Web Application Programming

Week 5: Introduction to Server-Side Programming

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com or jose@kumiuniversity.ac.ug

Agenda

1. Summary of previous lecture
2. Introduction to Server-Side programing
3. Basics of server-side programming languages (e.g., PHP, Python, Node.js)

Summary of previous lecture

1. JavaScript fundamentals: variables(looked declarations of varriables; Automatically, Using var, Using let, Using const and initialization of the same), data types(**Primitive Data Types:** Number, String, Boolean, Undefined, Null, Symbol (ES6) and BigInt (ES11), and **Reference Data Types:** Objects, Date, Arrays, functions etc), operators(Arithmetic, Assignment, Comparison, Logical Operators etc.), conditional statements, loops etc. and

Introduction to Server-Side programing

Introduction to Server-Side programing

Server-side programming refers to the practice of writing and executing code on a web server rather than on a user's device (such as a computer, smartphone, or tablet).

Server-side code is responsible for handling various tasks related to web applications, websites, and online services.

Server-side programming is a fundamental component of web development and is essential for creating dynamic and interactive web experiences.

Introduction to Server-Side programing

Here are some key aspects of server-side programming:

1. **Server-Side Languages:** There are several programming languages commonly used for server-side development. Some popular choices include:
    i. **JavaScript:** Using Node.js, JavaScript can be used for server-side programming as well as client-side scripting.
    ii. **PHP:** A popular scripting language designed for web development.
    iii. **Python:** Known for its simplicity and versatility, Python has many frameworks like Django and Flask for web development.

Introduction to Server-Side programing

Here are some key aspects of server-side programming:

1. **Server-Side Languages:** There are several programming languages commonly used for server-side development. Some popular choices include:
   iv. **Ruby**: Often used with the Ruby on Rails framework for web applications.
   v. **Java**: Widely used for building scalable and enterprise-level web applications.
   vi. **C#:** Used with ASP.NET for developing web applications on the Microsoft platform.
   vii. **Go (Golang):** Known for its performance and simplicity, Go is used for building web servers.

Introduction to Server-Side programing

Here are some key aspects of server-side programming:

2. **Web Servers:** To run server-side code, you need a web server. Common web servers include Apache, Nginx, Microsoft Internet Information Services (IIS), and Node.js, which can also function as a web server when using JavaScript for server-side development.

3. **Request-Response Model:** In server-side programming, the communication between clients and servers follows a request-response model. A client (usually a web browser) sends an HTTP request to the server, which processes the request and sends back an HTTP response.

Introduction to Server-Side programing

Here are some key aspects of server-side programming:

4. **Handling HTTP Requests:** Server-side code is responsible for handling different types of HTTP requests, such as GET, POST, PUT, and DELETE. The code processes these requests, interacts with databases if necessary, and generates responses.

5. **Databases:** Server-side applications often interact with databases to store and retrieve data. Common databases include MySQL, **PostgreSQL**, **MongoDB**, and **SQLite**. Server-side code connects to the database, performs queries, and manipulates data as needed.

Introduction to Server-Side programing

Here are some key aspects of server-side programming:

6. **Authentication and Authorization:** Server-side programming is crucial for implementing user authentication and authorization systems. It ensures that only authorized users can access certain parts of a website or perform specific actions.
7. **Session Management:** Many web applications require managing user sessions to maintain state information. Server-side code is responsible for creating and managing user sessions securely.

Introduction to Server-Side programing

Here are some key aspects of server-side programming:

8. **Security:** Server-side programming is critical for ensuring the security of web applications. It involves measures like input validation, protecting against SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) attacks.
9. **API Development:** Server-side programming is often used to create APIs (Application Programming Interfaces) that allow other applications, including mobile apps, to interact with the server and access its data and functionality.

Introduction to Server-Side programing

Here are some key aspects of server-side programming:

10. **Scalability:** As a web application grows, server-side code needs to be scalable to handle increased traffic. This might involve load balancing, caching, and optimizing database queries.
11. **Testing and Debugging:** Like any other software development, server-side code should be tested and debugged to ensure it functions correctly and reliably.

Introduction to Server-Side programing

Here are some key aspects of server-side programming:

12. **Deployment:** Once server-side code is developed and tested, it needs to be deployed on a web server, making it accessible to users on the internet.

Server-side programming is a broad field with various frameworks, libraries, and tools to streamline development. The choice of language and tools often depends on the specific requirements of the project and the developer's expertise.

