Awarding Great British Qualifications

Module Title

Topic 5:

Design and Build a Database (2)

Scope and Coverage

This topic will cover:

- An introduction to GET and POST
- An introduction to MySQL
  - Database queries
  - Data types and ranges
  - SQL statements

Learning Outcomes

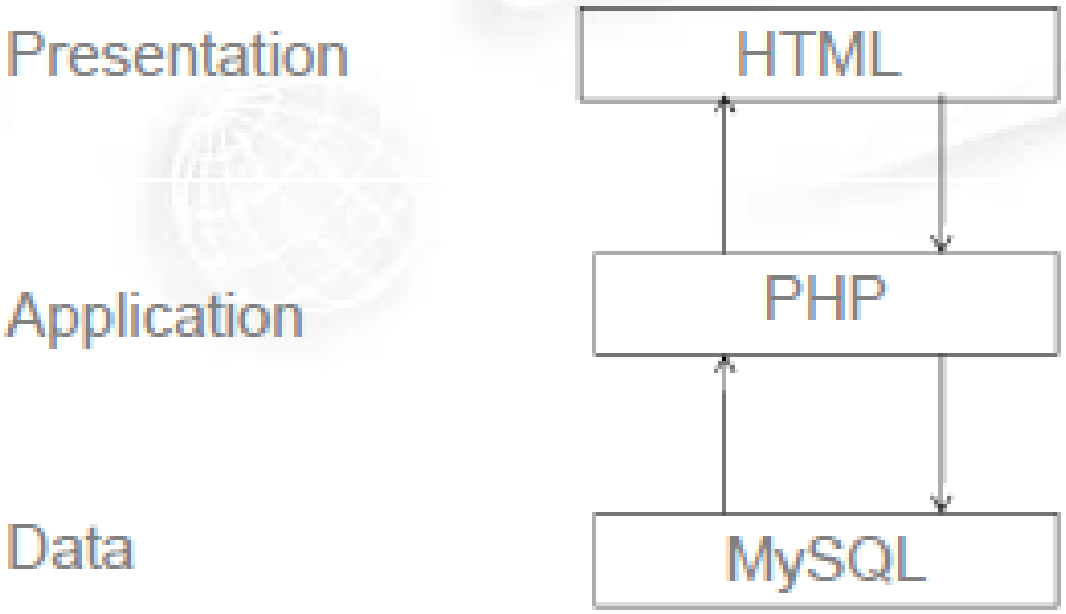
By the end of this topic students will be able to:

- Design and build a database which interacts with web page.

Introduction

- This lecture will introduce you to understanding how HTTP permits the sending of data to web pages and how to use MySQL into our dynamic websites architecture.
  - MySQL fits into the data layer of our N-Tier architecture.
- PHP acts as the mediator between the client and the database management system.
  - The presentation layer never communicates directly with the data layer.
  - The data layer never communicates directly with the client layer.

Our Architecture So Far



Databases and PHP

- In order to make use of a data layer, you must have access to a ***database***.
  - You will need one for the activities throughout this course.
  - You will need a username and password.
    - You will be provided with these as part of your course.
  - For the purpose of this course, we assume that your database is stored on a ***localhost***.
    - It doesn't have to be, but this would be the usual arrangement.

Connecting to the Database - 1

- For this module, we have created a user with login and password:
  - **monke13nccuser** with password **ncc1**
- When creating your own code you will use your **OWN** username and password.
- We will use a database called **monke13ncc.**
- This database is currently empty.
  - It has no tables.
  - It has no data.

Connecting to the Database - 2

- To make the connection to our MYSQL data, use the following code:

```
<?php
$host = “localhost”;
$user = “monke13_nccuser”;
$pass = “ncc1”;
$database = “monkey13_ncc”;

$connection = mysql_connect($host, $user, $pass)
or die (“Couldn’t connect to database”);

?>
```

Errors

- Depending on the level of reporting in your browser, you may or may not get meaningful error messages if this goes wrong.
- However, if you have a problem, then check the following:
  - The host is not correct
  - The username does not exist
  - The password does not match the user

A Database Connection

- Having made the connection to the server, you must select the database with which you were going to work.

```
<?php
        $host = “localhost”;
        $user = “monkey13_nccuser”;
        $pass = “ncc1”;
        $database = “monk13_ncc”;

        $connection = mysql_connect($host, $user, $pass)
          or die (“Couldn’t connect to database”);
        mysql_select_db ($database);

?>
```

Connection Made

- Once the connection is made, we can start manipulating our underlying database.
  - We do this using standard SQL queries, with which you should already be familiar.
- First we build a query in a variable, and then we pass the query and the database connection to a function called mysql_query.

