PHP
PHP Hypertext Preprocessor

Dr. Shaukat Ali
Department of Computer Science
University of Peshawar

Server-Side Basics

http://*server*/*path*/*file*

- Usually when you type a URL in your browser:
  - Your computer looks up the server's IP address using DNS
  - Your browser connects to that IP address and requests the given file
  - The web server software (e.g. Apache) grabs that file from the server's local file system, and sends back its contents to you

2

Server-Side Basics

- Some URLs actually specify *programs* that the web server should run, and then send their output back to you as the result:

  https://webster.cs.washington.edu/cse190m/quote.php

- The above URL tells the server webster.cs.washington.edu to run the program quote2.php and send back its output

3

Server-Side Programming

- It is used for making 'dynamic webpages'
- It is used for collecting form data from users using forms
- It is used to connect with databases
  - Adding and modifying data
- User verification
- Etc.

4

Server-Side Web Programming

- Server-side pages are programs written using one of many web programming languages/frameworks
  - examples: PHP, Java/JSP, Ruby on Rails, ASP.NET, Python, Perl

- The web server contains software that allows it to run those programs and send back their output
- Each language/framework has its pros and cons
  - we use PHP for server-side programming in this course

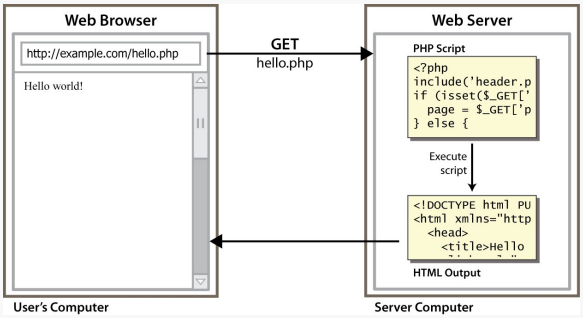
5

What is PHP?

- PHP stands for "PHP Hypertext Preprocessor"
- A Server-Side Scripting Language
- PHP is a widely used server side scripting language
  - PHP is used for dynamic developing web applications.
    - Provide different content depending on context
    - Interface with other services: database, e-mail, etc
    - Authenticate users
    - Process form information
  - Big applications like Wordpress, Facebook was developed in PHP

6

Lifecycle of a PHP Web Request



- Browser requests a `.html` file (**static content**): server just sends that file
- Browser requests a `.php` file (**dynamic content**): server reads it, runs any script code inside it, then sends result across the network

7

Why PHP?

- There are many other options for server-side languages: Ruby on Rails, JSP, ASP.NET, etc.
- Why choose PHP?
  - **Free and Open source:** Anyone can run a PHP-enabled server free of charge
  - **Compatible:** Supported by most popular operating systems and web servers and supports all major Database Management Systems
  - **Simple:** Lots of built-in functionality, familiar syntax, and many libraries
  - **Available:** Installed on UW's servers (Dante, Webster) and most commercial web hosts
  - **Well-Documented:** type php.net/*functionName* in browser Address bar to get docs for any function

8

Set Up for PHP programming

- Installing a webserver
  - Apache
  - Microsoft IIS
- Installing PHP
- Installing DBMS
  - MySQL
- Each webserver and DBMS has its own configuration settings.

http://www.php-intro.com/install.php

WAMP

- One easy way is to use WAMP
  - Linux, Apache, MySQL and PHP as one package
- WAMP alternatives available for Windows
  - WinLAMP
  - LAMP
  - XAMPP. Etc.
    - We will use WAMP.
    - Download and install.

10

WAMP

- In each variation of WAMP, you get one folder to put the PHP projects in.
  - www folder
- After installation of Webserver software, you can test the server by typing the following IP address in a web browser:
  - Localhost
  - 127.0.0.1
- If the webserver is successfully installed and running then you will get see a default page

11

WAMP

- In WAMP, the default directory to put PHP (and HTML) pages is
- **wamp64\www**
  - How to work with it
    - Create a folder inside www e.g., phy_files
    - Place files in it with a .php extension
    - For example
      - If you place **index.php** in a folder named say "**project1"** then it will be accessible by typing **http://localhost/project1/index.php**