Basics of server-side programming with PHP

Server-side programming with PHP-Hypertext Preprocessor is a popular choice for web development.

PHP is a scripting language designed specifically for building dynamic web applications and websites.

Here are the basic steps and concepts involved in server-side programming with PHP:

Basics of server-side programming with PHP

Steps and concepts involved in server-side programming with PHP:

1. **Setting Up Your Environment:**

   Install PHP: You need to install PHP on your web server or local development environment. You can download it from the official PHP website (php.net).

   Or

   Choose a Web Server: Common web servers for PHP development include Apache, Nginx, and LiteSpeed. Configure your web server to work with PHP.

Basics of server-side programming with PHP

Steps and concepts involved in server-side programming with PHP:

2. Creating PHP Files:

PHP code is typically embedded within HTML files or placed in separate **.php** files.

PHP code is enclosed in <?php and ?> tags.

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

Basics of server-side programming with PHP

Steps and concepts involved in server-side programming with PHP:

3. **Handling HTTP Requests:**

   PHP scripts can handle various types of HTTP requests, such as GET and POST, based on the client's interaction with your web application.

4. **Variables and Data Types:**

   PHP supports various data types, including integers, floats, strings, arrays, and objects. You can declare and manipulate variables as needed.

```php
$name = "John";
$age = 30;
```

Basics of server-side programming with PHP

Steps and concepts involved in server-side programming with PHP:

5. Conditional Statements:

You can use conditional statements like if, else, and switch to control the flow of your PHP code.

```php
if ($age >= 18) {
    echo "You are an adult.";
} else {
    echo "You are not an adult.";
}
```

Basics of server-side programming with PHP

Steps and concepts involved in server-side programming with PHP:

7. Loops:

PHP supports various types of loops, including for, while, and foreach, which allow you to iterate over data structures and perform repetitive tasks.

```php
for ($i = 0; $i < 5; $i++) {
    echo "Iteration $i<br>";
}
```

Basics of server-side programming with PHP

Steps and concepts involved in server-side programming with PHP:

8. Functions:

You can define and use functions to encapsulate and reuse blocks of code.

```php
function add($a, $b) {
    return $a + $b;
}

$result = add(5, 3); // $result is now 8
```

Basics of server-side programming with PHP

Steps and concepts involved in server-side programming with PHP:

9. Working with Forms:

PHP is commonly used to process form submissions. You can access form data using the **$_POST** and **$_GET** super global arrays and take action based on user input.

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST["username"];
    $password = $_POST["password"];
    // Process the form data
}
```

Basics of server-side programming with PHP

Steps and concepts involved in server-side programming with PHP:

10. Database Interaction:

PHP can connect to various database management systems like MySQL, PostgreSQL, and SQLite. You can use database-specific extensions or modern database libraries like PDO to interact with databases.

11. Outputting HTML

PHP is often used to generate dynamic HTML content that is sent to the client's browser. You can use echo or print statements to output HTML along with PHP variables and data.

