

Práctica 01

Programación Web estática

Fundamentos de Aplicaciones y Servicios Telemáticos

2º Curso Grado en Ingeniería de Tecnologías de Telecomunicación

Departamento de Ingeniería Telemática (DIT)

Universidad de Sevilla



Ignacio Campos Rivera

Francisco José Fernández Jiménez

José Ángel Gómez Argudo

Francisco Javier Muñoz Calle

Juan Antonio Ternero Muñiz

@2018

ÍNDICE

1	Objetivos y alcance	3
1.1	Introducción.....	3
1.2	Descarga de fuentes.....	3
1.3	Objetivo de la práctica.....	5
1.4	Documentación de apoyo	6
2	Estructura básica de páginas web	6
2.1	Despliegue	9
3	HTML: Usando el lenguaje.....	11
3.1	HTML5.....	11
3.2	Uso de etiquetas de texto.....	13
3.3	Uso de listas	16
3.4	Uso de tablas	16
3.5	Uso de imágenes.....	17
3.6	Uso de enlaces.....	18
3.7	Uso de formularios	20
3.8	XHTML.....	25
3.9	Validación	25
3.10	Más ejemplos y consultas.....	27
4	CSS: Cascading Style Sheets	28
4.1	Selectores	31
4.1.1	Selectores de etiqueta	31
4.1.2	Selectores de clase y de id.....	32
4.1.3	Combinando los selectores	33
4.2	Modelo de cajas.....	34
4.3	Posicionamiento básico	36
4.4	Posicionamiento flotante, maquetación, HTML5 y entornos de diseño web	36
4.5	Validación de CSS.....	38

5	Herramientas de desarrollo (Developer Tools)	38
6	HTTP	40
6.1	Cabecera Location:	40
6.2	Cabecera If- Modified-Since	41
7	XML	42
7.1	Reglas de sintaxis XML	43
7.2	Un fichero XML paso a paso	43
7.3	Bien formado y válido	46
7.4	Espacios de nombres	48
8	Anexo (no evaluable): Validación XML con XML Schema (XSD)	51

1 Objetivos y alcance

1.1 Introducción

Para el desarrollo de las prácticas, es necesario tener conocimientos básicos del sistema operativo UNIX. Puede repasar el apéndice “El Sistema Operativo UNIX” de la asignatura Fundamentos de la Programación I. También es recomendable tener los conocimientos básicos de TCP/IP impartidos en la asignatura Fundamentos de Internet.

1.2 Descarga de fuentes

En este apartado se recogen algunas cuestiones para facilitar el uso del entorno de trabajo y el desarrollo de la práctica.

NOTA: ADVERTENCIA SOBRE LA HERRAMIENTA "COPIAR-PEGAR"

Bien con el enunciado de esta práctica o con otros manuales que pueda encontrar, en ocasiones puede resultar útil la herramienta de "Copiar y Pegar" para hacer uso de los ejemplos escritos en un documento y así evitarse tener que escribirlos a mano. En caso de hacer uso de dicha herramienta, tenga en cuenta que con frecuencia la herramienta "Copiar y Pegar" cambia determinados caracteres (tales como los guiones "-" o las comillas) por otros de similar aspecto, pero que son otros caracteres en realidad, por lo que el intérprete podría dar errores de sintaxis. Si aprecia este comportamiento, revise dichos caracteres en su fichero.

DESCARGA DE LOS EJEMPLOS DE ESTA PRÁCTICA

Como irá observando, esta práctica se basa en el uso de múltiples ficheros. Para descargarlos realice los siguientes pasos:

- 1º Inicie una sesión en la máquina Linux con usuario `dit` (password `dit`).
- 2º Acceda a la página Web de la Asignatura en Enseñanza Virtual y vaya a la misma carpeta en la que se encuentra la memoria de esta práctica, donde encontrará los archivos necesarios para la práctica
- 3º Descargue el archivo "`workspace.tar.gz`". a su máquina local, dentro del directorio "`/home/dit`". Ejecute los siguientes comandos en un terminal para eliminar el directorio antiguo, y descomprimir y extraer el archivo:

```
cd /home/dit  
rm -r workspace  
tar xfvz ./workspace.tar.gz
```

- 4º Descargue el archivo "`FAST_t01-ficheros.tar.gz`" a su máquina local, dentro del directorio "`/home/dit`". Ejecute los siguientes comandos en un terminal para descomprimir y extraer el archivo:

```
cd /home/dit  
  
tar xfvz ./FAST_t01-ficheros.tar.gz
```

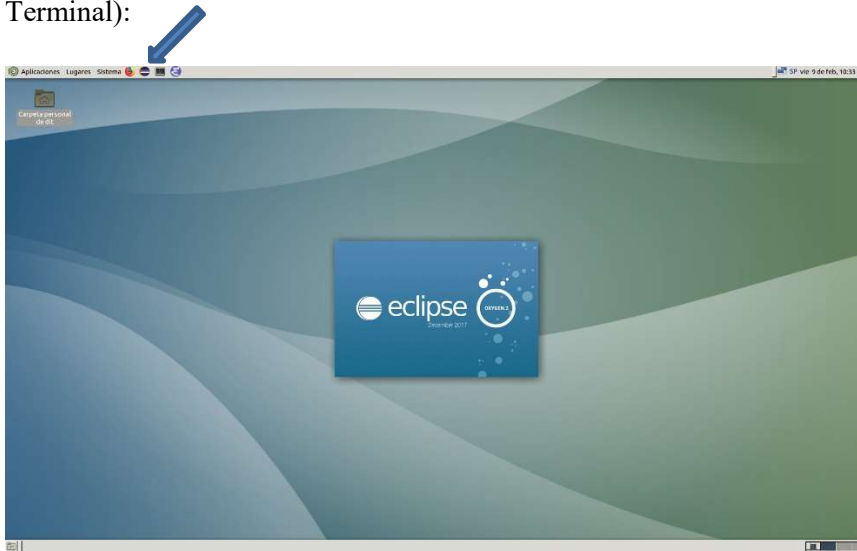
Nota: tar es una utilidad para almacenar/extraer archivos y directorios en un solo archivo. Para almacenar un directorio de forma comprimida en un archivo, su uso es:

```
tar cfvz archivo.tar.gz directorio
```

A partir de ese momento, dispondrá de los distintos ficheros en la carpeta `"/home/dit/workspace/AppWeb/WebContent/p01"`.

Si desea poder mostrar los ficheros de ejemplo en el servidor web, basta arrancar Tomcat (es un contenedor de Servlets que incorpora un servidor web) desde Eclipse (es un IDE o entorno de desarrollo integrado).

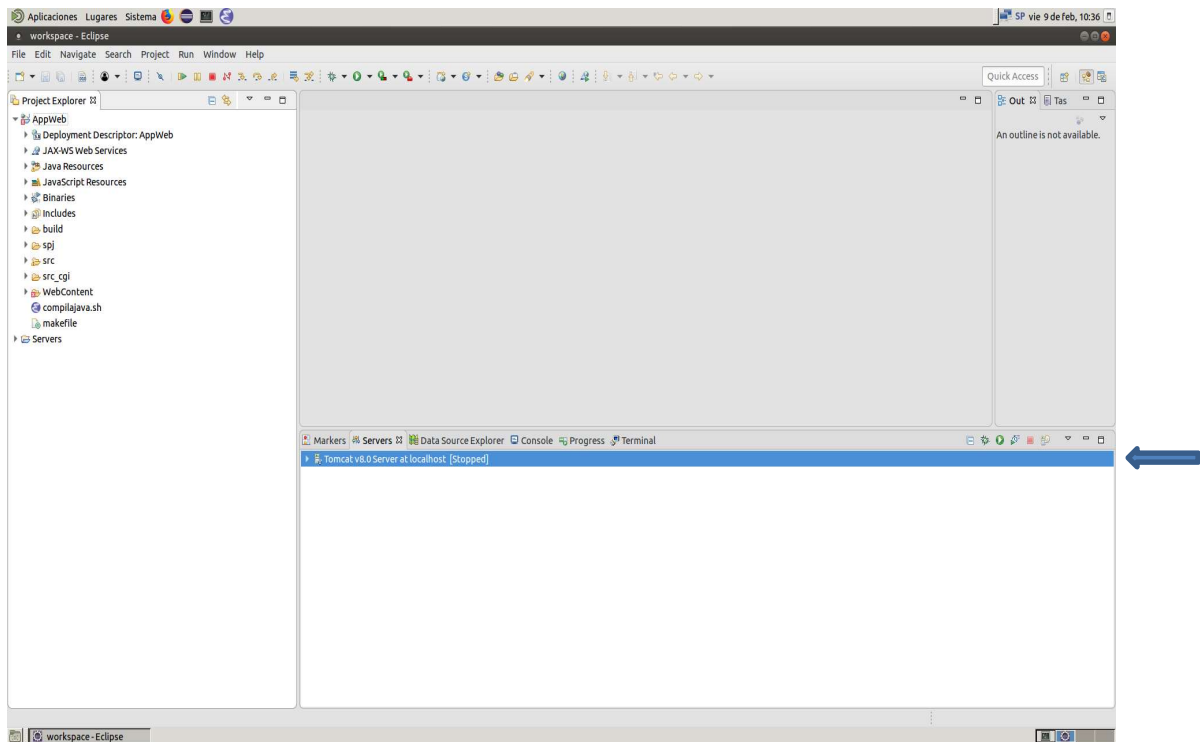
Primero arranque Eclipse con el lanzador cuyo icono está situado arriba a la izquierda (entre Firefox y Terminal):



Y seleccione el directorio de trabajo que aparece por defecto:

```
/home/dit/workspace
```

Una vez arrancado Eclipse, para arrancar Tomcat puede pulsar `Ctrl+Alt+R` o bien pulsar botón derecho sobre la línea correspondiente a Tomcat dentro de la pestaña Servers en la ventana inferior derecha y seleccionar Start:



Ahora podrá acceder a los ficheros desde el navegador Firefox de Mozilla usando la URL:

```
http://localhost:8080/AppWeb/p01/index.html
```

También es posible acceder a los ficheros de ejemplo en “local”, ya que los mismos no necesitan de la ejecución de código en el servidor. Basta usar la URL

```
file:///home/dit/workspace/AppWeb/WebContent/p01/index.html
```

o invocar desde la línea de comandos el navegador y pasarle como parámetro el fichero HTML:

```
dit@localhost:~/workspace/AppWeb/WebContent/p01$ pwd
```

```
/home/dit/workspace/AppWeb/WebContent/p01
```

```
dit@localhost:~/workspace/AppWeb/WebContent/p01$ firefox index.html &
```

Nota: cuando haya que analizar tráfico con Wireshark, puede ir al menú Aplicaciones > Internet > Wireshark o también puede usar la orden `sudo wireshark` desde un terminal. SI usa esta última, le pedirá el password de dit (que es dit), y aunque aparezca el error `Lua: Error during loading`, puede capturar y analizar el tráfico. Una de las formas de empezar a capturar tráfico es haciendo doble click sobre la interfaz any.

1.3 Objetivo de la práctica

El objetivo de esta práctica es introducir al alumno en el desarrollo de aplicaciones basadas en programación web estática. Consecuentemente, esta práctica se centrará inicialmente en la sintaxis de los lenguajes de marcas HTML y XHTML, de las hojas de estilos CSS y en el protocolo HTTP.

Posteriormente se desarrolla el uso del lenguaje de marcas XML y su sintaxis básica, junto al uso de los llamados espacios de nombres.

