University of Regina

CS 215
Web Oriented Programming

MySQL & PHP

Dr. Orland Hoeber
orland.hoeber@uregina.ca
http://www.cs.uregina.ca/~hoeber/cs215/

Readings

- Chapter 10 - 11
- http://www.w3schools.com/php/php_mysql_intro.asp

- Assignment 4 is due soon (Nov 21)

Client-Server Architecture

- Although the Web already operates using a client-server architecture, there is a second level of client-server communication when we use databases
  - web browser – web server
  - web server – database server
- Results in a three-tier architecture

Figure 13.6 Three-tier architecture of a Web site supported by databases

Three-Tier Architecture

- There are some important things to keep in mind regarding this three-tier architecture:
  - the Web server is both a server (to the browser) and a client (to the database server)
  - the Web browser is a client of the Web server, not the database server
  - the three-tier architecture remains valid even if the Web server and database server software are installed on the same computer (they run independently of one another)
  - PHP (or some other server-side programming language) is the glue between the server's server and client mode
  - It is important to know what is the communication mechanism between each tier

Database Access with PHP

- The first step in using a MySQL database within a Web application is to connect to the database
  - built-in PHP object: *mysqli*
    - this is the second implementation of the MySQL functions in PHP
    - the i stands for “improved”

```
$db = new mysqli(“<db server name>”, “<username>”,
“<password>”, “<database>”);
```

- The instance variable represents the connection the database, and will be used to issue commands to it

Checking the Connection

- Immediately after making the connection to the database, it is possible to check the status of the connection with the *connect_error* property
  - empty if the connection was made
  - contains an error description if the connection failed

```
$db = new mysqli("localhost", "hoeber", "pwd",
"hoeber");

if ($db->connect_error) {
   die ("Connection failed: " . $db->connect_error);
}
```

Closing the Connection

- The connection can be closed with the *close* method

```
$db->close();
```

- Note that PHP will automatically close the connection to the database at the end of the execution of the script (end of the page)
  - however, it is still good practice to explicitly close the connection
  - this is especially true of scripts that must do some non-trivial data processing after retrieving the data from the database
  - **I expect all of your connections will be explicitly closed**

Executing Queries

- After connecting to the database, other SQL operations can be executed via the *query* method:

```
$query = "SELECT * FROM Users;";
$result = $db->query($query);
```

- The return value is the data that resulted from the operation
  - FALSE on failure
  - TRUE on success for queries that do not return data (i.e., INSERT, UPDATE, DELETE)
  - for queries that return data (i.e., SELECT), the return value is an object that contains the result set
  - a set of methods exist for extracting the important data from this object

Dealing with the Results

- Determining the number of rows and columns:

```
$num_rows = $result->num_rows;
$num_cols = $result->field_count;
```

- Extracting the rows of data from the returned results:

```
$row = $result->fetch_assoc();
$row = $result->fetch_array();
$row = $result->fetch_object();
```

- Each of these only returns one row of data (starting from the beginning of the results set)
- They each return NULL when there are no more rows to show
- They differ in how they represent the data for access in PHP

Iterating Through the Results

- A simple loop can be used to iterate through the results

```
$db = new mysqli("localhost", "hoeber", "pwd", "hoeber");
if ($db->connect_error) {
   die ("Connection failed: " . $db->connect_error);
}
$query = "SELECT email FROM Users ORDER BY LastLogin DESC;";
$result = $db->query($query);
if ($result->num_rows > 0) {
     while ($row = $result->fetch_assoc()) {
        print ($row["email"] . "<br />");
     }
} else {
     print ("There are no users in the database.");
}
$db->close();
```

Overall Process for Database Access

- The overall process for accessing and using a database is:
  - connect to the database (create an instance of the *mysqli* object)
  - check that the connection has been created properly
  - issue the query (*query method*)
  - access the results set rows one by one (*fetch_assoc* method of the *mysqli_result* object)
  - disconnect from the database (*close* method)

- If your query does not return a result (insert, update, delete), then you will want to check the mysqli_result object to confirm that it resolves to True
- You should design your code to minimize the amount of time the database connection is kept open and the results set is kept active

Special Characters

- Since HTML has a number of special characters (&, <, >, etc.), care must be taken when extracting data from a database and inserting it into a Web page
  - there is a built-in PHP function for this very purpose:

```
$str = htmlspecialchars($str);
```

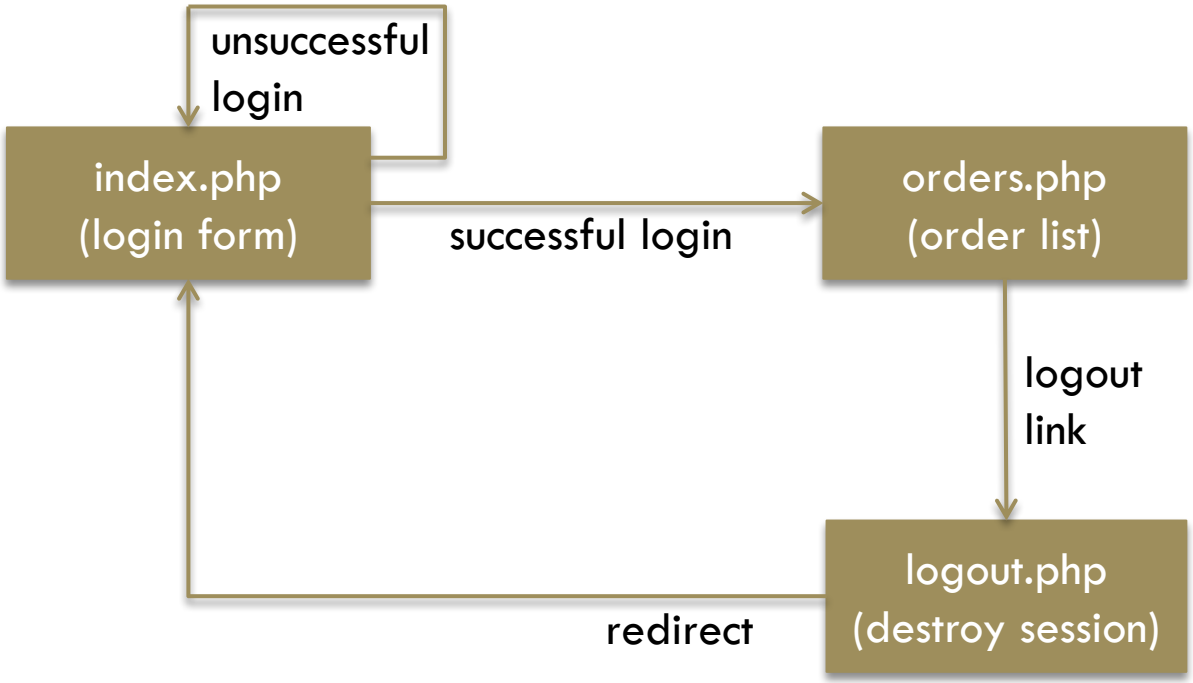
- There are also two related functions for automatically adding and stripping slashes from quotation marks:

```
$str = addslashes($str);
$str = stripslashes($str);
```

- A PHP server can be set to automatically perform the addslashes function (magic_quotes_gpc, set in the PHP.ini file, currently turned off on Hercules)

Login/Database/Session Example

- http://www2.cs.uregina.ca/~hoeber/teaching/cs215/2017F/examples/orders/index.php



Homework

- Read Chapter 17
- Next topic: AJAX
- Upcoming deadlines:
  - Assignment 4: Nov 21 @ 11:55 PM