**Instituto Nacional de Aprendizaje**

**Curso: Programador de aplicaciones informáticas**

**Módulo: Programación de Aplicaciones Empresariales**

**Tema: Investigación Nuevas Técnologías**

**Profesor: Luis Alonso Bogantes Rodríguez**

**Alumno: Claudio Mauricio Jiménez Castro**

**2023**

Contenido

[Introducción 1](#_Toc148950540)

[¿Qué es PHP? 2](#_Toc148950541)

[Instalación de PHP en Diferentes Sistemas Operativos 4](#_Toc148950542)

[Uso de XAMPP como Entorno de Desarrollo para PHP 6](#_Toc148950543)

[**Sintaxis de PHP:** 8](#_Toc148950544)

[Palabras Reservadas PHP 17](#_Toc148950545)

[Introducción a Symfony: 30](#_Toc148950546)

[Referencias: 36](#_Toc148950547)

# Introducción

El propósito fundamental de este trabajo es introducir a los compañeros y compañeras en dos temas de relevancia dentro del desarrollo web, para ello centrándonos en dos tecnologías fundamentales: PHP y Symfony. A medida que la la programación avanza e incorpora nuevas herramientas y lenguajes, la creación de aplicaciones web se ha convertido en un componente esencial en nuestro mundo moderno. PHP y Symfony son dos herramientas muy útiles que permiten a los desarrolladores web crear aplicaciones web dinámicas y eficientes, y su comprensión es importante para cualquiera que aspire a dedicarse como desarrollador(a) de software.

PHP, acrónimo de "Hypertext Preprocessor", es un lenguaje de programación ampliamente utilizado para el desarrollo web. Se destaca por su facilidad de uso y versatilidad, y es la columna vertebral de millones de sitios web en todo el mundo. A lo largo de este trabajo, exploraremos los conceptos generales y fundamentales de PHP, desde la sintaxis básica hasta ejemplos un tanto más completo pero simples.

Symfony, por otro lado, es un framework de desarrollo web escrito en PHP que se basa en los principios de la programación orientada a objetos. Symfony ofrece un marco estructurado para el desarrollo web, facilitando la creación de aplicaciones web más grandes y complejas. Al comprender Symfony, los desarrolladores pueden acelerar significativamente el proceso de desarrollo y mantener la calidad y la eficiencia de sus proyectos.

En este documento, abordaremos estos dos temas, brindando una visión general de PHP y Symfony, sus aplicaciones y ventajas, y cómo se complementan entre sí para impulsar el desarrollo web moderno. Además, proporcionaremos ejemplos y casos prácticos que ilustran cómo estas tecnologías se aplican en la creación de aplicaciones web del mundo real.

# ¿Qué es PHP?

PHP (acrónimo recursivo de "PHP: Hypertext Preprocessor") es un lenguaje de programación ampliamente utilizado para el desarrollo de aplicaciones web. Fue creado originalmente por Rasmus Lerdorf en 1994 y, desde entonces, ha evolucionado para convertirse en una de las herramientas más populares en el desarrollo web.

**Características Clave:**

**Lenguaje del Lado del Servidor:** PHP es un lenguaje de servidor, lo que significa que se ejecuta en el servidor web antes de que se envíe una página al navegador del usuario. Esto permite la generación de contenido dinámico y personalizado en tiempo real.

**Sintaxis Amigable:** La sintaxis de PHP es similar a la de otros lenguajes de programación, lo que facilita su aprendizaje para aquellos que ya tienen experiencia en la programación.

**Comunidad Activa:** PHP tiene una comunidad de desarrolladores muy activa que contribuyen constantemente a su desarrollo y creación de bibliotecas y marcos de trabajo (frameworks).

**Integración con Bases de Datos:** PHP se integra fácilmente con una amplia variedad de sistemas de gestión de bases de datos (DBMS) como MySQL, PostgreSQL, Oracle, y más. Esto permite a las aplicaciones web almacenar y recuperar datos de manera eficiente.

**Código Abierto:** PHP es un software de código abierto, lo que significa que es gratuito y está disponible para su modificación por parte de la comunidad de desarrolladores.

**¿Por qué PHP es Importante?**

PHP es una de las bases de la web dinámica moderna. Se utiliza en aplicaciones web de todos los tamaños, desde blogs personales hasta plataformas de comercio electrónico a gran escala. Además, PHP se complementa bien con otros lenguajes y tecnologías, lo que lo hace una opción versátil para los desarrolladores. Podemos además aseverar que php es importante por los siguientes puntos:

**Amplia Disponibilidad de Hosting:** La gran mayoría de proveedores de hosting web ofrecen soporte para PHP. Esto facilita el despliegue de aplicaciones PHP en servidores web compartidos o dedicados.

**Extensa Biblioteca de Funciones:** PHP proporciona una amplia gama de funciones integradas, desde manipulación de archivos hasta operaciones matemáticas y procesamiento de texto. Esto agiliza el desarrollo y reduce la necesidad de escribir código desde cero.

**Versatilidad:** PHP se utiliza tanto en aplicaciones pequeñas y sencillas como en proyectos empresariales a gran escala. Es un lenguaje versátil que puede adaptarse a una variedad de necesidades.

**Soporte para Bases de Datos:** La integración sencilla con sistemas de gestión de bases de datos (DBMS) permite a las aplicaciones PHP almacenar y recuperar datos de manera efectiva. La combinación de PHP con MySQL es particularmente popular.

**Comercio Electrónico:** Muchas de las plataformas de comercio electrónico más conocidas, como WooCommerce, PrestaShop y Magento, se basan en PHP. Esto facilita la creación de tiendas en línea y portales de comercio electrónico.

