Parul® University

Web Programming (203105353)

Prof. Mosam Patel, Assistant Professor
CSE

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-6

PHP

Parul®
University

DIGITAL LEARNING CONTENT

Topics

- Environment Setup
- Variable Types
- Constants
- Operator Types
- Decision Making
- Arrays
- Strings
- Web Concepts
- File Inclusion
- GET & POST
- Functions
- Cookies & Sessions
- File Uploading
- Object Oriented Programming with PHP

Introduction of PHP

•PHP - Hypertext Pre-processor.

•PHP is an open source, Object Oriented programming language for building dynamic & interactive Web pages.

•PHP is an interpreted language; i.e., the compilation is not required.

•PHP is quicker than other scripting languages, such as ASP and JSP.

•PHP is a scripting programming language on the server-side which is used to manage website dynamic content.

•PHP may be incorporated into HTML.

How PHP differs from HTML?

• A simple HTML page looks the same each time its displayed until the page is edited and a user cannot interact with a html page.

•A PHP page is Dynamic, interactive Web sites/ Web Pages which means- A page whose contents can be changed automatically each time the page is viewed. An interactive Web site is a site that responds to input from its visitor.

Features of PHP

- **Five important features make PHP possible in practical terms –**

- Simplicity

- Efficiency

- Security

- Flexibility

- Familiarity

Constituents of a PHP File

- PHP files can contain simple 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"

