Server side Database Programming with PHP

In this week, we will cover the following topics:

- *The topic of server-side programming.*
- *Introduction to the PHP programming*
  - *HTML-embedded PHP*
  - *Standalone command-line programming*
- *Typical server-side development environment*
- *PHP usage in context of MySQL relational database*

… and will result in the following learning outcomes

- *Appreciation of server-side programming*
- *Understanding of specifically how PHP is used in HTML*
- *Knowledge of the development environment needed to start developing your own server-side applications.*

Table of Contents

WEEK 8: SERVER SIDE DATABASE PROGRAMMING WITH PHP

8.1 Overview

During this week, we will look at the topic of server side database programming. There are different choices of programming language available for this purpose, but we will focus on the use of PHP. PHP is a very widely used scripting language and it is very useful in the context of web development.

Furthermore, the language contains libraries that can be used to easily interact with SQL databases in a web-based context. This feature of PHP is highly relevant and, therefore, to this course, and specifically to this part of the course, which is related to the practical use of databases.

The aim of this week not to provide you with a detailed review PHP, but to provide you with some insight to the use of PHP, embedded with HTML, to see how PHP can be used in this manner to:

- Connect to a database
- Send queries to the database and thereby
    - Rebate from a database, and display it on a webpage
    - Write data to a database

PHP is used for their creation of dynamic, interactive webpages. Dynamic, interactive webpages contrast with static web content. Static webpages are typically used to render static textual, and other (images, for example), information to a web browser. Creating a very simple static webpage is something that can be done when you first begin web-based programming.

However, in the real-world, for webpages to be useful they will often consist of static and dynamic content. For example, when a customer goes to a webpage to by a product they will often have to register their customer details, before purchasing. This information is stored by the company in the database. The kind of web-based programming required to perform this functionality comes under the dynamic web programming umbrella.

8.2 What is PHP?

The definition PHP is slightly circular. PHP means PHP Hypertext processor.

Several PHP features are worth noting, generally:

- **PHP is open source and highly relevant to web development, being embedded into HTML.**
- **Speedy access to database components/tiers of the server-side environment.**
- Scripts can be executed speedily with PHP.

- Supported widely (by operating systems, web servers etc.).
- Focussed on use for dynamic web content.
- Text processing features, parsing of XML for example.

- Also, a stand-alone programming language, capable of quite complex development tasks.

The first two features are the ones we will focus on this week, i.e. how PHP is used to programme on the server side, how it can be *embedded* within HTML code and then, specifically, how we can interact with relational databases in this manner.

Let's look at some PHP embedded within HTML:

```
<html>
<head>
<meta charset="UTF-8">
<title>PHP Hello World Test</title>
</head>
<body>
        <H1>Hello from HTML</H1>
        <P>HTML: "Hello world".</P>
        <H1>Hello from PHP</H1>
        <?php
        echo "Hello World. I am PHP a.k.a 'PHP Hypertext processor'";
        echo "<br> ";
        echo "...which could mean 'PHP Hypertext processor Hypertext processor'";
        echo "<br> ";
        echo "...due to the odd, recurcive definition";
        ?>
        <H1>Location of me:</H1>
        <P>http://localhost/database_scripts/test_php.php</P>
</body>
</html>
```

In this code, we can see (in green) the html mark-up tags. You should recognise from the mark-up the nested structure. So, everything exists *within* the start of the html (*<html>*) on the first line and the end of the html (*</html>*) and all text that is of the type 'top level heading' exists within the appropriate beginning and end tags (<H1> … </H1>) etc. The PHP code is embedded within the HTML code within the begging (*<?php*) and ending (*?>*).

Notice in the example that we have a HTML ' top-level heading' **Hello from PHP.** Everything after this and before heading **Location of me** can be thought of as PHP script. The Outputs from calling the file from browser is provided in Figure 1. Notice that, for this very simple example, we have not achieved anything with PHP that could not also be achieved by using straightforward HTML; all we have done is creates static content. The final heading in this example code**, Location of me:** we will return to below when we briefly discuss the installation PHP as part of the web development environment.

