PHP a gyakorlatban

Ismerkedés

A PHP (hivatalos nevén PHP: Hypertext Preprocesszor) egy szerver oldali HTML-be ágyazott scriptnyelv. Ugye, milyen bonyolultan hangzik? De máris tisztább lesz, ha azt mondom, hasonló a javascripthez. No nem mindenben, sőt! Hogy összehasonlíthassuk őket, ismételjük át – gyorsított tempóban – a javascript néhány, számunkra fontos momentumát.

Ha egy HTML oldalon a megszokott, statikus elemek helyett némi dinamizmust is szeretnék látni, erre nagyszerű lehetőséget biztosít a javascript. Bizonyos megszorításokkal látványos, felhasználóbarát lapokat készíthetünk vele. Nézzünk egy egyszerű példát:

```
<HTML>

<SCRIPT>
    var ora = (new Date()).getHours();

    if (ora<=6 || ora>=20) {
        document.writeln ("Jó estét, ");
    } else if (ora < 10) {
        document.writeln("Jó reggelt, ");
    } else {
        document.writeln("Jó napot, ");
    }
</SCRIPT>

kedves látogatóm!<HR>

</HTML>
```

Ha megnézzük a fentieket, láthatjuk, hogy az aktuális órától (napszaktól) függően az oldal üdvözli látogatóját. Figyeljük meg, hogy a javascript betét a <SCRIPT>…</SCRIPT> tagek közé van ékelve. Ezt hívjuk voltaképpen HTML-be ágyazásnak, hiszen a HTML forráson belül bárhol nyithatunk javascript betétet, legyen az akár egy sor közepe – természetesen a megfelelő szintaktikai szabályok betartásával. A PHP-vel ellentétben a javascriptet az aktuális böngésző hajtja végre, azaz a kliens (felhasználó) gépén fut, tehát függ annak teljesítményétől. A PHP részeket a PHP értelmező a szerveren futtatja le, s csak az eredményt adja vissza a böngészőnek. Ebből származik a PHP egyik nagy előnye: nem látható a forráskód. Egy javascriptet használó oldal forráskódját lekérve láthatjuk teljes forrását. Némi tudással bárki átírhatja a rajta található kódot saját képére, s használhatja azt. Ugye senki sem örülne, ha kemény munkával kidolgozott javascript rutinjait valaki kéretlenül használná? A PHP-s oldalak

forrását lekérve csupán a generált HTML részt fogják látni a kíváncsi szemek, tehát a PHP rutinok sohasem kerülnek forrásukban képernyőre. (Természetesen, hacsak nem akarom…) Hogy tisztább legyen, erre is lássunk egy nagyon egyszerű példát. A fenti javascriptes oldalrészletet valósítsuk meg PHP-ben. Magyarázatok a megoldás után:

```
<HTML>

<?php
    $ora = date ("g");

    if ($ora<=6 or $ora>=20) {
        echo "Jó estét, ";
    } else if ($ora < 10) {
        echo "Jó reggelt, ";
    } else {
        echo "Jó napot, ";
    }
?>

kedves látogató!<HR>

</HTML>
```

Láthatjuk, hogy minimális különbségek vannak a javascriptes változattal szemben. Az első, ami feltűnhet, hogy nem <SCRIPT>…</SCRIPT> tageket használunk. A PHP-s részek kezdéséhez a "<?php" előtagot használjuk, míg zárásához "?>" zárótagot kell írni. Ami e kettő között van, azt a PHP értelmezője fogja végrehajtani. A második, ami különböző, hogy az "ora" nevű változó előtt mindig szerepel egy "$" jel. Ez bizony kötelező, minden változó dollárjellel kezdődik. Hogy a date("g") mit jelent, egyelőre elégedjünk meg annyival, hogy az aktuális dátum órarészét adja vissza. A többi ugyanaz. Vagyis… Ha lekérnénk mindkét oldal forrását, beigazolódna, amit előzőleg említettem. Első esetben a teljes forrást megkapnánk, míg a PHP-s változat esetében a PHP rész helyett csak az aktuális napszakhoz tartozó köszöntés szövege jelenne meg.

Nos, nem véletlenül írtam az előző mondatot feltételes módban. Ha a javascriptes változatot valaki kimásolja és beilleszti egy HTML oldalba, bármilyen javascriptet ismerő böngészővel hehívva azt, megjelenik az eredmény: a napszaknak megfelelő köszöntés. Próbáljuk ki ugyanezt a PHP változattal. Nem lesz sok köszönet benne, ráadásul, ha lekérjük az oldal forrását, visszakapjuk szóról-szóra a begépelt PHP részletet is. Persze, hiszen a PHP-t nem értelmezi a böngésző, ehhez egy PHP értelmezőre van szükség. Lássuk tehát, hogyan építünk fel egy PHP fejlesztői környezetet, ahol elkészíthetjük és futtathatjuk PHP-ben készült munkáinkat.

Installálás

Ahhoz, hogy hozzákezdjünk bármilyen munkához, szükségünk lesz egy olyan webkiszolgálóra, ahová teljes hozzáférésünk van. Mivel ez nem biztos, hogy mindenki számára megoldható, legegyszerűbb, ha otthon mindenki épít egyet magának. Hozzávalók egy személyre: kell operációs rendszer. Én éppen egy Windows Millennium béta verziót nyúzok, ez alapján minden példám Win32 platformra vonatkozik majd. Remélem, ez senkinek sem okoz problémát. Bízom benne, hogy aki Linux-szal foglalkozik, annak sem okoz majd nehézséget adaptálni az elmondottakat. Az operációs rendszer alá szükségünk lesz egy feltelepített TPC/IP protokollra. Aki internetezik a gépén, annak biztosan van már telepítve, aki pedig nem, annak sem lesz bonyolult dolga. Be kell szerezni vagy egy modemet, vagy egy hálózati kártyát. Az utóbbiból egy régebbi típusú már párszáz, vagy alig több, mint ezer forintért is beszerezhető. Telepítsük fel, majd a Vezérlőpultban a Hálózatok alpontban adjunk hozzá egy TCP/IP protokollt, majd indítsuk újra a gépet. Ha mindent jól csináltunk, készen van a TCP/IP installálása.

Következő lépésként szerezzünk be néhány programot. Kell egy webszerver. Lehetőségeink, amit a PHP támogat: Windows NT Internet Information Server (IIS), ami bizony nem olcsó. Microsoft Personal WebServer (PWS), ami a telepítő CD-n megtalálható. Én nem találtam elég jónak, ráadásul nagyra terpeszkedik (mint a legtöbb MS program), de nem nyújt annyit. Használhatjuk még az OmniHTTPd nevű szerverprogramot is, de ez egy shareware, azaz bizonyos idő után lejár, fizetni kell érte. Egyetlen program marad, ami a világ egyik legelterjedtebb webszervere, sőt többplatformos, azaz fut Win32-es rendszeren és Linuxon is. Ez fontos, tudniillik a PHP szintén platformfüggetlen. Szóval szerezzünk be egy Apache webszervert. Ingyenesen letölthető az alábbi címről: http://www.apache.org/. Telepítése rendkívül egyszerű, alapértelmezésben létrehoz egy \Apache Group\Apache könyvtárat a Program Files rendszerkönyvtáron belül. Ha elindítjuk, az alábbi üzenetet kapjuk:

```
APACHE.EXE: cannot determine local host name.
Use the ServerName directive to set it manually.
```

És azonnal le is áll a program futása. Kapjuk elő valamelyik texteditorunkat, és töltsük be a \conf\ könyvtárban található httpd.conf nevű állományt. Keressük meg

valahol a 230. sor környékén a " #ServerName new.host.name" sort. Írjuk át. Szedjük le az elejéről a megjegyzést (#), majd a new.host.name helyett írjuk be a saját szerverünk nevét. Ez alapértelmezésben "localhost". Sorunk tehát: ServerName localhost. Természetesen a mondatvégi pont nélkül. Indítsuk újra az apache.exe-t. Eredményül az "Apache/1.3.x (Win32) running..." üzentet kapjuk egy DOS ablakban. Ha nem kapjuk vissza a promptot, akkor jól csináltunk mindent, fut a webszerver. Ha Windows 95 alatt próbálkozna valaki, jó tudni: az Apache futtatásához szükség van a Winsock2 modulra. Windows 98 és fölötte már ez beépített, Windows 95-re viszont installálni kell. Leszedhető az alábbi címről: http://www.microsoft.com/windows95/downloads/

Az Apache leállítása CTRL+C-vel történik, bár Windows NT alatt service-ként is futtatható, amiről bővebben az Apache dokumentációjában lehet olvasni.

