Introducción a PHP

- PHP: Lenguaje de scripting del lado del servidor adecuado para el desarrollo web.
- Origen del Nombre: PHP originalmente significaba "Página de Inicio Personal" y ahora es "Preprocesador de Hipertexto".
- Creador: Rasmus Lerdorf en 1994; mantenido por el equipo de desarrollo de PHP.

PHP como Lenguaje del Lado del Servidor

- Cuando se abre un sitio web, el navegador envía solicitud HTTP al servidor web.
- El servidor web procesa la solicitud y devuelve un documento HTML como respuesta.
- PHP se ejecuta en el servidor, procesa la solicitud y genera el HTML.
- Cliente: Navegador web; Servidor: Web server.

PHP como Lenguaje de Propósito General

- PHP es un lenguaje de programación diseñado para ser versátil y capaz de abordar una amplia gama de tareas y proyectos. No está limitado a un solo tipo de aplicación o industria específica, es flexible y adaptable, lo que lo hace adecuado para crear una amplia variedad de programas y soluciones

PHP como Lenguaje Multiplataforma

- Ejecutable en sistemas operativos como Linux, Windows y macOS.
- Compatibilidad con servidores web como Nginx, OpenBSD y Apache.
- Admite generación de imágenes PDF, GIF, JPEG y PNG.
- Soporte para varias bases de datos como MySQL, PostgreSQL, MS SQL, Oracle Database y MongoDB.

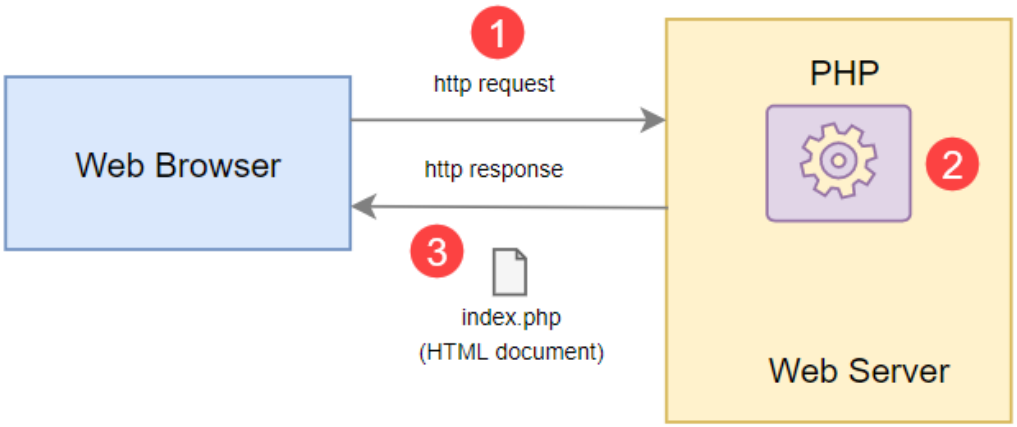
Capacidades de PHP

- Scripts del Lado del Servidor: Desarrollo de sitios y aplicaciones web dinámicos.
- Secuencias de Comandos de Línea de Comandos: Tareas administrativas como enviar correos electrónicos y generar archivos PDF.
- Enfoque principal de los tutoriales: Secuencias de comandos del lado del servidor.

Esta versión esquematizada presenta los aspectos clave de la introducción a PHP, destacando sus orígenes, funcionalidades y aplicaciones principales. Puede ser útil para presentar la información de manera concisa y accesible a tus estudiantes.

Cómo funciona PHP

A continuación, se ilustra cómo funciona PHP:



Cómo Funciona PHP

- Solicitud del Navegador: El navegador web envía una solicitud HTTP al servidor web (por ejemplo, index.php).
- Procesamiento en el Servidor: El preprocesador PHP en el servidor web procesa el código PHP para generar el documento HTML.
- Respuesta al Navegador: El servidor web envía el documento HTML generado de vuelta al navegador web.

Ventajas de PHP

- Simplicidad: PHP es fácil de aprender y comenzar a utilizar.
- Velocidad: Sitios web PHP tienden a funcionar de manera rápida.
- Estabilidad: PHP ha existido durante mucho tiempo y es conocido por su estabilidad.
- Código Abierto y Gratuito: PHP es de código abierto y gratuito, no requiere pago de licencia.
- Soporte Comunitario: Una comunidad en línea activa está disponible para ayudar en la resolución de problemas.

Instalación de PHP

La instalación de PHP en su computadora local permite el desarrollo y pruebas seguras de aplicaciones web sin afectar sistemas en producción.

