13. ÓRA

Kapcsolat a külvilággal

Ezen az órán olyan függvényekkel ismerkedünk meg, amelyek a „külvilággal” való érintkezést teszik lehetővé. Az óra során a következőkről tanulunk:

- Környezeti változók – részletesebben
- A HTTP kapcsolat felépítése
- Távoli kiszolgálón levő dokumentumok elérése
- Saját HTTP kapcsolat létrehozása
- Kapcsolódás más hálózati szolgáltatásokhoz
- Levélküldés programból

Környezeti változók

Már találkoztunk néhány környezeti változóval; ezeket a PHP a kiszolgáló segítségével bocsátotta rendelkezésünkre. Néhány ilyen változóval több dolgot is megtudhatunk weboldalunk látogatóiról, arra azonban gondoljunk, hogy lehet, hogy ezek a változók a mi rendszerünkön nem elérhetők, esetleg a kiszolgálóprogram nem támogatja azokat, így használatuk előtt érdemes ezt ellenőrizni. A 13.1. táblázat ezek közül a változók közül mutat be néhányat.

13.1. táblázat Néhány hasznos környezeti változó

Változó	*Leírás*
`$HTTP_REFERER`	A programot meghívó webhely címe.
`$HTTP_USER_AGENT`	Információ a látogató böngészőjéről és rendszeréről.
`$REMOTE_ADDR`	A látogató IP címe.
`$REMOTE_HOST`	A látogató számítógépének neve.
`$QUERY_STRING`	Az a (kódolt) karakterlánc, amely a webcímet kiegészíti (formátuma `kulcs=ertek&masikkulcs=masikertek`). E kulcsok és értékek elérhetővé kell, hogy váljanak programunk részére a megfelelő globális változókban is.
`$PATH_INFO`	Az esetleg a webcímhez illesztett további információ.

A 13.1. példaprogram ezen változók értékét jeleníti meg a böngészőben.

13.1. program Néhány környezeti változó felsorolása

```
1: <html>
2: <head>
3: <title>13.1. program Néhány környezeti változó
          felsorolása</title>
4: </head>
5: <body>
6: <?php
7: $korny_valtozok = array( "HTTP_REFERER",
   "HTTP_USER_AGENT", "REMOTE_ADDR", "REMOTE_HOST",
   "QUERY_STRING", "PATH_INFO" );
```

13.1. program (folytatás)

```
 8: foreach ( $korny_valtozok as $valtozo )
 9:      {
10:      if ( isset( $$valtozo ) )
11:          print "$valtozo: ${$valtozo}<br>";
12:      }
13: ?>
14: </body>
15: </html>
```

Figyeljük meg, hogyan alakítottuk a szövegként tárolt változóneveket valódi változókká. Ezt a módszert a negyedik órán tanultuk.

A 13.1. ábrán a 13.1. kód kimenetét láthatjuk. Az ábrán látható adatokat úgy kaptuk, hogy a programot egy másik oldal hivatkozásán keresztül hívtuk meg. A programot meghívó hivatkozás így nézett ki:

```
<A HREF='13.1.program.php/tovabbi/utvonal?nev=ertek'>gyerünk</A>
```

Amint láthatjuk, a hivatkozás relatív útvonalat használ a `13.1.program.php` meghívására.

13.1. ábra

Néhány környezeti változó kiíratása a böngészőben

Az elérési út azon része, mely programunk nevét követi, (jelen esetben a `/tovabbi/utvonal`) a `$PATH_INFO` változóban áll majd rendelkezésünkre.

A lekérdezés szövegét nem dinamikusan illesztettük a hivatkozásba (`nev=ertek`), de ettől függetlenül az a `$QUERY_STRING` változóban lesz elérhető. A lekérdező karakterláncokkal legtöbbször akkor találkozunk, ha egy `GET` metódust használó űrlap hívja meg a programot, de magunk is előállíthatunk ilyet, hogy segítségével adatokat továbbíthassunk lapról lapra. A lekérdező karakterlánc név–érték párokból áll, melyeket ÉS jel (&) választ el egymástól. Az adatpárok URL kódolású formában kerülnek a webcímbe, hogy a bennük szereplő, a webcímekben nem megengedett vagy mást jelentő karakterek ne okozzanak hibát. Ezt úgy oldották meg, hogy a problémás karaktereket hexadecimális megfelelőjükre cserélik. Bár a teljes karakterlánc elérhető a `$QUERY_STRING` környezeti változóban, nagyon ritkán lehet szükségünk rá, pontosan a kódolt mivolta miatt. De azért is mellőzük a használatát, mert az összes név–érték pár rendelkezésünkre áll globális változók formájában is (jelen esetben például létrejön a `$nev` változó és az értéke `ertek` lesz).

A `$HTTP_REFERER` változó értéke akkor lehet hasznos számunkra, ha nyomon szeretnénk követni, hogy programunkat mely hivatkozáson keresztül érték el. Nagy körültekintéssel kell eljárnunk azonban, mert ez és tulajdonképpen bármely másik környezeti változó is nagyon könnyen „meghamisítható”. (Az óra folyamán azt is megtudhatjuk, hogyan.) A nevet sajnálatos módon a kezdetekkor egy fejlesztő elírta (helyesen `HTTP_REFERRER`-nek kellene neveznünk), módosítása a korábbi változatokkal való összeegyeztethetőség megőrzése érdekében nem történt meg. Ezenkívül nem minden böngésző adja meg ezt az információt, így használatát célszerű elkerülnünk.

