

Kozmajer Viktor

PHP és MySQL az alapoktól

Kozmajer Viktor

PHP és MySQL az alapoktól

BBS-INFO Kiadó, 2011.

Minden jog fenntartva! A könyv vagy annak oldalainak másolása, sokszorosítása csak a kiadó írásbeli hozzájárulásával történhet.

Köszönet Hegedűs Gábornak, aki nélkül ez a könyv nem jöhetett volna létre

A könyv nagyobb mennyiségben megrendelhető a kiadónál: BBS-INFO Kiadó, 1630 Bp. Pf. 21. Tel.: 407-17-07

A könyv megírásakor a szerző és a kiadó a lehető legnagyobb gondossággal járt el. Ennek ellenére, mint minden könyvben, ebben is előfordulhatnak hibák. Az ezen hibákból eredő esetleges károkért sem a szerző, sem a kiadó semmiféle felelősséggel nem tartozik, de a kiadó szívesen fogadja, ha ezen hibákra felhívják figyelmét.

ISBN 978-963-9425-74-3

Kiadja a BBS-INFO Kft.

1630 Budapest, Pf. 21.

Felelős kiadó: a BBS-INFO Kft. ügyvezetője

Nyomdai munkák: Biró Family nyomda

Felelős vezető: Biró Krisztián

TARTALOMJEGYZÉK

1. Bevezető	9
2. APACHE, MySQL, PHP	11
2.1. Alapfogalmak	11
2.2. Szerveroldali szkriptek és működésük	13
2.3. Az APACHE2TRIAD telepítése	15
3. A PHP program szerkezete	19
3.1. Első PHP programunk	20
3.2. PHP kód blokkjainak kijelölésére szolgáló elemek	22
4. Függvények	23
4.1. A print függvény	23
4.2. A date függvény	26
4.3. Matematikai függvények	27
5. Változók	30
5.1. Változó típusának lekérdezése – a gettype() függvény	34
5.2. Változó típusának módosítása – a settype() függvény	37
6. Operátorok és kifejezések PHP-ben	39
7. Vezérlési szerkezetek	46
7.1. Elágazás – Az IF utasítás	46

7.1.1. Példaprogram az IF utasítás bemutatására.....	50
7.2.SWITCH szerkezet.....	54
7.2.1. Példaprogram az SWITCH szerkezet bemutatására:	55
8. CIKLUSOK.....	58
8.1.A WHILE() ciklus.....	59
8.2.A DO...WHILE() ciklus	61
8.3.A FOR() ciklus	63
8.3.1. Példaprogram a FOR () ciklus bemutatására	64
8.4.Egymásba ágyazott ciklusok	65
8.4.1. Példaprogram az egymásba ágyazott ciklusokhoz	65
9. Űrlapok	68
9.1.Űrlap elemek:	70
9.2.Űrlap: az action TAG és a paraméter átadás.....	73
10. Űrlapok használata a gyakorlatban	76
10.1. Regisztrációs űrlap készítése.....	76
10.2. Téglalap kerülete és területe űrlap segítségével.....	88
11. SQL adatbázisok	96
11.1. Bevezető, alapfogalmak	96
11.2. MySQL adatbázisok	97
11.3. Egy tábla felépítése MySQL belül.....	98
12. PHP MyAdmin	101
12.1. Legfontosabb adattípusok a MySQL-en belül	102
12.2. A MySQL működése	106
12.3. Új adatbázis létrehozása phpMyAdmin-ban.....	106
12.4. Adatbázis, adattábla készítése	109

13. SQL parancsok.....	115
13.1. Belépés a DOS-os (konzolos) felületre:	115
13.2. Lekérdezés SQL parancsokkal:	117
13.3. Kapcsolat létrehozása MySQL-ben.....	120
14. DML – adatmódosító parancsok.....	122
15. PHP és MySQL összekapcsolása	124
15.1. Példaprogram PHP és MySQL összekapcsolására	126
16. CMS rendszerek	132
16.1. A CMS-ek előnyei	133
16.2. Magyar nyelven is elérhető CMS rendszerek.....	135
16.3. CMS-ek telepítése általánosságban	136
17. Megvalósítások PHP-vel.....	139
17.1. Képgaléria PHP-ben 1.0	139
17.2. Képgaléria PHP-ben 2.0	143
17.3. Adatbázisok kimentése – DUMP-olás.....	146
17.4. Linkajánló készítése.....	151
17.4.1. Linkek adattábla létrehozása	152
17.4.2. Linkek.php elkészítése.....	154
17.5. Admin felület elkészítése.....	157
17.5.1. Admin.php elkészítése	158
17.5.2. Torol.php elkészítése.....	162
17.5.3. Linkfelvitel.php elkészítése.....	164
17.5.4. Felvisz.php elkészítése.....	168

1. Bevezető

Ez a zsebkönyv ajánlott mindazok számára, akik már rendelkeznek minimális ismeretekkel a HTML nyelvvel kapcsolatban, megtanulták a weboldalszerkesztés alapjait, de szeretnének tovább lépni, a dinamikus weblapok irányába. Szintén hasznos lehet azon olvasóknak is, akik már megismerkedtek más programnyelvekkel és most betekintést szeretnének nyerni, hogyan is működik mindez webes környezetben. Számukra (is) a legideálisabb választás a világ egyik legkedveltebb webprogramozói nyelve, a PHP és a szintén elterjedt MySQL webes adatbázis-kezelő rendszer, melynek számos előnye van. Segítségükkel professzionális honlapokat programozhatunk le, összeköthetjük őket az általunk készített, webes adatbázisokkal, amelyeket a világon bárhol elérhetünk az internet segítségével. Weboldalainkat dinamikussá téve, akár másodpercenként frissülő információkkal láthatjuk el, az adatok felvételét, módosítását és törlését elvégezhetjük pár

kattintással, a saját készítésű admin felületünkön és adatbázisunkban. És bármennyire is hihetetlen, mindkét fenti komponens teljesen ingyenes!

A könyv egymásra épülő fejezetein végighaladva, az olvasó megismerkedhet a PHP nyelv alapjaival, képes lesz önállóan is kisebb PHP programok megírására, a MySQL adatbázis-kezelés alapjait is megtanulhatja, így webes adatbázist is készíthet, amelyet az ismeretei alapján képes lesz PHP weboldallal összekötni. Megtanulja az űrlapkészítést, így akár saját adminisztrációs felülettel rendelkező, dinamikusan frissíthető honlapot is képes lesz elkészíteni. Megismerkedhet a CMS (tartalomkezelő) rendszerekkel, így minimális programozói tudással is képes lesz önállóan honlapokat telepíteni, majd azokat külsőleg testre szabni és tartalommal feltölteni.

A PHP kódrészletek és az illusztráló képernyőképek nagyban segítenek, hogy a leírtakat saját maguk is meg tudják valósítani (Macromedia/Adobe Dreamweaver, illetve PHP MyAdmin) segítségével.

2. APACHE, MySQL, PHP

2.1. Alapfogalmak

Mielőtt bármibe is belekezdenénk, ismerjük meg a legfontosabb alapfogalmakat.

APACHE: webservert alkalmazás, ami lehetővé teszi PHP programunk saját gépünkön történő futtatását.

MySQL: adatbázis-kezelő környezet.

PHP: programozási nyelv webes felületen (a Personal Home Page rövidítése). A PHP szerveroldali szkript nyelv, amivel egyszerűen tudunk aktív weboldalt készíteni. Létezik kliensoldali (pl.: Java) és szerveroldali kiszolgálás (pl.: PHP).

Az Apache-ba épül be a PHP nyelv, ezt alapjáraton nem ismeri fel a böngésző. A HTML oldal kódjában a PHP kódot: `<?php` és `?>` tag-ek közé kell írni. Ezek a hagyományos, PHP kódblokk kijelölésére szolgáló elemek.

A könnyebb átláthatóság és a programhibák könnyebb észlelése érdekében célszerű weblapszerkesztő programot használnunk (pl. Adobe Dreamweaver), mivel ezek tartalmazznak kódnézet funkciót is, amiben más-más színnel jelöli a program a különböző PHP-s kódrészleteket, függvényeket. Ha valamit elírunk a program begépelése során, akkor a kódszínezete is megváltozik majd, észrevehetőbbek a szintaktikai hibák, így könnyen észlelhető, hogy elrontottunk valamit, és azt is, hogy hol. Ez jelentősen megkönnyíti a munkánkat!

A PHP kezdetben csak egy makrógyűjteménynek indult, ami a személyes honlapok programozását hivatott támogatni. Neve is innen ered: **Personal Home Page** (egy másik forrás szerint Personal Homepage Programming).

A PHP azonban túlnőtt kezdeti feladatán, és mára már egy komplett web-programozói nyelvvé alakult. Legtöbbször ezt alkalmazzák, mert nyílt forráskódú és ezáltal ingyenes, az Internetről CMS (Content Management System) rendszereket tudunk letölteni, amik előre elkészített, komplett PHP oldalak és hozzájuk tartozó „design”-ok (témák, kinézetsémák a honlaphoz), kiegészítések, plugin-ok.

2.2. Szerveroldali szkriptek és működésük

A HTML nyelv eszközeivel együtt használva interaktívvá tehetjük weboldalainkat. A PHP tulajdonképpen egy szerveroldali szkript nyelv.

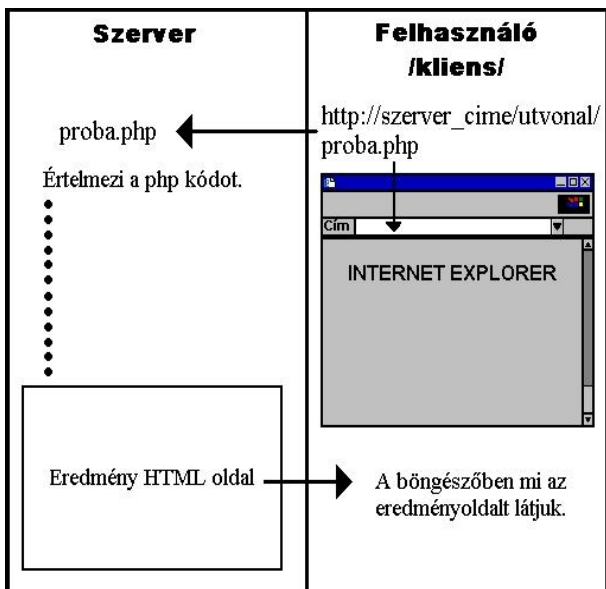
A szerveroldali szkriptek a következőképp működnek:

1. A böngészőbe beírjuk a PHP program nevét (elérési útvonalával együtt).
2. A szerver, ahol a PHP oldal található, megkeresi a PHP fájlt, és egy értelmező program segítségével lefuttatja azt.
3. Futtatás közben egy ún. „eredmény HTML oldalt” készít, melyet – miután a PHP kód végére ért – visszaküld a felhasználói oldalra a böngészőnek.
4. A felhasználó már csak ezt a PHP kód alapján készült HTML oldalt látja.

A Windows operációs rendszer alá több olyan programcsomag is létezik, amely minden szükséges összetevőt feltelepít számunkra a programozás elkészítéséhez.

Ilyenek pl.:

- PHP HOME
 - APP SERVER
 - APACHE2TRIAD
- (Mindhárom ingyenes.)



SZERVER pl.: Apache webservert; adatbázis-kezelő rendszer (MySQL) és beépülő PHP modul is megtalálható benne

2.3. Az APACHE2TRIAD telepítése

Az Apache webszerver segítségével PHP fájlokat futtathatunk az otthoni számítógépünkön. APACHE2TRIAD-nál a **HTDOCS mappa** a weben elérhető oldalakat tartalmazza. Ebbe a mappába kell helyoznunk azokat a fájlokat, amik PHP kódblokkot tartalmaznak.

A fájlokat úgy tekinthetjük meg, hogy begépeljük a böngészőnk címsorába a **http://localhost** címet (a http:// nem szükséges). Ezt az oldalt érdemes beállítani kezdőoldalként a böngészőnkben, ha sokat programozunk PHP-ben! A cím begépelése után megjelenik a webszerver „kezdőoldala” és innen tudunk belépni azokba a mappákba, ahova PHP oldalainkat elhelyeztük. Innen tudjuk elérni a webszerver MySQL adatbázis-kezelő modulját, a grafikus kezelőfelülettel rendelkező **phpMyAdmin**t.

Az Apache telepítése semmivel sem nehezebb, mint bármely más programé. A könyvben az 1.5.4-es verzió telepítését mutatjuk be ebben a részben.

A telepítés után már rendelkezni fog a gépünk beépített PHP fordítóval, illetve MySQL adatbázis-kezelő modullal (mindkettő 5-ös verzió lesz a 1.5.4-es Apache-nál), így ezeket nem kell külön telepítenünk, rögtön neki is állhatunk a PHP programozásnak!

Nézzük a telepítés lépéseit:

1., Töltsük le az Apache telepítőjét a <http://apache2triad.net>-ről és indítsuk el!

A megjelenő képernyőn beállíthatjuk, milyen komponenseket telepítsen a program (hagyjuk, hogy mindent telepítsen!). Ezek után kattintsunk a Next feliratú gombra a következő lépéshez.



2., A következő lépésben megadjuk, melyik meghajtóra szeretnénk telepíteni a webszervert. Ha ezt megtettük, kattintsunk a Next gombra!

3., A harmadik lépésben meg kell adnunk egy jelszót, amit az Apache webszerver használatakor fogunk alkalmazni (pl.: adatbázishoz kapcsolódunk, ami a helyi gépen van – ezekről bővebben később). A jelszó minimum 8, maximum 32 karakter hosszú lehet! Egyébként a hozzá tartozó

felhasználónevünk a 'root' lesz, legalábbis a localhost esetében.



4., Ezen lépések után már csak el kell fogadjuk a licenc-szerződést és indulhat is a telepítés! Miután a telepítő feltette gépünkre a megadott mappába az Apache-ot, a képernyőn még egyszer kérni fogja a jelszavunkat egy DOS-os ablakban. Gépeljük be a 3. pontban megadott jelszavunkat és nyomjunk Enter-t! Ezek után már csak újra kell indítanunk a számítógépünket és már használhatjuk is az APACHE2TRIAD-ot!

Fontos! Ha nem találjuk az apache2triad nevű mappát a megadott helyen, ne lepődjünk meg, a mappa alapbeállításként rejtett attribútumot kap.

Ha fájlkezelő programot használunk (pl.: Total Commander) állítsuk be, hogy mutassa a rejtett fájlokat is (Commander-nél: Beállítások → Általános Beállítások → Képernyő → Látszik a rejtett/system fájl)!

3. A PHP program szerkezete

Egy PHP program nem más, mint szokványos HTML kód, melybe a PHP kód a `<?php` és a `?>` jelek közé kerül. Többsoros kód esetén a sorok végére pontosvesszőt kell tennünk, ezzel zárjuk PHP-ben a parancsokat.

Figyelem! A tapasztalat szerint ezeknek a pontosvesszőknek a lefelejtése az egyik leggyakoribb kódolási hiba. Szintaktikailag hibás kód esetén a PHP értelmező hibaüzenetet jelenít meg a képernyőn, értelemszavaró hiba esetén a további végrehajtást befejezi. Nagy segítség, hogy a hibaüzenetben az értelmező kiírja a hiba helyét is.

A PHP értelmező program az alábbi szabályokat veszi figyelembe a PHP kód értelmezése közben:

- Ha HTML kódot talál, azt változtatás nélkül átmásolja az eredmény oldalra.

- Ha PHP kódot talál, azt lefuttatja. Ha azt a parancsot kapja például, hogy „Írd ki!”, akkor a kiírandó szöveget az eredményoldalra írja ki, mégpedig az aktuális pozícióba.
- Ha bármilyen más parancsot kap, azt végrehajtja és nem nyúl az eredményoldalhoz.

3.1. Első PHP programunk

Amikor PHP oldalt írunk, tudatnunk kell a feldolgozóval, mely részeket hajtsa végre. Ha nem adjuk meg, hogy a fájl mely részei tartalmazzanak PHP blokkokat, akkor mindent HTML-nek tekint és változtatás nélkül továbbküldi a böngésző számára.

A PHP blokk kijelölésének elhalasztása és ennek következménye egy példával jól szemléltethető:

1. Ez esetben nem jelöljük ki a PHP blokkot. Ekkor a böngésző, az előbb leírtak szerint HTML-nek néz mindent. A mi esetünkben a print függvényt (bővebben róla a következő fejezetben) egyszerű szöveggént értelmezi és a böngészőnk a print ("Helló világ!"); szöveget jeleníti meg. Vastag betűvel jelöltük a két forráskód azon részeit, amelyekben azok különböznek egymástól:

```
<html>
<head>
<title>Első PHP programunk</title>
</head>
  <body>
    print ("Helló világ!");
  </body>
</html>
```

2. Ez esetben már kijelöljük a PHP blokkot a következő táblázatban látható hagyományos `<?php` jelöléssel. Ekkor a PHP feldolgozó észleli, hogy mi a `print` függvényt használjuk a blokkban és e szerint végre is hajtja a parancsot. Böngészőnkben már csak a „Helló világ!” szöveg jelenik meg:

```
<html>
<head>
<title>Első PHP programunk</title>
</head>
  <body>
    <?php
      print ("Helló világ!");
    ?>
  </body>
</html>
```

3.2. PHP kód blokkjainak kijelölésére szolgáló elemek

Elnevezés	Kezdő elem	Záró elem
Hagyományos	<?php	?>
Rövid	<?	?>
ASP stílusú	<%	%>

Megjegyzés: A PHP blokkon belül fűzhetünk megjegyzéseket a programunkhoz! Ezt esetünkben fontos megemlíteni, mert a későbbi fejezetek példaként bemutatott forráskódokban szintén fogok használni ilyeneket. Az egyszerű HTML oldalak szerkesztésénél ilyen esetben a **<!-- Ide írjuk a megjegyzést -->** tag-et használjuk.

A PHP-nál a következő megjegyzési formák állnak a rendelkezésünkre:

```
// egysoros megjegyzés
# ez is egy egysoros megjegyzés
/* ezt a fajta megjegyzést már nem egy,
hanem több soros megjegyzésnél használjuk */
```

Ezek közül bármelyikkel is írunk megjegyzést, az nem fog látszani az eredmény oldalon, csupán a PHP oldalunk kódnézetében.

4. Függvények

A függvények olyan parancsok, amelyek valamilyen műveletet végeznek, többnyire attól függően, hogy milyen adatokat kapnak. A függvénynek átadott adatokat mindig zárójelbe kell tennünk a függvény neve után. Esetenként ezt elhagyhatjuk. A PHP-ban **több száz beépített függvényt** tudunk meghívni, ezek teljes listáját (valamint leírását és használatát) a **<http://www.php.net>** oldalon találhatjuk meg. Ezek sokszor kiegészülnek, újabb függvényeket írnak a már meglévők mellé. A következőkben megnézzük a beépített függvények közül néhányat (amelyet a későbbi példaprogramokban, forráskódokban is bemutatunk) a jobb megértés érdekében.

4.1. A print függvény

Az elsősorban szövegmegjelenítésre szolgáló print függvény esetében a zárójeleket elhagyhatjuk:

```
print ("szöveg, amit ki szeretnénk írni");
```

```
print "szöveg, amit ki szeretnénk írni";
```

A `print` függvénynek egy karakterláncot adtunk át. A karakterláncokat mindig (egyes vagy kettős) idézőjelbe kell tenni.

A `print(` helyett használhatjuk a `<?= ?>` szkriptet is szövegmegadásra. Ezen kívül még az `echo(` függvényt szokták használni ugyanerre a célra. Az utasításokat pontosvesszővel fejezzük be, ezzel tudatjuk a fordítóval, hogy befejeztük az utasítást (nem mindig szükséges kitenni a pontosvesszőt, például az utolsó sor végén).

A PHP-val nemcsak látható szöveget írhatunk ki az eredményoldalra, de HTML kódot is:

```
a.,  
<html>  
<head>  
<title>A print függvény</title>  
</head>  
<body>  
  <?="valami" ?>  
</body>  
</html>
```



```
b.,  
<html>  
<head>  
<title>A print függvény</title>  
</head>  
  <?php  
    print '<body>';  
    print ('valami');  
    print '</body>';  
  ?>  
</html>
```

Látható, hogy HTML kód beépítése a PHP oldalakba egyszerűen HTML tartalom begépelésből áll. A PHP feldolgozó figyelmen kívül hagy mindent a PHP nyitó és záró elemeken kívül. Egy dokumentumban a HTML elemek közé tetszőleges számú PHP kódblokk írható. A fenti példa mindkét esetben a *valami* szót írja ki a böngésző ablakába, csupán annyi a különbség, hogy a 2. esetben a `<body>` és `</body>` tag-eket a `print` függvénnyel írtuk ki.

Bár több kódblokkot helyezhetünk el egy dokumentumon belül, ezek együttesen alkotnak egy programot. Bármilyen, amit egy megelőző blokkban határoztunk meg (változók, függvények vagy osztályok), a dokumentumon belül elérhető lesz a későbbi blokkban is.

A több együttműködő, összességében egy nagyobb programot megvalósító PHP fájlt nevezzük **PHP alkalmazásnak**.

4.2. A date függvény

Ez a függvény a szervergépi idejét kéri le.

Szintaktikája:

date("Y-m-d, H:i:s");

Első paraméter: Y → év, m → hónap, d → nap

Második paraméter: H → óra, i → perc, s → másodperc (aktuális idő, frissítésre változik).

Fontos tudni, hogy különböző időformátumot kapunk, ha a különböző paraméterek kis- ill. nagybetűvel írjuk!

Pl.: A fenti szintaktika a következő dátum- és idő formátummal jelenik meg: 2011-02-18 18:30:25.

Ha például az Y mellett a hónapokat és a napokat jelölő paramétereket is nagybetűvel írjuk, akkor a következőt kapjuk: 2010-Feb-Fri 18:30:25.

A date függvénynek ezeken kívül vannak más paraméterei is. Például:

date("t"); → Ezzel a paraméterrel a date függvény azt mutatja meg, hogy hány nappal áll az aktuális hónap.

`date("w");` → Az aktuális nap „számbeli” megfelelője, számreprezentációja: Érthetőbben: a 0 a vasárnap, 1 a hétfő... egészen a 6-osig, ami a szombat. Ezt a paramétert még fogjuk használni a SWITCH szerkezet bemutatására!

➤ Ezen kívül még rengeteg paramétert sorolhatnánk fel, de ez számunkra most nem fontos, egyelőre ennyi nekünk elég.

Még egy fontos tudnivaló, hogy a paraméterek közé elhelyezett írásjel fog megjelenni a dátum-formátumban is. Például, ha „-” jelek helyett „.”-ot teszünk akkor a 2011.02.18.-as dátumot kapjuk.

Érdekességképen a `date` függvényhez megemlítenéd, hogy a

time függvény: időbélyeg, „internetidő”, az 1970. év a 0. év.

Szintaktikája: `time()`

4.3. Matematikai függvények

Most pedig nézzünk néhány beépített matematikai függvényt. Ezek mindegyikét nem biztos, hogy használjuk majd a későbbi fejezetekben, viszont hasznosak lehetnek a továbbiakban egyes felhasználók számára.

a. pi ()

A pi értékét adja vissza 13 számjegy pontossággal.

