SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | Approved by AICTE
QUALITY SYSTEM CERTIFICATION
DNV·GL
ISO 9001:2015

Subject Code : SITA1502

Subject Name: CUSTOMER INTERFACE DESIGN AND DEVELOPMENT

Prepared By: Dr. R. M. Gomathi/Dr. M. Malini Deepika

Unit – III Server Side Scripting

UNIT 3 SERVER SIDE SCRIPTING **9 Hrs.**

Introduction To PHP – Data Types – Control Structures – Arrays - Function – Html Form with PHP –Form Handling and Validation - File Handling – Cookies – Sessions – Filters – Exception Handling - Database Connectivity With MySQL.

Introduction to Scripting Languages

- All scripting languages are programming languages.
- The scripting language is basically a language where instructions are written for a run time environment.
- They do not require the compilation step and are rather interpreted. It brings new functions to applications and glue complex system together.
- A scripting language is a programming language designed for integrating and communicating with other programming languages.

Eg. PHP, Node js, Ruby, Python, Perl etc.,

Differences Between Programming vs Scripting

Programming

- Specific set of instructions that can be used to produce various kinds of output.
- Compiled to machine code and run on the hardware of the underlying Operating system.

  Eg. Java, Scala, C, C++, etc.

Scripting

- Scripting languages are generally interpreted. A scripting language's primary focus does not build the application but might provide behavior to an existing application.
- It can manipulate, customize and automate the facilities of an existing system.

  Eg. Javascript, Perl, VBScript, etc.,

Application of Scripting Languages

- Scripting languages are used in web applications. It is used in server side as well as client side.
- Client-side scripting languages are: JavaScript, VBScript, AJAX, jQuery etc.
- Server-side scripting languages are: PHP, ASP.Net, Perl etc.
- Scripting languages are used in system administration.

  Eg: Shell, Perl, Python scripts etc.
- It is used in Games application and Multimedia.
- It is used to create plugins and extensions for existing applications.

PHP - Introduction

- Acronym for "PHP: Hypertext Preprocessor“.
- PHP was conceived sometime in the fall of 1994 by Rasmus Lerdorf.
- PHP is a server-side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- PHP is a powerful tool for making dynamic and interactive Web Pages.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP Syntax is C-Like.
- PHP scripts are executed on the server
- PHP is free to download and use.
- Currently, the latest stable version is PHP 8.0.

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access and encrypt data

- The user may 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 OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is loosely typed language

Example: - "Hello World" Script in PHP

```
<html>
    <head>
    <title>Hello World</title>
  </head>
    <body>
    <?php echo "Hello, World!";?>
  </body>
</html>
```

Output:

Hello, World!

Parser Recognised comments

All PHP code must be included inside one of the three special markup tags which are recognised by the PHP Parser.

<?php PHP code goes here ?>

<? PHP code goes here ?>

<script language = "php"> PHP code goes here </script>

A most common tag is the <?php...?>

PHP echo and print Statements

There are two basic ways to get the output in PHP:

- echo
- Print

Difference between echo and print

echo

- echo is a statement, which is used to display the output.
- echo can be used with or without parentheses.
- echo does not return any value.
- We can pass multiple strings separated by comma (,) in echo.
- echo is faster than print statement.

print

- print is also a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than echo statement.

For Example (Check multiple arguments)

You can pass multiple arguments separated by a comma (,) in echo. It will not generate any syntax error

```
<?php
  $fname = "Gunjan";
  $lname = "Garg";
 echo "My name is: ".$fname,$lname;
?>
```

Output:

My name is: GunjanGarg

Note: print statement will generate a **syntax error** because of multiple arguments.

For Example (Check Return Value)

echo statement does not return any value. It will generate an error if you try to display its return value. But print returns a value, which is always 1

```
<?php
  $lang = "PHP";
  $ret = print $lang." is a web development language.";
  print "</br>";
  print "Value return by print statement: ".$ret;
?>
```

Output:

PHP is a web development language

Value return by print statement: 1

PHP Variables

- Denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.

PHP Data Types

PHP supports the following data types:

1. Integer
2. Float (floating point numbers - also called double)
3. Boolean
4. NULL
5. String
6. Array
7. Object
8. Resource

1. Integers

They are whole numbers, without a decimal point, like 4195. They are the simplest type .they correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions as following,

```
$int_var = 12345;

$another_int = -12345 + 12345;
```

2. Doubles

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code −

```
<?php
   $a = 2.2888800;
   $b = 2.2111200;
   $c = $a + $b;
   print("$a + $b = $c <br>");
   ?>
```

Output:

```
2.28888 + 2.21112 = 4.5
```

2. Boolean

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so −

```
if (TRUE)
   print("This will always print<br>");
else
   print("This will never print<br>");
```

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

- `$true_num = 3 + 0.14159;`
- `$true_str = "Tried and true"`
- `$true_array[49] = "An array element";`
- `$false_array = array();`
- `$false_null = NULL;`
- `$false_num = 999 - 999;`
- `$false_str = "";`

4. NULL

NULL is a special type that only has one value ‘NULL’. To give a variable the NULL value, simply assign it like this –

Example: $my_var = null;

or

$my_var = NULL;

```php
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

Output:

NULL

5. String

- A string is a sequence of characters, like "Hello world!". A string can be any text inside quotes. You can use single or double quotes.

- There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.
- Strings that are delimited by double quotes are preprocessed in both the following two ways by PHP .
- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of their values.

Example:

```
<?php
  $variable = "name";
  $literally = 'My $variable will not print!';
  print($literally);
  print "<br>";
  $literally = "My $variable will print!";
  print($literally);
?>
```

Output:

My $variable will not print!
My name will print

6. Array

An array stores multiple values in one single variable. In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

```
<html>
<body>

<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>

</body>
</html>
```

Output:

```
array(3) {
[0]=>  string(5) "Volvo"
[1]=>  string(3) "BMW"
[2]=>  string(6) "Toyota“
}
```

7. Object

- Classes and objects are the two main aspects of object-oriented programming.
- A class is a template for objects, and an object is an instance of a class.
- When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.
- Let's assume we have a class named Car. A Car can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties.
- When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.
- If you create a __construct() function, PHP will automatically call this function when you create an object from a class.

Example: -

```
<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
```

Output:

My car is a black Volvo!

My car is a red Toyota!

8. Resource

- The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.
- They are special variables that hold references to resources external to PHP (such as database connections).
- A common example of using the resource data type is a database call.

PHP Control Structures and Loops

- The control structure controls the flow of code execution in application.
- PHP supports a number of different control structures:

1. if
2. else
3. elseif
4. switch
5. while
6. do-while
7. for
8. foreach

1. PHP If Statement

The ‘if’ construct allows you to execute a piece of code if the expression provided along with it evaluates to true.

Eg:-

```
<?php
$age = 50;
if ($age > 30)
{
  echo "Your age is greater than 30!";
}
?>
```

Output: Your age is greater than 30!

2. PHP Else Statement

The ‘if’ construct, which allows you to execute a piece of code if the expression evaluates to true. On the other hand, if the expression evaluates to false, it won't do anything. More often than not, you also want to execute a different code snippet if the expression evaluates to false.

Eg:-

```
<?php
$age = 50;
 if ($age < 30)
{
  echo "Your age is less than 30!";
}
else
{
  echo "Your age is greater than or equal to 30!";
}
?>
```

Output: Your age is greater than or equal to 30!

3. PHP Else If Statement

We can consider the elseif statement as an extension to the if-else construct. If there are more than two choices to choose from, we can use the elseif statement.

Eg:-

```
<?php
$age = 50;
if ($age < 30)
  echo "Your age is less than 30!";
elseif ($age > 30 && $age < 40)
  echo "Your age is between 30 and 40!";
elseif ($age > 40 && $age < 50)
  echo "Your age is between 40 and 50!";
else
  echo "Your age is greater than 50!";
?>
```

Output : Your age is greater than 50!

4. PHP Switch Statement

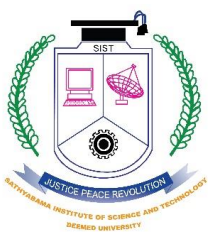
The switch statement is similar to a series of IF statements on the same expression. In many occasions, you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to.

Eg:-

```
<?php
$favourite_site = 'Code';
 switch ($favourite_site) {
  case 'Business':
    echo "My favourite site is business.tutsplus.com!";
    break;
  case 'Code':
    echo "My favourite site is code.tutsplus.com!";
    break;
  case 'Web Design':
    echo "My favourite site is webdesign.tutsplus.com!";
```

```
 break;
  case 'Music':
    echo "My favourite site is music.tutsplus.com!";
    break;
  case 'Photography':
    echo "My favourite site is photography.tutsplus.com!";
    break;
  default:
    echo "I like everything at tutsplus.com!";
}
?>
```

Output: My favourite site is code.tutsplus.com!



Loops in PHP

1. While Loop in PHP

The while loop is used when you want to execute a piece of code repeatedly until the while condition evaluates to false.

Eg:-

```
<?php
$max = 0;
echo $i = 0;
echo ",";
echo $j = 1;
echo ",";
$result=0;
 while ($max < 10 )
{
    $result = $i + $j;
     $i = $j;
    $j = $result;
     $max = $max + 1;
    echo $result;
    echo ",";
}
?>
```

Output:

It outputs the Fibonacci series for the first ten numbers.

0,1,1,2,3,5,8,13,21,34,55,89,

2. Do-While Loop in PHP

The do-while loop is very similar to the while loop, with the only difference being that the while condition is checked at the end of the first iteration. Thus, we can guarantee that the loop code is executed at least once, irrespective of the result of the while expression.

Eg:-

```
<?php
$handle = fopen("file.txt", "r");
if ($handle)
{
    do
    {
        $line = fgets($handle);
         // process the line content
     } while($line !== false);
}
fclose($handle);

?>
```

3. For Loop in PHP

Generally, the for loop is used to execute a piece of code a specific number of times. In other words, if you already know the number of times you want to execute a block of code, it's the for loop which is the best choice.

Eg:-

```
<?php
for ($i=1; $i<=10; ++$i)
{
  echo sprintf("The square of %d is %d.</br>", $i, $i*$i);
}
?>
```

Output:

The square of 1 is 1.
The square of 2 is 4.
The square of 3 is 9.
The square of 4 is 16.
The square of 5 is 25.
The square of 6 is 36.
The square of 7 is 49.
The square of 8 is 64.
The square of 9 is 81.
The square of 10 is 100.

The above program outputs the square of the first ten numbers. It initializes $i to 1, repeats as long as $i is less than or equal to 10, and adds 1 to $i at each iteration.

4. For Each in PHP

The foreach loop is used to iterate over array variables. If you have an array variable, and you want to go through each element of that array, the foreach loop is the best choice.

Eg:-

