19. ÓRA

Állapotok tárolása sütikkel és GET típusú lekérdezésekkel

A HTTP állapot nélküli protokoll. Ez azt jelenti, hogy ahány oldalt a felhasználó letölt kiszolgálónkról, annyi önálló kapcsolat létesül, ráadásul az egyes oldalakon belül a képek, külső fájlok letöltésére is külön kapcsolatok nyílnak. Másrészt viszont a felhasználók és a weboldalak készítői környezetként, azaz olyan térként érzékelik a weboldalakat, amelyben egyetlen oldal is egy nagyobb egész része. Így viszont nem meglepő, hogy az oldalról oldalra való információátadás módszerei egyidősek magával a Világhálóval.

Ezen az órán az információtárolás azon két módszerével fogunk megismerkedni, amelyek lehetővé teszik, hogy egy adott oldalon található adat a következő oldalakon is elérhető legyen. Az óra során a következőket tanuljuk meg:

- Mik a sütik és hogyan működnek?
- Hogyan olvassuk a sütiket?

- Hogyan írjunk sütiket?
- Hogyan tároljuk adatbázisban a weboldalak lekérési statisztikáját sütik segítségével?
- Mik azok a GET módú lekérések?
- Hogyan írjunk olyan függvényt, amely egy asszociatív tömböt egy lekérdező karakterlánccá alakít át?

Sütik

A Netscape munkatársai által kiötölt „csodasüti" (magic cookie) kifejezés még a Netscape 1 aranykorából származik. A kifejezés pontos eredete mindmáig viták tárgya, bár meglehetősen valószínűnek tűnik, hogy a szerencsesütinek valami köze lehetett hozzá. Azóta viszont már más böngészők is alkalmazzák ezt a szabványt.

ÚJDONSÁG A süti (cookie) kis mennyiségű adat, amelyet a felhasználó böngészője tárol egy kiszolgáló vagy egy program kérésének eleget téve. Egy gép legfeljebb 20 sütit tárolhat a felhasználó böngészőjében. Minden süti egy nevet, egy értéket, a lejárati időpontot és a gazdagépre és elérési útra vonatkozó információkat tartalmazza. Az egyes sütik mérete legfeljebb 4 KB lehet.

A süti értékeinek beállítása után csak az a számítógép olvashatja az adatokat, amely azokat létrehozta, ezzel is garantálva a felhasználó biztonságát. A felhasználó azt is beállíthatja böngészőjében, hogy az értesítse az egyes sütik létrehozásának kérelméről, vagy elutasíthatja az összes sütikérelmet. Ezekből adódóan a sütiket módjával kell használni, nem lehet teljesen rájuk támaszkodni a környezet megtervezésekor anélkül, hogy a felhasználót először ne értesítenénk.

Mindezekkel együtt a sütik kiválóan alkalmasak arra, hogy kis mennyiségű információt tároljunk velük, mialatt a felhasználó oldalról-oldalra lépdel, vagy akár hosszabb távon is, a webhely egyes látogatásai között.

A sütik felépítése

A sütik létrehozása általában a HTTP fejlécében történik (bár a JavaScript közvetlenül a böngészővel is létre tudja hozni a sütiket). Egy sütibeállító PHP program ilyesféle HTTP fejlécelemet kell, hogy létrehozzon:

```
Set-Cookie: zoldseg=articsoka; expires=Friday,
 ➥ 04-Feb-00 22:03:38 GMT; path=/; domain=kiskapu.hu
```

Mint ahogy azt láthatjuk, a `Set-Cookie` fejléc egy név–érték párt, egy greenwichi középidő szerinti lejárati időpontot (`expires`), egy elérési utat (`path`) és egy tartományt (`domain`) tartalmaz. A név és az érték URL felépítésű kell, hogy legyen. A lejárat mező (`expires`) arra ad utasítást a böngészőnek, hogy mikor „felejtse el" a sütit. Az elérési út a weboldalon azt a helyet jelöli, amelynek elérésekor a sütit vissza kell küldeni a kiszolgálóra. A tartomány mező azokat az internetes tartományokat jelöli ki, ahová a sütit el kell küldeni. A tartomány nem különbözhet attól, ahonnan a sütit küldték, viszont egy bizonyos szintű rugalmasságot is meghatározhat. Az előző példánál a böngésző a `www.kiskapu.hu` és a `leeloo.kiskapu.hu` kiszolgálóknak egyaránt hajlandó elküldeni a sütit. A HTTP fejlécéről a tizenharmadik óra anyagában többet is olvashattunk.

Ha a böngésző úgy van beállítva, hogy tárolja a sütiket, akkor az azokban tárolt adatok a süti lejártáig elérhetők maradnak. Ha a felhasználó a böngészővel olyan oldalra jut, amely egyezik a sütiben tárolt elérési útvonallal és tartománnyal, akkor elküldi a sütit a kiszolgálónak. Ehhez általában az alábbihoz hasonló fejlécelemet állít elő:

```
Cookie: zoldseg=articsoka
```