Requisitos

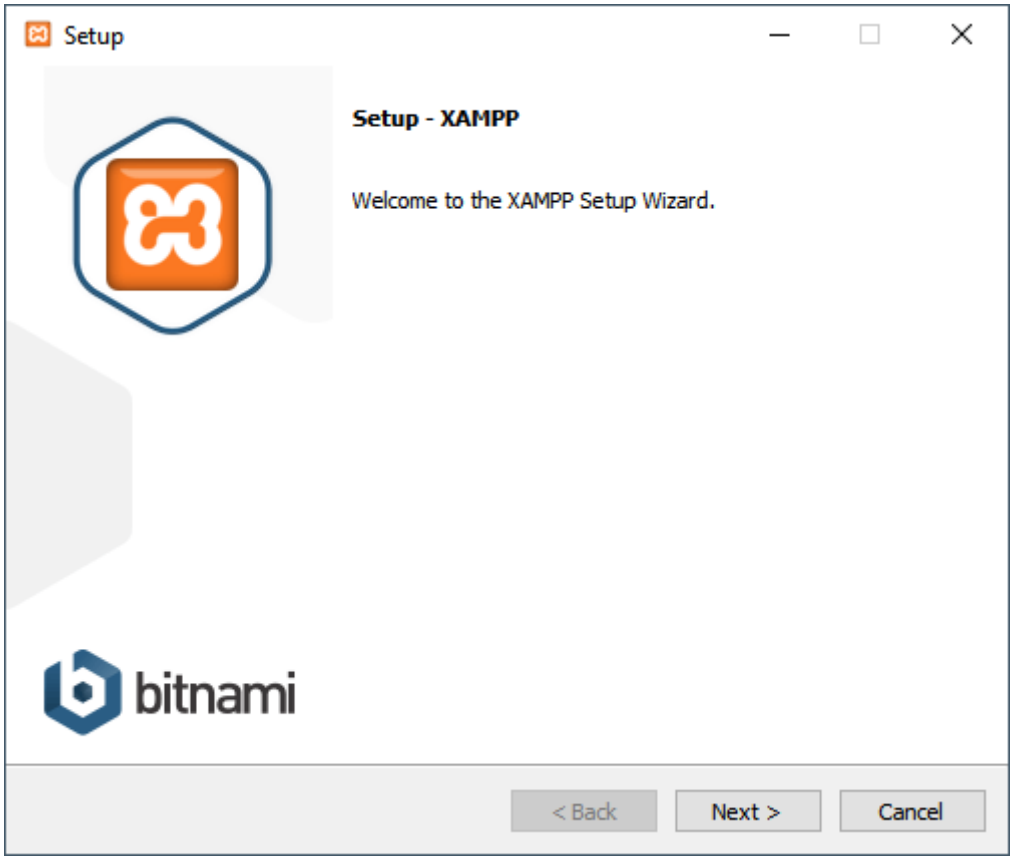
- PHP
- Servidor web que admita PHP (usaremos Apache).
- Servidor de base de datos (usaremos MySQL).

XAMPP

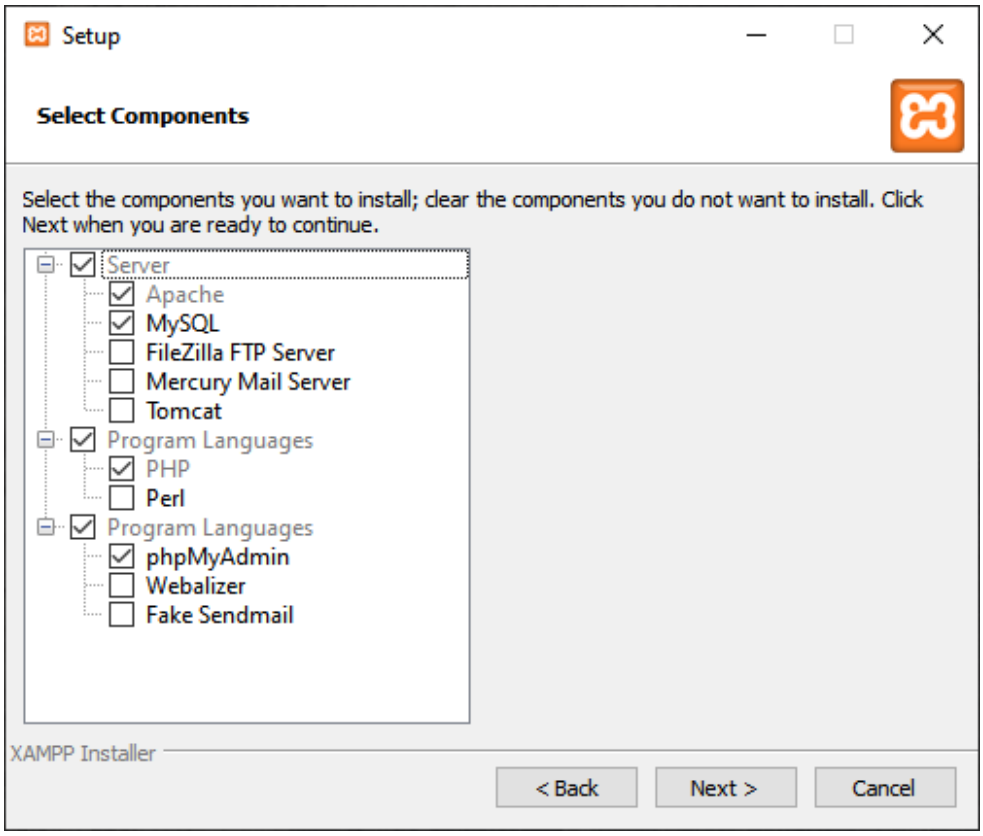
XAMPP es una distribución Apache que contiene PHP, MariaDB y Apache en un solo paquete. Es compatible con Windows, Linux y macOS.

