Databases and the Web – Exercise 4

This week we'll begin to develop some pages that pass information and act upon that information.

Following ex's 1–3 you should know how to back-up and recreate a database using phpMyAdmin & understand the structure and relationships between the three **world** tables.

From now on you'll be creating PHP pages on your PCs for use on StudentNet (or on your own machine with a `localhost`).

Task 1: Practice with PHP & a little debugging ;-)

1) Create a page on the PC that includes an HTML form, say "week4form.html", containing a submit button & a couple of named elements:

- *E.g.* a text field:
  ```
  <input type="text" name="textField" />
  ```
- and a pair of check-boxes, *e.g.*
  ```
  Yes
  <input type="checkbox" name="checkbox" value="yes" />
  <br />
  No
  <input type="checkbox" name="checkbox" value="no" />
  ```

2) Make the form's **action** attribute point to another page, say "week4form.php".

3) Create the "week4form.php" page in the same directory. Within it:

   a) Write out the contents of the **$_POST** and **$_GET** arrays using the **print_r** PHP function – look it up! (http://uk.php.net/manual/en/function.print-r.php)

   - If you wrap the function inside an HTML **<pre>** tag you can guarantee that the output displays as-is on the web page (otherwise HTML merges white-space.)

   b) Beneath that (on the same page)

   - connect to your StudentNet database,
   - execute a "**SELECT *** " query on *e.g.* the **country** table (use **LIMIT** to return just the 1st row),
   - extract the 1st row into an array using one of

     (1) **mysqli_fetch_row**

     (2) **mysqli_fetch_assoc**

     (3) **mysqli_fetch_object**

     (or their **PDO** or **mysql_** equivalents) and display the data using **print_r**.

4) Browse to the HTML page, experiment with the form settings and view the result in the PHP page…

This should (a) show you how useful "**print_r**" and **<pre>** are for examining the contents of complex datatypes and (b) hint at what happens when data are passed from a form to PHP … more on that next week!

Databases and the Web – Exercise 4

Task 2: Listing countries by continent

This exercise is a simple example of the use of XHTML **`form`** elements and PHP's database connectivity functions. In this exercise you will create a web page on which you can select a continent from a list and then press a button to be taken to a page that lists all the countries in that continent.

An example page is here:
http://staffnet.kingston.ac.uk/~ku13043/WebDB/ex/week04/ex1a.php

1. Create a PHP page to list the Continents from your world database as follows:

   a) Create a valid HTML4/5 or XHTML page in your **`"public_html"`** directory (on SUSE) or "**`www`**" on StudentNet.

      (i) Name it **`week4task1_1.php`**

      (ii) Use an appropriate DOCTYPE.

      (iii) Ensure it has it an appropriate **`<title>`** and a level 1 heading that describes its purpose.

   b) Inside that page add the PHP necessary to:

      (i) Connect to the MySQL DBMS.

      (ii) Use *your* database.

      (iii) Query the database for a list of unique continent names in alphabetical order.

   c) Add a **`form`** to the page that uses the GET **`method`** and whose **`action`** attribute points to the next file you will create (in step 2 below) that processes the form submission.

      - *E.g.* **`week4task1_2.php`**

   d) Inside the **`form`** write PHP that:

      (i) Writes a **radio button** into the page for each continent by combining the following within a **`for`** or **`while`** loop:

      - To *group* the radio buttons they all need a **`name`** attribute,
        e.g. **`<input type="radio" name="Continent" />`**
      - To associate the continent name with the button it's not sufficient to only have a **`<label>`**. Each **`radio`** button needs its **`value`** attribute set to the continent name,
        e.g. **`<input type="radio" value="Asia" />`**
      - To *label* the radio buttons so that the label works in Internet Explorer they each need a **unique `id`** attribute,
        e.g. **`<input type="radio" id="c0" />`** goes with
        `<label for="c0">`

(ii) For each continent, write into the page a continent name using **<label>** tags and the unique **id** above to associate the label with the appropriate button.