Így aztán a PHP program hozzáférhet a sütihez a `HTTP_COOKIE` környezeti változón keresztül (amely az összes süti nevét és értékét tárolja), de a `$zoldseg` globális változón keresztül vagy a `$HTTP_COOKIE_VARS["zoldseg"]` globális tömbváltozó segítségével is elérhetjük azt:

```
print "$HTTP_COOKIE<BR>"; // azt írja ki, hogy
 ➥ "zoldseg=articsoka"
print getenv("HTTP_COOKIE")."<br>"; // azt írja ki, hogy
 ➥ "zoldseg=articsoka"
print "$zoldseg<br>"; // azt írja ki, hogy "articsoka"
print "$HTTP_COOKIE_VARS["zoldseg"]<br>"; // azt írja ki, hogy
 ➥ "articsoka"
```

Sütik beállítása a PHP-vel

A PHP-vel kétféleképpen hozhatunk létre egy sütit. Egyrészt a kilencedik óra során megismert `header()` függvény segítségével beállíthatjuk a fejléc `Set-Cookie` sorát. A `header()` egy karakterláncot vár, amit az oldal HTTP fejlécébe ír. Mivel a fejlécek automatikusan kerülnek kiírásra, a `header()` függvényt még azelőtt kell meghívnunk, mielőtt bármi egyebet küldenénk a böngészőnek. Figyeljünk arra, hogy a PHP blokk kezdőeleme elé írt egyetlen szóköz vagy újsor karakter is kimenetként jelenik meg a böngészőben.

```
header ("Set_Cookie: zoldseg=articsoka; expires=Friday,
 ➥ 04-Feb-00 22:03:38 GMT; path=/; domain=kiskapu.hu");
```

Bár ez a módszer nem túl bonyolult, mégis írnunk kell egy olyan függvényt, amely előállítja a fejléc szövegét. Ez persze nem túl rázós feladat, hiszen csak a megfelelő formátumú dátumot, illetve az URL formátumú név–érték párt kell előállítanunk. Kár azonban ezzel vesződnünk, mert létezik egy olyan PHP függvény, amely pontosan ezt végzi el helyettünk.

A `setcookie()` függvény neve önmagáért beszél: a fejléc `Set-Cookie` sorát állítja elő. Ennek megfelelően azelőtt kell még meghívnunk, mielőtt valamit kiküldenénk a böngészőnek. A függvény bemenetét a süti neve, értéke, UNIX időbélyeg formátumban megadott lejárata, elérési útja, a feljogosított tartomány, valamint egy egész szám alkotja. Ezt a legutolsó paramétert 1-re állítva biztosíthatjuk, hogy a süti csak biztonságos kapcsolaton keresztül tudjon közlekedni. A süti nevén kívül egyik paraméter sem kötelező.

A 19.1. programban arra láthatunk példát, ahogy a `setcookie()` függvény segítségével beállítunk egy sütit.

19.1. program Süti értékének beállítása és kiíratása

```
 1: <?php
 2: setcookie( "zoldseg", "articsoka", time()+3600, "/",
 3:            "kiskapu.hu", 0 );
 4: ?>
 5: <html>
 6: <head>
 7: <title>19.1. program Süti értékének beállítása és
           kiíratása</title>
 8: </head>
 9: <body>
10: <?php
11: if ( isset( $zoldseg ) )
12: print "<p>Üdv, az ön által kiválasztott zöldség a(z)
           $zoldseg</p>";
13: else
14: print "<p>Üdvözöljük! Ez az ön első látogatása.</p>";
15: ?>
16: </body>
17: </html>
```

Bár a program első lefuttatásakor már beállítjuk a sütit, a `$zoldseg` változó ekkor még nem jön létre. A süti csak akkor lesz hozzáférhető, amikor a böngésző elküldi azt a kiszolgálónak. Ez csak akkor következik be, ha a felhasználó újra meglátogatja tartományunkat. A példában egy `"zoldseg"` nevű sütit hozunk létre, melynek értéke `"articsoka"`. A `time()` függvénnyel lekérdezzük a pillanatnyi időt, majd ebből 3600-at hozzáadva számítjuk ki a lejárat idejét. (A 3600 másodpercben értendő, tehát a süti 1 óra elteltével fog elavulni.) Útvonalként a `"/"`-t adjuk meg, amiből az következik, hogy sütink a webhely összes oldaláról elérhető lesz. A tartományt `"kiskapu.hu"`-ra állítottuk, aminek következtében ennek a tartománynak mindegyik kiszolgálója jogosult a süti fogadására (a `www.kiskapu.hu` tehát ugyanúgy megkapja a sütit, mint a `leeloo.kiskapu.hu`). Ha azt szeretnénk, hogy egy sütihez csak az a kiszolgáló férjen hozzá, amelyik a sütit létrehozó programot szolgáltatta, használjuk a `$SERVER_NAME` környezeti változót, ahelyett, hogy a tartomány, illetve kiszolgálónevet beépítenénk a programba. Ez a megoldás azzal az előnnyel is rendelkezik, hogy a programok módosítás nélkül átvihetők egy másik kiszolgálóra és ott is az elvárásoknak megfelelően fognak futni. Végül egy nullát is átadunk a `setcookie()`-nak, jelezvén, hogy a sütit nem biztonságos kapcsolaton keresztül is el szabad küldeni. Bár az első kivételével mindegyik paraméter elhagyható, hasznos, ha minden paramétert megadunk és csak a tartomány és biztonság adatok meghatározásától tekintünk el. Erre azért van szükség, mert bizonyos böngészők csak az útvonal megadása esetén képesek a sütik rendeltetésszerű használatára. Az útvonal elhagyásával a süti használatát az aktuális, illetve az alatta elhelyezkedő könyvtárak oldalai számára korlátozzuk.

