PHP

Hypertext Preprocessor

Introduction

- PHP stands for Hypertext Preprocessor.
- PHP is a scripting language that is embedded with the HTML page.
- It is a server side scripting language.
- A PHP code inside HTML page starts with **<?php** tag and ends with **?>** tag

A simple program

```
<html>
   <head>
    <title>hello world</title>
   </head>

   <body>
    <?php
        print(“hello world”);
    ?>
   </body>
</html>
```

PHP comments

- Comments can be applied same as in C and C++.

```
<html>  <head>  <title>Comments</title>  </head>
   <body>
    <?php
        /*
            We can print today's date using PHP
        */
        //print (“hello”);
        print("hello world");
    ?>
</body>
</html>
```

Variables

- Whenever a variable is encountered for the first time, a memory space is set aside for the contents.
- You do not need to specify the data types for the variables.
- In PHP, all variables are prefaced with the ‘$’ sign.

Example: Variables

l = A full textual representation of the day of the week	*Sunday* through *Sat urday*
F = A full textual representation of a month, such as January or March	*January* through *D ecember*
d = Day of the month, 2 digits with leading zeros	*01* to *31*
Y = A full numeric representation of a year, 4 digits	Examples: *1999* or *2003*

```
<html>
<head>
      <title>Variables</title>
</head>
<body>
<?php
      $Name = "A D";
      $Surname = "Patel";
      $id_no= "N22";
      $todaysDate = date("l F d, Y");

      print("Name is ".$Name );
      print("<br> Surname is ".$Surname);
      print("<br> Id is ".$id_no );
      print("<br> Date is ".$todaysDate);
?>
</body>
</html>
```

Send and Receive

```
<html> <head> <title>form</title>   </head>
<body>      <form action="retrieve.php" method="post">
        Name: <input type="text" name="aName"><br>
        Age: <input type="text" name="anAge"><br>
        Date of Birth: <input type="text" name="dob"><br>
        <input type="submit" value="Send Data">
</form> </body> </html>
---------------------------------------------------------------------
<html> <body>
<?php
print("Name is ".$_REQUEST['aName']."<br>");
print("Age is ".$_REQUEST['anAge']."<br>");
print("Birth date is ".$_REQUEST['dob']);
?>
</body> </html>
```

Decisions

L = Whether it's a leap year

1 if it is a leap year, 0 otherwise.

```
<html>    <body>
<?php
    $Today = date("l F d, Y");
    print("Today is $Today");
    $Today = date("L");
    if($Today == 1)
        print("<br>This year is a leap year!");
    else
        print("<br>This year is not a leap year");
?>
</body> </html>
```

Operators for Decisions

Operator	Operation Performed	Example
<	Less than	$num < 12
>	Greater than	$num > 12
<=	Less than equal to	$num <= 12
>=	Greater than equal to	$num >= 12
==	Equal to	$num == 12
!=	Not Equal to	$num != 12
AND, &&	Logical And	$num1 AND $num2 $num1 && $num2
OR, ||	Logical Or	$num1 OR $num2 $num1 || $num2
XOR	Exclusive OR	$num1 XOR $num2
!	Not	!$num

Switch Case

```
<html>
<body>
<?php
$Today = date("l F d, Y");
print("Today is $Today, <br>\n");
$diaryDate = date("d");
switch($diaryDate)
{
    case 03 : print("meeting");
     break;
    case 10 : print("appointment ");
     break;
    case 23 : print("club");
           break;

case 25 : print("conference");
           b
```

d	Day of the month, 2 digits with leading zeros	*01* to *31*

For Loop

```
<html>
<body>
<h1>I must learn my 7 times table</h1><br>
    <?php
        for($count = 1; $count <= 10; $count++)
        {
            print("7 * $count =".(7*$count)."<br>");
        }
    ?>
</body>
</html>
```

While loop

```
<html>
<body>
<h1>I must learn my 7 times table</h1><br>
      <?php
      $count=1;
      while( $count<=10 )
      {
       print("7 * $count =".(7*$count)."<br>");
       $count++;
      }
      ?>
</body>
</html>
```

do-while loop

```
<html>
<body>
<h1>I must learn my 7 times table</h1><br>
      <?php
      $count=1;
      do
      {
       print("7 * $count =".(7*$count)."<br>");
       $count++;
      }while( $count<=10 )
      ?>
</body>
</html>
```

break and continue

```
<html> <body>
    <?php
        for ($i = 0; $i <= 5; $i++)
        {
            if ($i == 2)
             continue;
            print "$i<br>";
            if($i==4)
             break;
        }
    ?>
</body> </html>
```

Arrays

Arrays are the variables that can hold many values under a single name

```
<html>
<body>
<?php
    $myarray = array();
    $myarray[0] = "This";
    $myarray[1] = " is ";
    $myarray[2] = " my array";
    echo($myarray[0].$myarray[1].$myarray[2]);
?>
</body>
</html>
```

```
$myarray = array(“This”, “is”, “myarray”);
$myarray = array(“This”, 1, 2.5);
$total = myarray[1]+myarray[2];

$preference= array(“red”, “white”, “blue”);
$preference[1]=“green”;
```