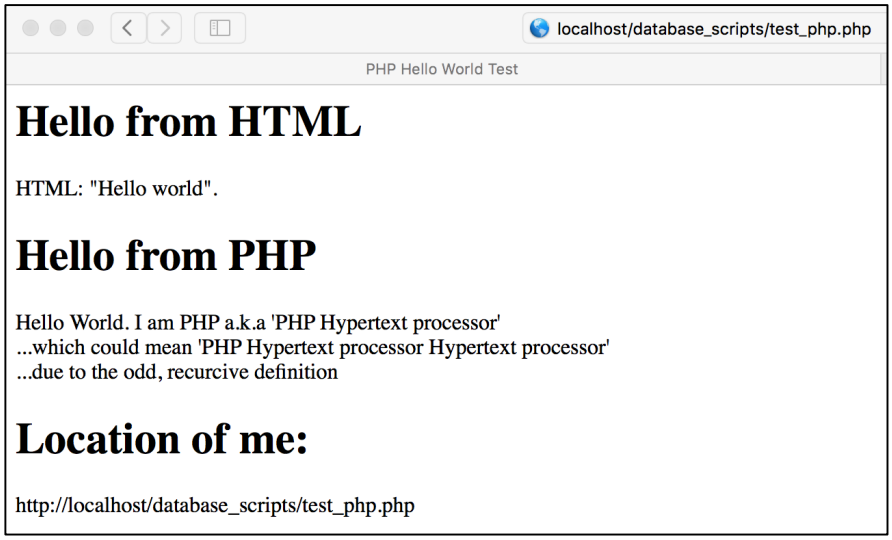


Figure 1: HTML-Embedded PHP: output from test_php.php.

As mentioned, above, PHP can be run from a stand-alone script, even though for this week we will be focusing on the alternative embedded-html usage. An example script is provided to illustrate this possibility, in Figure 2. In this way, PHP can be thought of as its own stand-alone programming/scripting language as well as being useful for web-based development.

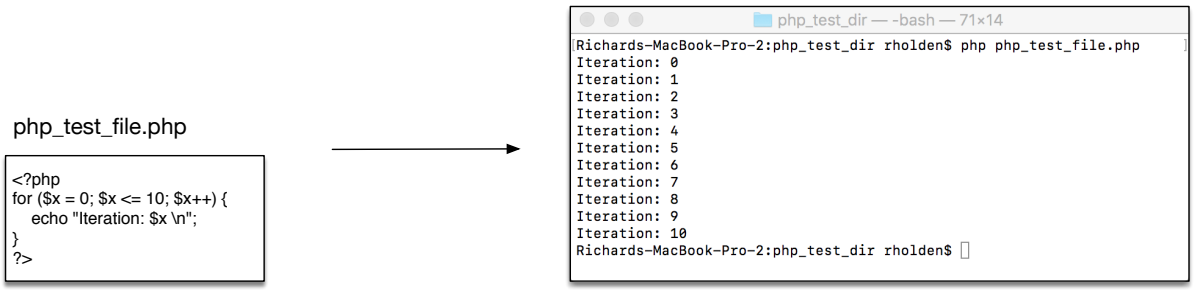


Figure 2: Pure PHP: output from test_php.php.

While PHP is very useful and widely used, there are some

One of the most confusing things about the use of PHP, HTML once an AMP stack is installed is where all all the components sit. We have already provided a commentary on the AMP stack and a general map of installations. The reason we have not provided very detailed installation instructions is that these are available on other courses, but also very specific features of the installations/configurations can change with different releases of the software. However, we can provide generally useful information about the development environment we assume for the example codes provided below, a topic to which we now turn.

8.3 The development environment

For the examples below, we assume that an AMP stack has been installed, that the apache program ***httpd*** is running on the localhost, and that a PHP interpreter is therefore also available.

With these pre-requisites in mind, let us consider a project structure, assuming a localhost setup.

8.3.1 The 'localhost'

Recall, a 'localhost' setup is typical for web development. While we have outlined previously client server setup, developers use the same computer to act both as the client *and* the server. While more generally we would refer two remote WebServer as hosting the website, when a local computer hosts the website it can be self-referenced using 'localhost', which loops back to an IP address of the local computer to act also as the server. Of course, during development the browser, and any other client programs also sit on the local machine, along with the database itself, the web-server, and web content. These aspects can be a little confusing because in the text books, the industrial configuration is usually presented, as we have also seen, previously, when discussing AMP stacks, but is rarely experienced as such as a developer, especially a budding developer.

For this reason, why have visualised how, during development, we often use a single machine to contain everything that would otherwise be connected over 'the network' proper. See Figure 3.

