Dr. Pál László, Sapientia EMTE, Csíkszereda

WEB PROGRAMOZÁS
5.ELŐADÁS

2023-2024
ősz

Sütik és munkamenetek kezelése

A HTTP protokoll – Emlékeztető

- A HTTP protokoll állapotmentes
- Nem emlékezik az előző kérés adataira
- Függetlenül kezeli a kéréseket
- Gyakran szükséges a kliens azonosítása és a hozzá kapcsolódó adatok tárolása
- Megoldás
  - Munkamenetek (Session)

Munkamenet példák

- Bejelentkezés alapú alkalmazások:
  - Levelezés
  - Internet bank
  - Facebook
- Bejelentkezés nélküli alkalmazások
  - Webshop – kosár

Állapotmentesség feloldása

- Az alkalmazásnak kell gondoskodnia az állapot megtartásáról
- Állapot megtartása: az adat kliensenként történő megőrzése
  - Pld. webshop kosara – kliensenként eltérő kosarak
- Megvalósítás
  - kliens oldalon
  - szerver oldalon

Kliens oldali állapottartás

- Az adatot a kliens gépén tároljuk
- Minden kérésnél felküldjük az adatokat a szerverre
- A szerver visszaadja a kliensnek
- Kliens oldali technológiák
  - URL
  - Rejtett mező
  - Süti

Állapottartás URL-ben

- URL querystring részében adjuk tovább az adatot
  - Pld. szamol.php?szamlalo=1
- Hátrányok:
  - Minden linkhez oda kell generálni
  - Sok adat nem fér el benne
    - URL hossza legfeljebb 2kB
  - Feltűnő (zavaró)
  - Könnyen átírható

Állapottartás URL-ben

- Példa:

```php
<?php
$szamlalo = 0;
if (isset($_GET['szamlalo'])) {
  $szamlalo = $_GET['szamlalo'];
}
$szamlalo += 1;

var_dump($szamlalo);
?>
<a href="szamlalo2.php?szamlalo=<?php echo $szamlalo; ?>">Növel</a>
```

Állapottartás rejtett mezőben

- Rejtett mező:

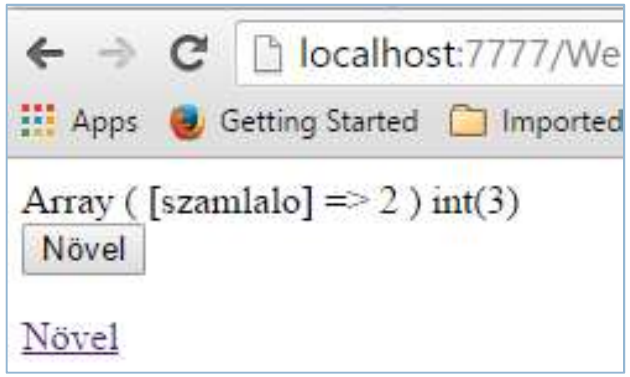
```
<input type="hidden" name="szamlalo" value="3">
```

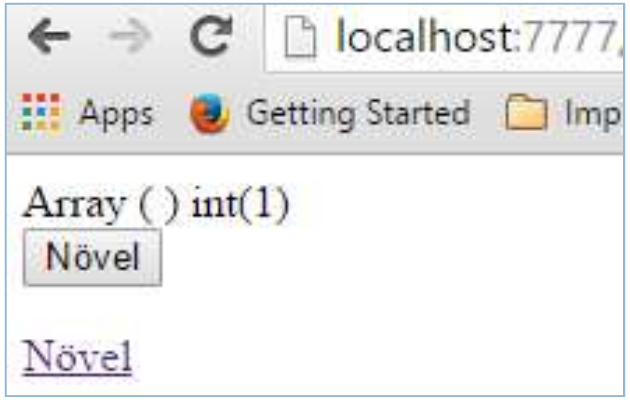
- Előnyök:
  - Sok adat
  - Nem feltűnő
- Hátrányok:
  - Manipulálható
  - Csak űrlapok esetén használható

