III. RÉSZ

Munka a PHP-vel

9. ÓRA

Űrlapok

A könyv eddigi példáiban egy nagyon fontos dolgot nem láttunk. Létrehoztunk változókat és tömböket, készítettünk és meghívtunk különböző függvényeket, különféle objektumokkal dolgoztunk, eddigi ismereteink azonban értelmetlenek, amíg a felhasználó által megadott adatokat kezelni nem tudjuk. Ebben az órában ezt a témakört tekintjük át.

A Világhálón alapvetően a HTML űrlapokon keresztül áramlik az információ a felhasználó és a kiszolgáló között. A PHP-t úgy tervezték, hogy a kitöltött HTML űrlapokat könnyen fel tudja dolgozni.

Ebben az órában a következőket tanuljuk meg:

- Hogyan férjünk hozzá a környezeti változókhoz és hogyan használjuk azokat?
- Hogyan férjünk hozzá az űrlapmezőkben megadott információkhoz?
- Hogyan dolgozzunk az űrlapok több elem kiválasztását engedélyező elemeivel?

- Hogyan készítsünk olyan kódot, amely egyszerre tartalmazza a HTML űrlapot és az ezt kezelő PHP programot?
- Hogyan mentsük az állapotot rejtett mezőkkel?
- Hogyan irányítsuk a felhasználót új oldalra?
- Hogyan készítsünk fájlfeltöltő HTML űrlapot és hogyan írjunk olyan PHP programot, amely kezeli ezt?

Globális és környezeti változók

Mielőtt elkészítenénk első igazi űrlapunkat, kis kitérőt kell tennünk, hogy újra áttekintsük a globális változókat. A globális változókkal először a függvényekről szóló hatodik órában találkoztunk. A globális változók azok a változók, amelyeket a program legfelső szintjén, azaz a függvényeken kívül vezettünk be. Minden függvény számára elérhető a beépített `$GLOBALS` nevű tömb. A `$GLOBALS` tömb használatát láthatjuk a 9.1. példában, amelyben egy ciklussal kiírjuk programunk összes globális változóját.

9.1. program A $GLOBALS tömb elemeinek kiírása

```
 1: <html>
 2: <head>
 3: <title>9.1. program A $GLOBALS tömb elemeinek
           kiírása</title>
 4: </head>
 5: <body>
 6: <?php
 7: $user1 = "Judit";
 8: $user2 = "István";
 9: $user3 = "János";
10: foreach ( $GLOBALS as $kulcs=>$ertek )
11:       {
12:       print "\$GLOBALS[\"$kulcs\"] == $ertek<br>";
13:       }
14: ?>
15: </body>
16: </html>
```

Három változót vezettünk be és a $GLOBALS tömb elemein végiglépkedve kiírtuk a változók neveit és értékeit a böngészőben. Láthatjuk az általunk megadott változókat, de a program ezeken kívül másokat is kiírt, a PHP ugyanis automatikusan bevezet néhány globális változót, amelyek a kiszolgáló és az ügyfél környezetét írják le. Ezeket a változókat környezeti változóknak nevezzük. A kiszolgálótól, a rendszertől és a beállításoktól függően különféle környezeti változók létezhetnek, ezek ismerete rendkívül fontos. A 9.1. táblázatban a leggyakoribb környezeti változókat soroltuk fel. A környezeti változók értéke közvetlenül vagy a $GLOBALS tömb részeként érhető el.

9.1. táblázat Környezeti változók

Változó	*Tartalma*	*Példa*
$HTTP_USER_AGENT	A böngésző neve és változatszáma	Mozilla/4.6 (X11;I; Linux2.2.6-15apmac ppc)
$REMOTE_ADDR	Az ügyfél IP címe	158.152.55.35
$REQUESTED_METHOD	A kérelem módja (GET vagy POST)	POST
$QUERY_STRING	A GET kérelmeknél az URL-hez kapcsolt kódolt adat	nev=janos&cim=ismeretlen
$REQUEST_URI	A kérelem teljes címe a lekérdező karaktersorozattal	/php-konyv/urlapok/9.14.program.php?nev=janos
$HTTP_REFERER	Az oldal címe, amelyről a kérelem érkezett	http://www.proba.hu/egy_oldal.html

A PHP létrehoz más környezeti változókat is.
Például a $GLOBALS["PHP_SELF"] az éppen futó program elérési útját adja meg. A szerző rendszerén az érték a következő volt:

```
/php-konyv/urlapok/9.1.program.php
```

A változó értéke közvetlenül is elérhető, $PHP_SELF néven. Ebben az órában még sokszor fogjuk használni ezt a változót. A HTML oldalt leíró és az űrlapot elemző PHP kódot gyakran tároljuk egy állományban. Az oldal nevének tárolásához a $PHP_SELF változó értékét a HTML FORM elemének ACTION paraméteréhez rendeljük.

A $GLOBALS tömb ezenkívül még sok másra is használható.

Adatok bekérése a felhasználótól

Jelenleg a HTML és PHP kódot külön állományban tároljuk. A 9.2. példában egy egyszerű HTML űrlap kódját láthatjuk.

