

UT2 INSERCIÓN DE CÓDIGO EN PÁGINAS WEB

Índice

- Inserción de código de servidor
- Ámbito de ejecución
- PHP: Hypertext Preprocessor
- Etiquetas de servidor
- Sintaxis básica
- Comentarios
- Imprimir: echo y print
- Variables y tipos de datos
- Expresiones y operadores
- Precedencia de operadores
- Combinación de etiquetas HTML y código PHP
- Insertar etiquetas dentro de un script PHP
- Insertar todas las etiquetas dentro de un script PHP
- Ámbito de las variables
- Cadenas de texto y concatenación
- Funciones relacionadas con los tipos de datos
- Operador ternario

Este material está basado en:

1. Libro de título: Desarrollo web en Entorno Servidor, de Juan Luis Vicente Carro, Editorial Garceta.
2. <https://cizacas.github.io/DWES2425/>
3. <https://www.php.net/manual/es/index.php>

Inserción de código de servidor

Una página web sencilla tiene tres partes:

- cabecera (definición de directivas),
- zona de metadatos
- cuerpo del documento (información a mostrar).

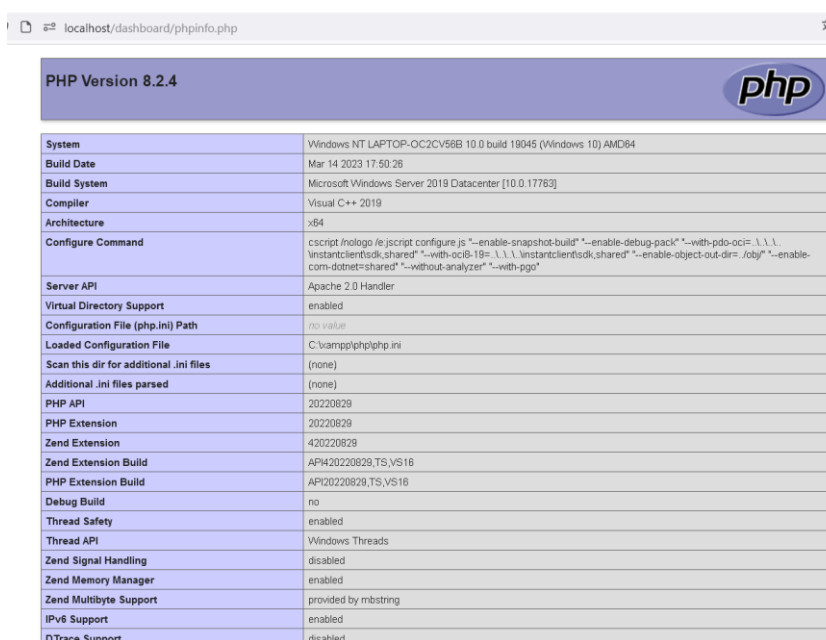
Una vez creada la página web estática, se inserta el código que será compilado e interpretado por el servidor. El código de servidor se ejecuta al procesar la petición HTTP del cliente.

Ámbito de ejecución

Los lenguajes de servidor solo pueden acceder a los recursos alojados en el servidor o a otros ofrecidos a través de la red.

No pueden acceder a recursos del cliente.

Poseen archivos de configuración, como php.ini, para definir el comportamiento de la aplicación.



System	Windows NT LPTOP-OC2CV56B 10.0 build 19045 (Windows 10) AMD64
Build Date	Mar 14 2023 17:50:26
Build System	Microsoft Windows Server 2019 Datacenter [10.0.17763]
Compiler	Visual C++ 2019
Architecture	x64
Configure Command	cmd.exe /X:Y /C:"php-src\configure.js --enable-snapshot-build --enable-debug-pack --with-pdo-oci=\\.\.\.\instantclient\shared --with-oci8-19=\\.\.\.\instantclient\shared --enable-object-out-dir=.obj" --enable-com-dotnet=shared --without-analyzer --with-pgsql
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	no value
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20220829
PHP Extension	20220829
Zend Extension	40220829
Zend Extension Build	API420220829,TS,VS16
PHP Extension Build	API20220829,TS,VS16
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled

PHP: Hypertext Preprocessor

- PHP es sensible a las mayúsculas.
- Es un lenguaje de código abierto.
- Permite su uso junto con documentos HTML mediante la inserción de fragmentos de código PHP delimitados por etiquetas especiales.
- El código PHP es interpretado por el servidor web y genera el correspondiente código HTML para mostrar en el navegador.
- Las páginas PHP tienen extensión .php.
- PHP tiene una interfaz de desarrollo basada en la programación orientada a objetos.
- Es compatible con diferentes tecnologías y servidores web, y soporta varios sistemas de bases de datos (para MySQL tiene librerías específicas).

