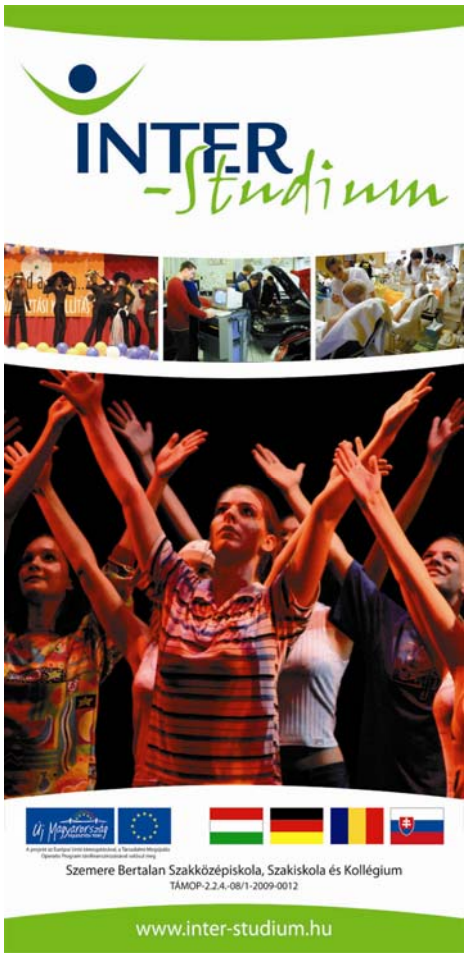
Webszerkesztés, a web programozás alapjai

3. modul

PHP programozás

Az egész életen át tartó tanulás fejlesztése az intézmények közötti nemzetközi együttműködéssel

TÁMOP-2.2.4.-08/1-2009-0012



Szemere Bertalan Szakközépiskola, Szakiskola és Kollégium

Szerkesztette: Lakatos Zsolt

Lektorálta: Molnár Gábor

A kiadvány az „INTER-STUDIUM - Az egész életen át tartó tanulás fejlesztése az intézmények közötti nemzetközi együttműködéssel” című, TÁMOP-2.2.4.-08/1-2009-0012 számú projekt keretében készült.

A projekt az Európai Unió támogatásával, a Társadalmi Megújulás Operatív Program társfinanszírozásával valósul meg

- 2010 -

Tartalom

1. óra. Bevezetés. A PHP története, feladata

A WEB megszületése óta léteznek olyam programozási lehetőségek, melyek a web oldalak dinamikus tartalommal való feltöltését hivatottak megtenni. A Unix-os környezetben természetes választás volt a C vagy C++. Ennek az alapvető problémája abból adódik, hogy könnyedén lehet benne biztonsági hibákat véteni. Ennek az eredménye pedig egy könnyen feltörhető, módosítható oldal lesz.
A következő triviális választás a Perl nyelv volt. A Perl-nek a legnagyobb problémája a feldolgozási sebességben rejlik. Bár nagyon gyorsan és egyszerűen fejleszthetőek alkalmazások, a fordítási, végrehajtási sebesség meggátolja azt, hogy igazán jó oldalak készüljenek a segítségével.
Itt kerül képbe a PHP. Ezt a programozási nyelvet kifejezetten azért alkották, hogy dinamikus web lapokat tudjunk készíteni. A PHP rendelkezik minden olyan beépített kapcsolattal és modullal, amiknek a segítségével fel tudjuk dolgozni a weblapokról érkező adatokat, és ezeket akár adatbázisban is tudjuk tárolni. Igazából semmi különös feladat nincs ami ahhoz kell, hogy a teljes dinamizmus megvalósítható legyen.
A Unix-os (Linux-os) világ teljesen természetesnek veszi a PHP használatát. A legtöbb ilyen operációs rendszerre automatikusan feltelepül a nyelv minden eleme. Sebességi problémáktól sem kell tartanunk. A nyelv felépítéséből adódóan garantált a lehető legjobb válaszidő. Természetesen támogatottak az objektum orientált megoldások is.

A PHP jelentése

A betűszó a következőt jelenti: Personal Homepage Page Tools. Amint a névből is kiderül ez eleve arra készült, hogy weblapokat tartsunk vele karban.
A hivatalos elnevezés a fejlesztőktől a következőképpen hangzik: PHP Hypertext Processor.
Ez azt jelenti, hogy egy olyan eszköz van a kezünkben, aminek a segítségével honlap tartalmakat tudunk megjeleníteni, vagy feldolgozni.

A PHP működése

A HTML nyelvvel szemben a kiszolgáló nem küldi el a PHP utasításokat a kliens részére feldolgozás céljából, hanem a feldolgozás kiszolgáló oldalon történik. A folyamat így zajlik:

1. Kérés a klienstől a kiszolgáló felé (www.mydomain.com/index.php)
2. Az index.php tartalmát a kiszolgáló (web szerver) továbbítja a PHP értelmező felé
3. A végrehajtott php kódból az értelmező HTML kódot generál
4. A kész HTML kód továbbítása a kliens felé
5. A kész oldal megjelenítése a böngészőben

Ez a végrehajtási módozat sokkal biztonságosabb, mintha a kliens hajtaná végre a kódot. Menetközben nem módosítható a kód, nem látszanak pl. egy adatbázis kapcsolat felépüléséhez szükséges adatatok, stb.

A nyelv jelenleg az 5.2.13-as verziónál tart. Ez a verziószám, a jegyzet írásának pillanatában érvényes. A fejlesztők viszonylag gyakran bocsájtanak ki újabb és újabb verziókat, amely ettől esetleg csak az utolsó számjegyben tér el. Ezek a verzióváltások nem tartalmaznak gyökeres változásokat a nyelv felépítésében, inkább a felfedezett hibák javításai történnek meg.

Miért érdemes a PHP-t választanunk, mint web fejlesztési eszközt?

Nő a fejlesztési sebesség. Mivel a nyelv nyílt forráskódú (GNU GPL) bármilyen problémára szinte azonnal találhatunk választ. Az interneten rengeteg fórum, levelező lista, tutoriál érhető el amelyek mindegyike a nyelvet támogatja. Ezek természetesen magyar és angol (rengeteg más) nyelven is elérhetőek. Több száz olyan oldalt találhatunk, ahol már elkészült programokat lehet böngészni. Szinte biztosan találunk megoldást minden felmerülő gondunkra.

Gyors futtatás, fordítás. A nyelv mögött álló Zend Engine garancia a futtatási sebességre. Ha összehasonlításokat kezdünk vizsgálni más nyelvekkel akkor azt tapasztaljuk, hogy sebesség terén a PHP a legjobb szinte minden esetben.

Hordozhatóság. A nyelv elérhető szinte minden operációs rendszeren és platformon. Szinte semmit nem kell módosítanunk, ha egy Windows rendszeren megírt programot Linux-on szeretnénk a továbbiakban futtatni.

Adatbázis támogatottság. A piacon elérhető adatbázis kezelő rendszerek szinte mindegyikére van támogatása a PHP-ban. Nem kell kompromisszumokat kötnünk, mert nem érhető el az a DBS[1] típus amit a megrendelő használ, vagy használni szeretne. Néhány ismertebb DBS a listából: MySql, PosrGreSQL, MSSQL, Oracle.

Összefoglalás

A PHP minimális programozási tapasztalatokkal felvértezve könnyen és gyorsan tanulható nyel. Hamar lehet a segítségével látványos eredményeket elérni.

[1] Database Management System – Adatbázis Kezelő rendszer

2. óra A PHP telepítési módozatai, WAMP Windows alatt

Amint az előző részben említettük a PHP szinte minden operációs rendszeren elérhető. Ha Linux-ot választunk, akkor szinte minden esetben az alap operációs rendszerrel együtt felkerül a PHP futtatásához szükséges minden elem. Ha ez mégsem így történik, akkor a következő lehetőségek közül választhatunk. Szinte minden Linux rendelkezik grafikus felülettel (X Windows). A grafikus felületen elérhető valamilyen csomagkezelő alkalmazás, melynek segítségével a megfelelő csomagot kiválasztva a telepítés szinte teljesen automatikusan megtörténik.

Ha nem áll rendelkezésre grafikus felület, akkor parancssori eszközt kell használnunk a telepítéshez. Ez pl. Debian disztribúció alatt az *apt-get install* paranccsal érhető el.

Mind a két módozat esetén figyeljünk arra, hogy működő internetkapcsolatra van szükség a telepítés sikeres végrehajtásához.

Ahhoz, hogy ki tudjuk próbálni a nyelvet nem kell Linux-ot telepítenünk. Lehetőség van a nyelvet minden elemével együtt bármilyen Windows-ra is telepíteni. Ebben az esetben is két út áll előttünk.

- Kézi módszer, amikor is minden szükséges dolgot letöltünk, telepítünk és kézzel konfigurálunk. Ez a nehezebb, de ez hasonlít legjobban ahhoz a futtatási környezethez, amit Linux alatt is kapnánk.
- Automatikus mód: Valamilyen előre összeállított csomagot töltünk le és használunk. Ennek a legnagyobb előnye az egyszerűség és gyorsaság. Számos ilyen kész csomag áll a rendelkezésünkre az interneten. Pl.: XAMP, WAMP, EasyPHP, stb. Ezeknek a csomagoknak vannak hátrányaik is. Nem minden esetben működnek teljesen jól. (Saját tapasztalat: XaMP-al nem tudtam a PDF nyomtatási lehetőségeket használni, mert folyamatosan hiányolt egy dll fájlt, annak ellenére, hogy az rendelkezésre állt a megfelelő könyvtárban.

A következőkben bemutatom mindkét telepítési, használati módozatot. Ne feledkezzünk meg a következőről: a használathoz nem csak a PHP-ra, hanem egy választott WEB szerverre is szükségünk lesz. Az általam választott szerver az Apache, mert ez is ingyenes, és az előbb felsorolt kész csomagok is ezt tartalmazzák. Természetesen elérhető a IIS[2] alatt futtatás is, az ehhez szükséges ismeretek megtalálhatóak a PHP hivatalos honlapján (http://php.net). A későbbi munkák érdekében az SQL kiszolgálót is célszerű most telepítenünk.

A választott környezet a következő:

Web szerver: Apache http Server. A most elérhető verzió a 2.2.14-es. Letöltés: http://httpd.apache.org/download.cgi

PHP. PHP 5.2.13, letöltés: http://hu.php.net/get/php-5.2.13-Win32.zip/from/a/mirror

Adatbázis szerver: MySQL. Az elérhető legfrissebb verzió 5.1.44. Letöltés: http://www.mysql.com/downloads/mysql/

Természetesen nem kell minden esetben a legújabb verziókat használnunk. Ha bevált egy csomag akkor bátran használjuk akkor is, ha van újabb elérhető csomag.

[2] Internet Information Server, a Microsoft Web kiszolgálója. Szinte minden Windows-ban elérhető

3. A nyelv alapelemei, alapvető szintaxis

A PHP feldolgozó (pharser) a feldolgozás során minden karaktert a kimenetre másol egészen addig, amíg valamilyen speciális jelölést nem talál amely egy PHP kód kezdetét jelzi a számára. Ebben az esetben az általa értelmezhető kódot lefuttatja, majd az eredményt elhelyezi a kimeneten.

A nyelv szabályai szerint négyféle lehetőség van a PHP módba való kerüléshez.[3]

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

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

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

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

Az első lehetőség a leginkább javasolt, mivel ezzel XML-konform dokumentumokban , mint például XHTML, is lehet PHP kódokat elhelyezni.

A második forma nem mindig használható, csak akkor, ha a rövid nyitó jelölések engedélyezve vannak. Engedélyezhetjük - PHP 3-ban - **short_tags()** függvényhívással, a short_open_tag beállítással a PHP konfigurációs fájlban, vagy a PHP fordításánál a **configure** program --enable-short-tags opciójával. Annak ellenére ha alapértelmezés szerint engedélyezve van is, a rövid nyitójelölések használata ellenjavallt

[3] Forrás: http://www.php.net, PHP dokumentáció

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

A leggyakoribb mód amikor az egész oldalunkat a PHP kezdő és záró tag-jei közé szerkesztjük (1.-es példa). Ebben az esetben a teljes oldal átfut az értelmezőn.

A kódot minden esetben, mint a példákon is látszik a ***<?php*** tag-el kell kezdeni. A kód befejezése a ***?>*** tag-el történik.

A szerkesztéshez célszerű olyan programot használni ami un. syntax highlight[4] tulajdonsággal bír. Ezek segítségével sokkal könnyebb szintaxikailag helyes kódot írnunk. Jelen jegyzetben az ingyenesen elérhető és magyarul „beszélő" NotePad++ programot használjuk. Információk, letőltés: http://notepad-plus.sourceforge.net/hu/site.htm. Természetesen nagyon sokféle – ingyenes - szerkesztő program elérhető. Ajánlom még a PsPad nevű programot, amely szintén elérhető magyar nyelven. Információ, letöltés: http://www.pspad.com/. A letöltés és telepítés angol nyelven zajlik, de a program futása már magyar nyelven történik.

Utasítások a PHP-ben

A C és C szerű nyelvekhez hasonlóan az utasítások lezárása minden esetben a ***;*** karakterrel történik. Természetesen ez alól is vannak kivételek amiket külön kiemelek a megfelelő helyeken.

Utasítás blokkok

Az utasítás blokkokat – szelekció, iteráció – a ***{ }*** karakterek kell, hogy határolják. Az utasítás blokkok tetszőleges mélységben egymásba ágyazhatóak. Ebben az esetben is nagy segítségünkre van a szerkesztőprogram, mert a zárójeleket „összepároztatja", megjelöli. Rögtön észrevehető, ha egy záró tagot esetleg lefelejtettünk.