Indítsuk el kedvenc böngészőnket, majd a címsorba írjuk be szerverünk címét: http://localhost. Enter után az Apache angol nyelvű üdvözlőképernyője jelentkezik be. Akkor van egy saját webszerverünk! Következhet a PHP installálása.

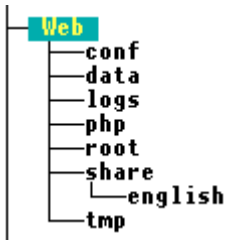
A PHP jelen pillanatban, két változatban is létezik. Az egyik a régi, de stabilabb és elterjedtebb változat. Ez a PHP3. Az újabb, több lehetőséggel rendelkező változat a PHP4. Mindkettő letölthető a készítők hivatalos webhelyéről a http://www.php.net/ címről. Töltsük le nyugodtan a 4.0-s verziót, szóbeszéd szerint máris megjelent a 4.01-es fordítás, amely máris stabilabb, jóval kevesebb gyermekbetegsége van. Letöltés után nincs mit installáljunk, kapunk kb. 10 állományt, amelyet bárhol elhelyezhetünk. No ez nem egészen igaz, lássuk a részleteket. A DLL kiterjesztésű állományokat helyezzük el a \Windows\System\ könyvtárába (NT esetén \WinNT\System32\), itt biztosan mindig megtalálja az oprendszer. (Vigyázzunk, a PHP-vel adott MSVCRT.DLL nevű állomány lehet, hogy régebbi, mint ami eredetileg a System könyvtárban található. Ellenőrizzük le, mielőtt felülírnánk. A hossz általában segít.) Van egy php.ini-dist nevű állomány, amelyet át kell nevezzünk php.ini-re. Indítsuk el a php.exe-t, majd a következő sorokba írjuk be a következőket:

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

Az utolsó sor után nyomjuk meg a CTRL+Z billentyűkombinációt, jelezve, hogy befejeztük a gépelést. Ha nem vétettünk hibát, egy hosszú HTML listát fogunk látni

elfutni a képernyőn. Amit látunk (illetve nem látunk…) az a fenti PHP nyelvű, egysoros programocska eredménye HTML formátumban. Nem kell aggódni, nem így fogunk tesztelgetni, hanem egybeépítjük az Apache-ot és a PHP-t. Ehhez kicsit lefosztjuk az Apache-ot, a teszteléshez nem lesz szükség mind a 2,5 MB-ra.

Hozzuk létre az alábbi könyvtárakat a főkönyvtárban. Azért ide, hogy minél kevesebbet kelljen majd írnunk.



A /conf/ könyvtárba hozzuk létre a már említett httpd.conf konfigurációs állomány, no persze nem kell egy 20 kilobyte-os állományt kézzel beírnunk, elég lesz ennyi is:

```
ServerRoot "c:/web"
DocumentRoot "c:/web/root"

ServerAdmin yinyan@westel900.net
ServerName localhost
Port 80

ErrorLog logs/error.log
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access.log common

DirectoryIndex index.html index.php

AddType application/x-httpd-php .php
ScriptAlias /php/ "c:/web/php/"
Action application/x-httpd-php "/php/php.exe"
```

Nézzük át, mi mit jelent.

- *ServerRoot:* az Apache útvonala.
- *DocumentRoot:* a HTML állományaink könyvtára. Itt kell majd elhelyeznünk az oldalunkat.
- *ServerAdmin:* a szerver adminisztrátorának jelszava. Ide mindenki írja a saját e-mail címét. Nagy szükségünk nem lesz rá, de jó, ha van.
- *ServerName:* a szerver neve. Erről már szóltam.
- *ErrorLog:* a /Web/logs/ könyvtárba fogja tenni az Apache a lapok letöltésekor keletkező hibákat, illetve az Apache hibaüzeneteit errors.log néven.

- *LogFormat:* a /Web/logs/ könyvtárban lesz még egy állomány, access.log néven. Ez a hozzáféréseket rögzíti, hibakeresési célokra ez is jól alkalmazható.
- *CustomLog:* az előző sor csak a log formátumát írta le, ez pedig a helyét is meghatározza.
- *DirectoryIndex:* ha egy könyvtárat adok meg a browsernek, melyek legyenek azok az állományok, amelyeket keresni fog, mint kezdőlap. Elsősorban az index.html lapot fogja betölteni, ha ez nincs, az index.php lesz a következő, amit keres. (Erről majd bővebben beszélünk még.)
- *AddType:* a következő három sor nagyon-nagyon fontos. Az alábbi bejegyzés a PHP filekiterjesztést ismerteti meg az Apache webszerverrel.
- *ScriptAlias:* a PHP.EXE helyét írjuk le. (Később erről is szó lesz.)
- *Action:* mit is kell tenni, ha a PHP kiterjesztéssel találkozunk? Itt található.

Ennyi tehát a httpd.conf állomány. Ugyanebbe a könyvtárba (/conf/) kerül a telepített APACHE mappából a mime.types nevű állomány.

A következő, hogy a /php/ könyvtárba másoljuk bele a PHP.EXE-t és a PHP.INI-t. Mint már megbeszéltük, a PHP DLL kiterjesztésű állományait helyezzük el a /Windows/System/ könyvtárban – vigyázva az MSVCRT.DLL-lel.

A /Web/ könyvtárba kerül az APACHE.EXE és az APACHECORE.DLL. Ezek után uninstallálhatjuk nyugodtan az Apache-ot, tovább nem lesz rá szükségünk ezt a négy elkülönített állományt leszámítva.

Készen is lennénk, ha nem gondolnánk a jövőre is. A PHP4-be már be van építve egy SQL szabványú adatbázis-kezelő rendszer. Ehhez viszont szükségünk lesz egy adatbázisszerverre, amely shareware verzió, és letölthető a http://www.mysql.com címről. Ez egy kb. 5 MB-os anyag, de ne aggódjunk – ezt is lefaragjuk. A felinstallált anyagból másoljuk át a MYSQLD.EXE-t a /Web/ könyvtárba, a /share/english/ könyvtárba pedig az ERRMSG.SYS-t és az ERRMSG.TXT-t. Kell még a felinstallált könyvtárból a /data/mysql/ alkönyvtár, ezt egy az egyben másoljuk át a /web/data/ könyvtárba. (Ez alapértelmezésben kb. 15 állományt jelent.)

Hátravan még egy batch állomány létrehozása, hogy ne kelljen mindig kézzel elindítgatnunk az SQL szervert és a webszervert. Készítsünk egy WEB.BAT állományt a /Web/ könyvtárban, az alábbi tartalommal:

```
@echo off
mysqld --basedir=c:\web
apache
```

A második sor elindítja a MySQL szerverét, paraméterként a szerver alapkönyvtárát adtuk meg. A harmadik sor pedig a webszervert indítja el.

