



## **Desarrollo web en entorno servidor**

### **Unidad 2: Inserción de código en páginas web**

## ÍNDICE

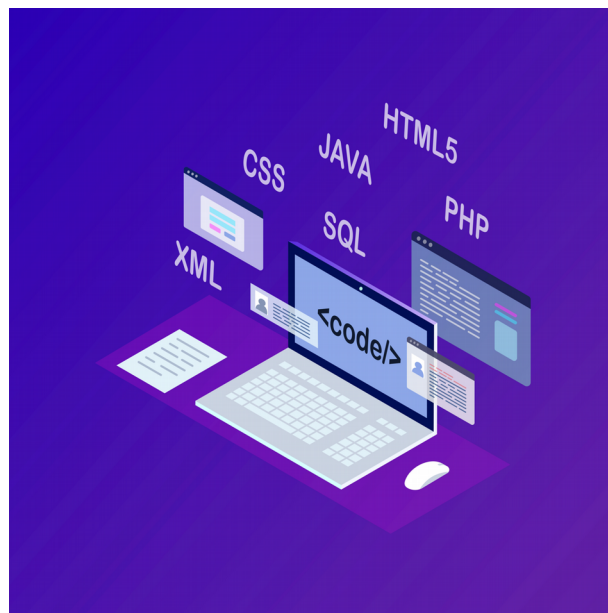
|   |    |
|---|----|
| INTRODUCCIÓN.....   | 3  |
| OBJETIVOS / CAPACIDADES.....  | 4  |
| PROYECTO DE LA UNIDAD.....  | 5  |
| 1. LENGUAJES EMBEBIDOS EN HTML. MECANISMOS DE<br>GENERACIÓN DE PÁGINAS WEB..... | 7  |
| 2. TECNOLOGÍA ASOCIADAS: PHP, ASP, JSP, SERVLETS, ENTRE<br>OTRAS.....           | 10 |
| Cuestionario.....   | 12 |
| 3. CONTENEDORES DE SERVLETS. EL CONTENEDOR WEB.....                             | 13 |
| 4. OBTENCIÓN DEL LENGUAJE DE MARCAS A MOSTRAR EN EL<br>CLIENTE.....             | 15 |
| 5. ETIQUETAS PARA INSERCIÓN DE CÓDIGO.....                                      | 18 |
| 6. BLOQUES DE CÓDIGOS.....  | 20 |
| 7. DIRECTIVAS.....  | 22 |
| Cuestionario.....   | 26 |
| 8. TIPOS DE DATOS. CONVERSIONES ENTRE TIPOS DE DATOS....                        | 27 |
| 9. VARIABLES. TIPOS. DECLARACIÓN, INICIALIZACIÓN Y ÁMBITO                       | 29 |
| 10. ÁMBITO DE UTILIZACIÓN DE LAS VARIABLES.....                                 | 31 |
| Cuestionario.....   | 34 |
| 11. OPERADORES DEL LENGUAJE. TIPOS.....   | 35 |
| RESUMEN.....  | 38 |
| RECURSOS PARA AMPLIAR.....  | 39 |
| BIBLIOGRAFÍA.....   | 40 |
| GLOSARIO.....   | 41 |

## INTRODUCCIÓN

Cualquier aplicación informática, ya sea para ordenador, dispositivo móvil o de otros tipos necesita ser programada antes de poderse ejecutar.

El elemento más importante de un programa es su **código fuente**, el cual dicta qué debe hacer la aplicación y cómo debe comportarse en cada momento.

En esta unidad veremos una introducción a los **lenguajes de programación** que tenemos a nuestra disposición para crear aplicaciones web y nos adentraremos en uno de ellos, **PHP**, que será el lenguaje que utilizaremos a lo largo del módulo.



En posteriores unidades se irán ampliando estos conceptos y añadiendo otros que nos permitirán realizar programas complejos con los que resolver diferentes problemas.



## OBJETIVOS / CAPACIDADES

*En esta unidad de aprendizaje, las capacidades que más se van a trabajar son:*

- ✓ Utilizar herramientas y lenguajes específicos, cumpliendo las especificaciones, para desarrollar e integrar componentes software en el entorno del servidor web.
- ✓ Programar y realizar actividades para gestionar el mantenimiento de los recursos informáticos.



## PROYECTO DE LA UNIDAD

Antes de empezar a trabajar el contenido, te presentamos la **actividad** que está relacionada con esta unidad de aprendizaje. Se trata de un **caso práctico** basado en una **situación real** con la que te puedes encontrar en tu puesto de trabajo. Con esta actividad se evaluará la puesta en práctica de los **criterios de evaluación** vinculados al resultado de aprendizaje que se trabaja en esta unidad. Para realizarla deberás hacer lo siguiente: lee el enunciado que te presentamos a continuación, dirígete al área general del módulo profesional, concretamente a la actividad de evaluación que se encuentra dentro de esta unidad, allí encontrarás todos los detalles sobre fecha y forma de entrega, objetivos... A lo largo de la unidad irás adquiriendo los conocimientos necesarios para ir elaborando este proyecto.

### Enunciado:

#### Instalación de XAMPP y ejecución de scripts.

Queremos ayudar a nuestra empresa a calcular las horas trabajadas de nuestros empleados.

Un empleado trabaja de lunes a viernes. Su jornada laboral semanal debería ser de 38 horas, pero por motivos ajenos al departamento de recursos humanos se observa que algunos trabajadores trabajan más horas y otros menos horas.

Crea un programa que tenga una variable con el nombre del trabajador. Crea también las variables necesarias para los días de la semana que sean laborables.

A cada día laborable asigne un valor numérico que representará las horas, y que puede contener decimales.

Haz los cálculos necesarios para que por pantalla salga un resumen de las horas trabajadas cada día por un trabajador, las horas totales trabajadas a lo largo de una semana, y la diferencia horaria respecto el convenio. Si el resultado es 0 se habrá cumplido el convenio. Si el resultado es  $>0$  el trabajador habrá hecho horas extra y si el resultado es  $<0$  el trabajador deberá horas a la empresa.

Calcula también el promedio de horas que el empleado trabaja cada día. Al haber una posible división entre 0, busca la directiva `display_errors` dentro del archivo `php.ini` y actívala.

