CH 5

PHP

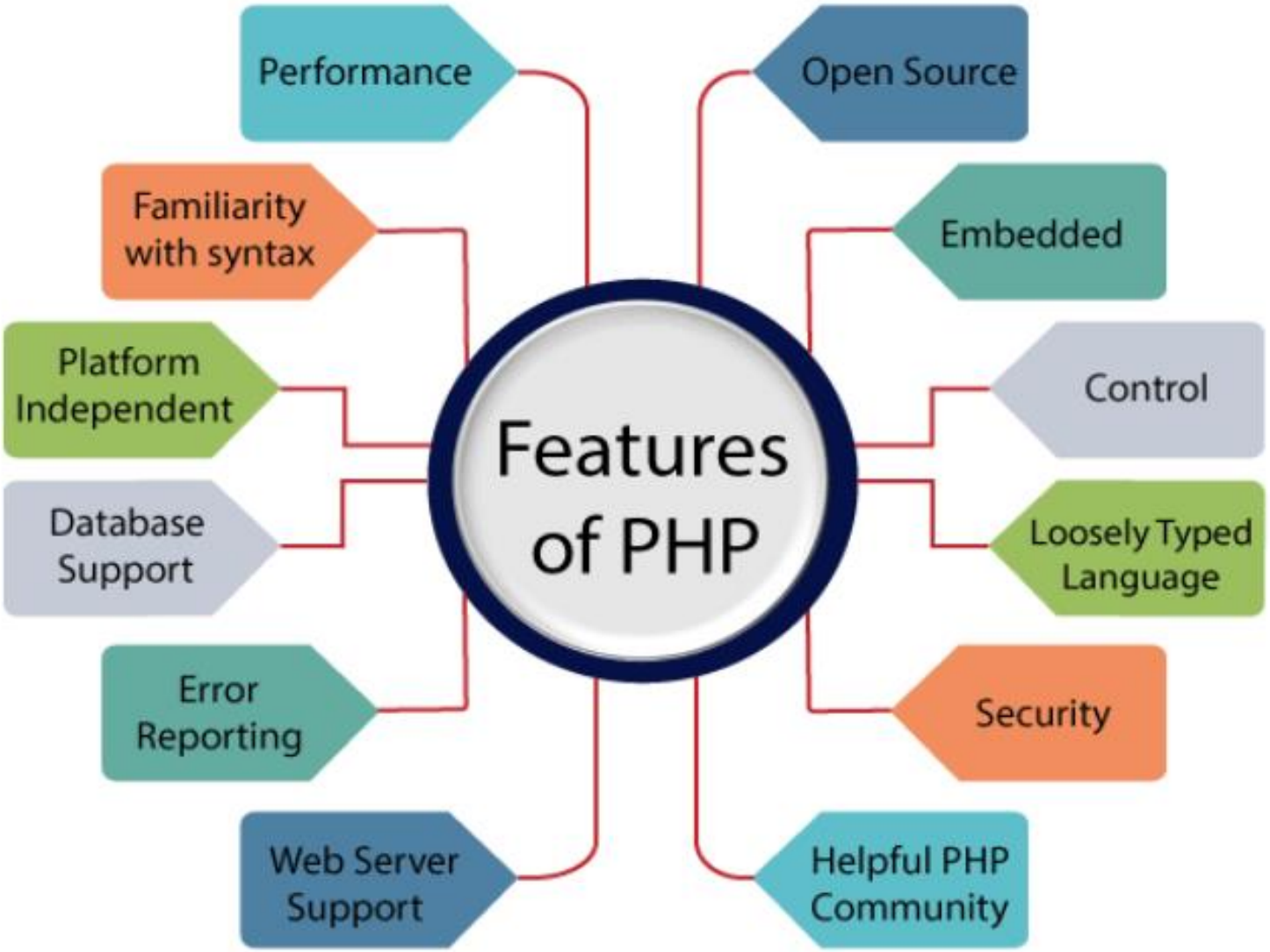
- 5.1. Installing and Configuring PHP
- 5.2. Introduction
- 5.2.1. PHP and the Web Server Architecture, PHP Capabilities
- 5.2.2. PHP and HTTP Environment Variables
- 5.2.3. Variables
- 5.2.4. Constants
- 5.2.5. Data Types
- 5.2.6. Operators
- 5.2.7. Working with Arrays
- 5.3. Decision Making, Flow Control and Loops
- 5.4. Introduction to Laravel
- 5.5. Creating a Dynamic HTML Form with PHP
- 5.6. Database Connectivity with MySQL
- 5.6.1. Performing basic database operations (CRUD)
- 5.7. Using GET, POST, REQUEST, SESSION,and COOKIE Variables