12

PHP working

- PHP file has an extension **.php** instead of .html
- PHP code is also be embedded in HTML code
- when a page request arrives, web server recognizes PHP content by the file extension
- The server execute the PHP code with in the page and substitutes the PHP output as HTML code
- The server sends back an HTML code to the client.
- The client never receives PHP code
  - Client only gets HTML and Javascript code

13

PHP Syntax

- PHP code is enclosed in delimiters
  - starts with **<?php**
  - and ends with **?>**

```
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

- Each statement ends with a semicolon';'
- PHP keywords are not case-sensitive
- But PHP variable names are case-sensitive

14

Hello, World Example

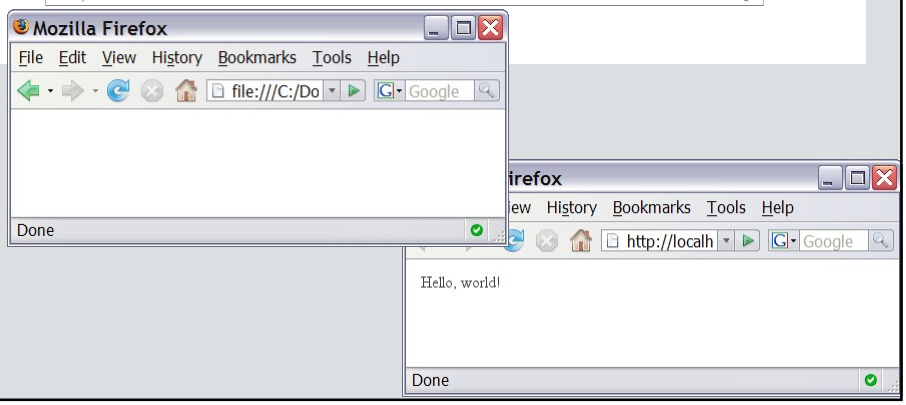
The following contents could go into a file hello.php:

```
<?php
print "Hello, world!";
?>
```

PHP

```
Hello, world!
```

output



PHP Variables

- In PHP, a variable starts with the $ sign, followed by the name of the variable
  - Variable Naming Rules
    - A variable name must start with a letter or the underscore character
    - A variable name cannot start with a number
    - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
    - Variable names are case-sensitive ($age and $AGE are two different variables)
    - Variable names longer than 30 characters are somewhat impractical
- PHP is loosely Typed language
  - That is we do not declare, initialize and specify data type of a_{16} variable

Variable Declaration and Initialization

Examples

```
$my_first_variable
$age = 16;
$user_name = "PinkHeartLuvr78";
$this_class_rocks = TRUE;
```

```
<?PHP
    $strVar = "Hello World";
    $numVar = 10;
    $floatVar = 3.14;
?>
```

- But PHP enables you to use variables at any point just by naming them

17

Strongly Typed and Loosely Typed

- Strongly Typed Languages
  - Requires explicit declaration of variable i.e., data type of variable
    - Generate error it used in incorrect operation
- Loosely Typed Languages
  - Don't require declaration of variable type
  - Automatically convert variable type depending upon the context in which they are used, and the operation performed on their values

18

PHP output

- We use **`echo`** and **`print`** statements to output strings. See example below:

echo “Welcome to PHP Programming”;

```
<html>

<?php
echo "Hello World <BR>";
echo ("Hello Again <BR>");
print "Hello from print<BR>";
Print("Hello from print Again");
?>

</html>
```

19

- Pr*in*ting variables:

```
<html>

<?php
$varValue = 10;
$str = "PHP Programming";

echo "Welcome to $str";
echo "Value of the variable is ". $varValue ;


?>
```

20

Variables scopes

- PHP variables also scopes like Java or Javascript
  - *Global* scope
    - A variable declared outside a function has a GLOBAL scope and can only be accessed any where in the program – in any function
  - *Local* scope
    - A variable declared inside a function has local scope - can be accessed inside the function

```php
<?php
$varX = 50; // has global scope

function myTest() {
    $localX = 5; // has local scope
    echo "<p>Variable x inside function is: $x</p>";
}
?>
```

21

Global Keyword

- We can access Global variable from inside a function or anther local scope
  - To get at outside variables from inside a function
- To access Global variables from inside a function, use the keyword *global*

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}
```