1.4 Documentación de apoyo

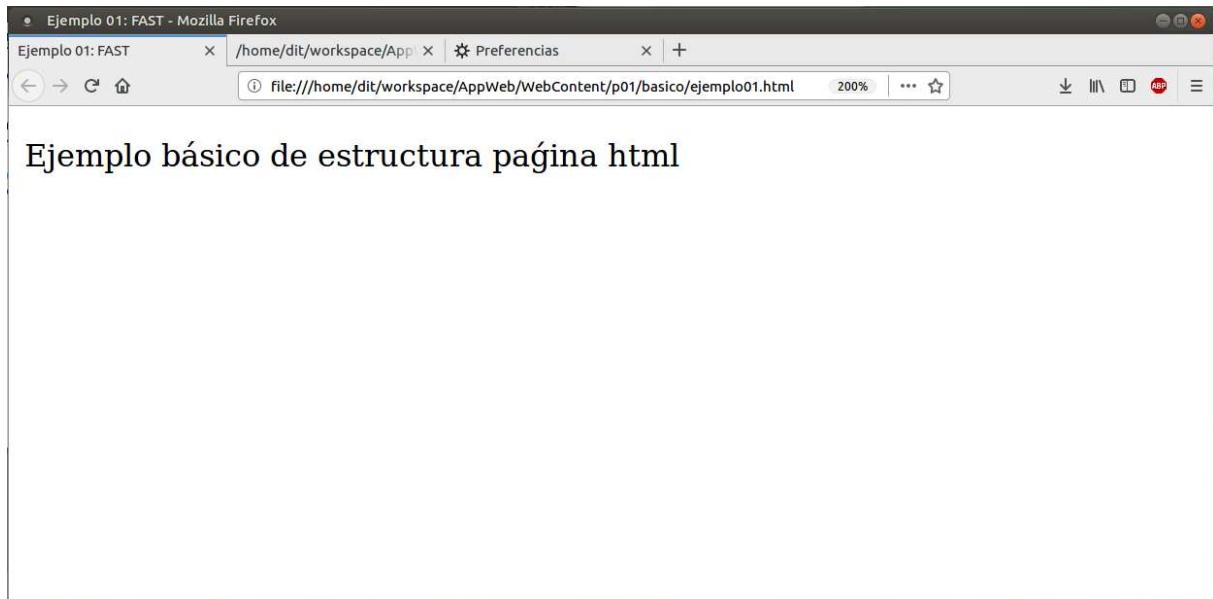
- Tutoriales W3schools HTML, CSS y XML: <http://www.w3schools.com/>
- Tutorial HTTP Mozilla: <https://developer.mozilla.org/en-US/docs/Web/HTTP>
- Estándar HTTP rfc 7230-5: <https://tools.ietf.org/html/rfc7230>
- Estándar HTML: HTML 5 A vocabulary and associated APIs for HTML and XHTML: <http://www.w3.org/TR/html/>
- Estándar HTML: HTML 4.01 <http://www.w3.org/TR/html4/>
- Estándar XHTML1.0: The Extensible HyperText Markup Language (Second Edition): <http://www.w3.org/TR/xhtml1/>
- Estándar CSS: CSS CURRENT STATUS https://www.w3.org/standards/techs/css#w3c_all
- Estándar XML: <http://www.w3.org/TR/xml/>
- Tutoriales Librosweb: <http://www.librosweb.es/>
- Mozilla Developer Tools: <https://developer.mozilla.org/es/docs/Tools/>
- WHATWG community: <https://whatwg.org/>

2 Estructura básica de páginas web

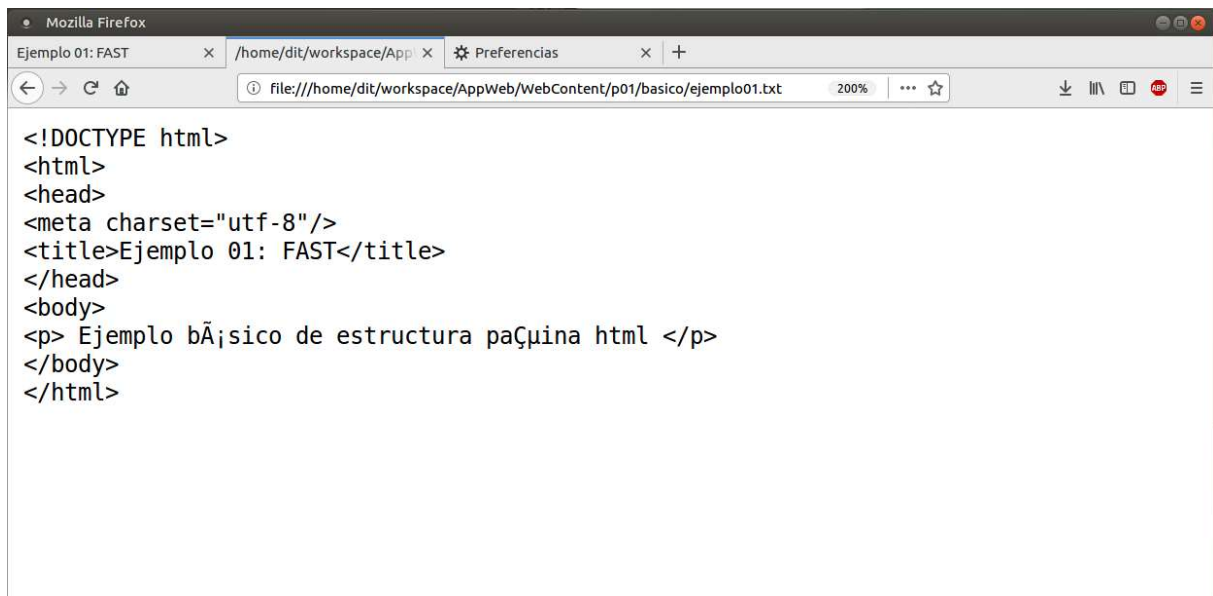
En su forma más básica, una página HTML es un simple fichero de texto que contenga al menos las etiquetas html, head (opcional) y body. Para ayudar a la identificación del contenido a partir del nombre del archivo, y sobre todo para que el navegador lo interprete correctamente, es recomendable que tenga la extensión ".html" (o ".htm"). Por ejemplo, el siguiente fichero sería una página web muy sencilla:

ejemplo01.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8"/>
<title>Ejemplo 01: FAST</title>
</head>
<body>
<p> Ejemplo básico de estructura página html </p>
</body>
</html>
```



Si este mismo fichero se almacena con extensión distinta, por ejemplo .txt, el navegador lo mostrará como un fichero de texto plano, y no “interpreta” las etiquetas html.



TAREAS

1º En un terminal, siendo el directorio de trabajo /home/dit/workspace/AppWeb/WebContent/p01/básico, abra el fichero ejemplo02.html con un editor de texto. Guarde de nuevo el fichero, pero con el nombre ejemplo02.txt.

2º Desde la propia línea de comandos invoque el navegador para que muestre el fichero html:

```
firefox ejemplo02.html &
```

3º Invoque de nuevo el navegador para que muestre el fichero txt. Observe la diferencia.

```
firefox ejemplo02.txt &
```

Observe que si accede al fichero haciendo doble click sobre él en un entorno gráfico o si se lo pasa como argumento al navegador desde un terminal (`firefox ejemplo02.html &`), entonces el acceso al fichero es directo, sin intervención de ningún servidor web (fíjese que la URL del navegador comienza por `file://`).



Recuerde que para poder crear y modificar un sitio web puede editar las páginas con Eclipse, con otros IDEs, e incluso con un editor de texto plano como emacs, ya que los ejemplos de estas prácticas son sencillos.

Por otra parte, si desea comenzar un nuevo fichero HTML, debe incluir su estructura básica y el DOCTYPE (que analizaremos posteriormente). En general los IDEs más completos facilitan plantillas para que al crear un nuevo documento HTML no sea necesario incluir la estructura básica. También puede comenzar a partir de un fichero anterior, o tener ya varios ficheros con dicha estructura. Puede encontrar la estructura básica y los distintos DOCTYPES en esta página del W3C:

<http://www.w3.org/QA/2002/04/valid-dtd-list.html>

A lo largo de esta memoria se puede trabajar con cualquier editor, y aunque los ejemplos se muestren con Eclipse, puede usar otro entorno si lo desea.

Por ejemplo, emacs, al crear un nuevo archivo con extensión html automáticamente indenta las etiquetas de los elementos html.. Bluefish permite crear un nuevo fichero usando la opción *Archivo > Nuevos desde Plantilla > XHTML 1.0*. Y Eclipse incluso permite seleccionar que tipo de documento html desea.

TAREAS

- 1º Abra un fichero nuevo con extensión html con el editor emacs y observe la estructura que añade y cómo destaca la sintaxis.
- 2º Abra el editor bluefish. Cree un nuevo fichero usando la opción Archivo > Nuevos desde Plantilla > HTML5.
- 3º En Eclipse, cree un nuevo fichero pulsando el botón derecho sobre el directorio p01/basico y usando la opción New > HTML File.

2.1 Despliegue

IMPORTANTE

El contenido de este apartado es útil para todas las prácticas de programación Web. Realícelo detenidamente.

Desplegar, implantar o publicar una aplicación web (deploy/publish en inglés) consiste en instalar los archivos necesarios y configurar el servidor web (si fuera necesario) para hacer accesible la aplicación web desde Internet. En general, este proceso puede variar de un servidor web a otro.

El servidor web instalado es el proporcionado por Tomcat. La instalación y configuración del servidor web no entra dentro del alcance de esta asignatura.

En la siguiente tabla se muestran los alias (URL de acceso) ofrecidos por el servidor y los directorios locales donde se buscan los ficheros Web:

URL (alias)	Directorio web
http://ip_equipo:8080/AppWeb/ https://ip_equipo:8080/AppWeb/	/home/dit/workspace/AppWeb/WebContent/

El servidor Web está configurado para usar "index.html" como fichero por omisión. Conforme a esta configuración, por ejemplo:

- "http://IP:8080/AppWeb/" equivale a
 - "http://IP:8080/AppWeb/index.html".
- "http://IP:8080/AppWeb/carpeta/" equivale a
 - "http://IP:8080/AppWeb/carpeta/index.html".

Debe tener en cuenta a qué usuario pertenece el proceso servidor web a la hora de asignar permisos a los ficheros y directorios que quiera publicar. Cuando copie ficheros al "Directorio web", estos deberán ser accesibles por el usuario correspondiente, debiendo tener al menos permiso de búsqueda en los directorios (x) y permiso de lectura en los ficheros (r). En muchas configuraciones el usuario correspondiente al proceso servidor web es normalmente el usuario "www-data", pero para esta asignatura es el usuario "dit" (el que arranca eclipse).

Por último, el servidor web guarda un registro de las acciones que realiza en los ficheros situados en el directorio `/home/dit/tomcat/logs/`

Por ejemplo, para consultar el fichero de log correspondiente al día 9/2/2018 puede utilizar el comando `tail` (muestra las últimas líneas, más información en `man tail`) desde el directorio anterior:

```
tail localhost_access_log.2018-02-09.txt
```

TAREAS

- 1º Compruebe que se han copiado (desplegado) los ficheros en el "Directorio web" del servidor Web para el usuario `dit`, es decir en `"/home/dit/workspace/AppWeb/WebContent"`. Como usuario `dit`, ejecute los siguientes comandos:

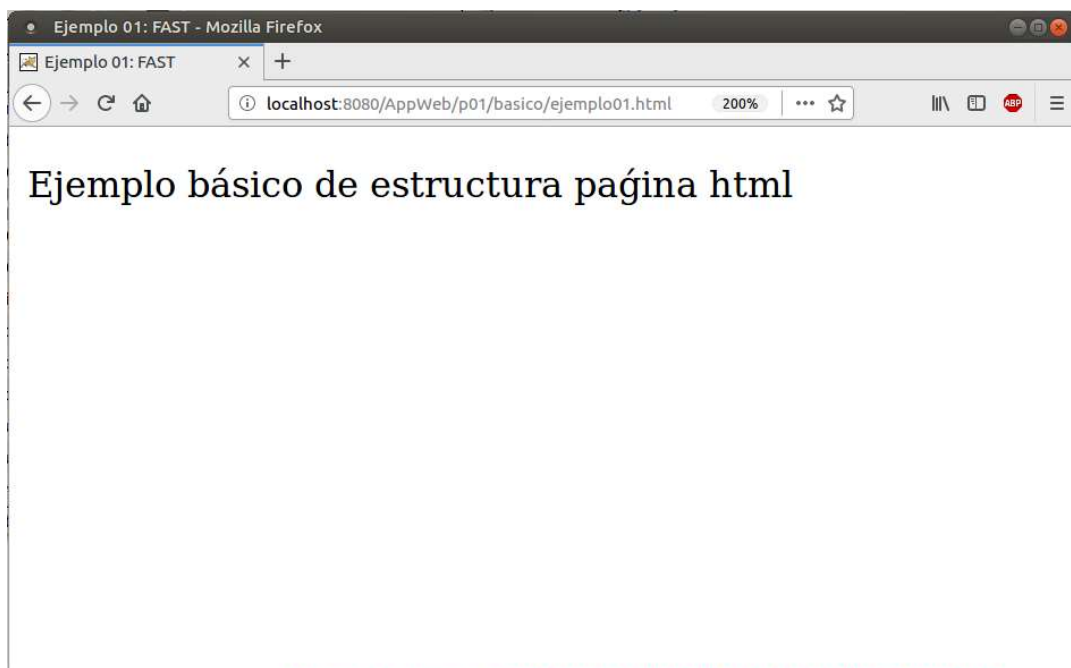
```
cd ~/workspace/AppWeb/WebContent  
ls -l p01
```

- 2º Compruebe que los permisos son adecuados para que el servidor pueda leer estos archivos. En general, permisos 755 a todos los directorios y permisos 644 a todos los ficheros regulares. Arranque el servidor web tal como se indicó anteriormente (Eclipse-Tomcat)

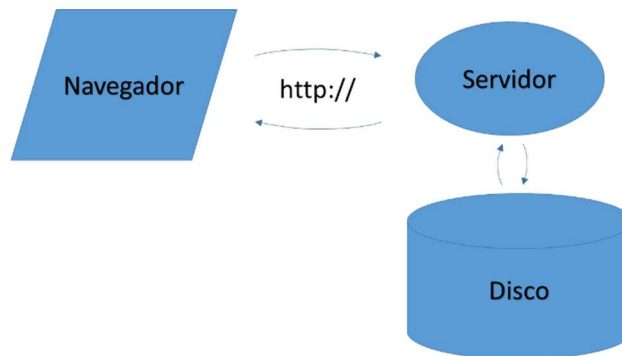
- 3º Desde el navegador acceda a la URL. Desde este índice podrá probar los ejemplos.
`http://localhost:8080/AppWeb/p01/index.html`

También puede consultar otros logs del servidor en la pestaña "Console", que está al mismo nivel que la pestaña "Servers", en la ventana inferior derecha dentro de Eclipse.

Cuando se accede a un fichero usando un navegador, pero a través de un servidor web (aunque está en la misma máquina), el resultado es similar a cuando se accede directamente:



Observe que al acceder al fichero poniendo en el navegador una URL que comienza por `http://`, entonces el acceso al fichero es a través de un servidor web (fíjese que la URL del navegador ya NO comienza por `file://`). Dependiendo de la configuración del navegador puede que no aparezca `http://` en la URL (aunque cuando se usa `https://` sí debe aparecer).



Este método es el que se debe usar SIEMPRE para poder comprobar el correcto funcionamiento.

TAREAS

- 1º Cree un subdirectorio de nombre `subbasico` dentro de `"/home/dit/workspace/AppWeb/WebContent/p01/basico"`.
- 2º Copie el fichero `ejemplo01.html` del directorio básico al subdirectorio `subbasico` y modifíquelo para que contenga su nombre y apellidos en un nuevo párrafo (etiqueta `<p>` y `</p>`).
- 3º Desde un navegador acceda a la URL
`http://localhost:8080/AppWeb/p01/basico/subbasico/ejemplo01.html`

3 HTML: Usando el lenguaje

En este apartado se presentan las principales etiquetas de las que disponemos en HTML y su uso básico.

3.1 HTML5

Aunque HTML4.01 es la versión más extendida, ya se ha publicado una recomendación del W3C que define HTML5. Puede encontrar la última versión de HTML en:

<https://www.w3.org/TR/html/>

También puede consultar la especificación conocida como “Living Standard” de WHATWG:

<https://html.spec.whatwg.org/multipage/>

Las diferencias más visibles son que el DOCTYPE es mucho más simple y que charset pasa a ser un atributo dentro de la etiqueta meta. En HTML5:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
...
```

El equivalente es más complejo en HTML4.01:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
...
```

En HTML5 aparecen nuevas etiquetas, algunas de ellas se verán en el apartado 4.4.

Puede consultar más información en:

http://www.w3schools.com/html/html5_intro.asp

Respecto a las distintas etiquetas de HTML, en teoría se han visto algunas etiquetas de bloque:

Párrafos y Títulos sección	<code><p></code>, <code><h1></code>, <code><h2></code>, <code><h3></code>, <code><h4></code>, <code><h5></code>, <code><h6></code>
Listas	<code></code>, <code></code>, <code></code>
Tablas	<code><table></code>, <code><tr></code>, <code><th></code>, <code><td></code>
Formulario	<code><form></code>
Resaltado texto	<code><blockquote></code>, ...
Varios	<code><body></code>, <code><hr></code>, <code><pre></code>, <code><div></code>, <code><noscript></code>, ...

Y también otras de línea:

Resaltado texto (NO son estilos)	<code></code>, <code></code>, <code><ins></code>, <code></code>, <code><q></code>, <code><sub></code>, <code><sup></code>, ...
Elementos formularios	<code><button></code>, <code><input></code>, <code><label></code>, <code><select></code>, <code><textarea></code>, ...
Varios	<code>
</code>, <code><a></code>, <code></code>, <code></code>, <code><script></code>, ...

Puede consultar su funcionamiento en:

<https://www.w3schools.com/tags/default.asp>

3.2 Uso de etiquetas de texto

El elemento básico a la hora de presentar contenidos textuales en una página Web es el párrafo, (<p>). Recuerde que es una etiqueta de bloque, y que dentro puede tener otras etiquetas anidadas, como aquellas que destacan de alguna forma palabras o frases dentro del párrafo.

A continuación, puede ver el contenido del fichero ejemplo02.html y la presentación que hace el navegador.

ejemplo02.html

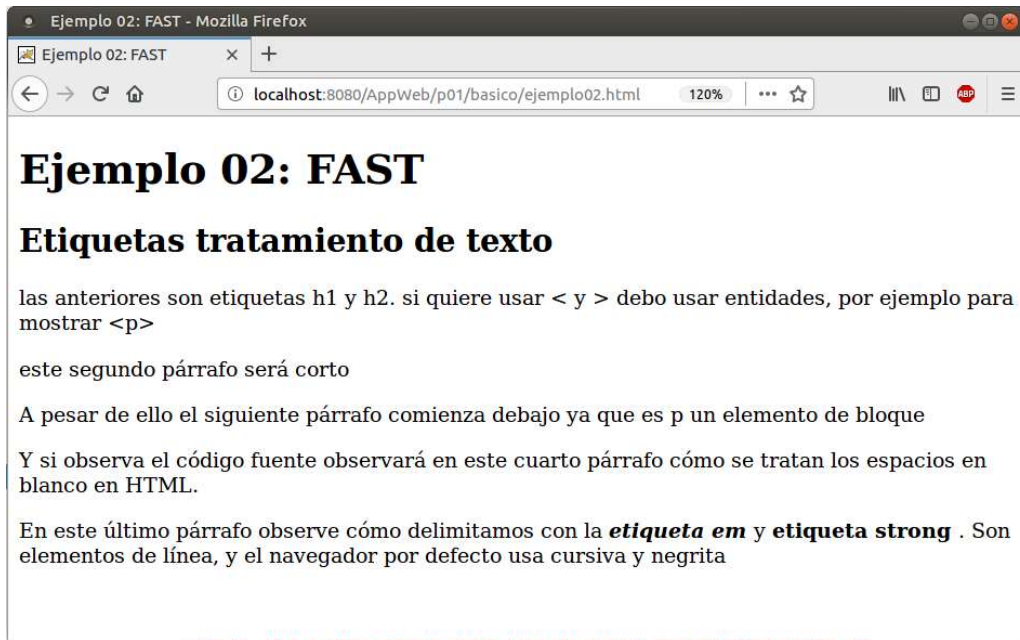
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Ejemplo 02: FAST</title>
</head>
<body>
<h1> Ejemplo 02: FAST</h1>
<h2> Etiquetas tratamiento de texto </h2>
<p> las anteriores son etiquetas h1 y h2.
si quiere usar &lt; y &gt; debo usar entidades,
por ejemplo para mostrar &lt;p>&gt;</p>
<p> este segundo párrafo será corto </p>
<p> A pesar de ello el siguiente párrafo comienza debajo
ya que es p un elemento de bloque</p>
<p> Y si observa el código fuente

    observará en este cuarto
párrafo                cómo

se tratan los espacios    en                blanco
en                        HTML.</p>

<p> En este último párrafo observe cómo delimitamos
con la <em> etiqueta em </em> y <strong> etiqueta strong </strong>.
Son elementos de línea, y el navegador por defecto usa cursiva y
negrita </p>

</body>
</html>
```



TAREAS

- 1º Edite el fichero `ejemplo02.html` que está dentro de la carpeta de ficheros de la práctica. Muestre dicho fichero en el navegador. Estudie el código fuente del mismo (botón derecho del ratón, opción *Ver código fuente de la página* o CTRL+U), y el resultado al visualizarlo. Observe las diferencias entre elementos de línea o bloque, y cómo se hace el tratamiento de espacios en blanco en HTML.
- 2º Modifique el fichero `ejemplo02.html`, y sustituya las etiquetas `<p>` del cuarto párrafo por la etiqueta `<pre>` (preformateado). Vuelva a cargar el fichero en el navegador (p.ej. botón de recarga a la derecha de URL) y estudie las diferencias.
- 3º Sustituya la etiqueta `<pre>` anterior por la etiqueta `<code>` (código). Encuentre y estudie las diferencias.
- 4º Aplique a algunas palabras del texto las etiquetas `<ins>` (insertado) y `` (borrado). ¿Son etiquetas de línea o de bloque? ¿Qué efecto tienen en la presentación?

Nota: Para pequeños cambios se puede editar html en el propio Firefox, para ello puede pulsar la tecla F12. Tenga en cuenta que al hacer esto lo que se modifica es una copia en memoria, no el fichero. Si desea insertar etiquetas, seleccione el elemento dentro de "Inspector" y use la opción "Editar como html" con el botón derecho. Encontrará más información en el apartado Herramientas de desarrollo: Developer Tools)
- 5º Sustituya las etiquetas de un elemento `<p>` por etiquetas `<blockquote>`. Marque o delimite una palabra de un párrafo con la etiqueta ``. Tras ver la presentación en el navegador. ¿Son etiquetas de línea o de bloque? ¿Qué efecto tienen en la presentación?
- 6º Elimine el elemento `<meta charset="utf-8" />` ¿qué ocurre con las tildes?.

3.3 Uso de listas

Si desea añadir una lista a su página deberá usar la etiqueta ``, (o la etiqueta `` si desea una lista ordenada). Estas etiquetas son de bloque. Para cada uno de los componentes de la lista deberá usar la etiqueta ``. Observe que en HTML no es necesario cerrar el elemento ``, ni siquiera que este aparezca en minúsculas. En XHTML ambos casos provocarían que el documento no fuera válido.

```
<ul>
  <li> primer item de la lista </li>
  <li> segundo item de la lista.</li>
  <li> tercer item de la lista.</li>
</ul>
```

Desde HTML 4.x se pretende que HTML no incluya capa de presentación (para que ésta sea controlada exclusivamente por hojas de estilo CSS). Dado que en las versiones anteriores de HTML como la 3.2 esto no se logró (mezclando estructura y presentación) y que HTML 4.x intenta ser compatible hacia atrás, atributos como "type" y "value" (en "li", por ejemplo), están marcadas como "deprecated", esto es, se mantienen por compatibilidad hacia atrás, pero se desaconseja su uso. Por esto se pueden modificar las viñetas directamente en HTML, con el atributo "type" y los valores "disc", "circle" o "square". Del mismo modo puede cambiar la forma de numerar una lista ordenada usando el atributo "type", y valores como "I", "a", "A", "i", etc. Por último, es posible anidar listas dentro de listas.

TAREAS

- 1º Modifique el fichero `ejemplo02.html` y añada una lista al comienzo del mismo. Compruebe su visualización.
- 2º Modifique la lista previa para que sea una lista ordenada. ¿Necesita modificar los apartados que forman parte de la lista?
- 3º Modificar el segundo punto de la lista previa para que a su vez sea una lista. El aspecto final debe ser similar a éste:
 - 1.Revisar cabecera
 2. Revisar etiquetas de cierre
 - A. Errores típicos en `
` y `<hr>`
 - B. Revisar elementos ``
 3. Revisar case-sensitive

3.4 Uso de tablas

El elemento `<table>` permite organizar la información en filas y columnas. Se apoya en los elementos `<tr>`, "table row", y `<td>`, "table data cell", y su uso es muy intuitivo. `<tr>` es el elemento contenedor de las filas, y cada una de las filas tendrá elementos `<td>` que contienen los datos de cada celda de la tabla. También es posible usar la etiqueta `<th>`, "tabla header", similar a `<td>`, pero que marca dicha celda como cabecera de una fila o columna (adicionalmente el navegador presenta el contenido centrado y destacado en negrita). La etiqueta `<caption>`, que sigue a la etiqueta `<table>`, permite ponerle un título a la tabla.

Es posible combinar varias celdas en una única celda. En horizontal se usa el atributo `colspan`, al que se le indica mediante un valor de tipo entero el número de celdas a su derecha que combina. Nótese que las celdas que "sobran" no se deben poner.

