

# Módulo 2: Inserción de Código Página Web

---

28 Septiembre 2022



Educar en la Verdad  
para ser libres



## Necesidades del Módulo 2: Inserción de código en páginas web

- 2.1 Lenguajes embebidos en HTML
- 2.2 Tecnologías asociadas: PHP, ASP, JSP, Servlets, entre otras
- 2.3 Contenedores de servlets
- 2.4 Obtención del lenguaje de marcas a mostrar en el cliente
- 2.5 Etiquetas para inserción de código
- 2.6 Bloques de código
- 2.7 Directivas
- 2.8 Tipos de datos. Conversiones entre tipos de datos
- 2.9 Variables
- 2.10 Ámbito de utilización de las variables



## 0. Introducción

En esta unidad aprenderemos cómo empezar a escribir un programa usando los lenguajes de código embebido en HTML, lo cual significa que escribiremos nuestros programas dentro de una página web usando una serie de marcas especiales que le indican al servidor qué código tiene que ejecutar y qué código no.

Los lenguajes de código embebido nos van a permitir la conversión de páginas web estáticas en páginas web dinámicas.

Los principales lenguajes de código embebido para utilizar en HTML son PHP, JSP (Java) y ASP (JavaScript y VBScript). Explicaremos las principales características de estos lenguajes y así poder decidir cuál usar en cada momento

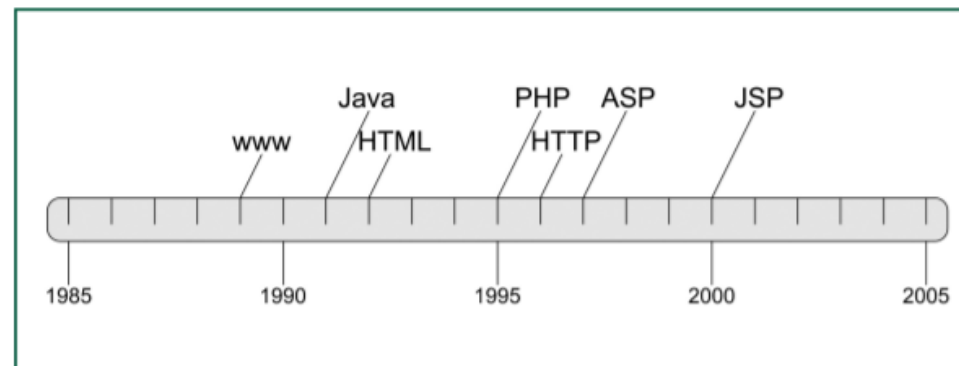


## 2.1. Lenguajes embebidos en HTML: PHP

HTML es el lenguaje que se utiliza para generar páginas web. Pero también hemos visto que estas páginas web son estáticas. Para poder hacerlas dinámicas necesitamos de lenguajes de programación en entorno de servidor.

Los lenguajes embebidos que están más extendidos son PHP, ASP y JSP. Para ser más precisos, ASP no es un lenguaje, sino el nombre que recibe la tecnología (Active Server Pages). Realmente los lenguajes que puede utilizar ASP son dos: JavaScript y VBScript.

La década de los 90 fue muy prolífica en cuanto a la generación del nuevo entorno web y de posibles lenguajes para su explotación (Figura 2.2).



**Figura 2.2**  
Cronología histórica web.



## 2.2. Tecnologías asociadas: PHP, ASP, JSP, Servlets, entre otras

### PHP (Hypertext Preprocessor)

Nació en el año 1995 y se ofrece de forma libre desde su página web oficial mediante una licencia de uso estilo GPL. Sus principales características son:

- Lenguaje imperativo con orientación a objetos.
- Manejo de excepciones.
- Integración natural con bases de datos MySQL, PostgreSQL o SQLite.
- Soporte de XML (eXtensive Markup Language) y DOM (Document Object Model).
- Permite nuevos módulos o extensiones.



## 2.2. Tecnologías asociadas: PHP, ASP, JSP, Servlets, entre otras

- Facilidad de uso y de aprendizaje.
- Instalable en cualquier sistema operativo moderno.