9.2. program Egy egyszerű HTML űrlap

```
 1: <html>
 2: <head>
 3: <title>9.2. program Egy egyszerű HTML űrlap</title>
 4: </head>
 5: <body>
 6: <form action="9.3.program.php" method="GET">
 7: <input type="text" name="felhasznalo">
 8: <br>
 9: <textarea name="cim" rows="5" cols="40">
10: </textarea>
11: <br>
12: <input type="submit" value="rendben">
13: </form>
14: </body>
15: </html>
```

Egy HTML űrlapot hoztunk létre, amely egy "felhasznalo" nevű szövegmezőt, egy "cim" nevű szövegterületet, és egy gombot tartalmaz. Könyvünk nem foglalkozik részletesen a HTML ismertetésével, ha mégis nehéznek találjuk a példákat, olvassunk el egy, a HTML nyelvvel foglalkozó kötetet vagy számítógépes leírást. A FORM elem ACTION paramétere a 9.3.program.php fájlra mutat, amely feldolgozza az űrlapon megadott adatokat. Mivel az ACTION csak a fájl nevét adja meg, a HTML és PHP kódot tartalmazó fájloknak egy könyvtárban kell lenniük.

A 9.3. példaprogram kiírja a felhasználó által megadott információkat.

9.3. program A 9.2. példa űrlapjának feldolgozása

```
 1: <html>
 2: <head>
 3: <title>9.3. program A 9.2. példa űrlapjának
           feldolgozása</title>
 4: </head>
 5: <body>
 6: <?php
 7: print "Üdvözlet <b>$felhasznalo</b><P>\n\n";
 8: print "A címe:<P>\n\n<b>$cim</b>";
 9: ?>
10: </body>
11: </html>
```

Ez a könyv első olyan programja, amelyet nem hivatkozáson keresztül hívunk meg és nem közvetlenül a böngészőbe írjuk be a címét. A 9.3. példában található kódot az `9.3.program.php` fájlban tároljuk. A fájlban lévő kódot akkor hívjuk meg, amikor a felhasználó kitölti a 9.2. példában található űrlapot. A program a `$felhasznalo` és a `$cim` változók értékét írja ki, amelyek a HTML űrlap `"felhasznalo"` szövegmezőjének és `"cim"` területének értékét tartalmazzák. Látható, hogy a PHP 4 segítségével milyen egyszerűen kezelhetőek az űrlapok. Bármilyen megadott információ globális változóként hozzáférhető és a változók neve megegyezik a megfelelő HTML elem nevével.

Több elem kiválasztása a SELECT elemmel

Előző példánkban olyan HTML űrlap szerepelt, amely egy elemhez egyetlen érték hozzárendelését engedélyezte. Most megtanuljuk a `SELECT` elem kezelését, melynek segítségével több elem kiválasztását engedélyezhetjük. Ha a `SELECT` elemnek egyszerű nevet adunk:

```
<select name="termekek" multiple>
```

akkor a feldolgozó program csak az egyik kiválasztott elemhez fér hozzá.
Ha minden elemet el szeretnénk érni a programból, az elem neve után írjunk üres szögletes zárójeleket. Ezt láthatjuk a 9.4. példában.

9.4. program SELECT elemet tartalmazó HTML űrlap

```
 1: <html>
 2: <head>
 3: <title>9.4. program SELECT elemet tartalmazó HTML
           űrlap</title>
 4: </head>
 5: <body>
 6: <form action="9.5.program.php" method="POST">
 7: <input type="text" name="felhasznalo">
 8: <br>
 9: <textarea name="cim" rows="5" cols="40">
10: </textarea>
11: <br>
12: <select name="termekek[]" multiple>
13: <option>Ultrahangos csavarhúzó
14: <option>Tricorder
15: <option>ORAC AI
16: <option>HAL 2000
17: </select>
18: <br>
19: <input type="submit" value="rendben">
20: </form>
21: </body>
22: </html>
```

Az űrlapot feldolgozó programban a `SELECT` elemben kiválasztott adatok a `$termekek` tömbben találhatóak. Ezt mutatjuk be a 9.5. példában

9.5. program A 9.4. példában látott űrlap feldolgozása

```
 1: <html>
 2: <head>
 3: <title>9.5. program A 9.4. példában látott űrlap
           feldolgozása</title>
 4: </head>
 5: <body>
 6: <?php
 7: print "Üdvözlet <b>$felhasznalo</b><P>\n\n";
 8: print "A címe:<P>\n\n<b>$cim</b>";
 9: print "A következő termékeket választotta:<p>\n\n";
10: print "<ul>\n\n";
```

9.5. program (folytatás)

```
11: foreach ( $termekek as $termek )
12:       {
13:       print "<li>$termek<br>\n";
14:       }
15: print "</ul>";
16: ?>
17: </body>
18: </html>
```

