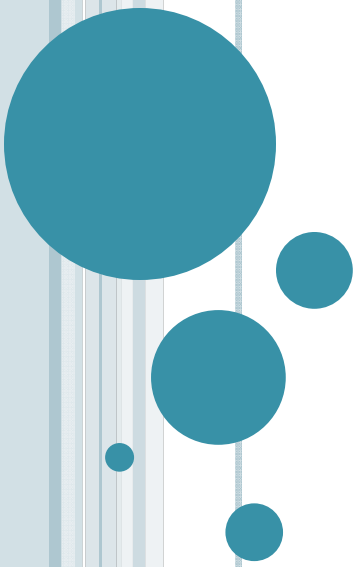
Maintaining State in PHP
Cookies & Sessions



Why to use Session and Cookie?

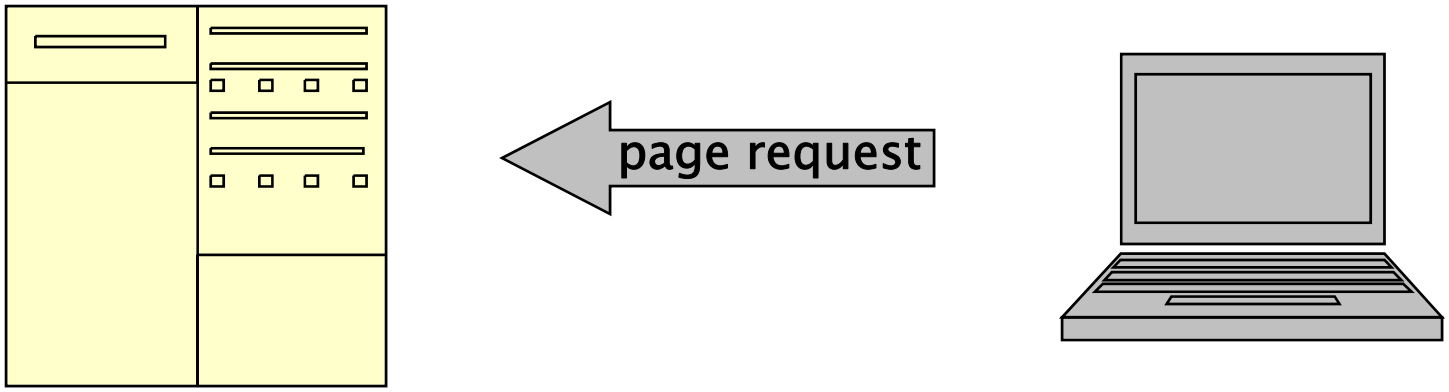
- HTTP is stateless Protocol.
- Any data you have stored is forgotten when the page is sent to the client and the connection is closed.
- Cookie is tiny bits of information that a web site could store on the client's machine that were sent back to the web site each time a new page was requested.
- Each cookie could only be read by the web site that had written it.

What is a Cookie?

- A cookie is a small text file that is stored on a user's computer.
- Each cookie on the user's computer is connected to a particular domain.
- Each cookie uses to store up to 4kB of data.
- A maximum of 20 cookies can be stored on a user's PC per domain.

