Guia PHP

PRINCIPIANTES

JORGE MOYA DELGADO

Indice

1. Introducción a PHP

1.1 ¿Qué es php?

PHP es un lenguaje de programación de código abierto utilizado principalmente para desarrollar aplicaciones web y páginas dinámicas.

Creado por Rasmus Lerdorf en 1994 como un conjunto de scripts para trabajar con formularios web, pero que ha evolucionado hasta convertirse en un completo y potente lenguaje de programación.

PHP es interpretado, lo que significa que el código fuente escrito por el desarrollador se compila en tiempo de ejecución. Esto hace que las páginas PHP sean muy dinámicas y flexibles, lo que a su vez las hace ideales para crear aplicaciones web interactivas.

1.2 ¿Para qué se usa PHP?

PHP se utiliza principalmente para el desarrollo de aplicaciones web dinámicas. Esto incluye todo, desde sitios web simples hasta aplicaciones web más complejas, como sistemas de administración de contenido (CMS), tiendas en línea, foros y redes sociales.

PHP se integra a la perfección con otras tecnologías web como HTML, CSS, JavaScript y bases de datos, lo que permite a los desarrolladores crear aplicaciones web interactivas y personalizadas para sus usuarios. Además, PHP es compatible con la mayoría de los servidores web, lo que lo hace muy popular entre los desarrolladores y las empresas que requieren aplicaciones web dinámicas y escalables.

1.3 Ventajas en el uso de PHP

- **Fácil de aprender**: PHP es un lenguaje de programación relativamente fácil de aprender para desarrolladores con experiencia en otros lenguajes de programación.
- **Flexibilidad**: Se puede utilizar en diversas aplicaciones web. Además, se puede integrar con otras tecnologías web como HTML, CSS, JavaScript y bases de datos, lo que permite a los desarrolladores crear aplicaciones web personalizadas y dinámicas.
- **Eficiencia**: PHP se ejecuta en el lado del servidor, lo que significa que el procesamiento se realiza en el lado del servidor, no en el navegador del usuario. Esto hace que las aplicaciones web sean más rápidas y eficientes.
- **Comunidad activa**: PHP tiene una gran comunidad de desarrolladores y usuarios que brindan soporte, herramientas y recursos para desarrollar aplicaciones web.
- **Coste**: PHP es gratuito y de código abierto, lo que lo convierte en una opción atractiva para empresas y desarrolladores que buscan una solución de desarrollo de aplicaciones web de bajo costo.

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hola Mundo";
    return 0;
}
```

Hola mundo en C++.