# 1. LENGUAJES EMBEBIDOS EN HTML. MECANISMOS DE GENERACIÓN DE PÁGINAS WEB



*Como ya vimos en la unidad 1, el lenguaje de marcas HTML nos describe cómo se representará la página web a los ojos del cliente. Esta página web la podemos crear, sin duda, de forma manual, generando un contenido estático. Pero lo interesante es ver cómo se genera de forma dinámica.*

Uno de los **lenguajes de script** que genera **contenido dinámico es PHP**. Existen varias formas de incluir contenido en la página web a partir del resultado de la ejecución de código PHP:

- Echo.
- Print.
- Printf.

- ➔ La forma más habitual es usando **echo**, que no devuelve nada (void), y genera como salida el texto de los parámetros que recibe.

Sintaxis de echo: `echo(string $arg1, string $... = ?): void`

- ➔ La alternativa a echo es **print** y funciona de forma similar. La diferencia más notable entre print y echo, es que print sólo puede recibir un parámetro y devuelve siempre 1.

Sintaxis de print: `print(string $arg): int`



*A pesar de la sintaxis descrita, tanto print como echo no son realmente funciones, por lo que no es obligatorio poner paréntesis cuando se utilicen.*



➔ Finalmente existe una alternativa más compleja: **printf**. Esta función puede recibir varios parámetros, el primero de los cuales es siempre una cadena de texto que indica el formato que se ha de aplicar. Esa cadena debe contener un especificador de conversión por cada uno de los demás parámetros que se le pasen a la función, y en el mismo orden en que figuran en la lista de parámetros.

Posibles valores de formato:

- ✓ **%%** - Devuelve un signo de porcentaje.
- ✓ **%b** - Número binario.
- ✓ **%c** - El carácter según el valor ASCII.
- ✓ **%d** - Número decimal con signo (negativo, cero o positivo).
- ✓ **%e** - Notación científica usando minúsculas (por ejemplo, 1.2e+2).
- ✓ **%E** - Notación científica usando mayúsculas (por ejemplo, 1.2E+2).
- ✓ **%u** - Número decimal sin signo (igual o mayor que cero).
- ✓ **%f** - Número de coma flotante (consciente de la configuración local).
- ✓ **%F** - Número de coma flotante (no tiene en cuenta la configuración local).
- ✓ **%g** - Más corto que %e y %f.
- ✓ **%G** - Más corto que %E y %f.
- ✓ **%o** - Número octal.
- ✓ **%s** - Cadena.
- ✓ **%x** - Número hexadecimal (letras minúsculas).
- ✓ **%X** - Número hexadecimal (letras mayúsculas).

Valores de formato adicionales. Estos se colocan entre el % y la letra (ejemplo %.2f).

Sintaxis de printf: `printf(string $format, mixed $args = ?, mixed $... = ?): int`



Vídeo: funciones `print`, `echo` y `printf`



Visualiza el siguiente vídeo en el que verás distintos ejemplos de cómo usar `print`, `echo` y `printf`.

## 2. TECNOLOGÍA ASOCIADAS: PHP, ASP, JSP, SERVLETS, ENTRE OTRAS

Una de las **diferencias** más apreciables entre los lenguajes de programación web es la manera en que se ejecutan en el servidor web. Se pueden distinguir los siguientes **grupos**:

- Lenguajes de guiones (scripts): Perl, Python, PHP, ASP, etc.
- Lenguajes compilados a código nativo: CGI programados con C, Bash, Perl, etc.
- Lenguajes compilados a código intermedio: servlets, JSP y ASP.net.

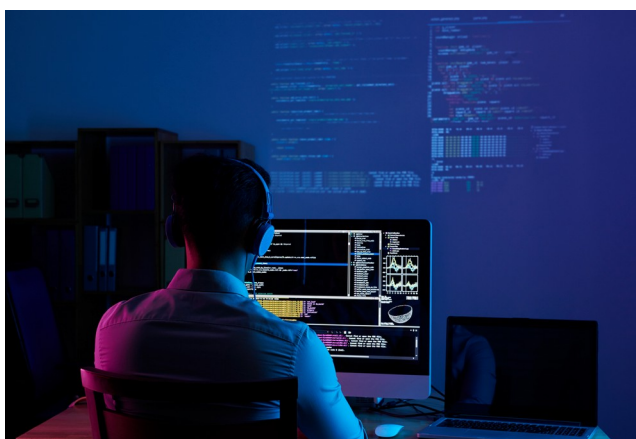


**PRESTA  
ATENCIÓN**

*No se debe confundir **ASP** con **ASP.net**. Ambos son tecnologías de Microsoft, pero ASP es un lenguaje de scripting y ASP.net es un framework donde los programas se suelen escribir en C#.*

La idea de los distintos tipos de lenguajes es la misma, **crear contenido web**, pero las formas de conseguirlo varían. Veamos algunas de sus **ventajas e inconvenientes**:

- ➔ Los **lenguajes de guiones** tienen la ventaja de que no es necesario compilar el código fuente original para ser ejecutados, lo que aumenta su portabilidad. Si se necesita realizar alguna modificación a un programa, se puede realizar con un sencillo editor. Por el contrario, el proceso de interpretación ofrece un peor rendimiento frente las otras alternativas.
- ➔ Los **lenguajes compilados a código nativo** son los de mayor velocidad de ejecución, pero tienen problemas en lo relativo a su integración con el servidor



web. Se escriben con lenguajes de propósito general que no están pensados para ejecutarse en el entorno de un servidor web. Por ejemplo, no se reutilizan los procesos para atender a varias peticiones. Además, los programas no son portables entre distintas plataformas de procesador.

- ➔ Los **lenguajes compilados a código intermedio** ofrecen un equilibrio entre las dos opciones anteriores. Su rendimiento es muy bueno y pueden portarse entre distintas plataformas en las que exista una implementación de la arquitectura (como un contenedor de servlets o un servidor de aplicaciones Java EE).

## Cuestionario

---



**Lee el enunciado e indica la opción correcta:**

La función echo de PHP devuelve:

- a. Un entero.
- b. Nada (void).
- c. Un booleano.



**Lee el enunciado e indica la opción correcta:**

La diferencia entre echo y printf es:

- a. Printf define el tipo de datos a mostrar.
- b. Printf puede recibir más parámetros que echo.
- c. Echo puede recibir más parámetros que printf.