Nem csak a SELECT elem engedélyezi több elem kiválasztását. Ha több jelölőnégyzetnek ugyanazt a nevet adjuk, ahol a felhasználó több elemet is kiválaszthat, csak az utolsó kiválasztott értéket kapjuk meg, de ha a nevet üres szögletes zárójelek követik, az adott nevű elemeket a PHP tömbként kezeli. A 9.4. példa SELECT elemét jelölőnégyzetekre cserélhetjük és ugyanazt a hatást érjük el:

```
<input type="checkbox" name="termekek[]" value="Ultrahangos
    ➥ csavarhúzó">Ultrahangos csavarhúzó<br>
<input type="checkbox" name="termekek[]"
    ➥ value="Tricorder">Tricorder<br>
<input type="checkbox" name="termekek[]" value=
    ➥ "ORAC AI">ORAC AI<br>
<input type="checkbox" name="termekek[]" value=
    ➥ "HAL 2000">HAL 2000<br>
```

Az űrlap minden mezőjének hozzárendelése egy tömbhöz

Az eddig megtanult módszerek mindaddig jól működnek, amíg programunk tudja, milyen mezőkkel dolgozik. Gyakran azonban olyan programra van szükségünk, amely az űrlap változásaihoz alkalmazkodik vagy egyszerre többféle űrlapot képes kezelni, anélkül, hogy keverné a mezők neveit. A PHP 4 globális változói erre a problémára is megoldást nyújtanak. Attól függően, hogy a kitöltött űrlap GET vagy POST metódust használt, elérhetővé válnak a $HTTP_GET_VARS vagy a $HTTP_POST_VARS változók. Ezek az asszociatív tömbök tartalmazzák az űrlap mezőinek neveit és a hozzájuk rendelt értékeket. A 9.6. példa azt mutatja be, hogyan listázható ki az űrlap minden mezője egy GET kérelem esetén.

9.6. program A $HTTP_GET_VARS tömb használata

```
 1: <html>
 2: <head>
 3: <title>9.6. program A $HTTP_GET_VARS tömb
           használata</title>
 4: </head>
 5: <body>
 6: <?php
 7: foreach ( $HTTP_GET_VARS as $kulcs => $ertek )
 8:       {
 9:       print "$kulcs == $ertek<BR>\n";
10:       }
11: ?>
12: </body>
13: </html>
```

A fenti kód a `GET` kérelmen keresztül érkezett paraméterek neveit és értékeit írja ki. Természetesen itt még nem kezeltük azt az esetet, amikor valamelyik átadott paraméter tömb. Ha a 9.4. példaprogrammal olyan HTML űrlapról érkező adatok feldolgozását végeztetjük, amelyben több elem kiválasztását engedélyező `SELECT` mező szerepel, valami hasonlót olvashatunk:

```
felhasznalo == Kiss Iván
cim == Budapest, Magyarország
termekek == Array
```

A termekek tömböt tartalmazza a `$HTTP_GET_VARS` tömb, de kódunk még nem tudja ezt kezelni. A 9.7. példaprogram ellenőrzi, hogy a paraméter típusa tömb-e és ha igen, kiírja a tömb elemeit.

9.7. program A $HTTP_GET_VARS tömb használata, ha tömböt tartalmaz

```
1: <html>
2: <head>
3: <title>9.7. program A $HTTP_GET_VARS tömb
          használata, ha tömböt tartalmaz</title>
4: </head>
5: <body>
6: <?php
```

9.7. program (folytatás)

```
 7: foreach ( $HTTP_GET_VARS as $kulcs => $ertek )
 8:       {
 9:       if ( gettype( $ertek ) == "array" )
10:          {
11:          print "$kulcs == <br>\n";
12:          foreach ( $ertek as $tombelem )
13:              print ".........$tombelem<br>";
14:          }
15:       else
16:          {
17:          print "$kulcs == $ertek<br>\n";
18:          }
19:       }
20: ?>
21: </body>
22: </html>
```

Mielőtt a `forearch` segítségével a `$HTTP_GET_VARS` tömb elemein végiglépkednénk, a `gettype()` függvénnyel ellenőrizzük, hogy a következő elem tömb-e. Ha az elem tömb, egy második `foreach` segítségével végiglépkedünk az elemein és kiírjuk azokat a böngészőbe.

Különbségek a GET és a POST továbbítás között

A program akkor rugalmas, ha el tudja dönteni, hogy `$HTTP_POST_VARS` vagy `$HTTP_GET_VARS` tömbből olvassa-e ki az elemeket. A továbbítás típusát (`GET` vagy `POST`) a legtöbb rendszerben a `$REQUEST_METHOD` környezeti változó tartalmazza, értéke értelemszerűen `"get"` vagy `"post"`. Érdemes tudni, hogy a `$HTTP_POST_VARS` tömb csak `POST` továbbítási mód esetén tartalmaz elemeket.

A 9.8. program mindig a megfelelő tömbből olvassa ki a paramétereket.