A függvény eredménye: 3,1415...

b. sqrt (\$szam)

A \$szam változóban (bővebben: következő fejezet) eltárolt számnak adja vissza a négyzetgyökét.

Pl.: sqrt (64)

A függvény eredménye: 8

c. abs (szám)

Egy szám (ez lehet egész vagy lebegőpontos) 0-tól való távolságát adja vissza (vagyis az abszolút értékét).

Pl.: abs (-34.56)

A függvény eredménye: 34.56

Mint azt már a „Függvények” fejezet bevezetőjében is írtuk, a PHP nyelvben megtalálható függvények teljes listája, leírása, bemutatása és szintaktikája megtalálható a <http://www.php.net> weboldalon.

Fontos még megemlíteni, hogy ugyanitt megtalálhatjuk a változók, konstansok, operátorok, elágazások, ciklusok stb. listáját PHP-ben, egy teljes körű ismertetővel a nyelv referencia oldalán:

<http://www.php.net/manual/en/langref.php>

A fenti weboldalon minden szükséges információt megtalálhatunk a PHP teljes körű megismeréséhez, így annak, aki a jövőben komolyabban el akar mélyülni eme programozási nyelv rejtelmeiben, ajánlatos átnézni és tanulmányozni ezen részletes referenciákat (csak érdekesség szintjén hoztuk most fel a PHP hivatalos honlapját, a következő fejezeteknél nem fogunk mindig külön kitérni erre a weboldalra, de már az elején fontosnak tartottuk ezt megemlíteni.)

5. Változók

A változó névvel ellátott memóriaterület a webszerver memóriájában. Ezekben adatokat tárolunk, mellyel a PHP programunkban műveleteket végezhetünk. Jelölése: \$változó neve. A változónak van **neve**, **típusa** és **értéke**. A változó nevének megadásakor figyeljünk arra, hogy a PHP megkülönbözteti a kis és nagybetűket!

Pl.: a \$szam nem ugyanaz, mint a \$Szam!

A név megadásakor figyeljünk arra is, hogy ne használjunk ékezetes betűket. Ezenkívül nem tartalmazhat még szóközt, valamint speciális karaktereket sem. Használhatjuk viszont az angol abc betűit, az aláhúzás _ karaktert és a számokat is.

Figyelem! A változók nevének megadásakor figyeljünk arra is, hogy a név „beszédese” és lehetőleg rövid legyen. Ez azért fontos, mert ha a program írása során vagy a későbbiekben használni akarjuk a változót, akkor így a nevére

könnyebben felismerhetjük, hogy a változó milyen adatot rejt:

Például: \$nev="Gabi"; → string-es változó

Például: \$szam=32; → egész (integer) változó

Most pedig nézzük milyen típusú változókat használhatunk PHP-ben:

1. Változó neve (angolul): **integer**

Magyarul: **egész szám**

Leírása: értéke csak pozitív, vagy negatív egész szám lehet.

Példa: \$szam = 21;

2. Változó neve (angolul): **float (vagy double)**

Magyarul: **lebegőpontos szám**

Leírása: tizedes számokat írhatunk le vele.

Tizedesjelként mindig pontot kell használni!

Példa: \$szam = 21.65;

3. Változó neve (angolul): **boolean**

Magyarul: **logikai érték**

Leírása: értéke lehet igaz (TRUE), vagy hamis (FALSE)

Példa: \$igaze = TRUE;

4. Változó neve (angolul): **array**

Magyarul: **tömb**

Leírása: egy egymáshoz rendelt adatcsomag közös neve.

Példa: \$a = array(21, 32, "szöveg");

5. Változó neve (angolul): **string**

Magyarul: **karakterlánc, szöveg**

Leírása: szövegek kiírására használjuk, tartalmazhat bármilyen karaktert - betűt, számot, írásjelet. A szöveget mindig egyszeres, vagy dupla idézőjelek közé kell tenni!

Példa: \$pelda = "Ez egy szöveg, amit string-es változóban tároltunk el!";

Bár a PHP nyelvtana nem írja elő, a változókkal végzett műveletek előtt azokat ajánlott definiálni, azaz megadni a kiinduló értékét és típusát.

Ha már megismerkedtünk a legfontosabb változó típusokkal, lássuk, hogyan is adhatunk nekik értéket. Természetesen ezt többféleképpen is megtehetjük:

- Függvény meghívásával. Pl.: \$negyzetgyok = sqrt(64);
- Összeadási művelettel. Pl.: \$osszeg = 2 + 6;
- Felhasználótól adat bekérésével (ezt az Űrlapok c. fejezetben részletezzük).
Pl.: \$aoldal = \$_GET['aoldal'];
- Adatbázisból való lekérdezéssel.
Pl.: \$tulajneve = \$sor[nev];

- Egy külső eszközről való beolvasással. Pl.: adatbázis és PHP program összekapcsolásakor ilyen az ún. „include”-olás (bővebben később foglalkozunk vele)
- Másik változóból átvett értékkel. Pl: \$szam1 = 32; \$szam2 = \$szam1;

Az adatok típusát azok definiálásakor a programozó határozza meg. A PHP az adattípusokat rugalmasan használja, azaz futásidőben átalakítja a környezettől függően szükséges típusra.

Adatunk érvényes típusát a **gettype()** beépített függvény segítségével kérdezhetjük le és a **settype()** függvénnyel módosíthatjuk. Adatainkat az alábbi módon is meghatározhatjuk/módosíthatjuk:

- (int), vagy (integer) - egészre konvertál
- (bool), (boolean) - logikai értékké konvertál
- (real), (double), (float) - lebegőpontos számmá konvertál
- (array) - tömbbé konvertál
- (object) - objektummá konvertál

Most nézzünk erre a két függvényre is példát (a forráskódokban ezek után **ki fogjuk hagyni** a html, body stb. tag-eket és csak a számunkra fontos PHP kódblokkot írjuk le)!

5.1. Változó típusának lekérdezése – a `gettype()` függvény

A `gettype()` szintaktikája egyszerű:

`gettype($valtozoneve)`

Zárójelekben megadjuk annak a változónak a nevét, aminek a típusát meg szeretnénk vizsgálni a függvénnyel.

Most pedig nézzünk rá egy példaprogramot:

```
<?php
```

```
$szam=32;
```

```
print ("A 'szam' változó értéke: $szam <br />"); //beszúrtunk egy sortörés TAG-et a változó kiírása után
```

```
print ("típusa: ".gettype ($szam)."<br /><br />"); //így a változó típusának kiírása már egy új sorba került.
```

```
/*
```

Egy sorban különálló szövegek összefűzésére PHP-ben a pont (.) szolgál. Jelen esetben a második `print` függvénynél megfigyelhető, hogy a szöveghez hozzáfűztük a `gettype()` függvény eredményét is, majd ehhez jött még a két sortörés TAG.

```
*/
```

```
$tizedesszam=32.5;
```

```
print ("A 'tizedesszam' változó értéke:
```

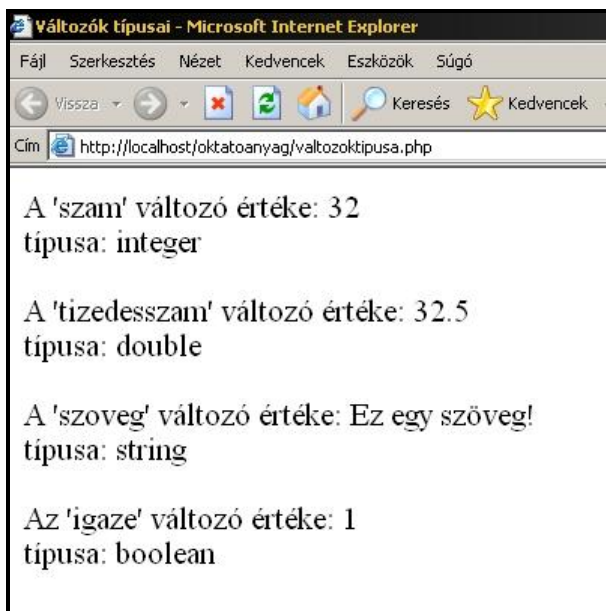
```
$tizedesszam <br />");
```

```
print ("típusa: ".gettype ($tizedesszam)."<br /><br />");
```

```
$szoveg="Ez egy szöveg!";  
print ("A 'szoveg' változó értéke:  
".$szoveg."<br />");  
print ("típusa: ".gettype ($szoveg)."<br  
/><br />");  
//a string-es változóknál mindenféleképpen oda  
kell figyelni a szövegek összefűzésére, különben  
hibaüzenetet kaphatunk!  
$igaze= true;  
print ("Az 'igaze' változó értéke: ".$igaze."<br  
/>");  
print ("típusa: ".gettype ($igaze)."<br /><br />");  
?>
```

Figyelem! Ahogy a kódrészletben is láthatjuk megjegyzésként, az egy sorban különálló szövegek összefűzésére PHP-ben a pont (.) szolgál. Erre különösen oda kell figyelnünk, mert könnyen szintaktikai hibát okozhatunk, ha nem, illetve ha rosszul használjuk!

Ha az előbbi példaprogramot megnézzük egy böngészővel, akkor a következőt láthatjuk majd:



Jól látható, hogy a különböző változók értékeit megvizsgálva a `gettype()` függvény visszaadta azok típusát!

5.2. Változó típusának módosítása – a `settype()` függvény

A `settype()` szintaktikája a következő:

`settype($valtozoneve, 'típus, amire konvertálni akarunk');`

<?php

```
$szam=32.5;  
print ("A 'szam' változó értéke: $szam <br />");  
print ("típusa: ".gettype ($szam)."<br /><br  
>");
```

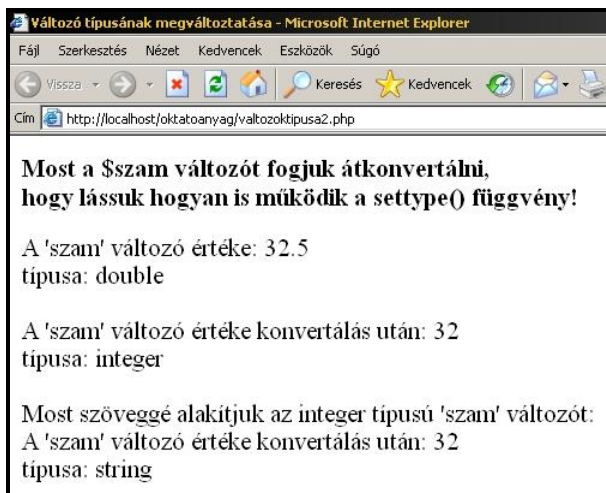
```
settype($szam, 'integer');  
print ("A 'szam' változó értéke konvertálás után:  
$szam <br />");  
print ("típusa: ".gettype ($szam)."<br /><br  
>");
```

```
print ("Most szöveggé alakítjuk az integer típusú  
'szam' változót:<br />");  
settype($szam, 'string');  
print ("A 'szam' változó értéke konvertálás után:  
$szam <br />");  
print ("típusa: ".gettype ($szam)."<br /><br  
>");  
?>
```

A kódrészlet jól szemlélteti, hogyan is használhatjuk a `settype()` függvényt korábban

deklarált változóink típusának megváltoztatására. Ahogy a megjegyzésben is látszik, a `settype()` szintaktikája a következőképpen épül fel: a függvény első paraméterében megadjuk a változó nevét, majd a második paraméterben, hogy milyen típusúvá szeretnénk konvertálni azt.

A böngészőnkben ez fog megjelenni, ha futtatjuk a példaprogramot:



6. Operátorok és kifejezések PHP-ben

Miután megismerkedtünk a változók deklarálásával és típusaival, nézzük meg, milyen műveleteket tudunk velük végezni. Ahhoz, hogy műveleteket (legyen az aritmetikai vagy más) végezhesünk PHP-ban, más programozási nyelvekhez hasonlóan itt is szükségünk lesz az operátorok használatára. Mielőtt azonban belemélyülnénk, nézzük meg mi is az az operátor:

Az operátor jel, az operandusok között helyezkedik el: pl.: $12 + 32$. Ebben az esetben az összeadás „+” jel az operátor, a 12 és a 32 pedig az operandusok. Ez a 3 alkotóelem így egy kifejezést alkot. Általában az operátor jelet mindkét oldalról egy operandus zárja be, persze vannak kivételek: ++i, --i (lásd a Vezérlési szerkezeteknél!).

Az előző fejezetekben már használtunk is egy operátort a változók deklarálásánál, ez pedig az

értékadó „=” jel operátor volt. Ezen kívül persze az operátoroknak is rengeteg csoportja van:

- **Aritmetikai operátorok:** ezek a hagyományos operátorok, amelyeket a matematikában is használunk: összeadás (+), kivonás (-), szorzás (*), osztás (/), negáció (-\$szam) illetve a maradékosztás (%). Ez utóbbi kettő lehet, hogy nem mindenki számára ismerős, ezért nézzünk rájuk példát: a negáció a kapott értéket negálja, vagyis, ha \$szam = 12, akkor a változó a negálás után -\$szam = -12. A maradékosztás sem annyira ördögös: a maradékosztás eredménye a bal oldalon lévő szám, jobb oldalon lévő számmal történő elosztásának maradéka.

Például: $13\%4 = 1$ (mivel $3*4 = 12$ -vel, maradék az egy)

$20\%3 = 2$ (mivel $6*3 = 18$, maradék a kettő)

$11\%4 = 3$ (mivel $2*4 = 8$, maradék a három) stb.

- **Összefűző operátor:** ennek használatát már megemlítettük az egyik forráskódban. Ezt különálló szövegek összefűzésére használjuk a PHP-ben. Ennek operátora a pont (.). **Figyelem**, a művelet szóközt nem rak az összefűzött tagok közé, azt nekünk kell a szövegbe begépelnünk:


```
<?php
$szoveg1 = "Helló ";           //a Helló
szó után raktunk egy szóközt is!!!
$szoveg2 = "világ!";
print ($szoveg1.$szoveg2);
?>
```

Ha ezt a kettőt összefűzzük és kiíratjuk a képernyőre, akkor azt kapjuk, hogy Helló világ!

- **Összetett értékadó operátorok:** Ezek lényeg, hogy két operátort kapcsolhatunk velük össze. Mindegyik ilyen operátornak két alakja van. A rövidebbeket elég gyakran használják PHP-ben, mivel egyszerűbben írhatunk le velük egy-egy kifejezést, és ezzel időt spórolhatunk a gépelés-kor és persze kevesebb karaktert kell felhasználnunk hozzájuk. Persze használhatjuk a hosszabb alakot is, az eredmény mindkét esetben ugyanaz lesz. Mindez a kezdők számára kicsit bonyolult lehet, de nem kell aggódni, könnyen megszokja az ember ezeket a rövidített formákat is:

Hagyományos forma	Rövidített forma
<code>\$a = \$a + 3</code>	<code>\$a += 3</code>
<code>\$a = \$a - 3</code>	<code>\$a -= 3</code>
<code>\$a = \$a * 3</code>	<code>\$a *= 3</code>
<code>\$a = \$a / 3</code>	<code>\$a /= 3</code>
<code>\$a = \$a % 3</code>	<code>\$a %= 3</code>
<code>\$a = \$a . „ egy szöveg.”</code>	<code>\$a .= „ egy szöveg.”</code>

Ide tartoznak még az utó- és előnövekményes operátorok, ezekről bővebben a Vezérlési szerkezetek című résznél.

- **Összehasonlító operátorok:** Ezek feladata, hogy a bal és jobb oldalon lévő operandust összehasonlítsák. Az összehasonlításnak két eredménye lehet: igaz (true, 1) vagy hamis (false, 0):

Operátor	Megnevezés	Igaz, ha:	Példa
<code>==</code>	egyenlő	A bal oldal egyenlő a jobb oldallal.	<code>\$a == 12</code>
<code>===</code>	azonos	A két oldal nemcsak érték szerint, hanem típusát tekintve is megegyezik.	<code>\$a === 12</code>
<code>!=</code>	nem egyenlő	A bal és a jobb oldal nem egyenlő.	<code>\$a != 12</code>
<code>!==</code>	nem azonos	A bal és a jobb oldal nem azonos.	<code>\$a !== 12</code>
<code>></code>	nagyobb, mint	A bal oldal nagyobb a jobbnál.	<code>\$a > 12</code>
<code><</code>	kisebb, mint	A bal oldal kisebb, mint a jobb.	<code>\$a < 12</code>
<code>>=</code>	nagyobb egyenlő	A bal oldal nagyobb vagy egyenlő a jobb oldallal.	<code>\$a >= 12</code>

< =	kisebb egyenlő	A bal oldal ki- sebb vagy egyenlő a jobb oldallal.	\$a < = 12
-----	-------------------	---	------------

Ezekkel az operátorokkal sokszor fogunk találkozni még programjaink során, különösen, ha elágazást vagy például ciklust készítünk.

- **Logikai operátorok:** Ezen operátorok is fontos szerepet töltenek be a PHP programozásban. Segítségükkel kép operandust vagy két összehasonlító feltétel által keletkezett logikai értéket hasonlíthatunk össze (ezek ismerősek lehetnek a Boole-algebrából):

Ope- rátor	Megne- vezés	Igaz, ha:	Példa
And	és	Mind \$a mind \$b igaz.	\$a and \$b
Or	vagy	\$a és \$b között van igaz.	\$a or \$b
Xor	kizáró vagy	\$a és \$b közül pontosan egy igaz.	\$a xor \$b
!	tagadás	\$a nem igaz.	! \$a

& &	és	Mind \$a mind \$b igaz.	\$a && \$b
	vagy	\$a és \$b között van igaz.	\$a \$b

Ezek a felsorolt csoportok a legfontosabb operátorok, amelyeket használni fogunk programjainkban. A példákban is sokszor fel fognak tűnni (pl.: a következő fejezetben) és feltétlenül fontos, hogy megértsük ezek használatát, mert jóformán semmi-re sem megyünk a PHP-val, ha ezeket nem ismerjük.

7. Vezérlési szerkezetek

Ez az anyagrész a kezdők számára kicsit bonyolult lehet, ám akik már használták a vezérlési szerkezeteket más programozási nyelvekben, azoknak könnyebben érthetik meg ezeket a PHP-n belül. Mindenesetre jól érthető példákon keresztül fogjuk bemutatni őket, mivel ezek is létfontosságúak lehetnek a későbbi, összetettebb programok készítése során.

7.1. Elágazás – Az IF utasítás

A program futása közben különböző esetek következhetnek be. Az elágazás feltételtől függően igaz vagy hamis ágba tereli a programot. Azaz a szerver kiértékeli a feltételt és dönt arról, hogy hol folytatódjon a program.

Az IF utasítás szintaktikája:

if (\$t>100)

{ igaz ág kezdete

A kapcsos zárójelek közötti utasítások akkor hajtódnak végre, ha a feltétel igaz.

} igaz ág vége

else

{ hamis ág kezdete

A kapcsos zárójelek közötti utasítások akkor hajtódnak végre, ha a feltétel hamis.

} hamis ág vége

Az IF kulcsszó után megadjuk a vizsgálni kívánt változót és a feltételt (a fenti példában \$t>10). Ha a feltétel igaz, akkor az igaz, ág, ha hamis, akkor a hamis ág fog végrehajtódni.

Fontos! Az érték adás = jellel, a vizsgálat == jellel történik (mint például a C#-ban).

a., Az IF szerkezete ettől eltérő is lehet, például megadhatjuk, hogy a program csak akkor hajtson végre valamilyen műveletet, ha a feltétel igaz (vagyis nem adunk meg hamis ágat):

if (\$t>100)

{

print ("A \$t nagyobb, mint száz.");

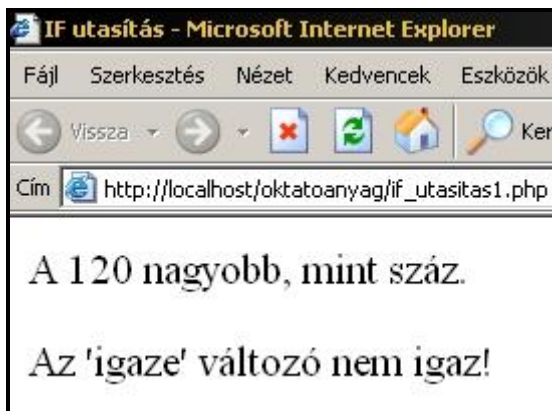
}

b., Olyan változat is lehetséges, hogy a fent leírt IF ... ELSE szintaktikát kibővítjük az IF ... ELSEIF ... ELSE szintaktikára. Ekkor következőképpen épül fel a program szerkezete:

```
<?php
$igaze = "nem";
if ($igaze == "igen")
{
    print ("Az 'igaze' változó tényleg igaz! ");
}
elseif ($igaze == "nem") //az ELSEIF
ágban új feltételt vizsgálunk meg
{
    print ("Az 'igaze' változó nem igaz! ");
}
else
{
    print ("Nem tudom, hogy igaz-e...");
}
?>
```

Ebben az esetben az ELSEIF ág hajtódik végre, mivel a feltétel vizsgálat értéke hamis lesz. Ha az \$igaze változónak 'igen' értéket adtunk volna, akkor a legelső ág hajtódott volna végre. Az ELSE ág akkor hajtódott volna végre esetünkben, ha a \$igaze változónak nem azt adtuk volna értékül, hogy „igen” vagy „nem”, hanem valami (elég, hacsak egyetlen egy karakterrel is) eltérő szöveget.

Ha ezt a két egy példa programba beillesztjük és lefuttatjuk, akkor a következő lesz az eredmény:



A fenti két példa alapján beláthatjuk, hogy az IF utasítás nem túl bonyolult, viszont annál fontosabb szerepe van a PHP programokban. Mindkét példa jól szemlélteti, hogy az elágazásokat milyen szintaktikával valósíthatjuk meg a PHP nyelvben. Ez az egyik legegyszerűbb vezérlési szerkezet, de most bonyolítjuk egy kicsit a következő példában (itt már néhány fontosabb operátor használatát is bemutatjuk, hogy azok használatát is gyakoroljuk!).

7.1.1. Példaprogram az IF utasítás bemutatására

<?php

\$szam1 = 237; //Deklarárunk 3 számot, amelyeket majd megvizsgálunk az IF utasítással.

