Сесии и бисквитки в PHP

Виолета Божикова

Какво е сесия?

- Когато работите с едно Desktop приложение, вие го отваряте, правите някакви промени в него и после го затваряте. Това е познато като Session. Компютърът знае кой сте вие. Той знае кога отваряте приложението и кога го затваряте.
- Но когато работите с едно web приложение, основавайки се само на средствата на HTTP протокола (HTTP протокола няма вграден начин за съхраняване на състоянието между две транзакции), web сървърът няма да разбере, че двете последователни заявки към един сайт, са от един и същ потребител, т. о. необходим е метод, с помощта на който web сървърът да проследи информацията за потребителя в течение на даден сеанс на връзка с даден Web-сайт.
- Приложението на сесиите и бисквитките е именно за проследяване на потребителите на web приложенията.

Сесиите и cookies - предназначение

- Сесиите и cookies се използват за съхранение на първоначални сведения за потребителите и последващо валидиране на потребителите, при преход между няколко страници или при последващо посещение на сайта (Интернет-магазин, форум и др., съхраняват информация за потребителя, например username, password...).
- При използване на сесии, данните за сесията (т.е наборът от сесийни променливи) се съхраняват във **временни файлове на сървъра** (последните съществуват само докато браузерът не бъде спрян, освен ако не е зададено нещо друго във файла php.ini).
- Бисквитките са малки **текстови файлове**, **съхранени на компютъра на клиента,** при първото посещение на даден web сайт. При първото посещение на даден web сайт, Server скриптът изпраща набор от "бисквитки" към браузъра, например: за име, възраст, идентификационен номер и др. Браузърът съхранява тази информация на локалната машина за бъдеща употреба. Когато следващия път, браузърът изпраща заявка към web сървъра за същия web сайт, той (браузърът) изпраща и тези "бисквитки", чрез които сървърът идентифицира потребителя.

Стартиране на сесия

- За да се съхрани информация за вашата PHP сесия, програмистът трябва да стартира такава сесия. Сесийните променливи съдържат информация за един единствен потребител и са валидни за всички страници на едно приложение.
- Стартирането на сесия става с функция **session_start**, която се извиква в началото на PHP-сценария:

session_start(); При стартиране на сесия, **PHP интерпретаторът** извършва следното:

1. Получавайки тази команда, сървърът създава нова сесия или възстановява текущата, основавайки се на идентификатора на сесията (SID), предаден към него автоматично, най-често чрез бисквитка PHPSESSID. Ако идентификаторът на текущата сесия вече съществува, то се зареждат регистрираните променливи на сесията.

Стартиране на сесия

- Забележка: Тази функция (**session_start**) трябва да се напише преди <html> tag, тоест:

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

2. Ако потребителят е нов, Php интерпретаторът генерира уникален низ от най-често **32 шестнадесетични символа**, който се използва за **сесиен идентификатор (SID, Session IDentifier),** например **3c7foj34c3jj973hjkop2fc937e3443**. На сървъра, в определена папка (например, \temp) се създава сесиен файл с име **sess_<идентификатор на сесията>,** в който ще се съхраняват сесийните данни (sess_3c7foj34c3jj973hjkop2fc937e3443). SID се генерира в този момент, когато ползователят влиза в сайта, и се унищожава, когато излезе от сайта.
3. Заедно с това, интерпретаторът автоматично изпраща генерирания идентификатор на сесията към потребителския компютър, чрез бисквитка, наречена **PHPSESSID**. **Чрез тази "бисквитка" ще се разпознае клиента, при следващите си обръщения към сървъра**. Ако по някаква причина работата на браузъра с бисквитки е забранена, тогава сесийният идентификатор се изпраща чрез URL-а. Например: адрес **http://.../test.php** се превръща в адрес **http://... /test.php?PHPSESSID=3c7foj34c3jj973hjkop2fc937e3443.**

Забележка: При забранени бисквитки, има и други начини за предаване на сесийния идентификатор, които няма да коментираме.

4. Когато браузърът поиска следваща страница на същия сайт, той ще включи в заявката си към сървъра и идентификатора на сесията, чрез изпращане на бисквитката (чрез хедър Set_Cookie).
5. Така PHP интерпретаторът получава уникалния сесиен идентификатор (от **PHPSESSID** бисквитката), търси сесиен файл с името на този идентификатор в съответната папка на сървъра (**session.save_path в php.ini** определя папката), и ако такъв файл има - то това е гаранция за това, че става въпрос за същия потребител.

Забележка:

- session_start() е нужно да се извиква във всички скриптове, в които предстои да се използват **променливите на сесията**, при това преди извеждане на каквито и да са данни в браузера. Това е свързано с това, че cookies се възстановяват само до първото извеждане на информация на екрана.
- За да извлечем идентификатора на текущата сесия: функция session_id().
- На сесията може да се зададе име чрез функция session_name([имя_сессии]). Извличане на името на текущата сесия може да стане с функция session_name() без параметри.