```
echo "<h1>Welcome, $name!</h1>";
```

Basics of server-side programming with PHP

Steps and concepts involved in server-side programming with PHP:

Error Handling:

Implement error handling mechanisms to catch and handle errors gracefully. PHP provides functions like try, catch, and die() for this purpose.

Security Considerations:

Be mindful of security best practices, such as input validation, to protect your PHP applications from common vulnerabilities like SQL injection and cross-site scripting (XSS) attacks.

Basics of server-side programming with PHP

12. Deployment:

To deploy a PHP application, upload your PHP files to your web server, configure any necessary server settings, and ensure your database is accessible.

Server-side programming with PHP is a versatile and widely used approach for building dynamic web applications, content management systems (CMS), e-commerce websites, and more.

Additionally, there are popular PHP frameworks like Laravel, Symfony, and CodeIgniter that provide structured ways to develop web applications, making development faster and more organized. See frameworks later.

Install PHP on your Local Machine

To run PHP scripts on any machine(server, computer etc) we need PHP installed on it.

In this tutorial, we will learn how to install PHP on our computer/laptop and setup the development environment.

Install PHP on your Local Machine+

Requirements for PHP

To run PHP scripts, we need the following services:

1. **PHP Parser**: To execute PHP scripts, PHP installation is required.
2. **Web Server**: Because PHP is mostly used to develop websites, hence most of its implementations comes bundled with Apache Web Server, which is required to host the application developed in PHP over HTTP.
3. **Database**: Any one database management system, which is generally MySQL, as PHP comes with a native support for MySQL

i. Now, you can install all the 3 services separately yourself, or you can simply download and install softwares which automatically installs all the above services.

ii. The most popular of such software package is **XAMPP**.

Install PHP on your Local Machine