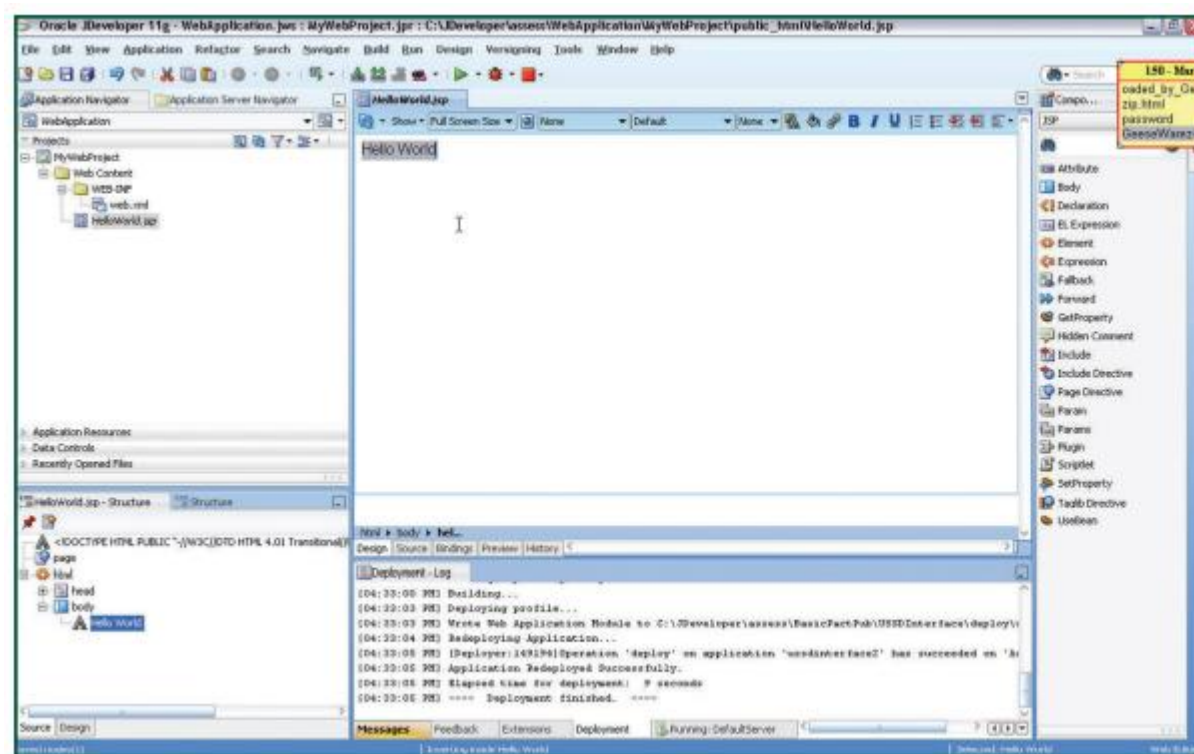
Y otras características que pueden ser un inconveniente son:

- Al ser un lenguaje interpretado, las páginas PHP pueden ser más lentas que en lenguajes compilados.
- No es rígido en la tipología de las variables.
- Su orientación a objeto no es nativa.

## 2.2. Tecnologías asociadas: PHP, ASP, JSP, Servlets, entre otras

### JSP (Java Server Pages) y Servlet (Server Applet)

En el año 2000 se creó la primera especificación de JSP. Es un lenguaje compilado automáticamente la primera vez que se ejecuta una página y utiliza la tecnología de Servlet para su ejecución (Figura 2.4).



**Figura 2.4**  
Ejemplo de proyecto en JSP.



## 2.2. Tecnologías asociadas: PHP, ASP, JSP, Servlets, entre otras

Como características principales tenemos:

- Es un lenguaje compilado y basado en Java; por tanto, es rápido y utiliza toda la potencia que proporciona Java.
- JSP y Servlet comparten el entorno de ejecución.
- Java es un lenguaje orientado a objetos de forma nativa.
- Es independiente de la plataforma en que se ejecuta y se integra fácilmente en dispositivos móviles

Aunque también tiene características que pueden ser inconvenientes:

- Es un lenguaje estricto y complejo; por tanto, más difícil de usar y aprender.
- Su portabilidad entre plataformas y versiones es muy grande pero no es total como se diseñó en un principio.