Регистрация (съхраняване) на сесийни променливи

- От версия PHP 4.2.0, нататък, **сесийните променливи се съхраняват** в глобалния асоциативен масив **$_SESSION[] :**

$_SESSION['username'] = "Maria";

$_SESSION['autoruser']=1;

- добавя се елемент с ключ **'username' и стойност "Maria"** в асоциативния масив $_SESSION
- добавя се елемент с ключ **'autoruser' и стойност 1** в асоциативния масив $_SESSION

Регистрация (съхраняване) на сесийни променливи

- Така, ако трябва да се предаде дадена сесийна променлива, например **“username”** за посетителя между две страници на даден сайт, то php страницата, от която се предава сесийната променлива **“username”** ще изглежда така:

```
<?php
session_start();
$_SESSION[‘username’]=”Maria”;
? ….
```

- Съответно, страницата, която ще използва **username** трябва да има следния фрагмент:

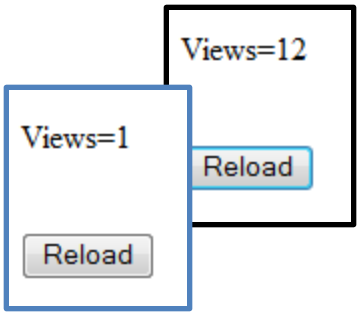
```
<?php
session_start (); // check to see if user has logged
if ($_SESSION[‘username’] == “Maria”) {….
```

Дали са регистрирани сесийни променливи? **isset() function**

- Със следващия пример създаваме елементарен брояч на страници.
- Използвана е **isset() function,** която да провери дали е регистрирана сесийна променлива (в случая - "views “), по следния начин: **if(isset($_SESSION['views'])) ...**
- Ако е регистрирана сесийната променлива views, то увеличаваме стойността и с 1. Ако не - то я създаваме, със стойност 1:

Пример: Стартираме сесия, проверяваме дали е установена сесийна променлива views, ако да – то се увеличава стойността и с единица, ако не – установява се на единица.

```
<?php
session_start();
if(isset($_SESSION['views']))
 {
 $_SESSION['views']=$_SESSION['views']+1;
 echo "Views=". $_SESSION['views'];
 }
 else
 {
 $_SESSION['views']=1;
 echo "Views=". $_SESSION['views'];
 }
 ?>
<form>
<br><input type="submit" value="Reload">
</form>
```



Пример: Стартираме сесия, проверяваме дали е установена сесийна променлива counter, ако да – то се увеличава стойността и с единица, ако не – установява се на единица.

```
<?php
  session_start();
  if( isset( $_SESSION['counter'] ) )
  { $_SESSION['counter'] += 1;   }
  else
  { $_SESSION['counter'] = 1;    }
  $msg = "You have visited this page ".  $_SESSION['counter'];
  $msg .= " in this session.";
  echo ( $msg );
?>
```

Резултат:

You have visited this page 1 in this session.

You have visited this page 2 in this session.

...

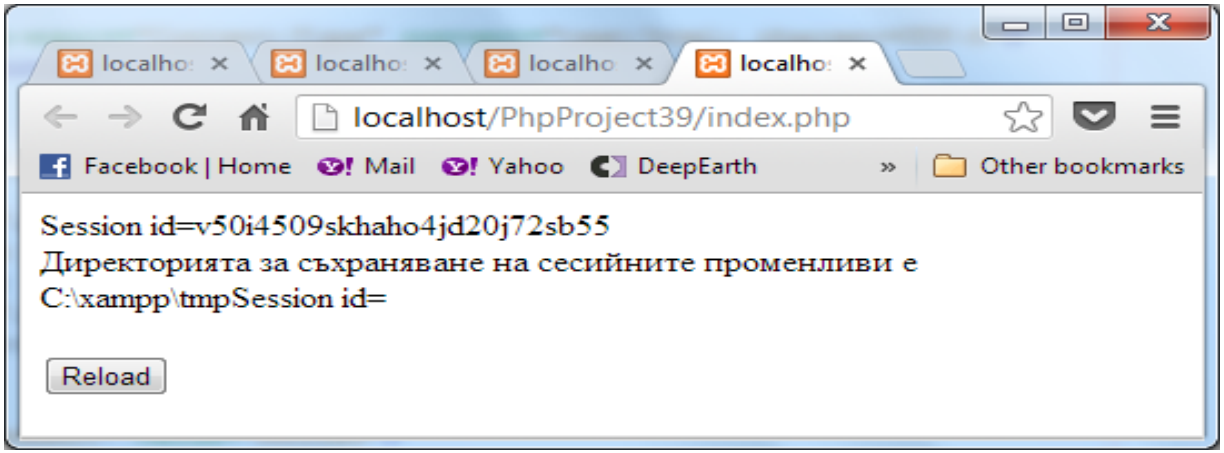
Разрушаване на сесия

- След завършване на работа със сесията е коректно първо да се разрегистрират всички променливи на сесията (ако има такива), а след това - да се разруши напълно сесията, чрез функция **session_destroy().**
- Забележка: Чрез session_destroy() прекратяваме сесията си и губим всички съхранени сесийни променливи.