Introduction

- PHP was originally an acronym for Personal Home Pages.
- Now a recursive acronym for **PHP: Hypertext Preprocessor.**
- PHP was created by **Rasmus Lerdorf in 1994** but appeared in the market in 1995.
- Latest version of PHP is **PHP 8.1. 19** Released! 11 May 2023.
- PHP is neither real programming language .
- PHP is a general-purpose scripting language widely used as a server-side language for creating dynamic web pages.
- Though its reputation is mixed, PHP is still extremely popular and is used in over 75% of all websites where the server-side programming language is known.

- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.

PHP Features



- **Performance:**
- PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP.
- **Open Source:**
- PHP source code and software are freely available on the web.
- **Familiarity with syntax:**
- PHP has easily understandable syntax. Programmers are comfortable coding with it.
- **Embedded:**
- PHP code can be easily embedded within HTML tags and script.
- **Platform Independent:**
- PHP is available for WINDOWS, MAC, LINUX & UNIX operating system.

- **Database Support:**
- PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.
- **Error Reporting -**
- PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.
- **Loosely Typed Language:**
- PHP allows us to use a variable without declaring its datatype.
- **Web servers Support:**
- PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.
- **Security:**
- PHP is a secure language to develop the website

- **Control:**
- Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code

Lexical structure

- The lexical structure of a programming language is the set of basic rules that governs how you write programs in that language.
- Case Sensitivity:
- PHP is case sensitive language.
- Names of user defined classes and functions, built in constructs and keywords are case insensitive but variables are case sensitive.
- Ex.
- Echo(“hello”)
- echo(“hello”)
- ECHO(“hello”) all three statements are same
- But variable name $name and $NAME are different

– Statements and semicolons:

–A statement is a collection of PHP code that does something.

– PHP uses semicolon(;) to separate statements.

–Comments

–All comments in PHP follow the # character.

–Ex.

–<?php

–#This is comment

–// single line comment

–/* This is multiline comment */

–?>

Define a variable

- A variable has a name and is associated with a value. To define a variable, you use the following syntax:
- $variable_name = value;
- When defining a variable, you need to follow these rules:
  - The variable name must start with the dollar sign ($).
  - The first character after the dollar sign ($) must be a letter (a-z) or the underscore (_).
  - The remaining characters can be underscores, letters, or numbers.

- PHP variables are case-sensitive. It means that **$message** and **$Message** variables are entirely different.
- Ex.
- <?php
- $title = "PHP is awesome!";
- ?>
- Shorter Way:
- <?= $variable_name ?>

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PHP Variables</title>
</head>
<body>
    <?php
            $title = 'PHP is awesome!';
    ?>
    <h1><?= $title; ?></h1>
</body>
</html>
```

Constants:

- PHP constants can be defined by 2 ways:
- Using define() function
- Using const keyword
- **define()**
- Syntax:
- define(name, value, **case**-insensitive)
- **name:** It specifies the constant name.
- **value:** It specifies the constant value.
- **case-insensitive:** Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.
- Ex. Define (“PI” 3.14);

- **const keyword**
- The constant defined using const keyword are **case-sensitive**.
- Example:
- **<?php**
- const MESSAGE="Hello";
- echo MESSAGE;
- **?>**

Basic structure of php

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**
- <?php
  // PHP code goes here
  ?>
- The default file extension for PHP files is ".php".

- **How to run PHP Program:**
- If you have installed XAMPP, all of your PHP code files will be placed in “htdocs” folder. The full path of htdocs folder will be something like c:\xampp\htdocs.
- Open notepad and create a simple program. You can copy the above example as well.
- Then save your file with PHP extension such as sample.php. You need to place or Save your **‘sample.php**’ file in htdocs folder.
- Now you can open any web browser such as chrome and run your PHP file using the localhost URL.
- For example, if your file is located: **‘c:\xampp\htdocs\sample.php**’, You need to type in browser URL: ‘http://localhost/sample.php’ and hit enter. You PHP program will run and give output such as ‘Hello World’

Operators

- **String Concatenation Operator**
  - String concatenation operator is dot (.).
  - Ex.
  - <?php
  - $a=“Hello”;
  - $b=$a.”World”;
  - Echo $b; ?>
- Increment & Decrement Operator (++, --)
- Arithmetic Operator (+,-,*,/, %)
- Comparison Operator (<,>,==,!=,<>,<=,>=)
- Logical Operator: (&&, ||, !)

