PHP: Hypertext Preprocessor

History

PHP 1.0 – Personal Home Page Tools (1994)

- Creator: Rasmus Lerdorf, a Danish–Canadian programmer.
- Background: Rasmus wanted to track visits to his online résumé. He wrote a set of small Perl scripts, later rewritten in C for performance.
- Original Name: *Personal Home Page Tools*.
- Features:
- Basic functionality for tracking visitors.
- Very limited — not a full programming language yet.

PHP 2.0 – PHP/FI (Forms Interpreter) (1995)

- Release: June 1995.
- Name: PHP/FI stood for *Personal Home Page / Forms Interpreter*.
- Key Features:
- Added form handling.
- Database integration with popular systems like mSQL.
- Early scripting capabilities.
- Impact: This version marked the beginning of PHP as a general-purpose tool for web development.

PHP 3.0 – The Turning Point (1997–1998)

- Developers: Zeev Suraski and Andi Gutmans (students at the Technion – Israel Institute of Technology).
- Reason for Rewrite: They found PHP/FI 2 too limited for their project, so they completely rewrote the parser.
- Release: June 1998.
- Key Features:
- First version simply called "PHP".
- Modular architecture → support for multiple databases, protocols, and APIs.
- Large open-source community involvement.
- Impact: This was the first widely used version of PHP.

PHP 4.0 – Zend Engine Era (2000)

- Core: Powered by the Zend Engine 1.0, created by Suraski & Gutmans.
- Release: May 2000.
- Key Features:

- Improved performance and scalability.
- Session handling introduced.
- Better integration with web servers.
- Impact: PHP 4.0 solidified PHP as a serious language for dynamic websites.

PHP 5.0 – Object-Oriented PHP (2004)

- Release: July 2004.
- Core: Zend Engine 2.0.
- Key Features:
- Full Object-Oriented Programming (OOP) support.
- Improved MySQL extension (MySQLi).
- Support for XML and SOAP web services.
- Exception handling improvements.
- Impact: PHP moved closer to being a modern programming language, not just a scripting tool.

PHP 7.0 – Performance Revolution (2015)

- Release: December 2015.
- Core: Zend Engine 3.0.
- Key Features:
- 2x faster execution than PHP 5.
- Reduced memory usage.
- Scalar type declarations (int, string, float, bool).
- Return type declarations.
- Improved error handling with Throwable interface.
- Impact: Made PHP competitive with modern web languages, especially for high-performance web applications.

PHP 8.0 – Modern PHP (2020)

- Release: November 2020.
- Core: Zend Engine with JIT (Just-In-Time Compiler).
- Key Features:
- JIT → significant performance boost for CPU-intensive tasks.
- Union types, attributes/annotations, and constructor property promotion.
- Improved type safety (strong typing).
- Enhanced error handling.
- Impact: PHP 8 positioned the language for modern development needs while keeping backward compatibility.

Zend Engine

The Zend Engine is the core scripting engine of PHP. It was created in 1999 by Zeev Suraski and Andi Gutmans. Think of it as the brain of PHP — it takes your PHP code and makes it run on the server.

Main Functions of the Zend Engine

- Parsing PHP Code
  - Reads your PHP script.
  - Checks for syntax errors.
- Compiling into Opcodes
  - Converts human-readable PHP code into opcodes (low-level instructions).
  - These opcodes are like "machine instructions" for PHP.
- Executing Code
  - Runs the opcodes line by line.
  - Interacts with memory, variables, and external resources (like databases).
- Memory Management
  - Allocates and frees memory automatically.
  - Includes garbage collection to clean unused variables.
- Error Handling
  - Detects runtime errors and exceptions.

Impact of the Zend Engine

- PHP 4 (2000) used Zend Engine 1.0 → big performance boost.
- PHP 5 (2004) used Zend Engine 2.0 → added Object-Oriented Programming (OOP).
- PHP 7 (2015) with Zend Engine 3.0 → 2x faster execution and lower memory use.
- PHP 8 (2020) → integrated JIT (Just-In-Time compiler) inside Zend Engine.

Summary

- PHP Code = Recipe written in English.
- Zend Engine = Translator + Chef who understands the recipe and actually cooks the food.
- Browser = Guest who only sees the final dish (HTML output), not the recipe or cooking process.

What is PHP?

- **PHP** stands for **PHP: Hypertext Preprocessor** (a recursive acronym).
- It is an **open-source, server-side scripting language** designed mainly for **web development**.

How it works:

- PHP code runs on the **server**, not in the browser.
- The server processes the PHP script → generates **HTML output** → sends it to the browser.
- The user only sees the final web page, not the PHP code.

Key Characteristics:

- Easy to learn and use.
- Can be **embedded directly inside HTML**.
- Works with almost all databases (**MySQL, PostgreSQL, MongoDB, etc.**).
- Supported on all major operating systems (**Windows, Linux, macOS**).
- Huge ecosystem of frameworks (**Laravel, CodeIgniter, Symfony**) and CMS (**WordPress, Drupal**).

In sort PHP is the **language of dynamic websites** — it powers millions of applications like WordPress, Facebook's early versions, and Wikipedia.

