CoSc3081
Web Programming

Instructor: Zinabu H.

zinabuscholar@gmail.com
zinabu@aku.edu.et

2024

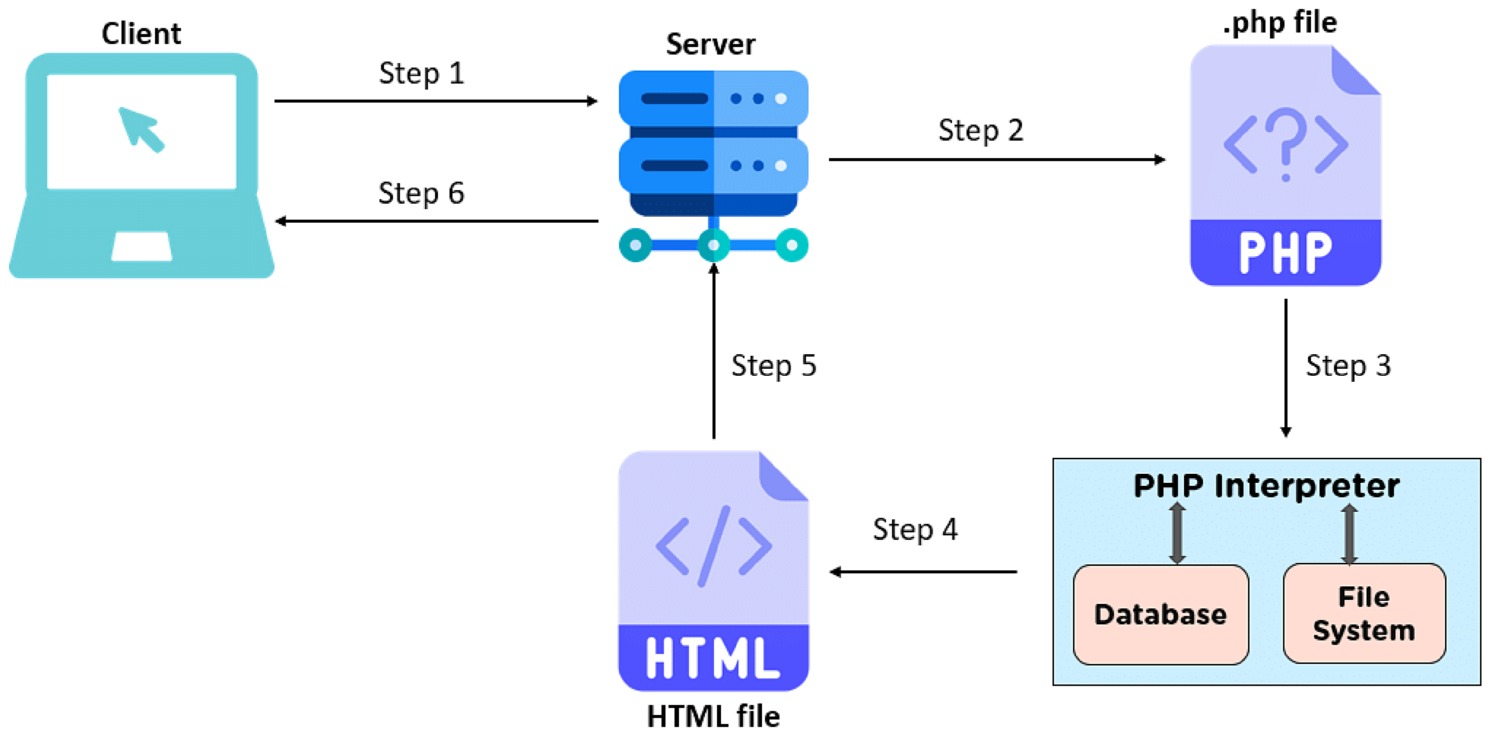
Chapter 5

Server Side Scripting (PHP)

- PHP Basics
- Arrays in PHP
- PHP Functions
- Form Processing in PHP
- Database Programming Using PHP
- PHP OOP

Introduction to PHP

- Server-side and general-purpose scripting language that is especially suited for web development.
- **How it works**



Basic PHP Syntax

- 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**"
- A PHP file normally contains HTML tags, and some PHP scripting code.

