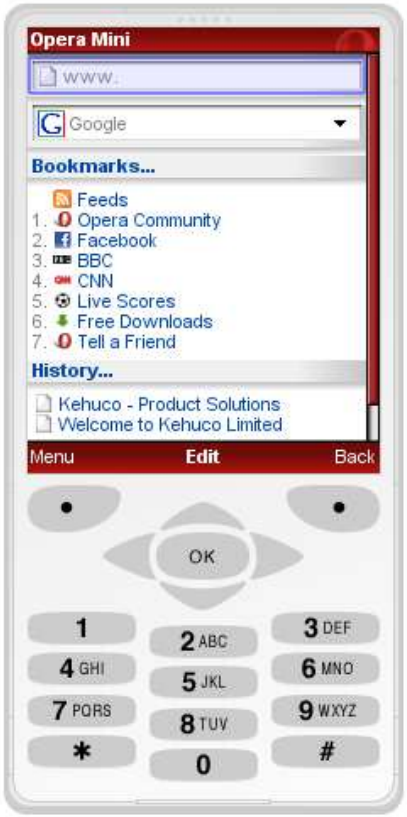
PHP and MYSQL



You Learn By Doing

eMobilis
Mobile Technology Training

Course Objectives

This module will benefit the students who are looking for ways to quickly build robust, cross platform mobile applications.

The students will learn how to;

Develop web content for mobile phones and Desktop PC using PHP and MySQL

Introduction
What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

Introduction Continued

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Where to Start?

- To get access to a web server with PHP support, you can:
- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support

Basic PHP Syntax

A PHP scripting block always starts with

<?php and ends with **?>**.

A PHP scripting block can be placed anywhere in the document.

On servers with shorthand support enabled you can start a scripting block with <? and end with ?>.

For maximum compatibility, we recommend that you use the standard form (<?php) rather than the shorthand form.

```
<?php


?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>
</html>
```

Comments in PHP

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

Variables in PHP

Variables are used for storing a values, like text strings, numbers or arrays.

When a variable is declared, it can be used over and over again in your script.

All variables in PHP start with a $ sign symbol.

The correct way of declaring a variable in PHP:

```
$var_name = value;
```

New PHP programmers often forget the $ sign at the beginning of the variable. In that case it will not work.

Let's try creating a variable containing a string, and a variable containing a number:

```
<?php
    $txt="Hello World!";
    $x=16;
    ?>
```

String Variables in PHP

- String variables are used for values that contains characters.
- After we create a string we can manipulate it. A string can be used directly in a function or it can be stored in a variable.
- Below, the PHP script assigns the text "Hello World" to a string variable called $txt:

```
<?php
$txt="Hello World";
echo $txt;
?>
```

The output of the code above will be:

Hello World

- Now, lets try to use some different functions and operators to manipulate the string.

The Concatenation Operator

- There is only one string operator in PHP.
- The concatenation operator (.) is used to put two string values together.
- To concatenate two string variables together, use the concatenation operator:

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

The output of the code above will be:

Hello World! What a nice day!

- If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

The strlen() function

The strlen() function is used to return the length of a string.

Let's find the length of a string:

```
<?php
echo strlen("Hello world!");
?>
```

The output of the code above will be:

12

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string).

The strpos() function

The strpos() function is used to search for character within a string.

If a match is found, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string:

```
<?php
echo strpos("Hello world!","world");
?>
```

The output of the code above will be:

6

The position of the string "world" in our string is position 6. The reason that it is 6 (and not 7), is that the first position in the string is 0, and not 1.

PHP Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

Conditional statements

Conditional statements are used to perform different actions based on different conditions.