How PHP Works:

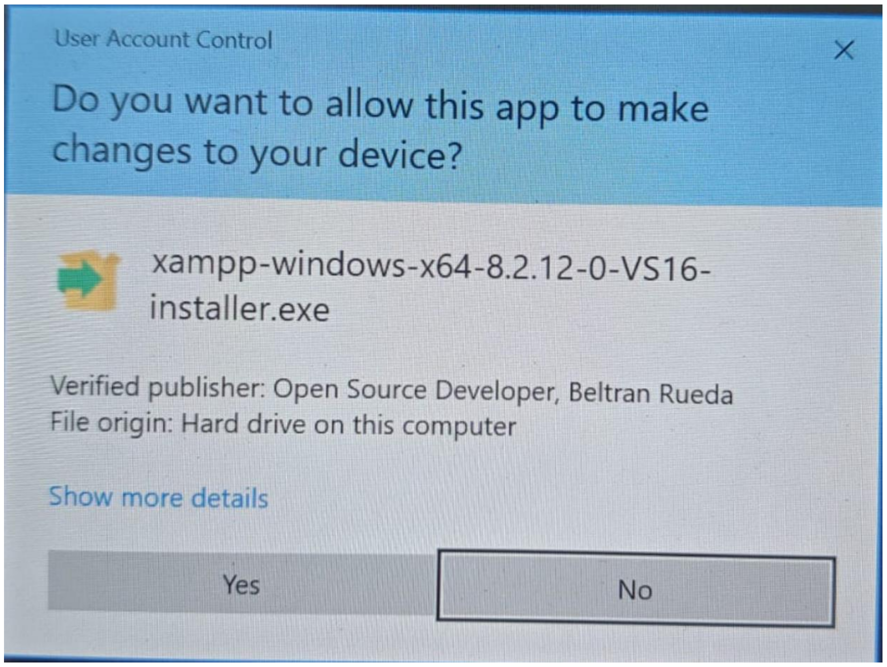
1. User Request (Client Browser)
   - A user opens their browser and types in a URL (for example: http://localhost/first.php).
   - The request is sent to the web server.
2. Web Server (Apache/Nginx/IIS)
   - The server receives the request.
   - It checks if the requested file is a PHP script.
   - If yes, the request is passed to the PHP interpreter (Zend Engine).
3. PHP Processing
   - The PHP interpreter executes the script.
   - Example:
     ```
     <?php
         echo "Hello, World!";
     ?>
     ```
   - If the script needs data, PHP can communicate with a database (e.g., MySQL).
4. Database Interaction (Optional)
   - PHP queries the database.
   - The database sends the required data back to PHP.
5. Response Generated
   - PHP processes the code + database results.
   - It generates a final HTML response.
6. Response to Browser
   - The server sends the HTML back to the client's browser.
   - The browser displays the page (e.g., *Hello, World!*).

"PHP sits between the web server and the database. The browser only sees the final HTML output — never the PHP code."

Setup

XAMPP

1. Download XAMPP xampp-windows-x64-8.2.12-0-VS16-installer .exe from https://www.apachefriends.org/
2. Double click on the exe and click **Yes**



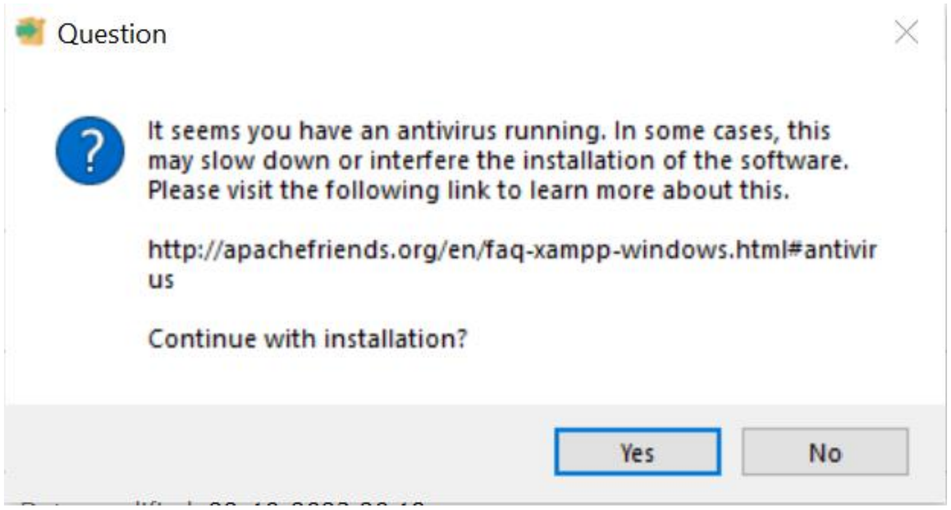
User Account Control

Do you want to allow this app to make changes to your device?

xampp-windows-x64-8.2.12-0-VS16-installer.exe

Verified publisher: Open Source Developer, Beltran Rueda
File origin: Hard drive on this computer