A `$HTTP_USER_AGENT` változó szétbontásával rájöhetünk, milyen operációs rendszert, illetve böngészőt használ a látogató, de tudnunk kell, hogy ezek az adatok is hamisíthatók. A változóban elérhető értéket akkor használhatjuk, ha a böngésző típusától vagy a változattól függően más és más HTML kódot vagy JavaScriptet szeretnénk elküldeni a látogatónak. A tizenhetedik és tizennyolcadik órában mindent megtanulunk arról, hogyan nyerhetjük ki a számunkra fontos adatokat ebből a karakterláncból.

A `$REMOTE_ADDR` változó a látogató IP címét tartalmazza, így a webhely látogatóinak azonosítására használható. Ne feledjük azonban, hogy a felhasználók IP címe többnyire nem állandó, hiszen az internetszolgáltatók általában dinamikusan osztják ki felhasználóiknak a címeket, így az minden csatlakozás során más lehet, tehát ugyanaz az IP cím különböző látogatókat jelenthet és a különböző IP címek is takarhatnak egyetlen felhasználót.

A `$REMOTE_HOST` változó nem biztos, hogy hozzáférhető; ez a kiszolgáló beállításaitól függ. Ha létezik, a felhasználó számítógépének nevét találhatjuk benne. A változó meglétéhez a kiszolgálónak az IP cím alapján minden kéréskor le kell kérdeznie a gép nevét, ami időigényes feladat, így a kiszolgáló ezen képességét

a hatékonyság kedvéért gyakran letiltják. Ha nem létezne ilyen változó, az információhoz a `$REMOTE_ADDR` segítségével juthatunk hozzá; később azt is látjuk majd, hogyan.

A HTTP ügyfél-kiszolgáló kapcsolat rövid ismertetése

A kiszolgáló és az ügyfél közti adatforgalom részletes ismertetése túlmutat könyvünk keretein, többek között azért, mert a PHP ezen részletek szakszerű kezeléséről is gondoskodik helyettünk. A folyamatot azonban nem hiábavaló alapjaiban megismerni, mert szükségünk lehet rá, ha olyan programot szeretnénk írni, amely weboldalakat tölt le vagy webcímek állapotát ellenőrzi.

ÚJDONSÁG

A *HTTP* jelentése *hiperszöveg-átviteli protokoll*. Nem más, mint szabályok halmaza, melyek előírják, hogy az ügyfél milyen kérésekkel fordulhat a kiszolgálóhoz és az milyen válaszokat kell, hogy adjon. Mind az ügyfél, mind a kiszolgáló információt szolgáltat magáról és a küldött adatokról. Ezen információk legtöbbjét a PHP-ből környezeti változókon keresztül érhetjük el.

A kérés

Az ügyfél szigorú szabályok szerint kérhet adatokat a kiszolgálótól. A kérés legfeljebb három részből állhat:

- A kérés sora
- Fejléc
- Törzs

A legfontosabb a kérés sora. Itt határozzuk meg a kérés fajtáját (`GET`, `HEAD` vagy `POST`), a kért dokumentum címét és az átviteli protokoll változatszámát (`HTTP/1.0` vagy `HTTP/1.1`). Egy jellemző kérés a következőképpen néz ki:

```
GET /egy_dokumentum.html HTTP/1.0
```

Ekkor az ügyfél egy `GET` kérést kezdeményez. Más szóval lekér egy dokumentumot, de adatokat nem küld. (Valójában kisebb mennyiségű adat küldésére a `GET` kérés alkalmazásakor is van lehetőség, ha azt lekérdező karakterlánc formájában hozzáírjuk az URL végéhez.) A `HEAD` módszer akkor alkalmazandó, ha nem magára a dokumentumra, csak annak tulajdonságaira vagyunk kíváncsiak, végül a `POST` kérés arra szolgál, hogy adatokat küldjünk az ügyféltől a kiszolgálónak. Ez legtöbbször HTML űrlapok esetében használatos.

A kifogástalan GET kéréshez egyetlen kérősor elküldése is elegendő. A kérés végét úgy tudathatjuk a kiszolgálóval, hogy egy üres sort küldünk neki.

A legtöbb ügyfél (általában böngészőprogram) a kérősoron kívül küld egy fejléc részt is, amely név–érték párokból áll. Az oldal lekérésével érkezett fejlécek legtöbbjéhez környezeti változók formájában hozzá is férhetünk programunkból. Az ügyfél fejlécének minden sora egy kettősponttal elválasztott névből és értékből áll. A 13.2 táblázat néhány lehetséges fejléc-nevet sorol fel.

