**Using PHP with MySQL**

**Connecting to MySQL**

Mysqli\_connect() – this function connects to the MySQL server. Store this in a variable.

$dbc = mysqli\_connect(HOSTNAME, USERNAME, PASSWORD, DB\_NAME);

Set the arguments as constants e.g. DEFINE(‘HOSTNAME’, ‘localhost’).

Hostname, username & password are based upon the users & privileges established within MySQL. This information can be found in: xampp/phpmyadmin/config.inc.php. The 4th argument is the name of the database to use.

If the connection was made, the $dbc variable (short for database connection but you can use any name) will become a reference point for all of your subsequent database interactions. Most of the PHP functions for working with MySQL will take this variable as its first argument.

If no connection was made run: mysqli\_connect\_error(); this returns the error message.

Once connected to the database you should set the encoding, which should match the PHP scripts: mysqli\_set\_charset($dbc, ‘utf8’);

Example:

$dbc = @mysqli\_connect(HOSTNAME, USERNAME, PASSWORD, DB\_NAME) OR die(‘could not connect to MySQL: ’ . mysqli\_connect\_error());

*//the @ symbol prevents the PHP error from being displayed in the web browser*

Save the database connection as a separate file to the file you wish to work and interact with the database. Save outside of the web document directory for security reasons. Create a copy in the web directory so that it is accessible from the web browser – in order to test the script. Remember to remove this copy from the web directory once you are happy for the site to go live.

**Executing simple queries**

Once you are connected to a database, you can start executing queries. The PHP function for executing a query is: *mysqli\_query();*

This takes 2 arguments: the database connection and the query itself.

It is good practice to first store the query as a variable prior to running this function, e.g. $q = “SELECT \* FROM tablename”;

For simple queries like INSERT, UPDATE and DELETE etc… (which do not return records), the $r variable (short for result), will be either true or false, depending on whether the query executed successfully.

For complex queries that return records (SELECT being the most important of these), $r will be a resource link to the results of the query if it worked or be false if it did not.

$r = mysqli\_query($dbc, $q);

If($r) { //if true

//code

} else {

Echo mysqli\_error($dbc);

}

It is optional, but good practice, to close the MySQL connection when finished: *mysqli\_close($dbc);*

Example (p274-280):

//this example performs an INSERT query to add a record to the users table

//check for form submission

If($\_SERVER[‘REQUEST\_METHOD’] == ‘POST’) {

$errors = array(); //initialise an error array

//check for a first name

If(empty($\_POST[‘first\_name’])) {

$errors[] = ‘you forgot to enter your first name’;

} else {

$fn = trim($\_POST[‘first\_name’]);

}

// do the same for last name, email and password etc…

If(empty($errors)) { //if there are no errors (by the errors array being empty)

//register the user in the database

Require(‘../mysqli\_connect.php’); //insert php file with db connection

//make the query

$q = “INSERT INTO users(first\_name, last\_name, email, pass, reg\_date) VALUES(‘$fn’, ‘$ln’, ‘$e’, SHA1(‘$p’), NOW() )”;

$r = mysqli\_query($dbc, $q); // run the query

If($r) { //if it ran ok

//print a message:

Echo “<h1>Thank you!</h1><p>You are now registered</p>;

}else{ //if it did not run ok

Echo “<h2>You could not be registered due to system error</h2>”;

//debugging message

Echo ‘<p>’ . mysqli\_error($dbc) .’<br />Query: ‘ .$q . ‘</p>’;

}

Mysqli\_close($dbc); //close the database connection

} else { //report the errors

“<p>The following errors occurred: <br />”;

Foreach($errors as $msg) { //print each error

Echo “- $msg<br /> \n”;

}

} //end of if (empty($errors)) IF

} //end of initial IF conditional

?> //create form below

**Retrieving query results**

The primary tool for handling SELECT query results is: *mysqli\_fetch\_array()*; which uses the query result variable ($r) and returns one row of data at a time, in an array format. You will want to use this function within a loop that will continue to access every returned row as long as there are more to be read. You will almost always want to use a while loop to fetch the results from a SELECT query.

While($row = mysqli\_fetch\_array($r)) {

// do something with $row

}

The mysqli\_fetch\_array() function takes an optional second parameter specifying what type of array is required: associative, indexed or both (default). Associative allows you to refer to column values by name, whereas indexed array requires you to use only numbers.

|  |  |
| --- | --- |
| Constant | E.g. |
| MYSQLI\_ASSOC | $row[‘columnname’]; |
| MYSQLI\_NUM | $row[0]; |
| MYSQLI\_BOTH (default) | $row[0]; or $row[‘columnname’]; |

\*it is good practice to use: mysqli\_free\_result($r); which removes the memory taken by $r, once you are finished with the query\*

Example (p282-284):

//make the query

$q = “SELECT CONCAT(last\_name, ‘, ‘, first\_name) AS name, DATE\_FORMAT(reg\_date, ‘%M %d, %Y’) AS dr FROM users ORDER BY reg\_date ASC”;

$r = @mysqli\_query($dbc, $q); //run the query

If($r) { //if it ran ok

//fetch and print all the records:

While($row = mysqli\_fetch\_array($r, MYSQLI\_ASSOC)) {

Echo $row[‘name’] . ‘<br />’;

}

Mysqli\_free\_result($r); //free up the resources

}else{ //if it did not run ok

Echo “<p>The current users could not be retrieved</p>”;

