Tema 1: Inserción de código en páginas web

¿Qué aprenderás?

- La arquitectura cliente/servidor usada en Internet.
- Tipos de aplicaciones web.
- Qué es y para qué sirve el lenguaje PHP.
- La sintaxis básica del lenguaje PHP.

¿Sabías que…?

- El lenguaje PHP fue creado por Rasmus Lerdorf en 1995.
- En la página web php.net encontrarás una documentación detallada de la sintaxis de PHP junto con funciones y métodos útiles.
- PHP significaba inicialmente Personal Home Page.

1. Introducción

1.1. Programación en entornos cliente/servidor

Sabemos que Internet es un vasto universo de información. Ésta se distribuye en diferentes equipos, los cuales se interconectan entre sí usando diferentes componentes hardware (como hubs, switchs, routers, etc.) y diferentes protocolos de comunicación (TCP, IP, HTTP, FTP, etc.).

Para el desarrollo de aplicaciones que trabajen en este contexto se usa la arquitectura cliente/servidor. El cliente es aquel que solicita una información o servicio. El servidor es el que atiende a la petición del cliente y le da respuesta. Cliente y servidor se comunican mediante el intercambio de mensajes.

Cuando nosotros nos conectamos a Internet a través de un navegador y accedemos a una página web, estamos haciendo de cliente. Al teclear la URL estamos solicitando un servicio a un servidor. Éste recibe nuestro mensaje y ejecuta una secuencia de instrucciones para generar una respuesta en forma de contenidos que visualizaremos en nuestro navegador. Podremos interactuar con el contenido que vemos, por ejemplo, añadiendo productos a un carrito de compra. Cada acción que hacemos es una solicitud nueva al servidor, que ejecuta un código y nos devuelve una respuesta. Este es un ejemplo de la interacción entre cliente y servidor.

1.2. Tipos de aplicaciones web

En función del contenido y la interacción con el usuario y ofrezcan, podemos diferenciar entre las siguientes aplicaciones web:

- Aplicaciones web estáticas: son las primeras páginas web que existieron en Internet. Estaban formadas por un contenido estático, que no cambiaba. Por tanto, el usuario no podía interactuar con él para editarlo o adaptarlo a sus necesidades. Este tipo de aplicaciones usaban básicamente código HTML.
- Aplicaciones web dinámicas: este tipo de aplicaciones permiten interacción con el usuario, es decir, éste puede realizar acciones que modifiquen el contenido de la página, usando, por ejemplo, formularios y otros elementos para enviar solicitudes al servidor. Hay una comunicación bidireccional entre cliente/servidor, y ofrecen más funcionalidades que las aplicaciones web estáticas. Las aplicaciones web dinámicas requieren una programación más compleja, usando diversos lenguajes como PHP, ASP, JSP, Perl o Python.

1.3. Creación de aplicaciones web dinámicas

Ya sabemos qué son las páginas web dinámicas, ahora toca ver cómo las podemos hacer. Los lenguajes que se usan son, entre otros, PHP, ASP, JSP, Perl, Python, que se denominan lenguajes de scripting. Éstos se caracterizan por intercalar su código entre las etiquetas HTML que tiene la web.

El código de los lenguajes de scripting es interpretado en el servidor por un módulo que es capaz de reconocer el código de estos lenguajes entre el código HTML y así realizar las operaciones pertinentes.

En el caso de PHP se usa Apache para interpretar el código.

Instalación del entorno de programación XAMPP

2. PHP (Hypertext Preprocessor)

2.1. ¿Qué es PHP?

PHP es un lenguaje de scripting diseñado específicamente para usarse en la web. Concretamente en el lado del servidor. Es una herramienta para crear páginas webs dinámicas. Es uno de los lenguajes más populares en la actualidad.

2.2. Agregar código PHP a una página HTML

El lenguaje HTML es casi interactivo. Es decir, los formularios en HTML permiten a los usuarios introducir la información que la página web puede recopilar. Sin embargo, no se puede tener acceso a esa información sin usar un lenguaje diferente a HTML. PHP procesa la información de los formularios sin necesidad de un programa aparte, y da espacio para realizar otras tareas.