Állapottartás rejtett mezőben

- Példa:

```php
<?php
print_r($_POST);
$szamlalo = 0;
if (isset($_POST['szamlalo'])) {
  $szamlalo = $_POST['szamlalo'];
}
$szamlalo += 1;

var_dump($szamlalo);
?>
<form action="szamlalo3.php" method="post">
  <input type="hidden" name="szamlalo"
         value="<?php echo $szamlalo; ?>">
  <input type="submit" value="Növel">
</form>

<!-- Ez nem működik -->
<a href="szamlalo3.php">Növel</a>
```





10

Állapottartás sütivel

Mi a süti (cookie)?

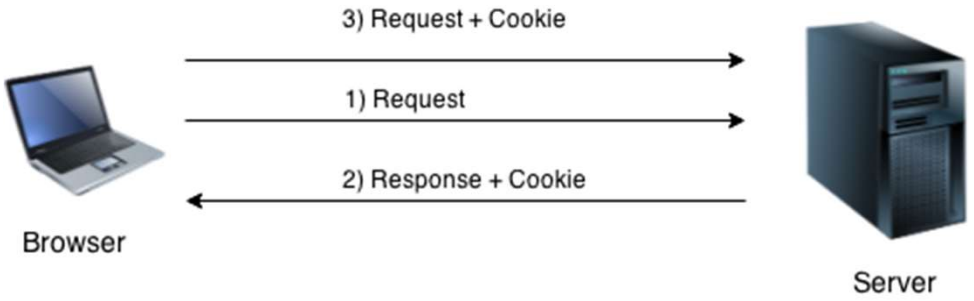
- Olyan kisebb adatmennyiségek, melyek egy-egy változó nevének és értékének tárolására szolgálnak, valamint arról a webhelyről is feljegyzik az információkat, melyről ezek az adatok származnak
- Kliensoldali adattárolást tesznek lehetővé
- A webhelyek általában csak a saját sütijeiket módosíthatják
- Az a webszerver érheti el és írhatja át, mely eredetileg kiküldte őket a kliensgépre

Mire használjuk a sütiket?

- Kisebb lélegzetű, hasznos, ám nem kritikus fontosságú feladatok esetében érdemes használni
  - Felhasználói beállítások tárolása a webhelyek megjelenítését befolyásoló, testre szabható paraméterek (színek, betűtípusok) kezelésére
  - Felhasználói azonosító kulcsok tárolása, melyekkel a felhasználókról tárolt személyes adatokat érhetjük el

Hogyan használjuk a sütiket?

- A kliens egy sütiket alkalmazó webhelyre látogat
- A webszerver arra utasítja a klienst, hogy későbbi használatra helyben tároljon el egy meghatározott adathalmazt
- A soron következő kérések alkalmával a kliens visszaküldi a megőrzött adatok másolatát
- A visszaküldött adatok alapján a szerver elvégzi a szükséges beállításokat



Hogyan használjuk a sütiket?

- A kliens oldalon az adatok tárolása egy előre meghatározott időtartam lejártáig tárolódnak, amit a szerver állít be másodpercben
- A sütik létrehozásuk után már a kliens fennhatósága alá tartoznak, tehát törölhetik ezeket akár a böngészőprogram segítségével is

Sütik létrehozása, beállítása

- A sütik kezelésére a PHP több módszert is felkínál. A módszerek között van olyan is ami direkt a header() paranccsal hajtódik végre, meg olyan is a mi függvényeket használ (setcookie())
- Megjegyzés: A HTTP protokoll szerint a szerver először ún. fejlécet küld, ennek a fejlécnek lesz része a süti is. A HTML oldal a fejléc után kerül küldésre

Sütik létrehozása, beállítása

- A ***setcookie*** függvény: létrehoz egy sütit, ami a többi header információval együtt kerül az olvasó böngészőjéhez
- Szintaxis:

  setcookie(name, value, expire, path, domain, …);