```
<?php
$a = 18;
$b = 4;
echo "$a + $b = " . ($a + $b); //addition
echo "\n";
echo "$a - $b = " . ($a - $b); //subtraction
echo "\n";
echo "($a * $b) = " . ($a * $b); //multiplication
echo "\n";
echo "$a / $b = " . ($a / $b); //division
echo "\n";
echo "$a % $b = " . ($a % $b); //modulus
echo "\n";
?>
```

Control structure

- If statement:
- Syntax:
- If (condition)
- {
- }
- If else statement:
- Syntax:
- If (condition)
- {
- }
- else{
- }

- **Example:**
- <?php
- $a=“hello”;
- $b=“java”;
- If($a==$b)
  - echo “Both Strings are equal”;
- else
  - echo “Both Strings are not equal”;
- ?>

- Switch:

```
<?php
   $ch = ‘u';
   switch ($ch)
   {
      case 'a':
         echo "Given character is vowel";
         break;
      case 'e':
         echo "Given character is vowel";
         break;
      case 'i':
         echo "Given character is vowel";
         break;
      case 'o':
         echo "Given character is vowel";
         break;
      case 'u':
         echo "Given character is vowel";
         break;
 }
?>
```

- while:
- Synatx:
- while (condition)
- {
- }
- Example:
- <?php
- $i=1;
- while($i<=10)
- {
  - echo $i. " ";
  - $i++;
- }

- For loop:
- for (*init counter; test counter; increment counter*)
- {
  code to be executed;
  }
- Example:
- <?php
- For($i=0;$i<=10;$i++)
- {
  echo $i. “ “;
- }

- **foreach loop:**
- Syntax:
- foreach($array as $current)
- {
- }
- Example:
- <?php
- $arr=array(1,2,3,4,5);
- foreach($arr as $value)
- {

  echo $value. “ “;
- }

PHP Variable Scope

- The scope of a variable is defined as its range in the program under which it can be accessed.
- PHP has three types of variable scopes:
- Local variable
- Global variable
- Static variable

- Local variable
- The variables that are declared within a function are called local variables for that function.
- These local variables have their scope only in that particular function in which they are declared.
- A variable declaration outside the function with the same name is completely different from the variable declared inside the function.

- Example:
- <?php
- **function** local_var()
- {
- $num = 45; //local variable
- echo "Local variable declared inside the functi on is: ". $num;
- }
- local_var();
- ?>

- Global variable:
- The global variables are the variables that are declared outside the function.
- These variables can be accessed anywhere in the program.
- To access the global variable within a function, use the **GLOBAL** keyword before the variable.
- these variables can be directly accessed or used outside the function without any keyword.

- Example:
- <?php
- $name = “ABC"; //Global Variable
- **function** global_var()
- {
- **global** $name;
- echo "Variable inside the function: ". $name;
- echo "</br>";
- }
- global_var();
- echo "Variable outside the function: ". $name;
- ?>

- Static variable
- use the static keyword before the variable to define a variable, and this variable is called as **static variable**.
- Static variables exist only in a local function, but it does not free its memory after the program execution leaves the scope.

- **Example:**
- <?php
- **function** static_var()
- {
- **static** $num1 = 3; //static variable
- $num2 = 6; //Non-static variable
- //increment in non-static variable
- $num1++;
- //increment in static variable
- $num2++;
- echo "Static: " .$num1 ."</br>";
- echo "Non-static: " .$num2 ."</br>";
- }
- //first function call
- static_var();
- //second function call
- static_var();
- ?>

$_GET Method

- Get request is the default form request.
- The data passed through get request is visible on the URL browser so it is not secured.
- You can send limited amount of data through get request. (max. 100 characters).
- **The $_GET variable is used to collect values from a form with method="get".**
- The $_GET variable is an array of variable passed to current script via the URL parameters.

- **Example:**
- <?php
  echo "Welcome ". $_GET['name']. "
";
  echo "You are ". $_GET['age']. " years old.";}
  ?>
- <html> <body>
- <form action = “Get_Ex.php" method = "GET">
- Name: <input type = "text" name = "name" />
- Age: <input type = "text" name = "age" />
- <input type = "submit" /> </form> </body> </html>

$_POST Method

- Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.
- The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.
- **The $_POST variable is used to collect values from a form with method="post".**
- The $_POST variable is an array of variable names and values sent by the HTTP POST method.