Working of PHP Script

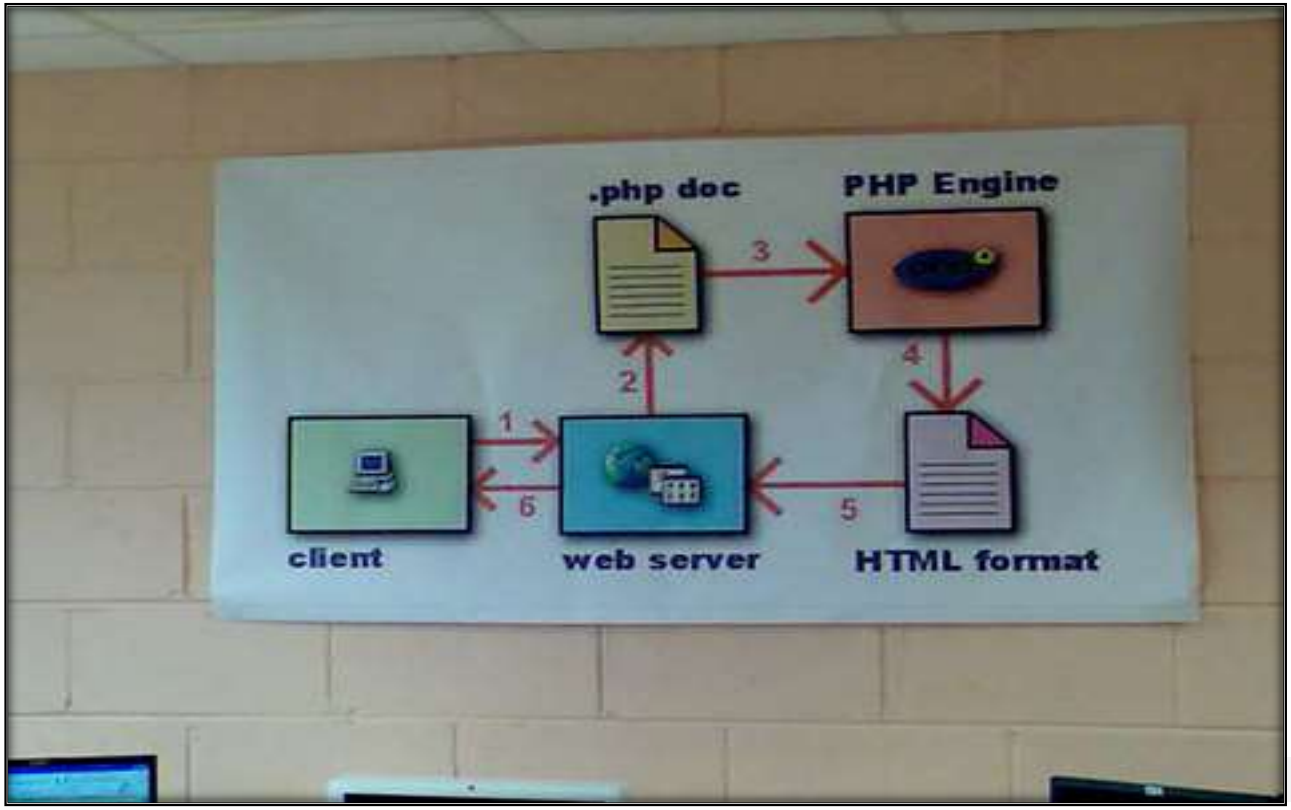

Image Source: flicker, google"

Environment Setup

To start using php on your PC, follow following steps:

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

PHP web server

The PHP Community Provides the following Server solutions under the GNU (General Public License):

- WAMP Server
- LAMP Server
- MAMP Server
- XAMPP Server

First Program in PHP

```
<?
echo " Welcome to the 6th sem's Web Programming Subject at PU";
?>
```

The above program print the message '*Welcome to the 6th sem's Web Programming Subject at PU'* on the web page. Echo method is used for displaying a message/tag on web page.

Three ways to write PHP code

PHP syntax 1 - Most frequently used, reliable and effective:

The most universal tag style in PHP is –

<? Php….? >

If you use this style, you can be positive that you will always interpret your tags correctly.

Three ways to write PHP code

PHP syntax 2 - short-open tags:

```
<?
echo " Welcome to the 6th sem’s Web Programming Subject at PU ";
?>
```

PHP syntax 3 – Enclosed in HTML script tags:

```
<script language="PHP">
    echo " Welcome to the 6th sem’s Web Programming Subject at PU ";
</script>
```

Parul®
University

DIGITAL LEARNING CONTENT

Variables

- $str="Web Programming @ PU"; // String Variable
- $x=2020; // Numeric Variable
- $y=12.6; // Floating point variable
- $bool = true; // Variable for Boolean value

Constants in PHP

- Constants are defined using the define()function of PHP, which accepts two arguments: the name and value of the constant.
- Constant name must follow the same rules as variable names, with one exception that constant names don't require the"$” prefix.

```
<?php
        define('ConstName', 'value');
?>
```

PHP Magic Constant

- There are a number of predefined constants available in PHP scripts.

We will use these constants as we need them; there's a sample:

__LINE__	The current line number of the file.
__FILE__	The full path and filename of the file.
__FUNCTION__	The function name
__CLASS__	The class name
__METHOD__	The class method name
PHP_VERSION	The PHP version
PHP_INT_MAX	The PHP integer value limit

PHP Magic Constant – Example

```
<?php
echo "The Line number : ". __LINE__;
echo  "Your file name :". __FILE__;
echo "Integer Maximum Value : ".PHP_INT_MAX;
?>
```

PHP data types

- **Scalar(It holds only single value)**

- Integer
- Float/double
- String
- Boolean

PHP data types

- **Compound Types (It can keep multiple costs)**

- Array

- Object

- **Special Types**

- resource

- NULL

Array Example

Array Data type

```
<?php
$dept = array ("PIET_CSE", "PIT_CSE", "PIET_IT");
var_dump($dept);    //the var_dump() function returns the datatype
and values
echo "</br>";
echo " Element1: $dept[0] </br>";
echo " Element2: $dept[1] </br>";
echo " Element3: $dept[2] </br>";
?>
```

Object Data type example

- **Object Data type**

```
<?php
class PU
{
public function pu_show()
{
echo "This is show method<br/>";
}
}
$PIET= new PU();
$PIET->pu_show();
var_dump($PIET);
?>
```

PHP Decision Statement

- The if Statement: if Output Statements appears when only Condition must be true.

- If-else Statement: if-else statements allow you to view output in both conditions (Show some message otherwise display other message if condition is true).