13.2 táblázat Néhány fejléc-kulcs

Név	*Leírás*
`Accept`	Az ügyfél által kezelhető dokumentumtípusok listája.
`Accept-Encoding`	Az ügyfél számára elfogadható kódolási és tömörítési formátumok listája.
`Accept-Charset`	Az ügyfél által előnyben részesített nemzetközi karakterkészlet.
`Accept-Language`	Az ügyfél által előnyben részesített nyelv (a magyar nyelv esetében „hu”).
`Host`	Az ügyfél kérése által megcímzett gép neve. Egyes kiszolgálók csak látszólagos (virtuális) kiszolgálók, a beérkező kéréseket nem önálló számítógép kezeli. Ilyen esetekben komoly jelentősége van ennek az adatnak.
`Referer`	Azon dokumentum címe, melynek egyik hivatkozása alapján a kérés létrejött.
`User-Agent`	Az ügyfélprogram típusa és változatszáma.

A GET és a HEAD eljárásoknál a kérést a fejléc és az utána következő üres sor zárja, a POST típusnál azonban az üres sort az üzenet törzse követi. A törzs tartalmazza a kiszolgálónak szóló összes adatot, jobbára URL kódolású név–érték párok, a lekérdező karakterláncokhoz hasonlóan.

A 13.2 példában egy jellemző, mindennapos kérést mutatunk be, melyet egy Netscape 4.7 kezdeményezett.

13.2. példa Jellemző Netscape-kérés

```
GET / HTTP/1.0
Referer: http://www.linuxvilag.hu/index.html
Connection: Keep-Alive
User-Agent: Mozilla/4.7 [en] (X11; I; Linux 2.2.13 i686)
Host: www.kiskapu.hu
Accept: image/gif, image/x-xbitmap, image/jpeg,
        image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

A válasz

Miután a kiszolgáló fogadta az ügyfél kérését, választ küld. A válasz általában a következő három részből tevődik össze:

- Állapot
- Fejléc
- Törzs

Amint látjuk, a kérés és a válasz szerkezete igen hasonló. A fejléc bizonyos sorai tulajdonképpen ügyfél és kiszolgáló által küldve egyaránt megállják helyüket, nevezetesen azok, amelyek a törzsre vonatkoznak.

Az állapot sor a kiszolgáló által használt protokollt (`HTTP/1.0` vagy `HTTP/1.1`) adja meg, illetve egy válaszkódot és egy azt magyarázó szöveget.

Számos válaszkód létezik, mindegyik a kérés sikerességéről vagy sikertelenségéről ad információt. A 13.3-as táblázat a leggyakoribb kódok jelentését tartalmazza.

13.3 táblázat Néhány válaszkód

Kód	*Szöveg*	*Leírás*
`200`	`OK`	A kérés sikeres volt és a törzsben megtalálható a válasz.
`301`	`Moved Permanently`	A kért adat nem található a kiszolgálón, de a fejlécben megtalálható annak új helye.
`302`	`Moved Temporarily`	A kért adatot ideiglenesen áthelyezték, de a fejléc elárulja, hogy hová.
`404`	`Not Found`	A kért adat nem található a megadott címen.
`500`	`Internal Server Error`	A kiszolgáló vagy egy CGI program komoly problémába ütközött a kérés teljesítése során.

Ennek megfelelően a jellegzetes válaszsor a következőképpen fest:

```
HTTP/1.1 200 OK
```

A fejléc rész a válaszfejléc soraiból áll, melyek formátuma a kérésfejléc soraihoz hasonló. A 13.4 táblázat a legsűrűbben alkalmazott fejlécelemekből szemezget.

13.4 táblázat A kiszolgáló válaszfejlécének leghétköznapibb elemei

Név	*Leírás*
`Date`	Az aktuális dátum
`Server`	A kiszolgáló neve és változatszáma
`Content-Type`	A törzsben szereplő adat MIME típusa
`Content-Length`	A törzs mérete bájtban
`Location`	Egy alternatív dokumentum teljes elérési útja (kiszolgálóoldali átirányítás esetén)

Miután a kiszolgáló elküldte a fejlécet és a szokásos üres sort, elküldi a törzset (a kérésben tulajdonképpen igényelt dokumentumot). A 13.3 példa egy jellemző választ mutat be.

13.3. példa A kiszolgáló válasza

```
 1: HTTP/1.1 200 OK
 2: Date: Sun, 30 Jan 2000 18:02:20 GMT
 3: Server: Apache/1.3.9 (Unix)
 4: Connection: close
 5: Content-Type: text/html
 6:
 7: <html>
 8: <head>
 9: <title>13.3. lista A kiszolgáló válasza</title>
10: </head>
11: <body>
12: Üdvözlet!
13: </body>
14: </html>
```

Dokumentum letöltése távoli címről

Bár a PHP kiszolgálóoldali nyelv, esetenként ügyfélként viselkedve adatokat kérhet egy távoli címről és a kapott adatokat programunk rendelkezésére bocsáthatja. Ha már jártasak vagyunk a kiszolgálón lévő fájlok kezelésében, nem okozhat komoly gondot a távoli helyen lévő adatok lekérése sem. Az a helyzet ugyanis, hogy a kettő között – formailag – semmi különbség nincs. Az `fopen()`-nel ugyanúgy megnyithatunk egy webcímet, ahogy azt egy fájl esetében tennénk. A 13.4. példaprogram egy távoli kiszolgálóhoz kapcsolódva adatokat kér le, majd megjeleníti azokat a böngészőben.

13.4. program Weboldal letöltése és megjelenítése az fopen()-nel