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

Извеждане на идентификатора на сесията и директорията за съхраняване на сесийните променливи: session_id() и session_save_path();

```
<? php session_start();?>
<?php
echo "Session id=".session_id()."<br>";
echo "Директорията за съхраняване на сесийните
     променливи е ".session_save_path();   //   C:\xampp\tmp
```




```
session_destroy();

echo "Session id=".session_id()."<br>“;   ?>
<form>
<br><input type="submit" value="Reload">
</form>
```

Обяснения на примера:

- **string session_id([string $id])** – връща текущия и/или задава нов сесиен идентификатор.

Пример:

- echo "Session id=".**session_id()."<**br>";
- string session_save_path([string $path]) – връща и/или установява път ($path) до директорията, в която да се съхраняват сесийните файлове.
- Ако $path липсва, то се използва предния зададен път или този по подразбиране (зададен е в php.ini, в частност C:\xampp\tmp – за XAMPP).

Пример:

```
echo session_save_path(); // отпечатва C:\xampp\tmp
```

Раз-регистриране на сесийни променливи

- Ако регистрацията е чрез използване на асоциативни масиви (в частност - $_SESSION[]), то за раз-регистриране на сесийните променливи се използва функция unset() по следния начин:

```
unset($_SESSION["username"]);
```

- За примера, който дадохме:

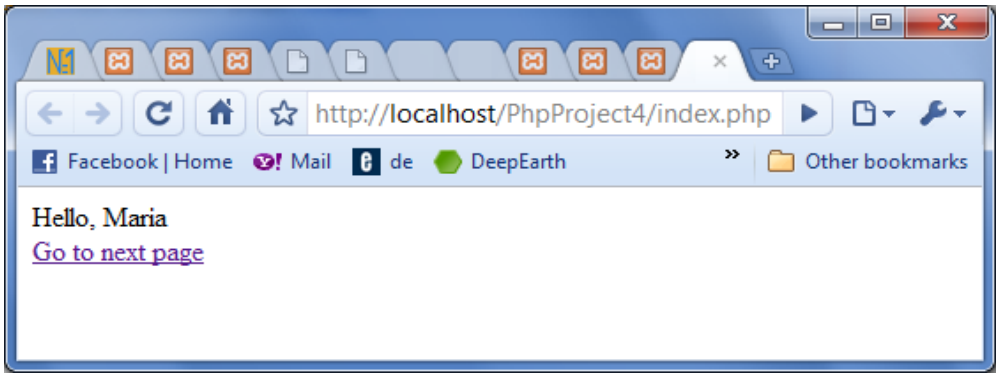
```
< ?php
unset($_SESSION['views']);
?>
```

Още един елементарен пример

- Да разгледаме един пример: стартиране на елементарна сесия, работеща с 3 web страници (index.php, page2.php и page3.php).
- Когато потребителят посети първата web страница (**index.php**) се открива сесия и се регистрира променлива **$username**. Съответстващият код е:

```
<?php
  session_start();
  $_SESSION['username'] = "Maria";
  echo 'Hello, '.$_SESSION['username']."<br>";
?>
<a href="page2.php">Go to next page </a>
```