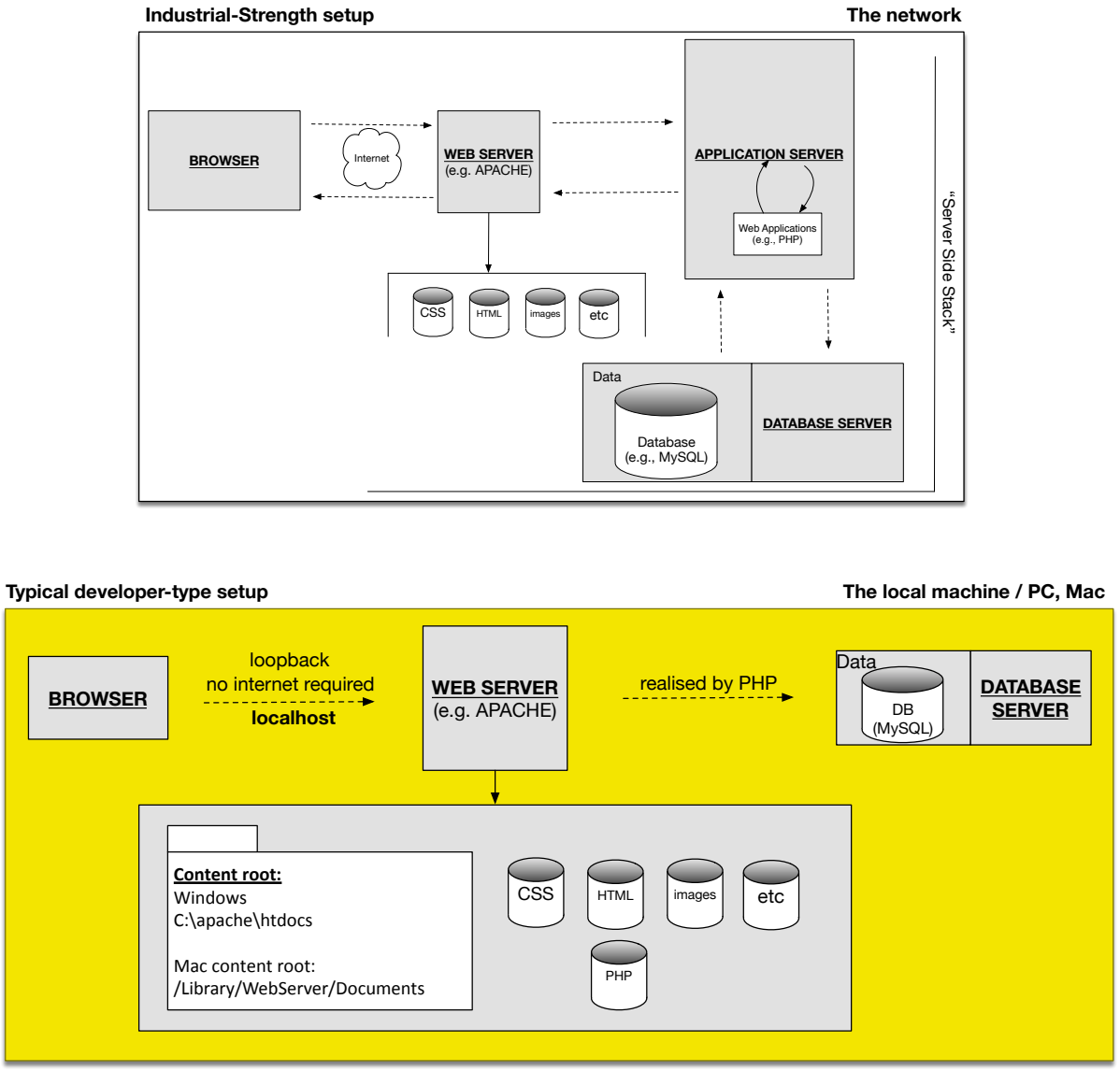


Figure 3: Development on localhost.

8.3.2 The web content folder accessed from IDE

Note, before we explain this section that the actual coding environment is very flexible. Some prefer to use an IDE and others prefer to code with a simple text editor. We will be using the eclipse IDE, to be consistent with when we come to develop some java code where an IDE is desirable. However, assuming that we have apache running (the AMP stack is installed and configured properly) when we change and save code within our edit (be it via an IDE or text edit), when load the page using a browser, the results of the code should work.

What do we mean when calling from the browser? In Figure 1, above, we deliberately noted the location of the <test_php.php> script on localhost:

```
<http://localhost/database_scripts/test_php.php>
```

In Figure 3, also note that the webserver is connected to by the browser via **localhost**. The contents of the website associated with localhost typically exist, on a windows machine, somewhere like the following:

```
C:\apache\htdocs
```

… or on a mac:

```
/Library/WebServer/Documents
```

These root folders provide the content specified by the various codes and resources (.css, .php, .html, images, etc.) in the root folder. Therefore, when we write:

```
<http://localhost/database_scripts/test_php.php>
```

…in the browser, we are running the code in <test_php.php>, which is in the <database_scripts> folder within the respective root folder of the machine. The root folder typically contains, by default, the index.html used initially for testing the installation of apache and, for testing the installation of PHP, a script, often names phpinfo.php, which calls the phpinfo() function. For example, on a windows machine we expect, given all the above, to see the content indicated in Figure 4.