3. Click **Yes**



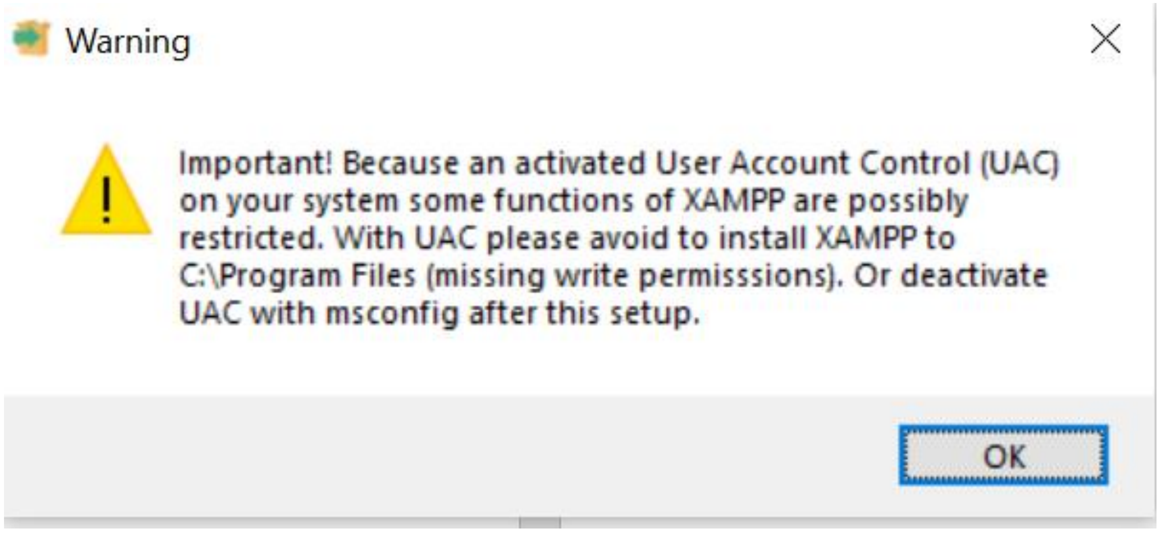
Question

It seems you have an antivirus running. In some cases, this may slow down or interfere the installation of the software. Please visit the following link to learn more about this.

http://apachefriends.org/en/faq-xampp-windows.html#antivirus

Continue with installation?

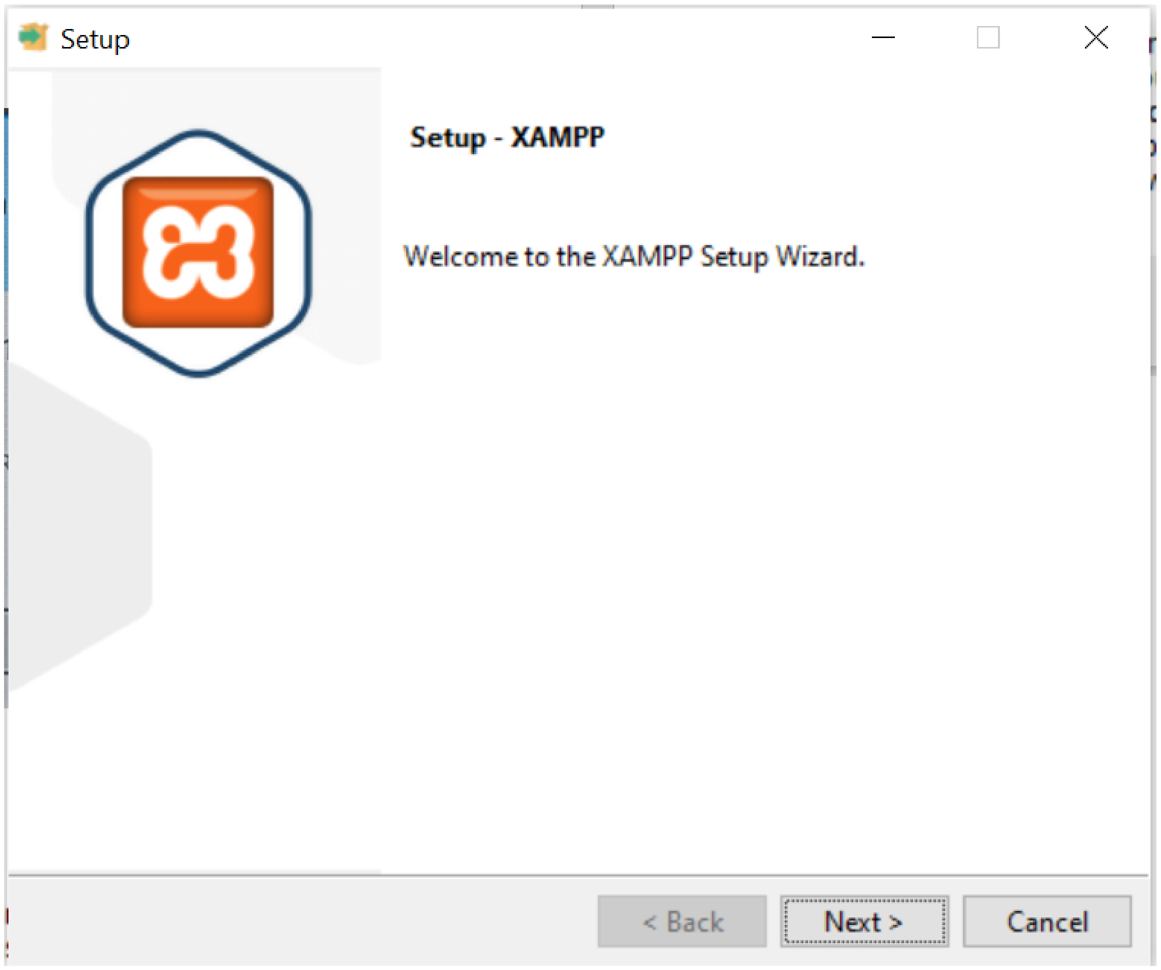
4. Click **OK**



Warning

Important! Because an activated User Account Control (UAC) on your system some functions of XAMPP are possibly restricted. With UAC please avoid to install XAMPP to C:\Program Files (missing write permisssions). Or deactivate UAC with msconfig after this setup.