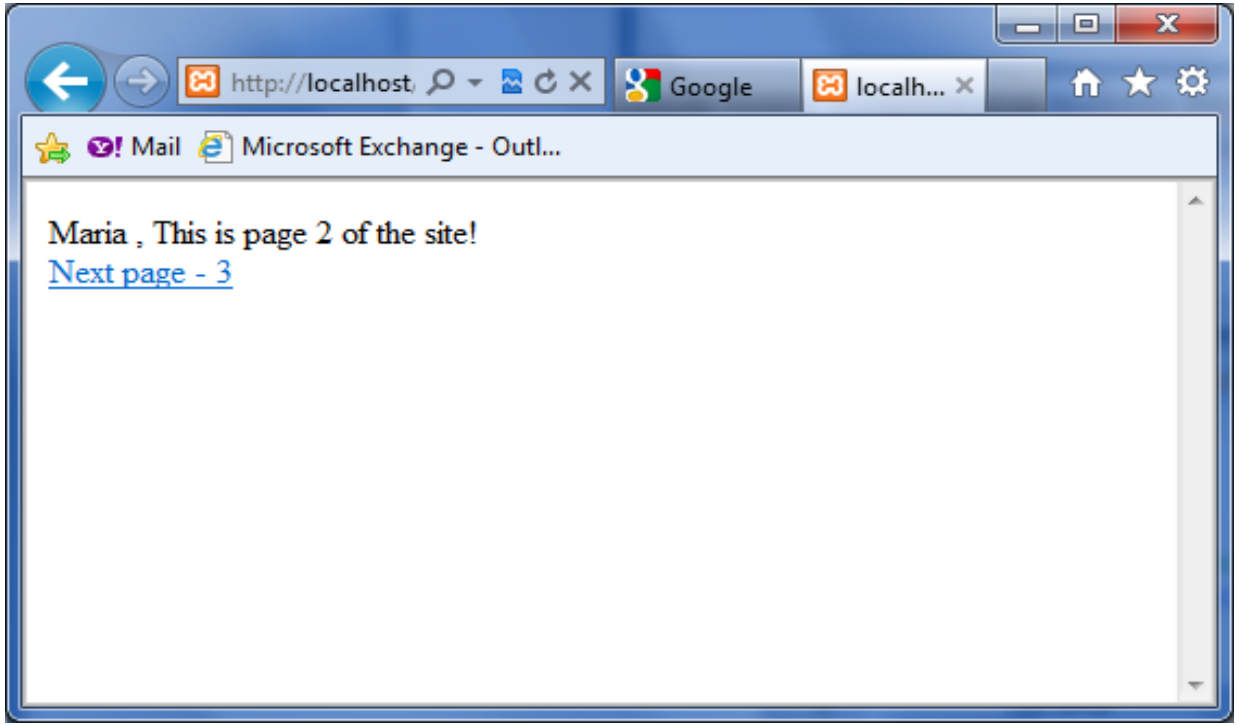
- Резултатът от работата на този сценарий е:



- Потребителят **"Maria"** щрака върху връзката и попада на страница **page2.php**, която има следния код:

```
<?php
session_start();
 echo $_SESSION['username'].' , This is page 2 of the
    site!';
  echo("<br>");
?>
  <a href="page3.php">Next page - 3 </a>
```

- Резултатът от работата на този скрипт:

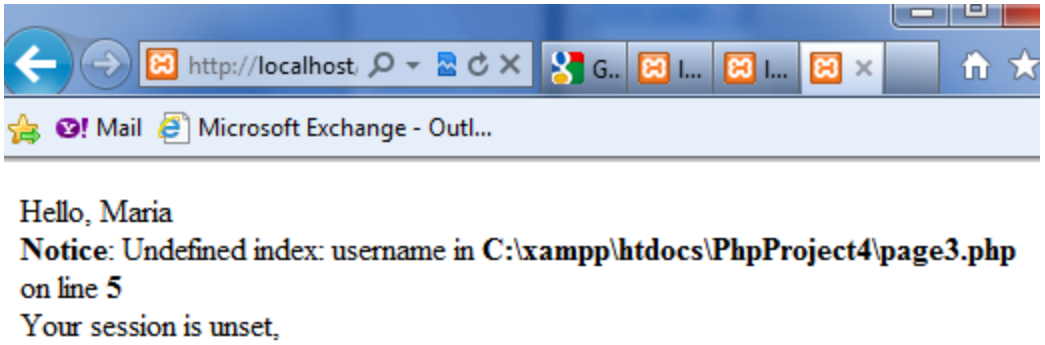


- При натискане на връзката, потребителят попада на страница page3.php, при което се извършва раз-регистрация на сеансовата променлива и унищожаване на сесията:

page3.php

```
<?php
session_start();
echo 'Hello, '.$_SESSION['username'];
unset($_SESSION['username']); // разрегистраме променливата
echo 'Your session is unset, '.$_SESSION['username'];
 /* Сега вече името на потебителя не се извежда */
session_destroy(); // разрушаване на сесия
?>
```

- Както се вижда след разрегистрация на сеансовата променлива стойността на масива **$_SESSION**['username'] е вече недостъпна:



Обяснение на примера:

- Имаме index.php, page2.php и page3.php. Ако не използваме **session_start() и в 3-те файла**, web сървърът по никакъв начин няма да разбере, че http заявките за 3-те файла са изпратени от един и същи потребител.
- В случая обаче и 3-те файла започват със session_start(), така при достъп до първия файл, интерпретаторът създава **нова** сесия. Но при поискване на 2-ра или 3-та страници, браузърът ще включи към заявката идентификатора на създадената сесия (чрез хедър Set_Cookie). Така session_start() функциите за 2-ра или 3-та страници, вместо да инициират нова сесия ще продължат да използват наличната, поради наличие на валиден сесиен идентификатор в http заявките.

