egyébként.

Ha az fopen-nel megnyitottunk egy filet írásra és egy karaktert vagy byte információt akarunk beolvasni, akkor az alábbi függvényt használhatjuk

$c =fgetch($fp);

Ha az fopen-nel megnyitottunk egy filet írásra, és egy text file egy sorát akarjuk beolvasni, akkor az alábbi függvényt használjuk:

$sor = fgets($fp[,sorhossz]);

A sorhossz paraméter nem kötelező, ha nem tesszük ki, akkor alapértéke 1024 karaktert olvas a függvény.

Ha az fopen-nel megnyitottunk egy filet írásra és, ki akarom szedni a beolvasott fileból az esetlegesen benne lév ő html tag-okat, akkor ezt kell használni.

$sor=fgetss($fp[,sorhossz]);

Ha nem adjuk meg a sorhosszat, akkor max 1024 hosszú stringet olvas be.

Ha egy text file összes sorát be akarjuk olvasni egy tömbbe, akkor az alábbi függvényt kell használni:

$aline =file($Filename[,int include_path])

A függvény beolvassa a megadottt $Filename elérési útvonalú text file sorait egy $aline nev ű tömbbe. A függvénnyel nemcsak a lokális filerendszerb ől, hanem URL-lel megadott file-ból is lehet olvasni! Ha a file nem létezik, akkor hibaüzenet keletkezik, és a tömb elemszáma 0 marad. A kezelésére nézzük az alábbi példát a Help alapján:

```
<?php
// A beolvasandó file egy URL-lel van megadva.
$filename =”http://www.szily.hu/szily/szily.css”;
$asor = @file($filename);
if(count($asor==0)){
 die(”Nem létez   ő file”);
}

// A beolvasott file feldolgozása és a sorok kiiratása.
// A html sorokat kiírja, mint html szöveg, sor és sorszámot is kiír.
foreach ($asor as $line_num => $sor) {
    echo "#<b>{$line_num}</b> sor : " . htmlspecialchars($sor) . "<br>\n";
}

// Ez a példa egy file-t beolvas, majd a sorokat egy stringgé összevonja.
$html = implode ('', file ('http://www.szily.hu/szily/'));
?>
```

Ha egy teljes file-t akarunk beolvasni, akkor az alábbi függvényt kell használnunk:

$t=fread($fp,$filesize($filenev));

A filesize($filenev) függvény megadja az adott file méretét byte-okban és a paraméter hatására a fileméretnek megfelelő puffert foglal le a memóriában a PHP. A visszatérési érték változója tartalmazza majd a file összes byte-ját, mint string. Ez a függvény bináris olvasáskor használható igazán, mert a beolvasott tartalommal semmiféle konverzió nem történik.

Egy írásra vagy írásra/olvasásra megnyitott fileokba való írásnál alapvető függvény az

int fwrite($fp,$c);

Ha a kiirandó érték egy karakter, akkor karakterenként írunk, ha egy textfile egy sora, akkor a sorvége jelet hozzá kell tennünk a kiirandó tartalomhoz az alábbi módon:

```
$c=$sor,”\r\n”;
```

Ha egy file tartalmát akarjuk kiiratni, akkor a változó tartalmazza a teljes tartalmat.

Álljon itt filekezelésre egy komplexebb példa. Egy file-ból olvasnom kell, de ehhez először meg kell nyitnom.

```
<!-

Ez a program megvalósít egy egyszerű karakter alapú számlálót. A számláló
pillanatnyi  értéke  a counter.txt  állományban van, amit a szkript beolvas
értékét minden egyes beolvasás során megnöveli, a megnövelt értéket ugyan-
abba a fileb visszaírja, majd a számláló állását megjeleníti. Ahhoz, hogy
program megfelelően müködjön a counter.txt állományt létre kell hoznunk,
melynek tartalma csak '1' legyen, ezt a filet fel kell töltenünk ugyanabba
a könyvtárba, ahol a program van, és írási jogosultságot kell adnunk rá.

-->
<hr>

Látogatók száma:<BR>
<?php
    $File = 'counter.txt';
    // A számláló file neve
    if (file_exists($File))
    // Létezik a file?
        {
        $fp = fopen($File,"r");
        // Megnyitjuk olvasásra
        $num = fread($fp, filesize($File));
        // Beolvassuk a tartalmát a $num változóba.
        fclose($fp);
        // Zárjuk a filet.
        $num=$num+1;
        // A változó értékét megnöveljük
        $fp = fopen($File,"w");
        // Megnyitjuk ugyanazt a filet, de most írásra
        fwrite($fp, $num, 10);
        // Kiírjuk a változó értékét.
        fclose($fp);
        // Zárjuk a filet
        echo('<b>'.chop($num).'</b>');
        // Kiírjuk a HTML-kódba a változó értékét.
        }
?>
<hr>
```