Kész! Rendezzünk egy ellenőrzést:

Van egy főkönyvtárból nyíló /Web/ nevű könyvtárunk, benne négy állomány (web.bat, apachecore.dll, apache.exe, mysqld.exe). Alkönyvtárai: /conf/, amely 2 állományt tartalmaz: (httpd.conf, mime.types), /data/, amely egy mysql nevű alkönyvtárban kb. 15 állományt tartalmaz. A következő könyvtár: /logs/, amely most üres, /php/ amely a php.exe-t és a php.ini-t tartalmazza. A /root/ könyvtárunk még üres, a /share/ viszont tartalmaz egy /english/ alkönyvtárat, benne két állományt (errmsg.sys, errmsg.txt). S végül a /tmp/ könyvtár szintén üres, továbbá nem feledkezünk el a /Windows/System/ könyvtárba másolt DLL kiterjesztésű állományokról sem.

Próbáljuk futtatni a web.bat nevű állományunkat. Ha nem kapunk hibaüzenetet és a DOS ablakban az "Apache/1.3.x (Win32) running…" üzenet is megjelent, akkor valószínűleg jól konfiguráltunk be mindent. Utolsó ellenőrzésképp nyomjuk le a CTRL+ALT+DEL kombinációt (csak egyszer!), s ha a tasklistában látjuk a mysqld nevét, akkor tényleg minden rendben. Idáig!

Első programjaink

Beszéltünk róla, hogy a /root/ könyvtárba kell tenni az elkészített lapjainkat. Kezdjük valami egyszerűvel. Hozzunk létre egy index.html nevű állományt a /root/ mappában, s írjunk bele valami izgalmasat:

```
<HTML>

    Hurrá, ez az első programunk!
    <br>
    Még nincs benne semmi PHP, de tesztnek jó!

</HTML>
```

Böngészőprogram betölt, cím http://localhost, s láthatjuk első, saját webszerverünkön futó oldalunkat. Nem sok, csupán két sor, de működik. Kérjük le a forráskódot, s láthatjuk, mit tettünk. Most próbáljunk valamit alkotni PHP-ben. Töröljük ki az előzőleg beírt sorokat, próbálkozzunk a következővel:

```
<HTML>

    Ez már a második programunk!
    <br>
    <?php phpinfo(); ?>

</HTML>
```

Frissítsük az oldalt. Az "Ez már a második programunk!" szöveg szépen meg is jelenik, de már nem. Ha lekérjük a forráskódot, meglepetés, de a PHP forrás is benn lesz, méghozzá szó szerint. Pedig a phpinfo() parancs a telepített PHP környezetről adna néhány oldalnyi információt. Mi a probléma? A megoldás egyszerű: bizony nem hiába készítettük kézzel a httpd.conf állományt. A három utolsó sorban van a megoldás. Csak akkor használhatunk PHP betéteket, ha .PHP kiterjesztést adunk a file-unknak. Nevezzük tehát át az index.html-t, index.php-re. Vigyázzunk, az index.html ne legyen a könyvtárban, mert ezt fogja keresni először a böngésző. Frissítsünk! Az oldal tartalmát nem fogom leírni, mindenki láthatja maga: egy szép, nagy táblázat, tele mindenféle információval. Nézzük meg újra a forrást! Ez már nem PHP, hanem HTML! Gyakorlásképp gépeljük be első és második programrészletünket, amely a napszaknak megfelelően köszönti az oldal nézőjét. Az elsőt mentsük el elso.html néven, majd hívjuk be a böngészőbe: http://localhost/elso.html. Ugye működik? A második neve masodik.php legyen (vigyázzunk, ide már PHP kiterjesztés kötelező!), ezt a http://localhost/masodik.php cím beírásával jeleníthetjük meg. Mi történne, ha az elsőt

nem html, hanem php kiterjesztéssel mentettük volna le? Semmi probléma, csak a cím változott volna egy picit. A legfontosabb tehát, hogy nyugodtan adhatunk php kiterjesztést állományainknak akkor is, ha az nem tartalmaz PHP betéteket. Visszafelé – tapasztalatunk szerint – nem működik, tehát html kiterjesztést ne adjunk PHP rutinokat tartalmazó állománynak.

PHP alapozás

A PHP igen engedékeny nyelv. Nem ragaszkodik sok olyan dologhoz, amely miatt más programozási nyelv már sikoltozna. Teljesen mindegy, hogy a PHP részt hol kezdem, illetve, hogy hol fejezem be. Kezdhetem közvetlenül a sor elején, de használhatok írás közben tabulátorokat a könnyebb olvashatóság kedvéért. Egy példa:

```
<?php echo "Ez egy egysoros PHP script"; ?>
```

Ez az egy sor kiírja a megadott szöveget. Egyanezt megoldhattam volna az alábbi módon is:

```
<?php
    echo "Ez egy többsoros PHP script";
?>
```

Voltaképpen addig, amíg az értelmező pontosvesszővel nem találkozik, egy sorként értelmezi a beírt anyagot. Hogy tovább fokozzam pozitív értelemben vett "igénytelenségét", újból példákkal illusztrálok:

```
<?php echo "1. példa"; ?>
<? echo "2. példa"; ?>
<script language="php">echo "3. példa";</script>
<% echo "4. példa"; %>
```

Nézzük meg ezt a négy példát. A különbség tulajdonképpen csak a PHP értelmező hívásában rejlik. Az első esetben a megszokott "<?php" kezdőtaggal indítottunk. Ez a legelterjedtebb, mondhatni a hivatalos verzió. A második példa könnyedebb, de ugyanúgy elfogadott. A harmadik azok kedvéért van, akik megszokták a Javascript nyelvezetét. Ezzel illik vigyázni, mert bizonyos HTML editorokba betöltve nemkívánatos eredményeket kaphatunk. A negyedik pedig azok számára lehet ismerős, akik ASP-ben otthon vannak. Az ASP a Microsoft szerver oldali scriptnyelve - sajnos, alacsony szintű támogatottsággal.

Hasonlóan engedékeny megjegyzések elhelyezése terén is:

```
<?php
    echo "Ez egy C++ stílusú megjegyzés"; // megjegyzés
    /* Bár ha akarom, több sorba is tördelhetem
    a megjegyzéseimet. */
    echo "vagy esetleg shell típusú?"; # ekkor így kell
?>
```

Lehetőségek tárháza, csak győzzük őket kihasználni. Érdemes általában egy stílus mellett megmaradni, hiszen mint tudjuk, madarat tolláról, programozót programjáról...

Változók

Annyit már tudunk a változókról, hogy "$" jellel kezdődnek. Fontos tudnunk róluk, hogy a változónevekben a kis- és nagybetűk különbözőek. Nem egyenértékű a "var" és a "Var" nevű változó! Ez elég nagy hibaforrást eredményezhet, főleg kezdők számára.

A PHP ötféle típust kezel, ami a PHP4 kibocsátásával 6-ra növekedett. Lássuk őket:

Integer (egész számok)

Floating-point (lebegőpontos számok - törtek)

Strings (karakterfüzérek - szövegek)

Arrays (tömbök)

Objects (objektumok)

Bool (logikai - csak a PHP4 verziójától kezdődően)

Egész

Nézzük először az egész típusú változókat:

```
$a = 1234; # decimális szám
$a = -123; # negatív szám
$a = 0123; # 8-as számrendszerbeli szám (ez 83 tizesben)
$a = 0x12; # hexadecimális szám (ez 18 tizesben)
```

Itt nincs magyaráznivaló, ilyen egyszerű. Nem kell meghatározni, a maximális és minimális értékeket, a PHP mindig annyi byte-ot foglal le, amennyi feltétlenül szükséges.