- Има случаи, когато потребителят не позволява съхраняване на сесийна бисквитка на своята машина.
- Използва се друг метод за изпращане на идентификатор на сесията към браузъра, а именно:
  - Сесийният идентификатор се прикрепва директно към URL адреса, като се използва константата **SID**, която се дефинира, когато сесията е стартирана.

```
<?php
session_start();
if (empty($_SESSION['count'])) {
   $_SESSION['count'] = 1;
} else {   $_SESSION['count']++;}
?>
<p>
Hello visitor, you have seen this
page <?php echo
$_SESSION['count']; ?> times.
</p>
<p>
To continue, <a
href="nextpage.php?<?php echo
htmlspecialchars(SID); ?>">click
here</a>.
</p>
```

Резултат:

Hello visitor, you have seen this page 1 times.

To continue, click here.

-Този пример показва как да регистрирате променлива и как да преминем правилно към друга страница, използвайки SID.

-Използва се **htmlspecialchars()** при отпечатване на SID, с цел да се предотвратят възможни XSS атаки (използват се за крадене на бисквитка и сесия).

- Една сесия приключва, когато потребителят затвори браузъра си.
- Също (дори и да не е затворен браузъра), ако потребителят напусне сайта, сървърът ще прекрати автоматично сесията, обикновено след един предварително определен период от време (обикновено 30 минути).

Бисквитки (**cookies**)

- "Бисквитката" е другият начин за да бъде идентифициран даден потребител.
- "Бисквитката" е малък текстов файл с информация, от вида "име-значение", (най-често за идентифициране на потребителя), който файл ,,сървърът вгражда на компютъра на потребителя при (първото) поискване на дадена web страница.
- Така, всеки следващ път, когато потребителят поиска същата web страница с браузъра на компютъра си, той изпраща и бисквитката, като част от своето искане към сървъра.
- С PHP, може едновременно да създаде и да се извлече стойността на бисквитката.

Ползата от бисквитки

- Използването на cookies (бисквитки) е удобно както за програмистите, така и за ползвателите.
- Ползвателите печелят от това, че не им се налага всеки път да въвеждат информация за себи си, а на програмистите бисквитките помагат лесно и надеждно да съхраняват информация за ползвателите.

Cookies работят по следния начин:

1. Потребителят заявява (чрез адрес на Web сайт в браузъра си) даден Web сайт. Ако сайта е открит успешно, то заявката е приета от Web сървъра, съхраняващ този сайт.
2. Ако Web сървъра приеме заявката, Web браузърът проверява за наличието на cookies за този сайт.

3. Ако се открият cookies, браузърът ги изпраща (т.е изпраща всички двойки име-стойност към сървъра като HTTP заявка). Освен това се изпраща срока на валидност на cookie, ако има такова и пътят. Датата показва, датата и часа, до който cookie ще бъде валидно. Пътят помага на Web сървъра да асоциира различните стойности в cookie в различните страници, които са били посетени. След като тази информация се получи от сървъра, тя се използва вътрешно от него.

4. Ако не се открият cookies , сървърът се уведомява за отсъствието им. В този случай сървърът генерира нов идентификатор за клиента, който инициира връзката и изпраща cookie, съдържаща двойки име-стойност, към Web браузъра на клиента. След това браузърът съхранява cookie на вашата машина.

Бисквитки (**cookies**)

- В допълнение към основната информация, която съхранява бисквитката "име-значение“ (**string name [, string value**), всяка бисквитка има и набор от атрибути:
  - изтичане срока на годност (**int expire)**,
  - валиден домейн (**string domain)**,
  - валиден път на домейн (**string path)** и
  - възможно - флаг за сигурност (**int secure**).
- Тези атрибути помагат да се гарантира, че браузърът изпраща правилните "бисквитка", когато подава заявка до сървъра.

Създаване на бисквитка

- Създаването на бисквитка става с помощта на функция **setcookie**. Тази **функция създава бисквитка,** която се изпраща към потребителския браузер, от web сървъра, чрез заглавната част на HTTP отговора на web сървъра:

bool setcookie (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]])

Функция **setcookie** има следните аргументи:

- **name** - име на създаваната cookie;
- **value** - значение, съхранявано в cookie с име **$name**;
- **expire** – задава времето в секунди, след изтичането на което, бисквитката става невалидна (интервалът се добавя към текущото време time()). Стойността по подразбиране на този параметър е докато браузъра се затвори. Това означава, че cookie е валидна само докато страницата е отворена в прозореца на браузъра. В момента, когато потребителя затвори страницата, cookie става неизползваема.
- **path** – път в URL адреса (www.domain.com/path1/path2/), с който се асоциира бисквитката, например стойност "/path1/" – означава бисквитка, валидна за path1 и поддиректориите, а стойност "/" означава валидност за целия домейн.