20 Grafika

A PHP-ban a grafikai modult nem a képek egyszer ű kitételére használhatjuk, hanem a szerver oldalon képekkel végzett manipulációkra. A PHP grafikához szükséges, hogy a szerver oldalon be legyen töltve a GD.DLL vagy a GD_2.DLL (Windows szerver esetén). A GD_2.DLL támogatja a BMP, JPEG, GIF, és PNG képekkel végzett manipulációkat. Az alábbi mintában egy fotoalbum készít ő és használó programot mutatunk be. A program az első meghívásakor az aktuális könyvtárban lévő képekről kisképeket készít, majd táblázatos formában megjeleníti azokat. A programot objektum-orientált formában írtam meg.

```
<?php
/*
 Simple PhotoAlbum 2.2.1.2004.06.02
 (c) by Zoltán Fábián, 1999-2004

 Use the program free. email: fz@szily.hu

 This script works on this site: http://fz.szily.hu,
http://www.alarmsystem.hu/fotoalbum

 Használata

 include("../photoalbum.php");
*/

if(stristr($_SERVER["PHP_SELF"], __FILE__)) die("Nem hívhatod meg a file-t
közvetlenül!...");

if(!defined("THUMBNAIL_WIDTH")) define("THUMBNAIL_WIDTH",150);

define("SERVERTARGET",str_replace(".","_",$_SERVER["SERVER_NAME"]));
define("DOCUMENT_ROOT",$_SERVER["DOCUMENT_ROOT"].((
substr($_SERVER["DOCUMENT_ROOT"],-1)!="/") ? "/" :""));                                    //The
define("PHOTOSCRIPT",relpath(__FILE__)); //The path of this script
define("PHOTOALBUM","__photoalbum.txt");
define("DELIMITER","#");
define("CSV_OK",True);

if(!isset($col)) $col =  4;   // Number of columns
define("COLUMNS", $col);

if(!isset($slide)) $slide = 5;  // timeout of the sliding
define("SLIDE", $slide);

//Needed gd extension
DEFINE("GD",True);
if(!extension_loaded ("gd")) {
 if(ini_get("enable_dl")) {
  DEFINE("GD",dl("gd.dll"));
 }else{
  DEFINE("GD",False);
 }
}
// ---------- Image osztály -----------
class image {
 var $Name; // Picture picture
 var $Thumb; // Thumbnail name
 var $dx;  // size of picture
 var $dy;
 var $Text; // text under the picture

 function image($Entry="",$Txt="") {
  global $aktdir,$newThumbnails;
  if(strlen($Entry)>0) {
```

```
   $this->Name = $Entry;
   //get size of picture
   if(GD) {
    $size = GetImageSize($aktdir.$this->Name);
   }else{
    $size=array(800,600);
   }
   $this->dx = $size[0];
   $this->dy = $size[1];

   //Text under the pictures
   if (strlen($Txt)>0){
    $this->Text = $Txt;
   }else{
    $pos = explode(".",$Entry);
    $this->Text = str_replace("_"," ",$pos[0]);
   }
   //Is there thumbnail of image?
   $thn = $aktdir."tn_".$this->Name;
   $origDate = filemtime ( $aktdir.$this->Name);
   $isthn  = is_file($thn);
   $thDate = $isthn ? filemtime ( $thn): 0;
   //print($origDate ." ".$thDate."<BR>");
   if((!$isthn || ($origDate > $thDate ))&& GD ) {
    $this->NewThumbnail();
    $newThumbnails++;
   }else{
    $this->ReadThumbnail();
   }
  }else{
   $this->Name = "";
   $this->dx = 0;
   $this->dy = 0;
   $this->Text = "";
  }
 }

 //Make new thumbnails from GIF, PNG or JPG | JPEG |bmp!
 function NewThumbnail () {
  global $gd, $aktdir;
  //size of original big picture
  $size = GetImageSize($aktdir.$this->Name);
  $dx = $size[0];
  $dy = $size[1];

  $a = eregi("\.gif",$aktdir.$this->Name);
  if((!eregi("\.gif",$aktdir.$this->Name)) &&
    (($dx> THUMBNAIL_WIDTH) |($dy>THUMBNAIL_WIDTH)))
  {
   if (eregi("\.jpg|\.jpeg",$aktdir.$this->Name))
   {
    $im = ImageCreateFromJPEG ($aktdir.$this->Name);
    //Empty destination image
    $dst_im = @ImageCreateTrueColor( THUMBNAIL_WIDTH,
          THUMBNAIL_WIDTH*$dy /$dx);
   }
   if (eregi("\.png",$aktdir.$this->Name)){
    $im = @ImageCreateFromPNG ($pics->ImagePath.$this->Name);
    $dst_im = @ImageCreateTrueColor(THUMBNAIL_WIDTH,
          THUMBNAIL_WIDTH*$dy /$dx);
   }
   if (eregi("\.wbmp",$aktdir.$this->Name)){
    $im = @ImageCreateFromwbmp ($pics->ImagePath.$this->Name);
    $dst_im = @ImageCreateTrueColor(THUMBNAIL_WIDTH,
```

```
          THUMBNAIL_WIDTH*$dy /$dx);
   }
   if (eregi("\.bmp",$aktdir.$this->Name)){
    $im = @ImageCreateFromwbmp ($aktdir.$this->Name);
    $dst_im = @ImageCreateTrueColor(THUMBNAIL_WIDTH,
          THUMBNAIL_WIDTH*$dy /$dx);
   }

   // Image resize and copy from source to destination
   $a = imagecopyresized ($dst_im, $im, 0, 0, 0, 0, THUMBNAIL_WIDTH,
          THUMBNAIL_WIDTH*$dy/$dx, $dx, $dy);

   $this->Thumb = $aktdir."tn_".$this->Name;
   //New image on disk
   Imagejpeg ($dst_im, $this->Thumb);
  }else{
   $this->Thumb = $aktdir."tn_".$this->Name;
   copy($aktdir.$this->Name,$aktdir."tn_".$this->Name);
  }
 }

 //Thumbnail reading
 function ReadThumbnail(){
  if(GD) {
   $this->Thumb = "tn_".$this->Name;
  }else{
   $this->Thumb = $this->Name;
  }

 }
 //Write out a thumbnail into the table
 function ShowThumbnail() {
  $str=$this->GetString();
  print($str);
 }
 //get the string of a thumbnail
 function GetString() {
  global $css_td,$css_link,$pics;
  $str  = "<TD ".$css_td." ALIGN=\"CENTER\" VALIGN=\"top\">";
  $str .= "<a href='javascript:picturepopup(\"".
                  PHOTOSCRIPT."?getimg=".$this->Name."&picpath=".
                  $pics->ImageWebPath."\",\"".SERVERTARGET."\")' ".
                  $css_link.">\n";
  $str .= "<IMG SRC=\"".$this->Thumb."\" ALT=\"".$this->Name.
                "\"  border=\"0\" width=\"".THUMBNAIL_WIDTH."\" ><BR>".
                $this->Text."</a>\n";
  $str .= "</TD>";
  return $str;
 }
}

//-------- Image list class -----------
class ImageList {
 var $aPictures = array(); //Array of images
 var $ImageNumber;
 var $RowNumber;    //number of rows
 var $ImagePath;    //The directory of list
 var $ImageWebPath;   //The Path in the website
 var $CSV_OK;

 var $css_file;    //styles
 var $css_link;
 var $css_body;
 var $css_table;
```

```
var $css_tr;
var $css_td;
var $css_tag;

//It makes a list from the actual directory
function ImageList($path) {
 $this->newThumbnails = 0;
 $this->ImageNumber = 0;
 $this->ImagePath = $path;
 $this->ImageWebPath = substr(relpath($path),0,-1);

 if(CSV_OK){
   $aCSV = array();
  if (file_exists(PHOTOALBUM)){
   $row = 1;
   $handle = fopen (PHOTOALBUM,"r");
   while ($row = fgetcsv ($handle, 1000, DELIMITER)) {
       $aCSV[$row[0]] = $row[1];
   }
   fclose ($handle);
  }else{
   $handle=fopen(PHOTOALBUM,"w");
   fclose($handle);
  }
 }

 //Make an Image object
 $d=opendir($path);
 while($entry=readdir($d)){
  if ($this->is_image($entry)) {

   $text=$entry; //Handling CSS text
   if(CSV_OK){
    if(isset($aCSV)){
     if(isset($aCSV[$entry])){
      $text=$aCSV[$entry];
     }else{
      $handle = fopen(PHOTOALBUM,"a");
      fwrite($handle,$entry.DELIMITER.$entry."\n");
      fclose($handle);
     }
    }else{
      $handle = fopen(PHOTOALBUM,"a");
      fwrite($handle,$entry.DELIMITER.$entry."\n");
      fclose($handle);
    }
   }
   $this->aPictures[] = new Image($entry,$text);
   $this->ImageNumber++;
  }
 }
 closedir($d);
 sort($this->aPictures);
 $this->RowNumber = (int) ($this->ImageNumber / COLUMNS );
}
//Is Image this filename
function is_image($str){
 //This is the right soluton.
 $str="_".strtolower($str);
 //exclude thumbnails
 $pos = strpos ($str, "tn_");
 if ($pos ==1){
  return FALSE;
 }else{
```

```
   return eregi("\.gif|\.jpg|\.jpeg|\.bmp|\.png",$str);
  }
 }
 function css(){
  global $css_file,$css_link,$css_body,$css_table,$css_tr,$css_td,$css_tr;
  $this->css_file   = ( isset($_GET["css_file"]) )? $_GET["css_file"] :
(isset($css_file) ? $css_link : False);
  $this->css_link   = ( isset($_GET["css_link"]) )? $_GET["css_link"] :
(isset($css_link) ? $css_link : "style=\"font-family: Verdana,Arial, Helvetica,
sans-serif; font-size: x-small; color: #000000\"");
  $this->css_body   = ( isset($_GET["css_body"]) )? $_GET["css_body"] :
(isset($css_body) ? $css_body : "style=\"background-color: #666666;\"");
  $this->css_table  = ( isset($_GET["css_table"]))? $_GET["css_table"]:
(isset($css_table)? $css_table: "style=\"border:0;\"");
  $this->css_tr     = ( isset($_GET["css_tr"]   ))? $_GET["css_tr"]   :
(isset($css_tr)   ? $css_tr   : "style=\"vertical-align:top;\"");
  $this->css_td     = ( isset($_GET["css_td"]   ))? $_GET["css_td"]   :
(isset($css_td)   ? $css_td   : "align=\"center\"");
  $this->css_tag    = ( isset($_GET["css_tag"]  ))? $_GET["css_tag"]  :
(isset($css_tag)  ? $css_tag  : "class=\"kapcsolat\"");
 }
 //It makes a table from the pictures
 function ShowList() {
  $str=$this->GetString();
  print($str);
 }


 function GetString() {
  global $css_table,$css_tr,$css_td,$newThumbnails;
  $str  = picturepopup();
  $str .= "<TABLE ".$css_table.">";// Border=\"0\">\n\r";
  $str .= "<TR ".$css_tr.">\n\r";
  $j=0;
  while (list ($key, $ertek) = each ($this->aPictures)) {
   $str .=$this->aPictures[$key]->GetString();
   $j++;
   if ($j == COLUMNS){
    $str .="</TR>\n\r<TR>";
    $j =0;
   }
  }

  while(0<$j && $j<COLUMNS){
   $str .= "<TD ".$css_td."> </TD>";
   $j++;
  }
  $str .="</TR>\n\r";
  $str .="</TABLE>\n\r";
  if ($newThumbnails>0){
   $str .="<p align='center' style='color: #ff0000'> Found ".
                   $newThumbnails." new pictures and made thumbnails!</p>";
   $newThumbnails=0;
  }
  Return $str;
 }

  //Searches an image in array
 function Search($image) {
  $i =0;
  while( ($i<sizeof($this->aPictures) ) &&
                ($this->aPictures[$i]->Name != $image) )
        {
   $i++;
```

```
  }
  return $i;
 }

 function PreviousImage($idx){
  if($idx==0) $idx = sizeof($this->aPictures);
  $idx--;
  Return $idx;
 }

 function NextImage($idx) {
  if($idx==sizeof($this->aPictures)-1) $idx=-1;
  $idx++;
  Return $idx;
 }
}

class popup {
 var $Image;
 var $PrevImage;
 var $NextImage;
 var $dx; //width of popup
 var $dy; //height of popup
 var $dxx;  //width of picture
 var $dyy; //height of picture

 function popup($getimg){
  global $pics;
  $idx= $pics->Search($getimg);
  $this->Image      = new Image();
  $this->PrevImage  = new Image();
  $this->NextImage  = new Image();
  $this->Image      = $pics->aPictures[$idx];
  $this->PrevImage  = $pics->aPictures[$pics->PreviousImage($idx)];
  $this->NextImage  = $pics->aPictures[$pics->NextImage($idx)];
  $this->dxx        = ($this->Image->dx<600) ? (600) :($this->Image->dx);

  if ($this->dxx>1024) $this->dxx =  1024;
  $this->dyy        = ($this->Image->dy) *$this->dxx/$this->Image->dx;
  $this->dx         = $this->dxx + 28;
  $this->dy         = $this->dyy + 40;
 }

 function GetString(){
  global $pics,$isslide;
    //Javascript for show the picture
  $str  ="<HTML>\n";
  $str .="<HEAD>\n";
  $str .="<TITLE>Kep:".$this->Image->Name."</TITLE>\n";
  if($isslide) {
   $str .="<META HTTP-EQUIV=REFRESH
CONTENT=\"".SLIDE.";URL=".basename($_SERVER["PATH_TRANSLATED"])."?getimg=".urlenco
de($this->NextImage->Name)."&isslide=1\">";
  }
  $str .="</HEAD>\n";
  $str .= picturepopup();
  $str .="<BODY style=\"background-color: #666666;\" style=\"margin:0\">\n";

  $str .="<DIV ALIGN=CENTER>\n";
  $str .="<A HREF='javascript:picturepopup(\"".PHOTOSCRIPT."?getimg=".$this-
>PrevImage->Name."&picpath=".$pics->ImageWebPath."\",\"".SERVERTARGET."\")'
style=\"font-family: Verdana,Arial, Helvetica, sans-serif; font-size: x-small;
color: #000000\" ".$pics->css_tag.">[::Prev::</A>\n";
```

```
  $str .="<A HREF='javascript:picturepopup(\"".PHOTOSCRIPT."?getimg=".$this-
>NextImage->Name."&picpath=".$pics->ImageWebPath."\",\"".SERVERTARGET."\")'
style=\"font-family: Verdana,Arial, Helvetica, sans-serif; font-size: x-small;
color: #000000\" ".$pics->css_tag.">::Next::</A>\n";
  $str .="<A HREF='javascript:picturepopup(\"".PHOTOSCRIPT."?getimg=".$this-
>NextImage->Name."&picpath=".$pics-
>ImageWebPath."&isslide=1\",\"".SERVERTARGET."\")' style=\"font-family:
Verdana,Arial, Helvetica, sans-serif; font-size: x-small; color: #000000\"
".$pics->css_tag.">::Slide::</A>\n";
  $str .="<A HREF='javascript:picturepopup(\"".PHOTOSCRIPT."?getimg=".$this-
>Image->Name."&picpath=".$pics->ImageWebPath."\",\"".SERVERTARGET."\")',
style=\"font-family: Verdana,Arial, Helvetica, sans-serif; font-size: x-small;
color: #000000\" ".$pics->css_tag.">::Stop::</A>\n";
  $str .="<A HREF=\"Javascript:void()\" onClick=\"self.close()\" ".$pics-
>css_tag.">::Close::]</A>\n";
     $str .="<BR>\n";
     $str .="<IMG SRC=\"".$pics->ImageWebPath."/".$this->Image->Name."\"
width=\"".$this->dxx."\">\n";
     $str .="</DIV>\n";

     $str .="<SCRIPT language=\"JAVASCRIPT1.4\">\n";
     $str .="    dx = ".$this->dx.";\n";
     $str .="    dy = ".$this->dy.";\n";
     $str .="    if(self.OuterWidth < dx) {\n";
     $str .="         dy = (int)(dy * self.OuterWidth /dx);\n";
  $str .="      dx = self.innerWidth;\n";
     $str .="    }\n";
     $str .="     if(self.OuterHeight < dy) {\n";
     $str .="         dx = (int)(dx * self.OuterHeight /dy);\n";
     $str .="         dy = self.OuterHeight;\n";
  $str .="    }\n";
  $str .="    self.resizeTo(dx, dy)\n";
  $str .="</SCRIPT>\n";
     $str .="</BODY>\n";
     $str .="</HTML>\n";
     return $str;
 }

 function Show()
 {
  print($this->GetString());
 }

}
// ************ Some functions ************
function picturepopup(){
 $str  = "<SCRIPT LANGUAGE=\"JAVASCRIPT1.2\">\n";
 $str .= " function picturepopup(url,target){\n";
 $str .= "
 winpops=window.open(url,target,\"status=0,resizable=1,location=0,menubar=0,scr
ollbars=0,toolbar=0\");\n";
 $str .= "   }\n";
 $str .= "</SCRIPT>\n";
 return $str;
}

function CSS_File_search() {
 $lines = file ($_SERVER["PATH_TRANSLATED"]);
 $keresendo = "text/css";
 $csere = array("<",">","\"");
 $lnk = " ";
 $ok = False;
 while ((list ($kulcs, $ertek) = each ($lines)) && !$ok){
  $e = strtolower($ertek);
```

```
  $p = strpos($e,$keresendo);
  if($p>0) {
   $ok = True;
   $p1 = strpos($e,"href");
   $e1 = substr($e,$p1+6);
   $e2 = explode(" ",$e1);
   $lnk= substr($e2[0],0,-1);
  }
 }
 Return $lnk;
}

//Searching for the css style after the body tag
function CSS_Body_search() {
 $fname =$_SERVER["PATH_TRANSLATED"];
 $handle = fopen ($fname, "r");
 $contents = fread ($handle, filesize ($fname));
 fclose ($handle);
 $text  =  strtolower($contents);
 $text  = str_replace(" ", "", $text);
 $atags = array();
 $atags =  explode("<body",$text);
 if (sizeof($atags) >1 ){
  $lnk ="";
  $p = explode("\"",$atags[1]);
     $lnk = $p[0];
 }else {
  $lnk ="style=\"background-color: #666666\"";
 }
 Return $lnk;
}

function relpath($dir){
 $dir=str_replace("\\","/",$dir);
 $len  = strlen(DOCUMENT_ROOT);
 return substr($dir,$len-1);
}

function pathcomp($p){ return (( substr($p,-1)!= "/") ? ($p."/") : ($p) ); }
/************* end of functions ***********/

/******************
* The main program
****************/
$newThumbnail =0;

//*********************** Calling parameters **************
$isslide = ( isset($_GET["isslide"] ) )? $_GET["isslide"]  : (isset($isslide) ?
$isslide :False);
$getimg  = ( isset($_GET["getimg"]  ) )? $_GET["getimg"]   : (isset($getimg)
? $getimg  : ""); //The name of viewing picture

//The path of actual pictures
if (!isset($aktdir)){   //from calling script first
 if (isset($_GET["picpath"])){  //From somewhere
  $aktdir =substr(DOCUMENT_ROOT,0,-1).$_GET["picpath"];
 } else {        //Slide
  $aktdir = dirname($_SERVER["PATH_TRANSLATED"])."/";
 }
}
$aktdir=pathcomp($aktdir);
$pics = new ImageList($aktdir);

//******************* searching for css style settings *****************
```

```
if(!isset($pics->css_file)) $pics->css_file = CSS_File_search();
if(!isset($pics->css_link)) $pics->css_body = CSS_Body_Search();
//****************** main things ****************

if( !$getimg){
 $pics->Showlist();
}else{      // ************* show one Image *****************
 $p = new popup($getimg);
 $p->show();
}
?>
```

