Chapter 4 – PHP

PHP: Introduction, Creating and running a PHP script, variables, constants, data types, arrays and functions, Reading and processing data from web form controls.

MySql: Connecting to MySql server using PHP, creating new database and tables, reading and displaying table data, Inserting, detection and updating records in database table using PHP..

Introduction:

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- It is a widely-used, open source scripting language
- PHP scripts are executed on the server
- It is free to download and use
- It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
- It is deep enough to run the largest social network (Facebook)!
- It is also easy enough to be a beginner's first server side language!
- Runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Compatible with almost all servers used today (Apache, IIS, etc.)
- Supports a wide range of databases
- Easy to learn and runs efficiently on the server side

PHP Can:

- generate dynamic page content
- create, open, read, write, delete, and close files on the server
- collect form data

- send and receive cookies
- add, delete, modify data in your database
- be used to control user-access
- encrypt data

PHP Files:

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

Creating and running a PHP Script:

- A PHP script can be placed anywhere in the document.
- Starts with **<?php** and ends with **?>**
- PHP files need to be saved with extension .php.
- **Syntax:**

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

- Example:

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

Running a PHP Script:

- Save the script to a location under 'web server root' and name it.
- Then enter the URL "http://localhost/Example.php in the web browser and click the Enter key.
- Using XAMPP Server:
    - Go to XAMPP server directory

      In Windows root server directory is "C:\xampp\htdocs\".
    - Create hello.php

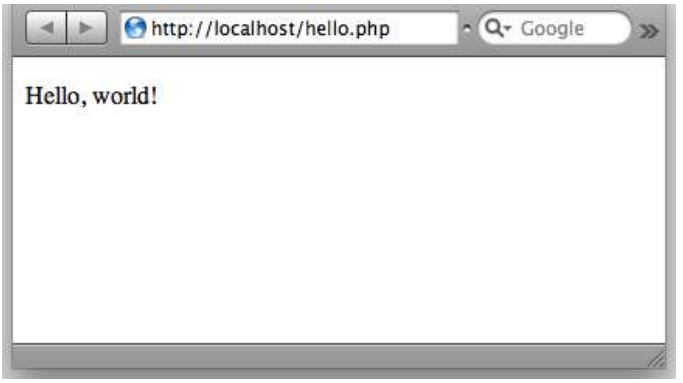
      Create a file and name it "hello.php"

      ```
      <?php
          echo "Hello, world"
      ?>
      ```
    - Open New Tab

      Run it by opening a new tab in your browser
    - Load hello.php

      On browser window type http://localhost/hello.php

Output:



Variables:

Creating/Declaring PHP Variables:

- In PHP, a variable starts with the $ sign, followed by the name of the variable.
- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
- Rules for PHP variables:
  - starts with the $ sign, followed by the name of the variable
  - must start with a letter or the underscore character
  - cannot start with a number
  - can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
  - Case-sensitive.
- Example:

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

The scope of variables:

- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
  - global
  - static

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

```
<?php
        $x = 5; // global scope
        function myTest() {
            // using x inside this function will
        generate an error
            echo "<p>Variable x inside function is:
        $x</p>";
        }
        myTest();
        echo "<p>Variable x outside function is:
        $x</p>";
?>
```

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

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>"; }
myTest();
// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

The Life time of a variable:

- Life time of a variable is confined to its Scope.
- Local variables will be destroyed once the end of the code block is reached. Hence the same named variables can be declared within different local scopes.

- Variables in global scope can be accessed from anywhere from outside a function or class independent of its boundary.

---

constants

PHP Constants are variables whose values, once defined, cannot be changed, and these constants are defined without a $ sign in the beginning. PHP Constants are created using define() function. This function takes two parameters first is the name, and the second is the value of the constant defined.

The name of the constant starts using letters or underscores and not with a number. It can start with a letter or underscore followed by letters, underscores or numbers. The name is case-sensitive and in uppercase. After a constant is defined, it cannot be undefined or redefined again. It remains the same throughout the script and cannot be changed as the variables do.

A constant is a name for a particular value. To define a constant, we have to use the define() function and to get the value of the constant; we just need to specify the name.

Syntax:

define(name, value, case-insensitive);

where name is the name of the constant,

value is the value of the constant,

case-insensitive is either true or false, by default, it is false.

define('TEXT', 'Hello World!');

A constant can also be defined using const construct.

```
<?php
const MSG = "WELCOME";
echo MSG;
```

```
?>
<?php
// program to demonstrate in PHP 7 using const keyword
const TEXT = 'PHP PROGRAMMING!';
echo TEXT;
echo constant("TEXT");
?>
```

Data Types:

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String:

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

Example:

```
<?php
$x = "Hello world!";
$y = 'Hello world!';
echo $x;
echo "<br>";
echo $y;
```

```
?>
```

PHP Integer:

- An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
- Rules for integers:
  - must have at least one digit
  - must not have a decimal point
  - can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)
- In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$x = 5985;
var_dump($x);
?>
```

PHP Float:

- A float (floating point number) is a number with a decimal point or a number in exponential form.
- In the following example $x is a float. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$x = 10.365;
var_dump($x);
?>
```

PHP Boolean:

- A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

- Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP 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:

Example

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

PHP Object:

- An object is a data type which stores data and information on how to process that data.
- In PHP, an object must be explicitly declared.

Example:

```
<?php
class Car {
   function Car() {
      $this->model = "VW";      }
}
// create an object
$herbie = new Car();
// show object properties
```

```
echo $herbie->model;
?>
```

PHP NULL Value:

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.
- Variables can also be emptied by setting the value to NULL

Example:

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

PHP 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.
- A common example of using the resource data type is a database call.