Two-dimensional Arrays

```
<html> <body>
<?php
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
</body> </html>
```

OUTPUT:

```
Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.
```

Arrays with loop

```
<html><body> <ol>
    <?php
    $preferences = array ("red", "white", "blue",
    "silver", "aqua", "cyan", "yellow");
    echo("The current preferences are");
    foreach($preferences as $value)
    {
     echo("<li>This preference is: $value </li>");
    }
    ?>
</ol> </body> </html>
```

Arrays with loop

```
<html> <body> <ol>
    <? php
    $preferences = array
    ("red","white","blue","silver","aqua","cyan", "yellow");
    for ($i=0; $i<sizeof($preferences); $i++)
    {
     $value = $preferences[$i];
     echo("<li>This preference is: $value</li>");
    } ?>
</ol> </body> </html>
```

Array Operations

- To add values at end of the array
  array_push($preferences, “black”, “gold”);

- To add values in beginning of the array
  array_unshift($preferences, “black”, “gold”);

- To remove an item from the start of the array
  array_shift($preferences);

- To remove an item from the end of the array
  array_pop($preferences);

Cont'd

- To sort an array
- **sort($preferences);**
- **rsort()** - sort arrays in descending order
- SORT_REGULAR compare items normally
- SORT_NUMERIC compare items numerically
- SORT_STRING compare items as strings

- For merging two arrays we have
- **$endarray=array_merge($array1,$array2);**
- To slice up an array
- **$endarray=array_slice($prefs,2,6);**

PHP Global Variables - Superglobals

- Some predefined variables in PHP 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.
- The PHP superglobal variables are:
  - $GLOBALS
  - $_SERVER
  - $_REQUEST
  - $_POST
  - $_GET
  - $_FILES
  - $_ENV
  - $_COOKIE
  - $_SESSION

$GLOBALS

```
<?php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

declare global variable in PHP

```
<?php
// Demonstrate how to declare global variable
// Declaring global variable
$x = “NMIMS";
$y = “STME";
$z = “Computer";
$a = 55;
$b = 100;

function concatenate() {
  // Using global keyword
  global $x, $y, $z;
  return $x.$y.$z;
}
function add() {
  // Using GLOBALS['var_name']
  $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}
```

$_SERVER

```
<html> <body>
<?php
    echo $_SERVER['PHP_SELF']; //page
    echo "<br>";
    echo $_SERVER['SERVER_NAME']; //localhost
    echo "<br>";
    echo $_SERVER['HTTP_HOST'];  //host name, here, localhost
    echo "<br>";
    echo $_SERVER['HTTP_REFERER'];  //complete URL
    echo "<br>";
    echo $_SERVER['HTTP_USER_AGENT'];  //Mozilla / chrome / safari
    echo "<br>";
    echo $_SERVER['SCRIPT_NAME']; //php script name
?>
</body> </html>
```

PHP Form Handling

exmple.php

```
<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>
```

welcome.php

```
<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

PHP Form Validation

- The Form Element
  - <form method="post" action="<?php echo htmlspecialchars ($_SERVER["PHP_SELF"]);?>">
- $_SERVER["PHP_SELF"]
  - The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.
- htmlspecialchars()
  - The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with &lt; and &gt;. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

PHP Form Security

- The $_SERVER["PHP_SELF"] variable can be used by hackers!
- If PHP_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.
- Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.

Vulnerability Example

- Assume we have the following form in a page named "test_form.php":

```
<form method="post" action="<?php echo
$_SERVER["PHP_SELF"];?>">
```

- Now, if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```
<form method="post" action="test_form.php">
```

- However, consider that a user enters the following URL in the address bar:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ea
lert('hacked')%3C/script%3E
```

- In this case, the above code will be translated to:

```
<form method="post" action="test_form.php/"><script>alert('hack
ed')</script>
```

How To Avoid $_SERVER["PHP_SELF"] Exploits?

- $_SERVER["PHP_SELF"] exploits can be avoided by using the htmlspecialchars() function.
- The form code should look like this:

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

- The htmlspecialchars() function converts special characters to HTML entities.
- Now if the user tries to exploit the PHP_SELF variable, it will result in the following output:

```
<form method="post"
action="test_form.php/&quot;&gt;&lt;script&gt;aler
t('hacked')&lt;/script&gt;">
```

Validate Form Data With PHP

```
<html> <head> </head>
<body>
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data); //Strip unnecessary characters (extra space, tab,
newline) from the user input data
  $data = stripslashes($data); // Remove backslashes (\) from the user
input data
  $data = htmlspecialchars($data);
  return $data;
} ?>
```

```
<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  E-mail: <input type="text" name="email">
 Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
<input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;        echo "<br>";
echo $email;       echo "<br>";
echo $website;          echo "<br>";
echo $comment;          echo "<br>";
echo $gender;
?>   </body> </html>
```