\$szam2 = 144;

\$szam3 = -51;

**print ("A 'szam1' változó értéke: \$szam1
");**

**print ("A 'szam2' változó értéke: \$szam2
");**

**print ("A 'szam3' változó értéke: \$szam3
");**

**print ("
");**

if (\$szam1>0 and \$szam1%2==0)

/* Azt vizsgáljuk meg, hogy a \$szam1 változó nagyobb-e, mint 0 és osztható-e kettővel. Láthatjuk, hogy az oszthatóság vizsgálata a maradékosztás operátor segítségével történik!*/

{

print ("A \$szam1 nagyobb, mint nulla és osztható kettővel!");

}

elseif (\$szam1<0 and \$szam1%2==0)

{

print ("A \$szam1 kisebb, mint nulla, és osztható kettővel!");

}

else //Az ELSE ágon belülre még egy IF elágazást helyezünk el.

```
{
    if ($szam1>0)
        /*Ez az egymásba ágyazott elágazás jelen esetben szükséges, mert, ha a program az ELSE ágba terelődik, akkor nem osztható a szám 2-vel. Ebben az esetben viszont még egy elágazással ettől függetlenül meg kell néznünk, hogy a 2-vel nem osztható szám negatív, vagy pozitív-e.*/
        {
            print ("A $szam1 nagyobb, mint nulla, viszont nem osztható kettővel!");
        }
    else
    {
        print ("A $szam1 kisebb, mint nulla, és nem osztható kettővel!");
    }
}
print ("  
>");
if ($szam2>0 and $szam2%2==0) //Most a $szam2 változót vizsgáljuk meg ugyanígy!
{
    print ("A $szam2 nagyobb, mint nulla és osztható kettővel!"); //Ebben az esetben az IGAZ ág hajtódik végre.
}
elseif ($szam2<0 and $szam2%2==0)
{

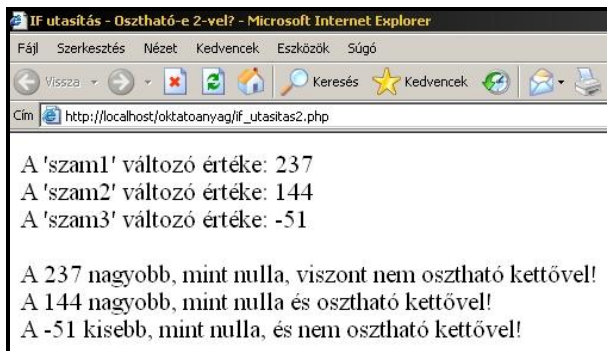
```

```
        print ("A $szam2 kisebb, mint nulla, és
        osztható kettővel!");
    }
else
{
    if ($szam2>0)
    {
        print ("A $szam3 nagyobb, mint nulla,
        viszont nem osztható kettővel!");
    }
    else
    {
        print ("A $szam3 kisebb, mint nulla, és
        nem osztható kettővel!");
    }
}
print ("<br />");
if ($szam3>0 and $szam3%2==0) //A
$szam3 változó már negatív szám, emel-
lett osztható 2-vel, így az ELSEIF ág haj-
tódik majd végre!
{
    print ("A $szam3 nagyobb, mint nulla és
    osztható kettővel!");
}
elseif ($szam3<0 and $szam3%2==0)
{
    print ("A $szam3 kisebb, mint nulla, és
    osztható kettővel!");
}
```

```
else
{
    if ($szam3>0)
    {
        print ("A $szam3 nagyobb, mint nulla,
        viszont nem osztható kettővel!");
    }
    else
    {
        print ("A $szam3 kisebb, mint nulla, és
        nem osztható kettővel!");
    }
}

?>
```

A példaprogram a fentiek alapján megírva a következőt fogja kiírni a képernyőnkre (láthatjuk, hogy a programunk helyesen lett megoldva, mert mindhárom számhoz a megfelelő leírás társul):



7.2. SWITCH szerkezet

Szintaktikája PHP-ben:

```
switch ($nap) { //Zárójelben a változó, amit  
vizsgálni akarunk.  
case 0: print("Ma vasárnap van."); break;  
case 1: print("Ma hétfő van."); break;  
case 2: print("Ma kedd van."); break;  
.  
.  
.  
default: print("Nem tudom milyen nap  
van!"); //egy default ágat is megadhatunk.  
}
```

A **switch** utasítás segítségével többirányú elágazást hozhatunk létre. Az utasítás után zárójelek között megadott változó értéke alapján ágaztatunk el. Kapcsos zárójelek között jön az elágazás kifejtése. Egyes ágakat a **case** paranccsal írhatjuk le, mely után jön az érték (pl.: case 0:). Ha a case kulcsszó után nem számot, hanem szöveget vizsgálunk, akkor azt természetesen "idéző jelek" közé kell tennünk! A kettőspont után kifejtjük { } jelek között (nem kötelező kirakni, anélkül is működik), hogy mit csináljon adott érték után.

Fontos! Minden ágat break-kel zárunk le (különben, ha kimarad az elágazás ágai túlcso-

dulnak a következő utasításokra, és helytelen program jön létre).

Ha a változó értékét az ágak között nem definiáltuk, a default ágon lévő utasítások fognak végrehajtódni.

Most nézzük meg, hogy a szintaktikai példaként leírt program hogyan is néz ki teljes egészében! A programban a korábban már említett `date()` függvényt, annak is a `'w'` paraméterét fogjuk eltárolni a `$nap` változóban, ami az aktuális nap számbeli megfelelőjét adja vissza (ez 0-tól – ami a vasárnap – 6-ig terjed – ami a szombat). A példa jól szemlélteti, hogy mire is tudjuk használni a `SWITCH` szerkezetet. Ez – mivel az `IF` utasításhoz hasonlóan – a vezérlési szerkezetekhez tartozik, a programot más-más irányba terelheti. Most pedig nézzük az aktuális dátum és nap kiíratását a `SWITCH`-csel.

7.2.1. Példaprogram az `SWITCH` szerkezet bemutatására:

<?php

```
$datum=date("Y.m.d.");           //A $datum változóban eltároljuk az aktuális nap dátumát.
```

```
print ("Mai dátum: $datum ");
```

```
$nap=date("w");                   //A $nap változóban eltároljuk az aktuális nap számbeli reprezentációját.
```

```
/*A SWITCH-el megvizsgáljuk, hogy az adott szám  
milyen számértéket ad vissza és ahhoz rendeljük  
hozzá a nap nevét magyarul és ezt ki is írjuk a  
képernyőre a dátum után!*/
```

```
switch ($nap)  
{  
    case 0: print("Ma vasárnap van."); break;  
    case 1: print("Ma hétfő van."); break;  
    case 2: print("Ma kedd van."); break;  
    case 3: print("Ma szerda van."); break;  
    case 4: print("Ma csütörtök van."); break;  
    case 5: print("Ma péntek van."); break;  
    case 6: print("Ma szombat van."); break;  
}
```

?>

A program ezzel a kóddal mindig az aktuális dátumot fogja kiírni számokkal, ponttal elválasztva, valamint az aktuális nap nevét. Pl.: *Mai dátum: 2007.09.29. Ma szombat van.*

Ezt a kis programot használhatjuk weboldalunkon, ha tájékoztatni szeretnénk a látogatóinkat a dátumról. A program ugyan egyszerű, de jól szemlélteti, hogy mit is jelent a „dinamikus weboldal” kifejezés. Minden nap más dátumot fogunk a honlapunkon látni, anélkül, hogy mi beleszerkesztenénk az oldalba. Ugyanígy a SWITCH-csel akár egy névnapköszöntőt is csinálhatunk. Igaz, ennek megírásához az év minden napjához hozzá kell rendelnünk az

aktuális névnapot. Ez persze nem kevés idő, de ha elszántak vagyunk és gyakorolni is szeretnénk, akkor nyugodtan nekiülhetünk!

8. CIKLUSOK

A ciklusok használata esetén egy vagy több utasítás újra és újra végrehajtódik. A ciklus egy utasításblokk többszöri végrehajtására alkalmazott programozói eszköz. A PHP – más programozási nyelvekhez hasonlóan lehetővé teszi mind az előltesztelő, mind a hátultesztelő, mind pedig a növekményes (más néven számláló) ciklusok használatát.

Ahhoz, hogy megértsük a példaprogramokat tisztában kell lennünk az inkrementáló és dekrementáló operátorok szerepével és jelölésével:

Inkrementáló művelet, inkrementáló operátor:

Növeljük a változó értékét 1-gyel.

$\$i++ \leftrightarrow \$i=\$i+1$

Dekrementáló művelet, dekrementáló operátor:

Csökkentjük a változó értékét 1-gyel.

$\$i- \leftrightarrow \$i=\$i-1$

8.1. A WHILE() ciklus

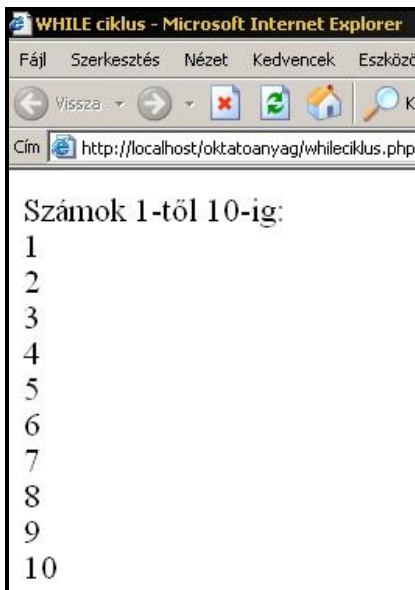
A WHILE() egy előltesztelő ciklus, amelynél a ciklusmag egy adott feltétel teljesülése esetén hajtódik végre.

Szintaktikája PHP-ben:

```
<?php
$i=1;
print ("Számok 1-től 10-ig: <br />");
while ($i<=10)      //végfeltétel megadása
{
    print ("$i <br />");    //ciklusmagban
    lévő utasításblokk
    $i++;
}
?>
```

A WHILE() a legegyszerűbben használható ciklusutasítás. A fenti példa futási eredményén is jól érzékelhetjük (lásd a képernyőképet), hogy ez a ciklus addig hatja végre a kapcsos zárójelek közt lévő utasításblokkot, amíg a while kulcsszó után következő, zárójelben lévő kifejezés (az az a végfeltétel) igaz. Ha a feltétel hamissá válik, a ciklus befejeződik. Olyan is előfordulhat persze, hogy a végfeltétel mindig hamis marad, ilyenkor a ciklus egyszer sem fog végrehajtódni. A másik – ellentétes – esetben, ha a végfeltétel folyamatosan igaz marad, könnyen az egyik legnagyobb programozói

hibába, a **végtelen ciklusba** eshetünk. Ez nem túl szerencsés, mert, ha lefuttatjuk a végtelen ciklust tartalmazó programunkat, akkor csak a böngésző leállításával tudjuk ezt leállítani.



Látható, hogy a fenti példa alapján elkészített WHILE ciklus kiírja a képernyőre a számokat 1-től 10-ig. A `$i` változót deklaráltuk a program elején, a WHILE ciklusban zárójelek között megadtuk azt a végfeltételt, hogy a ciklus addig hajtódjon végre, amíg a `$i` változó értéke kisebb/egyenlő nem lesz

10-zel. A ciklusmagban egy `print ()` függvény segítségével kiíratjuk a `$i` aktuális értékét, majd ezután beszúrtunk egy inkrementáló operátort, amely segítségével az `$i` értékét minden kiíratás után eggyel növeltük (`$i++`). Ezek után a ciklus az új értékre vonatkozóan is megvizsgálta a végfeltételt, és ha az igaznak bizonyult újra és újra végrehajtotta a ciklusmagban lévő utasításblokkot. A ciklus akkor fejeződött be, amikor a `$i` értéke a 10-et. Láthatjuk, hogy ez nem volt nagy ördögösség, de a következőkben bonyolódnak a ciklusok.

8.2. A DO...WHILE() ciklus

A `DO...WHILE ()` egy hátultesztelő ciklus, amelynél először a ciklusmag hajtódik végre, majd ezután történik a feltétel kiértékelése, ami eldönti, hogy kiléphetünk-e a ciklusból vagy nem.

Szintaktikája PHP-ben:

```
<?php
$i=1;
print ("Számok 1-től 10-ig
(DO...WHILE-al): <br/>");
do      //A "do" kulcsszóval kezdődik a
ciklus
```

```
{  
    print ("$_i <br />");    //A ciklusmagban  
    lévő utasítástömb mindenféleképpen lefut  
    egyszer!  
    $_i++;  
}  
while ($_i<=10)    //A ciklus végén található  
    a végfeltétel  
?>
```

Előfordulhat, hogy a ciklusba zárt utasításnak egyszer mindenképpen le kell futnia - akkor is, ha a feltétel mindig hamis. Ilyenkor használjuk a hátultesztelő DO...WHILE() ciklust, ahol a feltétel teljesülését a ciklus végén vizsgáljuk. Az első lefutásnak tehát a hamis feltétel sem akadálya.

Ha a példaként felhozott programot megtekintjük a böngészőnkben, akkor az ugyanazt az eredményt adja, mint az előző pontban bemutatott WHILE() ciklus példaprogramja (vagyis ugyanúgy kiírja a számokat 1-től 10-ig és aztán kilép a ciklusból). Tehát, ha azt szeretnénk, hogy a ciklus mindenféleképpen egyszer végrehajtódjon, használjuk a DO...WHILE()-t, ha nem akkor nyugodtan maradhatunk a sima WHILE() ciklusnál.

8.3. A FOR() ciklus

A FOR() ciklus egy növekményes, előltesztelő ciklus. A ciklusmagot egy előre meghatározott számszor hajtjuk végre. Szintaktikája PHP-ben:

```
for ($i=1; $i<=10; i=i+1) //kezdőérték,  
végfeltétel, lépés megadása, pontosvesz-  
szővel elválasztva
```

```
{
```

CIKLUSMAG: ide írjuk, milyen utasítást hajtson végre a ciklus, ameddig nem teljesül a végfeltétel.

```
}
```

A FOR ciklusnak a zárójelen belül 3 paramétere van:

- **Kezdőérték** (A fenti példában **\$i=1**, tehát a kezdőérték 1)
- **Végfeltétel, végérték** (Ez vagy igaz vagy hamis. A fenti példában a **\$i<=10**, tehát a ciklus addig tart, amíg az *i* változó értéke kisebb marad, mint 10 vagy egyenlő nem lesz 10-zel)
- **Lépés** (A ciklus a fenti példában az *i* értékét **növeli eggyel mindaddig**, amíg az *i* nem lesz egyenlő tízzel, tehát a végértékkel)

A FOR () ciklus a WHILE () egy speciális esete, számlálásra, illetve lista megjelenítésére könnyebben használható. A ciklusváltozót általában i-vel jelöljük. Megtehetjük azt is, hogy a zárójelben lévő paraméterek közül esetleg kihagyunk valamit, ekkor egy más után követi egymást két pontosvessző. Ez azonban **könnyen végtelen ciklushoz vezet**, így érdemes inkább mindegyik paramétert rendesen megadni.

Most nézzük, hogyan tudjuk a FOR ciklust használni! Készítsünk vele szorzótáblát!

8.3.1. Példaprogram a FOR () ciklus bemutatására

```
<?php
print ("5-ös szorzótábla:<br />");
for ($i=1; $i<=10; $i++)
{
    print ("5 * $i = ". 5*$i."<br />");
}
?>
```

A program a FOR ciklus segítségével kiírja az ötös szorzótáblát, az egymás alá a szorzás eredményével együtt.

8.4. Egymásba ágyazott ciklusok

Most nézzük meg a ciklusok zárásaként, hogy hogyan ágyazhatunk egymásba két ciklust. Sok esetben előfordulhat PHP-nál, hogy az adatokat táblázatba kell rendeznünk (pl.: adatbázisból való lekérdezés esetében). Ezt egy ciklus esetében is megtehetjük, ha a használjuk a HTML nyelvből ismert táblázatkészítő TAG utasításokat.

A példaprogramban továbbra is maradunk a szorzótáblánál, de most az egymásba ágyazást használva egy olyan táblázatot hozunk létre, amely tartalmazza a szorzótáblákat 1-től 10-ig!

8.4.1. Példaprogram az egymásba ágyazott ciklusokhoz

```
<table border="1">
<!-- Létrehozzuk a táblázatot, még a PHP kódblokk
előtt. -->
<?php
    for ($i=1; $i<=10; $i++) //i mutatja a so-
        rokat - A külső FOR ciklussal készülnek a sorok!
    {
        print("<tr>"); //új sor létrehozása
        for ($j=1;$j<=10;$j++) //j mutatja
            az oszlopokat - A belső FOR ciklus készíti
            az oszlopokat!
```

```

    {
        $s=$i*$j;           //Deklaráljuk az 's'
                             változó, ami a szorzás eredménye
        print("<td>$s</td>");    //A belső
                             ciklus felel azért is, hogy a szorzás ered-
                             ménye megjelenjen a képernyőn.
    }
    print("</tr>");
}
?>
</table>
<!-- Lezárjuk a táblázatot a PHP kódblokk után.-->

```

A végeredményünk egy 10 oszlop széles és 10 sor magas táblázat lesz, amely tartalmazza az 1-es szorzótáblától a 10-esig az összes szorzat eredményét, melyet könnyedén leolvashatunk a táblázatról.

Első ránézésre nehéznek tűnnek a ciklusok, legfőképpen a legutóbbi példában, de megfelelő szintű gyakorlás mellett gyerekjáték lesz a használatuk!

A következő fejezetben már a dinamikusabb PHP oldalak felé kacsingatunk. Megismerkedünk az űrlapokkal és megtanuljunk, hogyan is kell őket elkészíteni, és mi mindenre is tudjuk használni!

Egymásba ágyazott ciklusok - Microsoft Internet Exp

Fájl Szerkesztés Nézet Kedvencek Eszközök Súgó

Vissza Keresés

Cím <http://localhost/oktatoanyag/szorzotabla.php>

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

9. Úrlapok

A PHP-ben talán az egyik legfontosabb szerepe az úrlapoknak van. Ezek segítségével tudjuk igazán dinamikussá tenni weboldalainkat azáltal, hogy admin felületet kialakítva tudjuk tartalommal feltölteni oldalainkat. Ezek segítségével kérhetünk be adatot a felhasználóktól, amikkel PHP programunk dolgozni, fog. Az adminpanel segítségével felvehetünk, módosíthatunk, törölhetünk tartalmat oldalainkról, anélkül, hogy HTML-t szerkesztenénk. Ezek kialakítása természetesen nem kis időt vesz igénybe, azonban a későbbiekben az erre szánt idő többszörösét spórolhatjuk meg, és kényelmesebben szerkeszthetjük honlapunkat, ahonnan csak akarjuk, weblapszerkesztő program használata nélkül!

De ne rohanjunk ennyire előre, először is nézzük mik is az úrlapok elemei és hogyan is kell felépíteni egy úrlapot!

Űrlap: valamilyen adatbeviteli célból elkészített formanyomtatvány, amellyel segítjük a felhasználót a kitöltésben. (Másképpen pedig segíti a programozó dolgát a beérkező adatok feldolgozása során.)

Dreamweaver-ben: Form (Űrlap) eszköztár. A menüsor alatt található lenyíló listában találjuk ezt (amelyen alapesetben a Common felirat látható). Miután kiválasztottuk a Form eszköztárat, a következő elemeket láthatjuk majd:



Fontos! Az űrlap elemeket mindig egy Form belsejébe rakjuk (különben nem tudjuk feldolgozni)!!! Létrehozásához az eszköztáron a legelső, Form feliratú ikonra kattintsunk:



Most pedig nézzük meg melyek a számunkra legfontosabb űrlap elemek a fenti eszköztárról!

9.1. Űrlap elemek:

1. Text Filed



Tulajdonságai:

- a. Char width (karakter szélesség)
- b. Max chars (hány karakter befogadására alkalmas)
- c. Type (Single line, Multi line, Password)
Single line: egysoros beviteli mező.
Multiline: többsoros beviteli mező. Meglehet adni a sorok számát (Num lines) és a hosszú sorok tördelését (Wrap).
Password: jelszóbevitelre használható (a beírt szöveget * 'csillag' karakterrel rejti el).
- d. Initial value (Init val): Alapértelmezett érték, megjelenéskor mi legyen benne.
- e. Name: PHP szempontjából az egyik legfontosabb tulajdonság, mivel a feldolgozáskor a névvel tudunk hivatkozni a mező értékére (a benne lévő adathoz érdemes társítani a textfield nevét).

2. Hidden Field: rejtett mező**3. Textarea:** ugyanaz, mint a Multi line.**4. Checkbox:** igen/nem eldöntendő kérdésekhez

- a. Name
- b. Checked value: feldolgozásnál ez jelzi, hogy igaz az értéke
- c. Initial state: kezdő állapot:
 - Checked: pipás
 - Unchecked: alapértelmezésként nincs kipipálva

5. Radio button / radio group:

Akkor használjuk, ha több értékből egyet kell kiválasztani.

- a. Name
- b. Label (ez a szöveg jelenik meg)
- c. Value (ezt az értéket adja tovább)

6. List / Menü:

Ugyanaz a funkciója, mint az előzőleg leírtaknak, de a menü kisebb helyet foglal (mert legördülő lista)

A listánál (a lista típusánál) meg lehet adni a:

- a. magasságot (Height)
- b. és, hogy lehetőség van-e több elem kijelölésére (SELECTIONS – Allow multiple)

7. Button:

(alapértelmezettként Submit – feldolgoz – a neve)

- a. Value: gomb felirata
- b. Action:
 - submit: feldolgoz
 - reset: kezdő állapotba helyez
 - none: semmit sem csinál

9.2. Űrlap: az action TAG és a paraméter átadás

Ha megnézzük az űrlapot, amelyet a Form-ra kattintva szúrunk be a weblapszerkesztőn kód-nézetében, akkor láthatjuk, hogy az alábbi kód-részlettel kezdődik az űrlap:

```
< form id="form1" methods="post" action="">
```

Nézzük meg pontosan mik is ezek a tag-ek:

form id: az űrlap azonosítója (neve), ezzel tudunk rá hivatkozni.

methods: metódusok, itt adhatjuk meg, hogy milyen módon adja át az űrlap a paramétereket a feldolgozó oldalnak.

Az adatok átadásának – az action paraméterben megadott weboldalnak való átadásának – két módja van:

- **POST:** Ekkor a weboldal HEAD részében mennek át az adatok (header formában történnek).
- **GET:** Ez a módszer az URL címbe adja át a paramétereket, azaz bárki láthatja, sőt meg is változtathatja (Pl.: keresés a GOOGLE-ben).

Folyamata: A form kezdetekor a „method” nevű tulajdonságot GET-re állítjuk, így az URL címében fogja átadni a paramétereket.

Pl.: **ellenoriz.php?nev=HG&auto=checkbox**

Az oldal neve után jön a ? (kérdőjel), amely jelzi, hogy paraméterek következnek. Aztán jön a paraméter neve és az egyenlőség jel után az értéke. Ha több paraméter van, akkor & jellel választjuk el őket egymástól! Ebben az esetben az átvevő oldalon a \$_GET['nev']-vel vesszük át az adatot.

Header: header ("Location: siker.php");

A location-t **nagy L** betűvel kell írni!

Ezzel a paranccsal a böngészőt átirányíthatjuk automatikusan egy másik oldalra. Mivel az oldal nem is látszik, nem szabad semmit kiírni az eredmény oldalba! Azaz <? tag-gel kezdjük az oldalt, nincs HTML kód! (Arra is figyelni kell, hogy ne legyenek üres sorok az oldal forráskódjában, mert akkor szintén nem fog működni!)

action: Az action tag-be kell beírni azt az oldalt, ahova a Submit gomb megnyomásával lép az oldal. Ez az űrlap **feldolgozó oldala**.

A POST-tal és GET-tel történő paraméterátadásra példát találunk a „Megvalósítások PHP-vel” c. fejezetben!