[4] Syntax highlight: Szintaxis kiemelés: a szerkesztő program felismeri a programutasításokat, változókat, és különböző színekkel jelöli azokat

Megjegyzések a programon belül

A PHP-ban kétféle megjegyzése van lehetőség.

- egysoros megjegyzés: a **//** karakterek vezetik be, és a karakterek utáni rész lesz a megjegyzés, egészen a sor végéig. Nem kell lezárni a jelölést semmilyen jellel, de a következő sor már nem lesz megjegyzés formátum

```
<?php
…
….
// ez egy egysoros megjegyzés
..
..
?>
```

- többsoros megjegyzés: A **/*** karakterek vezetik be, és egészen a ***/** karakterekig tart a megjegyzésünk. Minden ami a két jelölés között van az nem fog végrehajtódni.

```
<?php
…
….
/* ez itt
egy
többsoros
megjegyzés blokk
*/
..
..
?>
```

4. Változók, változó nevek, foglalt nevek

Az előzőekből már látható volt, hogy minden változó nevet a ***$*** karakterrel kell kezdenünk. A nyelv érzékeny a kis és nagybetűk közötti különbségekre. Tehát a $alma nem ugyanazt a változót takarja mint a $Alma. A változó név betűvel vagy aláhúzás karakterrel kell, hogy kezdődjön.
A név tartalmazhatja az angol abc betűit, számokat és az aláhúzás (_) karaktert.

```
$var = "Géza";
$Var = "János";
echo "$var, $Var"; // kiírja, hogy "Géza, János"
$4site = ’ez nem jó’; // nem érvényes, mert számmal kezd˝odik
$_4site = ’ez ok’; // érvényes, aláhúzással kezd˝odik
$täyte = ’mansikka’; // érvényes, az ’ä’ az ASCII 228-as karaktere
$tükörfúrógép = "árvízt˝ur˝o"; // érvényes, ellen˝orizheted egy ASCII táblában
```

Előre definiált változók (foglalt nevek)

Általánosságban elmondható, hogy a PHP számos olyan névvel operál ami a külső kapcsolattartáshoz szükséges. Ezeket a foglalt neveket értelemszerűen mi már nem használhatjuk egyéb adatok tárolására. A szerencse az, hogy ezek a nevek az esetek nagy többségében valamilyen speciális karakterrel (legtöbbször az aláhúzás karakterrel) kezdődnek. A programozók nagy többsége nem is kezdi a saját változóit ezzel a karakterrel.

A foglalt nevek csoportjai:

1. **Apache változók:** csak abban az esetben állnak rendelkezésre, ha a rendszerünk Apache web-szerveren fut. Minden más esetben nincs garancia a változók létrejöttére és tartalommal való feltöltődésére.
2. **Környezeti változók**

3. **PHP változók**: ezeket maga a futtató környezet)pharser) állítja elő. Két nagyon fontos változó találálható ebben a csoportban:
    a. **$_POST**: HTTP POST metódussal által szolgáltatott adatokat tartalmazó asszociatív tömb, amely minden hatókörben elérhet˝o. PHP 4.1.0-ban került a nyelvbe.
    a. **$_GET**: HTTP GET metódussal által szolgáltatott adatokat tartalmazó asszociatív tömb, amely minden hatókörben elérhet˝o. PHP 4.1.0-ban került a nyelvbe.

Mindkét változót a HTML oldal és a feldolgozó PHP script közötti adatkommunikációra használjuk. Ugyanazt a feladatok másféle működéssel valósítják meg. Lásd 14-17. lecke.

Mivel az Apache és környezeti változók listája nagyon hosszú, szükség esetén nézz utána a dokumentációban, mert ennek a jegyzetnek a terjedelme nem teszi lehetővé az ismertetésüket.

Változók hatóköre

A változó hatásköre az a környezet, amelyben a változó definiált. A legtöbb esetben minden PHP változónak egyetlen hatásköre van. Ez az egyetlen hatáskör kiterjed az include és a require segítségével használt fájlokra is.

Amennyiben felhasználói függvényeket használunk, akkor jelennek meg az un. lokális hatókörrel rendelkező változók. A C nyelvben a globálisan definiált változók minden esetben elérhetőek a felhasználói függvényeken belül is. A PHP-ban ez nincs így. Nézzünk egy példát:[5]

```
$a = 1; /* globális hatáskör */
function Test ()
{
echo $a; /* egy helyi változót vár */
}
Test();
```

[5] forrás: PHP dokumentáció

A kimeneten nem fog megjelenni semmi, mert a $a nevű változónak lokálisan (a Test függvényen belül) nincs értéke.

Hogyan lehet ezt mégis megcsinálni? A ***global*** kulcsszó használatával. A függvényünkön belül globálisként definiáljuk a szükséges változó(ka)t és máris működni fog a dolog.Példa[6]:

```
$a = 1;
$b = 2;
function Osszead()
{
global $a, $b;
$b = $a + $b;
}
Ossszead();
echo $b;
```

Ennek a szkriptnek a kimenete 3 kell hogy legyen (annyi is lesz)

[6] [6] forrás: PHP dokumentáció

5. Adattípusok, adatszerkezetek

A PHP-ban nyolcféle változó típussal találkozhatunk. Ezek a következők:

A skalár típusok:

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

Az összetett típusok:

- array (tömb)
- object (objektum)

Speciális típusok:

- resource (erőforrás)
- NULL

A klasszikus programozási nyelvektől eltérően a változókat nem kell előre definiálni. Ez sok esetben könnyebbséget ad a programozónak, de sok hibára is okot adhat. Ez a könnyebbség megköveteli a programozótól, hogy mind a változók elnevezésében, mind pedig a használatukban nagyon konzekvens legyen. Miről is van szó?

```
<?php

$alma = 1234; //ez egy skalár típusú változó, egész számot tárolunk benne
…
…
…
…
$alma = „alma” // itt pedig már ugyanaz a változó string típust vesz fel.
```

A példában látható, hogy gond nélkül tudom ugyanazt a változó nevet két egymást követő sorban különféle adattípusok tárolására használni. Ez persze azzal jár, hogy az első sorban tárolt skalár érték elveszik, mert a később jövő string definíció felülírja az értéket. Innentől kezdve már a változónk string típussal bír.

Ez a PHP felfogásában a következőképpen hangzik: a változó típusát nem a programozó adja, hanem a futási környezet határozza meg azt. Természetesen van arra lehetőségünk, hogy lekérdezzük a változó aktuális állapotát. Ehhez a get_type() függvényt kell használjuk.

```
<?php
$alma =”karakteres típus” ;

echo get_type($alma) ;

?>
```

Típusok definiálása:
Logikai (boolean): a változó igaz vagy hamis (true/false) értékeket vehet fel.

```
$logikai = true ;
vagy
$logikai = false;
```

Egész számok (int): az egészek definiálhatók decimális, hexadecimális, és oktális formában pozitív avagy negatív előjellel.

```
$tizes = 123 ;
$tizes_negativ = -123 ;
$tizenhatos = 0x1AB ;
$nyolcas = 0123 ;
```

Megjegyzés: az oktális formát szinte soha, a hexadecimális formát nagyon ritkán használjuk.

Az értelmezési tartományt a futtató operációs rendszer határozza meg, a legtöbb esetben 32 bites előjeles számokat tárolhatunk.

Lebegőpontos számok (float): a következő szintaktikával deklarálhatók

```
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
```

Az értelmezési tartományt a futtató operációs rendszer határozza meg, a legtöbb esetben 1,8e308 a legnagyobb ábrázolható szám.

Karakter lánc (string): a PHP-ban egy karakter egy byte-nak felel meg. Jelenleg a PHP-ban nincs unicode támogatás. Karakteres változók létrehozása:
Aposztróf segítségével:
$alma = ’ez egy karakter lánc az alma nevű változóban” ;
Idézőjelek segítségével. Ebben az esetben elhelyezhetünk a karakterláncban számos vezérlő, vagy speciális karaktert
A következő speciális karakterek használhatóak:

\n	újsor (LF vagy 0x0A (10) ASCII kódú karakter)
\r	kocsi vissza (CR vagy 0x0D (13) ASCII kódú karakter)
\t	vízszintes tabulátor (HT vagy 0x09 (9) ASCII kódú karakter)
\\	vissza perjel
\$	dollárjel
\"	idézőjel

Tömbök (array)

Az összetett adatszerkezetek között a tömb a leggyakrabban használt típus. Ez különösen igaz a PHP-re, ahol számtalan esetben az adatkommunikáció tömbökön

keresztül történik. Ilyen pl. a $_POST és $_GET változó, de az SQL lekérdezések végeredménye is tömbbe érkezik meg. Így elmondhatjuk, hogy az egyik legfontosabb szerkezetről beszélünk. Mint minden más programozási nyelvben, itt is arról szól a dolog, hogy adott számú, azonos típusú elemet tartunk a szerkezetben, és az elemek sorszámozottak (indexeltek). A PHP-ben az indexelés minden esetben 0-val kezdődik. Más nyelvekkel ellentétben a tömböt nem kell előre definiálni, a mérete teljesen dinamikus. Nem igazán fordulhat elő az az eset, hogy „túlcímezzük" a tömböt, vagyis olyan elemre próbálunk hivatkozni, ami nem létezik.

Tömbök létrehozása

1. az array kulcsszó használatával

```
$tomb = array('Tavasz', 'Nyár', 'Ősz', 'Tél')
```

Ebben a példában egy négy elemű tömböt hoztunk létre (indexei: 0,1,2,3), melynek a típusa karakterlánc (string).

2. „röptében"

```
$tomb[] = 'Tavasz' ;
$tomb[] = 'Nyár' ;
…
…
```

Mivel nem adtunk meg a zárójelek között kulcsot (indexet) ezért a PHP megkeresi az utolsó tömbindex-et és megnöveli eggyel, majd ebbe az indexbe illeszti be az általunk megadott adatot. Az első sorban még a tömb nem jött létre, ezért létrehozza és a nulladik index-re beilleszti az adatot. A második sorban már megvan a tömbünk, csak a sorszámot kell növelni, és így tovább.

Asszociatív tömbök

Ez a fogalom kevés nyelvben ismert. Például nem találkozhatunk vele C-ben és C++-ban sem. Igazából a „modernebb" nyelvekre jellemző.

Az elgondolás a következő: a tömb elemeire ne sorszámmal, hanem megnevezéssel tudjunk hivatkozni. Miért jó ez?

A következő példát gondoljuk végig. Egy SQL lekérdezést hajtunk végre, aminek az eredménye egy tömbbe érkezik meg. A lekérdezés nagyon sok mezőből áll. Melyik

megoldás lehet a könnyebb, átláthatóbb? Ha az egyes adatokat sorszámmal kell kiíratnunk, vagy pedig a lekérdezés eredményére úgy tudunk hivatkozni, hogy a mezőneveket írjuk a zárójelek közé. Erre a megoldásra még visszatérek, amikor a PHP és SQL kapcsolatát tárgyalom. (22-23. óra).

Asszociatív tömb létrehozása:

```
$a['szín'] = 'piros';
$a['íz'] = 'édes';
$a['alak'] = 'kerek';
$a['név'] = 'alma';
```

6 Operátorok

Precedencia: a műveletvégzési sorrend, matematikából ismert fogalom. Általában a programozási nyelvekben nagyon hasonló a matematikához, de itt belép néhány új művelet.

Az operátorok precedenciáját a következő táblázat mutatja[7]:

,
or
xor
and
print
= += -= *= /= .= %= &= |= ^= ~= <<= >>=
?:
||
&&
|
^
&
== != === !==
< <= > >=
<< >>
+ - .
* / %
! ~ ++ -- (int) (float) (string) (array) (object) @
[]
new

Aritmetikai operátorok

Példa	Név	Eredmény
$a + $b	Összeadás	$a és $b összege
$a - $b	Kivonás	$a és $b különbsége
$a * $b	Szorzás	$a és $b szorzata
$a / $b	Osztás	$a és $b hányadosa
$a % $b	Modulus	$a / $b maradéka

[7] forrás:PHP kézikönyv, 204. oldal

Hozzárendelő operátorok

A legfontosabb, de nem egyetlen operátor az =. Alapesetben a bal oldalon álló változót egyenlővé teszi a jobb oldalon szereplő kifejezés értékével. A jobb oldalon lévő kifejezés sok „játékra” ad lehetőséget.Pl.:

```
$a = ($b = 5) + 6; // $a most 11, és $b 5

$a = 3
$a += 5
// $a-t 8-ra állítja, mintha $a = $a + 5-öt írtunk volna

$b = "Kala "
$b .= "Pál"

// $b "Kala Pál" lesz
```

Összehasonlító operátorok

Két érték összehasonlítására szolgálnak, pl. szelekció esetén

Példa	Név	Eredmény
$a == $b	Egyenlő	Igaz (TRUE), ha $a és $b értékeegyenl˝o
$a === $b	Azonos	Igaz (TRUE), ha $a és $b értéke egyenlő, és azonos típusúak (csak PHP 4)
$a != $b	Nem egyenlő	Igaz (TRUE), ha $a és $b értékei különböz˝ok
$a <> $b	Nem egyenlő	Igaz (TRUE), ha $a és $b értékei különbözők
$a !== $b	Nem azonos	Igaz (TRUE), ha $a és $b értékei vagy típusai különbözők (csak PHP 4)
$a < $b	Kisebb mint	Igaz (TRUE), ha $a szigorúan kisebb, mint $b
$a > $b	Nagyobb mint	Igaz (TRUE), ha $a szigorúan nagyobb, mint $b
$a <= $b	Kisebb, vagy egyenlő	Igaz (TRUE), ha $a kisebb, vagy egyenlő, mint $b
$a >= $b	Nagyobb, vagy egyenlő	Igaz (TRUE), ha $a nagyobb, vagy egyenlő, mint $b

Növelő/csökkentő operátorok

Példa	Név	Hatás
++$a	előnövekményes	Növeli $a-t eggyel, majd visszaadja $a értékét
$a++	utónövekményes	Visszaadja $a értékét, majd növeli $a-t eggyel
--$a	előcsökkentő	Csökkenti $a-t eggyel, majd visszaadja $a értékét
$a--	utócsökkentő	Visszaadja $a értékét, majd csökkenti $a-t eggyel

Az operátorok további fajtái, csak felsorolásban:

- bitorientált operátorok
- hibakezelő operátorok
- végrehajtó operátorok
- logikai operátorok
- string operátorok

Az előbb felsorolt operátorok közül néhányra külön kitérek majd a megfelelő fejezetben.