Ha a `setcookie()` szöveges paramétereiként `""`-t, azaz üres karakterláncot adunk át, a szám paramétereinek pedig 0-t, a függvény ezeket a paramétereket nem veszi figyelembe.

Süti törlése

A sütik törlésének „hivatalos” módja az, hogy a `setcookie()` függvényt egyetlen paraméterrel, mégpedig a törlendő süti nevével hívjuk meg:

```
setcookie( "zoldseg" );
```

Ebben a megoldásban nem bízhatunk meg teljesen, mert nem minden esetben működik kifogástalanul. Ezért célszerűbb, ha egy múltbeli dátummal új értéket adunk a sütinek:

```
setcookie( "zoldseg", "", time()-60, "/", "kiskapu.hu", 0);
```

Ennek alkalmazásakor viszont ügyelnünk kell, hogy a megadott útvonal, tartomány és biztonsági paraméter tökéletesen egyezzen a süti létrehozásakor megadottakkal.

Munkamenet-azonosító sütik

Ha olyan sütit szeretnénk létrehozni, amely csak addig létezik, amíg a felhasználó be nem zárja a böngészőt, csak annyi a teendőnk, hogy a süti élettartamát nullára állítjuk. Amíg ki nem lépünk a böngészőből, az így készített sütiket a kiszolgáló gond nélkül megkapja, ám ha bezárjuk a programot, a sütik megszűnnek, így hiába indítjuk újra a böngészőt, már nem érjük el azokat.

Ezen eljárás legfőbb alkalmazási területe a felhasználók azonosítása. A felhasználó elküldi a nevét és jelszavát, válaszként pedig egy olyan oldalt kap, amely egy sütit hoz létre a böngészőben. Ezek után a böngésző a sütit minden személyes adatokat tartalmazó laphoz történő hozzáféréskor visszaküldi a kiszolgálónak, az pedig ennek hatására engedélyezi a hozzáférést. Ilyen esetekben nem kívánatos, hogy a böngésző újraindítása után is hozzáférhessünk ezekhez a lapokhoz, mert nem lehetünk biztosak benne, hogy nem egy másik felhasználó indította el a böngészőt. A munkamenet-azonosítót hagyományosan *sessionid*-nek nevezik, a böngésző bezárásáig élő sütit pedig *session cookie*-nak.

```
setcookie( "session_id", "55435", 0 );
```

Példa: Webhelyhasználat nyomon követése

Képzeljük el, hogy egy tartalomszolgáltató azzal bízott meg bennünket, hogy sütik, valamint egy MySQL adatbázis segítségével kimutatást készítsünk a látogatókról. Szükség van a látogatók számára, a látogatások alkalmával letöltött lapok számának átlagára és az olvasók által a webhelyen töltött átlagidőre, látogatókra lebontva.

Az első dolgunk, hogy megértessük megbízónkkal a sütik alkalmazásának korlátait. Nem mindegyik felhasználó böngészőjében engedélyezett a sütik használata. Ilyen esetben a lekért oldal mindig úgy fog tűnni, mintha ez lenne az adott felhasználó első látogatása. Így hát a kapott eredményeket kissé meghamisítják azok a böngészők, amelyek nem tudják vagy nem akarják kezelni a sütiket. Továbbá abban sem lehetünk biztosak, hogy egy felhasználó mindig ugyanazt a böngészőt használja, és abban sem, hogy egy adott böngészőn nem osztozik-e esetleg több ember.

Ha megbízónk mindezt tudomásul vette, nekiveselkedhetünk a tényleges megvalósításnak. Egy működő mintát kevesebb, mint 90 sorból összerakhatunk!

Szükségünk lesz először is egy adatbázistáblára, mely a 19.1-es táblázatban felsorolt mezőkből áll.

19.1. táblázat Adatbázismezők

Név	*Típus*	*Leírás*
`vendeg_azon`	egész	Automatikusan növekvő mező, amely minden látogató számára egyedi azonosítót állít elő és tárol.
`elso_latogatas`	egész	A látogató első látogatásának időbélyege.
`utolso_latogatas`	egész	A legutóbb lekért lap időpontjának időbélyege.
`latogatas_szam`	egész	Az elkülöníthető látogatások száma.
`ossz_ido`	egész	A webhelyen töltött idő közelítő értéke.
`ossz_letoltes`	egész	A látogató összes letöltési kérelmeinek száma.

A `vendeg_naplo` MySQL tábla létrehozásához a következő `CREATE` utasításra lesz szükség:

```
create table vendeg_naplo (
                vendeg_azon INT NOT NULL AUTO_INCREMENT,
                PRIMARY KEY( vendeg_azon ),
                elso_latogatas INT,
                utolso_latogatas INT,
                latogatas_szam INT,
                ossz_ido INT,
                ossz_letoltes INT
                );
```

Most, hogy elkészítettük a táblát, amivel dolgozhatunk, írnunk kell egy programot, amely megnyit egy adatbázis-kapcsolatot és képes ellenőrizni a süti jelenlétét. Ha a süti nem létezik, új sort hoz létre a táblában és feltölti az új látogató adataival. Ennek megvalósítása látható a 19.2. példában.

