CMM007 – Cookies and sessions

Todays session is used to introduce the use of cookies and sessions on a website. This is a method that you can use to aid in authenticating users and also to create a ‘login’ system for your site.

This lab covers the following:

Table of Contents

Section 1 – Using Cookies 2

Setting Cookies 2

Reading Cookies 2

Deleting Cookies 2

Activity 1 – baking with your own cookies 3

Section 2 – using sessions 4

Adding Session Data 4

Reading Session Data 4

Removing Session Data 4

Ending a session 4

Checking Session Data 5

Activity 2 – Getting your code working with sessions 5

# Section 1 – Using Cookies

## Setting Cookies

Setting cookies can be accomplished by using the following code:

setcookie('flavour','chocolate chip');

The set cookie function can have more information passed into it.

setcookie('flavour','chocolate chip', 1417608000,'/products',www.example.com');

In total, there are 5 different pieces of information that you can send back. You don’t need to send back all of them – just the ones that you want. They are briefly explained below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Setcookie( | ‘flavour’, | ‘chocolate chip’, | 1417608000, | ‘/products’, | ‘www.example.com’); |
| Starts the set cookie function | Sets the name of the cookie | Sets the value of the cookie | Sets the expire time (as an EPOCH time stamp | Sets the path that cookies will be sent back under | Sets the domain that cookies will be sent back under |

So, for example, if you wanted to only set the cookie name, value and timestamp, you would have something like this:

setcookie('flavour','chocolate chip', 1417608000);

If you wanted to set the cookie name, value and a domain; you would have something like this:

setcookie('flavour','chocolate chip', ,'',www.example.com');

(notice how you just leave the bits in the middle blank)

## Reading Cookies

As well as setting cookies, you are also going to want to be able to read them at one point or another. This can be accomplished by looking in the $\_COOKIE superglobal array.

**if** (isset($\_COOKIE['flavor'])) {

**echo** "You ate a “ . $\_COOKIE['flavor'] . ” cookie.";

}

It is also possible to look through all of the cookies that you have in a particular session (when you are using third party plug ins they may send some without your knowledge…). To do this, code similar to below will be handy:

**foreach** ($\_COOKIE **as** $cookie\_name => $cookie\_value) {

**print** "$cookie\_name = $cookie\_value <br/>";

}

## Deleting Cookies

Sometimes your are going to want to delete your cookies. The easiest way to do this is to call the setcookie() function with no value in it and put the expiration time as some point in the past. For example:

setcookie('flavor','',1);

### Activity 1 – baking with your own cookies

Create a web application that allows a user to log in to a website. You will need to implement the following functionality:

1. Login Page – Displays a username and password box for a user
2. Check Login – Checks that the username and password entered are correct
   1. For this, just hard code in a username and password that can be used

$username = “Mike”;

$password = “mysecretpassword”;

* 1. If the password details are correct set two cookies. The first should be the username, the second should be an ‘access level’ for the user. Set the access level to be something similar to the following:

setcookie(‘access\_level’,'standarduser');

1. Create a page called *homepage.php* that a user can visit once they have logged in. This page will display their name and also their access level. Inlclude the script below in this page. Call on this script and pass in the *access\_level* cookie.

function displayAccessLevelInformation($accessLevel) {  
 if ($access\_level == "standarduser") {  
 echo "<p>You are currently logged in as a standard user</p>";  
 }  
 elseif ($access\_level == "root") {  
 echo "<p>You are currently logged in as a root user</p>";  
 echo "<p>You now have access to additional administrative features</p>";  
 }  
}

Once you have completed the above, download the *Edit This Cookie* extension for chrome. Use this to see if you can ‘trick’ the web application that you have made to give you root administrative rights.

# Section 2 – using sessions

A session is a combination of a server-side file containing all the data you wish to store, and a client-side cookie containing a reference to the server data. The file and the client-side cookie are created using the function *session\_start()* - it has no parameters, but informs the server that sessions are going to be used.

When you call *session\_start()*, PHP will check to see whether the visitor sent a session cookie - if it did, PHP will load the session data. Otherwise, PHP will create a new session file on the server, and send an ID back to the visitor to associate the visitor with the new file. Because each visitor has their own data locked away in their unique session file, you need to call *session\_start()* before you try to read session variables - failing to do so will mean that you simply will not have access to their data. Furthermore, as *session\_start()* needs to send the reference cookie to the user's computer, you need to have it before the body of your web page - even before any spaces.

## Adding Session Data

All your session data is stored in the session superglobal array, $\_SESSION, which means that each session variable is one element in that array, combined with its value. Adding variables to this array is done in the same way as adding variables to any array, with the added the bonus that session variables will still be there when your user browses to another page.

To set a session variable, use syntax like this:

$\_SESSION['var'] = $val;

$\_SESSION['FirstName'] = "Jim";

## Reading Session Data

Once you have put your data safely away, it becomes immediately available in the $\_SESSION superglobal array with the key of the variable name you gave it. Here is an example of setting data and reading it back out again:

<?php

$\_SESSION['foo'] = 'bar';

print $\_SESSION['foo'];

?>

## Removing Session Data

Removing a specific value from a session is as simple as using the function *unset()*, just as you would for any other variable. It is important that you should unset only specific elements of the $\_SESSION array, not the $\_SESSIONarray itself, because that would leave you without any way to manipulate the session data at all.

To extend the previous script to remove data, use this:

<?php

$\_SESSION['foo'] = 'bar';

print $\_SESSION['foo'];

unset($\_SESSION['foo']);

?>

## Ending a session

A session lasts until your visitor closes their browser - if they navigate away to another page, then return to your site without having closed their browser, *their session will still exist* . This behaviour is usually desirable - potentially your visitor's session data might last for days, as long as they keep browsing around your site, whereas cookies usually have a fixed lifespan.

If you want to explicitly end a user's and delete their data without them having to close their browser, you need to clear the $\_SESSION array, then use the *session\_destroy()* function. *Session\_destroy()* removes all session data stored on your hard disk, leaving you with a clean slate.

To end a session and clear up its data, use this code:

<?php

session\_start();

$\_SESSION = array();

session\_destroy();

?>

## Checking Session Data

You can check whether a variable has been set in a user's session using the function *isset()*, as you would a normal variable. Because the $\_SESSION superglobal is only initialised once *session\_start()* has been called, you need to call*session\_start()* before using *isset()* on a session variable. For example:

<?php

session\_start();

if (isset($\_SESSION['FirstName'])) {

/// your code here

}

?>

### Activity 2 – Getting your code working with sessions

Create a copy of the program that you had made previously. Adjust it so that it no longer uses cookies but uses sessions instead.

Once you have completed this, see what happens when you try and change the session cookie that is being used. Can you still change your *administrative rights* to be root?