DFTI Worksheet 2
Introduction to PHP

This lab exercise follows on from Week 1. This week you will write PHP scripts to read in the artist that the user entered in the HTML form from Week 1, and display it back to the user.

You must write the HTML forms from Worksheet 1 if you have not done so already.

This week you should make sure that you have done questions 1 to 5. If you do not complete them in class, finish in your own time. <u>These questions are essential to complete if you want to pass the unit</u>.

I would also strongly recommend completing Further Questions 6 to 8, in your own time if necessary.

If you are confident answering questions 1 to 8, you should attempt some or all of the advanced exercises from 9 to 13, covering arrays and associative arrays, in your own time if necessary.

<u>As I mentioned last time, you should keep a logbook for this unit. This can just be a Word document or similar. This is not assessed but is for your own benefit; you will find it useful for revision later.</u>

Core questions - essential to complete

Part I - A simple PHP script

1. In Notepad++, or in EPHP's own editor, write a PHP script to read the artist that the user entered into a variable, **$a**, and display a message which tells the user what artist they entered. Ensure that is saved as **searchresults.php**.

   So, if the user entered Oasis for the artist, the script should display:

```
You are searching for songs by Oasis
```

2. Upload your PHP to the server using FileZilla or EPHP.

3. **<u>When most people have got to this stage, we will demonstrate using EPHP to request and run PHP scripts.</u>**

   Using this demo to help you, test it in EPHP as follows.

   i) First enter the URL of your form <u>in the "Simulation" tab</u>:

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

Watch the HTTP animation as you did last time.

ii) When the form returns to the EPHP browser, try typing in an artist in the form within the EPHP browser. What happens?

iii) Correct your HTML form to fix the problem in ii). Upload the modified form using FileZilla.

iv) Type in an artist again, watch the animation, and watch what happens on the server carefully. Your PHP code will appear on the server once the request is received.

<u>If there were no errors in your PHP</u>, your PHP code will appear. EPHP shows a list of variables in the 'Variables' window of the bottom of the screen. Does $a contain the text you entered in the form? If not, try and figure out why not by looking at your PHP and looking at the "Develop" window on the client side, which should contain your HTML form.

<u>If a "syntax error" message is displayed in a popup window</u>, there's an error in your PHP code which prevents it running. Look at your code again and try to figure out what was wrong. Correct the error, re-upload, and start question 4 again

v) Click on "Run PHP and send back output" to send the HTTP response to the client. Answer these questions in your logbook:
- What happens in the EPHP browser?
- When you roll over the HTTP response, do you see PHP code? If not, what do you see instead? Why do you not see PHP? (answer in your logbook)

4. Once you've got it working in EPHP, try requesting it in the browser outside EPHP. In a new browser tab:
   - Enter the URL of your form (http://ephp.solent.ac.uk/~ephpXXX/index.html)
   - Enter an artist.
   - Your PHP should output the expected result.

5. Ensure you've completed and uploaded the signup form from Topic 1. <u>The form MUST have a method of "post" as signing up is an action which will update the database,and by convention, the "post" method is used whenever updates are being done</u>.
   Then, in Notepad++ or in EPHP, write the signup script to respond to the signup form. This should simply read in the details that the user entered in the form and display them back to the user, e.g.:

You have signed up with: Name Tim Smith, username ts282, Date of birth 2 June 1970

It should be saved as <u>signup.php</u>.

Test the signup script within EPHP again by entering the URL of your signup form, entering details, and testing whether the output is what you expect. Again, if a syntax error is reported, try and figure out why, and if the variables reported by EPHP are not what you expect, again try and figure out why.

Further exercises - also important to complete

6. You'll probably notice that the PHP signup script writes everything out on one line. Change the code so it writes out the message on separate lines, like this:

```
You have signed up with:
Name: Tim Smith
Username: ts282
Date of birth: 2 June 1970
```

7. Add code to your PHP signup script to check that the year is valid. If the year is before 1890 (no-one alive today was born before 1890), write a message telling the user that the year is invalid.

