**PHP and MySQL**

## **Session 6 Practical**

## **DB Connectivity & Functions**

## **Exercise 1**

The following exercises will only work if you have created the db1\_%USERNAME% (where your MySQL username is substituted for %USERNAME%) database from the week 5 practical.

We will now create an html file that contains a form, which will allow users to enter new records into the “test”.

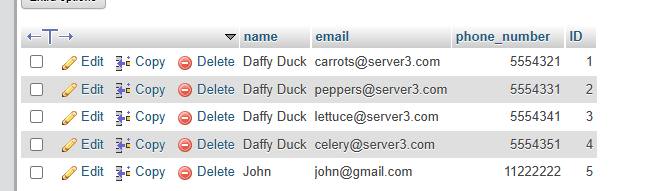
Create the file below and save it as wk6ex1.html.

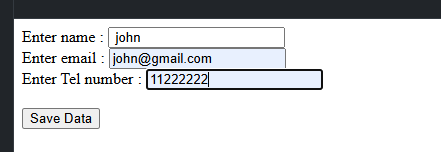


Now create the file below and save it as wk6ex1action.php. The script is incomplete and requires several additional lines (see comments), to connect to the server and execute the two SQL statements. Refer to your lecture notes for syntax required for the missing lines.

When complete run wk6ex1.html to see if it does save the new record.







As you can see, I have fully inserted the data and it has put the data inside of my test table, showing that I have gotten perfect results.

wk6ex1action.php (modified)

<?php

// Step 1: Connect to the database

$servername = "localhost";

$username = "root"; //

$password = "root"; //

$dbname = "db1\_gwalke01";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Step 2: Insert data into the table

$sql = "INSERT INTO test (name, email, phone\_number)

VALUES ('$\_POST[txtName]', '$\_POST[txtEmail]', '$\_POST[txtPhoneNumber]')";

if ($conn->query($sql) === TRUE) {

echo "New record created successfully!<br/>";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

// Step 3: Select all records from the table

$sql = "SELECT \* FROM test";

$result = $conn->query($sql);

// Step 4: Display the records

if ($result->num\_rows > 0) {

// Output data of each row

while($row = $result->fetch\_assoc()) {

echo "Name: " . $row["name"]. " - Email: " . $row["email"]. " - Phone: " . $row["phone\_number"]. "<br>";

}

} else {

echo "0 results";

}

// Step 5: Close the connection

$conn->close();

?>

## **Exercise 2**

The next application (containing two new scripts) that will display the people within your test table as a list of links. When you click on one of the links,that persons details will be displayed. Save the file below as wk6ex2.php. Remember to add the code necessary to connect to the server, select the database and execute the query.



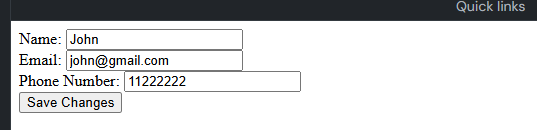
Now create the file below and save it as wk6ex2action.php. Remember to add the code necessary to connect to the server, select the database and execute the query.



Load wk6ex2.php into your browser and check it works.

Note how the id is sent to wk6ex2action.php as part of the query string and consequently we must use $\_GET[id] to pick up its value within the script.





wk6ex2action.php (modified.)

<?php

// Connect to server and select database

$servername = "localhost";

$username = "root";

$password = "root";

$dbname = "db1\_gwalke01";

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Retrieve the 'id' from the query string

$id = $\_GET['id'];

// Query to fetch the details of the selected person

$sql = "SELECT \* FROM test WHERE ID = $id";

$result = $conn->query($sql);

// Fetch and display the person's details

if ($result->num\_rows > 0) {

$row = $result->fetch\_assoc();

?>

<form action="wk6ex2action.php?id=<?php echo $row['ID']; ?>" method="post">

Name:

<input type="text" name="txtName" value="<?php echo $row['name']; ?>" readonly /><br>

Email:

<input type="text" name="txtEmail" value="<?php echo $row['email']; ?>" /><br>

Phone Number:

<input type="text" name="txtPhoneNumber" value="<?php echo $row['phone\_number']; ?>" /><br>

<input type="submit" name="update" value="Save Changes">

</form>

<?php

} else {

echo "No records found.";

}

$conn->close();

?>

wk6ex2.php (modified)

<?php

// Connect to server and select database

$servername = "localhost";

$username = "root";

$password = "root";

$dbname = "db1\_gwalke01";

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Query to fetch all records from the test table

$sql = "SELECT \* FROM test";

$result = $conn->query($sql);

// Display the records as clickable links

if ($result->num\_rows > 0) {

while($row = $result->fetch\_assoc()) {

echo "<a href=\"wk6ex2action.php?id=" . $row['ID'] . "\">" . $row['name'] . "</a><br>";

}

} else {

echo "No records found.";

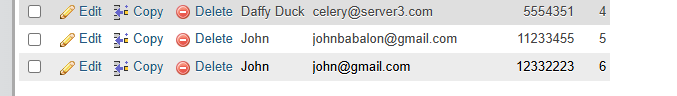
}

$conn->close();

?>

**Exercise 3.**

While exercise 2 allowed you to select a record and display its contents, it did not allow you to update the data within the record. Create a third script file that will save the changes made to the email and phone number fields within wk6ex2action.php.