7 Vezérlési szerkezetek: Elágazások

A szelekció (elágazás, *if*) a programozási nyelvek – köztük természetesen a PHP-val - egyik legfontosabb eleme. Az elágazás a C nyelvben ismert szabályok szerint működik.

A kulcsszó: ***if***

if (kifejezés)

(utasítás blokk)

A végrehajtás során a kifejezés logikai értéke kerül vizsgálatra. Amennyiben a kifejezés *true/igaz* akkor végrehajtódik az utasításblokkban felsorolt összes utasítás. Ellenkező esetben a végrehajtás az utasítás blokkot követő utasítással fog folytatódni. Ennek a szerkezetnek az egyetlen hibája az, hogy nem tudjuk kezelni azt az esetet, amikor a kifejezés értéke hamisat ad. Erre megoldás az *if* szerkezet *else* ággal történő bővítése, valahogy így:

if (kifejezés)
{utasítás blokk, ha a kifejezés igaz}
else
{utasítás blokk, ha a kifejezés hamis eredményt ad}

Többszörös elágazás

Mint inden programozási nyelvben, így a PHP-ban is lehetőség van az un. többszörös elágazásra. Ez olyan szerkezet, mintha egy sereg if-es szerkezetet írnánk egymás alá. Természetesen így is megoldhatók a feladatok, de szebb és elegánsabb a ***switch*** használata.

A szerkezet a következőképpen néz ki:

```
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";
}
```

Miről is van itt szó? Ebben a szerkezetben nem egy kifejezés igaz vagy hamis voltát vizsgáljuk, hanem egy változó konkrét értékét. (Igazából nem számít, hogy a változó milyen típusú. Leggyakrabban karakteres és skalár értékekkel használjuk). A változó értékétől függően fog a megfelelő ág lefutni. Abban az esetben, ha a változó egyik felsorolt értéket sem vette fel, akkor az utolsó részben használt *default* ág fog végrehajtódni. A case után tetszőleges utasítás sorozat állhat.

8. Vezérlési szerkezetek: Ciklusok

Természetesen a PHP-ban is rendelkezésre állnak azok a ciklusszervezési eszközök, mint bármely más programozási nyelvben. Mint tudjuk a ciklusokat két nagy csoportba sorolhatjuk:

- elöl tesztelő ciklusok
- hátul tesztelő ciklusok

A két típus közötti lényegi különbség a vizsgálat helyében van. Az elöl tesztelő esetében a vizsgálat a ciklusmag előtt található. Ez azt jelenti, hogy extrém esetben a ciklusmag egyszer sem fog lefutni. A hátul tesztelő esetében a vizsgálat később található mint a ciklus mag. Ez azt jelenti, hogy a ciklusmag legalább egyszer mindenképp le fog futni.
Nézzük sorban a lehetőségeket.

while ciklus

Általános alakja a következő:
while (kifejezés)
(utasítás blokk – ciklusmag)

A ciklusmag végrehajtása mindaddig fog folytatódni, amíg a kifejezés értéke ***true***.

Figyelem: vannak olyan nyelvek ahol ez fordítva működik, vagyis addig megy az iteráció amíg a kifejezés hamis értékű!

Példaprogram: írassuk ki a számokat 30-ig a képernyőre

```
<?php
$i=1 ;
while ($i<=30)
{
echo $i . „<br>” ;
```

```
$i++ ;
}
```

A *while* ciklust viszonylag ritkán használjuk ily módon. A legtöbb esetben valamilyen esemény bekövetkezésére várunk és addig hajtjuk végre a ciklusmagot amíg az be nem következik.

```
<?php
$i=1 ;
while ($i<>$b)
{
echo $i . „<br>” ;
$i++ ;
}
```

A példában látszik, hogy arra az eseményre várunk, hogy az $i változó értéke megegyezzen a $b változóéval. (A példában nincs kezelve a $b értéke)

For ciklus

A for ciklus szintén elöltesztelő tulajdonságú. Az előzhöz képest annyi a különbség, hogy előre tudjuk azt, hogy a ciklusunk hányszor fog lefutni.
Általános alak:
for (*paraméterek*)
{
    Ciklusmag
}

A paraméterek, melyeket meg kell adnunk:

- a ciklusváltozó kezdeti értéke
- a ciklusváltozóra vonatkozó feltétel
- a ciklusváltozó változtatásának mértéke (leggyakrabban ++, vagy --).

Megjegyzés: nem kötelező, de a programozók általában az i, j, k betüket használják a ciklusváltozók jelölésére.

Ez a leggyakrabban használt iterációs megoldás. A következőképpen kell használni:

```
<?php
for ($i=0;$i<100;$i++)
{
echo $i ;

}
```

A példában kiíratjuk a számokat 0-től 99-ig. Azért csak 99-ig, mert a ciklusfejben azt a feltételt adtuk meg, hogy addig csinálja, míg a $i értéke kisebb, mint 100. Tehát a 100-as értéknél már befejezi a futást.

Hátul tesztelő ciklus- do-while

Szinte minden programozási nyelvben ebben a formában van jelen.
Általános alakja a következő

```
do
{
ciklusmag
}
while (feltétel)
```

A feltétel ugyanúgy működik, mint a *while* ciklus esetén. Addig hajtjuk végre a ciklusmagot amíg a kifejezés értéke igaz.

Foreach ciklus
Ez a módozat nem található meg minden nyelvben. A PHP-ba is a Perl-ből került át.
Általános alak:

```
foreach (tömbkifejezés as érték)
{
```

ciklusmag
}
Ez az utasítás végigmegy egy tömbön, és annak értékeivel tud tulajdonképpen bármilyen műveletet végezni. A leggyorsabb módozat arra, hogy egy ismeretlen méretű tömb tartalmát a képernyőre kilistázzuk.

```
<?php
$honapok =array('Január','Február','Március','Április','Május','Június'
,'Július','Augusztus','Szeptember','Október','November','December')
foreach ($honapok as $ertek)
{
        echo $ertek . „<br>” ;
}
```

A példaprogram a hónapok tömb tartalmát írja ki a képernyőre soronként. Persze ebben az esetben viszonylag könnyedén tudnánk írni olyan ciklust ami ugyanezt a feladatot végzi el, de az esetek többségében nem ismerjük a tömb méretét és akkor már nem biztos, hogy jó egy másfajta megoldás.

Kilépés iterációból

Minden ciklusból „menet közben” ki lehet lépni. Ilyenkor nem foglalkozunk azzal, hogy teljesült e a ciklusmagban leírt feltétel. A break utasítás hatására elhagyjuk az iterációt, és minden változónk abban az állapotban marad, mint amilyen volt neki közvetlenül a break előtt.

9. Vezérlési szerkezetek: Függvények

Mint mindenhol a php-ban is van lehetőség a programoz számára, hogy függvényeket definiáljon. A saját függvények elhelyezkedhetnek a programunkon belül, vagy pedig külön fájlban is. A viselkedés és működés mindkét esetben ugyanaz.
Az általános alak a következő:

```
function pelda ($arg_1, $arg_2, ..., $arg_n)
{
echo "Példa függvény.\n";

return $retval;
}
```

A zárójelek között található *$arg_1, … $arg_n* változók a függvény paraméterei. Ezeket az értékeket kell átadnunk a számára, amikor meghívjuk. Ügyelni kell arra, hogy pontosan annyi paraméterrel hívjuk meg a függvényt, mint amennyit a definíciókor megadtunk, mert különben hibát fogunk kapni. A *$retval* változó a függvényünk visszatérési értéke lesz. A PHP alapesetben az érték szerintí paraméterátadást támogatja, tehát a híváskor a változók értékei – és nem a memória címek – fognak átadásra kerülni. A paraméterek típusában nincs megkötés, minden ismert egyszerű és összetett típus átadható.

Abban az esetben ha külső file-ban tartjuk a saját függvényeinket akkor, a program elején az include(*fájl_név*) utasítást kell használni. Ha a betöltendő fájl kritikus (nem létezése esetén meg kell szakítani a program futását) akkor a require(*fájl_név*) utasítást kell használni. Mind a két parancs beilleszti és feldolgozza a fájl tartalmát, de az *include* tovább megy, ha nem található a fájl.

10 TESZT

1. Sorolja fel milyen PHP telepítési módokat ismer. Ismertesse a PHP telepítését Windows operációs rendszer alá
2. Sorolja fel a változókra és neveikre vonatkozó szabályokat
3. Milyen adattípusokat és adatszerkezeteket ismer?
4. Ismertesse a PHP-ban használható ciklusszervezési módokat
5. Sorolja fel a használható operátorokat!
6. Hogyan kell függvényt definiálni, és milyen módon adhatóak át a paraméterek?

11 A PHP program elkészítésének és futtatásának módjai

Mint már a bevezetőben is olvashattuk a PHP módba kerülésre többféle lehetőség adódik. Mindenképpen tudatnunk kell a feldolgozóval, hogy mely részeket kell lefordítania, és mely részeket küldi tovább változtatás nélkül.

A PHP program kezdő és záró elemei a következőek:

Elnevezés	Kezdőelem	Záróelem
Hagyományos	<?php	?>
Rövid	<?	?>
ASP stílusú	<%	%>
Script elem	<SCRIPT LANGUAGE="PHP">	</SCRIPT>

Ahhoz, hogy a második és harmadik változatot használni tudjuk, némi beállítást kell eszközölni a PHP.INI fájlban.

Figyelem: éles rendszeren a php.ini módosítása nem kívánt problémákat okozhat. Sok esetben nincs is jogunk ezeket a beállításokat megtenni. Ilyen esetekben kérjük a rendszergazda segítségét.

Ahhoz, hogy a rövid kezdőelemeket használni tudjuk keressük meg a következő sort a php.ini-ben: short_open_tag =

Ez alapesetben így néz ki : *short_open_tag = off*. Az off-ot kell on-ra cserélni, majd a web szervert újraindítani.

Az ASP stílust a asp_tags kapcsoló **on** beállításával engedélyezhetjük.

A négy lehetőség közül leggyakrabban az elsőt használjuk. Néhányszor előfordul az utolsó lehetőség, a két középsőt viszont szinte soha nem használjuk.

Minden programozó első program a „*Helló világ*” így néz ki a különféle beillesztési módokon.

Hagyományos PHP beillesztés:

```
<?php
echo „Helló Világ” ;
?>
```

Rövid tag-ek

```
<?
echo „Helló Világ” ;
?>
```

ASP stílus

```
<%
echo „Helló Világ” ;
%>
```

Script elem

```
<html>
<head>
<title> Első oldalam </title>
<SCRIPT LANGUAGE="PHP">
print ("Hello Web!");
</SCRIPT>
</head>
```

Hogyan tudjuk megtekinteni a programunk végeredményét.

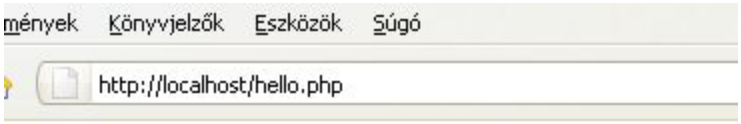
A kész programot el kell mentenünk, de nem mindegy, hogy milyen névvel és hová mentjük el. A fájlnév kiterjesztésének mindenképpen php-nak kell lenni. Ha a web szerver beállításaiban mást is megadtak akkor természetesen az a kiterjesztés is lehet, de az alap a php. A hely ahová mentenünk kell az pedig a web szerver dokumentum könyvtára. Ha a 2. fejezetben ismertetett WAMP/XAMP csomagokat használjuk akkor ez valószínűleg a c:\wamp\www vagy c:\xamp\www mappa lesz. Ez

azért nagyon fontos, mert a böngésző csak így fogja megtalálni a fájlokat és így lesz képes azokat lefuttatni. Ha nem ilyen web szervert használunk akkor meg kell keresnünk azt a mappát ahol a szerver alapértelmezetten eléri a fájlokat. Ha Apache szervert használunk akkor a httpd.conf fájlban a *document root* szekcióban találjuk meg ezt a helyet.