Etiquetas de servidor

La incrustación de código PHP en HTML se puede hacer de varias maneras, pero la forma compatible con todas las plataformas es:

```
<?php
// Código PHP aquí
?>
```

Ejemplo de página con PHP

```
<?php
    $expresion = "1";
    if ($expresion == "1") {
        print("1.- Empiezan líneas generadas por PHP <br>");
        print("2.- El texto está por instrucción print de PHP");
        echo 'Este otro texto está por instrucción echo';
    }
?>
```

Para saber más:

<https://www.php.net/manual/es/language.basic-syntax.phptags.php>

Sintaxis básica

- PHP es sensible a las mayúsculas.
 - Los espacios en blanco dentro del código embebido no afectan la ejecución.
 - El final de una instrucción se indica con punto y coma (;).
 - Los scripts embebidos pueden situarse en cualquier parte del código HTML.
 - El número de scripts es indefinido.
 - Al ejecutarse, el script se sustituye por el resultado de su ejecución.
-

Comentarios

```
Comentarios de una línea: // y #  
// Esto es un comentario  
# Esto también es un comentario  
Comentario de varias líneas: /* */  
/* Esto es un comentario de  
varias líneas */
```

Imprimir: echo y print

echo: muestra una o más cadenas, separadas por comas.

```
echo "Hola Mundo";  
echo "Hola", "Mundo";
```

print: muestra una única cadena.

```
print "Hola mundo";  
print "Hola "."mundo"; //Una única cadena que se concatena  
$resultado = print "Hola"; // Muestra "Hola" y $resultado será 1
```

Variables y tipos de datos

- El tipo de una variable no se suele especificar, se deduce en tiempo de ejecución, incluso puede cambiar.

```
$miVariable = 7;  
$miVariable = "cambio de tipo";
```

Reglas para nombrar variables:

- Deben comenzar con una letra o guión bajo (_).
- Solo pueden contener caracteres alfanuméricos y guiones bajos.
- No deben contener espacios en blanco.

En este curso vamos a adoptar la convención Lower Camel Case:

```
$ejemploDeLowerCamelCase
```

Tipos de datos:

- Booleano: true o false.
- Entero: números sin decimales (decimal, octal, hexadecimal).
- Real (float): números con decimales, también en notación científica.
- Cadena (string): texto entre comillas simples o dobles.
- Null: indica que la variable no tiene valor.

```
// Tipos escalares  
$entero = (int) 42;           // Entero  
$flotante = (float) 3.141592; // Flotante  
$booleano = (bool) true;     // Booleano  
$cadena = (string) "Hola mundo"; // Cadena de texto  
  
//El tipo de una variable no se suele especificar, se decide en tiempo de ejecución, incluso puede cambiar.  
$enteroSinDeclararDato = 42;           // Entero  
$flotanteSinDeclararDato = 3.141592;   // Flotante  
$booleanoSinDeclararDato = true;       // Booleano  
$cadenaSinDeclararDato = "Hola mundo"; // Cadena de texto
```

Para saber más

<https://www.php.net/manual/es/language.types.php>

Visualización de los valores de variables

```
<?php
$nombre = "Juan";
$edad = 30;

echo "Vamos a mostrar los valores de \$nombre y \$edad"; // Concatenar
variables con texto
echo "<br>";
echo "Mi nombre es " . $nombre . " y tengo " . $edad . " años.<br>"; //
Concatenar variables con texto
print "Mi nombre es " . $nombre . " y tengo " . $edad . " años.<br>"; //
Funciona de forma similar a echo
?>
```

Expresiones y operadores

- Operadores aritméticos: +, -, *, /, %, ++, --
- Operador de asignación: =
- Operadores de comparación: ==, !=, <, >, <=, >=
- Operador de control de error: @ (evita mensajes de error)
- Operadores lógicos: and (&&), or (||), !, xor
- Operadores de cadena:
 - Concatenación: . (punto)
 - Asignación con concatenación: .=

Precedencia de operadores (de mayor a menor)

1. ++, -- (unarios)
 2. *, /, %
 3. +, -
 4. <, <=, >, >=
 5. ==, !=
 6. &&
 7. ||
 8. and
 9. or
-

Combinación de etiquetas HTML y código PHP

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

Insertar etiquetas dentro de un script PHP

```
<?php  
echo "<h1> Hola mundo </h1>";  
?>
```