```
<?php
 echo "Welcome ". $_POST['name']. "<br />";
  echo "You are ". $_POST['age']. " years old.";
?>
<html> <body>
<form action = “Post_Ex.php" method = “POST">
 Name: <input type = "text" name = "name" />
Age: <input type = "text" name = "age" />
 <input type = "submit" />
 </form> </body> </html>
```

$_request method

- The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

- <?php

  echo "Welcome ". $_REQUEST['name']. "
";

  echo "You are ". $_REQUEST['age']. " years old.";

  ?>
- <html> <body>

  <form action = “Req_ex.php" method = "POST">

  Name: <input type = "text" name = "name" />

  Age: <input type = "text" name = "age" />

  <input type = "submit" /> </form> </body

  </html>

PHP Functions

- PHP Functions can be seen as ‘block of code’ or ‘set of statements’ that perform some tasks when executed.
- A function usually takes some input arguments, perform some action on them and returns some result.
- Example:
- **<?php**
- **function greetings()**
- **{**
- **echo "Hello";**
- **}**
- **greetings();**
- **?>**

- **PHP Function With Arguments:**
- <?php
- function add($num1, $num2)
- {
- $total = $num1 + $num2;
- echo "Sum of the two numbers is : $total";
- }
- add(10, 25);
- ?>

- **Function Returning Values**
- <?php
- function add($x, $y)
- {
- $z = $x + $y;
- return $z;
- }
- echo "The sum of 5 + 10 is: " . add(5, 10);
- ?>

- **/*Program to calculate area of rectangle*/**
- <?php
- function rect_area($length = 2, $width = 4)
- {
- $area = $length * $width;
- echo "Area Of Rectangle with length " . $length . " & width " . $width . " is " . $area ;
- }
- rect_area(); // function has been called.
- ?>

```
/*Factorial program in PHP using recursive function*/
<?php
function fact($number)
{
    if ($number < 2)
            return 1;
     else
        return ($number * fact($number-1));
 }
 echo fact(4);
?>
```

- **Function With Default Parameters Value**
- <?php
- function setAge($age = 25)
- {
- echo "My age is: $age";
- }

- setAge(50);
- echo "
";
- setAge(); // This time function will use the default value of 25
- echo "
";
- setAge(35);
- ?>

PHP echo and print Statements

- echo and print are more or less the same.
- They are both used to output data to the screen.
- The differences are small:
- echo has no return value while print has a return value of 1 so it can be used in expressions.
- echo can take multiple parameters while print can take one argument.
- echo is marginally faster than print.

- **Example(echo):**
- <?php
- $x = 5;
- $y = 4;
- echo "Addition " . ($x + $y);
- ?>
- **Example(print)**
- <?php
- $x = 5;
- $y = 4;
- print "Addition " . ($x + $y);
- ?>

Working with Arrays

- Arrays are the data structure where you can store data.
- **Types of Arrays in PHP.**
- There are three kinds of arrays that you can make. These are listed below.
- **Indexed arrays** – Arrays with a numeric index
- **Associative arrays** – Arrays with named keys
- **Multidimensional arrays** – Arrays containing one or more arrays

- **Indexed Arrays:**
- indexed array is an array where all key values are numeric and always start from zero.
- Example:
- <?php
- $colors = array("Red", "Green", "Blue");
- print_r($colors);
- ?>

- **print_r()**
- The print_r() function is useful for debugging purpose
- It prints content of array, objects and other things in human readable form.
- For Ex.
- $a=array(‘name’=> ‘aaa’ , ‘age’=> 40);
- print_r($a);
- Output:
- Array {
- [name]=> aaa
- [age]=> 40 }

- **isset():**
- The isset() function checks whether a variable is set, which means that it has to be declared and is not NULL.
- This function returns true if the variable exists and is not NULL, otherwise it returns false.
- **Note:** If multiple variables are supplied, then this function will return true only if all of the variables are set.
- **Tip:** A variable can be unset with the unset() function.

- **Syntax:**
- isset(*variable, ...*);

- **Example:**
- **<?php**
- $a = 20;
- if (isset($a))
- { echo "The Variable 'a' is now set.
"; }
- else { echo "The Variable 'a' is now unset.
"; }
- $b = null;
- if (isset($b))
- { echo "The Variable 'b' is now set.
"; }
- else { echo "The Variable 'b' is now unset.
"; }
- ?>

- **var_dump()**
- The var_dump() function is used to display structured information (type and value) about one or more variables.
- **Example**
- <?php
  $name=“aaa";
  $age=24;
  var_dump($name);
  var_dump($age);
  ?>
- **Output:**
- String(3) “aaa”
- int(24)

- **Associative Array**
- Associative arrays are arrays that use named keys that you assign to them using => symbol.
- **Example:**
- <?php
- $ages = array("Peter"=>22, "Clark"=>32, "John"=>28);
- print_r($ages);
- ?>

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

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

- **Multidimensional Arrays**
- A multidimensional array is an array containing one or more arrays.
- An example of a multidimensional array will look something like this:
- $row0=array(1,2,3);
- $row1=array(4,5,6);
- $row2=array(7,8,9);
- $multi=array($row0,$row1,$row2);
- $val=$multi[2][0]; //2nd row 0th column $val=7