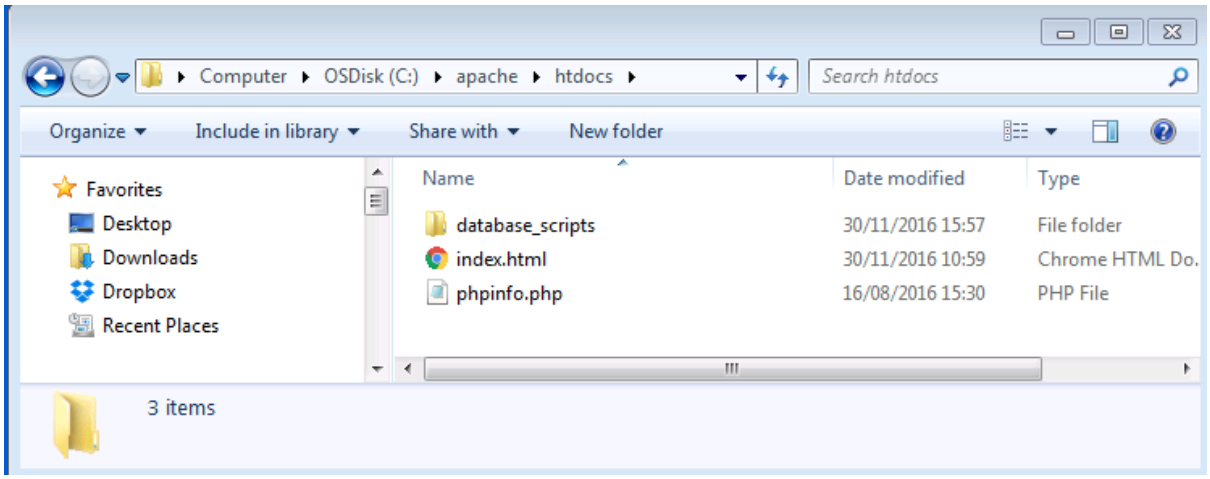


Figure 4: What to expect inside the web documents root folder?

8.3.3 Installing/configuring PHP to use mysqli library.

The purpose of this week is not to go into detail about how to install and configure PHP. However, for future reference it is worth noting that the installation of PHP, alone, does not necessarily provide access to the required PHP libraries.

Often the libraries can only be accessed if the PHP initialisation file has been properly configured. Therefore, it is important to know where the PHP initialisation file is located:

```
<C:\php\php.ini>
```

Within the PHP initialisation file are various options whose values can be set. To provide access to mysqli library, which is an extension, the extensions director must be specified along with the name of the dynamic link library which are depends on. These two parameters in the initialisation files are highlighted below.

extension_dir = "C:\php\ext"
extension=php_mysqli.dll

N.B., inside the initialisation file, you might find the following:

; extension_dir = ".\"
; extension=php_mysqli.dll

… and note the ';', which comments out the line in question.

Inside php.ini there are literally hundreds of parameters, but typically only a handful need to be altered when you would like to provide access to PHP extensions. For the practical sessions, your environment will have been installed and configured, but it is useful for you to know these things, if you were in interested in installing an AMP stack and configuring it yourself. Very often, when you run into trouble installing various resources, it can be very helpful to pose the relevant questions to:

http://stackoverflow.com

When we run the <phpinfo.php> script, in addition to the outputs provided by a default PHP installation, if the mysqli libraries are loaded then this should be indicated in the information table rendered in the browser. The specific table of interests for mysqli is presented in Figure 5.

mysqli

Mysqli Support	enabled
Client API library version	mysqlnd 5.0.11-dev - 20120503 - $Id: 76b08b24596e12d4553bd41fc93cccd5bac2fe7a $
Active Persistent Links	0
Inactive Persistent Links	0
Active Links	0

Figure 5: mysqli library is accessible!

8.4 Example PHP IDE

Editor
http://www.eclipse.org/pdt/

8.5 PHP libraries / APIs

If you go deeper into programming databases using PHP, you will come across different PHP API (application programming interfaces). We would like to point out two different kinds:

1. **mysqli**: this is an improvement over the original mysql API, and hence the appended 'i'. mysqli is very commonly used and supports two kinds of programming styles:
   - Procedural
   - Object Orientated

   In the example below we will be using the procedural syntax, which is slightly cleaner to look at for beginners. This api is also pre-configured for use with MySQL, which we have used on this course.

2. **PDO**: PHP Data Objects.

A tabular comparison of the high-level features of mysqli and PDO is contained in Table 1. This table is based on one presented in the following video, simplified for our purposes:

<PHP with MySQL Essential Training -> 14. Using PHP to Access MySQL -> Database API's in PHP>

	mysqli	PDO
Included with PHP	Yes	Yes
Pre-configured for MySQL	Yes	**No**
Other databases supported	**No**	Yes
Procedural	Yes	**No**
Object-Orientated	Yes	Yes

Table 1: PHP libraries for MqSQL. Key differences.

8.6 Introductory features of PHP language

The purpose of this section is merely to summarise some of the features PHP, such that you can recognise these features an example codes follow, below. For this purpose, we will cover

variables and some datatypes of relevance. For more in-depth introductions to PHP, please see the resources section.

