

# Tutorial básico de Java EE

**Sólo una tecnología puede programar  
todos estos dispositivos**



**En esta ocasión vamos a por este:**

*Java EE*

Copyright

Copyright (c) 2010, Abraham Otero. Este documento puede ser distribuido solo bajo los términos y condiciones de la licencia de Documentación de javaHispano v1.0 o posterior (la última versión se encuentra en <http://www.javahispano.org/licencias/>).

Para cualquier duda, consulta, insulto o tirón de orejas sobre este tutorial dirigirse a [abraham@javahispano.org](mailto:abraham@javahispano.org).



# Índice

Índice .....	3
Preámbulo .....	8
Guía para usar este tutorial .....	10
1 HTTP .....	12
1.1 Las URL .....	13
1.2 Peticiones HTTP .....	14
1.3 Respuestas HTTP .....	17
1.4 Códigos de estado del protocolo http .....	17
2 HTML .....	20
2.1 Las marcas o etiquetas .....	21
2.2 Caracteres especiales .....	23
2.3 Etiquetas relacionadas con la presentación de texto.....	24
2.3.1 Títulos de encabezamiento .....	24
2.3.2 Párrafos, líneas y tipos de fuente .....	24
2.4 Listas.....	30
2.5 Enlaces.....	33
2.6 Imágenes.....	36
2.7 Tablas .....	37
2.8 Creación de formularios .....	39
2.8.1 Campos de texto .....	40
2.8.2 RadioButton.....	41
2.8.3 Checkbox .....	42
2.8.4 Botones .....	43

2.8.5	Campos de password .....	44
2.8.6	Campos ocultos .....	44
2.8.7	Listas de selección .....	45
2.8.8	Áreas de texto .....	46
2.8.9	Ejemplo de un formulario .....	48
3	Breve introducción a las hojas de estilo (CSS) .....	52
3.1	Referenciando una hoja de estilo desde un documento .....	53
3.2	Sintaxis de las hojas de estilo .....	53
3.2.1	Identificadores y clases .....	55
3.3	Estilos CSS .....	58
3.3.1	Estilos para texto .....	58
3.3.2	Márgenes .....	61
3.3.3	Bordes .....	61
3.4	Enlaces .....	62
3.5	Un ejemplo .....	64
4	Un primer vistazo a los Servlets .....	68
4.1	El servidor de aplicaciones .....	68
4.2	Estructura de directorios de una aplicación web .....	69
4.3	Nuestro primer Servlet .....	71
4.4	HttpServletRequest y HttpServletResponse .....	77
4.5	Un Servlet que procesa un formulario .....	83
4.5.1	El ejemplo anterior apesta .....	87
5	Ciclo de vida de los Servlets .....	91
5.1	Configurando los Servlet: ServletConfig .....	92
5.2	Un ejemplo .....	94
6	Recordando al usuario: sesiones y cookies .....	96

6.1	HttpSession.....	96
6.1.1	Ejemplo: SessionServlet.....	99
6.2	Las Cookies .....	102
6.2.1	Un ejemplo: .....	104
6.3	Sesiones vs Cookies .....	106
7	Compartiendo información entre Servlets.....	108
7.1	ServletContext .....	108
7.2	Compartir información dentro de una misma petición.....	111
7.3	Guardando información en el contexto de la aplicación .....	114
7.4	Recapitulando: opciones para "recordar" información.....	117
8	Gestión declarativa de errores en la aplicación .....	119
8.1	Gestión de códigos de error http.....	120
8.2	Gestión declarativa de excepciones.....	121
9	Problemas de concurrencia con los Servlets.....	125
9.1	Un Servlet con problemas de concurrencia .....	125
9.2	¿Con qué variables pueden aparecer los problemas de concurrencia? .....	131
9.3	Una primera (cutre) solución a los problemas de concurrencia .....	132
9.4	Una mejor solución a los problemas de concurrencia .....	133
10	Bases de datos y los Servlets .....	136
10.1	Empleando JDBC directamente.....	136
10.2	Empleando el pool de conexiones .....	142
11	Reaccionando a eventos en el contenedor: listeners y filtros .....	146
11.1	ServletContextListener: arranque y parada de la aplicación .....	146
11.2	Escuchando eventos de sesión.....	149
11.3	Filtros.....	151
11.3.1	Un ejemplo de filtro.....	154

12	Páginas JSP .....	158
12.1	Despliegue de las páginas JSP .....	160
12.2	Mi primera página JSP .....	161
12.3	Sintaxis de las páginas JSP .....	162
12.3.1	Comentarios en las páginas JSP .....	162
12.3.2	Etiquetas de declaración .....	163
12.3.3	Etiquetas de scriptlet .....	164
12.3.4	Etiquetas de expresiones .....	166
12.3.5	Etiquetas de directivas .....	168
12.4	Variables implícitas en las páginas JSP .....	171
12.5	Un ejemplo de una página JSP .....	172
12.6	Tipos de errores en las páginas JSP .....	174
13	Acciones estándar JSP .....	176
13.1	Sintaxis de las acciones estándar JSP .....	177
13.1.1	Manipulación de Java Beans .....	177
13.1.2	Accediendo a los parámetros de un formulario .....	179
13.1.3	Redirigiendo peticiones entre páginas JSP .....	182
13.2	Poniendo todo esto junto: aplicación para salvar gatitos .....	182
13.2.1	Problemas con esta aplicación .....	189
13.3	Creando etiquetas a medida .....	190
14	Poniéndolo todo junto en una aplicación: BasicCRUDWebApp .....	198
14.1	Patrón MVC .....	198
14.2	BasicCRUDWebApp .....	199
14.3	El modelo y la persistencia (patrón DAO) .....	201
14.4	El controlador .....	211
14.5	La vista .....	219

14.6	Cada una de las interacciones en más detalle .....	224
15	¿Y ahora qué? .....	226
15.1	Si has llegado hasta aquí estás en deuda con la comunidad .....	227

# Preámbulo

Hace aproximadamente 10 años acababa de comenzar mi doctorado en la Universidad de Santiago de Compostela. No tenía ningún tipo de financiación, y en el departamento donde estaba trabajando me echaron un cable para conseguir un trabajo impartiendo cursos de programación Java en la Universidad. Para impartir los cursos debía crear unos apuntes. Así que comencé a escribir un documento al cual titulé "Tutorial básico de Java".

Esta intrascendente acción fue uno de esos pequeños hechos aleatorios que suceden en tu vida y que te la cambian completamente. El tutorial tuvo mucha aceptación entre los alumnos. Lo acabé enviando a una página web que se llamaba javaHispano.org para que lo publicasen, lo cual acabó haciendo que me uniese a los chicos que estaban detrás de la página. Con el tiempo, acabé siendo el presidente de la asociación sin ánimo de lucro que creamos a posteriori, y desde entonces prácticamente todos los años he estado involucrado en organizar una conferencia Java en España (entre otras muchas cosas) bajo el banner de javaHispano.

El tutorial, del cual desde el año 2000 he escrito 3 revisiones, es el documento más descargado de javaHispano.org, sumando a estas alturas bastante más de un cuarto de millón de descargas. Mucha gente me ha escrito para darme las gracias por ese documento; no pasa un mes sin que reciba al menos un e-mail al respecto. Eso 10 años después; en su día recibía varios a la semana. También me han ofrecido escribir un libro sobre Java a raíz de este tutorial. Y una revista me pidió que escribiese una serie de artículos introductorios al lenguaje Java para ella, que finalmente acabó publicando tanto por separado en números normales de la revista, como en un monográfico especial que sólo contenía mis artículos.

Mucha gente me ha escrito para pedir una continuación del "Tutorial básico de Java" donde se aborde Java EE. Sin duda, es la cosa que más veces me han pedido desde que me uní a javaHispano. Incluso una persona me ha dicho directamente que le indique un



número de cuenta y que me pagaba por escribir esa continuación (de verdad, no es una broma). La falta de tiempo siempre me había impedido emprender esta tarea.

Han pasado 10 años desde que escribí aquel primer tutorial. Ahora soy profesor en otra universidad, y el cuatrimestre que viene voy a impartir una asignatura de introducción a la programación web con Java por primera vez en esta universidad. Una excusa excelente para hacer unos buenos apuntes al respecto. Una excusa excelente para comenzar la escritura de este documento. No sé si será tan didáctico y tendrá tanto éxito como el primero; el listón está puesto bastante alto. Lo único que le puedo prometer al lector es que lo voy a intentar. Comencemos pues.

# Guía para usar este tutorial

Este tutorial trata de ser un curso de programación web con Java empezando desde 0. No se sume ningún conocimiento previo de HTML, del protocolo http o de programación web. Sí se asume que el lector tiene soltura con el lenguaje de programación Java. Aquellos lectores con conocimiento de HTML, http y CSS pueden a su propia discreción saltarse los primeros tres temas.

Este documento se distribuye junto con tres videos de apoyo. El primero hace una descripción general del código que acompaña a este tutorial. El segundo, es una introducción básica a Netbeans como herramienta para desarrollar aplicaciones web Java. El tercero, muestra cómo utilizar la consola de administración de Glassfish, así como la funcionalidad de administración de bases de datos de Netbeans.

Los códigos que acompañan este tutorial son aplicaciones Java EE estándar que deberían funcionar en cualquier contenedor de servletes Java EE. No tienen ninguna dependencia con Netbeans, ni con Glassfish. No obstante, se distribuyen como un proyecto de Netbeans. El motivo de esto es que considero Netbeans indudablemente la mejor herramienta Java para *comenzar* a programar aplicaciones web. Eclipse no es tan autocontenido y requiere de la instalación de plugins para tener cierta funcionalidad. Si estás leyendo este tutorial probablemente no tengas ni idea de programación web, y con total certeza no tienes ni idea de programación web con Java. En este escenario, mi clara recomendación es que uses Netbeans. En cualquier caso, los códigos deberían funcionar perfectamente con cualquier otro entorno de desarrollo (o servidor de aplicaciones).

Junto con el tutorial se distribuyen dos proyectos de Netbeans. El primero contiene todos los ejemplos que se irán mostrando desde el tema 4 hasta el tema 13. Este proyecto, cuyo nombre es CursoBasicoJavaEE, contiene un descriptor de despliegue completamente documentado y ordenado por temas. Estúdialo según vayas avanzando en los temas. También contiene anotaciones equivalentes a toda la información presente en el descriptor. Es decir, toda la información de configuración está por duplicado con fines pedagógicos.

El segundo proyecto, cuyo nombre es BasicCRUDWebApp, es un ejemplo de una aplicación web básica con soporte para las cuatro operaciones CRUD sobre un modelo sencillo. En este caso, la información de configuración sólo está en el descriptor de despliegue. Como veremos en el tema 14, dada la naturaleza de la aplicación es interesante que su información de configuración no esté empotrada en el código fuente.