- The if-elseif-else statement: if-else-if-else statement allows multiple if-else statements to be chained together, allowing the programmer to define actions for more than two possible outcomes.

- The switch Statement: The switch statement resembles a series of if statements on the same expression.

PHP - Looping

Four types of Loop used in PHP :

- for : In for loop specified condition is predefined.

- while : while loop executes the statement while a specified condition is true. In while loop first check the condition then execute the statement.

- do-while : do - while loop executes the statement once and next statement depends upon a specified condition is true.In do - while loop first execute the statement then check the condition, it means if condition is false in that case one time statement execute.

- foreach : Foreach loop executes the statement() of an associative array.

Parul®
University

DIGITAL LEARNING CONTENT

For loop in PHP

```
<?php
for ($cse=1; $cse<10; $cse++)
 {
echo "$cse ";
}
  ?>
```

Parul®
University

DIGITAL LEARNING CONTENT

PHP While Loop

```
<?php
$it = 1;
while ($it < 10) {
echo 'Web Programming @ PU';
$it++;
}
?>
```

PHP Array Types

- **There are three different types of arrays, and each value of the array is accessed using an ID, called the array index.**

-Indexed Array - An array with a digital index. Values are stored in linear fashions and accessed.

- Associative array – An array that contains strings as index. This stores values of elements not in a strict linear index order, but in association with key values.

- Multidimensional array – Access to an array containing one or more arrays and values using multiple indices.

Indexed Array

```
<?php
$depts=array("CSE","IT","CIVIL","MECHANICAL");

echo "Departments in University are: “. $depts[0] . $depts[1] . $depts[2] .
$depts[3]";

// Alternate to print values

foreach($student as $depts)
{
echo "Dept: ".$student;
}
?>
```

Associative Arrays – Example

```
<?php
$depts=array("cse"=>"550000","it"=>"250000","civil"=>"200000");
foreach($student as $depts)
{
echo "Dept: ".$student;
}
?>
```

Multidimensional Arrays

```
<?php
$depts = array
(
array(1,"cse",400000),
array(2,"it",500000),
array(3,"civil",300000)
);
for ($row = 0; $row < 4; $row++) {
   for ($col = 0; $col < 3; $col++) {
    echo $depts[$row][$col];
  }
 }
?>
```

Sorting Arrays

PHP array sort functions:

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort() - sort associative arrays in descending order, according to the value
- krsort() - sort associative arrays in descending order, according to the key

```
<?php
// sorts an associative array in ascending order, according to the value
$depts=array("cse"=>"550000","it"=>"250000","civil"=>"200000");
asort($depts);
?>
```

PHP String

Single quoted

The programmer can create a string in PHP merely by enclosing the text in a single quote. In general, this way is known as the easiest way to specify the string in PHP. To determine a single literal quote, you just have to escape it with a backslash (\), and you have to use the double backslash (\\) to specify a literal backslash (\). Any other instances with a backslash such as \r or \n will be their output just the same as they have specified, rather than having any particular meaning.

```
<?php
  $myStr='welcome PU';
  echo $myStr;
?>
```

PHP String

- **Double quoted**

A string can be specified in the PHP language by merely enclosing the text in the double quote. And the fact is that unlike in the single quote, only the use of double quote PHP strings will generally interpret the escape sequences and variables.

```
<?php
  $myStr="welcome PU";
  echo $myStr;
?>
```

PHP String Functions

- **strlen() - Return the Length of a String**

```
<?php
echo strlen("Hello Parul university"); // outputs 23
?>
```

- **str_word_count() - Count Words in a String**

```
<?php
echo str_word_count("Hello Parul university"); // outputs 3
?>
```

- **strrev() - Reverse a String**

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

PHP String Functions

- **strpos() - Search For a Text Within a String**

```
<?php
echo strpos("Hello PU", “PU"); // outputs 6
?>
```

- **str_replace() - Replace Text Within a String**

```
<?php
echo str_replace("world", “India", "Hello world!"); // outputs Hello India!
?>
```

PHP – Web Concepts

Identifying Browser & Platform

HTTP_USER_AGENT identifies the user's browser and operating system.