Basic PHP Syntax

- A **comment** in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

- There are three ways to add comments to code:
  - `// This is a single-line comment`
  - `# This is also a single-line comment`
  - `/* This is a multi-line comment */`

PHP Variables

- Creating (Declaring) PHP Variables
- In PHP, a variable starts with the $ sign, followed by the name of the variable
- Rules for PHP variables:
  - A variable starts with the $ sign, followed by the name of the variable
  - 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)

PHP 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 Data Types and Variables…

PHP Constants

- Constants are like variables, except that once they are defined they cannot be changed or undefined.
- To create a constant, use the **define()** function or **const** keyword.

```
Syntax

define(name, value, case-insensitive);
const var= value;
```

PHP Data Types and Variables…

Data Type Conversion

- Type casting allows you to convert a value of one type to another.
- Casting in PHP is done with these statements:
  - (string) - Converts to data type String
  - (int) - Converts to data type Integer
  - (float) - Converts to data type Float
  - (bool) - Converts to data type Boolean
  - (array) - Converts to data type Array
  - (object) - Converts to data type Object
  - (unset) - Converts to data type NULL

PHP Operators

What is an Operator?

- In PHP, an operator is a special symbol used to perform operations on operands (**values** and **variables**).
- **PHP Operator Types**
  - **Arithmetic operators**
  - Assignment operators
  - Comparison operators
  - Increment/Decrement operators
  - **Logical operators**
  - String operators
  - Array operators
  - Conditional assignment operators

PHP Operators

PHP Assignment Operators

- The commonly used assignment operator is =.
- You will understand other assignment operators such as +=, -=, *=,/= ,%=,**=,   .=
- ** Exponentiation Assignment

```
a **= 2; // a = a**2
```

PHP Operators : Comparison

- Comparison operators compare two values and return a boolean value, either **true** or **false**

Operator	Name	Description
==	Equal to	Return true if both operands are equal; otherwise, it returns false.
!=, <>	Not equal to	Return true if both operands are equal; otherwise, it returns false.
===	Identical to	Return true if both operands have the same data type and equal; otherwise, it returns false.
!==	Not identical to	Return true if both operands are not equal or not have the same data type; otherwise, it returns false.
>	Greater than	Return true if the operand on the left is greater than the operand on the right; otherwise, it returns false.
>=	Greater than or equal to	Return true if the operand on the left is greater than or equal to the operand on the right; otherwise, it returns false.
<	Less than	Return true if the operand on the left is less than the operand on the right; otherwise, it returns false.
<=	Less than or equal to	Return true if the operand on the left is less than or equal to the operand on the right; otherwise, it returns false.

PHP Operators …

PHP Logical Operators

- Logical operators perform logical operations and return a boolean value, either **true** or **false**

Operator	Description	Example
&&, and	**Logical AND**: true if both the operands are true, else returns false	x && y
||, or	**Logical OR**: true if either of the operands is true; returns false if both are false	x || y
!, not	**Logical NOT**: true if the operand is false and vice-versa.	!x

Conditional Statement

- Conditional statements are used to perform different actions based on different conditions.
- In PHP, there are three forms of the if...else statement.
  - **if** statement
  - **if...else** statement
  - **if...else if...else** statement

The syntax of the if statement is:

```
if (expression ) {
    // the body of if
}
```

Conditional Statement

The syntax of the if .. else statement is:

```
if ( expression ) {
   // block of code if condition is true
} else {
    // block of code if condition is false
 }
```

The syntax of the if...else if...else statement is:

```
if (expression1 ) {
    // code block 1
} else if (expression2){
    // code block 2
} else {
    // code block 3
}
```

Conditional Statement

PHP switch Statement

- If you need to make a choice between more than one alternatives based on a given test condition, the switch statement can be used