**Desarrollo Rápido:** PHP es conocido por su velocidad en el desarrollo de aplicaciones web. Los desarrolladores pueden implementar características rápidamente y realizar actualizaciones eficientes.

**Interacción con Tecnologías Web Modernas:** PHP puede trabajar en conjunto con otras tecnologías web como HTML, CSS, JavaScript y bibliotecas populares como jQuery, lo que permite crear aplicaciones web modernas y visualmente agradables.

PHP no solo es un lenguaje de programación popular y de código abierto, sino que es fundamental en la construcción de la web tal como la conocemos. Su versatilidad, amplia comunidad y disponibilidad en una variedad de servidores lo hacen esencial para el desarrollo web y la creación de aplicaciones interactivas y dinámicas.

# Instalación de PHP en Diferentes Sistemas Operativos

Para comenzar con el desarrollo en PHP, es crucial instalarlo adecuadamente en su sistema operativo. PHP es un lenguaje de programación ampliamente utilizado en la creación de aplicaciones web dinámicas. El punto bueno es que PHP es altamente compatible con una variedad de sistemas operativos, lo que permite a los desarrolladores trabajar en la plataforma de trabajo a la que este acostumbrado(a). A continuación, describiremos brevemente los pasos básicos para instalar PHP en sistemas Windows, MacOS y Linux.

**Instalación en Windows:**

XAMPP: Una de las formas más sencillas de configurar PHP en sistemas Windows es mediante el uso de XAMPP (Cross-Platform Apache, MySQL, PHP, and Perl). XAMPP es un paquete que incluye el servidor web Apache, PHP, MySQL y Perl. Los pasos para instalar PHP en Windows con XAMPP suelen ser los siguientes:

Descargar XAMPP desde el sitio web oficial.

Ejecutar el instalador y seguir las instrucciones.

Iniciar el servidor Apache y MySQL desde la interfaz de XAMPP.

Colocar tus archivos PHP en la carpeta "htdocs" y acceder a ellos a través de un navegador web.

Nota: Para los ejemplos en clase se trabajará con XAMPP

**Instalación en MacOS:**

**Homebrew:** En sistemas MacOS, puede utilizar Homebrew, un gestor de paquetes que facilita la instalación de PHP.

Abre la terminal y ejecuta el comando para instalar Homebrew.

Después, instala PHP con el comando "brew install php".

**Instalación en Linux:**

Administrador de Paquetes: La mayoría de las distribuciones de Linux ofrecen PHP a través de su administrador de paquetes. Los comandos para instalar PHP pueden variar según la distribución, pero en Debian/Ubuntu, puedes usar el siguiente comando:

**sudo apt-get install php**

**Herramientas Adicionales:**

Además de PHP en sí, es beneficioso instalar herramientas adicionales para el desarrollo web, como Apache, MySQL o MariaDB. XAMPP, WAMP y MAMP son paquetes que incluyen estas herramientas, lo que facilita la creación de entornos de desarrollo web locales completos en diferentes sistemas operativos.

**Verificación de la Instalación:**

Después de la instalación, es importante verificar que PHP esté funcionando correctamente. Puede hacerlo creando un archivo PHP simple con **<?php phpinfo(); ?>** y ejecutándolo en tu servidor local. Esto mostrará información detallada sobre la configuración de PHP.

La instalación de PHP en diferentes sistemas operativos es esencial para comenzar con el desarrollo web. Los pasos varían según la plataforma, pero siguiendo estos pasos básicos que he descrito, estará listo para escribir y probar aplicaciones PHP en su entorno elegido.

Es importante recordar que, además de la instalación, debe mantener su entorno actualizado con las últimas versiones de PHP y las herramientas relacionadas para garantizar un desarrollo suave y seguro

# Uso de XAMPP como Entorno de Desarrollo para PHP

Como bien se comentó líneas arriba, se utilizará XAMPP (Cross-Platform Apache, MySQL, PHP, and Perl) que es una solución integral que combina varios componentes esenciales para el desarrollo web en un solo paquete. Permite la creación de un entorno de desarrollo local completo que incluye un servidor web Apache, una base de datos MySQL, el intérprete PHP y Perl. XAMPP es ampliamente utilizado por desarrolladores de PHP en todo el mundo debido a su facilidad de instalación y configuración.

**Ventajas de Utilizar XAMPP:**

**Facilidad de Instalación:** XAMPP es conocido por su proceso de instalación sencillo. Los desarrolladores pueden descargar la versión específica de XAMPP para su sistema operativo y seguir un asistente de instalación que configura automáticamente los componentes necesarios.

**Plataformas Múltiples:** XAMPP es compatible con varios sistemas operativos, incluyendo Windows, MacOS y Linux. Esto significa que los desarrolladores pueden utilizar el mismo entorno de desarrollo en diferentes plataformas, lo que facilita la colaboración en proyectos web.

**Entorno Aislado:** XAMPP crea un servidor web local que no está conectado a Internet. Esto proporciona un entorno aislado en el que los desarrolladores pueden probar y depurar aplicaciones sin preocuparse por afectar a servidores web en producción.

**Gestión de Bases de Datos:** XAMPP incluye una instalación de MySQL o MariaDB, lo que permite a los desarrolladores crear bases de datos locales para sus aplicaciones. Pueden utilizar phpMyAdmin, una herramienta de administración de bases de datos web, para gestionar bases de datos de manera eficiente.

