DFTI Worksheet 5
Session-based login systems

Introduction

In this week's tutorial you will add a login system to the HitTastic! site. Users will only be able to access the site if they are logged in.

Please try and run this without EPHP this week, and only use EPHP if you are really stuck. Hopefully, after four weeks, you will now be comfortable with the basics of PHP, and you will not need EPHP to help you. This week therefore gives a good opportunity to make the transition to developing without EPHP.

To run your site in the browser directly, go to

http://ephp.solent.ac.uk/~ephpXXX/

Questions

Part 1 – Writing the login script

1. Create an HTML page to allow users to log in. You can do this by copying and pasting the HTML code below into Notepad++. **Upload it as login.html.**

```
<!DOCTYPE html>
<html>
<head>
<title>HitTastic! Login</title>
</head>
<body>
<h1>Log in to HitTastic!</h1>
<form method="post" action="login.php">

<label for="username">Username:</label>
<input name="username" id="username"/>
<br/>

<label for="password">Password:</label>
<input name="password" id="password" type="password"/>
<br/>

<input type="submit" value="Go!" />
</form>
</body>
</html>
```

2. Write the login script (name it ***login.php***). Your login script should check that there is a row in the database containing the username and the password that the user entered; if there is, you will know that it is a valid combination of username and password. If there is, the "gatekeeper" session variable should be set to the username, and the user redirected to your HitTastic! search form. If they do not match, an error message should be displayed to the user.

To do this:

a) Start the script by looking at the simple login example on the notes. Read the username and password from the form into two variables, $un and $pw, as is done on the example in the lecture notes.

b) In your PHP, write an SQL SELECT statement to search the database for any rows which contain both the username and the password that the user entered.

c) We now need to see if there were any rows returned from the SELECT statement. As we saw in lecture 3, we can do this by sending the query to the database and then trying to fetch a row from the results. $results ->fetch() returns the value "false" if there are no matching rows, or the first matching row otherwise. We can use this in a login script in conjunction with the SQL SELECT statement from part b), using code similar to that presented below:

```
// Query database to find a record containing that username and password
$results = $conn->query("SQL query to find a matching record");

// Try to fetch the row. If there is no matching row, fetch() will return false.

$row = $results->fetch();
if($row== false)
{
        // There were no matching rows
}
else
{
        // There were matching rows
}
```

Use an if/else statement like the one above, together with the authentication example in the lecture notes, to add code to your login script which:

- displays an error message "Invalid login!" if the user entered an incorrect username and password, or

- saves the username in a “gatekeeper” session variable, and redirects the user to the main page of the site (index.php) if the username and password were correct.

3. Test your login script.

Part 2 – Preventing people bypassing the login system

4. Prevent users being able to access the main index page (the page with the form allowing the user to search by artist) without logging in. To do this, add code to check for the existence of the “gatekeeper” session variable at the top of the index page, like the example in the lecture notes. Since you are adding PHP to the HTML page, you will need to re-save it as a PHP file. Call it “index.php”.

5.Test out your “gatekeeper” protection by entering the address of the search page directly into your browser.

Part 3 - Using the “gatekeeper” session variable to determine the identity of the logged-in user

6. Write a message on the main index page which informs the user who they are logged in as, e.g.

```
Logged in as shiggins
```

7.Add similar protection to the searchresults.php and download.php script, to prevent non-logged-in users being able to buy music.

Further questions

8. Once you've done this, alter your index.php so that it shows the balance of the current user, e.g.

```
Logged in as shiggins, your balance is 3.79
```

9. Also alter your download.php script so that the currently logged in user’s bank balance is reduced by 0.79.

If you’re not sure how to do this, look at the existing SQL statement in the download.php script. It’s doing something similar.

10. If you get that done:
a) prevent the user being able to buy the hit if they don't have enough money in their account and
b) reduce the user's bank balance by the actual price of the hit, rather than 0.79. (Assuming you're reading this in Week 5, there is now a "`price`" column in your `wadsongs` table).

Advanced questions

11. Think about how the “gatekeeper” system could be extended to distinguish between ordinary users and administrators. Implement your chosen system, ensuring that the “update hit details” page (Worksheet 3 advanced question) is only accessible to administrators.

12. At the moment the entire site is only accessible to logged in users. Is this ideal? Think about how a site such as HitTastic! might operate in the “real” world and re-code your pages accordingly.