A PHP nyelv

Rövid jegyzet a „Tanuljuk meg a PHP5 használatát” könyv alapján

Készült – 2016

Készítette – Nagy Zsolt

Tartalom

Alapok

PHP: Personal Home Page Tools

PHP: PHP Hypertext Preprocessor

Nyílt forráskódú, szerveroldali script nyelv.

A fejlesztéshez ingyenes rendszerek használhatóak:

- Apache webszerver,
- MySQL adatbáziskiszolgáló,
- PHP értelmező

Keretrendszerek: WAMP szerver, XAMPP szerver.

A forráskód elkészítéséhez használhatunk pl. Notepad++ editort.

PHP blokk

A PHP különbséget tesz kis- és nagybetű között!

Első PHP program

Mentsük elso.php néven.

Tartalma:

```
1:  <?php
2:    phpinfo();
3:  ?>
```

print() függvény

Adatkiírásra használhatjuk a következő függvényszerű nyelvi elemeket.

```
1:  <?php
2:    print ˝Helló˝;
3:    print 'Helló';
4:    echo ˝Helló˝;
5:  ?>
```

A szöveg kiíratás során (˝) jelet használva a változó értéke behelyettesítődik, (')-ot használva nincs változó behelyettesítés.

HTML és PHP kód

A HTML dokumentum doctype deklarációja elé, vagy a body részben szokták elhelyezni a PHP kódot. A PHP értelmezője figyelmen kívül hagy minden olyan kódot, amely a PHP nyitó és záró elemén `<?php ?>` kívül esik. Pl.

```
1:  <doctype html>
2:  <html>
3:  <head>
4:  <meta charset="utf-8">
5:  <title>_Oldal címe</title>
6:  </head>
7:  <body>
8:  <?php
9:  print ˝Ez egy mondat.";
```

```
10: print ˝Ez még mindig ugyanaz a sor”;
11: print ˝Ha új sort akarunk sortörés kell.<br />”;
12: print ˝Ez már új sorban van.”;
13: ?>
14: </body>
15: </html>
```

Megjegyzések

Egysoros megjegyzés: // jellel kezdődik.

Többsoros: `/* …. */`.

PHP - Alkotóelemek

Változók

A változók egy dollárjelből $ és egy névből állnak. A névre az azonosítóképzés szabálya vonatkozik – betűvel kezdődik, és betűvel vagy számmal folytatódik.

```
1:  $a;
2:  $nev;
3:  $_a;
```

Helytelen változónevek:

```
1:  $123;
2:  $1a;
3:  $-a;
4:  stb.
```

Értékadás:

```
1:  $a = 10;
2:  $nev = ˝Nagy Zsolt˝;
3:  $pi = 3.14;
4:  stb.
```

Adattípusok

A PHP gyengén típusos nyelv, ami azt jelenti, hogy a változó deklarációjakor nem kell megadni a típusát. Azt az értelmező az első adat ismeretében fogja hozzárendelni.

Típusok vannak, csak nem a programozó rendeli a változóhoz.

A PHP típusai:

- Integer – egész szám: 10,
- Double – lebegőpontos szám: 10.123,
- String – szöveg, karakterlánc: ˝helló˝,
- Boolean – logikai érték: true, false,
- Object – objektum,
- Array – tömb,
- Resource – erőforrásra való hivatkozás,
- NULL – kezdőérték nélküli változó.

Egy változó adattípusának lekérdezése: gettype($a).

Típus és érték lekérdezése: var_dump($a).

Egy bizonyos típus lekérdezése: `is_array($a), is_bool(), is_double(), is_int(), is_object(), is_string(), is_null(), is_resource()`.

Típus módosítása függvénnyel – a változó értéke módosul: `settype($a, string)`.

Típus módosítása típusátalakítással – a változó típusa nem módosul, csak egy érték másolat készül az új típussal: `(string)$a`.

Illetve: `doubleval(), intval(), strval()`.

Műveleti jelek és kifejezések

Aritmetikai műveletek

+, - * / %

Az osztás valós osztás, a % jel maradékképzés.

Összefűzés

˝alma˝.˝fa˝

˝almafa˝

Összetett értékadó műveletek:

+=, -=, *=, /=, %=, .=

Pl. $a = $a + 1;

$a += 1;

Összehasonlítás

==, != - egyenlő, nem egyenlő,

===, (!===)- azonosság, (nem azonosság) vizsgálat (típus és érték egyezik/ nem egyezik),

<, <=, >, >= - kisebb, nagyobb vizsgálatok.

Logikai műveletek

&&, or - és

||, and – vagy

xor – kizáró vagy

! – tagadás

Műveletek erősorrendje

Csökkenő erősorrend!