GET

- Using the GET method, the information that is encoded gets sent as an extension to the URL.
- It will appear as something like: http://<url>/dice_roll_get.php?num=6&faces=7
- This information is available to PHP via the $_GET variable.
- This action used to provided data to a PHP form influences the code that we use to access it.
- We can make use of the GET protocol by changing the action in our form to GET.

POST

- The POST protocol is most useful on a day-to-day basis.
- POST has no limitations on size of data.
- It has no limitations on data types.
- It works by placing the encoded data in a standard HTTP header.

Limitations of POST and GET

- Both protocols permit you to send data to a PHP script
- That data persists as long as the script is running (if the page is reloaded it will usually ask to resend the data).
- If we move outside of the confines of a single PHP script, we will lose the data.
- Now lets create the database table ……

Creating a Table 1

```
$query = “CREATE TABLE test-table (
            FirstName varchar (15), SurName varchar (15)
)”;

$ret = mysql_query ($query, $connection);

If ($ret) {
Echo “<p>Table created!</p>”;
}
Else {
Echo “<p>Something went wrong: “ . Mysql_error ();
            + “</p>;
}
```

Creating a Table 2

- This is a process that can be done only once.
  - After that, it will fail saying that the table already exists.
- Mysql_error is a useful function that returns the next of the last MySQL error that was encountered.
  - Databases can be frustrating, so get into the habit of using this for diagnostic purposes
- Once we have browsed to this PHP page we will have a table in our database.
  - Often this is done as a separate **setup.php** process

Putting Data in a Table

- Almost all MySQL manipulation is done through the mysql_query function.
  - It will be the primary mechanisms by which you achieve your objective.
- We can use it to execute any valid SQL:

```
$query = “INSERT INTO test_table (FirstName, SurName ) values
   (\”Michael\”, \”Heron\”)” ;
$ret = mysql_query ($query, $connection);

If ($ret) {
  echo “<p>Data Inserted!</p>”;
}
Else {
   echo “<p>Something went wrong:  “  .  Mysql_error(); + “</p>”;
```

Getting Data out of a Database

- Getting data out of a database too is done through queries.
- However, manipulating that data requires us to do some further process.
- The results come out in the form of an ***associative array***.
- Data can be retrieved using GET or POST as seen above.

Sending of Data to Web Pages

- HTTP permits the sending of data to web pages.
- Two methods for this are:
  - GET
  - POST
- When it is time to send information, it is encoded by the client and then sent in one of two ways.

The Associative Array

- Most arrays are indexed with a number.
- Associative arrays are indexed with other kinds of data, such as descriptive strings.
- They work the same way – the index provides the corresponding element.
  - The index in an associative array is often called a ***key***.
  - The element is often called the ***value.***

Manipulating the Results

- The results come out as an ***array*** of ***associative arrays.***
  - The keys of each associative array are the fields in the database.
  - The values are the contents of the database corresponding to those fields for a record.
- Thus, making use of the data we have queried for the database requires us to provide handling code in PHP.

Extracting the Data - 1

- PHP gives us a number of helper functions.
  - If we want how many records that were returned, we use the mysql_num_rows function.
- This is valuable information we will need if we are to perform operators on each of the records that were returned.
- The function takes the results of a query as a parameter, and gives an integer in return.

Extracting the Data - 2

```
<?php
        $host = “localhost”;
        $user = “monkey13_nccuser”;
        $pass = “ncc1”;
        $database = “monk13_ncc”;

        $connection = mysql_connect($host, $user, $pass)
          or die (“Couldn’t connect to database”);
        mysql_select_db ($database);

        $query = “SELECT * FROM test_table”;

        $ret = mysqul_query ($query, $connection);

        $num_results = mysqul_num_rows ($ret);

        echo “<p>There were $num_results results returned from the query.</p>”;

?>
```

Extracting the Data - 3

- We have two main ways of getting to the actual data.
  - Through the mysql_result function
  - Through the mysql_fetch_array function.
- With the former, we provide the results, the index of the specific row we wish to query and the field we wish to query:

```
$num_results = mysql_num_rows ($ret);
$name = mysql_result ($ret, 0 “FirstName”);
Echo “<p>There were $num_results results returned from the query”.
“, and the first FirstName was $name</p.”;
```

Fetching a Row - 1

- The results set we get from the query contains each of the rows, but it also contains an internal counter of the last row for which we asked.
  - We can use mysql_fetch_array to fetch each row in order.
  - Each call to the function increments the counter by one.
- We can use this to perform manipulation or output on each row of the results one by one.

