What is PHP

PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development.

PHP was created by **Rasmus Lerdorf in 1994** but appeared in the market in 1995. **PHP 7.4.0** is the latest version of PHP, which was released on **28 November**.

- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language,
- PHP can be embedded into HTML.

Why use PHP

PHP is a server-side scripting language, which is used to design the dynamic web applications with MySQL database.

- It handles dynamic content, database as well as session tracking for the website.
- You can create sessions in PHP.
- It can access cookies variable and also set cookies.

Install PHP

To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:

- **WAMP** for Windows
- **LAMP** for Linux
- **MAMP** for Mac
- **SAMP** for Solaris
- **FAMP** for FreeBSD
- **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, Mercury Mail, etc.

Download and Install WAMP Server

Click me to download WAMP server

Download and Install LAMP Server

Click me to download LAMP server

Download and Install MAMP Server

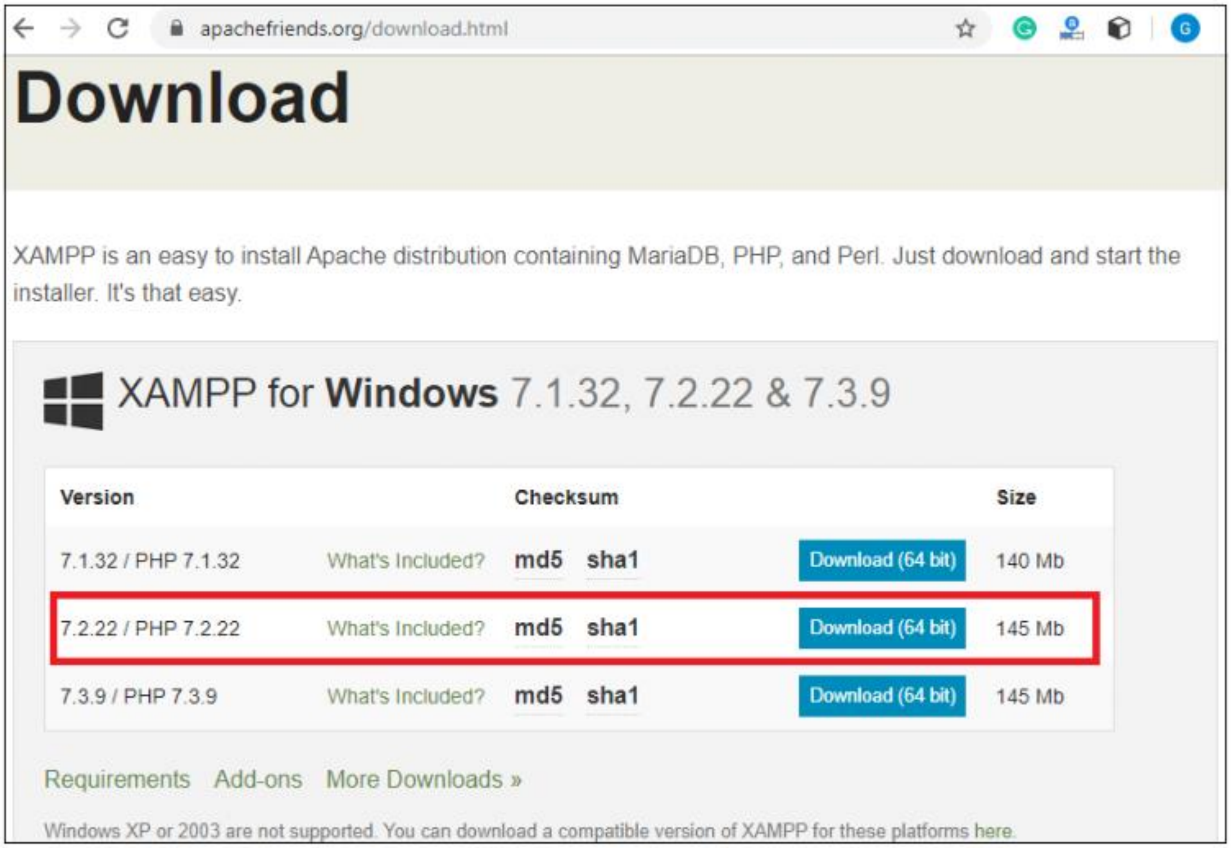
Click me to download MAMP server

Download and Install XAMPP Server

Click me to download XAMPP server

How to install XAMPP server on windows

Step 1: Click on the above link provided to download the **XAMPP server** according to your window requirement.