Miután megismerkedtünk a legfontosabb űrlap elemekkel, az action TAG-gal és a paraméterátadás lehetséges formáival, nézzük hogyan is valósítható meg ez a gyakorlatban. Ezek megértése nagyon fontos, mert a későbbiekben – például egy admin panel kialakításánál – együtt kell használnunk az eddig leírt ismereteinket, a később bemutatott adatbázis kezelési parancsokkal.

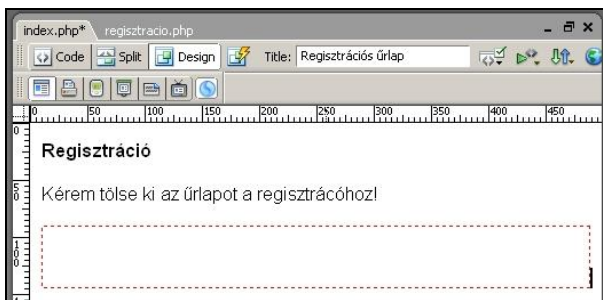
És most nézzünk egy pár példát űrlapok készítésére!


10. Űrlapok használata a gyakorlatban

10.1. Regisztrációs űrlap készítése

Elsőként egy regisztrációs űrlapot fogunk készíteni, amely elsőre elég összetettnek tűnhet, ám lépésről lépésre haladva átnézzük a készítés folyamatát! Kezdjünk is bele!

0., Előre megtervezzük, milyen adatokat is szeretnénk az űrlapon bekérni a felhasználótól! A példában mi kérni fogjuk a vezeték, illetve a keresztnévét, nemét, felhasználónevét és jelszavát, valamint a mobilszámát. Emellett megkérdezzük tőle, hogy van-e autója és, hogy dohányzik-e. A példában használni fogjuk a Form eszköztárról a **Text Field-et**, a **Checkbox-t**, a **Radio Button-t**, a **List/Menu-t** és persze a **Button-t** is.



1., Az előző lépés azért volt 0., mert mielőtt elkezdjük a munkát fontos a tervezés (így lesz ez majd akkor is, ha adatbázist tervezünk és készítünk). Most lássunk hozzá a tényleges készítéshez! A Dreamweaver-ben készítsünk egy új PHP oldalt: File → New → Dinamic page → PHP. Mentsük el index.php néven (persze nem kötelező ez a név, de a példában így fogunk rá hivatkozni). Az oldalunk – amibe most az űrlapot csinálni fogjuk – nem tartalmaz PHP blokkot. Fontos viszont, hogy az űrlap feldolgozásához szükséges **Form** belsejébe tegyük az űrlap elemeit, így első lépésként szúrjuk ezt be az eszköztár Form  ikonjára kattintva (A Dreamweaver design nézetében a Form határát egy szaggatott piros vonal jelzi, lásd: a képen)! A title TAG-et és a fenti szövegeket én rögtön a mentés után begépettük, de ezek számunkra nem fontosak, csak azért vannak feltüntetve, hogy lássuk mit is készítünk. Miután elhelyeztük a Form-ot az olda-

lon rögtön át is válthatunk kódnézetbe és ahol az űrlap action és method TAG-ját fogjuk beállítani:

```
<form action="regisztracio.php" method="post">
```

A mi példánkban állítsuk be, hogy az oldal a gombra való kattintással a regisztracio.php-ra ugorjon és, hogy a paramétereket POST metódussal (vagyis, hogy titkosítva, az oldal HEAD részében) adja át a feldolgozó oldalnak. Ezt persze később is megadhatjuk, de jobb előbb túlesni rajta, nehogy megfélekedezzünk róla (a regisztracio.php-t is létrehozhatjuk most is, ha akarjuk, de előbb az index.php-n készítjük el az űrlapot, aztán készítjük el a feldolgozó oldalt.)!

2., Miután az előző pontban felsoroltakat elvégeztük, elhelyezhetjük a Formon belül az űrlap elemeket! Lehetőleg ezeket a rendezettség miatt egy táblázatba helyezzük el (ez legyen 9 soros és 2 oszlopos – az egyikoszlopba megy, mit is kell írni az adott mezőbe, a másik oszlopba meg maga az űrlap elem). Most a 0. pontban leírtak szerint a felhasználótól mi kérni fogjuk a vezetéket, illetve a keresztnévét, nemét, felhasználónevét és jelszavát, valamint a mobilszámát. Emellett megkérdezzük tőle, hogy van-e autója és, hogy dohányzik-e. Eszerint töltsük ki értelemszerűen a táblázat bal oszlopának sorait!

Így fog kinézni az űrlapunk, ha felhelyeztük az űrlap elemeket! Nézzük ezeket sorjában!

Regisztráció

Kérem tölse ki az űrlapot a regisztrációhoz!

Vezetéknév:	<input type="text"/>
Keresztnév:	<input type="text"/>
Neme:	<input type="radio"/> Nő <input checked="" type="radio"/> Férfi
Felhasználónév:	<input type="text"/>
Jelszó: (max 8 karakter)	<input type="text"/>
Mobilszám:	20 <input type="text"/>
Van autója?	<input type="checkbox"/>
Dohányzik?	<input type="checkbox"/>
<input type="button" value="Regisztrálok!"/>	

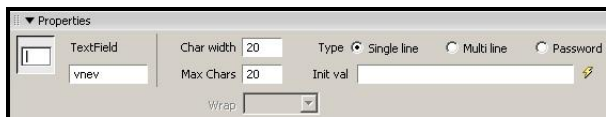
3., Űrlap elemek elhelyezése

a. Vezetéknév, keresztnév: Ez a két űrlap

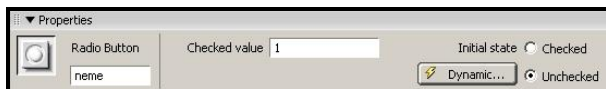


elem Text Field lesz. Az elsőnek a vnev-et, a másodiknak a knev-et adtok a Properties fülön. Ezekkel a nevekkel fogunk a feldolgozó oldalon hivatkozni a begépelt értékekre, a feldolgozó oldal PHP blokkjában. Emellett mindkét esetben beállítottuk, hogy a beviteli mező szélessége és a maximum karakterszám 20 karakternyi legyen. Ez elég kell, hogy legyen mind egy ke-

resztnév, mind egy vezetéknév esetében, ha csak nincs feltűnően hosszú neve valakinek. A mező típusa Single line, tehát egysoros beviteli mező.

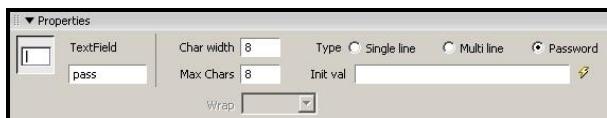


b. Neme: Hogy a felhasználó nemét megkérdezzük szükségünk lesz 2 darab Radio Button-ra. Mindkettőnek ugyanaz lesz a neve: neme. Az értékeknél viszont feltétlenül más értéket kell adnunk mind a Nő, mind a Férfi Radio Buttonnak (ennek okát majd meglátjuk a feldolgozó oldalon). A Nő-nél a „bepipált értéknek”, azaz a Checked value-nak egyet adtunk, a Férfinál kettőt. Utóbbinál azt is beállítottuk, hogy alaphoz legyen bejelölve, azaz, hogy a felhasználóhoz akkor is társuljon egy nem, ha véletlenül nem jelölte volna be.



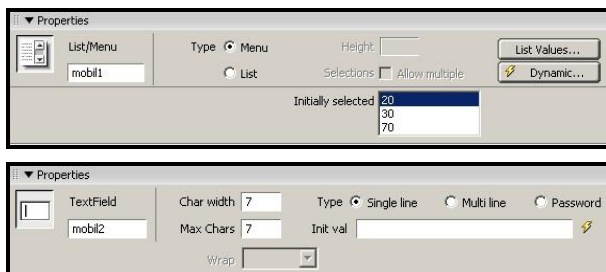
c. Felhasználónév, jelszó: A felhasználónév-nél nincs semmi újdonság, ugyanúgy Text Field mint az első kettő, csak kevesebbre állítottuk a max karakterszámot (16-ra, a mező nevét egyébként user-re állítottuk). Nekünk most a jelszó mező

érdekes, ami bár ugyanúgy Text Field, bár típusát tekintve nem Single line, hanem Password. Ennek a nevéen is látszik, hogy jelszó bevitelére használjuk, ugyanis csillagok jelzik a beírt karaktereket, tehát azok rejtve maradnak. Íme a jelszó beviteli mező beállításai:



d. Mobilszám: A mobilszámot két űrlap elemmel kérjük be. Számunkra az első lehet újdonság, ugyanis a mobilszolgáltató körzetszámát egy előre megadott listából választhatja ki a felhasználó. Ez az elem egy List/Menu. Mi a típusát tekintve Menu-ként használjuk (ha List-et használnánk nem legördülő lista lenne az eredmény – hanem esetünkben – egy 3 számot tartalmazó lista, ahonnan szintén választhatnánk). Ha valakinek mégis a List a szimpatikusabb válassza azt, csak ügyeljen arra, hogy a Selection beállításánál ne engedélyezze az Allow multiple-t, mert azzal elérhetővé válna több körzetszám bejelölése – persze ez itt felesleges, mert csak egyet kell megadni! A List values-re kattintva adhatjuk meg, hogy a listán mi jelenjen meg (Item label), illetve, hogy ahhoz milyen érték társuljon (Value - a feldolgozó oldalnak ez utóbbi a

fontos, pl.: betársíthatnánk az egyes körzetszámokhoz a mobilszolgáltató nevét, de ez most nekünk nem kell). A mobilszám második felét egy sima Text Field-el kérjük be, ebben nincs semmi különös, csak azt kell megadni, hogy a mobilszám max 7 karakter legyen (mivel jelenleg a mobilszámok hét számjegyből állnak). Íme a beállítások:



e. Van-e autója?, Dohányzik-e?: Ezt a két adatot két Checkbox („kipipálós doboz”) űrlap-elemmel kérjük be a felhasználótól. A példában az első elemnek auto, a másodiknak a dohányzik a neve. Úgy állítottuk be ezeket, hogy alapból ne legyenek bepipálva (Unchecked). A „Checked value”-hez, vagyis a „Bepipált értékhez” egyet írtunk, így a feldolgozó oldal felé küldött érték, mindkét űrlapelemnél egy lesz.

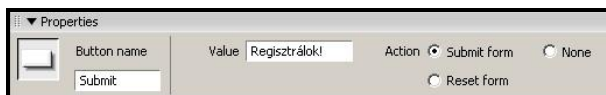


f. Regisztrálok! gomb: Mint az űrlapok elemeinek leírásánál tisztáztuk, hogy a Button elemmel tudunk arra a feldolgozó oldalra lépni, amelyet a Form-unk action TAG-jébe beállítottunk (Esetünkben ez a regisztracio.php). A gomboknak alapból Submit (Feldolgoz) a neve, ezen nem változtattunk. A Value mezőbe beírt szöveg fog a gombon megjelenni feliratként. Az Action checkbox-oknál bejelölhetjük, hogy milyen feladatot végezzen a gomb:

Submit form – a feldolgozó oldalra ugrik,

Reset form – kiüríti az űrlapot (minden töröl a mezőkből, mintha egy új űrlapot kértünk volna),

None – nem csinál semmit.



Ezzel át is vettük azon űrlap elemeket, amelyeket a példaprogramba beillesztettünk. Most nézzük a feldolgozó oldal felépítését!

4., A feldolgozó oldal (regisztracio.php) elkészítése.

Az előző pontokban elkészítettük az index.php-t, ami egy komplett űrlap, amelyet kitöltve a feldolgozó oldalra, a regisztracio.php-ra ugrik a program. Most ezt az oldalt készítjük el. Ebben már szerepel PHP blokk, viszont űrlap elem nem.

A szokásos módon mutatjuk be az oldal felépítését, a HTML tag-eket kihagyjuk, csak a PHP kódot illesztettük be. Lássuk!

<?php

\$vnev=\$_POST['vnev']; //A GET illetve a POST tömbökben tárolódnak az űrlap beviteli mezői és azok értékei.

\$knev=\$_POST['knev']; //Mi a POST metódust használjuk (tehát a paraméterek titkosítva adódnak át).

\$neme=\$_POST['neme']; //Láthatjuk, hogy PHP oldalon létrehozott változóknak érdemes ugyanazt a nevet adni, mint az eredeti űrlap elemnek.

\$user=\$_POST['user']; //Az összes űrlap elem adatát letároljuk ily módon egy változóba.

\$pass=\$_POST['pass']; //Így tudunk a deklarálások után műveleteket végezni az átvett adatokkal.

\$mobil1=\$_POST['mobil1'];

\$mobil2=\$_POST['mobil2'];

\$auto=\$_POST['auto'];

\$dohanyzik=\$_POST['dohanyzik']; //Miután átvettük az összes adatot, kiíratjuk a regisztráció eredményét a képernyőre.

**print ("Vezetéknév: \$vnev
");** //Az első 2 print függvénnyel kiíratjuk a felhasználó vezetéknévét, illetve keresztnévét.

**print ("Keresztnév: \$knev
");**

print ("Neme: ");

if (\$neme = 1) //A nem kiíratásához már szükségünk lesz egy IF elágazásra.

{ //Ha a Nő Radio Button van bejelölve, akkor annak értéke 1 lesz a feldolgozó oldalon.

**print ("Nő
");** //Tehát, ha a \$neme változó értéke 1, akkor azt írja ki, hogy Nő, ha nem akkor azt, hogy Férfi.

}

else

{

**print ("Férfi
");**

}

**print ("Felhasználó neve: \$user
");**

**print ("Jelszava: \$pass
");**

//Itt a jelszó már látható lesz!

**print ("Mobilszám: +36 \$mobil1 \$mobil2
");**

//A telefonszámot íratjuk ki. Előbb jön a körzet-szám, aztán a hétjegyű mobilszám.

if (\$auto = 1)

//A két Checkbox feldolgozása is IF elágazással történik, akárcsak a Radio Button-ok esetében.

{

**print ("Igen, van autóm.
");**

}

else //Ha nem jelöltük be a Checkbox-t, akkor a hamis ág fut le.

{

**print ("Nem, nincs autóm.
");**

}

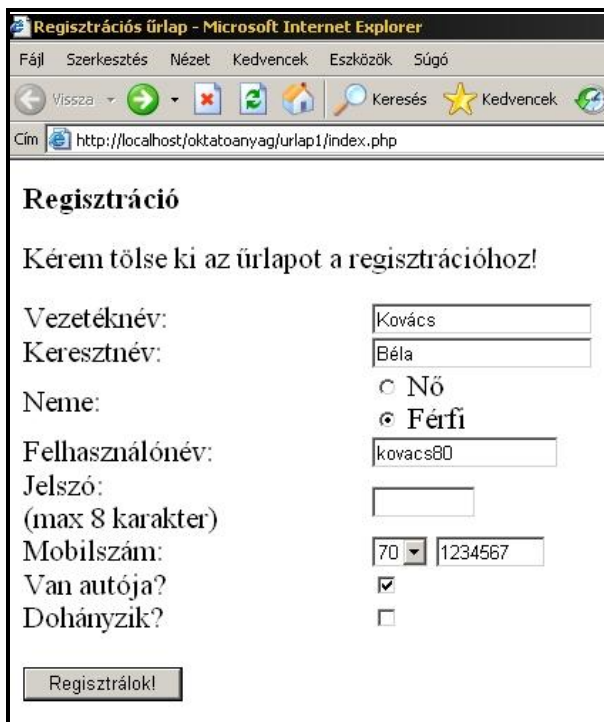
if (\$dohanyzik= =1) //Ugyanaz történik, mint az \$auto esetében.

```
{  
print ("Igen, dohányzom.<br />");  
}  
else  
{  
print ("Nem dohányzom.<br />");  
}
```

?>

Ezzel el is készítettük a feldolgozó oldalunkat. Mint látható az űrlap elemek elhelyezése, majd azok értékeinek átvétele és feldolgozása nem túl bonyolult, még ha elsőre annak is tűnhet! Fontos még megjegyezni a példaprogrammal kapcsolatban, hogy ez valójában nem továbbít semmilyen adatot a felhasználóról az adatbázisba, mivel nem használtunk semmilyen adatbázis kezelő parancsot a program során. Erre most még nem is volt szükségünk, mivel a példa lényege az volt, hogy megtanuljuk az űrlap elemek szerepét, paramétereit és ezek feldolgozásának folyamatát. A későbbiekben ezeket össze fogjuk kötni adatbázis ismereteinkkel és létre fogunk hozni egy admin felületről szerkeszthető kisebb PHP programot (ez egy linkadatbázis, linkajánló lesz tulajdonképpen).

A regisztrációs programunk a megírás után hasonlóképpen kell, hogy működjön:



Regisztrációs űrlap - Microsoft Internet Explorer

Fájl Szerkesztés Nézet Kedvencek Eszközök Súgó

Vissza Választás Újraöltöltés Tárolás Keresés Kedvencek

Cím <http://localhost/oktatoanyag/urlop1/index.php>

Regisztráció

Kérem tölse ki az űrlapot a regisztrációhoz!

Vezetéknév:

Keresztnév:

Neme: ☐ Nő ☒ Férfi

Felhasználónév:

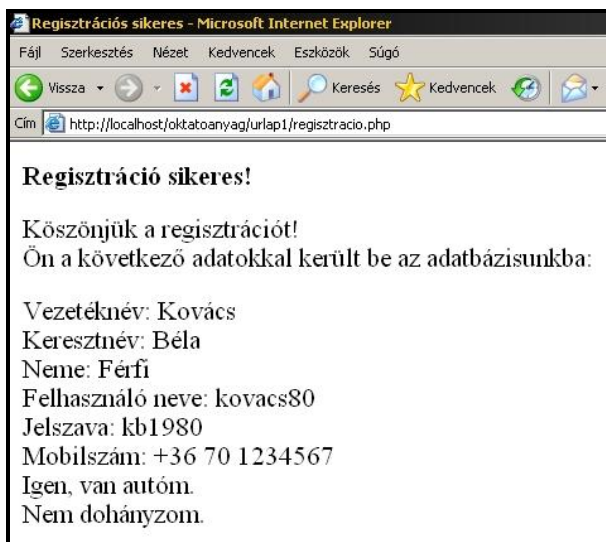
Jelszó:

(max 8 karakter)

Mobilszám:

Van autója? ☒

Dohányzik? ☐



10.2. Téglalap kerülete és területe űrlap segítségével

Ennél a programnál már könnyebb dolgunk lesz, mivel már az eddig megismert űrlap elemek közül csak a Text Field-el kell dolgoznunk. A program tulajdonképpen sokkal egyszerűbb, mint az előző, de a különbség annyi, hogy itt nem csak „szimpla” adatbekérést hajtunk végre. Most egy interaktív programot készítünk, amellyel a felhasználó ki tudja számolni egy téglalap kerületét

és területét azon számokkal, amelyeket ő gépel a beviteli mezőbe (ezt fontos volt megemlíteni, mert az úrlapok megismerés előtt mi csak ún. „programba beépített” változókat használtunk a példákban, vagyis a felhasználó semmiféleképpen nem tudott beleavatkozni a program futásába). Most pedig lássuk az elkészítést!

0., Mivel a téglalapnak 2 paraméterét kell bekérnünk a felhasználótól (ami az „a” és a „b” oldal lesznek), ezért a Form-on belül két Text Field-re lesz szükségünk. Készítünk egy úrlapot, amelyen bekérjük a két számot (index.php) és egy feldolgozó PHP oldalt (eredmeny.php), amelyen átvesszük a két számot és elvégezzük a számítást, majd kiíratjuk az eredményt a képernyőre.

1., Index.php elkészítése, úrlap elemek elhelyezés

a. Létrehozzuk az index.php-t. Beszúrunk egy Form-ot az eszköztárról.



Ezután kitöltjük az alábbiak szerint az aciton és method TAG-eket:

<form action="eredmeny.php" method="get">


Láthatjuk, hogy most a feldolgozó oldalunk az eredmeny.php lesz, valamint, hogy a GET-es paraméter átadást fogjuk használni. Utóbbi azért nem POST, mert most nem kell az átadott paramétereket titkosítanunk, így azok az oldal URL címében adódnak át a következőképpen:

`http://localhost/oktatoanyag/urlap2/eredmeny.php?aszam=4&bszam=6&Submit=Kisz%E1mol%21`

A fenti paraméterekben az **aszam** az oldalt takarja, a **bszam** a b oldalt a Submit pedig a gombra utal. Fontos megemlíteni, hogy a GET-tel történő paraméter átadásnál az oldal neve után kérdőjel jelzi, hogy paraméterek következnek, majd ezeket felsoroljuk & jellel elválasztva. Például:

`http://www.domainnev.hu/oldalneve.php?parameter1=0¶meter2=43¶meter3=7`

Szúrjunk be egy táblázatot a rendezettebb megjelenés érdekében, ebben helyezzük el majd a szövegeket és űrlap elemeket a következőképpen:



Téglalap kerülete, területe - Microsoft Internet Explorer

Fájl Szerkesztés Nézet Kedvencek Eszközök Súgó

Vissza Keresés Kedvencek

Cím `http://localhost/oktatoanyag/urlap2/index.php`

Téglalap területe és kerülete

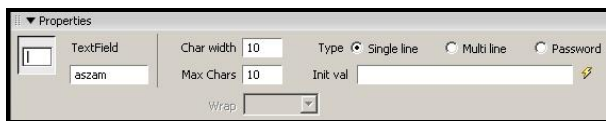
Kérem adja meg a téglalap a és b oldalának hosszát!

a oldal:

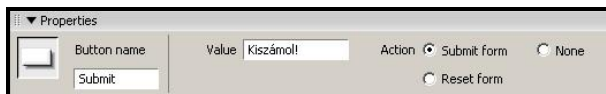
b oldal:

b. Most már elhelyezhetjük az űrlap elemeit! A számok bekéréshez két Text Field-et fogunk használni. Karakter szélességnek és max karakter-

nek én 10-et adtunk meg, típusa Single line. Az első neve aszam, a második-é bszam:



c. Ha az első két elemmel készen vagyunk, akkor már csak a gomb elhelyezésére van szükségünk az oldalon, amire kattintva az eredmény.php-ra ugrunk. A beállításoknál a Value (tehát a gomb felirata) jelen esetben Kiszámol!, az Action pedig Submit form:



2., Az űrlappal készen is volnánk, most jön a lényeg a feldolgozó oldal, vagyis az eredmény.php! Nézzük most ennek a PHP kódját!

<?php

\$aszam=\$_GET['aszam']; //Átvesszük a két értéket az űrlapról és letároljuk őket az azonos nevű változókba.