Los archivos de nuestras páginas web que incorporen código PHP pasarán de tener la extensión *.htm* o *.html* a *.php*, *.phtml, .php4*, *.php5, .php7 o .phps*. El código PHP lo escribiremos junto con las etiquetas HTML, englobado entre las siguientes etiquetas de inicio y fin de código PHP:

```
<?php          ?>
```

También podemos usar las etiquetas:

```
<?         ?>
```

PHP procesa todos los enunciados entre las dos etiquetas PHP. Una vez que la sección PHP se ha procesado, se descarta. O bien, si los enunciados PHP producen output, la sección PHP es reemplazada por dicho output. El explorador no ve la sección PHP, sólo su output, si lo hay.

Ejemplo de código “HolaMundo.php”:

```
<html>
     <head><title>Hola, mundo</title></head>
     <body>
          <?php echo "<p>HOLA MUNDO!</p>"?>
     </body>
</html>
```

En este ejemplo las etiquetas PHP encierran sólo una función: echo. La función echo se usa con mucha frecuencia. Simplemente produce como output el texto entre comillas dobles.

2.3. Escribir código PHP

Los enunciados PHP terminan con un punto y como (;). PHP no nota los espacios en blanco ni los finales de línea. Continúa leyendo un enunciado hasta topar con un punto y coma o la etiqueta PHP de cierre, sin importar cuántas líneas abarque el enunciado. Omitir el punto y coma es un error común, que produce un error a la hora de interpretar el código PHP.

En ocasiones un grupo de enunciados se combina para formar un bloque. Un bloque se encierra entre llaves ({}). Un bloque de enunciados se ejecuta en conjunto. Un uso común

para los bloques es en un bloque condicional, en el cual los enunciados se ejecutan sólo cuando ciertas condiciones son verdaderas.

En general, el lenguaje PHP no es *case sensitive*, esto quiere decir que no importa si las palabras clave las escribimos en mayúsculas o minúsculas. *Echo*, *echo*, *ECHO* y *eCHo* son todos iguales para PHP.

2.4. Mensajes de error y advertencia

PHP proporciona los siguientes mensajes de error y advertencias:

- Mensaje de error: este mensaje se recibe cuando el programa tiene un problema que no le deja ejecutarse. El mensaje tiene tanta información como sea posible, para ayudar al programador a identificar el problema.
- Mensaje de advertencia: este mensaje se recibe cuando el programa ve un problema que no es lo suficientemente serio como para impedir su ejecución. Los mensajes de advertencia indican que PHP cree que probablemente haya algo mal.
- Aviso: este mensaje se recibe cuando PHP ve una condición que podría ser un error o que podría estar perfectamente bien. Los avisos, como las advertencias, no causan que el script deje de ejecutarse. Los avisos tienen menos probabilidad de indicar problemas serios que las advertencias. Los avisos sólo le dicen al programador que está haciendo algo inusual y es mejor revisar nuevamente el código para asegurarse que está haciendo realmente lo que quiere hacer.

El nivel de mensaje de error se puede modificar en PHP (quizá sólo me interese ser informado de los errores y no de las advertencias). Esto lo podemos hacer desde el archivo *php.ini*, añadiendo la siguiente línea:

```
error_reporting(OPTIONS);
```

Por ejemplo, si queremos ver todos los mensajes, pondremos:

```
error_reporting(E_ALL);
```

Para ver todos los errores, pero no los avisos:

```
error_reporting(E_ALL & ~E_NOTICE);
```

2.5. Uso de variables

Las variables son espacios que se reservan en memoria para guardar información. Una variable tiene un nombre y, en ella, se almacena información, que se puede usar posteriormente en el programa.

En PHP, las variables se crean dándoles un nombre precedido del carácter $. Los nombres pueden contener letras, números y guiones, y no pueden empezar por número. Hay que ir

con cuidado con cómo escribimos el nombre de las variables, ya que PHP diferenciará entre mayúsculas y minúsculas.