## 2.3. Contenedores de servlets: Java

### Contenedores de servlets: Java

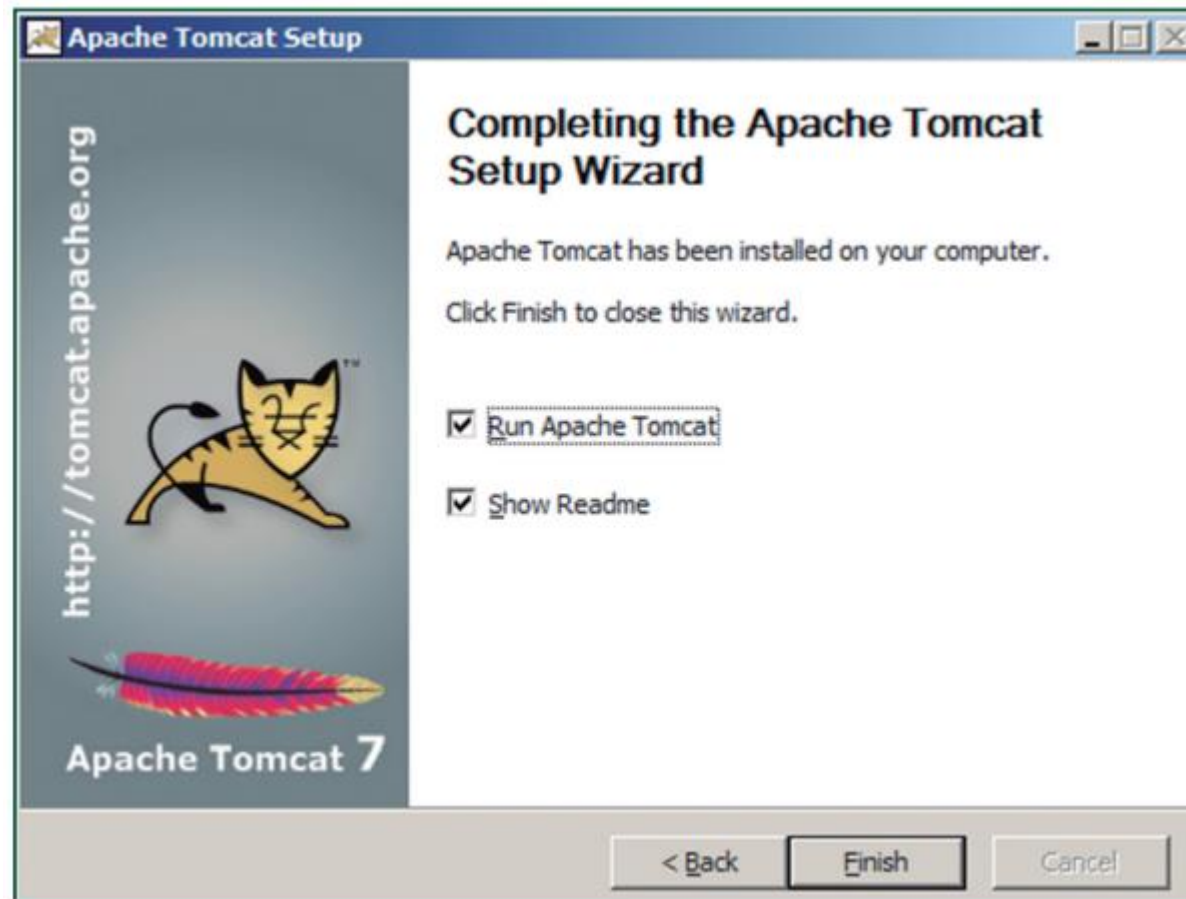
Un Servlet es un programa hecho en Java pero que debe cumplir ciertas condiciones:

- Debe estar basado en una clase llamada `HttpServlet` (extends `HttpServlet`).
- Los parámetros de entrada se reciben mediante la clase `HttpServletRequest`.
- La salida de resultados se realizará usando la clase `HttpServletResponse`.
- Debe implementar al menos el método `doGet()` o el método `doPost()`.

Veamos un ejemplo de cómo escribir a **HolaMundoServlet.java**: Pero, para poder ejecutar este Servlet, debemos tener no solamente un servidor web, sino también un servidor de aplicaciones con los módulos Java activados. Ambas cosas las podemos tener si, por ejemplo, instalamos Apache Tomcat.