**Intérprete PHP y Perl:** XAMPP proporciona intérpretes de PHP y Perl para ejecutar scripts en estos lenguajes. Esto es esencial para el desarrollo web, ya que PHP es uno de los lenguajes más utilizados en la creación de sitios web dinámicos.

**Compatibilidad con Aplicaciones Web:** XAMPP es compatible con muchas aplicaciones web populares y sistemas de gestión de contenidos (CMS) como WordPress, Joomla y Drupal. Los desarrolladores pueden utilizar XAMPP como plataforma de pruebas para estas aplicaciones.

**Uso de XAMPP en el Desarrollo Web:**

Una vez instalado XAMPP, los desarrolladores pueden crear y alojar sus proyectos web en el directorio "htdocs" o "www" dentro de la carpeta de instalación. Pueden acceder a sus sitios web escribiendo "<http://localhost/nombre-de-la-carpeta>" en un navegador web. Esto permite a los desarrolladores ver cómo se verá y funcionará su sitio en un servidor web real antes de la implementación en línea.

XAMPP es una herramienta esencial para los desarrolladores de PHP, ya que proporciona un entorno de desarrollo local completo y aislado. Facilita la creación y prueba de aplicaciones web antes de implementarlas en servidores de producción. A través de su combinación de Apache, MySQL, PHP y Perl, XAMPP simplifica el proceso de desarrollo web y acelera la creación de aplicaciones dinámicas y sitios web.

XAMPP se descargará del siguiente en enlace: <https://sourceforge.net/projects/xampp/>

Cabe mencionar también que para trabajar con PHP se utilizará un entorno ya conocido por todos(a) en este curso, hablamos de Visual Studio Code. Para poder trabajar en este entorno de desarrollo nos apoyaremos precisamente con XAAMPP, en cual se descargará del enlace de arriba, para posteriormente seguir las indicaciones para no solo instalarlo si no configurarlo para que sea detectado por Visual Studio e instalar ciertos complementos que será totalmente necesarios para poder trabajar con PHP.

Antes de ahondar en la instalación de XAAMPP repasemos algunas palabras reservadas propiamente del lenguaje PHP Y que para efectos de lograr una mejor comprensión es buena idea comprender C:\xampp\php

# **Sintaxis de PHP:**

**Etiquetas PHP (<?php ... ?>):** En PHP, las etiquetas <?php y ?> se utilizan para encerrar bloques de código PHP. Cualquier código dentro de estas etiquetas se interpreta como código PHP. Por ejemplo:

<?php

echo "¡Hola, mundo!";

?>

Comentarios en PHP: Los comentarios en PHP se utilizan para documentar el código y no se ejecutan. Hay dos formas de agregar comentarios:

Comentario de una sola línea:

// Este es un comentario de una sola línea

Comentario de múltiples líneas:

/\*

Este es un comentario de

múltiples líneas en PHP.

\*/

Variables y tipos de datos: PHP es un lenguaje débilmente tipado, lo que significa que no necesitas declarar el tipo de una variable antes de usarla. Algunos tipos de datos comunes en PHP incluyen:

Enteros (int): Números enteros, por ejemplo, $edad = 30;.

Flotantes (float o double): Números con decimales, por ejemplo, $precio = 19.99;.

Cadenas (string): Secuencias de caracteres, por ejemplo, $nombre = "Juan";.

Booleanos (bool): Valores verdadero o falso, por ejemplo, $activo = true;.

Arrays: Estructuras que pueden contener múltiples valores, por ejemplo, $colores = array("rojo", "verde", "azul");.

Null: Representa la ausencia de valor, por ejemplo, $nada = null;.

Operadores en PHP: PHP incluye una variedad de operadores para realizar cálculos y comparaciones. Algunos operadores comunes son:

Aritméticos: + (suma), - (resta), \* (multiplicación), / (división).

Comparación: == (igual), != (no igual), > (mayor que), < (menor que).

Operadores lógicos: && (y lógico), || (o lógico), ! (negación).

Concatenación: . (para unir cadenas de texto), por ejemplo, $nombreCompleto = $nombre . " " . $apellido;.

**Sentencias condicionales (if, else, switch):** Las sentencias condicionales permiten ejecutar diferentes bloques de código según se cumplan ciertas condiciones. Estas son algunas de las sentencias condicionales más comunes en PHP:

**if:** Permite ejecutar un bloque de código si se cumple una condición. Ejemplo:

if ($edad >= 18) {

echo "Eres mayor de edad";

}

**else:** Se utiliza junto con if para ejecutar un bloque de código si la condición no se cumple. Ejemplo:

if ($edad >= 18) {

echo "Eres mayor de edad";

} else {

echo "Eres menor de edad";

}

**switch:** Permite elegir diferentes bloques de código según el valor de una variable. Ejemplo:

$dia = "lunes";

switch ($dia) {

case "lunes":

echo "Hoy es lunes";

break;

case "martes":

echo "Hoy es martes";

break;

default:

echo "Hoy es otro día";

}

**Bucles (for, while, foreach):** Los bucles son fundamentales para repetir tareas en PHP. Estas son algunas de las estructuras de bucles más utilizadas:

**for:** Se utiliza para ejecutar un bloque de código un número específico de veces. Ejemplo:

for ($i = 1; $i <= 5; $i++) {

echo "Iteración número: " . $i . "<br>";

}

**while:** Permite ejecutar un bloque de código mientras se cumpla una condición. Ejemplo:

$contador = 0;

while ($contador < 3) {

echo "Iteración número: " . $contador . "<br>";

$contador++;