5. Click **Next**

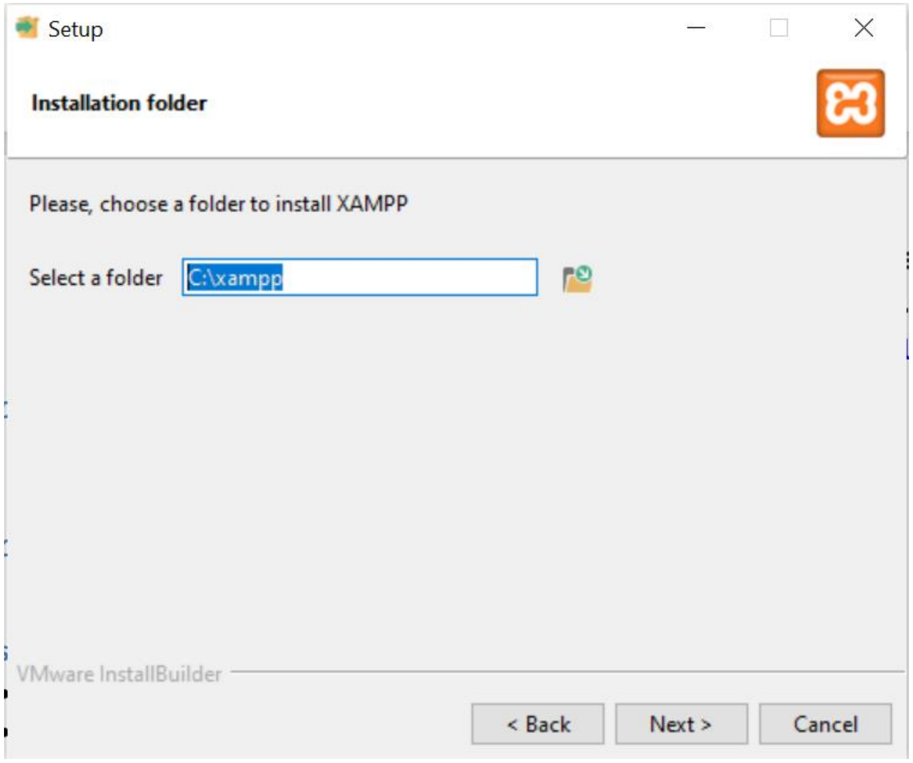


Setup

Setup - XAMPP

Welcome to the XAMPP Setup Wizard.

