# Tema 2: Inserción de código en aplicaciones web

## Herramientas y aplicaciones PHP

### Contenedor Docker

Contiene Servidor Web, PHP y Sistema gestor de BD.

### Editor de código

Extensiones útiles:

* PHP Intelephense
* PHP Debug
* VSCodium
* VisualStudio Code

### Depurador

Un **depurador** es un programa que se utiliza para probar una aplicación en desarrollo. Ayuda a detectar y corregir errores

Ejemplo: **XDebug** es una extensión de PHP que proporciona varias características útiles para el desarrollo:

◦ Depuración paso a paso

◦ Informes de errores

◦ Tracing

## Sintaxis básica de PHP

Una aplicación web en PHP está **compuesta por archivos** HTML, CSS y PHP (y JavaScript para el entorno cliente).

El código PHP puede diseñarse de dos maneras:

* Embebido dentro del código HTML
* Separando el código de apoyo a plantillas

Un archivo PHP tiene extensión **.php**

Todo código PHP debe comenzar con la etiqueta <?php

Y termina por ?>

Desde PHP 7.0 la etiqueta final es optativa si no hay código HTML detrás otros mecanismos de inserción están obsoletos

### Mezclando PHP y HTML

En PHP se puede mezclar código con HTML.

<!DOCTYPE html>

<html>

<head>

   <title>Ejemplo PHP + HTML</title>

</head>

<body>

   <h1>Bienvenido</h1>

   <?php

      $nombre = "Irene";

      echo "<p>Hola, $nombre</p>";

   ?>

</body>

</html>

Todo lo que esté fuera de las etiquetas <?php ... ?> se mostrará tal cual en la página. Además, un archivo PHP puede contener varias etiquetas de apertura y cierre. Gracias a esto, se pueden usar estructuras de control dentro del HTML, como if, else y endif, para decidir qué texto mostrar en función de una condición.

Texto

El contenido generado por IA puede ser incorrecto.

### Características de PHP

* PHP es un **lenguaje interpretado**.
* Las sentencias terminan con **punto y coma (;)** (igual que en Java o C).
* Los espacios en blanco, tabuladores y saltos de línea **se ignoran**.
* Las instrucciones y nombres son **sensibles a mayúsculas y minúsculas**.

### Mostrar texto en PHP

Se usan **echo** y **print()** para enviar texto al cliente.

### Comentarios

Los **comentarios** sirven para aclarar el código.

* **Hasta final de línea:**
  + // comentario
  + # comentario
* **Multilínea:**
  + /\* comentario \*/

### Reutilización de código

**include(archivo)**

* Inserta un archivo en la posición de la llamada.
* Si no encuentra el archivo → muestra advertencia (*warning*).

**require(archivo)**

* Inserta un archivo en la posición de la llamada.
* Si no encuentra el archivo → genera error y detiene la ejecución.

**Ruta de archivos**

* Se incluyen según la ruta dada o, si no se indica, según *include\_path*.

#### Variantes para evitar duplicados

**include\_once(archivo)** y **require\_once(archivo)**:

* Se aseguran de que el archivo solo se incluya una vez.
* Si el archivo se incluye varias veces y contiene funciones ya declaradas → error fatal.
* Son más lentas, pero recomendables por seguridad.

**Configuración en php.ini**

* + *auto\_prepend\_file*
  + *auto\_append\_file*

Funcionan de forma similar a *include()*. Permiten incluir archivos automáticamente en todas las páginas sin escribir la instrucción en cada una.

### Novedades de versiones

#### PHP 8.0

* **Argumentos nombrados**: se pueden pasar parámetros por nombre y en cualquier orden.
* **Promoción de propiedades constructivas**: inicializar propiedades directamente en el constructor.
* **Tipos de unión**: permiten declarar varias opciones de tipo.
* **Expresiones match**: alternativa a switch, más sencilla.
* **Operador nullsafe**: evita errores al acceder a valores nulos.
* **Comparaciones inteligentes** entre cadenas y números
* **Errores consistentes en funciones internas**: lanzan excepción si el parámetro no es válido.