++, --, (típusátalakítás),

/, *, %,

+, -,

<, <=, >, >=,

==, ===, !=,

&&,

||,

=, +=, -=, /=, %=, *=, .=,

and,

xor,

or

Állandók

```
define(˝ALLANDO_NEVE˝, 2000);

define(˝NEV˝, ˝Virág˝, true);
```

A harmadik paraméter azt mutatja, hogy a névben számít-e a kis- és nagybetű. Alapértelmezésben számít, de ha **true** értéket adunk, akkor nem veszi figyelembe.

PHP - Vezérlési szerkezetek

Elágazások

```
1:  if(feltétel) {
2:    utasítások;
3:  }
4:  ----------------
5:  if(feltétel) {
6:    utasítások_1;
7:  }
8:  else{
9:    utasítások_2;
10: }
11: ----------------
12: if(feltétel_1){
13:   utasítások_1;
14: }
15: else if(feltétel_2){
16:   utasítások_2;
17: }
18: else{
19:   utasítások_3;
20: }
21: ----------------
22: switch(egész_kifejezés){
23:   case érték_1: utasítások; break;
24:   case érték_2: utasítások; break;
25:   case érték_3: utasítások; break;
26:   …
27: ..default: utasítások;
28: }
29: -----------------
30: (feltétel) ? (utasítások_ha_igaz) : (utasítások_ha_hamis);
31: -----------------
```

Ciklusok

```
1:  while(feltétel){
2:    utasítások;
3:  }
4:  ----------------
5:  do{
6:    utasítások;
7:  while(feltétel);
8:  ----------------
9:  for($i = kezdő; $i < végérték; $i++){
10:   utasítások;
11: }
12: ----------------
13: foreach($tömb as $elem){
14:   utasítások;
15: }
16: ----------------
17: foreach($assoc_tömb as $kulcs => $érték){
18:   utasítások;
19: }
20: ----------------
```

Ciklus elhagyása

`break;` utasítással

Következő ismétlés elkezdése

`continue;` utasítással

PHP – függvények

Függvények hívása

Függvény hívása érték visszaadás elhagyásával.

```
gettype($proba);
```

Függvény hívása érték elkapásával.

```
$valtozo = gettype($proba);
```

Paraméteres függvény.

fgv_neve($elso_paraméter, ˝masodik_paraméter˝, 3);

Függvények létrehozása

```
1:  function fgv_név($param_1, $param_2){
2:    függvény_törzsének_utasításai;
3:  }
```

Érték visszaadása

```
1:  function fgv_név(){
2:    utasítások;
3:    return érték;
4:  }
```

Dinamikus függvényhívás

```
1:  function koszon(){
2:    print ˝Jó napot!<br />;
3:  }
4:  $fgv_tarolo = ˝koszon˝;
5:  $fgv_tarolo();
```

Változók hatóköre

A fgv-en belül bevezetett változók a függvényen kívülről nem érhetőek el, „helyi – lokális” változók.

global kulcsszó

Amennyiben függvényen belül akarunk hozzáférni függvényen kívül deklarált változókhoz, használni kell a „`global`” kulcsszót. Pl.

```
1:  …
2:  $kulso_valtozo = 4;
3:  function fgv(){
4:    global $kulso_valtozo;
5:    print ˝Változó értéke: $kulso_valtozo<br />˝;
6:  }
```

static kulcsszó

Ha egy változót a fgv-en belül a „`static`” kulcsszóval vezetünk be, a változó a függvény hívásai között is megőrzi a tartalmát. A változó a fgv-en kívülről nem érhető el, „helyi” marad.

Függvényparaméterek

Függvény paraméterek alapértelmezett értéke:

```
1:  function fgv($param1, $param2 = 4){}
```

Változókra való hivatkozás átadása függvénynek:

Amennyiben függvénynek paraméterként változót adunk át, a változónak csak az értéke fog átadódni. Ez azt jelenti, hogy a függvényen belül módosítunk a paraméterként kapott változó értékén, az nem lesz hatással a külső változóra → érték szerinti paraméter átadás.

A *cím szerinti paraméter átadásnál* nemcsak a változó értéke kerül át a függvényhez, hanem annak memória címe is, így a függvényen belüli változások visszahatnak a külső változóra.

```
1:  function fvg(&$param){
2:    $param += 10;
3:  }
4:  $szam = 1;
5:  fgv($szam);
6:  print ˝$szam˝;  //11-et fog kiírni
```

A fenti cím szerinti paraméterátadás már elavult módszer!

