|  |  |  |
| --- | --- | --- |
| **BESE 16-A/B** | **CS340 Web Technologies** | **Dr Naima Iltaf** |

**Lab 10 – Session and File Upload in PHP**

**Using Cookies in PHP:**

A cookie is an item of data that a web server saves to your computer’s hard disk via a web browser. Common uses include session tracking, maintaining data across multiple visits, holding shopping cart contents, storing login details, and more.

Because of their privacy implications, cookies can be read only from the issuing domain. In other words, if a cookie is issued by, for example, Hotmail.com, it can be retrieved only by a web server using that domain. This prevents other websites from gaining access to details to which they are not authorized.

**Setting a Cookie:**

To set a cookie in PHP is a simple matter. As long as no HTML has yet been transferred, you can call the setcookie function, which has the following syntax:

setcookie(name, value, expire, path, domain, secure, httponly);

**Accessing a Cookie:**

Reading the value of a cookie is as simple as accessing the $\_COOKIE system array. For example, if you wish to see whether the current browser has the cookie called user-name already stored and, if so, to read its value, use the following:

if (isset($\_COOKIE['username']))

$username = $\_COOKIE['username'];

Note that you can read a cookie back only after it has been sent to a web browser. This means that when you issue a cookie, you cannot read it in again until the browser reloads the page (or another with access to the cookie) from your website and passes the cookie back to the server in the process.

**Destroying a Cookie:**

To delete a cookie, you must issue it again and set a date in the past. It is important for all parameters in your new setcookie call except the timestamp to be identical to the parameters when the cookie was first issued; otherwise, the deletion will fail. Therefore, to delete the cookie created earlier, you would use the following:

setcookie('username', 'Hannah', time() - 2592000, '/');

As long as the time given is in the past, the cookie should be deleted. However, I have used a time of 2592000 seconds (one month) in the past in case the client computer’s date and time are not correctly set.

**PHP Sessions:**

A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.

**Storing a Session Variable:**

The correct way to store and retrieve session variables is to use the PHP $\_SESSION variable:

<?php

session\_start();

if(isset($\_SESSION['views']))

$\_SESSION['views']=$\_SESSION['views']+1;

else

$\_SESSION['views']=1;

echo "Views=". $\_SESSION['views'];

?>

**Destroying a Session:**

If you wish to delete some session data, you can use the unset() or the session\_destroy() function.

The unset() function is used to free the specified session variable:

<?php

session\_start();

if(isset($\_SESSION['views']))

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

?>

You can also completely destroy the session by calling the session\_destroy() function:

<?php

session\_destroy();

?>

session\_destroy() will reset your session and you will lose all your stored session data.

**Setting a timeout:**

There are other times when you might wish to close a user’s session yourself, such as when the user has forgotten or neglected to log out, and you wish the program to do it for them for their own security. The way to do this is to set the timeout, after which a logout will automatically occur if there has been no activity.

To do this, use the ini\_set function as follows. This example sets the timeout to exactly one day:

ini\_set('session.gc\_maxlifetime', 60 \* 60 \* 24);

If you wish to know what the current timeout period is, you can display it using the following:

echo ini\_get('session.gc\_maxlifetime');

**PHP - file upload:**

A very useful aspect of PHP is its ability to manage file uploads to your server. Allowing users to upload a file to your server creates a security issue, so be careful when enabling file uploads.

<form enctype="multipart/form-data" action="uploader.php" method="POST">

<input type="hidden" name="MAX\_FILE\_SIZE" value="100000" />

Choose a file to upload: <input name="uploadedfile" type="file" /><br />

<input type="submit" value="Upload File" />

</form>

Here is a brief description of the important parts of the above code:

* enctype="multipart/form-data" - Necessary for our to-be-created PHP file to function properly.
* action="uploader.php" - The name of our PHP page that will process the file.
* method="POST" - Informs the browser that we want to send information to the server using POST.
* input type="hidden" name="MA... - Sets the maximum allowable file size, in bytes, that can be uploaded. We have set the max file size to 100KB in this example.
* input name="uploadedfile" - uploadedfile is how we will access the file in our PHP script.

After the user clicks submit, the data will be posted to the server and the user will be redirected to uploader.php. This PHP file is going to process the form data and do all the work. After the user clicks submit, the data will be posted to the server and the user will be redirected to uploader.php. This PHP file is going to process the form data and do all the work.