  - **name:** a süti neve
  - **value:** a névhez rendelt érték
  - **expire:** az az idő intervallum, amíg a süti használható
  - **path:** az elérési út a szerveren
  - **domain:** a domain ahol érvényes a süti
  - **secure**: logikai, biztonságos küldés (HTTPS)
  - **httponly**: logikai, csak HTTP az elfogadott (no JavaScript)

Sütik létrehozása, beállítása

- ***Példa:*** Az alábbi példában létrehozunk egy *user* nevű sütit és az érvényességi időt egy órában határozzuk meg.

```php
<?php
    setcookie("user", "Kiss Istvan", time()+3600);
?>
```

Sütik kiolvasása

- A $_COOKIE[‘valtozo’] globális tömb segítségével történik a kiolvasás
- Példa:

```php
<?php
setcookie("user", "Kiss Manzo", time()+3600);
?>
<html>
    <body>
    <?php
        if (isset($_COOKIE["user"]))
            echo "Welcome " . $_COOKIE["user"] . "!<br />";
        else
            echo "Welcome guest!<br />";
    ?>
    </body>
</html>
```

Sütik kiolvasása

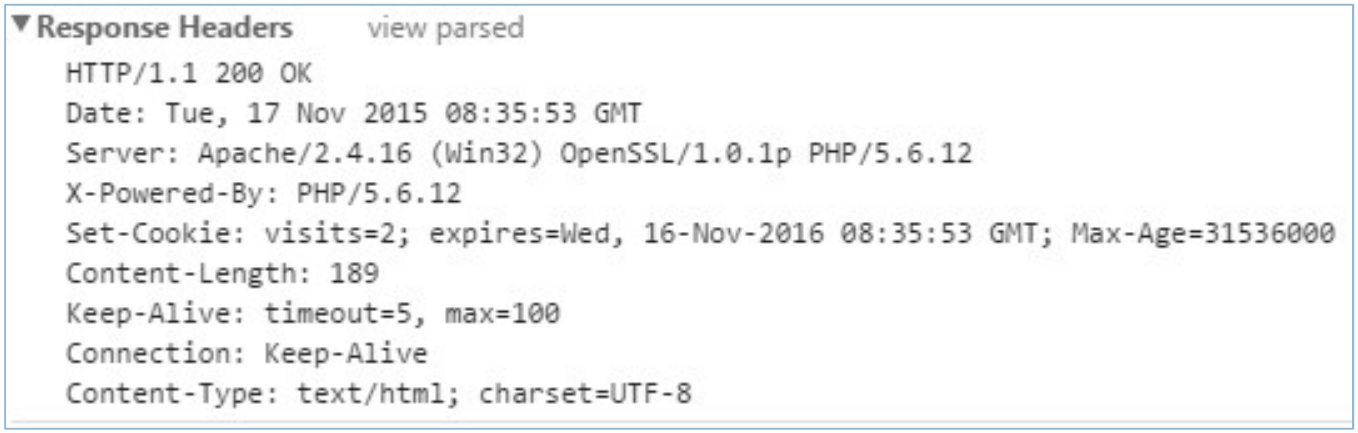
Példa: számláló sütivel

```
<?php
    if (!isset($_COOKIE['visits'])) $_COOKIE['visits'] = 0;
    $visits = $_COOKIE['visits'] + 1;
    setcookie('visits',$visits,time()+3600*24*365);
?>
<html>
<body>
<?php
    if ($visits > 1) {
        echo("Ez a $visits. látogatásod");
    } else {
    // Elso latogatas
    echo'Üdvözöllek a honlapomon!';
    }
?>
</body>
</html>
```