}

**foreach:** Se utiliza para recorrer elementos de un array o colección. Ejemplo:

$colores = array("rojo", "verde", "azul");

foreach ($colores as $color) {

echo "Color: " . $color . "<br>";

}

**Uso de operadores ternarios:** Los operadores ternarios permiten simplificar las sentencias condicionales en una sola línea. La sintaxis básica es condición ? valor\_si\_verdadero : valor\_si\_falso. Ejemplo:

$edad = 20;

$mensaje = ($edad >= 18) ? "Eres mayor de edad" : "Eres menor de edad";

echo $mensaje;

**Creación de funciones:** Las funciones son bloques de código reutilizable que realizan una tarea específica. Para crear una función en PHP, se utiliza la siguiente sintaxis:

function nombre\_de\_la\_funcion() {

// Código de la función

}

Por ejemplo, para crear una función que imprima "¡Hola, Mundo!", puede hacerlo de la siguiente manera:

function saludar() {

echo "¡Hola, Mundo!";

}

**Paso de parámetros:** Las funciones pueden aceptar parámetros (valores de entrada) que les permiten ser más flexibles y personalizables. Los parámetros se definen entre los paréntesis de la función. Aquí hay un ejemplo de una función que recibe dos parámetros:

function suma($numero1, $numero2) {

$resultado = $numero1 + $numero2;

return $resultado;

}

Para llamar a esta función y pasarle valores

$resultado = suma(5, 3);

echo "El resultado es: " . $resultado;

**Retorno de valores:** Las funciones pueden devolver un valor mediante la instrucción return. Esto permite que una función realice un cálculo y devuelva el resultado. En el ejemplo anterior, la función suma devuelve el resultado de la suma

function suma($numero1, $numero2) {

$resultado = $numero1 + $numero2;

return $resultado;

}

$resultado = suma(5, 3);

echo "El resultado es: " . $resultado;

**Uso de arrays unidimensionales y multidimensionales:** En PHP, un array es una estructura de datos que almacena una colección de elementos. Los arrays unidimensionales almacenan elementos en una sola dimensión, mientras que los arrays multidimensionales pueden tener múltiples dimensiones

**Array Unidimensional:**

$colores = array("rojo", "verde", "azul");

**Array Multidimensional:**

$personas = array(

array("Juan", 25),

array("María", 30),

array("Pedro", 28)

);

**Funciones relacionadas con arrays:** PHP ofrece una variedad de funciones incorporadas para trabajar con arrays, lo que facilita la manipulación de datos. Algunas de las funciones comunes son:

count($array): Devuelve la cantidad de elementos en un array.

array\_push($array, $element): Agrega un elemento al final de un array.

array\_pop($array): Elimina y devuelve el último elemento de un array.

array\_merge($array1, $array2): Combina dos o más arrays en uno solo.

sort($array): Ordena los elementos de un array en orden ascendente.

array\_key\_exists($key, $array): Verifica si una clave existe en un array asociativo.

Ejemplo de uso de algunas de estas funciones:

$edades = array(25, 30, 28);

echo "Cantidad de edades: " . count($edades);

array\_push($edades, 35);

echo "Nueva cantidad de edades: " . count($edades);

$ultimo = array\_pop($edades);

echo "Edad eliminada: " . $ultimo;

sort($edades);

echo "Edades ordenadas: ";

print\_r($edades);

**Recopilación de datos de formularios HTML:** Los formularios HTML permiten a los usuarios ingresar información y enviarla al servidor para su procesamiento. PHP es ampliamente utilizado para recopilar y procesar estos datos.

Para recopilar datos de formularios HTML en PHP, puedes utilizar la variable superglobal $\_POST o $\_GET, dependiendo del método utilizado en el formulario (POST o GET).

Ejemplo de formulario HTML:

<form method="post" action="procesar.php">

Nombre: <input type="text" name="nombre">

Email: <input type="text" name="email">

<input type="submit" value="Enviar">

</form>

En PHP (procesar.php), puedes acceder a los datos así:

$nombre = $\_POST['nombre'];

$email = $\_POST['email'];

**Validación de datos de entrada:** Es esencial validar los datos del formulario para garantizar que sean seguros y cumplan con ciertos criterios. Puedes usar funciones como filter\_var y expresiones regulares para validar datos, como direcciones de correo electrónico o números de teléfono

$email = $\_POST['email'];

if (filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

// Email válido

} else {

// Email no válido

}

Conexión a SQL mediante PHP: La siguiente solo es un ejemplo de la estructura básica para realizar la conexión

$serverName = "nombre\_servidor";

$connectionOptions = array(

"Database" => "nombre\_base\_de\_datos",

"Uid" => "nombre\_usuario",

"PWD" => "contrasena"

);

$conn = sqlsrv\_connect($serverName, $connectionOptions);

if (!$conn) {

die("La conexión a la base de datos ha fallado: " . print\_r(sqlsrv\_errors(), true));

}

Clases y Objetos en PHP:

Las clases son plantillas que definen cómo se crea un objeto. Puedes pensar en una clase como un plano o un modelo para construir objetos.

Los objetos son instancias de una clase. Cada objeto tiene su propia copia de las propiedades y métodos definidos en la clase.

Propiedades y Métodos de Clases:

Propiedades: Son variables asociadas a un objeto. Estas variables almacenan datos relacionados con el objeto.