```
<?php
$fruits = array('apple', 'banana', 'orange', 'grapes');
foreach ($fruits as $fruit)
{
  echo $fruit;
  echo "<br/>";
}

$employee = array('name' => 'John Smith', 'age' => 30, 'profession' => 'Software Engineer');
foreach ($employee as $key => $value)
{
  echo sprintf("%s: %s</br>", $key, $value);
  echo "<br/>";
}
?>
```

Output:

apple
banana
orange
grapes
name: John Smith

age: 30

profession: Software Engineer

Note: If you want to access array values, you can use the first version of the foreach loop, as shown in the above example. On the other hand, if you want to access both a key and a value, you can do it as shown in the $employee example above.

PHP Arrays

An array is a special variable, which can hold more than one value at a time.

Create an Array in PHP

In PHP, the array() function is used to create an array.

For example: If you have a list of items (a list of car names, for example), the array can be created as follows:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Output: I like Volvo, BMW and Toyota.

Count()

Get The Length of an Array - The count() Function

The count() function is used to return the length (the number of elements) of an array.

Eg:-

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

Output: 3

Types of arrays

- **Indexed arrays -** Arrays with a numeric index
- **Associative arrays -** Arrays with named keys
- **Multidimensional arrays -** Arrays containing one or more arrays

1. PHP Indexed Arrays

There are two ways to create indexed arrays. The index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo", "BMW", "Toyota");
```

or

the index can be assigned manually:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

Example:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Output: I like Volvo,BMW and Toyota

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:

Eg:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);
for($x = 0; $x < $arrlength; $x++) {
  echo $cars[$x];
  echo "<br>";
}
?>
```

Output:

Volvo

BMW

Toyota

2. PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them. There are two ways to create an associative array:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

(Or)

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

The named keys can then be used in a script.

Example:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Output: Peter is 35 years old.

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}
?>
```

Output:

Key=Peter, Value=35

Key=Ben, Value=37

Key=Joe, Value=43

3. PHP - Multidimensional Arrays

A multidimensional array is an array containing one or more arrays. PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. The dimension of an array indicates the number of indices you need to select an element.

- For a two-dimensional array , we need two indices to select an element
- For a three-dimensional array , we need three indices to select an element

PHP - Two-dimensional Arrays

- A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);
```

Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column. To get access to the elements of the $cars array we must point to the two indices (row and column):

Example:

```
<?php
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>";
?>
```

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.

We can also put a for loop inside another for loop to get the elements of the $cars array (we still have to point to the two indices):

```
<?php
for ($row = 0; $row < 4; $row++)
{
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++)
{
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

Output:

```
Row number 0
Volvo
22
18
Row number 1
BMW
15
13
Row number 2
Saab
5
2
Row number 3
Land Rover
17
15
```

PHP Functions

PHP has more than 1000 built-in functions, and in addition we can create our own custom functions.

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

PHP Date/Time Functions

Function		Description
date_add()	-	Adds days, months, years, hours, minutes, and seconds to a date
date_diff()	-	Returns the difference between two dates
date_format()	-	Returns a date formatted according to a specified format

PHP String Functions

Function		Description
print()	-	Outputs one or more strings
printf()	-	Outputs a formatted string
str_pad()	-	Pads a string to a new length
str_repeat()	-	Repeats a string a specified number of times

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create our own functions.

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

Create a User Defined Function in PHP

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

Syntax

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

Note: A function name must start with a letter or an underscore.

Function names are NOT case-sensitive.

Example:-

```
<?php
function writeMsg() {
  echo "Hello world!";
}
writeMsg(); // call the function
?>
```

Output: Hello world!

PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma. The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example:

```
<?php
function familyName($fname) {
  echo "$fname <br>";
}
 familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

Result:

Jani

Hege

Stale

Kai Jim

Borge

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is. PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives us an option to specify the expected data type when declaring a function, and by adding the `strict` declaration, it will throw a "Fatal Error" if the data type mismatches.

In the following example we try to send both a number and a string to the function without using `strict`:

Example:

```
<?php
function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it will return 10
?>
```

Result: 10

To specify strict we need to set declare(strict_types=1);. This must be on the very first line of the PHP file.
In the following example we try to send both a number and a string to the function, but here we have added the strict declaration:

```
<?php declare(strict_types=1);       // strict requirement
function addNumbers(int $a, int $b)
{
  return $a + $b;
}
echo addNumbers(5, "5 days");      // since strict is enabled and "5 days" is not an integer, an error will be thrown
?>
```

Result:

PHP Fatal error: Uncaught TypeError:

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example:

```
<?php
function setHeight(int $minheight = 50) {
  echo "The height is : $minheight <br>";
}
 setHeight(350);
setHeight();
setHeight(135);
setHeight(80);
?>
```

Result:

The height is : 350
The height is : 50
The height is : 135
The height is : 80

PHP Functions - Returning values

To let a function return a value, use the **return** statement:

Example:

```
<?php
function sum(int $x, int $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);
?>
```

Result:

5 + 10 = 15
7 + 13 = 20
2 + 4 = 6

HTML Form with PHP

An HTML form is used to collect user input. The user input is most often sent to a server for processing. The PHP superglobals $_GET and $_POST are used to collect form-data.

PHP - A Simple HTML Form
The example below displays a simple HTML form with two input fields and a submit button:
Example

```
<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>
```

Result:

Name:
E-mail:
Submit

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:

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

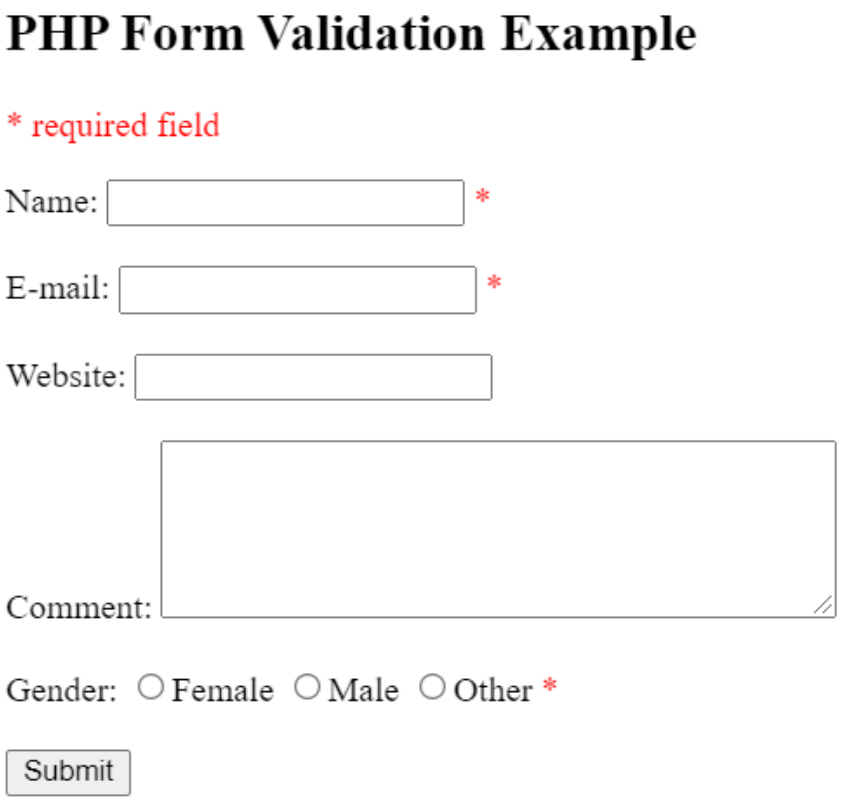
The output could be something like this:

Welcome John

Your email address is john.doe@example.com

Form Validation

- Proper validation of form data is important to protect your form from hackers and spammers!
- The HTML form we will be working at in these chapters, contains various input fields: required and optional text fields, radio buttons, and a submit button:



HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

- 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>

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

- 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

The Form Element

The HTML code of the form looks like this:

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

When the form is submitted, the form data is sent with method="post".

- What is the **$_SERVER["PHP_SELF"] variable**?

  The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.

So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

➢ What is the **htmlspecialchars() function**?

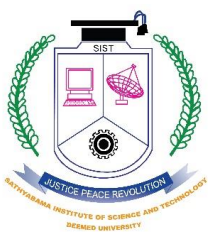
The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

The validation rules for the form above are as follows:

- **Field** **Validation Rules**
- Name - Required. + Must only contain letters and whitespace
- E-mail - Required. + Must contain a valid email address (with @ and .)
- Website - Optional. If present, it must contain a valid URL
- Comment - Optional. Multi-line input field (textarea)
- Gender - Required. Must select one

```php
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }
```

```
if (empty($_POST["website"])) {
  $website = "";
 } else {
  $website = test_input($_POST["website"]);
 }

 if (empty($_POST["comment"])) {
  $comment = "";
 } else {
  $comment = test_input($_POST["comment"]);
 }

 if (empty($_POST["gender"])) {
  $genderErr = "Gender is required";
 } else {
  $gender = test_input($_POST["gender"]);
 }
}
?>
```



PHP - Display The Error Messages

Then in the HTML form, we add a little script after each required field, which generates the correct error message if needed (that is if the user tries to submit the form without filling out the required fields):

Example:

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

Name: <input type="text" name="name">
<span class="error">* <?php echo $nameErr;?></span>
<br><br>
```

E-mail:

<input type="text" name="email">

* <?php echo $emailErr;?>

Website:

<input type="text" name="website">

<?php echo $websiteErr;?>

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

* <?php echo $genderErr;?>

<input type="submit" name="submit" value="Submit">

</form>

PHP Forms - Validate E-mail and URL

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters, dashes, apostrophes and whitespaces. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

The preg_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise.

PHP - Validate E-mail

The easiest and safest way to check whether an email address is well-formed is to use PHP's filter_var() function.