- **Conditional Statements**
- Very often when you write code, you want to perform different actions for different decisions.
- You can use conditional statements in your code to do this.
- In PHP we have the following conditional statements:
- **if statement -** use this statement to execute some code only if a specified condition is true
- **if...else statement -** use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else statement -** use this statement to select one of several blocks of code to be executed
- **switch statement -** use this statement to select one of many blocks of code to be executed

The if Statement

- Use the if statement to execute some code only if a specified condition is true.
- **Syntax**

if (*condition*) *code to be executed if condition is true;* The following example will output "Have a nice weekend!" if the current day is Friday:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

The if...else Statement

- Use the if....else statement to execute some code if a condition is true and another code if a condition is false.
- **Syntax**
- if (*condition*)

  code to be executed if condition is true;

  else

  code to be executed if condition is false;

Example

- The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

Switch

- Conditional statements are used to perform different actions based on different conditions.
- **The PHP Switch Statement**
- Use the switch statement to select one of many blocks of code to be executed.
- **Syntax**
- switch (*n*)
  {
  case *label1:*
  code to be executed if n=label1;
  break;
  case *label2:*
  code to be executed if n=label2;
  break;
  default:
  code to be executed if n is different from both label1 and label2;
  }

Example

```
<html>
<body>

<?php
$x=1;
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>

</body>
</html>
```

While loop

- **PHP Loops**
- Often when you write code, you want the same block of code to run over and over again in a row.
- In PHP, we have the following looping statements:
- **while** - loops through a block of code while a specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as a specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

Syntax

- **The while Loop**
- The while loop executes a block of code while a condition is true.
- **Syntax**
- while (*condition*)

  {

  code to be executed;

  }

Example

- The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i++;
  }
?>

</body>
</html>
```

The do...while Statement

- The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true.
- **Syntax**
- do
  {
  code to be executed;
  }
- while (*condition*);

Example

- The example below defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

```
<html>
<body>

<?php
$i=1;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<=5);
?>

</body>
</html>
```

The for Loop

The for loop is used when you know in advance how many times the script should run.

- **Syntax**
- for (*init; condition; increment*)

  {

  code to be executed;

  }

Parameters:

- *init*: Mostly used to set a counter
- *condition*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment*: Mostly used to increment a counter

Example

- The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:
-

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
  {
  echo "The number is " . $i . "<br />";
  }
?>

</body>
</html>
```

PHP Arrays

- An array stores multiple values in one single variable.
- **What is an Array?**
- A variable is a storage area holding a number or text. The problem is, a variable will hold only one value.
- An array is a special variable, which can store multiple values in one single variable.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:
- $cars1="Saab";
  $cars2="Volvo";
  $cars3="BMW";

Example

- In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] ." and " . $cars[1] ." are
Swedish cars.";
?>
```

Research and Assignment

Student to Read on their own the following Topics

- PHP Arrays

Assignment:

- Using if…else…if and comparison operator,s create a grading system that will give a student his/her grade (A, B+, B, C+, C, D+, D, Fail) according the marks scored

Create a PHP Function

- A function will be executed by a call to the function.
- **Syntax**
- function *functionName*()
  {
  code to be executed;
  }
- PHP function guidelines:
- Give the function a name that reflects what the function does
- The function name can start with a letter or underscore (not a number)

Example

- A simple function that writes my name when it is called:

```
<html>
<body>

<?php
function writeName()
{
echo “John Mutai";
}

echo "My name is ";
writeName();
?>

</body>
</html>
```

PHP Functions - Adding parameters

- To add more functionality to a function, we can add parameters. A parameter is just like a variable.
- Parameters are specified after the function name, inside the parentheses

Example

- The following example will write different first names, but equal last name:

```
<html>
<body>

<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

PHP Functions - Return values

- To let a function return a value, use the return statement.

Example

- ```
  <html>
  <body>

  <?php
  function add($x,$y)
  {
  $total=$x+$y;
  return $total;
  }

  echo "1 + 16 = " . add(1,16);
  ?>

  </body>
  </html>
  ```

Output: 1 + 16 = 17
  ```

PHP Forms and User Input

- The PHP $_GET and $_POST variables are used to retrieve information from forms, like user input.
- **PHP Form Handling**
- The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will **automatically** be available to your PHP scripts.

Example

- The example below contains an HTML form with two input fields and a submit button:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>

</body>
</html>
```

Save file as “student.html”

When a user fills out the form above and click on the submit button, the form data is sent to a PHP file, called "welcome.php":

"welcome.php" looks like this:

- <html>
  <body>

  Welcome <?php echo $_POST["fname"];
  ?>!
  You are <?php echo $_POST["age"]; ?> years
  old.