19.2 program A MySQL adatbázist új felhasználó adataival bővítő program

```
 1: <?php
 2: $adatbazis = mysql_connect( "localhost", "janos",
                  "majomkenyerfa" );
 3: if ( ! $adatbazis )
 4:     die( "Nem lehet a mysql-hez kapcsolódni!" );
 5: mysql_select_db( "minta", $adatbazis ) or die (
                    mysql_error() );
 6: if ( ! isset ( $vendeg_azon ) )
 7:     $vendeg_allapot = ujvendeg( $adatbazis );
 8: else
 9:     print "Örülünk, hogy ismét körünkben
              üdvözölhetjük, $vendeg_azon<P>";
10: function ujvendeg( $adatbazis )
11:     {
12:     // egy új vendég!
13:     $vendeg_adatok = array (
14:                   "elso_latogatas" => time(),
15:                   "utolso_latogatas" => time(),
16:                   "latogatas_szam" => 1,
17:                   "ossz_ido" => 0,
18:                   "ossz_letoltes" => 1
19:                   );
20:     $lekerdezes = "INSERT INTO vendeg_naplo
                       ( elso_latogatas,
21:                    utolso_latogatas,
22:                    latogatas_szam,
23:                    ossz_ido,
24:                    ossz_letoltes ) ";
25:     $lekerdezes .= "values ( ".
             $vendeg_adatok["elso_latogatas"].", ".
26:          $vendeg_adatok["utolso_latogatas"].", ".
27:          $vendeg_adatok["latogatas_szam"].", ".
28:          $vendeg_adatok["ossz_ido"].", ".
29:          $vendeg_adatok["ossz_letoltes"]." )";
30:     $eredmeny = mysql_query( $lekerdezes );
31:     $vendeg_adatok["vendeg_azon"] =
                     mysql_insert_id();
32:     setcookie( "vendeg_azon",
                  $vendeg_adatok["vendeg_azon"],
33:                time()+(60*60*24*365*10), "/" );
34:     return $vendeg_adatok;
35:     }
36: ?>
```

A szokványos módon csatlakozunk a MySQL kiszolgálóhoz és kiválasztjuk azt az adatbázist, amelyben a `vendeg_naplo` tábla található (a MySQL adatbáziskiszolgáló kezelését a tizenkettedik óra tartalmazza). Ellenőrizzük a `$vendeg_azon` változó jelenlétét, amely a felhasználót azonosító egyedi egész számot tartalmazza. Ha ez a változó nem létezik, feltételezzük, hogy új felhasználóról van szó és bejegyzéséhez meghívjuk az `ujvendeg()` nevű függvényt.

Az `ujvendeg()` függvény egy hivatkozás-azonosítót kér és egy tömböt ad vissza. A függvényben létrehozunk egy `$vendeg_adatok` nevű tömböt. A tömb `elso_latogatas` és `utolso_latogatas` elemeit a pillanatnyi időre állítjuk. Mivel ez az első látogatás, így a `latogatas_szam` és az `ossz_letoltes` elemeket 1-re állítjuk. Az `ossz_ido` 0 lesz, hiszen ezen látogatás alkalmával még nem telt el idő.

A létrehozott tömb alapján elkészítjük a táblába illesztendő új sort, úgy, hogy a tömb mindegyik elemét a tábla azonos nevű oszlopának adjuk értékül. Mivel a `vendeg_azon` mező automatikusan növekszik, megadásától eltekinthetünk. A süti beállításakor azonban szükség lesz erre az újonnan előállított azonosítóra. Ezt válaszként a `mysql_insert_id()` függvénytől kaphatjuk meg. Most, hogy elláttuk az új látogatót egy azonosítóval, nincs más hátra, mint kibővíteni a tömböt az azonosítóval, amely így már megfelel az adatbázisrekordnak, amit az imént létrehoztunk.

Végső lépésként létrehozzuk a `vendeg_azon` nevű sütit egy `setcookie()` hívással és visszaadjuk a `$vendeg_adatok` tömböt a hívó programnak.

Legközelebb, amikor a látogató működésbe hozza a programot, a PHP már látni fogja a `$vendeg_azon` változón keresztül a `vendeg_azon` sütit. A program ezt úgy értékeli, hogy egy régi ismerős visszatéréséről van szó és ennek megfelelően frissíti a `vendeg_naplo` táblát, majd üdvözli a felhasználót.

A frissítés elvégzéséhez meg kell még vizsgálnunk, hogy a program futását egy folyamatban lévő böngészés eredményezte vagy új látogatás első lépéséről van szó. Ennek eldöntésében egy globális változó lesz segítségünkre, mely egy másodpercben mért időtartamot tárol. Ha a program által érzékelt legutolsó letöltés ideje az előbb említett időtartamnál régebbi, akkor a mostani letöltés egy újabb látogatás kezdetének tekintendő, ellenkező esetben a folyamatban lévő látogatás részének tekintjük.

A 19.3. példa a `regivendeg()` függvénnyel bővíti a 19.2. programot.