The syntax of the switch statement is:

```
switch(expression) {
    case value1:
        // body of case 1
        break;
    case value2:
        // body of case 2
        break;
    case valueN:
        // body of case N
        break;
    default:
        // body of default
}
```

Conditional Statement

PHP loops

- In programming, loops are used to repeat a block of code
- In PHP, there are three forms of loops
  - **for** loop
  - **do while** loop
  - **while** loop

The syntax of the **for** loop is:

```
for (initialExpression; condition; updateExpression) {
    // for loop body
}
```

Conditional Statement

PHP loops

The syntax of the **while** loop is:

```
while (condition) {
  // body of loop
}
```

The syntax of the **do-while** loop is:

```
do{
  // body of loop
} while (condition)
```

Note: do...while loop is similar to the while loop. The only difference is that in do…while loop, the body of loop is executed at **least once**.

Conditional Statement

for Vs while Loop

- A for loop is usually used when the number of **iterations is known**
- And while and do…while loops are usually used when the number of **iterations are unknown**.

Conditional Statement

PHP break Statement

- The break statement is used to terminate the loop immediately when it is encountered

PHP continue Statement

- The continue statement is used to skip the current iteration of the loop and the control flow of the program goes to the next iteration.

Array in PHP

- An array stores multiple values in one single variable:

Create an Array

- Creating an array using **array()** construct or **[ ] JSON** notation, separated by commas.
- Example

```
$scores = array(1, 2, 3); or
$scores = [1, 2, 3];
```

Access Elements of an Array

- Each element of an array is associated with a number called an **index**. The index specifies the position of the element inside the array.

```
Example: echo $scores[0];
```

Array in PHP...

Add Element to an Array

- We can add elements to an array using

syntax

```
$array_name[] = new_element;
```

Change the Elements of an Array

- We can add or change elements by accessing the index value

syntax

```
$array_name[index]=new_element;
```

Array in PHP…

Remove Elements from an Array

- To remove an element from an array, you use the **unset()** function

Example

```
// remove one element at the index
  unset($array_name[index]);
```

Array in PHP…

Size of an Array

- To get the number of elements in an array, you use the **count()** function

Syntax

```
count($array_name);
```

Array in PHP…

PHP Associate Arrays

- arrays that allow you to keep track of elements by names rather than by numbers
- To create an associative array, you use the array() construct or [ ] JSON notation:

Example

```php
<?php
   $html = array(); //or
   $html = [];
   $html['title'] = 'PHP Associative Arrays';
```

Array in PHP…

PHP foreach statement

- PHP provides you with the foreach statement that allows you to iterate over elements of an array, either an **indexed array** or an **associative array**.
- Iterates over all elements in an array, one at a time. It starts with the first element and ends with the last one. Therefore, you don't need to know the number of elements in an array upfront.

Example

```php
<?php
   foreach ($array_name as $element) {
         // process element here
   }
```

Array in PHP…

PHP foreach statement

- **associative array.**

```
<?php
   foreach ($array_name as $key => $value){
       // process element here
   }
```

Array in PHP…

PHP Multidimensional Array

- A multidimensional array is an array that contains another array.

Create a Multidimensional Array

For example

```
$tasks = [
            ['Learn PHP programming', 2],
             ['Practice PHP', 2],
            ['Work', 8],
            ['Do exercise', 1],
        ];
```

Array in PHP…

Access Elements of a Multidimensional Array

- To access an element in an multidimensional array, you use the square brackets ([])

For example

```
$tasks = [
            ['Learn PHP programming', 2],
             ['Practice PHP', 2],
            ['Work', 8],
            ['Do exercise', 1],
        ];

echo $tasks[0][1];
```

Array in PHP…

Add an Element to a Multidimensional Array

- To add an element to a multidimensional array, you use the following syntax:

```
$array[] = [element1, element2, ...];
```

Example

```
$tasks[] = ['Build something matter in PHP', 2];
```

Array in PHP…

Remove an Element from a Multidimensional Array

- To remove an element from a multidimensional array, you can use the **unset()** function