Функция **setcookie** има следните аргументи:

- **domain** – домейн, с който се асоциира бисквитката; За да се направи бисквитката валидна за всички суб-домейни на домейн example.com – то трябва да се запише ".example.com". Ако запишем www4k.example.com – бисквитката ще е валидна само за суб-домейн www4k.
- **secure** (Сигурност) – определя дали, бисквитката да се изпраща само по криптиран канал за комуникация (протокол HTPPS - Secure Hypertext Transfer Protocol) или не. По подразбиране този параметър има стойност 0, което означава възможност за достъп по обичайна HTTP заявка.

setcookie() / setrawcookie()

- Стойността (value) на създадената с функция **setcookie()** бисквитка, автоматично се кодира към URL-а (URLencoded), когато се изпраща бисквитката (от браузъра) към сървъра и автоматично се декодира, когато се получава от браузъра.
- За да се избегне URLencoding, може да използвате setrawcookie() вместо setcookie() функция.

Пример за елементарно приложение с бисквитки

- Да създадем елементарен сценарий, който да дава възможност да се отчита с помощта на cookie количеството на обръщенията на посетителя към web страница.
- В **cookie** с име **counter** се съхранява броя на посещенията на страницата от потребителя:

Пример:

- Използване на асоциативния масив $_COOKIE за достъп до бисквитка:

```
<?php
 $_COOKIE["counter"];
 setcookie("counter",6);
 echo 'The page is visited by '.$_COOKIE['counter'].' times!!!';
?>
 // The page is visited by 5 times!!!
```

Пример:

- Използване на асоциативния масив $_COOKIE

```
<?php
$returnCookie=setcookie("user","joe");
if($returnCookie)
{echo $_COOKIE['user'];}
else
{var_dump($returnCookie);}
?>
//joe
```

Задаване на срока годност на cookies

- По подразбиране, cookies се създават за един сеанс на работа с браузъра, но може да им зададем по дълъг срок.
- Това е много удобно за потребителя: ако предостави свои данни при посещение на даден сайт, тогава няма да му се налага да ги задава при всяко посещение на сайта.

Задаване на срока годност на cookies

- **Пример:**

```
<?php
 setcookie("c_name", "Adam", time()+60);
//създаване на бисквитка c_name, със срок 1
   минута
 echo $_COOKIE["c_name"];
//извеждане на бисквитка
 ?>
// Adam
```

Изчистване на стойността на бисквитка и четене на достъпни бисквитки

- Извиква се функция **setcookie** и се предава само името на бисквитката, чиято стойност трябва да се изчисти:

```
setcookie("name");
```

- **Четене на достъпните бисквитки чрез $_COOKIE**

```
<?php
echo $_COOKIE["user"]; // Отпечатване на
    стойността на бисквитка user
print_r($_COOKIE); // Отпечатване на всички
    бисквитки
 ?>
```

четене на достъпни бисквитки - пример

```
<?php
 setcookie("name", "John", time()+60);
 setcookie("age", "22", time()+60);
 setcookie("c_name", "Adam", time()+60);
 //Setting Cookies with PHP, time 60 seconds
 echo $_COOKIE["c_name"]."<br />"; //displays Cookie c_name
 echo $_COOKIE["name"]. "<br />";//displays Cookie name
echo $_COOKIE["age"] . "<br />";//displays Cookie age
print_r($_COOKIE); //displays all Cookies
?>
<form>
<br><input type="submit" value="Reload">
</form>
```

Изход:

```
Adam
John
22
Array ( [name] => John [age] => 22 [c_name] => Adam )
```

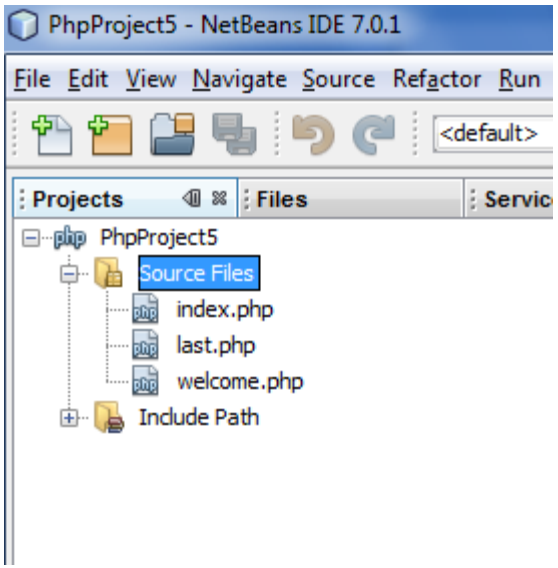
Един демонстрационен проект, включващ 3 файла:

- index.php, създава бисквитка

"user" - "Alex Porter".