**Lee el enunciado e indica la opción correcta:**

Perl, Python, PHP, ASP son:

- a. Lenguajes compilados a código nativo.
- b. Lenguajes de guiones (script).
- c. Lenguajes compilados a código intermedio.

### 3. CONTENEDORES DE SERVLETS. EL CONTENEDOR WEB



Un **servlet** es un programa Java que se ejecuta en un servidor Web y construye o sirve páginas web. De esta forma se pueden construir páginas dinámicas, basadas en diferentes fuentes variables: datos proporcionados por el usuario, fuentes de información variable (páginas de noticias, por ejemplo), o programas que extraigan información de bases de datos.

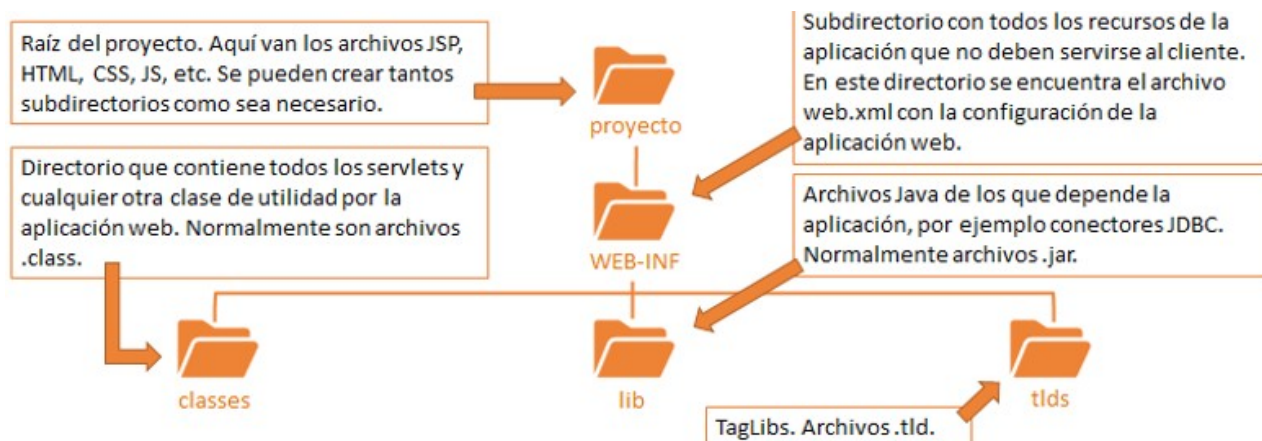
Comparado con un CGI, un **servlet** es más sencillo de utilizar, más eficiente (se arranca un hilo por cada petición y no un proceso entero), más potente y portable. Con los servlets podremos, entre otras cosas, **procesar**, **sincronizar** y **coordinar** múltiples **peticiones** de clientes, reenviar peticiones a otros servlets o a otros servidores, etc.

Normalmente al hablar de servlets se habla de **JSP** y viceversa, puesto que ambos conceptos están muy interrelacionados.

➔ Para trabajar con ellos se necesitan **tener presentes algunos recursos**:

- ✓ Un **servidor web** que dé soporte a servlets / JSP (contenedor de servlets y páginas JSP). Ejemplos de estos servidores son Apache Tomcat, Glassfish, Jetty, JRun, Java Web Server, BEA WebLogic, etc.
- ✓ Las **librerías (clases)** necesarias para trabajar con servlets / JSP. Al desarrollar nuestra aplicación, deberemos incluir las librerías necesarias en el CLASSPATH para que compilen los ficheros.





*Estructura de ficheros de un proyecto Java Web.*

- ➔ La **diferencia entre Servlets y JSP** es que los Servlets son clases que deben implementar la clase abstracta `HttpServlet`, en especial el método `doGet()` o `doPost()` y deben ser previamente compilados, mientras que los archivos JSP contienen código Java entre código HTML utilizando los símbolos `<%` y `%>`.
- ➔ Un servidor Web para Servlets y JSP como **Jakarta Tomcat** es una aplicación escrita en Java que mantiene una Java Virtual Machine en ejecución para compilar los archivos JSP y ejecutar Servlets.
- ➔ El **tiempo que demora** en la compilación inicial de un JSP es contrarrestado por su rápido tiempo de respuesta posterior ya que para procesar un requerimiento sólo tiene que levantar un proceso liviano o "thread" dentro de la misma JVM para ejecutar un archivo **.class** y no crear un proceso pesado como un intérprete de Perl para programas CGI.

Vídeo: desarrollo web con JSP y Servlets



Visualiza el siguiente vídeo en el que verás las diferencias en la ejecución de una página HTML y JSP.  
<https://www.youtube.com/embed/g2NOIQwl0g>

## 4. OBTENCIÓN DEL LENGUAJE DE MARCAS A MOSTRAR EN EL CLIENTE

Hemos explicado que un servidor web “sólo sabe” mandar contenido **estático**, es decir, un documento HTML con los posibles elementos multimedia asociados.

Este documento HTML estático final, quizás se haya construido de forma dinámica, por ejemplo, con PHP. Y la forma de construir este documento es **embebiendo etiquetas PHP dentro de una estructura HTML** o generando directamente una estructura HTML con PHP gracias a funciones como echo, print o printf.



*Recuerda que si un documento lleva código PHP hay que nombrarlo con la extensión **.php**, por ejemplo, **index.php**.*

Veamos un mismo resultado de documento web, primero generado con **etiquetas embebidas** y después creado íntegramente con **PHP**:

→ Web **generada con etiquetas** embebidas de PHP dentro de HTML:



```

<!DOCTYPE html>
<html>
<head>
<title>Ejemplo fecha</title>
</head>
<body>
<h1>Hoy es</h1>
<h2><?php date("d/m/Y") ?></h2>
</body>
</html>

```

➔ Web **generada con PHP** para crear la estructura HTML:

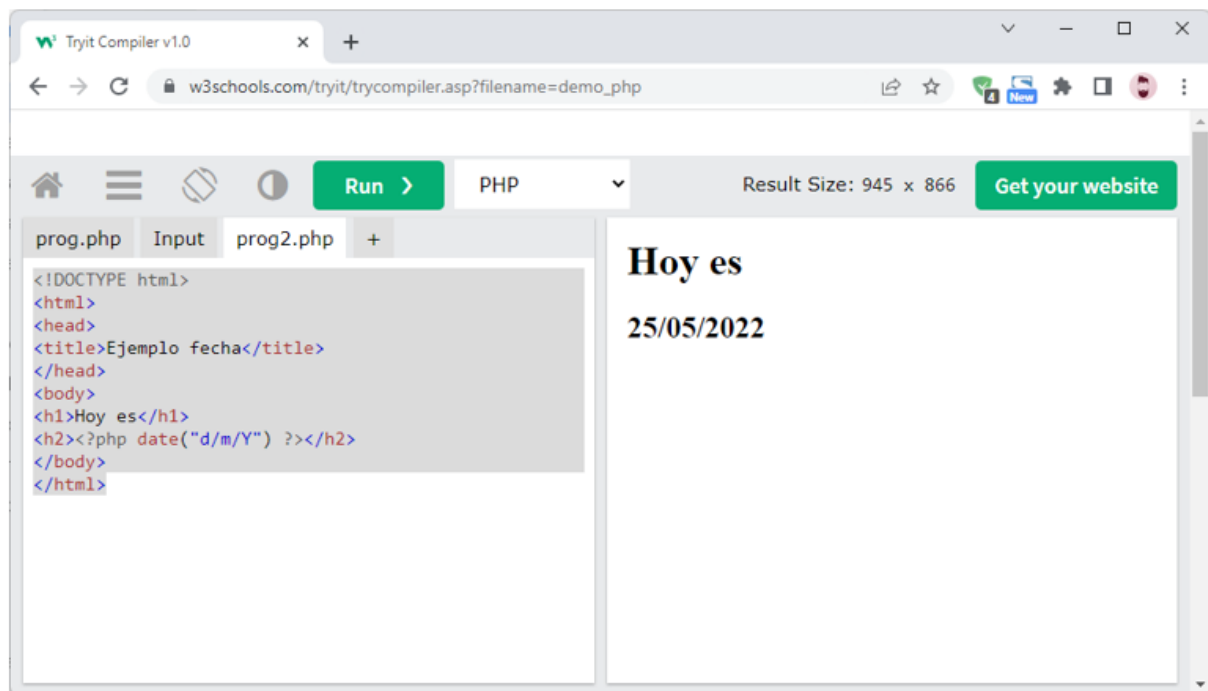
```

<?php
echo "<!DOCTYPE html>";
echo "<html>";
echo "<head>";
echo "<title>Ejemplo fecha</title>";
echo "</head>";
echo "<body>";
echo "<h1>Hoy es </h1>";
echo "<h2>" . date("d/m/Y") . "</h2>";
echo "</body>";
echo "</html>";

```

Fíjate que si en un documento sólo hay código PHP no hace falta la etiqueta de cierre `?>`

➔ En ambos casos el **resultado** generado es el mismo:

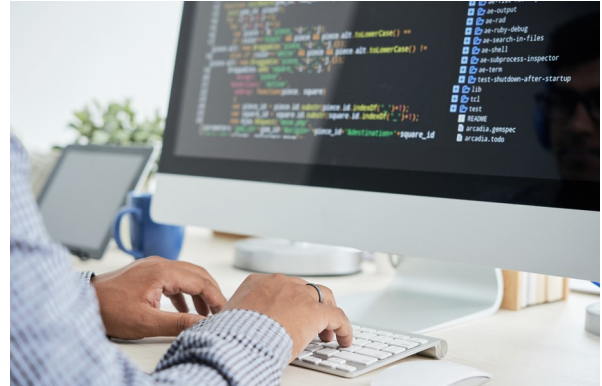


*Resultado de la ejecución del código de ejemplo.*

## 5. ETIQUETAS PARA INSERCIÓN DE CÓDIGO

A continuación, veremos algunas **características** de las **etiquetas en PHP**:

- Cuando PHP analiza un archivo busca **etiquetas de apertura y cierre**, que normalmente son `<?php` y `?>`, e indican a PHP cuando empezar y terminar de interpretar código. PHP también permite la etiqueta de apertura `<?`, pero se desaconseja.



Cualquier código fuera de las etiquetas de apertura y cierre es ignorado por el intérprete de PHP. Esto permite embeber PHP en HTML:

```
<p> Texto ignorado por el intérprete de PHP.</p>
<?php echo 'Texto que va a ser interpretado por PHP.'; ?>
<p> Texto también ignorado.</p>
```

Como en C o en Perl, PHP requiere que cada instrucción se termine con punto y coma ";" al final de cada sentencia.

- En PHP, las palabras clave (if, else, while, echo, etc.), clases, funciones y funciones definidas por el usuario no distinguen entre **mayúsculas y minúsculas**.

En el siguiente ejemplo, las tres declaraciones de echo a continuación son iguales y válidas:

```
<?php
ECHO "Hola mundo!<br>";
echo "  Hola mundo!<br>";
EcHo "  Hola mundo!<br>";
?>
```

A pesar de ello se recomienda seguir un mismo criterio a la hora de escribir código.



*Los nombres de las variables en PHP son case-sensitive.*

- ➔ A pesar de que PHP no distingue las mayúsculas de las minúsculas hay una **excepción: las variables.**

En el siguiente ejemplo sólo la primera declaración mostrará el valor de la variable `$color`. Esto se debe a que `$color`, `$COLOR` y `$coLOR` se tratan como tres variables diferentes:

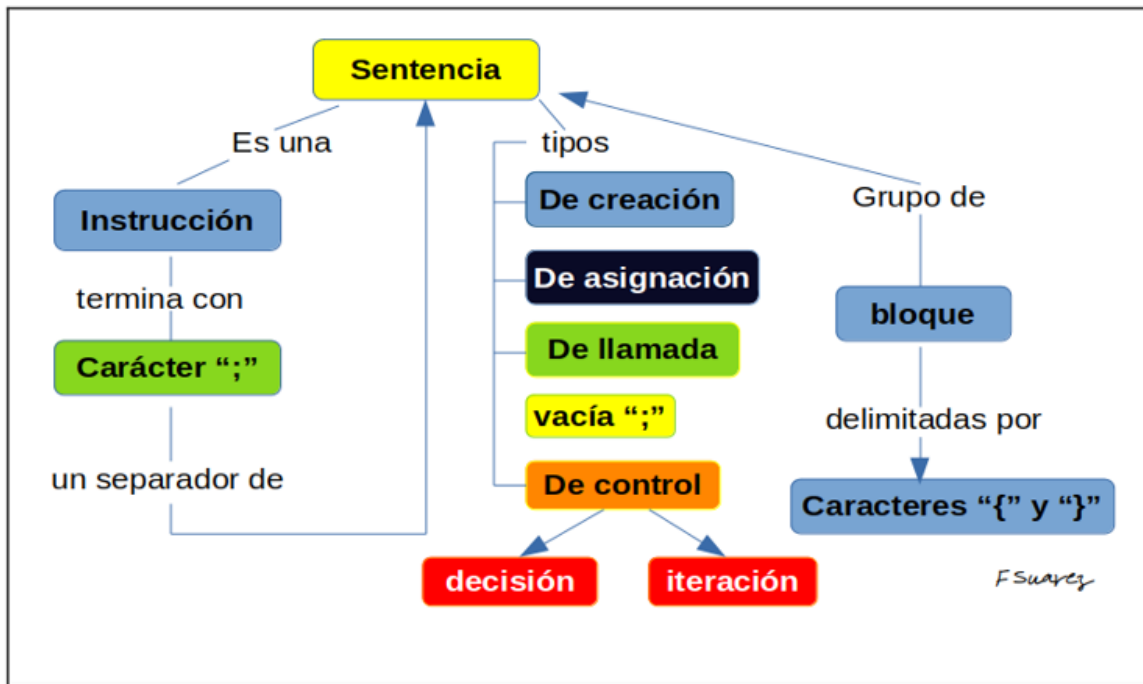
```
<?php
$color = "rojo";
echo "Mi coche es " . $color . "<br>";
echo "Mi televisor es " . $COLOR . "<br>";
echo "Mi barco es " . $coLOR . "<br>";
?>
```

## 6. BLOQUES DE CÓDIGOS

Cualquier aplicación básica consta de una serie de **bloques o partes** bastantes bien definidas:

- ➔ **Declaración**: se crearán las variables necesarias para almacenar los datos referentes al problema.
- ➔ **Entrada**: se pedirá al usuario, ya sea mediante consola o interfaz gráfica los datos.
- ➔ **Resolución**: se implementarán los algoritmos necesarios para la resolución del problema.
- ➔ **Salida**: se presentarán al usuario los resultados obtenidos.

A pesar de que la sintaxis puede variar entre distintos lenguajes de programación, la idea de **bloque de código** es la misma para todos.



Estructura de un bloque de código. Fuente:  
<https://github.com/cisnefy/EjerciciosJAVA/wiki/Qu%C3%A9-es-una-sentencia-en-java-y-bloques-de-c%C3%B3digo>. Esta imagen se reproduce acogiendo al derecho de cita o reseña (art. 32 LPI), y está excluida de la licencia por defecto de estos materiales.



Cuanto más complejo es el problema a resolver más lo será también la manera de resolverlo, por lo que estos bloques pueden repetirse a lo largo del programa, y no siempre necesariamente en el mismo orden.

## 7. DIRECTIVAS

Algunos **parámetros de configuración** influyen de forma crucial en el rendimiento de las aplicaciones montadas sobre PHP. Estos parámetros de configuración se llaman **directivas**.

Estas directivas de configuración se introducen en el archivo de configuración de PHP (**php.ini**).

### ➔ Configuración PHP:

¿Recuerdas que en la actividad 3 de la UD1 mostrábamos toda la configuración de PHP con la función `phpinfo()`? Salía algo así por pantalla:





|                |   |  |
|----------------|---|--|
| HTTPS_PROXY    | Yes   |  |
| MULTI_SSL      | No  |  |
| BROTLI         | No  |  |
| Protocols      | dict, file, ftp, ftps, gopher, gophers, http, https, imap, imaps, ldap, ldaps, mqtt, pop3, pop3s, rtsp, scp, sftp, smb, smbs, smtp, smtps, telnet, tftp |  |
| Host           | x86_64-pc-win32   |  |
| SSL Version    | OpenSSL/1.1.1n  |  |
| ZLib Version   | 1.2.12  |  |
| libSSH Version | libssh2/1.10.0  |  |

| Directive   | Local Value                            | Master Value                           |
|-------------|--|--|
| curl.cainfo | C:\xampp\apache\bin\curl-ca-bundle.crt | C:\xampp\apache\bin\curl-ca-bundle.crt |

**date**

|                                   |               |  |
|-----------------------------------|---------------|--|
| date/time support                 | enabled       |  |
| timelib version                   | 2021.11       |  |
| "Olson" Timezone Database Version | 2022.1        |  |
| Timezone Database                 | internal      |  |
| Default timezone                  | Europe/Berlin |  |

| Directive              | Local Value   | Master Value  |
|------------------------|---------------|---------------|
| date.default_latitude  | 31.7667       | 31.7667       |
| date.default_longitude | 35.2333       | 35.2333       |
| date.sunrise_zenith    | 90.833333     | 90.833333     |
| date.sunset_zenith     | 90.833333     | 90.833333     |
| date.timezone          | Europe/Berlin | Europe/Berlin |

