PHP 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.

PHP Cookies

- Cookies are small files that are stored in the visitor's/client's browser.
- Cookies can be used to identify return visitors, keep a user logged into a website indefinitely, track the time of the user's last visit, and much more.
- Cookies accept seven different arguments, but only the "name" is required.

PHP Cookies

- (Keep in mind that all values are stored on the visitor's computer, so the data is not private. Never store passwords in cookies, for example!) The options are:

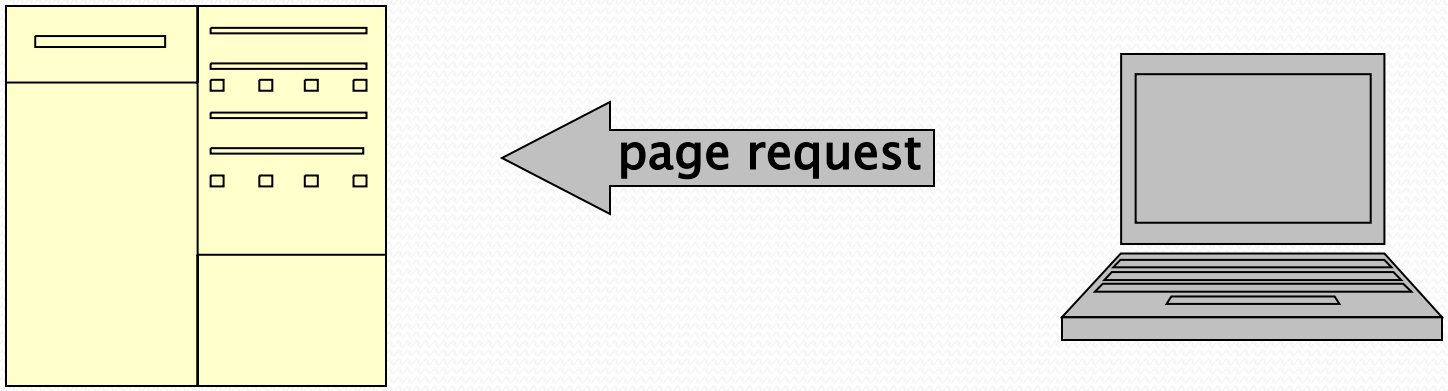
Argument	Description
name	Name of the Cookie
value	Value of the Cookie
expire	Time When Cookie Expires (Unix Timestamp) (If "0", Or Omitted, Cookie Will Expire When Browser Closes) (Set to Client's Time, Not Server's)
path	Server Path Where Cookie Is Available (If Path Is the Root Directory, Cookie Is Available In Entire Domain) (Default Value Is Current Directory)
domain	Domain That Cookie Is Available
secure	Indicates That Cookie Should Only Be Transmitted Over a Secure HTTPS Connection From Client
httponly	When TRUE, Cookie Is Only Accessible Through HTTP Protocol

- PHP allows you to create, retrieve and update cookies.
- The setcookie() function is used to first create a cookie.
- This function must be run before any other data is sent to the browser, such as the opening <html> tag or random whitespace.
- The syntax is:

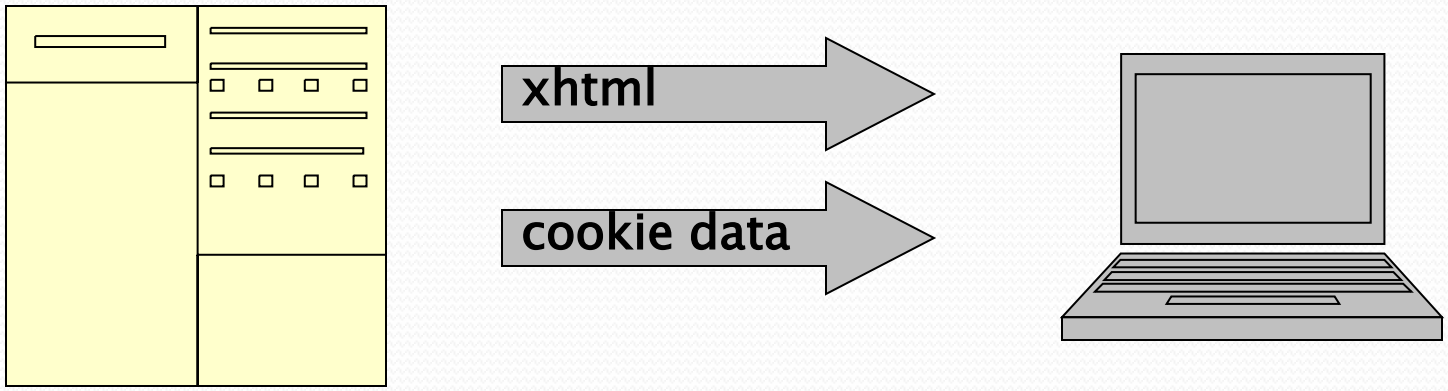
  E.g. setcookie(name, value, expire, path, domain);