Hivatkozás visszaadása függvényből

Alapértelmezés szerint a return utasítással *értéket* adunk vissza. Pl.

```
1:  return $valtozo;
```

Ez megváltoztatható, ha egy & jelet teszünk a függvény neve elé. Ilyenkor a változó címkomponense adódik vissza a hívó programnak.

```
1:  function &fgv($param){
```

```
2:     $param += 5;
3:     return $param;
4:   }
5:   --------------------
6:   ---hívásnál pedig---
7:   --------------------
8:   $valami = 5;
9:   $valt = &fgv($valami);
10:  &valami++;
11:  print ($valt);  // 11-et ír ki
```

A cím szerinti paraméterátadás új módszere!

Függvény létezésének ellenőrzése

function_exists() – true, ha létezik, false, ha nem

Tömbök

Olyan több rekeszes változónak tekinthető, amely sok „azonos” érték tárolására alkalmas. Az eltárolt értékek összetettek is lehetnek. Az egyes elemeket a tömb béli helyük sorszámaival „indexekkel” érhetjük el. Az indexek alapértelmezés szerint 0-tól sorszámozódnak. Egy adott sorszámú elemet az (index-1) kifejezéssel kapunk meg.

Tömb létrehozása

array() függvény segítségével:

```
1:   $tomb = array(˝alma˝, ˝körte˝, ˝citrom˝);
```

vagy idexek alkalmazásával:

```
2:   $tomb[] = ˝alma˝;
3:   $tomb[] = ˝körte˝;
4:   $tomb[] = ˝citrom˝;
```

Lehet így is, de nem ajánlott:

```
5:   $tomb[0] = 2;
6:   $tomb[10] = 4;
7:   $tomb[100] = 20;
```

Ebben az esetben a tömbnek 3 eleme lesz, de az utolsó index értéke 100.

Tömb bővítése:

```
8:   $tomb = array(˝alma˝, ˝körte˝, ˝citrom˝);
9:   $tomb[] = ˝banán˝;
```

Tömb feltöltése az array_fill() fgv-el:

```
10:  $tomb = array_fill(0, 4, ˝alapértelmezett_érték˝);
```

Működése: 0. indextől 4 elemet szúr be, ˝alapértelmezett_érék˝értékkel.

Társításos (asszociatív) tömb

A tömb elemeinek nem indexei lesznek, hanem nevei. Vagyis név ➔ érték párokat alkotnak.

Létrehozása az array() fgv-el:

```
1:   $assoc_tomb = array(
```

```
2:     ″nev″ => ″János″,
3:     ″foglalkozas″ => ″informatikus″,
4:     ″kereset″ => 300000,
5:     ″munkahely″ => ″Google″
6:   );
```

Az elemek elérése:

```
$assoc_tomb['nev']
```

Létrehozás közvetlen értékadással:

```
$assoc_tomb[″nev″] => ″János″;
```

stb.

Többdimenziós tömb

```
1:   $tobbdim = array(
2:     array(
3:       ″nev″ => ″János″,
4:       ″foglalkozas″ => ″informatikus″
5:     ),
6:     array(
7:       ″nev″ => ″Ottó″,
8:       ″foglalkozas″ => ″vegyész″
9:     )
10:  );
```

Tömbök elérése

Indexeken vagy a nevükön keresztül.

```
$tomb[index]
```

```
$assoc_tomb['nev']
```

Műveletek tömbökkel

Tömb méretének lekérdezése

```
count($tomb).
```

Tömbök bejárása

```
1:   ------------------------
2:   ----indexelt tömbnél----
3:   ------------------------
4:   foreach($tomb as $elem){
5:     utasításokkal feldolgozható a $elem;
6:   }
7:   --------------------------
8:   ----társításos tömbnél----
9:   --------------------------
10:  foreach($assoc_tomb as $kulcs => $ertek){
11:    utasításokkal a $kulcs és a $ertek feldolgozható;
12:  }
13:  ----------------------------
14:  ----többdimenziós tömbnél----
15:  ----------------------------
```

```
16: foreach($kulso_tomb as $belso_tomb){
17:   foreach($belso_tomb as $ elem){
18:     utasításokkal a $elem feldolgozható;
19:   }
20: }
21: ------------------------------
22: ----tömbök gyors kiíratása----
23: ------------------------------
24: print_r()
```

Egyéb műveletek

Tömbök egyesítése:

```
1:  $egyesitett_tomb = array_merge($elso_tomb, $masodik_tomb);
```

Egyszerre több elem hozzáadása:

```
2:  $elemszam = array_push($tomb, 1, 2, 3);
```

Első elem eltávolítása:

```
3:  $elem = array_shift($tomb);
```

Tömb részének kinyerése:

```
4:  $resz_tomb = array_slice($tomb, kezdo_index, darabszam);
```

Tömbök rendezése

Indexelt tömbök növekvő/csökkenő rendezése:

```
sort($tomb);

rsort($tomb);
```

Asszociatív tömb növekvő/csökkenő rendezése:

```
asort($tomb);

arsort($tomb);
```

A fenti rendező függvények az elemek értékei alapján rendez szám, vagy szöveg szerint.