21 Beléptetés, jelszavak alkalmazása, titkosítás

Az alábbi forrást használtam fel a következő oldalak leírásához:

http://hotwired.lycos.com/webmonkey/00/05/index2a_page2.html?tw=programming

Amikor egy könyvtárat vagy html oldalt le akarunk védeni azonosító névvel és jelszóval, akkor el ször is meg kell szerveznünk a WEB szerver oldali védelmet. Ez a különböz ő WEB szerverek esetén mindig egy kicsit másképpen megy. Az Apache WEB szerver esetén is több lehet ség kínálkozik, de talán a következ ő a legegyszerűbb:

Tegyük fel, hogy a /wwwroot/ceg nevű könyvtárat szeretnénk levédeni:

A védett könyvtárban létre kell hozni a .htaccess nevű file-t. Ennek a tartalma:

```
AuthType Basic
AuthName "Vedett terület"
AuthUserFile /wwwroot/ceg/.htpasswd
AuthGroupFile /dev/null

<Limit GET POST PUT>
require valid-user
</Limit>
```

A szerveren be kell állítani a megfelelő könyvtárra az alábbi sorokat:

```
httpd.conf

<Directory "/wwwroot">   //itt a www gyökérkönyvtáráról van szó
 ...
 AllowOverride
 AuthConfig
 ...
</Directory>

// a későbbiek során be kell töltve lenniük az alábbi soroknak

LoadModule anon_auth_module modules/mod_auth_anon.so
LoadModule dbm_auth_module modules/mod_auth_dbm.so
LoadModule digest_auth_module modules/mod_auth_digest.so
LoadModule digest_module modules/mod_digest.so
LoadModule expires_module modules/mod_expires.so
LoadModule headers_module modules/mod_headers.so
LoadModule info_module modules/mod_info.so
LoadModule status_module modules/mod_status.so
LoadModule php4_module x:/php/sapi/php4apache.dll    #Ez itt a PHP4 miatt kell
```

Az alábbi könyvtárban /apache/bin le kell futtatni az alábbi parancsot:

```
 htpasswd /wwwroot/ceg/.htpasswd usernév
```

Ennek hatására létrejön egy text file, amiben a usernév és a hozzá tartozó jelszó található.

Ha több usert akarunk azonosíttatni, akkor létre kell hozni egy user groupot és abban a usereket definiálni. Például létrehozzuk a .htgroup nevű file-t, amiben definiáljuk a csoportot:

```
userek: bela geza user1 user2
```

Lefuttatjuk a jelszógenerálást többször

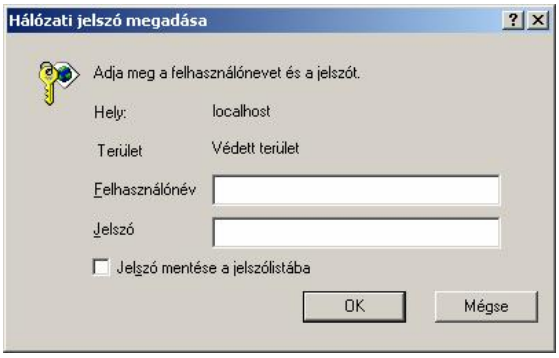
```
htpasswd /wwwroot/ceg/.htpasswd bela
htpasswd /wwwroot/ceg/.htpasswd geza
htpasswd /wwwroot/ceg/.htpasswd user1
htpasswd /wwwroot/ceg/.htpasswd user2
```

Módosítjuk a htaccess file tartalmát:

```
AuthUserFile /wwwroot/ceg/.htpasswd
AuthGroupFile /wwwroot/ceg/.htgroup
AuthName userek
AuthType Basic

<Limit GET>
require group dorks
</Limit>
```

Az adott könyvtár oldalának lekérésekor a szerver elküldi a böngész őnek a 401-es hibakódot, amely utasítja a böngészőt, hogy jelenítse meg az adatok bevitelére alkalmas oldalt párbeszédablakot.