What is XAMPP?

XAMPP stands for:

1. **X**: Cross Platform, as it supports all the modern operating systems like Windows, Mac OSX, Linux etc.
2. **A**: Apache Web Server
3. **M**: MySQL database management system.
4. P: PHP installation
5. P: Perl scripting language. (Admin, 2021).

Install PHP on your Local Machine

You can easily download and install XAMPP from this link. The installation process is simple and once installed you will see a small window like this, showing the status of various services which are running.

Install PHP on your Local Machine
Installing a Webserver (Apache)

Installation of Apache server is automatically done during the installation of a webserver system like xampp or wampserver.

To install Apache, you must be having either of the above systems, however we are going to use xampp in this tutorial. If you do not have one please go to google and download xampp.

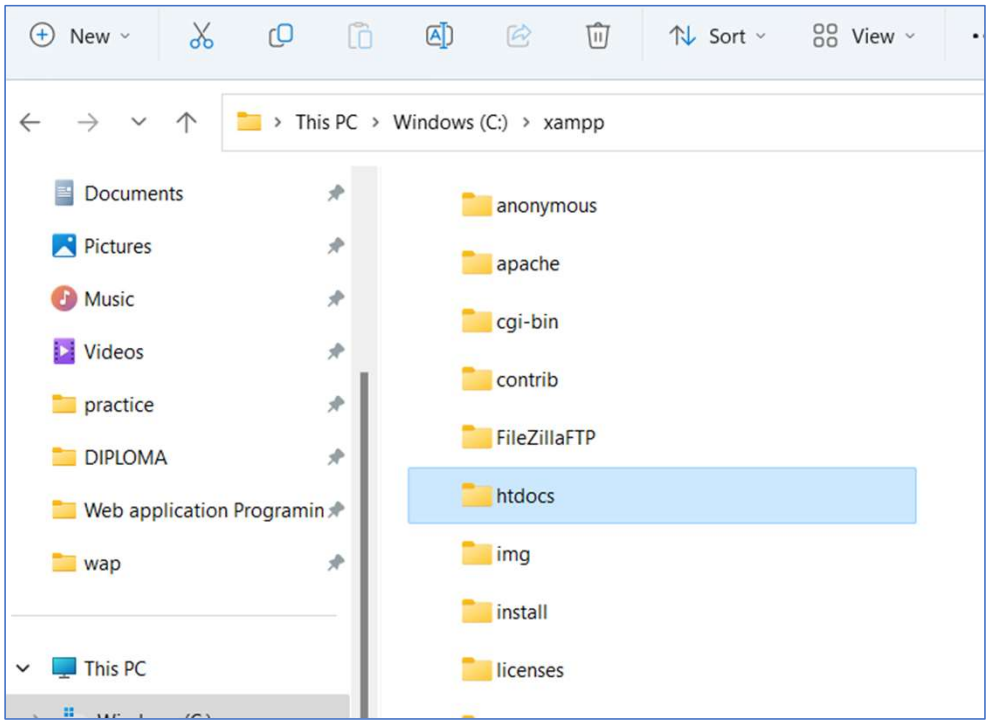
Installation process.

1. Open the folder containing xampp setup then double click on it
2. Click Next, Next until you finish the installation

Install PHP on your Local Machine

Upon successful installation, a folder with name **xampp** will be created in the C drive(by default).

In the folder xampp there are many sub-folders like **apache**, **cgi-bin**, **FileZillaFTP** etc, but the most important sub-folders are:



Install PHP on your Local Machine

htdocs: This is the folder in which we will keep all our PHP files.

mysql: This folder contains all the files for the MySQL database. By default the MySQL databse runs on port number 3306.

php: This folder holds all the installation files for PHP. All the configurations for the current PHP installation is saved in php.ini file which is stored in this folder.

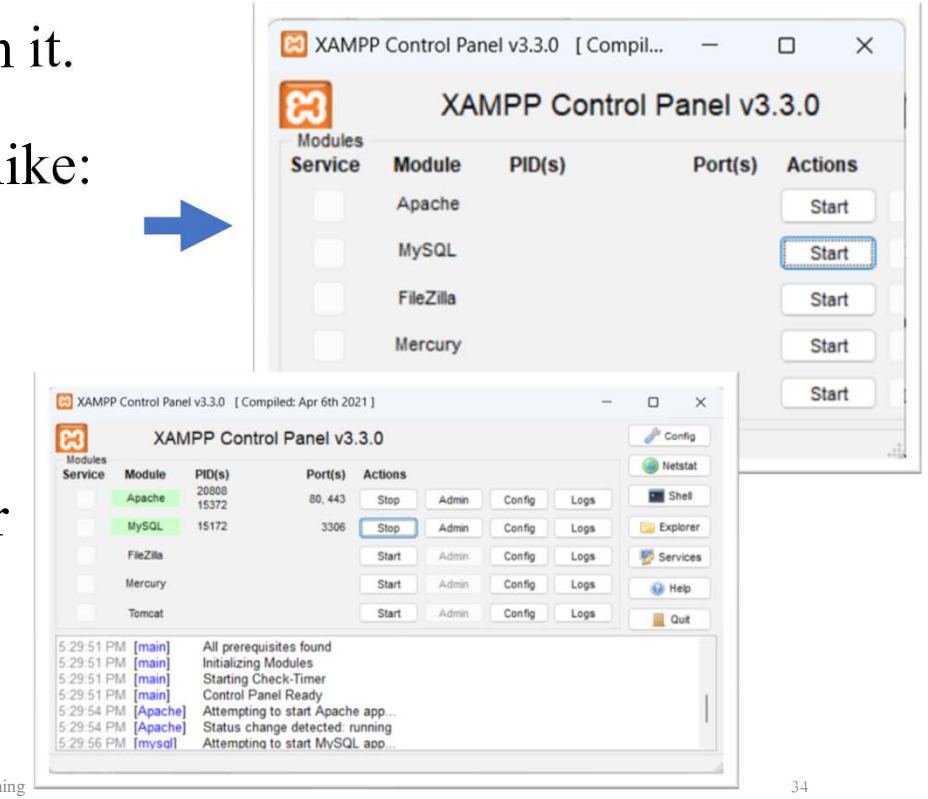
Other Software Packages

XAMPP is not the only available package, although it is the most widely used one. Here are a few other Operating System specific options that you can download and install:

1. WAMP: It's for Windows (Window, Apache, MySQL and PHP)
2. MAMP: It's for Mac OSX (Macintosh, Apache, MySQL and PHP)