Uno de los ejemplos del tema 10, y uno de los tres mecanismos alternativos de persistencia empleados en BasicCRUDWebApp requieren de la configuración de un pool de conexiones en el servidor de aplicaciones. En uno de los videos que acompañan a este tutorial se explica cómo configurar dicho pool en Glassfish.

A lo largo del tutorial se irá indicando el momento en el cual el lector debe consultar cada uno de los tres videos. Cuando se indique que se debe consultar uno de estos videos, recomiendo al lector que no continúe leyendo el tutorial y que lo mire.

Ahora es el momento de ver el primer video.



<https://www.youtube.com/watch?v=PJtdpR1FDDE>

# 1 HTTP

Para desarrollar aplicaciones web es necesario tener al menos unas nociones básicas del funcionamiento del protocolo empleado para la transferencia de los recursos que componen una web. El Hypertext Transfer Protocol (protocolo de transferencia de hipertexto) o HTTP es el protocolo empleado con este fin. Se trata de un protocolo orientado a transacciones que sigue un esquema de petición-respuesta entre un cliente y un servidor. Al cliente que efectúa una petición (habitualmente, un navegador web) se le denomina "user agent". La información transmitida se denomina recurso y se la identifica mediante un localizador uniforme de recursos (URL, uniform resource locator). Como su nombre indica, una URL permite localizar un recurso en Internet. Las URL son siempre únicas, hecho que garantiza el W3C, y cada URL en Internet identifica unívocamente un recurso (página web, imagen, Applet, flash, etc.).

El protocolo no tiene estado, esto es, no guarda ninguna información sobre conexiones anteriores. Una conversación en este protocolo habitualmente comienza por una petición del cliente de un determinado recurso al servidor. El servidor, responde sirviendo dicho recurso. Pero no guarda ningún tipo de registro o memoria de qué recursos ha servido a qué clientes. Si a continuación el mismo cliente vuelve a pedir otro recurso, la respuesta del servidor será idéntica al caso en el que un cliente diferente, totalmente nuevo, haya pedido el recurso. El hecho de tratarse de un protocolo sin estado hace que sea altamente escalable y ha sido uno de los secretos de su éxito. También es una de sus principales limitaciones con vistas a desarrollar aplicaciones web, y parte de la funcionalidad que nos proporcionarán los servidores de aplicaciones es precisamente mitigar este problema. Pero ya llegaremos allí. En este capítulo vamos a ver la estructura básica de las peticiones y respuestas del protocolo http.

## 1.1 Las URL

Las URL indican cómo localizar en Internet un determinado recurso. Su formato es:

```
protocolo://maquina:puerto/camino/fichero
```

El protocolo habitualmente es http (HyperText Transport Protocol) o https (HyperText Transport Protocol Secure), aunque pueden emplearse otros protocolos como ftp (File Transfer Protocol). La máquina es el nombre o la IP del servidor al cual nos queremos conectar. Habitualmente, se emplea un nombre (como Google.com) que es traducido a una IP por el servicio de DNS. Después de la máquina, separado por ":" puede indicarse el puerto al cual nos queremos conectar. Por defecto, cada protocolo tiene un puerto asignado; por ejemplo http tiene asignado el puerto 80 y https tiene asignado el puerto 443. Al publicar una página web en Internet, lo habitual es emplear el puerto por defecto; sin embargo cuando estamos construyendo esa página web en nuestro equipo local a menudo se emplean puertos diferentes para depuración y testeo de dicha página.

Camino es la ruta en el sistema de archivos de la máquina remota donde se encuentra el recurso al cual queremos acceder. Dicha ruta es relativa al directorio raíz de la web. Fichero es el recurso concreto al que queremos acceder dentro de un directorio de la máquina. Por ejemplo:

```
http://javaHispano.org:4040/ejemplo/inicio.html
```

accedería empleando el protocolo http al puerto 4040 de la máquina javaHispano.org. Dentro de esa máquina debería haber un servidor web sirviendo el contenido de algún directorio de la máquina. Partiendo de ese directorio, el servidor web iría al directorio "ejemplo" y buscaría el recurso con nombre inicio.html, que serviría al cliente que lo ha solicitado.

## 1.2 Peticiones HTTP

Las peticiones de http siguen el siguiente formato:

```
Método SP URL SP Versión Http CRLF
(nombre-cabecera: valor-cabecera (, valor-cabecera)*CRLF)*
Cuerpo del mensaje
```

SP significa espacio en blanco; los espacios en blanco introducidos en el texto han sido introducidos por motivos de legibilidad, sólo los indicados mediante SP son requeridos en el protocolo. CRLF significa retorno de carro. Todo lo que aparece entre parentesis es opcional. Los paréntesis que están dentro de los paréntesis indican contenido opcional dentro del propio paréntesis. Cuando después de un paréntesis hay un \*, significa que el contenido del paréntesis puede aparecer repetido cero o más veces.

La primera parte de la petición es el método que se emplea para realizar la petición (habitualmente GET o POST). A continuación, separada mediante un espacio en blanco, se indica la URL del recurso que se está pidiendo. Después, separada por un nuevo espacio en blanco, se indica la versión del protocolo http que se está empleando.

A continuación se pueden encontrar una o varias cabeceras. Las cabeceras tienen asociado un nombre y después llevan ":". Después de los ":" debe ir un valor para la cabecera. Es posible que una misma cabecera tenga asociados varios valores. En este caso, los valores se separan por ","; los espacios en blanco dentro del valor de una cabecera se considera que forman parte del valor, y no que son separadores.

En el mensaje puede haber varias cabeceras separadas por retorno de carro. Después de las cabeceras, viene el cuerpo del mensaje, que puede ser vacío (es el caso del método GET) o no (es el caso del método POST). Por ejemplo, una petición http podría ser:

```
GET /en/html/dummy?name=MyName&married=not+single&male=yes HTTP/1.1
Host: www.explainth.at
User-Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11) Gecko/20070312
Firefox/1.5.0.11
Accept: text/xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
```

```
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.explainth.at/en/misc/httpreq.shtml
```

En este caso, el cuerpo está vacío. Un ejemplo de petición http con cuerpo no vacío podría ser:

```
POST /en/html/dummy HTTP/1.1
Host: www.explainth.at
User-Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11) Gecko/20070312
Firefox/1.5.0.11
Accept: text/xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.explainth.at/en/misc/httpreq.shtml
Content-Type: application/x-www-form-urlencoded
Content-Length: 39

name=MyName&married=not+single&male=yes
```

Este último caso podría ser el resultado del envío de un formulario del cliente al servidor. Esta petición tipo post sería completamente equivalente a la anterior petición get en lo que a envío de datos de un formulario al servidor se refiere. En este caso, el contenido del mensaje es sólo la última línea. En el formulario había tres campos, uno con nombre "name", otro con nombre "married", y el último con nombre "male". En la respuesta, cada uno de los tres campos está separado por el símbolo "&". Cada campo va seguido del valor que el usuario ha introducido en el formulario para dicho campo, y separando el nombre del campo y su valor va el signo "=". En el siguiente capítulo veremos cómo funcionan los formularios en más detalle y cómo crearlos.

El protocolo de conexión http define varios métodos (algunas veces también llamados "verbos") que indican la acción que desea efectuar sobre el recurso indicado en la petición. Ese recurso podría ser un archivo que reside en un servidor, o podría ser un programa que se está ejecutando en dicho servidor. Los métodos son:

- **HEAD**: pide al servidor que le envíe una respuesta idéntica a la que enviaría a una petición GET, pero sin el cuerpo de la respuesta. Esto es útil para la recuperación de la meta-información contenida en las cabeceras de la petición.
- **GET**: pide al servidor que le envíe un recurso.
- **POST**: envía datos al servidor para que sean procesados por el recurso especificado en la petición. Los datos se incluyen en el cuerpo de la petición. Este método podría crear un nuevo recurso en el servidor, o actualizar un recurso ya existente.
- **PUT**: envía un recurso determinado (un archivo) al servidor. A diferencia que POST, este método crea una nueva conexión (socket) y la emplea para enviar el recurso, lo cual resulta más eficiente que enviarlo dentro del cuerpo del mensaje.
- **DELETE**: elimina el recurso especificado.
- **TRACE**: pide al servidor que le envíe un mensaje de respuesta. Se suele emplear para diagnosticar posibles problemas en la conexión.
- **OPTIONS**: pide al servidor que le indique los métodos HTTP que soporta para una determinada URL.
- **CONNECT**: se emplea para transformar una conexión ya existente a una conexión encriptada (https).
- **PATCH**: se emplea para modificar parcialmente un recurso ya existente en el servidor.



## 1.3 Respuestas HTTP

Una respuesta del servidor en el protocolo http sigue la siguiente estructura:

```
Versión-http SP código-estado SP frase-explicación CRLF
(nombre-cabecera: valor-cabecera ("," valor-cabecera)* CRLF)*
Cuerpo del mensaje
```