Lebegőpontos

Lebegőpontos számok esetén még egyszerűbb, csak a tizedespontot kell a megfelelő helyre kitennünk:

```
$a = 1.234;
$a = 1.2e3;
```

Az első eset a szokásos forma, a második pedig egy úgynevezett tudományos forma (exponenciális alak). Jelentése: 1.2 * 10^3, azaz 1.2 * 1000, még pontosabban 1200.

Szöveg

Amilyen egyszerűek a számok, annyira sokoldalúak a stringek. Ha valaki ismerős egyéb nyelvekben, tudja, hogy a karakteres értékeket vagy idézőjelek ("), vagy aposztrófok (') közé kell tenni. Nos, a PHP mindkettőt használja.

A legfontosabb különbség a kettő között, hogy az idézőjelek közé tett kifejezés kiértékelődik, míg aposztrófok között nem. Ahhoz, hogy világosabb legyen, újból példákhoz folyamodunk:

```
<?php
    $a = "Ez egy string";

    $a = $a . ", meg még egy kicsi...";
    // a PHP-ben hozzáfűzésre nem "+" jelet, hanem pontot
    // használunk

    $a .= " a végére.";
    // egy másik módszer az összefűzésre,
    // talán ismerős más nyelvekből

    $num = 5;        // vezessünk be egy számot is

    $str = "A szám értéke: $num";
    echo $str;
    // Az eredmény:
    //        A szám értéke: 5

    $str = 'A szám értéke: $num';
    echo $str;
    // Az eredmény:
    //        A szám értéke: $num

?>
```

Itt már több ismeretlen is szerepel. Egyrészt, mint láttuk, szöveg összefűzésre nem "+" jelet, hanem "."-ot használunk. A másik módszer kicsit ravaszabb, nem kell kétszer kiírni a módosítandó változót, ettől eltekintve ugyanaz, mint az előző. Alatta láthatjuk a különbséget az idézőjelek és az aposztrófok között. Ha idézőjelek között szerepel egy változó, az értelmező kiszámítja azt, és behelyettesíti. Ha aposztrófok közé tesszük, minden értelmezés nélkül kiírja. Ez a kétféle lehetőség egy kicsit megzavarhatja a műveleteket:

```
<?php
    $a = 1 + "10.5";                // $a = 11.5
    $a = 1 + "-1.3e3";              // $a = -1299
    $a = 1 + "bob-1.3e3";           // $a = 1
    $a = 1 + "bob3";                // $a = 1
    $a = 1 + "10 kicsi indián";     // $a = 11
    $a = "10.0 indián " + 1;        // $a = 11
?>
```

Láthatjuk, hogy számnak értelmez mindent a PHP, amíg egy nem-szám karakterrel találkozik. Jó példa erre a negyedik sor (1+"bob3"), hiszen a végeredmény nem négy, csupán egy lesz, míg az alatta levő sorban 1+"10 kicsi indián"-ból a 10 még számként értelmezhető.

Sok más egyéb nyelvhez hasonlóan (C, Perl) lehetőség van "escape" karakterek használatára. Ennek akkor vehetjük hasznát, amikor vezérlőjelet szeretnénk elhelyezni a karaktersorozatban, vagy olyan jelet, aminek beírásával szintaktikailag helytelen kifejezéshez jutnánk. Erre egy jó példa, ha idézőjelet szeretnénk elhelyezni a szövegben. Egy táblázatban összefoglaltam a lehetőségeket:

Megnevezés	**Jelentés**
\n	Új sor
\r	Carriage Return (kocsivissza)
\t	Tabulátor
\\	Backslash (\)
\$	Dollárjel (amit ugye a változók jelölése miatt nem lehetne…)
\"	Idézőjel

```
<?php
    echo "Így kell idézőjelet kiírni: \"";
    echo "Soremelés következik\n";
    echo "- ez már új sorban van.";
?>
```

A szöveges típusra szintén jellemző a dinamizmus, azaz csak annyi helyet foglal el, amennyire mindenképpen szüksége van. Ez lehet egy szónyi információ, egy mondat, de egy többboldalas szöveget is érthetünk alatta.

Tömb

A következő típus a tömb. Minden nyelvben van egyfajta tömb, amely ugyanúgy használandó, mint itt:

```
<?php
    $a[1] = 100;
    $a[2] = "példa";
    $a[3] = 3.1415926;
?>
```

A példa alapján látható az első nagy különbség. A tömb egy összetett adattípus, minden eleme egy egyszerű típus (integer, floating-point, string), és ezt úgy keverhetjük, ahogy tetszik, nem kötelező egy tömbön belül ugyanazt a típust használnunk. A második különbség a megadási módban rejlik:

```
<?php
    $a[1] = 100;
    $a[] = "példa";
    $a[] = 3.1415926;
?>
```

Ha nem adunk indexet, akkor automatikusan a tömb végéhez fűződik az elem, azaz az előző két példa teljesen megegyezik.

Természetesen használhatunk többdimenziós tömböket is.

```
<?php
    $a[1][2] = "első sor második eleme";
    $a[1][2][3][4] = "ez egy négydimenziós tömb";
?>
```

A tömb elemekkel való feltöltésére két mód is kínálkozik. Az egyik a szokásos, megszokott változat:

```
<?php
    $a[1] = "alma";
    $a[2] = "körte";
    $a[3] = "barack";
    $a[4] = "szilva";

    echo $a[3];
?>
```

Míg a másik sokkal egyszerűbb, amolyan PHP-s módszer:

```
<?php
    $a = array ("alma", "körte", "barack", "szilva");
    echo $a[3];
?>
```

Vigyázat, a két módszer mégsem egyenértékű! A végeredmény bizony más. Első esetben a barack lesz a megjelenő gyümölcs, míg a második példában a szilva. Ennek

oka, hogy ha nem adok meg indexet, a PHP a nulladik (0.) elemtől kezdi el a tömb feltöltését. Vigyázzunk vele, érdemesebb, ha mi is a nulladik elemtől kezdjük a számozást. S nem utolsósorban takarékosabb!

Most pedig egy olyan pozitívumát ismerhetjük meg a PHP tömbkezelésének, amely nem található meg csak nagyon kevés nyelvben. Ez az úgynevezett asszociatív tömbök használata. Ez annyit jelent, hogy a tömb indexe helyén nem szám, hanem egy karakteres azonosító szerepe. Példával talán egyszerűbb lesz:

```
<?php
    $a[1] = "piros ";
    $a["gyümölcs"] = "alma";

    echo $a[1];
    echo $a["gyümölcs"];
?>
```

Mondhatnánk a "gyümölcsödik" elem az alma. Meglátjuk, milyen hasznos lesz a későbbiekben, le sem tudunk majd szokni róla. Ha asszociatív tömböt használunk, elemeinek megadása egy kicsit változik:

```
<?php
    $a["szín"]        = "piros";
    $a["íz"] = "édes";
    $a["forma"]       = "gömbölyű";
    // Ez volt a régi forma

    $a = array(
         "szín"       => "piros",
         "íz"         => "édes",
         "forma"      => "gömbölyű"
         );
    // Ez pedig az új módszer
?>
```

Látjuk, hogy a hozzárendelés a "=>" jelsorozattal történik. Most pedig lássunk egy példát, amely ötletesen bemutatja a módszer előnyét:

```
<?php
  $a = array(
      "alma"      => array(
          "szín"  => "piros",
          "íz"    => "édes",
          "forma" => "gömbölyű"
      ),
      "narancs"   => array(
          "szín"  => "narancssárga",
          "íz"    => "fanyar",
          "forma" => "gömbölyű"
      ),
      "citrom"    => array(
          "szín"  => "sárga",
          "íz"    => "savanyú",
          "forma" => "gömbölyded"
```

```
    )
);

echo $a["narancs"]["íz"];

?>
```

Mit is csinál a fenti példa? Próbáljuk megfejteni működését, próbáljuk begépelni, letesztelni, megváltoztatni, s újra tesztelni. Vigyázzunk, hova teszünk pontosvesszőt (ugye emlékszünk, csak a kifejezés végére...), illetve vesszőt. Sok sikert!

