20. ÓRA

Állapotok tárolása munkamenet-függvényekkel

Az előző órában áttekintettük, hogy hogyan lehet süti vagy lekérdező karaktersorozat használatával menteni az oldalak állapotát. A PHP 4 ezeken kívül további lehetőségeket is nyújt, különböző függvényei vannak a felhasználó állapotának mentésére is. Ezen függvények használata nagyon hasonló az előző órában tanultakhoz, de mivel közvetlenül a nyelvbe vannak építve, az állapot mentése egyszerű függvényhívásokkal végezhető el.

Ebben az órában a következőkről tanulunk:

- Mik azok a munkamenet-változók és hogyan használhatók?
- Hogyan indítsunk el és folytassunk egy munkamenetet?
- Hogyan tartsuk nyilván a munkamenetek változóit?
- Hogyan semmisítsünk meg egy munkamenetet?
- Hogyan töröljük egy munkamenet egyes változóit?

Mik azok a munkamenet-függvények?

A munkamenet-függvényekkel egyszerűen végezhetjük el azokat a műveleteket, amelyekről az előző órában tanultunk. Ezek egy különleges felhasználói azonosítót biztosítanak, amellyel oldalról oldalra különböző információkat vihetünk át, hozzákapcsolva azokat ehhez az azonosítóhoz. Az előzőekhez képest igen nagy előny, hogy a függvények használatával kevesebb munka hárul ránk. Amikor a felhasználó lekér valamilyen munkamenetet támogató oldalt, kap egy azonosítót, vagy a régebbi látogatásakor szerzett meglévő azonosítója kerül felhasználásra. Ezt követően a munkamenethez kapcsolt összes globális változó elérhető lesz a kódban.

A munkamenet-függvények mindkét, az előző órában tanult módszert támogatják. Alapértelmezés szerint a sütiket használják, ugyanakkor az azonosítókat minden, a saját oldalunkra mutató kereszthivatkozásban elhelyezhetjük, így biztosítva, hogy az oldal a sütiket nem támogató böngészővel is használható legyen.

A munkamenet állapotát a rendszer általában egy ideiglenes fájlban tárolja, de a megfelelő modulok használatával az adatokat adatbázisba is tehetjük.

Munkamenet indítása a session_start() függvénnyel

Hacsak nem változtattuk meg a `php.ini` fájlt, a munkameneteket mindig saját magunknak kell elindítanunk, alapállapotban azok nem indulnak el automatikusan. A `php.ini` fájlban található a következő sor:

```
session.auto_start = 0
```

Változtassuk a `session.auto_start` értékét `1`-re, így a PHP dokumentumokban a munkamenetek rögtön elindulnak. Ha nem változtatjuk meg az értéket, mindig meg kell hívnunk a `session_start()` függvényt.

Miután a munkamenet elindult, a `session_id()` függvénnyel rögtön hozzáférhetünk a felhasználó munkamenet-azonosítójához. A `session_id()` függvénnyel lekérdezhetjük vagy módosíthatjuk az azonosítót. A 20.1. példában található program a munkamenet-azonosítót írja ki a böngészőbe.

20.1. program Munkamenet indítása vagy folytatása

```
 1: <?php
 2: session_start();
 3: ?>
 4: <html>
 5: <head>
 6: <title>20.1. program Munkamenet indítása vagy
           folytatása</title>
 7: </head>
 8: <body>
 9: <?php
10: print "<p>Üdvözöljük! Az Ön munkamenet-azonosítója "
          .session_id()."</p>\n\n";
11: ?>
12: </body>
13: </html>
```


A fenti program első lefutása után létrehoz egy munkamenet-azonosítót. Ha a felhasználó később újratölti az oldalt, ugyanazt az azonosítót kapja meg. Ehhez természetesen szükséges, hogy a felhasználó engedélyezze a sütiket a böngészőjében. Vizsgáljuk meg a 20.1. példa kimenetének fejléceit. A süti a következőkben lett beállítva:

```
HTTP/1.1 200 OK
Date: Sun, 07 Jan 2001 13:50:36 GMT
Server: Apache/1.3.9 (Unix) PHP/4.0.3
Set-cookie: PHPSESSID=2638864e9216fee10fcb8a61db382909; path=/
Connection: close
Content-Type: text/html
```