Itt beviszi a felhasználó az adatokat, amit a szerver ellenőriz.

A fenti módszer használatához nincsen szükség PHP közrem űködésére, ugyanakkor nagy számú felhasználó esetén nem túl gyors a módszer, csak a szerveren lehet módosítani a userlistát és a jelszavakat.

A beléptetésre több megfelelő módszer kínálkozik a PHP esetén.

Ha modulként futtatjuk a PHP-t, akkor viszonylag egyszer űen biztosíthatjuk, hogy egy HTTP bekérést szimulálhassunk, csak az alábbi sorokat kell elküldeni a böngészőnek:

```
   header('WWW-Authenticate: Basic realm="Magánterület!"');
   header('HTTP/1.0 401 Unauthorized');
   echo 'Authorization Required.';
   exit;
```

Ennek hatására a $PHP_AUTH_USER és $PHP_AUTH_PWD változók fogják tartalmazni a bevitelnél adott választ. Ekkor sessionok segítségével minden oldalon felhasználhatjuk a változók értékeit.

Ha CGI módszerrel futtatjuk a PHP-t, akkor sajnos nem egyszer ű a fenti változók használata, ekkor gondoskodni kell arról, hogy minden oldal hívásakor a változó tartalma átadódjon.

A továbbiakban bemutatjuk két példán, hogy mi történik akkor, ha egy párbeszédablakban bekértük a két változó értékét.

Egy lehet őség, hogy létrehozzunk egy beviteli ablakot egy HTML form segítségével, majd POST metódussal elküldjük az eredményt a következő oldalnak.

```
<Form method=”POST” ACTION=”login.php”>
 <p>Kérem a usernevet: <INPUT Type=”text” NAME=”PHP_AUTH_USER” SIZE=60><BR>
 Kérem a jelszót: <INPUT TYPE =”PASSWORD” NAME=”PHP_AUTH_PWD”><BR>
 <INPUT TYPE=”SUBMIT” NAME=”OK”>
 <INPUT TYPE=”RESET” NAME=”RESET”>
</Form>
```

A login.php oldal leellen őrzi, hogy a megfelel ő usernév jelszó páros szerepel-e benne. Az ellen őrzés történhet adatbázisból vagy sima text file-ból is.

```
<?php
 // Megvizsgáljuk, hogy az el    őző form-ból kapott adatok között létezik-e
 //a $PHP_AUTH_USER változó és tartalmaz-e adatokat.
```

```
 if (!isset($PHP_AUTH_USER)) {

  // Ha nem létezik, a változó, akkor elküldünk egy fejlécet a böngész       őnek.
  header('WWW-Authenticate: Basic realm="Magánterület!"');
  header('HTTP/1.0 401 Unauthorized');
  exit;

 } else if (isset($PHP_AUTH_USER)) {

  // Ha nem üres a változó, akkor megnézzük, hogy az adatbázisban megvan-e
  // a megfelel       ő usernév jeszó páros
  // MySQL kapcsolatot hozunk létre. Az itteni usernév nem azonos a beléptet       ő userrel!
  mysql_connect("hostname", "username", "password")
   or die ("Unable to connect to database.");

  // Megnyitjuk a jelszóadatbázist
  mysql_select_db("Userlista") or die ("Unable to select database.");


  // Elküldjük a lekérdezést a users táblára

  $sql = "SELECT * FROM users
       WHERE username='$PHP_AUTH_USER' and password='$PHP_AUTH_PW'";

  // Végrehajtjuk, majd az eredmény a $result változóba kerül

  $result = mysql_query($sql);

  // Ha az eredménynek 0 sora van, akkor nincs ilyen user és jelszó páros,
  // ha 1, akkor van

  $num = mysql_numrows($result);
  if ($num != "0") {
   echo "<P>Isten hozott!</p>";
   exit;
  } else {
   header('WWW-Authenticate: Basic realm="Magánterület!"');
   header('HTTP/1.0 401 Unauthorized');
   echo 'Authorization Required.';
   exit;
  }
 }
?>
```

Hasonló megoldás, amikor egy text file-t használunk az authentikációhoz:

```
<?php

 // Megvizsgáljuk, hogy az el    őző form-ból kapott adatok között létezik-e
 //a $PHP_AUTH_USER változó és tartalmaz-e adatokat.

 if (!isset($PHP_AUTH_USER)) {

  // Ha nem létezik, a változó, akkor elküldünk egy fejlécet a böngész       őnek.
  header('WWW-Authenticate: Basic realm="Magánterület!"');
  header('HTTP/1.0 401 Unauthorized');
  exit;

 } else if (isset($PHP_AUTH_USER)) {

  // Ha nem üres a változó, akkor megnézzük, hogy egy text file-ban megvan-e
  // a megfelel       ő usernév jeszó páros. Beolvassuk a file-t a $file_tartalom nevű
  // változóba

  $filename = "/wwwroot/ceg/file.txt";
  $fp = fopen($filename, "r");
  $file_tartalom = fread($fp, filesize($filename));
  fclose($fp);

  // A text file sorait egy $sor nev       ű tömbbe visszük be.
  //A sorok végét a \n  karakter jelzi.

  $sor = explode("\n", $file_tartalom);
```

```
  // Végignézzük a $sor tombot, hogy a megfelel       ő értéket megtaláljuk-e benne
  // A sor minden elemét két részre bontjuk, usernév és jelszó párra.
  // Az elválasztó jel a :
  // A következ       őben a keresés programozási tételt alkalmazzuk.

  $i = 0;
  $adatok = explode(":", $sor[$i]);

  while($i<=sizeof($sor))&&
   !(($adatok[0]=="$PHP_AUTH_USER")
    &&($adatok[1]== "$PHP_AUTH_PW"))
  {
   $data_pair = explode(":", $sor[$i]);
   i++;
  }
  if ($i<=sizeof($sor)) $auth = 1;
  else      $auth = 0;


  if ($auth == "1") {
   echo "<P>Isten hozott!</p>";
   exit;

  } else {
   header('WWW-Authenticate: Basic realm="Magánterület!"');
   header('HTTP/1.0 401 Unauthorized');
   echo 'Authorization Required.';
   exit;
 }
?>
```

A fenti rutinokat kissé módosítva beírhatjuk őket egy include file-ba. Ha minden php oldalon beszúrjuk a file-t, akkor automatikusan az oldal bekérésekor lefut és ellen őrzi a jogosultságot. Mindenesetre ekkor biztosítani kell, hogy átlépve a következő oldalra a $PHP_AUTH_USER és $PHP_AUTH_PWD változók tartalma megmaradjon és elérhető legyen.

Egy harmadik és talán elég jól használható megoldás. Alapja, hogy egy scriptet minden védett oldal elejére beszúrunk, ami az oldal hívásakor lefut és SESSION változóban tároljuk a beléptetés eredményét, illetve a beléptetett user-t.

```
$name ="";
$pwd ="";
Session_start();
if(!isset($_SESSION["logged_in"])) $_SESSION["logged_in"] = False;

If (!$_SESSION["logged_in"]){

 if( isset($_POST["name"]) && isset($_POST["pwd"])){
  //Ha kés        őbb login formot használunk POST-tal
  $name = $_POST["name"];
  $pwd  = $_POST["pwd"];

 //Ha a Böngész    ő authhentikációs ablakát használjuk
 // if (isset($PHP_AUTH_USER) && isset ($PHP_AUTH_PW)){
 //  $name = $PHP_AUTH_USER;
 //  $pwd  = $PHP_AUTH_PW;
 //

  //A jelszó és usernév ellen        őrzése adatbázisból
  mysql_connect("hostname", "username", "password")
   or die ("Unable to connect to database.");

  // Megnyitjuk a jelszóadatbázist
  mysql_select_db("Userlista") or die ("Unable to select database.");

  // Elküldjük a lekérdezést a users táblára
  $sql = "SELECT * FROM users
       WHERE username='$name' and password='$pwd'";

  // Végrehajtjuk, majd az eredmény a $result változóba kerül
  $result = mysql_query($sql);

  // Ha az eredménynek 0 sora van, akkor nincs ilyen user és jelszó páros,
  // ha 1, akkor van

  $num = mysql_numrows($result);

  if($num>0){ //Ide kell betenni a jelszóellen            őrzést
   $_SESSION["logged_in"] = True; //vagy sima text vagy mysql lekérdezés stb...
  }
 }
}

If (!($_SESSION["logged_in"])){
 //Vagy egy login formot használunk, ekkor POST-tal küldjük el a megfelel    ő értékeket
 include ("login.html");

 // vagy szimuláljuk a WEBböngész    ő beléptet ő oldalát.
 //header('WWW-Authenticate: Basic realm="Beléptetés!"');
 //header('HTTP/1.0 401 Unauthorized');
 exit;
}
```