#### PHP 8.1

* **Enumeraciones (enum)**: definición directa de tipos enumerados.
* **Propiedades de solo lectura (readonly)**: atributos que no pueden modificarse tras inicialización.
* **First-class Callable Syntax**: obtener referencia directa a funciones.
* **Expresión new en constructor**: objetos instanciados como parámetros por defecto.
* **Tipos de intersección pura**: un valor debe cumplir varios tipos a la vez.
* **Tipo de retorno never**: funciones que no devuelven nada (terminan con die(), exit() o excepción).
* **Constantes de clase final**: no se pueden sobrescribir en clases hijas.
* **Notación octal explícita**: 0o16 === 14 (true)
* **Fibers**: implementación ligera de concurrencia (pausar/reanudar código).
* **Soporte de expansión de arrays**: también con claves string.

#### PHP 8.2

* **Clases de solo lectura**: toda la clase es inmutable.
* **Tipos de forma normal disyuntiva (DNF)**: combinan unión e intersección de tipos.
* **Tipos independientes null, false y true**
* **Propiedades dinámicas en desuso**: asignar propiedades sin declararlas es deprecated.
* **Nueva extensión random**: API orientada a objetos para números aleatorios.
* **Constantes en rasgos**: accesibles solo desde la clase que usa el trait.

## Tipos de datos en PHP

### Características del tipado en PHP

El tipo de una variable **no lo declara el programador**, lo decide PHP en tiempo de ejecución.

PHP es un lenguaje de **tipado débil**: las variables cambian de tipo al asignarles valores distintos.

### Funciones útiles para tipos

* **var\_dump($value)** → muestra el tipo y el valor de una variable o expresión.
* **gettype($value)** → devuelve el tipo de una variable como string.

#### Tipos escalares

* **bool / boolean** → valores true o false.
* **int / integer** → número entero.
* **float / double** → número de coma flotante (antes llamado real).
* **string** → cadena de caracteres.

#### Tipos compuestos

* **array** → matriz, mapa ordenado que asocia valores con claves.
* **object** → instancia de una clase.
* **callable** → llamadas de retorno.
* **iterable** → pseudotipo para usar en foreach (PHP 7.1+).

#### Tipos especiales

* **resource** → referencia a un recurso externo.
* **null** → variable sin valor asignado (o destruida con unset()).
* **mixed** → acepta cualquier valor (PHP 8.0+).
* **void** → tipo de retorno que indica que la función no devuelve valor.

### Control de tipos

**Comprobar si existe una variable**

isset($var) → devuelve true si existe.

**Destruir una variable**

unset($var) → elimina el valor de la variable.

**Comprobar si existe una variable y está vacía**

empty($var) → devuelve true si no tiene valor.

### Forzado de tipos

* Se puede tratar una variable como si fuera de otro tipo usando **casting**.
* El tipo se escribe entre paréntesis delante de la variable.

$cantidad = 10;  
$resultado = (double)$cantidad;  
var\_dump($cantidad); // int(10)  
var\_dump($resultado); // float(10)

### Conversión explícita permitida

* (real) → alias de float, obsoleto desde PHP 8.0
* (string) o (binary)
* (array)
* (object)
* (int) → integer
* (bool) → boolean
* (float) o (double)

**settype()**

También se puede usar la función settype($var, $type) para cambiar el tipo de una variable. Establece el tipo de $var al tipo especificado en $type.

### Reinterpretación de variables

Permiten convertir una variable al tipo apropiado:

* intval($value, $base = 10) → int
* floatval($value) o doubleval($value) → float
* strval($value) → string

## Constantes en PHP

* Almacenan valores que no se pueden modificar.
* Se pueden definir constantes escalares y también de arrays (desde PHP 7).
* Se definen con: define(nombre, valor, case\_insensitive).
* También se puede usar const, que define en tiempo de compilación.
* Diferencia:
  + **define()** → tiempo de ejecución.
  + **const** → tiempo de compilación (no admite expresiones).
* Convención: los nombres se escriben en MAYÚSCULAS.
* Se referencian sin $.

define('PRECIO',100);   
echo PRECIO;

## Variables en PHP

* Una variable es un espacio en memoria que almacena un valor.
* Se crean con $ + nombre.
* No necesitan declaración previa.