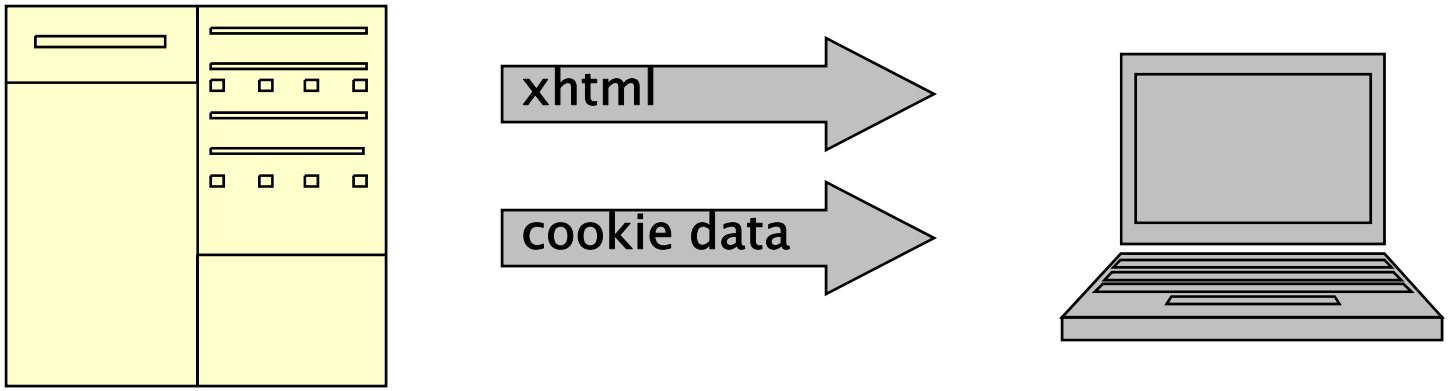
Example

1. User sends a request for page at www.testcookie.com for the *first* time.



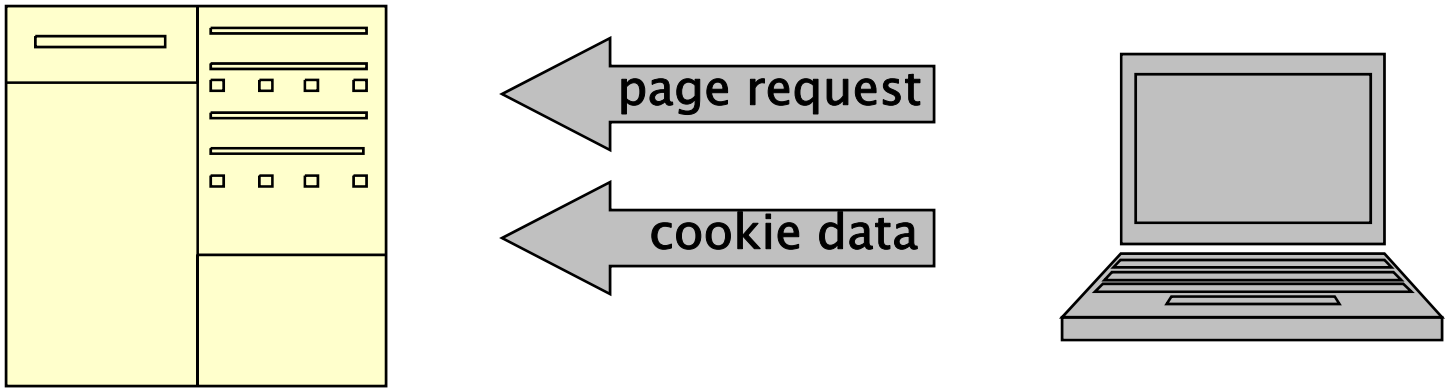
Example

2. Server sends back the page xhtml to the browser AND stores some data in a cookie on the user's PC.



Example

3. At the next page request for domain www.testcookie.com, all cookies data associated with this domain is sent too.



SET A COOKIE

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

name = cookie name

value = data to store (string)

expire = UNIX timestamp when the cookie expires. Default is that cookie expires when browser is closed.

path = Path on the server within and below which the cookie is available on.

domain = Domain at which the cookie is available for.

secure = If cookie should be sent over HTTPS connection only. Default false.

SET A COOKIE - EXAMPLES

```
setcookie('name','Robert')
```

- This command will set the cookie called name on the user's PC containing the data Robert.
- It will be available to all pages in the same directory or subdirectory of the page that set it (the default path and domain).
- It will expire and be deleted when the browser is closed (default expire).