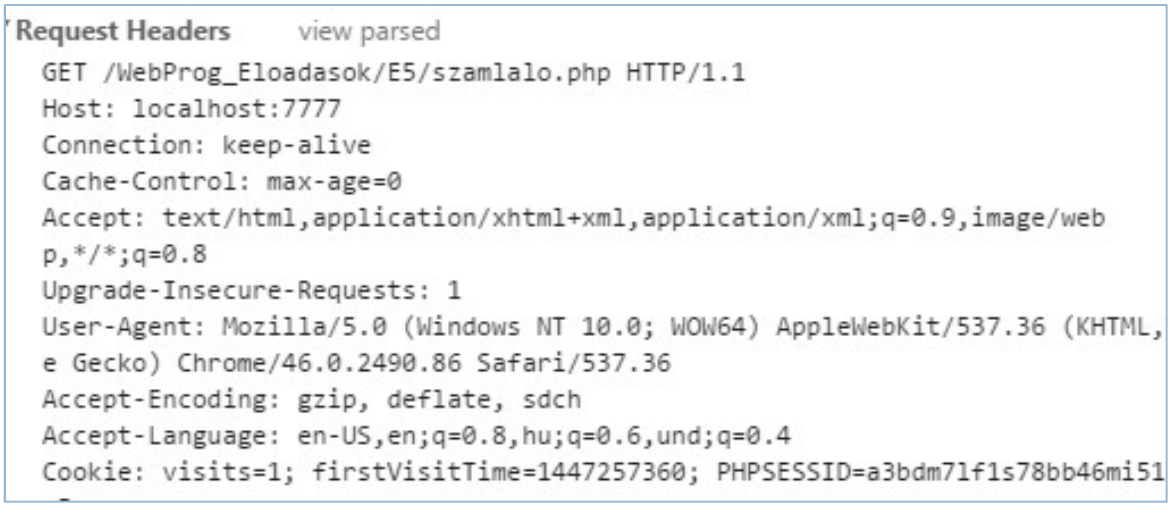
Sütik kiolvasása

- Előző példa HTTP kérése és válasza:

Kérés



Válasz



Süti törlése

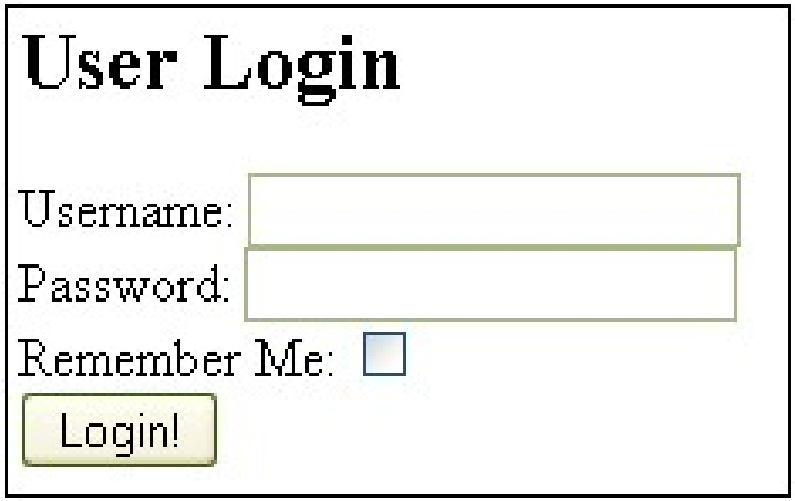
- A süti törlése gyakorlatilag azt jelenti, hogy a sütit lejártnak állítjuk be (az aktuális időből kivonunk egy számot)

```
<?php
    setcookie("user", "", time()-3600);
?>
```

Példa: login ablak sütivel

login.html

```
<form name="login" method="post" action="login.php">
  Username: <input type="text" name="username"><br>
  Password: <input type="password" name="password"><br>
  Remember Me: <input type="checkbox" name="rememberme" value="1"><br>
 <input type="submit" name="submit" value="Login!">
</form>
```



Példa: login ablak sütivel

login.php: süti létrehozása vagy törlése

```php
$user = 'alma';
$pass = 'alma';
if (isset($_POST['username']) && isset($_POST['password'])){
    if (($_POST['username'] == $user) && ($_POST['password'] == $pass)) {

        if (isset($_POST['rememberme'])) {
            setcookie('username', $_POST['username'], time()+60*60*24*365);
            setcookie('password', md5($_POST['password']), time()+60*60*24*365);
        }
        else {
            setcookie('username', $_POST['username'], time()- 3600);
            setcookie('password', md5($_POST['password']), time()-3600);
        }
        header('Location: index.php');

    } else {
        echo 'Username/Password Invalid';
    }

} else {
    echo 'You must supply a username and password.';
}
```