8.6.1 Variables and data types

In PHP, variables are always prefixed with a dollar symbol, '$' and the data type is determined by the assignment, rather than explicitly, in-code.

PHP	Java
$three = 3;	int three = 3;
$three_and_half = 3.5;	float three_and_half = 3.5;
$message = "PHP";	String message = "JAVA';

Table 2: Variables in PHP v Java

There are two different kinds of array in PHP. There are straightforward arrays, which can be constructed using the array constructor, *array().* For example:

```
<?php
$persons = array("Dave", "Adam", "Ralph");
$person = $persons[0];
echo $person . "\n";
?>

-- program output when called from comman line --

Dave
```

The second kind of array is the associative array. This array allows it to associate an array value element with and non-integer index. Another way of saying this is that each 'entry' India right is, in effect, a key-value pair. Therefore, if we want to access the value of the pair we do this by specifying the key. For example:

```
<?php
$food = array("Friend" => "Sarah", "Best friend" => "Usain");

$afriend = $food['Friend']; // access value 'Sarah' with 'Friend' key
echo $afriend . "\n";

$afriend = $food['Best friend']; // access value 'Usain with 'Friend' key

echo $afriend . "\n";
?>

-- program output when called from comman lin --

Sarah
Usain
```

Variable scope: In the examples below, we will see that the scope variables are per-web-page. It is also possible for variables to be within-function scope, as with the other programming languages such as Java, C++ etc., where the resources for the variable I allocated one functions called and freed in the function returns.

8.6.2 Functions we will use in examples

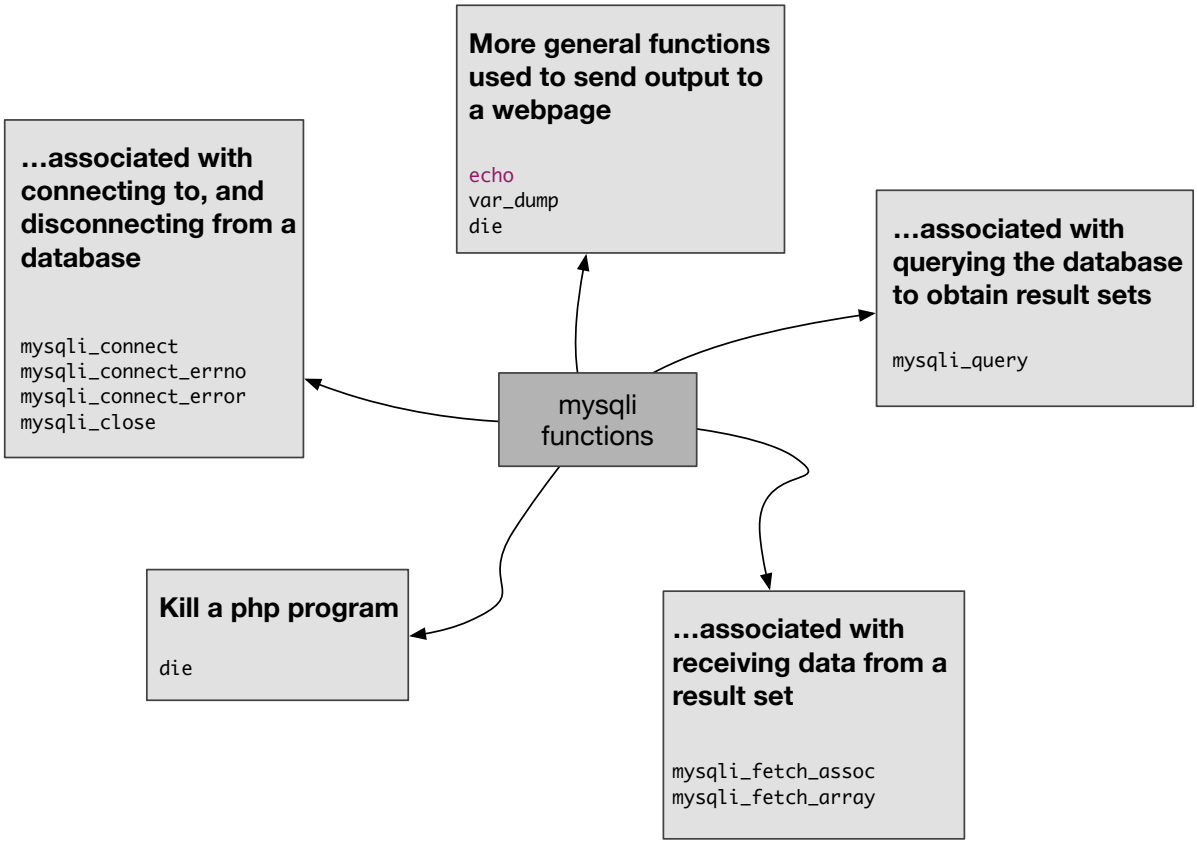


Figure 6: Some mysqli functions

8.7 PHP progressive examples

Five embedded-in-html php code examples are presented. Each script builds on previous scripts and, therefore, where the codes are identical a php comment is included to indicate as such, to simplify the presentation and to provide more of a focus on the newly introduced functions/code. A general description of each respective script is provided, here, where we indicate the new code to look out for in each script using the **emboldened** font:

- Example 1:
  - **Connects to a database, outputting errors where appropriate.**
  - **Closes the same database, outputting errors where appropriate.**
- Example 2:
  - Connects to a database, outputting errors where appropriate.

