UNIVERSITY OF CALGARY
DEPARTMENT OF COMPUTER SCIENCE
Fall 2019
CPSC 471: Data Base Management Systems
Prepared By: Tamer N. Jarada

Objectives:

- Be familiar with web application architecture
- Introduction to PHP
- PHP Installation
- Get Started with Basic PHP
- Be familiar with PHP User Defined Functions
- Be familiar with PHP Form Handling
- Get Started with MySQL
- Connect PHP with MySQL Server
- Do INSERT, SELECT, UPDATE and DELETE on MySQL form PHP
- Be familiar with PHP Cookies
- Be familiar with PHP Sessions

Web Application Architecture:

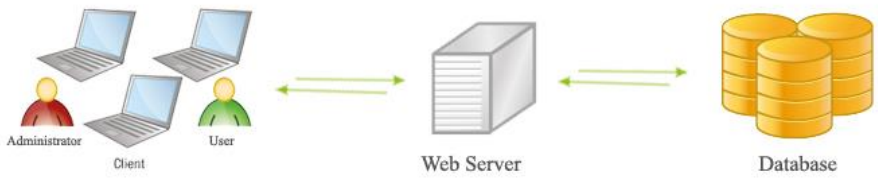

Figure (1): Web Application Architecture

Introduction to PHP:

PHP (Personal Home Page) is a server scripting language, and is a powerful tool for making dynamic and interactive Web pages quickly.

PHP is a widely-used, free, and efficient alternative to competitors such as ASP.Net.

What You Should Already Know

You should have a basic understanding of the following:

- HTML
- CSS
- JavaScript

If you want to study these subjects first, you may see http://www.w3schools.com, http://www.codecademy.com or any free online courses website.

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, and close files on the server
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can restrict users to access some pages on your website
- PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is easy to learn and runs efficiently on the server side
- PHP supports a wide range of databases
- PHP is free.

PHP Installation:

What Do I Need?

To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host with PHP Support

If your server has activated support for PHP you do not need to do anything. Just create some .php files, place them in your web directory, and the server will automatically parse them for you.

You do not need to compile anything or install any extra tools. Because PHP is free, most web hosts offer PHP support.

Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

- Install a web server
- Install PHP
- Install a database, such as MySQL

The official PHP website (PHP.net) has installation instructions for PHP: http://php.net/manual/en/install.php

However, many people know from their own experience that it's not easy to install an Apache web server and it gets harder if you want to add MySQL, PHP. AppServ is one solution for that problem; it is easy to install Apache distribution containing MySQL and PHP, just download (http://www.appservnetwork.com), and install.

Get Started with Basic PHP:

A PHP script can be placed anywhere in the document. A PHP script starts with **<?php** and ends with **?>**

Code

```
<?php
// PHP code goes here
?>
```

The default file extension for PHP files is ".php". A PHP file normally contains HTML tags, and some PHP scripting code.

Hello world example:

Code

```
<html>
<body>

<h1>My first PHP page</h1>
<?php
    echo "Hello World!";
?>
</body>
</html>
```

Comments in PHP

A comment in PHP code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is editing the code!

Comments are useful:

- To let others understand what you are doing - Comments let other programmers understand what you were doing in each step (if you work in a group)
- To remind yourself what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

PHP supports three ways of commenting:

Code

```
<html>
<body>
<?php
// This is a single line comment

# This is also a single line comment

/*
This is a multiple lines comment block
that spans over more than
one line
*/
?>
</body>
</html>
```

PHP Case Sensitivity

In PHP, all user-defined functions, classes, and keywords (e.g. if, else, while, echo, etc.) are **case-insensitive**.

In the example below, all three echo statements below are legal (and equal):

Code

```
<html>
<body>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

However; in PHP, all variables are case-sensitive.

In the example below, only the first statement will display the value of the $color variable (this is because $color, $COLOR, and $coLOR are treated as three different variables):

Code

```
<html>
<body>

<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case sensitive ($y and $Y are two different variables)

Code

```
<html>
<body>

<?php
$txt="Hello world!";
$x=5;
$y=10.5;

echo $txt;
echo "<br>";

echo $x+$y;


?>

</body>
</html>
```

Note: PHP is a Loosely Type Language which means PHP automatically converts the variable to the correct data type, depending on its value. In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

Local and Global Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function.

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function.