Mivel a `session_start()` megpróbálja beállítani a sütit, amikor munkamenetet hoz létre, el kell indítani, mielőtt bármit kiíratnánk a böngészővel. Vegyük észre, hogy nincs lejárati idő. Ez azt jelenti, hogy a munkamenet csak addig tarthat, amíg a böngésző aktív. Amikor a felhasználó leállítja a böngészőt, a sütit nem menti, de ez a viselkedés a `php.ini` fájl `session.cookie_lifetime` beállításának megváltoztatásával módosítható. Az alapértelmezett beállítás `0`, itt kell a lejárati időt másodpercben megadni. Ezzel mindegyik munkamenetnek megadjuk a működési határidőt.

Munkamenet-változók

A munkamenet-azonosító hozzárendelése a felhasználóhoz csak az első lépés. Egy munkamenet során tetszőleges számú globális változót vezethetünk be, amelyeket később bármelyik, a munkameneteket támogató oldalunkról elérhetünk.

A változó bejegyzéséhez, bevezetéséhez a `session_register()` függvényt kell használnunk. A `session_register()` függvény paramétereiben karaktersorozatokat kell megadni, így határozva meg egy vagy több bevezetendő változó nevét. A függvény visszatérési értéke `true`, ha a bejegyzés sikeres volt. A függvény paramétereiben csak a változók nevét kell megadni, nem pedig a változók értékeit.

A 20.2. példában két változó bejegyzését láthatjuk.

20.2. program Változók bejegyzése

```
 1: <?php
 2: session_start();
 3: ?>
 4: <html>
 5: <head>
 6: <title>20.2. program Változók bejegyzése</title>
 7: </head>
 8: <body>
 9: <?php
10: session_register( "termek1" );
11: session_register( "termek2" );
12: $termek1 = "Ultrahangos csavarhúzó";
13: $termek2 = "HAL 2000";
14: print session_encode();
15: print "A változókat bejegyeztük";
16: ?>
17: </body>
18: </html>
```

A 20.2. példában látható kódnak nincs túl sok értelme, amíg a felhasználó be nem tölt egy új oldalt. A 20.3. példában egy olyan PHP program látható, amely a 20.2. példában bejegyzett változókat használja.

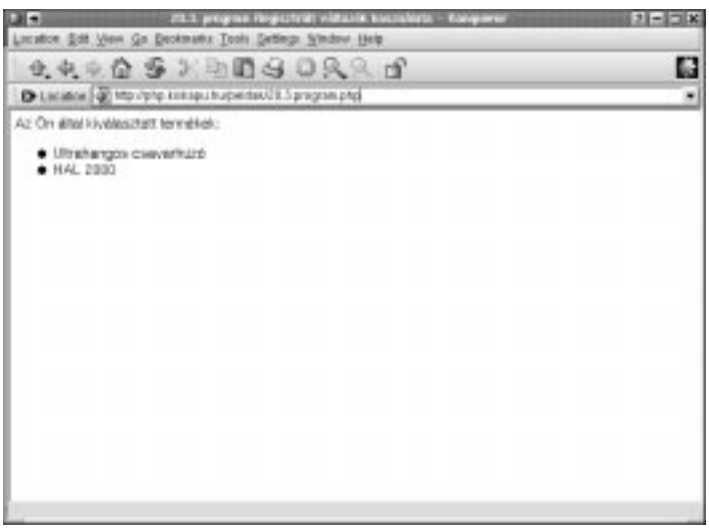
20.3. program Bejegyzett változók használata

```
 1: <?php
 2: session_start();
 3: ?>
 4: <html>
 5: <head>
 6: <title>20.3. program Bejegyzett változók
           használata</title>
 7: </head>
 8: <body>
 9: <?php
10: print "Az Ön által kiválasztott termékek:\n\n";
11: print "<ul><li>$termek1\n<li>$termek2\n</ul>\n";
12: ?>
13: </body>
14: </html>
```


A 20.3. program eredménye a 20.1. ábrán látható. Látjuk, hogy az oldalról hozzáférünk a másik programban bejegyzett `$termek1` és `$termek2` változókhoz.