**PHP - file upload: uploader.php**

When the uploader.php file is executed, the uploaded file exists in a temporary storage area on the server. If the file is not moved to a different location it will be destroyed! To save our file we are going to need to make use of the $\_FILES associative array.

The $\_FILES array is where PHP stores all the information about files. There are two elements of this array that we will need to understand for this example.

* uploadedfile - uploadedfile is the reference we assigned in our HTML form. We will need this to tell the $\_FILES array which file we want to play around with.
* $\_FILES['uploadedfile']['name'] - name contains the original path of the user uploaded file.
* $\_FILES['uploadedfile']['tmp\_name'] - tmp\_name contains the path to the temporary file that resides on the server. The file should exist on the server in a temporary directory with a temporary name.

<?php

$target\_path = "C:/Users/Misbah/Desktop/uploads/";

$target\_path = $target\_path . basename( $\_FILES['uploadedfile']['name']);

**if**(move\_uploaded\_file($\_FILES['uploadedfile']['tmp\_name'], $target\_path)) {

**echo** "The file ". basename( $\_FILES['uploadedfile']['name']).

" has been uploaded";

} **else**{

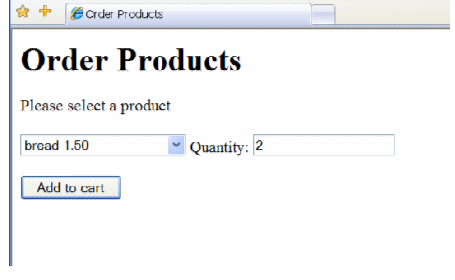
**echo** "There was an error uploading the file, please try again!";

}

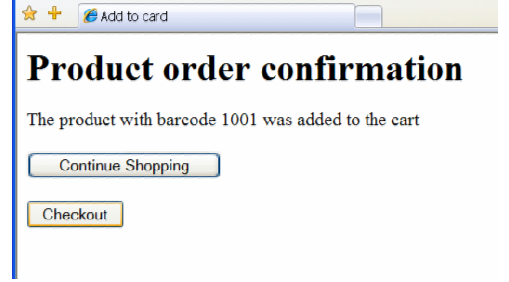
?>

**Lab Tasks**

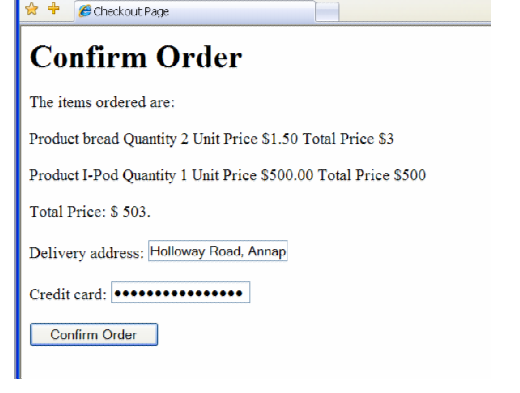
1. **Implement an online shopping cart using PHP session and MySQL. It should have the following properties: (7)**
2. **The products stored in database should appear in the dropdown box. The database should also consist of unit price of each item for total cost collection. A textbox should take the quantity of product as input.**



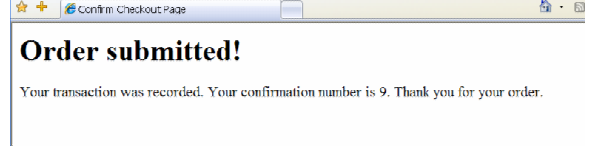
1. **When user presses Add to cart button, following page should appear.**



**If the user presses the Continue shopping button, redirect it to the previous page. Using session variable, also keep record of the products the user is adding to the cart. If the user presses Checkout button, following page should appear.**



**When user confirms order, save this information into database as well as assign a confirmation number to the customer. This step will also destroy the session and close the connection to database. The customer should not see any items in the shopping cart, even if it executes the checkout code again.**



**2. Write a PHP program for the following specifications. (3)**

**Create Account**

New user should be able to create his/her account on your website. You should have a form for this. Two types of accounts should exist, user and admin accounts.

**Log In (User)**

Check the name and password that the user has entered. If they match with your database, redirect the user to a page with his/her profile details and admin to his profile page. If they do not, return the user to the index page.

On users profile page, user should be able to:

* Edit the information he has entered and update it.

Whereas on admin’ profile page, admin should be able to:

* See the list of users all having account on website.
* Able to delete any users account.