DFTI Worksheet 3
Connecting to databases from PHP

This week you will write a PHP script to connect to the HitTastic! database, and search the database to find all tracks by the artist that the user entered on the form. You should use the example on the lecture notes as your main guidance.

You will also write a signup script to add a user to the database.

The most important section to complete is the Standard Questions, which allow you to perform basic database searches and insert with PHP. You must finish these; if you cannot in class, you should complete in your own time and ask us for help.

If you are aiming for a higher mark in the unit, I would recommend also trying the further questions, and for a very high mark, the advanced questions: again you should finish in your own time if necessary.

SQL Worksheet

If you have not done SQL before, I would recommend doing the SQL worksheet (available on the website) first. If you have, or if you don't feel the need to practise with SQL, skip this and begin the main worksheet below.

Standard Questions

1) I have setup music databases on Neptune for all of you. You can view these by logging into PHPMyAdmin on Neptune at

http://ephp.solent.ac.uk/phpmyadmin/

Use your EPHP username and password to login. The table is called "wadsongs".

2) Load Notepad++ and open your searchresults.php from the previous exercise (Worksheet 3). Add this code to the PHP script to connect to the database.

```
$conn = new PDO ("mysql:host=localhost;dbname=ephpXXX;", "Neptune
login", "Neptune password");
```

Use your Neptune login and password to connect.

3) Now add a line of PHP code to your searchresults.php script to do an SQL query to search for all songs by the user's chosen **artist** in the database. See the example in the lecture notes for how to do this.

4) Again using the lecture notes to help you, add a **while** loop to go through all the hits returned from the database and echo the **song title**, **artist**, **year** and **genre** of each back to the browser.

5) Upload your script using FileZilla or EPHP.

6) Test it in the simulator in EPHP. Enter the URL of your index, wait for the form to be sent back, enter an artist (choose a well-known artist as the database only stores number 1 hits) <u>and watch carefully what happens on the server</u>. Watch how each line of code is stepped through and try and relate this to each line of the database results being highlighted in turn.

INSERT statements

Now you are going to write a script which use a different type of query: the INSERT statement.

7) Login to PHPMyAdmin on Neptune (http://ephp.solent.ac.uk/phpmyadmin) **<u>with your Neptune username and password</u>**.

8) Run Notepad++ and open your signup.php script from last time. Modify it so that it adds a new record into the ht_users table, containing the details that the user entered in the form. You will need to:

- Connect to the database (use your username and password)
- Send an INSERT statement to the database to add a new record;
- Display some sort of confirmation message.

When testing, **<u>do not use your real date of birth, date of birth of family members, or anything resembling your real date of birth (e.g. right day, wrong year)</u>** (for security reasons – e.g. identity theft). The databases on Neptune are not secure from other students.

If you are not sure how to do this, look in the **<u>SQL revision</u>** lecture notes for an example of an SQL INSERT statement.
If you do not have a signup.php script from last time, start one from scratch now.

Remember to access files uploaded, you go to the URL

http://ephp.solent.ac.uk/~yourusername/filename

Use PHPMyAdmin to test whether it worked (i.e. whether it inserted the user).

Further questions

1) If the user didn't enter anything in the form, display an error to the user. Only perform the database search if the user entered something. Hint: test for a blank field ("")

2) Modify your code so that an error ("Sorry, no results" or similar) is shown if there are no results from the database search. See the example in the lecture notes to get started with this.

3) Add a new way to search for music on your site, one which allows the user to enter a search term and then to choose (by means of a select box) what to search on: title, artist or year. Add a **new** HTML form to your index page as an entirely separate HTML form (do not remove your old form!) and create an entirely separate, new PHP script to do the search.

UPDATE statement

You will all get the chance to practice with UPDATE statements next week. However, if you have completed the core exercises above, you can start to explore it now.

4) Create a new file. Copy and paste this code into it. It is an HTML form to allow the site administrator to update the password of a user. Save it as **updatepass.html**.

```
<html>
<head>
<title>Change password</title>
</head>
<body>
<h1>Update a password</h1>
<form method="post" action="fill this in">
Username: <br />
```

```
<input name="username" /><br/>
New password:
<br />
<input name="newpassword" type="password" /><br />
<input type="submit" value="go"/>
</form>
</body>
</html>
```

If you are not sure how to do an UPDATE query, look in the Databases and SQL lecture notes.

5) Write a PHP script called **updatepass.php** which reads in the username and the new password from the form and updates the password to the value supplied. It should use an UPDATE query.

Upload to the server using FileZilla and test.

6) Add error-checking to the update hit script, so that an error message is displayed if the username is invalid (there is no user with that username in the database). **Hint**: use a SELECT statement to search the database for a user with that username, and then test whether any results were returned. See the example in the SQL revision notes which tests if any results were returned from a query.

Advanced Question – Change Song Details

This question should only be done if you are thoroughly comfortable with everything you've done so far.

7) Below is a third form, which allows the administrator to update the details of an existing song (chart position and price) by ID. (This might be used in case of errors in the database, e.g. typos)

```
<html>
<head>
<title>Change details of existing song</title>
</head>
<body>
<h1>Change details of an existing song</h1>
<form method="post" action="changedetails1.php">
<label for="id">ID of song: </label> <br />
<input name="id" /><br/>
<input type="submit" value="Go!" />
</body>
</html>
```

When the administrator enters a song ID, another form should come up (you'll write this in Question 8 below), with the fields filled with the details of that song The administrator can then change one or more fields and then submit that form.

8) Write a new script, ***changedetails1.php***, to respond to the first form and auto-generate the second form with the chart position and price fields filled in. The script should read in the ID from the form, retrieve the details of the song with that ID from the database, and display (using echo statements) **a new form** with fields for chart position and price. Each form field should be filled in with the current value (retrieved from the database) for that song. The form should also contain a field for ID, but this should be a **hidden field** so that the user cannot change the ID. A hidden field can be done HTML similar to that below:

```
     <input type='hidden' name='id' value='the song ID' />
```

The user will then be able to alter the values in the fields and hit Go.

Not sure how to do this? Remember that in PHP you can echo out **any** HTML tags. So write down, in your logbook, the HTML for each form field required, and then work out how to echo out that field from PHP.

9) You should then write a **further** script, ***changedetails2.php***, which actually does the update. To update several values at once use the multi-field-update version of the SQL UPDATE statement:

```
UPDATE table SET column1='value1', column2='value2' WHERE
column3='value3'
```