Insertar todas las etiquetas dentro de un script PHP

Otra filosofía, menos utilizada, es incrustar el código HTML en un script PHP.

```
<?php  
echo "<!DOCTYPE html>";  
echo "<html>";  
echo "<head>";  
echo "<meta charset='UTF-8'>";  
echo "<title>Hola mundo</title>";  
echo "</head>";  
echo "<body>";  
echo "<h1>Hola mundo</h1>";  
echo "</body>";  
echo "</html>";  
?>
```

Ámbito de las variables

Como ya hemos dicho, el lenguaje PHP es interpretado, es decir, no permanece en la memoria, sino que, una vez ejecutado el script de la página php, se pierde el contenido de las variables utilizadas en el código. Así pues, cada variable sólo tiene validez en el contexto en que está se defina, bien aparezca dentro de un servidor, de un script php o de una función. Por tanto, por defecto las variables que forman parte de una página o función tienen un ámbito exclusivamente local.

¿Cómo podemos hacer que una variable tenga un valor global, es decir, que se conserve tanto en una función como en una página o entre páginas diferentes?

Para alterar el ámbito de una variable disponemos de dos modificadores:

1. **global**: Este modificador permite acceder a una variable en la página php cuando se invoca a una función.

Vamos a ver los siguientes ejemplos:

```
<?php
function ver_texto(){
    echo "<br>";
    echo "No se va a ver el contenido de la variable \$texto dado que solo tiene ambito local";
    echo "<br>";
    echo $texto;
}

$texto = "Un, dos, tres...";

ver_texto();

function ver_texto_2(){
    global $texto;
    echo "<br>";
    echo "Ahora sí que se va a visualizar el contenido de \$texto";
    echo "<br>";
    echo $texto;
}

$texto = "Responda otra vez.";
ver_texto_2();
```


2. Otro método es mediante la matriz `$GLOBALS()`. Es un array que asocia en cada uno de sus elementos el nombre de la variable con su contenido de tal forma que accediendo al primero podemos saber su contenido dentro de un ámbito global.

```
//Con la matriz GLOBALS, podremos acceder a variables locales como si fueran globales
$x = 3;
$v = 6;

function Sumar(){

    $GLOBALS["z"] = $GLOBALS["x"] + $GLOBALS["v"];
    echo "<br>";
    echo "El resultado es:" . $GLOBALS["z"];
}

Sumar();
```

3. **Static**: Este modificador hace que la variable conserve el mismo valor en sucesivas ocasiones en que se invoque a la función, pero no pierde su valor cuando la ejecución del script php abandona este ámbito. Por eso, es necesario crearla como estática dentro de la función y asignarle valor en este ámbito.

```
function ver_valor()
{
    echo "<br>";
    static $valor=0;
    echo $valor;
    $valor++;
}

echo "La variable \$valor, aún siendo una variable local de la
función, "
. "aumenta cada vez que invocamos la función si la declaramos como
static";

ver_valor();
ver_valor();
ver_valor();
```

Resumen:

- Se pueden utilizar en cualquier lugar del programa.
 - Si la variable no existe se reserva espacio en memoria.
 - Si una variable se define dentro de una función es una **variable local** a la función.
 - Si aparece fuera de la función se considera distinta a la definida en la función.
 - Si dentro de una función se quiere usar una variable definida fuera hay que usar la palabra **global**.
 - Si se quiere mantener el valor de una función declara dentro al salir de esta, se debe definir como **static**.
-

Cadenas de texto

- Se definen tanto con comillas simples como dobles.
- Cuando se pone una variable dentro de unas comillas dobles, se procesa y se sustituye por su valor.

En el siguiente ejemplo definimos una variable como cadena de texto y mostramos su valor.

```
<?php
    $modulo="DWES";
    echo "<p> Módulo: $modulo </p>";
?>
```

PHP al encontrar la variable la sustituye por DWES. Para que PHP distinga correctamente el texto que forma la cadena del nombre, a veces es necesario encerrarla **entre llaves**.

```
<?php
    $modulo="DWES";
    echo "<p> Módulo: ${modulo} </p>";
?>
```

Para saber más:

<https://www.php.net/manual/es/language.types.string.php>

Concatenación de cadenas

- Concatenación punto (.)
- Operador de asignación y concatenación (.=) concatena al argumento del lado izquierdo la cadena del lado derecho.

```
<?php
    $a="Módulo ";
    $b= $a."DWES";
    //ahora $b contiene "Módulo DWES"

    $a .="DWES";
    // ahora $a también contiene "Módulo DWES"

?>
```

Funciones relacionadas con los tipos de datos en PHP

Obtener y establecer el tipo de datos

`gettype($variable)`

- **Descripción:** Devuelve el tipo de la variable como una cadena.

`settype($variable, "tipo")`

- **Descripción:** Convierte la variable al tipo especificado.