In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

PHP - Validate URL

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";
}
```

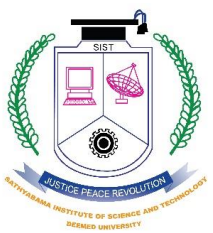
PHP - Validate Name, E-mail, and URL

Now, the script looks like this:

Example

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }
```

```
if (empty($_POST["email"])) {
   $emailErr = "Email is required";
 } else {
   $email = test_input($_POST["email"]);
   // check if e-mail address is well-formed
   if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
     $emailErr = "Invalid email format";
   }
 }
```



```
if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this
regular expression also allows dashes in the URL)
    if  (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-
z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }
```


```
if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }


  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

Result:

PHP Form Validation Example

* required field

Name: Raj *

E-mail: raj@gmailcom *

Website:

Comment:

Gender: Female Male Other *

Submit

Your Input:

PHP Form Validation Example

* required field

Name: *

E-mail: * Invalid email format

Website:

Comment:

Gender: Female Male Other * Gender is required

Submit

Your Input:

Raj
raj@gmailcom

After hitting the Submit button, the error message will be generated in the form output like this,

PHP File Handling

File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP Manipulating Files

PHP has several functions for creating, reading, uploading, and editing files.

Note: When you are manipulating files you must be very careful. You can do a lot of damage if you do something wrong. Common errors are: editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.

PHP readfile() Function

- The readfile() function reads a file and writes it to the output buffer.
- The readfile() function is useful if all you want to do is open up a file and read its contents.
- Assume we have a text file called **"webdictionary.txt",** stored on the server, that looks like this:

➢ 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

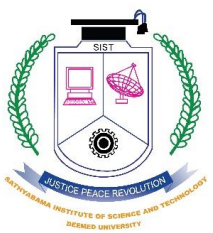
The PHP code to read the file and write it to the output buffer is as follows (the readfile() function returns the number of bytes read on success):

Example

```
<?php

echo readfile("webdictionary.txt");

?>
```

Result:

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 Language236



PHP File Open/Read/Close

PHP Open File - fopen()

A better method to open files is with the fopen() function. This function gives you more options than the readfile() function.

The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the fopen() function is unable to open the specified file:

PHP Read File - fread()

The fread() function reads from an open file. The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

The following PHP code reads the "webdictionary.txt" file to the end:

```
fread($myfile,filesize("webdictionary.txt"));
```

PHP Close File - fclose()

The fclose() function is used to close an open file. It's a good programming practice to close all files after finished with them. You don't want an open file running around on your server taking up resources!

The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

Example:

```
<?php

$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");

echo fread($myfile,filesize("webdictionary.txt"));

fclose($myfile);