## 2.3. Contenedores de servlets: Java



## 2.3. Contenedores de servlets: Java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class HolaMundoServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hola Mundo usando Servlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hola Mundo!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

## 2.4 Obtención del lenguaje de marcas a mostrar en el cliente

Cuando recibe una petición URL, el servidor realiza un proceso previo de análisis de dicha URL figura 2,5 . Como ya sabéis una URL se escribe de esta forma:

`protocolo://nombreServidorWeb:puerto/ruta/pagina?parametro1=valor1&parametro2=valor2`

El análisis del servidor descompone la URL y encuentra los elementos descritos: ruta, página y parámetros.

La **ruta** es una carpeta del servidor donde se almacenan, entre otras cosas, las páginas web, las imágenes, los vídeos, etc. Cuando solicitamos la **página**, el servidor la busca en la ruta que se le ha indicado anteriormente, la lee y la procesa utilizando los **parámetros** que se han indicado.

Procesar es la ejecución de dicha página web, si es que debe ser ejecutada, puesto que si es solamente una página web con HTML, no hay nada que procesar y se debe enviar tal cual al navegador del cliente



## 2.4 Obtención del lenguaje de marcas a mostrar en el cliente

En este último caso el servidor web únicamente realizará la tarea de dispensador de páginas y nada más, puesto que no será necesaria ninguna ejecución de código embebido.

En el resto de casos el servidor web indicará al servidor de aplicaciones que se ha realizado una petición a cierta página y con ciertos parámetros.



**Figura 2.5**  
Ejemplo de URL que utiliza el  
protocolo de acceso http.

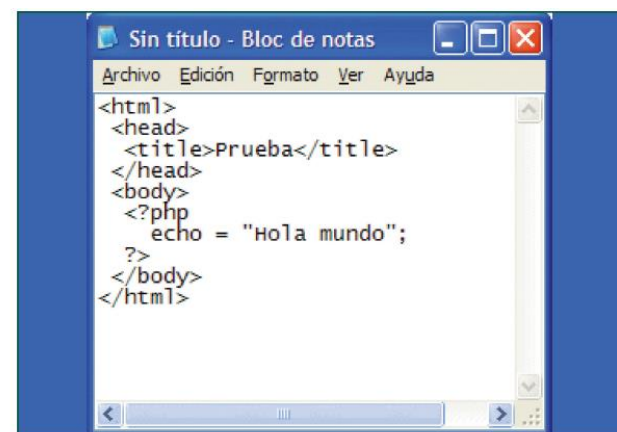


## 2.5 Etiquetas para inserción de código

Todas las marcas tienen una característica en común; hay dos marcas: la marca de inicio de código y la marca de final de código.

Recordad que cada lenguaje utiliza un tipo de marcas diferentes ejemplo: (Figura 2.6):

- PHP. La marca de inicio es “<?php” y la marca de final es “?>”.
- ASP. La marca de inicio es “<%” y la marca de final es “%>”.
- JSP. La marca de inicio también es “<%” y la marca de final también es “%>”



The screenshot shows a Notepad window titled "Sin título - Bloc de notas". The menu bar includes "Archivo", "Edición", "Formato", "Ver", and "Ayuda". The text content is as follows:

```
<html>
<head>
  <title>Prueba</title>
</head>
<body>
  <?php
    echo = "Hola mundo";
  ?>
</body>
</html>
```

**Figura 2.6**  
Ejemplo de código embebido  
PHP en HTML.

## 2.6 Bloques de código

Los bloques de código son los trozos de código de nuestro programa que colocamos dentro de una página HTML usando las marcas de inicio y de final de código.

En este ejemplo puedes observar que se han escrito dos bloques de código en esta página PHP: bloques.php

```
<html>
<head>
<title>Hola Mundo usando dos bloques de código</title></head>
<body>
<h1><?php echo `;Hola`; ?> <?php echo `Mundo!`; ?></h1>
</body>
</html>
```



## 2.6 Bloques de código

Los bloques de código pueden ser todo lo grandes que se desee que sean y una página web puede contener uno o muchos bloques de código.

Los servidores de aplicaciones ejecutan las páginas de forma secuencial. Esto significa que se ejecutan sentencia a sentencia empezando por el principio de la página y en el orden en el que se lee la página. De todos modos, aunque la ejecución sea siempre secuencial, es posible realizar alteraciones en la línea de código utilizando sentencias condicionales, iterativas o de llamadas a métodos o funciones. Esto lo vamos a aprender en la siguiente unidad.