Példa: login ablak sütivel

index.php: ha létezik a süti, beléptet bejelentkezés nélkül

```
$user = 'alma';
$pass = 'alma';

if (isset($_COOKIE['username']) && isset($_COOKIE['password'])){

    if (($_COOKIE['username'] != $user) || ($_COOKIE['password'] != md5($pass)))
        header('Location: login.html');
    else
        echo 'Welcome back ' . $_COOKIE['username'];
}
else
    header('Location: login.html');
```

Átirányítás: header függvény

- Header függvény:
  - Kulcsfontosságú szerepet játszik az HTTP fejlécek manipulálásában és irányításában
  - Egyik gyakori használat a böngésző átirányítása egy másik URL-re (például jogosultságok hiányában)
  - Pld: header("Location: login.php");
- Megjegyzés:
  - Meghívása előtt nem szabad semmilyen kimenetet küldeni a PHP szkriptből (pl. echo vagy HTML tartalom)

Kliensoldali megoldások

- Adat a kliensen van
- Manipulálható
- Sok adat esetén feleslegesen sok adat megy oda-vissza a kliens és szerver között

Szerveroldali megoldások

- Tároljuk az adatot a szerveren
  - nem manipulálható kliens oldalon
  - nem kell sok adatot küldözgetni
- A kliens megkülönböztetése továbbra is szükséges
  - ID-t kap, amivel azonosítja magát és hozzáfér a ID-hoz tartozó adatokhoz
- Az ID kliensoldali megoldással közlekedik
  - süti (alapértelmezett)
  - url (ha nincs süti)

28

Munkamenet (session) kezelés

Munkamenet - Bevezető

- Ismert tény, hogy a HTTP protokoll állapotmentes, azaz a kliens nem követhető két kérés között
- A valós életben előfordulnak olyan helyzetek, amikor szükséges a kliens nyomon követése különböző kérések között
- Session: a kliens és a webszerver között egyedileg végbemenő műveletek sorozatai, amelyek hosszabb időn keresztül is fennállnak