?>
```

Result:

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 Read Single Line - fgets()

The fgets() function is used to read a single line from a file. The example below outputs the first line of the "webdictionary.txt" file:

Example:

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

Result:

AJAX = Asynchronous JavaScript and XML

Note: After a call to the fgets() function, the file pointer has moved to the next line.

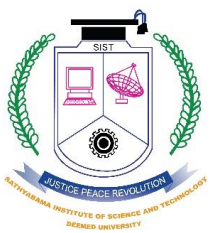
PHP Check End-Of-File - feof()

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. The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:

Example:

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

Cookies and Sessions



Need for Cookies and Sessions

- HTTP is a stateless protocol
- This means that each request is handled independently of all the other requests and it means that a server or a script cannot remember if a user has been there before
- However, knowing if a user has been there before is often required and therefore something known as cookies and sessions have been implemented

Similarity between Cookies and Sessions

- Different mechanisms of the same solution
- **Cookies**

  A mechanism for storing data in the remote browser and thus tracking or identifying return users

- **Sessions**

  It is support in PHP consists of a way to preserve certain data across subsequent accesses. This enables you to build more customized applications and increase the appeal of your web site.

What is a Cookie?

- A cookie is a piece of text that a Web server can store on a users hard disk
- A cookie is a variable, sent by the server to the browser
- Cookies allow a Web site to store information on a users machine and later retrieve it. The pieces of information are stored as name-value pairs
- With PHP, one can both create and retrieve cookie values

What is a Cookie?

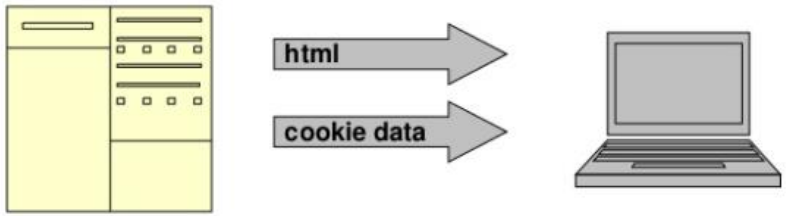
- Each cookie on the user's computer is connected to a particular domain
- Each time the same computer requests a page with a browser, it will send the cookie too
- Each cookie can store up to 4kB of data
- A maximum of 20 cookies can be stored on a user's PC per domain

When are Cookies Created?

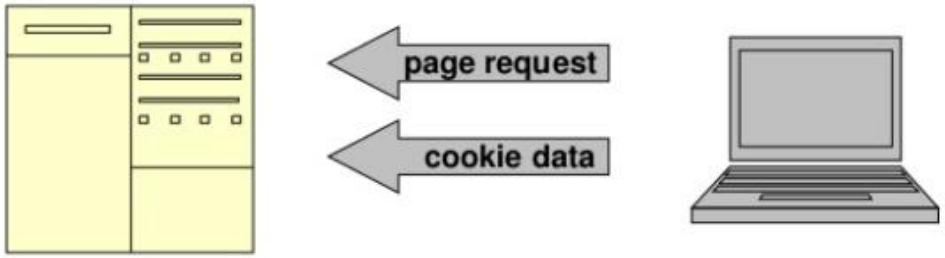
- When a new webpage is loaded
- If the user has elected to disable cookies then the write operation will fail, and subsequent sites which rely on the cookie will either have to take a default action

Example (1) User sends a request for page at www.example.com for the first time. page request

Example (2) Server sends back the page html to the browser AND stores some data in a cookie on the user's PC. html cookie data



Example (3) At the next page request for domain www.example.com, all cookie data associated with this domain is sent too. page request cookie data



Parts of a Cookie

Each cookie is effectively a small lookup table containing pairs of (key, data) values - for example (firstname, John) (lastname,Peter)

Once the cookie has been read by the code on the server or client computer, the data can be retrieved and used to customise the web page appropriately

Syntax for setting up a cookie

setcookie(name [,value [,expire [,path [,domain [,secure]]]]])

- name = cookie
- namevalue = data to store (string)
- expire = UNIX timestamp when the cookie expires. Default is that cookie expires when browser is closed.
- path = Path on the server within and below which the cookie is available on.
- domain = Domain at which the cookie is available for.
- secure = If cookie should be sent over HTTPS connection only. Default false.

Examples for setcookie

setcookie(‘name’,’Robert’)

This command will set the cookie called name on the user’s PC containing the data Robert. It will be available to all pages in the same directory or subdirectory of the page that set it (the default path and domain). It will expire and be deleted when the browser is closed (default expire)

Examples for setcookie

setcookie(‘age’,’20’,time()+60*60*24*30)

This command will set the cookie called age on the user’s PC containing the data 20. It will be available to all pages in the same directory or subdirectory of the page that set it (the default path and domain). It will expire and be deleted after 30 days.

Examples for setcookie

setcookie(‘gender’,’male’,0,’/’)

This command will set the cookie called gender on the user’s PC containing the data male. It will be available within the entire domain that set it. It will expire and be deleted when the browser is closed.

Read cookie data

- All cookie data is available through the superglobal

$_COOKIE:

$variable = $_COOKIE[‘cookie_name’] or

$variable = $HTTP_COOKIE_VARS[‘cookie_name’];

Example: $age = $_COOKIE[‘age’]

Delete a cookie

- To remove a cookie, simply overwrite the cookie with a new one with an expiry time in the past

setcookie(‘cookie_name’,’’,time()-6000)

Note that theoretically any number taken away from the time() function should do, but due to variations in local computer times, it is advisable to use a day or two.

Problems with Cookies

- Browsers can refuse to accept cookies
- Additionally, it adds network overhead to send lots of information back and forth.
- There are also limits to the amount of information that can be sent
- Some information you just don’t want to save on the client’s computer.

Sessions

- A Session allows to store user information on the server for later use (i.e. username, shopping cart items, etc)
- However, this session information is temporary and is usually deleted very quickly after the user has left the website that uses sessions
- 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
- The session_start() function must appear BEFORE the <html> tag.

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

Storing a Session Variable

- The $_SESSION variable can be store and retrieved in PHP

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

</body>
</html>
```

Retrieving a Session Variable

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

Display:

Pageviews = 1

Destroying a Session

- 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();
?>
```

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

Cookies vs. Sessions

Cookies

- Cookies are stored on client side
- Cookies can only store strings
- Cookies can be set to a long lifespan

Sessions

- Sessions are stored on server
- Sessions can store objects
- When users close their browser, lifespan. they also lost the session

Filters- Introduction

- PHP Filter is an extension that filters the data by either sanitizing or validating it
- It plays a crucial role in security of a website, especially useful when the data originates from unknown or foreign sources, like user supplied input
- For example data from a TML form

- There are mainly two types of filters
    1. **Validation**
    2. **Sanitization**

Types of filters

- **Validation**

Used to validate or check if the data meets certain qualifications or not

Example: passing in FILTER_VALIDATE_URL will determine if the data is a valid url, but it will not change the existing data by itself

- **Sanitization**

Unlike validation, sanitization will sanitize data so as to ensure that no undesired characters by removing or altering the data

Example passing in FILTER_SANITIZE_EMAIL will remove all the characters that are inappropriate for an email address to contain. That said, it does not validate the data

Example PHP program to validate

Program to validate URL using FILTER_VALIDATE_URL filter

```php
<?php
// PHP program to validate URL

// Declare variable and initialize it to URL
$url = "https://www.geeksforgeeks.org";

// Use filter function to validate URL
if (filter_var($url, FILTER_VALIDATE_URL)) {
    echo("valid URL");
}
else {
    echo("Invalid URL");
}
?>
```

Program to validate email using FILTER_VALIDATE_EMAIL filter

```php
<?php
// PHP program to validate email

// Declare variable and initialize it to email
$email = "xyz@gmail.com";

// Use filter function to validate email
if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Valid Email";
}
else {
        echo "Invalid Email";
}