Társításos tömb *kulcs* szerinti rendezése:

```
ksort($tomb);

krsort($tomb);
```

Karakterláncok kezelése

Karakterláncok formázása

A `printf(), sprintf()` függvényekkel.

```
printf(˝formázó karakterlánc˝, $ertek1, $ertek2, stb);
```

A formázó karakterlánc *kiíratandó karaktereket és típusleírókat* tartalmaz.

```
printf(˝Nevem: %s, életkorom: %d˝, $nev, $eletkor);  //egy sorban

printf(˝Nevem: %s<br />\n, Életkorom: %d˝, $nev, $eletkor);  //több
sorban
```

Típusleírók:

- d: decimális szám,
- b: egész szám bináris formában jelenik meg,
- c: egész szám ASCII megfelelője jelenik meg,
- f: lebegőpontos számként,
- o: egész szám oktális számként jelenik meg,
- s: karakterláncként,
- x, X: egész szám hexadecimális számként jelenik meg.

Mezőszélesség megadása - a % jelet követő egész szám, ha nincs kitöltő karakter:

```
printf(˝%20s˝, ˝könyvek˝);  //……könyvek
```

Pontosság megadása – közvetlenül a típusleíró elé írt pontból és számból áll:

```
printf(˝%.2f˝, 3.14159);  //3.14 lesz kiírva
```

Kimenet kitöltése adott szélességűre, adott karakterrel – közvetlenül a % után áll:

```
printf(˝%04d”, 36);  //0036 lesz kiírva
```

Formázott karakterlánc tárolása:

```
$formazott_szoveg = sprintf(˝%.2f˝, 3.1415927);
```

Indexelés

```
$szoveg = ˝almafa˝;

$szoveg[0]

$szoveg[1]
```

Karakterlánc függvények

Szöveg hosszának megállapítása: `strlen($szoveg)` – a szöveg hosszát adja vissza.

Szövegrész megkeresése: `strstr($miben, ˝mit˝)` – Ha nincs bene, false-ot ad vissza, ha benne van, a keresett karakterlánccal kezdődő részt.

Részlánc elhelyezkedésének meghatározása: `strpos($miben, ˝mit˝, melyik_idextől)` – Ha nincs benne false, ha benne van, akkor a keresett szöveg kezdő pozícióját adja. A harmadik paraméter nem kötelező.

Szöveg részének kinyerése: `substr($miben, mettől, hány_karaktert)`.

Karakterlánc elemekre bontása: `strtok($miben, $hatarolo)` – A határoló karakter mentén fogja részekre bontani, tokenizálni a szöveget. A határoló jelet minden hívásnál meg kell adni.

Szöveg tisztítása, fehér szóközök levágása: `trim(), ltrim(), rtrim()`.

Karakterlánc részének lecserélése: `substr_replace($miben, ˝mit˝, kezdő_index, hossz)`.

Összes részlánc lecserélése: `str_replace($mit, $mire, $miben)`.

Kis- és nagybetű közötti váltás: `strtolower($szoveg), strtoupper($szoveg)`.

Sortörések kialakítása: `nl2br($szoveg)` – a \n karatert HTML sortöréssé
 alakít.

Sortörés adott karakterszámnál: `wordwrap($szoveg, karakterszam, ˝HTML_sortörés_jele˝)` – a `wordwrap()` alapértelmezés szerint \n karakterekkel töri a sorokat, amik a HTML-ben nem jelennek meg. Ezért kell a harmadik paraméter pl. ˝
\n˝ - a \n karakterpárt csak a *fordított forrás* (böngészőben – forrás megtekintése ctrl + u) szebb kimenete miatt érdemes alkalmazni.

Karakterlánc tömbbé alakítása: `explode(˝határoló˝, $szöveg)`.

Űrlapok használata

Szuperglobális változók

Olyan tömbökről van szó, amelyeket beépítettek a PHP-be. Automatikusan töltődnek fel értékekkel, és a program bármely részén elérhetőek.

$_COOKIE	böngészősüti kulcs-érték párok,
$_ENV	program héjkörnyezetének kulcs-érték párjai,
$_FILES	feltöltött fájl információi,
$_GET	HTTP GET módszerrel elküldött kulcs-érték párok,
$_POST	HTTP POST módszerrel küldött kulcs-érték párok,
$_REQUEST	$_GET, $_POST, $_COOKIES szuperglobális tömbök értékei,
$_SERVER	kiszolgáló által elérhetővé tett változók,
$_GLOBALS	minden globális változót tartalmaz, ami az adott programhoz tartozik.