Algunos ejemplos de creación de variables:

```
$nombre, $Nombre, $var1, $x, $Cantidad_Total
```

2.6. Asignación de valores a las variables

Las variables pueden guardar números o cadenas de caracteres. Podemos asignar un valor a un variable mediante el signo de igual (=). Por ejemplo:

```
$nombre = "José Pérez";

$precio = 9.95;

$suma = $numero1 + $numero2;
```

Para ver por pantalla el valor que tiene una variable, podemos usar la función echo:

```
echo $nombre;
```

En el último ejemplo vemos como se le asigna a una variable el valor que resulta de la suma de los valores de otras dos variables. Obviamente, a las variables $numero1 y $numero2 se les debe haber asignado un valor previamente, sino obtendríamos un mensaje de error al intentar hacer la suma (¿qué sumamos?).

Las variables se crean la primera vez que le asignamos un valor. A lo largo del programa, ese valor lo podemos modificar con el uso del signo de igual (=). Si queremos destruir una variable que ya no necesitamos, lo haremos usando la función unset. En el siguiente ejemplo, destruimos la variable "nombre":

```
unset($nombre);
```

Una vez se ejecute la línea anterior, la variable "nombre" dejará de existir, ya no la podremos utilizar en nuestro programa (para usarla deberíamos volver a crearla).

2.7. Conversiones entre tipos de datos

Al declarar una variable en PHP hemos visto que en ningún momento estamos indicando el tipo de dato que va a guardar esa variable. PHP es un lenguaje que no requiere la definición explícita del tipo de datos con el que va a trabajar una variable.

Al crear una variable, ésta puede almacenar números y cadenas de caracteres. Esto vendrá en función de las sucesivas asignaciones que hagamos.

```
$var = 10; //La variable contiene un entero

$var = "Texto"; //La misma variable ahora contiene texto
```

El operador suma (+) puede hacer que una variable que contenga texto convierta automáticamente su valor a un número entero. No obstante, el resultado de la suma será impredecible.

Podemos forzar el cambio de tipos de datos en una variable de una forma implícita, de la siguiente manera:

```
$var = 10; //La variable contiene un entero

$var2 = (boolean)10; //La variable es un booleano
```

2.8. Información sobre una variable

PHP nos ofrece unas funciones que nos ofrecen información de una variable, como es el tipo de dato que guarda y su valor. Estas funciones son var_dump y print_r:

```
$var = 10;
var_dump($var);
print_r($var);
```

2.9. Estado de una variable

PHP nos ofrece unas funciones útiles para saber si una variable ha sido definida o si no tiene un valor asignado. Estas funciones son `isset()` y `empty()`.

2.9.1. Isset

La función isset() nos dice si una variable tiene un valor no nulo.

```
<?php
   $variable1 = "valor";
   $variable2 = null;
   echo isset($variable1); //IMPRIMIRÁ CIERTO
```

```
    echo isset($variable2); //IMPRIMIRÁ FALSO
?>
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

2.9.2. empty

La función empty() nos dice si una variable tiene un valor.

```
<?php
    $variable1 = "valor";
    $variable2 = null;
    echo empty($variable1); //IMPRIMIRÁ FALSO
    echo empty($variable2); //IMPRIMIRÁ TRUE
?>
```

En la siguiente tabla podemos ver el resultado que darían las funciones `isset` y `empty` en diferentes casos:

Declaración	isset($var)	empty($var)
$var = null	Falso	Cierto
$var = 0	Cierto	Cierto
$var = true	Cierto	Falso
$var = false	Cierto	Cierto
$var = "0"	Cierto	Cierto
$var = ""	Cierto	Cierto
$var = "hola"	Cierto	Falso

2.10. Uso de constantes

Las constantes son muy parecidas a las variables, en cuanto tienen un nombre y almacenan un valor. Pero a diferencia de las variables, con las constantes no podemos cambiar su valor a lo largo del código.

Las constantes se declaran mediante la función define. En el siguiente ejemplo, declaramos la constante "EDAD_MINIMA" con el valor 18:

```
define("EDAD_MINIMA", 18);
```

Ahora, podemos mostrar el valor de esa variable de la siguiente forma:

```
echo EDAD_MINIMA;
```

2.11. Operaciones con números

Operador	Descripción
+	Suma dos números
-	Resta el segundo número del primero
*	Multiplica dos números
/	Divide el primer número entre el segundo
%	Módulo. Encuentra el residuo cuando el primer número se divide entre el segundo número
++	Incremento, suma 1 al valor de la variable
--	Decremento, resta 1 al valor de la variable

Se pueden hacer varias operaciones aritméticas de una sola vez. El orden en que se realiza la operación aritmética es importante. PHP multiplica y divide primero, y luego lleva a cabo las sumas y las restas. La operación se efectúa, teniendo en cuenta el orden de las operaciones, de izquierda a derecha. Podemos cambiar el orden en el cual se lleva a cabo la operación usando paréntesis. La operación aritmética entre paréntesis se realiza primero.

PHP nos ofrece operaciones para representar los números en el formato deseado. Por ejemplo, si queremos mostrar un número que representa el precio de un producto, con dos decimales, escribiremos lo siguiente:

```
$precio = 20;

$f_precio = sprintf("%.2f", $precio);
```

La función "sprintf" nos mostrará el valor de la variable "precio" con dos decimales (20,00).

Mientras que los operadores de suma, resta, multiplicación, división y módulo son binarios, requieren dos valores para operar, el incremento y el decremento son unarios. Veamos un ejemplo de su uso:

```
$contador = 0;

$contador++;

echo $contador;
```

La salida será 1, porque hemos incrementado el valor inicial de $contador, que era 0. El operador de decremento funciona análogamente.

A veces podremos usar las operaciones aritméticas vistas de una forma especial, usando atajos:

```
$contador+=3;

$contador-=2;

$contador*=4;

$contador/=2;
```

Estas operaciones suman 3 al $contador, restan 2 al $contador, multiplican el $contador por 4, y dividen el $contador entre 2.

2.12. Uso de cadena de caracteres

Cuando almacenamos una cadena de caracteres en una variable, le indicamos a PHP dónde empieza y dónde termina la cadena usando comillas dobles o sencillas. Por ejemplo, los dos siguientes enunciados son iguales:

```
$string = "Hola, mundo";
```

```
$string = 'Hola, mundo!';
```

Si queremos que nuestra cadena de caracteres contenga una comilla simple (') como valor, se lo indicaremos a PHP con el carácter barra (\). Un ejemplo:

```
$string = 'El restaurante Tom\'s es bueno';
echo $string;
```

El fragmento de código anterior imprimirá por pantalla "El restaurante Tom's es bueno". De no haber puesto el carácter \, imprimiría "El restaurante Tom".

Podemos insertar saltos de línea y tabuladores en nuestras cadenas de caracteres escribiendo \n y \t, respectivamente.

Si encerramos una variable entre comillas dobles, PHP usa el valor de la variable. Sin embargo, si la encerramos entre comillas sencillas, PHP usa el nombre literal de la variable. Por ejemplo:

```
$edad = 12;
$resultado1 = "$edad";
$resultado2 = '$edad';
echo $resultado1;
echo "<br>";
echo $resultado2;
```

El código anterior imprimirá:

```
12
$edad
```

Para juntar cadenas de caracteres, usaremos el operador de concatenación punto (.). Un ejemplo:

```
$cadena1 = "Hola";
$cadena2 = "mundo";
echo $cadena1." ".$cadena2;
```

Imprimirá la cadena "Hola mundo", dejando un espacio en blanco entre las dos cadenas de caracteres.

Operador	Descripción	Ejemplo
echo	Imprime una cadena	echo “Hola”
explode	Separa una cadena en partes, usando un delimitador, y lo guarda en un array	explode(“,”, “Hola,mundo”) Devuelve: array(“Hola”, “mundo”)
strcmp	Compara dos cadenas. Devuelve 0 si son iguales y diferente de 0 si no lo son	strcmp(“hola”, “hola”) Devuelve: 0
strlen	Devuelve la longitud de una cadena	strlen(“hola”) Devuelve: 4

2.13. Uso de fechas y horas

PHP tiene la capacidad de reconocer fechas y horas y manejarlas en forma diferente que las cadenas de caracteres simples. La función que más utilizaremos será “date”, que convierte una fecha u hora en el formato que especifiquemos. El formato general es:

```
$miFecha = date(“formato”, $timestamp);
```

$timestamp es una variable que contiene la fecha almacenada. Podemos usar la función “date” sin pasarle como parámetro la variable con la fecha. De este modo, PHP cogerá la fecha del sistema. Por ejemplo:

```
$hoy = date(“Y/d/m”);
```

Esto devuelve la fecha de hoy en formato año/día/mes.

En la siguiente tabla se pueden ver los símbolos que se pueden usar para formatear una fecha. Las partes de la fecha se pueden separar con guiones, puntos, contrabarras o espacios.

Símbolo	Significado	Ejemplo
M	Mes en texto, abreviado	ene
F	Mes en texto, sin abreviar	enero
m	Mes en números precedido por ceros	02, 12
n	Mes en números sin cero precedente	1, 12
d	Día del mes; dos dígitos precedidos por ceros	01, 14
j	Día del mes sin el cero precedente	3, 30
l	Día de la semana en texto, sin abreviar	viernes
D	Día de la semana en texto, abreviado	vie
w	Día de la semana en números	De 0 (domingo) a 6 (sábado)
Y	Año en cuatro dígitos	2018
y	Año en dos dígitos	18
g	Hora entre 0 y 12 sin ceros precedentes	2, 10
G	Hora entre 0 y 24 sin ceros precedentes	2, 15
h	Hora entre 0 y 12 precedida por ceros	01, 10
H	Hora entre 0 y 24 precedidas por ceros	00, 23
i	Minutos	00, 59
s	Segundos	00, 59
a	am o pm en minúsculas	am, pm

A	AM o PM en mayúsculas	AM, PM

2.14. Almacenar tiempo en una variable

Para almacenar la fecha actual en una variable escribiremos:

```
$hoy = time();
```

Otra forma de almacenar la fecha actual sería:

```
$hoy = strtotime("today");
```

Podemos usar la función strtotime con diversos parámetros:

```
$fecha = strtotime("tomorrow");
$fecha = strtotime("now + 24 hours");
$fecha = strtotime("2 weeks ago");
```

2.15. Operadores de comparación

En PHP tenemos los siguientes operadores de comparación simple:

Operador	Descripción
==	¿Son los dos valores iguales?
>	¿Es el primer valor mayor que el segundo valor?
>=	¿Es el primer valor mayor o igual que el segundo valor?
<	¿Es el primer valor menor que el segundo valor?
<=	¿Es el primer valor menor o igual que el segundo valor?
!=	¿Son los dos valores diferentes entre sí?
<>	¿Son los dos valores diferentes entre sí?

Estos operadores de comparación podemos usarlos con valores numéricos y con cadenas. En este último caso, la comparación se realiza teniendo en cuenta el valor del código ASCII asociado a cada carácter.

Ejemplos de comparaciones:

```
$edad < 18;
$valor1 != $valor2;
$nombre == "Dani";
```

Hay que tener cuidado con el operador de comparación, que suele confundirse con el operador de asignación. No es lo mismo:

```
$nombre == "Dani";
$nombre = "Dani";
```

En la primera sentencia comprobamos si el valor de la variable $nombre es igual al literal "Dani". En la segunda sentencia, almacenamos en valor "Dani" dentro de la variable $nombre.

Los operadores de comparación los utilizaremos sobre todo en bloques condicionales, que se ejecutarán si se cumple cierta condición, que evaluaremos con comparaciones. Esto se verá en el capítulo siguiente.

2.16. Patrones de caracteres

Las comparaciones entre cadenas de caracteres admiten patrones que nos permiten realizar comparaciones a otro nivel, como por ejemplo buscar si una cadena empieza por una determinada letra, si contiene unos caracteres, o cumple ciertas condiciones.

El uso de patrones puede ser útil al recoger datos de un formulario, para comprobar que lo que el usuario ha escrito tiene algún sentido antes de almacenarlo en una base de datos (por ejemplo, la dirección de correo contiene una @, o el nombre de usuario no contiene espacios.

Los patrones los construiremos usando caracteres literales y caracteres especiales. Los caracteres literales son caracteres normales, sin ningún otro significado especial (una c es una c). Los caracteres especiales tienen un significado especial en el patrón. A continuación se pueden ver estos caracteres especiales y su significado.

Carácter	Significado	Ejemplo	Concuerda	No concuerda
^	Inicio de línea	^h	hola	¡hola!
$	Final de línea	a$	casa	Casas
.	Cualquier carácter	..	Cualquier cadena con dos caracteres	asd
?	Carácter precedente opcional	lea?n	lean, len	loan
()	Agrupa caracteres literales en una cadena que debe concordar exactamente	l(ea)n	lean	len, ln
[]	Encierra un conjunto de caracteres literales opcionales	l[ea]n	len, lan	lean, ln
-	Representa todos los caracteres entre dos caracteres	m[a-c]n	man, mbn, mcn	mdn, mun, maan
+	Uno o más de los elementos anteriores	puerta[1-3]+	puerta111, puerta13, puerta2	puerta, puerta55
*	Cero o más de los elementos anteriores	puerta[1-3]*	puerta, puerta32	puerta4, puerta55
{ , }	Los números de inicio y final en un rango de repeticiones	a{2,5}	aa, aaaaa	a, xx3
\	El siguiente carácter es literal	m*n	m*n	men, mean

(|)	El conjunto de cadenas alternas	(Tom | Tommy)	Tom, Tommy	Thomas, To

Con los caracteres especiales anteriores podemos construir complejos patrones. Una cadena se compara con el patrón y, si concuerda, la comparación es verdadera. A continuación veremos algunos ejemplos:

1. Cadenas que empiecen con una letra mayúscula: ^[A-Z].*
   - ^[A-Z] : letra mayúscula al inicio de la cadena.
   - .* : una cadena de caracteres con uno o más caracteres.
2. Dos cadenas alternas: Buenas (tardes | noches)
   - Buenas : caracteres literales.
   - (tardes | noches) : tardes o noches.
3. Cualquier código postal: ^[0-9]{5}(\-[0-9]{4})?$
   - ^[0-9]{5} : cualquier cadena de cinco números.
   - \- : literal
   - [0-9]{4} : cualquier cadena de cinco números.
   - ( )? : agrupa las dos últimas partes del patrón y las hace opcionales.
4. Cualquier cadena que tenga @ y termine en .com: ^.+@.+\.com$
   - ^.+@ : Cualquier cadena de uno o más caracteres al inicio, seguida de @.
   - .+ : cualquier cadena de uno o más caracteres
   - \. : un punto literal.
   - com$: una cadena com literal al final de la cadena.

Para comparar una cadena con un patrón usaremos la función ereg, tal y como sigue:

```
ereg("[0-9]*", "1234");

ereg($patron, $cadena);
```

En el primer ejemplo, se han utilizado literales para el patrón y la cadena, y en el segundo ejemplo se han usado variables.

2.17. Operadores AND, OR, XOR

Los operadores AND, OR y XOR los usaremos para conectar diversas comparaciones, que queremos que se evalúen a la vez.

Operador	Sinónimo	Es cierta si	Ejemplo
AND	&&	Ambas comparaciones son ciertas	$apellido == “Santiago” and $provincia == “Barcelona”
OR	||	Una de las comparaciones o ambas son ciertas	$edad > 8 or $edad < 64
XOR		Una de las comparaciones es cierta pero no ambas	$provincia == “Barcelona” xor $provincia == “Madrid”

Por supuesto podremos concatenar tantas comparaciones como queramos mediante los operadores anteriores. Las comparaciones que usan AND se comprueban primero, seguidas de las comparaciones que usan XOR y, finalmente, se comprueban las que usan OR. Aunque, este orden lo podemos modificar con el uso de paréntesis. No funcionarán, pues, de la misma manera los siguientes enunciados:

```
$edad == 20 || $edad == 30 && $nombre == "Dani"

($edad == 20 || $edad == 30) && $nombre == "Dani"
```

En la primera línea, primero se mira si $edad es igual a 30 y si $nombre es igual a “Dani”. Si ambos concuerdan, la condición es verdadera y el programa no necesita revisar el operador or. Si las condiciones del operador and no son ambas ciertas, entonces el programa revisa el operador or.

En la segunda línea, mira si $edad es 20 o 30. Si es así, esta parte de la condición es verdadera. Sin embargo, la comparación en el otro lado de and también debe ser verdadera, de modo que la comprueba a continuación.

2.18. Comentarios

Los comentarios son notas que están insertadas en el programa mismo. Se emplean para facilitar la tarea de los programadores ya que no realizan ningún papel activo en la generación del código. Suelen usarse para explicar fragmentos del código o hacer anotaciones al respecto.

Pueden insertarse comentarios en los programas en cualquier parte, siempre y cuando indiquemos a PGP que son comentarios. Para eso se usan los siguientes caracteres:

```
/*Aquí empieza un comentario de varias líneas. Todo lo
que escriba a continuación PHP lo ignorará

…

Aquí acabo el comentario*/

#Esto es un comentario de una línea

//Esto también es un comentario de una línea
```

2.19. Función echo

Como ya se ha indicado a lo largo de este documento, la función echo se usa para producir output. Esta salida es enviada al explorador del usuario, el cual maneja el output como HTML.

El formato general de un enunciado echo es:

```
echo objetosalida1, objetosalida2, objetosalida3…
```

El objetosalida puede ser un número, una cadena o una variable. Una cadena debe estar encerrada entre comillas. Podemos indicar tantos objetosalida como queramos, separados por coma. El resultado que veremos en el navegador será el valor de todos los objetosalida concatenados (sin espacios, a no ser que los pongamos nosotros).

2.20. Función exit

Si queremos que el programa deje de ejecutarse, que se pare en algún punto intermedio, podemos usar la función exit. El formato es:

```
exit("mensaje");
```

Entre comillas pondremos el texto que queremos que se muestre como mensaje (output) cuando el programa para su ejecución.

La función die es otra forma de parar la ejecución de nuestros programas. Funciona exactamente igual que la función exit.

Ejemplo de inserción de código PHP en páginas web

Test de autoevaluación

¿Qué instrucción muestra el contenido de una variable de PHP en una página HTML?

a) echo
b) show
c) println

¿Qué función de PHP nos permite saber si una variable ha sido declarada?

a) empty()
b) unset()
c) isset()

¿Cómo se define una constante en PHP?

a) define("CONSTANTE", 10)
b) define($constante, 10)333
c) define(10, $constante)

Recursos y enlaces

- Descarga del entorno XAMPP
- Manual oficial de PHP

Conceptos clave

- **Arquitectura cliente/servidor**: es un modelo para el diseño de aplicaciones en que las tareas se reparten entre el que provee recursos, el servidor, y el que los demanda, el cliente.
- **Aplicaciones web estáticas**: son aquellas aplicaciones en las que el usuario obtiene una página estática, es decir, cuyo contenido no cambia con sus acciones. No existen peticiones del cliente hacia el servior, ni respuestas del servidor hacia el cliente.
- **Aplicaciones web dinámicas:** son las aplicaciones web cuyo contenido se va construyendo y modificando en función de las peticiones que haga el usuario. Hay una comunicación constante entre el cliente y el servidor.
- **PHP:** es un lenguaje de programación web que se ejecuta en el servidor usado para crear aplicaciones web dinámicas. Es uno de los lenguajes más populares en la actualidad.

Ponlo en práctica

Actividad 1

Crea una aplicación web usando PHP en la que se declaren dos variables que contengan dos valores numéricos. Haz la operación de suma, resta, multiplicación y división y muestra los resultados. En el caso de la división, haz que se muestren únicamente dos decimales.

SOLUCIÓN