Example

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

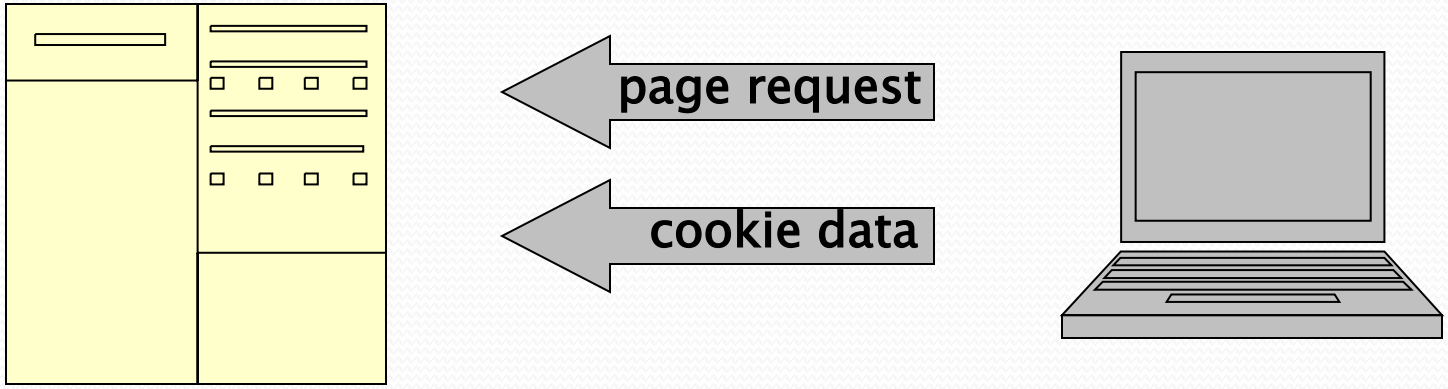


2. Server sends back the page xhtml to the browser AND stores some data in a cookie on the user's PC.



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 = Required. Specifies the name of the cookie.

value = Optional. Specifies the value of the cookie.

expire = Optional. Specifies when the cookie expires. The value: time()+86400*30, will set the cookie to expire in 30 days. If omitted or set to 0, the cookie will expire at the end of the session (when the browser closes). Default is 0.

path = Optional. Specifies the server path of the cookie. If set to "/", the cookie will be available within the entire domain. If set to "/php/", the cookie will only be available within the php directory and all sub-directories of php. The default value is the current directory.

domain = Optional. Specifies the domain name of the cookie. To make the cookie available on all subdomains of php.com, set domain to “php.com". Setting it to www.php.com will make the cookie only available in the www subdomain

secure = Optional. Specifies whether or not the cookie should only be transmitted over a secure HTTPS connection. Default is False.