También es posible combinar en vertical, usando el atributo `rowspan` de una forma similar.

ejemplotablas.html

```
...
<!-- <table border="1" align="center"> No soportado en HTML5 -->
<table>
<caption> Título de la tabla</caption>
<tr>
  <th>col 1</th><th>col 2</th><th>col 3</th><th>col 4</th><th>col 5</th>
</tr>
<tr>
  <td colspan="2">1.1</td><td>1.3</td><td>1.4</td><td>1.5</td>
</tr>
<tr>
  <td>2.1</td><td>2.2</td><td>2.3</td><td rowspan="3">2.4</td><td>2.5</td>
</tr>
<tr>
  <td>3.1</td><td>3.2</td><td>3.3</td><td>3.5</td>
</tr>
<tr>
  <td>4.1</td><td>4.2</td><td>4.3</td><td>4.5</td>
</tr>
</table>
...
```

Título de la tabla				
col 1	col 2	col 3	col 4	col 5
1.1		1.3	1.4	1.5
2.1	2.2	2.3		2.5
3.1	3.2	3.3	2.4	3.5
4.1	4.2	4.3		4.5

3.5 Uso de imágenes

Observe que la etiqueta `` es una etiqueta vacía, no tiene etiqueta de cierre. Si estamos usando XHTML todas las etiquetas deben estar cerradas. Para ello cerramos la etiqueta con `</>` al final de la etiqueta de inicio (es aconsejable dejar un espacio antes de `</>` por compatibilidad con navegadores antiguos).

```
<!-- Válido HTML -->
```

```

```

```
<!-- Válido XHTML -->
```

```

```

La norma no especifica los formatos soportados, pero los más aceptados son JPG, GIF y PNG. Por otro lado es conveniente especificar con los atributos `width` y `height` la anchura y altura con la que se muestra la imagen. Esto permite que el navegador conozca antes de cargar la imagen el tamaño que ocupará la misma en pantalla.

TAREAS

- 1º Obtenga una imagen (formato jpg, png o gif) , por ejemplo de la página web de la ESI o de la Universidad de Sevilla. Descárguela en local. Modifique la página `ejemplotablas.html` para que incluya esa imagen antes de la tabla.
- 2º Modifique el ejemplo previo, indicando **incorrectamente** la localización del archivo. Configure un texto alternativo (atributo `alt`) que deberá aparecer en lugar de la imagen.
- 3º Incluya en el ejemplo anterior otra imagen, pero no copie la imagen en local, sino que la referencia debe ser su localización externa. Puede obtener la referencia de una imagen en una página, con el botón derecho, copiar localización de la imagen.

La URL que se usa en el atributo `src` puede ser absoluta o relativa, y es similar a la que se usa en los enlaces (ver siguiente apartado).

3.6 Uso de enlaces

Los enlaces se utilizan para establecer relaciones entre dos recursos. Aunque la mayoría de enlaces relacionan páginas web, también es posible enlazar otros recursos como imágenes, documentos y archivos (los enlaces pueden ser a ficheros html o de otros tipos, como .pdf, .txt, etc...). O incluso se pueden usar otros protocolos, como https, ftp, etc. Un posible enlace puede ser:

```
<a href="ftp://ftp.rediris.es/pub/FreeBSD/ls-lR.gz">enlace</a>
```

El elemento usado es `<a>`, y su atributo principal es `href`.

Los enlaces a otro recurso pueden ser caminos absolutos o relativos y residir en distinta o en la misma máquina. En el caso de absolutos comienzan con `http://` seguidos de la dirección de la máquina y el camino al recurso, por ejemplo:

```
http://www.us.es/acerca/index.html
```

Si comienza con `/` es relativo al host donde está el fichero que contiene el enlace, como `/acerca/index.html`. Si no comienza con los anteriores es relativo a la localización del fichero que contiene el enlace, `teoria/index.html`. Si comienza con el símbolo `#` se refiere a una sección del documento, como la referencia `#apartado03`.

Además puede usar los conocidos `“..”` para acceder al directorio padre del actual.

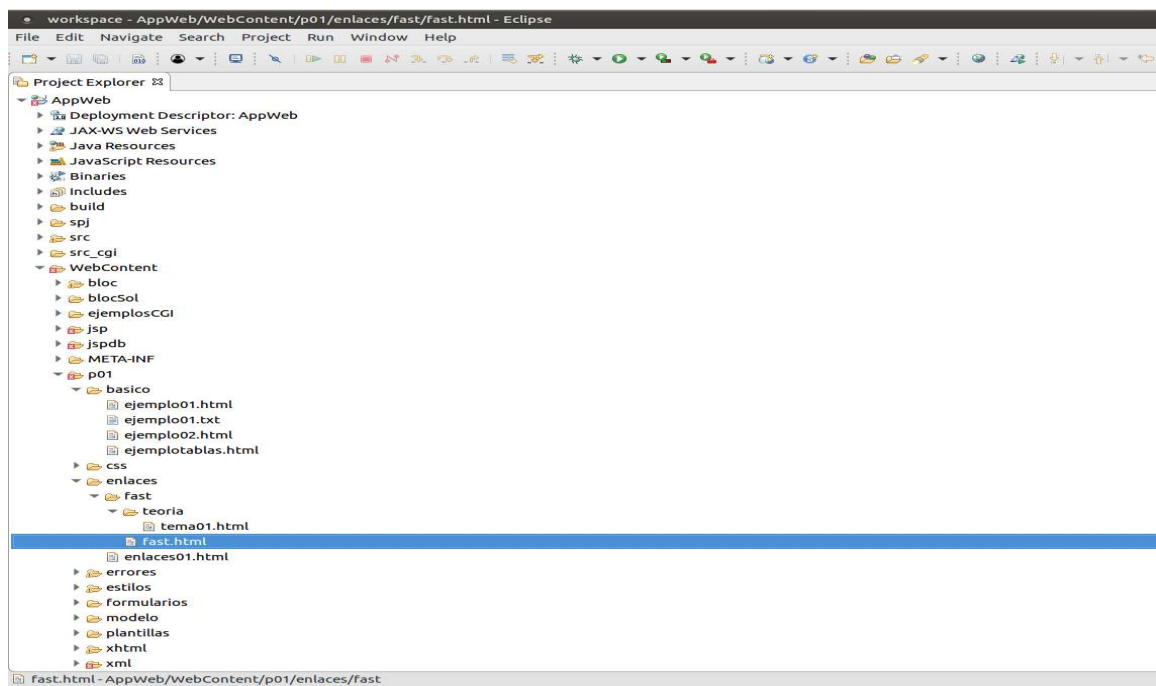
enlaces01.html

```

<h1> Absolutos</h1>
<ul>
<li> <a href="http://www.google.es">Google</a> Enlace Absoluto. </li>
<li> <a href="https://ev2.us.es">Enseñanza virtual</a> Enlace seguro </li>
</ul>
<h1> Relativos</h1>
<ul>
<li> <a href="fast/fast.html">Fast</a> Un nivel por debajo</li>
<li> <a href="fast/teoria/tema01.html">Tema 01 de fast</a> dos niveles por debajo</li>
<li> <a href="fast/teoria/tema01.html#apartado03">Tema 01 de fast</a> dos niveles por debajo y salta al apartado 03</li>
<li> <a href="fast/teoria/tema01.html#apartado04">Tema 01 de fast</a> dos niveles por debajo y salta al apartado 04 mediante id</li>
</ul>

```

Puede observar enlaces absolutos y enlaces relativos. La estructura de directorios es:



Para definir un enlace a una zona determinada de una página hay que previamente fijar una marca en la zona destino. Esto se puede hacer de la siguiente forma¹:

- Mediante el atributo `id` aplicado a un elemento ya existente en la página.

Define destinos

¹ Hasta HTML4.01 se podía hacer también con una etiqueta `<a>` con atributo `name` y sin atributo `href`, pero no en HTML5.

```

...
<a name="apartado03"></a> <!-- no en HTML5 -->
<h2> Apartado 03</h2>
...
<h2 id="apartado04"> Apartado 04</h2>
...

```

Y en los enlaces se usa como destino (el valor del atributo href) la marca correspondiente, precedida del carácter “#”

enlaces locales a la página

```

...
<li> <a href="#apartado03"> Apartado 03</a> </li>
<li> <a href="#apartado04"> Apartado 04</a> </li>
...

```

TAREAS

- 1º Analice los ficheros de la carpeta enlaces, los tipos de enlaces definidos y la estructura de directorios usada. Pruebe que todos los enlaces funcionan a partir de la página `enlaces01.html`
- 2º Modifique el fichero `tema01.html` para que incluya enlaces a los apartados 1 y 2 de la propia página. Ya existen los enlaces a los apartados 3 y 4.
- 3º Incluya en el fichero `enlaces01.html` un enlace a un fichero externo en formato pdf, por ejemplo al calendario académico de este curso, que se encuentra en la página web de la ESI, <http://www.esi.us.es/>

Aunque el atributo `id` (que es global y también el atributo `class`) es de los más usados, puede consultar una lista de atributos globales (se pueden aplicar a cualquier elemento) en:

http://www.w3schools.com/tags/ref_standardattributes.asp

En cambio, el atributo `href` se aplica al elemento `<a>` o al elemento `<link>`, entre otros, pero no a cualquiera:

```

http://www.w3schools.com/tags/tag_a.asp
http://www.w3schools.com/tags/att_a_href.asp
http://www.w3schools.com/tags/tag_link.asp
http://www.w3schools.com/tags/att_link_href.asp

```

3.7 Uso de formularios

Para comprender el funcionamiento de los formularios deberemos trabajar al menos en dos ficheros: uno con el formulario, y otro con código que reciba los datos del formulario y los procese. El segundo fichero será una página codificada en JSP (u otro lenguaje dinámico de ejecución en el servidor). Esta segunda página se especifica (como verá posteriormente) en el atributo `action` del formulario.

En esta práctica usaremos un ejemplo con código JSP, que recibe los valores del formulario y simplemente los presenta en pantalla. Profundizará en programación en servidor web en prácticas posteriores.

El elemento contenedor de un formulario es el elemento `<form>`, y sus atributos más importantes son `action` y `method`. El atributo `action` indica la URL que se encargará de procesar los datos que se han introducido en el formulario, y de posiblemente generar una respuesta.

El atributo `method` establece cómo se envían los datos del formulario al servidor. Las dos posibilidades que nos ofrece HTTP son GET y POST. Ya conoce las diferencias entre ambos. En el caso de POST los valores viajan dentro del mensaje HTTP, como parte de los datos. En el caso de GET, los parámetros viajan en la propia URL de la petición. Por lo tanto, el atributo `method` tendrá los valores `get` o `post` en función de la cantidad de datos que se envíen o de la privacidad que le quiera dar a los mismos.

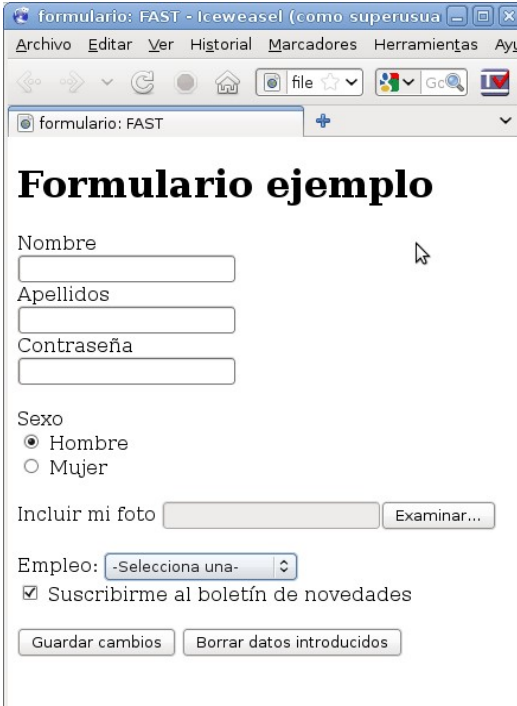
```
<form method="post" action="formulario_cv_verdatos.jsp" >
```

El elemento final de todo formulario suelen ser dos botones. El primero permite el envío de los datos y el segundo eliminar los datos introducidos o seleccionados previamente. Son elementos `<input>` con atributo `type` con los valores `submit` o `reset`.

```
<input type="submit" name="enviar" value="Enviar datos" />
```

```
<input type="reset" name="limpiar" value="Borrar datos" />
```

El resto del formulario serán las entradas de datos solicitadas, construidas con el elemento `<input>`, que tiene como uno de sus atributos más importantes el nombre, `name`. El valor del atributo `name` será el que utiliza la aplicación del servidor para obtener el valor de cada campo del formulario. Otro atributo imprescindible es el tipo de la entrada, `type`, que puede tener los valores `text`, `password`, `checkbox`, `radio`, etc. Observe en la figura y en el fichero distintas posibilidades a la hora de construir un formulario



The screenshot shows a web browser window with the title 'formulario: FAST - Iceweasel (como superusua)'. The browser's address bar shows 'file'. The page content is a form titled 'Formulario ejemplo'. The form contains the following elements:

- Three text input fields labeled 'Nombre', 'Apellidos', and 'Contraseña'.
- A 'Sexo' section with two radio buttons: 'Hombre' (selected) and 'Mujer'.
- An 'Incluir mi foto' section with a file input field and an 'Examinar...' button.
- An 'Empleo:' section with a dropdown menu showing '-Selecciona una-'.
- A checkbox labeled 'Suscribirse al boletín de novedades' which is checked.
- Two buttons at the bottom: 'Guardar cambios' and 'Borrar datos introducidos'.

formulario_cv.html

```
...  
<form action="" method="post">  
Nombre <br/>  
<input type="text" name="nombre" /> <br/>  
Apellidos <br/>  
<input type="text" name="apellidos" /><br/>  
Contraseña <br/>  
<input type="password" name="contrasena"/><br/>  
<br/>  
Sexo <br/>  
<input type="radio" name="sexo" value="hombre"  
    checked="checked" /> Hombre <br/>  
<input type="radio" name="sexo" value="mujer" />  
Mujer <br/>  
<br/>  
Incluir mi foto  
<input type="file" name="foto" /><br/>  
...  
<input type="submit" name="enviar"  
    value="Guardar cambios" />  
<input type="reset" name="limpiar"  
    value="Borrar datos introducidos" />  
</form>  
...
```

TAREAS**En esta tarea será necesario utilizar un servidor web.**

- 1º Usará el fichero formulario_cv.html. Compruebe su visualización en un navegador externo. Rellene los datos del formulario y pulse el botón de enviar. El formulario no hará nada, ya que no está definida la URL que procesará los datos del formulario. Antes de seguir, compruebe que el directorio `formulario` está accesible para el servidor web en `/AppWeb/p01`.
- 2º Compruebe que el fichero `formulario_cv_verdatos.jsp` está en el directorio anterior, junto al fichero `formulario_cv.html`. Haga que en `formulario_cv.html` la URL que procesa los datos sea `formulario_cv_verdatos.jsp`. Tenga en cuenta que la referencia puede ser relativa (sólo el nombre del fichero), o absoluta. Compruebe los permisos y arranque el servidor web y el servidor jsp con Tomcat-Eclipse (tal como se indicaba al principio de la práctica).
- 3º Desde el navegador Firefox, acceda al formulario indicando la ruta: `http://localhost:8080/AppWeb/p01/formularios/formulario_cv.html`. Rellene los datos y pulse enviar. El navegador ahora mostrará los datos que ha recibido del formulario.

Si repite el ejercicio anterior capturando el tráfico en el interfaz local sobre el que está haciendo las pruebas, obtendremos que el mensaje HTTP que el navegador envía al servidor al pulsar el botón de enviar es:

captura_post

```

...
POST /AppWeb/p01/formularios/formulario_cv_verdatos.jsp
HTTP/1.1
Host: localhost:8080
User-Agent: ...
Accept: ...
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://localhost:8080/AppWeb/p01/formularios/formulario_cv.h
tml
Cookie: JSESSIONID=5B6E7B514257A2E98DE9A6525C8BE9AB
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 134

nombre=Alfredo&apellidos=Garc%C3%ADa&contrasena=12345678&sex
o=hombre&foto=&empleo=Estudiante&suscribir=suscribir&enviar=
Enviar+cambios
...

```

Los parámetros viajan en el cuerpo del mensaje HTTP. Observe que es una cadena de pares `name=valor` separados por el carácter `&`.

TAREAS

- 1º Inicie una captura de tráfico (mediante *wireshark*) en la interfaz local, acceda al formulario usando el navegador, y envíe los datos. En la captura compruebe el funcionamiento del método `POST`.
- 2º Modifique el `formulario_cv.html` para que el método HTTP de envío sea `GET` (`method="get"`). Inicie la captura de tráfico, rellene y envíe el formulario, y observe en el navegador la URL de la dirección a la que está accediendo. Analice el tráfico y compruebe cómo se envían los parámetros en el mensaje HTTP.

El resultado de la captura debe ser similar al siguiente. Observe los parámetros justo a continuación del `GET`, en la primera línea. El cuerpo del mensaje HTTP está vacío. Y en la barra de direcciones, al final del nombre del fichero, aparecen todos los parámetros enviados, separados por el carácter `?`.

`http://localhost:8080/AppWeb/p01/formulario/formulario_cv_verdatos.jsp?nombre=Ana&apellidos=Rivera&contrasena=1234&sexo=mujer&foto=&empleo=Ingeniero&suscribir=suscribir&enviar=Enviar+cambios`

captura_get


```
GET
/AppWeb/p01/formulario/formulario_cv_verdatos.jsp?nombre=Ana&apellidos=Rivera&contrasena=1234&sexo=mujer&foto=&empleo=Ingeniero&scribir=suscribir&enviar=Enviar+cambios HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 ...
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-es,es;q=0.8,en-us;q=0.5,en;q=0.3
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://localhost:8080/AppWeb/p01/formulario/formulario_cv.html
```

La información de los formularios se envía mediante la codificación URL que convierte los caracteres al juego de caracteres ASCII, reemplazando los que no son ASCII con un “%” seguido de 2 dígitos hexadecimales y los espacios con un signo más “+” o con %20. Consulte en la siguiente página en qué se convierte un %:

http://www.w3schools.com/tags/ref_urlencode.asp

3.8 XHTML

Ya se han indicado en teoría las principales diferencias entre HTML y XHTML. Recordemos que las más importantes son que en XHTML:

- todos los identificadores de elementos, atributos y valores reservados se especifican en minúsculas.
- todos los elementos deben tener una etiqueta de cierre. En HTML es opcional cerrar elementos como ``, `<hr>`, `
`, etc.
- todos los atributos deben tener asociado un valor de forma explícita (no se permiten atributos sin valor) y su valor estar delimitado por comillas dobles o simples.
- se distinguen mayúsculas y minúsculas (case sensitive).
- DOCTYPE es obligatorio, y el atributo `xmlns` en `<html>` también.
- `<html>`, `<head>`, `<title>`, and `<body>` son obligatorios.

Por lo tanto, una página HTML que no cumpla estas restricciones adicionales no pasaría el control de validez de XHTML. Esto es interesante porque aunque el navegador de un ordenador podría presentarla, el navegador de un dispositivo con menos recursos podría tener problemas.

El DOCTYPE y la etiqueta `html` con su atributo `xmlns` para XHTML 1.1 es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

Busque cuál sería el equivalente para XHTML 1.0 en:

<http://www.w3.org/QA/2002/04/valid-dtd-list.html>

TAREAS

- 1º Muestre en el navegador el fichero `demo_xhtml_01_10Trans.xhtml`, y averigüe el estándar de XHTML y la versión o tipo (strict, transitional, frames).
- 2º Muestre en el navegador el fichero `demo_xhtml_01_10Trans_con_errores.xhtml`, e intente averiguar todos los errores (más adelante utilizará una herramienta para ello). Intente corregir el primer error.
- 3º Cambie la extensión del fichero anterior por `html` y muéstrelo en el navegador. ¿Se muestra algún error ahora?

3.9 Validación

Incluso una vez especificada la norma o estándar a seguir por una página HTML 4.01, entre las distintas versiones o tipos (strict, transitional, frameset) hay diferencias. El tipo de documento, especificado con DOCTYPE, nos determina qué estándar, y dentro de éste, qué versión o tipo del mismo debería “respetar”. Por ejemplo, el siguiente código HTML

```
...
<body> texto no delimitado por etiqueta
<p identificador="error" color="noexiste"> párrafo cerrado con P,
atributo que no existe y con valor de atributo no válido </P>
<elemento_inventado> Este elemento no existe
</elemento_inventado>
</BODY>
```

provoca que al validar, el número de avisos cambie en función de que el tipo de documento sea strict o transitional.

A la hora de validar puede usar las herramientas que incorpore su IDE, herramientas de validación on-line como las del W3C (<http://validator.w3.org/>), herramientas integradas en el navegador (Firefox y HTML Validator), e incluso utilidades en línea de comandos como `tidy` (vea `man tidy`). `tidy` tiene como propósito principal validar, corregir y producir una salida en formato “pretty-print”.

TAREAS

- 1º Proceda a validar los ficheros (ubicados en el directorio errores de la práctica) `error01.html` y `error02.html` con la herramienta de validación del W3C: <http://validator.w3.org/>. Observe que la única diferencia entre ellos es el tipo de documento, DOCTYPE, y compruebe que al validar los dos tienen errores, pero en el segundo (strict) aparece un error más: en modo estricto no se permite texto en el body que no esté dentro de un elemento contenedor como `p`, `h1`, etc. Compruebe que **la presentación en el navegador no muestra ningún error** en ninguno de los dos casos.
- 2º Valide los ficheros `prueba01.html` y `prueba02.html` con la herramienta online del W3C. Observe cómo funciona la validación de tidy respecto al doctype del documento.
- 3º Valide el fichero `demo_xhtml_01_10Trans_con_errores.xhtml` con la herramienta online del W3C. Observe los errores encontrados.
- 4º Valide el fichero `index.html` (HTML5) con la herramienta online del W3C. Observe el resultado. Copie el fichero anterior en otro de nombre `index5.html`. Elimine la línea que contiene la etiqueta meta. Valide este otro fichero ahora ¿qué errores encuentra?
- 5º Instale el plugin para Mozilla Firefox HTML Validator, que permite comprobar la validez del documento HTML. Una vez instalado, marca con un icono el número de errores. Si hace clic sobre el mismo, se abre la vista de código fuente, y en esa vista se muestra información detallada de los errores. Puede acceder e instalar a este complemento desde
<http://users.skynet.be/mgueury/mozilla/>
<https://addons.mozilla.org/es/firefox/addon/html-validator/>
- 6º Evalúe las distintas opciones de tidy (vea las opciones con `man tidy`) con los ficheros `prueba01.html` y `prueba02.html`.

```
tidy -e prueba01.html
```

```
tidy -q -e prueba02.html
```

3.10 Más ejemplos y consultas

Observe en la siguiente URL que en la barra de navegación de la izquierda puede acceder a numerosos ejemplos prácticos.

<http://www.w3schools.com/html/>

Recorra alguna de las secciones, y utilice las utilidades “Try it yourself”, que permiten editar código html y ver cómo se presentaría en un navegador. Observe que en el apartado “HTML Reference” tiene una lista de todas las etiquetas disponibles, atributos, etc. En esta práctica solo se presentan algunas de ellas.

- 1º Navegando por www.w3schools.com puede llegar a encontrar la lista con todas las etiquetas html:

<http://www.w3schools.com/tags/default.asp>

- 2º Busque en esa lista una etiqueta que sea nueva en HTML5 (por ejemplo `<aside>`) y averigüe su significado. Compruebe que aparece especificada en la norma:

<https://www.w3.org/TR/html5/>
<https://www.w3.org/TR/html5/sections.html#the-aside-element>

- 3º Haga lo mismo con la etiqueta `<article>`.

- 4º Busque ahora una etiqueta que no esté soportada en HTML5 (por ejemplo ``). Compruebe que aparece especificada en la norma anterior:

<https://www.w3.org/TR/1999/REC-html401-19991224/>
<https://www.w3.org/TR/1999/REC-html401-19991224/index/elements.html>

- 5º Haga lo mismo con la etiqueta `<center>`.

4 CSS: Cascading Style Sheets

“Cascading Style Sheets” (CSS) es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores. Puede consultar el estándar:

<https://www.w3.org/standards/techs/css>

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información de estilo puede ser adjuntada como un documento separado, y es lo aconsejable.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

Observe en un primer ejemplo cómo se define una hoja de estilos y el aspecto que tiene:

base.css

```
h1 { color: blue; text-align: center; }
p  { color: red; background-color: grey}
```

Observe que hay dos reglas, la primera se aplica a los elementos con etiqueta `<h1>` (cabeceras de nivel 1), y provoca que el color del contenido sea azul, y que el texto esté centrado. La segunda se aplica a los elementos con etiqueta `<p>` (párrafos) y provoca que el color del contenido sea rojo con un fondo gris.

A la primera parte de cada regla se la conoce como selector, y a la segunda, la delimitada por las llaves, declaración:

- Selector: En teoría se han expuesto distintas posibilidades, pero la más simple es usar una etiqueta, y la declaración se aplica a todos los elementos que coincidan con esa etiqueta.
- Declaración: En ella tendremos una lista, separada por “;” de parejas propiedad y valor, separadas a su vez por “:”. Cada propiedad permite modificar alguna característica del aspecto de un elemento, y el valor indica el valor que toma a partir de ese momento esa característica.