Métodos: Son funciones asociadas a un objeto. Estos métodos pueden realizar operaciones en los datos almacenados en las propiedades del objeto.

class Persona {

// Propiedades

public $nombre;

public $edad;

// Método constructor

public function \_\_construct($nombre, $edad) {

$this->nombre = $nombre;

$this->edad = $edad;

}

// Método

public function saludar() {

echo "Hola, soy {$this->nombre} y tengo {$this->edad} años.";

}

}

// Crear un objeto Persona

$persona1 = new Persona("Juan", 30);

// Acceder a las propiedades y métodos

echo $persona1->nombre; // Imprimirá "Juan"

$persona1->saludar(); // Imprimirá "Hola, soy Juan y tengo 30 años."

# Palabras Reservadas PHP

**Die():** Este constructor de lenguaje es equivalente a [exit()](https://www.php.net/manual/es/function.exit.php).

**\_\_halt\_compiler:** Detiene la ejecución del compilador

Break: break finaliza la ejecución de la estructura for, foreach, while, do-while o switch en curso.

Ej: <?php  
$arr = array('uno', 'dos', 'tres', 'cuatro', 'pare', 'cinco');  
while (list(, $val) = each($arr)) {  
if ($val == 'pare') {  
break; /\* Se puede también escribir 'break 1;' aquí. \*/  
}  
echo "$val<br />\n";  
}

Clone: Para crear una copia de un objeto se utiliza la palabra clave clone (que invoca, si fuera posible, al método \_\_clone() del objeto)

**Empty:**  Determina si una variable está vacía

**end:** Estructura alternativa en algunas estructuras de control en las llaves de cierre, se emplean algunas como:  endif;, endwhile;, endfor;, endforeach;, o endswitch. Ej:

<?php if ($a == 5): ?>  
A es igual a 5  
<?php endif; ?

PHP también tiene una función llamada end() que se utiliza para mover el puntero interno de un array hasta su último elemento.

$frutas = array("manzana", "plátano", "naranja");

echo end($frutas); // Esto mostrará "naranja"

**final:** impide que las clases hijas sobrescriban un método, antecediendo su definición con final. Si la propia clase se define como final, entonces no se podrá heredar de ella. Ej:

<?php  
class BaseClass {  
public function test() {  
echo "llamada a BaseClass::test()\n";  
}  
  
final public function moreTesting() {  
echo "llamada a BaseClass::moreTesting()\n";  
}  
}  
  
class ChildClass extends BaseClass {  
public function moreTesting() {  
echo "llamada a ChildClass::moreTesting()\n";  
}  
}  
// Devuelve un error Fatal: Cannot override final method BaseClass::moreTesting()  
?>

**include\_once:** Tiene un comportamiento similar al de la sentencia [include](https://www.php.net/manual/es/function.include.php), siendo la única diferencia de que si el código del fichero ya ha sido incluido, no se volverá a incluir, e include\_once devolverá true. Como su nombre indica, el fichero será incluido solamente una vez.

**Include:**  permite incluir el contenido de un archivo externo dentro de otro archivo. Ejemplo, se tiene un archivo llamado “saludo.php”, que contiene lo siguiente:

<?php

echo "Hola, bienvenido a mi sitio web.";

?>

Luego en otro archivo llamado “principal.php” que contiene el siguiente código:

<html>

<head>

<title>Mi sitio web</title>

</head>

<body>

<?php

include 'saludo.php'; //Acá incluimos el archivo externo junto con el código descrito anteriormente y se ejecutará justo donde es llamado

?>

<p>Este es el contenido de mi página principal.</p>

</body>

</html

**Isset:** Determina si una variable está definida y no es null

**Require:** require es idéntico a [include](https://www.php.net/manual/es/function.include.php) excepto que en caso de fallo producirá un error fatal de nivel E\_COMPILE\_ERROR. En otras palabras, éste detiene el script mientras que [include](https://www.php.net/manual/es/function.include.php) sólo emitirá una advertencia (E\_WARNING) lo cual permite continuar el script.

**Llamadas de retorno (Callbacks / Callables**): permite revisar que los contenidos de una variable contengan el nombre de una función válida, o que un array contenga un objeto adecuadamente codificado y un nombre de función

//El objetivo principar de este proceso de de los callbacks es mayor nivel de reutilización y flexibilidad en el código.

<?php

// Definimos una función que toma una llamada de retorno como argumento

function procesarDatos($datos, $callback) {

// Verificamos si la llamada de retorno es válida

if (is\_callable($callback)) {

// Si es válida, llamamos a la función de llamada de retorno

$resultado = call\_user\_func($callback, $datos);

return $resultado;

} else {

return "Llamada de retorno no válida";

}

}

// Definimos una función de llamada de retorno

//Esta es la función de llamada de retorno que se aplicará a los datos que necesiten ser procesados

function calcularDoble($numero) {

return $numero \* 2;

}

// Datos de ejemplo

$miNumero = 5; //variable que contiene el dato que será procesado en la función “procesarDatos”

// Llamamos a procesarDatos con la función de llamada de retorno

$resultado = procesarDatos($miNumero, 'calcularDoble');

// Mostramos el resultado

echo "El resultado es: " . $resultado;

?>

**Rasgos (Traits):** En PHP son una forma de reutilizar código en las clases sin utilizar la herencia. Los rasgos le permiten incluir métodos y propiedades en una clase de manera compartida, lo que es útil cuando deseas compartir funcionalidades entre diferentes clases sin crear una jerarquía de herencia complicada. Los rasgos son una característica de PHP que promueve la reutilización del código y la composición en lugar de la herencia

trait Saludar {

public function saludar() {

echo '¡Hola! ';

}

}

class Persona {

use Saludar;

private $nombre;

public function \_\_construct($nombre) {

$this->nombre = $nombre;

}

public function presentarse() {

$this->saludar();

echo 'Soy ' . $this->nombre;

}

}

$persona = new Persona('Juan');

$persona->presentarse();

En este ejemplo, hemos definido un rasgo llamado Saludar, que contiene el método saludar. Luego, la clase Persona utiliza este rasgo con use Saludar y, por lo tanto, hereda el método saludar. Cuando creamos una instancia de Persona y llamamos al método presentarse, utiliza el método saludar del rasgo para imprimir "¡Hola!" antes de mostrar el nombre de la persona.