\$bszam=\$_GET['bszam'];

if ((\$aszam>0) and (\$bszam>0))

/*A számolás megkezdéséhez szükségünk van egy IF elágazásra, mert ugyebár nem engedhetjük, hogy a felhasználó 0-át vagy annál ki-

sebb számot írjon be oldal hosszának, mert ugye olyan téglalap nem létezik. A feltétel tehát úgy szól, hogy: ha az a szám és a b szám is nagyobb nullánál, akkor hajtsa végre az igaz ágat, ha nem írjon ki hibaüzenetet! */

```

{
    $kerulet=2*($aszam+$bszam);
    //Ha a és b nagyobb, mint nulla elvégezzük
    a két műveletet.
    $sterulet=$aszam*$bszam;
    //Mind a kerület, mind a terület kiszámítá-
    sát egy változóban adtuk meg,
    print("A téglalap kerülete (K):
$kerulet <br />");
    //így kiíratáskor már csak a $kerulet és
    $sterulet változókat kell begépelni.
    print("A téglalap területe (T):
$sterulet");
}
else //Ha a hamis ág fut le, akkor a
program nem ad vissza semmiféle eredményt,
csak egy piros hibaüzenetet!
{
    print("<font color='red'>A téglalap
egyik oldala sem lehet kisebb vagy egyen-
lő nullával!<br />Kérem csak pozitív
számot adjon meg!</font><br />");
}
?>

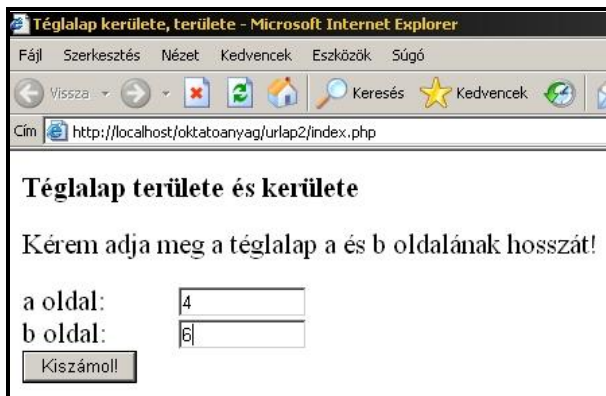
```

A PHP blokkon kívül szúrjunk még be egy linket, ami az index.php-ra vezet, hogy ha valaki hibás számot írt be vagy újra akar számoltatni valamit, vissza tudjon lépni!

```
<p><a href="index.php">Vissza az oldalak megadásához</a></p>
```

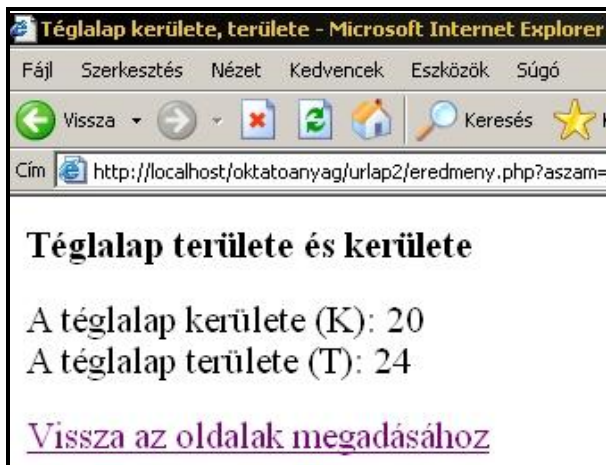
Amennyiben mindent jól csináltunk, akkor a következőképpen fog működni a programunk:

1. Első lépésben beírjuk a két értéket a két beviteli mezőbe:



The screenshot shows a Microsoft Internet Explorer window with the title "Téglalap kerülete, területe - Microsoft Internet Explorer". The address bar shows the URL "http://localhost/oktatoanyag/urlop2/index.php". The main content area displays the text "Téglalap területe és kerülete" followed by the instruction "Kérem adja meg a téglalap a és b oldalának hosszát!". Below this, there are two input fields: "a oldal:" with the value "4" and "b oldal:" with the value "6". At the bottom left, there is a button labeled "Kiszámoll!".

2. Ezek után a „Kiszámol!” gombra kattintva az oldal kiírja a téglalap kerületét és területét:



Ha szeretnénk még gyakorolni az űrlapok készítését, akkor nyugodtan készítsünk magunknak a példaprogramokhoz hasonlóakat! Pl.:

- Kör kerülete, területe
- Háromszög területe, kerülete
- Kocka vagy téglatest felszíne, térfogata
- Gömb felszíne, térfogata
- Újabb regisztrációs űrlapok készítése stb.

Ezek segítenek abban, hogy megfelelő rutint szerezzünk az űrlapok elkészítésében és hozzájárulnak – az előző programhoz hasonlóan – például a

vezérlési szerkezetek gyakorlásához is. Ez fontos, mert a következő anyagrészen megismerkedünk az adatbázis kezelésével MySQL-ben, majd azt a tudást vegyítve űrlap ismereteinkkel komolyabb programokat is írunk!

11. SQL adatbázisok

11.1. Bevezető, alapfogalmak

Adatbázis: logikailag összefüggő információ vagy adatgyűjtemény. Adott célból összeállított adatok és objektumok gyűjteménye.

Adatbázis-kezelők: Speciális programrendszer, amellyel lehetséges az adatbázisok kezelése.

Relációs adatbázis-kezelő rendszer: olyan program, amelyik az adatokat számítógépen, táblázatokban tárolja, rendezi, illetve onnan keresi vissza. Pl.: *Microsoft Access*

Jellemzőik:

- Adatbázis létrehozása, kezelése
- Biztonság
 - Felhasználók jogköre
 - Adatmentés, tükrözés (pl.: feliratok.hu)
- Elérhetőség
- Több felhasználó

11.2. MySQL adatbázisok

A MySQL egy nyílt forrás-kódú adatbázis rendszer. A MySQL-es adatbázis-kezelés megértéséhez és elsajátításához szükségünk lesz korábban megszerzett tudásunkra. Ha korábban tanultunk adatbázis-kezelést pl.: Access-ben, akkor könnyű dolgunk lesz a MySQL megismerését és kezelését illetően.



Adatbázis → Adattábla: A MySQL-ben az adatok adatbázisokban, azokon belül pedig adattáblákban tárolódnak. Egy adattábla az alábbiak szerint épül fel:

Rekord →				

↑ **Mező**

Egy EGYEDET jellemzünk.

Adattábla: logikailag összetartozó adatok sorokból (rekordok) és oszlopokból (mezők) álló elrendezése.

Rekord: egyed összes adata. Az adattábla egyetlen sora, az információ csoport egyetlen elemének összes adatát tartalmazza.

Mező: egyed egyetlen tulajdonsága.

Fontos! Mező- és táblanevekben **nem használunk** ékezetes betűt!!! Az SQL nyelvet az angol nyelvten alapján építették fel. Az SQL nem érzékeny a kis és nagy betűs különbségekre (ám azonban ne felejtjük el, hogy a PHP igen!). Ami az SQL parancsokat illeti a jövőben nagybetűs formában írjuk majd őket, a többi kiegészítő szöveget (mint például a táblaneveket, mezőneveket) pedig kisbetűvel.

11.3. Egy tábla felépítése MySQL belül

Figyeljük meg milyen adattípusokat is használunk a példában, és hogy milyen attribútumai vannak a bevitt adatoknak!

Mezőnév	Tipus	Hossz	Kulcs	
Azonosító	int		✓	Auto_increment
Márkanév	char	40		FK: Foreign Key (idegen kulcs)
Alkohol	float			
Ár	int	40		
Gyártó	varchar			
Kiszerelés	int			
Szeretem-e	bool			
Kiadás	date			

Ha feltöltjük a táblát pár rekorddal, akkor a kilistázáskor hasonló láthatunk:

ID	Marka	Alkohol	Ar	Gyarto	Kiszereles*	Szereteme	Kiadas
1	Duff Ser	4,8	359	Duff Gyár	1	I	1961
2	Gold Water	4,0	250	GW Gyár	4	N	1992

* A Kiszerelés mezőhöz készítünk még egy segédtáblát (kapcsolótáblát):

ID	Azonosító
1	Üveg
2	Doboz
3	Hordó
4	Palack
5	Egyéb

PK: Primary Key
(Elsődleges kulcs)

Figyelem! Ezt a táblát példának hoztuk fel, hogy lássuk, hogyan is épül fel egy MySQL adat-tábla. A készítéséhez szükséges lépéseket a későbbiekben vesszük át, utána példafeladatként mi magunk is készítünk egy adatbázist, benne egy adattáblával. Az SQL parancsok bemutatásánál szintén ezt a Sör nevű táblát hozzuk fel, persze ékezet nélkül (sor), pl.:

```
SELECT      *   FROM    sor;
```

12. PHP MyAdmin

A phpMyAdmin az Apache web-szerverhez kapcsolódó adatbázis-kezelő modul, amely



grafikus kezelőfelülettel rendelkezik. Ezen a felületen könnyebben tudjuk kezelni az adatbázisokat, mint a konzolos felületen. Az Apache-csal együtt a phpMyAdmin is feltelepül így a telepítés után már használatba is vehetjük. Elérése: beírjuk a böngészőnkbe a <http://localhost> címet és a listából kiválasztjuk a `phpmyadmin` linket. Begépeljük a szerverünkhöz tartozó felhasználó nevet és jelszót és ezzel be is léptünk a modulba.

Amielőtt elkezdenénk új adatbázist és adattáblákat létrehozni, tisztáznunk kell, hogy milyen mezőtípusokat is tudunk használni a MySQL-en belül. Aki már használt Access-t, az tudja, hogy az adattáblákban szereplő adatok más-más attribútummal rendelkeznek. Más típusa van az egyes

szám, szöveg és dátumformátumoknak és a lefoglalt tárhely mérete is különbözik. Most nézzük meg mely adattípusok számunkra legfontosabbak az adatbázis-kezeléshez. Ezek egy részét a következő táblázatban bemutatjuk,, a többit megtalálhatjuk az interneten a MySQL referencia-könyvben. Akit részletesebben érdekelnek ezek, látogasson el a <http://www.php.net>, illetve a <http://www.mysql.com> oldalra, ahol a teljes lista megtalálható!

12.1. Legfontosabb adattípusok a MySQL-en belül

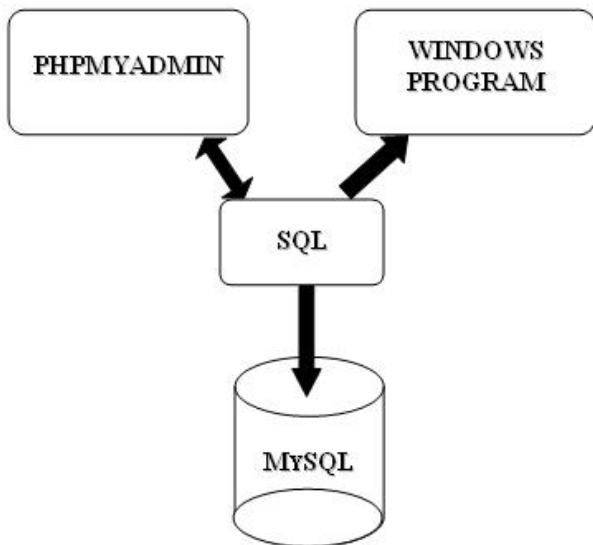
MySQL ADATTÍPUS	TÁRHELY IGÉNYE	ADATTÍPUS JELLEMZŐI
<i>Numerikus adattípusok</i>		
bigint	8 bájt	A legnagyobb egész típus (tárolható érték-tartomány ~ 10 ¹⁹ - - 10 ¹⁹ , előjel nélkül 0-10 ²⁰ -ig).
smallint (előjellel)	2 bájt	- 32 768 és 32 767 közötti egész számok.
smallint (előjel nélkül)	2 bájt	0 és 65 535 közötti egész számok.

tinyint (előjellel)	1 bájt	- 127 és 128 közötti egész számok.
tinyint (előjel nélkül)	1 bájt	0 és 255 közötti egész számok.
int, integer	4 bájt	- 2 147 483 648 és 2 147 483 647 közötti egész számok tárolá- sára alkalmas.
decimal	pontosság + 2 bájt	Lebegőpontos értékek tárolására használjuk, ahol a pontosság is fontos (pl.: 21,8).
<i>Szöveges adattípusok</i>		
char	1 - 255	Rögzített hosszúságú szöveges mező. Ak- kor használjuk legin- kább, mikor egy ka- raktert szeretnénk tárolni.
varchar	1 - 255 bájt	Változó hosszúságú szöveg tárolására alkalmas (maximális hossza 255 karakter lehet). Akkor használ- juk, mikor az attribú- tum értékének hossza változó (ugyanis ez az

		adattípus a beírt szövegnek megfelelően dinamikusan változtatja tárhely igényét, vagyis ténylegesen csak annyi helyet foglal el, amennyi karaktert begépelünk!).
text	0- 65 535 bájt	Szöveges értékek tárolására használatos. A varchar-ral ellentétben akkor is lefoglalja a teljes tárhelyet, ha nem is gépeltünk bele annyi karaktert. (Ezt inkább a nagyobb terjedelmű szövegek tárolására használjuk)
<i>Dátum adattípusok</i>		
date	3 bájt	A Gergely-naptár szerinti dátumok tárolására alkalmas. A tárolható értéktartomány 1000. január 1. – 9999. december 31. között van.

		Formátuma: ÉÉÉÉ- HH-NN Pl.: 2010-12-08
datetime	8 bájt	A Gergely-naptár szerinti dátumok és időpontok tárolására alkalmas. A tárolható értéktartomány 1000. január 1. 00:00:00 – 9999. december 31. 23:59:59 között van. Formátuma: ÉÉÉÉ- HH-NN ÓÓ:PP:MM Pl.: 2011-02-14 22:14:00
time	3 bájt	Időpont tárolására szolgál éjfél és éjfél – 1 másodperc között Formátuma: ÓÓ:PP:MM Pl.: 14:03:13
year	1 bájt	A Gergely naptár szerinti éveket tárolhatunk benne 1900 és 2155 között.

12.2. A MySQL működése

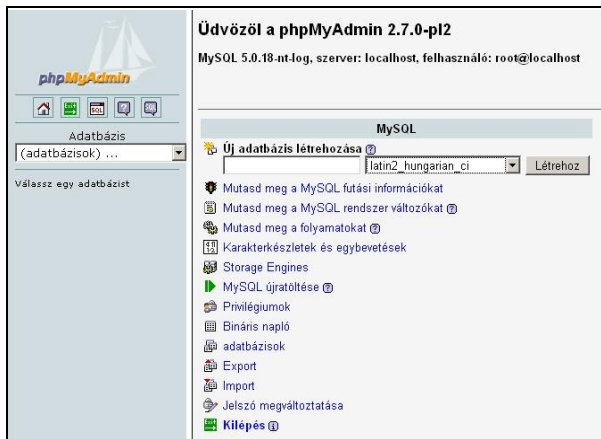


12.3. Új adatbázis létrehozása phpMyAdmin-ban

- Belépünk a localhostba (a böngésző címsorába írjuk be)
- Belépünk azon belül a phpmyadmin mappába
- Begépeljük a felhasználó nevünket és a jelszavunkat, és már bent is vagyunk!

- Bal frame-ben tudunk adatbázist választani
- Jobb frame-en hozhatunk létre új adatbázist (Az Egybevetés lenyíló panelén válasszuk a latin2_hungarian_ci-t). Ugyanitt, jobb oldalt tudunk minden más műveletet elvégezni.

Ha beléptünk, a következő képernyőt láthatjuk:



A bemutatott példákban a 2.7-es verziót használjuk jelenleg az Apache-csal, de az Interneten lévő szervereken általában már ennél újabb verziókkal is találkozhatunk. Azonban nem kell megijedni, a régebbi és újabb verziókkal is könnyen elboldogulunk majd, ha már megtanultuk kezelni.

Jól látszik a képernyőképen, hogy a bal oldalt lévő mezőbe egy nevet gépelve tudunk adatbázist

készíteni, amely így a **Létrehoz** gombra való kattintással el is készül. A példán az egybevetést már beállítottuk **latin2_hungarian_ci**-re. Ez azért fontos, hogy ékezetes betűket is tudjunk használni. Ám ennyi nem biztos, hogy elég is lesz. Ha a szerverünkön nincs megfelelően beállítva a karakterkódolás, akkor könnyen **?**-et kaphatunk vissza például az **ő** és **ű** betűk helyett. Sajnos az ingyenes tárhelyeken ez elég sokszor előfordul (persze nem mindenhol), de erre is nézünk később egy kis praktikát! Ha begépeltük a nevet és létrehoztunk egy új adatbázist, akkor a bal oldalt látható **Adatbázis** felirat alatt lévő lenyíló listán tudjuk kiválasztani az elkészített adatbázist, hogy abba adattáblákat készíthessünk. Meg kell jegyezni, ha ingyenes tárhelyen szeretnénk adatbázist kezelni, azt általában előbb aktiválni kell az adott szolgáltató honlapján. Ezek után hozhatunk létre új táblákat. Adatbázisból csak egyet kezelhetünk ekkor (ami általában ugyanazt a nevet viseli, mint az ingyenes tárhelyünk domain neve, amit mi adtunk meg). Ez kicsit zavaró lehet, főleg, ha sok adattáblát akarunk használni, de ugye az ingyenességnek is ára van...

Ezek a kezdő lépések új adatbázis létrehozásakor. A következőkben egy példán keresztül készítünk egy adatbázist, megtanulunk táblát létrehozni, beállítani a mezők attribútumait, majd rekordokat feltölteni. Fontos, hogy a példa szerint készítjük el az adatbázist, mert erre alapozva a későbbi-

ekben fogunk még példaprogramokat is felhozni! És most lássunk is neki!

12.4. Adatbázis, adattábla készítése

Az előbbi ábrán látható képernyőtől elindulva végigvesszük egy adatbázis és azon belül egy adattábla készítését. Egy nyilvántartó adatbázist készítünk, amiben szerepelni fog az egyén neve, anyja neve, születési helye, születési dátuma. A példa kedvéért ennyi mező nekünk egyelőre elég is lesz. Most nézzük a lépéseket!


1., Hozzunk létre egy új adatbázist a képernyő jobb oldalán az előző fejezetben leírtak szerint! Az adatbázisunk neve **peldafeladat** legyen (ha nagyon akarjunk, természetesen más nevet is adhatunk neki, de mi a példák során ezt a nevet fogjuk használni). Az egybevetés lenyíló panelén válasszuk ki a **latin2_hungarian_ci**-t. Kattintsunk a **Létrehoz** – ra!



2., Miután létrehoztuk automatikusan felkínálja a php MyAdmin új tábla létrehozását az újonnan készített adatbázisunkba. Legyen a tábla neve **nyilvantartas**, a mezők száma pedig 5!

Kattintsunk a **Végrehajt** gombra!

Nincs tábla az adatbázisban.

 Új tábla létrehozása a(z) **peldafeladat** adatbázisban

Név: Mezők száma:

Ezután már meg is nyílik az üres adattábla, ahol el is kezdhetjük a mezők neveinek és attribútumainak beállítását!

3., Az első mezőnk az **id** lesz. A neve az angol **identity** szóból származik, (személy)azonosságot jelent és általában ez a szokásos névhasználat kulcsmezők esetében. Ez a mező fogja a későbbiekben beazonosítani az aktuális rekordot. Ennél a mezőnél be kell állítanunk az **EXTRA** feliratú oszlopban, hogy **auto_increment**. Ez annyit tesz, hogy minden egyes új rekord beszúrásakor ez a mező magától növeli az értékét eggyel (mert ugyebár ez **INT típusú** lesz, az első rekordnál az id 1 lesz). Mivel magától növekszik az értéke, nem szabad új rekord feltöltésekor beleírunk semmit! Ez a mező lesz a kulcs mezőnk is egyben, ezért az **EXTRA** oszlop melletti, **kulcs ikonnal jelzett oszlop checkbox-át** jelöljük be. Ebben az oszlopban tudjuk ezáltal az elsődleges kulcsot beállítani. Az id ezzel kész is! Nézzük a többi mezőt, hogyan is kell kitöltenünk!

localhost ▾ példafeladat ▾ nyilvantartas									
Mező	Típus	Hossz	Érték	Egybevetés	Tulajdonságok	Null	Alapértelmezett	Extra	
id	BIT					not null		auto_increment	
nev	VARCHAR	30		latin2_hungarian_ci		not null			
anyjaneve	VARCHAR	30		latin2_hungarian_ci		not null			
szhely	VARCHAR	50		latin2_hungarian_ci		not null			
szdatum	DATE					not null			

A 2. mező neve: nev

Típusa: VARCHAR Hossza: 30

A 3. mező neve: anyjaneve

Típusa: VARCHAR Hossza: 30

A 4. mező neve: szhely

Típusa VARCHAR Hossza: 50

Az 5. mező neve: szdatum

Típusa: DATE Hossz: nem kell beírni semmit!

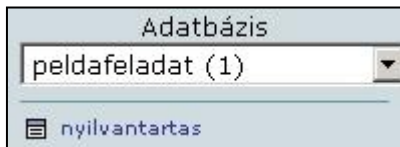
A 3 szöveges típusú mezőnél állítsuk be az egybevetést **latin2_hungarian_ci** – re!

Kattintsunk a **Ment** gombra és már el is készült az új adattáblánk, feltölthetjük rekordokkal!

Ha mindent jól csináltunk, a phpMyAdmin következő táblastruktúrát fogja mutatni:

Mező	Típus	Egybevetés	Tulajdonságok	Null	Alapértelmezett	Extra	Parancs
<input type="checkbox"/> id	int(1)			Nem		auto_increment	     
<input type="checkbox"/> nev	varchar(30)	latin2_hungarian_ci		Nem			     
<input type="checkbox"/> anyjaneve	varchar(30)	latin2_hungarian_ci		Nem			     
<input type="checkbox"/> szhely	varchar(50)	latin2_hungarian_ci		Nem			     
<input type="checkbox"/> szdatum	date			Nem			     

4., Miután elkészültünk, a phpMyAdmin bal felén, az Adattábláknál



kiválasztásra került a **peldafeladat** adatbázis és alatta megjelent a **nyilvantartas** tábla linkje (bal

oldali ábra). Ahhoz, hogy a táblával műveletet végezzünk, a fenti eszköztárat kell használnunk, ami a jobb frame tetején van (ld. ábra).



A **Beszúr** földre kattintva tudunk rekordot beszúrni a phpMyAdmin-on keresztül. A többi fül, ami nekünk fontos lehet:

- **Tartalom:** a kiválasztott tábla tartalmát mutatja (vagyis a rekordokat listázza ki)
- **Struktúra:** az adott tábla felépítését mutatja (a 3. pont alján látható ábrához hasonlóan)
- **SQL:** szöveges SQL parancsokat tudunk ide begépelni és lefuttatni, mintha csak a konzol felületet használnánk.
- **Keresés:** megadhatunk keresési feltételeket akár mezőnként is, ha az adattáblán belül keresünk valamilyen rekordot.
- **Export, Import:** adatbázismentést (DUMP-olást) tudunk végezni az elsővel, a másodikkal adatbázist beimportálni DUMP fájl segítségével (ezekről később)
- **Tevékenységek:** táblákat helyezhetünk, másolhatunk át egyik adatbázisból a másikba, ellenőrizhetjük, javíthatjuk a táblákat stb.
- **Kiürít:** kiüríti az adattáblát (kitörli a rekordokat, de a tábla megmarad!)

- **Eldob:** eldobja az adattáblát (véglegesen törli a táblát is, minden adat elveszik!!!)