El código de estado es un código que indica si la petición ha tenido éxito o habido algún error con ella. La frase de explicación suele proporcionar alguna explicación del error. Las cabeceras siguen la misma estructura que en las peticiones. Después de las peticiones, podemos encontrar la respuesta del servidor. Un ejemplo de una respuesta del servidor podría ser:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<html>
<head> <title> Título de nuestra primera página </title> </head>
<body>
¡Hola mundo!
</body>
</html>
```

## 1.4 Códigos de estado del protocolo http

Los códigos de estado del protocolo http son números de tres dígitos que forman parte de las respuestas http. Estos códigos explican qué ha sucedido al intentar llevar a cabo una petición. Estos códigos son:

- Códigos 1xx : Mensajes
  - 100-111 Conexión rechazada
- Códigos 2xx: Operación realizada con éxito
  - 200 OK
  - 201-203 Información no oficial
  - 204 Sin Contenido
  - 205 Contenido para recargar
  - 206 Contenido parcial
- Códigos 3xx: Redirección
  - 301 Mudado permanentemente
  - 302 Encontrado
  - 303 Vea otros
  - 304 No modificado
  - 305 Utilice un proxy
  - 307 Redirección temporal
- Códigos 4xx: Error por parte del cliente
  - 400 Solicitud incorrecta
  - 402 Pago requerido
  - 403 Prohibido
  - 404 No encontrado
  - 409 Conflicto
  - 410 Ya no disponible
  - 412 Falló precondition
- Códigos 5xx: Error del servidor
  - 500 Error interno

- 501 No implementado
- 502 Pasarela incorrecta
- 503 Servicio no disponible
- 504 Tiempo de espera de la pasarela agotado
- 505 Versión de HTTP no soportada

## 2 HTML

HTML (HyperText Markup Language, Lenguaje de Marcado de Hipertexto) es el lenguaje en el que se construyen las páginas web. La primera descripción de HTML disponible públicamente fue un documento llamado HTML Tags (Etiquetas HTML), publicado por primera vez en Internet por Tim Berners-Lee, un científico del CERN, en 1991. Su propósito era definir un lenguaje que permitiese crear vínculos electrónicos entre distintos documentos, de tal modo que desde un documento se pudiese referir a otro y navegar fácilmente hasta él.

La primera versión de HTML que se convirtió en un estándar fue la 3.0, cuando el World Wide Web Consortium (W3C) lo aceptó como uno de sus estándares en 1995. En 1999 llegó HTML 4.0. En la actualidad, se está definiendo la versión 5.0.

El propósito de este capítulo no es, ni mucho menos, hacer una presentación detallada del estándar HTML. Eso requeriría un tutorial completo. El propósito de este capítulo es proporcionar al lector sin experiencia previa en HTML unas nociones básicas que le permitan aprender a crear aplicaciones de servidor.

Como ya irás teniendo más claro según vayas avanzando en el tutorial, el crear una aplicación web requiere por un lado realizar trabajo de programación en algún lenguaje (Java en nuestro caso) y trabajo con HTML y otras tecnologías relacionadas (CSS y Javascript fundamentalmente). A menudo los programadores se especializan en una u otra tarea. A los primeros, se le suele llamar programadores propiamente dichos y se dice que se encargan de implementar el "Back end" de la aplicación, la parte del servidor. A los segundos, a menudo se les llama diseñadores y se dice que se encargan de implementar el "Front end" de la aplicación, la parte del cliente; lo que el usuario finalmente ve en su navegador. Este tutorial hace énfasis en la parte del servidor. El propósito de este capítulo es poco más que proporcionar unas nociones básicas de HTML para que seamos capaces de generar unas páginas web mínimas que visualicen el resultado de la programación que hemos realizado en el servidor.

Sobra decir que si el lector tiene experiencia previa en HTML, puede saltarse este capítulo.

## 2.1 Las marcas o etiquetas

Como las siglas de HTML indican, HTML es un lenguaje basado en marcas o etiquetas. La sintaxis de las etiquetas es:

```
<nombre de la etiqueta [atributo1 = "valor1", atributo2 = "valor2"  
, ...]>  
[contenido de la etiqueta]  
[</nombre de la etiqueta> ]
```

Las partes de la etiqueta que van entre "[]" son opcionales. Una etiqueta siempre va entre "<...>". Opcionalmente, puede tener un conjunto de atributos y valores. Algunos atributos no tienen un valor asociado; a estos atributos se les suele denominar compactos. En caso de tener un valor asociado, el valor del atributo debe ir entre comillas. El contenido de la etiqueta puede ser texto, u otras etiquetas. Algunas etiquetas no tienen ningún contenido. Habitualmente, todas las etiquetas deben cerrarse.

Todo el contenido de un documento HTML se encuentra entre las etiquetas <html> </html>. Cada página se divide a su vez en cabecera (delimitada entre las etiquetas <head> </head>) y cuerpo (delimitado entre las etiquetas <body> </body>). Toda la información que se incluye en la cabecera no se renderiza al visualizar el documento. Se considera "meta información". Es buena práctica incluir dentro de la cabecera de la página web un título, empleando la etiqueta <title>. Éste será el título que nuestro navegador web muestre en su barra. Este título también es considerado por los buscadores de Internet, como Google, a la hora de determinar sobre qué trata nuestra página.

A continuación presentamos nuestra primera página web, en cuyo cuerpo hemos incluido el tradicional mensaje de hola mundo. El texto que va entre <!--... --> es un comentario que será ignorado por el navegador web.

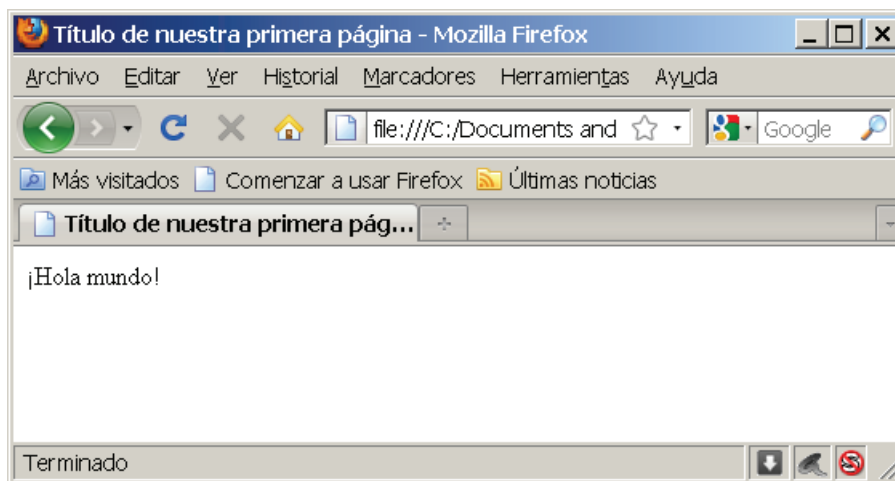
```

<!-- Ejemplo1.html -->

<html>
<head> <title> Título de nuestra primera página </title> </head>
<body>
¡Hola mundo!
</body>
</html>

```

Si guardas este documento en un archivo de texto con extensión ".html" y lo abres con tu navegador web verás algo similar a:



HTML no distingue entre mayúsculas y minúsculas en el nombre de las marcas. Es decir, <html> es lo mismo que <HTML>. Los espacios en blanco, tabulaciones y retornos de carro que haya entre marcas serán ignorados por el navegador web. De ahí que podamos ponerlos a nuestro placer, para formatear adecuadamente el texto. Por ejemplo, el texto de Ejemplo1.HTML podría haberse escrito así:

```

<html><head> <title> Título de nuestra primera página
</title></head><body>¡Hola mundo!</body></html>

```

Aunque obtendríamos el mismo resultado en nuestro navegador web, la página web sería bastante menos legible para el programador, y por tanto más difícil de modificar y mantener en el futuro.

## 2.2 Caracteres especiales

Como en todo lenguaje de programación, en HTML hay algunos caracteres que tienen significado especial, como por ejemplo `<` y `>`. Por ello, no pueden emplearse directamente en el código HTML, sino que debe emplearse una secuencia de escape equivalente. En la siguiente tabla mostramos las secuencias de escape más comunes:

Secuencias de escape más comunes de HTML	
<code>&amp;lt;</code>	<code>&lt;</code>
<code>&amp;gt;</code>	<code>&gt;</code>
<code>&amp;amp;</code>	<code>&amp;</code>
<code>&amp;quot;</code>	<code>"</code>
<code>&amp;aacute;</code> <code>&amp;eacute;</code> <code>&amp;iacute;</code> <code>&amp;oacute;</code> <code>&amp;uacute;</code>	á, é, í, ó, ú
<code>&amp;ntilde;</code>	ñ
<code>&amp;iquest;</code>	¿

Como habrás podido observar, todas las secuencias de escape empiezan con un `"&"` y terminan en un `";"`. La mayor parte de los navegadores modernos permiten escribir vocales con acento y caracteres como la `"ñ"` directamente en el HTML. No obstante,

suele ser una buena idea curarse en salud y emplear las secuencias de escape por compatibilidad con navegadores antiguos.

## 2.3 Etiquetas relacionadas con la presentación de texto

En esta sección vamos a ver varias etiquetas relacionadas con la presentación de texto en páginas web.

### 2.3.1 Títulos de encabezamiento

Permiten describir secciones, subsecciones, subsubsecciones... del documento HTML. Existen seis tipos de cabecera diferentes; de más relevante a menos relevante son:

```
<h1>Encabezado 1</h1>  
<h2>Encabezado 2</h2>  
<h3>Encabezado 3</h3>  
<h4>Encabezado 4</h4>  
<h5>Encabezado 5</h5>  
<h6>Encabezado 6</h6>
```

El texto del encabezamiento aparecerá habitualmente con un tamaño de fuente más grande y, en general, que resalta más cuanto mayor es el número del encabezamiento.

### 2.3.2 Párrafos, líneas y tipos de fuente

La etiqueta **<p>** sirve para delimitar párrafos. Además de introducir un retorno de carro, fuerza un segundo retorno de carro para dejar una línea en blanco entre dos párrafos consecutivos. Por defecto los párrafos están justificados a la izquierda. Pero podemos modificar la alineación horizontal de un párrafo mediante el atributo **ALIGN** de la marca **<p>**, que puede tomar los valores:

- **LEFT**: justificado a la izquierda. Es el valor por defecto.



- CENTER: el párrafo está centrado
- RIGHT: justificado a la derecha

La etiqueta **<br/>** introduce un retorno de carro en la posición en la que es colocada. Se trata de una etiqueta compacta, es decir, una etiqueta que se abre y se cierra en el mismo punto.

Es posible indicar que una frase, palabra, un conjunto de letras/letra de una palabra, deben mostrarse en negrita, cursiva, con algún tipo de énfasis... empleando las siguientes etiquetas:

```
<b>Negrita </b>
<i>Cursiva </i>
<tt> Emplea una fuente de tamaño fijo </tt>
<em>Énfasis </em>
<strong>Más énfasis </strong>
```

La etiqueta **<TT>** se emplea para usar fuente de ancho fijo. Al emplear esta fuente, un carácter como "." ocupa lo mismo que un carácter como "H". Este tipo de fuente suele emplearse para, por ejemplo, mostrar listados de código fuente dentro de un documento HTML. Todas estas etiquetas pueden afectar a un párrafo completo, a una palabra, o incluso a un conjunto de letras de una palabra. Por ejemplo, "ne**<b>gri</b>ta", se mostraría en un navegador web como "negrita".**

Puede modificarse el tamaño de la fuente utilizando el atributo size de la etiqueta **<font>**:

```
<font size="tamaño">Texto</font>
```

HTML define siete tamaños de fuente distintos, siendo el tamaño 3 el tamaño por defecto. Podemos indicar el tamaño mediante un número entero entre 1 y 7, o también podemos indicar el número de veces que queremos que esa fuente sea mayor o menor que la fuente por defecto. Por ejemplo,

```
| <font size="2">Texto</font>
```

significa usar el tamaño "2" para la fuente, mientras que

```
| <font size="+2">Texto</font>
```

significa usar una fuente con un tamaño igual al tamaño por defecto más 2 unidades. Del mismo modo,

```
| <font size="-2">Texto</font>
```

significa usar una fuente con un tamaño igual al tamaño por defecto menos 2 unidades. Es posible especificar el tamaño de la fuente por defecto mediante la etiqueta:

```
| <basefont size="tamaño">
```

aunque es etiqueta está "deprecated" en HTML 4.

En la etiqueta `<font>` también puede especificarse el color de la fuente. Para ello se emplea el atributo "color". En HTML, los colores se expresan en RGB como tres números hexadecimales consecutivos. Por ejemplo, el blanco será FFFFFFFF y el negro 000000. Muchos navegadores también permiten especificar el color como una cadena de caracteres con el nombre del color en inglés. Así, por ejemplo,

```
| <font color="red">Texto</font>
```

es equivalente a

```
<font color="FF0000">Texto</font>
```

Para separar distintas secciones en una página web a veces es útil emplear una línea horizontal. Podemos crear esta línea horizontal con etiqueta **<hr/>**.

El contenido que se encuentra dentro de la etiqueta **<pre>** respetará los espacios en blanco, tabulaciones y retornos de carro que contenga, además de mostrarse con una fuente de ancho de tamaño fijo. Esta etiqueta suele ser útil para, por ejemplo, visualizar código fuente dentro de una página web.

Pongamos todo esto junto:

```
<!-- Ejemplo2.html-->

<html>
<head>
<Title>Ejemplo2.html</Title>
</head>

<body>
<h1> Este ser&acute; el t&iacute;tulo de la p&acute;gina </h1>
<h2>P&acute;rrafos</h2>
<p> Este es el primer p&acute;rrafo. Aqu&iacute; hay una ruptura de
línea<br/>de texto </p>
<p>Y &eacute;ste es otro p&acute;rrafo. Observa como hay una
l&iacute;nea en blanco entre ellos al
mostrarse en el navegador web.</p>
<p>
<B>Negrita </B><br/>
<I>Cursiva </I><br/>
<TT> Emplea una fuente de tama&ntilde;o fijo: *.* ocupa lo mismo que
*A* </TT><br/>
<EM>&eacute;nfasis </EM><br/>
<STRONG>Gran &eacute;nfasis </STRONG> </p>

<pre> al renderizar esto          se van a respetar
```

```
los espacios en blanco y retornos de carro </pre>

<hr/>
<p> Observa las líneas horizontales </p>
<hr/>
<h1>Encabezado 1</h1>
<h2>Encabezado 2</h2>
<h3>Encabezado 3</h3>
<h4>Encabezado 4</h4>
<h5>Encabezado 5</h5>
<h6>Encabezado 6</h6>

</body>
</html>
```

Al abrir esto en un navegador web obtendremos algo similar a la imagen:

# Este será el título de la página

## Párrafos

Este es el primer párrafo. Aquí hay una ruptura de línea de texto

Y éste es otro párrafo. Observa como hay una línea en blanco entre ellos al mostrarse en el navegador web.

### Negrita

*Cursiva*

Emplea una fuente de tamaño fijo: \*.\* ocupa lo mismo que \*A\*

*énfasis*

### Gran énfasis

al renderizar esto                      se van a respetar

los espacios en blanco y retornos de carro

---

Observa las líneas horizontales

---

## Encabezado 1

## Encabezado 2

### Encabezado 3

### Encabezado 4

### Encabezado 5

### Encabezado 6

A partir de ahora, para incrementar la legibilidad de este documento y porque la mayor parte de los navegadores web actuales soportan el uso directo de acentos en el texto sin emplear secuencias de escape, emplearemos directamente los acentos en el texto y no usaremos más las secuencias de escape. No obstante, cuando se crea una página web destinada a ser empleada por una audiencia muy amplia suele ser recomendable emplearlas.

## 2.4 Listas

Las listas son una forma muy práctica de presentar información. En HTML existen varios tipos diferentes de listas. Las primeras, las listas ordenadas, se definen con etiqueta **<ol>**. En estas listas, cada uno de los elementos de la lista se numera consecutivamente, de ahí el nombre de lista ordenada. Cada elemento de la lista ordenada se define empleando la etiqueta **<li>**. Por ejemplo

```
<ol>
<li> Java </li>
<li> Python </li>
<li>C++</li>
</ol>
```

Se renderizará en un navegador web como:

- |  |
|--|
| <ol style="list-style-type: none"><li>1. Java</li><li>2. Python</li><li>3. C++</li></ol> |
|--|

También es posible anidar una lista dentro de otra

```
<ol>
<li> En la plataforma Java: <ol>
<li> Java </li>
<li> Groovy </li>
<li> JRuby </li>
</ol></li>
<li> Python </li>
<li>C++</li>
</ol>
```

Esto daría como resultado al mostrarse en un navegador:

1. En la plataforma Java:

1. Java
  2. Groovy
  3. JRuby
2. Python
3. C++

También podemos crear listas desordenadas mediante la etiqueta `<ul>`. En estas listas, cada uno de los elementos de la lista (elementos que se definen con la misma etiqueta que en el caso anterior: `<li>`) aparecerá precedido de algún tipo de símbolo (un punto, un cuadrado, etc.). Al igual que en el caso de las listas ordenadas, podemos anidar listas desordenadas, o anidar listas ordenadas dentro de ellas.

El último tipo de lista es la lista de definición. Habitualmente, estas listas suelen emplearse para mostrar términos seguidos de su definición. En este caso, los elementos de las listas no emplean ningún tipo de marca ni numeración para identificar cada elemento, sino que emplean sangrado entre los párrafos. Las listas de definición se crean con la etiqueta `<dl>`. Los términos a definir se etiquetan con `<dt>`, y sus correspondientes definiciones con `<dd>`. Al igual que los otros tipos de listas, éstas también pueden anidarse.

A continuación mostramos una página web donde se ejemplifican los tres tipos de lista que hemos visto:

```
<!-- Ejemplo3.html-->

<html>
<head>
<Title>Ejemplo3.html</Title>
</head>

<body>
<h2>Listas</h2>
<h3> Lista ordenada </h3>
```

```

<p> Los <font size="+1"><i> lenguajes de programación </i></font> que
mas me

gustan son <i> (en orden de preferencia): </i></p>
<ol>
<li> Java </li>
<li> Python </li>
<li>C++</li>
</ol>
<h3> Listas desordenadas </h3>
<p>Sin un orden particular, mis <font color="red"><b> comidas
favoritas </b></font> son las siguientes:</p>
<ul>
<li> La paella </li>
<li>El cocido gallego </li>
<li>Merluza</li>
</ul>

<h2> Listas de definición </h2>
<dl>
<dt> Java </dt>
<dd> Es el mejor lenguaje de programación del mundo</dd>
<dt>.NET</dt>
<dd>Es el lado oscuro de la fuerza</dd>
</dl>
</body>
</html>

```

Al mostrar esta página en un navegador web veremos algo parecido a esto:



## Listas

### Lista ordenada

Los *lenguajes de programación* que mas me gustan son (*en orden de preferencia*):

1. Java
2. Python
3. C++

### Listas desordenadas

Sin un orden particular, mis **comidas favoritas** son las siguientes:

- La paella
- El cocido gallego
- Merluza

### Listas de definición

Java

Es el mejor lenguaje de programación del mundo

.NET

Es el lado oscuro de la fuerza

## 2.5 Enlaces

Si hay algo que caracteriza a los documentos HTML es la posibilidad de crear enlaces. Esto es, vínculos entre un documento y otros documentos del mismo servidor web, documentos de cualquier otro servidor web, o secciones del mismo documento. Estos enlaces son lo que han hecho Internet lo que es a día de hoy.

Para crear enlaces se utiliza la etiqueta <a> cuya sintaxis es:

```
<a href="direccion">Texto del enlace</a>
```

El atributo href indica a dónde apunta el enlace. El texto que va dentro de la etiqueta es lo que se renderizará en la navegador web. Habitualmente, los enlaces suelen mostrarse en azul o en otro color especial distinto del resto de los textos. La dirección del enlace

no se ve directamente en la página web, aunque habitualmente los navegadores la muestran en la barra que está situada al fondo de todo del navegador cuando el usuario pone el ratón encima del enlace.

href puede apuntar a un documento que se encuentre en nuestro servidor web. Por ejemplo,

```
<a href="C:\servidor\web\documentospersonales\CV.html"> Mi currículum  
</a>
```

Aunque la sintaxis del anterior enlace es correcta y válida, nunca debe emplearse. No es buena idea indicar la ruta absoluta de un recurso en nuestro propio servidor. Si, por ejemplo, cambiamos el directorio donde se encuentra la página web o cambiamos de sistema operativo, cambiará la localización de las páginas. Es mucho mejor usar rutas relativas, que van a permanecer constantes aunque cambiemos de máquina o de directorio. Por ejemplo, suponiendo que el enlace está en una página que se encuentra en el directorio web, podríamos cambiar el enlace anterior por:

```
<a href="\documentospersonales\CV.html"> Mi currículum </a>
```

Los enlaces también pueden apuntar a cualquier otra URL de cualquier otro servidor web:

```
<a href="http://google.com"> Enlace a Google</a>
```

El tercer sitio al cual puede apuntar un enlace es a otra sección de la propia página. A veces, cuando la página es muy extensa, nos puede interesar dar un salto desde una posición a otra. Para ello tendremos que indicar empleando la etiqueta <a> cuáles son los posibles sitios a los cuales se puede "saltar" dentro del documento. La sintaxis es:

```
<a name= "nombre"></a>
```

Para saltar al punto del documento donde se halle esa etiqueta emplearemos un enlace de la forma

```
<a href="#nombre">volver al índice</a>
```

observa como el valor del atributo href debe ser el nombre que hemos dado al punto del documento al cual debemos saltar, precedido por el signo de #. Es posible desde otra página web enlazar a una sección concreta de otro documento. Por ejemplo, supongamos que la pagina <http://javaHispano.org/ejemplo.html> tiene una marca `<a name= "fin"></a>` en alguna región del documento. Desde otra página, podríamos saltar directamente a esa región del documento mediante el enlace:

```
<a href=" http://javaHispano.org#fin"> ir al fin </a>
```

A continuación mostramos un ejemplo donde se usan varios enlaces.