Feladat:

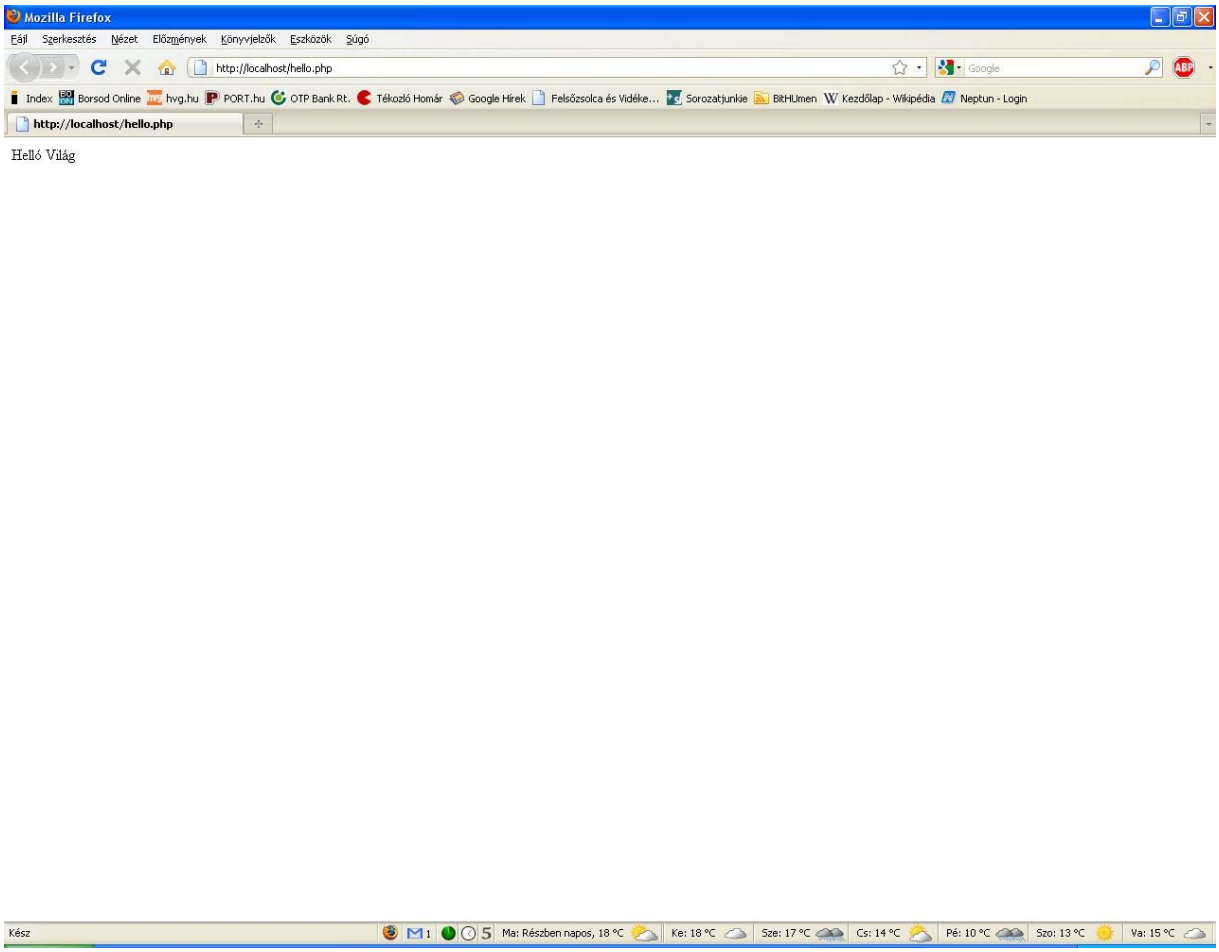
Készítsük el az előbb bemutatott helló világ programot, és próbáljuk ki a böngészőnkben.

A megoldás menete a következő:
A notepad++ alkalmazással írjuk meg a programunkat, majd mentsük el a megfelelő helyre ***hello.php*** néven. Majd indítsuk el a böngészőprogramunkat és a címsorba gépeljük be a következőt: http://localhost/hello.php, majd üssük le az Enter billentyűt.



1. ábra

Ha a program megírásánál mindent jól csináltunk, akkor az eredménynek ehhez hasonlónak (ilyennek) kell lenni:


Mozilla Firefox
Fájl Szerkesztés Nézet Előzmények Könyvjelzők Eszközök Súgó
http://localhost/hello.php
Google
Index Borsod Online hvg.hu PORT.hu OTP Bank Rt. Tékozló Homár Google Hírek Felsőzsolca és Vidéke... Sorozatjunkie BitHUmen Kezdőlap - Wikipédia Neptun - Login
http://localhost/hello.php
Helló Világ
Kész
Ma: Részben napos, 18 °C
Ke: 18 °C
Sze: 17 °C
Cs: 14 °C
Pé: 10 °C
Szo: 13 °C
Va: 15 °C

2. ábra

12. – 13. HTML form-ok kezelése

A szerver oldali programozások leggyakoribb esete az, amikor egy HTML lapra beviteli mezőket helyezünk el, és az azokba beírt tartalmat dolgozzuk fel a programunk segítségével. Valószínűleg már mindenki használt – legalább nézelődés szintjén – web áruházat. Egy mezőbe beírjuk a keresett termék nevét, és a megfelelő gombra kattintás után megjelenik a böngészőben az a tartalom amire kíváncsiak voltunk.

A következőkben áttekintjük a HTML formok-ból érkező adatok feldolgozását. A következő példákban két fájllal fogunk dolgozni: az egyik lesz az amelyik a böngészőben előállítja a megfelelő beviteli mezőket, a másik pedig a feldolgozó program. Mindét kód folyamatos bővítésével jutunk el addig, hogy minden űrlapelem feldolgozását megismerjük. Az egyszerűség és gyakorlás kedvéért az űrlapelemeket is egy PHP programban fogjuk előállítani, a feldolgozás mással nem is menne.

Az egyik legegyszerűbb eset az, amikor az oldalunkon csak egyszerű szöveges mezők vannak. Lássuk az első programot:

```
<?php
echo '<form action ="feldolg.php" method="POST">
        Vezeték név : <input type="text"  name="v_nev"> <br>
        Kereszt név : <input type="text"   name="k_nev"> <br>
        <input type="submit" value="OK">
        </form>
        ';
?>
```

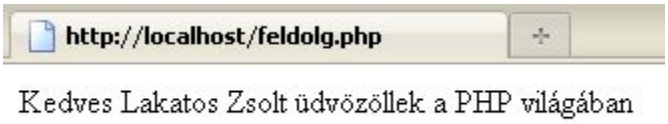
Ha készen vagyunk a begépeléssel, akkor mentsük el ***bevitel.php*** névvel a megfelelő könyvtárba. Ha helyesen jártunk el az eredménynek ilyennek kell lenni:

3. ábra

A form tag action paraméterében azt adtuk meg, hogy az űrlapelemek tartalmát a feldolg.php program fogja végezni. Így most készítsük el azt is. A program egyelőre semmi mást nem fog csinálni, minthogy kiírja a begépelt nevet és egy üdvözlő szöveget.

```
<?php
echo "Kedves " ;
echo $_POST['v_nev'] ;
echo " " ;
echo $_POST['k_nev'] ;
echo " üdvözöllek a PHP világában" ;
?>
```

A végeredmény a következő:



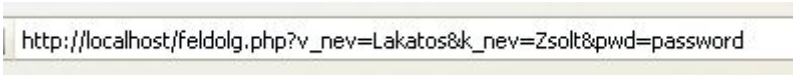
4. ábra

Természetesen a Lakatos Zsolt felirat a beviteli mezőkbe került begépelésre.
Bővítsük a HTML kódot egy jelszó mezővel. A feladatunk annyi mindösszesen, hogy ez után a sor elé : ***<input type="submit" value="OK"> ,*** begépeljük a következőt: *Jelszó : <input type =”password” name=pwd”>
*
A feldolgozó programunkat a következő sorral bővítsük: *echo „
Az Ön által begépelt jelszó a következő: „ . $_POST[’pwd’];*

Általánosan elmondhatjuk az űrlapokkal kapcsolatban a következőket:

- a HTML kódban a ***name*** paraméter szolgál arra, hogy a beírt adatokat a két oldal között át tudjuk adni.
- Az itt megadott név lesz a feldolgozó oldalon a $_POST, vagy $_GET asszociatív tömbben az a név amivel a megfelelő adatra tudunk hivatkozni.

Mi a különbség a POST és GET metódus között? Mindösszesen csak annyi, hogy a beírt adatokat „hol" mozgatja a PHP. Ha GET metódust használunk akkor a beírt adatok, és változók a böngészőnk címsorában „közlekednek", vagyis minden begépelt adat a címsorban látszik, valahogy így:



5. ábra

Jól látszik, hogy látszanak a változónevek és azok értékei is. Azt is megfigyelhetjük, hogy a beviteli oldalon hiába nem látszik a jelszó amit beírtunk, az a böngésző ablakok között titkosítatlanul megy át. Ezek után kijelenthetjük, hogy a GET metódus kerülendő, mert bárki könnyen tudomást szerezhet arról, hogy milyen adatokat adunk át az oldalak között, és ez sok-sok biztonsági problémát vet fel. Ezt a megoldást a program készítési és tesztelési fázisában célszerű alkalmazni. Amikor már meggyőződtünk arról, hogy minden tökéletesen működik, akkor állítsuk át az egészet POST metódusra.

Feladat:

Készítsen egy olyan beviteli és feldolgozó oldalt, melyben szerepelnek a következő űrlapelemek: szövegmező 2 db, jelszó mező, rádió gomb 4 db, check box 2 db, OK gomb és Reset gomb. A beviteli mezők tartalmát a feldolgozó program soronként írassa ki. Mind a két oldal PHP-ban készüljön el!

Megoldás:

Beviteli oldal:

```
<?php
echo '<form action ="feldolg2.php" method="POST">
        Vezeték név : <input type="text"name="v_nev"> <br>
        Kereszt név : <input type="text" name="k_nev"> <br>
        Jelszó : <input type="password"name="pwd"><br>
        Rádió gombok: <br>
        Első <input type="radio" name="radio" value="egyes"> <br>
        Második <input type="radio" name="radio" value="kettes"> <br>
        Harmadik <input type="radio" name="radio" value="hármas"> <br>
        Negyedik <input type="radio" name="radio" value="négyes"> <br>

        Check Box<br>
        Első <input type="checkbox" name="egyes_chk"> <br>
        Második <input type="checkbox" name="kettes_chk"> <br>


        <input type="submit" value="OK">   <input type="reset"
```

```
value="Törlés">
            </form>
            ';
           ?>
```

Feldolgozó program:

```
<?php
echo "Kedves " ;
echo $_POST['v_nev'] ;
echo " " ;
echo $_POST['k_nev'] ;
echo " üdvözöllek a PHP világában" ;
echo "<br>Az Ön által begépelt jelszó a következő: " . $_POST['pwd'] . "<br>";

echo "A következő rádió gomb volt bekapcsolva : " . $_POST['radio'] .  "<br>";

echo "Első check box állapota : " . $_POST['egyes_chk'] .  "<br>";
echo "Második check box állapota : " . $_POST['kettes_chk'] .  "<br>";

?>
```

A feldolgozás során a következőt tapasztalhatjuk: ha a check box nincs kipipálva, akkor a kiírásban nem jelenik meg semmi. Kipipált esetnél megjelenik az ***on*** felirat. Ha szükségünk van a kikapcsolt állapot kezelésére, akkor azt külön kivételként egy szelekcióval le kell kezelnünk egyedileg.

14. – 15. Tömbök kezelése, tömbkezelő függvények

Mint már korábban is említettük a PHP-ben, hasonlóan sok más programozási nyelvhez a tömb az egyik leggyakrabban használt összetett adattípus. Az előző leckében is látszott, hogy az oldalak közötti adatcsere tömbön ($_POST vagy $_GET) keresztül történik. Ezért ennek külön szakaszt kell szentelni, aminek az is az oka, hogy eléggé sok tömbkezelő függvény áll a rendelkezésre a munkánk egyszerűsítésére.

Ismétlésképpen nézzük meg azokat a módozatokat melyekkel tömböket tudunk létrehozni.

Tömb létrehozás az *array()* függvény segítségével

```
$evszak = array(„Tavasz”, „Nyár”, „Ősz”, „Tél”);
```

Tömb létrehozása, vagy elem hozzáadása a szögletes zárójelek segítségével

```
$evszak[] =”Tavasz”;
```

Ebben az esetben, ha a tömb még nem létezett, akkor az első elem (Ne felejtsük el, hogy az indexelés 0-val kezdődik) lesz az egyenlőség jel után megadott érték. Tehát most a 0.-ik elem a Tavasz. További elemeket ugyanezzel a technikával lehet a tömbbe beilleszteni. A zárójelek között nem kell értéket (tömbindex) megadnunk mert a PHP nyilvántartja, hogy éppen hol járunk, és a következő sorszámra fogja a megadott elemet beszúrni. Az is működik, ha mi magunk adjuk meg az elem sorszámát, de lássuk be azt, hogy mi biztosan pontatlanabbul fogunk számolni mint a PHP, és ezért előfordulhatnak olyan esetek, amikor felülírunk egy elemet, vagy pedig lukak lesznek a tömbben (nem veszünk használatba minden sorszámot). Ezeket elkerülendő bízzuk ezt a programozási nyelvre, amely ezt biztosan jól fogja csinálni.