21.1 Titkosított átvitele a kliens és a szerver között: MD5()

A PHP-ban van egy jól használható függvény, amellyel adatok módosítatlanását tudjuk vizsgálnialapvet ő titkosítását tudjuk elvégezni, illetve a titkosított adatokat tudjuk visszakódolni plain text-é. Az adatoknak karakter formátumúaknak kell lenni. A függvény az

String md5(string)

```
/*
 *  md5.jvs 1.0b 27/06/96
 *
 * Javascript implementation of the RSA Data Security, Inc. MD5
 * Message-Digest Algorithm.
```

```
 *
 * Copyright (c) 1996 Henri Torgemane. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for any purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies.
 *
 * Of course, this soft is provided "as is" without express or implied
 * warranty of any kind.
 *
 * $Id: md5.js,v 1.2 1998/11/22 14:27:42 sas Exp $
 *
 */


function array(n) {
  for(i=0;i<n;i++) this[i]=0;
  this.length=n;
}

/* Some basic logical functions had to be rewritten because of a bug in
 * Javascript.. Just try to compute 0xffffffff >> 4 with it..
 * Of course, these functions are slower than the original would be, but
 * at least, they work!
 */

function integer(n) { return n%(0xffffffff+1); }

function shr(a,b) {
  a=integer(a);
  b=integer(b);
  if (a-0x80000000>=0) {
    a=a%0x80000000;
    a>>=b;
    a+=0x40000000>>(b-1);
  } else
    a>>=b;
  return a;
}

function shl1(a) {
  a=a%0x80000000;
  if (a&0x40000000==0x40000000)
  {
    a-=0x40000000;
    a*=2;
    a+=0x80000000;
  } else
    a*=2;
  return a;
}

function shl(a,b) {
  a=integer(a);
  b=integer(b);
  for (var i=0;i<b;i++) a=shl1(a);
  return a;
}

function and(a,b) {
  a=integer(a);
  b=integer(b);
  var t1=(a-0x80000000);
  var t2=(b-0x80000000);
  if (t1>=0)
    if (t2>=0)
      return ((t1&t2)+0x80000000);
    else
      return (t1&b);
  else
    if (t2>=0)
      return (a&t2);
    else
      return (a&b);
```

```
}

function or(a,b) {
  a=integer(a);
  b=integer(b);
  var t1=(a-0x80000000);
  var t2=(b-0x80000000);
  if (t1>=0)
    if (t2>=0)
      return ((t1|t2)+0x80000000);
    else
      return ((t1|b)+0x80000000);
  else
    if (t2>=0)
      return ((a|t2)+0x80000000);
    else
      return (a|b);
}

function xor(a,b) {
  a=integer(a);
  b=integer(b);
  var t1=(a-0x80000000);
  var t2=(b-0x80000000);
  if (t1>=0)
    if (t2>=0)
      return (t1^t2);
    else
      return ((t1^b)+0x80000000);
  else
    if (t2>=0)
      return ((a^t2)+0x80000000);
    else
      return (a^b);
}

function not(a) {
  a=integer(a);
  return (0xffffffff-a);
}

/* Here begin the real algorithm */

    var state = new array(4);
    var count = new array(2);
 count[0] = 0;
 count[1] = 0;
    var buffer = new array(64);
    var transformBuffer = new array(16);
    var digestBits = new array(16);

    var S11 = 7;
    var S12 = 12;
    var S13 = 17;
    var S14 = 22;
    var S21 = 5;
    var S22 = 9;
    var S23 = 14;
    var S24 = 20;
    var S31 = 4;
    var S32 = 11;
    var S33 = 16;
    var S34 = 23;
    var S41 = 6;
    var S42 = 10;
    var S43 = 15;
    var S44 = 21;

    function F(x,y,z) {
 return or(and(x,y),and(not(x),z));
    }

    function G(x,y,z) {
 return or(and(x,z),and(y,not(z)));
    }
```

```
   function H(x,y,z) {
return xor(xor(x,y),z);
   }

   function I(x,y,z) {
return xor(y ,or(x , not(z)));
   }

   function rotateLeft(a,n) {
return or(shl(a, n),(shr(a,(32 - n))));
   }

   function FF(a,b,c,d,x,s,ac) {
       a = a+F(b, c, d) + x + ac;
a = rotateLeft(a, s);
a = a+b;
return a;
   }

   function GG(a,b,c,d,x,s,ac) {
a = a+G(b, c, d) +x + ac;
a = rotateLeft(a, s);
a = a+b;
return a;
   }

   function HH(a,b,c,d,x,s,ac) {
a = a+H(b, c, d) + x + ac;
a = rotateLeft(a, s);
a = a+b;
return a;
   }

   function II(a,b,c,d,x,s,ac) {
a = a+I(b, c, d) + x + ac;
a = rotateLeft(a, s);
a = a+b;
return a;
   }

   function transform(buf,offset) {
var a=0, b=0, c=0, d=0;
var x = transformBuffer;

a = state[0];
b = state[1];
c = state[2];
d = state[3];

for (i = 0; i < 16; i++) {
    x[i] = and(buf[i*4+offset],0xff);
    for (j = 1; j < 4; j++) {
 x[i]+=shl(and(buf[i*4+j+offset] ,0xff), j * 8);
    }
}

/* Round 1 */
a = FF ( a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
d = FF ( d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
c = FF ( c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
b = FF ( b, c, d, a, x[ 3], S14, 0xc1bdceee); /* 4 */
a = FF ( a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
d = FF ( d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
c = FF ( c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
b = FF ( b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
a = FF ( a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
d = FF ( d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
c = FF ( c, d, a, b, x[10], S13, 0xffff5bb1); /* 11 */
b = FF ( b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
a = FF ( a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
d = FF ( d, a, b, c, x[13], S12, 0xfd987193); /* 14 */
c = FF ( c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
b = FF ( b, c, d, a, x[15], S14, 0x49b40821); /* 16 */
```

```
/* Round 2 */
a = GG ( a, b, c, d, x[ 1], S21, 0xf61e2562); /* 17 */
d = GG ( d, a, b, c, x[ 6], S22, 0xc040b340); /* 18 */
c = GG ( c, d, a, b, x[11], S23, 0x265e5a51); /* 19 */
b = GG ( b, c, d, a, x[ 0], S24, 0xe9b6c7aa); /* 20 */
a = GG ( a, b, c, d, x[ 5], S21, 0xd62f105d); /* 21 */
d = GG ( d, a, b, c, x[10], S22,  0x2441453); /* 22 */
c = GG ( c, d, a, b, x[15], S23, 0xd8a1e681); /* 23 */
b = GG ( b, c, d, a, x[ 4], S24, 0xe7d3fbc8); /* 24 */
a = GG ( a, b, c, d, x[ 9], S21, 0x21e1cde6); /* 25 */
d = GG ( d, a, b, c, x[14], S22, 0xc33707d6); /* 26 */
c = GG ( c, d, a, b, x[ 3], S23, 0xf4d50d87); /* 27 */
b = GG ( b, c, d, a, x[ 8], S24, 0x455a14ed); /* 28 */
a = GG ( a, b, c, d, x[13], S21, 0xa9e3e905); /* 29 */
d = GG ( d, a, b, c, x[ 2], S22, 0xfcefa3f8); /* 30 */
c = GG ( c, d, a, b, x[ 7], S23, 0x676f02d9); /* 31 */
b = GG ( b, c, d, a, x[12], S24, 0x8d2a4c8a); /* 32 */

/* Round 3 */
a = HH ( a, b, c, d, x[ 5], S31, 0xfffa3942); /* 33 */
d = HH ( d, a, b, c, x[ 8], S32, 0x8771f681); /* 34 */
c = HH ( c, d, a, b, x[11], S33, 0x6d9d6122); /* 35 */
b = HH ( b, c, d, a, x[14], S34, 0xfde5380c); /* 36 */
a = HH ( a, b, c, d, x[ 1], S31, 0xa4beea44); /* 37 */
d = HH ( d, a, b, c, x[ 4], S32, 0x4bdecfa9); /* 38 */
c = HH ( c, d, a, b, x[ 7], S33, 0xf6bb4b60); /* 39 */
b = HH ( b, c, d, a, x[10], S34, 0xbebfbc70); /* 40 */
a = HH ( a, b, c, d, x[13], S31, 0x289b7ec6); /* 41 */
d = HH ( d, a, b, c, x[ 0], S32, 0xeaa127fa); /* 42 */
c = HH ( c, d, a, b, x[ 3], S33, 0xd4ef3085); /* 43 */
b = HH ( b, c, d, a, x[ 6], S34,  0x4881d05); /* 44 */
a = HH ( a, b, c, d, x[ 9], S31, 0xd9d4d039); /* 45 */
d = HH ( d, a, b, c, x[12], S32, 0xe6db99e5); /* 46 */
c = HH ( c, d, a, b, x[15], S33, 0x1fa27cf8); /* 47 */
b = HH ( b, c, d, a, x[ 2], S34, 0xc4ac5665); /* 48 */

/* Round 4 */
a = II ( a, b, c, d, x[ 0], S41, 0xf4292244); /* 49 */
d = II ( d, a, b, c, x[ 7], S42, 0x432aff97); /* 50 */
c = II ( c, d, a, b, x[14], S43, 0xab9423a7); /* 51 */
b = II ( b, c, d, a, x[ 5], S44, 0xfc93a039); /* 52 */
a = II ( a, b, c, d, x[12], S41, 0x655b59c3); /* 53 */
d = II ( d, a, b, c, x[ 3], S42, 0x8f0ccc92); /* 54 */
c = II ( c, d, a, b, x[10], S43, 0xffeff47d); /* 55 */
b = II ( b, c, d, a, x[ 1], S44, 0x85845dd1); /* 56 */
a = II ( a, b, c, d, x[ 8], S41, 0x6fa87e4f); /* 57 */
d = II ( d, a, b, c, x[15], S42, 0xfe2ce6e0); /* 58 */
c = II ( c, d, a, b, x[ 6], S43, 0xa3014314); /* 59 */
b = II ( b, c, d, a, x[13], S44, 0x4e0811a1); /* 60 */
a = II ( a, b, c, d, x[ 4], S41, 0xf7537e82); /* 61 */
d = II ( d, a, b, c, x[11], S42, 0xbd3af235); /* 62 */
c = II ( c, d, a, b, x[ 2], S43, 0x2ad7d2bb); /* 63 */
b = II ( b, c, d, a, x[ 9], S44, 0xeb86d391); /* 64 */

state[0] +=a;
state[1] +=b;
state[2] +=c;
state[3] +=d;

   }

   function init() {
count[0]=count[1] = 0;
state[0] = 0x67452301;
state[1] = 0xefcdab89;
state[2] = 0x98badcfe;
state[3] = 0x10325476;
for (i = 0; i < digestBits.length; i++)
    digestBits[i] = 0;
   }

   function update(b) {
var index,i;

index = and(shr(count[0],3) , 0x3f);
```

```
 if (count[0]<0xffffffff-7)
   count[0] += 8;
        else {
   count[1]++;
   count[0]-=0xffffffff+1;
          count[0]+=8;
        }
 buffer[index] = and(b,0xff);
 if (index  >= 63) {
     transform(buffer, 0);
 }
    }

    function finish() {
 var bits = new array(8);
 var padding;
 var i=0, index=0, padLen=0;

 for (i = 0; i < 4; i++) {
     bits[i] = and(shr(count[0],(i * 8)), 0xff);
 }
        for (i = 0; i < 4; i++) {
     bits[i+4]=and(shr(count[1],(i * 8)), 0xff);
 }
 index = and(shr(count[0], 3) ,0x3f);
 padLen = (index < 56) ? (56 - index) : (120 - index);
 padding = new array(64);
 padding[0] = 0x80;
        for (i=0;i<padLen;i++)
   update(padding[i]);
        for (i=0;i<8;i++)
   update(bits[i]);

 for (i = 0; i < 4; i++) {
     for (j = 0; j < 4; j++) {
  digestBits[i*4+j] = and(shr(state[i], (j * 8)) , 0xff);
     }
 }
    }

/* End of the MD5 algorithm */

function hexa(n) {
 var hexa_h = "0123456789abcdef";
 var hexa_c="";
 var hexa_m=n;
 for (hexa_i=0;hexa_i<8;hexa_i++) {
   hexa_c=hexa_h.charAt(Math.abs(hexa_m)%16)+hexa_c;
   hexa_m=Math.floor(hexa_m/16);
 }
 return hexa_c;
}


var ascii="01234567890123456789012345678901" +
          " !\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ"+
          "[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~";

function MD5(entree)
{
 var l,s,k,ka,kb,kc,kd;

 init();
 for (k=0;k<entree.length;k++) {
   l=entree.charAt(k);
   update(ascii.lastIndexOf(l));
 }
 finish();
 ka=kb=kc=kd=0;
 for (i=0;i<4;i++) ka+=shl(digestBits[15-i], (i*8));
 for (i=4;i<8;i++) kb+=shl(digestBits[15-i], ((i-4)*8));
 for (i=8;i<12;i++) kc+=shl(digestBits[15-i], ((i-8)*8));
 for (i=12;i<16;i++) kd+=shl(digestBits[15-i], ((i-12)*8));
 s=hexa(kd)+hexa(kc)+hexa(kb)+hexa(ka);
 return s;
```

```
}
```