```php
<?php
echo "Hola Mundo";
?>
```

Hola mundo en PHP.

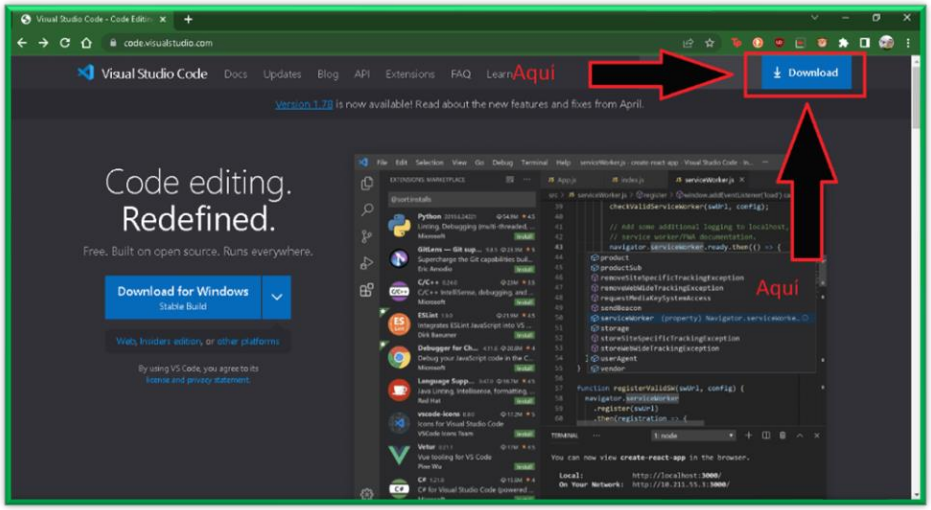
2. Configuración del entorno de desarrollo

2.1 Instalación de XAMPP

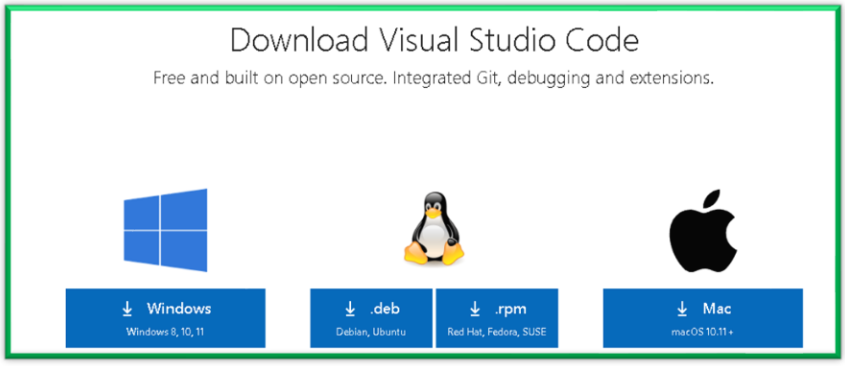
Para la instalación de XAMPP podemos seguir los primeros minutos del siguiente vídeo: clic para ir al video.

2.2 Instalación de Visual Studio Code

Si nos dirigimos a la página de Visual Studio Code veremos que aparece la sección de descargas:



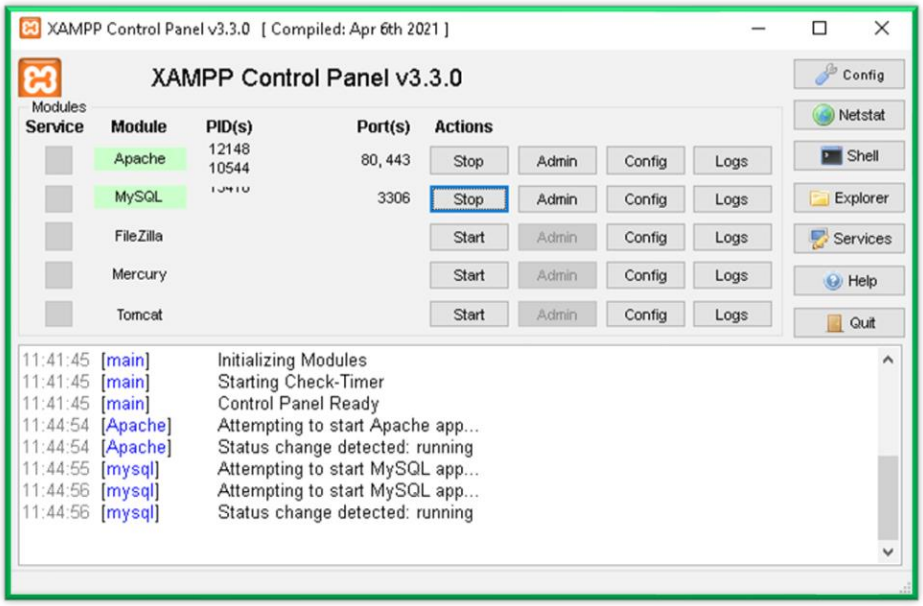
Al clicar nos aparecerá algo como esto:



Seleccionamos nuestro sistema operativo y ejecutamos el fichero que descarguemos, la instalación, como cualquier otra, dándole **si a todo** como si supiéramos lo que estamos haciendo.

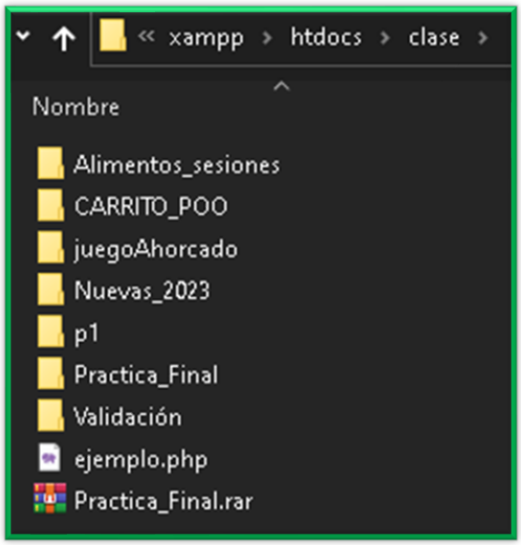
2.3 Configuración de XAMPP

En esta parte mejor no toquetear mucho las cosas, si hemos seguido al chico del vídeo en la instalación, nuestro XAMPP debería verse así:



La ruta donde debemos crear nuestras webs dinámicas es: **C:\xampp\htdocs**

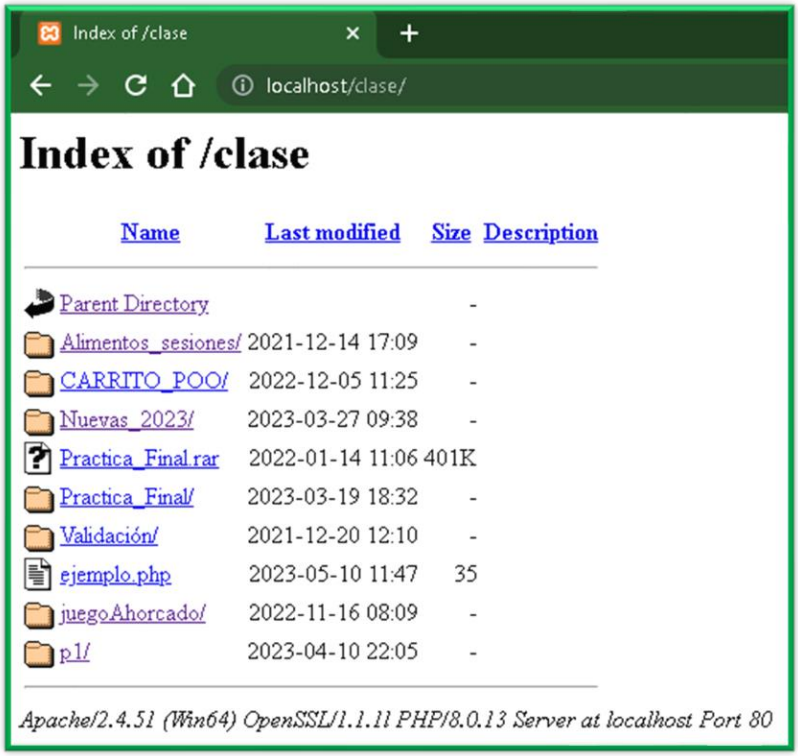
Una vez en ésta carpeta, podemos crear directamente nuestro Script o crear un directorio para cada web (lo mas recomendado si no queremos terminar con 231237 ficheros así: ejer1.php, ejer1Final.php, ejer1Final_finalisimo.php, ejer1-copia.php)



Como podemos observar, casi ni usando el truco de un directorio para cada web podemos evitar un pequeño caos, pero mejor eso que nada.

Ahora ya tenemos nuestro fichero “ejemplo.php” localizado en un ruta accesible para nuestro Apache, XAMPP, PHP.

Para comprobarlo, simplemente abrimos nuestro Visual Studio Code y creamos el pequeño HolaMundo.php del ejemplo anterior. Ejecutamos el Script y nuestro navegador debería mostrarlo:



Muy importante la ruta del navegador para acceder a ésta vista de los ficheros y directorios de nuestro C:\xampp\htdocs\clase

Ahora simplemente hacemos clic en el fichero del ejemplo y debería mostrarse:

Cómo vemos, el maestro debe albergar siempre una lección más que el alumno.

3. Variables y tipos de datos

3.1 Variables en PHP

Como ya hemos visto, bueno no se ve, pero está en el código. Hay diferentes variables en PHP ¿QUÉ ES UNA VARIABLE? Pues es una manera que tenemos de almacenar datos/info/cosas. Por ejemplo, imagina que tienes 12348 usuarios para una web de compra-venta de peonzas, cada uno de estos usuarios tendrá una edad, nombre, id (importante para las BD) una dirección, la cual nunca deberíamos de usar para fines ilícitos, esas cosillas.

Pues PHP nos permite guardar esos datos en variables, hay diferentes tipos y a continuación vamos a ver algunos de ellos, más adelante se profundiza en algunos de estos tipos de datos/variables.

3.2 Tipos de datos en PHP

Como toca, primero viene la parte teórica, estos son los tipos de datos más comunes que veremos en PHP:

- **Entero (int)**: es un número entero sin decimales, por ejemplo, 10, -5, 0, etc.
- **Flotante (float)**: es un número con decimales, por ejemplo, 3.14, -0.5, etc.
- **Cadena de caracteres (string)**: es un conjunto de caracteres alfanuméricos y especiales encerrados entre comillas simples o dobles, por ejemplo, "Hola Mundo", '123', etc.
- **Booleano (bool)**: es un tipo de dato que solo puede tener dos valores: verdadero (true) o falso (false).
- **Arreglo (array)**: es una colección ordenada de elementos que pueden ser de diferentes tipos de datos, por ejemplo, un arreglo que contiene números, cadenas de caracteres, booleanos, etc.
- **Objeto (object)**: es una instancia de una clase que contiene propiedades y métodos que pueden ser utilizados para manipular la información.
- **Nulo (null)**: es un valor especial que indica que una variable no tiene un valor asignado.

Bueno, lo normal si vienes con 0 conocimientos en lenguajes de programación, es que los booleanos, los array, los object y los null, te suenen a chino PERO hay un truquillo y es: que aunque tengas conocimientos, sabes que te va a tocar chapar un par de días hasta comprenderlos, como a todo KISKI.