In continuation with previous slide

PHP Advanced

PHP Include Files

- Assume we have a standard menu file called "menu.php":

```
<?php
echo '<a href="/default.asp">Home</a> -
<a href="/html/default.asp">HTML Tutorial</a> -
<a href="/css/default.asp">CSS Tutorial</a> -
<a href="/js/default.asp">JavaScript Tutorial</a> -
<a href="default.asp">PHP Tutorial</a>';
?>
```

- All pages in the Web site should use this menu file. Here is how it can be:

```
<html> <body>
<div>        <?php include 'menu.php';?> </div>
<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
</body> </html>
```

Cont’d

- The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.
- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

PHP File Handling

- PHP has several functions for creating, reading, uploading, and editing files.
- PHP readfile() Function
  - The readfile() function reads a file and writes it to the output buffer.

```
<?php
echo readfile(“file.txt");
?>
```

file.txt

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

PHP File Open/Read/Close

```
<?php
$myfile = fopen("file.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("file.txt"));
fclose($myfile);
?>
```

Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist

- PHP Read Single Line - fgets()

```
<?php
$myfile = fopen(“file.txt", "r") or die("Unable to open file!");
echo fgets($myfile);
fclose($myfile);   ?>
```

- PHP Check End-Of-File - feof()

```
<?php
$myfile = fopen(“file.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);   ?>
```

- PHP Read Single Character - fgetc()

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);   ?>
```

- **PHP Write to File - fwrite()**

fwrite(file, string, length)

Parameter	**Description**
file	Required. Specifies the open file to write to
string	Required. Specifies the string to write to the open file
length	Optional. Specifies the maximum number of bytes to write

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = “NMIMS\n";
fwrite($myfile, $txt);
$txt = “STME\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

PHP file upload: Rules

- First, ensure that PHP is configured to allow file uploads.
  - In your "php.ini" file, search for the file_uploads directive, and set it to On

    file_uploads = On
- Create The HTML Form
  - Make sure that the form uses method="post"
  - The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form
- Create The Upload File PHP Script

PHP file upload: HTML Form

```html
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```

PHP file upload: Adding restrictions

```
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
```

```
// Check file size
if ($_FILES["fileToUpload"]["size"] >
500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
```

```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png"
            && $imageFileType != "jpeg“ && $imageFileType != "gif" )
{
   echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
   $uploadOk = 0;
}
```

Upload File PHP Script

```
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType
= strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
   $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
   if($check !== false) {
      echo "File is an image - " . $check["mime"] . ".";
      $uploadOk = 1;
   } else {
      echo "File is not an image.";
      $uploadOk = 0;
   }
}
```

Click for the complete PHP script

Functions used in PHP file upload script

- **isset():** Check whether a variable is empty. Also check whether the variable is set/declared
- **basename():** Return filename from the specified path
- **pathinfo():** returns information about path: either an associative array or a string, depending on options
  - pathinfo (string $path [, int $options = PATHINFO_DIRNAME | PATHINFO_BASENAME | PATHINFO_EXTENSION | PATHINFO_FILENAME])
- **strtolower():** Convert all characters to lowercase
- **getimagesize():** function will determine the size of any supported given **image** file and return the dimensions along with the file type and a height/width text string to be used inside a normal HTML IMG tag and the correspondent HTTP content type
- **$_FILES**: HTTP File Upload variables

PHP Cookies

- What is a Cookie?
  - 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.
  - A cookie is created with the **setcookie()** function.

setcookie(name, value, expire, path, domain, secure, httponly);

Cookie example

```
<?php
$cookie_name = “NMIMS";
$cookie_value = “STME";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>    //before HTML tag

<html>    <body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
<p><strong>Note:</strong> You might have to reload the page to see the value of the
cookie.</p>
</body>  </html>
```

PHP Sessions

- A session is a way to store information (in variables) to be used across multiple pages.
  - Unlike a cookie, the information is not stored on the users computer.
- When you work with an application, you open it, do some changes, and then you close it.
  - This is much like a Session. The computer knows who you are. It knows when you start the application and when you end.
  - But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc).
- By default, session variables last until the user closes the browser.
- So; Session variables hold information about one single user, and are available to all pages in one application.
- If you need a permanent storage, you may want to store the data in a database.

Start a PHP Session

sessionstart.php

```
<?php
// Start the session ; must before HTML tag
session_start();
?>

<html>  <body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>  </html>
```

Get PHP Session Variable Values

sessionretrieve.php

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

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body> </html>
```

More on PHP session

- Another way to show all the session variable values for a user session is to run the following code

```
<?php
print_r($_SESSION);
?>
```

- Modify a PHP Session Variable

```
<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
```

- Destroy a PHP Session

```
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?>
```