Descarga e Instalación

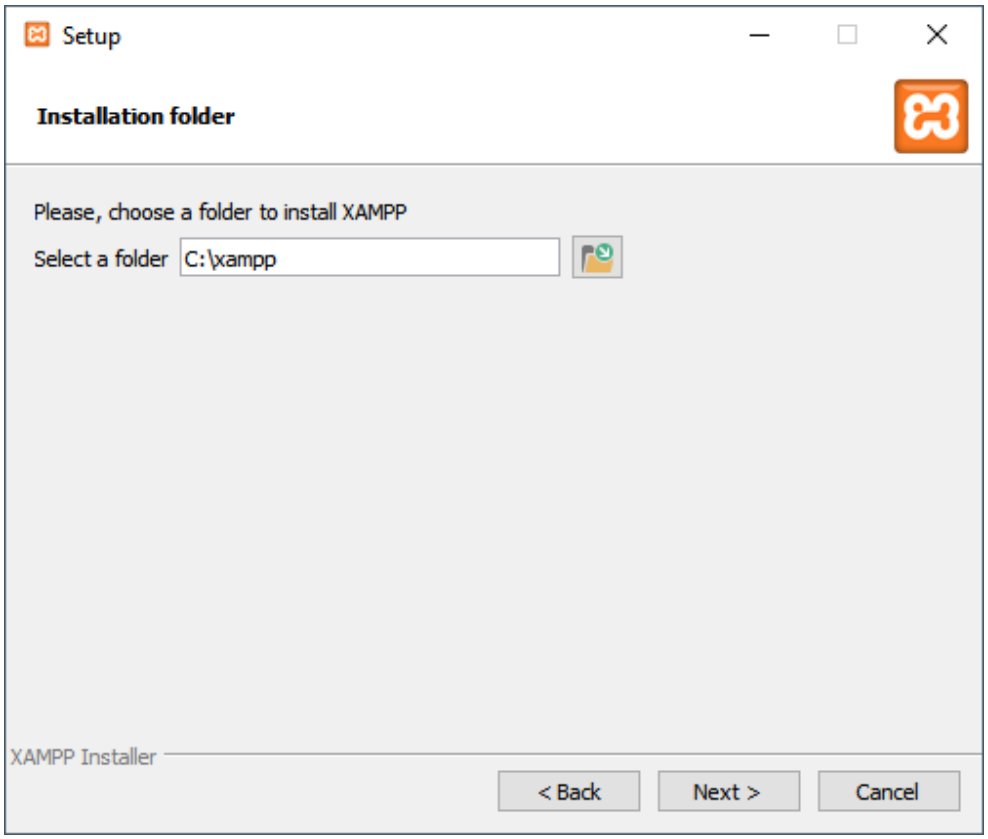
- Descargue la versión adecuada de XAMPP desde el sitio oficial.
- Siga los pasos para instalar XAMPP:



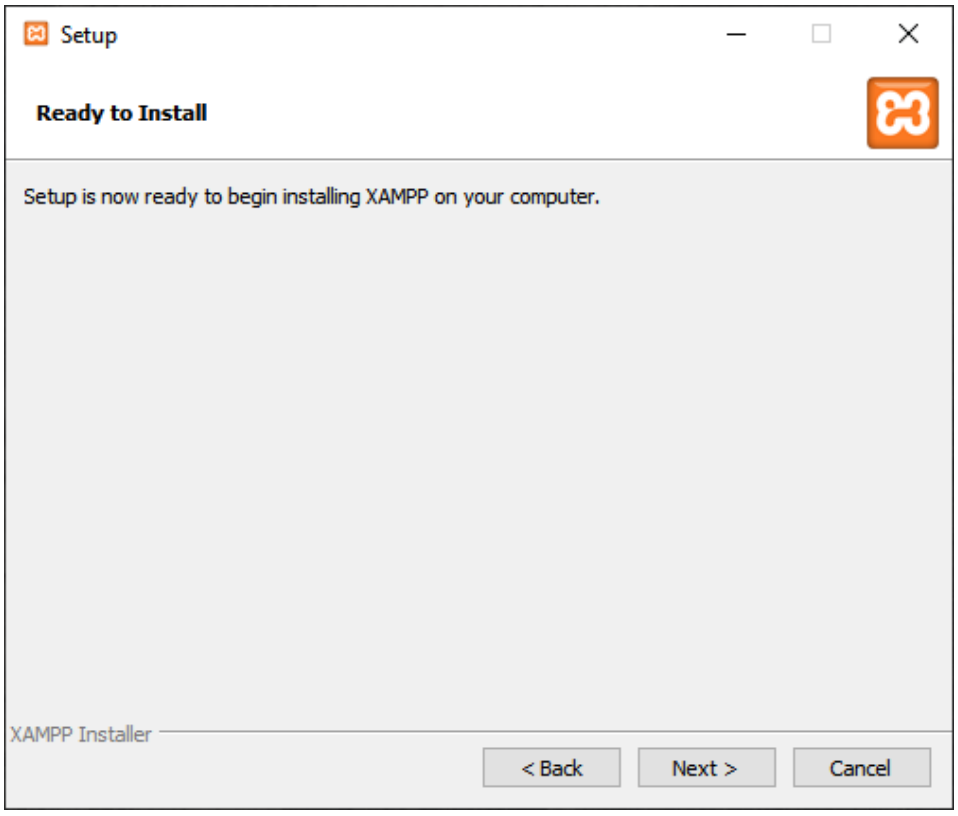
Seleccione los componentes que desea instalar:



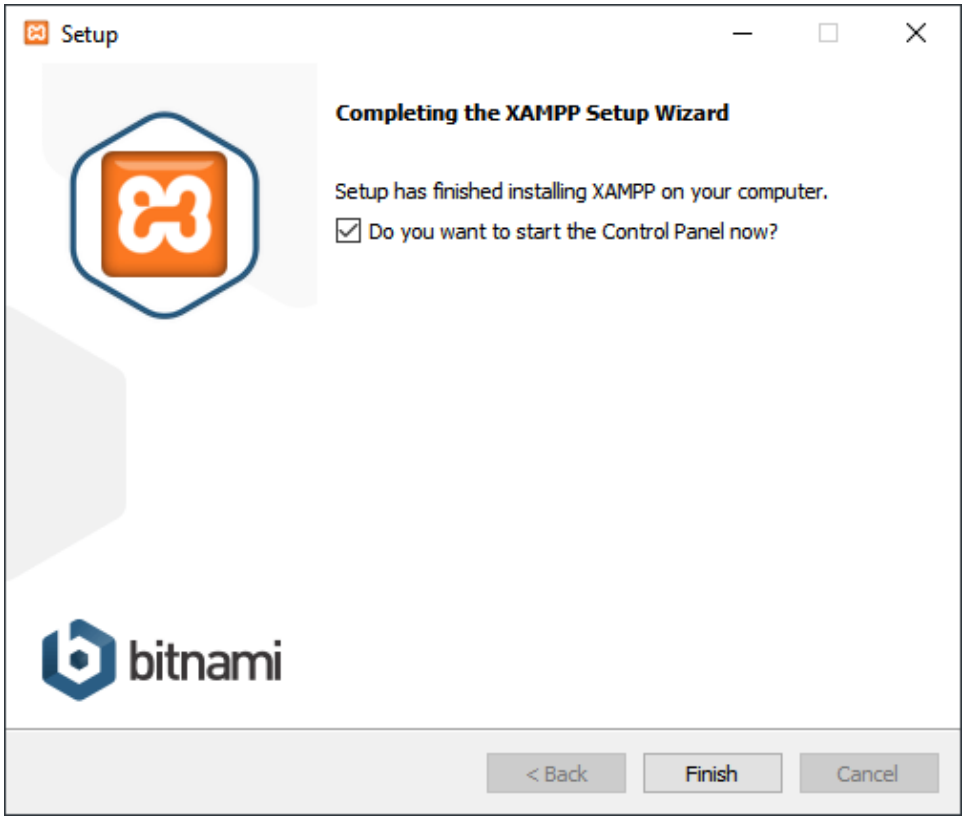
Se recomienda instalar XAMPP en la carpeta c:\xampp:



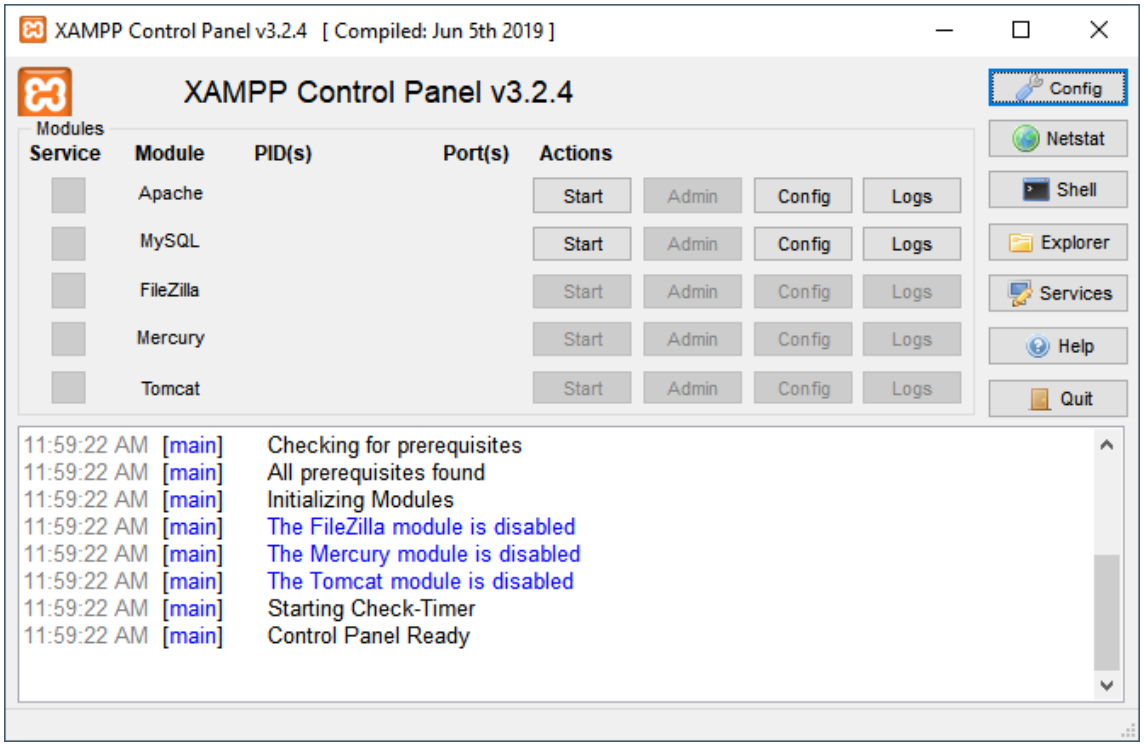
Y ahora estás listo para instalar XAMPP. Haga clic en el botón Siguiente para iniciar la instalación. Tardará unos minutos en completarse.