22

PHP Data Types

- String
- Integer
- Float (or double)
- Boolean
  - `$x = true;`
- Array
  - We use the function array to create arrays.

```
$cars = array("Volvo","BMW","Toyota");
$numbers = array( 1, 2, 3, 4, 5);
```

- Array is accessed using index number
- $numbers[0]
- $numbers[1]
- $numbers[3]

23

Converting Data Types in PHP

- PHP includes built-in functions for casting a type
  - The gettype( ) function returns type of variable
  - Also contain functions to check for specific type e.g., is_string(), is_int()

- Example

```
<?php
$my_var = 1995;
echo "The variable is now a " . gettype($my_var);
$my_var = settype($my_var, "string"); //convert to string
Echo "The variable is now a " . gettype($my_var);
?>
```

- Temporary Type Casting

  $ age = **(int)** "21"

24

Comments in PHP

- Single line:

```
//comments
```

- Multiline comment:

```
/*
Comments
comments
*/
```

25

PHP Constants

- Constants are like variables except that once they are defined they cannot be changed or undefined.
- There is **no need to write a dollar sign ($) before a constant** name.
- By convention, constant identifiers are always UPPERCASE
- Constant is created using **`define()`** function.

`define(name,value,case-insensitive)`

- *name:* Specifies the name of the constant
- *value:* Specifies the value of the constant
- *case-insensitive:* Specifies whether the constant name should be case-insensitive. Default is false

26

PHP Constant

- constants are automatically global across the entire program
  - PHP constant examples

```
// Constant LENGTH is defined
define("LENGTH", 50);
echo LENGTH;

// Constant LENGTH is defined as case insensitive
define("LENGTH", 50, true);
echo Length;

// Constant GREETING is defined.
define("GREETING","Welcome to W3Schools" ,true);
echo greeting;
```

27

PHP Operators

- PHP divides the operators in the following groups:
  - Arithmetic operators
    - +, -, /, *, %, ** (power of, 2**3=8)
  - Assignment operators
    - =, +=, -=, …
  - Comparison operators
    - ==, === (equal value and type), !=, <,>, <= …
  - Increment/Decrement operators
    - ++, --,
  - Logical operators
    - and, or, &&, ||, !
  - String operators
    - . (concatenation), .= (concat. Assignment)
  - Array operators

28

PHP Conditional Statements

- if
- if …else
- if … elseif… else

```
if ($marks > 50) {
   echo "You passed!";
}
```

```
if ($marks > 50) {
  echo "You passed!";
else
  echo "You failed";
```

29

switch statement

```
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
  break;
}
```

30

Loops

- Loops are like C
  - **while**
  - **do … while**
  - **for**

```
for ($x = 0; $x <= 10; $x++)
{
    echo "The number is: $x <br>";
}
```

31

- Foreach loop is like in Javascript
  - The **foreach** loop works only on arrays, and is used to loop through each key/value pair in an array.

```
//printing array values
$colors = array("red","green","blue", "yellow");
foreach ($colors as $value)
{
    echo "$value <br>";
}
```

- For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

32

Functions

- Function syntax:

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

- Function Call

    functionName();

- Function names are NOT case-sensitive.

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

writeMsg();
?>
```

33

Function 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.

```
<?php

function addNum( $x1, $x2)
{
$sum = $x1 + $x2;
echo "<BR>The sum is $sum";
}

addNum( 4,5);
addNum( 10,20);

?>
```

34

Function Parameters

The following example will write different first names, but equal last name:

```
<html>
<body>

<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

35

Function Parameters

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>

</body>
</html>
```

This example adds different punctuation.

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?
```

36

Pass by Value

- Copy of actual parameters is passed into the formal parameters
- If the value of a variable within the function is changed, **it doesn't get changed outside** of the function.

```
<?php

 function increment($num)
{

 $num = $num + 1;
 echo "The function output is =".$num."<BR>";
}

$n = 1;
echo "The value before calling is=".$n."<BR>";
increment($n); /* Output 2 */

echo "The value after calling is=".$n; /* Output 1 */

?>
```

```
The value before calling is=1
The function output is =2
The value after calling is=1
```