```
 1: <html>
 2: <head>
 3: <title>13.4. program Weboldal letöltése és megjele-
nítése az fopen()-nel</title>
 4: </head>
 5: <body>
 6: <?php
 7: $weboldal = "http://www.php.net/index.php";
 8: $fajlmutato = fopen( $weboldal, "r" ) or
                  die("A $weboldal nem nyitható meg");
```

13.4. program (folytatás)

```
 9: while ( ! feof( $fajlmutato ))
10:     print fgets( $fajlmutato, 1024 );
11: ?>
12: </body>
13: </html>
```

Ha megnézzük ezt az oldalt, akkor a PHP honlapját kell látnunk, azzal a kis különbséggel, hogy a képek nem jelennek meg. Ennek oka, hogy az oldalon található `IMG` hivatkozások általában relatívak, így az oldal megjelenítésekor a böngésző a mi kiszolgálónkon keresi azokat. Ezen úgy segíthetünk, hogy a `HEAD` HTML elemet az alábbi sorral bővítjük:

```
<base href="http://www.php.net/index.php">
```

Az esetek többségében azonban nem a letöltött adatok megjelenítése, hanem a feldolgozás a feladat.

Az `fopen()` egy fájlmutatóval tér vissza, ha sikerült felépítenie a kapcsolatot és `false` (hamis) értékkel, ha nem jött létre a kapcsolat vagy nem találta a fájlt. A kapott fájlmutató a továbbiakban ugyanúgy használható az olvasáshoz, mint a helyi fájlok kezelése esetében. A PHP a távoli kiszolgálónak úgy mutatkozik be, mint egy PHP ügyfél. A szerző rendszerén a PHP a következő kérést küldi el:

```
GET /index.php  HTTP/1.0
Host: www.php.net
User-Agent: PHP/4.0.3
```

Ez a megközelítés elég egyszerű, így az esetek többségében elegendő, ha ezt alkalmazzuk weblapok letöltéséhez. Szükségünk lehet viszont más hálózati szolgáltatásokhoz való kapcsolódásra, vagy épp csak többet szeretnénk megtudni a dokumentumról, elemezve annak fejlécét. Ekkor már más eszközökhöz kell folyamodnunk. Hogy ezt hogyan tehetjük meg? Nos, az óra folyamán lesz még erről szó.

Átalakítás IP címek és gépnevek között

Ha kiszolgálónk nem is bocsátja rendelkezésünkre a látogató gépének nevét a `$REMOTE_HOST` változóban, a látogató címéhez minden bizonnyal hozzájuthatunk a `$REMOTE_ADDR` változóból. A változóra alkalmazva a `gethostbyaddr()` függvényt megtudhatjuk a látogató gépének nevét. A függvény egy IP címet ábrázo-

ló karakterláncot vár paraméterként és az annak megfelelő gépnévvel tér vissza. Ha hiba lép fel a név előállítása során, kimenetként a beadott IP címet kapjuk meg változatlanul. A 13.5-ös program a `$REMOTE_HOST` hiánya esetén a `gethostbyaddr()` függvény segítségével előállítja a felhasználó gépének nevét.

13.5. program Gépnév lekérdezése a gethostbyaddr() segédletével

```
 1: <html>
 2: <head>
 3: <title>13.5. program Gépnév lekérdezése
           a gethostbyaddr() segédletével</title>
 4: </head>
 5: <body>
 6: <?php
 7: if ( isset( $REMOTE_HOST ) )
 8:     print "Üdvözöljük $REMOTE_HOST<br>";
 9: elseif ( isset ( $REMOTE_ADDR ) )
10:        print " Üdvözöljük
                ".gethostbyaddr( $REMOTE_ADDR )."<br>";
11: else
12:     print " Üdvözöljük, akárki is Ön.<br>";
13: ?>
14: </body>
15: </html>
```


Ha a `$REMOTE_HOST` változó létezik, egyszerűen megjelenítjük annak értékét. Ha nem létezik, megpróbáljuk a `$REMOTE_ADDR` alapján a `gethostbyaddr()` függvénnyel előállítani a gépnevet. Ha minden kísérletünk csődöt mondott, általános üdvözlő szöveget jelenítünk meg.

Egy gép címének a név alapján történő meghatározásához a `gethostbyname()` függvényt használhatjuk. Ennek paramétere egy gépnév, visszatérési értéke pedig hiba esetén ugyanez a gépnév, sikeres átalakítás esetén viszont a megnevezett gép IP címe.

Hálózati kapcsolat létesítése

Eddig minden feladatunkat könnyen meg tudtuk oldani, mert a PHP a távoli weblapok lekérését közönséges fájlkezeléssé egyszerűsítette számunkra. Néha azonban nagyobb felügyeletet kell gyakorolnunk egy-egy hálózati kapcsolat felett, de legalábbis a szokásosnál több jellemzőjét kell ismernünk.

A hálózati kiszolgálók felé az `fsockopen()` függvénnyel nyithatunk meg egy kapcsolatot. Paramétere egy IP cím vagy gépnév, egy kapuszám és két változóhivatkozás. Változóhivatkozásokat úgy készítünk, hogy ÉS (`&`) jelet írunk a változó neve elé. Megadhatunk továbbá egy nem kötelező időkorlát-paramétert is, amely annak az időtartamnak a hossza (másodpercben), ameddig a függvény vár a kapcsolat létrejöttére. Ha sikerült megteremtenie a kapcsolatot, egy fájlmutatóval tér vissza, egyébként pedig `false` értéket ad.

A következő kódrészlet kapcsolatot nyit egy webkiszolgáló felé.