$_SERVER fontosabb elemei

$_SERVER['PHP_SELF']	Az aktuális program. Hivatkozásokban és form elemek action paraméterében használható.
$_SERVER['HTTP_USER_AGENT']	Az ügyfélprogram neve és változatszáma.
$_SERVER['REMOTE_ADDR']	Az ügyfél IP címe.
$_SERVER['REQUESTED_METHOD']	A kérelem módja GET vagy POST.
$_SERVER['QUERY_STRING']	A GET kérelmeknél az URL-hez kapcsolt kódolt adat.
$_SERVER['REQUEST_URI']	A kérelem teljes címe a lekérdező karakterlánccal.
$_SERVER['HTTP_REFERER']	Az oldal címe, amelyről a kérelem érkezett.

Felhasználói adatok bekérése

Az adatokat HTML űrlapokon keresztül különböző vezérlőket felhasználva tudjuk megadni. A `form` elemnek az `action` tulajdonság értéke határozza meg a feldolgozó fájl (valami.php) nevét, illetve a `method` tulajdonsága pedig a küldés típusát (get, post) adja meg. A feldolgozó programban az űrlap vezérlőin keresztül bevitt adatokat a `$_GET`, és a `$_POST` szuperglobális tömbök kulcs-érték párjain keresztül érhetjük el.

index.html fájl tartalma:

```
1:  <!--Téglalap kerület és terület számítása űrlap
    segítségével, szerver oldali kóddal.-->
2:  <!doctype html>
3:  <html lang="hu">
4:  <head>
5:    <meta charset="utf-8">
6:    <title>Első PHP űrlap</title>
7:  </head>
8:  <body>
9:    <h1>Első PHP űrlap</h1>
10:   <p>Téglalap területének és kerületének számítása.</p>
11:   <form action="feldolgoz.php" method="get">
12:     <fieldset>
13:       <legend>Add meg a téglalap adatait.</legend>
14:       <p><label>a: <input type="text" name="a"
    id="a"></label></p>
15:       <p><label>b: <input type="text" name="b"
    id="b"></label></p>
16:     </fieldset>
17:     <fieldset>
18:       <legend>Vezérlő</legend>
19:       <p>
20:         <button type="submit">Feldolgozás</button>
21:         <button type="reset">Törlés</button>
22:       </p>
23:     </fieldset>
24:   </form>
25: </body>
26: </html>
```

feldolgoz.php fájl tartalma:

```
1:  <?php
2:  $a = $_GET['a'];
3:  $b = $_GET['b'];
4:  $terulet = $a * $b;
5:  $kerulet = 2 * ($a + $b);
6:  ?><!doctype html>
7:  <html lang="hu">
8:  <head>
9:    <meta charset="utf-8">
10:   <title>Első PHP űrlap</title>
11: </head>
12: <body>
13:   <h1>Első PHP űrlap</h1>
14:   <p>Téglalap területének és kerületének számítása.</p>
15:   <p>Terület: <?php print $terulet; ?></p>
16:   <p>Kerület: <?php print $kerulet; ?></p>
17: </body>
18: </html>
```

Felhasználói adatok beemelése globális hatókörbe

Lehetséges, de nem javasolt az űrlap mezőinek globális változókká alakítása.

```
import_request_variable(˝g˝, ˝import_˝);
```

Az első paraméter lehet: g, p, c, rendre a GET, POST, COOKIES neveknek megfelelően.

A második paraméterrel egy változó előtag adható meg.

Űrlap elemeinek elérése felhasználói tömbökkel

Az *index.html* fájl tartalma ugyan az, mint az első, csak az `action` elemnek *feldolgoz_2.php* az értéke.

feldolgoz_2.php tartalma:

```
1:  <?php
2:      $a = $_GET['a'];
3:      $b = $_GET['b'];
4:  function terulet(){
5:      global $a;
6:      global $b;
7:      $terulet = $a * $b;
8:      return $terulet;
9:  }
10: function kerulet(){
11:     global $a;
12:     global $b;
13:     $kerulet = 2 * ($a + $b);
14:     return $kerulet;
15: }
16: ?><!doctype html>
17: <html lang="hu">
18: <head>
19:     <meta charset="utf-8">
20:     <title>Első PHP űrlap</title>
21: </head>
22: <body>
23:     <h1>Első PHP űrlap</h1>
24:     <p>Téglalap területének és kerületének számítása.</p>
25:     <p>Terület: <?php print terulet(); ?></p>
26:     <p>Kerület: <?php print kerulet(); ?></p>
27: </body>
28: </html>
```