**dom**

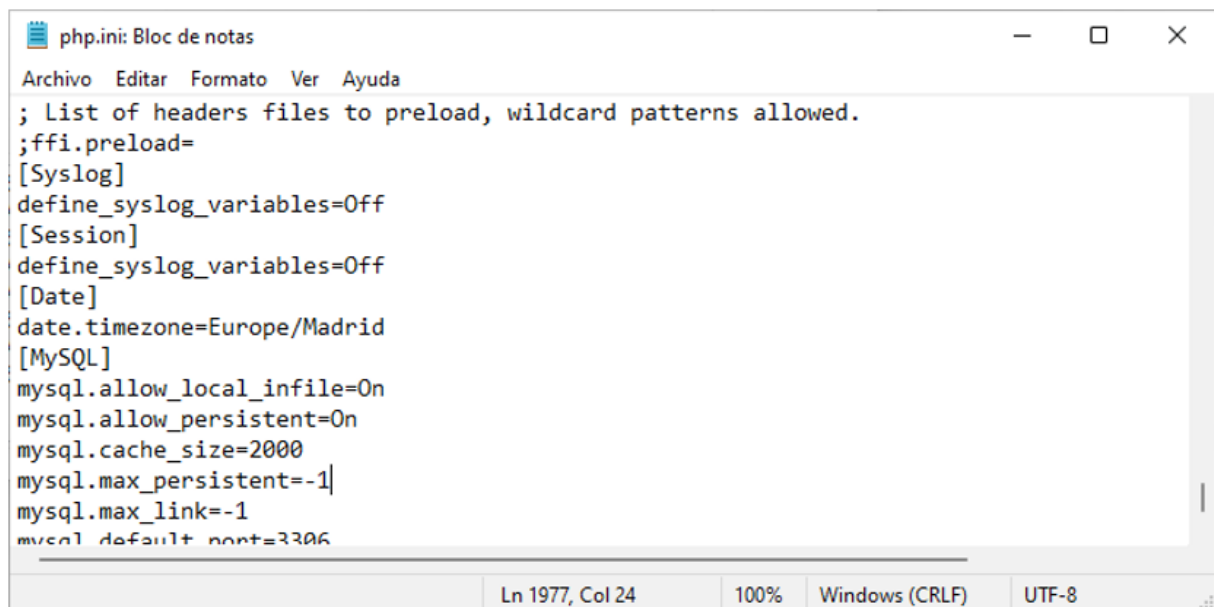
|                     |          |  |
|---------------------|----------|--|
| DOM/XML             | enabled  |  |
| DOM/XML API Version | 20031129 |  |

## Configuración de PHP

Una de las configuraciones que apreciamos es la zona horaria. Por defecto está establecida en Europe/Berlin. Si quisiéramos poner la zona horaria de España deberíamos editarlo en el archivo **php.ini**. Este archivo se encuentra en la ruta donde se ha instalado PHP. Si usas Windows estará probablemente en c:/xampp/php/php.ini.

### ➔ Edición de php.ini:

Si cambiamos la línea del timezone tendremos correctamente configurada nuestra zona horaria:



```
php.ini: Bloc de notas
Archivo  Editar  Formato  Ver  Ayuda
; List of headers files to preload, wildcard patterns allowed.
;ffi.preload=
[Syslog]
define_syslog_variables=Off
[Session]
define_syslog_variables=Off
[Date]
date.timezone=Europe/Madrid
[MySQL]
mysql.allow_local_infile=On
mysql.allow_persistent=On
mysql.cache_size=2000
mysql.max_persistent=-1
mysql.max_link=-1
mysql.default_port=3306
Ln 1977, Col 24    100%    Windows (CRLF)    UTF-8
```

Edición de *php.ini*



*Dentro de un script en PHP también se pueden definir directivas que afectarán a ese código en concreto. Y con Apache, también es posible definir directivas de configuración de PHP en el archivo de configuración de Apache (por ejemplo, *httpd.conf*) o en un archivo *.htaccess*.*

### Actividad de aprendizaje 1: directivas

Crea dos archivos **.php** y guárdalos en la carpeta pública de tu servidor web. En ambos archivos se intentará dividir un número entre cero (algo que debería ser erróneo).

Copia este script en el primer archivo:

```
<?php
    error_reporting(E_ALL);
    echo 10/0;
?>
```

Ahora copia este otro script en el segundo archivo:

```
<?php
    error_reporting(0);
    echo 10/0;
?>
```

Comparte en el foro los resultados de ambas ejecuciones y la conclusión sobre qué crees que pasa. Discute con tus compañeros cuando nos puede interesar tener las directivas configuradas como en el primer script y cuando puede interesar tenerlas como en el segundo script.

## Cuestionario

---



**Lee el enunciado e indica la opción correcta:**

Un servlet es:

- a. Un programa en Java que se ejecuta en un servidor web.
- b. Un programa en PHP que se ejecuta en un servidor web.
- c. Un programa en Java que se ejecuta en un cliente.



**Lee el enunciado e indica la opción correcta:**

PHP es case-sensitive respecto:

- a. Las palabras reservadas.
- b. Las variables.
- c. PHP no es case-sensitive.



**Lee el enunciado e indica la opción correcta:**

Las directivas en PHP:

- a. Son bloques de código.
- b. Son configuraciones del lenguaje.
- c. PHP no tiene directivas.

## 8. TIPOS DE DATOS. CONVERSIONES ENTRE TIPOS DE DATOS

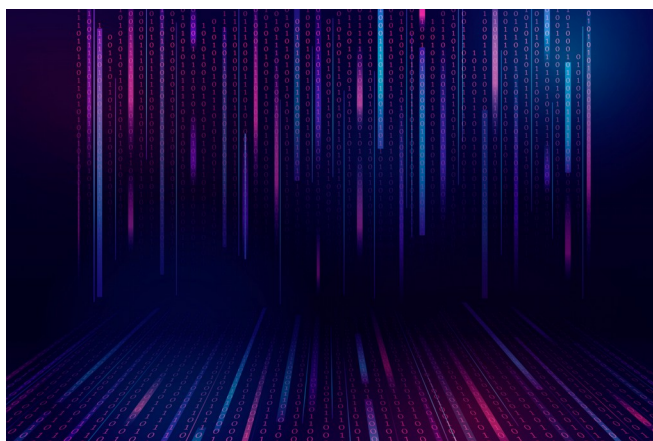


*PHP (a diferencia de Java, por ejemplo) no tiene variables tipadas (algunos autores dicen que PHP tiene un tipado dinámico). Esto significa que no es necesario declarar de qué tipo debe ser una variable y que además puede cambiar de tipos a lo largo de una ejecución.*

PHP admite **diez tipos primitivos**:

### ➔ Cuatro tipos escalares:

- ✓ Boolean.
- ✓ Integer.
- ✓ Float (número de punto flotante, también conocido como double).
- ✓ String.



### ➔ Cuatro tipos compuestos:

- ✓ Array.
- ✓ Object.
- ✓ Callable.
- ✓ Iterable.

### ➔ Y, finalmente, **dos tipos especiales**:

- ✓ Resource.
- ✓ Null.

El tipo de una variable usualmente no lo declara el programador; al contrario, es decidido en tiempo de ejecución por PHP dependiendo del contexto en el que se emplea dicha variable.