Asszociatív tömb létrehozása

```
$karakter = array
(
"v_nev" => "Kala",
"k_nev" => "Pál",
"kor" => 33,
"nem" => "ferfi”
);
```

Tömbelemek elérése

Normál tömbök esetében a sorszám megadásával lehetséges az elérés, asszociatív tömböknél
az elemek elérése az elnevezéssel történik. Tehát a szögletes zárójelek közé ne sorszámot kell írnunk, hanem aposztrófok (**’)** közé a tömb mezőjének elnevezését.

```
echo $evszak[0]; // az eredmény Tavasz lesz a képernyőn
echo $karakter['v_nev'] . ' ' . $karakter['k_nev'] ; // az eredmény Kala Pál lesz a képernyőn
```

Tömb méretének lekérdezése

Mivel a tömbök mérete dinamikusan változik, ezért fontos, hogy bármikor meg tudjuk mondani a teljes méretet, ami tulajdonképpen az elemek számát jelenti. Erre szolgál a ***count()*** függvény. A függvény visszatérési értéke a tömbben lévő elemek száma lesz, a bemeneti paramétere pedig egy tömb típusú változó kell hogy legyen.

```
$evszak = array(„Tavasz”, „Nyár”, „Ősz”, „Tél”); //létrehoztuk a tömbböt
$meret = count($evszak) ;
echo $meret ; // a kimeneten 4-nek kell megjelennie.
```

Figyelem!! A *count* nem az utolsó használatos index-et adja vissza. Mivel a sorszámozás 0-val kezdődik, emiatt az utolsó index mindig a *count* által visszaadott érték mínusz egy.

Elem keresése a tömbben

Sokszor előfordul az az eset, hogy pusztán arra vagyunk kíváncsiak, hogy egy érték szerepel e a tömbben. Erre egy lehetőség, hogy írunk egy ciklust ami végighalad az összes elemen és megvizsgálja az általunk megadott feltétel szerint azokat. A másik – sokkal egyszerűbb - megoldás az in_array() függvény használata.

Lássuk a következő példát:

```
$evszak = array(„Tavasz”, „Nyár”, „Ősz”, „Tél”); //létrehoztuk a tömbböt
if (in_array(„Tavasz”, $evszak)
{ echo „Az értéket tartalmazza a tömb” ;}
else
{ echo „Az értéket nem tartalmazza a tömb” ;}
```

Elem eltávolítása a tömbből

Ha az elemet kinullázzuk akkor az indexe és tulajdonképpen egy valamilyen értéke megmarad. Ha teljesen el szeretnénk tüntetni az adott elemet akkor az ***unset()*** függvényt kell használnunk. A függvény kétféleképpen alkalmazható. A bemeneten megadható tömbindex, vagy érték. Ha az index alkalmazzuk akkor egyértelműen az adott elemet töröljük ki, ha pedig étéket akkor azokat az indexeket (természetesen lehet több is) melyek az általunk megadott értéket tartalmazzák.
Figyelem! Az ***unset()*** függvény nem sorszámozza újra a tömb elemeit. Tehát látszólag lukas lesz a sorozat, de ezt a problémát a ***foreach*** alkalmazása megoldja.

Két tömb egyesítése

A array_merge() függvény segítségével gyorsan össze tudunk illeszteni két tömböt egy harmadikba. Az új tömb a következőképpen jön létre: az elejére kerül az első paraméterként megadott tömb tartalma, és utána következnek sorban a második tömb elemei.

```
$elso = array(1,2,3,4,5) ;
$masodik = array(6,7,8,9,10) ;
$harmadik = array_merge($elso, $masodik) ;
// a $harmadik tömb így néz ki : 1,2,3,4,5,6,7,8,10
```

Tömb részének kinyerése

Ahhoz hogy egy tömb közbülső darabját ki tudjuk másolni az array_slice függvény kell.
Három paramétere van:

- a bemeneti tömb amelyből ki kell nyernünk egy darabot
- az az index ahonnan a vágást kezdjük
- hány darab elemet szeretnénk kinyerni

```
$elso = array(1,2,3,4,5,6,7,8,9,10) ;
$masodik = array_slice($elso, 3,3) ;

//A $masodik tartalma: 4,5,6 lesz
```

Tömb rendezése

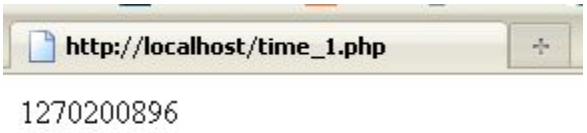
Talán ez egyik leghasznosabb találmány a PHP-ban. Nem kell bonyolult rendezési algoritmusokat írnunk, mert egy utasítással sorba rakhatjuk a tömb elemeit. Ha a tömb elemei karakterek akkor az angol ABC, ha pedig számok akkor nagyság szerinti a rendezés.

```
$elso = array(3,7,4,1,9,20,13) ;
sort($elso) ;

//A tömb így fog kinézni: 1, 3, 4, 7, 9, 13, 20
```

16. – 17. Dátum, idő kezelése

Sokszor előfordul az az eset, hogy valamilyen dátummal vagy idővel kapcsolatos kérdés merül fel a programírás során. Ilyen például amikor egy web oldalon látjuk, hogy ki van írva az aktuális névnap. Az első és talán legfontosabb idetartozó függvény a ***time()***. Ennek a kimenete teljesen meglepő módon egy egész szám, amely a külső szemlélő számára semmitmondó. Persze, ha tudjuk, hogy miképpen keletkezett ez a nagy szám akkor már teljesen másképp tekintünk rá. Mi is ez? A ***time()*** által visszaadott érték az 1970.01.01 00:00-tól eltelt másodpercek számát adja meg. Ezt az értéket sok hivatkozásban időbélyegként vagy time stamp-ként is fellelhetjük.



6. ábra

Természetes ezzel így nem sok mindent tudunk kezdeni. Ezt az értéket át kell alakítanunk „emészthető formára". Szerencsére ez is nagyon könnyen megoldható. Rendelkezésünkre áll a ***date()*** függvény, melynek segítségével bármilyen formában ki tudjuk íratni a dátumot és/vagy időt.
A ***date()***függvénynek rengeteg paramétere van amit a következő táblázat mutat meg.[8]

Formátum	Leírás	Példa
a	.am. (délelőtt) vagy .pm. (délután), kisbetűvel	pm
A .	AM. (délelőtt) vagy .PM. (délután), nagybetűvel	PM
d	A hónap napja (bevezető nullákkal írt szám)	05
D	A hét napja (három betűvel)	Thu
F	A hónap neve	January
h	Óra (12 órás formátum, bevezető nullákkal)	03
H	Óra (24 órás formátum,	05

[8] forrás: PHP dokumentáció, www.php.net

	bevezető nullákkal)	
g	Óra (12 órás formátum, bevezető nullák nélkül)	3
G	Óra (24 órás formátum, bevezető nullák nélkül)	5
i	Perc	47
j	A hónap napja (bevezető nullák nélkül)	5
l	A hét napja (névvel)	Thursday
L	Szökőév (.1. ha igen, .0. ha nem)	1
m	Az év hónapja (számmal, bevezető nullákkal)	01
M	Az év hónapja (három betűvel)	Jan
n	Az év hónapja (számmal, bevezető nullák nélkül)	1
s	Az óra percei	24
U	Időbélyeg	948372444
y	Év (két számjegy)	00
Y	Év (négy számjegy)	2000
z	Az év napja (0-365)	19
Z	A greenwichi középidőtől való eltérés másodpercben	0

Hogyan kell ezt használni?

```
<?php

echo date ("Y.m.d. H:i:s<br>", time());

?>
```

Ennek az eredménye a következő lesz a böngészőben:

7. ábra

A táblázatban felsorolt értékekkel nagyon könnyen testre szabható a dátum és idő kiíratása az oldalunkra. A beépített formátumoknak egyetlenegy hátránya van. A hónapok és napok neveinek kijelzését csak angol nevekkel tudja. Ahhoz, hogy a nálunk használt formátumban történjen a kiírás, némi trükközésre van szükség. Definiálnunk kell egy tömböt a hónapok magyar neveinek és egyet a napoknak, és abból kell kiválasztani a megfelelő értékeket.

```
<?php

$honapok = Array("","január", "február", "március","április","május", "június", "július",

"augusztus","szeptember", "október", "november", "december");

echo date("Y") ."-" . $honapok[date("n")] . "-" . date("d") ;


?>
```

A kimenet a következő lesz

8. ábra

Ugyanezzel a módszerrel megoldható a hét napjainak kiírása is.

Dátumok ellenőrzése

Nagyon sokszor előfordul olyan eset, amikor a felhasználó által bevitt dátumot kell feldolgoznunk. Ehhez szükség van valamilyen ellenőrzésre, mert után nézne ki, ha valakinek a születési dátumában pl. 14. hónap 52. napot találnánk. Erre is megvan a PHP beépített függvénye, mégpedig a ***checkdate()***. Ez mindösszesen annyit tesz, hogy leellenőrzi a következőket:

- a hónap 1 és 12 közé esik
- a nap befogadható az adott hónapra és évre (a függvény tudja, hogy mely hónap 30 vagy 31 napos) és ismeri a szökőéveket is.
- az év 0 és 32767 közé esik

Használata:

```
if (checkdate(4,5,2010))
{echo „a beírt dátum helyes”;}
```

A paramétereket a következő sorrendben kell megadni: nap, hónap, év

Feladat: készítsünk programot, mely kiírja a mai névnapot

A megoldáshoz szükség van a következő tömbökre, melyek a névnapokat tartalmazzák.

```
$januar=array('','Fruzsina','Ábel,Alpár','Genovéva,Benjámin','Titusz,Leona','Simon','B
oldizsár','Attila,Ramóna','Gyöngyvér','Marcell','Melánia','Ágota','Ernő','Veronika','Bódo
g','Lóránt,Loránd','Gusztáv','Antal,Antónia','Piroska','Sára,Mária','Fábián,Sebestyén','
Ágnes','Vince,Artúr','Zelma,Rajmund','Timót','Pál','Vanda,Paula','Angelika','Károly,Kar
ola','Adél','Martina,Gerda','Marcella');
```

```
$februar=array('','Ignác','Karolina,Aida','Balázs','Ráhel,Csenge','Ágota,Ingrid','Dorotty
a,Dóra','Tódor,Rómeó','Aranka','Abigél,Alex','Elvira','Bertold,Marietta','Lídia,Lívia','Ella
,Linda','Bálint,Valentin','Kolos,Georgina','Julianna,Lilla','Donát','Bernadett','Zsuzsanna
','Aladár,Álmos','Eleonóra,Zelmira','Gerzson','Alfréd','Mátyás','Géza','Edina','Ákos,Bát
or','Elemér');
$marcius=array('','Albin','Lujza','Kornélia','Kázmér','Adorján,Adrián','Leonóra,Inez','Ta
más','Zoltán','Franciska,Fanni','Ildikó','Szilárd','Gergely','Krisztián,Ajtony','Matild','Krist
óf','Henrietta','Gertrúd,Patrik','Sándor,Ede','József,Bánk','Klaudia','Benedek','Beáta,Iz
olda,Lea','Emőke','Gábor,Karina','Irén,+Źrisz','Emánuel','Hajnalka','Gedeon,Johanna','
Auguszta','Zalán','Árpád');
$aprilis=array('','Hugó','Áron','Buda,Richárd','Izidor','Vince','Vilmos,Bíborka','Herman','
Dénes','Erhard','Zsolt','Leó,Szaniszló','Gyula','Ida','Tibor','Anasztázia,Tas','Csongor','
Rudolf','Andrea,Ilma','Emma','Tivadar','Konrád,Zelmira','Csilla,Noémi','Béla','György','
Márk','Ervin','Zita','Valéria','Péter','Katalin,Kitti');
$majus=array('','Fólöp,Jakab,Zsaklin','Zsigmond','Tímea,Irma','Mónika,Flórián','Györg
yi','Ivett,Frida','Gizella','Mihály','Gergely,Katinka','Ármin,Pálma','Ferenc','Pongrác','Sze
rvác,Imola','Bonifác','Zsófia,Szonja','Mózes,Botond','Paszkál','Erik,Alexandra','Ivó,Mil
án','Bernát,Felícia','Konstantin','Júlia,Rita','Dezső','Eszter,Eliza','Orbán','Fülöp,Evelin','
Hella','Emil,Csanád','Magdolna','Janka,Zsanett','Angéla,Petronella');
$junius=array('','Tónde','Kármen,Anita','Klotild,Cecília','Bulcsú','Fatime,Fatima','Norbe
rt,Cintia','Róbert','Medárd','Félix','Margit,Gréta','Barnabás','Villő','Antal,Anett','Vazul','J
olán,Vid','Jusztin','Laura,Alida,Alina','Arnold,Levente','Gyárfás','Rafael','Alajos,Leila','
Paulina','Zoltán','Iván','Vilmos','János,Pál','László','Levente,Irén','Péter,Pál','Pál');
$julius=array('','Tihamér,Annamária','Ottó','Kornél,Soma','Ulrik','Emese,Sarolta','Csab
a','Apollónia','Ellák','Lukrécia','Amália','Nóra,Lili','Izabella,Dalma','Jenő','Örs,Stella','He
nrik,Roland','Valter','Endre,Elek','Frigyes','Emília','Illés','Dániel,Daniella','Magdolna','L
enke','Kinga,Kincső','Kristóf,Jakab','Anna,Anikó','Olga,Liliána','Szabolcs,Alina','Márta,
Márti,Flóra','Judit,Xénia','Oszkár');
$augusztus=array('','Boglárka','Lehel','Hermina','Domonkos,Dominika','Krisztina','Bert
a,Bettina','Ibolya','László','Emőd','Lőrinc','Zsuzsanna,Tiborc','Klára','Ipoly','Marcell','Má
ria','Ábrahám','Jácint','Ilona','Huba','István','Sámuel,Hajna','Menyhért,Mirjam','Bence','
Bertalan','Lajos,Patrícia','Izsó','Gáspár','Ágoston','Beatrix,Erna','Rózsa','Erika,Bella');
$szeptember=array('','Egyed,Egon','Rebeka,Dorina','Hilda','Rozália','Viktor,Lőrinc','Za
kariás','Regina','Mária,Adrienn','Ádám','Nikolett,Hunor','Teodóra','Mária','Kornél','Szer
éna,Roxána','Enikő,Melitta','Edit','Zsófia','Diána','Vilhelmina','Friderika','Máté,Mirella','
Móric','Tekla,Líviusz','Gellért,Mercédesz','Eufrozina,Kende','Jusztina','Adalbert','Venc
el','Mihály','Jeromos');
$oktober=array('','Malvin','Petra','Helga','Ferenc','Aurél','Brúnó,Renáta','Amália','Kopp
ány','Dénes','Gedeon','Brigitta','Miksa','Kálmán,Ede','Helén','Teréz','Gál','Hedvig','Luk
ács','Nándor','Vendel,Irén','Orsolya','Előd','Gyöngyi','Salamon','Blanka,Bianka','Dömöt
ör','Szabina','Simon, Szimonetta','Nárcisz','Alfonz','Farkas');
$november=array('','Marianna','Achilles','Győző','Károly','Imre','Lénárd','Rezső','Zsom
bor','Tivadar','Réka','Márton','Jónás,Renátó','Szilvia','Aliz','Albert,Lipót','Ödön','Horten
zia,Gergő','Jenő','Erzsébet','Jolán','Olivér','Cecília','Kelemen,Klementina','Emma','Kat
alin','Virág','Virgil','Stefánia','Taksony','András,Andor');
$december=array('','Elza','Melinda,Vivien','Ferenc','Borbála,Barbara','Vilma','Miklós','
Ambrus','Mária','Natália','Judit','Árpád,Árpádina','Gabriella','Luca,Otília','Szilárda','Valé
r','Etelka,Aletta','Lázár,Olimpia','Auguszta','Viola','Teofil','Tamás','Zénó','Viktória','Ádá
m,Éva','Eugénia','István','János','Kamilla','Tamás,Tamara','Dávid','Szilveszter');
```