37

Pass by reference

- Address is passed instead of value
- we prepend an **ampersand (&)** to the argument name in the function definition.
- Any change in variable value within a function can reflect the change in the original value of a variable

```
<?php

 function increment(&$num)
{

 $num = $num + 1;
 echo "The function output is =".$num."<BR>";
}

$n = 1;
echo "The value before calling is=".$n."<BR>";
increment($n); /* Output 2 */

echo "The value after calling is=".$n; /* Output 1 */

?>
```

```
The value before calling is=1
The function output is =2
The value after calling is= 2
```

38

Default Argument Values

- A function may define C++-style default values for scalar arguments
- The default value must be a constant expression, not (for example) a variable, or a function call.
- Using default arguments, defaults should be on right side of any non-default arguments; otherwise, things will not work as expected.

```
<?php
function makecoffee($type = "cappuccino")
{
    return "Making a cup of $type.\n";
}
echo makecoffee() ."<BR>";
echo makecoffee(null)."<BR>";
echo makecoffee("espresso")."<BR>";
?>
```

```
Making a cup of cappuccino.
Making a cup of .
Making a cup of espresso.
```

39

Arrays (more)

- In PHP, the **`array()`** constructor is used to create an array:

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

- Three types of arrays:
  - Indexed arrays:
    - Arrays with a numeric index
    - Array elements addressed by a number
    - **`$colors = array('red','blue','green','yellow');`**
  - Associative arrays:
    - Arrays with named keys
    - Array elements addressed by a name
    - **`$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");`**

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

  - Multidimensional arrays:
    - Arrays containing one or more arrays

40

String Operators and Functions

- Concatenation Operator (.)
  - Concatenation operator (.) can be used between string values to join them together.

```
<?php
$first_name = “Shaukat”;
$last_name = “Ali”;
$whole_Name = $first_name . “ “ . $last_name;
Echo “First name plus last name =
<b>$whole_name</b>”;
?>
```

41

String Operators and Functions

- Strlen( ) Function
  - The strlen( ) function find the length of a string – it counts all characters in the string and returns the total

```
<?php
$first_name = “Shaukat”;
$last_name = “Ali”;
$whole_Name = $first_name . “ “ . $last_name;
$string_length = strlen($whole_name);
Echo “Length of string is = <b> $string_length
</b>”;
?>
```

42

String Operators and Functions

- Strstr ( ) Function
  - The strstr( ) function gets any part of a string that is after the first instance of a particular character or string within a string

```
<?php
$first_name = “Shaukat”;
$last_name = “Ali”;
$whole_name = $first_name . “ “ . $last_name;
$part_after_space = strstr($whole_name, “ “);
Echo “The part of the string after the space is <b>
$part_after_space</b>”;
?>
```

43

String Operators and Functions

- Strpos ( ) Function
  - The strops ( ) function determines a search string exist within a searched string and retunes a numeric value indicating location at the search string begins

```
<?php
$first_name = “Shaukat”;
$last_name = “Ali”;
$whole_name = $first_name . “ “ . $last_name;
$letter_position = strpos($whole_name, “a“);
Echo “The position of the first letter s is <b>
$letter_position</b>”;
?>
```

44

$_GET Variable

- The $_GET variable is an array of variable names and values sent by the HTTP GET method.
- The $_GET variable is used to collect values from a form with method="get". Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and it has limits on the amount of information to send (max. 100 characters).

45

$_GET Variable

```
<form action="welcome.php" method="get">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

When the user clicks the "Submit" button, the URL sent could look something like this:

```
http://www.w3schools.com/welcome.php?name=Peter&age=37
```

The "welcome.php" file can now use the $_GET variable to catch the form data (notice that the names of the form fields will automatically be the ID keys in the $_GET array):

```
Welcome <?php echo $_GET["name"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
```

46

$_POST Variable

- The $_POST variable is an array of variable names and values sent by the HTTP POST method.
- The $_POST variable is used to collect values from a form with method="post". Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

47

$_POST Variable

```
<form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

When the user clicks the "Submit" button, the URL sent could look something like this:

```
http://www.w3schools.com/welcome.php
```

The "welcome.php" file can now use the $_POST variable to catch the form data (notice that the names of the form fields will automatically be the ID keys in the $_POST array):

```
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old!
```

48

Cookies in PhP

- A cookie is a small text file that lets you store a small amount of data (nearly 4KB) on the user's computer.
- They are typically used to keeping track of information such as username that the site can retrieve to personalize the page when user visit the website next time.
- Once a cookie has been set, all page requests that follow return the cookie name and value.
- A cookie can only be read from the domain that it has been issued from. For example, a cookie set using the domain www.xyz.com can not be read from the domain www.abc.com.