Display Random values

The rand() function is used to generate a random number.

Browser Redirection

The PHP header() function supplies raw HTTP headers to the browser which can be used to redirect it to another location.

PHP – File Inclusions

The include() Function
The require() Function

The include and require statements are identical, except upon failure:

-the require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- the include will only produce a warning (E_WARNING) and the script will continue.

PHP – File Inclusions- Examples

"pu_foot.php":

```
<?php
echo "<p>Copyright &copy; 2020-" . date("Y") . " Paruluniversity.ac.in</p>";
?>
```

To include the pu_foot.php file in a html page, using the include statement:

```
<html>
<body>
<h1>Welcome to University!</h1>
<p>Web Programming course.</p>
<p>203105353.</p>
<?php include 'pu_foot.php';?>
</body>
</html>
```

To include the pu_foot.php file in a html page, using the include statement:

```
<html>
<body>
<h1>Welcome to University!</h1>
<p>Web Programming course.</p>
<?php require 'pu_foot.php‘; echo “ CSE-IT”;
?>
</body>
</html>
```

PHP - Get & Post Methods

Two ways by which a browser client can send information to the web server:

The GET Method- is unsecured method because it display all information on address bar/ url. By default method is get method. Using GET method limited data sends. GET method is faster way to send data.

The POST Method- It is secured method because it hides all information. Using POST method unlimited data sends . POST method is slower method comparatively GET method.

Form GET Method

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

and "welcome_PU.php" looks like this:

```
<html> <body>
Welcome to subject <?php echo $_GET["Subject"]; ?><br>
Your subject code is: <?php echo $_GET["Code"]; ?>
</body> </html>
```

Form POST Method

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

and "welcome_PU.php" looks like this:

```
<html> <body>
Welcome to subject <?php echo $_POST["Subject"]; ?><br>
Your subject code is: <?php echo $_POST["Code"]; ?>
</body> </html>
```

GET vs. POST

• The array is created by both GET and POST (e.g. array(key1= > value1, key2= > value2, key3= > value3, ...)). This array contains key / value pairs, where key names are the names of the form controls, and values are the user input data.

• GET and POST are both treated as both $ GET and $ POST. These are superglobals, meaning they are always accessible, regardless of scope-and you can access them from any function, class or file without having to do anything special about them.

• $_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.

PHP Function

•In PHP function, we can pass the information through arguments which are separated by comma.

•PHP supports Call by Value (default), Call by Reference, Argument Default values, and Argument Length List of Variables.

Example -

```
<?php
function PU($name){
echo "Webprogramming@” . $name;
}
PU("CSE");
PU("IT");
?>
```

Passing Arguments by Reference

```
<?php
function pu_adder(&$stud)
{
$stud .= “CSE-IT”;
}
$stu = 'Web programming@ PU ';
Pu_adder($stu);
echo $stu;
?>
```

PHP Functions returning value

```
<?php
function pu_cube($n){
return $n*$n*$n;
}
echo "Cube of 20 is: ".cube(20);
?>
```

PHP Functions Default Argument value

```
<?php
function pu($name="CSE"){
echo "Webprogramming @ $name<br/>";
}
pu("PIET");
pu();//passing no value
pu("PIT");
?>
```

PHP Cookie

• A cookie in PHP is a small piece of information stored on the browser of the client. It's used for user recognition.

• There are three steps to identify returning users –

-A set of cookies is sent to the browser by a server script—for example, name, age, or ID number, etc.
-The browser will store this information for future use on a local machine.
-When the browser sends any request to the web server next time, it sends the information about those cookies to the server, and the server uses that information to identify the user.

PHP setcookie() function

It is used to set HTTP response cookies. Once a cookie has been established, it can be accessed via the superglobal variable $ COOKIE.