18. – 19. Karakterláncok kezelése, kapcsolódó függvények

Karakterformázás a ***printf()*** függvény segítségével. A C/C++ nyelvből már ismerős lehet ez a függvény. Nagyon sokban hasonlít az ott megismertekhez a PHP-s változat, de vannak eltérések is. A bemenet egy formázó karakterlánc, amely a következő paraméterként megadott számot írja ki a megfelelő formátumban.

```
printf(„a szám kiírása %d”,  123) ;
```

A % jelel együtt megadott karakter az un. formátum kód. Nézzük milyen formátumkódokat ismer a PHP.[9]

Paraméter	Leírás
d	A paramétert decimális számként jeleníti meg.
b	Egész számokat bináris számként jelenít meg.
c	Egy egész számot annak ASCII megfelelőjeként jelenít meg.
f	A paramétert lebegőpontos számként ábrázolja.
o	Egy egész számot oktális (8-as számrendszerű) számként jelenít meg.
s	A paramétert karakterlánc-állandónak tekinti.
x	Egy egész számot kisbetűs hexadecimális (16-os számrendszerű) számként jelenít meg.
X	Egy egész számot nagybetűs hexadecimális (16-os számrendszerű) számként jelenít meg

A karakterláncok tárolási módja tulajdonképpen megegyezik a tömbök tárolási módjával. Nézzük a következő példát:

```
$szoveg = „Ez egy karakterlánc” ;
echo $szoveg ; //megjeleníti a változó tartalmát
echo $szoveg[5] ; //megjeleníti a karaktersorozat 5-ös indexű karakterét, vagyis az y
//karaktert
```

Figyelem!! Itt is, mint a tömböknél az első index minden esetben 0.

Szöveg hosszának megállapítása

[9] forrás: PHP referencia könyv, 932. oldal

Az strlen() függvény segítségével lehetőségünk van arra, hogy egy ismeretlen hosszúságú karakterlánc hosszát megállapítsuk és átadjuk egy változóba. A bemenet egy karakterlánc, vagy egy ilyen típusú változó kell hogy legyen.

```
$szoveg = „PHP” ;
$hossz = strlen($szoveg) ;
echo $hossz; // a kimeneten 3-nak kell megjelennie
//egyszerűbben
$szoveg = „PHP” ;
echo strlen($szoveg) ;
//a végeredmény ugyanaz
```

Karaktersorozat keresése egy karakterláncban

Sok esetben van arra szükségünk, hogy megállapítsuk azt, hogy egy karakterlánc tartalmaz e bizonyos sorozatot. Pl.: ezzel a technikával meg tudjuk állapítani, hogy egy beírt e-mail cím tartalmazza e a @ karaktert. Erre szolgál az ***strpos()*** függvény.A függvény nem csak azt mondja meg hogy a részlánc megtalálható, hanem ha igen akkor annak a helyét is.

```
$email = valaki@valahol.hu

if (strpos($email,”@”))
{echo „az e-mail cím rendben van”;}
else {echo „nem szabályos e-mail cím” ;}
```

Szöveg tisztítása

A trim() függvény segítségével eltávolíthatjuk a karakterláncból a felesleges elválasztó karaktereket. A bemenete egy szöveg, vagy szöveg típusú változó

```
$szoveg = "\t\t\teléggé levegős ez a szöveg ";
$szoveg = trim( $szoveg );
print $szoveg
// a kimeneten a következő jelenik meg: eléggé levegős ez a szöveg
```

Karakterlánc egy részének kicserélése

A ***substr_replace()*** függvény megkeresi a részláncot és kicseréli az általunk megadottra. Három paramétere van a függvénynek:

- átalakítandó karakterlánc (direktben megadva, vagy változóként)
- a csereszöveg
- index ahonnan a cserét kezdeni kell

```
$szoveg = „valamilyen elrontott szoveg” //a szoveg szóban kellene az o-t ö-re cserélni
$szoveg = substr_replace($szoveg,”ö”, 24) ;
//az eredmény a következő lesz: valamilyen elrontott szöveg
```

Minden részlánc megkeresése és cseréje

Az str_replace() függvény hasonlóan működik mint az előző, de az átalakítandó karakterláncban cseréli az össze rész előfordulást. Ennek a függvénynek is három paramétere van:

- a keresett (cserélendő karakterlánc)
- csereszöveg, vagyis mire kell cserélnünk az előfordulásokat
- forrásszöveg

```
$szoveg = „valamilyen elrontott szoveg” //a szoveg szóban kellene az o-t ö-re cserélni
$szoveg = str_replace(„o”, „ö”,$szoveg) ;
//az eredmény a következő lesz: valamilyen elröntött szöveg
```

Kis és nagybetűk cseréje

Két függvény áll a rendelkezésünkre. Az ***strtoupper()*** segítségével minden kisbetűt nagyra, az ***strtolower()***-el pedig minden nagybetűt kicsire tudunk cserélni.

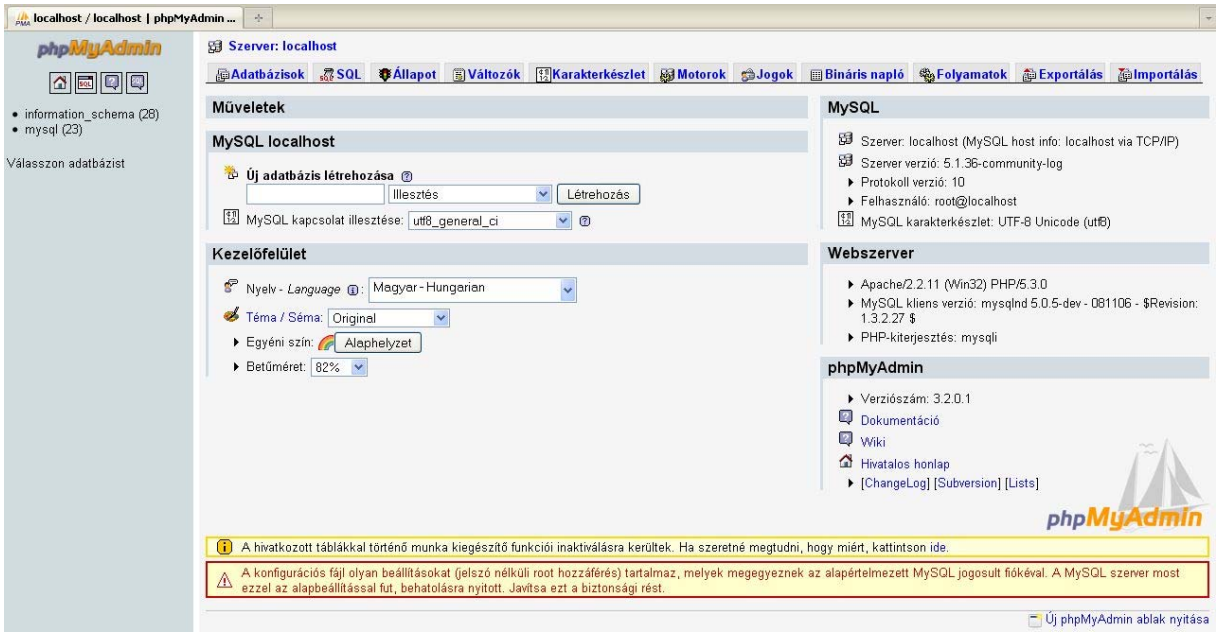
```
$szoveg = „php”;
$szoveg = strtoupper($szoveg) ;
//az eredmény PHP lesz
strtolower($szoveg) ;
//az eredmény ismét php lesz
```

20. Teszt

1. Mi a különbség a GET és POST metódusok között?
2. Milyen típusú változóban mennek át az adatok két oldal között?
3. Hogyan lehet egy tömb elemszámát megtudni?
4. Milyen függvény segítségével lehet két tömböt egyesíteni?
5. Mely függvény segítségével lehet egy tömböt rendezni?
6. Mely függvényt használnánk a dátum formázására?
7. Milyen átalakító paramétert használnánk a printf() függvényben egy egész szám lebegőpontos számként való megformázására?
8. Milyen függvényeket használnánk egy szöveg hosszának kiderítéséhez?
9. Hogyan alakítanánk át egy karakterláncot csupa nagybetűsre?

21. Adatbázis kapcsolatok létrehozása

A bevezetőben már említettem, hogy a PHP segítségével nagyon könnyen lehet szabványos adatbázisokhoz csatlakozni. Az esetek nagy többségében a MySQL adatbázis szervert használjuk, de ez nem jelenti azt, hogy más adatbázis típusok ki lennének zárva. A dokumentáció tanulmányozásával minden itt felsorolt MySQL függvény megfelelőjét megtalálhatjuk szinte bármilyen kiszolgáló típusra.
A MySQL adatbázisok kezeléséhez nagy segítséget tud nyújtani a PHPMyAdmin csomag. Ez ingyenesen letölthető a http://www.phpmyadmin.net/home_page/downloads.php oldalról.



9. ábra

Ebben a leírásban nem foglalkozom az adatbázis kezelés és SQL elméletével és alapismeretével. Feltételezem azt, hogy az alapfogalmakkal mindenki tisztában van.

Csatlakozás MySQL kiszolgálóhoz

Az adatbázis kezelés első legfontosabb lépése a kapcsolódás. Erre a ***mysql_connect()*** függvény szolgál. Három paramétert vár el tőlünk:

- az adatbázis szerver nevét, vagy elérését

- felhasználónév
- jelszó

A második két paraméter ismerete nélkül nincs lehetőségünk az adatbázis szerverhez kapcsolódni. Ha nem ismerjük ezeket, akkor kérjünk segítséget a rendszergazdától. A példában a root felhasználóval csatlakozunk a szerverhez, jelszó nélkül. Ez az eset csak a legritkább esetben fordul elő, leginkább a saját számítógépünkön lévő tesztkörnyezetben.

```
$link = mysql_connect(„localhost”,”root”, „”)
or die („Nem sikerült kapcsolódni az adatbázis szerverhez”);
```

A függvény kimenete egy erőforrás típusú adat, amely azonosítja a továbbiakban a kapcsolatot. Természetesen erre az erőforrásra a későbbiekben is szükségünk lesz.

Adatbázis kiválasztása

Miután kapcsolódtunk a kiszolgálóhoz meg kell határoznunk, hogy mely adatbázissal kívánunk a továbbiakban dolgozni. Ennek kiválasztására a ***mysql_select_db()*** függvény szolgál. Bemenetként egyetlen egy paramétert vár, és az a kiválasztandó adatbázis neve. Opcionális paraméterként megadható a kapcsolódás erőforrás azonosítója is. Ha nincs megadva akkor a legutolsó kapcsolódási azonosítót próbálja meg használni. Erre akkor lehet szükség, ha egy időben több szerverhez is kapcsolódunk.

```
$link = mysql_connect(„localhost”,”root”, „”)
or die („Nem sikerült kapcsolódni az adatbázis szerverhez”) ;

mysql_select_db(„teszt”)
or die („nem sikerült kiválasztani az adatbázist”) ;
```

A példában a teszt nevű adatbázishoz próbálunk kapcsolódni. Mindkét esetben a ***DIE*** elágazás a hibakezelést szolgálja. Ezt megvalósíthatnánk ***IF***-es szerkezettel is, de ez így gyorsabb és egyszerűbb.

Az adatbázis táblákat nem kell külön kiválasztani, hanem a megfelelő utasításnál (select, update, stb.) kell azokat meghatározni, pontosan ugyanúgy mintha normál SQL utasításokat írnánk.

