Distributed Programming (03NQVOC)
Distributed Programming I (03MQPOV)

Laboratory exercise n.6

Objective: writing dynamic pages in PHP. 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 environment. 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 (as 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.

Note: to simplify the debug, 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.

Exercise 6.1.1

Create an HTML page containing a form with a text input field and two buttons, Reset and Submit.

Develop then a PHP page that receives data from the form and shows a page having as first-level title "Text received" followed by a paragraph containing the value of the input text field.

Exercise 6.1.2

Create an HTML page containing a form with a drop-down menu allowing to choose an integer number N between 1 and 10

Develop then a PHP page that receives data from the form, verifies the correctness of the input and shows a table with the squares and the cubes of the numbers ranging from 1 to the value N introduced in the form.

Exercise 6.1.3

Create an HTML page containing a form allowing to insert name, surname and sports played (multiple choice between Basket, Football, Swimming, Ski and Volley).

Develop then a PHP page that receives the form data and produces a two-column table. In the first column all the form input field names must be listed, along with the server variable names (found in $_SERVER), while the second column must list their corresponding values.

Exercise 6.1.4

Create an HTML page containing:

- a form with two text inputs:
    - in the first one put the name (max length of 30 characters)
    - in the second one put the age (max value 120 years)
- two buttons to reset all the fields and to send data to the server.

Develop then a PHP that receives the form data and that - if the data respects the limits - replies with a confidential greeting by using the name of the user and by adding the suffix:

“young boy” if the age is < 10

“young friend” if 10 <= age < 30

“gentleman” if age >= 30

As example, by putting in the form the data “Dante” and “56”, the browser should show a reply like “Hello gentleman Dante, how are you today?”

Exercise 6.1.5

Create an HTML page containing a form (named “calculator”) with the following fields:

- two text inputs, admitting just numbers (called operand1 and operand2)
- the type of operation (a closed-choice value between +, -, *, /)

- two buttons, Reset and Submit.

Develop then a PHP page that indicates the performed operation and its result.

Exercise 6.2.1

Create a page A showing the text “Italia!” that 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 it shows it.

To verify the correct behavior of the pages perform the following operations:

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

Exercise 6.2.2

Create a PHP page showing a table containing all the cookie 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 6.2.3

Modify the page A of exercise 6.2.1 by setting as expiring date of the cookie the 31th december 2010 and then repeat visits of pages A and B as described in exercise 6.2.1.

Exercise 6.2.4

Create an HTML page containing a form with two text inputs, “name” and “surname”, and two buttons, Reset and Submit.

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

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

Exercise 6.2.5

Create a page (ONE) with just a button called “Enter” needed to access to another page (TWO) containing the text of a joke and a button called “Exit” that brings you back to the first page.

The access of the page TWO must be denied if the user is already reading that page for more than two times. A “reading session” is considered to be started when the “Enter” button is pressed, and is terminated when the “Exit” button is pressed.

Note: to simulate the multiple access to the page is enough to open multiple browser tabs.

Exercise 6.2.6

Develop a PHP page containing a series of products organized in a table in which:

- the first column must contain the product name (CD, DVD, SD memory, USB memory);
- the second one must contain a brief product description;
- the third one must contain the product price;
- the fourth one must contain some input text fields used to specify the product quantities.

By pressing a button it must be possible to access to another page showing the total quantities and the total price of the selected products.

By accessing again to the starting page, the fourth column must already contain the quenatities selected the last time.

Note: to ensure the proper functioning of the pages, do not use the “back” button but re-load the starting page by writing the URL or by pressing the Reload button.

Exercise 6.2.7

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

- The first page must list the available products, and for each one it must contain an input field allowing to specify the quantity of products the user wants to buy.
- By clicking to 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 it concludes the buying process.