apachefriends.org/download.html

Download

XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

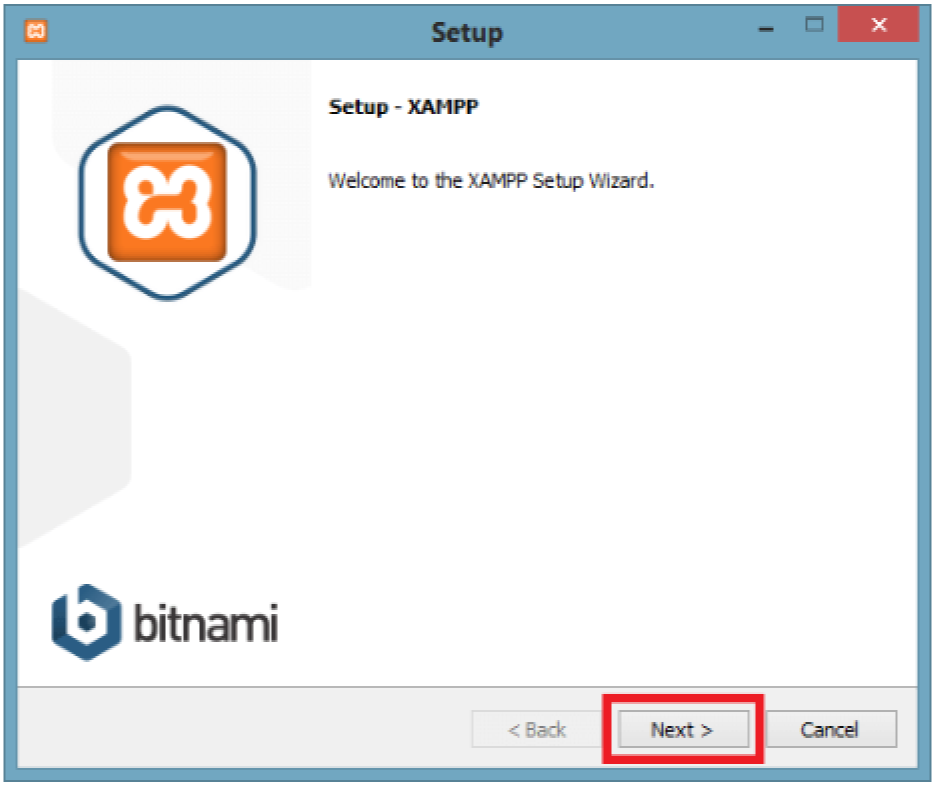
XAMPP for **Windows** 7.1.32, 7.2.22 & 7.3.9

Version		Checksum			Size
7.1.32 / PHP 7.1.32	What's Included?	**md5**	**sha1**	Download (64 bit)	140 Mb
7.2.22 / PHP 7.2.22	What's Included?	**md5**	**sha1**	Download (64 bit)	145 Mb
7.3.9 / PHP 7.3.9	What's Included?	**md5**	**sha1**	Download (64 bit)	145 Mb

Requirements Add-ons More Downloads »

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms here.

Step 2: After downloading XAMPP, double click on the downloaded file and allow XAMPP to make changes in your system. A window will pop-up, where you have to click on the **Next** button.