$nombre="Mundo";  
echo "Hola, $nombre";

#### Características de los nombres de variables

* Deben ser descriptivos.
* Longitud apropiada (ni muy corta ni muy larga).
* Sin símbolos (excepto guion bajo \_).
* Usar guion bajo o camelCase.
* No pueden empezar con número.
* Son sensibles a mayúsculas y minúsculas (user ≠ User ≠ USER).

**Asignación de variables**

Por defecto las variables en PHP se asignan **por valor** → la nueva variable recibe una copia.

Se pueden asignar **por referencia** con & → ambas variables apuntan al mismo contenido (si una cambia, la otra también).

Si no se inicializa una variable, PHP le da un **valor por defecto**:

* Boolean → false
* Entero y flotante → 0
* Cadena → "" (vacía)
* Array → [] (vacío)

**Variables variables**

Permiten crear nombres de variables de forma dinámica.

Se usa $$ delante del nombre.

El valor de una variable se convierte en el nombre de otra.

**Ámbito de las variables**

El **ámbito** define desde dónde se puede acceder a una variable.

* **Globales**: visibles en todo el script, pero no dentro de funciones directamente.
* **Locales**: visibles solo dentro de la función en la que se declaran.
* **Globales dentro de funciones**: para usar una variable global dentro de una función se necesita:
  + La palabra clave global, o
  + El array superglobal $GLOBALS.

**Variables superglobales**

Son arrays especiales que están disponibles en todo el script, sin importar el ámbito:

* **$GLOBALS** → referencias a todas las variables globales.
* **$\_SERVER** → información del servidor y del entorno de ejecución.
* **$\_GET** → variables recibidas por parámetros en la URL.
* **$\_POST** → variables recibidas por formularios (POST).
* **$\_FILES** → ficheros subidos por formulario.
* **$\_COOKIE** → variables de cookies.
* **$\_SESSION** → variables de sesión.
* **$\_REQUEST** → combina $\_GET, $\_POST y $\_COOKIE.
* **$\_ENV** → variables de entorno.

## Operadores

#### Aritméticos

|  |  |
| --- | --- |
| + | Suma |
| - | Resta / Negación |
| \* | Multiplicación |
| / | División |
| % | Módulo (resto) |
| \*\* | Exponenciación |

#### Asignación

|  |  |  |
| --- | --- | --- |
| Operador | Ejemplo | Equivalente |
| = | $a = 5 | asigna |
| += | $a += 5 | $a = $a + 5 |
| -= | $a -= 2 | $a = $a - 2 |
| \*= | $a \*= 3 | $a = $a \* 3 |
| /= | $a /= 2 | $a = $a / 2 |
| .= | $a .= "txt" | concatena |

#### Incremento / Decremento

|  |  |
| --- | --- |
| ++$a | Preincremento (suma y luego usa) |
| $a++ | Postincremento (usa y luego suma) |
| --$a | Predecremento |
| $a-- | Postdecremento |

#### Comparación

|  |  |
| --- | --- |
| == | Igual valor |
| === | Igual valor y tipo |
| != ó <> | Diferente valor |
| !== | Diferente valor o tipo |
| < , > , <= , >= | Comparaciones numéricas |
| <=> | Nave espacial → -1, 0 o 1 |

#### Arrays

|  |  |
| --- | --- |
| + | Unión de arrays |
| == | Igualdad (mismos pares clave-valor) |
| === | Identidad (igualdad + orden + tipo) |
| != ó <> | Desigualdad |
| !== | No idéntico |

#### Lógicos

|  |  |
| --- | --- |
| ! | Negación |
| && / and | Y lógico |
| || / or | O lógico |
| xor | O exclusivo |

#### Bit a bit

|  |  |
| --- | --- |
| ~ | NOT |
| & | AND |
| | | OR |
| ^ | XOR |
| << | Desplazar a la izquierda |
| >> | Desplazar a la derecha |

#### Otros operadores

* **Concatenación (.)** → une cadenas.
* **Ternario (?:)** → condicional corto.
* **Fusión de nulo (??)** → devuelve el primer valor no nulo.
* **Ejecución (``)** → ejecuta comandos del sistema.
* **Control de errores (@)** → suprime warnings.
* **instanceof** → comprueba si un objeto pertenece a una clase.

#### Precedencia de operadores

Tabla

El contenido generado por IA puede ser incorrecto.