49

Setting Cookies in PHP

- PHP provided **setcookie()** function to set a cookie. Set before the <HTML> tag.
- This function requires arguments

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

Parameter	Description	Which type of data
name	Name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.	String
value	Value of the cookie, stored in clients computer.	String
expire	After this time cookie will become inaccessible. The default value is 0. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.	Integer
path	Specify the path on the server for which the cookie will be available. If set to /, the cookie will be available within the entire domain.	String
domain	To which domain the cookie is available. For example, www.example.com	String
secure	If set true, the cookie is available over a secure connection only.	Boolean
httponly	If set true, the cookie is available over HTTP protocol only. Scripting languages like JavaScript won't be able to access the cookie.	Boolean

0

Setting Cookies in PHP

- Example:
  - use setcookie() function to create a cookie named **username** and assign the value **shaukat ali** to it.
  - It also specify that the cookie will expire after **30 days** (30 days * 24 hours * 60 min * 60 sec).

```
<?php
setcookie("username", "John Carter", time()+30*24*60*60);
?>
```

  - **Note:**
  - All the arguments except the name are optional.
  - You may also replace an argument with an empty string ("") in order to skip that argument, however to skip the expire argument use a zero (0) instead, since it is an integer.

51

Example

```
<?php
  setcookie("name", "Paleey Khan", time()+3600, "/","", 0);
  setcookie("age", "5", time()+3600, "/", "",  0);
?>
<html>

  <head>
    <title>Setting Cookies with PHP</title>
  </head>

  <body>
    <?php echo "Set Cookies"?>
  </body>

</html>
```

52

Accessing Cookie Value

- The PHP $_COOKIE super global variable is used to retrieve a cookie value.
- It's a good practice to check whether a cookie is set or not before accessing its value.
  - To do this you can use PHP isset() function

```
<?php

if(isset($_COOKIE["name"])){
        echo $_COOKIE["name"]. "<br />";

        echo $_COOKIE["age"] . "<br />";

} else{
        echo "Welcome Guest!";
}
?>
```

53

Deleting Cookie in PHP

- You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string) - however this time you need the set the expiration date in the past

```
<?php
        setcookie( "name", "", time()- 60, "/","", 0);
        setcookie( "age", "", time()- 60, "/","", 0);
?>
<html> <head>
        <title>Deleting Cookies with PHP</title>
        </head>
        <body>
<?php
        echo "Deleted Cookies" ;
?>
</body> </html>
```

54

PHP Sessions

- Cookies can have security issues
  - Cookies are stored on user's computer it is possible for an attacker to easily modify a cookie content to insert potentially harmful data in your application that might break your application
- Each time a browser requests a URL to the server
  - All the cookie data for a website is automatically sent to the server within the request
  - It means if you have stored 5 cookies on user's system, each having 4KB in size, the browser needs to upload 20KB of data each time the user views a page, which can affect your site's performance

55

PHP Sessions

- A session creates a global file in a temporary directory on the server
  - Registered session variables and their values are stored
  - Data will be available to all pages on the site during that visit
- Sessions have the capacity to store relatively large data compared to cookies
- The location of the temporary file is determined by a setting a variable in the **php.ini** file called **session.save_path**

56

PHP Sessions

- When a session is started the following things happen
  - PHP first creates a unique identifier for that particular session called **Session ID** which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443
    - The **Session ID** is used to retrieve stored values
  - A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique **Session ID** string.
  - A session file is automatically created on the server in the designated temporary directory and bears the name of the **Session ID** prefixed by "sess". For example, sess_3c7foj34c3jj973hjkop2fc937e3443.
-

57

PHP Sessions

- When a PHP script wants to retrieve the value from a session variable
  - PHP automatically gets the **Session ID** string from the **PHPSESSID** cookie
  - Looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values
- A session ends when the user loses the browser or after leaving the site
  - The server will terminate the session after a predetermined period of time of absence of user activity, commonly 30 minutes duration
    - You can adjust this timeout duration by changing the value of session.gc_maxlifetime variable in the PHP configuration file (php.ini).