- **Queries the database, outputting errors where appropriate.**
- Closes the same database, outputting errors where appropriate.

- Example 3:
  - Connects to a database, outputting errors where appropriate.
  - Queries the database, outputting errors where appropriate.
  - **Outputs the query result to the browser, as an array type.**
  - Closes the same database, outputting errors where appropriate.

- Example 4:
  - Connects to a database, outputting errors where appropriate.
  - Queries the database, outputting errors where appropriate.
  - Outputs the query result to the browser, **as an associated array type.**
  - Closes the same database, outputting errors where appropriate.

- Example 5:
  - Connects to a database, outputting errors where appropriate.
  - Queries the database, outputting errors where appropriate.
  - Outputs the query result to the browser, **as an associated array type marked-up in html using the html table format.**
  - Closes the same database, outputting errors where appropriate.

We will not go into detail about each line in the. php scripts, but refer to to the practical sessions, which accompany these lectures . However, we do provide outputs from the scripts as they appear in the browser, from Figure 7 (example 1), to Figure 11 (Example 5).

8.7.1 PHP and MySQL: example 1

```
<html>
<title> Example 1 </title>

<?php
$dbhost = "localhost";
$dbusername = "root";
$dbpassword = "password";
$dbname = "retail_db";


echo "Attempting to connect to a database... <br> \n";
echo "<br> \n";
$connection = mysqli_connect ( $dbhost, $dbusername, $dbpassword, $dbname );

if (mysqli_connect_errno ()) {
       die ( "Database connection failure!! ..." . mysqli_connect_error () . " (" .
mysqli_connect_errno () . ")" );
} else {
       echo "Username: $dbusername<br> \n";
       echo "is connected to the database: $dbname <br> \n";
       echo "<br> \n";
}
?>
```

```
<body>
</body>
</html>

<?php
mysqli_close ( $connection );

if (mysqli_connect_errno ()) {
        echo "database $dbname <br> \n";
        echo " has not been detached (or does not exist) <br> \n";
} else {
        echo "Username: $dbusername <br> \n";
        echo "detached from the database: $dbname <br> \n";
        echo "<br> \n";
}
?>
```

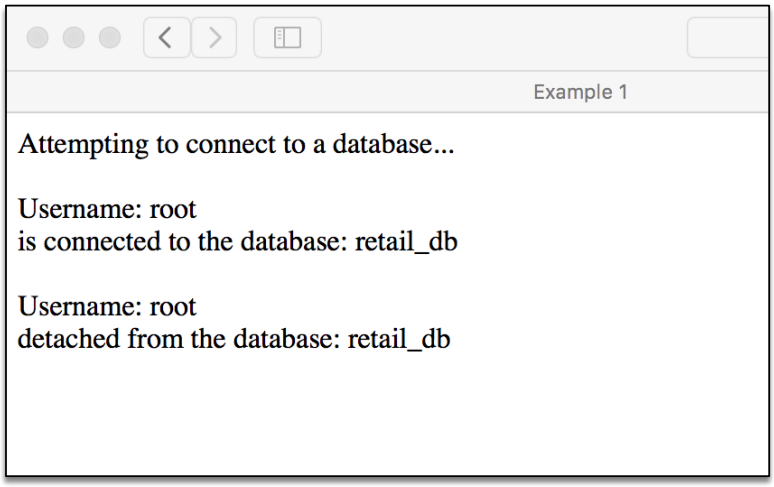


Figure 7: Example 1 output

8.7.2 PHP and MySQL: example 2

```
<html>
<title>Example 2</title>


<?php
// connect to db here, as above in Example 1
?>


<?php
$query = "SELECT * FROM department";
$query_result = mysqli_query ( $connection, $query );

if (! $query_result) {
        die ( "Query: " . $query . " failure!! " );
} else {
        echo "Query: $query <br> \n";
        echo "... successful <br> \n";
        echo "<br> \n";
}
```

```
?>

<?php
mysqli_free_result ( $query_result );
?>


<body>
</body>
</html>

<?php
// disconnect from db here, as above in Example 1
?>
```