20.1. ábra

Bejegyzett változók használata



Hogyan működik mindez? A PHP 4 a háttérben folyamatosan fenntart egy ideiglenes fájlt a munkamenet számára. A `session_save_path()` függvény használatával e fájl elérési útját deríthetjük ki, de a függvénynek egy könyvtár elérési útját is megadhatjuk. Ha ezt meghatározzuk, az ideiglenes munkamenet-fájlok ezután

ide kerülnek. Ha a függvényt paraméter nélkül futtatjuk, visszatérési értéke annak a könyvtárnak az elérési útja lesz, ahová a rendszer az ideiglenes munkamenet-fájlokat menti. A szerző rendszerén a

```
print session_save_path();
```

visszatérési értéke `/tmp`. A `/tmp` könyvtár például a következő fájlokat tartalmazhatja:

```
sess_2638864e9216fee10fcb8a61db382909
sess_76cae8ac1231b11afa2c69935c11dd95
sess_bb50771a769c605ab77424d59c784ea0
```

Amikor megnyitjuk azt a fájlt, melynek neve a 20.1. példában visszaadott munkamenet-azonosítót tartalmazza, láthatjuk, hogyan tárolódnak a bejegyzett változók:

```
termek1|s:22:"Ultrahangos csavarhúzó";termek2|s:8:"HAL 2000";
```

Amikor meghívjuk a `session_register()` függvényt, a PHP beírja a fájlba a változó nevét és értékét. Később innen kapjuk meg a változó értékét.

Ha a `session_register()` függvénnyel bejegyeztünk egy változót, annak a PHP program futása során módosított értéke a munkamenet-fájlban is meg fog változni.

A 20.2. példa bemutatta a `session_register()` függvény használatát, de a függvény működése nem tűnik túl rugalmasnak. Mit tehetünk, ha a felhasználó többféle terméket tartalmazó változókat szeretne bevezetni? Szerencsére a `session_register()` függvény paraméterében tömb típusú változót is megadhatunk.

A 20.4. példában a felhasználó több termék közül választhat. Ezután munkamenet-változókkal létrehozunk egy egyszerű „bevásárlókosarat”.

20.4. program Tömb típusú változó bejegyzése

```
1: <?php
2: session_start();
3: ?>
4: <html>
5: <head>
6: <title>20.4. program Tömb típusú változó
          bejegyzése</title>
7: </head>
```

20.4. program (folytatás)

```
 8: <body>
 9: <h1>Termékböngésző lap</h1>
10: <?php
11: if ( isset( $termekek_urlap ) )
12:      {
13:      $termekek = $termekek_urlap;
14:      session_register( "termekek" );
15:      print "<p>A termékeit bejegyeztük!</p>;
16:      }
17: ?><p>
18: <form method="POST">
19: <select name="termekek_urlap[]" multiple size=3>
20: <option> Ultrahangos csavarhúzó
21: <option> HAL 2000
22: <option> Kalapács
23: <option> TV
24: <option> Targonca
25: </select>
26: </p><p>
27: <input type="submit" value="Rendben">
28: </form>
29: </p>
30: <a href="20.5.program.php">Tartalom</a>
31: </body>
32: </html>
```


Először elindítjuk a munkamenetet a `session_start()` függvénnyel. Ezzel hozzáférünk minden korábban beállított munkamenet-változóhoz. A HTML dokumentumon belül a `FORM` elem `ACTION` tulajdonságát beállítjuk a jelenlegi dokumentumra. Létrehozunk egy `termekek_urlap[]` nevű `SELECT` elemet, amely a termékek számával megegyező `OPTION` összetevőt tartalmaz. Emlékezzünk rá, hogy a HTML dokumentum azon elemeinek, amelyek engedélyezik több elem kiválasztását, szögletes zárójellel együtt kell megadni a `NAME` paraméterét. Így a felhasználó választása egy tömbbe kerül.