PHP és HTML kód összekapcsolása

index_2.php fájl tartalma:

```
1:  <!--Téglalap kerület és terület számítása űrlap
    segítségével, szerver oldali kóddal.-->
2:  <?php
3:  if(!empty($_GET)){
4:      $a = (int)$_GET['a'];
5:      $b = (int)$_GET['b'];
6:      $terulet = $a * $b;
7:      $kerulet = 2 * ($a + $b);
8:  }
9:  else {
```

```
10:     $a = "";
11:     $b = "";
12:     $terulet = "";
13:     $kerulet = "";
14: }
15: ?><!doctype html>
16: <html lang="hu">
17: <head>
18:     <meta charset="utf-8">
19:     <title>Első PHP űrlap</title>
20: </head>
21: <body>
22:     <h1>Első PHP űrlap.</h1>
23:     <p>Téglalap területének és kerületének számítása.</p>
24:     <form action="<?php print $_SERVER['PHP_SELF']; ?>"
    method="get">
25:         <fieldset>
26:             <legend>Add meg a téglalap adatait.</legend>
27:             <p><label>a: <input type="text" name="a"
    id="a" value="<?php print $a; ?>"></label></p>
28:             <p><label>b: <input type="text" name="b"
    id="b" value="<?php print $b; ?>"></label></p>
29:         </fieldset>
30:         <fieldset>
31:             <legend>Terület és kerület</legend>
32:             <p><label>terület: <input type="text"
    name="t" id="t" value="<?php print $terulet;
    ?>"></label></p>
33:             <p><label>kerület: <input type="text"
    name="k" id="k" value="<?php print $kerulet;
    ?>"></label></p>
34:         </fieldset>
35:         <fieldset>
36:             <legend>Vezérlő</legend>
37:             <p>
38:                 <button
    type="submit">Feldolgozás</button>
39:                 <button type="reset">Törlés</button>
40:             </p>
41:         </fieldset>
42:         <p>Hibakezeléshez használható a print_r()
    tömbkiíró függvény.</p>
43:         <p><?php print_r($_GET); ?></p>
44:     </form>
45: </body>
46: </html>
```

Állapot mentése rejtett mezőkkel

```
1:  <input type=˝hidden˝ value=˝<?php print $ertek; ?>˝>
```

A felhasználó átirányítása

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 semmit, különben a függvényhívás a PHP elküldi a saját fejlécét. Ez mindenféle kiírás, még sortörés és szóköz karakter esetén is bekövetkezik. A `header()` fgv. használatakor semmi nem lehet a függvényhívás előtt, így ellenőrizni kell a felhasználandó külső állományokat is.

Egy jellemző PHP fejléc

```
1:  HEAD /eleresi_ut/valami.php http/1.0
2:  Host: webhely.hu
3:  http/1.1 200 OK
4:  Date: Mon, 10 aug 2016 10:21:10 GMT
5:  Server: Apache/2.4.18 (Unix) PHP/7.0.9
6:  X-Powered-By: PHP/7.0.9
7:  Connection: close
8:  Content-Type: text/html; charset=UTF-8
```

A böngésző másik lapra irányítása

```
header(˝Location: http://www.szerencsiszakkepzo.sulinet.hu˝);
```

A header() fgv-t követő kódrész kihagyásához ki kell adni az exit; utasítást.

Fájlfeltöltő űrlapok

Fájl feltöltésekor meg kell adni a form elemnek egy enctype paramétert:

```
1:  <form enctype=˝multipart/form-data˝>…</form>
```

Korlátozhatjuk egy rejtett mezben a feltöltendő fájl méretét (byte-ban), pl. 4Kb-os méret korlát:

```
2:  <input type=˝hidden˝ name=˝MAX_FILE_SIZE˝ value=˝4096˝>
```

Egy fájlt az input elemmel tudjuk kiválasztani, melynek típusa: ˝file˝.

```
3:  <input type=˝file˝ name=˝feltoltes˝>
```

A fájl feltöltése a ˝submit˝ gomb megnyomásával lehet.

```
4:  <input type=˝submit˝ value=˝Feltöltés˝>
```

Vagy a HTML 5 button elemével:

```
5:  <button type=˝submit˝>Feltöltés</button>
```

A feltöltött fájl elérése

Amikor a fájl sikeresen feltöltöttük, az egyedi nevet kap, és egy ideiglenes könyvtárban tárolódik (alapbeállítás a /tmp). Ez egy biztonsági elem, mert itt a fájlt meg lehet vizsgálni és, ha minden rendben, akkor át lehet másolni a végleges helyére.

A fájlról információkat a $_FILES szuperglobális tömbből szerezhetünk.

$_FILES['feltoltes']['name']	A feltöltött fájl neve.
$_FILES['feltoltes']['tmp_name']	Az ideiglenes fájl elérési útja.
$_FILES['feltoltes']['size']	A feltöltött fájl mérete bájtban.