9.8. program A GET és POST kérelmek kezelése

```
 1: <html>
 2: <head>
 3: <title>9.8. program A GET és POST kérelmek
           kezelése</title>
 5: </head>
 6: <body>
 7: <?php
 8: $PARAMETEREK = ( count( $HTTP_POST_VARS ) )
 9:      ? $HTTP_POST_VARS : $HTTP_GET_VARS;
10: foreach ( $PARAMETEREK as $kulcs => $ertek )
11:      {
12:      if ( gettype( $ertek ) == "array" )
13:         {
14:         print "$kulcs == <br>\n";
15:         foreach ( $ertek as $tombelem )
16:             print ".........$tombelem<br>";
17:         }
18:      else
19:         {
20:         print "$kulcs == $ertek<br>\n";
21:         }
22:      }
23: ?>
24: </body>
25: </html>
```

A feltételes műveletjellel beállítjuk a `$PARAMETEREK` változó értékét. A `count()` függvénnyel ellenőrizzük, hogy a `$HTTP_POST_VARS` változó tartalmaz-e elemeket. A `count()` visszatérési értéke a tömb elemeinek száma: pozitív, ha a paraméterében megadott változó tartalmaz elemeket és 0 (`false`), ha nem.
Ha a `$HTTP_POST_VARS` tartalmaz elemeket, akkor a `$PARAMETEREK` változót egyenlővé tesszük vele, egyébként a `$PARAMETEREK` a `$HTTP_GET_VARS` értékét veszi fel. Most már kiírathatjuk a `$PARAMETEREK` tömb tartalmát a korábban látott módon.

PHP és HTML kód összekapcsolása egy oldalon

Néhány esetben szükség lehet rá, hogy az űrlapot leíró HTML kódot és az űrlapot kezelő PHP kódot egyetlen fájlban tároljuk. Ez akkor lehet hasznos, amikor a felhasználó számára többször adjuk át ugyanazt az űrlapot. Természetesen kódunk sokkal rugalmasabb, ha dinamikusan írjuk meg, de ekkor elveszítjük a PHP használatának előnyét. A HTML és PHP kódot ne keverjük a közös fájlon belül, mert így a forrás nehezebben lesz olvasható és módosítható. Ahol lehetséges, készítsünk HTML kódból meghívható PHP függvényeket, amelyeket később fel tudunk használni.

A következő példákban olyan oldalt fejlesztünk, amely az űrlapról beérkező egész számról megmondja, hogy kisebb vagy nagyobb-e egy előre megadott egész értéknél.

A 9.9. példában a fenti feladat megoldását tartalmazó HTML kódot láthatjuk. A kód egy egyszerű szövegmezőt és némi PHP kódot tartalmaz.

9.9. program Saját magát meghívó HTML kód

```
 1: <html>
 2: <head>
 3: <title>9.9. program Saját magát meghívó
           HTML kód</title>
 4: </head>
 5: <body>
 6: <form action="<?php print $PHP_SELF?>" method="POST">
 7: Ide írja a tippjét: <input type="text" name="tipp">
 8: </form>
 9: </body>
10: </html>
```

A fenti űrlap saját magát hívja meg, mert a `FORM` elem `ACTION` paraméterének a `$PHP_SELF` értéket adtunk. Vegyük észre, hogy nem készítettünk gombot, amely elküldi a beírt számot. Az újabb böngészők a szövegmező kitöltése és az `ENTER` lenyomása után automatikusan elküldik a megadott adatot, viszont néhány régebbi böngésző ezt nem teszi meg.

A 9.9. példában lévő program nem dinamikus, hiszen nem ír ki semmit, ami a felhasználótól függ. A 9.10. példában további PHP kódot építünk az oldalba. Először meg kell adnunk a számot, amit a felhasználónak ki kell találnia. A teljes változatban valószínűleg véletlenszámot állítanánk elő, most azonban egyszerűen megadjuk, mondjuk a 42-t. Ezután megnézzük, hogy az űrlapot kitöltötték-e már, különben

olyan változóval számolnánk, amely még nem létezik. A `$tipp` változó létezésének ellenőrzésével megtudhatjuk, hogy az űrlapot kitöltötték-e már. Amikor az űrlap elküldi a `"tipp"` paramétert, a `$tipp` változó globális változóként hozzáférhető lesz a programban. A `$tipp` változó létezése esetén egészen biztosak lehetünk benne, hogy az űrlapot a felhasználó kitöltötte, így elvégezhetjük a további vizsgálatokat.

9.10. program Számkitalálós PHP program

```
 1: <?php
 2: $kitalalando_szam = 42;
 3: $uzenet = "";
 4: if ( ! isset( $tipp ) )
 5:    {
 6:    $uzenet = "Üdvözlöm a számkitalálós játékban!";
 7:    }
 8: elseif   ( $tipp > $kitalalando_szam )
 9:    {
10:    $uzenet = "A(z) $tipp túl nagy, próbáljon
                  egy kisebbet!";
11:    }
12: elseif   ( $tipp < $kitalalando_szam )
13:    {
14:    $uzenet = "A(z) $tipp túl kicsi, próbáljon
                  egy nagyobbat!";
15:    }
16: else // egyenlők kell, hogy legyenek
17:    {
18:    $uzenet = "Telitalálat!";
19:    }
20: ?>
21: <html>
22: <head>
23: <title>9.10. program Számkitalálós PHP program</title>
24: </head>
25: <body>
26: <h1>
27: <?php print $uzenet ?>
28: </h1>
29: <form action="<?php print $PHP_SELF?>" method="POST">
30: Ide írja a tippjét: <input type="text" name="tipp">
31: </form>
32: </body>
33: </html>
```

