University of Stirling
Department of Computing Science
PHP – Server Side Scripting

Practical 3 – Databases, JavaScript and Cookies

You will find it helpful to have your lecture notes to hand.

1. Log-on in the usual way. The webserver to be used in the practicals is `shark.cs.stir.ac.uk`. All files you are using will have to be on `shark`. To access your file space on shark, you will need to map a network drive. In Windows Explorer, select the `Map Network Drive` option from the `Tools` menu. Select a drive and link this to the folder: `\\shark\www\`{your_username}. Navigate to the newly created drive and create a directory `ITNP70`.
2. Navigate to `Groups on Wide` and find the folder `ITNP70`. Inside that folder you will find another folder called `Prac3`. Copy that folder into the `ITNP70` folder you created in the previous step.
3. Close the window on the `ITNP70` Group folder, so you do not confuse its contents with your own copy.

Databases

4. Change your MySQL database password at http://www.cs.stir.ac.uk/cgi-bin/sam/dept/cmpf.pl. (your default password is the same as your username).
5. Open Textpad. Open the file `dbsetup.php` into the Textpad window. This file is the management PHP file as mentioned in the lectures. It sets up the MySQL database. Make sure you understand the code by going through the program line by line. You also may want to have a look at `dbinitial.sql` which is loaded by `dbsetup.php`. Make sure you understand the loading and use of the user credentials (in dbcredentials.php).
6. Change the Data Source Name in the `connect()` function. You need to change xxx to your own student domain username (three times: username, password, database name). Try and execute the file on `shark` (using a web browser). You should direct the browser to: `shark.cs.stir.ac.uk/`{user_name}`/ITNP70/Prac3/dbsetup.php`. Change the program so that you can follow in the browser which SQL command is being executed.
7. Design a webpage (with PHP code) which connects to your database and which uses the tables created in step 5. The webpage should display all courses with the course id and course name together with the corresponding course coordinator with their names, email addresses and office numbers. Before you start coding, it is a good idea to think about the structure of your webpage with the PHP. What needs to happen in what order? What PHP functions do you need to call? What should be the SQL query to get the data? You will find parts of this code in your lecture notes.

8. Change your program to make it interactive. The user should be able to specify a course unit (e.g. ITNP70) and then the information related only to this course unit is displayed. Alternatively the user may specify 'All'. This should produce the same result as in step 6. To do this you will need an HTML form (with 1 text field and submit button) and process the passed information. You have done something similar already in an earlier practical.

Checkpoint

Client-side Scripting

9. Your `Prac3` folder has a subfolder `jscript`. This contains seven files which you should try out in a browser. In each case, inspect the contents of the files, and "View the page source" in the browser. Make sure you understand how they correspond.
10. Note that `js3.html` and `js4.html` involve the use of JavaScript functions, called after a `buttonclick` event, causing output to be displayed in an INPUT field on the form.
11. Think about `js3.html`. Another way of implementing such a count might be to have a (JavaScript) global variable declared in the header before the increment function. Do this:

    ```
    var count = 0;
    ```

    and adapt the increment function so that it increments this variable, then assign the new value to the form element for output. Now try it out.
12. Try `js5.html` with various kinds of input, to see that it accepts only numbers ...
13. ... Hold on ...! It seems to think that "47qxts" is a number. (This is what the function `parseInt` does - if the string has a prefix which constitutes a number, the rest is ignored, and if it does not have such a prefix, the value returned is NaN (not a number).)
14. Extend `js6.html` so that it includes another button, which says "CSG", and, when clicked, makes "4B81" show in the text field.
15. Note that `js7.html` may transmit multiple values for a single form field to the PHP script. PHP needs to recognise this to be able to pick up all values and not just one. You signal this case to PHP buy ending the name of the form field with `[]`. Experiment with removing `[]`.
16. Your `Prac3` folder has a subfolder called `initials`. Point your browser to your `shark` copy of `initialform.html`.
17. You should find that you get a form that will tell you the title and initials of students, if you enter the correct registration number. It works like this: when you click the "Click to submit" button, `initialform.html` sends your query to the serverside PHP script `initials.php`, which consults the text file `initials.txt` and then outputs the answer, wrapped in HTML. That output comes back to your browser (i.e. to the client side), which displays it.
18. You will perhaps have noticed that there is a significant difference between the Rooms (`js5.html`) and Initials examples: the Initials page includes some clientside verification. If the user submits a blank registration number, or submits with the field still saying "Enter registration number", the data is not submitted and an error message is shown instead. Check that this validation works as described in the previous paragraph.

19. The validation is rather weak, however. It will still allow invalid registration numbers, such as "grub" (not a number at all) or "73" (too short). As with `js5.html` above, we would like to check that the user has entered a number, and we want to check that it is in a certain range. Modify your `initialform.html` so that it checks that it is a number which has been entered. You will recall that `parseInt` is rather permissive. Try using the (builtin) function `Number` instead:

    ```
    var n = Number(regno);
    ```

    You can look up the builtin functions in "PART 1. Core Language Features" of `http://www.cs.stir.ac.uk/doc/jscript/ClientGuideJS13`, and navigate through links "Functions", and "Predefined Functions".)

20. Extend the validation so only numbers greater than 1200000 and less than 1400000 are allowed.

Use of Cookies

21. Your `Prac3` folder has a further subfolder called `cookies`. Try and run the example and understand who this is working. Here is what happens (take note of the URLs that appear in the browser's "Location" field each time, and you'll see the PHP aspect of this working):

    a. The example starts at `start.php`. You see a "Welcome to Trader Joe's" initial page. It includes an invitation to click a link to sign in.

    b. This leads to a "sign-in" page (`sign.html`). Sign in using any name you like (but don't include any nonalphabetic characters).

    c. Click on `Submit`, and you will see an order form for cakes. Enter the number of cakes you wish to order, and click to submit your order. You see an acknowledgement of your order.

    d. Now we come to the interesting bit. Use the `Back` button several times to return to the `start.php` page, and click the `Refresh` button on the browser.

    e. This time you get a different first page! Your previous visit has been remembered. Who you are has been remembered through the use of a cookie. If the cookie exists, the script `start.php` accesses it, and constructs the "welcome back" page, and otherwise (as on the first occasion) the script simply constructs a page containing a link to the "sign-in" page.

    f. At this point, if you are the same person as before, you can go straight to the "order cakes" page. But if you are a different person (who just happens to be accessing Trader Joe's from the same place) then you can click to go to the "sign-in" page.

    The "sign-in" page sends the name of the user to the script `order.php` by normal HTML procedure, using a GET request. `Order.php` accesses the user's name, and incorporates it into the cookie, which is then sent back with the constructed HTML page. The cookie is then stored on the user's machine. Next time the user wants to order cakes, the cookie will be sent back to the server, and the script `start.php` will use its value to identify the user. The final aspect: the order form contains a "hidden field", to contain the user's name, so that this is returned to the server when the order is submitted, and is accessed by the script (`goodbye.php`) which produces the (personalised) acknowledgement.