```
$fajlmutato = fsockopen( "www.kiskapu.hu", 80, &$hibakod,
➥ &$hibaszoveg, 30 );
```

A webkiszolgálók általában a 80-as kapun várják a kéréseket.

Az első hivatkozott változó, a `$hibaszam` sikertelen művelet esetén egy hibakódot tartalmaz, a `$hibaszoveg` pedig bővebb magyarázattal szolgál a hibáról.

Miután visszakaptuk a kapcsolat fájlmutatóját, a kapcsolaton keresztül az `fputs()` és `fgets()` függvények segítségével írhatunk és olvashatunk is, ahogy ezt egy közönséges fájl esetében is tehetjük. Munkánk végeztével a kapcsolatot az `fclose()` függvénnyel bonthatjuk.

Most már elegendő tudás birtokában vagyunk ahhoz, hogy kapcsolatot létesíthessünk egy webkiszolgálóval. A 13.6-os programban megnyitunk egy HTTP kapcsolatot, lekérünk egy fájlt, majd egy változóban tároljuk azt.

13.6. program Weboldal lekérése az fsockopen() használatával

```
 1: <html>
 2: <head>
 3: <title>13.6. program Weboldal lekérése az fsockopen()
           használatával</title>
 4: </head>
 5: <body>
 6: <?php
 7: $kiszolgalo = "www.kiskapu.hu";
 8: $lap = "/index.html";
 9: $fajlmutato = fsockopen( "$kiszolgalo", 80,
                  &$hibaszam, &$hibaszoveg);
10: if ( ! $fajlmutato )
```

13.6. program (folytatás)

```
11: die ( "Nem sikerült kapcsolódni a $kiszolgalo
          géphez:\nA hiba kódja: $hibaszam\nA hiba
          szövege: $hibaszoveg\n" );
12:
13: $lekeres = "GET $lap HTTP/1.0\r\n";
14: $lekeres .= "Host: $kiszolgalo\r\n";
15: $lekeres .= "Referer:
                 http://www.linuxvilag.hu/index.html\r\n";
16: $lekeres .= "User-Agent: PHP browser\r\n\r\n";
17: $lap = array();
18: fputs ( $fajlmutato, $lekeres );
19: while ( ! feof( $fajlmutato ) )
20:     $lap[] = fgets( $fajlmutato, 1024 );
21: fclose( $fajlmutato );
22: print "A kiszolgáló ".(count($lap))."
         sort küldött el!";
23: ?>
24: </body>
25: </html>
```

Figyeljük meg, milyen kérésfejlécet küldtünk a kiszolgálónak. A távoli gép webmestere a fejléc `User-Agent` sorában feltüntetett adatokat látni fogja a webkiszolgáló naplójában. A webmester számára ráadásul úgy fog tűnni, mintha a `http://www.linuxvilag.hu/index.html` címről mutatott volna egy hivatkozás a kért oldalra és mi ennek segítségével kértük volna le az oldalt. Emiatt programjainkban némi fenntartással kell kezelnünk az ezen fejlécekből előálló környezeti változók tartalmát és sokkal inkább segédadatoknak kell azokat tekintenünk, mintsem tényeknek.

Vannak esetek, amikor meg kell hamisítani a fejléceket. Szükségünk lehet például olyan adatokra, amelyeket a kiszolgáló csak Netscape böngészőnek küld el. Ezek elérésének egyetlen módja, ha a `User-Agent` fejlécsorban egy Netscape böngészőre jellemző karakterláncot küldünk. Ennek a webmesterek nem igazán örülnek, hiszen döntéseiket a kiszolgáló gyűjtötte statisztikák alapján kell meghozniuk. Hogy segíthessünk ebben nekik, lehetőleg ne hamisítsuk meg adatainkat, hacsak feltétlenül nem szükséges.

A 13.6-os program nem sokkal bővítette a PHP beépített lapkérő módszereit. A 13.7-es példa azonban már az `fsockopen()`-es módszert alkalmazza egy tömbben megadott weboldalak letöltéséhez és közben a kiszolgáló által visszaadott hibakódokat is ellenőrzi.

13.7. program Az állapotsor megjelenítése webkiszolgálók válaszaiból