?>
```

Filter Functions

- The filter function is used to filter the data coming from insecure source

filter_var(): Filters a specific variable
filter_var_array(): Filters multiple variable i.e. array of variable
filter_has_var(): Check if the variable of specific input type exists or not
filter_id():helps to get filter id of the specified filter name
filter_list(): Returns a list of supported filter name in the form of array
filter_input(): Gets an external variable and filters it if set to do so
filter_input_array():same as filter_input() but here Gets multiple variables i.e. array of variable and filters them if set to do so

Predefined Filter Constants

- **Validate filter constants**

FILTER_VALIDATE_BOOLEAN: Validates a boolean
FILTER_VALIDATE_INT: Validates an integer
FILTER_VALIDATE_FLOAT: Validates a float
FILTER_VALIDATE_REGEXP: Validates a regular expression
FILTER_VALIDATE_IP: Validates an IP address
FILTER_VALIDATE_EMAIL: Validates an e-mail address
FILTER_VALIDATE_URL: Validates an URL

Predefined Filter Constants

- **Sanitize filter constants**

- **FILTER_SANITIZE_EMAIL:** Removes all illegal characters from an e-mail address
- **FILTER_SANITIZE_ENCODED:** Removes/Encodes special characters
- **FILTER_SANITIZE_MAGIC_QUOTES:** Apply addslashes() function
- **FILTER_SANITIZE_NUMBER_FLOAT:** Remove all characters, except digits, +- and optionally ., eE
- **FILTER_SANITIZE_NUMBER_INT:** Removes all characters except digits and + –
- **FILTER_SANITIZE_SPECIAL_CHARS:** Removes special characters
- **FILTER_SANITIZE_FULL_SPECIAL_CHARS** Encoding quotes can be disabled by using FILTER_FLAG_NO_ENCODE_QUOTES.
- **FILTER_SANITIZE_STRING :** Removes tags/special characters from a string
- **FILTER_SANITIZE_STRIPPED :** Alias of FILTER_SANITIZE_STRING
- **FILTER_SANITIZE_URL:** Removes all illegal character from s URL

Predefined Filter Constants

- **Other filter constants**

FILTER_UNSAFE_RAW : Do nothing, optionally strip/encode special characters

FILTER_CALLBACK : Call a user-defined function to filter data

Exception Handling in PHP

- An exception is unexpected program result that can be handled by the program itself. Exception Handling in PHP is almost similar to exception handling in all programming languages

PHP provides following specialized keywords for this purpose

- **try:** It represent block of code in which exception can arise
- **catch:** It represent block of code that will be executed when a particular exception has been thrown
- **throw:** It is used to throw an exception. It is also used to list the exceptions that a function throws, but doesn't handle itself
- **finally:** It is used in place of catch block or after catch block basically it is put for cleanup activity in PHP code

Need for Exception Handling in PHP

Following are the main advantages of exception handling over error handling

- **Separation of error handling code from normal code:** In traditional error handling code there is always if else block to handle errors. These conditions and code to handle errors got mixed so that becomes unreadable. With try Catch block code becomes readable

- **Grouping of error types:** In PHP both basic types and objects can be thrown as exception. It can create a hierarchy of exception objects, group exceptions in namespaces or classes, categorize them according to types

Exception handling in PHP

```php
<?php

// PHP Program to illustrate normal
// try catch block code
function demo($var) {
    echo " Before try block";
    try {
        echo "\n Inside try block";

        // If var is zero then only if will be executed
        if($var == 0)
        {

            // If var is zero then only exception is thrown
            throw new Exception('Number is zero.');

            // This line will never be executed
            echo "\n After throw (It will never be executed)";
        }
    }

    // Catch block will be executed only
    // When Exception has been thrown by try block
    catch(Exception $e) {
            echo "\n Exception Caught", $e->getMessage();
        }

        // This line will be executed whether
        // Exception has been thrown or not
        echo "\n After catch (will be always executed)";
}

// Exception will
not be rised
demo(5);

// Exception will
be rised here
demo(0);
?>
```

Output

Before try block Inside try block After catch (will be always executed) Before try block Inside try block Exception CaughtNumber is zero. After catch (will be always executed)

Using Custom Exception Class

```php
<?php
class myException extends Exception {
    function get_Message() {

        // Error message
        $errorMsg = 'Error on line '.$this-
>getLine().
                    ' in '.$this->getFile()
        .$this->getMessage().' is number
zero';
        return $errorMsg;
    }
}

function demo($a) {
    try {

        // Check if
        if($a == 0) {
            throw new myException($a);
        }
    }

    catch (myException $e) {

        // Display custom message
        echo $e->get_Message();
    }
}

// This will not generate any exception
demo(5);