Ahora vamos a mostrar en código un ejemplo de cada uno de estos datos para que podáis experimentar y ver qué hace cada uno. O en cristiano, copiar y pegar las cosas en vuestro Visual Studio Code (si, durante todo el documento pienso escribirlo siempre entero, queda flama).

3.3 Conversión de tipos de datos

A veces, vamos a necesitar convertir ciertos datos a otro tipo para realizar operaciones o comparaciones dentro de nuestros Scripts, en PHP tenemos diferentes métodos para llevar a cabo estas conversiones:

- **Conversiones implícitas**: PHP realiza automáticamente conversiones implícitas de tipos de datos en ciertas situaciones. Por ejemplo, si se agrega un número entero a un flotante, PHP convertirá automáticamente el entero en un flotante antes de realizar la operación.
- **Conversiones explícitas**: puede convertir explícitamente un tipo de datos a otro utilizando los siguientes métodos:
  - ***(int) o intval()***: convierte el valor en un número entero.
  - ***(flotante) o floatval()***: Convierte el valor en un flotante.
  - ***(cadena) o strval()***: convierte el valor en una cadena.
  - ***(bool) o boolval()***: Convierte el valor a un valor booleano.
  - ***(matriz) o settype()***: convierte el valor en una matriz.
  - ***(objeto)***: Convierte el valor en un objeto.

Es importante tener en cuenta que las conversiones de tipos de datos pueden provocar la pérdida de precisión o información, por lo que es importante probar y verificar que los resultados sean los esperados.

4. Operadores

4.1 Operadores aritméticos

4.2 Operadores de comparación

4.3 Operadores lógicos

5. Estructuras de control de flujo

5.1 Estructuras condicionales (if, else; elseif…)

5.2 Estructuras de repetición (While, do-while, foreach)

5.3 Interrupción de estructuras de repetición (break, continue)