  </body>
  </html>

PHP $_GET Function

- The built-in $_GET function is used to collect values in a form with method="get".
- **The $_GET Function**
- The built-in $_GET function is used to collect values from a form sent with method="get".
- Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send (max. 100 characters)

Example

- <form action="welcome.php"
  method="get">
  Name: <input type="text" name="fname"
  />
  Age: <input type="text" name="age" />
  <input type="submit" />
  </form>

Explanation

- The "welcome.php" file can now use the $_GET function to collect form data (the names of the form fields will automatically be the keys in the $_GET array):
- Welcome <?php echo $_GET["fname"]; ?>.
 You are <?php echo $_GET["age"]; ?> years old!

- **When to use method="get"?**
- When using method="get" in HTML forms, all variable names and values are displayed in the URL.
- **Note:** This method should not be used when sending passwords or other sensitive information!

PHP $_POST Function

- The built-in $_POST function is used to collect values in a form with method="post".
- **The $_POST Function**
- The built-in $_POST function is used to collect values from a form sent with method="post".
- Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
- **Note:** However, there is an 8 Mb max size for the POST method, by default (can be changed by setting the post_max_size in the php.ini file).

Example

- <form action="welcome.php"
  method="post">
  Name: <input type="text" name="fname"
  />
  Age: <input type="text" name="age" />
  <input type="submit" />
  </form>

Explanation

- The "welcome.php" file can now use the $_POST function to collect form data (the names of the form fields will automatically be the keys in the $_POST array):
- Welcome <?php echo $_POST["fname"]; ?>!
 You are <?php echo $_POST["age"]; ?> years old.

- **When to use method="post"?**
- Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
- However, because the variables are not displayed

The PHP $_REQUEST Function

- The PHP built-in $_REQUEST function contains the contents of both $_GET, $_POST, and $_COOKIE.
- The $_REQUEST function can be used to collect form data sent with both the GET and POST methods.

Example

- **Welcome**

```
<?php echo $_REQUEST["fname"]; ?>!
<br />
You are <?php echo $_REQUEST["age"]; ?>
years old.
```

Research and Assignment

Student to Read on their own the following Topics

- Read Further on PHP $GET and $POST

Assignment:

- Using $GET or $POST create a Student Registration Form called "registration.html" that's contains (Surname, Firstname, Dateofbirth, Gender, Telephone/Mobile, Address, City, Email) and on clicking Submit, all the details entered are displayed in a PHP file called "studentrecord.php

The PHP Date() Function

- The PHP date() function formats a timestamp to a more readable date and time.
- A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.
- **Syntax**
- date(*format*,*timestamp*)
- timestamptimestampOptional. Specifies a timestamp. Default is the current date and time

PHP Date() - Format the Date

- The required *format* parameter in the date() function specifies how to format the date/time.
- Here are some characters that can be used:
- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)

- ```
  <?php
  echo date("Y/m/d") . "<br />";
  echo date("Y.m.d") . "<br />";
  echo date("Y-m-d")
  ?>
  ```
  ```

PHP File Handling

- The fopen() function is used to open files in PHP.

Opening a File

- The fopen() function is used to open files in PHP.
- The first parameter of this function contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened:
- <html>
  <body>

  <?php
  $file=fopen(“emobilis.txt","r");
  ?>

  </body>
  </html>

The file may be opened in one of the following modes:

Modes	Description
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

Fopen()

- **Note:** If the fopen() function is unable to open the specified file, it returns 0 (false).
- **Example**
- The following example generates a message if the fopen() function is unable to open the specified file:
-
```
<html>
<body>