// It will cause an exception
demo(0);
?>
```

Output

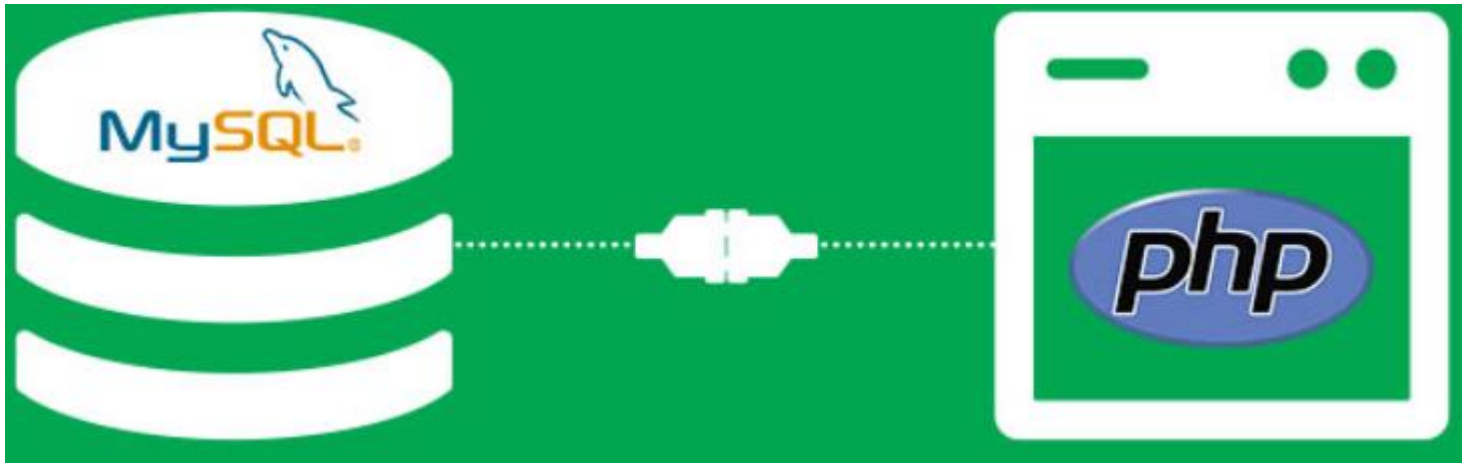
Error on line 20 in /home/45ae8dc582d50df2790517e912980806.php0 is number zero

Set a Top Level Exception Handler

- The **set_exception_handler()** function set all user defined function to all uncaught exception
- Example

```
<?php
function exception_handler($exception) {
  echo "Uncaught exception: " , $exception->getMessage(), "\n";
}

set_exception_handler('exception_handler');

throw new Exception('Uncaught Exception');
echo "Not Executed\n";
?>
```

Database Connectivity With MySQL



BASIC DATABASE SERVER CONCEPTS

- world's most popular open source database because of its consistent **fast performance, high reliability and ease of use**

BASIC DATABASE SERVER CONCEPTS

- **Database runs as a server**
  Attaches to either a default port or an administrator specified port
  Clients connect to database
  For secure systems
  authenticated connections
  usernames and passwords
  Clients make queries on the database
  Retrieve content
  Insert content
  SQL (Structured Query Language) is the language used to insert and retrieve content

MY SQL

- MySQL can be controlled through a simple command-line interface; however, we can use phpMyAdmin as an interface to MySQL

- phpMyAdmin is a very powerful tool; it provides a large number of facilities for customising a database management system

CONNECTING to MY SQL

- In order for our PHP script to access a database we need to form a connection from the script to the database management system

resourceId = mysql_connect(server, username, password);

Server is the DBMS server

username is your username
password is your password

CONNECTING to MY SQL DBMS

- In order for our PHP script to access a database we need to form a connection from the script to the database management system

resourceId = mysql_connect(server, username, password);

The function returns a resource-identifier type

- a PHP script can connect to a **DBMS** **anywhere in the world**, so long as it is connected to the internet

- We can also connect to multiple DBMS at the same time

SELECTING A DATABASE

- **Once connected to a DBMS, we can select a database**

  mysql_select_db(databasename, resourceId);

Create HTML form for connecting to database

- you need to create a working folder first and then create a web page with the name “contact.html”
- If you install xampp your working folder is in folder this “E:\xampp\htdocs”
- You can create a new folder “contact” on your localhost working folder
- Create a “contact.html” file and paste the following code

Code create HTML form for connecting to database

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Contact Form - PHP/MySQL Demo Code</title>
</head>

<body>
<fieldset>
<legend>Contact Form</legend>
<form name="frmContact" method="post" action="contact.php">
<p>
<label for="Name">Name </label>
<input type="text" name="txtName" id="txtName">
</p>
<p>
<label for="email">Email</label>
<input type="text" name="txtEmail" id="txtEmail">
</p>
<p>
<label for="phone">Phone</label>
<input type="text" name="txtPhone" id="txtPhone">
</p>
<p>
<label for="message">Message</label>
<textarea name="txtMessage" id="txtMessage"></textarea>
</p>
<p> </p>
<p>
<input type="submit" name="Submit" id="Submit" value="Submit">
</p>
</form>
</fieldset>
</body>
</html>
```

Create a PHP page to save data from HTML form to your MySQL database

- The contact HTML form action is on “contact.php” page. On this page, we will write code for inserting records into the database.
- For storing data in MySQL as records, you have to first connect with the DB. Connecting the code is very simple. The mysql_connect in PHP is

mysqli_connect.

Example:

```
$con = mysqli_connect("localhost","your_localhost_database_user",
 "your_localhost_database_password","your_localhost_database_db");
```

Local Host

- You need to place value for your localhost username and password. Normally localhost MySQL database username is root and password blank or root

Example

```
$con = mysqli_connect('localhost', 'root', '',’db_contact’);
The “db_contact” is our database name that we created before.
After connection database you need to take post variable from the form. See the below code
$txtName = $_POST['txtName'];
$txtEmail = $_POST['txtEmail'];
$txtPhone = $_POST['txtPhone'];
$txtMessage = $_POST['txtMessage'];
```