```
<?php
$emp=array
(
        array(1,"aa",400000),
        array(2,"bbb",500000),
        array(3,"ccc",300000)
);
for($row=0;$row<3;$row++)
{
  for($col=0;$col<3;$col++)
    {
        echo $emp[$row][$col]."    ";
    }
  echo"<br/>";
}  ?>
```

Sorting Arrays

- functions most commonly used for sorting arrays.
- sort() and rsort() — For sorting indexed arrays
- asort() and arsort() — For sorting associative arrays by value
- ksort() and krsort() — For sorting associative arrays by key

sort()

- The sort() function is used for sorting the elements of the indexed array in ascending order <?php

- $colors = array("Red", "Green", "Blue", "Yellow");
- sort($colors);
- print_r($colors);
- ?>
- o/p : Array ([0] => Blue [1] => Green [2] => Red [3] => Yellow)
- Similarly you can sort the numeric elements of the array in ascending order.

rsort()

- The rsort() function is used for sorting the elements of the indexed array in descending order
- <?php
- $colors = array("Red", "Green", "Blue", "Yellow");
- rsort($colors);
- print_r($colors);
- ?>
- Array ([0] => Yellow [1] => Red [2] => Green [3] => Blue)
- Similarly you can sort the numeric elements of the array in descending order.

asort()

- The asort() function sorts the elements of an associative array in ascending order according to the value.
- <?php
- $age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);
- asort($age);
- print_r($age);
- ?>
- Output:
- Array ([Harry] => 14 [Peter] => 20 [Clark] => 35 [John] => 45)

arsort()

- The arsort() function sorts the elements of an associative array in descending order according to the value.
- <?php
- $age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);
- arsort($age);
- print_r($age);
- ?>
- Array ([John] => 45 [Clark] => 35 [Peter] => 20 [Harry] => 14)

ksort()

- The ksort() function sorts the elements of an associative array in ascending order ascending to the key.
- <?php
- $age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);
- ksort($age);
- print_r($age);
- ?>
- Array ([Clark] => 35 [Harry] => 14 [John] => 45 [Peter] => 20)

krsort()

- The krsort() function sorts the elements of an associative array in descending order ascending to the key.
- <?php
- $age = array("Peter"=>20, "Harry"=>14, "John"=>45, "Clark"=>35);
- krsort($age);
- print_r($age);
- ?>
- Array ([Peter] => 20 [John] => 45 [Harry] => 14 [Clark] => 35)

- **Getting size of array:**
- The count() and sizeof() is used to return the number of elements in array.
- Syntax: count($array);
- Example:
- $a=array(1,2,3,4,5);
- $s=count($a); or $s=sizeof($a);
- Output: 5

- **Traversing Arrays:**
- Traversing an array means to visit each and every element of array using looping structure or iterator functions.
- There are several ways to traverse arrays in PHP