22. – 23. MySQL lekérdezések létrehozása

A MySQL-el kapcsolatos példákhoz a MySQL demó adatbázisát fogjuk használni. Ez letölthető a következő helyről: http://dev.mysql.com/doc/index-other.html.

Example Databases

	Download DB	HTML Setup Guide	PDF Setup Guide
employee data (large dataset, includes data and test/verification suite)	Launchpad	View	US Ltr | A4
world database (all versions, used in MySQL certifications and training)	Gzip | Zip	View	US Ltr | A4
sakila database (requires MySQL 5.0 or later)	TGZ | Zip	View	US Ltr | A4
menagerie database (all versions)	TGZ | Zip		

10. ábra

Itt a második sorban lévő „world database (…. „lehetőséget válasszuk. A kattintás után a böngészőnkben beállított helyre letöltődik egy **world.sql.zip** nevű állomány, amit valahová ki kell csomagolnunk. Az adatbázis használatbavételére kétféle lehetőség adott:

- a mysql parancssori betöltés
- ha rendelkezésünkre áll a PHPMyAdmin, akkor ott az import lehetőség

Mysql konzol változat

A következő lépések sorozata szükséges:

1. csomagoljuk ki a letöltött fájlt a c:\wamp\
2. indítsuk el a MySQL konzolt a WAMP szerver vezérlőpultjából

11. ábra

3. a konzolban adjuk ki egymás után a következő utasításokat:
    - **CREATE DATABASE world;**
    - **USE world;**
    - **SOURCE c:\\wamp\\world.sql (**A dupla perjelekre a szerver mintaillesztése miatt van szükség)
    - **show tables;**

PHPMyAdmin változat

A következőket kell tennünk:

1. csomagoljuk ki a letöltött fájlt a c:\wamp\
2. Kattintsunk a WAMP szerver konzoljában a PHPMyAdmin feliratra (ennek hatására a böngészőnkben megjelenik az adminisztrációs felület)

12. ábra

3. A PHPMyAdmin kezelőfelületén kattintsunk az Adatbázis fülre, és a képernyő alsó részében lévő beviteli mező segítségével hozzuk létre a world nevű adatbázis.
4. A baloldali részben, ahol a meglévő adatbázisok vannak felsorolva válasszuk ki az előzőekben létrehozottat, és a felső menüsoron válasszuk az ***import*** pontot.
5. A megjelenő ablakban a szövegfájl nyomógomb segítségével tallózzuk ki a kicsomagolt world.sql fájlt, és az indítás gombra kattintva elindul a folyamat
6. Ha mindent jól csináltunk akkor az sql fájl tartalma néhány másodperc alatt bekerül az adatbázisba és készen áll arra, hogy munkát végezzünk vele

A PHP-ban a megfelelő függvények használatával az SQL lekérdezések pontosan ugyanúgy működnek mintha az adatbázis szerver konzol képernyőjén hajtanánk végre őket.
A folyamat így néz ki:

1. SQL lekérdezés „összerakása”
2. A lekérdezés futtatása a szerveren
3. Az eredmény(ek) feldolgozása

Lássunk egy példát. Kérdezzük le a city táblában szereplő városneveket és a hozzájuk tartozó népességi adatokat, és ezeket jelenítsük meg a böngészőnkben.

```
<?php
$link = mysql_connect("localhost", "root","")
or die("nem sikerült kapcsolódni az adatbázishoz") ;

mysql_select_db("world")
or die("nem sikerült kiválasztani az adatbázist") ;

$sql = "select name, population from city " ;
$ered = mysql_query($sql) ;
echo "<table border=1> <tr> <th>Városnév</th><th>Lakosság</th> </tr>" ;


while ($eredmeny= mysql_fetch_row($ered) )
{
        echo  "<tr><td>" . $eredmeny[0] . "</td> <td>" . $eredmeny[1] . "</th></tr>" ;
}

echo "</table>" ;

?>
```

Nézzük a program működését:

Az első néhány sor az adatbázishoz való kapcsolódáshoz és a adatbázis kiválasztásához szükséges. Az első újdonság ez a sor: *$sql = "select name, population from city " ;*Itt nem történik más mint összerakjuk a lekérdezésünket az adatbázis szerver felé. A lekérdezés eredménye a ***$ered*** erőforrás típusú változóba fog megérkezni majd. Ennek a tartalmát direktben nem tudjuk megjeleníteni, további feldolgozást igényel a dolog. A következő ***echo***-val kezdődő sor a táblázatos megjelenítéshez szükséges. Ezek után következik egy ***while*** ciklus melynek a feltételében kibontjuk az erőforrás típusú adatunkat egy ***$eredmény*** változóba. Az iteráció egészen addig zajlik, amíg a ***$ered*** tartalmaz adatot. A ciklus magjában történik meg a kiíratás a táblázatba.

Az a függvény ami az iteráció feltételében van a következő: ***mysql_fetch_row()***. Ez egy normál tömbbe másolja be az adatokat. Ennek a függvénynek van még néhány változata:

- mysql_fetch_assoc(): asszociatív tömbbe kerülnek az adatok
- mysql_fetch_object(): objektum típusba kerülnek az adatok
- mysql_tech_array(): normál tömbbe kerülnek az adatok, a feldolgozás kényelmesebb mint a mysql_fetch_row() esetében

A programozók szempontjából talán az az eset a legkényelmesebb és legkönnyebben kezelhető amikor asszociatív tömb a kimenet.

Feladat: alakítsuk át az előző programunkat, hogy az eredmények egy legördülő listába kerüljenek be.
Megoldás:

```
<?php
$link = mysql_connect("localhost", "root","")
or die("nem sikerült kapcsolódni az adatbázishoz") ;

mysql_select_db("world")
or die("nem sikerült kiválasztani az adatbázist") ;

$sql = "select name, population from city order by name" ;
$ered = mysql_query($sql) ;

echo "<form> <select>";

while ($eredmeny= mysql_fetch_row($ered) )
{
echo "<option>" . $eredmeny[0] . "</option>" ;

}

echo "</select><input type=submit> </form>" ;

?>
```

A képernyőn ennek kell megjelennie:

13. ábra

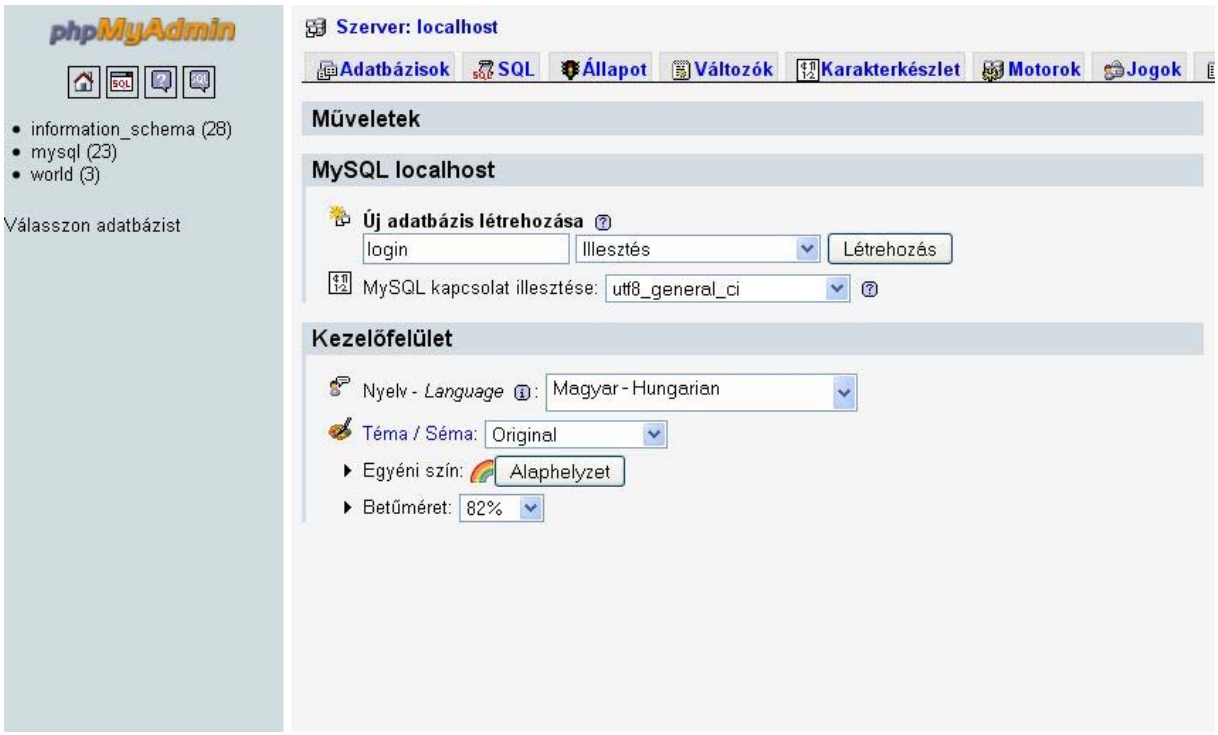
Ugyanúgy mint az előző esetbena feldolgozás kissé sokáig tart, mert kb. 4000 sort kell feldolgoznunk.

24. – 25. Adattáblák módosítása feltöltése

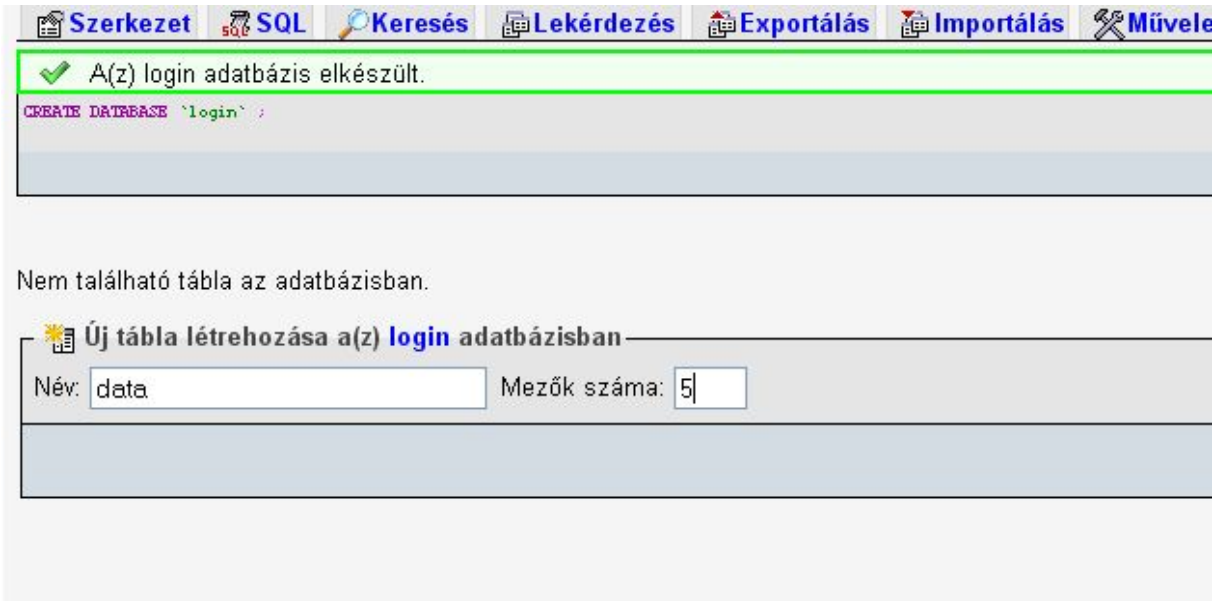
Az adatbázis műveletek egyik nagy csoportja a lekérdezések mellett az adatfelvitel, adatmódosítás. Ezek a feladatok leginkább az adat táblákban található tartalomra korlátozódnak. Az adatbázisok szerkezetét a legritkább esetben módosítjuk, alakítjuk PHP programból. Ennek az oka az, hogy a tervezési fázis után az adatbázis szerver konzolján keresztül, vagy valamely külső program segítségével létrehozzuk az adatbázist és a táblákat. Ezután szerkezeti módosítás nem igazán kell már.
A most következő részben is hasonlóan fogunk eljárni. A PHPMyAdmin segítségével létrehozzuk az adatbázist és egy táblát, majd ezt fogjuk PHP segítségével adatokkal feltölteni. A feladat amit megvalósítunk a következő: egy egyszerű regisztrációs programot készítünk. Az új felhasználók megadják néhány adatukat, és ezek be kerülnek egy MySQL adatbázisba.