Los rasgos son útiles para compartir comportamientos comunes en diferentes clases, sin forzar una relación de herencia complicada. Esto hace que tu código sea más modular y fácil de mantener.

**Insteadof:** es una característica útil para resolver conflictos de nombres de métodos cuando se utilizan traits en PHP, permitiendo que la clase que los utiliza defina cuál de los métodos con el mismo nombre debe utilizarse.

trait TraitA {

public function sayHello() {

echo 'Hello from TraitA';

}

}

trait TraitB {

public function sayHello() {

echo 'Hello from TraitB';

}

}

class MyClass {

use TraitA, TraitB {

TraitA::sayHello insteadof TraitB;

}

}

$obj = new MyClass();

$obj->sayHello(); // Output: Hello from TraitA

**Echo:** Muestra una o más cadenas, y concatenarlas con el valor de variables para imprimirlas luego. Ej: echo "Hola mundo";

Eval: Evaluar una cadena como código de PHP

<?php  
$cadena = 'taza';  
$nombre = 'café';  
$str = 'Esto es una $cadena con mi $nombre en ella.';  
echo $str. "\n";  
eval("\$str = \"$str\";");  
echo $str. "\n";  
?>

**Funciones Flecha:** PHP 7.4 introdujo las funciones flecha, también conocidas como funciones arrow o funciones de flecha. Estas funciones ofrecen una forma más concisa de definir funciones anónimas y son especialmente útiles para devolver valores desde una expresión simple.

Las funciones flecha son útiles para operaciones simples y rápidas, como transformaciones de datos o filtrado de arreglos. Puede usarlas en lugar de definir funciones anónimas completas con function.

Ej:

$cuadrado = fn($numero) => $numero \* $numero;

echo $cuadrado(4); // Imprime 16

Las funciones flecha en PHP tienen una diferencia clave con respecto al ámbito de las variables externas en comparación con las funciones anónimas tradicionales. En una función flecha, las variables del ámbito declaradas anteriormente son accesibles directamente. Esto significa que las variables se heredan desde el contexto en el que se declaró la función flecha.

En contraste, en las funciones anónimas tradicionales, las variables externas deben importarse explícitamente utilizando la palabra clave use. Un ejemplo que muestra esta diferencia sería:

$valor = 10;

// Función anónima tradicional

$funcionAnonima = function ($parametro) use ($valor) {

return $parametro + $valor;

};

// Función flecha

$funcionFlecha = fn($parametro) => $parametro + $valor;

echo $funcionAnonima(5); // Imprime 15

echo $funcionFlecha(5); // Imprime 15

**goto:** El operador goto puede ser usado para saltar a otra sección en el programa. El punto de destino es especificado mediante una etiqueta seguida de dos puntos y la instrucción es dada como goto seguida de la etiqueta del destino deseado. Este goto no es completamente sin restricciones. La etiqueta de destino debe estar dentro del mismo fichero y contexto, lo que significa que no se puede saltar fuera de una función o método, ni se puede saltar dentro de uno. Tampoco se puede saltar dentro de cualquier clase de estructura de bucle o switch. Se puede saltar fuera de estos y un uso común es utilizar un goto en lugar de un break multi-nivel.

<?php  
goto a;  
echo 'Foo';  
  
a:  
echo 'Bar'; //va hasta acá e imprime  
?>

Ej 2:

<?php  
for($i=0,$j=50; $i<100; $i++) {  
while($j--) {  
if($j==17) goto end;  
}   
}  
echo "i = $i";  
end:  
echo 'j alcanzó 17'; //Se brinca el resto del código e imprime acá  
?>

**instanceof** se utiliza para determinar si una variable de PHP es un objeto instanciado de una cierta [clase](https://www.php.net/manual/es/language.oop5.basic.php#language.oop5.basic.class):

<?php  
class MyClass  
{  
}  
  
class NotMyClass  
{  
}  
$a = new MyClass;  
  
var\_dump($a instanceof MyClass);  
var\_dump($a instanceof NotMyClass);  
?>

El resultado del ejemplo sería:

bool(true)

bool(false)

**match:** es una estructura de control en PHP que permite realizar comparaciones precisas entre una expresión y varios valores posibles. Es similar a un interruptor switch, pero ofrece una sintaxis más concisa y expresiva. Puede pensar en match como una manera de buscar un valor específico y realizar acciones basadas en ese valor. Ej de estructura:

$resultado = match ($expresion) {

valor1 => "Acción 1",

valor2 => "Acción 2",

valor3 => "Acción 3",

default => "Acción por defecto",

};

Ej práctico:

$dia = "Lunes"

$mensaje = match ($dia) { //=> determina la acción

"Lunes", "Martes", "Miércoles", "Jueves", "Viernes" => "Días laborables",

"Sábado", "Domingo" => "Fin de semana",

default => "Día no válido",

};

echo $mensaje; // Imprimirá "Días laborables" porque $dia es "Lunes"

**array:** en PHP es una estructura de datos que permite almacenar múltiples valores en una sola variable. Los elementos de un array pueden ser de diferentes tipos, como números, cadenas de texto, objetos, otros arrays, etc. Para efectos de estudiar los arreglos en php, resaltaremos de momento dos tipos que llaman la atención particularmente la atención por su uso.