- foreach loop:
- Example:
- $a=array(‘a’,’b’,’c’.’d’.’e’);
- Foreach($a as $value)
- {
- Echo “$value\n”;
- }

- **include and require:**
- It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.
- **The include and require statements are identical, except upon failure:**
- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

- if you want the execution to go on and show users the output, even if the include file is missing, use the include statement.
- Otherwise, in case of FrameWork, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution.
- This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

- However, there is one big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute.
- If we do the same example using the require statement, the echo statement will not be executed because the script execution dies after the require statement returned a fatal error.

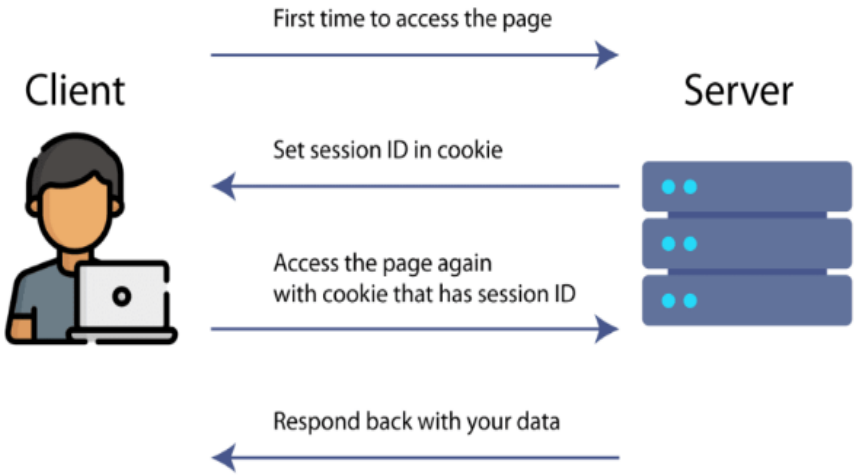
- **Syntax:**
- include '*filename*';
  or
  require '*filename*';
- Example:
- **a.php**
- <?php
  $color='red';
  $car='BMW';
  ?>

- **Main.php**
- <html>
  <body>
  <h1>Welcome to my home page!</h1>
  <?php include ’a.php';
  echo "I have a $color $car.";
  ?>
  </body>
  </html>

Cookies and Session

Sr No	Session	Cookies
1	Data are stored on server side	Data stored on client side or users browser.
2	Session data are more secure because they never travel on every HTTP request.	Cookies data are less secure because they travel with each and every HTTP request.
3	You can store objects in session(store large amount of data)	You can store only string type of data in cookies (Max file size 4kB)
4	Session can not be used for future references. (default expire time 24min)	Cookies mostly used for future references

- Cookies are automatically saved whenever a new web page is opened or reloaded.
- Whenever cookies request user information from the server. The server sets a Session ID in the cookies. The server uses that session ID to identify the cookies where the request is coming from.



- **Cookies:**
- Cookies are small files of information that are sent to a browser to store a user's information from a particular visited website.
- Cookies stores user information from a website in the browser only and use that information to identify the user when next the user tries to use visit the same website in the browser

- **Setting Cookies in PHP:**
- **Syntax:**
- Setcookie (name, value, expire, path, domain, secure)
- The setcookie() function should be called first before any other code is called or executed.

Parameter	Description
Name	This contains the name of the cookie.
Value	This contains the value of the cookie. This could be in a string or integer form