22 Biztonság, tippek és trükkök

Egy nyilvános szerveren futó PHP program futtatásának a biztonságossá tétele rendkívül fontos dolog, és két tényezőn, a szerver beállításain és a programozó gondosságán is múlik.

Bemenő adatok ellenőrzése és szűrése

A bemenő adatokat két oldalon lehet vagy kell ellenőrizni.

- A user oldalon a formok adatbevitelének felprogramozásával

- A szerver oldalon a már elküldött adatok érvényességének vizsgálatával

Az első változat arra való csakis és kizárólag, hogy a felhasználó user hibáit kiküszöböljk illetve minimalizáljuk.

Az adatbevitel általában egy ormon keresztül zajlik, tehát a formon való adatbevitelt célsze ű ellen őrizni. Erre a célra általában Javascript kódot használhatunk, amely például akkor fut le,

- ha a Form submit gombját megnyomjuk – ekkor a Form onSubmit eseményére indulhat el a Javascript kód,
- vagy minden egyes mez ő értékének változásakor lefuttatunk egy ellen őrző rutint – ekkor a mez ő onChange eseményére indulhat a javascript kód.

Mely mezőket kell ellenőrizni? Gondos programozással csak néhányat.

A HTML Formoknál az alábbi beviteli mezők vannak:

- Input mező Text – ellenőrizni kell

TEXTarea – ellenőrizni kell

Checkbox – ellenőrizni lehet

RadioButton – célszerű ellenőrizni

Select menü – a kiválasztottságot célszerű ellenőrizni

File mező – Igazából a file létezését ellene ellen őrizni, de nem illik. Csak annyit leet, hogy a file mez őben van-e valamiféle bejegyzés.

A fentiek alapján llátszik, hogy egyszer ű esetben csak a text és a textarea ellen őrzése igazán fontos. Ott ellenőrizhetjük a string hosszát, azt hogy bizonyos karakterek benne vannak-e vagy nincsenek. Mevizsgálhatjuk, hogy a bevitt adat numerikus-e, és ha igen, akkor milyen tartományba esik.

A szerver oldalon már ellenőrzött adatot kapunk, miért kell a szerver oldalon ismételten ellenőrizni?

A a kliens oldali ellen őrzés sokkal gyorsabb, továbbá a userek sanda szándékait a szerver oldalon lehet csak elkapni.

Itt is minden fenti dolgot ellenrizhetünk, azaz:

- Egy változó értékét
- Egy változó formáját (milyen hosszú, milyen karakterek vannak bene vagy hiányoznak). Itt kell kisz űrni az esetlegesen bevitt HTML tag-eket, a ” és a ’ jeleket, az esetlegesen bevitt SQL utasításneveket (SQL injection).
- Az adat numerikus-e, és ha igen, akkor megfelelő-e az értéke.
- Ez azt jelenti, hogy itt már a változók létét, a formátumát, és értékét kell csak ellen őrizni. Itt kell felkészülni esetleges sanda szándékú userek datbevitelére is.

File-k elérésének a kezelése

A PHP.INI megfelelő (biztonságos) beállítása

Biztonságos kapcsolatok, kapcsolati módok használata

23 Sablonok – Template-ek, LIB-ek

Template vagy más előregyártott library-t akkor használunk, ha

1. Gyorsítani akarjuk a PHP alkalmazás futtatását
2. Gyorsítani akarjuk a programfejlesztést
3. El akarjuk választani a fejlesztés design és a logikai részét
4. Nem akarjuk feltalálni a spanyolviaszt, azaz a más által jól megírt kódot használni akarjuk.

23.1 Template-ek, Smarty

A template rendszerek olyan PHP-ban vagy egyéb módon kifejlesztett alkalmazások, amelyek segítségével el tudjuk választani egymástól az üzleti logika és a design elemeket. Ezek a gyakorlatban azt jelentik, hogy a design elemek egy HTML-hez hasonló kódolási rendszerben jönnek létre, z adatbázis, és egyéb prorgramozási részek a kifejleszett PHP kód alapján, és a futás közben amTemplate rendszer összeszerkeszti a design template-eket és a PHPkódot, majd az így létrejött kódot futtatja.

Felvetődik a kérdés, hogy mennyire lassítja le az alkalmazást az így kifejlesztett kód? A tapasztalatok azt mutatják, hogy egyes template rendszerek használata esetén a kód futási sebessége nő, mivel nem a PHP-nak kell előállítania a teljes HTML tartalmat, hanem azt HTML kódban tartjuk.

A PHP programozónak és a Designernek könnyű elválasztania a munkáját, hiszen a programozó és a designer egy jól definiált felhasználói interface-en keresztül kapcsolódik egymáshoz.

A programnak az a feladata, hogy az általa előállított PHP kód eredményét helyezze el egy tipikusan string típusú változóba, amit aztán a sablon feldolgozó modulja beszerkeszt a Sablon tartalmába és elküldi a WEB szerveren keresztül a kliensnek.

Tekintettel arra, hogy a Web oldalak jelentős része még akkor is statikusnak tekinthető, ha PHP-val állítkjuk elő, egyes sablon rendszerek arra is képeske, hogy az összeszerkesztett oldalakat cache-eljék, azaz az összeszerkesztett oldallal szolgáljanak ki bizonyos kéréseket.

Az alábbiakban a Smarty nevű népszerű alkalmazáscsomaggal ismerkedünk meg egy kicsit.