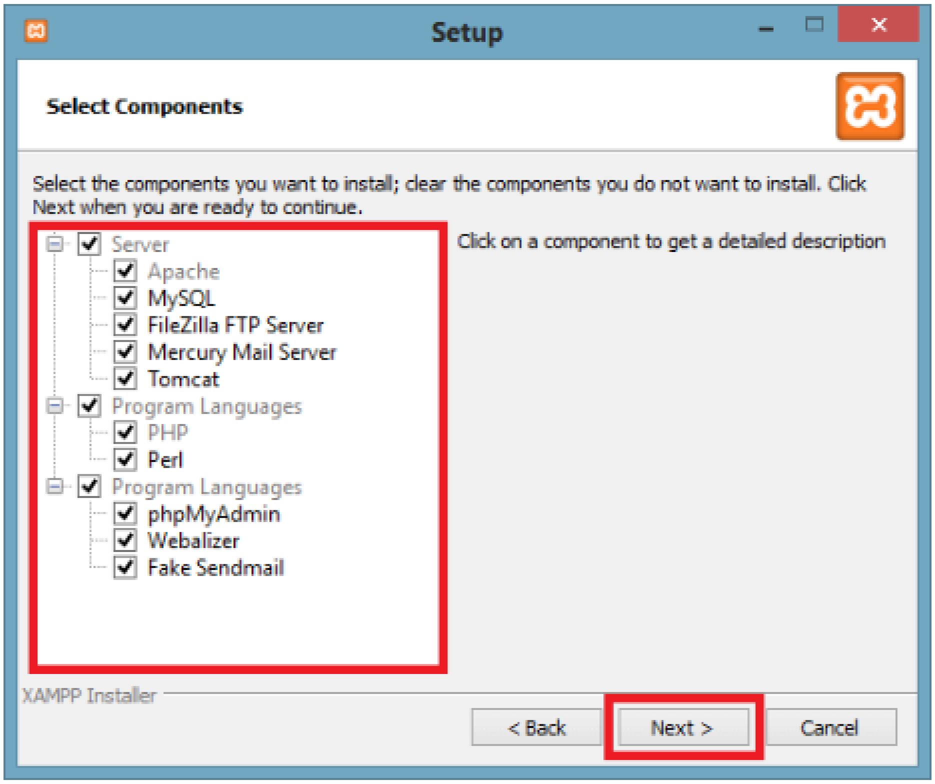


Setup

Setup - XAMPP

Welcome to the XAMPP Setup Wizard.

Step 3: Here, select the components, which you want to install and click **Next**.

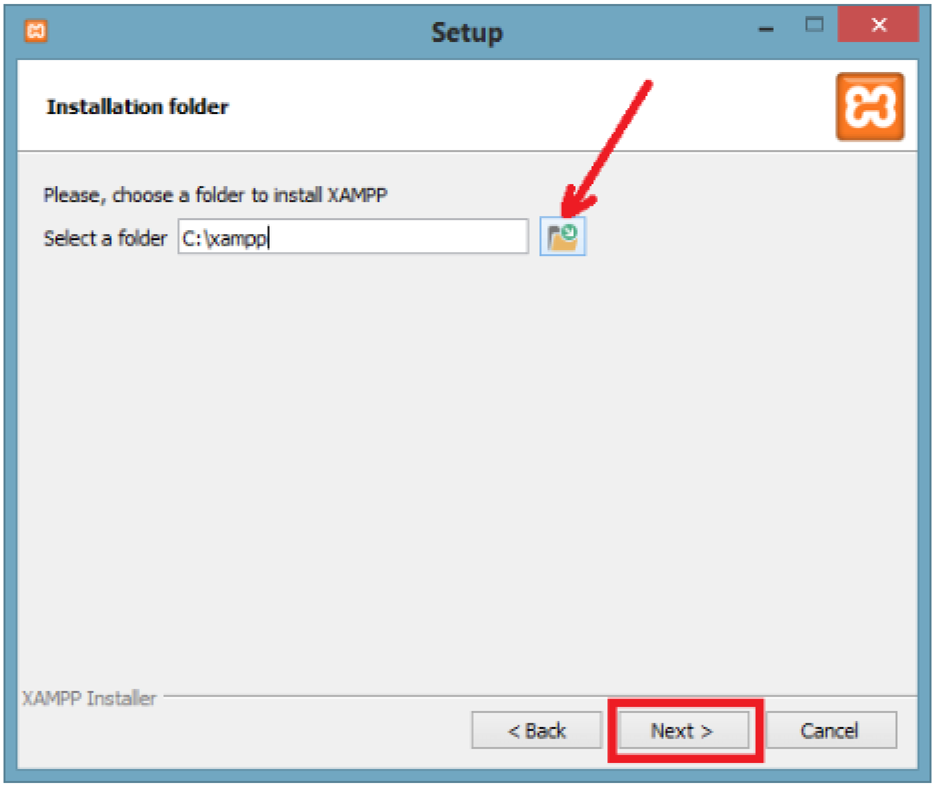


Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☑ Server
  - ☑ Apache
  - ☑ MySQL
  - ☑ FileZilla FTP Server
  - ☑ Mercury Mail Server
  - ☑ Tomcat
- ☑ Program Languages
  - ☑ PHP
  - ☑ Perl
- ☑ Program Languages
  - ☑ phpMyAdmin
  - ☑ Webalizer
  - ☑ Fake Sendmail