*Para forzar la conversión de una variable a un cierto tipo, se puede amoldar la variable o usar la función `settype()` con ella.*

## 9. VARIABLES. TIPOS. DECLARACIÓN, INICIALIZACIÓN Y ÁMBITO

Todo programa que pretenda resolver un problema requiere de una serie de datos. Éstos se almacenarán en lo que se denomina variables. En PHP, una variable comienza con el \$signo, seguido del nombre de la variable:

```
<?php
$txt = "Hola mundo!";
$x = 5;
$y = true;
?>
```

Después de la ejecución de las instrucciones anteriores, la variable `$txt` contendrá el valor `Hola mundo!`, la variable `$x` contendrá el valor `5` y la variable `$y` contendrá el valor booleano `true`.

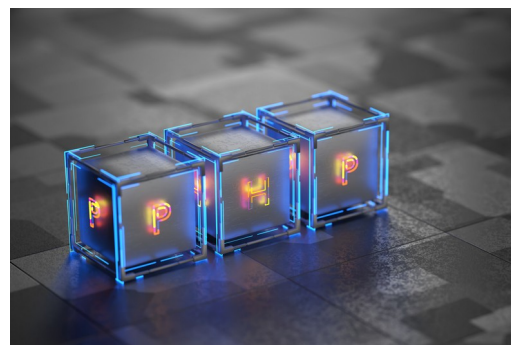


*Cuando se asigna un valor de texto a una variable, deben colocarse comillas alrededor del valor. A diferencia de otros lenguajes de programación, PHP no tiene ningún comando para declarar una variable. Se crea en el momento en que le asigna un valor por primera vez.*

Una variable puede tener un **nombre corto** (como `x` e `y`) o un nombre más **descriptivo** (edad, nombre del coche, volumen\_total).

Veamos las **reglas** para **variables de PHP**:

- ➔ **I.** Una variable comienza con el signo `$`, seguido del nombre de la variable.
- ➔ **II.** Un nombre de variable debe comenzar con una letra o el carácter de subrayado (`_`).





- ➔ **III.** Un nombre de variable no puede comenzar con un número.
- ➔ **IV.** Un nombre de variable solo puede contener caracteres alfanuméricos y guiones bajos (Az, 0-9 y \_).
- ➔ **V.** Los nombres de las variables distinguen entre mayúsculas y minúsculas (`$edad` y `$EDAD` son dos variables diferentes).

## 10. ÁMBITO DE UTILIZACIÓN DE LAS VARIABLES



En PHP, las variables se pueden **declarar en cualquier parte del script**. El alcance de una variable es la parte del script donde se puede hacer referencia o utilizar la variable.

PHP tiene **tres ámbitos** variables diferentes:

- Local.
- Global.
- Estático.

### → Global en comparación con local.

Ejemplo de ámbito (scope) global en comparación con el ámbito local:



```
function saludar() {  
    $saludo = "Hola"; // Función de ámbito local  
    print $saludo;  
}  
  
$saludo = "Buenos días"; // Función de ámbito global  
  
saludar(); // Devuelve: Hola  
print $saludo; // Devuelve: Buenos días
```

La ejecución del script empieza en `$saludo = "Buenos días"`, y después llama a la función `saludar()`. Esta función establece `$saludo` a "Hola" y lo imprime, y después devuelve el control al script principal donde se imprime `$saludo`. Puede comprobarse que sus valores son diferentes y no se afecta el uno al otro debido al ámbito de ejecución.

Normalmente, cuando se completa/ejecuta una función, se eliminan todas sus variables. Sin embargo, a veces queremos que no se elimine una variable local ya que puede sernos útil para otro trabajo.

### → **Ámbito estático.**

Para hacer esto, se usa la palabra reservada `static` cuando se declara la variable por primera vez:

```
<?php
function incremento() {
    static $x = 0;
    echo $x;
    $x++;
}
incremento();
incremento();
incremento();
?>
```

Cada vez que se llame a la función `incremento()`, la variable `$x` aún tendrá la información que contenía desde la última vez que se llamó a la función. Por lo tanto, el resultado por pantalla sería: 0, 1 y 2, ya que se ha llamado 3 veces.

---

### Actividad de aprendizaje 2: variables y tipos de datos

Escribe el siguiente código:

```
$a = "Hola";  
$b = 100;  
$c = 10.55;  
$d = true;  
$e = array("ASIX", "DAM", "DAW");  
$f = ["ASIX", "DAM", "DAW"];  
$g = new Car();  
$h = null;
```

Usando la función `var_dump($a)` y la sentencia `echo gettype($a)` para cada una de las variables comparte en la wiki qué tipo de dato contiene cada variable. También encontrarás algún error (deliberado). Trata de entender por qué pasa discutiéndolo con tus compañeros.

## Cuestionario

---



**Lee el enunciado e indica la opción correcta:**

Los tipos de datos escalares en PHP son:

- a. Null, resource.
- b. Boolean, integer, float, string.
- c. Array, object, callable, iterable.



**Lee el enunciado e indica la opción correcta:**

Para forzar la conversión de una variable a un cierto tipo, se puede amoldar la variable o usar la función:

- a. Settype().
- b. Gettype().
- c. Converse().



**Lee el enunciado e indica la opción correcta:**

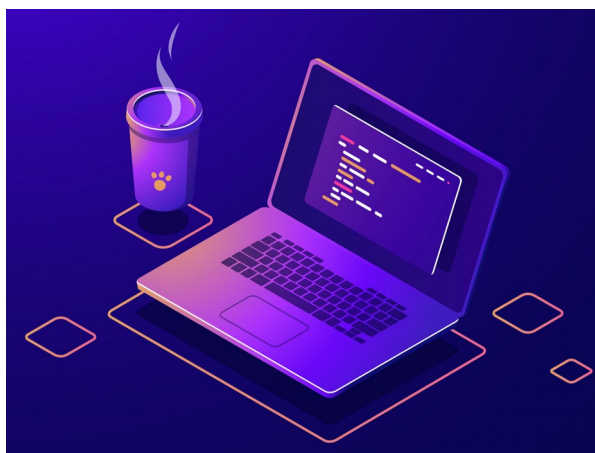
PHP tiene tres ámbitos variables diferentes:

- a. Interno, externo y superficial.
- b. Comunitario, regional y provincial.
- c. Local, global y estático.