(iii) Finally add a submit button, *e.g.*

- **<input type="submit" value="View countries" name="submit" />** ensures that the **$_GET** array contains an entry like **('submit' => 'View countries')**

Verify that it works as expected using a modern browser & IE ;-)

- Do you get a list of buttons and continents?
- Can you click on the **<label>** next to the radio button to select it?
- If you press "submit", does it go somewhere?

2. Create the *destination* for the previous page's **form** (the file listed in its **action** attribute) as follows:

a) Again, create a valid page with a title and heading.

(i) Add the continent name to the heading as in the example.

b) Write HTML that opens an unordered list.

c) Write PHP within the page that:

(i) Connects to your MySQL DBMS on StudentNet.

(ii) Uses *your* database.

(iii) Queries the database using the supplied continent name and retrieves all of the country names for that continent

- *E.g.* **$_GET['Continent']**
- Retrieve *only* the country name.
- **NB** *copy/paste* is your friend! (**Ctrl-Ins/Shift-Ins** or **Ctrl-C/Ctrl-V**)

(iv) Write each retrieved name as a list item **<li>**.

d) Write HTML that closes the unordered list.

- *E.g.* you end up with something like:

```
<ul>
<?php
      while ($arr=mysqli_fetch_row…) {
            /* …code… */
      }
?>
</ul>
```

- Can you arrange it so that the list items are indented by one tab and appear on separate lines in the HTML source?
  (It makes the HTML easier to read…)

e) For debugging purposes you might like to add the following code to the bottom of your HTML page:

```
<h2>Debugging: See what $_GET contains</h2>
<pre><?php print_r($_GET); ?></pre>
```

This dumps the contents of the $_GET array into a preformatted HTML block so you can easily see what your form is sending to the page…

You could easily adapt this two page methodology to a single page that contains the form *and* the code to process it, using a branch like

```
if ($_GET['submit']!='View countries') {
        /* HTML FORM stuff */
} else {
        /* PHP FORM processing stuff */
}
```

3. Try it!
   a) Save a copy of your first file as *e.g.* **week4task1_3.php**
   b) Add the **if** test to the appropriate place.
   c) Modify the form so its **action** attribute points back to the same page using **PHP_SELF** (see the week 4 lecture slides.)
   d) Copy/paste the code from **week4task1_2.php** inside the **else** clause.
   e) Verify it works – when you submit the form the same base URL should remain in the browser location bar with the GET request query string appended.
   f) Finally change **week4task1_3.php** so that it uses POST instead of GET.
      - Now when you submit the form there should be no query string. Bookmarks will not work but the data are ever-so slightly more secure…
4. Back-up your work!

Task 3: Listing countries and language by continent

This exercise builds on "Task 2" and the queries you have already written to list the countries and their languages for a given continent. An example page is here: http://staffnet.kingston.ac.uk/~ku13043/WebDB/ex/week04/ex2.php

1. Take a copy of your **week4task1_3.php** and name it **week4task2.php**
2. Modify the query so that it returns the joined contents of **Country** and **CountryLanguage** for the chosen continent.
   a) This is similar to a query from exercise 3.

b) Arrange the query so that the data are ordered by country name, the official/unofficial status of the language and then the language name.

3. Initially (for simplicity's sake!) list the data in a single list like:

   - Algeria, Arabic, T
   - Algeria, Berberi, F *etc*. (for Africa.)

4. Once that's working for every continent, including a sensible error message for Antarctica ☺, can you arrange it so that:

   a) Each country appears as a separate heading or unordered/definition list item, inside of which is a (nested) list of spoken languages?

   b) The official language(s) are highlighted in some way?

      - You could use a neat CSS trick here so that, for example, "Gibraltar: English, Arabic" is *marked-up* something like this:

```
<li>Gibraltar
          <ul>
               <li class="T">English</li>
               <li>Arabic</li>
          </ul>
</li>
```

With CSS rules like:

```
li li {font-weight:normal;}
li li.T {font-weight:bold;}
```

Extra time?

If you have extra time and want to occupy your mind ;-) you could:

1. Experiment by using a **<select>** drop-down list instead of the radio boxes.
2. Allow for multiple check boxes or selections so that you might list countries from more than one continent.