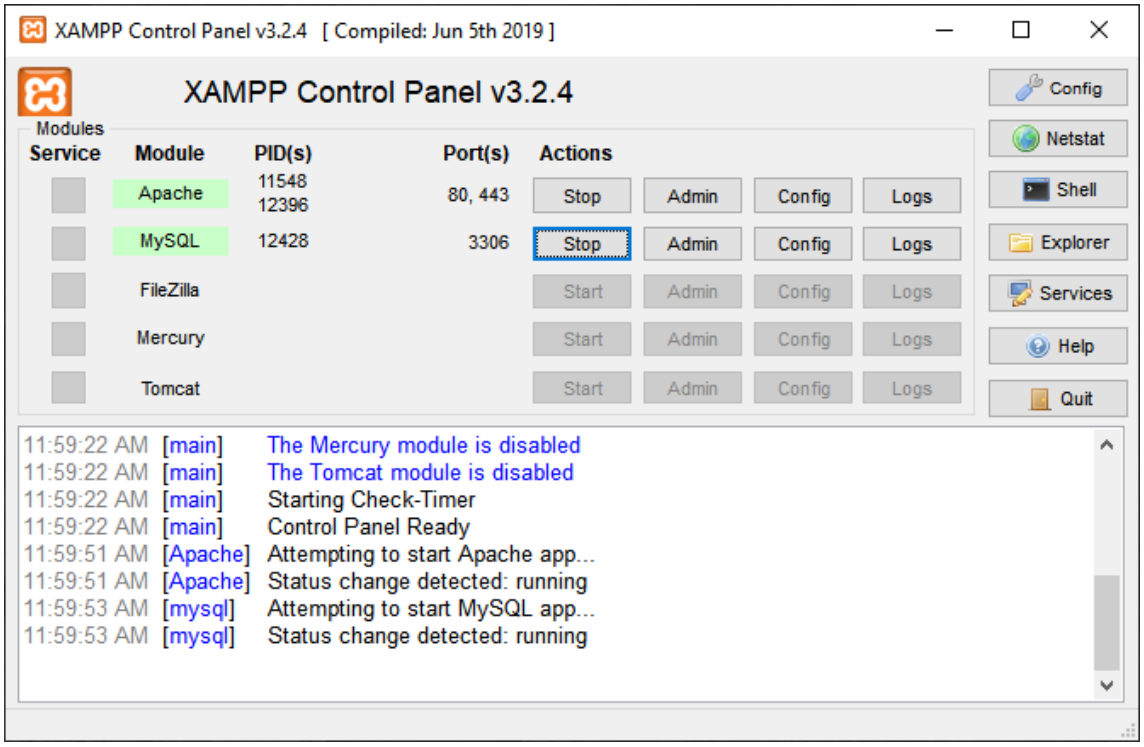
Finalizar para iniciar el Panel de control de XAMPP:



El Panel de control de XAMPP enumera los servicios instalados. Para iniciar un servicio, haga clic en el botón Iniciar correspondiente:



A continuación, se muestra que el servidor web Apache y MySQL están en funcionamiento. El servidor web Apache escucha en los puertos 80 y 443 mientras que MySQL escucha en el puerto 3306:



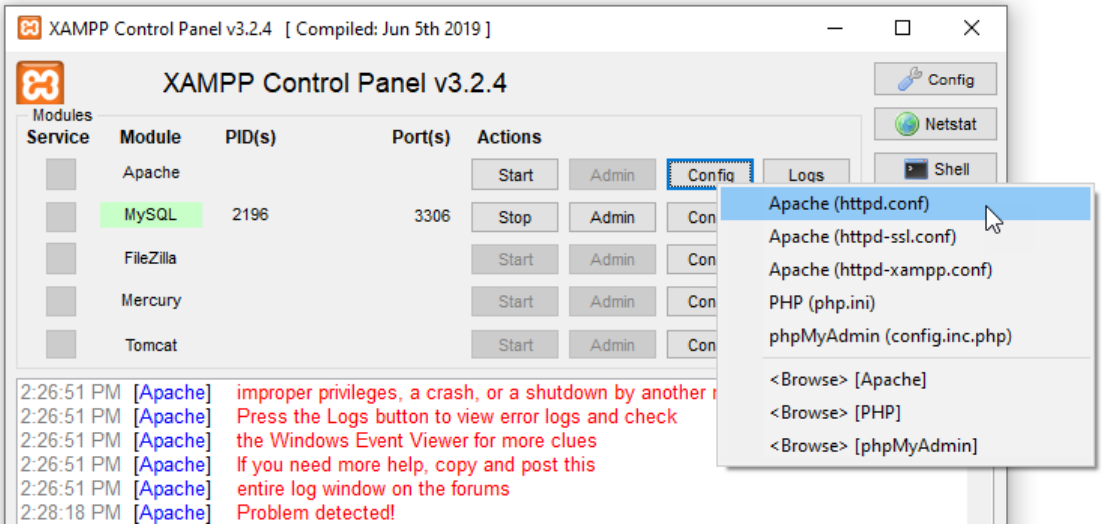
Iniciar XAMPP → URL: http://localhost/.

Si la instalación se completa con éxito, verá la pantalla de bienvenida de XAMPP.