SET A COOKIE - EXAMPLES

```
setcookie('age','20',time()+60*60*24*30)
```

- This command will set the cookie called age on the user's PC containing the data 20.
- It will be available to all pages in the same directory or subdirectory of the page that set it (the default path and domain).
- It will expire and be deleted after 30 days.

SET A COOKIE - EXAMPLES

```
setcookie('gender','male',0,'/')
```

- This command will set the cookie called gender on the user's PC containing the data male.
- It will be available within the entire domain that set it.
- It will expire and be deleted when the browser is closed.

Read cookie data

- All cookie data is available through the superglobal **$_COOKIE:**

```
$variable = $_COOKIE['cookie_name']
```

or

```
$variable = $HTTP_COOKIE_VARS['cookie_name'];
```

e.g.

```
$age = $_COOKIE['age']
```

Storing an array..

- Only **strings** can be stored in Cookie files.
- To store an array in a cookie, convert it to a string by using the `serialize()` PHP function.
- The array can be reconstructed using the `unserialize()` function once it had been read back in.
- Note : Cookie size is limited!

Delete a cookie

- To remove a cookie, simply overwrite the cookie with a new one with an expiry time in the past…

setcookie('cookie_name',",time()-6000)

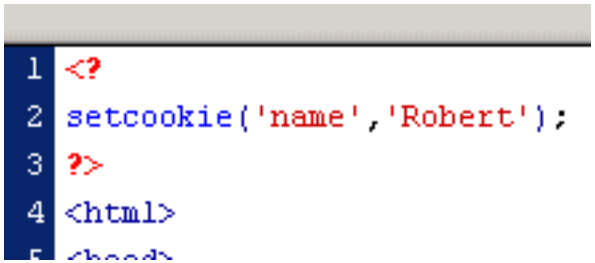
- Note that theoretically any number taken away from the time() function should do, but due to variations in local computer times, it is advisable to use a day or two.
-

PHP Sessions

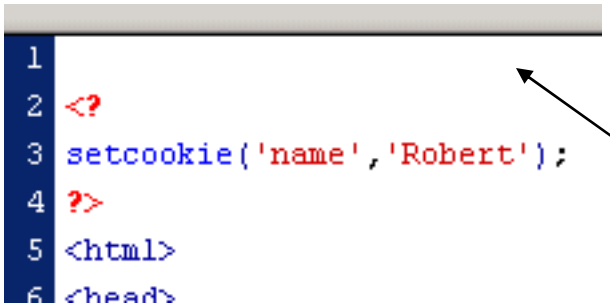
- You can store user information (e.g. username, items selected, etc.) in the server side for later use using PHP session.
- Sessions work by creating a unique id (UID) for each visitor and storing variables based on this UID.
- The UID is either stored in a cookie or is propagated in the URL.

Where to write code for Cookie…

- As the **setcookie** command involves sending a HTTP header request, it must be executed **before any xhtml is echoed to the browser, including whitespace**.



correct!



echoed whitespace before **setcookie**

incorrect.

How do 'Sessions' work?

- They are based on assigning each user a unique number, or **session id**.
- Even for extremely heavy use sites, this number can for all practical purposes can be regarded as **unique.**

e.g.

26fe536a534d3c7cde4297abb45e275a

How do 'Sessions' work?

- This **session id** is stored in a cookie, or passed in the URL between pages while the user browses.
- The data to be stored (e.g. name, log-in state, etc.) is stored **securely server-side in a PHP superglobal**, and referenced using the session id.

When should you use sessions?

- Need for data to stored on the server
- Unique session information for each user
- Transient data, only relevant for short time
- Data does not contain secret information
- Similar to Cookies, but it is stored on the server
- More secure, once established, no data is sent back and forth between the machines
- Works even if cookies are disabled
- Example: we want to count the number of “hits” on our web page.

How to Start?

- Before you can store user information in your PHP session, you must first start up the session.

session_start() function must appear BEFORE the **<html>** tag.