Para aplicar estilos a una página HTML lo aconsejable es definir los estilos en un fichero separado, cuyo contenido serían las reglas como las del ejemplo visto previamente. Además deberá enlazarlo con el fichero html, utilizando la etiqueta `link` en la cabecera del documento html:

```
<link href="base.css" rel="stylesheet" type="text/css" />
```

Otras posibilidades (desaconsejadas) son: Definir las reglas en la propia página html, en la sección `<head>` utilizando la etiqueta `<style>`, o definir las propiedades en el lugar de su aplicación en la etiqueta de inicio del elemento al que se aplican usando el atributo `style`.

TAREAS

- 1º Edite el fichero `estilos01.html`. Analice su contenido, y observe la presentación por defecto que hace el navegador, ya que la zona de estilos está comentada.
- 2º Elimine los comentarios que delimitan la etiqueta `style` y guarde los cambios. Visualice ahora la página, viendo el efecto de aplicar los estilos.
- 3º Copie el fichero `estilos01.html` en `estilos03.html`. Cree un nuevo fichero, `estilos03.css`, cuyo único contenido serán las reglas de estilo que aparecen dentro de la etiqueta `style`. Elimine la etiqueta `style`, y en su lugar utilice el elemento `link` para enlazar `estilos03.html` con `estilos03.css`.

La hoja de estilos, como su propio nombre indica, se aplica en cascada: Si no define reglas para un determinado elemento se aplica la hoja de estilos por defecto. Si sólo define un valor para una propiedad de un elemento, el resto de propiedades se mantienen. Si define una propiedad (que sea heredable) a un elemento, esa propiedad se hereda a los elementos que estén dentro. Las reglas se aplican de lo más general a lo más específico.

Por ejemplo, si define que el color del elemento `body` es azul, todos los elementos de la página heredarán ese color, a no ser que alguno de ellos especifique en una regla un nuevo valor para esa propiedad. Más información en:

<http://www.w3schools.com/css/default.asp>
http://www.w3schools.com/css/css3_intro.asp
http://www.w3schools.com/css/css_examples.asp

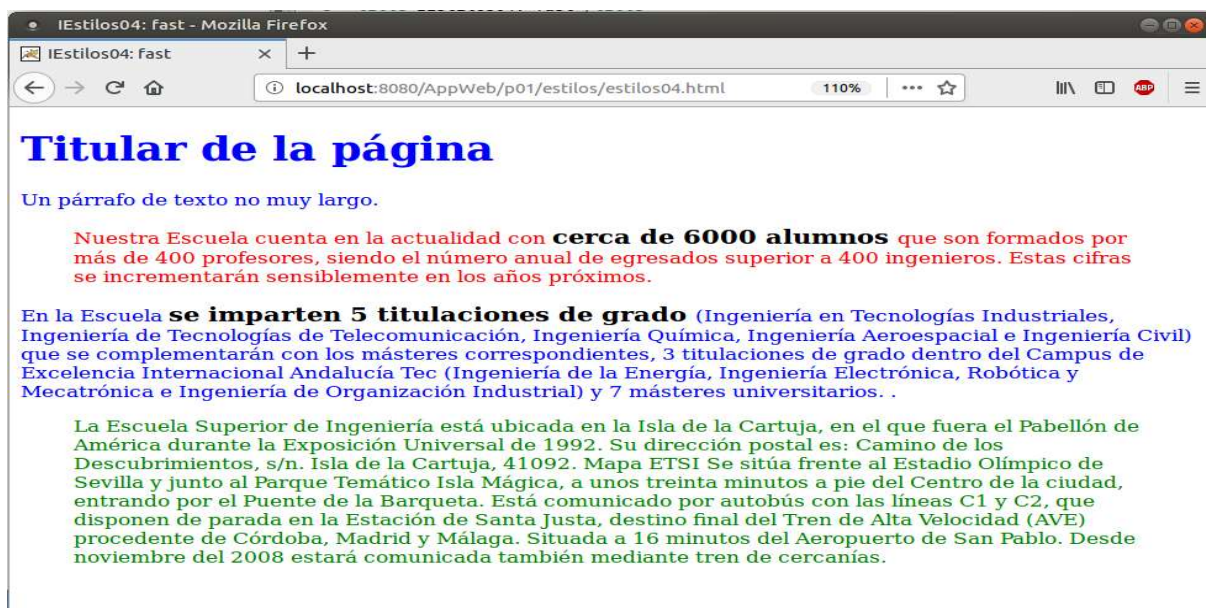
TAREAS

- 1º Edite el fichero `estilos04.html`. Analice su contenido. Observe que el color definido para `body` es azul, y para `blockquote` rojo. Observe que tenemos un elemento `strong`. Analice de qué color aparece cada elemento. Observe que hay un segundo texto delimitado por `blockquote` que no sigue la misma regla que el primero. Analice el porqué de ese comportamiento. Puede consultar las propiedades en:

<http://www.w3schools.com/cssref/default.asp>

- 2º Cree el fichero `estilos04.css`, y realice los pasos similares a `estilos03.css`. Modifique los estilos para que el elemento `strong` se presente con una fuente de tamaño superior, en cursiva y de color verde. Para ello puede utilizar la propiedad `font-size: 1.4em`. Las unidades más usadas son `em` (2em es 2 veces el tamaño de fuente) y `px` (1px es 1/96 de pulgada). Puede consultar las unidades en:

http://www.w3schools.com/cssref/css_units.asp



4.1 Selectores

Ya se han presentado algunos tipos de selectores en teoría. Aunque existen muchas posibilidades, nos vamos a detener en lo más usados, los selectores de tipo o etiqueta, los de clases y combinaciones de los anteriores.

Selectores	Elementos	Ejemplo
*	Todos	*
etiqueta	etiqueta	p
[atributo]	Con atributo	[href]
[atributo op valor]	Con atributo y valor (=), (~=) o (=)	[href="web.html"]
#ID	Con "id=ID"	#AAAA
.clase	Con "class=clase"	.c1
:característica	Pseudo-clase	:visited

4.1.1 Selectores de etiqueta

Son los que se han utilizado hasta ahora en los ejemplos. Seleccionan todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector. Solamente es necesario indicar el nombre de una etiqueta HTML (sin los caracteres < y >) correspondiente a los elementos que se quieren seleccionar.

Si se quiere aplicar los mismos estilos a dos etiquetas diferentes, se pueden encadenar los selectores. CSS permite agrupar todas las reglas individuales en una sola regla con un selector múltiple. Para ello, se incluyen todos los selectores separados por una coma (.). También es posible que distintas reglas se apliquen a la misma etiqueta. En las siguientes reglas, a h1 h2 y h3 se le aplican primero el color y alineación, y posteriormente a h1 se le aplica un tamaño de fuente mayor del que le corresponde por defecto:

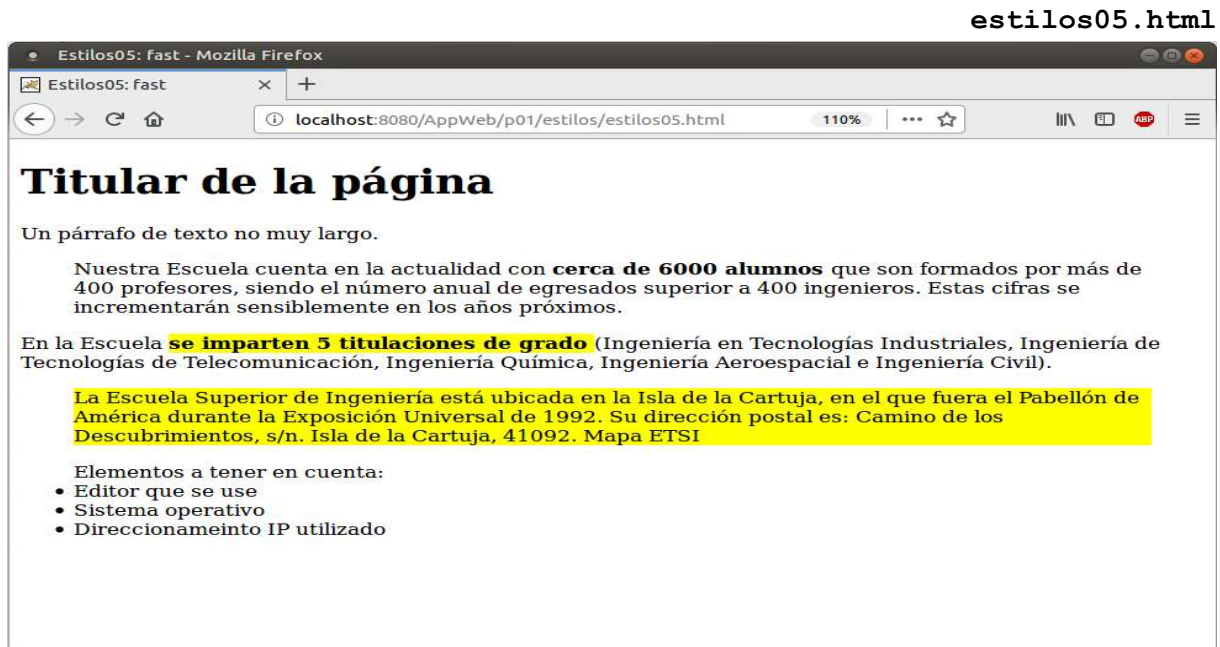
```
h1, h2, h3 { color: blue; align-text: center; }
```

```
h1 { font-size: 2em; }
```


4.1.2 Selectores de clase y de id

Si no desea aplicar el mismo estilo a todos los elementos que tengan la misma etiqueta puede utilizar el selector de clase. También se puede utilizar a elementos de distinta etiqueta.

Para lograrlo será necesario utilizar el atributo `class` en todos aquellos elementos a los que desee aplicar el estilo, e identificarlos con un nombre común. Por ejemplo:



En este caso se ha aplicado el mismo estilo a uno de los dos elementos `blockquote`, y a uno de los elementos `strong`.

estilos con atributo class

```
...
<p>
En la Escuela <strong class="destacado"> se imparten 5
titulaciones de grado </strong> (Ingeniería en Tecnologías
...</p>

<blockquote class="destacado">
La Escuela Superior de Ingeniería está ubicada en la Isla
de la Cartuja, en el que fuera el
...
</blockquote>
...
```

Y habrá que aplicar la regla precediendo el nombre de la clase con un punto:

```
.destacado { background-color: yellow; }
```

El atributo `class` puede tomar varios valores, simplemente separando con un espacio, por ejemplo:

```
class="izquierda resaltado"
```

Puede ver las ventajas de esto en:

`http://www.w3schools.com/tags/att_global_class.asp`

El selector `id` es similar al de clase, pero se aplica al elemento cuyo atributo `id` sea el indicado. El atributo `id` debe ser único dentro de una página html, luego sólo se aplicará a un elemento dentro de una página. El selector usa el valor del atributo `id` precedido del carácter `#`.

```
#principal { background-color: yellow; }
```

TAREAS

- 1º Edite el fichero `estilos05.html`. Analice su contenido. Observe los resultados al presentar la página web en el navegador.
- 2º Cree el fichero `estilos05.css`, y realice los pasos similares a `estilos03.css`. Modifique el fichero para aplicar el estilo de clase destacado al primer elemento de la lista que aparece en la página web.
- 3º Modifique el fichero para aplicar el estilo de `id` de valor `principal` y reglas `{background-color: green; color: red}` al bloque que también tiene aplicado el estilo de clase (sin eliminar el atributo `class`). Observe qué propiedad prevalece en caso de “colisión”.
- 4º Cree otra regla que aplique color marrón (`brown`) a los elementos que tengan el atributo `class` con el valor “resaltado”. Modifique el atributo `class` de un elemento para que se le aplique tanto la regla con el selector `".destacado"` como la regla con el selector `".resaltado"`. Recuerde que si el atributo `class` tiene más de un valor, deben ir separados por un espacio

4.1.3 Combinando los selectores

Se puede ser más específico a la hora de aplicar un selector. Una de las posibilidades es aplicar un estilo de clase sólo a un tipo de elemento. Si se quiere aplicar un cierto estilo sólo a los párrafos que tengan el atributo `class` con el valor `destacado`, la sintaxis consistiría en unir ambos selectores (operador lógico AND):

```
p.destacado { background-color: yellow; }
```

Otra posibilidad es el selector descendiente, que se aplica a los elementos que están incluidos dentro de otro elemento. Se considera que es descendiente cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento. En este caso el selector se forma encadenando las etiquetas del tipo contenedor y del tipo contenido o descendiente:

```
p .destacado { background-color: yellow; }
```

Si compara los dos ejemplos el primero se aplica a cualquier elemento `<p>` que tenga el atributo `class` al valor `destacado`. El segundo se aplica a cualquier elemento que esté incluido en un elemento `<p>` y que tenga el atributo `class` al valor `destacado`.