$_FILES['feltoltes']['error']	Egy PHP állandónak megfelelő hibakód. Pl. UPLOAD_ERR_FROM_SIZE
$_FILES['feltoltes']['type']	A feltöltött fájl típusa. Pl. image/jpg

A $_FILES hibaállandói

UPLOAD_ERR_OK	0	Minden rendben.
UPLOAD_ERR_INI_SIZE	1	A fájl mérete meghaladja a php.ini upload_max_filesize beállításban megadottat.
UPLOAD_ERR_FROM_SIZE	2	A fájl mérete meghaladja a MAX_FILE_SIZE rejtett elemben megadottat.
UPLOAD_ERR_PARTIAL	3	A fájl csak részben töltődött fel.
UPLOAD_ERR_NO_FILE	4	A fájl nem töltődött fel.

Fájlok használata

Fájlok beágyazása

Forrásfájlokat illeszthetünk a kódunkba, amely úgy fog végrehajtódni, mintha eredetileg is része lett volna. Amennyiben nem található a fájl, a program fut tovább.

`include();`

Abban az esetben, ha a fájl hiánya esetén meg szeretnénk állítani a program futását:

`require();`

Ha meg szeretnénk akadályozni, hogy már egy beemelt fájlt újra beágyazzunk, használjuk:

`include_once();` illetve,

`require_once();` utasításokat.

Érték visszaadása beágyazott fájlból

A beemelt fájlban használjuk a `return` érték; utasítást.

Majd a programban felfoghatjuk az értéket:

```
$valtozo = include(˝beszurt_file.php˝);
```

Fájlok vizsgálata

Fájl létezésének ellenőrzése: `file_exists()`, amely igazat vagy hamisat ad vissza.

Fájl vagy könyvtár: `is_file()` vagy `is_dir()`.

Fájl állapotának lekérdezése: `is_readable(), is_writeable(), is_executable()`.

Fájl méretének lekérdezése: `filesize()`. Pl.

```
1:  $atime = fileatime(˝valami.txt˝);
```

Fájl utolsó módosításának lekérdezése: `filetaime()`. A visszaadott érték UNIX időbélyeg formátumú, amelyet átalakíthatunk a `date()` függvénnyel. Pl.

```
2:  print date(˝Y.m.d H:i˝, $atime);
```

Fájl utolsó módosításának ideje: `filemtime()`.

Fájl utolsó változásának időpontja: `filectime()`.

Fájlok létrehozása, megnyitása

Fájl létrehozása: `touch("valami.txt");`

Fájl törlése: `unlink("valami.txt");`

Fájl megnyitása olvasásra: `$fp = fopen("valami.txt", 'r');`

Fájl megnyitása írásra: `$fp = fopen("valami.txt", 'w');`

Fájl megnyitása hozzáfűzésre: `$fp = fopen("valami.txt", 'a');`

A karakterkódok jelentése: r – read/olvasás, w – write/írás, a – append/hozzáfűzés.

Olvasás fájlból

Sorok olvasása: `$sor = fgets($fp, 1024);`

Az `fgets()` függvény addig olvas a fájlból, amíg újsor karakterhez ("\n") nem ér, vagy a megadott bájtnyi adatot ki nem olvasta, vagy a fájl végéhez nem ér.

A fájl végének vizsgálata: `feof()`.

Tetszőleges mennyiségű adat olvasása fájlból: `$adatok = fread($fp, 16);`

Az olvasás kezdetének beállítása: `fseek($fp, 64);`

Fájl karakterenkénti olvasása: `fgetc()` függvénnyel. Pl.

```
1:  $fajlnev = "valami.txt";
2:  $fp = fopen($fajlnev, 'r') or die("$fajlnev nem nyitható
    meg");
3:  while(!feof($fp)){
4:    $karakter = fgetc($fp);
5:    print "$karakter<br />";
6:  }
```

Fájl tartalmának beolvasása egy változóba

`$tartalom = file_get_contents("valami.txt");`

Írás, hozzáfűzés fájlba

Karakterlánc kiírása: `fwite($fp, "karakterlánc"), fputs($fp, "karakterlánc")`.
Használat előtt a fájlt meg kell nyitni.

Adatok fájlba írása előzetes megnyitás nélkül (PHP 5-től):
`file_put_contents("valami.txt", "szöveg")`, illetve
`file_put_contents("valami.txt", "szöveg hozzáfűzése", FILE_APPEND)`.

Könyvtárak

Könyvtár létrehozása: `mkdir()` függvénnyel. Pl. `mkdir("valami.txt", 0777)` – teljes írási/olvasási/végrehajtási jogok.

Könyvtár törlése (a könyvtárnak üresnek kell lenni, és megfelelő jogosultsággal kell rendelkezni):
`rmdir()`.

Könyvtár megnyitása olvasásra: `$kvt = opendir(˝konyvtar˝).`

Könyvtár tartalmának olvasása: `readdir()` függvénnyel. A függvény visszatérési értéke a könyvtár következő elemének neve, vagy `false`, ha a könyvtár végére értünk.