Cuando se ejecuta una página, las marcas HTML únicamente se leen y se almacenan tal cual para generar el resultado final que se enviará al cliente. Los lenguajes orientados a objetos también se ejecutan de forma secuencial con

la excepción del punto de inicio de la ejecución que viene definido por la clase de nivel superior





## 2.7 Directivas

Podemos hacer que una misma página web se ejecute de forma diferente solamente modificando ciertas directivas.

Una directiva es una instrucción incluida en la página web que no ejecuta código, sino que modifica el comportamiento de dicha ejecución.

La sintaxis de una directiva es la siguiente:

```
<%@ directiva atributo=valor %>
```

## 2.7 Directivas

De los tres lenguajes que estamos analizando, PHP es diferente, ya que sus directivas no se incorporan en la página web, sino que se definen en el archivo de configuración php.ini, que puede incluir el siguiente código:

```
; RFC2616 compliant header.
; Default is zero.
; http://php.net/cgi.rfc2616-headers
; cgi.rfc2616_headers = 0

;;;;;;;;;;;;;;;;;;;;;;;;;
; File Uploads      ;
;;;;;;;;;;;;;;;;;;;;;;;;;

; Whether to allow HTTP file uploads
; http://php.net/file-uploads
file_uploads = On

; Temporary directory for http uploaded files (will use system
; default if not
; specified).
; http://php.net/upload-tmp-dir
upload_tmp_dir = "C:\xampp\tmp"

; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
upload_max_filesize = 128M
```



## 2.7 Directivas

Conforme estos lenguajes avanzan, las directivas también se modifican y amplían proporcionando cada vez mayores posibilidades de configuración.

Estos mismos avances hacen que las directivas sean muy diferentes entre lenguajes siendo unas pocas las que son comunes tanto en nombre como en funcionalidad.

La directiva básica es la directiva Page que tiene, según el lenguaje, estos atributos:

- ASP. Buffer, ClassName, CodePage, ContentType, Culture, Debug, Description, EnableSessionstate, ErrorPage, Language...
- JSP. AutoFlush, Buffer, ContentType, Extends, Import, Info, IsThreatSafe, IsErrorPage, Language, Session...

Es conveniente acceder a la documentación oficial de cada lenguaje para obtener toda la información al respecto de estas directivas o de cualquier otro aspecto del lenguaje.





## 2.8 Tipos de datos. Conversiones entre tipos de datos

Para las personas, los conceptos uno y 1 son el mismo concepto: el número uno.

Sin embargo, para una máquina, son conceptos completamente diferentes porque se escriben de formas diferentes.

Los tipos de datos son el nombre genérico que reciben los grupos de conceptos

que se escriben usando el mismo modo de escribirlos. Si escribimos letras entonces tenemos el tipo de datos texto. Si escribimos números entonces tenemos el

tipo de datos numérico, como en los siguientes ejemplos:



## 2.8 Tipos de datos. Conversiones entre tipos de datos

```
<?php
//Array almacenando cadena de caracteres como valores
$text = array('zumo','leche','arroz');

//Array almacenando valores numéricos
$numerico = array(1,45,5);

//Array almacenando cadena de caracteres
//Los números solo representan posiciones en el array,
//no se almacenan como valores
$config = array(
    0 => 'Casas';
    1 => 'Coche';
    2 => 'Árbol';
    3 => 'Puente'
);

?>
```



## 2.8 Tipos de datos. Conversiones entre tipos de datos

Los tipos de datos se dividen en dos grandes grupos:

- Tipos básicos, también llamados simples.
- Tipos compuestos, también llamados complejos.

Los tipos básicos son:

- Tipo texto (string). Grupos de caracteres alfanuméricos de longitud no fija.
- Tipo numérico (decimal, integer, float). Grupos de dígitos numéricos expresados en base 10 de longitud fija, no fija o flotante.
- Tipo booleano (boolean). Expresión de un valor cierto o falso

## 2.8 Tipos de datos. Conversiones entre tipos de datos

Los tipos compuestos se definen como agrupaciones de tipos básicos o complejos donde no se almacena únicamente un valor sino que pueden almacenarse más de un valor simultáneamente y que además pueden ser de tipos básicos diferentes o complejos. Por ejemplo, una dirección podría almacenarse de dos formas completamente diferentes:

- Usando un tipo básico de texto. “Avenida Mi Calle, 23 Escalera B 4o 1a”.
- Usando un tipo complejo compuesto del texto. “Avenida Mi Calle”, número 23, texto escalera “B”, número planta 4, número piso 1.