Operador	Elementos afectados
S1S2	AND
S1 , S2	OR

S1 S2	Descendiente: (hijo, nieto, ...)
S1 > S2	Hijo
S1 + S2	Adyacente
S1 ~ S2	Mismo nivel

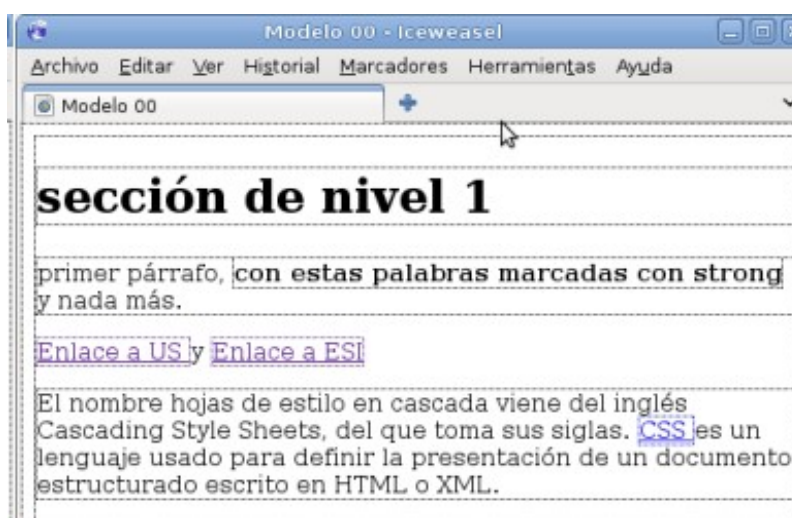
TAREAS

- 1º Edite el fichero `estilos06.html`. Analice su contenido. Observe los resultados al presentar la página web en el navegador. Hay párrafos que se destacan, otros que no. Hay elementos `strong` o enlaces dentro de párrafos que se destacan, otros no.
- 2º Puede consultar más información de las hojas de estilo CSS en la URL <http://www.w3schools.com/css/default.asp>. Recorra alguna de las secciones, y utilice las utilidades “Try it yourself”, que permiten editar código y ver cómo se presentaría en un navegador. Observe que en el apartado “CSS Reference” tiene una lista de todas las propiedades. En esta práctica solo se presentan algunas de ellas

4.2 Modelo de cajas

Ya conoce que en HTML los elementos son de bloque o de línea. Los elementos de bloque, a la hora de posicionarse en la página, “provocan” un salto de línea previo y posterior al mismo. Observe en la figura cómo tras el contenido de la etiqueta `h1` (elemento de bloque), cuyo texto no ocupa todo el ancho de la línea, no hay nada. El párrafo comienza una línea después. Al mismo tiempo el segundo párrafo no ocupa todo el ancho de la segunda línea, en cambio el párrafo siguiente empieza en la línea siguiente.

Frente a esto, observe las palabras destacadas con `strong`, dentro del párrafo, cuyo contenido se inserta en el párrafo. Observe por último los dos enlaces entre los párrafos, uno a continuación del otro.



modelo00.html

```
<!-- omitido inicio fichero -->
<style type="text/css" >
* { border: dotted ; }
</style>
</head>
<body>
<h1> sección de nivel 1</h1>
<p>primer párrafo, <strong> con estas palabras marcadas con
strong</strong>
y nada más.</p>

<a href="http://www.us.es"> Enlace a US </a>
Y
<a href="http://www.esi.us.es"> Enlace a ESI </a>

<p class="margen">El nombre hojas de estilo en cascada viene
del inglés Cascading Style Sheets,
del que toma sus siglas.
<a href="http://www.w3schools.com/css/default.asp">CSS </a>
es un lenguaje usado para definir la presentación de un
documento
estructurado escrito en HTML o XML.</p>

</body>
</html>
```

TAREAS

- 1º Analice el contenido del fichero modelo00.html, y observe la presentación que hace el navegador.
- 2º Modifique el marcado del primer párrafo, justo debajo de <h1>, sustituyendo la etiqueta <p> por la etiqueta <code> (no olvide modificar la etiqueta de cierre). ¿Qué ha cambiado en la presentación respecto a la distribución de espacios? ¿Dónde están ahora los enlaces?

Recuerde de teoría el modelo de cajas CSS para comprender cómo se distribuyen los elementos en la página web, y como se aplican las propiedades:

- **content:** se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- **padding:** espacio libre opcional existente entre el contenido y el borde. Por defecto vale cero. Por detrás del contenido y el espacio de relleno se puede mostrar una imagen, y un color de fondo:
 - **background-image:** imagen que se muestra por detrás del contenido y el espacio de relleno.
 - **background-color:** color que se muestra por detrás del contenido, del espacio de relleno y de la imagen de fondo. El valor por defecto es transparente.
- **border:** línea que encierra completamente el contenido y su relleno. Por defecto no hay borde.

- **margin:** separación opcional existente entre la caja y el resto de cajas adyacentes. Por defecto vale cero.

Las propiedades `border`, `margin` y `padding`, se pueden aplicar de forma independiente a cada uno de los lados de la caja: `top`, `bottom`, `left` y `right`. Y en el caso de `border` podemos aplicar anchura, color o estilo del borde. Observe dos ejemplos de estilos:

```
h1 { border-top-width: 2px; border-top-style: double; }
```

También se pueden aplicar de forma agrupada, por tipo de propiedad, etc.

```
p { border: 1px solid red; }
```

Para conocer las distintas opciones puede consultar alguna de las referencias sobre CSS, por ejemplo la guía rápida de W3schools:

```
http://www.w3schools.com/cssref/default.asp
```

Para comprender la diferencia entre relleno y margen observe el ejemplo del fichero `modelo01.html`

En un párrafo se ha aplicado margen, en el otro relleno:

```
.margen { border: dotted; margin: 3em; background-color: yellow; }  
.relleno { border: dotted; padding: 3em; background-color: yellow; }
```

El párrafo 1 tiene margen en todos los lados, y por lo tanto el contenido queda separado de los párrafos anterior y posterior. Además observe cómo el borde de la caja está justo alrededor del contenido, ya que el `padding` es cero en ese caso. Recuerde y observe que el borde no es el límite de la caja, los límites de la caja llegan más allá si hay margen, que es este caso.

En el segundo caso, el párrafo 2, al no tener definido valor para el margen se supone cero, y el borde coincide con los límites de la caja. Por otra parte, al haber aplicado `padding`, el contenido está separado del borde de la caja.

Puede consultar más ejemplos en:

```
http://www.w3schools.com/css/css\_boxmodel.asp
```

4.3 Posicionamiento básico

CSS permite modificar la posición en la que se muestra la caja que conforma un elemento. La propiedad que lo permite es `position`. Sus valores posibles son: `static`, `relative`, `fixed`, `absolute`.

Puede mirar los ejemplos y consultar su funcionamiento en:

```
http://www.w3schools.com/css/css\_positioning.asp
```

4.4 Posicionamiento flotante, maquetación, HTML5 y entornos de diseño web

Es un modo especial de posicionamiento en que las cajas se desplazan horizontalmente a la derecha o a la izquierda respecto de su posición original. Su comportamiento se controla con la propiedad `float`,

que puede tomar los valores `left` o `right`. Cuando se aplica, el resto de elementos que le siguen ocupan el espacio dejado por la caja flotante (se dice que la caja flotante no forma parte del flujo normal de la página). Destacar que los elementos sólo pueden flotar horizontalmente, no hacía arriba o abajo. Los elementos anteriores no se ven afectados por este posicionamiento. Puede consultarlo (y también la propiedad `clear`) en:

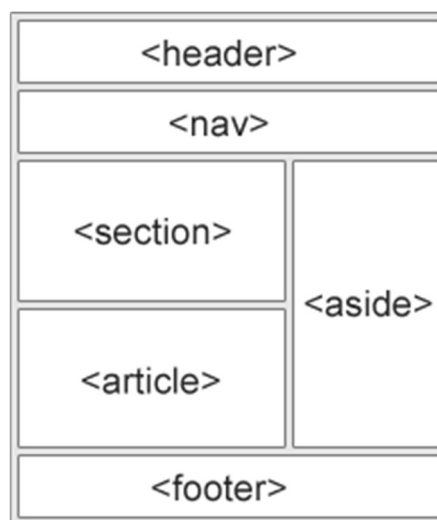
http://www.w3schools.com/css/css_float.asp

Uno de los usos más frecuentes de `float` es con imágenes o para organizar la estructura (layout) de la página.

Para hacer la maquetación de la página web, es decir, decidir dónde se presentan cada una de sus partes, en HTML4.01 se solía usar la etiqueta genérica `<div>` (recuerde que `<div>` es de bloque y `` es de línea) que nos permite marcar un bloque e identificarlo con un atributo (`class` o `id`) para posteriormente, mediante hojas de estilo, posicionarlo y asignarle propiedades.

Es frecuente que todas las páginas, en el cuerpo o `body` tengan una estructura similar: una parte superior de títulos, logos, etc., un zona de contenidos, con un menú a la derecha (o a la izquierda), y una zona central de contenidos, que a su vez a puede estar en una o dos columnas.

Con HTML5 se usan nuevas etiquetas para definir diferentes partes de una página web:



Puede consultar la semántica de estas etiquetas en:

http://www.w3schools.com/html/html5_semantic_elements.asp

Puede ver la equivalencia en maquetación entre HTML4.01 y HTML5 en:

http://www.w3schools.com/html/html5_migration.asp

Actualmente en la maquetación se intenta hacer un diseño web que se adapte a distintos tipos de dispositivos, lo que se conoce como Responsive Web Design (RWD), usando HTML y CSS. Puede encontrar información en:

http://www.w3schools.com/css/css_rwd_intro.asp

Para facilitar esta tarea, hay múltiples entornos (Frameworks):

`http://www.w3schools.com/css/css_rwd_frameworks.asp`

En concreto, uno de los más simples es W3.CSS:

`http://www.w3schools.com/w3css/default.asp`

4.5 Validación de CSS

Existe una herramienta online para validar su código CSS, y comprobar si es sintácticamente correcto:

`http://www.css-validator.org/`

Por otra parte, aunque su hoja de estilos fuese sintácticamente correcta, es posible que no funcione como espera. Para buscar el problema puede usar herramientas como las Developer Tools que verá a continuación y comprobar qué reglas CSS se están aplicando realmente sobre un determinado elemento.

TAREAS

- 1º Acceda al validador online del W3C, e intente validar el fichero `estilos01.css` del directorio `css`:

`http://www.css-validator.org/`


- 2º Intente ahora validar los ficheros `estilos02_error.css` y `estilos03_error.css` que se encuentran en el mismo directorio, y analice los errores que se muestran. Revise y corrija los ficheros para que superen el proceso de validación.

5 Herramientas de desarrollo (Developer Tools)

Los navegadores web suelen incluir una serie de herramientas destinadas a facilitar el trabajo de detección de fallos, errores en la aplicación de estilos y errores en el código JavaScript. Además existen complementos, extensiones o plugins que incorporan funcionalidades similares o mejoradas de esas herramientas. El objetivo de este apartado es introducirle en el uso de una de ellas, las “developer tools” que incorpora el navegador Mozilla Firefox.

Para acceder puede usar el menú *> Desarrollador > Alternar Herramientas (Ctrl+Mayús.+I)*, o sobre algún elemento de la página HTML pulsar el botón derecho y seleccionar “*Inspeccionar elemento*”, o también puede pulsar F12.

Observará cómo en la parte inferior aparece la Caja de Herramientas (“ToolBox”). Si ha seleccionado el Inspector, entonces ya estará seleccionado el elemento. En la parte inferior izquierda aparece el panel de código HTML, y a la derecha el panel CSS con los estilos aplicados

Si hace clic sobre el icono “seleccionar elemento con el ratón” , a la izquierda del panel HTML, a medida que mueve el ratón por la página, el elemento bajo el puntero del ratón se resalta con un borde punteado y una anotación muestra su etiqueta HTML. Al mismo tiempo, se muestra la definición de ese elemento en el panel HTML.

Si hace clic sobre un elemento de la página HTML, ese elemento se “fija” en el panel HTML. Al mismo tiempo, si sobre ese elemento HTML se han aplicado estilos CSS, se muestran en el panel CSS, a la derecha. El panel CSS tiene varias vistas, de las cuales nos centraremos en la vista Reglas.