```
<!-- Ejemplo4.html-->

<html>
<head>
<Title>Ejemplo4.html</Title>
</head>

<body>
<h2><a name="indice"></a> Índice</h2>
<ul>
<li><a href="#uno">Sección uno</a><br/><br/><br/><br/><br/>
```

```

<li><a href="#dos">Sección dos</a><br/><br/><br/><br/><br/>
<li><a href="http://javaHispano.org"> Enlace javaHispano </a><br/>
<li><a href="http://google.com"> Enlace a Google </a><br/>
<li><a href="."> Enlace al directorio de tu equipo donde has guardado
esta página </a>
</ul>
<h3><a name="uno">Sección uno</a></h3>
<p>Esta es la sección 1 de la pagina de enlaces locales</p>
<p>Click aqui para <a href="#indice">volver al índice</a>

<h3><a name="dos">Sección dos</a></h3>
<p>Esta es la sección 2 de la pagina de enlaces locales</p>
<p>Click aqui para <a href="#indice">volver al índice</a>
</body>

```

Para que la navegación dentro del propio documento funcione correctamente, deberás reducir el tamaño de la ventana del navegador web de tal modo que aparezca el scroll a la derecha. Si toda la página web se puede mostrar dentro de la pantalla del ordenador, y no hace falta scroll, no va a suceder nada cuando intentes hacer clic en los enlaces a secciones dentro de la propia página, ya que todos ellos se están mostrando ya.

## 2.6 Imágenes

Podemos empotrar imágenes dentro de una página web empleando la etiqueta

```
</img>
```

el atributo src indica la fuente, la localización, de la imagen. Puede ser una URL a cualquier imagen situada en cualquier página web de Internet, o una ruta relativa o ruta absoluta de esa imagen dentro de nuestro servidor web. Es posible especificar el alto y el ancho de la imagen con los atributos width y height. El alto y el ancho se medirá en píxeles. Si especificamos ambos valores, es posible que la imagen se distorsione para adecuarse a ellos. Por lo general, es una buena práctica especificar sólo uno de los

valores. En este caso, el navegador web escalará proporcionalmente la otra dimensión de la imagen y evitará distorsiones.

El atributo alt de la etiqueta imagen permite especificar un texto alternativo, que se mostrará en pantalla en caso de no poderse mostrar la imagen. Algunos usuarios navegan por Internet sin mostrar imágenes para hacerlo más rápido, o por motivos de seguridad. Los invidentes, suelen emplear software que lee la pantalla del ordenador al navegar por Internet, y en el caso de las imágenes lee este texto.

Por último, también es posible indicar cómo deseamos realizar la alineación de las imágenes y el texto mediante el atributo align, que puede tomar los valores : TOP, TEXTOP, CENTER, ABSCENTER, MIDDLE, ABSMIDDLE, BOTTOM, LEFT, y RIGTH.

La etiqueta que se muestra a continuación empotra el logo de Google en una página, con un ancho de 500 píxeles y un tamaño de borde de cuatro píxeles. En caso de no poder mostrarse la imagen, se mostrará el texto "Logotipo de Google".

```
<img width= "500", border= "4" , alt= "Logotipo de Google", align=
"center" src= "http://www.google.es/images/logo.png"> </img>
```

Es el posible usar una imagen como enlace; es decir, que al hacer clic sobre la imagen nos lleve a otra página web. Para ello, simplemente dentro de la etiqueta del enlace en vez de el texto que habitualmente suelen llevar los enlaces, colocamos una etiqueta de imagen:

```
<a href="enlace">  </a>
```

## 2.7 Tablas

Las tablas son una forma compacta y clara de mostrar información. En HTML, se definen mediante la etiqueta **<table>**. Dentro de la etiqueta de tabla, emplearemos la etiqueta **<tr>** para crear filas, y dentro de esta etiqueta emplearemos la etiqueta **<td>** para delimitar el contenido de cada celda. Empleando el atributo border podemos

indicar, en píxeles, el ancho del borde de la tabla. Si no indicamos ningún borde, la tabla tampoco dibujará la rejilla correspondiente con las celdas.

La siguiente etiqueta define una tabla de  $3 \times 3$  empleando un borde de un píxel:

```
<table border="1">
<tr><td>Celda A1</td><td>Celda B1</td><td>Celda C1</td></tr>
<tr><td>Celda A2</td><td>Celda B2</td><td>Celda C2</td></tr>
<tr><td>Celda A3</td><td>Celda B3</td><td>Celda C3</td></tr>
</table>
```

Al renderizarse veremos algo similar a:

Celda A1	Celda B1	Celda C1
Celda A2	Celda B2	Celda C2
Celda A3	Celda B3	Celda C3

Podemos indicar la anchura de la tabla mediante el atributo `width`, que puede ser o bien la anchura total de la tabla en píxeles, o un porcentaje. Si es un porcentaje, es el porcentaje del ancho total de pantalla que la tabla va a ocupar. Por ejemplo,

```
<table width="80%"> ..... </table>
```

Esta tabla ocupará el 80% del espacio disponible.

Anidando la etiqueta `<caption>` dentro de una tabla podemos especificar un título para la tabla

## 2.8 Creación de formularios

Los formularios se emplean para recoger datos del usuario en una página web y enviarlos al servidor. Estos datos pueden recogerse mediante campos de texto, checkboxes, listas de selección, etc. En el servidor, debe haber algún programa (como los que aprenderemos a desarrollar lo largo de este curso) capaz de recoger esos datos y procesarlos de algún modo. HTML y http se encargan sólo de enviarlos al servidor, pero no definen que se hace ni proporcionan ningún mecanismo para hacer algo con ellos.

Los formularios se crean con etiqueta **<form>**. Dentro de esta etiqueta es donde debemos colocar todos los elementos del formulario destinados a recoger la entrada del usuario. Los atributos de la etiqueta de formularios son:

- **action** = "ruta programa". Especifica el programa en el servidor que se va a encargar de procesar la información enviada por el formulario. Es posible enviar los datos a través de correo electrónico especificando el valor del atributo: "mailto: direccion\_de\_correo"; en este caso también deberemos añadir el atributo **enctype**="text/plain" para que el correo recibido no resulte ilegible. No obstante, lo más habitual es enviar los datos a una URL en el servidor usando el método GET o POST. La URL puede especificarse tanto de modo relativo como de modo absoluto.
- **method** = " POST / GET ". Indica el método que se va a emplear para enviar la información del formulario al servidor. POST envía los datos dentro del cuerpo de la petición. El método GET añade los argumentos del formulario a la URL especificada en el atributo **action**, utilizando como separador de las distintas piezas de información el signo de interrogación "?". El método más usado es POST. Si queremos mandar el formulario a una dirección de correo electrónico, es obligado usar este método.

Veamos a continuación qué elementos podemos colocar dentro de las etiquetas de un formulario. El primero es la etiqueta **<input>**, que nos va a permitir definir la mayoría de los diferentes campos del formulario. Esta etiqueta tiene un atributo con nombre **type** que permite especificar el tipo concreto de campo de entrada que queremos emplear.

## 2.8.1 Campos de texto

Si queremos incluir una caja de texto simple (lo equivalente a un `TextField`) en el formulario, deberemos indicar que `type= "text"`. Al emplear este valor de atributo, podemos además añadir los siguientes atributos a la etiqueta `input`:

- **name** = "nombre": asigna un nombre que identificará el campo de texto. Cuando desde un programa en el servidor queramos recuperar el contenido del campo de texto deberemos indicar este nombre.
- **maxlength** = "n": especifica el número máximo de caracteres que el usuario podrá incluir en el campo de texto. Es útil para realizar validaciones sobre la entrada del usuario.
- **size** = "n": establece el tamaño del campo de texto en la pantalla.
- **value** = "texto": permite especificar el valor por defecto que va a aparecer en la caja de texto; el valor que especificamos en este atributo será lo que inicialmente aparezca en el campo de texto al renderizar el formulario.
- **disabled**: desactiva el campo de texto, por lo que el usuario no podrá escribir nada en él. No es necesario especificar ningún valor para este atributo.

A continuación mostramos un ejemplo de formulario que contiene un único campo de texto con nombre `lenguaje`, y donde el usuario no va a poder escribir más de 20 caracteres. Antes del campo de texto, dentro del formulario, introducimos un mensaje para indicarle al usuario qué dato debe de introducir

```
<form action=" http://javaHispano.org/ejemplo" method="post "  
enctype="text/plain" name="ejemplo">  
  Introduce tu lenguaje de programación favorito:  
  <input type="text" maxlength="20" size="20" name="lenguaje">  
</form>
```



Al visualizarse en un navegador web veremos algo parecido a:

Introduce tu lenguaje de programación favorito: <input type="text"/>
--

## 2.8.2 RadioButton

Otro posible valor para el atributo `type` es `radio`, que permite definir un control equivalente a un `JRadioButton`. Junto con este valor de atributo, se pueden emplear los siguientes atributos:

- **name** = "nombre": asigna un nombre único a la pieza de información que representarán los `RadioButtons` que pertenezcan al mismo grupo. Éste es el identificador que emplearemos en el servidor para recuperar esta pieza de información. Debe ser el mismo para todos los elementos `radio` de un mismo grupo.
- **value** = "valor" define el valor que tomará la variable indicada en el atributo anterior y que se enviará al servidor. Es decir, al seleccionar uno de los `RadioButtons` el valor que tomará la variable que se envía al servidor será este atributo.
- **checked**: permite indicar cuál es el elemento que está seleccionado por defecto dentro de un grupo.
- **disabled**: desactiva el elemento, impidiendo que el usuario lo seleccione

Por ejemplo, al incluir esto dentro de un formulario:

```
¿Cuánto tiempo llevas programando en él?<br/>
<input type="Radio" name="tiempo" value="0-2" checked> Menos de dos
años<br/>
<input type="Radio" name="tiempo" value="2-5"> De dos a cinco
años<br/>
<input type="Radio" name="tiempo" value="5-"> Más de cinco años<br/>
```

el navegador web nos mostrará algo parecido a:

¿Cuánto tiempo llevas programando en él?

☒ Menos de dos años

☐ De dos a cinco años

☐ Más de cinco años

### 2.8.3 Checkbox

Otro posible valor para el atributo type es checkbox, que define una o más casillas de las cuales el usuario puede marcar una, varias o ninguna. Junto con este valor de atributo, podemos especificar los siguientes atributos:

- **name** = "nombre": asigna un nombre único a la pieza de información que representa un grupo de casillas. Éste es el identificador que emplearemos en el servidor para recuperar esta pieza de información. Debe ser el mismo para todas las casillas que pertenezcan a un mismo grupo.
- **checked**: permite indicar si una casilla está o no seleccionada por defecto.
- **disabled**: desactiva la casilla.

Si incluimos estas etiquetas dentro de un formulario:

```
<br/> <br/> ¿En qué entornos has usado ese lenguaje?:<br/>
  <input type="checkbox" name="entornos" value="descriptorio" checked>
Descriptorio<br/>
  <input type="checkbox" name="entornos" value="web" checked> Web<br/>
  <input type="checkbox" name="entornos" value="movilidad">
Movilidad<br/>
  <input type="checkbox" name="entornos" value="empotrado">
Dispositivos empotrados<br/>
```

El navegador web mostrará algo parecido a:

¿En qué entornos has usado ese lenguaje?:

- ☒ Escritorio
- ☒ Web
- ☐ Movilidad
- ☐ Dispositivos empotrados

## 2.8.4 Botones

Otro posible valor para el atributo `type` es `button`, que permite crear botones. Este botón puede ser usado con diferentes fines. A menudo se le proporciona funcionalidad mediante el lenguaje JavaScript, a partir del evento "OnClick". Al usar este valor de atributo, podemos emplear también los siguientes atributos:

- **name** = "nombre": asigna un nombre al botón, que puede ser útil para definir las acciones que el botón va a efectuar desde JavaScript.
- **value** = "valor": permite especificar el texto que se va a mostrar en el botón.
- **disabled**: desactiva el botón, de tal modo que el usuario no pueda hacer clic sobre él.

Esta etiqueta