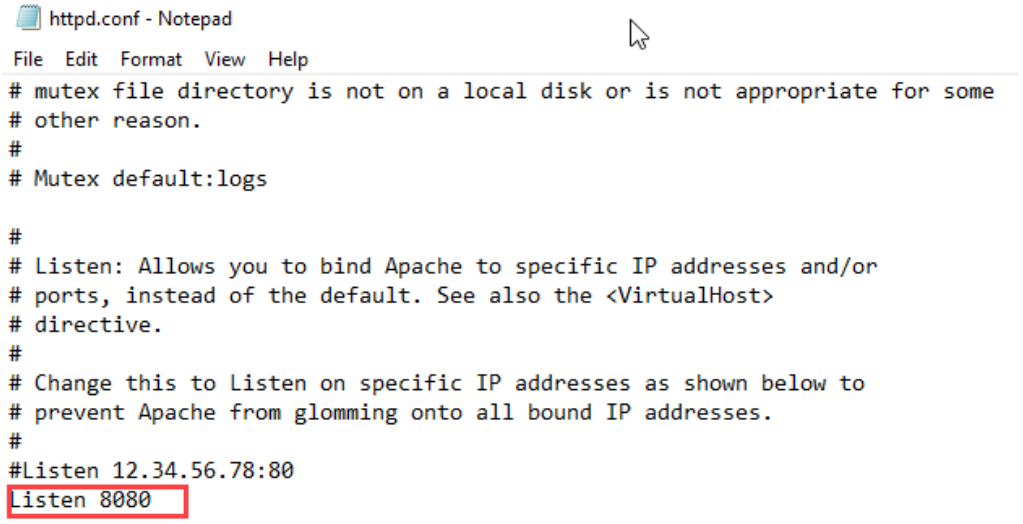
Solución de Problemas: Cambiar el Puerto de Apache en XAMPP

Si experimenta un problema en el que Apache no puede iniciar debido a que el puerto 80 está ocupado por otro servicio, puede solucionarlo cambiando el puerto en XAMPP. Siga estos pasos:

- Abra el Panel de Control de XAMPP.
- Junto al módulo de Apache, haga clic en el botón "Config" para abrir la configuración de Apache.

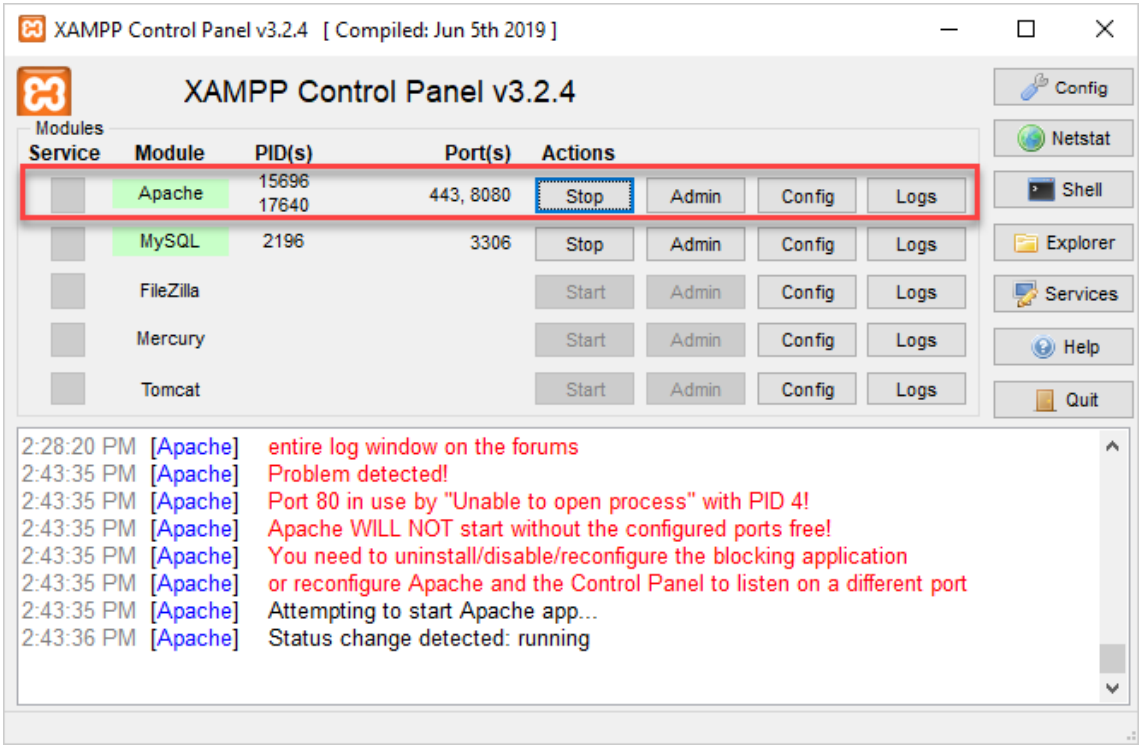


- En el archivo de configuración de Apache, busque la línea que contiene el número de puerto 80. Puede parecer algo así: Listen 80
- Cambie el número de puerto 80 a un número de puerto libre, como 8080: Listen 8080



- Guarde los cambios en el archivo de configuración.

- Regrese al Panel de Control de XAMPP y haga clic en el botón "Start" junto al módulo de Apache para intentar iniciar Apache nuevamente.