Ejemplo de gettype y settype

```
<?php
    $x = "10";
    settype($x, "integer");
    echo gettype($x); // integer

?>
```

Comprobación del tipo de datos

Estas funciones devuelven true si la variable es del tipo indicado:

- `is_array($var)`
- `is_bool($var)`
- `is_float($var)`
- `is_int($var)`
- `is_string($var)`
- `is_object($var)`
- `is_null($var)`

```
<?php

// Variables iniciales
$entero = 42;                // Entero
$flotante = 3.141592;        // Flotante
$booleano = true;            // Booleano
$cadena = "Hola mundo";      // Cadena de texto
$array = (array) [1, 2, 3];   // Array
$nulo = null;                 // Variable nula

// gettype: obtener el tipo de cada variable
echo "Tipo de \$entero: " . gettype($entero) . "<br>";

// settype: cambiar el tipo de una variable
settype($cadena, "integer");
echo "Nuevo tipo de \$cadena (convertido a integer): " . gettype($cadena)
. "<br><br>";

// is_array
if (is_array($array)) {
    echo "\$array es un array.<br>";
} else {
    echo "\$array NO es un array.<br>";
}
```

Otras funciones útiles

`isset($variable)`

- **Descripción:** Verifica si la variable está definida y no es null.

```
<?php
    if (isset($nombre)) {
        echo "La variable está definida";
    }
?>
```

`unset($variable)`

- **Descripción:** Destruye la variable.

```
<?php
    $nombre = "María";
    unset($nombre);
?>
```

Funciones para Fechas

Funciones para **Fechas**: no hay un tipo específico. La información fecha y hora se almacena como un número entero y hay una serie de funciones en PHP para trabajar con ellas:

La función `date()` en PHP **formatea una fecha/hora local** y devuelve una cadena con el formato especificado.

- **\$format**: Una cadena que especifica el formato de salida (por ejemplo, "Y-m-d H:i:s").
- **\$timestamp** (*opcional*): Un entero que representa una marca de tiempo Unix. Si no se proporciona, se usa la hora actual.

```
<?php
    $fecha = date (string $formato [, int $fechahora]); //Devuelve
un string
?>
```

Algunos formatos comunes

- Y - Año completo (e.g., 2025)
- m - Mes (01 a 12)
- d - Día del mes (01 a 31)
- H - Hora en formato 24h (00 a 23)
- i - Minutos (00 a 59)
- s - Segundos (00 a 59)
- l - Día de la semana (e.g., Monday)

```
<?php
    echo date("Y-m-d H:i:s"); // Muestra la fecha y hora actual
    echo date("l", strtotime("2025-09-24")); // Muestra el día de
    la semana para esa fecha
?>
```

Si queremos establecer una zona horaria disponemos del siguiente comando

```
<?php

    date_default_timezone_set();

?>
```

Esto asegura que todas las funciones relacionadas con fecha y hora (como `date()`, `time()`, `strtotime()`, etc.) usen la zona horaria de Madrid, España.

La función `getdate()` en PHP es muy útil cuando necesitas obtener **todos los componentes de la fecha y hora actual** (o de una marca de tiempo específica) en forma de un **array asociativo**.

```
<?php

    $array = getdate([int $timestamp]);

?>
```

```
<?php
    date_default_timezone_set('Europe/Madrid');
    $fecha = getdate();

    echo "Hoy es " . $fecha['weekday'] . ", " . $fecha['mday'] . " de " .
    $fecha['month'] . " de " . $fecha['year'];
?>
```

Esto es lo que devuelve en un array

```
(  
    [seconds] => 40  
    [minutes] => 10  
    [hours] => 15  
    [mday] => 24  
    [wday] => 3  
    [mon] => 9  
    [year] => 2025  
    [yday] => 266  
    [weekday] => Wednesday  
    [month] => September  
    [0] => 1695561040  
)
```

Para saber más

<https://www.php.net/manual/es/ref.datetime.php>

Operador ternario

Es una forma compacta de escribir una estructura condicional if-else.

```
<?php  
  
$a = 1;  
$b = 3;  
$imprimir = ($a < $b) ? "a es menor que b" : (($a > $b) ? "a es mayor  
que b" : "a es igual a b");  
print $imprimir;  
?>
```

Explicación paso a paso

1. Primera condición:

$(\$a < \$b)$ → si es verdadera, se imprime "a es menor que b".

2. Si no es verdadera, se evalúa la segunda parte:

$(\$a > \$b)$ → si esta es verdadera, se imprime "a es mayor que b".

3. Si ninguna de las anteriores es verdadera, se imprime "a es igual a b".

Este tipo de estructura es útil para simplificar condicionales anidados en una sola línea.