Array Asociativo:

$miArrayAsociativo = [

"nombre" => "Juan",

"edad" => 30,

"ciudad" => "Madrid"

];

echo $miArrayAsociativo["nombre"]; // Imprime "Juan"

echo $miArrayAsociativo["edad"]; // Imprime 30

echo $miArrayAsociativo["ciudad"]; // Imprime "Madrid"

Array en PHP que usa índices numéricos personalizados.

<?php  
$array = array(1, 1, 1, 1, 1, 8 => 1, 4 => 1, 19, 3 => 13);  
print\_r($array);  
?>

El resultado del ejemplo sería:

Array

(

[0] => 1

[1] => 1

[2] => 1

[3] => 13

[4] => 1

[8] => 1

[9] => 19

)

**unset** es una función en PHP que se utiliza para eliminar una variable o múltiples variables.

Básicamente, "destruye" la variable y la libera de la memoria, lo que significa que la variable ya no existe ni tiene ningún valor.

Estructura: unset($variable);

Se pueden eliminar múltiples variables: unset($variable1, $variable2, $variable3);

Ejemplo largo

$nombre = "Juan";

$edad = 30;

unset($nombre); // Elimina la variable $nombre

echo $nombre; // Esto generará un error ya que $nombre ya no existe

echo $edad; // Esto funcionará correctamente, $edad aún existe

# Introducción a Symfony:

Symfony es un framework PHP de código abierto ampliamente utilizado en el desarrollo de aplicaciones web. Está diseñado para ser un marco de trabajo versátil y eficiente que ayuda a los desarrolladores a construir aplicaciones web robustas, escalables y de alta calidad.

Symfony se ha convertido en una elección popular debido a su estructura modular y su arquitectura MVC (Model-View-Controller) bien definida. Esto facilita la organización y el mantenimiento de proyectos web de cualquier tamaño.

**Características clave de Symfony:**

**Arquitectura MVC:** Symfony sigue el patrón de diseño Modelo-Vista-Controlador, que separa claramente la lógica de negocio (Model), la presentación (View) y el control (Controller). Esta separación fomenta un código limpio y una fácil mantenibilidad.

**Componentes Reutilizables:** Symfony se basa en componentes reutilizables que pueden integrarse en otras aplicaciones PHP o utilizarse por separado. Esto acelera el desarrollo y promueve las mejores prácticas de programación.

**Gestión de Dependencias:** Symfony utiliza un contenedor de servicios para administrar las dependencias de la aplicación. Esto facilita la inyección de dependencias y la creación de código modular y mantenible.

**Historia de Symfony:**

Symfony fue creado por SensioLabs en 2005 y desde entonces ha evolucionado significativamente. La comunidad de Symfony ha contribuido a su crecimiento y mejora continua. El lanzamiento de Symfony 2 marcó un hito importante al introducir componentes independientes y permitir una mayor flexibilidad en el desarrollo de aplicaciones.

Symfony se basa en PHP, por lo que es una excelente opción para desarrolladores de PHP que desean crear aplicaciones web sólidas y escalables. A lo largo de los años, Symfony ha demostrado ser una herramienta valiosa para empresas y desarrolladores independientes en todo el mundo. Su comunidad activa y su compromiso con las prácticas de desarrollo de alta calidad continúan manteniendo a Symfony como un marco de referencia en el desarrollo de aplicaciones web en PHP.

**Beneficios de Symfony:**

**Eficiencia en el Desarrollo:** Symfony automatiza tareas comunes de desarrollo y proporciona herramientas para la creación rápida de aplicaciones.

**Flexibilidad:** Puedes usar Symfony para desarrollar aplicaciones de cualquier tipo, desde sitios web simples hasta aplicaciones empresariales complejas.

**Seguridad y Mantenimiento:** Symfony prioriza la seguridad y se actualiza regularmente para garantizar la estabilidad y la compatibilidad con las últimas versiones de PHP.

Symfony ofrece a los desarrolladores las herramientas y la estructura necesarias para crear aplicaciones web de alta calidad de manera eficiente y efectiva. Con un sólido compromiso con las mejores prácticas de desarrollo y una comunidad activa, Symfony sigue siendo una elección principal para proyectos web en PHP.

**Arquitectura MVC:**

Symfony se adhiere estrictamente al patrón MVC, lo que separa claramente la lógica de negocio (Model), la presentación (View) y el control (Controller) de una aplicación.

**1. Modelo (Model):**

El Modelo en Symfony representa la capa de lógica de negocio y la interacción con la base de datos. Aquí se definen las entidades y la lógica de la aplicación. Symfony utiliza el Object-Relational Mapping (ORM) para mapear objetos a la base de datos, lo que simplifica la gestión de datos.

**2. Vista (View):**

La Vista se encarga de la presentación de los datos al usuario. En Symfony, las Vistas suelen estar formateadas en Twig, un lenguaje de plantillas que simplifica la generación de HTML. Esto permite separar de manera efectiva la lógica de presentación del código del controlador y del modelo.

**3. Controlador (Controller):**

El Controlador actúa como intermediario entre el Modelo y la Vista. Responde a las solicitudes del usuario y procesa la lógica de la aplicación. En Symfony, los controladores son clases que gestionan las rutas y las acciones asociadas a esas rutas. Los controladores se encargan de recuperar datos del Modelo y pasarlos a la Vista para su presentación.