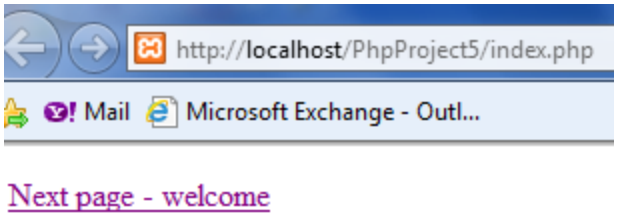
- welcome.php, проверява дали съществува бисквитката и разпечатва всички бисквитки
- last.php – унищожава

бисквитката



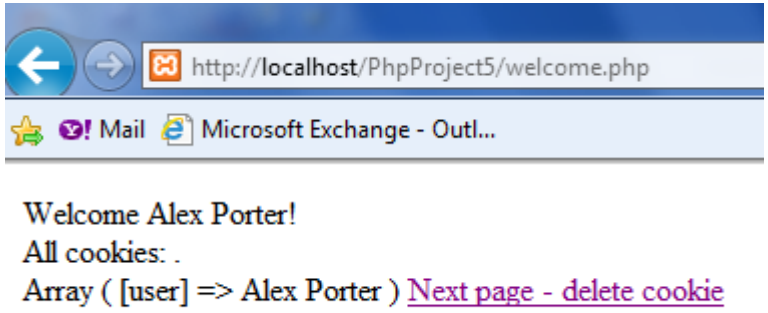
B index.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
   <body>
<?php
 $expire=time()+3600;
 setcookie("user", "Alex Porter", $expire);
 ?>
 <a href="welcome.php">Next page - welcome </a>
    </body>
</html>
```



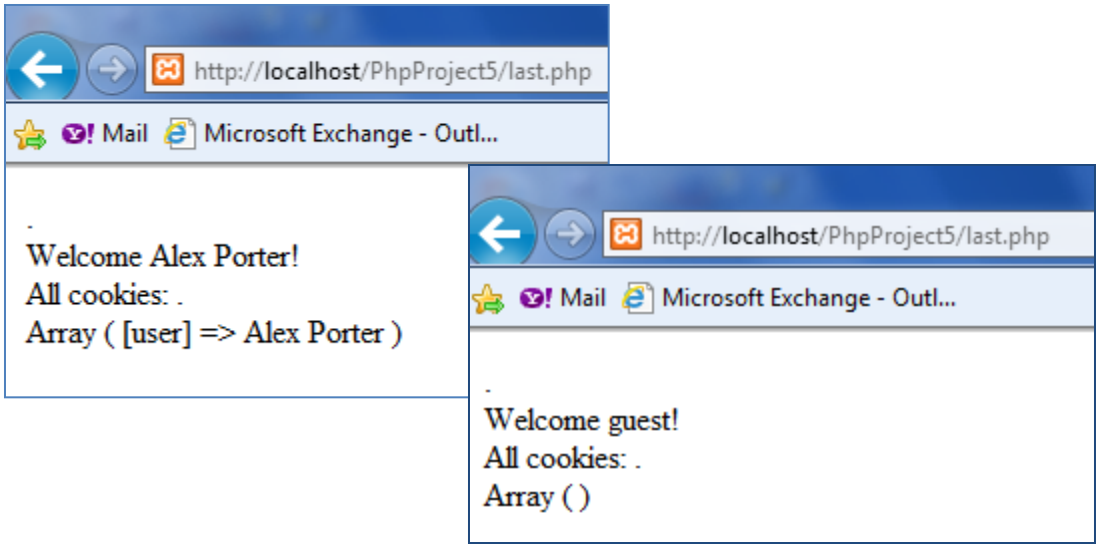
welcome.php

```
<?php
if (isset($_COOKIE["user"])) //Print a cookie
   echo "Welcome " . $_COOKIE["user"] . "!<br />";
 else
   echo "Welcome guest!<br />";
?>
All cookies: .<br />
 <?php
print_r($_COOKIE); // A way to view all cookies
?>
<a href="last.php">Next page - delete cookie </a>
```



last.php