Click on a component to get a detailed description

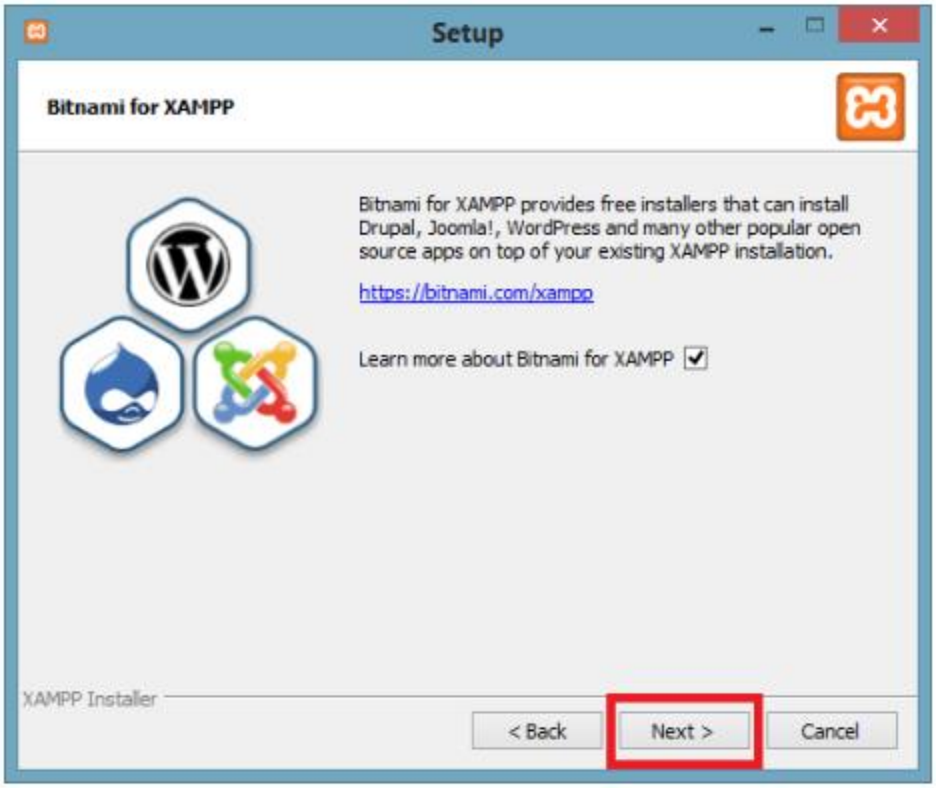
Step 4: Choose a folder where you want to install the XAMPP in your system and click **Next**.



Please, choose a folder to install XAMPP

Select a folder C:\xampp

Step 5: Click **Next** and move ahead.