A PHP kód blokkjában megvizsgáljuk, hogy létezik-e a `$termekek_urlap` tömb. Ha a változó létezik, feltehetjük, hogy a kérdőívet kitöltötték. Ezt a változót hozzárendeljük a `$termekek` változóhoz, amit a `session_register()` függvénnyel bejegyzünk. A `$termekek_urlap` tömböt közvetlenül nem jegyezzük be, mert ez egy későbbi kitöltés során ütközést okozhat az ugyanilyen nevű, `POST` eljárással érkezett változóval.

A kód végén megadunk egy hivatkozást, ahol bemutatjuk, hogy valóban hozzáférünk a bejegyzett tömbhöz. Ez a kód a 20.5. példában található.

20.5. program Hozzáférés bejegyzett tömb típusú változóhoz

```
 1: <?php
 2: session_start();
 3: print session_encode(); //kódolt karaktersorozat kiírása
 4: ?>
 5: <html>
 6: <head>
 7: <title>20.5. program Hozzáférés bejegyzett
          tömb típusú változóhoz</title>
 8: </head>
 9: <body>
10: <h1>Tartalom lap</h1>
11: <?php
12: if ( isset( $termekek ) )
13:     {
14:     print "<b>A bevásárlókocsi tartalma:</b><ol>\n";
15:     foreach ( $termekek as $egytermek )
16:         print "<li>$egytermek";
17:     print "</ol>";
18:     }
19: ?>
20: <a href="20.4.program.php">Vissza
a termékválasztáshoz</a>
21: </body>
22: </html>
```

A `session_start()` használatával folytatjuk a munkamenetet. Ellenőrizzük, hogy létezik-e a `$termekek` változó. Ha létezik, elemeit egyenként kiírjuk a böngészőbe.

Természetesen egy igazi „bevásárlókosár-program” megírásakor a termékeket érdemes adatbázisban, és nem a futtatandó kódban tárolni. A 20.4. és 20.5. példa csak azt mutatta be, hogyan érhető el egy tömb típusú változó egy másik oldalról.

A munkamenet és a változók bejegyzésének törlése

A munkameneteket és a bejegyzett változókat a `session_destroy()` függvénnyel törölhetjük. A `session_destroy()` függvénynek nincsenek paraméterei, futtatásához azonban szükséges egy folyamatban lévő munkamenet. A következő kódrészlet létrehoz, majd rögtön megsemmisít egy munkamenetet:

```
session_start();
session_destroy();
```

Amikor olyan oldalt töltünk be, amely egy munkamenetet vár, az már nem látja a korábban törölt munkamenetet, ezért létre kell hoznia egy újat. Minden korábban bejegyzett változó elvész.

Valójában a `session_destroy()` függvény nem semmisíti meg a bejegyzett változókat. Abban a kódban, amelyben a `session_destroy()` függvényt futtattuk, a változók továbbra is elérhetők. A következő kódrészletben látható, hogy az `5`-re állított `$proba` változó a `session_destroy()` meghívása után is elérhető marad. A munkamenet megszüntetése nem törli a bejegyzett változókat.


```
session_start();
session_register( "proba" );
$proba = 5;
session_destroy();
print $proba; // kiírja, hogy 5
```

A munkamenetből a bejegyzett változókat a `session_unset()` függvénnyel távolíthatjuk el. A `session_unset()` a munkamenet-fájlból és a programból is eltávolítja a változókat, ezért óvatosan használjuk.

```
session_start();
session_register( “proba” );
$proba = 5;
session_unset();
session_destroy();
print $proba; // nem ír ki semmit. a $proba változó
 ➥ már nem létezik.
```

A munkamenet megszüntetése előtt meghívjuk a `session_unset()` függvényt, amely eltávolítja a `$proba`, és minden más bejegyzett változót a memóriából.