A program törzsében egy `if` szerkezettel vizsgáljuk a `$tipp` változót és adunk értéket az `$uzenet` változónak. Ha a `$tipp` változó még nem létezik, a felhasználó csak most lépett az oldalra, ezért üdvözöljük. Különben megvizsgáljuk az előre megadott számot és a `$tipp` értékét, és ez alapján rendeljük az `$uzenet` változóhoz a megfelelő szöveget. Végül a HTML oldal törzsében csak ki kell írnunk az `$uzenet` értékét. A program persze még továbbfejleszthető. A PHP és a HTML kódot szétválasztva tartva egy grafikus könnyen módosíthatja az oldalt, a programozó közreműködése nélkül.

Állapot mentése rejtett mezőkkel

A 9.10. példában található program nem tudja, hogy a felhasználó hányszor tippelt, egy rejtett mező bevezetésével azonban ez is számon tartható. A rejtett mező ugyanúgy működik, mint a szövegmező, de a felhasználó nem látja, hacsak meg nem nézi a HTML forráskódot. A 9.11. példában található programban bevezetünk egy rejtett mezőt, amely a találgatások számát tartalmazza.

9.11. program Állapot mentése rejtett mezővel

```
 1: <?php
 2: $kitalalando_szam = 42;
 3: $uzenet = "";
 4: $probalkozasok = ( isset( $probalkozasok ) ) ?
    ++$probalkozasok : 0;
 5: if ( ! isset( $tipp ) )
 6:     {
 7:     $uzenet = "Üdvözlöm a számkitalálós játékban!";
 8:     }
 9: elseif    ( $tipp > $kitalalando_szam )
10:     {
11:     $uzenet = "A(z) $tipp túl nagy, próbáljon
                   egy kisebbet!";
12:     }
13: elseif    ( $tipp < $kitalalando_szam )
14:     {
15:     $uzenet = "A(z) $tipp túl kicsi, próbáljon
                   egy nagyobbat!";
16:     }
17: else // egyenlők kell, hogy legyenek
18:     {
19:     $uzenet = " Telitalálat!";
20:     }
```

9.11. program (folytatás)

```
21: $tipp = (int) $tipp;
22: ?>
23: <html>
24: <head>
25: <title>9.11. program Állapot mentése rejtett
          mezővel</title>
26: </head>
27: <body>
28: <h1>
29: <?php print $uzenet ?>
30: </h1>
31: Tippelés sorszáma: <?php print $probalkozasok?>
32: <form action="<?php print $PHP_SELF?>" method="POST">
33: Ide írja a tippjét:
34: <input type="text" name="tipp" value="<?php
     print $tipp?>">
35: <input type="hidden" name="probalkozasok"
    value="<?php print $probalkozasok?>">
36: </form>
37: </body>
38: </html>
```

A rejtett mező neve `"probalkozasok"`. A PHP segítségével kiírjuk a `"probalkozasok"` és a `"tipp"` mező értékét, hogy a felhasználó tudja, hányszor próbálkozott és mit tippelt utoljára. Ez az eljárás akkor lehet hasznos, amikor a kitöltött űrlapot elemezzük. Ha az űrlapot rosszul töltötték ki, a felhasználó tudni fogja, mit rontott el. A beadott `tipp` értékét egész számmá alakítjuk a kérdőívbe írás előtt.

Egy kifejezés értékének kiírásához a böngészőben a print() vagy az echo() utasításokat használhatjuk. PHP módban ezt egyszerűbben is megtehetjük. Ha egyenlőségjelet (=) teszünk a PHP blokkot nyitó elem után, a böngésző az azt következő kifejezés értékét kiírja, így a

```
<? print $proba?>
```

helyett írhatjuk a következőt is:

```
<?=$proba?>
```

A PHP kód törzsében egy feltétellel megvizsgáljuk, hogy kell-e növelni a `$probalkozasok` változó értékét. Ha a változó létezik, akkor növeljük az értékét, különben 0-ra állítjuk. A HTML kód törzsében folyamatosan kiírjuk, hányszor próbálkozott a felhasználó.

Ne bízzunk meg feltétel nélkül a rejtett mezőkben. Nem tudhatjuk, hogy honnan származik az értékük. Nem mondjuk, hogy ne használjuk őket, csak azt, hogy nagy körültekintéssel tegyük. A felhasználó a forrás megváltoztatásával könnyedén csalhat a programban. A rejtett mezők használata nem biztonságos.

A felhasználó átirányítása

Programunknak van egy nagy hátulütője. Az űrlap mindig újratöltődik, akár kitalálta a felhasználó a számot, akár nem. A HTML nyelvben sajnos elkerülhetetlen az egész oldal újratöltése, a felhasználót azonban átirányíthatjuk egy gratuláló oldalra, ha kitalálta a számot.