19.3 program Felhasználókövető program sütik és MySQL adatbázis felhasználásával

```
 1: <?php
 2: $lhossz = 300; // a látogatás hossza 5 perc,
                      másodpercben kifejezve
 3: $adatbazis = mysql_connect( "localhost", "janos",
                  "majomkenyerfa" );
 4: if ( ! $adatbazis )
 5:     die( "Nem lehet a mysqld-hez kapcsolódni!" );
 6: mysql_select_db( "minta"", $adatbazis ) or
    die ( mysql_error() );
 7: if ( ! isset ( $vendeg_azon ) )
 8:     $vendeg_allapot = ujvendeg( $adatbazis );
 9: else
10:     {
11:     $vendeg_allapot = regivendeg( $adatbazis,
                            $vendeg_azon, $lhossz );
12:     print "Örülünk, hogy ismét körünkben
               üdvözölhetjük, $vendeg_azon<P>";
13:     }
14:  function ujvendeg( $adatbazis )
15:      {
16:      // egy új vendég!
17:      $vendeg_adatok = array (
18:                    "elso_latogatas" => time(),
19:                    "utolso_latogatas" => time(),
20:                    "latogatas_szam" => 1,
21:                    "ossz_ido" => 0,
22:                    "ossz_letoltes" => 1
23:                    );
24:      $lekerdezes = "INSERT INTO vendeg_naplo
                         ( elso_latogatas,
25:                       utolso_latogatas,
26:                       latogatas_szam,
27:                       ossz_ido,
28:                       ossz_letoltes ) ";
29:     $lekerdezes .= "values ( ".
              $vendeg_adatok["elso_latogatas"].", ".
30:           $vendeg_adatok["utolso_latogatas"].", ".
31:           $vendeg_adatok["latogatas_szam"].", ".
32:           $vendeg_adatok["ossz_ido"].", ".
33:           $vendeg_adatok["ossz_letoltes"]." )";
34:      $eredmeny = mysql_query( $lekerdezes );
35:      $vendeg_adatok["vendeg_azon"] =
                      mysql_insert_id();
```

19.3 program folytatás

```
36:       setcookie( "vendeg_azon",
                     $vendeg_adatok["vendeg_azon"],
37:                  time()+(60*60*24*365*10), "/" );
38:       return $vendeg_adatok;
39:       }
40: function regivendeg( $adatbazis, $vendeg_azon, $lhossz )
41:      {
42:      // egy ismerős vendég,
            aki már járt nálunk korábban!
43:      $lekerdezes = "SELECT * FROM vendeg_naplo WHERE
                        vendeg_azon=$vendeg_azon";
44:      $eredmeny = mysql_query( $lekerdezes );
45:      if ( ! mysql_num_rows( $eredmeny ) )
46:           // süti van ugyan, de azonosító nincs — tehát
                mégiscsak új vendég
47:              return ujvendeg( $adatbazis );
48:      $vendeg_adatok = mysql_fetch_array( $eredmeny,
                          $adatbazis );
49:      // ez most egy újabb letöltés, tehát növeljük
            a számlálót
50:      $vendeg_adatok["ossz_letoltes"]++;
51:      if ( ( $vendeg_adatok["utolso_latogatas"]
                + $lhossz ) > time() )
52:             // még mindig ugyanaz a látogatás,
                tehát növeljük az összidőt.
53:             $vendeg_adatok["ossz_ido"] +=
54:           ( time() - $vendeg_adatok["utolso_latogatas"] );
55:      else
56:           // ez már egy új látogatás,
57:           $vendeg_adatok["latogatas_szam"]++;
58:      // ennek megfelelően módosítjuk az adatbázist
59:      $lekerdezes  = "UPDATE vendeg_naplo SET
                         utolso_latogatas=".time().",
60:                 "latogatas_szam=".$vendeg_adatok
                    ["latogatas_szam"].", ".
61:             "ossz_ido=".$vendeg_adatok["ossz_ido"].", ".
62:             "ossz_letoltes=".
                $vendeg_adatok["ossz_letoltes"]." ";
63:      $lekerdezes .= "WHERE vendeg_azon=$vendeg_azon";
64:      $eredmeny = mysql_query( $lekerdezes );
65:      return $vendeg_adatok;
66:      }
67: ?>
```

Látható, hogy a programot az $lhossz nevű változóval bővítettük. Ebben adjuk meg azt az időtartamot, amely alapján megítéljük, hogy a letöltés mely látogatáshoz tartozik. A $vendeg_azon változó jelenlétének érzékelése azt jelenti számunkra, hogy kaptunk egy sütit. Erre válaszul meghívjuk a regivendeg() függvényt, átadva annak az adatbázisra hivatkozó, a $vendeg_azon és az $lhossz változót, amit 300 másodpercre, azaz 5 percre állítottunk.

A regivendeg() függvény első lépésként lekérdezi a felhasználóról tárolt adatokat. Ezt egyszerűen úgy tesszük, hogy kikeressük a vendeg_naplo tábla azon sorát, melyben a vendeg_azon mező egyenlő a $vendeg_azon változó értékével. A mysql_query() által visszaadott sorok számát megtudhatjuk, ha meghívjuk a mysql_num_rows() függvényt az eredménysorok azonosítójával. Ha a mysql_num_rows() válasza 0, akkor mégis új felhasználóról van szó, tehát meg kell hívnunk az ujvendeg() függvényt.

Tegyük fel, hogy találtunk egy megfelelő sort, melynek azonosítója egyezik a süti értékével. Ezt a sort a mysql_fetch_array() függvénnyel emelhetjük át egy PHP tömbbe (esetünkben a $vendeg_adatok nevűbe). A program mostani futása egy oldal letöltésének következménye, tehát a $vendeg_adatok["ossz_letoltes"] változó értékét eggyel növelnünk kell, hogy rögzítsük ezt a tényt.