8. Apply the CSS stylesheet you wrote last week to your two PHP scripts. You will need to mix HTML and PHP for this!

Loops, Arrays and Associative Arrays

9. Alter your signup form so that it's dynamically created using PHP, using loops and arrays. (You'll need to rename it from 'signup.html' to 'signup_form.php').

You are going to change the day, month and year fields so that each is a <select> box, containing appropriate values, rather than a simple input box - and generate each <select> box dynamically using PHP.

Remember that the syntax of an HTML select box is:

```
<select name='day'>
<option>1</option>
<option>2</option>
... etc...
<select>
```

a) Use a for loop within PHP to dynamically create a select box for the day of birth, containing the numbers 1-31 as its options.

b) Create an array containing the months of the year, January through to December. Use a foreach loop to loop through this array and create another select box for the month of birth, with an <option> for each month of the year.

c) Use getdate() (see the lecture notes) and a for loop, to create a select box for the year of birth. This should contain every year from 1890 to the current year minus 18, so that it works no matter what the current year is.

10. Write a new HitTastic! "about" page, about.php (why .php and not .html? - You are going to add some PHP to it shortly).

This about page should contain some information about the HitTastic! company. If you can't think of anything, use this:

HitTastic! was founded in 1998 by Alex Martin in a garden shed in rural Northern California in order to provide an online search facility for music. It was one of the first sites to use the newly-released PHP 3.0. The site was highly successful and by linking up with record companies, expanded to the search and download site that you see today. HitTastic! now has its own premises in Silicon Valley, and can be contacted at:

HitTastic! Inc,
One HitTastic! Way,
Sunnyvale, California.

You are going to use a PHP associative array to generate three hyperlinks, to the index page, the signup form and the about page.

Resave your index page as index.php, and write some PHP code to create an associative array containing the hyperlink text and URL. The associative array should be as follows:

Key	Value
Index	index.php
Signup	signup_form.php
About	about.php

Use a foreach loop to loop through this associative array and generate the three hyperlinks on index.php. Use the key for the hyperlink text, and the value for the URL.

Repeat this code on signup_form.php and about.php, so that each page is linked to each other.

11. Preview of functions - preventing code repetition

Obviously it's rather inefficient to repeat this code on three pages. Wouldn't it be better to write it just once and use it in all three pages?

You can do this with PHP functions. We will look at these later, but here's a quick introduction. A function is a reusable block of code which can be called at any point from your code. In PHP, it's common practice to put your functions in an external include file and include them into each script which needs them. Here is an example of a function:

```
function hello()
{
     echo "Hello world!";
}
```

To use this function:

- place the function in a separate PHP file, functions.php;
- include functions.php in each PHP script which needs it with the include() statement, e.g.

```
     include("functions.php");
```

- call the function at the point in your PHP code where it is needed, e.g.:

```
     hello();
```

Exercise: write the code to generate your hyperlinks in its own function. Save it in an external PHP file, include it in each of your three pages (index, signup form and about), and call the function from your PHP code at the point where you want to display the hyperlinks.

If you get that done, try writing a second function, writeCSSLink(), to generate your link to your CSS stylesheet. This function should take a parameter of the CSS filename (ask your tutor for help on parameters if need be), and write out the

```
<link rel='stylesheet' ... />
```

line so that it contains the parameter passed in.

Change each of your PHP scripts so that this function is called to write out your CSS link, rather than placing the <link> tag directly in your file.

Further advanced exercise – Seasonal stylesheets

We can get the current month using getdate(), in a similar way to getting the current year. For example:

```
$d = getdate();
$m = $d["mon"];
```

will store the month number in the variable $m.

12. In your index.php, write some PHP to display “Happy New Year!” if it's January, “Happy Pancake Day/Mardi Gras” (whichever name is most appropriate to you) if it’s February, or “Happy Easter!” if it's March or April.

13. Create two CSS files, winter.css and spring.css. Set winter.css to a look and feel suitable for winter e.g. combinations of dark blue and white; set spring.css to a look and feel suitable for spring, e.g. combinations of green, blue and yellow.

Add some PHP to the search page and search results page so that it imports the appropriate stylesheet depending on the season (if it's January or February, import the winter stylesheet, if it’s March or April, import the spring stylesheet). Test it out now, in the browser. If you can remember, test it out again in March.