Munkamenet-azonosítók a lekérdező karakterláncban

Áttekintettük a sütin alapuló munkamenetek kezelését. Tudnunk kell azonban, hogy ez nem a legbiztosabb módszer. Nem feltételezhetjük ugyanis, hogy a látogató böngészője támogatja a sütik használatát. A probléma elkerülése végett építsünk be programunkba egy, a munkamenet-azonosító továbbadására alkalmas lekérdező karakterláncot. A PHP létrehozza a `SID` állandót, ha a böngésző nem küldött vissza azonosítót tartalmazó sütit. A `SID` név–érték alakú állandó karakterlánc, amelyet beágyazhatunk a munkamenetet támogató oldalra mutató HTML hivatkozásokba:

```
<a href="masik_lap.php?<? print SID; ?>">Másik lap</a>
A böngészőben ez már így jelenik meg:
<a href="masik_lap.php?PHPSESSID=
 ➥ 08ecedf79fe34561fa82591401a01da1">Másik lap</a>
```

A munkamenet-azonosító automatikusan átvételre kerül, amikor a céloldalon meghívjuk a `session_start()` függvényt, és az összes korábban bejegyzett változó a szokásos módon elérhető lesz.

Ha a PHP 4-et az `--enable-trans-sid` kapcsolóval fordítottuk le a fenti lekérdező karakterlánc automatikusan hozzáadódik minden HTML hivatkozáshoz. Ez a lehetőség alapállapotban kikapcsolt, de használatával kódjaink hordozhatóbbá válnak.

Munkamenet-változók kódolása és visszafejtése

Amikor megnéztük a munkamenet-fájl tartalmát, láttuk, hogyan menti és kódolja a PHP a bejegyzett változókat. A kódolt karaktersorozathoz a `session_encode()` függvény használatával bármikor hozzáférhetünk. Ez hasznos lehet a munkamenetek nyomkövetésénél. A következő kódrészletben láthatjuk, hogyan működik a `session_encode()` függvény.

```
session_start();
print session_encode()."<br>";
// minta eredmény: termekek|a:2:{i:0;s:8:
 ➥ "HAL 2000";i:1;s:8:"Targonca";}
```

Látható, hogy a függvény tárolási alakjukban adja vissza a bejegyzett változókat, így ellenőrizhetjük, hogy a változó valóban be lett-e jegyezve és értéke megfelelő-e. A `session_encode()` függvény használatával bejegyzett változókat tartalmazó adatbázisokat is készíthetünk.

A session_decode() függvény használatával a változók kódolt formájából visszanyerhetjük az értéküket. A következő kódrészlet ezt mutatja be:

```
session_start();
session_unset(); // nem szabad, hogy létezzenek
 ➥ munkamenet-változók
session_decode( "termekek|a:2:{i:0;s:8:\"HAL 2000\";
 ➥ i:1;s:8:\"Targonca\";}" );
foreach ( $termekek as $egytermek )
     {
     print "$egytermek<br>\n";
     }
// Kimenet:
// HAL 2000
// Targonca
```

A szokásos módon elindítjuk a munkamenetet. Hogy tiszta lappal induljunk, a session_unset() függvénnyel töröljük az összes korábban bejegyzett változót. Ezután átadjuk a session_decode() függvénynek a változók kódolt formáját. A függvény párosítja a változóneveket a megfelelő értékekkel. Ezt ellenőrizzük is: a $termekek tömb minden elemét kiíratjuk.