The following example tests variables with local and global scope:

Code

```
<html>
<body>

<?php
$x=5; // global scope

function myTest()
{
$y=10; // local scope
echo "<p>Test variables inside the function:</p>";
echo "Variable x is: $x";
echo "<br>";
echo "Variable y is: $y";
}
```

```
myTest();

echo "<p>Test variables outside the function:</p>";
echo "Variable x is: $x";
echo "<br>";
echo "Variable y is: $y";
?>


</body>
</html>
```

In the example above there are two variables $x and $y and a function myTest(). $x is a global variable since it is declared outside the function and $y is a local variable since it is created inside the function.

When we output the values of the two variables inside the myTest() function, it prints the value of $y as it is the locally declared, but cannot print the value of $x since it is created outside the function.

Then, when we output the values of the two variables outside the myTest() function, it prints the value of $x, but cannot print the value of $y since it is a local variable and it is created inside the myTest() function.

PHP The global Keyword

The global keyword is used to access a global variable from within a function.

To do this, use the global keyword before the variables (inside the function):

Code

```
<html>
<body>

<?php
$x=5;
$y=10;

function myTest()
{
global $x,$y;
$y=$x+$y;
}

myTest();
echo $y; // outputs 15
?>


</body>
</html>
```

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the static keyword when you first declare the variable:

Code

```
<html>
<body>

<?php

function myTest()
{
static $x=0;
echo $x;
$x++;
}
myTest();
myTest();
myTest();
?>
</body>
</html>
```

PHP User Defined Functions:

Besides the built-in PHP functions, we can create our own functions.

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

A user defined function declaration starts with the word "function":

Code

```
function functionName()
{
code to be executed;
}
```

Note: A function name can start with a letter or underscore (not a number).

PHP Function with Arguments:

Code

```
<?php
function familyName($fname,$year)
{
echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege","1975");
familyName("Stale","1978");
familyName("Kai Jim","1983");
?>
```

PHP Function with Default Argument Value:

Code

```
<?php
function setHeight($minheight=50)
{
echo "The height is : $minheight <br>";
}
setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

PHP Function with Returning Value:

Code

```
<?php
function sum($x,$y)
{
$z=$x+$y;
return $z;
}
echo "5 + 10 = " . sum(5,10) . "<br>";
echo "7 + 13 = " . sum(7,13) . "<br>";
echo "2 + 4 = " . sum(2,4);
?>
```

PHP Form Handling:

The example below displays a simple HTML form with two input fields and a submit button:

Code

```
<html>
<body>
<form action="welcome.php" method="post">
   Name: <input type="text" name="name"><br>
   E-mail: <input type="text" name="email"><br>
   <input type="submit">
</form>
</body>
</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the **HTTP POST** method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

Code

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?> <br>
Your email address is: <?php echo $_POST["email"]; ?>
</body>
</html>
```

The same result could also be achieved using the **HTTP GET** method:

Code

```
<html>
<body>
<form action="welcome.php" method="get">
   Name: <input type="text" name="name"><br>
   E-mail: <input type="text" name="email"><br>
   <input type="submit">
</form>
</body>
</html>
```

and "welcome.php" looks like this:

Code

```
<html>
<body>

Welcome <?php echo $_GET ["name"]; ?> <br>
Your email address is: <?php echo $_GET ["email"]; ?>

</body>
</html>
```

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

GET vs. POST

Both GET and POST create an array (e.g. array(key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as **$_GET** and **$_POST**. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$_GET is an array of variables passed to the current script via the URL parameters.

$_POST is an array of variables passed to the current script via the HTTP POST method.

When to use GET?

Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL).

GET also has limits on the amount of information to send. The limitation is about 2000 characters.

However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

Information sent from a form with the POST method is invisible to others (all names/values is embedded within the body of the HTTP request) and has no limits on the amount of information to send.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Note: POST is the preferred way by developers to send form data.

Get Started with MySQL:

MySQL is the most popular database system used with PHP.

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform).

Access your MySQL server:

You may access your MySQL server by using different ways, one of these ways is to use MySQL-Front (http://www.mysqlfront.de/)

Connect PHP with MySQL Server:

Before we can access data in a database, we must open a connection to the MySQL server.

In PHP, this is done with the **mysqli_connect()** function.

Code

```
mysqli_connect(host,username,password,dbname);
```

- **host:** Either a host name or an IP address
- **username:** The MySQL user name
- **password:** The password to log in with
- **dbname:** The default database to be used when performing queries

Note: There are more available parameters, but the ones listed above are the most important.

In the following example we store the connection in a variable ($con) for later use in the script:

Code

```
<?php
// Create connection
$con=mysqli_connect("localhost","root","123456","471");
// Check connection
if (mysqli_connect_errno($con))
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }
?>
```

The connection will be closed automatically when the script ends. To close the connection before, use the mysqli_close() function:

Code