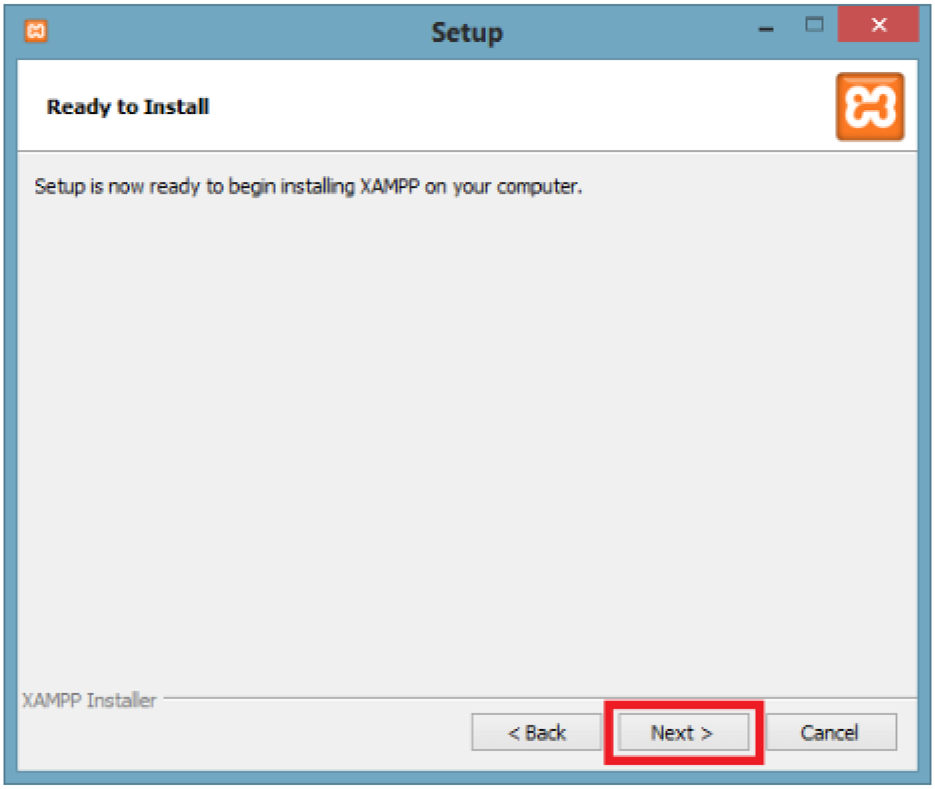
Bitnami for XAMPP

Bitnami for XAMPP provides free installers that can install Drupal, Joomla!, WordPress and many other popular open source apps on top of your existing XAMPP installation.

https://bitnami.com/xampp

Learn more about Bitnami for XAMPP ☑

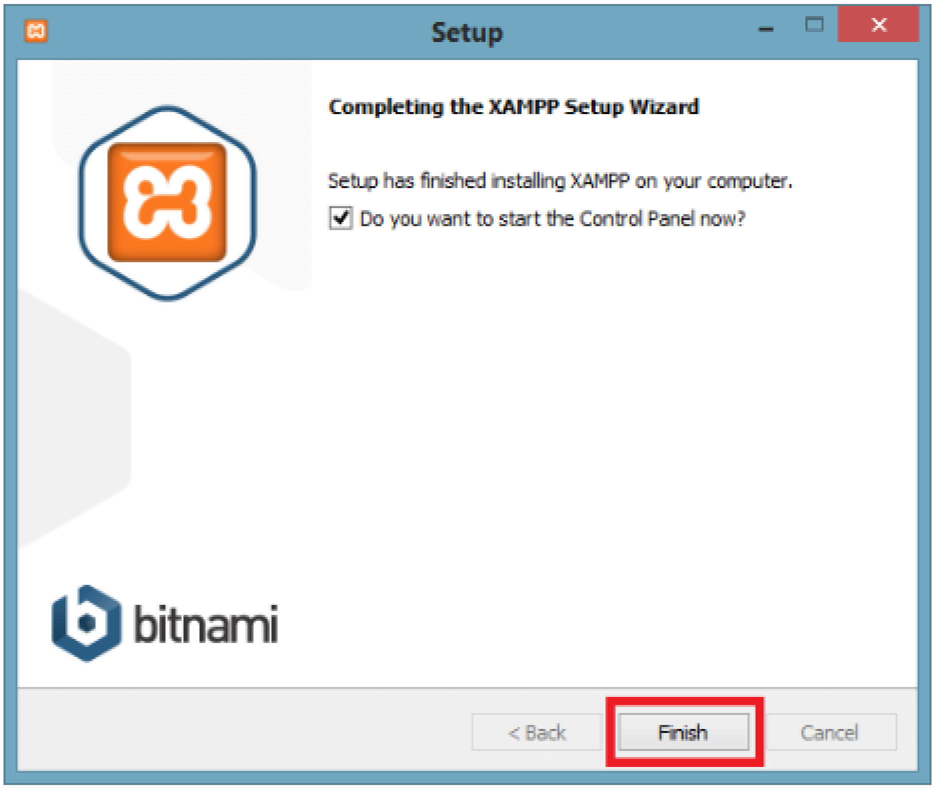
Step 6: XAMPP is ready to install, so click on the **Next** button and install the XAMPP.



Ready to Install

Setup is now ready to begin installing XAMPP on your computer.

Step 7: A finish window will display after successful installation. Click on the **Finish** button.