6. Select only MySQL and phpMyAdmin and click **Next** Click **Next**



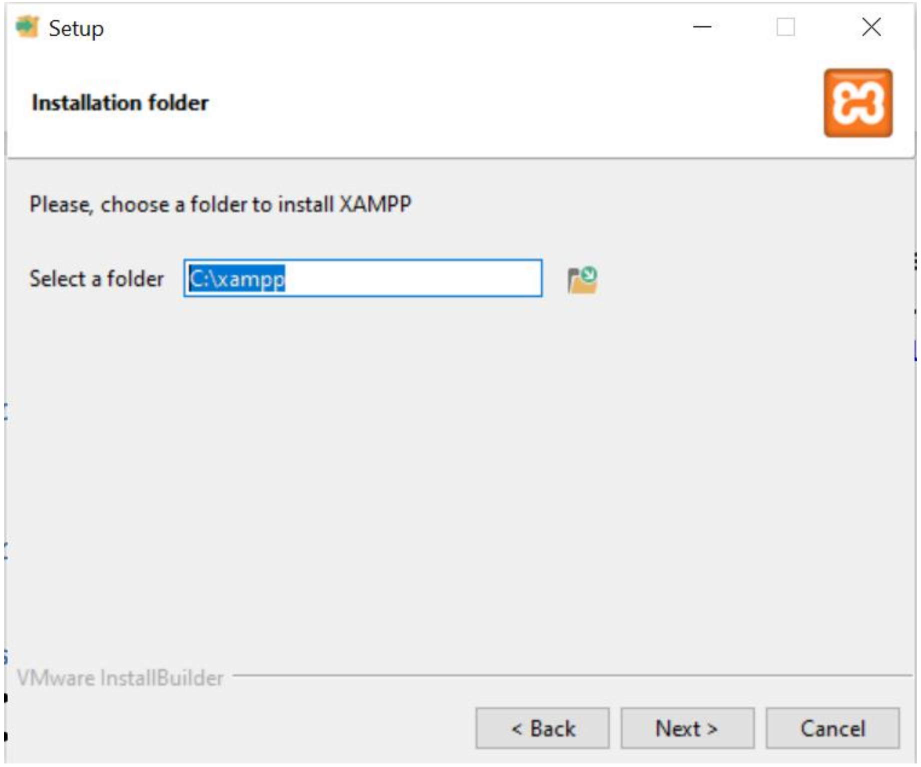
Setup

Installation folder

Please, choose a folder to install XAMPP

Select a folder: C:\xampp

7. Click **Next**



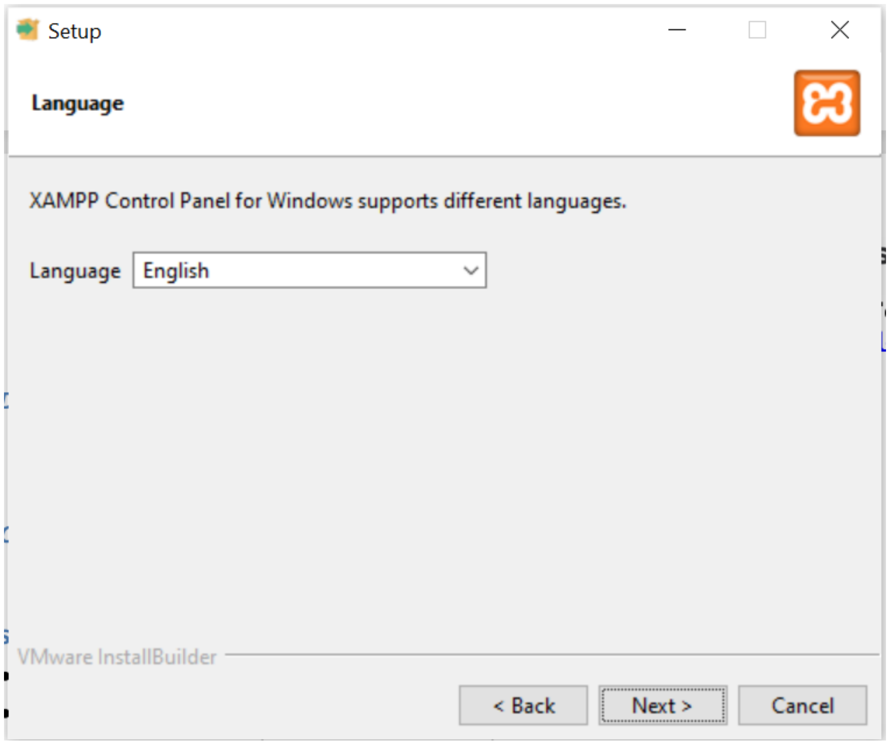
Setup

Installation folder

Please, choose a folder to install XAMPP

Select a folder: C:\xampp

8. Click **Next**

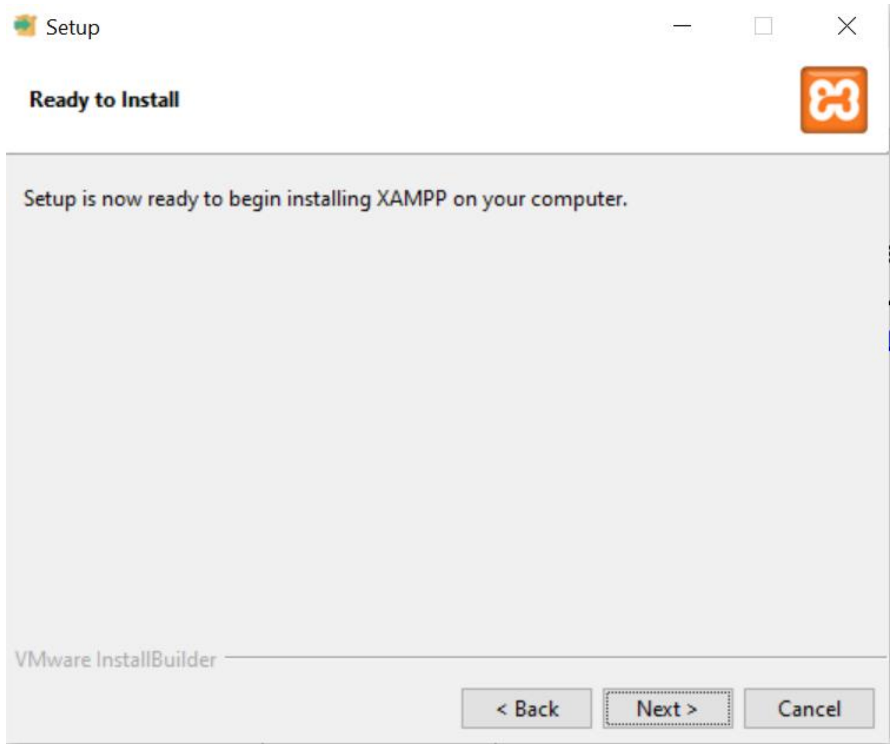


Setup

Language

XAMPP Control Panel for Windows supports different languages.