58

Starting a PHP Session

- A PHP session is easily started by making a call to the **session_start()** function
  - This function first checks if a session is already started and if none is started then it starts one
  - It is recommended to put the call to **session_start()** at the beginning of the page
- Session variables are stored in array called **$_SESSION[]**. These variables can be accessed during lifetime of a session

59

Example

- The following example starts a session then register a variable called **counter** that is incremented each time the page is visited during the session
- Make use of **isset()** function to check if session variable is already set or not

60

Example

```php
<?php
   session_start();

   if( isset( $_SESSION['counter'] ) ) {
      $_SESSION['counter'] += 1;
   }else {
      $_SESSION['counter'] = 1;
   }

   $msg = "You have visited this page ".  $_SESSION['counter'];
   $msg .= "in this session.";
?>

<html>

   <head>
      <title>Setting up a PHP session</title>
   </head>

   <body>
      <?php  echo ( $msg ); ?>
   </body>

</html>
```

Output will be

```
You have visited this page 1in this session.
```

61

Destroying a PHP Session

- A PHP session can be destroyed by **session_destroy()** function
  - This function does not need any argument and a single call can destroy all the session variables

```php
<?php
   session_destroy();
?>
```

- If you want to destroy a single session variable then you can use **unset()** function to unset a session variable.

```php
<?php
   unset($_SESSION['counter']);
?>
```

62

Comparison between Cookies and Sessions

- Both cookies and sessions are used for storing persistent data. But there are differences for sure.
  - Sessions are stored on server side. Cookies are on the client side.
  - Sessions are closed when the user closes his browser. For cookies, you can set time that when it will be expired.
  - Sessions are safe than cookies. Because, since stored on client's computer, there are ways to modify or manipulate cookies.

63

Opening Database Connection

- In order to store or access the data inside a MySQL database, you first need to connect to the MySQL database server
- PHP offers two different ways to connect to MySQL server:
  - **MySQLi extension** (the "i" stands for improved)
  - **PDO (PHP Data Objects)**
- PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.
  - MySQLi extension however provides an easier way to connect to, and execute queries on, a MySQL database server.

64

Connecting to MySQL Database Server

- In PHP you can easily do this using the mysqli_connect() function
- All communication between PHP and the MySQL database server takes place through this connection
  - **Syntax**

    $link = mysqli_connect("hostname", "username", "password", "database");
- Mysqli_connect() returns a boolean value
  - Ture if successfully connected
  - False if not connected

65

Example

```
<?php

/* Attempt MySQL server connection. Assuming you are running MySQL
server with default setting (user 'root' with no password) */

$link = mysqli_connect("localhost", "root", "");

// Check connection
if($link === false)
{
die("ERROR: Could not connect. " . mysqli_connect_error());
}
else
{
// Print host information
echo "Connect Successfully. Host info: " . mysqli_get_host_info($link);
}

?>
```

66

Closing the MySQL Database Server Connection

- The connection to the MySQL database server will be closed automatically as soon as the execution of the script ends
- However, if you want to close it earlier you can do this by simply calling the PHP mysqli_close() function

67

Example

```
<?php

/* Attempt MySQL server connection. Assuming you are running MySQL
server with default setting (user 'root' with no password) */

$link = mysqli_connect("localhost", "root", "");

// Check connection
if($link === false)
{
die("ERROR: Could not connect. " . mysqli_connect_error());
}
else
{
// Print host information
echo "Connect Successfully. Host info: " . mysqli_get_host_info($link);
}

// Close connection
mysqli_close($link);
?>
```

Creating MySQL Database Using PHP

- The CREATE DATABASE query is used to create a new database in MySQL
- Execute the SQL query through passing it to the PHP mysqli_query() function to finally create database

```
// Attempt create database query execution
$sql = "CREATE DATABASE demo";
if(mysqli_query($link, $sql))
{
echo "Database created successfully";
}
Else
{
echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}
```