Amikor az ügyfélprogram elkezdi a kapcsolatot a kiszolgálóval, elküld egy fejlécet, amely különböző információkat tartalmaz az azt követő dokumentumról. A PHP ezt automatikusan megteszi, de a `header()` függvénnyel mi is beállíthatjuk a fejléc egyes elemeit. Ha a `header()` függvényt szeretnénk használni, biztosnak kell benne lennünk, hogy a böngészőnek nem írtunk ki semmit, különben a függvényhívás előtt a PHP elküldi a saját fejlécét. Ez mindenféle kiírás, még sortörés és szóköz karakter esetén is bekövetkezik. A `header()` függvény használatakor semmi nem lehet a függvényhívás előtt, így ellenőriznünk kell a felhasználandó külső állományokat is. A 9.12. példában egy jellemző PHP fejlécet láthatunk.

9.12. példa Egy jellemző PHP fejléc

```
1: HEAD /php-konyv/urlapok/9.1.program.php HTTP/1.0
2: HTTP/1.1 200 OK
3: Date: Sun, 09 Jan 2000 18:37:45 GMT
4: Server: Apache/1.3.9 (Unix) PHP/4.0
5: Connection: close
6: Content-Type: text/html
```

A fejlécet a Telnet programmal írathatjuk ki. Kapcsolódjunk a 80-as kapun a webkiszolgálóhoz és gépeljük be a következőt:

```
HEAD /utvonal/fajl.html HTTP/1.0
```

Az ügyfélprogram ekkor kiírja a fejlécet.

Egy `"Location"` fejléc elküldésével a böngészőt egy másik lapra irányíthatjuk:

```
header( "Location: http://www.kiskapu.hu/" );
```

Tegyük fel, hogy készítettünk egy `"gratulacio.html"` oldalt, ahova átirányítjuk a felhasználót, amikor kitalálta a megadott számot. A megoldást a 9.13. példában találjuk.

9.13. **program** Fejléc küldése a header() függvénnyel

```
 1: <?php
 2: $kitalalando_szam = 42;
 3: $uzenet = "";
 4: $probalkozasok = ( isset( $probalkozasok ) ) ?
    ++$probalkozasok : 0;
 5: if ( ! isset( $tipp ) )
 6:     {
 7:     $uzenet = "Üdvözlöm a számkitalálós játékban!";
 8:     }
 9: elseif   ( $tipp > $kitalalando_szam )
10:     {
11:     $uzenet = "A(z) $tipp túl nagy, próbáljon
                   egy kisebbet!";
12:     }
13: elseif   ( $tipp < $kitalalando_szam )
14:     {
15:     $uzenet = "A(z) $tipp túl kicsi, próbáljon
                   egy nagyobbat!";
16:     }
17: else // egyenlők kell, hogy legyenek
18:     {
19:     header( "Location: gratulacio.html" );
20:     exit;
21:     }
22: $tipp = (int) $tipp;
```

9.13. program (folytatás)

```
23: ?>
24: <html>
25: <head>
26: <title>9.13. program Fejléc küldése a header()
függvénnyel</title>
27: </head>
28: <body>
29: <h1>
30: <?php print $uzenet ?>
31: </h1>
32: Tippelés sorszáma: <?php print $probalkozasok?>
33: <form action="<?php print $PHP_SELF?>" method="POST">
34: Ide írja a tippjét:
35: <input type="text" name="tipp" value="<?php print
          $tipp?>">
36: <input type="hidden" name="probalkozasok"
37:     value="<?php print $probalkozasok ?>">
38: </form>
39: </body>
40: </html>
```

Az `if` szerkezet `else` ágában a böngészőt átirányítjuk a `"gratulacio.html"` oldalra. A `header()` függvény meghívása után az oldalnak `exit` utasítással kell végződnie, hogy befejezze az űrlap vizsgálatát.

Fájlfeltöltő űrlapok és programok

Megtanultuk, hogyan kérhetünk be adatokat a felhasználótól, de a Netscape Navigator 2-től és az Internet Explorer 4-től kezdve lehetőségünk van fájlok feltöltésére is. Ebben a részben azzal foglalkozunk, hogyan lehet a fájlok feltöltését kezelni a PHP 4 segítségével.

Először létre kell hoznunk egy HTML űrlapot, amely tartalmazza a fájlfeltöltő mezőt, melynek egyik paramétere kötelezően a következő:

```
ENCTYPE="multipart/form-data"
```

A feltöltő mező előtt meg kell adnunk egy rejtett mezőt is. Ennek neve `MAX_FILE_SIZE` kell, hogy legyen, és a fogadandó fájl lehetséges legnagyobb méretét adja meg, bájtban. Nem lehet nagyobb, mint a `php.ini` fájlban megadott

upload_max_filesize értéke, amely alapértelmezés szerint 2 megabájt. Miután kitöltöttük a MAX_FILE_SIZE mezőt, a fájlt egy egyszerű INPUT elemmel tölthetjük fel, melynek TYPE paramétere "file". Bármilyen nevet adhatunk neki. A 9.14. példában a fájlt feltöltő HTML kódot láthatjuk.