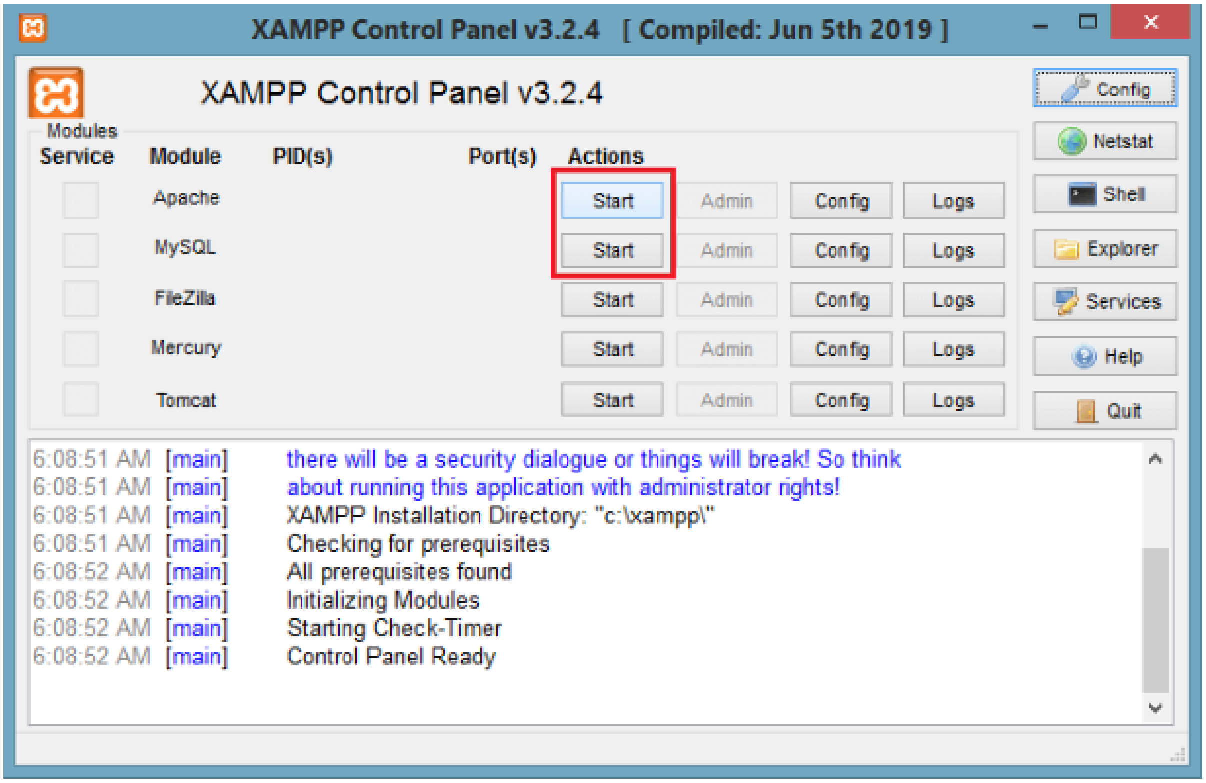
Completing the XAMPP Setup Wizard

Setup has finished installing XAMPP on your computer.

☑ Do you want to start the Control Panel now?

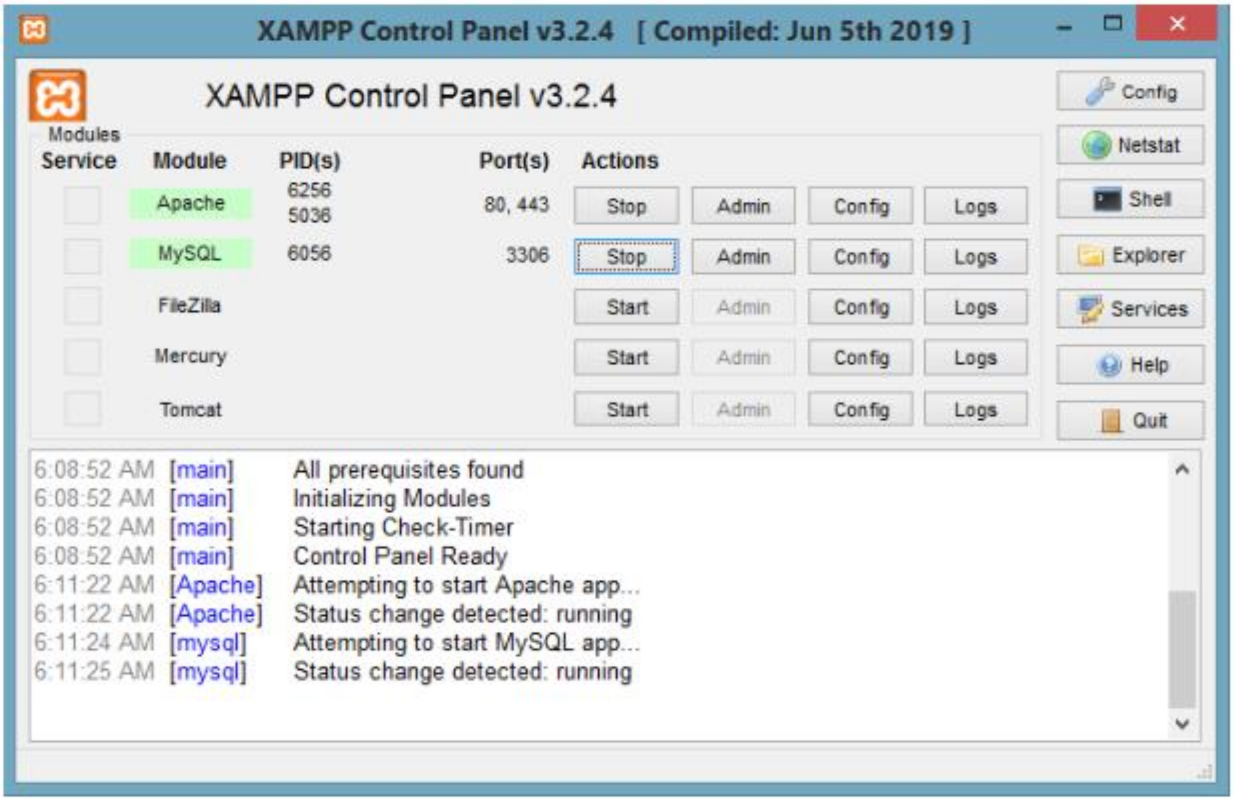
Step 10: XAMPP is ready to use. Start the Apache server and MySQL and run the php program on the localhost.