Tehát kattintsunk a **Beszúr** fülre, ami után megjelenik egy beviteli űrlap, ahol egyszerre akár 2 rekordot is fel tudunk tölteni egyszerre. Töltsük fel a táblánkat a következő rekordokkal:

Név	Anyja neve	Születési helye	Születési dátuma
Nagy János	Kovács Mária	Budapest	1989-08-05
Kis Jolán	Szabó Anna	Kaposvár	1962-06-03
Kovács Gábor	Szűcs Tímea	Budapest	1976-10-12
Antal Iván	Takács Éva	Szeged	1992-02-05
Varga Béla	Nagy Eszter	Salgótarján	1985-06-25
Kozma Miklós	Pásztor Jolán	Győr	1961-05-03
Dobó István	Csikós Anita	Eger	1952-12-20
Szabó Elemér	Juhász Ildikó	Pécs	1987-03-13

Ha fel akarjuk vinni az űrlapot, kattintsunk a **Végrehajt** gombra! Ezzel a begépelt adatok bekerülnek egy új rekordként az adattáblába.

Figyeljünk arra, hogy az id mezőbe ne írjunk semmit, hagyjuk üresen! A Funkció lenyíló panel sem kell nekünk, ahhoz ne nyúljunk! A dátum beírása a fenti táblázatban megjelenő dátumformával történik: ÉÉÉÉ-HH-NN.

A következő fejezetekben megtanuljuk azokat az SQL parancsokat, melyek elengedhetetlenek adatbázis kezelő és PHP ismereteinkhez egyaránt. Megtanuljuk, hogyan is kell egy adatbázist összekötni PHP oldalakkal, hogy az adattáblák tartalmát egy weboldalba listázva megtekinthessük.

13. SQL parancsok

Az adatbázisokkal több módon is végezhetünk műveleteket. Leginkább a grafikus felületet használata javasolt (a PHP MyADMIN-t), de nagy előny az is, ha valaki tud adatbázist kezelni konzolos felületen is (mert a következő parancsok függetlenek az adatbázis-kezelő rendszertől, így egyaránt működnek a MySQL-ben és egyéb rendszerekben is).

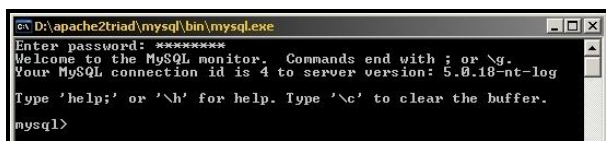
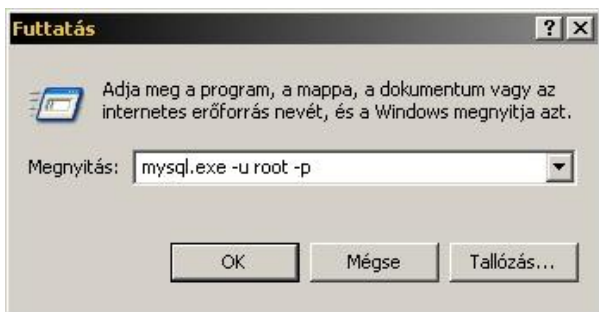
13.1. Belépés a DOS-os (konzolos) felületre:

A Start menü → Futtatás sorával előhívható ablakba gépeljük be a következőket:

mysql.exe -u felhasználónevünk -p

Ezzel be tudunk lépni a konzolos felületre. A **-u** kapcsoló után meg kell adnunk a felhasználónevünket a **-p** kapcsolóval pedig jelezzük, hogy a jelszavunkat is meg akarjuk adni a belépéshez.

Ha megjelent a képernyőnkön a **mysql>** prompt, akkor már neki is kezdhünk a parancsok gépeléséhez.



- | | |
|------------------------|--------------------------------|
| show databases; | → adatbázisok mutatása |
| show tables; | → táblák mutatása |
| use proba; | → próba adatbázis használata |
| describe sor; | → sör tábla mutatása (leírása) |

Figyeljünk oda a parancsok után található ; (pontosvesszőkre), mert ezek elhagyása esetén a parancs nem hajtódik végre, hibaüzenetet kaphatunk!

13.2. Lekérdezés SQL parancsokkal:

1. **SELECT * FROM sor;**
- ↓ ↓ ↓ ↓
- lekér Joker honnan a táblánk
- karakter neve
- (minden
- mezőt
- kiír)

A csillagot felválthatjuk a mezők nevével, vesszővel elválasztva. Ha nem vagyunk biztosak a mezők nevével, a lekérdezés előtt írassuk ki egy describe-bal a táblát.

2. **SELECT [mezonev1, mezonev2, ...] FROM tabalanev;**

Csak adott mezők értékeit kérjük le az adattáblából.

3. **SELECT [mezonev] AS "átnevezett mezonev";**

Átnevezés a lekérdezésben. Nem az adattáblában! (A [] jeleket nem kell begépelni a lekérdezésbe.)

4. Aggregáló függvények

a., Megszámlálás:

SELECT COUNT(*) FROM sor;

Jelen esetben a sör táblában számolja meg a rekordokat. Ennél a függvénynél is használható az „AS”-es átnevezés (ugyanúgy az ezt követőekben is). Az aggregáló függvényeket MINDIG a SELECT után írjuk be.

b., Szummázás:

SELECT SUM(mezonev) FROM tablanev;

Összeadja a mezőhöz tartozó adatokat (csak numerikus adat összeadására képes).

c., Minimum kiválasztás:

SELECT MIN(mezonev) FROM tablanev;

A mező legkisebb értékét adja vissza.

d., Maximum kiválasztása:

SELECT MAX(mezonev) FROM tablanev;

A mező legnagyobb értékét adja vissza.

e., Átlag számítás:

SELECT AVG(mezonev) FROM tablanev;

5. SELECT záradékok – Feltétel

WHERE kulcsszó után történik a feltétel megadása.

A feltétel után <, >, <=, >=, =, != (nem egyenlő) jelek szerepelhetnek.

Pl.: ...WHERE ar != 120;

- „**És**” kulcsszó a feltételek összefűzésére: **AND**
Pl.: ...WHERE ar < 200 AND alkohol > 4.5;
- „**Vagy**” kulcsszó: vagy az egyik, vagy a másik feltétel teljesül: **OR**;
- „**Közt**” kulcsszó: „-tól -ig” határ: **BETWEEN**
Pl.: ...WHERE ar BETWEEN 180 AND 230;
- **IN** kulcsszó: ezután zárójelek között egy feltétel megadása
- **NOT IN** kulcsszó: az IN ellentéte, komplementere
- **LIKE** kulcsszó: LIKE után meg kell adni mi legyen a keresett feltétel

LIKE-nál a joker karakter a % (**százalék**) jel.

Pl.: ...WHERE mezonev LIKE "%a%"; → olyan szöveget keres, amiben van a betű. Ha az első % jelet nem íránk, akkor csak "a" betűvel kezdődő szavakra keres rá!

%a% → szótöredék, több betűre, karakterre keres
_ (**aláhúzás karakter**) → egy karakterre használjuk

Pl.: ...WHERE markanev LIKE "_m%"; → Olyan márkánévre keres rá, amiben az első betű bármi lehet, a 2. csak „m” betű, azután pedig bármilyen karakter lehet.

- **ORDER BY** feltétel: rendezés valamilyen feltétel szerint növekvő sorrendben (ASC)
- **ORDER BY** feltétel **DESC**: ugyanaz, mint az előző, csak csökkenő sorrendben
Másodlagos rendezési szempont is megadható.
Pl.: ... ORDER BY mezonev1 DESC, mezonev2
Szöveges rendezés esetén ABC sorrend van, vesszővel megadhatunk több szempontot is.

A numerikus adatokat tudjuk változtatni a lekérdezésben a +, -, *, / műveleti jelek segítségével.

EMPTY SET → üzenet, amit a MySQL-től kapunk: nincs ilyen, nincs találat a feltételre

13.3. Kapcsolat létrehozása MySQL-ben

Több tábla összekapcsolásának szintaktikája (a kiválasztott mezőkkel, hogy kapcsolatot tudjunk létrehozni):

1. **SELECT** tablanev1.mezonev1,
tablanev2.mezonev2 **FROM** tablanev1, tablanev2

2. Tényleges kapcsolat: az 1. pontban leírt lekérdezéshez hozzáfűzzük a következőt:

... **WHERE** tablanev1.mezonev1 =
tablanev2.azonosito

Az azonosító (ID) segítségével párosíthatjuk össze a táblát, mivel ez volt a kulcsunk (Az "SQL adatbázisok" című fejezetben példaként felhozott *sör* táblát ezzel a módszerrel tudjuk összekapcsolni a *Kiszereles* mezőhöz tartozó segédtablával.).

3. Ha párosította az idegen és elsődleges kulcsokat és létrejött a kapcsolat, **AND** kulcsszóval újabb lekérdezéseket készíthetünk (mivel a WHERE-hez fűzünk hozzá az AND-del).

14. DML – adatmódosító parancsok

14.1. Mi is az a DML?

A DML jelentése Data Manipulate Language, azaz adatmódosító nyelv az SQL-en belül.

14.2. 1. Új rekord beszúrása

INSERT INTO tablanev (zárójelben itt megadjuk, milyen mezőknek akarunk értéket adni) **VALUES** (zárójelben itt megadjuk azoknak a mezőknek a felvitelét, amiket az előbb felsoroltunk – ékezetes betűs char-nál " " jel kell);

14.3. 2. Rekord módosítása:

UPDATE sor **SET** ar = 150; → A sör tábla minden árát növeli 150-nel.

UPDATE sor **SET** ar =160 **WHERE** id = 2; → Ennél csak annak a rekordnak változtatja meg az

árát, aminek az azonosítója 2 (WHERE záradék után kell megadni).

Erre lehet alkalmazni több változtatást is, ezeket vesszővel választjuk el.

Pl.: **UPDATE** sor **SET** alkohol = 4.7, ar = 120, markanev = "SPATEN" **WHERE** id = 2;

Az UPDATE parancsot WHERE záradék nélkül meggondolatlanul soha ne használjuk!

14.4. 3. Törlés – DELETE:

DELETE FROM tablanev; → Minden adatot töröl!!!

DELETE FROM sor **WHERE** id = 3; → Kitörli azt a rekordot, aminek az azonosítója a 3-as.

WHERE után megadjuk feltételként, hogy mit szeretnénk törölni:

Pl.: **DELETE FROM** sor **WHERE** alkohol < 4.5;
→ Törli azt a rekordot a sör táblából, aminek az alkoholtartalma kisebb, mint 4,5.

Egy rosszul kiadott **DELETE** paranccsal könnyen el tudjuk rontani – akár törölni is az egész – eddig elkészített adatbázisunkat, ezért érdemes időnként biztonsági mentést készíteni!

DUMP-olás: biztonsági mentés, adatbázis visszaállítása (**INSERT INTO**-s **SQL** parancsokkal vissza tudjuk állítani az egész adatbázisunkat).

Fontos! A DELETE parancsot WHERE záradék nélkül ne használjuk!

15. PHP és MySQL összekapcsolása

1. Kapcsolat megadása a PHP blokkon belül:

```
$kapcsolat = mysql_connect  
("LOCALHOST","ROOT","KOZI1987");
```

↓ ↓ ↓
adatbázis helye felhasználónév jelszó
(IP cím is lehet)

\$kapcsolat → ez az ún. connection string

A felső sor után OR-ral hozzáfűzhetjük a hiba okát és egy hibaüzenetet is:

```
OR DIE (print "HIBA!".mysql_error());
```

A fenti példa működése: csatlakozás a **localhost**-hoz, **root** felhasználóval és **kozi1987** jelszóval. Ha nem sikeres, akkor a PHP program véget ér, kiírja, hogy HIBA és a hiba okát lekéri a MySQL szervertől.

Ha sikeres a csatlakozás a \$kapcsolat változó tárolja a kapcsolatot.

2. Adatbázis kiválasztása:

```
mysql_select_db("proba", $kapcsolat);
```

Itt is lehet az OR-ral hozzáfűzni a hibaüzenetet.

A fenti paranccsal kiválasztjuk a \$kapcsolatból a használni kívánt adatbázist, ha nincs ilyen, akkor hibával jelezzük.

3. Parancs meghatározása:

```
pl.: $parancs = "SELECT * FROM tablanev";
```

4. Parancs kiadása:

A \$parancs szöveges változóban lévő parancsot futtatjuk le az adatbázison. A \$kapcsolat elhagyható, ilyenkor az utoljára megnyitott kapcsolaton fog dolgozni. Ha nem jó a parancs hibát jelez vissza!

```
$eredmeny = mysql_query($parancs, $kapcsolat);
```

A visszaadott eredményt a \$eredmeny változó tartalmazza.

5. Feldolgozás WHILE ciklussal,

```
mysql_fetch_array
```

(A feldolgozás és a parancs megértésére nézünk konkrét példát a későbbiekben.)

6. majd az adatbázis lezárása:

Adatbázis lezárása: `mysql_close($kapcsolat);`

A fenti lépések FORM használatakor (űrlapok esetében) a következőképpen változnak:

1. Kapcsolat kialakítása
2. Adatbázis kiválasztása
3. SQL parancs megadása
4. Futtatás
5. Listázás
6. Átirányítás bezárása

15.1. Példaprogram PHP és MySQL összekapcsolására

Most megnézzük egy viszonylag egyszerű példán keresztül, hogyan is történik egy adatbázis és egy PHP oldal összekapcsolása. Ennek lényege, hogy a PHP segítségével minden adatbázisban tárolt adatot képesek vagyunk a képernyőre íratni weblap formájában. Erre épülnek a modern dinamikus honlapokon például a hírek, fórumok, vendégkönyvek, chatbox-ok és minden más olyan építő elem, melyet adatbázisban tárolunk és admin panelről tudunk szerkeszteni: felvenni, módosítani vagy törölni.

A mostani feladatban, egy PHP oldal formájában, kilistázzuk a korábban már elkészített **peldafeladat** adatbázisunk **nyilvantartas** tábláját, méghozzá név szerint növekvő sorrendben. A parancsokat már átnéztük ehhez és a folyamatot is ismertettük az előző pontban, most pedig megnézzük a forráskódon keresztül a készítés folyamatát! Nagyon oda kell figyelniünk, hogy hol nyitunk és hol zárunk PHP kódblokkot és persze a szintaktikára is ügyelniünk kell! Minden lépést leírtuk a megfelelő sorhoz a már megszokott megjegyzés formában! Most pedig lássuk a kódot (<body> tag-tól a </body> tag-ig):

<body>

```
<p align="center"><h2>Nyilvantartási jegyzék</h2></p>
```

```
<p>
```

```
<!-- Készítünk egy táblázatot, amely a PHP kódblokkon kívül szerepel és beleírjuk a nyilvantarto nevű adattábla mezőinek nevét. -->
```

```
<table width="600" border="0" cellpadding="0" cellspacing="0">
```

```
<tr>
```

```
<td width="150"><strong>Név</strong></td>
```

```
<td width="150"><strong>Anyja neve
```

```
</strong></td>
```

```
<td width="150"><strong>Születési hely
```

```
</strong></td>
```

```
<td width="150"><strong>Születési dátum
</strong></td>
</tr>
</table>
</p>
<p>
```

```
<?php
$kapcsolat = mysql_connect("localhost",
"root", "kozi1987") or die (print
"HIBA!".mysql_error());
/* 1. Csatlakozunk a helyi gépen (localhost) lévő
szerver adatbázisához ROOT felhasználónévvel és
KOZI1987 jelszóval (ha mi más felhasználót vagy
jelszót használunk, természetesen azt kell begépel-
nünk). Ha nem sikeres, akkor a PHP program véget
ér, kiírja, hogy HIBA! és a hiba okát lekéri a mysql
szervertől. Ha sikeres a $kapcsolat változó tárolja a
kapcsolatot. */
```

```
mysql_select_db("peldafeladat", $kapcsolat);
//2. Kiválasztjuk a peldafeladat adatbázist, a $kap-
csolat connection string-ből.
```

```
mysql_query("SET NAMES latin2");
/*Ezt a sort nem kötelező kiraknunk, ez egy kis
parktika, hogy ha a webszerver (ahova az oldalunkat
feltöltjük) nem engedi a karakterkódolás beállítását,
akkor ez a kis függvény automatikusan latin2-es kó-
dolásúra teszi a PHP oldalt, így az ékezetes ő és ú
betűk is gond nélkül megjelennek!*/
```

```
$parancs = "SELECT * FROM nyilvantartas
ORDER BY nev";
```


//3. A \$parancs szöveges változóban lévő parancsot futtatjuk le az adatbázison. A \$kapcsolat elhagyható, ilyenkor az utoljára megnyitott kapcsolaton fog dolgozni. A mostani esetben lekérdezzük a nyilvantartas tábla összes mezőjét és a 'név' mező szerint növekvő sorrendbe rakjuk.

\$eredmeny = mysql_query(\$parancs, \$kapcsolat);

//4. A \$eredmeny változóban eltároljuk, hogy a \$parancs-ot hajtsa végre a PHP fordító a \$kapcsolatban megjelölt adatbázison.

while (\$sor = mysql_fetch_array(\$eredmeny))

/* 5. Egy WHILE ciklussal feldolgozzuk a rekordokat. A visszatért eredmény egy 2 dimenziós táblázat lesz. Ezt soronként dolgozzuk fel a mysql_fetch_array parancssal, amely egy sorát adja a \$eredmenynek a \$sor nevű változóba (amely nevéből adódóan az adattábla egy sorára, tehát egy rekordjára utal). Addig íratjuk ki a sorok tartalmát, ameddig minden soron végig nem mentünk! (Ezért is kell a WHILE ciklust alkalmazni.)

A soron belül a táblában lévő mezőnévvel hivatkozhatunk az egyes adatokra (\$sor[anyjaneve]) */

{ //ezek után lezárjuk az 1. PHP blokkot, mivel most HTML rész következik!

?>

<!-- A WHILE ciklus ciklusmagjában HTML kód van, mégpedig egy táblázaté. A táblázat egy-egy oszlopába kiíratjuk az adattábla minden mezőjét. A ciklusmag által minden rekord új sorba kerül és a lista az összes rekordot tartalmazni fogja. -->

```

<table width="600" border="0" cellpadding="1"
cellspacing="1">
  <tr>
    <td width="150"><?= $sor[nev]; ?></td>
    <!-- Mint már korábban említettük ezt a szintaktikát is
    használhatjuk kiíratásra a print függvény helyett.
    Azért is használjuk ezt, mert ezekben rövid PHP
    blokkban kiíratást végzünk és utána le is zárjuk a
    kódblokkot, hogy visszatérjünk a HTML részbe! -->
    <td width="150"><?= $sor[anyjaneve]; ?></td>
    <td width="150"><?= $sor[szhely]; ?></td>
    <td width="150"><?= $sor[szdatum]; ?></td>
  </tr>
</table>

```

```

<?php
} //Ezt a PHP blokkot azért készítettük, hogy
lezárja a WHILE ciklust az első kapcsoszárojel pár-
jával. Ez nagyon fontos, soha ne felejtsük el kiten-
ni!

```

```

mysql_close($kapcsolat);
//6. Az adatbázis kapcsolatot is lezárjuk!
?>
</p>
</body>

```

Ezzel el is készült a PHP oldalunk! Ha mindent jól gépeltünk és megnézzük a böngészőnkben az oldalt, láthatjuk, hogy a programunk pontosan azokat a rekordokat íratta ki a képernyőre, amelyeket mi egy korábbi feladat során felvittünk:

Cím  http://localhost/oktatoanyag/php_mysql_kapcsolata/nyilvantartas.php			
Nyilvántartási jegyzék			
Név	Anyja neve	Születési hely	Születési dátum
Antal István	Takács Éva	Szeged	1992-02-05
Dobó István	Csikós Anita	Eger	1952-12-20
Kis Jolán	Szabó Anna	Kaposvár	1962-06-03
Kovács Gábor	Szűcs Tímea	Budapest	1976-10-12
Kozma Miklós	Pásztor Jolán	Győr	1961-05-03
Nagy János	Kovács Mária	Budapest	1989-08-05
Szabó Elemér	Juhász Ildikó	Pécs	1987-03-13
Varga Béla	Nagy Eszter	Salgótarján	1985-06-25

16. CMS rendszerek

CMS: Content Managment System (Tartalomkezelő rendszer).

Általában kevés szaktudást igénylő internetes alkalmazás, mely lehetővé teszi, hogy szinte bárki összetett weboldalt birtokoljon. Az alkalmazás jellegéből adódóan a weboldal rövid idő alatt összeáll, programozási tudást általában nem, vagy egészen keveset igényel.

A CMS olyan internetes alkalmazás, amely által bárki összeállíthat egy weboldalt, képeket, tartalmakat vagy akár navigációt változtathat akkor is, ha nem vagy alig rendelkezik technikai/programozási ismeretekkel. A tartalom (content) a kinézettől, design-tól (layout-tól) elkülönítve, webes adatbázisban van tárolva.

Layout = kinézet

Content = tartalom

A CMS alkalmazható többek között web-áruházaknál, portáloknál, web-magazin szerkesztésénél illetve más egyéb alkalmazásnál.

A tartalmak szerkesztéséhez egy kezelőfelület (admin felület) áll rendelkezésre. Az adatbevitel szöveg- és feltöltő mezőkön (űrlapok, checkbox-ok, radiobutton-ok stb.) keresztül történik, amelyek segítségével a CMS feltölti a megváltoztatott fájlokat a webszerverre. A tartalmak azonnal láthatók online, az oldal frissítése után.

A CMS-eket a praktikusságuk és könnyen kezelhetőségük mellett azért (is) érdemes megemlíteni, mert a lentebb felsorolt tartalomkezelő rendszerek mindegyike nyílt forráskódú, ingyenesen letölthető és használható, PHP-ben íródott, valamint MySQL adatbázist használ. Így nekünk, jelen ismereteinkkel sem fog különösebb gondot okozni ezek telepítése és alapvető konfigurálása

16.1. A CMS-ek előnyei

- **Komoly szaktudást nem igényel**
- **Adatbázisban tárolt tartalom**

A feltöltött tartalom adatbázisban tárolódik, így hordozhatóvá és több eszköz (böngésző, mobiltelefon, PDA, nyomtatás) számára is könnyen átalakíthatóvá, formázhatóbbá válik.

- **Dinamikus tartalom**

Külön modulok segítségével akár vendégkönyveket, híreket, fórumot, szavazásokat, látogatottsági statisztikákat, az egész honlapra kiterjedő keresést vagy webshop-ot integrálhatunk honlapunkba. Ezen fajta kiegészítők száma rengeteg, bőséges választékból válogathatunk a CMS-eknél.

- **Naprakész frissítés**

A tartalom módosításához nincs szükség külön cégre, programozókra, extra kiadásokra – az oldal frissítését saját erőforrásból végzi el!