9.14. program Fájlfeltöltő űrlap

```
 1: <html>
 2: <head>
 3: <title>9.14. program Fájlfeltöltő űrlap</title>
 4: </head>
 5: <body>
 6: <form enctype="multipart/form-data"
     action="<?print $PHP_SELF?>" method="POST">
 7: <input type="hidden" name="MAX_FILE_SIZE"
     value="51200">
 8: <input type="file" name="fajl"><br>
 9: <input type="submit" value="feltöltés!">
10: </form>
11: </body>
12: </html>
```

Vegyük észre, hogy a program meghívja az oldalt, amely tartalmazza, hogy PHP kóddal kezeljük a fájlt. A legnagyobb fájlméretet 50 kilobájtban adjuk meg és "fajl"-nak nevezzük el.

Amikor a fájlt feltöltöttük, az egy ideiglenes könyvtárban tárolódik (ez alapbeállításban a /tmp, ha UNIX rendszerről van szó). A fájl elérési útját a betöltő mező nevével megegyező globális változó tartalmazza (példánkban a $fajl). A fájlról a különböző információkat a PHP globális változókban tárolja. Nevük a fájl elérési útját tartalmazó változónév, kiegészítve a "name", "size" és "type" utótagokkal. A változók jelentését a 9.2. táblázat tartalmazza.

9.2. táblázat Feltöltött fájlhoz kapcsolódó globális változók

Változónév	*Tartalma*	*Példa*
$fajl	A fájl ideiglenes elérési útja	/tmp/php5Pq3fU
$fajl_name	A feltöltött fájl neve	test.gif
$fajl_size	A feltöltött fájl mérete bájtban	6835
$fajl_type	A feltöltött fájl típusa	image/gif

A PHP 4 a feltöltött fájlok adatainak tárolására beépített tömb típusú változókat használ. Ha egy vagy több fájlt töltünk fel egy űrlapon keresztül, a fájlok adatai a `$HTTP_POST_FILES` tömbben tárolódnak. A tömb indexei a feltöltést meghatározó mezők nevei. A 9.3. táblázatban láthatjuk a tömb elemeinek nevét és értékét.

9.3. táblázat Feltöltött fájlhoz kapcsolódó globális változók

Elem	*Tartalma*	*Példa*
`$HTTP_POST_FILES["fajl"]["name"]`	A feltöltött fájl neve	`test.gif`
`$HTTP_POST_FILES["fajl"]["size"]`	A feltöltött fájl mérete bájtban	`6835`
`$HTTP_POST_FILES["fajl"]["type"]`	A feltöltött fájl típusa	`image/gif`

A 9.15. példában látható program kiírja a rendelkezésre álló információkat a feltöltött fájlról, és ha az GIF formátumú, megjeleníti.

9.15. program Fájlfeltöltő program

```
 1: <html>
 2: <head>
 3: <title>9.15. program - Fájlfeltöltő program</title>
 4: </head>
 5: <?php
 6: $feltoltes_konyvtar = "/home/httpd/htdocs/feltoltesek";
 7: $feltoltes_url = "http://php.kiskapu.hu/feltoltesek";
 8: if ( isset( $fajl ) )
 9:     {
10:     print "elérési út: $fajl<br>\n";
11:     print "név: $fajl_name<br>\n";
12:     print "méret: $fajl_size bájt<br>\n";
13:     print "típus: $fajl_type<p>\n\n";
14:     if ( $fajl_type == "image/gif" )
15:         {
16: copy ( $fajl, "$feltoltes_konyvtar/$fajl_name") or die
          ("Nem lehet másolni");
17:
18:         print "<img
src=\"$feltoltes_url/$fajl_name\"><p>\n\n";
19:         }
20:     }
```

9.15. program (folytatás)

```
21: ?>
22: <body>
23: <form enctype="multipart/form-data" action="<?php print
     $PHP_SELF?>" method="POST">
24: <input type="hidden" name="MAX_FILE_SIZE"
           value="51200">
25: <input type="file" name="fajl"><br>
26: <input type="submit" value="feltöltés!">
27: </form>
28: </body>
```

A 9.15. példában először ellenőrizzük, hogy a `$fajl` globális változó létezik-e. Ha létezik, feltehetjük, hogy a feltöltés sikeres volt (legalábbis e példa erejéig).

Mindig csak megbízható forrásból fogadjunk fájlt és ellenőrizzük, hogy programunk számára megfelelő formátumú-e. A rosszindulatú látogatók olyan űrlapot készíthetnek, amelyben megegyező nevű elemek találhatóak, de teljesen más tartalommal, így programunk nem a várt adatokat kapja meg. Lehet, hogy ez paranoiának tűnik, de jegyezzük meg, hogy a kiszolgálóoldali programozásban a paranoia jó dolog. Soha ne bízzunk külső forrásból érkező adatokban, még akkor sem, ha a külső forrás egy olyan űrlap, amit éppen mi készítettünk.