Adatbáziskapcsolat - SQL

Csatlakozás MySQL kiszolgálóhoz

Adatbázis kiválasztása

Hibakeresés

Adatok hozzáadása

Automatikusan növekvő mező értékének lekérdezése

Adatok lekérdezése

Adatok frissítése

Dátumok kezelése

Dátum kinyerése

A PHP `time()` függvény egy UNIX időbélyeget ad vissza (1970. jan. 1. éjfél óta eltelt másodpercek száma). Pl.

```
1:  print time();
2:  //egy lehetséges kimenet: 1060005000
```

Időbélyeg átalakítása

Az időbélyegből a `getdate()` függvény egy társításos tömböt készít, amelyből kinyerhetjük a szükséges információkat. Pl. Ha nem adunk meg paramétert, akkor az aktuális időt fogja használni.

```
1:  $datum_tomb = getdate();
2:  print
    $datum_tomb['year'].˝/˝.$datum_tomb['mon'].˝/˝.$datum_tomb['
    mday'];
```

A `getdate()` által visszaadott társításos tömb elemei:

seconds	A percből eltelt másodpercek száma (0-59),
minutes	Az órából eltelt percek száma (0-59),
hours	A nap órája (0-23),
mday	A hónap napja (1-31),
wday	A hét napja (0-6),
mon	Az év hónapja (1-12),
year	Év (négy szánjeggyel),
yday	Az év napja (0-365),
weekday	A hét napja (névvel),
month	Az év hónapja (névvel),

0 időbélyeg.

A dátum formázott karakterlánc formájában való visszaadása `date()` függvénnyel. Pl.

```
print date(˝Y.m.d. H:i:s˝, time());
```

Időbélyeg készítése

Időbélyeget az `mktime()` függvénnyel készíthetünk, amelyet a `getdate()` vagy a `date()` függvényekkel hozhatunk olvasható formátumra. A függvény bemenete hat egész szám, a következő sorrendben: *óra, perc, másodperc, hónap, hónap napja, év*. Pl.

```
$ido = mktime(14, 30, 0, 5, 1, 2016);
```

Dátum ellenőrzése

A `checkdate()` függvény három egész számot vár: hónap, nap, év sorrendben. Igazat ad vissza, ha a hónap 1 és 12 között van, a nap elfogadható az adott hónapban és évben, és az év 0 és 32767 közé esik.

```
checkdate(4, 4, 1066);
```

Objektumok

Osztály, Objektum létrehozása

Osztály létrehozása

```
1:  class Ember{
2:    var $nev;
3:    var $fizetes;
4:  }
```

Objektum létrehozása

```
$ember1 = new Ember();
```

Objektumtulajdonságok

PHP4-től var kulcsszóval vezetjük be.

Hivatkozás az objektum egy tulajdonságára:

```
$ember1->nev
```

Az osztály, objektum tagfüggvényei

```
1:  class Ember{
2:    var $nev;
3:    var $fizetes;
4:    function getNev(){
5:      return $this->nev;
6:    }
7:    function setNev($n){
8:      $this->nev = $n;
9:    }
10: }
```

Hozzáférés korlátozása

```
public var $nev;
```

```
public function getNev(){}

protected var $nev;

protected function vedett(){}

private var $nev;

private function titkos(){}
```

A konstruktor

PHP5 előtt a konstruktor neve megegyezett az osztály nevével.

```
1:  class Ember{
2:    var $nev;
3:    var $fizetes;
4:    function Ember($n, $f){
5:      $this->nev = $n;
6:      $this->fizetes = $f;
7:    }
8:  }
```

Öröklés

Osztály öröklése az `extends` kulcsszóval.

Egy felülírt – szülő osztályban lévő – metódus meghívása: `parent::szulo_metodus()`.

Osztályok és objektumok ellenőrzése

Objektum osztályának megállapítása: `get_class($obj)`.

Objektum családjának megállapítása: `is_a($obj, ˝osztaly_neve˝)`.

Osztály létezésének ellenőrzése: `class_exists($osztaly_neve)`.

Tagfüggvény létezésének ellenőrzése: `method_exists($obj, ˝metodus_neve˝)`.

Objektumok tárolása és kinyerése

Objektumok tárolása szerializálással (sorosítással):

```
1:  class Osztaly{
2:    var $elem = ˝elem_neve˝;
3:  }
4:  $obj = new OSztaly();
5:  $obj_tarolo = serialize($obj);
6:  ---------------------------------
7:  ----visszaalakítás objektummá----
8:  ---------------------------------
9:  $obj_tarolo2 = unserialize($obj_tarolo);
```

VÉGE