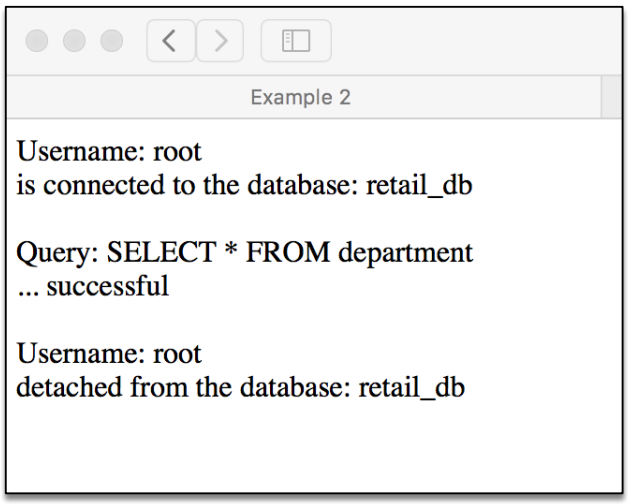
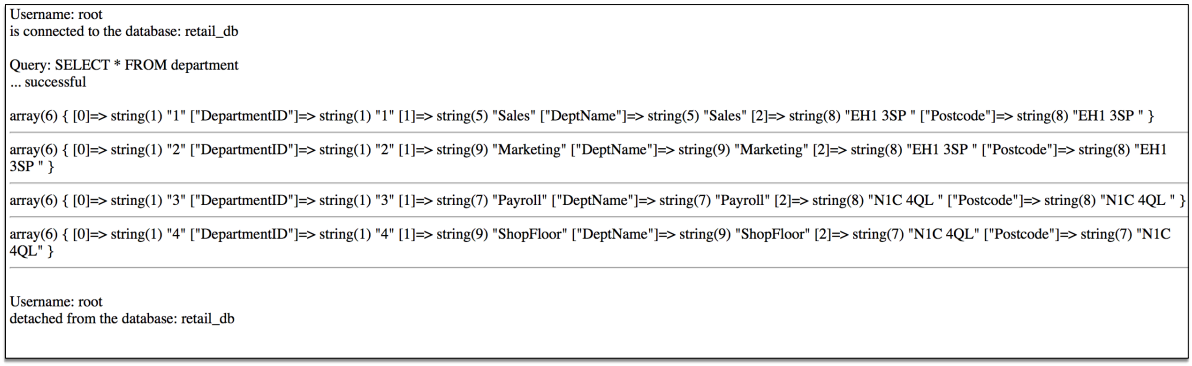


Figure 8: Example 2 output

8.7.3 PHP and MySQL: example 3

```
<html>
<title>Example 3</title>

<?php
// connect to db here, as above in Example 1
?>

<?php
// query db here, as above in Example 2
?>

<?php
while ( $department = mysqli_fetch_array ( $query_result ) ) {
        var_dump ( $department );
        echo "<hr />";
}
echo "<br> \n";
?>

<?php
mysqli_free_result ( $query_result );
?>
```

```
<body>
</body>
</html>

<?php
// disconnect from db here, as above in Example 2
?>
```


Username: root
is connected to the database: retail_db

Query: SELECT * FROM department
... successful

array(6) { [0]=> string(1) "1" ["DepartmentID"]=> string(1) "1" [1]=> string(5) "Sales" ["DeptName"]=> string(5) "Sales" [2]=> string(8) "EH1 3SP " ["Postcode"]=> string(8) "EH1 3SP " }

array(6) { [0]=> string(1) "2" ["DepartmentID"]=> string(1) "2" [1]=> string(9) "Marketing" ["DeptName"]=> string(9) "Marketing" [2]=> string(8) "EH1 3SP " ["Postcode"]=> string(8) "EH1 3SP " }

array(6) { [0]=> string(1) "3" ["DepartmentID"]=> string(1) "3" [1]=> string(7) "Payroll" ["DeptName"]=> string(7) "Payroll" [2]=> string(8) "N1C 4QL " ["Postcode"]=> string(8) "N1C 4QL " }

array(6) { [0]=> string(1) "4" ["DepartmentID"]=> string(1) "4" [1]=> string(9) "ShopFloor" ["DeptName"]=> string(9) "ShopFloor" [2]=> string(7) "N1C 4QL" ["Postcode"]=> string(7) "N1C 4QL" }

Username: root
detached from the database: retail_db

Figure 9: Example 3 output

8.7.4 PHP and MySQL: example 4

```
<html>
<title>Example 4</title>

<?php
// connect to db here, as above in Example 1
?>

<?php
// query db here, as above in Example 2
?>

<body>

<?php
// Display some results from a query using
// ************************************
while ($department  = mysqli_fetch_assoc($query_result)) {
       // output the data per row
       echo $department["DepartmentID"] . "<br />";
       echo $department["DeptName"] . "<br />";
       echo $department["Postcode"] . "<br />";
       //echo "<br> \n";
       echo "<hr />";
}
echo "<br> \n";
?>

</body>
</html>
```

```
<?php
mysqli_free_result ( $query_result );
?>

<?php
// disconnect from db here, as above in Example 2
?>
```