3. LAMP: It's for Linux (Linux, Apache, MySQL and PHP

Confirming the installation status

1. After the installation, double click on xampp shortcut icon found on the desktop to run it.
2. When it opens, you will see some thing like:
3. Click Start for Apache and MySQL
4. Now minimize the window
5. Accessing the server on the web browser

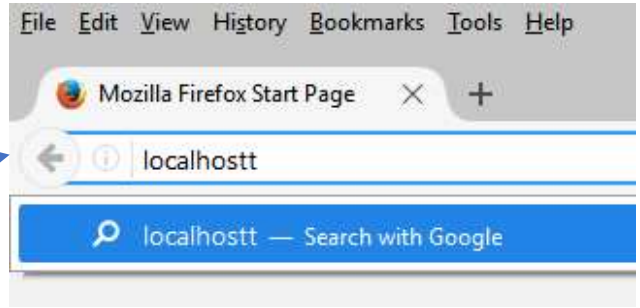


Confirming the installation status+

Accessing the server on the web browser

Now that the Apache and MySQL is running, do the following, to test the server through the web browser

1. Open the web brower (e.g Mozilla Firefox)
2. Type “localhost” in the address bar
3. Press Enter
4. You should see

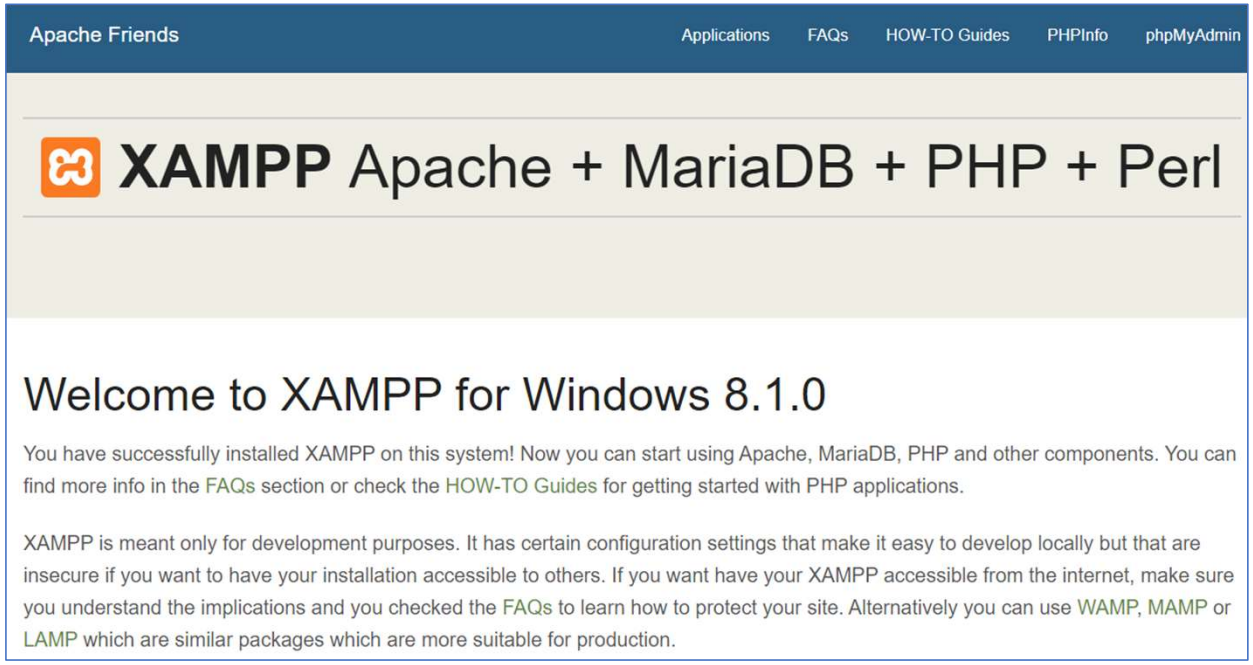


Success registered if you are seeing this picture on you browser

Install PHP on your Local Machine

If everything seems fine and the apache server is running(check in the XAMPP control panel),

Open your **Web browser** and enter **localhost** in the address bar and hit enter. You will see the default page of Apache web server.



IDE or Editor for writing PHP code

As PHP code is not compiled, you can even use a simple editor like Notepad to create PHP code files. But it's always good to have a nice looking Editor which can highlight the code, provide you suggestions while coding, and even analyze your code for syntax errors as your write it.

Before we checkout the list of IDEs and Editors, we must know the difference between both. An **Editor** is a simple text editor with enhanced capabilities like syntax highlighting etc, many modern editors also allows for language specific plugin download to support more features.

IDE or Editor for writing PHP code

Whereas, an **IDE** is an integrated development environment, which is a self-contained package that allow you to write, compile, execute and debug code in the same place.

So if you want a light weight software, go for any Editor, and if you want a heavy weight IDE, you can choose any one from the list below.

So here are a few good IDEs and Editors:

1. Netbeans IDE
2. PyStorm IDE
3. Atom Editor
4. Brackets Editor