Megvizsgáljuk, hogy a $vendeg_adatok["utolso_latogatas"] és az $lhossz összege a pillanatnyi időnél későbbi időpont-e. Ha későbbi, akkor az azt jelenti, hogy a legutóbbi letöltés óta kevesebb, mint $lhossz idő telt el, vagyis a legutóbbi látogatás még folyamatban van és ez a letöltés még annak a része. Ennek tényét úgy őrizzük meg, hogy a legutóbbi letöltés óta eltelt időt hozzáadjuk a $vendeg_adatok["ossz_ido"] változóhoz.

Ha úgy találtuk, hogy már új látogatásról van szó, egyszerűen növeljük a $vendeg_adatok["latogatas_szam"] változó értékét.

Befejezésül a $vendeg_adatok tömb tartalmával frissítjük a vendeg_naplo táblát és a hívó program is megkapja ezt a tömböt. Bár ezt külön nem emeltük ki, természetesen az utolso_latogatas mezőt is a pillanatnyi időhöz igazítottuk.

Most, hogy a feladaton túl vagyunk, írhatunk egy kis programot, hogy ellenőrizzük, az elmélet valóban működik-e a gyakorlatban. Elkészítjük az allapotMegjelenites() függvényt, amely kiszámolja a lekérő felhasználó átlagértékeit és megjeleníti azokat a böngészőben. A valóságban persze komolyabb megjelenésű, átfogóbb képet kell adnunk a látogatókról, de a lényeg megértéséhez ez a példa is bőven elegendő. A függvény megvalósítását a 19.4. példában találjuk. A korábbi példák kódjához az include() függvény alkalmazásával férhetünk hozzá.

19.4 program A 19.3-as program által begyűjtött letöltési statisztikák megjelenítése

```
 1: <?php
 2: include("19.3.program.php");
 3: allapotMegjelenites();
 4: function allapotMegjelenites()
 5:      {
 6:      global $vendeg_allapot;
 7:      $letoltesek = sprintf( "%.2f",
 8:                    ($vendeg_allapot["ossz_letoltes"]/
                       /$vendeg_allapot["latogatas_szam"])
                       );
 9:      $idotartam =  sprintf( "%.2f",
10:               ($vendeg_allapot["ossz_ido"]/
                  /$vendeg_allapot["latogatas_szam"]) );
11:      print "<p>Üdvözöljük! Az Ön azonosítója".
                $vendeg_allapot["vendeg_azon"]."</p>\n\n";
12:      print "<p>Az Ön látogatásainak száma
13:           $vendeg_allapot["latogatas_szam"]."</p>\n\n";
14:      print "<p>Letöltések átlagos száma
                látogatásonként: $letoltesek</p>\n\n";
15:      print "<p>Átlagos látogatási időtartam: $idotartam
                másodperc</p>\n\n";
16:      }
17: ?>
```

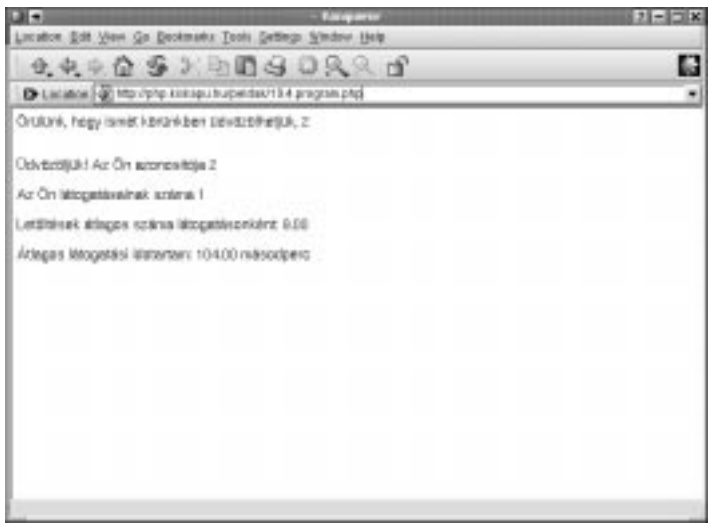
A 19.1 ábra a 19.4 program kimenetét mutatja. Az `include()` sorral biztosítottuk, hogy a felhasználókövető kód lefusson. Hasonló sorral kell bővítenünk webhelyünk minden olyan oldalát, melyet be szeretnénk vonni a statisztikába. Az `allapotMegjelenites()` függvény a `$vendeg_allapot` globális változó alapján dolgozik. Ezt vagy az `ujvendeg()`, vagy a `regivendeg()` függvény hozza létre és tölti fel a `vendeg_naplo` tábla megfelelő sorának értékeivel.

Azt, hogy egy felhasználó látogatásonként átlagosan hányszor kattintott olyan hivatkozásra, mely a webhely valamelyik oldalára mutatott, úgy számoljuk ki, hogy elosztjuk a `$vendeg_allapot["ossz_letoltes"]` változót a látogatások számával. Ugyanezzel az értékkel a `$vendeg_allapot["ossz_ido"]`-t elosztva a látogatások átlagidejét kaphatjuk meg. Ezeket a számokat az `sprintf()` segédletével kerekítjük két tizedesjegyre, majd mondatokká formálva megjelenítjük azokat a felhasználónak.