- Nem lehet beleavatkozni a weboldal kinézetébe. Erre csak akkor van lehetőség, ha kellő szakértelemmel kicseréljük, felülírjuk a weboldalt alkotó képeket, grafikákat illetve átírjuk az adott témához tartozó CSS stílusfájlokat a megfelelő helyen és módon.
- Másik megoldás, hogy külső fejlesztőtől beszerzünk, letöltünk különböző oldaltémákat (themes). Sokféle, rendszerezett, különböző kategóriákba besorolt, ingyenes témát találhatunk a lent felsorolt CMS-ek hivatalos honlapjain, illetve a velük foglalkozó, egyéb (sok esetben magyar nyelvű) weblapokon is! Így könnyedén kereshetünk weboldalunk tartalmához illeszkedő, számunkra megfelelő design, kinézetet, amit aztán pár kattintással azonnal telepíthe-

tünk is, és ami ezt követően azonnal alkalmazható az oldal témájaként.

16.2. Magyar nyelven is elérhető CMS rendszerek

- PHP-Nuke (<http://phpnuke.org>)
- e107 (<http://e107.org>)
- Drupal (<http://drupal.org>)
- Joomla (<http://www.joomla.org>)
- GuppyY (<http://www.freeguppy.org>)

Ezen CMS-ek nagy része megtalálható magyar nyelven is, így használatuk még könnyebb és kényelmesebb. Általában találhatunk olyan magyar nyelvű weboldalakat is, ahol az adott tartalomkezelő rendszerrel foglalkoznak és az aktuális frissítésekről, hírekről, újabb bővítményekről és témákról kezdve a telepítési útmutatón át a hibajavításokig, sok mindent megtalálhatunk rajtuk, amik segíthetnek a munkánkban. Érdekes mindegyik rendszerhez – persze a hivatalos mellett – egy ilyen honlapot is felkeresni az érdeklődő webfejlesztők, webprogramozók és érdeklődők számára.

Egy példa: PHP NUKE

Egy CMS rendszer. Integrálja magában az összes eszközt, amelyet információs honlap vagy

portál létrehozása (széles értelemben véve) alkalmazunk.

Példa a felhasználására:

- Intranet (belső) oldalak
- E-kereskedelmi rendszerek
- Vállalati portálok
- Nyilvános ügynökségek
- Újságíró cégek
- Online cégek
- Információs honlapok
- E-tanulás rendszerek
- Hobby weboldalak

16.3. CMS-ek telepítése általánosságban

Letöltjük a CMS telepítő csomagját, ami általában tömörített fájlban található. Ajánlatos az adott CMS hivatalos honlapjáról letölteni, ahol a legújabb verziókat, valamint a magyar ékezetes betűket is tartalmazó verziókat is megtalálhatjuk. Ezután:

1. A csomagban található, telepítéshez szükséges fájlokat kicsomagoljuk és mindegyiküket feltöltjük honlapunk FTP szerverére.
2. Adatbázis konfigurálása (ez két módon történhet):

- a. **Setup**-on keresztül (általában ez az **install.php**-t kell futtatnunk, ennek elérési útját kell begépelnünk a böngészőnk címsorába).

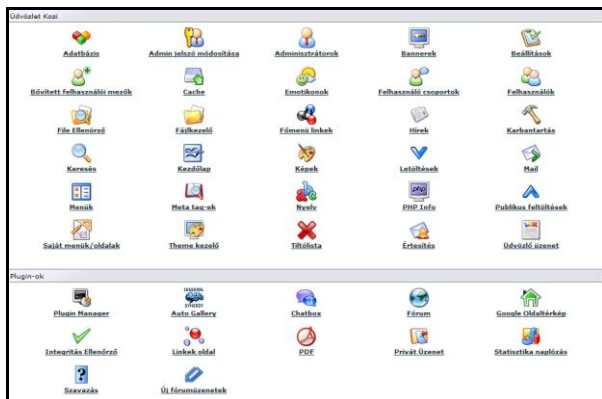
Fontos! A sikeres telepítést követően azonnal töröljük az `install.php` fájlt (erre általában külön figyelmeztetést is kapunk a telepítéskor)!

- b. vagy **.ini** fájlon keresztül
ekkor adatbázis importálásra (DUMP) van szükségünk
3. Admin fiók kialakítása, admin felhasználónévünk és jelszavunk megadása, amelyekkel később belépve, minden módosításra jogosultak leszünk a honlapon és annak admin felületén (ezeket az adatokat soha ne tegyük közzé!).
4. Tartalom, forma, blokkok kialakítása. A honlap témájának, kinézetének beállítása a meglévő sémákból (további témákat tölthetünk le az adott CMS honlapjáról illetve az ezekkel foglalkozó weboldalakról), plugin-ek, kiegészítő funkciók bekapcsolása/telepítése, személyre szabás, tartalommal való feltöltés

A telepítés során a CMS-ek egyértelműen megadják, hogy melyik mezőbe mit kell kitöltenünk, így a kezdő PHP programozók is könnyűszerrel

telepíthetik őket pár kattintással, minimális programozási ismeretekkel is.

A telepítés után egy **grafikus admin felületen** keresztül könnyen szerkeszthető, látványos külalakgal és hasznos kiegészítőkkel rendelkező honlapot kapunk. Mindenképp érdemes őket kipróbálni!



Az ábrán az e107 tartalomkezelő rendszer grafikus admin felületének egy részlete látható. Jó példa arra, hogy a CMS-eknél milyen átlátható kezelőfelületen tudjuk beállítani telepített webpünk külső-belső tulajdonságait, illetve milyen blokkok kialakítására van lehetőségünk.

17. Megvalósítások PHP-vel

17.1. Képgaléria PHP-ben 1.0

Most a megismert PHP nyelv segítségével készítünk egy képgalériát. A képeinkből, amiket be akarunk illeszteni a galériánkba, először is miniatűröket (thumbnail) készítünk, amiket (egy táblázatba rendezve) beszúrunk egy weboldalba. Ez lesz a példánkban a **galeria.php**.

Helyezzük el a nagyméretű képeket a **nagy**, a thumbnail képeket a **mini** mappába. Az oldal, amelyikre a **galeria.php**-ről ugrunk (tehát, amelyiken megjelennek a nagy képek, lesz a **kep.php**.

Az alábbi módon szűrjük be a PHP programunkba a megjelenítendő képeket:

```

<head>
<title>PHP képgaléria</title>
</head>
<body>
<?php
$foto=$_GET['foto'].'.jpg';           //a kép
meghívása a galeria.php-ből GET-tel, tehát ebben
a változóban tároljuk a kép nevét. Ezek után még
hozzá kell fűznünk az átvett paraméterhez a képfájl
kiterjesztését (ami esetünkben, minden képnél JPG
lesz) és ezáltal letároltuk a $foto változóban a beil-
lesztéshez szükséges összes adatot!
?>
<p align="center">PHP képgaléria</p>
<table width="100" border="3" align="center">
  <tr>
    <td><b>img src="nagy/<?= $foto ?>"
border="0"></td> //aktuális kép beszúrása
  </tr>
</table>
<p align="center"><strong><a
href="galéria.php">Vissza</a></strong></p>
//beszúrunk egy visszalépő linket, amely a
galéria.php-re mutat, hogy új képet nyithassunk
meg!
</body>
</html>
```

17.2. Képgaléria PHP-ben 2.0

Az előző példában elkészítettük a képgalériánkat. Mint már említettük, ez úgy működik, hogy a `galeria.php`-n lévő kisképekre kattintva átugrunk a `kep.php`-ra, ahol megtekinthetjük a képet nagyobb méretben, a fent leírtak alapján. Ezek után visszaléphetünk a kisképek oldalára. Ám kényelmesebb lenne, ha nem kellene mindig visszalépkednünk a thumbnail képeinkhez, hanem mondjuk hivatkozások segítségével a `kep.php`-n is tudnánk előre és vissza is lapozni, a nagy képek között. Most ezt a megoldást mutatjuk be, kiegészítjük tehát az eddig elkészült képgalériánk forráskódját!

[Ide jönnek a `html`, `head`, `title`, `body` tag-ek]

```
<?php
```

```
$foto=$_GET['foto'].'.jpg';
```

```
?>
```

```
<p align="center">PHP képgaléria</p>
```

```
<table width="100" border="3" align="center">
```

```
<tr>
```

```
<td></td>
```

```
</tr>
```

```
</table><!-- Eddig a pontig a forráskód ugyanaz, csak most kezdünk egy új PHP blokkot! -->
```

```
<?php
```

```
//Deklarálunk két változót, melyeket majd az „előző” illetve „következő” linkekhez használunk majd fel. Ezek feladata az, hogy az átvett foto paraméter
```

értékéből kivonjanak illetve hozzáadjanak egyet (így mutatva a PHP programnak, hogy melyik az a kép, amelyik az aktuális megjelenített előtt, illetve után van!). Nagyon fontos, hogy csak akkor tudjuk lapozhatóvá tenni a galériánkat ezzel a módszerrel, ha (egész) számértékeket adunk a képeink nevének (string-es névből nehéz is lenne kivonni egyet...).

```
$elozo = $_GET[foto] -1;
```

```
$kovetkezo = $_GET[foto] +1;
```

```
?>
```

```
<p align="center">
```

```
//Több kódblokkból fog állni a linkek elhelyezés,  
mert folyamatosan oda-vissza kell „ugrálnunk”  
PHP-ből HTML-be!
```

```
<?php
```

```
/*Hogy a lapozás ne zavarodjon meg az első vagy  
utolsó kép elérésénél, egy IF elágazással biztosíta-  
nunk kell, hogy az első képnél ne jelenjen meg az  
„Előző” link, az utolsó képből meg értelemszerűen  
a „Következő” link: */
```

```
if ($_GET[foto] !=1)
```

```
{ //Ha a paraméter nem egyenlő egy-  
gyel, akkor beilleszthető az „Előző” link! A hi-  
vatkozásba is PHP-t csempésztünk, mégpedig para-  
métert adunk át: a linkre kattintva az aktuális képnél  
eggyel előrébb lévő képet jeleníti meg a kep.php!
```



```
?>
<a href="kep.php?foto=<?= $elozo;
?>">Előző</a>
<?php
    }          //A HTML-ből visszalépve egy újabb
kódblokkban lezárjuk az IF utasítást a nyitó kapcsos
zárójel párjával!
?>
<strong><a
href="index.php">Vissza</a></strong>
<?php //Ismét szükségünk lesz egy IF elágazásra:
ha az átvett foto paraméter nem egyenlő az utolsó
kép számával (a példában 22 képünk van a galériá-
ban), akkor beilleszti a „Következő” linket is! Fontos,
hogy mindig az utolsó kép számát vizsgáljuk ebben az
elágazásban! A hivatkozásba ismét PHP-t csempé-
szünk, paramétert adunk át: a linkre kattintva az
aktuális képnél eggyel hátrébb lévő képet jeleníti meg
a kep.php!
if ($_GET[foto] !=22)
    {
?>
<a href="kep.php?foto=<?= $kovetkezo;
?>">Következő</a>
<?php          //Az első IF-hez hasonlóan itt is
vissza kell lépnünk PHP-be a hivatkozás beszúrása
után, hogy lezárjuk az elágazás igaz ágát.
    }
?>
</p></body></html>
```

Ezzel a kis kiegészítéssel lapozhatóvá tettük PHP képgalériánkat. HTML ismereteinket felhasználva dekoratívvá tehetjük a galériát különböző szín- és betűformázások használatával! Fontos azonban, hogy ne felejtsük el az **utolsó IF elágazásban lévő feltételvizsgálatnál** mindig az aktuális képmennyiség utolsó elemét vizsgálni! A példában 22 képet helyeztünk a galériába, tehát, ha a felhasználó eléri a 22. képet, nem fog megjelenni neki a „Következő” feliratú link, hanem csak a „Vissza” és „Előző” linkek. Ha újabb képeket szúrunk, **ezt ne felejtsük el átírni!** Ha 50 képünk van, akkor 50-et írunk a feltételvizsgálatba, ha például 100, akkor 100-at és így tovább. A lényeg, hogy ezt észben tartsuk, különben a lapozhatóság nem fog megfelelően működni! (Természetesen a képek számát is tárolhatjuk változóban, vagy vehetjük adatbázisból, így tetszőleges képszámmal is dolgozhatunk.)

17.3. Adatbázisok kimentése – DUMP-olás

A tapasztalt PHP programozók (is) tudják, hogy nagyon fontos biztonsági mentéseket készíteni az adatbázisokról, különösképpen, ha azok nagyon sok táblát tartalmaznak. Az adatbázisok elkészítése sok időbe telhet és még fontosabb, hogy a bennük tárolt adatok elvesztése nagy fejtörést

okozhat. Ezért is ajánlott legalább hetente egyszer biztonsági másolatot készíteni róla, arra az esetre, ha a rendszerrel valami történik, esetleg mi ejtünk valami hibát (például olyan rekordot vagy táblát törölünk, ami nem éppen szerencsés). Most megnézzük lépésenként, hogyan is tudunk manuálisan adatbázist kimenteni – „**dump**”-olni – és ezeket hogyan is illeszthetjük be újból!

1., Lépünk be a phpMyAdmin-ba!

2., Válaszuk azt az adatbázist a bal oldali listából, amelyet ki szeretnénk menteni (a példa során most a peldafeladat adatbázissal fogunk dolgozni, amit korábban készítettünk).

3., Ha nem az egész adatbázis szeretnénk kimenteni, hanem annak csak egy konkrét tábláját, akkor kattintsunk a tábla nevére bal oldalt. Mi most az egész adatbázist fogjuk menteni, így válasszuk ki a nevét bal oldalt (peldafeladat), majd a képernyő jobb oldalán, fenn kattintsunk az **Export** fülre:



Ezek után a következő ablak jelenik meg jobb oldalt:

Adatbázis kiírás (vázlat) megnézése

Export

nyilvantartas

Mindent kijelöl / Mindet töröl

- ☒ SQL
- ☐ LaTeX
- ☐ Microsoft Excel 2000
- ☐ Microsoft Word 2000
- ☐ MS Excel CSV adat
- ☐ CSV adat
- ☐ XML

SQL beállítások

Egyedi megjegyzés hozzáadása a fejléchez (n törli a sorokat):

☐ Export lezárása a tranzakcióban

☐ Idegen kulcsok ellenőrzésének letiltása

SQL export kompatibilitás: NONE

☒ Struktúra

- ☐ Tábla eldobás hozzáadása
- ☐ IF NOT EXISTS hozzáadása
- ☒ AUTO_INCREMENT érték hozzáadása
- ☒ Idézőjelek használata a tábla- és mezőneveknél

Hozzáadás a megjegyzéshez:

- ☐ Dátum Készítés/Módosítás/Ellenőrzés

☒ Adat

- ☐ Mezőneveket is hozzáadja
- ☐ Kiterjesztett beszűrások

Lekérdezés maximális hossza: 50000

- ☐ Időzített beszűrés használata
- ☐ Mellőző beszűrások használata
- ☒ Bináris mezők hexadecimálisként

Export típus: INSERT

☐ Fájlnév megadása

Átmeneti fájlnev (): DB (☒ emlékezzon az átmeneti névre)

Tömörítés: ☒ Nincs ☐ "zip-pel tömörítve" ☐ "gzip-pel tömörítve" ☐ "bzp-pel tömörítve"

Nekünk, mint kezdőknek csak a **bal oldali lista**,
fent, az Export felirat alatt és az **Export típus lenyí-
ló lista** most a lényeges, a többi nem fontos jelen-
leg! A fenti listából kiválaszthatjuk, hogy az adat-
bázis mely táblákkal legyen kidumpolva. Nekünk
most csak egy adattáblánk van, a nyilvantartas
nevű. Ezt ki is jelöltük. A lista alatt található radio
button-ok segítségével más formátumokban is
elmenthetjük az adatbázisunkat (a Microsoft Excel
2000-et kijelölve **.xls** kiterjesztésű Excel munkafü-
zetet készít a rekordjainkkal, a Microsoft Word
2000-re kattintva egy **.doc** kiterjesztésű Word fájlt,
melyben táblázatba rendezve jeleníti meg az adat-
tábláinkat - azok szerkezetét és rekordjait) stb.

Mi maradjunk most az **SQL radio button**-nál! A jobb oldalt, lent látható az export típusának megadása egy lenyíló lista formájában. Itt 3 féle kimenetről gondoskodhatunk:

- **INSERT:** ezzel úgy menti ki az adatbázist a phpMyAdmin, hogy **INSERT INTO**-s parancsokat készít a meglévő tábláink alapján, így a dump fájl tartalmát beillesztve egy teljesen üres adatbázisba, visszanyerhetjük azt.
- **UPDATE:** mint a neve is mutatja, frissíti a meglévő adatbázist és annak adattábláit, rekordjait. Ezt akkor érdemes használnunk, ha nem vett el az adatbázisunk inkább csak a régi rekordokat szeretnénk visszaállítani.
- **REPLACE:** teljes egészében kicseréli az adattáblát, és annak rekordjait a dump fájlban lévő értékekre.

4., Mi most használjuk az **INSERT** típust, így egy üres adatbázisba beilleszthetjük a korábban elkészített nyilvantartas táblánkat! Kattintsunk a képernyő jobb oldalának alján lévő, **Végrehajt gombra!**

5., A képernyő jobb oldalán megjelenik egy szövegdoz, amely **SQL** parancsok formájában fogja tartalmazni a biztonsági mentést. Ezt a szöveget jelöljük ki teljes egészében (bár csak a legelső **SQL** parancstól szükséges, a többi mellékes infor-

mációt (szerver verzió, PHP verzió, kimentés időpontja) tartalmaz. Egérrel jelöljük ki a szöveget (vagy kattintsunk valahova a szövegbe és Ctrl+A billentyűkombinációt használva) és másoljuk be ezt egy szöveges fájlba (mondjuk Jegyzetömbbe és mentjük el dump.txt néven). Célszerű a nevébe is beírni a kimentés dátumát, hogy tisztában legyünk mikori is a dump fájl.

Mivel az adatbázisunkat valahogy vissza is kell helyeznünk a dump fájl segítségével, ezt kétféleképpen is végezhetjük:

- **IMPORT** földre kattintva jobb oldalt fenn. Ekkor meg kell adnunk a szöveges fájl helyét, amibe a dump-olást elmentettük, emellett csak annyi a dolgunk, hogy a **Fájl karakterkészlete** lenyíló listán a **latin2**-t válasszuk (ha a kódolást nem jól állítjuk be, az ékezetes betűkkel problémák lesznek, nem fognak megjelenni!), majd a Végrehajt gombot megnyomjuk.



Import

Importálandó fájl

A szövegfájl helye (Legnagyobb méret: 2 048KB)

A fájl karakterkészlete:

Imported file compression will be automatically detected from: Nincs, gzip, bzip2, zip

- **SQL** parancs futtatásával: Ebben az esetben a fenti fülek közül válasszuk az SQL-t, ahol, mint

tudjuk, SQL lekérdezéseket tudunk futtatni, akár csak konzolos felületen. Ekkor csak annyit kell tennünk, hogy a megjelenő beviteli mezőbe beillesztjük a szöveges dump fájlunk tartalmát, majd megnyomjuk a Végrehajt gombot. Ezzel el is készült az biztonsági mentés beillesztése.

A két verzió közül teljesen mindegy melyiket használjuk, az eredmény ugyan az lesz! Javasolt az utóbbit használni, de akinek az IMPORT fül a szimpatikusabb nyugodtan használja azt (csak ne **felejtjük el a karakterkódolást**)! Ezzel meg is tanultunk dump-olni és a biztonsági mentést visszailleszteni. Használjuk ezt a gyakorlatban is, nehogy a munkánk kárba vesszen!

17.4. Linkajánló készítése

Ebben a példában egy linkajánlót készítünk, melyet dinamikusan tudunk majd bővíteni egy egyszerű admin felületen keresztül. Ehhez szükségünk lesz egy új adattáblára (amely tárolni fogja a linkek nevét, URL címét és leírását), egy `linkek.php` nevű oldalra (ezen fogjuk kilistázni az adattábla rekordjait) valamint egy admin mappára (melyben majd a linkajánló szerkesztéséhez, az admin panel megfelelő működéséhez szükséges php fájlokat tároljuk). Ezek elkészítését most nézzük lépésről-lépésre:

17.4.1. Linkek adattábla létrehozása

Hozzunk létre egy új táblát a már meglévő, *példafeladat* adatbázisunkba! A neve legyen *linkek*. Az alábbi szerkezetet alapul végre készítjük el a táblát:

Mező	Típus	Egybevetés
<u>id</u>	int(11)	
nev	varchar(100)	latin2_hungarian_ci
cim	varchar(100)	latin2_hungarian_ci
leiras	text	latin2_hungarian_ci

	Mező	Típus	Egybevetés	Tulajdonságok	Null	Alapértelmezett	Extra
<input type="checkbox"/>	id	int(11)			Nem		auto_increment
<input type="checkbox"/>	nev	varchar(100)	latin2_hungarian_ci		Nem		
<input type="checkbox"/>	cim	varchar(100)	latin2_hungarian_ci		Nem		
<input type="checkbox"/>	leiras	text	latin2_hungarian_ci		Nem		

Fontos! Az **id** mezőnk lesz a **kulcsmezőnk**. Ennél a mezőnél feltétlenül állítsuk be az **auto_increment** értéket, így automatikusan nő majd a mező értéke eggyel, mindenegyes új sor beszúrásánál!

Töltsük fel az elkészült táblát néhány rekorddal:

Név	URL cím	Leírás
Google	www.google.com	A világ legnépszerűbb kereső oldala!
A PHP nyelv	www.php.net	A népszerű PHP programozási nyelv hivatalos weboldala, sok hasznos információval.
MySQL website	www.mysql.com	Az egyik legelterjedtebb adatbázis-kezelő rendszer honlapja, sok infóval és referenciákkal.

A **http://** **protokolljelzést** szándékosan nem tettük az URL cím elé, mert úgy oldjuk meg a kilistázást, hogy a protokollt ne kelljen mindig a cím elé gépelni, ezzel helyet és időt is spórolhatunk! Ha elkészült az adattábla és feltöltöttük a rekordokat, jöhet a PHP és MySQL összekapcsolása!

17.4.2. Linkek.php elkészítése

Ezzel a résszel sem kell különösebben sokat bajlódunk, az eddigi ismereteink alapján könnyen elkészíthetjük a rekordok kilistázását egy táblázatba, a weblapon belül. Erre a célra készítsünk egy 3 soros, 1 oszlopos táblát a létrehozott linkek.php oldalra, amelybe az egyes mezők adatai kerülnek!