Munkamenet-változó bejegyzésének ellenőrzése

Mint láttuk, a bejegyzett változó jelenlétét az isset() függvénnyel ellenőrizhetjük, de más módon is megvizsgálhatjuk a változó létezését. Erre való a session_is_registered() függvény. A függvény paraméterében egy változó nevét tartalmazó karakterláncot kell megadni, a visszatérési érték pedig true, ha a változó bejegyzett.

```
if ( session_is_registered ( "termekek" ) )
     print "A 'termekek' változó bejegyzett!";
```

Ez akkor lehet hasznos, ha biztosak akarunk lenni a változó forrásában, azaz, hogy a változó tényleg a munkamenet bejegyzett változója vagy csak egy GET kérelem eredményeként kaptuk.

Összefoglalás

Ebben és az előző órában áttekintettük, hogyan menthetjük az állapotokat egy állapot nélküli protokollban. Különféle módszereket alkalmaztunk: sütiket és lekérdező karakterláncokat használtunk, néha ezeket fájlokkal és adatbázisokkal ötvöztük. Mindegyik megközelítési módnak megvannak a maga előnyei és hátrányai.

A süti nem igazán megbízható és nem tárolhat sok információt. Ezzel szemben hosszú ideig fennmaradhat.

Az adatok fájlban vagy adatbázisban való tárolása miatt a sebesség csökkenhet, ami gondot okoz egy népszerű webhelyen. Ugyanakkor egy egyszerű azonosítóval nagy mennyiségű adathoz férhetünk hozzá.

A lekérdező karakterlánc a süti ellentéte. Elég csúnyán néz ki, de viszonylag nagy mennyiségű információt hordoz magában. Használatáról a körülmények mérlegelése után kell dönteni.

Ebben az órában megtanultuk, hogyan indítsunk el munkamenetet a `session_start()` függvénnyel. Láttuk, hogyan jegyezhetünk be változókat a `session_register()` függvénnyel, hogyan ellenőrizhetjük a bejegyzést a `session_is_registered()` függvénnyel, valamint hogyan semmisíthetjük meg a változókat a `session_unset()` függvénnyel. A munkamenet megsemmisítésére a `session_destroy()` függvényt használtuk.

Hogy minél több felhasználó érhesse el a munkamenetet támogató környezetet, lekérdező karakterláncainkban használjuk a `SID` állandót a munkamenet-azonosító átadására.

Kérdések és válaszok

Van valamilyen veszélye a munkamenet-függvények használatának, amiről még nem tudunk?

A munkamenet-függvények alapvetően megbízhatók. De emlékezzünk rá, hogy a sütik használata nem lehetséges több tartományon keresztül. Ezért, ha több tartománynevet használunk ugyanazon a kiszolgálón (elektronikus kereskedelemi alkalmazásoknál ez gyakori), tiltsuk le a sütik használatát a `session.use_cookies` `0`-ra állításával a `php.ini` fájlban.

Műhely

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

Kvíz

1. Melyik függvényt használjuk egy munkamenet elindítására?

2. Melyik függvény adja vissza a jelenlegi munkamenet azonosítóját?

3. Hogyan rendelhetünk változókat a munkamenethez?

4. Hogyan semmisíthetjük meg a munkamenetet és törölhetjük el minden nyomát?

5. Hogyan semmisíthetjük meg az összes munkamenet-változót a futó programban és az egész munkamenetben?

6. Mi a `SID` állandó értéke?

7. Hogyan ellenőrizhetjük, hogy a `$proba` nevű változó be van-e jegyezve egy munkameneten belül?

Feladatok

1. Az előző óra Feladatok részében készítettünk egy programot, amely sütit vagy lekérdező karakterláncot használ, hogy oldalról oldalra vigye tovább a felhasználó válaszait. Ezután a környezet minden oldala a beállított háttérszínnel indult és név szerint üdvözölte a felhasználót. Készítsük el ezt a programot a PHP 4 munkamenet-függvényeivel.

2. Készítsünk programot, amely munkamenet-függvények használatával emlékezik rá, hogy a felhasználó a környezet mely oldalait látogatta meg. Készítsük el a felhasználó lépéseit tartalmazó hivatkozások sorozatát, hogy mozgása nyomon követhető legyen.