Polytechnic University of Cartagena

Higher Technical School of Telecommunications Engineering

APPLICATIONS ON THE INTERNET

Lab 4: COOKIES AND DATABASES WITH PHP

Teachers:

Esteban Egea Lopez
Juan José Alcaraz Espín

1. Goals

– Understand how cookies and use with PHP.
– Understand the basics of SQL and use with PHP.

2. Lab work

You will work with cookies and database data using PHP scripts. To do this, you have a MySQL/MariaDB database server installed the server.
In this lab we will start practicing with cookies and then make queries to the database and show the results.

1) **Implement a visit counter in PHP, which shows the number of visits for each user**.
   - The counter uses a cookie called *visit* whose value is the number of visits that the user has made to the web. The script will read the value of the cookie (if it exists), display it on the screen, and send the cookie back to the client with its value increased by one unit.
   - To do this, look at the documentation for the PHP *setcookie*() function. What is the path of your cookie? Do the same visit counter but now only for an *images* subdirectory (which you will need to create).

- **Make a program in PHP that shows all the videos available in your database as an HTML table**. **Only the title, country and associated photo will be displayed.**
- Review the summary of DB access with PHP that is provided at the end of this booklet. Also check the online documentation.
- To check the name of the tables, fields and data types, you can use phpMyAdmin. It is a web interface for the administration and management of MySQL servers. Its operation is intuitive. Open the browser and type the following URL: http://labit601.upct.es/phpmyadmin
- The script will have to read the corresponding table from the DB and then render it in HTML table format. What query have you used?
- Take a look at URL for the images in the database. Where are you going to store the images in your server? How do you insert them?

2) **Make a program that allows you to insert new videos in your database.**
   - First create a suitable HTML form that allows you to submit the information associated with the video. Regarding the image, you have two options: (1) provide only the URL of the image to insert it in the database or (2) upload the image using the form so that the script saves it in a folder and inserts the corresponding URL in the database.
   - Create a PHP script that processes the form and insert the video in

the database.

3) **Modify exercise 2 so that the table now shows hyperlinked titles, so that by clicking on the title a page is accessed which also shows the description of the video and a form to be able to add comments.**
   - The hyperlinks have to refer to the script that will show the description of the video and have a query string to identify the video in the database.
   - The script that displays the description will use the video id provided to it to display its description.
   - In addition, it will generate an HTML form that allows the insertion of comments. Although the script that will process the insertion of comments is not asked, consider what information that script should receive, what information should it receive and how did you include it in the form?

4) **Use cookies to identify users.**

   a) Create a simple login form, in which a user will simply enter his / her nickname.

   b) The script that processes this form checks in the database that the nick exists (attention to checking strings may need to use the LIKE string comparison clause) and also retrieves the user id and its name.

   c) The user id will be used in a cookie that will serve to the server in subsequent requests to know that the user was previously identified. Set a life time.

   d) Modify the previous exercises so that when the complete description of a video is shown, the form to add users is updated with the name of the identified user.

   e) Finally, you can alternatively use PHP sessions. It is a mechanism that simplifies the establishment of session variables that are conserved between requests. Briefly, internally the server uses a cookie with a unique identifier to establish a session and saves the session variables in local files, which are then retrieved. To work with sessions, the session_start () and superglobal $_SESSION array are used.

5) Show the description of the videos along with all the associated comments. The nick of the user who left the comment will be displayed.

6) Make the nick of the user associated with a comment is hyperlinked, so that by clicking on it we are shown a new page with the user information and all comments.

How to access to database with PHP

To access a database with PHP we have several options: you can use specific libraries for a particular database (MySQL, Postgre, etc.) or we can use PDO (PHP Data Objects) which is a library that abstracts the internal operation of the Particular database offering a common interface for any of the available database. That is, with PDO we can exchange the database that we use

without having to modify the code. In these practices we will use PDO (http://php.net/manual/es/book.pdo.php). In our case, we used a MySQL database, so we have to configure it to use the MySQL driver for the corresponding functions.

To work with a database usually follow the following steps:

Connect to the database by creating a PDO object with the corresponding configuration string. To use a different database simply change the corresponding configuration string. Although only shown here for the creation of the PDO, the whole code would be included inside the try block

```
try {
        $pdo = new PDO('mysql:host=localhost;dbname=catalogo, 'user',
'password')

} catch (PDOException $e) {

    echo 'Connection failed: ' . $e->getMessage();
};
```

From this moment on, the desired query is performed, creating a string of characters with the necessary SQL statements. In this case a c variable is used in the query.

```
$query = 'SELECT * FROM catalogo WHERE `categoria`='.$c;
```

The query is executed with *query ()*. The results, although returned as a PDOStatement object, are internally saved as table (array) where each row corresponds to the corresponding row in the database (depending on the query).

```
$result = $pdo->query($query);
```

Iterate the result matrix with the function fetch (), or some other available ones. Each call to the function returns a row in the table until it reaches the end.

```
// Printing results in HTML
        while ($line = result->fetch(PDO::FETCH_ASSOC)) {
                echo "<tr>";
                echo "\t\t<td><input type=\"radio\" name=\"id\"
value=\"".$line["id"]."\"></td>\n";
                echo "<td>".$line["descripcion"]."</td>\n";
                echo "<td>".$line["precio"]."</td>\n";
                echo "<td>".$line["unidades"]."</td>\n";
                echo "\t</tr>\n";
        }
```

Memory is freed. For large queries this improves performance. The connection to the database is closed when there are no more queries to

execute. This is done by assigning null objects.

```
$result=null;
$pdo=null;
```

These last steps are repeated as many times as necessary, ie for each query that needs to be done.

The syntax of the most common queries is as follows:

Delete:

```
DELETE FROM Tabla WHERE user = 'jcole'
```

Insert:

```
INSERT INTO Tabla (col1, col2, col3) VALUES(15, 76, 43)
```

Update one row:

```
UPDATE Tabla SET dni=48423234
UPDATE Tabla SET age=age+1
```

Select data:

```
SELECT * FROM Tabla;
SELECT dni, edad FROM Tabla
```

To select, update or delete you can set conditions adding a WHERE clause at the end. This clause may use expressions:

```
SELECT * FROM table WHERE ID = 47373621
SELECT * FROM Table 15 WHERE units = AND price <100
SET UPDATE Table age = 26 WHERE dni = 48423234
```

Although multiple successive queries with PHP could be performed, it is almost always preferable to combine queries over several tables. For example:

```
SELECT tabla.dni, tabla2.salario FROM table1, table2
  WHERE t1.dni = t2.dni;
```

The above query selects elements of relational tables, such as:

dni	first name	surnames	age
4523345	Juan	García	46
1234223	Ana	Pérez	37

dni	salary	HireDate
4523345	15000	11-12-2005
3232535	26000	09-10-2005

The above query would return the user to pay dni 4523345 since it is the only overlap in the field dni between Table 1 and Table 2.