Ahora, cuando acceda a su servidor local, especifique el nuevo puerto en la URL. Por ejemplo, si su dirección local es "http://localhost", cambie la URL a "http://localhost:8080".

Creación de un "Hola Mundo" en PHP

Para crear un sencillo "Hola Mundo" en PHP, siga estos pasos:

- Abra la carpeta htdocs en la carpeta de instalación de XAMPP. Por lo general, se encuentra en C:\xampp\htdocs.
- Cree una nueva carpeta llamada helloworld.
- En la carpeta helloworld, cree un nuevo archivo llamado index.php y coloque el siguiente código:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>PHP - Hello, World!</title>
</head>
<body>
    <h1><?php echo 'Hello, World!'; ?></h1>
</body>
</html
```

El código de index.php es similar a un documento HTML normal, pero incluye código PHP entre las etiquetas **<?php y ?>**. La línea **<?php echo 'Hello, World!'; ?>** imprime el mensaje "Hello, World!" dentro de una etiqueta h1.

- Abra un navegador web y visite la URL: http://localhost:8080/helloworld/.

Si ve el mensaje “Hello, ¡World!" en el navegador, ha ejecutado con éxito su primer script PHP.

- También puede ejecutar el script desde la línea de comando:

c:\xampp\htdocs\helloworld>php index.php

Nota: Si el archivo contiene solo código PHP y no incluye HTML, no es necesario usar la etiqueta de cierre **?>** al final del archivo.

Sintaxis básica de PHP

La sintaxis en PHP es fundamental para escribir programas correctamente. Aquí hay una introducción a la sintaxis básica de PHP:

Etiquetas de PHP

Para comenzar un bloque de código PHP, use la etiqueta de apertura **<?php**. Si está mezclando código PHP con HTML, puede abrir y cerrar bloques de código PHP dentro de las etiquetas **<?php y ?>.** Si el archivo contiene solo código PHP, la etiqueta de cierre es opcional.

Ejemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PHP Syntax</title>
</head>
<body>
    <h1><?php echo 'PHP Syntax'; ?></h1>
</body>
</html>
```

Sensibilidad entre Mayúsculas y Minúsculas

En PHP, existen distinciones parciales entre mayúsculas y minúsculas. Algunos elementos no distinguen entre mayúsculas y minúsculas, como construcciones PHP (if, else, while, etc.), palabras clave (true, false) y nombres de funciones y clases definidos por el usuario. Sin embargo, las variables distinguen entre mayúsculas y minúsculas.

```php
$message = "Hello"; // Variable
if ($is_new_user) { // Construcción if
    send_welcome_email(); // Función
}
```

Declaraciones

Las declaraciones son piezas individuales de código que realizan acciones, como asignar valores a variables o llamar a funciones. Cada declaración termina con un punto y coma ;. Las declaraciones pueden ser simples o compuestas.

Declaración simple:

```php
$message = "Hello";
```

Declaración compuesta:

```php
if ($is_new_user) {
    send_welcome_email();
}
```

Espacios en Blanco y Saltos de Línea

En PHP, los espacios en blanco y los saltos de línea generalmente no tienen un significado especial. Puede colocar declaraciones en una línea o distribuirlas en varias líneas para mejorar la legibilidad.

```php
login($username, $password); // Una línea
```

```php
login(
    $username,
    $password
 ); // Varias líneas
```

Variables en PHP

En PHP, las variables son utilizadas para almacenar y manipular datos en programas. Aquí aprenderás cómo utilizar variables en PHP:

Para definir una variable en PHP, utiliza la siguiente sintaxis:

```php
$nombre_variable = valor;
```

Al definir una variable, ten en cuenta estas reglas:

- **El nombre de la variable debe comenzar con el símbolo de dólar ($).**
- **El primer carácter después del símbolo de dólar ($) debe ser una letra (a-z) o un guión bajo (_).**
- **Los caracteres siguientes pueden ser guiones bajos, letras o números.**
- **Las variables distinguen entre mayúsculas y minúsculas, lo que significa que $mensaje y $Mensaje son variables diferentes.**

Mostrar el Valor de una Variable

Para mostrar el valor de una variable en una página web, utiliza la construcción echo. Puedes incluir el valor de la variable en tu HTML de la siguiente manera:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Variables en PHP</title>
</head>
<body>
    <?php
        $titulo = '¡PHP es increíble!';
        echo '<h1>' . $titulo . '</h1>';

      // forma más corta
        $titulo_dos = '¡PHP es increíble!';
    ?>
    <h1><?= $titulo_dos ?></h1>

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

Separación de Preocupaciones (Separation of concerns)

Para mantener tu código organizado y mantenible, es recomendable dividir el código en archivos diferentes y luego incluirlos cuando sea necesario.

```
<?php
$titulo = '¡PHP es increíble!';
require 'index.view.php';
?>
```

index.view.php:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Variables en PHP</title>
</head>
<body>
    <h1><?= $titulo ?></h1>
</body>
</html>
```

Comentarios en PHP

Los comentarios son partes esenciales del código que proporcionan información útil para entenderlo. PHP admite dos tipos de comentarios:

1) Comentarios de una línea

- Inician con # o //.
- Se colocan al final de la línea o en el bloque actual.

```
$tarifa = 100;
$horas = 173;
$pago = $horas * $tarifa; // Cálculo del pago
```

2) Comentarios de varias líneas

- Inician con /* y terminan con */.
- Útiles para abarcar comentarios en múltiples líneas.