Típuskonverziók

A PHP alapértelmezésben nem támogatja a konkrét típusdeklarációt, azaz egy változó típusát csupán tartalma határozza meg:

```
$a = "0";                       // $a típusa karakteres, értéke "0"
$a++;                           // $a típusa karakteres, értéke "1"
$a += 1;                        // $a típusa egész, értéke 2
$a = $a + 1.3;                  // $a típusa lebegőpontos, értéke 3.3,
                                // mivel egyik összetevője szintén
                                // lebegőpontos
$a = 5 + "10 kicsi indián"; // $a típusa egész, értéke 15
```

Természetesen lehetőség van egyértelmű konverzióra is. Ez nagyon hasonlít a C-re, azaz a konvertálandó típus nevét zárójelbe írjuk a változó elé:

```
$a = 10;                        // $a egész típusú
$b = (double) $a;               // $b lebegőpontos
```

Lehetőségeink:

- (int), (integer) - egész konverzió
- (real), (double), (float) - lebegőpontos (double) konverzió
- (string) - string konverzió
- (array) - tömbbé konvertál
- (object) - objektum típusúra konvertál

Gyakorlati alapproblémák

A HTML, javascripttel keverve, továbbá egy is CSS-sel (Cascading Style Sheets – stíluslapok) megfűszerezve, csodákra képes. A piacon egyre inkább magára maradó Internet Explorer új 5.5-ös verziója már eléggé stabil, gyors ahhoz, hogy ne a böngésző, az operációs rendszer, hanem a vonalsebesség legyen a gyenge pontja a rendszernek. Ha valóra válnak azok a tervek, amelyek a jelenlegi internetes vonalak sebességét a többszörösére emelik, ez sem fog útban állni. S miért ne válhatna valóra? Az egyedüli, amiért aggódom, az Interneten megtalálható információk minősége. Egy felmérés szerint az internetes oldalak 70%-a "szemét", azaz információértéke majdnem nullával egyenlő. Nagy része valóban az alábbi sémára épül: "X.Y. vagyok, tizenx éves, itt-meg-itt lakom, imádom az alábbi énekeseket: blablablabla…, nem szeretem őket: blablabla. Itt láthatod tavaly nyáron készült képeimet a Balatonról (Görögországból, sítáborból). Ha teszik, ide írj.". Ehhez tökéletesen elegendő a HTML.

Mikor kell mégis olyan fegyverhez nyúlnunk, mint például a PHP. Miért több, mint a javascript? Tömböket az is tud kezelni, a változók és a meglevő funkciók szintén elégségesek interaktív weblapok elkészítéséhez. Azért egy óriási különbség mégis van: a javascript forráskódja letöltődik a felhasználó gépére, tehát bárki számára hozzáférhetővé válik. Éppen ebből következik, hogy fileműveletekre nem is gondolhatunk. Képzeljük el, hogyan adnánk hozzáférési jogokat, ha a jelszavak listájához bárki hozzáférhetne. Az adatkezelés algoritmusa végtelenül egyszerű. A felhasználó megad néhány bemenő adatot (input), végrehajtunk a bevitt adatokon valamilyen műveletet, vagy egy meglevő adatbázisban keresést végzünk az input alapján (query), végül valamilyen kimenetet (output) produkálunk. Az input beviteléhez és az output megjelenítéséhez a HTML lehetőségeit fogjuk használni, viszont az adatfeldolgozást nem bízhatjuk rá. Ezért a PHP a felelős. Egyelőre elégedjünk meg annyival, hogy a PHP ugyanúgy képes szöveges állományok kezelésére, mint régi dBase adatállományaink manipulálására, de erejét az SQL adatbázisok használatai során mutatja meg. Csak hogy a leggyakoribbakat említsük. Adabas D, Empress, FilePro, Informix, Interbase, mSQL, MySQL, Oracle, PostgreSQL, Solid, Sybase, Velocis, Unix dbm. Nem feladatom méltatni a régi adatkezelő rendszerekkel szemben az SQL előnyeit, csak annyit mondhatok, hogy nem hiába használják a legnagyobb rendszerek

kiszolgálásához az SQL-t. Ezekhez a kiszolgálókhoz való csatlakozást a PHP-hez letölthető DLL állományok segítségével tudjuk megvalósítani. Szerencsénkre a legelterjedtebb SQL kiterjesztést készen kapjuk a PHP-hez (a 4-es verzióba már készen beépítették, a PHP3-hoz a php3_mysql.dll állomány szükségeltetik). Ez a MySQL. Sokak szerint ez az az SQL implementáció, amely leginkább illeszkedik a PHP-hez. Most, ha visszalapozunk az installálás fejezetének utolsó lapjaihoz, érthetővé válhat, miért van szükségünk a mysqld.exe nevű állományra. Ez lesz az SQL szerver, ehhez csatlakozunk PHP-ból, amikor adatbáziskezeléssel foglalkozunk majd. Nem is oly' sokára, ez is bekövetkezik.

Az adatkezelésről általában

Mondottam, hogy az adatok bevitelét és a kimenetet HTML-ben fogjuk produkálni, s csak a feldolgozást végezzük PHP-ben. Ez bizony azt is jelenti, hogy vége a kényelmes, WYSIWYG honlapszerkesztéseknek, mondhatnám, sutba dobhatjuk FrontPage, DreamWeaver, Adobe PageMill és ehhez hasonló, rendkívül hatékony HTML editorainkat. Egyelőre ugyanis még egyikük sem tudja kezelni a PHP betéteket tökéletes módon. Marad a régi bevált módszer, a kézzel történő szerkesztgetés. Tudom, most sokan felhördülnek, hogy micsoda mazochista technikákat alkalmazunk céljaink eléréséhez, de sajnos (dehogy sajnos!) más lehetőségünk nincs. Azok kedvéért, akik nincsenek tökéletesen képben a HTML-t illetően, megengedjük, hogy használják editoraikat olyan részek elkészítésében, amelyek még bizonytalanul mennek. Hogy beilleszthessék eme részeket a PHP betétek közé, ki kell metszeni a szükséges részletet a szerkesztőprogram forrásablakából, s nem árt, ha tudjuk mit, és miért teszünk. Egyben biztos vagyok: egy jól működő, jól használható, komplex PHP-s feldolgozóprogram elkészítése után mindenki profi HTML kóder lesz!

Ismételjük át, amit a HTML adatbeviteli lehetőségeiről tudnunk kell. HTML-ben úgynevezett űrlapokat hozhatunk létre a <FORM> objektum beszúrásával. Ez fogja egybe a különböző beviteli formákat, mint például a szöveges mező, vagy a checkbox. Ezekről később szólunk. A <FORM> objektumnak különböző attribútumai (jellemzői, paraméterei) lehetnek:

```
<FORM
        target = "ablaknév"
        action = "végrehajtó script neve"
        method = GET | POST
        name = "űrlap neve"
        onReset = "reset_rutin"
        onSubmit = "submit_rutin ">
```

Nem kell megijedni, megmagyarázom őket:

- target: megadhatom, hogy az eredmény melyik ablakban/frameben jelenjenek meg.
- action: a Rögzítés (Submit) gomb megnyomása után ennek a scriptnek fogja elküldeni az űrlap adatait a böngésző. Ha nem adjuk meg, akkor a HTML állomány önmaga hívódik meg! Ez fontos!