<?php
$file=fopen("emobilis.txt","r") or exit("Unable to open file!");
?>

</body>
</html>
```

Closing a File

- The fclose() function is used to close an open file:
- <?php
  $file = fopen("emobilis.txt","r");

  //some code to be executed

  fclose($file);
  ?>

Check End-of-file

- The feof() function checks if the "end-of-file" (EOF) has been reached.

  The feof() function is useful for looping through data of unknown length.
- **Note:** You cannot read from files opened in w, a, and x mode!
- if (feof($file)) echo "End of file";

Reading a File Line by Line

- The fgets() function is used to read a single line from a file.
- **Note:** After a call to this function the file pointer has moved to the next line.

Example

- The example below reads a file line by line, until the end of file is reached:
- ```
  <?php
  $file = fopen("emobilis.txt", "r") or exit("Unable to open file!");
  //Output a line of the file until the end is reached
  while(!feof($file))
    {
    echo fgets($file). "<br />";
    }
  fclose($file);
  ?>
  ```
  ```

Reading a File Character by Character

- The fgetc() function is used to read a single character from a file.
- **Note:** After a call to this function the file pointer moves to the next character.

- **Example**

- The example below reads a file character by character, until the end of file is reached:
- ```
  <?php
  $file=fopen("emobilis.txt","r") or exit("Unable to open file!");
  while (!feof($file))
    {
    echo fgetc($file);
    }
  fclose($file);
  ?>
  ```
  ```

PHP Uploading Images

- With PHP, it is possible to upload files to the server.
- **Create an Upload-File Form**
- To allow users to upload files from a form can be very useful.
- Look at the following HTML form for uploading files:

```
<html>
<head><title>Upload Images</title></head>
<body>
<form enctype="multipart/form-data" action="upload.php"
method="POST">
<input type="hidden" name="MAX_FILE_SIZE" value="10000000" />
Choose file to upload: <input name="uploadedfile" type="file" /><br />
<input type="submit" value="Upload File" />
</form>
</body>
</html>
```

Save file as "fileupload.html"

Create The Upload Script

Create folder called "upload" in the same location with **"fileupload.html" and "upload.hphp"**

```
<?php

$target_path = "upload/";

$target_path = $target_path . basename( $_FILES['uploadedfile']['name']);

if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path)) {
    echo "The file ".  basename( $_FILES['uploadedfile']['name']).
    " has been uploaded";
} else{
    echo "There was an error uploading the file, please try again!";
}
?>
```

Save file as "upload.php"

Upload to server

- By using the global PHP $_FILES array you can upload files from a client computer to the remote server.
- The first parameter is the form's input name and the second index can be either "name", "type", "size", "tmp_name" or "error". Like this:
- $_FILES["file"]["name"] - the name of the uploaded file
- $_FILES["file"]["type"] - the type of the uploaded file
- $_FILES["file"]["size"] - the size in bytes of the uploaded file
- $_FILES["file"]["tmp_name"] - the name of the temporary copy of the file stored on the server
- $_FILES["file"]["error"] - the error code resulting from the file upload
- This is a very simple way of uploading files. For security reasons, you should add restrictions on what the user is allowed to upload.

PHP Sessions

- A PHP 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 PHP 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:
- <?php session_start(); ?>

```
<html>
<body>

</body>
</html>
```

Storing a Session Variable

- The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:
-
```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>