For example

```
unset($tasks[2]);
```

Remove the third element of the $tasks array

Array in PHP…

Iterating over Multidimensional Array

- To iterate a multidimensional array, you use a nested **foreach** loop like this:

For example

```
foreach ($tasks as $task) {
        foreach ($task as $task_detail) {
                echo $task_detail . '<br>';
        }
}
```

Array in PHP…

Array Methods

Some of the methods

Method	**Description**
array_unshift ()	prepend one or more elements to the beginning of an array
array_push()	adds one or more elements to the end of an array
array_pop()	removes an element from the end of an array and returns that element.
array_shift()	remove an element from the beginning of an array.
array_keys()	get the keys of an array
in_array()	check if a value exists in an array.

PHP Functions

- A function is a named block of code that performs a specific task.

Declaring a Function

The syntax to declare a function is:

```
function function_name() {
    // function body
}
```

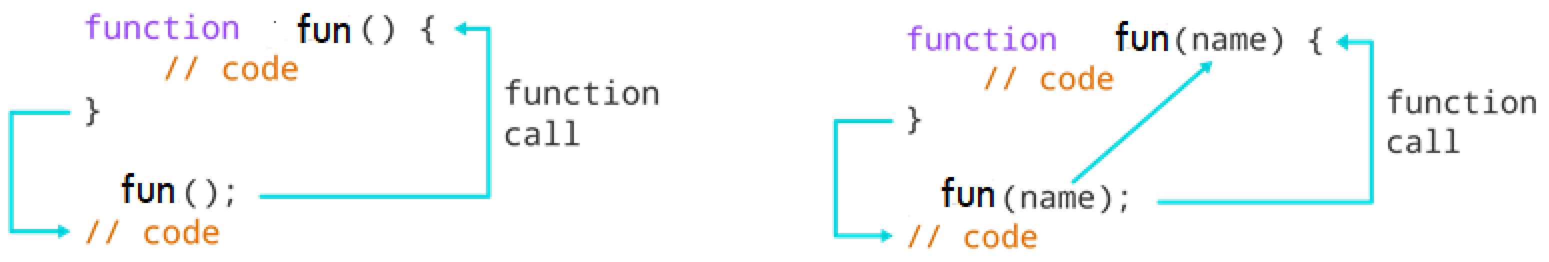
Note:

- A function is declared using the function keyword.
- The name of the function needs to start with a letter or underscore followed by zero or more letters, underscores, and digits.

PHP Functions…

Calling a Function

```
function_name() ;
```



```
function  fun() {
    // code
}

  fun();
// code
```

```
function   fun(name) {
    // code
}

  fun(name);
// code
```

Note:

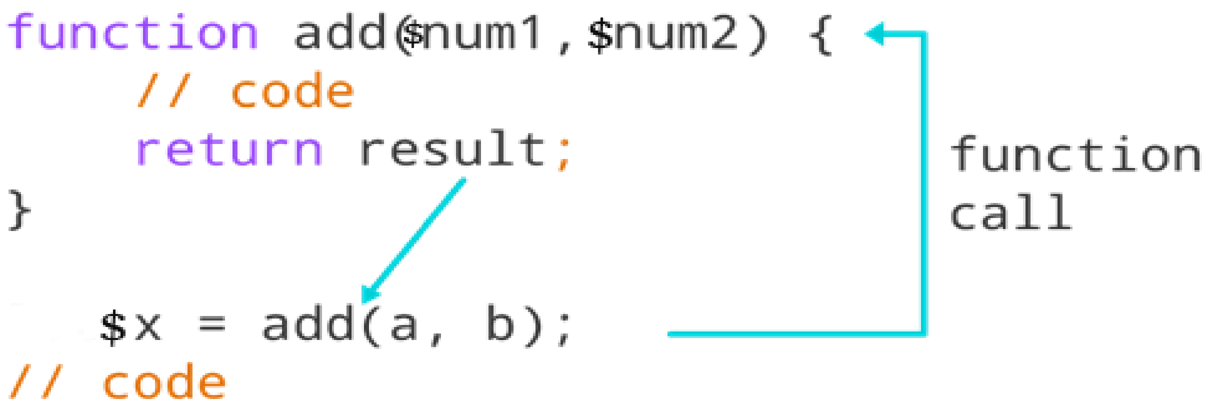
A function can also be declared with parameters.