El patrón MVC de Symfony facilita la creación de aplicaciones organizadas y mantenibles. Al separar las responsabilidades de la lógica de negocio, la presentación y el control, Symfony promueve buenas prácticas de desarrollo, como la reutilización de código y la escalabilidad de las aplicaciones.

Symfony ofrece una implementación sólida de MVC que permite a los desarrolladores crear aplicaciones web eficientes y flexibles. La estructura MVC de Symfony es una de las razones clave por las que se ha convertido en un marco de referencia en el desarrollo de aplicaciones web en PHP.

los componentes reutilizables son una característica destacada de Symfony. Symfony proporciona una amplia gama de componentes que abarcan desde el manejo de formularios hasta la autenticación y la manipulación de imágenes. A continuación, profundicemos en este aspecto:

**Componentes Reutilizables:** Symfony ofrece más de 30 componentes independientes que cubren una variedad de funcionalidades comunes en el desarrollo web. Algunos ejemplos incluyen:

**HttpFoundation:** Proporciona una abstracción de la solicitud y la respuesta HTTP, lo que facilita el manejo de datos entrantes y salientes.

**Routing:** Permite definir las rutas de URL y sus controladores asociados de una manera muy flexible.

**Form:** Facilita la creación y el procesamiento de formularios HTML y su validación.

**Security:** Ofrece herramientas para gestionar la seguridad, como la autenticación y la autorización.

**Translation:** Permite la internacionalización y localización de la aplicación.

**Validator:** Proporciona un marco para la validación de datos.

**Console:** Permite la creación de comandos de consola para tareas automatizadas.

**Gestión de Dependencias:** Symfony hace un uso intensivo del concepto de inyección de dependencias y proporciona un contenedor de servicios para gestionarlas. Esto significa que puedes definir servicios y sus dependencias, lo que facilita la organización y el intercambio de componentes dentro de su aplicación. Symfony te permite administrar todas las dependencias y configuraciones en un solo lugar, lo que mejora la mantenibilidad y la flexibilidad de tu aplicación.

**Entorno de Desarrollo Profesional:** Con Symfony, puede realizar pruebas unitarias y funcionales, depurar código de manera eficiente y automatizar tareas comunes, como la generación de código. Además, Symfony integra las mejores prácticas de desarrollo web, como la manipulación de bases de datos y la seguridad.

**Symfony Flex:** Symfony Flex es un administrador de aplicaciones que facilita la configuración y el manejo de proyectos Symfony. Ofrece una forma más sencilla y eficiente de gestionar las extensiones y configuraciones de Symfony. Symfony Flex instala y configura automáticamente las dependencias, lo que acelera el desarrollo y garantiza una mejor compatibilidad.

**Bundles:** Los bundles son paquetes de código que encapsulan funcionalidades específicas. Symfony incluye una amplia gama de bundles predefinidos que puedes utilizar en tu aplicación. También puedes crear tus propios bundles para modularizar y organizar tu código. Esto promueve la reutilización de código y facilita la colaboración en proyectos más grandes.

**Rutas y Enrutamiento:** Symfony proporciona un sistema de enrutamiento flexible que permite asignar URL a controladores y acciones. Puedes definir rutas de manera clara y legible, lo que facilita la navegación en tu aplicación.

**Twig:** Symfony utiliza Twig como motor de plantillas para la vista. Twig es conocido por su sintaxis sencilla y segura. Facilita la creación de plantillas HTML y evita problemas comunes, como ataques de inyección de código.

**Doctrine ORM:** Symfony se integra con Doctrine, un ORM (Mapeo Objeto-Relacional), que simplifica la interacción con bases de datos. Doctrine permite representar entidades de bases de datos como objetos en PHP, lo que facilita la manipulación de datos y la creación de consultas.

**Validación y Formularios:** Symfony proporciona componentes para la validación de datos y la creación de formularios. Puedes definir reglas de validación de manera clara y garantizar que los datos de entrada sean seguros. La creación de formularios se simplifica y se integra perfectamente con la validación.

**Seguridad:** Symfony incluye un sólido sistema de seguridad que cubre la autenticación, la autorización y la gestión de usuarios. Puedes proteger tus aplicaciones contra amenazas comunes, como ataques CSRF y XSS, de manera efectiva.

**Pruebas y Depuración:** Symfony proporciona herramientas integradas para realizar pruebas unitarias y funcionales. También puedes depurar tus aplicaciones de manera efectiva, lo que facilita la identificación y corrección de errores.

**Administración y Panel de Control:** Symfony facilita la creación de áreas de administración y paneles de control para la gestión de contenido. Puedes construir interfaces de administración personalizadas de manera eficiente.

**Comunidad Activa:** Symfony goza de una comunidad activa de desarrolladores. Esto significa que tienes acceso a documentación detallada, soporte en línea y una gran cantidad de bundles y soluciones complementarias desarrolladas por la comunidad.

**Seguridad y Mantenimiento:** Symfony se mantiene con altos estándares de seguridad y se actualiza regularmente. Esto garantiza la estabilidad y la compatibilidad con las últimas versiones de PHP y protege tus aplicaciones contra vulnerabilidades conocidas.

**Compatibilidad con Plataformas:** Symfony es compatible con diversas plataformas y servicios de alojamiento web. Esto te permite implementar tus aplicaciones en una amplia gama de entornos y elegir el que mejor se adapte a tus necesidades.

Nota: Para la instalación y trabajo con Symfony se ahondar en la presentación de Power Point, que se enfoca más en lo práctico

# Referencias:

https://www.php.net/