```
 1: <html>
 2: <head>
 3: <title>13.7. program Az állapotsor megjelenítése
          webkiszolgálók válaszaiból</title>
 4: </head>
 5: <body>
 6: <?php
 7: $ellenorzendo = array (     "www.kiskapu.hu" =>
                                "/index.html",
 8:              "www.virgin.com" =>
                 "/nincsilyenlap.html",
 9:              "www.4332blah.com" =>
                 "/nemletezogep.html"
10:              );
11: foreach ( $ellenorzendo as $kiszolgalo => $lap )
12:     {
13:     print "Csatlakozási kísérlet a $kiszolgalo géphez
                ...<BR>\n";
14:     $fajlmutato = fsockopen( "$kiszolgalo", 80,
                      &$hibaszam, &$hibaszoveg, 10);
15:     if ( ! $fajlmutato )
16:         {
17:         print "A $kiszolgalo géphez nem sikerült
                      csatlakozni:\n<br>A hiba kódja:
                      $hibaszam\n<br>Szövege: $hibaszoveg\n";
18:         print "<br><hr><br>\n";
19:         continue;
20:         }
21:     print "A $lap oldal fejlécének letöltése ...<br>\n";
22:     fputs( $fajlmutato, "HEAD $lap HTTP/1.0\r\n\r\n" );
23:     print fgets( $fajlmutato, 1024 );
24:     print "<br><br><br>\n";
25:     fclose( $fajlmutato );
26:     }
27: ?>
28: </body>
29: </html>
```

Először létrehozunk egy asszociatív tömböt az ellenőrzendő kiszolgálónevekből és az oldalak címeiből. Ezeket sorra vesszük a `foreach` utasítás segítségével. Minden elemnél az `fsockopen()`-nel kezdeményezünk egy kapcsolatot az adott címmel,

de a válaszra csak 10 másodpercig várunk. Ha ezen belül nem érkezne válasz, üzenetet jelenítünk meg a böngészőben, majd a `continue` utasítással a következő címre lépünk. Ha a kapcsolatteremtés sikeres volt, kérelmet is küldünk a kiszolgálónak. A kéréshez a `HEAD` eljárást használjuk, mert a lap tartalmára nem vagyunk kíváncsiak. Az `fgets()` segítségével beolvassuk az első sort, az állapotsort.
A jelen példában nem elemezzük a fejléc többi részét, ezért lezárjuk a kapcsolatot az `fclose()` függvénnyel, majd továbblépünk a következő címre.

A 13.2-es ábrán a 13.7. program kimenete látható.

13.2. ábra

Kiszolgálók válaszát megjelenítő program

NNTP kapcsolat létrehozása az fsockopen()-nel

Az `fsockopen()` függvény tetszőleges internetkiszolgálóval képes kapcsolatot teremteni. A 13.8. példában egy NNTP (Usenet) kiszolgálóhoz kapcsolódunk: kiválasztunk egy hírcsoportot, majd megjelenítjük első üzenetének fejlécét.

13.8. program Egyszerű NNTP kapcsolat az fsockopen() használatával

```
1: <html>
2: <head>
3: <title>13.8. program Egyszerű NNTP kapcsolat az
          fsockopen() használatával</title>
4: </head>
5: <body>
6: <?php
7: $kiszolgalo = "news"; // ezt saját hírkiszolgálónk
                  címére kell beállítanunk
```

13.8. program (folytatás)

```
 8: $csoport = "alt.test";
 9: $sor = "";
10: print "<pre>\n";
11: print "— Csatlakozás a $kiszolgalo
          kiszolgálóhoz\n\n";
12: $fajlmutato = fsockopen( "$kiszolgalo", 119,
                             &$hibaszam, &$hibaszoveg, 10 );
13: if ( ! $fajlmutato )
14:     die("Csatlakozás a $kiszolgalo géphez
             sikertelen\n$hibaszam\n$hibaszoveg\n\n");
15: print "— Kapcsolat a $kiszolgalo géppel
          felvéve\n\n";
16: $sor = fgets( $fajlmutato, 1024 );
17: $allapot = explode( " ", $sor );
18: if ( $allapot[0] != 200 )
19:     {
20:     fputs( $fajlmutato, "close" );
21:     die("Hiba: $sor\n\n");
22:     }
23: print "$sor\n";
24: print "— A $csoport kiválasztása\n\n";
25: fputs( $fajlmutato, "group ".$csoport."\n" );
26: $sor = fgets( $fajlmutato, 1024 );
27: $allapot = explode( " ", $sor );
28: if ( $allapot[0] != 211 )
29:     {
30:     fputs( $fajlmutato, "close" );
31:     die("Hiba: $sor\n\n");
32:     }
33: print "$sor\n";
34: print "— Az első üzenet fejlécének lekérése\n\n";
35: fputs( $fajlmutato, "head\n" );
36: $sor = fgets( $fajlmutato, 1024 );
37: $allapot = explode( " ", $sor );
38: print "$sor\n";
39: if ( $allapot[0] != 221 )
40:     {
41:     fputs( $fajlmutato, "close" );
42:     die("Hiba: $sor\n\n");
43:     }
```

13.8. program (folytatás)