Syntax:

```
bool setcookie( string $name [, string $value [, int $expire = 0 [, string $path
[, string $domain [, bool $secure = false [, bool $httponly = false ]]]]] )
```

Example:

```
setcookie("CName", "CValue");/* defining the name and value only*/
setcookie("CName", "CValue", time()+3*60*60);//using expiry in 3 hour(3*60*60
seconds or 3600 seconds)
setcookie("CName", "CValue", time()+3*60*60,"/mypupath/", "mypudomain.com",
1);
```

PHP Delete Cookie

If you set the expiration date in the past, the cookie will be deleted.

```
<?php
setcookie ("CName", "", time() - 7200);
// set the expiration date to two hour ago
?>
```

PHP Session

•PHP session is used temporarily (until the user closes the website) to store and pass information from one page to another.

•Session in PHP is widely used in shopping websites where we need to store and pass on cart information from one page to another, e.g., username, product code, product name, product price, etc.

•To recognize the user and avoid conflict b/w multiple browsers, the PHP session creates unique user ids for each browser.

PHP $_SESSION

PHP $_SESSION is an associative array that contains all session variables.

It is used to set and get session variable values.

Example: Store information

- $_SESSION["user"] = "Sachin";

Example: Get information

- echo $_SESSION["user"];

PHP File Uploading

•PHP configuration file php.ini describes the temporary directory that is used for file uploads as upload_tmp_dir and the maximum permitted size of files that can be uploaded is stated as upload_max_filesize.

• **php.ini" file Configuration**

To ensure that PHP is configured to allow file uploads, in the "php.ini" file perform following steps.

file_uploads = On

PHP File Uploading

Step1: Create the HTML Form

```
<html>
<body>

<form action="puupload.php" method="post" enctype="multipart/form-data">
//enctype="multipart/form-data" specifies which content-type to use when
      //submitting the form
Select course to upload:
<input type="file" name="coursefileUpload" id="coursefileUpload">
<input type="submit" value="Upload course" name="upload">
</form>
</body>
</html>
```

PHP File Uploading

Step 2: Create the puUpload.php File - PHP Script contains the code for uploading a file:

```
 <?php
$pu_dir = "puuploads/";
$pu_file = $pu_dir . basename($_FILES["coursefileUpload"]["name"]);
$puuploadDone = 1;
$cFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
?>
```

Step 3: Check if File Already Exists

```
if (file_exists($pu_file)) {
 echo "Sorry, the course file already exists.";
 $puuploadDone = 0;
}
```

PHP File Uploading

Step 4: Limit File Size

If the file is larger than 450KB, an error message is displayed, and $uploadOk is set to 0:

```
if ($_FILES["coursefileUpload"]["size"] > 450000) {
  echo "Sorry, your Course file is too large.";
  $puuploadDone = 0;
}
```

Step 5: Limit File Type

```
if($cFileType != "txt" && $cFileType != "doc" && $imageFileType != "docx"
&& $imageFileType != "pdf" )
{
  echo "Sorry, only txt, document or pdf files are allowed.";
  $puuploadDone = 0;
}
```

PHP WITH OOPS CONCEPT

Defining PHP Classes

```
<?php
  class puClass
 {
     var $cse;
     var $it = "Web programming";

     function pufunc ($course)
     {   $this->cse = $course;     }
    function pudisp()
     {   echo $cse . “  “ . $it;      }
  }
?>
```

PHP WITH OOPS CONCEPT

Creating Objects in PHP

The objects of above created puClass is as follows:

```
$piet = new puClass;

$pit = new puClass;
```

PHP WITH OOPS CONCEPT

Calling Member Functions

```
$piet->pufunc( "Advanced Web Programming" );
$pit-> pufunc( "Sem-6  Web Programming" );

$piet->pudisp();
$pit->pudisp();
```

PHP WITH OOPS CONCEPT

Constructor Functions

```
function __construct( $par1, $par2 ) {
   $this->cse = $par1;
   $this->it = $par2;
}
$piet = new puClass( "Advanced Web Programming", "Sem-6 WP");
$pit = new puClass ( "Adv wp", "Sem-7 Python" );

/* Get the set values */
$piet->pudisp();
$pit->pudisp();
```