```
/*
   Este es un ejemplo de un comentario
   de varias líneas.
*/
```

Constantes en PHP

Las constantes en PHP son nombres que representan un valor único, y una vez definidas, su valor no cambia durante la ejecución del script. Puedes utilizar la función define() o la palabra clave const para definir constantes.

Definir constantes con define()

```php
define('WIDTH', '1140px');
echo WIDTH; // Imprime: 1140px
```

- Los nombres de constantes se escriben en mayúsculas.
- Los nombres de las constantes no comienzan con el signo de dólar $.
- Las constantes distinguen entre mayúsculas y minúsculas (a menos que se especifique lo contrario).

Definir constantes con la palabra clave const

```php
const SALES_TAX = 0.085;

$gross_price = 100;
$net_price = $gross_price * (1 + SALES_TAX);
echo $net_price; // Imprime: 108.5
```

- Las constantes definidas con const son creadas en tiempo de compilación.
- Pueden contener valores escalares como números, cadenas y arreglos.

Diferencias entre define() y const

- define() es una función, mientras que const es una construcción del lenguaje.
- Puedes usar define() condicionalmente en tiempo de ejecución, pero no const.
- define() permite definir constantes con nombres generados por expresiones.

Eligir define() si se necesita mayor flexibilidad o si se quiere definir constantes condicionalmente. Utilizar const para definir constantes de manera más clara y directa.

Función var_dump() en PHP

La función var_dump() muestra información detallada sobre una variable:

- Puedes usarla para mostrar el tipo y valor de una variable.
- Envolver var_dump() en <pre> mejora la legibilidad.
- Crear una función d($data) para una salida más conveniente:

```php
function d($data) {
    echo '<pre>';
    var_dump($data);
    echo '</pre>';
}

$balance = 100;
d($balance);
```

- Se puede combinar var_dump() con die() para mostrar y detener:

```php
function d($data) {
    echo '<pre>';
    var_dump($data);
    echo '</pre>';
}

$balance = 100;
d($balance);
```

- Crea una función dd() para mayor reutilización:

```php
function dd($data) {
    echo '<pre>';
    var_dump($data);
    echo '</pre>';
    die();
}

$message = 'Dump and die example';
dd($message);
```

La función **var_dump()** es útil para depuración y comprensión de variables en tu código. Personaliza la salida para legibilidad y usa funciones auxiliares como dd() para mejorar el flujo de trabajo.

Tipos de Datos en PHP

Tipos Escalares

- **bool:** Representa valores verdaderos o falsos.
- **int**: Representa números enteros.
- **float**: Representa números de punto flotante (decimales).
- **string**: Representa cadenas de texto.

```php
$is_admin = true;     // Tipo bool
$age = 25;            // Tipo int
$price = 10.99;       // Tipo float
$name = "John";       // Tipo string
```

Tipos Compuestos

- **array**: Representa una colección de valores, ya sea indexada o asociativa.
- **object**: Representa una instancia de una clase, con propiedades y métodos.

```php
$numbers = [1, 2, 3];          // Array indexado
$person = new Person();        // Objeto de la clase Person
```

Tipos Especiales

- **null**: Representa la ausencia de valor.
- **resource**: Contiene una referencia a un recurso externo, como un archivo o una conexión a una base de datos.

```php
$no_value = null;              // Valor null
$file_handle = fopen('file.txt', 'r'); // Recurso de archivo
```

Los tipos de datos en PHP son fundamentales para almacenar y manipular información en tus programas. Comprender estos tipos te permitirá trabajar eficientemente con variables y estructuras de datos en tu código.

Cadenas en PHP

En PHP, una cadena es una secuencia de caracteres. PHP te proporciona cuatro formas de definir una cadena literal: comillas simples, comillas dobles, sintaxis heredoc y sintaxis nowdoc.

Comillas simples y Comillas dobles

Puedes definir una cadena usando comillas simples o comillas dobles:

```php
$tituloConComillasSimples = 'PHP es genial';
$tituloConComillasDobles = "PHP es genial";
```

En las cadenas con comillas dobles, puedes incluir variables y caracteres especiales como \n, \r y \t sin escapar.

Acceso a caracteres en una cadena

Puedes acceder a un carácter específico en una cadena usando un índice basado en cero o dentro a un bucle:

```php
$cadena = 'Hola Mundo';
echo $cadena[0]; // 'H'
```

Obtener la longitud de una cadena

Para obtener la longitud de una cadena, puedes usar la función strlen():

```php
$cadena = 'Hola Mundo';
echo strlen($cadena); // 10
```

Cadenas heredoc y nowdoc

PHP ofrece dos formas de definir cadenas multilínea sin escapar caracteres especiales ni expandir variables: heredoc y nowdoc.

Heredoc

Las cadenas heredoc te permiten definir cadenas multilínea sin tener que escapar comillas y expandir variables. La sintaxis es la siguiente:

```php
$cadenaHeredoc = <<<ETIQUETA
Texto multilínea aquí
Puedes incluir variables: $variable
ETIQUETA;
```

Nowdoc

Las cadenas nowdoc son similares a las heredoc, pero no expanden variables. La sintaxis es la siguiente:

```php
$cadenaNowdoc = <<<'ETIQUETA'
Texto multilínea aquí
No se expanden variables: $variable
ETIQUETA;
```