Miután feltöltöttük a fájlt, a böngészőben megjelenítjük a `$fajl`, `$fajl_name`, `$fajl_size` és `$fajl_type` változók tartalmát, amelyek a fájlról információkat tartalmaznak.

Ezután ellenőrizzük a `$fajl_type` változót. Ha értéke `"image/gif"`, akkor az érkezett állományt GIF képként megjeleníthetjük. A típust a fájlkiterjesztés alapján állapítja meg a rendszer, ezért érdemes ellenőrizni, hogy a fájl valóban GIF formátumú-e. Ennek módjáról a tizenhetedik, karakterláncokkal foglalkozó órában tanulunk.

A `copy()` függvénnyel a feltöltött fájlt a kiszolgáló másik könyvtárába másolhatjuk. A `copy()` függvénynek két paramétert kell megadni: az első a másolandó fájl elérési útja, a második a fájl új elérési útja. A fájl eredeti elérési útja a `$fajl` változóban található. Az új elérési út számára létrehozunk egy `$feltoltes_konyvtar` nevű változót, amihez hozzáfűzzük a `$fajl_name` értéket. Ezután a `copy()` függvénnyel átmásoljuk a fájlt. UNIX rendszerben a kiszolgálóprogramok a `'nobody'` felhaszná-

lónév alatt futnak, ezért mielőtt másolatot készítenénk a fájlról, ellenőriznünk kell, hogy a művelet engedélyezett-e. Az `or` művelettel a `die()` függvényt használjuk, ha a másolás sikertelen. Ezt a módszert a tizedik, a fájlokkal foglalkozó órában részletesebben áttekintjük.

Másolás után az eredeti fájlt nem kell törölnünk, a PHP ezt megteszi helyettünk. Ha nem másoljuk vagy helyezzük át a feltöltött állományt, a fájl elvész, mivel az a PHP kód végrehajtása után törlődik az ideiglenes könyvtárból.

Amikor létrehozzuk a `$feltoltes_konyvtar` változót, létrehozzuk a `$feltoltes_url` változót is, a célkönyvtár URL címének tárolására. A lemásolt fájlt HTML kép elemként jelenítjük meg.

Összefoglalás

Eddig olyan dolgokat tanultunk, amelyek segítségével igazi interaktív környezetet hozhatunk létre. Van azonban még néhány tanulnivaló. A felhasználóról képesek vagyunk információkat szerezni, de mit tegyünk ezekkel? Például fájlba írhatjuk. Ez lesz a következő óra témája.

Ebben az órában megtanultuk, hogyan használjuk a `$GLOBALS` tömböt, az űrlapok adatait, a feltöltött fájlokat és a globális változókat. Megtanultuk, hogyan küldhetünk fejlécet az ügyfélnek a böngésző átirányítása érdekében. Megnéztük, hogyan listázzuk ki az űrlapról kapott adatokat és rejtett mezőket használtunk, adatok átadására a programtól a böngészőnek.

Kérdések és válaszok

Létre lehet hozni tömböt olyan elemek tárolására is, amelyek nem választómezővel vagy jelölőnégyzetekkel megadottak?
Igen. Minden elem, amely nevének végén üres szögletes zárójel van, tömbként tárolódik, így az elemek csoportba rendezhetők.

A header() függvény nagyon hasznosnak tűnik. Tanulunk még a HTTP fejlécekről?
Magáról a HTTP-ről még beszélünk a tizenharmadik órában.

Az elemek nevének automatikus változónévvé alakítása veszélyesnek tűnik. Kikapcsolható valahol ez a lehetőség?
Igen a `php.ini` fájl `register_globals` elemét kell `off`-ra állítani a lehetőség kikapcsolásához.

Műhely

A műhelyben kvízkérdések találhatók, melyek segítenek megszilárdítani az órában szerzett tudást. A válaszokat az A függelékben helyeztük el.

Kvíz

1. Melyik környezeti változó tartalmazza a felhasználó IP címét?

2. Melyik környezeti változó tartalmaz a böngészőről információkat?

3. Hogyan nevezzük el az űrlap mezőjét, hogy a rá hivatkozó globális változó `$urlap_tomb` nevű tömb legyen?

4. Melyik beépített tömb tartalmazza a `GET` kérelmen keresztül érkező változókat?

5. Melyik beépített tömb tartalmazza a `POST` kérelmen keresztül érkező változókat?

6. Melyik függvényt használjuk a böngésző átirányítására? Milyen karaktersorozatot kell átadnunk a függvénynek?

7. Hogyan lehet korlátozni egy űrlapról feltölthető fájl méretét?

8. Hogyan korlátozható az összes űrlapról vagy programból feltölthető fájl mérete?

Feladatok

1. Készítsünk számológép-programot, amelyben a felhasználó megadhat két számot és a rajtuk végrehajtandó műveletet (összeadás, kivonás, szorzás vagy osztás – ezeket ismerje a program).

2. Készítsünk programot, amely rejtett mező használatával megmondja, hogy a felhasználó hányszor használta az első feladatban szereplő számológépet.