```
<?php
// унищожане на бисквитката
// времето за изтичане се установява се намалява с 1 час по-рано от текущото време
setcookie("user", "", time()-3600); // setcookie("user");
?>.<br />
<?php
if (isset($_COOKIE["user"])) //Отпечатваме cookie, ако го има
  echo "Welcome " . $_COOKIE["user"] . "!<br />";
 else
  echo "Welcome guest!<br />";
?>
//Разпечатваме всички cookies
All cookies: .<br />
 <?php
print_r($_COOKIE);
?>
```



- При първо заявяване на модула last.php масивът $_COOKIE не е празен, но след Refresh се вижда, че вече няма бисквитки!!!
- Ако вместо setcookie("user", "", time()-3600); зададем setcookie("user"); То резултатът е:
- Welcome !

  All cookies: .

  Array ([user] =>)

Създаване на Cookie в заглавната част на HTML сорса

```
<!DOCTYPE html>
<HTML><HEAD>
<meta http-equiv="Set-Cookie" content="my_name=Mary">
<meta http-equiv="Set-Cookie" content="my_age=23">
</HEAD>
<BODY ><PRE>
<?php
setcookie("my_school","TU-Varna",time()+60);
//setcookie("my_school","TU-Varna",time()-60);
setcookie("city","Varna");
print_r($_COOKIE);
?>
</BODY></HTML>
```

```
Array
(   [my_school] => TU-Varna
    [city] => Varna
    [my_name] => Mary
    [my_age] => 23
)
```

Изпращане на заглавна част към потребителския браузер - функция header()

Ако вашият браузер не поддържа бисквитки, Php предоставя функция **header**, с която към Web браузера може да се изпраща произволна заглавна част, включително **бисквитка**:

```
<?php
setcookie("counter","Hello!!!");
header($_COOKIE['counter']);
print_r($_COOKIE);
?> //Array ( [counter] => Hello!!!)
```

или Пример 2:

```
<?php
header("Set-Cookie: user=joe");
/* header ("Set-Cookie: user=$_POST[email]; expires=Tue, 17-May-12 14:39:58
    GMT;path=/; domain=yourdomain.com");  */
print_r($_COOKIE);
?> //Array ( [user] => joe)
```

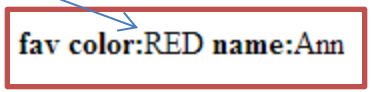
Изпращане на заглавна част към потребителския браузер - функция header()

- Следва един пример, демонстриращ предаване на информация за потребителя, чрез HTML форма и функция header ("Location: $url") – за изпращане на бисквитка към браузъра.
- Функцията header може да се използва и за съвсем други цели, например за зареждане в прозореца на браузера на друг документ.

```
<?php
header('Location:http://www.yahoo.com')
 ?>
```

Изпращане на бисквитка чрез форма и функция header()

```
<html>
<head>
<meta http-equiv="content-type"
content="text/html;charset=iso-8859-1" />
<title>Set a cookie</title>
</head>
<body>
<form action="form_data.php"
method="post">
<p>Please enter your  favorite color and
name:</p>
<p>Favorite color: <input type="text"
name="color" size="20" />
Name: <input type="text" name="name"
size="20" /></p>
<input type="submit" name="submit"
value="Submit" />
</form>
</body>
</html>
```



form_data.php:

```
?php
 if (!empty($_POST)) {
// set the cookie with the submitted user data
  setcookie('color',$_POST['color']);
  setcookie('name', $_POST['name']);
// redirect the user to final landing page so
cookie info is available
  header("Location:form_data.php");
 } else {
  echo "<b>fav color:</b>".$_COOKIE['color'];
  echo "<b>name:</b>".$_COOKIE['name'];
 }
?>
```

Разлика между бисквитки и сесии. Какво да използваме, бисквитка или сесия? Трябва да знаете разликата, за да решите

- Бисквитката съществува на локалния ви компютър, като малък текстов файл, съдържащ основна информация за вас като потребител на сайт (потребителско име и парола).
- Тя се съхранява в браузъра на потребителя, докато се изтрие, така че да не трябва отново да се въвежда всеки път, когато се достъпва уебсайта за който се отнася.
- Проблемът е, че потребителят може да блокира бисквитките или да ги изтрие по всяко време.

Разлика между бисквитки и сесии: Трябва ли да използвате бисквитка или сесия? Трябва да знаете разликата, за да решите

- Сесиите не зависят от потребителя, той няма власт да ги блокира.
- Проблемът със сесиите обаче е, че когато затворите браузъра си, те се губят. Така че, ако посещавате сайт, който изисква вход (login), потребителят ще бъде принуден да се логва отново всеки път, когато го посещава. Така, ако затворите Amazon.com, преди да сте направили всички покупки, вашата сесия приключва, независимо че бисквитката, която идентифицира вашето потребителско име все още съществува на локалния ви компютър.
- Разбира се, вие можете да се възползвате от най-доброто от двете неща! Можете да използвате комбинация от бисквитки и сесии, за да направите вашия сайт да работи точно по начина, който искате.