## 11. OPERADORES DEL LENGUAJE. TIPOS

Los operadores se utilizan para realizar operaciones en variables y valores. PHP divide a los operadores en los siguientes grupos:

- Operadores **aritméticos**.
- Operadores de **asignación**.
- Operadores de **comparación**.
- Operadores de **incremento/decremento**.
- Operadores **lógicos**.
- Operadores de **cadena**.
- Operadores de **matriz**.
- Operadores de **asignación condicional**.



Veamos unas tablas con los **operadores más usuales**:

### ➔ Operadores aritméticos:

| Ejemplo       | Nombre         | Resultado                                     |
|---------------|----------------|---|
| $\$a + \$b;$  | Suma           | Suma de $\$a$ y $\$b$                         |
| $\$a - \$b;$  | Resta          | Diferencia entre $\$a$ y $\$b$                |
| $\$a * \$b;$  | Multiplicación | Producto de $\$a$ y $\$b$                     |
| $\$a / \$b;$  | División       | Cociente de $\$a$ entre $\$b$                 |
| $\$a \% \$b;$ | Módulo         | Resta o residuo de $\$a$ dividido entre $\$b$ |

### ➔ Operadores de asignación:

| Ejemplo                          | Nombre                     | Resultado                                    |
|----------------------------------|----------------------------|--|
| <code>\$a = \$b;</code>          | Asigna                     | \$a toma el valor de \$b                     |
| <code>\$a += \$b;</code>         | Suma y asigna              | Es lo mismo que <code>\$a = \$a + \$b</code> |
| <code>\$a -= \$b;</code>         | Resta y asigna             | Es lo mismo que <code>\$a = \$a - \$b</code> |
| <code>\$a *= \$b;</code>         | Multiplica y asigna        | Es lo mismo que <code>\$a = \$a * \$b</code> |
| <code>\$a /= \$b;</code>         | Divide y asigna            | Es lo mismo que <code>\$a = \$a / \$b</code> |
| <code>\$a %= \$b;</code>         | Calcula el módulo y asigna | Es lo mismo que <code>\$a = \$a % \$b</code> |
| <code>\$cadena .= "Juan";</code> | Concatena y asigna         | Concatena "Juan" al final de \$cadena        |

### ➔ Operadores de comparación:

| Ejemplo                        | Nombre            | Resultado  |
|--------------------------------|-------------------|--|
| <code>\$a == \$b;</code>       | Igual             | Devuelve true si \$a es igual a \$b                            |
| <code>\$a === \$b;</code>      | Idéntico          | Devuelve true si \$a es igual a \$b y son del mismo tipo       |
| <code>\$a != \$b;</code>       | No igual          | Devuelve true si \$a no es igual a \$b                         |
| <code>\$a &lt;&gt; \$b;</code> | No igual          | Devuelve true si \$a no es igual a \$b                         |
| <code>\$a !== \$b;</code>      | No idéntico       | Devuelve true si \$a no es igual a \$b o no son del mismo tipo |
| <code>\$a &gt; \$b;</code>     | Mayor que         | Devuelve true si \$a es mayor que \$b                          |
| <code>\$a &lt; \$b;</code>     | Menor que         | Devuelve true si \$a es menor que \$b                          |
| <code>\$a &gt;= \$b;</code>    | Mayor o igual que | Devuelve true si \$a es mayor o igual que \$b                  |
| <code>\$a &lt;= \$b;</code>    | Menor o igual que | Devuelve true si \$a es menor o igual que \$b                  |



Puedes encontrar toda la lista de operadores en "Listado de operadores en PHP" del apartado recursos para ampliar.



### Vídeo: uso de variables y operadores



*Visualiza el siguiente vídeo que trata sobre el uso de variables y operadores.*

### Actividad de aprendizaje 3: algoritmo



*La actividad se elaborará en parejas.*

Una vez instalado el entorno de desarrollo, y a modo de prueba te juntan con otro nuevo empleado para que entre los dos resolváis vuestro primer problema.

Se pide escribir un algoritmo que dados 3 números enteros que representan horas, minutos y segundos (guardados en 3 variables), dé el resultado equivalente en segundos.

Por ejemplo, 1 horas, 10 minutos y 6 segundos son 4206 segundos totales.

Ponte de acuerdo con algún compañero y entregad el archivo **.php** con el código creado al espacio habilitado para ello.

## RESUMEN

En esta unidad hemos podido hacer una pequeña introducción a los **lenguajes de programación** del lado del servidor y ya hemos entrado de lleno en uno de ellos: **PHP**.

Hemos visto aspectos básicos como la **creación de un primer script**, con todas sus fases, desde el análisis hasta la ejecución.

Además, nos hemos podido familiarizar con los primeros **conceptos** a tener en cuenta a la hora de crear un programa informático, como son los bloques de código, las palabras reservadas, las variables y sus operadores.

Con los conocimientos adquiridos hasta ahora ya somos capaces de **crear los primeros algoritmos** que se comportan como una incipiente aplicación web.

En próximas unidades ampliaremos estos conceptos con otros que nos permitirán realizar programas cada vez más complejos.

## RECURSOS PARA AMPLIAR



### PÁGINAS WEB

- Tutorial sobre ASP de la plataforma w3schools:  
<https://www.w3schools.com/asp/default.asp> [Consulta noviembre 2022].
- Listado de operadores en PHP:  
[https://www.w3schools.com/php/php\\_operators.asp](https://www.w3schools.com/php/php_operators.asp) [Consulta noviembre 2022].



## BIBLIOGRAFÍA



### PÁGINAS WEB

- Referencia oficial de PHP: <https://www.php.net/> [Consulta noviembre 2022].
- Tecnologías de Java: <https://www.oracle.com/java/technologies/> [Consulta noviembre 2022].



## GLOSARIO

- **ASP**: Active Sever Pages.
- **Directiva**: parámetro de configuración.
- **JMV**: Java Virtual Machine (máquina virtual de Java).
- **JSP**: Java Server Pages.
- **PHP**: PHP Hypertext Preprocessor (acrónimo recursivo).
- **SCOPE**: ámbito de una variable. Delimita el uso de una variable dentro de un script.
- **Servlet**: los servlets son módulos java que nos sirven para extender las capacidades de los servidores web.