5. Notepad ++
6. Sublime Text

Download and install anyone you want. I personally use **Notepad++** Editor, Netbeans CodeBloks

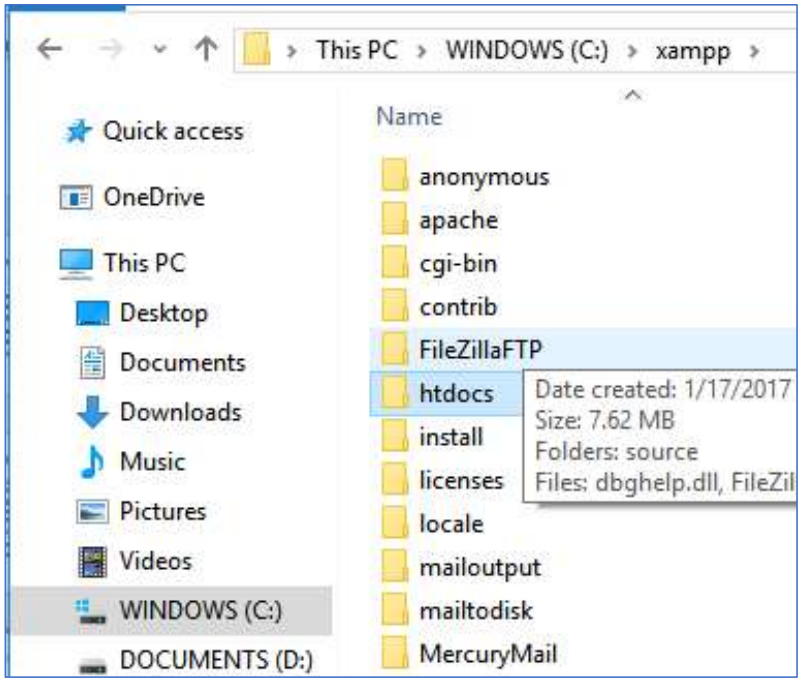
Locating the webserver root directory (htdocs)

Xampp Main folder appears in C:\xampp by default. However, this may be different if you installed xampp in a different Disk.

Just Open Computer then click Local Disk (C:)

How?

1. Click Start
2. Click File Explorer
3. Click this PC, or Computer or My computer depending on the
4. version of windows you are using.
5. Then double click Local Disk (C:)
6. Open xampp
7. Then Open htdocs.



Creating domain folders in the root directory

Having known that the server is running well, and where the htdocs is located C:\xampp\htdocs, you can now open htdocs and create the folder of your choice e.g **wap.**

How?

1. Right click on any empty space on your htdocs
2. Point at New
3. Click Folder
4. Rename it to **wap** then click and empty space to confirm the name and exit.

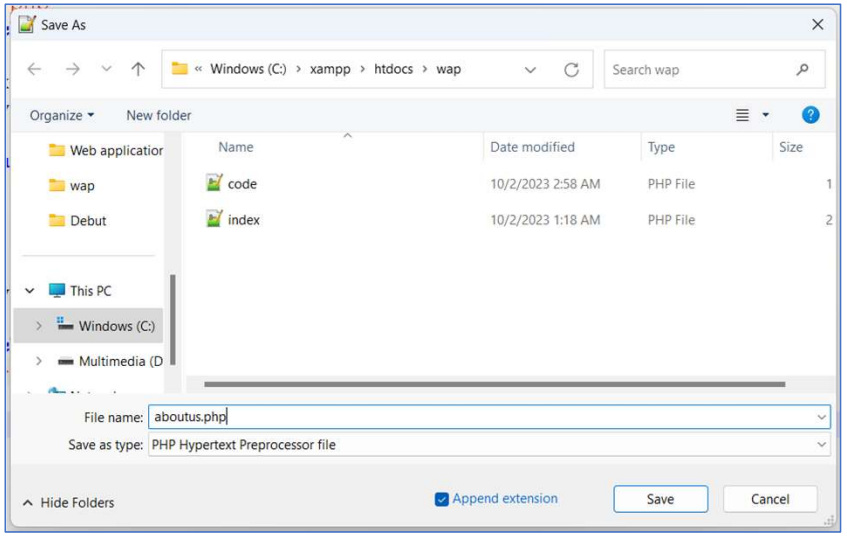
Creating .php Files in the domain folder

Open any Code editor e.g. Notepad++ if it is installed, other wise download it and install

1. Click File
2. Click New

Saving the file with .php extension

1. Click File again and select Save As
2. Navigate to your folder wap which could be found in C:\xampp\htdocs\wap
3. Type the name of the file e.g. index.php
4. Select **All types(*.*)** in the Save as Type dropdown menu
5. Click save.



Creating .php Files in the domain folder

Using the same procedure above, create two more files below in wap domain

1. login.php
2. register.php
3. courses.php

Accessing a page from a web browser

Open the browser

Type **localhost/wap/** in the wed address and press enter

Note! That index.php is automatically accessed by the server.

== **To visit a specific page not index.php,**

type the following **“Localhost/wap/the name of the page”** you want to access, then press enter.

Linking Web pages created in the domain folder

Now enter the following codes into your pages found in wap folder created above.

Save all then

Reload the web Page.