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

<html>
<body>

</body>
</html>
```

PHP Sessions

- Starting a PHP session:

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

- This tells PHP that a session is requested.
- A **session ID** is then **allocated** at the server end.

- **session ID** looks like:

sess_f12347812374681237 68asjkhfa7891234g

Session variables

- $_SESSION
- e.g., $_SESSION[“intVar”] = 10;

- Testing if a session variable has been set:

```
session_start();

if(!$_SESSION['intVar']) {...} //intVar is set or not
```

Registering session variables

- Instead of setting superglobals, one can register one's own session variables

```
<?php
    $barney = "A big purple dinosaur.";
    $myvar_name = "barney";
    session_register($myvar_name);
?>
```

- $barney can now be accessed "globally" from session to session
- This only works if the **register_globals** directive is enabled in **php.ini** - nowadays this is turned off by default

Use of session_register() is deprecated!

MAKE YOUR OWN SESSION VARIABLES

- With **session_start()** a default session variable is created - the name extracted from the page name
- To create your own session variable just add a new key to the **$_SESSION superglobal**

$_SESSION['dug'] = "a talking dog.";

Use of $_SESSION is preferred, as of PHP 4.1.0.

Session Example 1

```
<?php
    session_start();
    if (!isset($_SESSION["intVar"]) ){
        $_SESSION["intVar"] = 1;
    } else {
        $_SESSION["intVar"]++;
    }
    echo "<p>In this session you have accessed this page " .  $_SESSION["intVar"] . "times.</p>";
?>
```

Session Example 2

```
<?php session_start();?>
<?php
$thisPage = $_SERVER['PHP_SELF'];

$pageNameArray = explode('/', $thisPage);
$pageName = $pageNameArray[count($pageNameArray) - 1];
print "The name of this page is: $pageName<br/>";

$nameItems = explode('.', $pageName);
$sessionName = $nameItems[0];
print "The session name is $sessionName<br/>";

if (!isset($_SESSION[$sessionName])) {
        $_SESSION[$sessionName] = 0;
        print "This is the first time you have visited this page<br/>";
}
else {
        $_SESSION[$sessionName]++;
}
print "<h1>You have visited this page " . $_SESSION[$sessionName] .
   " times</h1>";
?>
```

Ending sessions

unset(**$_SESSION**[‘**name**’])

-Remove a session variable

session_destroy()

- Destroys all data registered to a session
- does not unset session global variables and cookies associated with the session
-Not normally done - leave to timeout

Destroying a session completely

```php
<?php
// Initialize the session.
// If you are using session_name("something"), don't forget it now!
session_start();

// Unset all of the session variables.
$_SESSION = array();

// If it's desired to kill the session, also delete the session cookie.
// Note: This will destroy the session, and not just the session data!
if (ini_get("session.use_cookies")) { // Returns the value of the configuration option
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}

// Finally, destroy the session.
session_destroy();
?>
```

returns the name of the current session

Session Example 3

```
<?php
session_start();

if(!isset($_SESSION['strColourBg']))   $_SESSION['strColourBg'] = "red";
else echo "Currently Bg set to " . $_SESSION['strColourBg'] . "<br>";
if(!isset($_SESSION['strColourFg'])) $_SESSION['strColourFg'] = "yellow";
else echo "Currently Fg set to " . $_SESSION['strColourFg'];

if(isset($_POST["submit"]) ) {
  $strColourBg = $_POST["strNewBg"];
  $strColourFg = $_POST["strNewFg"];
  $_SESSION['strColourBg'] = $strColourBg;
  $_SESSION['strColourFg'] = $strColourFg;
  echo "<br>New Settings";
}
else {
  $strColourBg = $_SESSION['strColourBg'];
  $strColourFg = $_SESSION['strColourFg'];
  echo "<br>Keep old settings";
}
?>
```