How to run PHP programs on XAMPP, see in the next tutorial.



```
6:08:51 AM  [main]    there will be a security dialogue or things will break! So think
6:08:51 AM  [main]    about running this application with administrator rights!
6:08:51 AM  [main]    XAMPP Installation Directory: "c:\xampp\"
6:08:51 AM  [main]    Checking for prerequisites
6:08:52 AM  [main]    All prerequisites found
6:08:52 AM  [main]    Initializing Modules
6:08:52 AM  [main]    Starting Check-Timer
6:08:52 AM  [main]    Control Panel Ready
```

Step 10: If no error is shown, then XAMPP is running successfully.



XAMPP Control Panel v3.2.4 [Compiled: Jun 5th 2019]

XAMPP Control Panel v3.2.4

Modules

Service	Module	PID(s)	Port(s)	Actions			
□	Apache	6256 5036	80, 443	Stop	Admin	Config	Logs
□	MySQL	6056	3306	Stop	Admin	Config	Logs
□	FileZilla			Start	Admin	Config	Logs
□	Mercury			Start	Admin	Config	Logs
□	Tomcat			Start	Admin	Config	Logs

```
6:08:52 AM  [main]    All prerequisites found
6:08:52 AM  [main]    Initializing Modules
6:08:52 AM  [main]    Starting Check-Timer
6:08:52 AM  [main]    Control Panel Ready
6:11:22 AM  [Apache]  Attempting to start Apache app...
6:11:22 AM  [Apache]  Status change detected: running
6:11:24 AM  [mysql]   Attempting to start MySQL app...
6:11:25 AM  [mysql]   Status change detected: running
```

PHP Variables

In PHP, a variable is declared using a **$ sign** followed by the variable name. Here, some important points to know about variables:

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP is given below:

```
$variablename=value;
```

Rules for declaring PHP variable:

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

PHP Variable: Declaring string, integer, and float

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

PHP Variable: Sum of two variables

```
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

PHP Variable: case sensitive

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

PHP Variable: Rules

PHP variables must start with letter or underscore only.

PHP variable can't be start with numbers and special symbols.

```
<?php
$a="hello";//letter (valid)
$_b="hello";//underscore (valid)

echo "$a <br/> $_b";
?>
```

Example Number

```
<?php
$4c="hello";//number (invalid)
$*d="hello";//special symbol (invalid)
echo "$4c <br/> $*d";
?>
```

Types of Variable

PHP has three types of variable scopes:

1. Local variable
2. Global variable
3. Static variable

Local variable

The variables that are declared within a function are called local variables for that function.

```
<?php
   function local_var()
```

```
  {
     $num = 45;  //local variable
     echo "Local variable declared inside the function is: ". $num;
  }
  local_var();
?>
```

Example 2 :

```
<?php
  function mytest()
  {
     $lang = "PHP";
     echo "Web development language: " .$lang;
  }
  mytest();
  //using $lang (local variable) outside the function will generate an error
  echo $lang;
?>
```

Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program.