Első lépésként készítsük el az adatbázist és a táblát. Az adatbázis neve legyen: **LOGIN**, a tábla neve pedig: **DATA**.
A tábla a következő szerkezetű legyen:

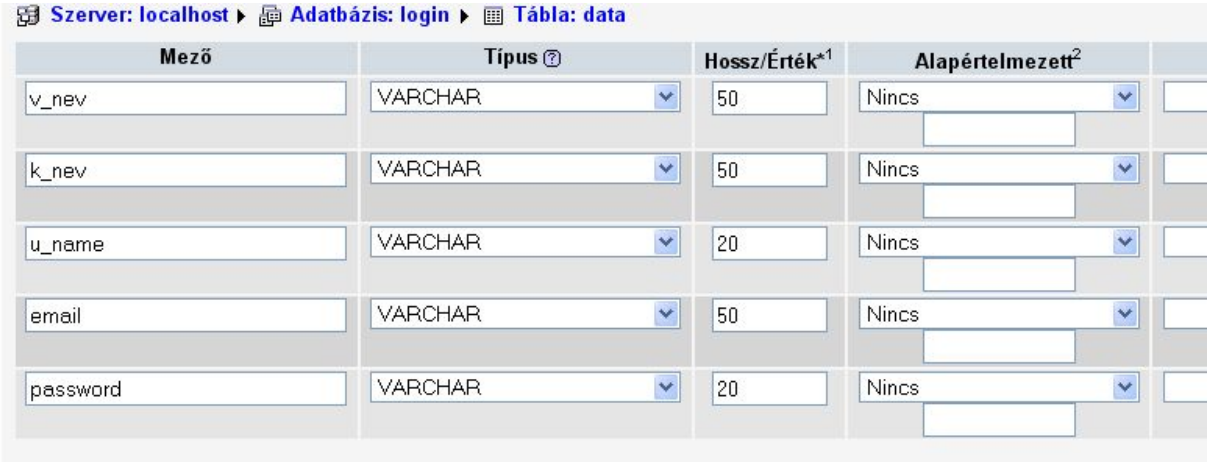
Mezőnév	Mezőtípus	Hossz	Leírás
v_nev	varchar	50	Vezetéknév
k_nev	varchar	50	Keresztnév
u_name	varchar	20	Bejelentkezési név
email	varchar	50	E-mail cím
password	varchar	20	Jelszó



14. ábra



15. ábra



16. ábra

Az adatbázis elkészülte után az első lépés annak a HTML oldalnak a létrehozása, melyben a felhasználó megadja az adatait.

A kész oldalnak valahogyan így kell kinéznie:

17. ábra

A forráskód a következő:

```
<html>
<head>
<title> Felhasználói regisztráció </title>
</head>
<body>
<p align="center"> <b> <big>Felhasználói adatok megadása </big></b></p>
<form action="feldolgoz.php" method="POST">
<table border=0 width=60%>
<tr>
<td> Vezeték név</td> <td> <input type="text" name="v_nev"></td>
</tr>
<tr>
<td> Kereszt név</td> <td> <input type="text" name="k_nev"></td>
</tr>
<tr>
<td> Felhasználó név</td> <td> <input type="text" name="u_name"></td>
```

```
</tr>
<tr>
<td> Email cím       </td> <td> <input type="text" name="email"></td>
</tr>
<tr>
<td> Jelszó</td> <td> <input type="password" name="password"></td>
</tr>
<tr>
<td> <input type="submit" value="Elküld"></td> <td> <input type="reset"
value="Törlés"></td>
</tr>
</table>
</form>
</body>
</html>
```

Természetesen ez a feladat sokféleképpen megoldható. Ez az egyik legegyszerűbb mód az adatok bekérésére és a szöveg igazítására. Ha igazán szépen akarnánk ezt megcsinálni akkor nem táblázat eszközökkel illesztenénk a szöveget, hanem az előző modulban megtanult CSS eszközeivel.
A forráskódból látszik, hogy az adatfeldolgozást a **feldolgoz.php** nevű alkalmazás fogja elvégezni. Most ezt a programot kell elkészíteni.

A programnak a következő részekből kell állnia:

- Adatbázis szerverhez kapcsolódás, adatbázis kiválasztás
- Az adatbeviteli lekérdezés összerakása
- A lekérdezés futtatása
- Felhasználó tájékoztatása a sikeres regisztrációról

A feladat első pontja már ismerős az előző fejezetből. Így néz ki a hozzátartozó kódrészlet:

```
$link = mysql_connect("localhost", "root","")
or die("nem sikerült kapcsolódni az adatbázishoz") ;

mysql_select_db("world")
or die("nem sikerült kiválasztani az adatbázist") ;
```

A következő részhez leginkább SQL tudás szükséges. Adatbeszúrás a következő módon történhet a táblába direkt adatmegadással:

```
INSERT INTO `login`.`data` (`v_nev` ,`k_nev` ,`u_name` ,`email` ,`password` ) VALUES ('Lakatos',
'Zsolt', 'zsoltl', 'zsolt.lakatos@gmail.com', 'jelszo');
```

Nincs is más dolgunk mint ezt lefordítani PHP-re. Az eltérés annyi lesz, hogy a beírandó adatokat nem direktben fogjuk megadni, hanem azok helyén valamilyen változóknak kell szerepelni. Milyen változóknak? Azoknak amik a $_POST asszociatív tömbbel érkeznek meg a feldolgozó programba. Nézzük a kódot:

```
<?php
```

```
$link = mysql_connect("localhost", "root","")
or die("nem sikerült kapcsolódni az adatbázishoz") ;

mysql_select_db("login")
or die("nem sikerült kiválasztani az adatbázist") ;

$v_nev = $_POST['v_nev'] ;
$k_nev = $_POST['k_nev'] ;
$u_name = $_POST['u_name'] ;
$email = $_POST['email'] ;
$password = $_POST['password'] ;

$sql = "insert into data values  ('$v_nev', '$k_nev', '$u_name', '$email', '$password');
" ;

$query = mysql_query($sql) ;

if (mysql_errno() ==0)
{ echo "Az adatfelvitel sikerült" ;}

else
        {echo mysql_error() ;}

?>
```

Ebben a kódban is találhatunk olyan részeket ami ismeretlen lehet:
Az SQL lekérdezés összerakásának egyszerűsítése miatt találhatjuk a program elején azt a részt, amikor a $_POST tömb elemeit külön változókba kimásolom.
A legvégén pedig egy szelekciós szerkezetet találunk, aminek a feltétele a mysql_errno() függvény. Ennek a függvénynek a kimenete 0 abban az esetben, ha az előtte lévő SQL parancs hiba nélkül lefutott. Ha valamilyen hiba volt akkor azt a mysql_error függvény segítségével lehet a képernyőn szöveges formában megjeleníteni.

Ebből a rövid programból is látszik az, hogy igazából minden az adatbázishoz intézett parancson múlik. Tulajdonképpen ugyanazokat a függvényeket kell alkalmaznunk, ha lekérdezést rakunk össze, ha adatot viszünk fel, vagy esetleg adatot módosítunk egy táblában.

26 Képek kezelése

A PHP-ban számos lehetőség áll rendelkezésre a képek kezeléséhez. „Röptében” tudunk képeket készíteni, információkat tudunk kinyerni a képfájlokból, FLASH animációt tudunk lejátszani, vezérelni.
A PHP a következő képformátumokat képes kezelni: GIF, PNG, JPG. A GIF formátumoz csak megjeleníteni lehetséges. Régebben lehetőség volt a GIF-ek manipulálására is, de mivel a formátum jogdíjas lett az ide kapcsolódó függvény könyvtárakat törölték a rendszerből.

A képekkel kapcsolatban az egyik legfontosabb függvény a getimagesize(). A bemeneti paraméter a képfájl neve ha szükséges a teljes elérési útvonallal együtt. A bemeneti formátum JPG. GIF, PNG és SWF lehet.
Figyelem: UNIX/LINUX rendszerek alatt a kis és nagybetű eltérések nagy odafigyelést igényelnek.
Az eredménye egy négy elemű tömb lesz mely a következőképpen épül fel:

- kép szélessége pixelben
- kép magassága pixelben
- Egy jelző mely a típusra utal: 1=GIF, 2=JPG, 3=PNG, 4=SWF
- egy string mely a következő formát ölti: height=xxx width=xxx. Ez azért hasznos, mert egy az egyben be lehet illeszteni egy HTML img tag-be.

Példa:

```
<?php
$forma = array('','GIF', 'JPEG', 'PNG', 'Flash') ;
$size= getimagesize("php_9.jpg") ;
echo "A kép szélessége: " . $size[0] . "pixel <br>" ;
echo "A kép magassága: " . $size[1] . "pixel <br>" ;
echo "A kép típusa: " . $forma[$size[2]] . "<br>" ;
echo "A képhez tartozó string: " . $size[3] . "<br>" ;
?>
```

A getimagesize() paraméterében található képfájl a wwwroot mappában található, és így nem kell elérési útvonalat megadni hozzá.

A többi függvény segítségével akár rajzolhatunk is a képernyőre, vagy akár egy új képet hozunk létre fájlból vagy egy internetes hivatkozásból. A lehetőségek széles száma miatt ezeket a függvényeket itt most nem ismertetem. A leírás megtalálható angol nyelven a http://php.net/manual/en oldalon angol nyelven, vagy pedig részben magyarul a http://www.php-blog.hu/php-magyar-kezikonyv oldalon. A részben azt jelenti, hogy nem minden függvény leírása van lefordítva.

27. – 28. PDF dokumentumok kezelése, létrehozása

Nagyon sokszor fordul elő az az eset, hogy a weblapunk kimenetét nyomtatóra is át kell vinni. Ez nagyon nehéz feladat lenne abban az esetben, ha minden elérhető nyomtatótípusra megpróbálnánk felkészíteni a programot. Válasszuk inkább azt a megoldást, hogy elkészítjük a nyomtatást egy PDF fájlba és mivel a felhasználók nagy többségénél van telepítve valamilyen PDF OLVASÓ, annak segítségével kinyomtatható a dokumentum. Arról nem is beszélve, hogy így megadjuk azt a lehetőséget is a felhasználóink számára, hogy a tartalmat el tudják menteni későbbi felhasználás céljából. Amásik felhasználási lehetőség a PDF dokumentumok megjelenítése. Nagyon sok vállalatnál a leírások, dokumentációk PDF formában állnak rendelkezésre. Ha ezeket (vagy egy részüket) meg akarják osztani a látogatóikkal, akkor a programnak képesnek kell lenni a PDF fájlok megnyitására is.
Kezdjük a második probléma megoldásával, mert ez az egyszerűbb eset. A PDF-ek megjelenítésének az egyetlen feltétele az, hogy a futtató számítógép rendelkezzen megjelenítő programmal. Ez tulajdonképpen bármi lehet, de leggyakrabban a felhasználók az Adobe Reader-t használják. Ha esetleg nem áll rendelkezésre, akkor letölthető a http://get.adobe.com/reader/ internetes oldalról. Akkor sincs semmi gond, ha a felhasználó gépén más program van telepítve, mert a megjelenítést az nem befolyásolja.
Célszerű a megjeleníteni kívánt állományokat egy külön mappába másolni.
A megjelenítés nem túl bonyolult, nézzünk egy példát:

```
<?php
$filename ="./pdf/Honda.pdf";
header("Content-type: application/pdf");
readfile($filename);
?>
```

A program a következőképpen működik: az első sorban beállítjuk annak a PDF fájlnak a nevét és elérési útvonalát melyet meg szeretnénk jeleníteni Mindenképpen szükség van a HTML oldal fejlécének beállítására. Ez „mondja” meg a böngészőnek,

hogy most nem kell az oldalt lefordítania, hanem az érkező tartalmat egy beépített vagy külső megjelenítő eszköz (program) segítségével kell megjeleníteni a felhasználó számára. Az utolsó sor feladat pedig a fájl beolvasása és kiírása a sztenderd output-ra, ami az esetek többségében a monitor.

Kicsit keményebb dió a PDF fájlok előállítása. A PHP-ban két függvénykönyvtár áll rendelkezésre a PDF-ek előállításához. Az egyik a ClibPDF, másik pedig a PDFLib. Mindkét csoportnak ugyanaz a feladata, mindösszesen a megvalósítás más kissé. Én most a PDFLib függvénykönyvtárat fogom használni a következőkben. A használathoz a következőket kell megtenni. Mivel a PHP-ban alapból nincs benne a PDFLib támogatás ezt kézzel kell megtegyük. Lépések sorban:

1. töltsük le a PDFLib-et a http://www.pdflib.com/download/pdflib-family/pdflib-8/ oldalról. Itt ki tudjuk választani, hogy milyen Windows verziót (32 bit vagy 64 bit) használunk, és milyen fejlesztői környezethez kell a könyvtár. A mi esetünkben PHP.
2. A letöltött tömörített állományból válasszuk ki azt a PHP verziót amely a mi gépünkre van telepítve és a benne lévő libpdf_php.dll állományt csomagoljuk ki.
3. A kicsomagolt dll állományt másoljuk be a PHP-nk ext könyvtárába. Ez Wamp szerver esetén a c:\wamp\bin\php\php5.3.0\ext mappa lesz. Ha nem tudjuk hogy hol van az ext mappa akkor segítségünkre lehet a phpinfo() függvény. Írnunk kell egy egysoros programot ***<?php phpinfo(); ?>***, és azt megfuttatva meg tudjuk nézni az ext mappa helyét.

18. ábra

4. Ezek után újra kell indítanunk a web szerverünket. Sok esetben nem elég a restart funkció, teljesen le kell állítani, majd újraindítani a szervert.

29. – 30. Komplex vizsgafeladat elkészítése

Készítsen egy WEB oldalhoz kapcsolódó általános bejelentkezési lapot!

Az adatbázisban a következő adatokat tárolja:

- Teljes név
- Felhasználói név
- Születési dátum
- E-mail cím
- Jelszó

Az adatok bevitelénél a következő feltételeket tartsa be:

Név:	maximum 100 karakter hosszúság
Felhasználói név:	maximum 15 karakter hosszúság, ellenőrizze, hogy létezik e már a megadott név
Jelszó:	minimum 6 karakter hosszúság
E-mai cím:	maximum 100 karakter hosszúság, Ellenőrizze a cím helyességét
Születési dátum:	Ellenőrizze a bevitt adat helyességét

A következő lapokat kell elkészítenie:

1. Regisztrációs úrlap
2. Bejelentkezési lap
3. Adminisztrációs felület

Az adminisztrációs felületen lehessen a regisztrált felhasználókat kilistázni, és a listákat PDF formátumba nyomtatni.