<html>
<body>

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

</body>
</html
```

Example

- In the example below, we create a simple page-views counter. The isset() function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1:
- <?php
session_start();

if(isset($_SESSION['views']))
$_SESSION['views']=$_SESSION['views']+1;
else
$_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>

Destroying a Session

- 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:
- <?php
  unset($_SESSION['views']);
  ?> You can also completely destroy the session by calling the session_destroy() function:
- <?php
  session_destroy();
  ?>

PHP Sending E-mails

- PHP allows you to send e-mails directly from a script.
- **The PHP mail() Function**

PHP Simple E-Mail

- The simplest way to send an email with PHP is to send a text email.
- In the example below we first declare the variables ($to, $subject, $message, $from, $headers), then we use the variables in the mail() function to send an e-mail:
- <?php
$to = "someone@example.com";
$subject = "Test mail";
$message = "Hello! This is a simple email message.";
$from = "someonelse@example.com";
$headers = "From: $from";
mail($to,$subject,$message,$headers);
echo "Mail Sent.";
?>

Save file as "email.php"

PHP E-Mail Example

Name:	
Address:	
Tel/Mobile:	
Email:	
Company:	
Position:	
Comment:	

Send Clear

Using html form to send data using mail function

PHP E-Mail Example: HTML

```
<html>
<head><title>contact</title></head>

<form name="form1" method="post" action="‘mailsend.php">
    <table border="0">
      <tr>
        <td bgcolor="#FFCC99"><b>Name:</b></td>
        <td><input name="Name" type="text" size="40"></td>
      </tr>
      <tr>
        <td bgcolor="#FFCC99"><b>Address:</b></td>
        <td><input name="Address" type="text" size="40"></td>
      </tr>
      <tr>
        <td bgcolor="#FFCC99"><b>Tel/Mobile:</b></td>
        <td><input name="Tel" type="text" size="40" ></td>
      </tr>
      <tr>
        <td bgcolor="#FFCC99"><b>Email:</b></td>
        <td><input name="Email" type="text" size="40"></td>
      </tr>
```

```
<tr>
        <td bgcolor="#FFCC99"><b>Company:</b></td>
        <td><input name="Company" type="text" size="40"></td>
      </tr>
      <tr>
        <td bgcolor="#FFCC99"><b>Position:</b></td>
        <td><input name="Position" type="text" size="40"></td>
      </tr>
    <tr>
        <td valign="top" bgcolor="#FFCC99"><b>Comment:</b></td>
          <td><textarea name="MsgBody" cols="35" rows="10"></textarea></td>
    </tr>
      <tr>
        <td> </td>
        <td><input type="submit" name="Submit"
          value="Send"> <input type="reset" name="Reset"
          value="Clear"></td>
      </tr>
      <tr>
        <td> </td>
      </tr>
    </table>
  </form>
</body>
</html>
```

Save file as "formemail.html"

PHP E-Mail Example: PHP

```
<?php
$recipient = ‘email@email.com'; //Replace with your email address here
            $address = $_POST['Address'];
            $tel = $_POST['Tel'];
            $email = $_POST['Email'];
            $company = $_POST['Company'];
            $position = $_POST['Position'];

    $subject = "WebEnquires";
    $from = $_POST['Name'];
    $msg = "Message from: $from\n\n Address: $address\n\n Tel: $tel\n\n Email:
$email\n\n Company: $company\n\n Position: $position\n\n".$_POST['MsgBody'];
    if (mail($recipient, $subject, $msg))
                    echo "Thank You. Your Comment has been sent Successfully";
    else
      echo "Comment failed to send";

?>
```

Save file as “mailsend.php”

Research and Assignment

Student to Read on their own the following Topics

- PHP Error Handling
- PHP Sessions and PHP Cookies

Assignment:

- Submit a simple example showing how PHP Sessions and PHP Cookies are implemented

PHP Error Handling

- The default error handling in PHP is very simple. An error message with filename, line number and a message describing the error is sent to the browser.
- **PHP Error Handling**
- When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

Basic Error Handling: Using the die() function

- The first example shows a simple script that opens a text file:
- <?php
  $file=fopen("welcome.txt","r");
  ?> If the file does not exist you might get an error like this:
- **Warning**:
- fopen(welcome.txt) [function.fopen]: failed to open stream:no such file or directory in **C:\webfolder\test.php** on line **2**

Error handling

- To avoid that the user gets an error message like the one above, we test if the file exist before we try to access it:
-
```
<?php
if(!file_exists("welcome.txt"))
  {
  die("File not found");
  }
else
  {
  $file=fopen("welcome.txt","r");
  }
?>
```

PHP Exception Handling

- Exceptions are used to change the normal flow of a script if a specified error occurs

Basic Use of Exceptions

- When an exception is thrown, the code following it will not be executed, and PHP will try to find the matching "catch" block.
- If an exception is not caught, a fatal error will be issued with an "Uncaught Exception" message.

Example