Expire	This will contain the expiration date, of the cookie. If omitted, it will take the default value(0s), and immediately after the user reloads or closes the web page the data in the cookies will be lost. **Optional** (time() + 3600 for 1 hour.)
Path	This will contain the path of the cookie in the webserver. **Optional.** The forward slash "/" means that the cookie will be made available on the entire domain
Domain	This will contain the domain works. For example, www.example.com. **Optional**
Secure	**Optional.** the default is false It is used to determine whether the cookie is sent via https if it is set to true or http if it is set to false.

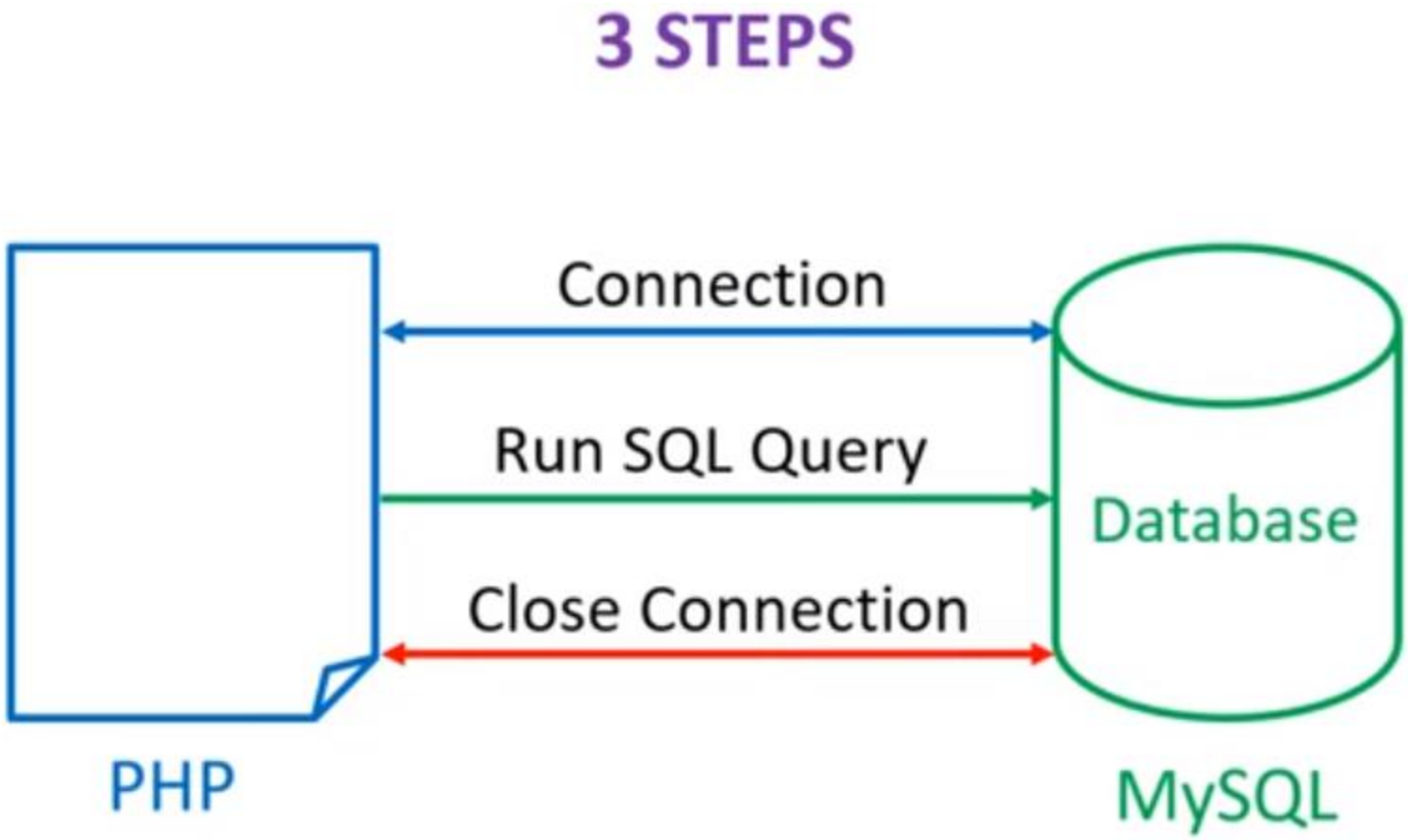
- **Example:**
- <?php
- //Setting cookie
- setcookie('Username', 'Dennis', time() + 86400, '/');
- ?>
- <html lang="en">
- <body>
- <?php
- if (isset($_COOKIE['Username']))
- {
- echo "The Username is" . $_COOKIE['Username'];
- }
- else
- echo "No Username is found";
- ?>
- </body>
- </html>

- **Sessions in PHP**
- A session is a global variable stored on the server.
- Each session is assigned a unique id which is used to retrieve stored values.
- Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL
- Sessions have the capacity to store relatively large data compared to cookies.
- The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.
- Just like the $_COOKIE array variable, session variables are stored in the $_SESSION array variable. Just like cookies, the session must be started before any HTML tags.

- **Start a Session in PHP**
- first call the PHP session_start function and then store your values in the $_SESSION array variable.
- $ _SESSION[identifier] = value;
- Example:
- **<?php**
- session_start();
- **?>**
- <!DOCTYPE html>
- <html>
- <body>
  <?php
- // Set session variables
- $_SESSION["name"] = “DY Patil College";
- $_SESSION["ID"] = "123";
- **?>**
  </body>
- </html>

- **how we can retrieve session variables in PHP**
- **<?php**
- session_start();
- **?>**
- <!DOCTYPE html>
- <html>
- <body>
  <?php
- // Echo session variables that were set on previous page
- echo "College name is " . $_SESSION["name"] . ".
";
- echo "College ID is " . $_SESSION["ID"] . ".";
- **?>**
  </body>
- </html>

Database


3 STEPS
Connection
Run SQL Query
Close Connection
Database
PHP
MySQL

- 1. connection:
- mysqli_connect(servername, username, password, database)

- 2.Run SQL query:
- mysqli_query(connection_name,query)

- 3.Close connection:
- mysqli_close(connection_name)

CRUD Operations

- CRUD stands for create, read, update, and delete.
- **1. Create:**
- In CRUD operations, 'C' is an acronym for **create,** which *means to add or insert data into the SQL table.* So, firstly we will create a table using CREATE command and then we will use the INSERT INTO command to insert rows in the created table.