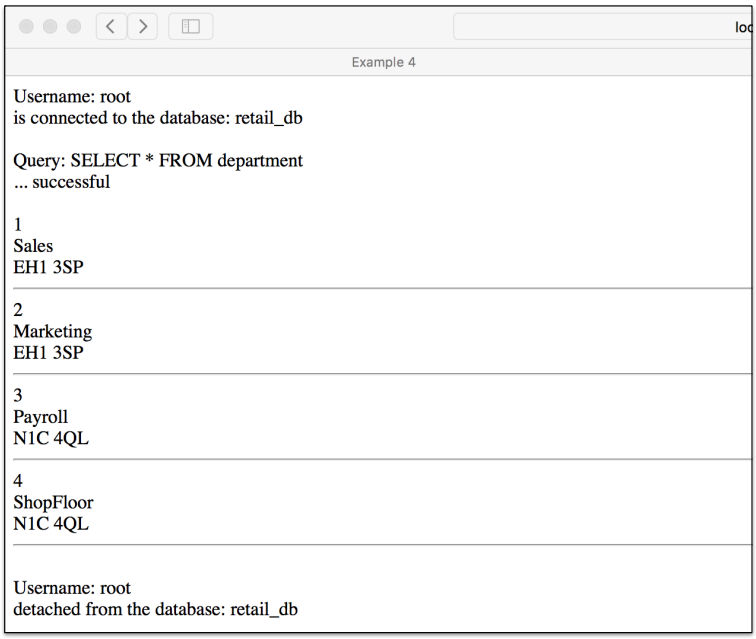


Figure 10: Example 4 output

8.7.5 PHP and MySQL: example 5

```
<html>
<title>Example 5</title>

<?php
// connect to db here, as above
?>

<?php
// query db here, as above
?>

<body>
<?php
echo "<table border='1'>";
while ( $department = mysqli_fetch_assoc ( $query_result ) ) {
        $name = $department ['DepartmentID'];
        $address = $department ['DeptName'];
        $content = $department ['Postcode'];
        echo "<tr><td>" . $name . "</td><td>" . $address . "</td><td>" . $content .
"</td></tr>";
}
echo "</table>";
echo "<br> \n";
?>
```

```
</body>

<?php
mysqli_free_result ( $query_result );
?>

</html>

<?php
// close db connection code here, as above
?>
```

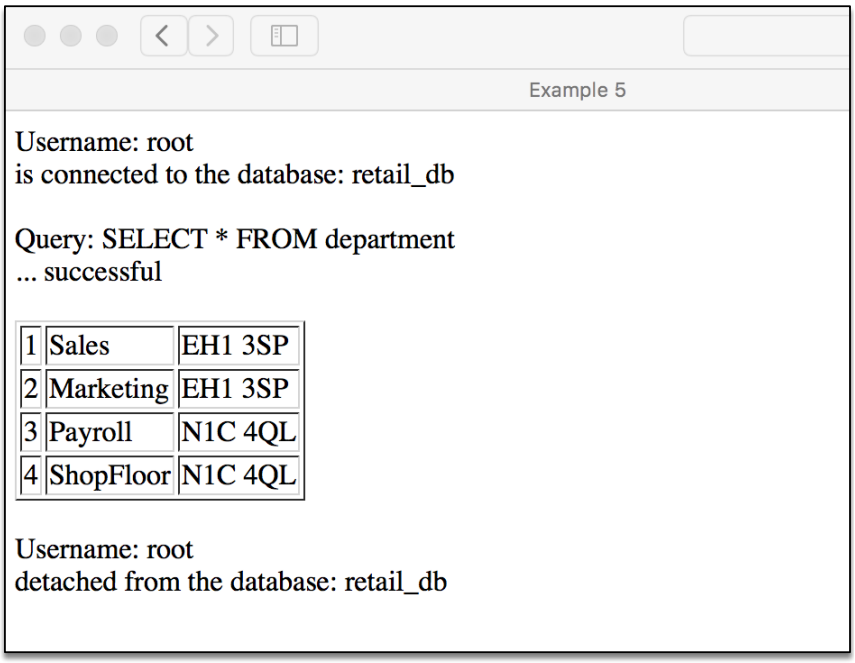


Figure 11: Example 5 output

8.8 Summary

- The PHP programming language was introduced.
- We looked at the structure PHP has an embedded within HTML, the main use case considered on this course.
- We covered the local machine as localhost, web server, etc, as a development environment.
- We described the mysqli Library. Variables and datatypes in PHP were introduced and examples code given.

~~~~~
~~~~~