En ciertos casos los tipos de datos se convierten de forma automática de un tipo a otro. Esto solo pasa cuando son tipos básicos. Cuando escribimos un número en la pantalla lo que realmente se escribe es un texto. Cuando realizamos la división de dos números el resultado siempre se convertirá a número en coma flotante (float)



## 2.9 Variables

Una variable es un depósito de información temporal que puede usarse en cualquier momento dentro de nuestro código.

Su contenido no es fijo y por este motivo se denominan variables. Los depósitos de información cuyo contenido es fijo se denominan constantes. Las variables pueden contener valores de distintos tipos, desde tipos básicos a tipos complejos y también objetos.

Además del contenido, la otra cosa más importante de una variable es su nombre, puesto que será a través de ese nombre que utilizaremos la variable.



## 2.9 Variables

Cada lenguaje tiene sus limitaciones para los nombres de las variables, aunque de forma general estas limitaciones suelen ser:

- Límites en cuanto a la longitud del nombre. Hay que utilizar nombres “cortos”, no más de 30 caracteres de longitud, y claros para saber qué van a contener.
- Límites en cuanto a qué caracteres no se pueden utilizar para el nombre. Caracteres como `/ \ ( ) { } & = ? ¿`, acentos y espacios en blanco normalmente no son permitidos.
- Límites en cuanto al nombre en sí mismo. No se pueden utilizar palabras reservadas del lenguaje ni ninguna otra palabra que ya signifique algo para ese lenguaje.

Palabras como ***if***, ***for*** o ***while*** no podrán ser utilizadas como nombres de variables.

## 2.9 Variables

Con las variables podemos realizar básicamente tres tipos de acciones:

- **Definición.** Las variables solo pueden definirse una vez dentro de su ámbito de actuación.
- **Asignación o modificación.** Antes de usar una variable debemos asignarle un valor que posteriormente podrá ser modificado o no.
- **Utilización.** El objetivo de una variable es su lectura para utilizarse su contenido en cualquier momento.

Veamos, por ejemplo, cómo realizar estas acciones según el lenguaje de programación PHP.

## 2.9 Variables

- **Variables.** Empiezan por el signo dólar y son sensibles a mayúsculas y minúsculas.

Esto significa que aunque sea el mismo nombre serán dos variables diferentes si tienen algún carácter una en mayúsculas y la otra en minúsculas, como en el siguiente ejemplo:

```
<?php

//válido: empieza por una letra
$mejor_amigo = 'Pablo';

//válido: empieza por guión bajo
$_num_preferido = 8;

//no válido
$3_aficiones = 'nadar, bailar, pasear';

//no válido: palabra reservada
$TRUE = 'amistad';

?>
```

## 2.9 Variables

- Definición. No es necesaria. Se realiza de forma implícita en la asignación del primer valor de la variable.
- Asignación o modificación. `$nombre_de_variable = valor`
- Utilización. `echo $texto_a_escribir; $b = 1; $a = $b + 1;`

### Para saber más

**CONSTANTE:** si queremos trabajar con un valor que ha de permanecer intacto durante toda la ejecución del código, tenemos que definir y utilizar una constante. En PHP tenemos que utilizar la función `define()`. `define("nombre_de_la_constante", valor_de_la_constante);`

## 2.10 Ámbito de utilización de las variables

El objetivo principal de una variable es almacenar información para poder ser leída posteriormente, pero ¿dónde va a ser leída? Esta pregunta se responde mediante el ámbito de las variables; en concreto, mediante lo que se conoce como visibilidad de las variables.

El ámbito de una variable es el bloque o los bloques de código en los que esa variable puede ser utilizada. Según como se haya definido la variable, podrá ser usada en todos los bloques de código o solamente en ciertos bloques.