```
<?php
$servername = "localhost";
$username = "root";
$password = " ";
$dbname = "mca_db";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error)
 {
  die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests2 (id INT(6) PRIMARY KEY,firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,email VARCHAR(50))";

if ($conn->query($sql) === TRUE) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}
$conn->close();
?>
```

2. Insert

Example 1:

```
<?php
 $con=mysqli_connect("localhost", "root", "", “db1");
 if(mysqli_connect_error())
      echo "Connection Error.";
  else
      echo "Database Connection Successfully.";
mysqli_query($con, "insert into my_team values(1, 'Shikhar', 'Dhawan', 'Delhi', 'India')");
 echo "Records Inserted ...“;
 mysqli_close($con);
?>
```

- **3.Update:**
- <html>
- <head>
- <title>PHP MySQL connection example</title>
- </head>
- <body>
- <form method="post">
- Enter Student ID : <input type="text" name="id">
- Enter Student Phone no : <input type="text" name="pno">
- <input type="submit" value="Update" name="Submit1">
- </form>
- </html>

- <?php
- if(isset($_POST['Submit1']))
- {
- $n=$_POST['id'];
- $pn=$_POST['pno'];
-
- $con = mysqli_connect("localhost", "root"," ", "fy"); //Creating a connection
- $sql = "update stud set phonenumber=$pn where sno=$n";

```
if(mysqli_query($con,$sql))
{
   echo "record updated";
}
$query = "select * from stud where sno=$n"; //Executing the multi query
$res = mysqli_query($con, $query); //Retrieving the records
echo "<table border=1>";
     echo "<tr>";
      echo "<th>Sno</th>";
      echo "<th>Sname</th>";
      echo "<th>email</th>";
      echo "<th>phonenumber</th>";
      if($row=mysqli_fetch_array($res))
      {
         echo "</tr>";
         echo "<tr>";
         echo "<td>$row[0]</td>";
         echo "<td>$row[1]</td>";
         echo "<td>$row[2]</td>";
         echo "<td>$row[3]</td>";
         echo "</tr>";
      }
      echo "</table>";
mysqli_close($con); //Closing the connection
}
?>
```

- **4.Delete**
- <html> <head>
- <title>PHP MySQL Delete example</title> </head>
- <body>
- <form method="post">
- Enter Student No : <input type="text" name="id">
- <input type="submit" value="Delete" name="Submit1">
- </form> </html>

- <?php
- if(isset($_POST['Submit1']))
- {
- $n=$_POST['id']; //html rollno
- $con = mysqli_connect("localhost", "root"," ", "fy"); //Creating a connection
- echo $n;
- $sql = "delete from stud where sno=$n";
- if(mysqli_query($con,$sql))
- {
- echo "record deleted";
- }
- mysqli_close($con); //Closing the connection
- }
- ?>

- **5. Read: Accessing rows in resultset.**
- 1.mysqli_fetch_array()
  - Fetch a result row as a numeric array and as an associative array:
  - **mysqli_fetch_array(*result,resulttype*)**

```
<?php
$con = mysqli_connect("localhost", "root", " ", "mydb"); //Creating a connection
$query = "SELECT * FROM players"; //Executing the multi query
$res = mysqli_query($con, $query, MYSQLI_USE_RESULT); //Retrieving the records
 if ($res)
 {
        while ($row = mysqli_fetch_array($res))
        {
              print("Name: ".$row[0]."\n");
              print("Age: ".$row[1]."\n");
       }
  }
   mysqli_close($con); //Closing the connection
?>
```