```
<html><head><title>Front Page</title></head><body>

<center><h1>Business Zone</h1></center>
<hr>
<center><table>
<tr>
<td><a href="Home.php">Home</a></td>
<td><a href="Aboutus.php">About us</a></td>
<td><a href="services.php">Our Services</a></td>
<td><a href="product.php">products</a></td>

</tr>
</table></center>
<?php

echo " Do you have a question, feel free to reach us.";

?>

</body></html>
```

Sample Code for index.php

```
<html><head><title>IWelcome</title></head><body>


<center><h1>Business Zone</h1></center>
<hr>
<center><table>
<tr>
<td><a href="Aboutus.php">About us</a></td>
<td><a href="services.php">Our Services</a></td>
```

```
<td><a href="Contactus.php">Contact Us</a></td>
<td><a href="product.php">products</a></td>
</tr>
</table></center>
<?php


echo " This is the first page of our website.";


?>
</body></html>
```

Sample page code for services.php

```
<html><head><title>Front Page</title></head><body>


<center><h1>Business Zone</h1></center>
<hr>
<center><table>
<tr>
<td><a href="Home.php">Home</a></td>
<td><a href="Aboutus.php">About us</a></td>
<td><a href="Contactus.php">Contact Us</a></td>
<td><a href="product.php">products</a></td>
</tr>
</table></center>
<?php

echo " Our best offers";

?>

</body></html>
```

Sample page code for contactus.php

```
<html><head><title>Front Page</title></head><body>


<center><h1>Business Zone</h1></center>
<hr>
<center><table>
<tr>
<td><a href="Home.php">Home</a></td>
<td><a href="Aboutus.php">About us</a></td>
```

```
td><a href="services.php">Our Services</a></td>
<td><a href="product.php">products</a></td>


</tr>
</table></center>
<?php
<
echo " Do you have a question, feel free to reach us.";


?>
</body></html>
```

Viewing Library file /php installation details

Example of files you may need to see their details are: **Db**, **php.ini**, versions of php etc.

We can easily do this by calling **phpinfo()** function into our page e.g. index.php

How?

Open index.php in the code editor then type the following

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

//this will call phpinfo(); for us to view all the needful details about our php installation

Accessing php.ini

What is php.ini for?

Php.ini is the php configuration file that controls how php works. Know how it works and where it is a wholesome deal.

Where is it located?

Php.ini in located in php folder found in xampp main folder "C:\xampp\php"

Basic PHP Syntax

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".

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

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

Example

```
<!DOCTYPE html>
<html>
<body>

<?php

    echo "My first PHP script!";

?>

</body>
</html>
```

Comments in PHP

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

Comments can be used to:

1. Let others understand what we are doing
2. Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

PHP supports several ways of commenting:

PHP supports several ways of commenting:

```php
<?php
	// This is a single-line comment
	# This is also a single-line comment
	/*
		This is a multiple-lines comment block
		that spans over multiple
	lines
	*/
	// You can also use comments to leave out parts of
	//a code line
	$x = 5 /* + 15 */ + 5;

	echo $x;

?>
```

PHP Case Sensitivity

In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are **NOT case-sensitive**.

In the example below, all three echo statements below are legal (and equal):

PHP Case Sensitivity+ Example

```
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
```

However; all variable names are case-sensitive.

PHP Case Sensitivity+ Example

In the example below, only the first statement will display the value of the $color variable (this is because $color, $COLOR, and $coLOR are treated as three different variables):

```php
<?php
$color = "red";     $ColOR="Blue";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

PHP 5 Variables

Variables are "containers" for storing information.

Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

E.g.

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

PHP 5 Variables

After the execution of the statements above, the variable **$txt** will hold the value **Hello world!**, the variable **$x** will hold the value **5**, and the variable **$y** will hold the value **10.5**.

Note:

1. When you assign a text value to a variable, put quotes around the value.

2. Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Think of variables as containers for storing data.

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

1. A variable starts with the $ sign, followed by the name of the variable
2. A variable name must start with a letter or the underscore character
3. A variable name cannot start with a number
4. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
5. Variable names are **case-sensitive** (`$age and $AGE` are two different variables)

Output Variables

The PHP echo statement is often used to output data to the screen.

The following example will show how to output text and a variable:

```
<?php
    $txt = “Kumi University";
    echo "I love $txt!";
?>
```

Output Variables

The following example will output the sum of two variables:

```
<?php
    $x = 5;
    $y = 4;
    echo $x + $y;
?>
```

Note: You will learn more about the echo statement and how to output data to the screen in the next chapter.

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically converts the variable to the correct data type, depending on its value.

In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

1. local
2. global
3. static

Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can be accessed outside from any where:

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

Variables

Same name variables in different functions

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

PHP The global Keyword

```php
<?php
    $x = 5;
    $y = 10;
    function myTest() {
        global $x, $y;
        $y = $x + $y;
    }

    myTest();
    echo $y; // outputs 15
?>
```

PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly. The example above can be rewritten like this:

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

function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the **static** keyword when you first declare the variable:

```
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
myTest();
myTest();
?>
```

PHP The static Keyword

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function

Echo and Print Statements

In PHP there are two basic ways to get output: **echo** and **print**.