Language: English

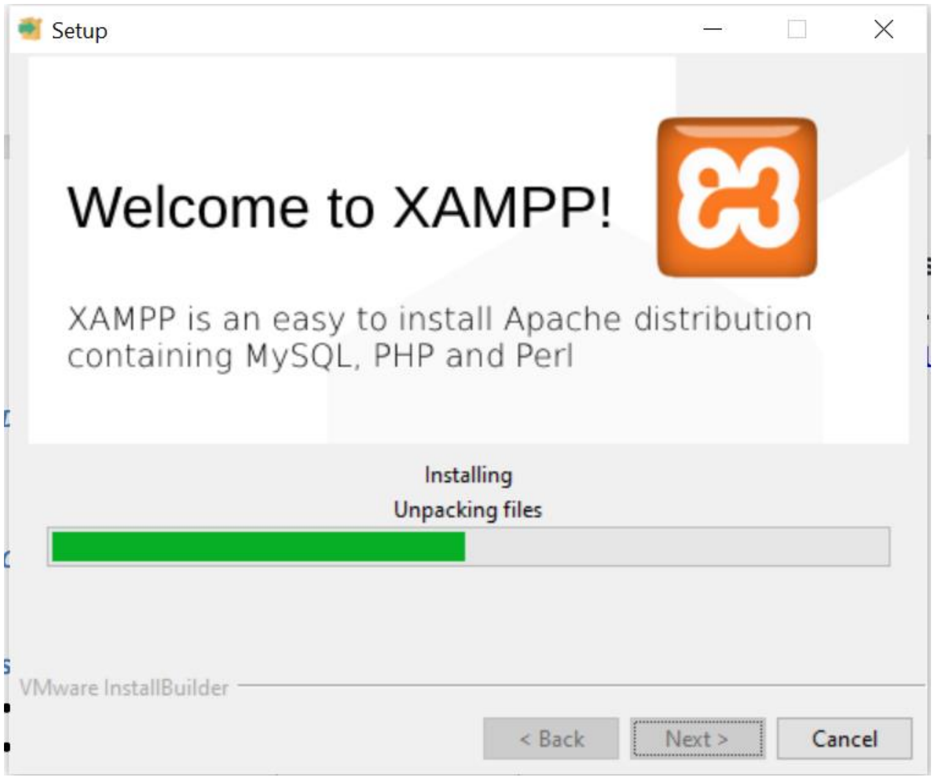


Setup

Ready to Install

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

9. Setup started



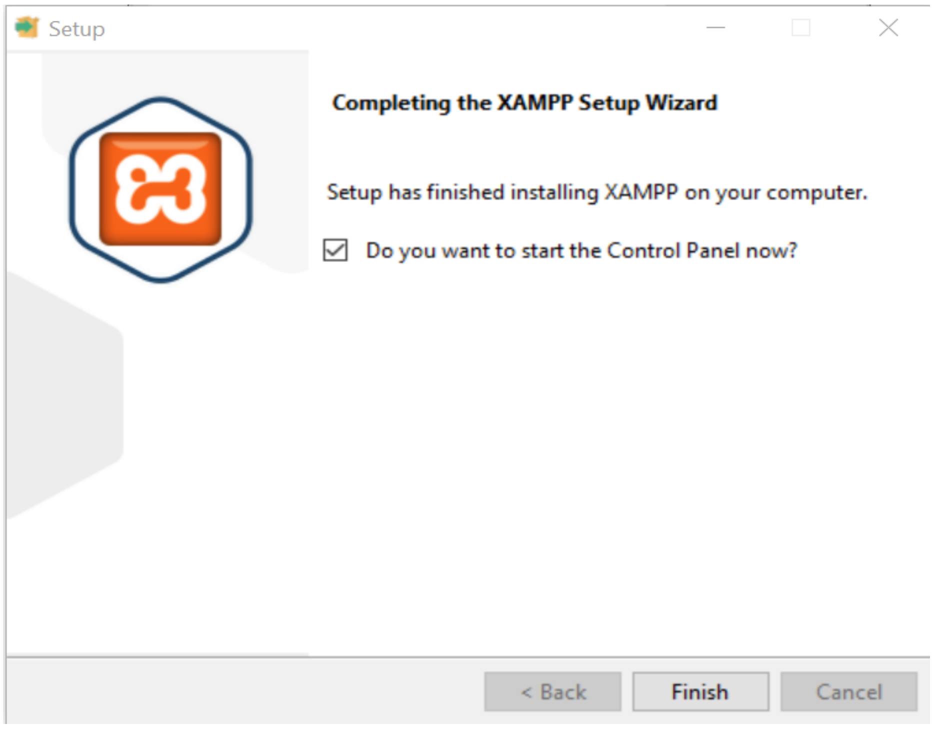
Welcome to XAMPP!

XAMPP is an easy to install Apache distribution containing MySQL, PHP and Perl

Installing

Unpacking files

10. Click **Finish**

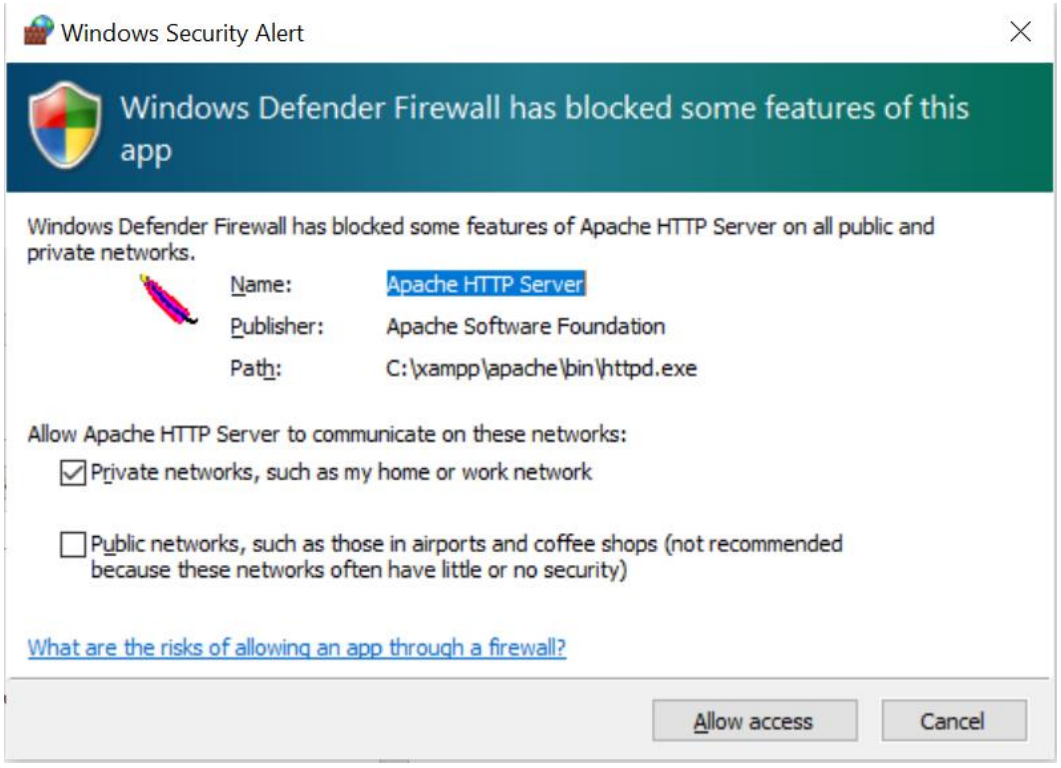


Completing the XAMPP Setup Wizard

Setup has finished installing XAMPP on your computer.

☑ Do you want to start the Control Panel now?

11. If you get this pop-up click **Allow Access**



Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of Apache HTTP Server on all public and private networks.