Nuevamente hay que decir que cada lenguaje tiene su política de ámbito de variables, aunque de forma general casi todos los lenguajes siguen las mismas normas:

- Ámbito global de las variables. Cuando una variable se define con ámbito global, esta variable podrá ser utilizada en todos los bloques de código de la página.

## 2.10 Ámbito de utilización de las variables

- Ámbito local de las variables. Cuando una variable se define con ámbito local, esta variable solo podrá ser utilizada en ciertos bloques de código de la página.

El ámbito global es el más fácil, ya que toda la página podrá utilizar esa variable, aunque muchas veces la definición global no es útil o incluso es errónea para el concepto que representa esa variable.

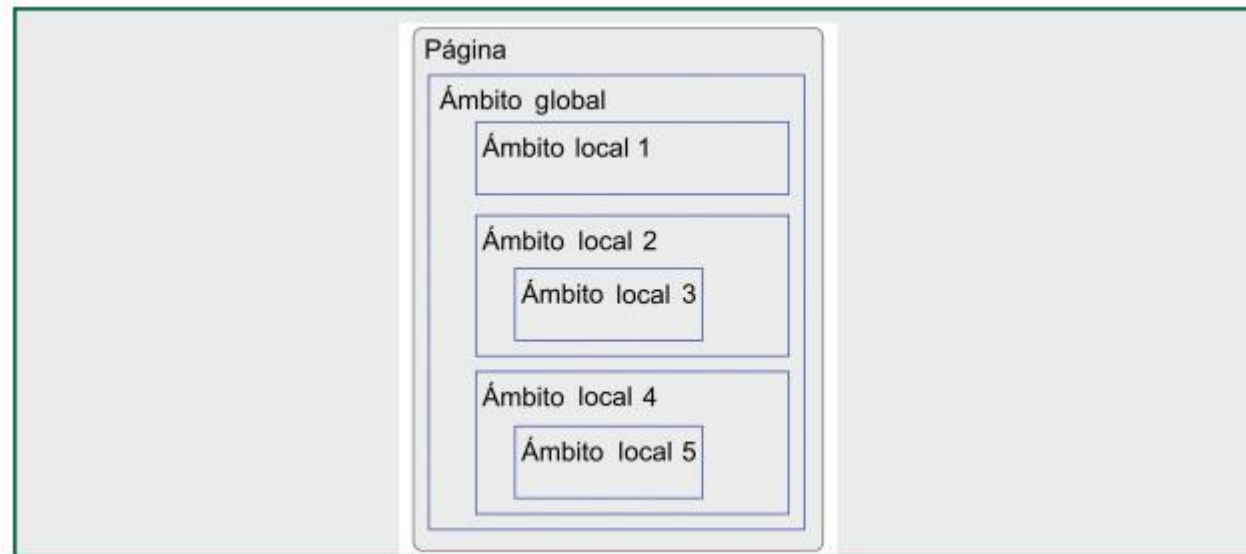
La política de ámbito local es la más utilizada puesto que la definición y el ámbito de uso de las variables queda restringido únicamente a los bloques de código donde realmente son necesarias esas variables.

Hay un aspecto especialmente complejo en el ámbito de las variables ligado con su definición, y es que podemos definir dos variables con el mismo nombre pero en dos ámbitos diferentes.

## 2.10 Ámbito de utilización de las variables

Como están definidas en dos ámbitos diferentes, se puede utilizar el mismo nombre para definirlas. Si utilizamos nombres diferentes se acabó el problema, pero habrá veces que será necesario ponerles el mismo nombre.

Los ámbitos se organizan utilizando una estructura de árbol. Un ámbito local estará contenido dentro de un solo ámbito local o global y un ámbito local o global puede contener más de un ámbito local (Figura 2.10).



**Figura 2.10**  
Organización de ámbitos  
en árbol.





## RESUMEN

Los lenguajes embebidos en HTML permiten generar páginas de contenido dinámico utilizando lenguajes de servidor. Esto significa que escribimos nuestros programas dentro de una página web, eso sí, usando una serie de marcas especiales que le indican al servidor qué cosas tiene que ejecutar y qué no.

Los servidores de aplicaciones al procesar las páginas web desestiman el texto HTML y se centran en ejecutar solamente el código que está escrito entre las marcas de inicio y final de código.

Los lenguajes embebidos más extendidos son PHP, ASP y JSP en el lado del servidor y JavaScript para dinamizar la página en el navegador del cliente.