A parameter is a value that is passed when declaring a function.

PHP Functions…

Function return

- The return statement can be used to return the value to a function call.



```
function add($num1,$num2) {
    // code
    return result;
}

    $x = add(a, b);
// code
```

Benefits of Using a Function

- Function makes the code reusable. You can declare it once and use it multiple times.
- Function makes the program easier as each small task is divided into a function.
- Function increases readability.

PHP form processing

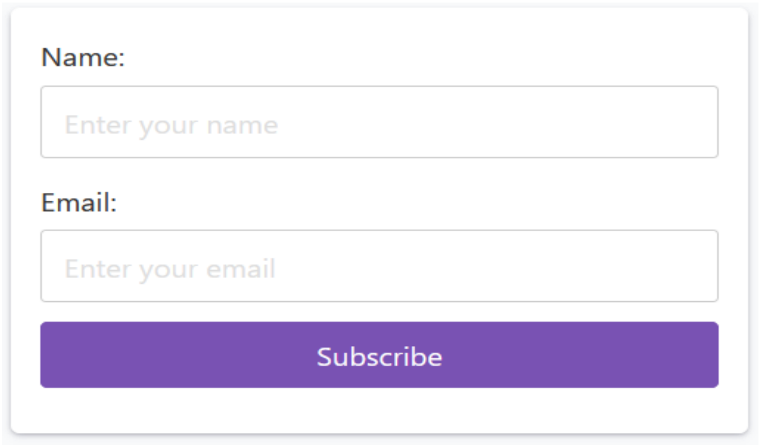
- To create a form, you use the <form> element as follows:

```
<form action=“form.php" method=“post">
 </form>
```

- The <form> element has two important attributes:
  - **action**: specifies the URL that processes the form submission. In this example, the form.php will process the form.
  - **method**: specifies the HTTP method for submitting the form. The most commonly used form methods are POST and GET. In this example, the form method is post.

PHP form processing...

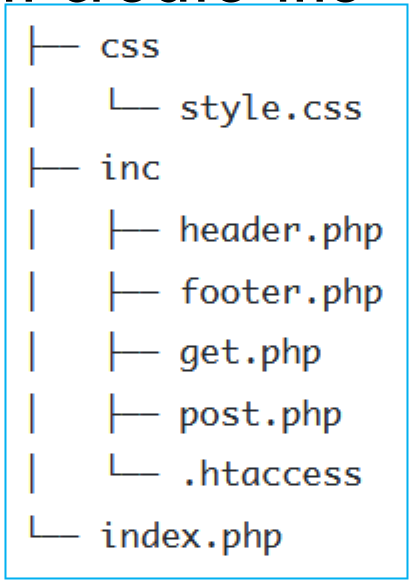
- Example



- Use the <form> tag to create an HTML form.
- Specify the URL that processes the form submission in the action attribute.
- Use either **GET** or **POST** method for the method attribute of the form for submission.
- Use the **$_GET** or **$_POST** to access the form data.
- Use the **htmlspecialchars**() function to escape the user input before showing it on a webpage.

PHP form processing…

- Mixing PHP & HTML is not always a good practice.
- To make the code more organized, you can create the following file & directory structure:



```
├── css
│   └── style.css
├── inc
│   ├── header.php
│   ├── footer.php
│   ├── get.php
│   ├── post.php
│   └── .htaccess
└── index.php
```

- The index.php file in the root directory will include the header.php and footer.php.
- If the request method is GET, the index.php file loads the form in the get.php file. Otherwise, it loads the code from the post.php file for processing the POST request.