```
<?php
//create function with an exception
function checkNum($number)
  {
  if($number>1)
    {
    throw new Exception("Value must be 1 or below");
    }
  return true;
  }

//trigger exception
checkNum(2);
?>
```

PHP Filter

- PHP filters are used to validate and filter data coming from insecure sources, like user input.
- **What is a PHP Filter?**
- A PHP filter is used to validate and filter data coming from insecure sources.
- To test, validate and filter user input or custom data is an important part of any web application.
- The PHP filter extension is designed to make data filtering easier and quicker.

Functions and Filters

- To filter a variable, use one of the following filter functions:
- filter_var() - Filters a single variable with a specified filter
- filter_var_array() - Filter several variables with the same or different filters
- filter_input - Get one input variable and filter it
- filter_input_array - Get several input variables and filter them with the same or different filters

Example

```
<?php
$int = 123;

if(!filter_var($int, FILTER_VALIDATE_INT))
  {
  echo("Integer is not valid");
  }
else
  {
  echo("Integer is valid");
  }
?>
```

Options and Flags

- Options and flags are used to add additional filtering options to the specified filters.
- Different filters have different options and flags.
- In the example below, we validate an integer using the filter_var() and the "min_range" and "max_range" options:

```
<?php
$var=300;

$int_options = array(
"options"=>array
  (
  "min_range"=>0,
  "max_range"=>256
  )
);

if(!filter_var($var, FILTER_VALIDATE_INT, $int_options))
  {
  echo("Integer is not valid");
  }
else
  {
  echo("Integer is valid");
  }
?>
```

Research and Assignment

Student to Read on their own the following Topics

- PHP Error Handling
- PHP Sessions and PHP Cookies

Assignment:

- Submit a simple example showing how PHP Sessions and PHP Cookies are implemented

Introduction

- **What is MySQL?**
  - MySQL is a database server
  - MySQL is ideal for both small and large applications
  - MySQL supports standard SQL
  - MySQL compiles on a number of platforms
  - MySQL is free to download and use

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

PHP MySQL Introduction

What is MySQL?

- MySQL is the most popular open-source database system.
- The data in MySQL is stored in database objects called tables.
- A table is a collections of related data entries and it consists of columns and rows.
- Databases are useful when storing information categorically. A company may have a database with the following tables: "Employees", "Products", "Customers" and "Orders".

Database Tables

Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.

Below is an example of a table called "Persons":

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

Queries

- A query is a question or a request.
- With MySQL, we can query a database for specific information and have a recordset returned.
- query
- SELECT LastName FROM Persons
- The query above selects all the data in the "LastName" column from the "Persons" table,

Connect to Database

- **PHP MySQL Connect to a Database**
- **Create a Connection to a MySQL Database**
- Before you can access data in a database, you must create a connection to the database.
- This is done with the mysql_connect() function.
- **Syntax**
- mysql_connect(servername,username,password);

Example

- In the following example we store the connection in a variable ($con) for later use in the script. The "die" part will be executed if the connection fails:
- <?php
$con = mysql_connect("localhost",“root","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

// some code
?>

Closing a Connection

- The connection will be closed automatically when the script ends. To close the connection before, use the mysql_close() function:
-
```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

// some code

mysql_close($con);
?>
```

Create Database

- **PHP MySQL Create Database and Tables**
- A database holds one or multiple tables.
- **Create a Database**
- The CREATE DATABASE statement is used to create a database in MySQL.
- **Syntax**
- CREATE DATABASE database_name

Example

- The following example creates a database called "my_db":