Természetesen a példát úgy is bővíthetnénk, hogy a program rögzítse a felhasználó érdeklődési területeit a webhelyen belül, de akár naplóztathatnánk a látogatáshoz használt böngésző típusát vagy a látogatók IP címeit is. Hogy mire lehet mindezt használni? Könnyíthetjük felhasználóink eligazodását webhelyünkön, úgy, hogy böngészési szokásaikat kielemezve, vastag betűvel kiemeljük a számukra feltehetőleg különösképp érdekes mondanivalót.

19.1. kép

A használati statisztika megjelenítése



Lekérdező karakterláncok használata

A sütik hatalmas hátránya azok felhasználó-függősége. Nem csak a felhasználó kénye-kedvének vagyunk kitéve, ha sütiket használunk, de a felhasználó böngészőjétől is függ használatuk. A felhasználó letilthatja a sütik használatát, a böngészők pedig nem mindig egyformán valósítják meg a szabványt. Egyikük-másikuk a sütik kezelése területén dokumentált hibákkal is rendelkezik. Ha csak egy látogatás folyamán szeretnénk állapot-adatokat tárolni, jobban járunk, ha egy sokkal hagyományosabb megoldáshoz folyamodunk.

Ha egy űrlapot a `GET` eljárással küldünk el, akkor annak mezőit és azok értékeit URL kódolásban hozzáillesztjük ahhoz a címhez (URL-hez), ahová az űrlapot küldjük. Az értékek ekkor elérhetők lesznek a kiszolgáló, illetve az általa futtatott programok számára. Vegyünk példaként egy olyan űrlapot, amely egy `felhasznalo` és egy `nev` mezőt tartalmaz. Ekkor a lekérés a következőképp fog festeni:

```
http://php.kiskapu.hu/proba5.php?nev=
  ➥ 344343&felhasznalo=Szy+Gyorgy
```

Minden mező nevét egy egyenlőségjel (=) választja el annak értékétől; ezeket a név–érték párokat pedig ÉS jelek (&) határolják. A PHP visszafejti a jeleket, a talált adatokat a $HTTP_GET_VARS asszociatív tömbben teszi elérhetővé a program számára, és emellett a mezők nevével azonos nevű globális változókat is létrehoz, a megfelelő tartalommal. A felhasznalo nevű GET változóhoz így az alábbi két módon férhetünk hozzá:

```
$HTTP_GET_VARS["felhasznalo"];
$felhasznalo;
```

A GET lekérések szerencsére nem csak űrlapokkal kapcsolatban használhatók; viszonylag könnyen készíthetünk mi is ilyen karakterláncokat, így a fontos adatokat könnyedén továbbíthatjuk lapról lapra.

Lekérdező karakterlánc készítése

Alapvető feladatunk a lekérdező karakterláncok készítésekor, hogy a kulcs–érték párokat URL formára alakítsuk. Tegyük fel, hogy egy címet (URL-t) szeretnénk átadni egy lapnak paraméterként, azaz egy lekérdező karakterlánc részeként. Erre a PHP urlencode() függvénye használható, mely egy tetszés szerinti karakterláncot vár és annak kódolt változatával tér vissza:

```
print urlencode("http://www.kiskapu.hu");
// azt írja ki, hogy http%3A%2F%2Fwww.kiskapu.hu
```

Az URL kódolású szövegek előállítása így nem okoz gondot, akár saját GET lekéréseket is összerakhatunk. A következő kódrészlet két változóból épít fel egy lekérdező karakterláncot:

```
<?php
$erdeklodes = "sport";
$honlap = "http://www.kiskapu.hu";
$lekerdezes = "honlap=".urlencode( $honlap );
$lekerdezes .= "&erdeklodes=".urlencode( $erdeklodes );
?>
<A HREF="masiklap.php?<?print $lekerdezes ?>">Gyerünk!</A>
```

Ennek hatására a böngészőhöz az URL már a kódolt lekérdező karakterlánccal bővítve jut el:

```
masiklap.php?honlap=http%3A%2F%2Fwww.kiskapu.hu&
 ➥ erdeklodes=sport
```

A `honlap` és az `erdeklodes` paraméterek a `masiklap.php` programon belül ugyanilyen nevű, URL-ből visszafejtett globális változókként lesznek elérhetők.

Ez így a problémának kissé kezdetleges megoldása, mert a változónevek kódba ágyazásával megnehezítjük a kód újrahasznosítását. Ahhoz, hogy hatékonyan küldhessünk adatokat lapról-lapra, egyszerűvé kell tennünk a lekérdező karakterláncokat tartalmazó hivatkozások készítését. Ez különösen fontos, ha meg szeretnénk tartani a PHP azon hasznos tulajdonságát, miszerint a nem kifejezetten programozók is könnyedén eligazodhatnak rajta.

A 19.5. példa egy olyan `lsztring()` nevű függvény megvalósítását tartalmazza, amely a kapott asszociatív tömb alapján elkészít és visszaad egy lekérdező karakterláncot.