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small:

1. **echo** has NO return value while print has a return value of 1 so it can be used in expressions.
2. echo can take **multiple parameters** (although such usage is rare) while **print can take one argument**.
3. echo is marginally faster than print.

The PHP echo Statement

4. The echo statement can be used with or without parentheses: **echo** or **echo()**.

Display Text

The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

```
<?php
    echo "<h2>PHP is Fun!</h2>";
    echo "Hello world!<br>";
    echo "I'm about to learn PHP!<br>";
    echo "This ", "string ", "was ", "made
", "with multiple parameters.";
?>
```

Display Variables

```php
<?php
    $txt1 = "Learn PHP";
    $txt2 = "Kumi University";
    $x = 5;
    $y = 4;

    echo "<h2>" . $txt1 . "</h2>";
    echo "Study PHP at " . $txt2 . "<br>";
    echo $x + $y;
?>
```

PHP Data Types

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

PHP supports the following data types:

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

PHP String

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

```
<?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:

1. An integer must have at least one digit
2. An integer must not have a decimal point
3. An integer can be either positive or negative
4. Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

PHP Integer

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```
<?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:

```
<?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:

```
<?php

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

?>
```

You will learn a lot more about arrays in later chapters of this tutorial

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.

First we must declare a class of object. **For this**, we use the **class keyword.**

A class is a structure that can contain properties and methods:

See. Eg. Below

PHP Object - Example

```php
<?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.

Tip: If a variable is created without a value, it is automatically assigned a value of NULL.

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.

We will not talk about the resource type here, since it is an advanced topic.

PHP String Functions

In this chapter we will look at some commonly used functions to manipulate strings.

Get The Length of a String

The PHP strlen() function returns the length of a string.

The example below returns the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

The output of the code above will be: 12.

Count The Number of Words in a String

The PHP str_word_count() function counts the number of words in a string:

```
<?php

echo str_word_count("Hello world!"); // outputs 2

?>
```

The output of the code above will be: 2.

Reverse a String

The PHP strrev() function reverses a string:

```
<?php

echo strrev("Hello world!"); // outputs !dlrow olleH

?>
```

The output of the code above will be: !dlrow olleH.

Search For a Specific Text Within a String

The PHP **strpos**() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

The example below searches for the text "world" in the string "Hello world!":

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

Tip: The first character position in a string is 0 (not 1).

Replace Text Within a String

The PHP **str_replace()** function replaces some characters with some other characters in a string.

The example below replaces the text "world" with "Dolly":

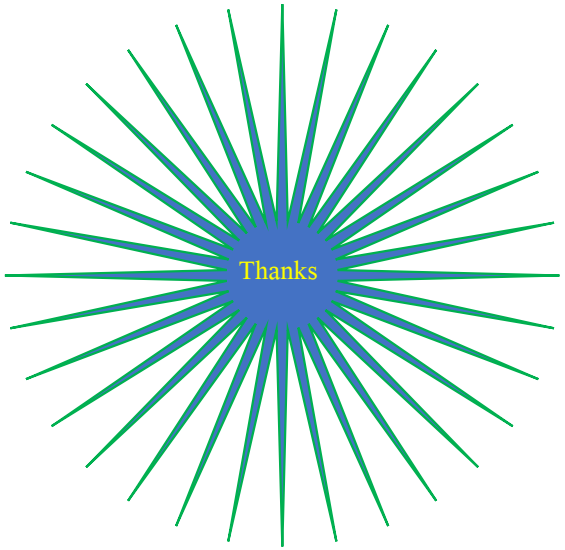
```
<?php
// outputs Hello Dolly!
echo str_replace("world", "Dolly", "Hello world!");
?>
```

Summary

Summary

1. Summary of previous lecture
2. Introduction to Server-Side programing
3. Basics of server-side programming languages (e.g., PHP, Python, Node.js)
4. Installation of PHP through the server, Code editor, learnt some basics of php,

Thank you for Listening



References

Include JavaScript in HTML. Studytonight.com. (n.d.). https://www.studytonight.com/javascript/include-javascript-in-html

ECMA-262. Ecma International. International , E. (2023, June 30). https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

PHP full form - hypertext preprocessor. Admin. (2021, January 20).. BYJUS. https://byjus.com/full-form/php-full-form/#:~:text=The%20full%20form%20of%20PHP%20is%20Hypertext%20Preprocessor.,build%20web%20applications%20or%20websites.