- ```
  <?php
  $con = mysql_connect("localhost","peter","abc123");
  if (!$con)
    {
    die('Could not connect: ' . mysql_error());
    }

  if (mysql_query("CREATE DATABASE my_db",$con))
    {
    echo "Database created";
    }
  else
    {
    echo "Error creating database: " . mysql_error();
    }

  mysql_close($con);
  ?>
  ```
  ```

Insert

- **PHP MySQL Insert Into**
- The INSERT INTO statement is used to insert new records in a table.
- **Insert Data Into a Database Table**
- The INSERT INTO statement is used to add new records to a database table.
- **Syntax**
- 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:

Syntax of insert

- INSERT INTO table_name
  VALUES (value1, value2, value3,...) The second form specifies both the column names and the values to be inserted:
- INSERT INTO table_name (column1, column2, column3,...)
  VALUES (value1, value2, value3,...)

Research and Assignment

Student to Read on their own the following Topics

Assignment:

- Create Database named eMobilis with three tables namely student records, parentRecords and college assets. Using SQL commands add a list of 10 students to the database

Insert Data From a Form Into a Database

- Now we will create an HTML form that can be used to add new records to the "Persons" table.

- Here is the HTML form:
- <html>
<body>

<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname" />
Lastname: <input type="text" name="lastname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>

</body>

PHP MySQL Select

- The SELECT statement is used to select data from a database.
- **Select Data From a Database Table**
- **Syntax**
- SELECT column_name(s)
  FROM table_name

Example

- The following example selects all the data stored in the "Persons" table (The * character selects all the data in the table):

- ```
  <?php
  $con = mysql_connect("localhost","peter","abc123");
  if (!$con)
    {
    die('Could not connect: ' . mysql_error());
    }

  mysql_select_db("my_db", $con);

  $result = mysql_query("SELECT * FROM Persons");

  while($row = mysql_fetch_array($result))
    {
    echo $row['FirstName'] . " " . $row['LastName'];
    echo "<br />";
    }

  mysql_close($con);
  ?>
  ```
  ```

Display the Result in an HTML Table

- The following example selects the same data as the example above, but will display the data in an HTML table:
-
```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons");

echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>";

while($row = mysql_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>" . $row['FirstName'] . "</td>";
  echo "<td>" . $row['LastName'] . "</td>";
  echo "</tr>";
  }
echo "</table>";

mysql_close($con);
?>
```

Where clause

- **PHP MySQL The Where Clause**
- The WHERE clause is used to filter records.
- **The WHERE clause**
- The WHERE clause is used to extract only those records that fulfill a specified criterion.
- **Syntax**
- SELECT column_name(s)
  FROM table_name
  WHERE column_name operator value

Example

- The following example selects all rows from the "Persons" table where "FirstName='Peter':
-
```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons
WHERE FirstName='Peter'");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }
?>
```

update

- **PHP MySQL Update**
- The UPDATE statement is used to modify data in a table.
- **Update Data In a Database**
- The UPDATE statement is used to update existing records in a table.
- **Syntax**
- 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!

Example

- The following example updates some data in the "Persons" table:
- <?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

mysql_query("UPDATE Persons SET Age = '36'
WHERE FirstName = 'Peter' AND LastName = 'Griffin'");

mysql_close($con);
?>

Delete

- The DELETE statement is used to delete records in a table.
- **Delete Data In a Database**
- The DELETE FROM statement is used to delete records from a database table.
- **Syntax**
- DELETE FROM table_name
  WHERE some_column = some_value

Example of delete

- The following example deletes all the records in the "Persons" table where LastName='Griffin':
- <?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

mysql_query("DELETE FROM Persons WHERE
LastName='Griffin'");

mysql_close($con);
?>

Reference:

- HTML & XHTML: The Definitive Guide, 5th Edition By Chuck Musciano and Bill Kennedy (O'Reilly)
- www.google.com search Topic PHP and MySQL Tutorials