PHP File Upload

- The <input> element with the **type="file"** allows you to select one or more files from their storage and upload them to the server via the form submission.

```
<input type="file" id="file" name="file">
```

- To upload multiple files, you add the multiple attribute to the <input> element like this:

```
<input type="file" id="file" name="file" multiple>
```

- If you use multiple file type specifiers, you need to separate them using a comma (,).

```
<input type="file" accept="image/png, image/jpeg" name="file">
```

PHP File Upload...

- The <form> element that contains the file input element must have the **enctype** attribute with the value **multipart/form-data**:

```
 <form enctype="multipart/form-data" action="upload.php"
   method="post">

</form>
```

- If it doesn't, the browser won't be able to upload files.

PHP file upload configuration

- PHP has some important options that control the file upload.
- These options are in the php.ini file
- If you don’t know where to find your php.ini file, you can use the **php_ini_loaded_file()** function as follows:

```
<?php
  echo php_ini_loaded_file();
```

- It’ll return the following file path if you use XAMPP on Windows:

```
C:\xampp\php\php.ini
```

PHP file upload configuration…

- Important settings for file uploads in the php.ini file:

```
; Whether to allow HTTP file uploads.
file_uploads=On

; Temporary directory for HTTP uploaded files
(will use system default if not
; specified).
upload_tmp_dir="C:\xampp\tmp"

; Maximum allowed size for uploaded files.
upload_max_filesize=2M

; Maximum number of files that can be uploaded
via a single request
max_file_uploads=20
```

Handling File uploads in PHP

- To access the information of an uploaded file, you use the **$_FILES** array.
- For example, if the name of the file input element is file, you can access the uploaded file via **$_FILES['file'].**
- The $_FILE['file'] is an associative array that consists of the following keys:
  - **name**: is the name of the uploaded file.
  - **type**: is the MIME type of the upload file e.g., image/jpeg for JPEG image or application/pdf for PDF file.
  - **size**: is the size of the uploaded file in bytes.
  - **tmp_name**: is the temporary file on the server that stored the uploaded filename. If the uploaded file is too large, the tmp_name is "none".
  - **error**: is the error code that describes the upload status e.g., UPLOAD_ERR_OK means the file was uploaded successfully

PHP Cookies and Session

What is a Cookie?

- A cookie is often used to identify a user.
- A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

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

- Only the *name* parameter is required. All other parameters are optional.

PHP Cookies and Session…

What is a Session?

- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.

Start a PHP Session

- A session is started with the **session_start()** function.
- Session variables are set with the PHP global variable: **$_SESSION**.

```
// Set session variables
    $_SESSION[“username"] = “zinabu";
```

```
// echo session variables set on different page
echo “WelCome”.$_SESSION[“username"];
```

PHP Cookies and Session…

Destroy a PHP Session

- To remove all global session variables and destroy the session, use **session_unset()** and **session_destroy()**

- // remove all session variables

  session_unset();

- // destroy the session

  session_destroy();

Database Programming using PHP

PHP MySQL Database

- MySQL is the most popular database system used with PHP.

PHP + MySQL Database System

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

PHP Connect to MySQL

- PHP 5 and later can work with a MySQL database using:
  - **MySQLi extension** (the "i" stands for improved)
  - **PDO (PHP Data Objects)**

Note

PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.

Database Programming 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:

Syntax(mysqli)

```
// Create connection
$conn = new mysqli($servername, $username, $password,$myDB);

// Check connection
  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
   }
echo "Connected successfully";
```

Database Programming using PHP…

Open a Connection to MySQL…

Syntax (PDO)

```
try {
  $conn = new PDO("mysql:host=$servername;dbname=myDB",
$username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully";
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
```

Close the Connection

```
//Mysqli
  $conn->close();
```

```
//PDO
 $conn=null;
```

Database Programming using PHP…

PHP MySQL Insert Data

Syntax (mysqli)

```
$sql = "INSERT INTO tableName(attributes) VALUES (values)";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
```

Database Programming using PHP…

PHP MySQL Insert Data

Syntax (PDO)

```
$sql = "INSERT INTO tableName(attributes) VALUES (values)";

  // use exec() because no results are returned
  $conn->exec($sql);
  echo "New record created successfully";

$conn = null;
```