A telepítése során a **smarty.inc.php** oldalt kell beszerkesztenünk az oldalaink elejére, ami egyúttal inicializálja a rendszert.

```
<?php
/**********************
* index.php az alkalmazás főoldala
* HTML neve: main
**********************/
Include(”config.php”); //színek és egyéb paraméterek beállítása

session_start();

// SESSION Után megnyitandó modulok
if(file_exists("libs/Smarty.class.php"))
 define("SMARTY_DIR","libs/");
else
 define("SMARTY_DIR",SMARTY_ROOT);

require_once("Smarty.class.php");  //Smarty beszerkesztése

$smarty = new Smarty;
$smarty->template_dir = SMARTY_ROOT."templates/";  //Smarty szükséges könyvtárai
$smarty->compile_dir  = SMARTY_ROOT."templates_c/";
$smarty->config_dir   = SMARTY_ROOT."configs/";
$smarty->cache_dir    = SMARTY_ROOT."cache/";

$smarty->register_modifier("numberformat","numberformat");

//Menürendszer előállítása
$Menu = Menu();

//------------- Interface --------------------
//Aktuális ID meghatározása
```

```
 // ---------- Smarty megjelenítési rész -------------------
 $smarty->Assign("Menu",$Menu);
 //
 $smarty->Assign("mnutxtcolor",MNUTXTCOLOR);
 $smarty->Assign("mnubgcolor",MNUBGCOLOR);
 $smarty->Assign("mnuhightxtcolor",MNUHIGHTXTCOLOR);
 $smarty->Assign("mnuhighcolor",MNUHIGHCOLOR);
 $smarty->Assign("css",dirname($_SERVER["REQUEST_URI"]));

 $editlink = "edit.php?ID=".$ID."&LASTID=".$ID."&cmd=LOAD&tblname=Muszer#".$tab;
 $smarty->Assign("editlink",$editlink);

 session_write_close();
 // ------------- Kimenet tömörítve menjen ki -----------

 $smarty->Assign("SID",session_id());
 $smarty->display(PATH."templates/index.tpl"); //A megjelenítés sablonja alapján a
         //megjelenítést végz                                    ő modul
?>
```

A példában egy program el őállítja a menü kódját, a Config.php-ben beállítjuk a zsínekkel kapcsolatos információkat. Az alábbi utasításokban átadjuk a sablonrendszernek egy-egy változó tartalmát:

$smarty->Assign(”azonosító”,$tartalom);

Az azonosító a sablonokon szintén változóként jelentkezik. A sablonok HTML tartalmába a smarty motor behelyettesíti az átadott akutális értékeket és az így kicserélt tartalommal jelennek meg a html oldalak.

A Sablon egyszerű esetben az alábbi lehet.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>

<title>{$Title}</title>
<meta http-equiv="Content-Type" content="text/html; charset={$langheader}">
<link href="{$css}/muszer.css" rel="stylesheet" type="text/css">
</head>

{literal}
  <script language="javascript1.3" src="js/fz_scripts.js" type="text/javascript"></script>
{/literal}
{$htmlheader}
<body {$firstpopup}>
<table style="width:980px;padding:0px;margin:0px;border:0px">
<tr>
  <td>
    <table style="padding:0px;margin:0px;border:0px">
      <tr>
        <td>
          <table style="padding:0px;margin:0px;border:0px">
            <tr>
             <td>
               <applet code="apPopupMenu" Archive="menu/apPopupMenu.jar" Width="450" Height="24"
mayscript>
               <param name="key" value="g|1iqpfks/phu">
               <param name="Copyright" value="Apycom Software - www.apycom.com">
               <param name="isHorizontal" value="true">
               <param name="3DBorder" value="true">
               <param name="solidArrows" value="false">
               <param name="buttonType" value="3">
               <param name="status" value="link">
               <param name="alignText" value="left">
               <param name="backColor" value="{$mnubgcolor}">
               <param name="backHighColor" value="{$mnuhighcolor}">
               <param name="fontColor" value="{$mnutxtcolor}">
               <param name="fontHighColor" value="{$mnuhightxtcolor}">
               <param name="font" value="Arial,10,0">
               <param name="menuItems"  value="
               {$Menu}">
               </applet>
             </td>
           </tr>
```

```
          <tr>
            <td>
            {$List}
            </td>
          </tr>
        </table>
 </td>