69

Example

```
<?php
/* Attempt MySQL server connection. Assuming you are running MySQL
server with default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "");

// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}

// Attempt create database query execution
$sql = "CREATE DATABASE demo";
if(mysqli_query($link, $sql)){
    echo "Database created successfully";
} else{
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}

// Close connection
mysqli_close($link);
?>
```

70

Creating Tables inside MySQL Database Using PHP

- The SQL CREATE TABLE statement is used to create a table in database.
- Steps
  - First make a SQL query using the CREATE TABLE statement
  - Execute the SQL query through passing it to the PHP mysqli_query() function to finally create table

71

Example

```
<?php

/* Attempt MySQL server connection. Assuming you are running MySQL server with
    default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection
if($link === false) {
die("ERROR: Could not connect. " . mysqli_connect_error()); }

// Attempt create table query execution
$sql = "CREATE TABLE persons( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(30) NOT NULL, last_name VARCHAR(30) NOT NULL, email
    VARCHAR(70) NOT NULL UNIQUE )";

if(mysqli_query($link, $sql))
{ echo "Table created successfully."; }
Else
{ echo "ERROR: Could not able to execute $sql. " . mysqli_error($link); }

// Close connection
mysqli_close($link);
?>
```

72

Inserting Data into a MySQL Database Table

- The INSERT INTO statement is used to insert new rows in a database table.
- Steps
  - First make a SQL query using the INSERT INTO statement with appropriate values
  - Second execute the insert query through passing it to the mysqli_query() function to insert data in table

73

Example

```
<?php
/* Attempt MySQL server connection. Assuming you are running
   MySQL server with default setting (user 'root' with no password)
   */
$link = mysqli_connect("localhost", "root", "", "demo");
// Check connection
if($link === false)
{ die("ERROR: Could not connect. " . mysqli_connect_error()); }
// Attempt insert query execution
$sql = "INSERT INTO persons (first_name, last_name, email) VALUES
   ('Peter', 'Parker', 'peterparker@mail.com')";
if(mysqli_query($link, $sql))
{ echo "Records inserted successfully."; } else
{ echo "ERROR: Could not able to execute $sql. " .
   mysqli_error($link); }
// Close connection
mysqli_close($link);
?>
```

74

Selecting Data From Database Tables

- The SQL SELECT statement is used to select the records from database tables.
- Its basic syntax is as follows:

SELECT *column1_name*, *column2_name*, *columnN_name* FROM *table_name*;

- Steps:
  - First make a SQL query using the SELECT statement
  - Secondexecute this SQL query through passing it to the mysqli_query() function to retrieve the table data

75

Example

```
// Attempt select query execution
$sql = "SELECT * FROM persons";
if($result = mysqli_query($link, $sql)) {

if(mysqli_num_rows($result) > 0) {

while($row = mysqli_fetch_array($result)) {
Echo $row['id'] . " " . $row['first_name'] . " ".
  $row['last_name'] . " " .  $row['email'] . "<br>"; }

// Free result set
mysqli_free_result($result); }
else { echo "No records matching your query were
  found."; }
else { echo "ERROR: Could not able to execute $sql. " .
  mysqli_error($link); }
```

76

Deleting Database Table Data

- The basic syntax of the DELETE statement can be given with:

```
DELETE FROM table_name WHERE column_name=some_value
```

- Steps:
  - First make a SQL query using the DELETE statement and WHERE clause
  - Second execute this query through passing it to the PHP mysqli_query() function to delete the tables records

77

Example

```
<?php
/* Attempt MySQL server connection. Assuming you are running
   MySQL server with default setting (user 'root' with no password)
   */
$link = mysqli_connect("localhost", "root", "", "demo");
// Check connection
if($link === false)
{ die("ERROR: Could not connect. " . mysqli_connect_error()); }
// Attempt delete query execution
$sql = "DELETE FROM persons WHERE first_name='John'";
if(mysqli_query($link, $sql))
{ echo "Records were deleted successfully."; }
Else
{ echo "ERROR: Could not able to execute $sql. " .
   mysqli_error($link); }
// Close connection
mysqli_close($link);
?>
```

78