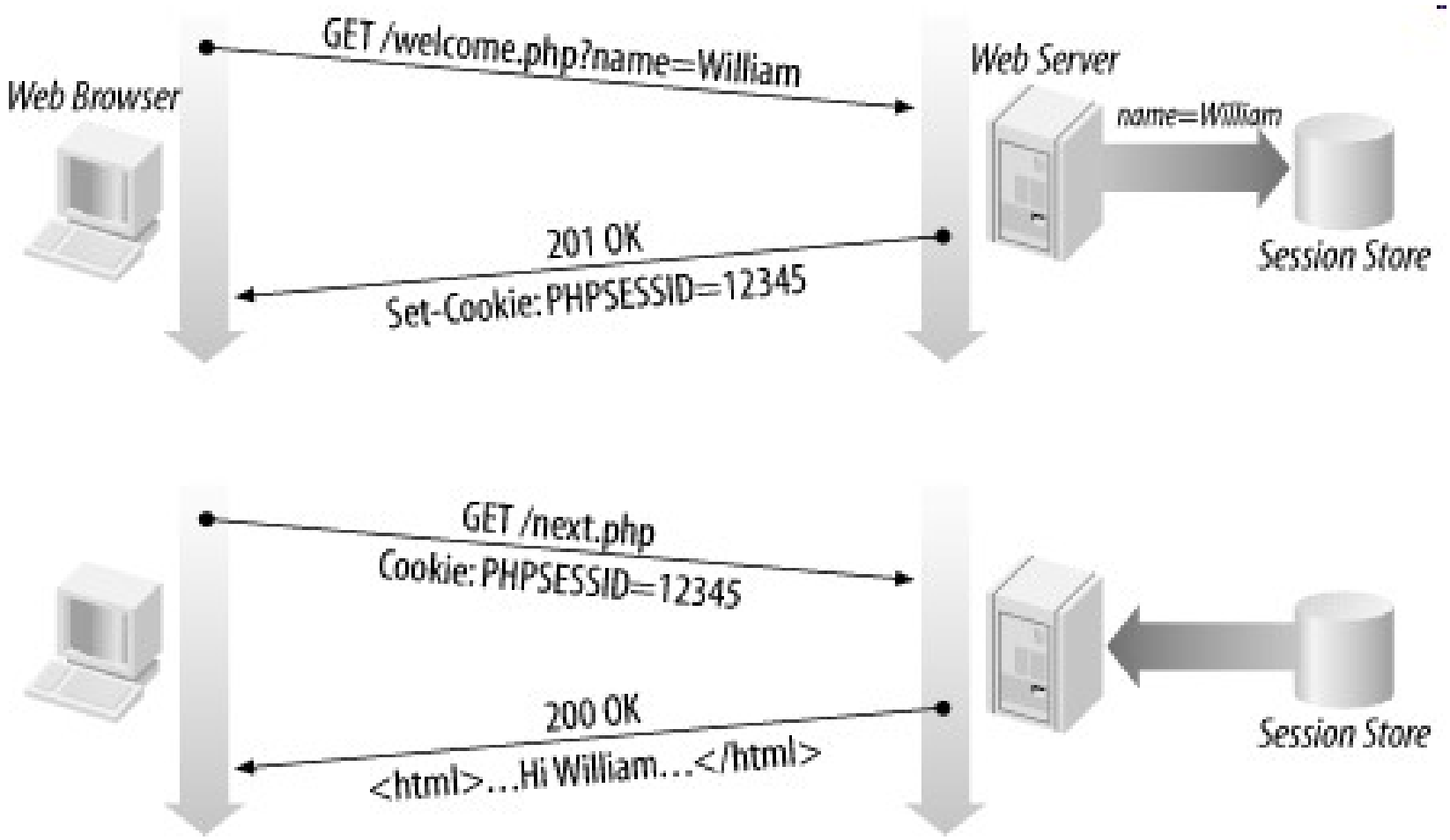
Munkamenet - Bevezető

- Munkafolyamat lehet például valamilyen internetes tranzakció kezelés vagy elektronikus postafiók ellenőrzés, stb.
- A sütiktől eltérően, a munkafolyamat-ban az adatokat a szerveren tartjuk, a kliensnek biztosítunk egy egyedi kulcsot (***Session ID***), amely egyértelműen azonosítja azt és a hozzátartozó szerveroldali adatokat is.

Hogyan működik?

- Amikor a látogató először érkezik oldalunkhoz, generálunk egy azonosítót, amit eljuttatunk a klienshez oly módon, hogy azt a kliens a látogató következő oldal lekérésekor eljutassa a szerverre.
- Ezután egy ilyen azonosító (sessionId) egy adott felhasználó munkamenetét fogja jelképezni, és ezen azonosító alapján a szerveren állapotinformációkat, adatokat tárolhatunk
- Kliens oldalon ez az egyedi azonosító (session id) vagy sütiben vagy a böngésző címsorában(ez a ritkább) tárolódik

Hogyan működik?



Munkamenet kezelése PHP-ben

- Számos függvény létezik ezek kezelésére
- Munkamenet indítása: mielőtt információkat tárolnánk, el kell indítani a munkamenetet (*session_start()*)

```
<?php session_start(); ?>
<html>
    <body>
    </body>
</html>
```

- Az előbbi kód regisztrálja a session-t a szerveren, engedélyezi a felhasználói információk mentését, és a session azonosítót továbbítja a felhasználó számára

Munkamenet kezelése PHP-ben

- Adatok elhelyezése a munkamenetben: a *session_start()* parancsot követően létrejön a $_SESSION nevű tömb, ami egyrészt tartalmazza a munkamenet során már korábban elhelyezett adatokat, valamint újabbakat tehetünk bele