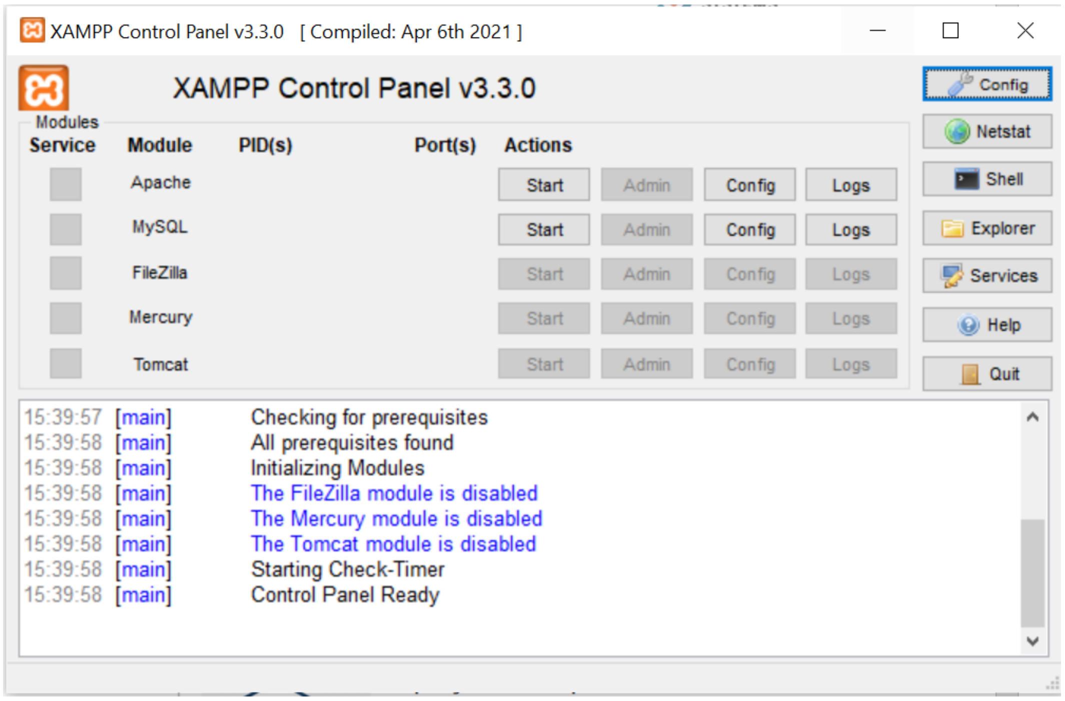
Name:	Apache HTTP Server
Publisher:	Apache Software Foundation
Path:	C:\xampp\apache\bin\httpd.exe

Allow Apache HTTP Server to communicate on these networks:

- [✓] Private networks, such as my home or work network
- [ ] Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

12. If setup is successfully, you should get



XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]

XAMPP Control Panel v3.3.0

Service	Module	PID(s)	Port(s)	Actions			
	Apache			Start	Admin	Config	Logs
	MySQL			Start	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
15:39:57  [main]     Checking for prerequisites
15:39:58  [main]     All prerequisites found
15:39:58  [main]     Initializing Modules
15:39:58  [main]     The FileZilla module is disabled
15:39:58  [main]     The Mercury module is disabled
15:39:58  [main]     The Tomcat module is disabled
15:39:58  [main]     Starting Check-Timer
15:39:58  [main]     Control Panel Ready
```

13. Check the Webserver
    a. Click Start button inline with Apache
    b. Go to http://localhost, You should get the Web page
14. Check the MySQL
    a. Click start button inline with MySQL
    b. Go to http://localhost/phpmyadmin/
    c. Go to SQL and run basic query "Show Databases;" and click on Go
    d. It should show all the available database.
15. Check php installation

- Find PHP folder For XAMPP → C:\xampp\php
- Copy the path (e.g., C:\xampp\php).
- Add to Environment Variables:
  - Press Win + R → sysdm.cpl
  - Go to Advanced → Environment Variables
  - Under System variables, select Path → Edit
  - Click New → paste your PHP path → OK
- Run php -v
- Create hello.php and keep it in C:\xampp\htdocs and go to http://localhost/hello.php
- Or simple run php hello.php

VS Code

1. Download VS Code https://code.visualstudio.com/

Let's Start with PHP

echo and print

- Use **echo** when you just want to print something (most common).
- Use **print** if you also want to **return a value** inside an expression.

Feature	echo	print
Argument	Multiple	Only one
Return Value	None	Return 1
Speed	Slightly faster	Slightly slower
Use of Expression	No	Yes
Common Usage	General Output	Output+ return

Variable and Constants

Variables

- A variable is used to store data (like numbers, text, etc.).
- In PHP, variables always start with a $ sign.

Syntax:

```
$age = 25;
$name = "Prabhat";
$price = 99.50;
echo $name;   // Output: Prabhat
```

Rules for Variables:

- Must start with $ + a letter or underscore.
- Case-sensitive ($Name ≠ $name).
- Can hold any type (string, integer, array, object, etc.).

Constants

- A constant is like a variable, but its value cannot change once defined.
- Used for fixed values (e.g., PI, DB name).

Syntax :

```
// 1. Using define()
define("SITE_NAME", "Swacode");
echo SITE_NAME;   // Output: Swacode

// 2. Using const
const PI = 3.14159;
echo PI;   // Output: 3.14159
```

Rule for Constants:

- No $ sign used.
- Are global and can be accessed anywhere.
- Faster than variables for fixed values.

Primitives (Basic Data Types)

Primitives are the **basic building blocks** of data in PHP.

- **Integer** → whole numbers (e.g., 85)
- **Float/Double** → decimal numbers (e.g., 75.5)
- **String** → sequence of characters (e.g., "Prabhat")
- **Boolean** → true or false values (e.g., true)
- **NULL** → a variable with no value assigned

PHP is **loosely typed** → you don't need to declare variable types; they are decided at runtime.

Operations

Operations allow us to perform **calculations and assignments** on variables.
Arithmetic Operators → +, -, *, /, %, **

Assignment Operators → =, +=, -=
Increment / Decrement → ++, --
Example:
$x = 10; $y = 3;
$x + $y → 13
$x % $y → 1
$x += 5 → 15

Expressions

An **expression** is a combination of variables, values, and operators that **evaluates to a result**.

- **Comparison Operators** → >=, <=, ==, ===
- **Logical Operators** → &&, ||, !
- **Ternary Expression** → short form of if/else
  ($marks >= 40) ? "Pass" : "Fail"
- **Null Coalescing Operator** →

$name ?? "Guest"   // returns "Guest" if $name is null

Control Statements

- Control statements decide the **flow of execution**.
- **if / else if / else** → for range-based conditions
- **switch** → for fixed value comparisons