```
<input type="Button" name="boton" value="No hace nada">
```

Creería este botón en un navegador web, que no haría absolutamente ninguna acción al ser pulsado:

No hace nada

También podemos crear botones con la etiqueta `<button>`. Cuando un botón creado con esta etiqueta está dentro de un formulario, la acción por defecto de dicho botón será realizar el envío del formulario servidor. Un ejemplo de uso de esta etiqueta sería:

```
<button> Enviar formulario</button>
```

También podemos conseguir un botón que envíe los datos del formulario asignando al atributo `type` el valor `submit`; de este modo estaremos incorporando al formulario un botón que permite realizar el envío de los datos. Cuando el usuario haga clic en ese botón, todos los datos del formulario serán enviados al programa del servidor o a la dirección de correo indicada en `action`. Junto con este atributo podemos emplear los siguientes atributos:

- **value** = "valor": permite definir el texto que se va a mostrar en el botón de envío.
- **disabled**: desactiva el botón.

Asignando al atributo `type` el valor `reset` obtendremos un botón que resetea todos los campos del formulario. Junto con este atributo podemos emplear los mismos atributos que con `submit`.

### 2.8.5 Campos de password

Cuando el atributo `type` toma el valor `password`, obtenemos un campo de texto en el cual cuando se teclea no se ve el contenido tecleado, sino que se muestran sólo "\*" u otro carácter similar. Su propósito es permitir introducir passwords.

### 2.8.6 Campos ocultos

Cuando el atributo `type` toma el valor `hidden`, estaremos creando un campo del formulario invisible, que no se mostrará en pantalla de ningún modo. Aunque ahora mismo esto pueda parecer bastante ridículo, cuando hagamos programas de servidor que procesen formularios veremos su utilidad. Junto con este valor, podemos usar los siguientes atributos:

- **name** = "nombre": asigna un nombre al campo oculto.
- **value** = "valor": valor que toma el campo oculto, y que el usuario no puede modificar de ningún modo.

## 2.8.7 Listas de selección

La etiqueta `<select>` permite definir listas desplegables (equivalentes a un `JComboBox` o `JList`) en las cuales el usuario podrá elegir uno o varios valores. Esta etiqueta soporta los siguientes atributos:

- **name** = "nombre": asigna un nombre a la lista; este será el nombre mediante el cual podremos identificar en el servidor la selección del usuario.
- **size** = "n": permite definir el número de opciones que van a ser visibles en la lista. Si `n=1` en la lista se presentará como una lista desplegable (al estilo de un `JcomboBox`), mientras que si se indica otro valor la lista se representará como una caja como una scroll vertical (al estilo de un `JList`).
- **multiple**: permite seleccionar varias opciones a la vez. Si se omite este atributo, el usuario sólo podrá seleccionar una de las opciones.
- **disabled**: desactiva la lista.

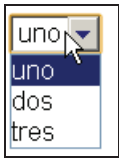
Para especificar cada una de las opciones de la lista se utiliza la etiqueta `<option>`, que admite los siguientes parámetros:

- **value**: indica el valor que será asociado al parámetro `name` de `<select>` cuando se envíe el formulario.
- **selected**: indica que esta opción va a estar seleccionada por defecto. De no indicarse en ninguna de las opciones de la lista que está seleccionada por defecto, no aparecerá ninguna seleccionada.

Por ejemplo, incluyendo estas etiquetas en un formulario:

```
<select>
  <option value="uno">uno</option>
  <option value="dos">dos</option>
  <option value="tres">tres</option>
</select>
```

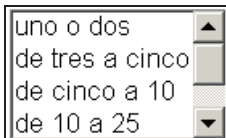
Obtenemos una lista como esta:



Mientras que incluyendo estas etiquetas:

```
<select size=3>
  <option value="uno">uno o dos </option>
  <option value="dos"> de tres a cinco </option>
  <option value="tres"> de cinco a 10</option>
  <option value="cuatro"> de 10 a 25 </option>
  <option value="cinco"> de 25 a 50 </option>
  <option value="seis"> más de 50</optio>
</select>
```

Obtenemos una lista como ésta:



### 2.8.8 Áreas de texto

Para incluir campos de texto que puedan contener múltiples líneas empleamos la etiqueta **<textarea>**. A diferencia que la etiqueta input cuando el atributo type = "text", en este campo de texto el usuario podrá introducir múltiples líneas. Si no escribimos ningún texto entre la etiqueta de apertura y de cierre, el campo de texto aparecerá vacío en el formulario. Si introducimos algún texto, éste será valor por defecto del campo de texto en el formulario. Al igual que suele suceder con cualquier campo de texto, si el

usuario introduce más texto del que se puede representar en él, automáticamente aparecerá una scroll vertical.

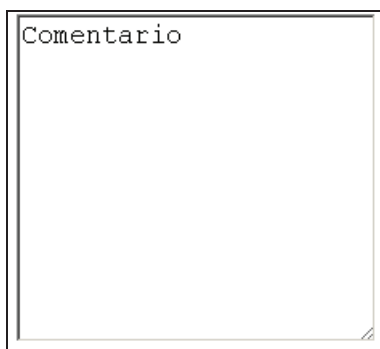
Esta etiqueta soporta los siguientes atributos:

- **name** = "nombre": especifica el nombre con el cual se va a identificar el campo de texto; este será el nombre asociado en el servidor al texto introducido por el usuario.
- **cols** = "c": especifica el número de columnas visibles del área de texto.
- **rows** = "r": especifica el número de filas visibles del área de texto.
- **disabled**: deshabilita el área de texto.
- **readonly**: convierte el área de texto en un área de sólo lectura que no puede ser modificada por el usuario.

Con las etiquetas:

```
<textarea name="comentario" cols="20" rows="10">Comentario</textarea>
```

Obtendremos un área de texto como:



## 2.8.9 Ejemplo de un formulario

Finalmente, veamos un ejemplo de un formulario con varios campos:

```
<!-- Ejemplo5.html-->

<html>
<head>
<Title>Ejemplo5.html</Title>
</head>

<body>
<h2>Ejemplo de formulario </h2>

<form action=" http://javaHispano.org/ejemplo" method="post"
enctype="text/plain" name="ejemplo">

    Introduce tu lenguaje de promoción favorito:
    <input type="text" maxlength="20" size="20" name="lenguaje">

    <br/> <br/> ¿Cuánto tiempo llevas programando en él?<br/>
    <input type="Radio" name="tiempo" value="0-2" checked> Menos de dos
años<br/>
    <input type="Radio" name="tiempo" value="2-5"> De dos a cinco
años<br/>
    <input type="Radio" name="tiempo" value="5-"> Más de cinco años<br/>

    <br/> <br/> ¿En qué entornos has usado ese lenguaje?:<br/>
    <input type="checkbox" name="entornos" value="descriptorio" checked>
Descriptorio<br/>
    <input type="checkbox" name="entornos" value="web" checked> Web<br/>
    <input type="checkbox" name="entornos" value="movilidad">
Movilidad<br/>
    <input type="checkbox" name="entornos" value="empotrado">
Dispositivos empotrados<br/><br/>

    <input type="Button" name="boton" value="No hace nada"><br/><br/>
```



```

¿Cuántos lenguajes de programación has usado además de tu favorito?
<select>
<option value="uno">uno</option>
<option value="dos">dos</option>
<option value="tres">tres</option>
</select>

<br/>¿En cuantos proyectos de software has participado ?<br/>
<select size=3>
<option value="uno">uno o dos </option>
<option value="dos"> de tres a cinco </option>
<option value="tres"> de cinco a 10</option>
<option value="cuatro"> de 10 a 25 </option>
<option value="cinco"> de 25 a 50 </option>
<option value="seis"> más de 50</optio>
</select>

<br/><br/>¿Quieres introducir algún comentario adicional?<br/>

<textarea name="comentario" cols="20" rows="10">Comentario</textarea>
<br/>

<button> Enviar formulario</button>

</form>
</body>
</html>

```

Al abrir esta página web en navegador obtendremos algo parecido a esto:

## Ejemplo de formulario

Introduce tu lenguaje de programación favorito:

¿Cuánto tiempo llevas programando en él?

- ☒ Menos de dos años
- ☐ De dos a cinco años
- ☐ Más de cinco años

¿En qué entornos has usado ese lenguaje?:

- ☒ Descriptorio
- ☒ Web
- ☐ Movilidad
- ☐ Dispositivos empotrados

¿Cuántos lenguajes de programación has usado además de tu favorito?

¿En cuantos proyectos de software has participado ?

- 
- 
- 
- 

¿Quieres introducir algún comentario adicional?

Comentario

Si el lector hace clic en el botón de "Enviar formulario" en el formulario se enviará a la url <http://javahispano.org/ejemplo>, donde no hay ningún tipo de programa esperando recibir dicho formulario, y por tanto se mostrará un error 404. Si el lector cambia el método de envío del usuario de "post" a "get", podrá ver en la URL toda la información que se envía del formulario; por ejemplo:

```
http://javahispano.org/ejemplo?lenguaje=&tiempo=0-  
2&entornos=descriptorio&entornos=web&comentario=Comentario
```

Lejos de haber cubierto todo el lenguaje de HTML, nos detendremos aquí. Quizás al lector le interese seguirse aventurando por su cuenta en HTML, y también en Javascript; hay muchos tutoriales en Internet para ello. Respecto al último tema (Javascript), puedes encontrar uno que escribí hace bastante tiempo aquí: <http://www.javahispano.org/contenidos/es/javascript/>. Si bien esto no son más que unas nociones muy básicas, son suficientes para los propósitos de este tutorial.

### 3 Breve introducción a las hojas de estilo (CSS)

CSS (Cascading Style Sheets) es un lenguaje que permite modificar la presentación del contenido de un documento HTML. Esto podría llevarse a cabo también empleando etiquetas de formato que nos permitan especificar el tipo de fuente, color, tamaño, etc. del texto de distintas secciones del documento. Sin embargo, si decidimos que queremos emplear un color azul, por ejemplo, para la fuente de todos los párrafos (el texto que vaya entre las etiquetas `<p>`) de nuestro documento, tendríamos que dentro de cada uno de estos párrafos incluir la adecuada etiqueta `<font>`. Si después decidimos que no ha sido una buena idea hacer la fuente azul y queremos volver a hacer que sea negra, tendríamos que retirar todas las etiquetas. Y si más tarde decidimos que queremos cambiar el tipo de fuente de los párrafos tendríamos que volverlas e incluir, pero esta vez especificando un tipo de fuente en vez de un color...

En el mundo real, si estuviésemos manteniendo una web compleja, seguro que nos olvidaremos de cambiar el tipo de fuente en algún párrafo o en alguna página. Esta no es una solución óptima.

Las hojas de estilo nos permiten especificar cómo se va a presentar el contenido de las distintas etiquetas HTML en un documento independiente del propio documento HTML. De este modo, podemos cambiar la presentación del documento HTML sin necesidad de modificar este documento. Es más, cuando se emplean de modo adecuado las hojas de estilo, habitualmente empleamos una única hoja de estilo para toda una web. Con sólo modificar una línea de dicha hoja de estilo podemos de un modo simple hacer que el color de la fuente de los párrafos de nuestro documento sea azul, o vuelva a ser negra, o emplee un tipo de fuente diferente.

## 3.1 Referenciando una hoja de estilo desde un documento

Para indicar dentro de un documento HTML que la maquetación de dicho documento se va a realizar conforme a lo especificado en una hoja de estilo debemos incluir una etiqueta dentro de la cabecera (<head>) del documento:

```
<link rel="stylesheet" title="Nombre de la hoja CSS" type="text/css"
href="estilo.css">
```

El atributo rel indica qué tipo de recurso estamos referenciando; una hoja de estilo en nuestro caso. title es el nombre que le queramos dar a nuestra hoja de estilo. type siempre toma el valor indicado aquí; nunca nadie ha creado una hoja de estilo en un formato diferente de texto plano (o si alguien lo ha hecho no ha tenido demasiado éxito que digamos...). El atributo href debe indicar la ruta de la hoja de estilo que queremos emplear para maquetar nuestra página web. Como ya hemos dicho, idealmente emplearemos una única hoja de estilo para todo un portal, de tal modo que cambiando esa única hoja de estilo cambiemos la apariencia de todo el portal.

## 3.2 Sintaxis de las hojas de estilo

Para dentro de una hoja de estilo redefinir la apariencia de una etiqueta HTML se emplea la siguiente sintaxis:

```
etiqueta {
<estilos CSS>
}
```

Donde "etiqueta" es la etiqueta HTML que queremos redefinir. Por ejemplo:

```
P {  
  font-size : 14pt;  
}
```

estaría indicando que el texto que vaya entre las etiquetas <p> debe emplear un tamaño de fuente de 14 puntos.

Es posible redefinir varias etiquetas a la vez, separándolas por comas:

```
etiqueta1,etiqueta2,etiqueta3 {  
<estilos CSS>  
}
```

Es posible especificar que el estilo sólo se va a aplicar a una etiqueta si está contenida dentro de otra empleando la sintaxis:

```
contenedor etiqueta {  
<estilos CSS>  
}
```

En este caso, el estilo será aplicado a "etiqueta" sólo cuando "etiqueta" se encuentre dentro de la etiqueta "contenedor". Por ejemplo:

```
ul li {  
  font-size : 10pt;  
  background-color : 666666;  
}
```

está especificando un estilo que se aplicará a las etiquetas <li> sólo cuando esas etiquetas estén dentro de otra etiqueta <ul>; si esas etiquetas estuviesen dentro de la etiqueta <ol> este estilo no se aplicaría.

### 3.2.1 Identificadores y clases

En ocasiones nos va a interesar que una misma etiqueta aparezca con presentaciones diferentes en sitios diferentes. Con este fin se pueden emplear los identificadores y las clases. Ambas clases suelen emplearse en conjunción con las etiquetas `<span>` y `<div>`. Estas etiquetas no tienen presentación en el documento HTML por sí mismas, sino que suelen emplearse "de apoyo" para las hojas de estilo.

La etiqueta `<span>` puede emplearse dentro de un texto para indicar que un fragmento de dicho texto va a representarse empleando un estilo diferente que el resto. Por ejemplo:

```
<p> Mi madre tiene ojos <span class="azul"> azules </span> y pelo  
<span class="amarillo"> rubio </span>.</p>
```

Esto nos va a permitir que las palabras "azules" y "rubio" tengan una presentación diferente que el resto del texto que aparece dentro del párrafo.

La etiqueta `<div>` tiene un uso similar, sólo que esta etiqueta puede contener otras etiquetas del documento HTML. Esta etiqueta define una división o sección en el documento que puede emplear un conjunto de estilos diferentes que el resto del documento. Por ejemplo:

```
<div class="especial">  
  <h3> Esto es una cabecera </h3>  
  <p> Esto es un párrafo </p>  
</div>
```

esta etiqueta nos permitiría aplicar un estilo especial a las etiquetas `<h3>` y `<p>` que aparecen dentro de ella, un estilo que puede ser diferente del estilo que se aplicará a estas etiquetas cuando no aparezcan dentro de `<div>`.

Para indicar que se deben usar estilos diferentes pueden emplearse tanto identificadores como clases. La principal diferencia entre ambos es que los identificadores tienen que ser únicos en todo el documento HTML, mientras que las clases pueden repetirse cuantas veces se quiera. Para indicar que queremos aplicar un determinado estilo a un identificador se emplea la sintaxis:

```
#identificador {  
  <estilos CSS>  
}
```

Podemos indicar que queremos aplicar un determinado estilo a una etiqueta sólo cuando la etiqueta se encuentre afectada por un identificador determinado:

```
#identificador etiqueta {  
  <estilos CSS>  
}
```

Por ejemplo,

```
#especial p{  
  font-size : 16pt;  
}
```

cambiará el tamaño de fuente del primer párrafo que se muestra a continuación, pero no del segundo:

```
<div id="especial" >  
<p> Este es el primer párrafo </p>  
</div>  
<p>Y este es el segundo párrafo </p>
```



Para indicar el estilo correspondiente con una determinada clase se emplea la sintaxis:

```
.clase {  
<estilos CSS>  
}
```

Para cambiar el estilo de una etiqueta sólo cuando se encuentra en una sección marcada con una determinada clase se emplea la sintaxis:

```
etiqueta.clase {  
<estilos CSS>  
}
```

Así por ejemplo, el estilo

```
.especial{  
font-size : 16pt;  
}
```

afectará al primer párrafo que se muestra a continuación, pero no al segundo:

```
<p class="especial" > Este es el primer párrafo </p>  
<p>Y este es el segundo párrafo </p>
```

El estilo

```
p.especial{  
font-size : 16pt;  
}
```

sólo afectará a etiquetas <p>, y no a cualquier otra etiqueta, aunque esa otra etiqueta también tenga asignada la clase "especial".

## 3.3 Estilos CSS

La sintaxis para los atributos de los estilos es la siguiente:

```
atributo: valor;
```

A continuación veremos algunos de los estilos que más comúnmente se emplean en las hojas de estilo.

### 3.3.1 Estilos para texto

El tipo de fuente se declara empleando la etiqueta:

```
font-family: <fuente>;
```

Siempre suele ser buena idea emplear una lista de varias fuentes, y no una única fuente, ya que corremos el riesgo de que el usuario no tenga instalada dicha fuente en su equipo. Por ejemplo:

```
font-family: Times, "Times New Roman", serif ;
```

Este estilo quiere decir que deberemos intentar usar la fuente Times; en caso de que esta fuente no esté disponible se intentará usar "Times New Roman". Si ésta tampoco estuviese disponible se emplearía una fuente predeterminada del grupo de Fuentes "serif".

Para especificar color se emplea la etiqueta:

```
color: <color>;
```

Los colores en CSS se pueden indicar en formato hexadecimal como #RRGGBB.

El tamaño de la fuente se indica mediante la etiqueta

```
font-size: <tamaño>;
```

donde el tamaño habitualmente se expresa en puntos:

```
font-size: 16pt;
```

Se pueden indicar distintas decoraciones para el texto empleando el atributo:

```
text-decoration: <decoración>;
```

donde <decoración> puede tomar los siguientes valores:

- **underline**: subraya el texto.
- **overline**: dibuja una línea encima del texto.
- **line-through**: tacha el texto.
- **none**: modo normal, sin emplear ningún tipo de decoración. Aunque este modo es el predeterminado en algunas etiquetas, en otras como <a>.

Podemos indicar el estilo de la fuente empleando la etiqueta

```
font-style: <estilo>;
```

donde <estilo> puede tomar los siguientes valores:

- **italic**: el texto se mostrara en cursiva.
- **oblique**: el texto será oblicuo (muy parecido a la cursiva).
- **normal**: el texto se mostrará normal.

Una fuente puede ponerse con distintos grosores empleando el atributo:

```
font-weight: <grosor>;
```

donde <grosor> puede tomar los siguientes valores:

- **bold**: mostrará el texto negrita.
- **bolder**: mostrará el texto en una negrita todavía más gruesa.
- **Un número del 100 al 900**: donde el grosor mínimo será el número 100, y el máximo el número 900.
- **normal**: mostrará el texto empleando un grosor normal.

Para configurar el alineado del texto podemos emplear la etiqueta:

```
text-align: <alineado>;
```

donde <alineado> puede tomar los valores left, right, center o justify.

El color de fondo puede configurarse mediante la etiqueta:

```
background-color: <color>;
```

### 3.3.2 Márgenes

Es posible configurar los márgenes que hay alrededor de un documento empleando los atributos:

```
margin-top: <cantidad>;  
margin-bottom: <cantidad>;  
margin-left: <cantidad>;  
margin-right: <cantidad>;
```

Donde <cantidad> se expresa en píxeles.

### 3.3.3 Bordes

Las hojas de estilo nos permiten ponerle bordes a todo tipo de elementos. Para ello emplearemos la etiqueta:

```
border: <tipo> <grosor> <color>;
```

El grosor es el grosor del borde en píxeles, y color es su color. La etiqueta tipo puede tomar los siguientes valores:

- **solid**: un borde sólido.
- **dashed**: un borde con línea discontinua.
- **dotted**: un borde con una línea punteada.
- **double**: un borde construido con dos líneas sólidas.

- **none**: ningún borde.

También es posible construir bordes especificando las características para cada uno de los lados del borde con las etiquetas:

```
border-top: <tipo> <grosor> <color>;  
border-bottom: <tipo> <grosor> <color>;  
border-left: <tipo> <grosor> <color>;  
border-right: <tipo> <grosor> <color>;
```

por ejemplo:

```
.borde {  
border-top: solid 2px #A00000;  
border-bottom: double 3px #00FF00;  
border-left: dotted 2px #A00000;  
border-right: dashed 2px #0000FF;  
}
```

mostrará un borde con los cuatro lados diferentes alrededor de cualquier elemento que tenga como clase "borde".

## 3.4 Enlaces

En CSS existen unas clases especiales que se llaman pseudoclases que afectan a comportamientos especiales de ciertos componentes (como los enlaces). Un ejemplo de estos comportamientos es pasar el ratón por encima, hacer clic, etc. Para definir una pseudoclase se pone después del nombre de la etiqueta ":" y el nombre de la pseudoclase que vamos a definir:

```
etiqueta:pseudoclase {  
<Formatos CSS>  
}
```

Por ejemplo, en el caso de los enlaces tenemos las siguientes pseudoclases:

- **hover**: esta pseudoclase se aplica mientras el ratón está encima del objeto.
- **visited**: esta pseudoclase se aplica a los enlaces que ya han sido visitados.
- **link**: se corresponde con enlaces en estado normal, que no han sido visitados ni tienen el ratón encima.
- **active**: esta pseudoclase se aplica mientras el enlace está activo.

Por ejemplo, estos estilos nos permitirían modificar el comportamiento por defecto de los enlaces, cambiando su color y haciendo que se subrayen cuando el ratón esté encima de ellos:

```
A:link {  
text-decoration:none;color:#0000cc;  
}  
A:visited {  
text-decoration:none;color:#ffcc33;  
}  
A:active {  
text-decoration:none;color:#ff0000;  
}  
A:hover {  
text-decoration:underline;color:#999999;font-weight:bold  
}
```

## 3.5 Un ejemplo

A continuación mostramos una hoja de estilo que permite cambiar la apariencia por defecto de varias etiquetas. La hoja de estilo emplea clases, identificadores, y pseudoclases. Podrás encontrar esta hoja de estilo en el directorio EjemplosHTML de los ejemplos de este tutorial. La primera línea de la hoja de estilo es un comentario indicando su nombre (en las hojas de estilo los comentarios comienzan con /\* y terminan en \*/)

```
/*estilos.css*/

P {/*cambiamos apariencia de los párrafos*/
  font-size : 12pt;
  font-family : arial,helvetica;
  font-weight : normal;
}

H1 {/*cambiamos la apariencia de los encabezados de primer nivel*/
  font-size : 36pt;
  font-family : verdana,arial;
  text-decoration : underline;
  text-align : center;
  background-color : Teal;
}

ul>li {/*cambiamos la apariencia de los elementos de las listas
desordenadas*/
  font-size : 10pt;
  font-family : verdana,arial;
  background-color : 666666;
}

BODY /*cambiamos la apariencia de todo el cuerpo*/ {
  background-color : #001600;
  font-family : arial;
  color : White;
}
```



```

p.especial{/*clase que se aplica sólo a los párrafos*/
  font-size : 16pt;
}

#enorme{/* mediante identificador*/
  font-size : 28pt;
}

.borde {/*clase para bordes*/
border-top: solid 2px #A00000;
border-bottom: outset 3px #00FF00;
border-left: dotted 2px #A00000;
border-right: dashed 2px #0000FF;
}

A:link {/*pseudoclase para los enlaces*/
text-decoration:none;color:#0000cc;
}

A:visited {
text-decoration:none;color:#ffcc33;
}

A:active {
text-decoration:none;color:#ff0000;
}

A:hover {
text-decoration:underline;color:#999999;font-weight:bold
}

```

A continuación mostramos una página web HTML que emplea esta hoja de estilo. En la página web se emplean etiquetas `<span>` y `<div>` para tomar ventajas de los estilos. También se emplea el atributo `id` en algunas etiquetas para indicar el identificador del estilo css a emplear, así como el atributo `class` para indicar la clase css a emplear.

```

<!-- Ejemplo6.html-->

<html>
<head>
<Title>Ejemplo2.html</Title>
<link rel="stylesheet" title="Nombre de la hoja CSS" type="text/css"
href="estilos.css">
</head>

<body>
<h1> Este será el título de la página </h1>
<h2>Párrafos</h2>

<div class="especial" >
<p> Este es el primer párrafo. Aquí hay una ruptura de línea<br/>de
texto </p>
</div>

<p class="borde"> Este texto tiene borde </p>
<p class="especial">Este párrafo tiene letra más grande. Y esto será
un enlace:
<a href=" http://javaHispano.org"> javaHispano.org</a></p>
<p id="enorme">Esto se ver  enorme.</p>

Una lista; cómo es desordenada tendrá color de fondo:
<ul>
<li>Primer elemento</li>
<li>Segundo elemento</li>
<li>Tercer elemento</li>
</ul>

Una lista ordenada:
<ol>
<li>Primer elemento</li>
<li>Segundo elemento</li>
<li>Tercer elemento</li>
</ol>

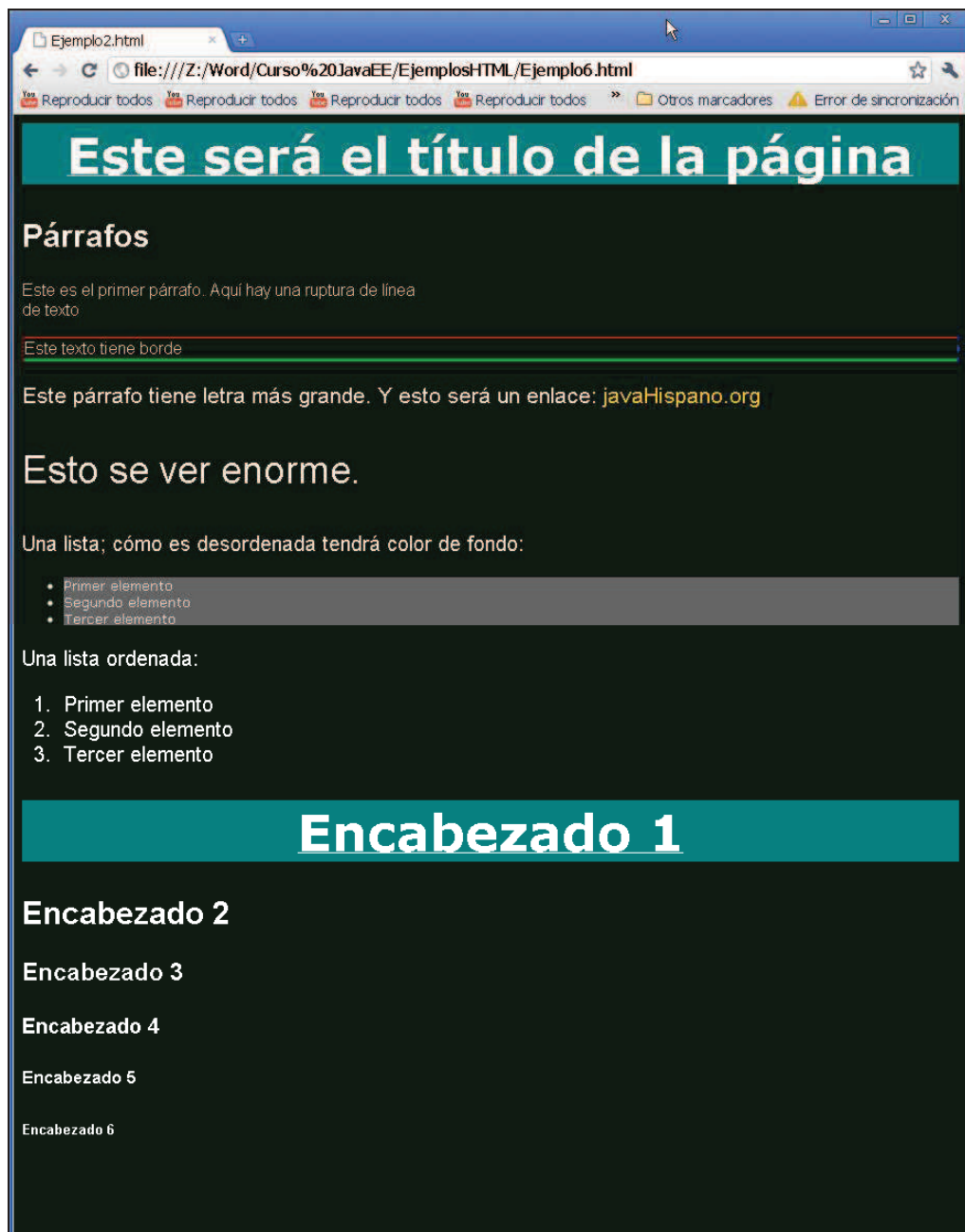
<h1>Encabezado 1</h1>
<h2>Encabezado 2</h2>

```

```
<h3>Encabezado 3</h3>
<h4>Encabezado 4</h4>
<h5>Encabezado 5</h5>
<h6>Encabezado 6</h6>

</body>
</html>
```

Esta será la apariencia de la página web:



## 4 Un primer vistazo a los Servlets

### 4.1 El servidor de aplicaciones

Actualmente, cuando se programa una aplicación web normalmente se toma ventaja de un servidor de aplicaciones. Un servidor de aplicaciones es un software de infraestructura que proporciona una serie de servicios a aplicaciones que corren en su interior. Es un concepto similar al de sistema operativo: un sistema operativo es quien controla la ejecución de las aplicaciones en un ordenador, y además les proporciona una serie de servicios a estas aplicaciones (como por ejemplo, compartir la CPU y memoria RAM entre varios programas de un modo transparente para el programador, acceso al sistema de ficheros...). El servidor de aplicaciones va a ser quien ejecute y controle la ejecución de nuestras aplicaciones web. A su vez, nos prestará una serie de servicios como por ejemplo permitir almacenar estado entre distintas peticiones del cliente, facilitar el almacenamiento de información de modo persistente (en una base de datos, por ejemplo), repartir la carga de la aplicación web entre varios servidores, etc.

El servidor de aplicaciones será quien realmente reciba la petición http. Él analizará dicha petición y almacenará toda la información relativa a ella (como las cabeceras o la información de los campos de un formulario) en objetos Java. A continuación, si existe algún programa registrado para gestionar peticiones a la URL a la cual ha llegado la petición, invocará a dicho programa y le pasará como parámetros objetos Java que contienen toda la información relativa a la petición. Ese es el momento en el cual se comienza a ejecutar nuestro código fuente. Empleando el API que el servidor proporciona con este fin, nuestro programa Java podrá acceder a la información de la petición, procesarla, y generar la respuesta adecuada. Para generar la respuesta adecuada nuevamente el servidor nos proporciona un API compuesta por varios objetos Java. Nosotros invocaremos métodos sobre esos objetos Java, y el servidor finalmente generará una respuesta a la petición a partir de ellos, respuesta que enviará por la red.

En este tutorial emplearemos el servidor de aplicaciones Glassfish 3.1 junto con Netbeans 7.0 para la ejecución de los ejemplos. No obstante, la gran mayoría de la