Munkamenet kezelése PHP-ben

- Példa: egyszerű kattintás számláló

```php
<?php
session_start();

if (isset($_SESSION['counter'])){
  print $_SESSION['counter'];
  $_SESSION['counter']++;
  print "<a href=\"counter.php\">növel</a>";
}
else{
  $_SESSION['counter'] = 1;
  print $_SESSION['counter'];
  print "<a href=\"counter.php\"> növel</a>";
}

print "<br> A szesszióazonosító:".session_id();

?>
```

Kimenet

22növel
A szesszióazonosító:evtrcgitp8vl1fh8oq2hb8lmr5

Munkamenet kezelése PHP-ben

- Példa: egyszerű kattintás számláló

Request Headers view parsed

```
GET /WebProg_Laborok/L8/sessions/counter.php HTTP/1.1
Host: localhost:7777
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/web
p,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
e Gecko) Chrome/46.0.2490.86 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,hu;q=0.6,und;q=0.4
Cookie: firstVisitTime=1447257360; PHPSESSID=a3bdm71f1s78bb46mi51v8d9n5
```

Response Headers view parsed

```
HTTP/1.1 200 OK
Date: Tue, 17 Nov 2015 09:30:49 GMT
Server: Apache/2.4.16 (Win32) OpenSSL/1.0.1p PHP/5.6.12
X-Powered-By: PHP/5.6.12
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
Pragma: no-cache
Content-Length: 87
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Munkamenet kezelése PHP-ben

- Megjegyzések:
  - A $_SESSION nem tartalmazhat erőforrás típusú adatoka (pld. adatbázsi kapcsolat)
  - Sütiket használnak, így azok kikapcsolása nem javasolt
  - A munkamenet adatok fájlokban tárolódnak egy speciális helyen
  - A *session_start* utasítás előtt nem lehet más utasítás (még üres sor sem)
  - A munkamenet megszűnésével valamennyi adat elvesz
  - A munkamenet alapértelmezetten legfennebb a kliens böngésző bezárásáig tart

Hasznos függvények

- *session_start()*
  - Munkamenet indítása
- *session_name()*
  - *Lekéri vagy beállítja a munkamenet nevét*
- *session_set_cookie_params()*
  - *A munkamenetben használt süti paraméterek beállítása*
- *session_write_close()*
  - *Adatok explicit mentése (unlock files) -> más program is hozzáfér az adatokhoz*
- *session_reset()*
  - Újra inicializálja a munkamenet tömböt az eredeti értékekkel

Hasznos függvények

- Példa:

```php
/* First start the Session */
session_start();

/* Unset all $_SESSION variables */
$_SESSION = array();

/* Clear the Session Cookie */
$cookie_par = session_get_cookie_params();
setcookie(session_name(), '', time() - 86400, $cookie_par['path'],
    $cookie_par['domain'], $cookie_par['secure'], $cookie_par['httponly']);

/* Destroy the session data */
session_destroy();
```

Munkamenet törlése

- A session adatok törlése az *unset()* vagy a *session_destroy()* függvényekkel történik.
  - Az *unset()* függvényt a session változó törlésére használjuk
  - A session teljes törlése a *session_destroy()* függvénnyel lehetséges

```php
<?php
    unset($_SESSION['counter']);
?>
```

Példa: adat továbbvitele másik oldalra


```php
<?php
session_start();
?>
<html>
  <head>
    <title>sessionok gyakorlása</title>
  </head>
  <body>
    <form action="" method="post">
      Név: <input type="text" name="nev" />
      <input type="submit" value="mehet" name="mehet" />
    </form>
  </body>
</html>
<?php
if (isset($_POST['mehet']))
{
  $_SESSION['nev'] = $_POST['nev'];
  print "<a href=\"kiir.php\">Tovább</a>";
}
?>
```

Példa: adat továbbvitele másik oldalra

kiir.php

```
<?php
session_start();
?>
<html>
  <head>
    <title>Session_gyakorlás</title>
  </head>
  <body>
<?php
  print "<h1>Üdvözöllek ".$_SESSION['nev']."</h1>";
?>
  </body>
</html>
```

Hitelesítés munkamenetekkel

Login.php