Podemos modificar el comportamiento de ejecución de las páginas web dinámicas modificando ciertas directivas en los archivos de configuración o bien en el propio código de la página.





## RESUMEN

Las variables son elementos de almacenamiento temporal de datos para poder ser usados en las páginas por los mismos bloques de código o por otros bloques de código.

La estructura de los ámbitos es similar a la de un árbol, donde desde una rama podemos “retroceder” a las ramas anteriores hasta la raíz, pero no es posible “saltar” de una rama a otra.

## 2.11 EJERCICIOS: JAVA

Programa que escribe en pantalla los números impares entre 1 y 1000:

```
- /**
 * Programa que escribe en pantalla los números impares entre 1 y 1000
 *
 * @author
 */
public class Program1_Impares {
-     public static void main(String[] args) {
        int contador = 1;
        while (contador < 1001) {
            System.out.println("Número: " + contador);
            contador = contador + 2;
        }
    }
}
```

## 2.11 EJERCICIOS: JAVA

\* Programa que lee un número positivo por teclado y escribe en pantalla cuántas cifras tiene

```
public class Programa3_Cifras {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int numero;  
        //Se validan los datos de entrada adecuadamente  
        do{  
            System.out.println("Introduzca un número mayor que 0:");  
            //Comprobamos si el valor introducido es numérico  
            while(!sc.hasNextInt()){  
                sc.next();  
            }  
            numero=sc.nextInt();  
            if(numero<=0){  
                System.out.println("Error: El número no es mayor que 0");  
            }  
        }  
        while(numero<=0);  
        //Cálculo del número de cifras mediante divisiones sucesivas  
        int cifras = 1;  
        while(numero>9){  
            cifras++;  
            numero/=10;  
        }  
        System.out.println("El número tiene "+cifras+" cifras");  
    }  
}
```

## 2.11 EJERCICIOS: JAVA

\* Programa que lee por teclado un número entero positivo y comprueba si es primo

```
public class Programa5_Primo {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int numero;  
        boolean primo = true;  
        String esPrimo = " es primo";  
        //Se lee el valor del número por teclado  
        do {  
            System.out.println("Introduzca un número entero positivo:");  
            //Comprobamos si el valor introducido es numérico entero  
            while (!sc.hasNextInt()) {  
                sc.next();  
            }  
            numero = sc.nextInt();  
            if (numero <= 0) {  
                System.out.println("El número debe ser mayor que 0");  
            }  
        } while (numero <= 0);  
    }  
}
```

```
//El 1 no se considera primo  
if (numero == 1) {  
    primo = false;  
}  
//Ningún número par es primo excepto el 2  
} else if (numero > 2 && numero % 2 == 0) {  
    primo = false;  
}  
//Caso general para números impares mayores que 2  
} else {  
    int contador = 3;  
    int limite = (int) Math.sqrt(numero);  
    while (primo && contador <= limite) {  
        if (numero % contador == 0) {  
            primo = false;  
        }  
        contador += 2;  
    }  
}  
if (!primo) {  
    esPrimo = " NO es primo";  
}  
System.out.println("El número " + numero + esPrimo);  
}
```



## Bibliografía

- Vicente, J. (2013) “Desarrollo Web en entorno servidor”, editorial: Ibergaceta publicaciones, S.L. ISBN:978-84-1545-292-8
- Comesaña, L. (2017) “Desarrollo Web en entorno servidor”
- Desarrollo web en entorno servidor. Realización: ITACA (Interactive Training Advanced Computer Applications, S.L.) Colaboradores: Departamento de Producto de Centro de Estudios CEAC © Planeta DeAgostini Formación, S.L.U. Barcelona (España), 2016 ISBN: 978-84-9063-804-0 (Obra completa) ISBN: 978-84-9128-797-1 (Desarrollo web en entorno servidor)



## BIBLIOGRAFÍA

- Desarrollo web en entorno servidor. Realización: ITACA (Interactive Training Advanced Computer Applications, S.L.)

Colaboradores:

Departamento de Producto de Centro de Estudios CEAC © Planeta DeAgostini Formación, S.L.U. Barcelona (España), 2016

ISBN: 978-84-9063-804-0 (Obra completa)

ISBN: 978-84-9128-797-1 (Desarrollo web en entorno servidor)