Set a cookie - examples

- This method returns true on successfully setting a cookie and False otherwise.
- PHP 5.2 - The httponly parameter was added.
- httponly : If set to TRUE the cookie will be accessible only through the HTTP protocol.
- E.g. `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*3
0)
```

- 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.

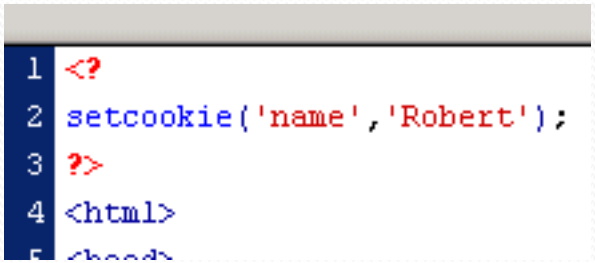
```
$age = $_COOKIE['age']
```

Storing an array..

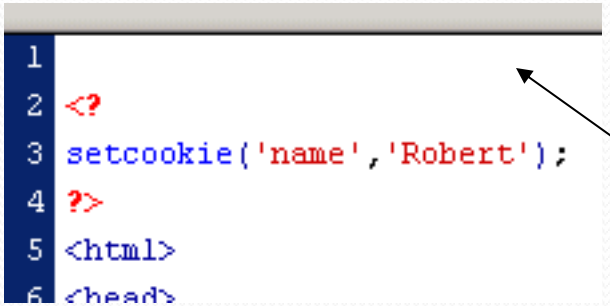
- Only **strings** can be stored in Cookie files.
- To store an array in a cookie, convert it to a string by using the `serialize()` PHP function.
- The array can be reconstructed using the `unserialize()` function once it had been read back in.
- Note : Cookie size is limited!

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.

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.

When should you use sessions?

- Suppose you are building one E-commerce site, to allow any one to buy the product you must ask them to log-in with their user name and until they log out your system must track the user in every step, this concept is called as “session tracking”.
- Now why do we need to track the session?
  - HTTP is state less protocol, and when you refresh the page, it lost everything, which your project should not !

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.

Why and when to use Sessions?

- You want to store important information such as the user id more securely on the server where malicious users cannot temper with them.
- You want to pass values from one page to another.
- You want the alternative to cookies on browsers that do not support cookies.
- You want to store global variables in an efficient and more secure way compared to passing them in the URL
- You are developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

Creating a Session

- In order to create a session, you must first call the PHP session_start function and then store your values in the $_SESSION array variable.
- Let's suppose we want to know the number of times that a page has been loaded, we can use a session to do that.
- The code below shows how to create and retrieve values from sessions

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

What is a Session?

- A session is a global variable stored on the server.
- Each session is assigned a unique id which is used to retrieve stored values.
- Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL
- Sessions have the capacity to store relatively large data compared to cookies.
- The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.
- Just like the $_COOKIE array variable, session variables are stored in the $_SESSION array variable. Just like cookies, the session must be started before any HTML tags.
- You want to store important information such as the user id more securely on the server where malicious users cannot temper with them.
- You want to pass values from one page to another.
- You want the alternative to cookies on browsers that do not support cookies.
- You want to store global variables in an efficient and more secure way compared to passing them in the URL
- You are developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

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.

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>
```

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

How to Start?

Turning on Auto Session

You don't need to call start_session() function to start a session when a user visits your site if you can set **session.auto_start** variable to 1 in **php.ini** file.

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 68asjkhfa78912 34g

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>";
?>
```

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 Session Variables

- The session_destroy() function is used to destroy the whole Php session variables.
- If you want to destroy only a session single item, you use the unset() function.
- The code below illustrates how to use both methods.
- session_destroy(); //destroy entire session
- unset($_SESSION['product']); //destroy product session item
- Session_destroy removes all the session data including cookies associated with the session. Unset only frees the individual session variables. Other data remains intact.

Sessions without cookies

- There may be a case when a user does not allow to store cookies on their machine.
- There is another method to send session ID to the browser.
- Alternatively, you can use the constant SID which is defined if the session started.
- If the client did not send an appropriate session cookie, it has the form session_name = session_id.
- Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.

Hidden From Fields

- A hidden field is not displayed on the page.
- It simply stores the text value specified in the value attribute.
- Hidden fields are great for passing additional information from the form to the server.

Summary

- Cookies are small files saved on the user's computer
- Cookies can only be read from the issuing domain
- Cookies can have an expiry time, if it is not set, then the cookie expires when the browser is closed
- Sessions are like global variables stored on the server
- Each session is given a unique identification id that is used to track the variables for a user.
- Both cookies and sessions must be started before any HTML tags have been sent to the browser.