Nézzük a linkek.php forráskódját (<body> tagtól </body> tag-ig):

```
<body>
<p><h1>Linkajánló</h1></p>
<p>
<?php //Itt kezdődik az 1. php blokk. Csatla-
kozzunk a localhost-on lévő adatbázishoz, mely
csatlakozást eltárol a $kapcsolat connection string.
$kapcsolat = mysql_connect("localhost",
"root", "kozi1987") or die (print
"HIBA!".mysql_error());
mysql_select_db("peldafeladat", $kapcsolat);
//Kiválasztjuk a $kapcsolatból a 'peldafeladat'
adatbázist.
mysql_query("SET NAMES latin2");
$parancs = "SELECT * FROM linkek ORDER
BY nev"; //Kilistázzuk az összes rekordot a
'linkek' táblából a 'nev' mező szerint növekvő sor-
rendbe rendezve.
$eredmeny = mysql_query($parancs, $kap-
csolat);
```

```
//Lefuttatjuk a kérést($parancs) a megnyitott kapcsolaton($kapcsolat).
```

```
while ($sor = mysql_fetch_array($eredmeny))
```

```
//A WHILE ciklussal feldolgozzuk az eredményt.  
Addig listáz ki, ameddig van rekord a táblában.
```

```
{
```

```
?>
```

```
<!-- Itt újra HTML kezdődik, egy táblázatot szúrunk be, melynek soraiba kiíratjuk az aktuális rekord 'nev', 'cim' illetve 'leiras' mezőit -->
```

```
<table width="600">
```

```
<tr>
```

```
<td><strong>&nbsp;<?= $sor[nev];
```

```
?></strong></td>
```

```
<!-- Az első cellába jön a link neve -->
```

```
</tr>
```

```
<tr>
```

```
<td><a href="http://<?= $sor[cim];
```

```
?>" target="_blank"><?= $sor[cim];
```

```
?></a></td>
```

```
<!-- A második cellába jön maga a link. A HTML kód már eleve tartalmazza a http:// protokollt, így ezért nem kellett beírunk az adatbázisba minden linknél! -->
```

```
</tr>
```

```
<tr>
```

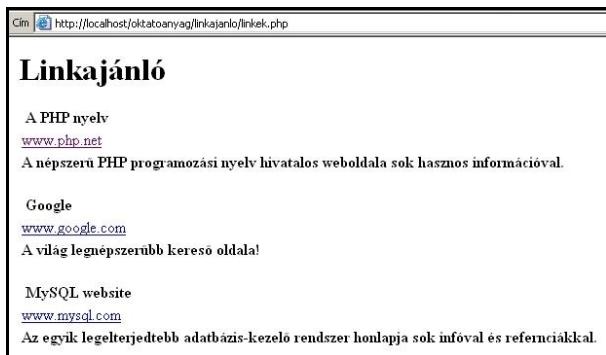
```
<td><p align="justify"><strong><?=
```

```
$sor[leiras]; ?></strong></p></td>
```

```
<!-- A harmadik cellába jön a link leírása --
```

```
>
```

```
</tr>
</table>
<br />
<p>
<?php //Miután a kiíratás rész befejeztük, nyitunk
még egy PHP blokkot és lezárjuk a WHILE ciklust!
Ezzel kész is az oldal!
    }
?>
</p>
</p>
</body>
```



A képernyőképen láthatjuk, hogy böngészőnkben megnyitva, az oldal kilistázza az adatbázisban lévő rekordokat. Láthatjuk, hogy a kilistázás nem volt nehezebb, mint mondjuk a *nyilvantartas.php* esetében. A folyamat lényegében ugyanaz, ezért megfelelő gyakorlás után már minden egyes pont-

ját rutinszerűen fogjuk gépelni. Ám a *nyilvantartas.php*-vel ellentétben, ez az oldal rendelkezni fog egy admin panellel, mely elkészítése ugyan több időt vesz igénybe, de ezáltal, az admin felületről dinamikusan szerkeszthetjük majd a linkjeinket (újabb rekordokat szűrhatunk be, illetve törölhetünk a linkek adattáblából a megismert DML parancsok és a PHP – MySQL összefűzésének segítségével. A következő pontban az ehhez szükséges PHP oldalakat készítjük el.

17.5. Admin felület elkészítése

Fontos, hogy megértsük az admin felület elkészítésének lépéseit, mert ezek ismeretében magunk is készíthetünk újabb admin felületet, vagy a mostanit is bővíthetjük a későbbiekben újabb funkciókkal! Legelőször is készítsünk egy mappát abba a könyvtárba, ahol a *linkek.php* oldal található és adjuk neki az **admin** nevet. Ebbe a mappába fogjuk elkészíteni az **admin.php**-t és a hozzá tarozó PHP oldalakat, amelyek segítségével szerkeszthetjük majd a linkeket, egyetlen egy HTML kód begépelése nélkül.

Lássunk is neki!

17.5.1.Admin.php elkészítése

A létrehozott admin mappába készítsünk egy új PHP oldalt, ez lesz az **admin.php**. Ez tulajdonképpen arra szolgál, hogy kilistázzon nekünk egy mezőt az adattáblából (mely alapján azonosítani tudjuk a megfelelő rekordot) és lehetőséget ad az adott rekord törlésére illetve új rekord felvételére – persze anélkül, hogy belépnénk a phpMyAdminba. A főoldal kódja a következőképpen fog kinézni:

```
<body>
<table width="650" align="center" border="3">
  <tr>
    <td><h2>Linkek - Admin panel</h2></td>
  </tr>
</table>
<table width="650" border="3" align="center">
  <tr>
    <td width="200"><strong>Név</strong></td>
    <td width="350"><strong>URL
cím</strong></td>
    <td width="100"><strong>Törlés (végle-
ges)</strong></td>
  </tr>
<?php
//A korábbi listázásokhoz hasonlóan egy táblázatba
helyezzük el a kiíratandó mezőket. A PHP blokkban
csatlakozunk a 'peldafeladat' adatbázishoz és
```

ugyanazt a parancsot futtatjuk le, mint a linkek.php esetében.

```
$kapcsolat = mysql_connect("localhost",  
"root", "kozi1987");  
mysql_select_db("peldafeladat", $kapcsolat);  
$parancs = "SELECT * FROM linkek ORDER BY  
nev";  
$eredmeny = mysql_query($parancs);  
while ($sor = mysql_fetch_array($eredmeny))  
{  
?>
```

<!-- Itt visszaugrunk a HTML részbe, hogy kiírassuk a táblázat megfelelő oszlopába a megfelelő mezőhöz tartozó értékeket! Most csak a link nevét, valamint URL-jét íratjuk ki, ez bőven elég, hogy beazonosítsuk a rekordot, amit éppen törölni akarunk. -->

```
    <tr>  
        <td bordercolor="#FFFFFF"><?= $sor[nev];  
?></td>  
        <td bordercolor="#FFFFFF"><?= $sor[cim];  
?></td>  
        <td bordercolor="#FFFFFF"><a  
href="torol.php?id=<?= $sor[id]; ?>">Tör-  
lés</a></td>
```

<!-- A 3. oszlopban szereplő PHP kód lehet első ránézésre furcsa. Ez egy link, amelyre kattintva a torol.php-ra ugrik a program. Az érdekessége, hogy most használjuk a táblázat id (azonosító) mezőjét, melyet paraméterátadással csatolunk a linkhez (amit majd a torol.php fog fogadni). Így

kell megoldanunk, hogy a törlés csak arra az egy adott rekordra vonatkozzon, amelyiknek a sorában a 'Törlés' linkre kattintottunk! Ennek a paraméterátadásnak a fontossága a torol.php forráskódjában jól látható lesz! -->

```
</tr>
<tr>
  <td colspan="5" bordercolor="#FFFFFF"><hr
width="100%" color="#000000"></td>
</tr>
<?php //Miután kilistáztuk az összes rekordot
lezárjuk a ciklust!
    }
```

```
?>
```

```
</table>
<table width="650" align="center" border="3">
  <tr>
    <!-- Beszúrunk egy linket az admin oldal aljára,
    amely a linkfelvitel.php-ra mutat, ahol új rekordot
    tudunk majd az adattáblánkba beilleszteni. -->
    <td>
      <p align="center"><strong><a
href="linkfelvitel.php">Új link felvitele</a>
</strong></p>
    </td>
  </tr>
</table>
</body>
```


Nagy valószínűséggel ennek az oldalnak az elkészítése sem jelentett komolyabb elkészítést azok számára, akik megértették a tananyag ide vonatkozó fejezeteit. Az újdonság – mint már azt a forráskódban is leírtuk – az ID mező paraméter átadással történő továbbküldése a `torol.php`-nek. Persze ez sem új teljesen, mivel paraméter átadást számos esetben használtunk, mind az űrlapoknál, mind a képgalériánál, ennek az esetnek a jelentősége, hogy ez most különösen azért fontos, mert a Törlés linkre kattintva csak az a rekord törlődik, amelynek a sorában a hivatkozás volt.

Ha mindent a fenti forráskód szerint csináltunk, akkor a következő felületet láthatjuk a böngészőnkben, ha megnyitjuk az `admin.php`-t:

Linkek - Admin panel		
Név	URL cím	Törlés (végleges)
A PHP nyelv	www.php.net	Törlés
Google	www.google.com	Törlés
MySQL website	www.mysql.com	Törlés
Új link felvitele		

Természetesen, ha mostani állapotában az oldalon a Törlésre kattintunk, nem fog csinálni semmit, mivel a `torol.php`-t nem készítettük el. Ugyanígy új linket sem tudunk jelenleg beszúrni az adatbázisba, a **linkfelvitel.php** és **felvisz.php** nélkül!

17.5.2. Torol.php elkészítése

Létrehozunk egy új, **.php** kiterjesztésű fájlt az admin mappánkba, a neve legyen: **torol**. Ez az oldal egy rövid PHP blokkból fog állni mindösszesen. Ki kell hangsúlyozni, hogy ez az oldal semmilyen HTML tag-et nem fog tartalmazni (és nem is fogjuk látni a képernyőn), kizárólag PHP-t (így, ha Dreamweaver-rel dolgozunk, nyugodtan töröljük ki mindent kódnézetben)! A forráskód, tehát **<?php** –vel kezdődik és **?>** – vel zárul. **Figyeljünk az üres sorokra is**, mert ha a kód elején véletlenül hagyunk egyet, az is **hibát eredményezhet** a működésben. Nézzük, hogyan is néz ki a **csak PHP kódot tartalmazó torol.php**:

```
<?php //Az oldal forráskódja a PHP blokk nyitásával kezdődik, és annak bezárásával fejeződik be!!!  
$kapcsolat = mysql_connect("localhost",  
"root", "kozi1987");  
mysql_select_db("peldafeladat", $kapcsolat);  
//Eddig a pontig már ismerjük a lépéseket: kapcsolódás, majd az adatbázis kiválasztása a kapcsolatból.  
$parancs = "DELETE FROM linkek WHERE id  
= $_GET[id]";  
/* Kiadjuk a parancsot: a törlés DML parancsot adjuk ki. A törlés lefordítva: Töröld a 'linkek' táblából azt a rekordot, amelynek az azonosítója: $_GET[id]. Vagyis a WHERE záradék után megad-
```

juk, hogy az előző oldal 'Töröl' linkkel átadott id paraméterét vegye át, és csak az átvett azonosítójú rekordot törölje az adattáblából! WHERE záradék nélkül az összes rekordunkat törölné az oldal!!! */

mysql_query(\$parancs); //Lefuttatjuk a parancsot, itt nem kell WHILE ciklus.

mysql_close(\$kapcsolat); //Lezárjuk a kapcsolatot.

header ("Location: admin.php"); //a header ("Location: "); függvénnyel átirányítjuk az oldalt vissza az admin.php-re, tehát, ha a program törli a kiválasztott rekordot, visszaugrik a főoldalra.

?>

Mint láthatjuk, ez a PHP oldal sem okoz már különösebb fejtörést, mivel a szükséges parancsokat már megismertük korábban. **Amit ne felejtünk el a későbbiekben sem az ilyen oldalaknál:**

- Ne tegyünk ilyen feldolgozó oldalakba HTML tag-et!
- Ne hagyjunk üres sorokat (főleg ne az elején)!
- Ne használjuk a DELETE parancsot WHERE záradék nélkül (csak indokolt esetben)!
- Figyeljünk oda a header függvényre, a zárójelben megadott **Location**-t mindig **nagy L** betűvel írjuk, ne kicsivel, mert különben nem fog működni!

Amennyiben ezeket a figyelmeztetéseket betartjuk nem valószínű, hogy oldalunk olyan hibát jelez vissza, amit nem ismerünk és nem tudjuk mi okozhatja. Az admin felületünk ezzel már félkész állapotban van, még szükségünk van **a linkfelvitel.php-ra és annak feldolgozó oldalára** (a felvisz.php-re), hogy új rekordokat is képesek legyünk beszúrni.

17.5.3.Linkfelvitel.php elkészítése

Ebben a pontban elkészítünk egy beviteli űrlapot, melyet kitöltve – és a feldolgozó oldalnak elküldve – a program beszúr egy új rekordot, anélkül, hogy beléptünk volna a phpMyAdmin-ba. A linkfelvitel.php-t szintén admin mappába hozzuk létre. A készítés során az űrlapok fejezetnél tanulatokat fogjuk hasznosítani. Vegyük a készítés lépéseit pontról-pontra:

1., Beszúrunk az üres weblapra egy Form-ot a Dreamweaver Form eszköztárából. Elvileg a következőképpen fog kinézni a form tag-je kódnézetben:

```
<form id="form1" name="form1"  
method="post" action="felvisz.php">
```

Nekünk ebből a method és az action a fontosak. Post metódussal adjuk át a feldolgozó oldalnak az adatokat (tehát titkosítva). Az action paraméterben

megadjuk azt az oldalt, amelyre az űrlap ugrik a 'Felvitel' gombra való kattintással. Ez lesz a felvisz.php, amelynek a szerkezet a torol.php-hoz lesz hasonló.

2., A form-on belül beszúrunk egy táblázatot, amelybe elhelyezzük az űrlap elemeket (persze ez nem kötelező, de így esztétikusabb és könnyebben formázható lesz az űrlapunk. Négy űrlap elemet fogunk használni, mégpedig három Text Field-et és egy Button-t. Nézzük is ezeket egyenként!

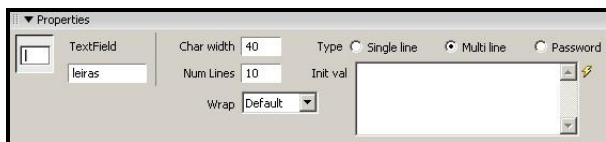
3., Az első Text Field, amit beszúrunk, a link nevének megadására szolgál. A szövegmező neve **nev** lesz. Íme a beállítása Dreamweaver-ben:



4., A második beviteli mező – ahol az URL címet adjuk meg – szintén Text Field lesz. Ugyanazokat az értékeket állítottuk be, mint a név esetében, csak itt a mező neve **cim** lesz.

5., A harmadik mező űrlap elem is Text Field, de itt már nem Single line-t fogunk használni, hanem Multi line-t, azaz többsoros beviteli mező. Mivel az adatbázisunkban a 'leiras' mező típusa TEXT volt, ezért biztosítani kell az űrlapon a meg-

felelő helyet a hosszabb leíró szövegek begépeléséhez is:



Láthatjuk, hogy a **leiras** nevű beviteli mező karakter szélességét 40-re, magasságát 10-re állítottuk, ezáltal 10 sor magas lett a Text Field. A sortörés módját (Wrap) hagyjuk Default-on (az alapbeállításon).

6., Most már csak egy űrlap elem hiányzik: a Button. Helyezzük az oldal aljára és adjuk meg a Value értékének, hogy Felvitel. Ez a szöveg fog ezáltal megjelenni a gombon, az Action beállítás pedig marad az alpból beállított Submit form (vagyis feldolgozó forma).

Ezek elhelyezése és beállítása után tulajdonképpen már el is készült az űrlapunk, melyet kitöltve új rekordot szúrhatunk be a *peldafeladat* adatbázisunkba, a *linkek* táblába. Persze még nem fog működni, mivel még egyetlen oldal elkészítése hátra van, ez az oldal pedig a *felvisz.php* nevű, *torol.php*-hez hasonló szerkezetű feldolgozó oldal.

Ha a fenti lépések szerint készítjük el a beviteli űrlapunkat, valami hasonlót kell látnunk a böngészőnkben a *linkfelvitel.php* megnyitása után:

Link feltöltése

Név	<input type="text"/>
URL cím	<input type="text"/>
Leírás	<div><div></div></div>

Figyelem! Fontos megemlíteni, hogy az URL cím nevű mező kitöltésénél ne feledjük, hogy csak a **www**-vel kezdődő címet kell megadnunk, a **http:// előjel nem kell**. Ha mégis kiteszük a linkünk hibás lesz a linkek.php oldalon (ahol már a HTML kódba eleve elhelyeztük a protokoll nevét, hogy ne kelljen minden egyes rekordnál újra eltárolni).

Ha véletlenül mégis így töltjük fel, akkor hibás URL-t kapunk: pl.: `http://http://www.google.com`). Erre figyeljünk oda a feltöltéskor!

17.5.4. Felvisz.php elkészítése

Az admin felületünk elkészítésének utolsó lépéshez érkeztünk. Már csak a **linkfelvitel.php feldolgozó oldalát** kell elkészítenünk, ami a `torol.php`-hez hasonlóan csak PHP kódblokkból fog állni. Itt szintén egy DML-es SQL parancsot fogunk erre a célra használni, mégpedig az `INSERT INTO`-t és természetesen az űrlapról átvett paramétereket fogjuk az adattáblánkba beszúrni. Lássuk:

```
<?php
```

```
$kapcsolat = mysql_connect("localhost",  
"root", "kozi1987");  
mysql_select_db("peldafeladat", $kapcsolat);
```

```
//Csatlakozás, adatbázis kiválasztása.
```

```
$parancs = "INSERT INTO linkek (nev, cim,  
leiras) VALUES ('$_POST[nev],  
'$_POST[cim]', '$_POST[leiras]')";
```

```
/* A $parancs változóban megadjuk az adattáblába  
illesztés SQL parancsát. A parancs hétköznapi  
nyelvre lefordítva: Illeszd be a 'linkek' táblába a  
(nev, cim és leiras mezőkbe) a következő értékeket  
(átvett nev, átvett cim, átvett leiras). */
```

```
mysql_query($parancs);
```

```
//Lefuttatjuk a parancsot.
```

```
mysql_close($kapcsolat);
```

```
//Lezárjuk a kapcsolatot.
```



```
header ("Location: admin.php"); //Átírányítjuk  
az oldalt az admin.php-re, ahol már szerepelni fog  
az új rekord a listában!  
?>
```

Most már teljes egészében elkészült az admin felületünk. Immár dinamikusan bővíthetjük a link-ajánlónkat, anélkül, hogy HTML-t szerkesztenénk, vagy a phpMyAdmin-t használnánk. Természetesen a példa kiegészíthető módosítással, az admin felület jelszóval való védelmével, stb.

Reméljük a könyvben szereplő példák érthetőnek és hasznosak bizonyultak a PHP nyelv iránt érdeklődők számára és ezt a tudást használni is fogják a jövőben!

Jó webprogramozást mindenkinek! ☺

A BBS-INFO KIADÓ AJÁNLATA

- 35 NAP – Lipták Béla 1956-os naplója (224 oldal, 1970 Ft.)
Adobe Photoshop zsebkönyv (192 o. 870 Ft.)
Az internet és lehetőségei (240 oldal, 1970 Ft.)
Csodálatos úticélok (160 színes oldal, keménytáblás, B6, 1970 Ft.)
GTK+ programozás Free Pascalban (244 oldal, 2490 Ft.)
Hétköznapi marketing (4 féle könyv, 870-970 Ft.)
Hogyan használjam? Minden amit a számítógép használatához tudni kell (560 o. 2490 Ft.)
Hogyan kezdjem? (240 o. 1430 Ft.)
Informatikai füzetek (ECDL vizsgákhoz)
 1. Az információ technológia alapfogalmai (152 o. 970 Ft.)
 2. Operációs rendszerek (248 o. 1750 Ft.)
 3. Szövegszerkesztés (242 o. 1430 Ft.)
 4. Táblázatkezelés (192 o. 1190 Ft.)
 5. Adatbázis-kezelés (168 o. 1190 Ft.)
 6. Prezentáció-készítés (104 o. 1190 Ft.)
 7. Információ és kommunikáció (94 o. 870 Ft.)
 9. ECDL Képszerkesztés (144 o. 970 Ft.)
 10. ECDL Webkezdő (144 o. 970 Ft.)
JavaScript kliens oldalon (188 o. 2900 Ft.)
Kezdő bűnözők kézikönyve (216 o. 970 Ft.)
Macromedia Flash MX (204 o. 1970 Ft.)
Makróhasználat Excelben (118 oldal, 1750 Ft.)
MySQL.NET (540 oldal, 3920 Ft.)
Nevelés és értékkutatás - A pedagógiai gyakorlat pszichológiája (200 o. 1970 Ft.)
Office XP (352 o. 2190 Ft.)
Office 2007 (464 oldal, 3920 Ft.)
OpenOffice.org (400 oldal, 2990 Ft.)
Ötletek és tanácsok Windows felhasználóknak (176 o. CD mell. 1970 Ft.)
PC hardver kézikönyv (360 o. 2870 Ft.)
Programozás Turbo Pascal nyelven (232 o. 1290 Ft.)
Puskázási kézikönyv (128 o. B6, keménytáblás, 970 Ft.)
Szerver oldali Webprogramozás (192 o. 2900 Ft.)
Szövegszerkesztési feladatgyűjtemény + CD mellékl. (144 o. A4, 1970 Ft.)
Titkos hadműveletek pénztárcánk ellen (240 o. 540 Ft.)
Videószerkesztés házilag (240 o. 1970 Ft.)
Webes adatbázis-kezelés MySQL és PHP használatával (160 oldal, 2490 Ft.)
Weblapkészítés házilag (200 o. 1760 Ft.)
Windows XP részletesen (368 oldal, 2490 Ft.)
Windows 7 mindenkinek (392 oldal, 2990 Ft.)
XHTML - A HTML megújulása XML alapokon (200 o. 2900 Ft.)
Zsebkönyvek (Word-Excel-Outlook-PP-Access 2007, 2010, Windows XP, Vista, 7 stb.)

**RÉSZLETES INFORMÁCIÓK, ÚJDONSÁGOK
ÉS TARTALOMJEGYZÉKEK AZ INTERNETEN:**

WWW.BBS.HU

A fenti könyvek megrendelhetők a kiadó címén: **BBS-INFO Kft.** 1630 Bp., Pf. 21.
Nagyobb mennyiségű rendelés esetén kedvezményeket tudunk biztosítani.

SECONDORB

Ingyen is játszható internetes városépítő stratégiai játék



- építsd fel saját városod kedved szerint
- készítsd el saját épületed és tölts fel
- gazdálkodj okosan a nyersanyagokkal
- hívd ki barátaidat logikai játékokra
- kereskedj másokkal
- ha kedved van, csatázhatsz is másokkal
- válaszolj tesztkérdésekre
- nyerj valódi ajándékokat

WWW.SECONDORB.HU WWW.SECONDORB.COM

BETŰVETÉS

**Olvasson regényeket, történeket, írásokat akár ingyen!
Publikálja írásait ingyen vagy pénzért!**

Nálunk nem kell megvennie a könyvet ahhoz, hogy elolvashassa.

Írások minden témakörben: Bulvár - Család - Életmód - Hobbí - Dokumentumminták - Erotika - Kultúra - Vallás - Mese - Ifjúsági - Mondások - Idézetek - Művészet - Receptek - Gasztronómia - Regények - Novellák - Sport - Szakirodalom - Számítástechnika - Történelem - Utazás - Versek - Viccek - Egyéb

WWW.BETUVETES.HU