Database Programming using PHP…

PHP MySQL Fetch Data

Syntax (mysqli)

```
$sql = "SELECT query";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo $row[“attrname1]. " " . $row["attrname2"];
  }
}
```

PHP File Input-Output

File Handling

- File handling is an important part of any web application
- PHP has several functions for creating, reading, uploading, and editing files

open() function

Before reading from or writing to a file, you need to open it

Syntax

```
fopen ( string $filename , string $mode , bool $use_include_path
= false , resource $context = ? ) : resource
```

PHP File Input-Output…

PHP Read File

- To read the contents from a file, you follow these steps:
  - **Open the file** for reading using the fopen() function.
  - **Read the contents** from the file using the fread() function.
  - **Close** the file using the fclose() function.

fread() function

Syntax

```
fread ( resource $stream , int $length ) : string|false
```

PHP File Input-Output…

PHP Write to File

- The **fwrite()** function is used to write to a file.
- The first parameter of fwrite() contains the **name** of the file to write to and the second parameter is the **string** to be written.

Syntax

```
fwrite ( $filename ,  $content )
```

PHP Date and Time

- The PHP **date()** function is used to format a date and/or a time.

Syntax

```
date(format, timestamp)
```

Parameter	**Description**
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time

- **Get a Date**
  - d - Represents the day of the month (01 to 31)
  - m - Represents a month (01 to 12)
  - Y - Represents a year (in four digits)
  - l (lowercase 'L') - Represents the day of the week

PHP Date and Time...

- **Get a Time**
  - H - 24-hour format of an hour (00 to 23)
  - h - 12-hour format of an hour with leading zeros (01 to 12)
  - i - Minutes with leading zeros (00 to 59)
  - s - Seconds with leading zeros (00 to 59)
  - a - Lowercase Ante meridiem and Post meridiem (am or pm)

PHP Math

- PHP has a set of math functions that allows you to perform mathematical tasks on numbers.

function	Description
pi()	returns the value of PI
min() and max()	used to find the lowest or highest value in a list of arguments
abs()	returns the absolute (positive) value of a number
sqrt()	returns the square root of a number
round()	rounds a floating-point number to its nearest integer
rand()	generates a random number

PHP OOP

- PHP OOP allows you to structure a complex application into a simpler and more maintainable structure.
- **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.
- **Define a Class**
  - A class is defined by using the **class** keyword, followed by the **name** of the class and a pair of curly braces (**{}**). All its properties and methods go inside the braces:

Syntax

```
<?php
     class className{
          // code goes here...
     }
?>
```

PHP OOP…

- **Define Objects**
  - Objects of a class are created using the **new** keyword.

  Syntax

```
<?php
      $objName = new ClassName();
?>
```

- **Add properties to a class**
  - PHP has three access modifiers: public, private, and protected.
  - To add properties to class, you place variables inside it.
- **Add methods to a class**

```
<?php
   modifier function methodName(parameter_list) {
         // implementation
   }
```

PHP OOP…

- **Accessing properties and methods in PHP Class**
  - To access a property, you use the object operator (**->**) like this:

  Syntax

```
<?php
     $objectName->propery;

     $objectName->methodName(arguments);
?>
```

- **Chaining methods**

```
     $objectName->methodName1()
                -> methodName2();
```

PHP OOP…

- **PHP Inheritance**
  - Inheritance allows a class to **reuse the code** from another class without duplicating it.
  - To define a class inherits from another class, you use the extends keyword.

Syntax

```
<?php
   class ChildClass extends ParentClass {
       //Other codes
   }
?>
```

- **How to Call the Parent Constructor**

```
modifier function __construct($par1,$par2){
     parent::__construct($par1);
     $this->property= $par2;
}
```

PHP OOP…

Note !

As you have taken an **Object Oriented Programing** Course, the remaining OOP concepts will be covered/discussed in the class and/or Lab sessions.

End of Chapter 5

Aksum University- AIT

2024