    <td valign="top" >
    <iframe src="{$editlink}" scrolling="no" name="EditFrame"
style="width:540px;height:{$editheight}px;border-width:1px;border-color:
#e6e6e6;padding:0px;margin:0px">
       </iframe>
    </td>
  </tr>
</table>
</body>
</html>
```

A fenti példában a {$azonosító} alakú részek helyére kerül be a sablonrendszernek átadott aktuális érték.

Lehetséges az is, hogy feltételektől függően heélyezzünk el értékeket a sablonon:

{if $valtozo}

Egyik tartalom

{else}

Másik tartalom

{/if}

Lehetséges ciklusokat szervezni és egyéb lehetőségek is rendelkezésre állnak.

További információkat a http://smarty.php.net-en találhatunk.

23.2 Előre gyártott osztályok, LIB-ek

Amikor egy viszonylag összetett feladatot kell elvégezni, gyakori, hogy azt már elvégezte más, és igazából nekünk csak az ő munkáját kellene hazsnosítanunk. Hála a jó égnek a világon rendkívül sok olyan programozóvan, aki szabaddá teszi a forráskódjait, így azt mások is használhatják. A PHP ráadásul az a világ, ahol a programozók el őszeretettel osztják meg egymás között az információt. Talán két ilyen fontos gy űjtemény létezik a PHP világában, ahol nagyon sok feladatra találunk megoldást:

PEAR Csomag

A pear a http://pear.php.net–en megtalálható programcsomag. Segítségével sokféle feladatot elvégeztethetünk ingyenesen használható modulok segítségével. A rendszer jellegzetessége, hogy azoknak a programozóknak, akik a saját megoldásaikat el akarják helyezni a gyűjteményben, először minősíttetni kell a kódjukat formai és tartalmi szempontból is. Amennyiben megfelel a kód, azután már a gy űjtemény részévé válik. A kifejlesztett modulok szabványosan kapcsolódhatnak egymáshoz, használhatják egymás tulajdonságait. A PEAR használatához a szervert futtató gépen telepíteni kell a PEAR csomagot és minden olyan modult, amelyre a hazsnálni kívánt modul hivatkozik. Ez kicsit hasonlít a LINUX filozófiájához, ugyanis a csomagokat nevükke és verziószámukkal azonosítják és csak azok települnek a gépre, amelyekre szükség van. Minden modulhoz tartozik szabványos formátumú Dokumentáció is, amit a programozó kiegészíthet még saját egyéb doksival is.

PHPClasses

Szintén nagyon sok feladatra találunk megoldást. Ennek a gyűjteménynek a különlegessége, hogy csakis kizárólag Objektum Orientált megoldásokat helyezhetnek el a programozók az oldalakon. A megodlások általában függetlenek egymástól. Minden modulhoz tartozik, tartozhat dokumentáció, példaalkalmazás is. Az egyes modulokra szavazni lehet. A site-nak van hírlevele, amire feliratkozva naponta 1 levelet kapun k, az új vagy változott csomagok rövid leírásával.

24 Objektum Orientált programozás PHP-ben

Először az igényekről. Amikor elkezd valaki programozást tanulni, hazsnálja az egyszerű utasításokat és egyszerű vezérlési szerkezeteket, majd egyre bonyolultabb alkalmazásokat kezd el írni. Van, aki megmarad az egyszerű szinten. Neki soha, vagy csak nagyon ritkán lesz szüksége az OOP használatára. Abban az esetben, amikor adatbázis is kerül az alkalmazásba, sok feltételnek megfelelő összetett programot írunk, el őbb-utóbb nem elkerülhető az OOP használata. A mai ablakozós rendszerek mindegyike OOP technikával készült.

Egyszerű adatszerkezetek

Az alábbiakban egy kis programozáselmélet következik.

Az egyszerű adatok, mint például az integer, real, karakter vagy hasonló adattípusok felhazsnála úgy történik, hogy egy változó létrehozásakor megmondjuk a változó nevét és típusát, esetleg az értékét.

Az így generált változók egymástól függetlenül jöhenek létre, és kezelésük is némileg független egymástól. A tipikus programszerkezet az ilyen változók feldolgozására **a szekvencia, azaz utasítások egymásutánja**.

Az ilyen adatszerkezetet **egyszerű adatszerkezetnek** hívjuk.

Összetett adatszerkezetek

A programozás során szükséges, hogy több adatot összekössünk egymással és azokat együt kezeljük. A hagyományos programozásban két lehetőségünk van erre.

Tömbök

Ha azonos típusú és méretű adatokat akarunk együtt kezelni, akkor **tömböket** hozunk létre.

A tömb tehát olyan adatszerkezet, amelyben az adatok egyforma méretűek. A tömböt alkotó adatok a tömb elemei. A tömb elemeinek az eléréséhez pedig indexeket használunk. Az index a programozási nyelvtől függően pozitív egész szám, amely nullától (C-ben) vagy, mint a Pascalban a megadott kezdőindextől kezdődik.

Asszociatív tömbnek hívjuk azt a tömböt, amelyben a tömb indexeket nem egélsz számmal, hanem stringekkel azonosítjuk.

A tömbök feldolgozása általában **ciklus vezérlési szerkezetet** igényel, ugyanis a tömb elemeit általában egymás után érjük el sorban, és a tömb elemeken általában ugyanazt vagy hasonló, esetleg feltételektől függő tevékenységet végzünk el.

Abban az esetben ha két indexxel hivatkozhatunk a tömb elemeire, akkor két dimenziósnak hívjuk a tömböt, ha n db indexxel hivatkozhatunk egy elemre, akkor n dimenziós tömbről beszélünk.

Az n dimenziós tömbök feldolgozása általában **n db egymásba ágyazott ciklussal** zajlik. Tipikusan a for vagy while ciklust hazsnálhatjuk ilynekor.

Asszociatív tömbök esetén azonban nem minden nyelvnek van jó megoldása. Pascalban a tömb következő, illetve előző, első és utolsó elemére hivatkozhatunk. PHP-ban speciális ciklus a foreach(), amivel végig lehet menni a tömb elemein.

Rekordok

Ha nem azonos típusú adatokból akarunk összetett adatszerkezetet létrehozni, akkor az ilyen adatszerkezetet rekordnak (Pascalban), struktúrának (C-ben) hívjuk, illetve PHP-ban minden tömb lehet különböző típusú, tehát ebben a nyelvben a tömb és a rekord szinte azonos. Általában a rekord típus lehet ennek a közös neve.

A rekord adattípus feldolgozása szelekciót igényel, azaz a különböző típusú adatokat különböző kódok, programrészek dolgozzák fel. Erre a feltételes elágazás (if…then…else…) vagy a többirányú elágazás (switch) alkalmas.

Sturktúrált programozás

A nem túlságosan összetett feladatok megoldása során használható az a módszer, hogy a programban az adatszerkezeteket definiáljuk és az adatszerkezetek alapján hozzuk létre a programszerkezetet. Ez egy eljáráshívási rendszert hoz létre, amelyben az egyes eljárásoknak, függvényeknek átadjuk a paramétereket, majd a visszakapott eredményeket más eljárásokkal, függvényekkel tovább feldolgozzuk.

A struktúrált programozás akkor kezd nehézkessé válni, amikor a programokkal különböző adattípusokkal kell ugyanazt a fajta műveletet elvégezni, mint például a grafiklus felhasználói interface létrehozása. A grafikus elemek különböznek egymástól, ugyanakkor a feladatok ugyanazok.

Ilyen esetben célszerű lenne olyan programot írni, amely „tudja”, hogy milyen paramétereket adunk át neki, és ennek alapján ki tudja választani a megfelelő kezelő alkalmazást.

Az is célszerű, hogy amennyire lehet újrafelhasználható kódot írjunk.

Célszerű az is, hogy az egyes kódrészek csak szabványos paraméterlistán és függvényeken keresztül tartsanak egymással kapcsolatot (ezt már a struktúrált programozás is tudja).

Új gondolat az, hogy legyünk képesek nagy bonyolultságú adatstruktúrák egyben történő kezelésére, mint például egy tömb vagy egy rekord egyszerre történő kezelése, ugyanakkor tetszőleges bonyolultású adatokat tudjunk ily módon kezelni.

A fenti problémákra a megoldás a következő.

Létesítsünk olyan adatszerkezetet, amelyben az adatszerkezetben tárolt adatok írjuk le, hanem azt is, hogy annak az adatszerkezetnek az adatait melyik kód írja le. **Az adat és a kód elválaszthatalanul tartozzon össze**. (Struktúrált programozásnál lehetséges az, hogy egy függvénynek olyan paramétert adunk át, amelyet nem tud feldolgozni, ekkor fatális hiba vagy fodítási hiba keletkezik). Ha az összetartozást leírjuk, akkor a kód saját maga „tudja”, hogy melyik adatokon kell dolgoznia. Az így definiált adattípust **osztálynak** hívjuk (Class).

Ez tehát egy adattípus lesz, ami a korábban leírt adattípusokhoz, adatszerkezetekhez hasonlóan akkor használható, ha létrehozunk egy változót, amelynek a típusa ez az adattípus lesz. Ezt hívják **példányosításnak**. Ebben az esetben a **létrejött adatszerkezetet objektumnak hívjuk**.

Az osztályban vagy objektumban található adatokat **az osztály tulajdonságainak** (property) hívjuk.

Az osztályban vagy objektumban található függvényeket és eljárásokat **az osztály metódusainak** hívjuk.

<Kitérő>

Amikor egy programot írunk és lefordítjuk, akkor nyilvánvalóan az osztályban található eljárásoknak memóriát kell foglalni. Ezt a fordító rendszer természetesen elvégzi helyettünk. Amikor a kódunk létrehoz egy objektumot, akkor létre kell hozni a memóriában az osztály tulajdonságainak megfelelő memóriaterületet, továbbá az így lefoglalt memóriaterületet össze kell kötni az osztály kódjával is. Egyszerű megoldás lenne, ha minden objektum egy az egyben lemásolódna az eredeti oszttályról, mint egy sablonról, de ebben az esetben nqagyon sok memóriára lenne szükségünk, ezért a programok ilynekor a kódot csak egy példányban tárolja a program és az objektum csak hivatkozásokat tárol az eredeti osztályban definiált eljárásokra.

A memóriafoglalás ugyanakkor meglehet ősen bonyolulttá is válhat, ugyanis szükség lehet a pédlányostás során kezdőértékek adására is, ezt pedig nem lehet mindenre általnosan megfogalmazni. Az osztály példányosítása tehát összetett folyamat.

</Kitérő>

Az osztályok mindig tartalmaznak egy (vagy több) olyan metódust, amely az osztály példányosítása vagy másképpen az objektum létrehozásakor lefut. **Ezt hívják konstruktornak.** A konstruktor feladata lefoglalni a megfelelő memóriaterületet az objektum résére, illetve minden olyan beállítást megvalósítani, amely az objektum létrehozása során szükséges lehet. Ha egy osztályban nem definiálunk konstruktort, akkor a rendszer létrehoz egy default konstruktort.

Hasonló probléma az, hogyha egy objektumot megszüntetün, akkor illik felszabadítani a memóriaterületét, és lezárni, illetve elengedni a menet közben lefoglalt erőforrásokat. Azt a metódust, amely ezt megteszi **destruktornak hívják**. Minden objektumnak van default destruktora.

A PHP-ban az osztályok definiciója az alábbiakban történik.

```
Class osztalynev {
   Var $tul1;
   Var $tul2;
   Var $tul3;
   Function osztalynev(){  //Saját konstruktor
   }
   Function Method1(){
   }
```

```
    Function Method2(){
  $a = $this->tul2;
          Print($a);
   }
}
```

Az osztály példányosítása az alábbiakban zajlik.

```
$a = new osztalynev();
```

Az OOP szabályai szerint vannak olyan tulajdonságok és metódusok, amelyek **publikusak** és vannak olyanok, amelyeket csak az osztályból magából lehet elérni. Ezek a **private** metódusok és tulajdonságok. A PHP4 egyáltalán nem, a PHP5 már részben támogatja ezeket az OOP tulajdonságokat, ezért az alábbiakban. A PHP-ban minden metódus és property publikus.

Az osztályon kívülről az alábbi módon tudjunk értéket adni egy osztály egy tulajdonságának:

```
$a->tul2 = ”helló”;
```

Az objektumon belüli kód azonban nem tudja, hogy saját magának mi a neve, ezért egységesen az objektumon belüli tulajdonságok elérésénél az alábbi módot használjuk:

```
$this->tul1 =  $this->tul2 . ” tetszik?”;
```

Minden osztály ugynais saját magáról azért tud, csak a sjaát nevét nem tudja.

Az OOP-nak olyan további tulajdonsága az öröklődés.

Ha definiáltunk egy olyan osztályt, amely csak bizonyos tulakjdonságaiban b ővebb egy korábban definiáltnál, akkor a korábbi definiíciót kiterjeszthetjük és a régi osztályra alapozva egy új osztályt hozhatunk létre. Ezt egy **osztály kiterjesztésének** hívjuk. Ilyenkor a kiterjesztett osztály **örökli** az **ős-osztá**ly tulajdonságait és metóduait. Az **ős-osztályt szülőnek** hívjuk, a kiterjesztettet **gyerek osztálynak**.

```
Class Osztalykit extends Osztalynev{
     Var $tul3;
      Function Osztalykit(){
          $Parent::osztalynev();
      }
      Function MehodKit(){
            $this->Method2()
            $this->tul3 = $this->tul2;
        Print($this->tul3);
      }
}
$b = new Osztalykit();
```

A fenti példában a korábban definiált osztalynev() class-t kiterjesztettük. A konstruktorban kénytelenek voltunk meghívni a szülő osztály konstruktorát $parent:: osztalynev().

A gyerek osztály eléri a szülő osztály metódusait és tulajdonságait is. A szülő osztály azonban nem éri el a gyerek osztályét.

A PHP-ban egy gyerek osztálynak csak egy szülője lehet. Ez más nyelveken nem feltétlenül van így.

Az egyes osztályokban lehetnek ugyanolyan nev ű metódusok, s őt a szül ő-gyermek kapcsolatban lév ő osztálystruktúrákban is lehet azonos nev ű metódus vagy változó. Az OOP esetén ha egy osztályban meghívunk egy metódust vagy egy tulajdnoságra hivatkozunk, a rendszer tudni fogja, hogy az esetlegesen azonos nev ű példányok közül mikor melyiket kell meghívni. Mindig az éppen aktuális osztály megfelelő példányát.