PHP WITH OOPS CONCEPT

Inheritance

```
class piet extends PuClass {
   var $sem;

   function setsem($vsem){
      $this->sem = $vsem;
   }
   function getsem(){
      echo $this->sem;
   }
}
```

PHP WITH OOPS CONCEPT

Function Overriding

```
function pudisp() {
   echo $this->sem;
   return $this->sem;
}
```

Interfaces

```
interface puMail {
   public function pusendMail();
}
```

Abstract Classes:

Defines an Abstract class and an abstract function.

```
abstract class puAbstractCls
{
  abstract  function  puAbstractFunc  ()
{   }
}
```

PHP WITH OOPS CONCEPT

- **Abstraction in PHP – Example:**

```
<?php
abstract class Animal
{
public $name;
public $age;
public function Describe()
{
return $this->name . ", " . $this->age
. " years old";
      }
abstract public function Greet();
}
class cat extends
Animal
{
public function
Greet()
{
return "Lion!";
}
public function
Describe()
{
return
parent::Describe() . ",
and I'm a cat!";
}
}
$animal = new cat();
$animal->name =
"Seru";
$animal->age = 5;
echo $animal-
>Describe();
echo $animal-
>Greet();

?>
```

PHP WITH OOPS CONCEPT

- **PHP – Interface- Example:**

```
<?php
   interface a
   {
      public function dis1();
   }
   interface b
   {
      public function dis2();
   }
class demo implements a,b
{
   public function dis1()
   {
      echo "method 1...";
   }
   public function dis2()
   {
      echo "method2...";
   }
}
$obj= new demo();
$obj->dis1();
$obj->dis2();
 ?>
```

PHP WITH OOPS CONCEPT

Static Keyword

```
<?php
  class pu {
     public static $static_cse = 'piet';
      public function staticVal() {
        return self::$static_cse;
      }
  }
  print pu::$ static_cse;
  $pit = new pu();
  print $pit->staticVal();
?>
```

PHP WITH OOPS CONCEPT

- **EXAMPLE : Public member:**

```
<?php
class pu
{
Public $name="Web
Programming";
Function disp()
{
echo $this->name."<br/>";
}
}
class pit extends pu
{
function show()
{
echo $this->name;
}
}
$obj= new pit;
echo $obj->name."<br/>";
$obj->disp();
$obj->show();
?>
```

PHP WITH OOPS CONCEPT

- **EXAMPLE : Private member:**

```
<?php
Class PU
{
private $name=“Web
Programming";
private function show()
{
echo "This is private method of
parent class";
}
}
class piet extends PU
{
function show1()
{
echo $this->name;
}
}
$obj= new piet;
$obj->show();
$obj->show1();
?>
```

PHP WITH OOPS CONCEPT

- **EXAMPLE : Protected member:**

```
<?php
Class PU
{
protected $x=500;
protected $y=100;
   function add()
{
echo $sum=$this->x+$this-
>y."<br/>";
}
   }
class pit extends PU
{
function sub()
{
echo $sub=$this->x-$this->y;
}
}
$obj= new pit;
$obj->add();
$obj->sub();
?>
```

PHP WITH OOPS CONCEPT

- **Property Overloading – Example :**

```
class PUOverload
{
   private $data = array();
   public function __set($name, $value)
   {
      $this->data[$name] = $value;
   }
public function __get($name)
   {
   return $this->data[$name];
   }
public function __isset($name)
   {
      return isset($this->data[$name]);
   }
   public function __unset($name)
   {
      unset($this->data[$name]);
   }
}
$piet = new PUOverload;
$piet->name = “Web Programming";
echo $piet->name; // Web
Programming
```

PHP WITH OOPS CONCEPT

Method Overloading Example:

```
<?php
class puOverloading
{
   public function __call($name, $arguments)
   {
      echo "Calling object method '$name' "
          . implode(', ', $arguments). "\n";
   }
public static function __callStatic
($name, $arguments)
{
  echo "Calling static method '$name' "
          . implode(', ', $arguments);
 }
}

$pit = new puOverloading;
$pit->runTest('in object context');
MethodTest::runTest('in static
context');
```

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in