Functions:

- A function is a block of statements that can be used repeatedly in a program.
- will not execute immediately when a page loads.
- 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;
}
```

Example:

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

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.

Example:

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

familyName("Hege", "1975");
familyName("Stale", "1978");
```

```
familyName("Kai Jim", "1983");
?>
```

Returning values:

- To let a function return a value, use the `return` statement:

Example

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

Arrays:

- An array is a special variable, which can hold more than one value at a time.
- An array can hold many values under a single name, and you can access the values by referring to an index number.

Create an Array in PHP:

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

```
array();
```

In PHP, there are three types of arrays:

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

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";
```

The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

Example

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

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.";
?>
```

Multidimensional Arrays:

- A multidimensional array is an array containing one or more arrays.

Example:

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

<?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>";
?>
```

Processing a Web form:

- Forms are used to collect user input

- Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.
- The PHP superglobals $_GET and $_POST are used to collect form-data.
- $_GET is an array of variables passed to the current script via the URL parameters.

  Syntax: $_GET['name of the form field']
- $_POST is an array of variables passed to the current script via the HTTP POST method.
- Syntax: $_POST['name of the form field']

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>
```

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

Validating Login form using PHP:

```
<html>
<body>
<center>
<form name="f1" action="Login.php" method=”post”>
<label> Username: </label>
<input type="text" name="t1"><br><br>
<label> Password: </label>
<input type="password" name="t2"><br><br>
<input type="submit" value="SUBMIT" >
<input type="reset" value="RESET" >
</form>
</center>
</body>
</html>
```

Login.php:

```
<?php
$u=$_POST["t1"];
$u=$_POST["t2"];
$msg="";
$flag=true;
if($u=="")
{
        $flag=false;
        $msg="Please enter Username";
}
else if($p=="")
{
        $flag=false;
        $msg="Please enter Password";
```

```
}
else if(strlen($p)<6)
{
        $flag=false;
        $msg="Password should be minimum of 6
characters";
}
if($flag==false)
        echo $msg;
else
        echo "Login Success <br> Welcome $u";
?>
```

Introduction to MYSQL:

- MySQL is an open source, fast reliable, and flexible relational database management system, used with PHP.
- It is a database system, used for developing web-based software applications.
- used for both small and large applications.
- It is a relational database management system *(RDBMS).*
- Fast, reliable and flexible and easy to use.
- Supports standard SQL *(Structured Query Language).*
- Free to download and use.
- Presently developed, distributed, and supported by Oracle Corporation.
- Written in C, C++.

Main Features of MySQL:

- MySQL server design is multi-layered with independent modules.

- It is fully multithreaded by using kernel threads. It can use multiple CPUs if they are available.
- Provides transactional and non-transactional storage engines.
- It has a high-speed thread-based memory allocation system.
- Supports in-memory heap table.
- Handles large databases.
- Works in client/server or embedded systems.
- Works on many different platforms.

Connecting to MySql Server using PHP:

Open a Connection to MySQL:

- Before we can access data in the MySQL database, we need to be able to connect to the server:

```
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = mysqli_connect($servername,
$username, $password);
// Check connection
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Create a MySQL Database:

The CREATE DATABASE statement is used to create a database in MySQL

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$conn = mysqli_connect($servername,
$username, $password);
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " .
mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Create a MySQL Table:

- A database table has its own unique name and consists of columns and rows.
- The CREATE TABLE statement is used to create a table in MySQL.

Example: Create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
$conn = mysqli_connect($servername, $username,
$password, $dbname);
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " .
mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Insert Data Into MySQL:

- Some syntax rules to follow:
  - The SQL query must be quoted in PHP
  - String values inside the SQL query must be quoted
  - Numeric values must not be quoted
  - The word NULL must not be quoted

- The INSERT INTO statement is used to add new records to a MySQL table:
- **Syntax:** INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
$conn = mysqli_connect($servername,
$username, $password, $dbname);
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}
$sql = "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('John', 'Doe',
'john@example.com')";
if (mysqli_query($conn, $sql)) {
    echo "New record created
successfully";
} else {
    echo "Error: " . $sql . "<br>" .
mysqli_error($conn);
}
mysqli_close($conn);
?>
```

- **Select Data From a MySQL Database:**
- The SELECT statement is used to select data from one or more tables:
  - SELECT column_name(s) FROM table_name
- or we can use the * character to select ALL columns from a table:
  - SELECT * FROM table_name

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
$conn = mysqli_connect($servername,
$username, $password, $dbname);
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}
$sql = "SELECT id, firstname, lastname FROM
MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
        while($row =
mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name:
" . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
mysqli_close($conn);
?>
```

- **Delete Data From a MySQL Table:**
  - The DELETE statement is used to delete records from a table:

DELETE FROM table_name WHERE some_column = value

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
$conn = mysqli_connect($servername,
$username, $password, $dbname);
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " .
mysqli_error($conn);
}
mysqli_close($conn);
?>
```

- **Update Data In a MySQL Table:**
    - The UPDATE statement is used to update existing records in a table:

UPDATE table_name SET column1=value, column2=value2,...

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
$conn = mysqli_connect($servername,
$username, $password, $dbname);
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}
$sql = "UPDATE MyGuests SET lastname='Doe'
WHERE id=2";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " .
mysqli_error($conn);
}
mysqli_close($conn);
?>
```