```
44: while ( ! ( strpos($sor, ".") === 0 ) )
45:       {
46:       $sor = fgets( $fajlmutato, 1024 );
47:       print $sor;
48:       }
49: fputs( $fajlmutato, "close\n" );
50: print "</pre>";
51: ?>
52: </body>
53: </html>
```

A 13.8. program egy kicsit többet mutat annál, hogyan nyissunk az `fsockopen()`-nel NNTP kapcsolatot. Valós alkalmazás esetén a visszakapott sorok elemzését célszerű egy függvénnyel végezni, egyrészt azért, hogy megszabaduljunk az ismétlésektől, másrészt azért, hogy több adatot is kinyerhessünk a válaszból. Mielőtt azonban újra feltalálnánk a kereket, célszerű, ha megismerkedünk a PHP IMAP függvényeivel, amelyekkel mindez a munka automatizálható.

A fenti programban a `$kiszolgalo` változó tárolja a hírkiszolgáló nevét, a `$csoport` pedig annak a csoportnak a nevét, melyhez kapcsolódni szeretnénk. Ha ki szeretnénk próbálni ezt a programot, javítsuk át a `$kiszolgalo` változó értékét az internetszolgáltatónk által megadott hírkiszolgáló nevére, a `$csoport`-ot kedvenc csoportunkra. Az `fsockopen()`-nel a kiszolgáló 119-es kapujára csatlakozunk, mert ez a hírcsoport szolgáltatás szabványos kapuja. Ha nem kapunk vissza használható fájlmutatót, a `die()` függvény segítségével megjelenítünk egy hibaüzenetet a böngészőben és kilépünk a programból. Kapcsolódás esetén a kiszolgálótól kapnunk kell egy nyugtázó üzenetet, amit az `fgets()` függvénnyel próbálunk fogadni. Ha minden zökkenőmentesen zajlott, a visszakapott szöveg a 200-as állapotkóddal kezdődik. Ennek ellenőrzéséhez az `explode()` függvénnyel a szóközök mentén szétdaraboljuk a `$sor` változót és a darabokat egy tömbbe helyezzük. Az `explode()` függvénnyel részletesebben a tizenhetedik órában foglalkozunk. Ha a kapott tömb első eleme (a nullás indexű) 200, akkor továbblépünk, egyébként pedig befejezzük a programot.

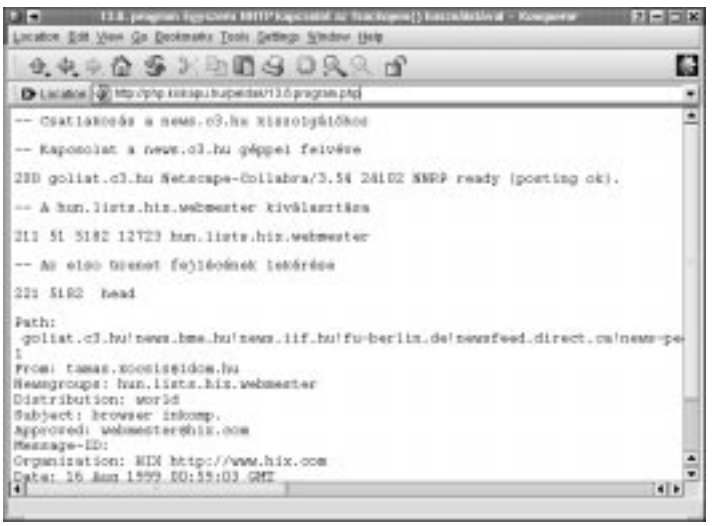
Ha minden az elvárásoknak megfelelően haladt, akkor a `"group"` paranccsal kiválasztjuk a kívánt hírcsoportot. Ha ez is sikerült, a kiszolgálónak egy 211-es állapotkóddal kezdődő szöveget kell válaszul küldenie. Ezt is ellenőrizzük és sikertelenség esetén kilépünk a programból.

A hírcsoport kiválasztása után küldünk egy `"head"` parancsot. Ennek hatására visszakapjuk a csoport első üzenetének fejlécét. Természetesen kérésünkre ez esetben is először egy nyugta érkezik, mely a 211-es állapotkódot kell, hogy tartalmazza. A fejléc csak ezt követően érkezik, számos hasznos információt tartalmazó sorral. A fejlécet záró sor egyetlen pontot tartalmaz. Ezt a sort egy `while` ciklussal keressük meg. Mindaddig, míg a kiszolgáló válaszsora nem ponttal kezdődik, további sorokat olvasunk be és mindegyiket megjelenítjük a böngészőben.

Végső lépésként bontjuk a kapcsolatot. A 13.3-as ábra a 13.8-as program egy lehetséges eredményét mutatja be.

13.3. ábra

NNTP kapcsolat megteremtése



Levél küldése a mail() függvénnyel

A PHP leveszi a vállunkról az elektronikus levelezés gondját is. A `mail()` függvény három karakterláncot vár paraméterként. Az első a címzett, a második a levél tárgya, a harmadik pedig maga az üzenet. A `mail()` `false` értékkel tér vissza, ha problémába ütközik a levél küldése során. A következő kódrészlet egy rövid levél küldését példázza:

```
$cimzett = "valaki@tartomany.hu";
$targy = "üdvözlet";
$uzenet = "ez csak egy próba üzenet! ";
mail( $cimzett, $targy, $uzenet ) or
 ➥ print "A levél elküldése sikertelen";
```

Ha a PHP-t UNIX rendszeren futtatjuk, a Sendmailt fogja használni, más rendszereken a helyi vagy egy távoli SMTP kiszolgálót fog feladata elvégzéséhez igénybe venni. A kiszolgálót a `php.ini` fájl `SMTP` utasításával kell beállítani.

Nem kell a `mail()` kötelező paraméterei által előállított fejlécekre szorítkoznunk. A függvénynek van ugyanis egy elhagyható negyedik paramétere is, mellyel szabadon alakíthatjuk az elküldendő levél fejléceit. Az ebben felsorolt fejlécsorokat a CRLF (`\r\n`) karakterpárral kell elválasztanunk. Az alábbi példában egy `From (Feladó)` és egy `X-Priority` (Fontosság) mezővel bővítjük a fejlécet. Ez utóbbit csak bizonyos levelezőrendszerek veszik figyelembe. A példa a következő:

```
$cimzett = "valaki@tartomany.hu";
$felado = "masvalaki@masiktartomany.hu";
$targy = "üdvözlet ";
$uzenet = "ez csak egy proba üzenet! ";
mail( $cimzett, $targy, $uzenet,
 ➥ "From: $felado\r\nX-Priority: 1 (Highest)" )
or print "A levél elküldése sikertelen";
```

Összefoglalás

Ezen az órán megtudhattuk, hogyan vethetjük be a környezeti változókat, ha több adatot szeretnénk megtudni látogatóinkról. Ha nem tudjuk a látogató gépének nevét, a `gethostbyaddr()` függvénnyel kideríthetjük.

Láthattuk miként zajlik egy HTTP kapcsolat megteremtése a kiszolgáló és az ügyfél között, megtanultuk, hogyan töltsünk le egy dokumentumot a webről az `fopen()` segítségével és hogyan építsük ki saját HTTP kapcsolatunkat az `fsockopen()` függvény felhasználásával. Az `fsockopen()`-nel más hálózati szolgáltatásokhoz is kapcsolódtunk, végül pedig elektronikus levelet küldtünk a programból a `mail()` függvény egyszerű meghívásával.

Kérdések és válaszok

A HTTP meglehetősen misztikusnak tűnik. Tényleg szükség van az ismeretére, ha jó PHP kódot szeretnénk írni?
Nem. Kitűnő programok készíthetők a kiszolgáló-ügyfél párbeszéd részletes ismerete nélkül is. Másfelől azonban szükség van ilyen alapvető ismeretekre, ha kicsit bonyolultabb feladatokat szeretnénk programozottan végrehajtani, például weboldalakat letölteni.

Ha én is könnyedén küldhetek hamis fejléceket, akkor mennyire bízhatok a mások fejlécei alapján létrehozott környezeti változókban?
Az olyan környezeti változókban, mint a `$HTTP_REFERER` vagy a `$HTTP_USER_AGENT`, nem szabad megbíznunk, amennyiben ezen információk pontos ismerete programunkban alapvető követelmény. Az ügyfélprogramok tekintélyes hányada azonban nem hazudik: ha a böngészőtípust és az egyéb felhasználói jellemzőket olyan céllal gyűjtjük, hogy az abból készített statisztikák elemzése alapján a felhasználókat jobban szolgálhassuk, nincs értelme foglalkoznunk azzal, hogy nem minden adat feltétlenül helytálló.

Műhely

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

Kvíz

1. Mely környezeti változó árulja el nekünk annak a lapnak a címét, amely a programunkra hivatkozott?

2. Miért nem felel meg a `$REMOTE_ADDR` változó a látogató nyomon követésére?

3. Minek a rövidítése és mit jelent a HTTP?

4. A fejléc mely sorából tudhatja meg a kiszolgáló, hogy milyen ügyfélprogramtól érkezett a kérés?

5. Mit takar a kiszolgáló 404-es válaszkódja?

6. Anélkül, hogy külön kapcsolatot építenénk fel egy kiszolgálóval, mely függvénnyel tölthetünk le egy weboldalt róla?

7. Adott IP cím alapján hogyan deríthető ki a hozzá tartozó gép neve?

8. Melyik függvény használható hálózati kapcsolat létrehozására?

9. A PHP mely függvényével küldenénk elektronikus levelet?

Feladatok

1. Készítsünk egy programot, amely egy webcímet kér be (például `http://www.kiskapu.hu/`) a felhasználótól, majd egy `HEAD` kérést küldve felveszi a kapcsolatot azzal. Jelenítsük meg a kapott választ a böngészőben. Ne feledkezzünk meg arról, hogy a kapcsolat nem mindig jön létre.

2. Készítsünk olyan programot, amely a felhasználónak lehetővé teszi, hogy begépeljen némi szöveget, majd elektronikus levél formájában továbbítsa azt a mi e-mail címünkre. Áruljuk el a felhasználónak, hogy a környezeti változók mit is állítanak róla.