Distributed Programming I (03MQPOV)

Laboratory exercise n.6

Objective: writing dynamic pages in PHP, using the features illustrated in the course. A PHP manual is available to the URL http://php.net.

In this exercise the XAMPP suite will be used. It is already installed in the laboratory machines in the Windows OS. After launching XAMPP, activate Apache from the control panel. The folder in which the files will need to be placed is **c:\xampp\htdocs**. **Create a sub-folder** and work just inside that one to avoid destroying xampp content (like its own links, etc).

In order to write PHP pages the files in which the code is stored must be terminated with the ".php" extension, otherwise the pages will not be recognized by the Apache web server.

Moreover, to verify the proper functioning of the PHP pages, it is mandatory to access them by using the web server address (for example: http://localhost/myfolder/page.php) and not by using their local physical path.

To each exercise it is suggested to alternate the POST and the GET methods to send form data (the last one is particularly useful to debug the pages because it shows the data in the URL).

To debug the proper working of PHP pages it is suggested to use the "Eclipse for PHP Developers" software, which is already installed in the lab machines. Set as workspace the folder C:\xampp\htdocs, so that the created projects will be available also to the Apache web sever. To create a new project use: File/New Project/PHP, set a folder name and continue by pressing OK. Through File/New/PHP file, create a new file, as example containing <?php phpinfo(); ?> and execute it through Run/Run as/PHP Web page and by confirming the proposed URL. This configuration will allow to set breakpoints to debug the developed PHP code.

It is also possible to analyse the contents of the DB used by MySQL by means of the web interface, accessible at the URL http://localhost/**phpmyadmin**.

For the exercises that require the use of a database, you are advised to first create a "sample" database by means of the web application phpMyAdmin. In this database you should import the .sql file given with the lab material, which inserts some books into the books table (with fields id, Title, Genre, SubGenre, Price, Publisher). You can add columns and/or tables to the sample database in order to store other information, such as users, passwords, shopping cart, etc.

In order to connect to the MySQL DB, you can use user='root' and pass=''.

Note: just in order to simplify debugging, it is advised to use the .php file extension also for the files just containing HTML code: in this way it is possible to set these pages as

starting points for a debug session. In “Debug Configurations”, verify that “Xdebug” is set as “Server Debugger”. It is also possible to set the flag “Break at first line”, if desired.

Note: the files in c:\xampp\htdocs are visible to all the users of the computer, independently of the username they used for login. Do NOT leave personal files or files that you need to protect, such as the solutions of your exam web assignment!

Exercise 6.1.1

Create a page A showing the text “Italia!” which is able to set a cookie with the name “Country” and the value “IT”.

Create then a page B that reads the “Country” cookie value and shows it.

In order to verify the correct behavior of the pages, perform the following operations:

- visit first page A and then page B
- close the browser and re-start it
- visit first page B, then page A and finally page B again

Exercise 6.1.2

Create a PHP page showing a table containing all the cookies the web server is able to read from the browser while the page is visited. In particular, the first column of the table must show the cookie names, while the second one must contain their values.
Finally, compare the page result to what is possible to obtain by showing the cookie list directly from the browser menu (e.g. in Firefox: tools/options/privacy/cookies).

Exercise 6.1.3

Modify the page A of exercise 6.1.1 by setting as expiring date of the cookie the 31th December 2010 and then repeat the visits of pages A and B as described in exercise 6.1.1.

Exercise 6.1.4

Create an HTML page containing a form with two text fields, “name” and “surname”, and two buttons, respectively for resetting the form and for submitting it.

The form must send data to a PHP page containing a brief description of a topic you like and, for each page access next to the first one, also a custom greeting like, for example:

> *Welcome back, dear <name> <surname>, to my humble site.*

Exercise 6.1.5

Create a page (ONE) with just one button labelled “Enter”, which gives the user access to another page (TWO) containing the text of a joke and a button labelled “Exit” that lets the user go back to the first page.

The access of the page TWO must be allowed only if the user has not yet requested to view the page more than twice. A request to view is considered as issued when the “Enter” button has been pressed, and the reading is considered as finished when the “Exit” button has been pressed.

Note: to simulate multiple accesses to the page it is enough to open multiple browser tabs or windows.

Exercise 6.1.6

Develop a PHP web site that allows the user to “buy” a series of products. The website must be composed of several pages:

- The first page must list the available products, and for each one it must contain an input field allowing the user to specify the quantity of items the user wants to buy.
- By clicking on a product name it is possible to reach another page with a brief description of the product and the price of that product.
- Every page must contain a link to a “summary” page, representing the “shopping basket” of the products acquired until that moment, that shows the products that the user plans to buy along with the selected quantity. This page must contain a link to the starting page, a “Buy” button to confirm the order, and it must be possible to modify the quantities written in the product list and save the new quantities by pressing the “Update” button.
- By pressing the “Buy” button a summary (read-only) page must be shown, containing the selected products, for each selected product, the total price (product price * quantity) and the total price of the order. Finally, this page must show a button that confirms the order and that concludes the buying process.

Exercise 6.2

Create a page A containing a form to send the following data to URL B:

- a name, composed of at most 10 alphabetical characters, in which the first one must be upper case and the other ones must be lower case.
- an age (a numerical value ranging from 0 to 199)
- a phone number, composed of a prefix (two or three digits, in which the first one must be zero) followed by the character “-” and then by a number (six or seven digits) that may contain another character “-” after the first three digits.

In case one of the data items does not respect its specification, an error must be signalled (client side, if javascript is enabled, or server side otherwise). In order to disable Javascript in the browser, go to the browser settings and look for “cookie”, then go to the menu item “Enable/Disable Javascript”.

Page B must show the entered data, if correct, otherwise it must show an error message and propose a link to go back to page A.

Exercise 6.3.1

Create a page that queries the database and shows all the inserted values, by filling up a table composed of three columns:

- the first column must contain the product name
- the second column must contain the available quantity of the product
- the third column must contain the genre of the product
- the fourth column must contain the product price

Exercise 6.3.2

Repeat the previous exercise by just showing **10 records a time**. Insert a button that allows loading the next block of data. The current page number must NOT be passed as a form parameter. Please avoid loading, at each page view, all the data present in the DB.

Exercise 6.3.3

Create a page A listing the products present in the database (by showing ID, Description, Genre, and Price) and also containing a form to place a buy order, specifying the ID of the chosen product and the desired quantity.

Create then a page B that receives the form data and returns the total price for the selected product and the desired quantity.

Exercise 6.3.4

Starting from the solution of the previous exercise, update it so that page B also modifies the DB, by subtracting the bought quantity (if available) or by signalling an error (if the desired quantity exceeds the current product availability). The page must also contain a link or button to return to page A.

Extend then the page for placing a buy order so that it is possible to buy more than one product, each one with its quantity, in a single buy operation. In this case, when the buy request is performed, insert a new page that, after having presented the list of products that are going to be bought, each one with its quantity, requests a final confirmation from the user, by pressing a specific button. A click on the button will finally confirm the buy operation and will make the corresponding modifications to the database.

The confirmation page must also have another button that lets the user cancel the buy operation and that takes the user to the initial page.

Also, adding products to be bought must be possible only for users who have been authenticated by means of username and password. Usernames and passwords have to be stored in a table of the DB, with passwords not stored as cleartext but converted into digests by means of MD5 or SHA.