- method: ez határozza meg, hogy az adatok hogy kerülnek a scripthez. Ha GET, akkor az űrlap adatai az URL után csapódnak egy kérdőjellel (pl. http://www.sajatdomain.com/?mezo1=ertek1&mezo2=ertek2). Láthatjuk, hogy a különböző űrlapmezők az "&" jellel vannak elválasztva. Ha POST, akkor a mezők értékeit a standard inputon keresztül kapják meg a scriptek. Szerencsére PHP-ben ezzel nem kell törődnünk, leginkább a POST metódust fogjuk használni.
- name: megadja az űrlap nevét. Ez csak esetleges javascript betéteinknél jelent majd hivatkozási alapot.
- onReset: ha megnyomjuk a FORM-hoz tartozó Törlés gombot, akkor az itt megadott javascript függvény kerül végrehajtásra, mielőtt törlődnek az adatok az űrlapról.
- onSubmit: Mint az előző, csak a Rögzítés gombra vonatkozólag. Ez nagyon hasznos lehet, ha nem akarjuk addig átküldeni az adatokat a scriptnek (action), amíg egy megadott feltétel nem teljesül.

Az űrlapot a </FORM> zárótaggel kell zárni. Közé tehetjük az űrlapmezőket. Hogy milyen lehetőségeink vannak, tekintsük át őket.

Beszúrhatunk egyszerű TEXT (szöveges) mezőt:

```
<INPUT TYPE = "TEXT"
        name = "név"
        value = "kezdőérték"
        size = "méret"
        maxlength = "hossz">
```

- name: a szöveges mező neve. Rendkívül fontos szerepet kap a következőkben.
- value: ha kezdőértéket adunk a mezőnek, itt megtehetjük. Ha nem adjuk meg, a szöveges mező üres lesz.
- size: a mező szélessége karakterekben mérve.
- maxlength: a mezőbe írható karakterek maximális száma.

Lehetőségünk van többsoros szövegablak (TEXTAREA) létrehozására:

```
<TEXTAREA
        name = "név"
        rows = "sorok száma"
        cols = "oszlopok száma">

KEZDŐSZÖVEG

</TEXTAREA>
```

- name: a többsoros beviteli mező neve. Később ezzel hivatkozunk rá.
- rows: megjelenítendő sorok száma. Ha nem fér ki a beírt szöveg, görgetősávok használatával lehet a kilógó részeket megtekinteni.
- cols: Ugyanaz, mint rows, csak oszlopokra.

Ugye látjuk, hogy itt zárótag is kötelező?

A következő elem a PASSWORD, azaz jelszó objektum. Nem ragozom, hiszen csak deklarációjában különbözik a TEXT beviteli mezőtől, no meg persze abban, hogy a beírt szöveg minden karaktere helyett "*" jelenik meg. Jelszavak beolvasására jól alkalmazható:

```
<INPUT TYPE = "PASSWORD"
        name = "név"
        value = "kezdőérték"
        size = "méret"
        maxlength = "hossz">
```

Legyen a következő a SELECT objektum. Ez egy olyan választhatóan többsoros, szükség esetén gördíthető listát képez, amelyből a felhasználó egy vagy több elemet kiválaszthat. Formátuma a következő:

```
<SELECT
        name = "név"
        size = "méret"
        MULTIPLE>

        <OPTION VALUE = "érték" SELECTED> szöveg
        <OPTION VALUE = "érték" SELECTED> szöveg
        ...

</SELECT>
```

- name: a lista hivatkozási nevét adhatjuk meg.
- size: a látható sorok száma

- MULTIPLE: ha egyszerre több választás is megengedett, akkor ezt a MULTIPLE kulcsszóval jelezhetjük. Ilyenkor a szokásos módon, a CTRL, SHIFT billentyűk segítségével tudunk többszörös választást eszközölni.
- <OPTION…: láthatjuk, hogy az előbb lezártuk a SELECT taget. Jöhetnek a listából választható elemek leírásai. Egy elemnek három tulajdonsága lehet. Első az értéke, ez megy át a scripthez. A második a SELECTED opció, amely ha létezik, akkor az aktuális elem kiválasztott elem, míg ha nem írjuk be, akkor nem lesz kiválasztva indításkor. A harmadik az aktuális elemhez tartozó, képernyőn (listában) megjelenő szöveg. Egy példa:

```
<OPTION VALUE=1>Első sor
<OPTION VALUE=2 SELECTED>Második sor
```

  A példa alapján két értéke lesz a listának, a második érték lesz alapértelmezettként kiválasztva. Ha változtatás nélkül megnyomjuk a Rögzítés gombot, akkor a scriptnek a második érték megy át, pontosabban ennek a sornak a VALUE tagja (2).

A végén ezt se feledjük lezárni </SELECT> taggel.

Alapértelmezésben a SELECT egy legördülő lista, amelynek tetején az éppen kiválasztott elem található. Gondoljunk csak például a drive (meghajtó) választó listára a Windows bármelyik töltés/mentés dialógusablakában. Ha viszont a fent említett paraméterek közül a SIZE vagy a MULTIPLE bármelyikét megadjuk, akkor kapjuk az egyszerű listát, amelyből többet is ki lehet választani, és ez már több soros is lehet.

Következő elemünk a lefordíthatatlan RADIOBUTTON nevet viseli. Talán rádiógombnak lehetne fordítani? Ezek azok a kör alakú gombok, amelyek közül mindig egy lehet kiválasztva. Formátuma a következő:

```
<INPUT TYPE = "RADIO"
        name = "név"
        value = "érték"
        CHECKED>
```

- name: a már megszokott hivatkozási név. Ha több rádiógomb csoportot is létrehozunk, akkor a következő csoportnak más nevet kell adnunk. Minden csoportban egy gomb lehet bekapcsolva.

- value: ha be van kapcsolva a rádiógomb, akkor ezt az értéket küldi el a scriptnek a Rögzít gomb megnyomása után.
- CHECKED: szintén egy olyan kulcsszó, amely elhagyható. Ha viszont meg van adva, akkor alapértelmezésben a rádiógomb be lesz kapcsolva. Természetesen mindig csak EGY gomb lehet bekapcsolva. Ha többet is meghatározunk, az utolsó CHECKED lesz az érvényes.

A CHECKBOX objektum (Ellenőrző doboz? Maradjunk a checkboxnál…) egy kétállású kapcsolót reprezentál. Lehetőségei megegyeznek a rádiógombnál leírtakkal, azt leszámítva, hogy itt több kiválasztott gomb is lehet.

Végül az utolsó objektumtípus, amit űrlapon elhelyezhetünk a nyomógomb, azaz a BUTTON. Ennek három fajtáját különbözteti meg a HTML. Első, amit megemlítünk a valódi BUTTON. Definíciója a következőképpen történik:

```
<INPUT TYPE = "BUTTON"
        name = "név"
        value = "felirat"
        onClick = "click_rutin">
```

- name: a gomb hivatkozási neve.
- value: a gomb felirata
- onClick: a gombra történő kattintáskor az itt megadott javascript rutint hajtja végre.

A következő gombtípus a SUBMIT gomb, azaz magyarul "Rögzít"-nek fordíthatjuk. Ezen gomb megnyomásakor a böngésző elküldi az űrlap tartalmát az ACTION részben megadott scriptnek:

```
<INPUT TYPE = "SUBMIT"
        name = "név"
        value = "felirat">
```

Hivatalosan itt is van onClick eseménykezelő, de ritkán van rá szükség, ezért itt, most nem tárgyaljuk.