Now the changes are updated using the code I made for updating.

wk6ex2update.php

<?php

// Connect to server and select the database

$servername = "localhost";

$username = "root";

$password = "root";

$dbname = "db1\_gwalke01";

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Retrieve the 'id' from the query string

$id = $\_GET['id']; // We passed this id from wk6ex2.php

// Retrieve the updated values from the form

$name = $\_POST['txtName'];

$email = $\_POST['txtEmail'];

$phone\_number = $\_POST['txtPhoneNumber'];

// Prepare the SQL query to update the record

$sql = "UPDATE test SET email='$email', phone\_number='$phone\_number' WHERE ID=$id";

// Execute the update query

if ($conn->query($sql) === TRUE) {

echo "Record updated successfully!";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

// Close the database connection

$conn->close();

?>

CHANGED THE ACTION FILE SO THAT IT REDIRECTS TO THE UPDATE.

<form action="wk6ex2update.php?id=<?php echo $row['ID']; ?>" method="post">

Name: <input type="text" name="txtName" value="<?php echo $row['name']; ?>" readonly /><br>

Email: <input type="text" name="txtEmail" value="<?php echo $row['email']; ?>" /><br>

Phone Number: <input type="text" name="txtPhoneNumber" value="<?php echo $row['phone\_number']; ?>" /><br>

<input type="submit" value="Save Changes" />

</form>

**Exercise 4.**

Modify the scripts to allow a user to delete a record from the test table.

CODE FOR WK6EX2.PHP UPDATED:

<?php

// Connect to server and select database

$servername = "localhost";

$username = "root";

$password = "root";

$dbname = "db1\_gwalke01";

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Query to fetch all records from the test table

$sql = "SELECT \* FROM test";

$result = $conn->query($sql);

// Display the records as clickable links with a delete option

if ($result->num\_rows > 0) {

while ($row = $result->fetch\_assoc()) {

echo "<a href=\"wk6ex2action.php?id=" . $row['ID'] . "\">" . $row['name'] . "</a> ";

echo "<a href=\"wk6ex2delete.php?id=" . $row['ID'] . "\">Delete</a><br>";

}

} else {

echo "No records found.";

}

$conn->close();

?>

To make this easier, I have decided to also create a new file which is called wk6ex2delete.php, which allows for an easier way to structure the deletion code.

<?php

// Connect to server and select the database

$servername = "localhost";

$username = "root";

$password = "root";

$dbname = "db1\_gwalke01";

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Get the ID from the URL query string

$id = $\_GET['id'];

// SQL query to delete the record with the given ID

$sql = "DELETE FROM test WHERE ID = $id";

if ($conn->query($sql) === TRUE) {

echo "Record deleted successfully!<br>";

echo "<a href='wk6ex2.php'>Back to the list</a>";

} else {

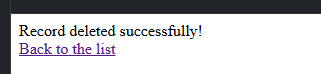
echo "Error: " . $sql . "<br>" . $conn->error;

}

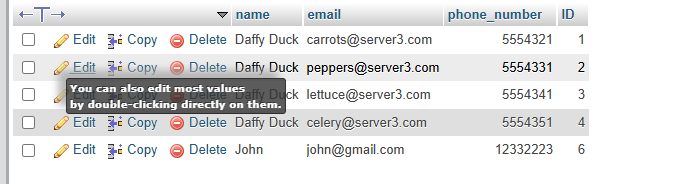
// Close the database connection

$conn->close();

?>







I forgot to screenshot before and after, but as you can see, one of the johns’ records has now been deleted.

## **Exercise 5**

Like most languages there is a function construct within php, that allows you to name a group of statements, which are executed when the function name is called within your scripts. The two functions below illustrate, how functions might be used to create a basic html document. Create a new file containing the two functions below and name it myfunctions.inc



Now you must write the script that will call these functions.

Create a new script file containing the script below and save it as wk6ex5.php. Notice how we have passed the name of the file that contains our two functions to the php include function. This will allow us to access these functions even though they are not declared within wk6ex5.php. Now run the script; 

The argument “My first function demo” is passed to the $pageTitle parameter of the html\_header function where it is output as the html page title (in the blue bar at the top of your browser). The argument to the html\_h1 function is displayed within the <h1> (heading 1) html tags.