//debugging message:

Echo mysqli\_error($dbc) . ‘<br />Query: ‘ .$q;

}//end of if ($r) IF

Mysqli\_close($dbc); //close the database connection

**Ensuring secure SQL**

Database connection with respect to PHP comes down to 3 broad issues:

1. Protecting the MySQL access information
2. Not revealing too much about the database
3. Being cautious when running queries, particularly those involving user-submitted data
4. Accomplished via securing the MySQL connection script outside of the web directory so that it is never viewable through a web browser
5. Attained by not letting the user see PHP’s error messages or your queries
6. Numerous steps you should take:
   1. Validate that some value has been submitted or that it is of the proper type (number, string etc…)
   2. Use regular expressions to make sure that submitted data matches what you would expect it to be
   3. Typecast some values (discussed in chapter 13)
   4. Run user-submitted data through the mysqli\_real\_escape\_string() function. This function makes data safe to use in a query by escaping what could be problematic characters: $safe = mysqli\_real\_escape\_string($dbc, data); This function should be used on every text input in a form

Example p286-289:

$firstname = mysqli\_real\_escape\_string($dbc, trim($\_POST[‘first\_name’]));

\*trim() removes whitespace and other predefined characters from both sides of a string\*

**Counting returned records**

Mysqli\_num\_rows()

This function returns the number of rows retrieved by a SELECT query. It takes one argument; the query result variable.

$num = mysqli\_num\_rows($r);

It’s a good idea to use this function before you attempt to fetch any results using a while loop (because there’s no need to fetch the results if there aren’t any).

Example p290-291:

$num = mysqli\_num\_rows($r);

If($num > 0) { //if ran ok

//display records

Echo “there are $num records”;

//fetch and print records using while loop

//then free up memory using mysqli\_free\_result($r);

}else{ //if no results

//display message

Echo “no records returned”;

}

//close database connection

**Updating records with PHP**

To update, first you need an UPDATE query using the mysqli\_query() function. Then verify using the mysqli\_affected\_rows() function, which shows the number of rows affected by an INSERT, UPDATE or DELETE query. The argument takes the database connection.

Example p292-297:

//this page lets a user change their password

If($\_SERVER[‘REQUEST\_METHOD’]) {

Require(‘../mysqli\_connect.php’);

$errors = array();

//check for email address

If(empty($\_POST[‘email’])) {  
 $errors[] = ‘you forgot to enter your email address’;

}else{

$e = mysqli\_real\_exape\_string($dbc, trim($\_POST[‘email’]));

}

//do the same to check for current password – store current password as $p

//check for a new password and match against the confirmed password

If(!empty($\_POST[‘pass1’])) {

If($\_POST[‘pass1’] != $\_post[‘pass2’]) {

$errors[] = ‘your new password did not match the confirmed password.’;

}else{

$np = mysqli\_real\_escape\_string($dbc, trim($\_POST[‘pass1’]));

}

}else{

$errors[] = ‘you forgot to enter your new password’;

}

If(empty($errors)) { //if everthing ok

//check that you have entered the right email/password combination

$q = “SELECT user\_id FROM users WHERE (email = ‘$e’ AND pass=SHA1(‘$p’) )”;

$r = @mysqli\_query($dbc, $q);

$num = @mysqli\_num\_rows($r);

If($num == 1) { //match was made

//get the user id:

$row = mysqli\_fetch\_array($r, MYSQLI\_NUM);

//make the update query

$q = “UPDATE users SET pass=SHA1(‘$np’) WHERE user\_id=$row[0]”;

$r = @mysqli\_query($dbc, $q);

If(mysqli\_affected\_rows($dbc) == 1) { //if it ran ok

Echo “thank you, your password has been changed”;

}else{ //if it did not run ok

Echo “your password could not be changed due to system error”;

//debugging message

Echo mysqli\_error($dbc) .’<br />Query: ‘ .$q;

}

//close db connection – mysqli\_close($dbc);

}else{ //invalid email/password combination

Echo “<h1>Error!</h1><p>The email and password do not match</p>”;

}else{ //report the errors

Echo “The following errors occurred: ”;

Foreach($errors as $msg) {

Echo “ - $msg<br />\n”;

}

}//end of if(empty($errors) IF.

Mysqli\_close($dbc); //close the db connection

} //end of the main submit conditional

//close PHP tag and create HTML form to change password

**Test**

**Part 1**

1. Connect to MySQL using function – ensure to use constants as the function arguments
2. How do you find the information needed for the above function arguments if you do not already know them?
3. If the connection was not made, what function do you use to display the error message?
4. What does the @ symbol do?
5. Where should you save the file with the database connection reference point?

**Part 2**

1. How do you execute queries once you are connected to the database?
2. It is considered good practice to close the database connection once you are finished with it – how is this achieved?
3. Give an example of inserting, deleting & updating queries in a database
4. What is the primary function used for handling SELECT query results? Give basic example
5. The function above (4) takes an optional second argument – what is it? Give example
6. How can you remove the memory taken up by the query once you are finished with it?
7. Give an example of retrieving data (SELECT) from a database

**Part 3**

1. What are the 3 security concerns when working with databases in PHP & how can these be minimised?
2. What does mysqli\_real\_escape\_string() do? Give example
3. What does trim() do?
4. What function can you use to find the number of records returned from a SELECT query?
5. What does mysqli\_affected\_rows() do? Give example
6. Give an example of how you can change a password in a database