Az utolsó, RESET gomb az a bizonyos "Törlés" gomb, amely alaphelyzetbe állítja az űrlap minden mezőjét, azaz mindegyik a módosítások előtti (kezdeti) állapotba kerül. Definíciója:

```
<INPUT TYPE = "RESET"
        name = "név"
        value = "felirat">
```

Nos, ennyi lenne az űrlapok kezelésére vonatkozó ismétlésünk. Ha valaki többre kíváncsi, érdemes átlapozni a legutolsó HTML specifikációt, amelyet a http://www.w3.org címen mindig megtalálunk. Másik lehetőség, hogy előveszünk egy magyar nyelvű kézikönyvet, kikeressük a megfelelő fejezeteket, kipróbáljuk a példákat és addig nem állunk fel a gép mellől, amíg tisztán nem látjuk a témakör összes aspektusát. Ne feledjük: befektetés nélkül nincs eredmény! Nos, ennyi bölcsesség után térjünk rá következő fejezetünkre, amelyből megtudhatjuk, hogy hogyan mennek át űrlapunk mezői PHP scriptjeinkbe.

Adatátvitel a HTML és a PHP között

A cím egy kicsit megtévesztő. A HTML egy lapleíró nyelv, nincs szüksége semmilyen változóra, hiszen statikus adatokkal dolgozik. Az előző mondatom igaz is, meg nem is. A HTML valóban nem használ a szó szoros értelmében vett változókat, viszont minden, a weblapon található objektum sok-sok tulajdonsággal rendelkezik, amelyet nagyrészt magam szabhatok meg. Természetesen főként az űrlapok objektumaira gondolok. Hogy világosabbá váljék gondolatmenetem, készítsünk egy egyszerű űrlapot, amely bekéri a felhasználó nevét és születési évét. Valami ilyesmit fogunk írni:

```
<HTML>
    <FORM name=adatok>
        Neved:<br>
            <input type=text name=nev><br>
        Születési éved:<br>
            <input type=text name=szev><br>

        <br>
        <input type=submit name=submit value="Mehet">
    </FORM>
</HTML>
```

Az űrlap neve "adatok", rajta két mező ("nev" és "szev" névvel). Szükségünk lehet egy "Mehet" gombra is, amellyel átadjuk az űrlap tartalmát a <FORM> tag action részében megadott scriptnek. Aki figyelt, észrevette, hogy én ezt a részt kihagytam. Beszéltünk róla, hogy ha üresen hagyom, akkor a HTML oldal önmagát fogja visszahívni a Mehet (Submit) gomb lenyomása után. Ezt ki is próbálhatjuk. Ha begépeljük a fenti forrást, töltsük ki a két mezőt, majd nyomjuk le a "Mehet" gombot. A lap újrahívódik, ami azt jelenti, hogy törli a két mező tartalmát. Viszont ezzel még nincs vége! Nézzük csak a címsort! Egy kicsit megváltozott:

http://localhost/?nev=Joc&szev=1971&submit=Mehet

Ugye, nem kell mondanom, hogy mindenkinek más lesz a neve és a születési éve? Ha visszalapozunk a FORM leírásához, nézzük meg újból a "method" paraméterét. Ott azt mondtuk, ha GET, akkor egy kérdőjellel fűzi hozzá az URL-hez az űrlap mezőinek értékeit. Ez az! A method attribútum kihagyása esetén ugyanis az alapértelmezett metódus lép életbe, ez pedig a GET. Mielőtt tovább mennénk, nézzünk rá erre a sorra, s elemezzük ki, mit látunk.

Odáig biztosan mindenkinek tiszta, hogy http://localhost/. Ez ugye a szerverünk neve (illetve URL címe). Ezek után a fent említett kérdőjel jön, majd ez egyes űrlapmezők neve és értéke:

```
nev = Joc
szev = 1971
submit  = Mehet
```

Tényleg három objektumot helyeztünk el az űrlapra, két szövegmezőt, s egy nyomógombot. Fontos, hogy a nyomógomb értékét (value) mi adtuk meg, ezt a felhasználó nem tudja megváltoztatni, a két szövegmező viszont szabad préda, azt ír bele, amit akar. Ha akarja, üresen hagyja, de olyat is elkövethet, hogy a születési év mezőbe számokat ír. Sebaj, megoldjuk ezen problémákat is.

Folytassuk tovább eszmefuttatásunkat, amelyet a metódusoknál hagytunk abba. Ha ezt POST-ra módosítom, nem fog megjelenni a címsorban semmi, de sebaj, a PHP egyszerű megoldást kínál, ennek lekezelésére. Az egyszerűség kedvéért megadom a megoldást, s később jön a magyarázat:

```
<?php

    if (getenv(REQUEST_METHOD)==="POST")
    {

        echo $nev;

    } else

    {

?>

<HTML>
    <FORM name=adatok method=post>
        Neved:<br>
            <input type=text name=nev><br>
        Születési éved:<br>
            <input type=text name=szev><br>

        <br>
        <input type=submit name=submit value="Mehet">
    </FORM>
</HTML>


<?php } ?>
```

Mentsük el a fenti forrást index.php néven (ugye mindenki tudja, hogy a C:/web/root/ könyvtárba mentünk minden anyagot?), majd hívjuk be a böngészőt, s

kérjük le a http://localhost URL-t. Ha minden tökéletes (a webszerver is el van indítva), akkor egy üres űrlapnak kell megjelennie a weblapon. Kövessük végig a folyamatot, értelmezzük az egyes sorokat.

```
<?php
    if (getenv(REQUEST_METHOD)==="POST")
    {
        echo $nev;
    } else
```

Nézzük meg egyelőre idáig. Az első sor egy szokásos PHP nyitás, azt jelenti, hogy most PHP nyelvű rész következik. Utána egy vizsgálatot végzünk: ha a REQUEST_METHOD nevű környezeti változó értéket POST, akkor… De nem siessünk. Szóval az "if" a többi nyelvhez hasonlóan itt is a logikai vizsgálatot végzi el. Maga a kiértékelendő feltétel zárójelben következik. Használunk benne egy getenv() függvényt, amely környezeti változók tartalmát olvassa be. Ezek közül egy a REQUEST_METHOD. Ez a FORM method nevű attribútumát adja át a PHP-nek. Az egyenlőség-vizsgálat néhány nyelvtől eltérően nem "=" hanem "==", azaz dupla egyenlőségjel. Kezdők egyik kedvenc hibája, hogy feltételek megfogalmazása estén szimpla egyenlőségjelet használnak. Nem helyes, ugyanis az "=" az értékadás jele!