19.5. program Lekérdező karakterláncot készítő függvény

```
 1: <html>
 2: <head>
 3: <title>19.5 program Lekérdező karakterláncot készítő
          függvény</title>
 4: </head>
 5: <body>
 6: <?php
 7: function lsztring( $lekerdezes )
 8:     {
 9:     global $QUERY_STRING;
10:     if ( ! $lekerdezes ) return $QUERY_STRING;
11:     $lsztring = "";
12:     foreach( $lekerdezes as $kulcs => $ertek )
13:         {
14:         if ( strlen( $lsztring ) ) $lsztring .= "&";
15:         $lsztring .= urlencode( $kulcs ) .
                        "=" . urlencode( $ertek );
16:         }
17:     return $lsztring;
18:     }
19: $lekerdezes = array ( "nev" => "Kis Lajos Bence",
20:                "erdeklodes" => "Filmek (főleg
                   művészfilmek)",
21:           "honlap" => "http://www.kiskapu.hu/lajos/"
22:           );
23: print lsztring( $lekerdezes );
```

19.5. program (folytatás)

```
24: // azt írja ki, hogy nev=Kis+Lajos+Bence&erdeklodes=
       Filmek+%28f%F5leg+m%FBv%E9szfilmek
25: // %29&honlap=http%3A%2F%2Fwww.kiskapu.hu%2Flajos%2F
26: ?>
27: <p>
28: <a href="masiklap.php?<? print lsztring($lekerdezes)
            ?>">Gyerünk!</a>
29: </p>
30: </body>
31: </html>
```


Az `lsztring()` egy asszociatív tömböt vár paraméterként, amit jelen esetben a `$lekerdezes` változó képében adunk át neki. Ha a `$lekerdezes` még nem kapott volna értéket, a programot tartalmazó lap lekérdező karakterláncát adjuk vissza, amit a `$QUERY_STRING` változóval érhetünk el. Ezzel oldhatjuk meg azt, hogy a `GET`-tel továbbított, módosítatlan űrlapadatok könnyedén továbbadhatóak legyenek más lapok számára.

Ha a `$lekerdezes` tömb nem üres, a visszaadandó lekérdező karakterláncot az `$lsztring` változóba helyezzük.

A `foreach` segítségével sorra vesszük a tömb elemeit, úgy, hogy azok a `$kulcs` és `$ertek` változókon keresztül legyenek elérhetők.

A kulcs–érték párokat `&` jellel kell elválasztanunk, így ha nem a ciklus első lefutásánál tartunk – azaz az `$lsztring` változó már nem üres –, ezt a karaktert is hozzátesszük a lekérdező karakterlánchoz.

A `$kulcs` és `$ertek` változókat az `urlencode()` függvénnyel kódolva és egy egyenlőségjellel (`=`) elválasztva adjuk hozzá az `$lsztring` végéhez.

Ezek után már csak vissza kell adnunk az összeállított, kódolt karakterláncot.

E függvény alkalmazásával pár sornyi PHP kód is elég, hogy adatokat adhassunk át egyes lapjaink között.

Összefoglalás

Ezen az órán a weblapok közti információátadás két módjával ismerkedhettünk meg. Segítségükkel több oldalból álló webes alkalmazások alakíthatók ki, melyek a felhasználók igényeit is figyelembe veszik.

Tudjuk, hogyan használjuk a `setcookie()` függvényt sütik létrehozására. Ennek kapcsán láthattuk, hogyan tárolhatók adatok a webhely felhasználóinak szokásairól sütik és egy adatbázis felhasználásával. Láttunk lekérdező karakterláncokat is. Tudjuk, hogyan kell azokat kódolni és egy hasznos eszközt is a magunkénak mondhatunk, amely jelentősen egyszerűsíti ezek használatát.

Kérdések és válaszok

Van e valamiféle komoly biztonságot vagy személyiségi jogokat sértő velejárója a sütik használatának?
A kiszolgáló csak a saját tartományán belül levő gépek sütijeihez férhet hozzá. Azon túl, hogy a süti a felhasználó fájlrendszerében tárolódik, azzal semmiféle további kapcsolatba nem kerül. Lehetőség van azonban arra, hogy sütit állítsunk be egy kép letöltésének válaszaként. Tehát ha sok webhely jelentet meg egy külső reklámszolgáltató vagy számláló programot oldalain, ezek szolgáltatója képes lehet a látogatók követésére különböző webhelyek között is.

Műhely

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

Kvíz

1. Milyen függvénnyel vehetjük rá a weboldalunkat letöltő felhasználó böngészőjét, hogy létrehozzon egy sütit és tárolja azt?

2. Hogyan törölhetjük a sütiket?

3. Milyen függvénnyel tehetünk egy tetszőleges karakterláncot alkalmassá arra, hogy lekérdező karakterláncban szerepelhessen?

4. Melyik beépített változó tartalmazza a lekérdező karakterláncot nyers formában?

5. A lekérdező karakterlánc formájában elküldött név–érték párok globális változókként válnak elérhetővé. Létezik azonban egy tömb is, amely tartalmazza őket. Mi ennek a tömbnek a neve?

Feladatok

1. Tegyük lehetővé, hogy a webhelyünket meglátogató felhasználó beállíthassa a weboldalak háttérszínét, valamint hogy megadhassa, milyen néven köszöntsék a webhely oldalai. Oldjuk meg sütikkel, hogy minden oldal köszönjön, illetve a megfelelő háttérszínnel jelenjen meg.

2. Írjuk át az előbbi feladat programját úgy, hogy az adatokat lekérdező karakterláncban tároljuk, ne sütikben.