Fetching a Row - 2

```
Echo “<table width = \”100%\”>”;
Echo “<tr>”;
Echo “<th align = \”left\”>Record</th>”;
Echo “<th align = \”left\”>First Name</th>”;
Echo “<th align = \”left\”>Surname</th>”;
Echo “</tr>”;

For ($i = 0; $i <$num_results; $i++) {
$row = mysql_fetch_array ($ret);
Echo “<tr>”;
Echo “<td>$i</td>”;
Echo “td>” . $row[“FirstName”] . “</td>”;
Echo “td>” . $row[“SurName”] . “</td>”;
Echo “</tr>”;
}

Echo “</table>”;
```

User Input - 1

- We can incorporate user input using the techniques we have learned so far.

```
<html>
<head>
<title>Database Form</title>
</head>
<body>
<form action = “database_search.php” method = “POST”>
<p>First Name</p>
<input type = “text” name = “firstname”>
<p>Surname</p>
<input type = “text” name = “surname”>
<input type = “submit” value = “Search that Data”>
<input type = “reset” value = “Clear values”>
</body>
</html>
```

User Input - 2

- We can incorporate user input into queries into the same way we can into regular output:

```
$firstname = $_POST["firstname"];
$surname = $_POST["surname"];

$query = "SELECT * from test_table WHERE FirstName = \"$firstname\"
OR SurName = \"$surname\"";

$ret = mysql_query ($query, $connection);

$num_results = mysql_num_rows ($ret);
```

SQL Injection - 1

- Whenever we insert text from a user into a query, we run the risk of an SQL injection attach.
  - When someone incorporates their own SQL into their form input.
- Many browsers help combat this now by automatically sanitizing some of the input.
  - However, not all do, and not all PHP access will come via a browser.
- You should ***sanitise*** all user data using the mysql_real_escape_string()function.

Sanitised Input

- $firstname = $_POST[“firstname”];
- $surname = $_POST[“surname”];

- $firstname = mysql_real_escape_string ($firstname);
- $surname = mysql_real_escape_string ($surname);

- $query = “SELECT * from test_table WHERE
- FirstName = \”$firstname\” OR SurName = \”$surname\””;

- $ret = mysql_query ($query, $connection);

MySQL and PHP - 1

- A proper treatment of how SQL can be used to query a MySQL database is outside the scope of this module.
  - However you will have encountered it in previous modules.
- There are often two ways to accomplish any given data retrieval task using these tools.
  - A broad query with heavy PHP processing
  - A specific query with little PHP process

MySQL and PHP - 2

- With judicious use of joins, ordering and grouping by, you can create very sophisticated SQL queries.
  - These queries are often large and unwieldy.
  - However, MySQL is a platform designed for very efficient storage and retrieval of large amounts of data.
- PHP allows you to query a broad data set (such as select * from <table>) and then manually manipulate it for meaning.
  - This is inefficient, but can be useful for very fine detail work.

MySQL and PHP - 3

- In all cases, you must be mindful of several things:
  - Efficiency of data access
  - Readability of your code
  - Maintainability of your code
  - Portability of your architecture

- You want to make MySQL do as much of the work as it can.
  - However if you have to contort your queries to accomplish a simple goal, PHP can be a more effective tool.

Web Development Frameworks

- *Web development frameworks are software frameworks that are designed to support wed applications development.*
- *Web development framework* makes it easier to write, maintain and scale web applications.
- Web development framework provide tools and libraries that simplify common web development tasks e.g. interacting with databases, supporting sessions and user authorization, and etc.
- 2 examples of web development frameworks:
  - ASP.NET
  - RubyonRails

Ruby on Rails - 1

- Ruby on rails is a software programme that is designed as a framework for web development.
  - It is based on Ruby software
  - It is fun
  - It allows you to write less code
  - It is designed so that when you are writing code you don’t have to repeat yourself
  - Convention over configuration is the preferred method of coding.

Ruby on Rails - 2

```
# The Hello Class
class Hello

def initialise( name )
@name = name.capitalise
end
def salute
puts "Hello #{@name}!"
end
end
# Create a new object
h = Hello.new(“Andy")
# Output "Hello Andy!"
h.salute
```

Conclusion

- PHP offers integrated support for MySQL and other databases.
  - MySQL is the one we use throughout this module.
- PHP and MySQL integration is handled via the use of queries.
  - The data that comes back from a MySQL query can be further refined by PHP as needed.
- It is important to sanitise the input we receive from a user.
  - Or we risk SQL injection attacks.

Terminology

Associative Array - An array which is indexed by data types (usually strings) other than just integers.

SQL Injection - A (often malicious) piece of SQL that is provided by the user to influence the result of a query.

Sanitization - The process of taking potentially suspect user input and converting into a form that will not be harmful for database queries.

Ruby on Rails – web framework that is built on Ruby programming language.

References

- Rubyonrails, 201. [online] Available at: http://guides.rubyonrails.org/getting_started.htm

Awarding Great British Qualifications

Topic 5 – Design and Build a Database (2)

Any Questions?