Bezártam a feltételhez tartozó zárójelet, amely után a logikai "IGAZ" ág következik. A PHP ezt kapcsos-zárójelek közé teszi. Jelen pillanatban egy sor van közöttük, ilyen esetekben nem kötelező használni őket, de inkább szokjuk meg, strukturáltabbá és áttekinthetőbbé teszi programunkat. Szóval a kapcsos-zárójelek között mit művelünk? Kiírunk valamit, hiszen az "echo" parancs a kiírást jelenti PHP-ben. Kiírunk egy $nev nevű változót. Mi ez a változó, honnan jött? Remélem, sokan kitalálták. Ez bizony az űrlapunk első szövegmezője, amit "nev" névvel illetünk. Ezért jegyeztem meg a FORM objektumainak leírásakor, hogy adjunk nevet az egyes mezőknek. Mivel a PHP-ben minden változó "$" jellel kezdődik, itt sincs kivétel. Tehát kiírjuk az első szövegmező tartalmát, már ha egyáltalán kedves felhasználónk kitöltötte. Ezek után egy "else" kulcsszó következik, ami a logikai "HAMIS" ág bevezetője. Lássuk, mit tartalmaz ez az ág.

```
    {

?>

<HTML>
    <FORM name=adatok method=post>
        Neved:<br>
            <input type=text name=nev><br>
        Születési éved:<br>
            <input type=text name=szev><br>

        <br>
        <input type=submit name=submit value="Mehet">
    </FORM>
</HTML>


<?php } ?>
```

Figyeljük meg, hogy a "HAMIS" ág is ugyanúgy kapcsos-zárójelek közé teendő, mint az "IGAZ" ág. Csakhogy egy érdekes konstrukció következik. Amint megnyitom a "HAMIS" ágat, máris kilépek a PHP környezetből, visszalépek HTML-be. Sebaj, a PHP követi ezt, s tudja, hogy a most következő HTML rész is a "HAMIS"ághoz tartozik, egészen addig, amíg egy záró kapcsos-zárójelre nem talál. Mivel a zárótag szintén PHP nyelvű, a legutolsó sorban nyitnom kellett egy PHP részt, s ebben helyeztem el a záró kapcsos-zárójelet. Közötte pedig az előzőleg már megtárgyalt űrlapot írtam le, vigyázva arra, hogy a küldési metódus POST legyen. (Egy megjegyzés: a HTML részben mindegy, hogy nagybetűvel, vagy kisbetűvel írom a paramétereket. Ugyanez viszont nem érvényes a PHP-ben. Már egyszer említettem, de nem árt újra felelevéníteni: a PHP megkülönbözteti a kis/nagybetűvel írt változókat egymástól.)

Kövessük csak a lap betöltődésének és értelmezésének folyamatát! A vezérlés először a REQUEST_METHOD vizsgálatára fut rá. Mivel a lap első betöltésekor ennek nem POST az értéke (csak akkor lesz POST, ha megnyomom a "Mehet" gombot), továbbfut a "HAMIS" ágra. Ez kiírja az űrlapot, s vége. Amint a felhasználó megnyomja a "Mehet" gombot, az űrlap újrahívja az oldalt. Immár van értéke a REQUEST_METHOD-nak, méghozzá pont POST, tehát az "IGAZ" ág fog végrehajtódni, a "HAMIS" ág kimarad. Az oldal letöltése a "HAMIS" ág utáni részen folytatódik, de mivel jelen helyzetben itt már nincs semmi, kész az oldal.

Gyakorlásképpen próbáljuk megoldani, hogy a gomb megnyomása után a lap köszöntse nevén a felhasználót, és írja ki, hány éves. Valami ilyenre gondoltam:

Szia, kedves Tamás! Ebben az évben töltöd be 20. életéved.

Egy megoldási lehetőség:

```
if (getenv("REQUEST_METHOD")=="POST")
    {
        $kor = 2000 - $szev;
        echo "Szia, kedves $nev! Ebben az évben töltöd $kor. életéved.";
    }
```

Csak az "IGAZ" ágat írtam le, hiszen módosítani csak itt kellett. Azért nézzük át: bevezettem egy $kor nevű változót. Ha a jelen évből kivonom a születési évet, akkor az aktuális évben betöltött évek számát kapom. Ez egyértelmű. A kiírás viszont sokak számára érdekes lehet, kihasználtam, hogy az idézőjelek közé változókat a PHP kiértékeli, azaz a $nev és a $kor változók helyett azok értékeit fogja behelyettesíteni.

Tudom, elkészíthettük volna szebben is. Például kiemelhetnénk a felhasználó nevét félkövérrel:

```
echo "Szia, kedves <b>$nev</b>! Ebben az évben töltöd $kor. életéved.";
```

Nos, jó példa, milyen hatékonyan lehet keverni a PHP-n belül a HTML forrást. Ha arra támad kedvünk, a teljes HTML kódot kiírhatjuk a PHP segítségével. Lássuk az előző programot, csak PHP nyelven:

```
<?
    if (getenv("REQUEST_METHOD")=="POST")
    {
        $kor = 2000 - $szev;
        echo "Szia, kedves <b>$nev</b>! Ebben az évben töltöd $kor. életéved.";
    } else
    {
    echo "<HTML>";
    echo "   <FORM name=adatok method=post>";
    echo "            Neved:<br>";
    echo "                    <input type=text name=nev><br>";
    echo "            Születési éved:<br>";
    echo "                    <input type=text name=szev><br>";
    echo "            <br>";
    echo "            <input type=submit name=submit value=\"Mehet\">";
    echo "   </FORM>";
    echo "</HTML>";
    }
?>
```

Hogy miért nem szoktuk ezt a megoldást alkalmazni? Kérjük csak a forráskódot, s rögtön megértjük. Persze, van megoldás: ha minden echo sort egy "\n" (soremelés) jelsorozattal fejezünk be, máris szebb lesz a látvány. De vajon megéri-e?

Ha valaki figyelmesen átolvasta az előző kódot, észrevehette, hogy a "Mehet" gomb deklarálásánál az idézőjeleket kicseréltem. Ennek magyarázata, hogy az "echo" parancs paraméterét szintén idézőjelek közé kellett tennem. Ilyenkor több megoldás is kínálkozik. Az egyik, hogy valamelyik idézőjelpárt aposztrófpárra cserélem. A másik, hogy Escape-szekvenciát alkalmazok (\"), amit már összefoglaltam egy táblázatban:

```
echo "      <input type=submit name=submit value=\"Mehet\">";
echo "      <input type=submit name=submit value='Mehet'>";
echo '      <input type=submit name=submit value="Mehet">';
```

A fenti példák mindegyike helyes, a lényeg, hogy kerüljük ki az idézőjelen belüli idézőjel problémáját, mert egy kövér "Parse error…" hibaüzeneten kívül mást nem fogunk kapni.

A fenti egyszerű feladatra egy nagyon egyszerű megoldást adtunk. Bonyolultabb weblapokon sokszor előfordul, hogy nem egy, de több űrlap található egy lapon. Ha továbbra is a REQUEST_METHOD lekérdezése alapján döntjük el, hogy volt-e űrlapküldés, akkor gondjaink támadhatnak. Honnan tudjuk, melyik űrlap gombját nyomták meg? Erre is van megoldás. Térjünk vissza oda, amikor megnéztük, milyen értékeket ad át az űrlap a scriptnek. Átadja a szövegmezők tartalmát, és átadja a nyomógomb értékét is. A szövegmezőket a felhasználó adja meg, a gomb értékét viszont én írhatom be a forrásba. Ez fontos szempont, hiszen úgymond felhasználó-független. Ezt kihasználva, a következőképpen módosíthatom az előző program forrását:

```
if (isset($submit))
    {
        $kor = 2000 - $szev;
        echo "Szia, kedves <b>$nev</b>! Ebben az évben töltöd $kor. életéved.";
    }
```

Az új infromáció az első sor. A $submit változó a "Mehet" gomb neve. Vigyázzunk, hogy ugyanúgy írjuk le, ahogy a HTML forrásban használtuk, jelen helyzetben végig

kisbetűvel. Az isset() függvény azt vizsgálja le, hogy a zárójelben szereplő változónak van-e tartalma, tehát létezik-e egyáltalán a változó. Ha egy következő űrlapon az "Elküld" gombot máshogy nevezzük el (name=…), akkor lehetőségünk nyílik arra, hogy megkülönböztessük a küldő űrlapokat egymástól. Lássunk erre egy példát:

```
if (isset($gomb1))
    {
        echo "Az első űrlapról jött a hívás.";
    } else
if (isset($gomb2))
    {
        echo "Második űrlapról jött a hívás.";
    } else
// jöhet az űrlapok leírása, mivel egyik gombot sem nyomták meg.
```