Create your own function within myfunctions.inc that will display the text passed to it in <h2> (heading 2) html tags. Now include a line within wk6ex5.php that will call this function (passing some text as an argument).

modified myfunction.inc:

<?php

// Function to display text in <h2> tags

function html\_h2($text) {

echo "<h2>$text</h2>";

}

?>

created my own function within the functions.inc file

<?php

// Include the myfunctions.inc file which contains our functions

include("myfunctions.inc");

// Call the html\_h2 function and pass some text as an argument

html\_h2("This is my custom h2 header!");

?>

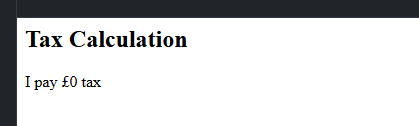
## **Exercise 6**

In addition to generating html code, user defined functions can perform repetitive calculations as in the tax calculating function below. The function takes two arguments, your salary and the % tax rate. Add the calculatetax function to the myfunctions.inc file.



Create a new script file containing the script below and save it as wk6ex6.php. Now run the script. This script assumes your salary is £15,000 and the tax rate is 22%, change these figures to reflect you own income.





Me personally, I do not pay any tax in my current area of work, so I inserted 0% tax.

## **Exercise 7**

We can assign a default value to a parameter by setting its value in the functions declaration (as below). These default values will be assigned to a parameter, if you don’t pass a value when you call the function. Amend the calculatetax function contained within myfunctions.inc to provide a 40 percent default value to the tax rate.



Now remove the tax rate argument from the call to the calculatetax function defined in wk6ex6.php and run the script.



## Updated code:

<?php

// Include the functions from myfunctions.inc

include("myfunctions.inc");

// Set your salary

$salary = 15000; // Replace this with your actual salary

// Calculate the tax using the calculatetax function (without passing the tax rate)

$tax = calculatetax($salary);

// Output the result

html\_h2("Tax Calculation");

echo "I pay £" . $tax . " tax";

?>

## **Exercise 8**

The calculatetax function does not take into account the tax allowance (that is the first x pounds you earn are exempt from tax). Add a third parameter to the calculatetax function and include the tax allowance within the calculation.

Remember to pass the tax allowance within the function call within wk6ex6.php.

UPDATED CODE (calculatetax) TO PASS TAX ALLOWANCE:

functions.inc

<?php

// Function to calculate tax with tax allowance

function calculatetax($salary, $rate = 40, $taxAllowance = 0) {

// Deduct tax allowance from the salary

$taxableIncome = $salary - $taxAllowance;

// If taxable income is less than 0, set tax to 0

if ($taxableIncome < 0) {

$taxableIncome = 0;

}

// Calculate tax on the taxable income

$tax = ($taxableIncome \* ($rate / 100));

return $tax;

}

?>

UPDATED WK6EX6.PHP to use the tax allowance:

<?php

// Include the functions from myfunctions.inc

include("myfunctions.inc");

// Set your salary, tax rate, and tax allowance

$salary = 15000; // Replace with your actual salary

$rate = 22; // Replace with your actual tax rate

$taxAllowance = 5000; // Replace this with the actual tax allowance amount

// Calculate the tax using the calculatetax function (with tax allowance)

$tax = calculatetax($salary, $rate, $taxAllowance);

// Output the result

html\_h2("Tax Calculation with Allowance");

echo "I pay £" . $tax . " tax after a £" . $taxAllowance . " tax allowance.";

?>

**Composer and Phinx PHP Database Migrations**

**Exercise 9**

Install composer on your intweb environment via the command line and demonstrate it has been successfully installed.

The command to run is as follows:

curl -s https://getcomposer.org/installer | php

*(You can either connect directly from the command line using SSH or you can use PuTTY on the university machines)*

**Exercise 10**

Install Phinx on your intweb environment into a new project folder within “public\_html” via the command line by using composer.

*(Refer to the screencast demonstration / lecture recording for more details.)*

**Exercise 11**

Create a new database migration that makes some change to your database using the **change()** method. Screenshot your code and evidence of the migration having been run.

Hint: once you have coded your migration, the CLI command you can use is (assuming you are using the “production” environment):

phinx migrate -e production

**Exercise 12**

Create a new database migration that makes some change to your database using the **up()** method. You will need to create a **down()** method also. Screenshot your code and evidence of the migration having been run.

Hint: once you have coded your migration, the CLI command you can use is (assuming you are using the “production” environment):

phinx migrate -e production

**Exercise 13**

Rollback your migrations to check that they revert successfully and bring your database back into its original state (e.g. if you started with an empty database before running the migrations, after you’ve reverted you migrations you should end up with an empty database again).

Hint: try this command to rollback all migrations (assuming you are using the “production” environment):

phinx rollback -e production -t 0