```
<?php
  $name = "Sanaya Sharma";       //Global Variable
  function global_var()
  {
     global $name;
     echo "Variable inside the function: ". $name;
     echo "</br>";
  }
  global_var();
  echo "Variable outside the function: ". $name;
?>
```

Example 2

```
<?php
    $name = "Sanaya Sharma";        //global variable
    function global_var()
    {
        echo "Variable inside the function: ". $name;
        echo "</br>";
    }
    global_var();
?>
```

Using $GLOBALS instead of global

Another way to use the global variable inside the function is predefined $GLOBALS array.

```
<?php
    $num1 = 5;      //global variable
    $num2 = 13;     //global variable
    function global_var()
    {
            $sum = $GLOBALS['num1'] + $GLOBALS['num2'];
            echo "Sum of global variables is: " .$sum;
    }
    global_var();
?>
```

Static variable

It is a feature of PHP to delete the variable, once it completes its execution and memory is freed. Sometimes we need to store a variable even after completion of function execution. Therefore, another important feature of variable scoping is static variable. We use the static keyword before the variable to define a variable, and this variable is called as **static variable**.

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

//first function call
    static_var();

    //second function call
    static_var();
?>
```

PHP $ and $$ Variables

The **$var** (single dollar) is a normal variable with the name var that stores any value like string, integer, float, etc.

The **$$var** (double dollar) is a reference variable that stores the value of the $variable inside it.

To understand the difference better, let's see some examples.

```
<?php
$x = "abc";
$$x = 200;
echo $x."<br/>";
echo $$x."<br/>";
echo $abc;
?>
```

Example2

```
<?php
 $x="Sindh";
```

```
$$x="karachi";
echo $x. "<br>";
echo $$x. "<br>";
echo "Capital of $x is " . $$x;
?>
```

Example3

```
<?php
$name="Cat";
${$name}="Dog";
${${$name}}="Monkey";
echo $name. "<br>";
echo ${$name}. "<br>";
echo $Cat. "<br>";
echo ${${$name}}. "<br>";
echo $Dog. "<br>";
?>
```

PHP Data Types

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types (predefined)
2. Compound Types (user-defined)
3. Special Types

PHP Data Types: Scalar Types

It holds only single value. There are 4 scalar data types in PHP.

1. boolean
2. integer
3. float
4. string

PHP Data Types: Compound Types

It can hold multiple values. There are 2 compound data types in PHP.

1. array
2. object

PHP Data Types: Special Types

There are 2 special data types in PHP.

1. resource
2. NULL

PHP Boolean

Example:

```
<?php
   if (TRUE)
      echo "This condition is TRUE.";
   if (FALSE)
      echo "This condition is FALSE.";
?>
```

PHP Integer

```
<?php
   $dec1 = 34;
   $oct1 = 0243;
   $hexa1 = 0x45;
   echo "Decimal number: " .$dec1. "</br>";
   echo "Octal number: " .$oct1. "</br>";
   echo "HexaDecimal number: " .$hexa1. "</br>";
?>
```

PHP Float

```
<?php
   $n1 = 19.34;
```

```
    $n2 = 54.472;
    $sum = $n1 + $n2;
    echo "Addition of floating numbers: " .$sum;
?>
```

PHP String

A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.

String values must be enclosed either within **single quotes** or in **double quotes**. But both are treated differently. To clarify this, see the example below:

```
<?php
    $company = "ICreativez";
    //both single and double quote statements will treat different
    echo "Hello $company";
    echo "</br>";
    echo 'Hello $company';
?>
```

PHP Array

```
<?php
    $bikes = array ("Royal Enfield", "Yamaha", "KTM");
    var_dump($bikes);   //the var_dump() function returns the datatype and values
    echo "</br>";
    echo "Array Element1: $bikes[0] </br>";
    echo "Array Element2: $bikes[1] </br>";
    echo "Array Element3: $bikes[2] </br>";
?>
```

PHP object

Objects are the instances of user-defined classes that can store both values and functions.

```
<?php
     class bike {
         function model() {
             $model_name = "Royal Enfield";
             echo "Bike Model: " .$model_name;
           }
     }
     $obj = new bike();
     $obj -> model();
?>
```

PHP Resource

Resources are not the exact data type in PHP. Basically, these are used to store some function calls or references to external PHP resources.

For example - a database call. It is an external resource.

PHP Null

Null is a special data type that has only one value: **NULL**. There is a convention of writing it in capital letters as it is case sensitive.

The special type of data type NULL defined a variable with no value.

```
<?php
   $nl = NULL;
   echo $nl;   //it will not give any output
?>
```