```
<?php
mysqli_close($con);
?>
```

Do INSERT, SELECT, UPDATE and DELETE on MySQL form PHP:

Insert Data into a Database Table:

The INSERT INTO statement is used to add new records to a database table. It is possible to write the INSERT INTO statement in two forms.

The first form doesn't specify the column names where the data will be inserted, only their values:

Code

```
INSERT INTO table_name VALUES (value1, value2, value3,...)
```

The second form specifies both the column names and the values to be inserted:

Code

```
INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2,
value3,...)
```

The following example adds new record to a table.

To get PHP to execute the statements above we must use the mysqli_query() function. This function is used to send a query or command to a MySQL connection:

Code

```
<?php
$name = $_POST["name"];
$email = $_POST["email"];
echo $name. "<br>". $email. "<br>";

// Create connection
$con=mysqli_connect("localhost","root","123456","471");

// Check connection
if (mysqli_connect_errno($con))
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }
   $sql = "INSERT INTO users (Name, Email) VALUES ('". $name."','". $email
."')";
  if (!mysqli_query($con,$sql))
  {
  die('Error: ' . mysqli_error($con));
  }
else
   echo "1 record added";

mysqli_close($con);
?>
```

Select Data from a Database Table:

The SELECT statement is used to select data from a database.

Code

```
SELECT column_name(s)FROM table_name
```

The following example selects all the data from a table and displays it in a table:

Code

```
<?php
// Create connection
$con=mysqli_connect("localhost","root","123456","471");

// Check connection
if (mysqli_connect_errno($con))
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }


$result = mysqli_query($con, "SELECT * FROM Users");

echo "<table border='1'>
<tr>
<th>ID</th>
<th>Name</th>
<th>Email</th>
</tr>";

while($row = mysqli_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>" . $row['ID'] . "</td>";
  echo "<td>" . $row['Name'] . "</td>";
  echo "<td>" . $row['Email'] . "</td>";
  echo "</tr>";
  }
echo "</table>";

mysqli_close($con);
?>
```

Update Data in a Database:

The UPDATE statement is used to update existing records in a table.

Code

```
UPDATE table_name SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Note: Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

The following example updates some data in a table:

Code

```
<?php
// Create connection
$con=mysqli_connect("localhost","root","123456","471");


// Check connection
if (mysqli_connect_errno($con))
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }

$sql = "update users set name='".$name. "', email='". $email. "' where ID=".
$ID;

$result = mysqli_query($con, $sql);

mysqli_close($con);

?>
```

Delete Data in a Database

The DELETE FROM statement is used to delete records from a database table.

Code

```
DELETE FROM table_name
WHERE some_column = some_value
```

Note: Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

The following example deletes some data in a table:

Code

```
<?php
// Create connection
$con=mysqli_connect("localhost","root","123456","471");

// Check connection
if (mysqli_connect_errno($con))
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }

$result = mysqli_query($con,"Delete from Users where ID=". $ID );

mysqli_close($con);

?>
```

PHP Cookies:

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

The setcookie() function is used to set a cookie.

Code

```
setcookie(name, value, expire, path, domain);
```

Note: The setcookie() function must appear BEFORE the <html> tag.

Creating a Cookie:

In the example below, we will create a cookie named "user" and assign the value "Alex Porter" to it. We also specify that the cookie should expire after one hour:

Code

```
<?php
setcookie("user", "Alex Porter", time()+3600);
?>

<html>

…
```

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

Retrieving a Cookie Value:

In the example below, we retrieve the value of the cookie named "user" and display it on a page:

Code

```
<html>
<body>

<?php
if (isset($_COOKIE["user"]))
  echo "Welcome " . $_COOKIE["user"] . "!<br>";
else
  echo "Welcome guest!<br>";
?>

</body>
</html>
```

Deleting a Cookie:

When deleting a cookie you should assure that the expiration date is in the past.

Code

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time()-3600);
?>
```

PHP Sessions:

A session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.

Starting a Session:

Before you can store user information in your PHP session, you must first start up the session.

Note: The session_start() function must appear BEFORE the <html> tag:

Code

```
<?php session_start(); ?>

<html>
<body>

</body>
</html>
```

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

Storing a Session Variable:

The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:

Code

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>

<html>
<body>

<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?>

</body>
</html>
```

If you wish to delete some session data, you can use the unset() or the session_destroy() function.

The unset() function is used to free the specified session variable:

Code

```
<?php
session_start();
if(isset($_SESSION['views']))
  unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the session_destroy() function:

Code

```
<?php
session_destroy();
?>
```

Note: session_destroy() will reset your session and you will lose all your stored session data.