```
Example:
if ($marks >= 90) echo "A";
elseif ($marks >= 75) echo "B";
else echo "C or below";

switch ($role) {
  case "admin": echo "Admin"; break;
  case "editor": echo "Editor"; break;
  default: echo "Other";
}
```

Arrays

Arrays store **multiple values in a single variable**.

- **Indexed Arrays** → keys are numbers.
  Example: [85, 78, 92]

- **Associative Arrays** → keys are words.
  Example: ["AI"=>85, "DS"=>78, "ML"=>92]

Use foreach loop to iterate through arrays easily.

Functions

- Functions are **reusable blocks of code** that perform specific tasks.
- Defined using the function keyword.
- Can take **parameters** and return **values**.
- Support **type hints** and **return types** for clarity.

  Example:
  ```
  function greet(string $name="Guest"): string {
    return "Hello, $name!";
  }
  echo greet("Prabhat");   // Hello, Prabhat
  ```
- Functions help in:
- Reducing repetition
- Improving modularity
- Making code easy to maintain

Handling HTML Forms with PHP

Forms are an essential way to collect input from users in web applications. PHP makes it simple to handle and process form data.

Handling forms with $_GET and $_POST

- $_GET Method:
  - Sends form data via the URL as query parameters.
  - Example: form.php?name=John&age=22
  - Suitable for: search forms, filters, and when data is not sensitive.
  - Limitation: data length is limited (around 2000 characters in most browsers).
  - Example :
    ```
    <?php
    if (isset($_GET['name'])) {
       echo "Hello, " . $_GET['name'];
    }
    ?>
    ```
- $_POST Method:
  - Sends form data in the HTTP request body.
  - Suitable for: login forms, payment processing, and sensitive information.
  - No size limitations for data.
  - Example:
    ```
    <?php
    if (isset($_POST['username'])) {
       echo "Welcome, " . $_POST['username'];
    }
    ?>
    ```

Using Cookies and Sessions

Cookies:

- Small text files stored in the user's browser.
- Can store preferences, login states, or tracking information.
- Example:

```
// Setting a cookie
setcookie("user", "John", time() + 3600, "/");

// Retrieving a cookie
if (isset($_COOKIE['user'])) {
   echo "Welcome back, " . $_COOKIE['user'];
}
```

Sessions:

- Data is stored on the server, only a session ID is sent to the browser.
- More secure compared to cookies.
- Example:

```
session_start();
$_SESSION['username'] = "John";
echo "Session started for " . $_SESSION['username'];
```

Introduction to Session Tracking

- Session tracking is the mechanism of maintaining user interactions across multiple requests.
- PHP assigns a unique session ID for each user.
- Common use cases:
  - Shopping carts
  - Login authentication
  - Personalized dashboards

File Upload, Read, and Write in PHP

File Upload

HTML Form with file input:

```
<form action="upload.php" method="post" enctype="multipart/form-data">
   <input type="file" name="myfile">
   <input type="submit" value="Upload">
</form>
```

PHP script (upload.php):

```
if (move_uploaded_file($_FILES['myfile']['tmp_name'], "uploads/" .
$_FILES['myfile']['name'])) {
   echo "File uploaded successfully!";
} else {
   echo "Error uploading file.";
```

}

File Read/Write

- Reading a file:
  ```
  $file = fopen("data.txt", "r");
  echo fread($file, filesize("data.txt"));
  fclose($file);
  ```
- Writing to a file:
  ```
  $file = fopen("data.txt", "w");
  fwrite($file, "Hello PHP File Handling!");
  fclose($file);
  ```

Connecting PHP with MySQL and CRUD Operations

Connecting to MySQL

```
$conn = new mysqli("localhost", "root", "", "mydb");
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully!";
```

CRUD Operations

- Create (Insert)
  ```
  $sql = "INSERT INTO users (name, email) VALUES ('John', 'john@example.com')";
  $conn->query($sql);
  ```
- Read (Select)
  ```
  $result = $conn->query("SELECT * FROM users");
  while ($row = $result->fetch_assoc()) {
     echo $row['name'] . " - " . $row['email'] . "<br>";
  }
  ```
- Update
  ```
  $sql = "UPDATE users SET email='newemail@example.com' WHERE name='John'";
  $conn->query($sql);
  ```
- Delete
  ```
  $sql = "DELETE FROM users WHERE name='John'";
  $conn->query($sql);
  ```

Overview of WAP and WML

- Wireless Application Protocol (WAP):
- Developed in the late 1990s to allow mobile devices with limited hardware to access internet content.
- Optimized for small screens and low bandwidth.
- Works over 2G and GSM networks.
- Wireless Markup Language (WML):
- An XML-based markup language designed for WAP browsers.

- Similar to HTML but lightweight.
  Example WML code:
  ```
  <?xml version="1.0"?>
  <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
  "http://www.wapforum.org/DTD/wml_1.1.xml">
  <wml>
    <card id="welcome" title="Welcome">
      <p>Hello, WAP World!</p>
    </card>
  </wml>
  ```

Relevance Today:
WAP and WML are now obsolete due to modern smartphones and responsive web design. However, they represent an important step in the history of mobile web access.