```php
<?php
session_start();
?>

<html>
<form name="form1" method="post">
    <table>
        <tr>
            <td>Username</td>
            <td><input type="text" name="username"></td>
        </tr>
        <tr>
            <td>Password</td>
            <td><input type="password" name="password"></td>
        </tr>
        <tr>
            <td><input type="submit" value="SignIn" name="submit"></td>
        </tr>
    </table>
</form>
</html>
```


```php
<?php
if (isset($_POST['submit'])) {
    $user = $_POST['username'];
    $pswd = $_POST['password'];
    if ($user == "admin" && $pswd == "admin") {
        $_SESSION['user'] = $user;
        $_SESSION['logged'] = true;
        header( header: 'Location: welcome.php');
    } else {
        echo "Please enter the username or password again!";
    }
}
?>
```

Hitelesítés munkamenetekkel

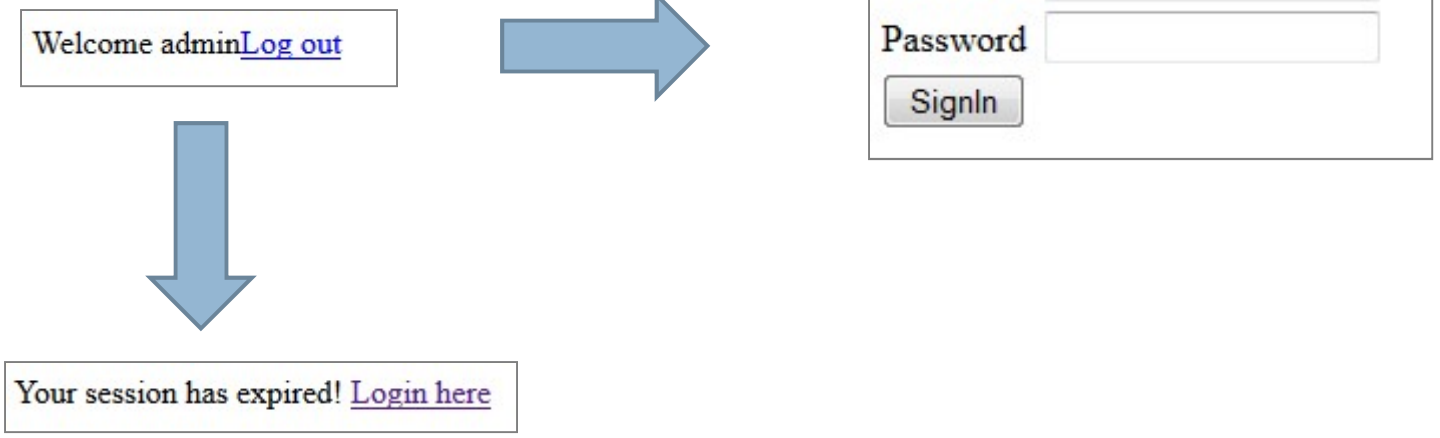
welcome.php

```php
<?php
session_start();

if (isset($_SESSION['logged']) && $_SESSION['logged'] == true) {

    print '<h3>Welcome to the group, ' . $_SESSION['user'] . '!</h3>';
    print '<p><a href="logout.php">Click here to logout.</a></p>';
} else
    print 'Vedett oldal!';
?>
```

Hitelesítés munkamenetekkel

LogOut.php

```php
<?php
session_start();
session_destroy();
print "<p>Your session has expired</p>";
print '<p><a href="login.php">Click here to login.</a></p>';
?>
```



Kérdések

- Milyen állapottartási technikák léteznek kliens oldalon?
- Milyen állapottartási technikák léteznek szerver oldalon?
- Sütik használatának menete?
- Hogyan használjuk a munkameneteket?

Könyvészet

- http://nyelvek.inf.elte.hu
- http://hu.wikipedia.org
- https://www.w3schools.com/php