TAREAS

- 1º Abra el fichero `modelo03.html` en el navegador, y active las “Developer Tools”. Pruebe a ir recorriendo los distintos elementos, y observe cómo se muestra la etiqueta HTML (y su `class` si la tiene definida, como por ejemplo los párrafos con clase `margen` o `relleno`).
- 2º Fije el elemento `p` con clase `margen`, y una vez fijado observe el panel CSS donde se muestran las reglas CSS que se están aplicando sobre dicho elemento.

Sobre el panel CSS es posible modificar las reglas CSS que se aplican para ver el resultado inmediato sobre la página HTML. Si hace clic a la izquierda de una propiedad aparecerá un cuadro que permite activar o desactivar esa propiedad. Si la desactiva, la propiedad aparecerá tachada, y en ese mismo momento deja de aplicarse en el navegador.



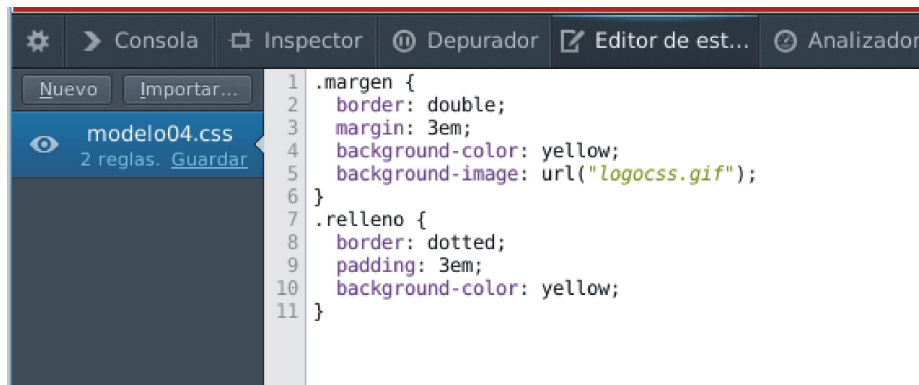
También es posible, si hace clic sobre los valores de la propiedad, modificarlos y darle nuevos valores. Incluso haciendo clic sobre la regla CSS (sobre la llave de cierre) es posible añadir nuevas propiedades. Si se introduce una regla CSS incorrecta el panel CSS la muestra tachada y con un símbolo de aviso para llamar nuestra atención.



Si las reglas CSS están definidas en un fichero externo, al lado de cada regla se muestra el fichero y la línea donde está especificada.



Tenga en cuenta que todas las modificaciones que haga sobre el panel CSS se “pierden” al recargar la página web. Para ocultar el editor basta hacer clic sobre el botón del “Inspector”.



TAREAS

- 1º Continúe con el fichero `modelo03.html`, y desactive la propiedad que fija la imagen de fondo. Observe el resultado, aparece el fondo amarillo. Justifique la razón.
- 2º Añada en el panel CSS la propiedad `background-repeat` y asígnele el valor `no-repeat`; Observe que a medida que escribe el panel le “ofrece” valores posibles para la propiedad y para sus valores. Observe el resultado.
- 3º Añada en el panel CSS la propiedad `padding` y dele un valor suficiente para que la imagen no se solape con el texto. Observe el resultado. Haga uso de F12 para ver el modelo de cajas.
- 4º Modifique en el panel CSS la propiedad `border` a los valores `thick blue double`. Observe el resultado.

A partir de este momento, siempre que tenga algún problema con la aplicación de estilos, podrá utilizar tanto el validador CSS como estas herramientas. Puede encontrar más información en la web de Mozilla:

<https://developer.mozilla.org/en-US/docs/Tools>

6 HTTP

En este apartado se van a ver detalles del protocolo HTTP.

Los mensajes de petición más usados son los que tienen el método GET o el método POST. Puede consultar las diferencias más importantes en:

https://www.w3schools.com/tags/ref_httpmethods.asp

Para ver las cabeceras de los mensajes del protocolo se puede utilizar Wireshark o las Developer Tools.

Con las Developer Tools es posible además editar un mensaje ya enviado y volverlo a enviar. También es útil cuando se usa HTTPS, ya que Wireshark no puede descifrar los mensajes encriptados por TLS.

6.1 Cabecera Location:

Location es una cabecera de respuesta que indica la URL a la que redirigirse en los mensajes de redirección. Puede consultar su funcionamiento en la RFC 7231 o en:

<https://developer.mozilla.org/es/docs/Web/HTTP/Headers/Location>

TAREAS

- 1º Abra el navegador Firefox y pulse F12 para acceder a las Developer Tools. Seleccione la pestaña Red (justo a la derecha de Memoria). Acceda a la URL `http://ev.us.es` (fíjese que se intenta usar `http` y no `https`) y observe los mensajes enviados (si desaparecen las Developer Tools vuelva a pulsar F12 y a acceder a la URL indicada).
- 2º Haciendo click sobre el primer mensaje debe aparecer a la derecha el contenido de éste y su respuesta. En la pestaña Cabeceras busque la cabecera Location. ¿Qué contiene? ¿De qué tipo (código numérico) es la respuesta? Mire también las pestañas Cookies, Parámetros y Respuesta. ¿Qué contiene la Respuesta?
- 3º Haga click en el segundo mensaje enviado y averigüe cuál es la URL solicitada. ¿Qué relación tiene con la cabecera Location del mensaje anterior? ¿De qué tipo es la respuesta a este segundo mensaje? ¿Contiene la cabecera Location? ¿Qué contiene ésta? ¿Qué contiene la Respuesta?
- 4º Haga click en el tercer mensaje enviado y repita los pasos del apartado anterior.

6.2 Cabecera If- Modified-Since

If-Modified-Since es una cabecera de petición que permite que el servidor no envíe un recurso si no se ha modificado desde un determinado instante. Puede consultar su funcionamiento en la RFC 7232 o en:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-Modified-Since>

TAREAS

- 1º Usando las Developer Tools como en el apartado anterior, acceda a la URL `www.esi.us.es`
- 2º ¿Se produce redirección? Busque la cabecera Location. ¿Qué contiene? ¿De qué tipo (código numérico) es la respuesta?
- 3º Entre todos los mensajes enviados haga click en el mensaje GET que solicita el archivo `pause.png` (si no aparece pruebe a recargar la página). En la pestaña Cabeceras busque la cabecera Content-Length en la respuesta y compruebe que su valor es “1378”. En la parte derecha pulse sobre “Editar y volver a enviar” y borre la línea que contiene la cabecera If-None-Match. A continuación, compruebe que hay una línea con la cabecera If-Modified-Since, y pulse “Enviar “.
- 4º Analice el mensaje enviado y su respuesta. ¿De qué tipo es la respuesta? ¿Contiene la cabecera Content-Length?
- 5º Edite el mensaje anterior (pulse sobre “Editar y volver a enviar”), borre la línea que contiene la cabecera If-Modified-Since, y pulse “Enviar “.
- 6º Analice el mensaje enviado y su respuesta. ¿De qué tipo es la respuesta? ¿Contiene la cabecera Content-Length? ¿Coincide su valor con alguno de los mensajes anteriores?
- 7º Repita todos los pasos anteriores, pero además capturando el tráfico con Wireshark (se pueden ver los mensajes HTTP porque no se está usando HTTPS). Compruebe que cuando el mensaje de respuesta es del tipo “304 Not Modified”, efectivamente el cuerpo del mensaje está vacío.

7 XML

El lenguaje extensible de marcas (XML) es un subconjunto de SGML. Su objetivo es permitir que SGML genérico pueda ser servido, recibido y procesado en la web en la misma manera que hoy es posible con HTML. Si desea ampliar información sobre SGML, puede ampliar información empezando en:

<https://www.w3.org/TR/WD-html40-970708/intro/sgmltut.html>

XML ha sido diseñado de tal manera que sea fácil de implementar y buscando interoperabilidad tanto con SGML como con HTML. Puede consultar su especificación en:

<https://www.w3.org/standards/techs/xml>

Entre los objetivos de diseño para XML está:

- XML debe ser compatible con SGML.
- El diseño de XML debe ser formal y conciso.
- La brevedad en la marcación es de mínima importancia.

7.1 Reglas de sintaxis XML

Se puede, de una forma abreviada, considerar que un documento XML es sintácticamente correcto (bien formado o “well-formed”) si contiene un único elemento raíz, todo elemento tiene etiqueta de cierre, los elementos están correctamente anidados y los atributos están delimitados por comillas dobles (o simples).

En XML el nombre de un elemento (y de un atributo) debe seguir estas reglas: el primer carácter puede ser una letra del alfabeto inglés en mayúsculas o minúsculas y el guion bajo o subrayado. El resto de caracteres pueden ser los mismos que el primer carácter, pero además también se puede usar el guion, el punto y los números.

En XML se distinguen las mayúsculas y minúsculas, es un lenguaje sensible a mayúsculas y minúsculas. Los comentarios en XML se encierran dentro de `<!-- Comentario -->`.

Por último, en XML existen las secciones `CDATA`, que permiten que el analizador XML ignore ciertas secciones del documento. En el contenido de la sección `CDATA` se pueden emplear los caracteres especiales como `&`, `<` y `>` sin peligro de que sean interpretados por el analizador XML.

Si lo desea, puede ver ejemplos de violación de estas reglas en la documentación de W3Schools:

`http://www.w3schools.com/xml/xml_syntax.asp`

TAREAS

- 1º Acceda a la página web `http://www.w3schools.com/xml/xml_syntax.asp` y analice la información presentada.
- 2º Cree un fichero xml, e incluya información con sintaxis incorrecta en el mismo, siguiendo las indicaciones de la página web indicada. Compruebe que al intentar mostrar el fichero en un navegador genera un error en pantalla.

7.2 Un fichero XML paso a paso

Un fichero xml tendrá una estructura similar a esta:

tablon01_correcto.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- DTD asociada -->
<!DOCTYPE tablon SYSTEM "tablon01.dtd">
<!-- Hoja de estilos asociada -->
<?xml-stylesheet type="text/css" href="tablon01.css"?>
<tablon>
  <anuncio codigo="A000" vendido="no">
    <fecha> 22/09/2014</fecha>
    <asunto> Vendo bicicleta </asunto>
    <texto> 2 años, cambios en buen estado</texto>
    <precio moneda_pago="eur"> 500 euros</precio>
    <contacto visible="si"> 95 456 456 </contacto>
  </anuncio>
  <anuncio codigo="A001" vendido="no">
    <fecha> 22/09/2011 </fecha>
    <asunto> Vendo moto </asunto>
    <texto> buena, bonita, barata</texto>
    <precio moneda_pago="eur"> 500 euros</precio>
    <contacto visible="si"> 666 123 123 </contacto>
  </anuncio>
  <anuncio codigo="A002" vendido="si">
    <fecha> 24/09/2011 </fecha>
    <asunto> Vendo bici</asunto>
    <texto> muy bonita, con candado</texto>
    <precio> 90 euros</precio>
    <contacto visible="no"> a@ese.com </contacto>
  </anuncio>
</tablon>

```

En la primera línea se encuentra la declaración de documento xml, con la codificación asociada al fichero, en este caso UTF-8. Esta línea se llama prólogo, y es opcional, pero si existe debe ser la primera.

```
<?xml version="1.0" encoding="UTF-8"?>
```

Tras ella hay dos instrucciones de proceso opcionales. En la primera se enlaza el fichero a un fichero externo, tablon01.dtd, que servirá para validar la estructura, elementos y atributos del fichero xml.

```
<!DOCTYPE tablon SYSTEM "tablon01.dtd">
```

En la segunda se enlaza el fichero xml a una hoja de estilos que se usará para presentar los datos:

```
<?xml-stylesheet type="text/css" href="tablon01.css"?>
```

A partir de ese momento comienzan los datos, con el elemento raíz que, recuerde, debe ser único. Observe que anuncio es un elemento que no tiene texto asociado, es un contenedor de otros elementos: fecha, asunto, texto, precio y contacto. Algunos de ellos además de texto tienen atributos con sus correspondientes valores, entrecomillados para no provocar un error sintáctico.

tablon01_correcto.xml

```
<tablon>
  <anuncio codigo="A000" vendido="no">
    <fecha> 22/09/2014</fecha>
    <asunto> Vendo bicicleta </asunto>
    <texto> 2 años, cambios en buen estado</texto>
    <precio moneda_pago="eur"> 500 euros</precio>
    <contacto visible="si"> 95 456 456 </contacto>
  </anuncio>
</tablon>
```

Recuerde que XML es mucho más estricto que HTML en su sintaxis. Observe qué ocurriría si se intenta mostrar un fichero XML en el que no se ha respetado la delimitación de los atributos, o se ha equivocado al cerrar la etiqueta de un elemento y la ha puesto en mayúsculas, etc.

tablon01_mal_formado.xml

```
<tablon>
  <anuncio codigo="A000" vendido="no">
    <fecha> 22/09/2014</fecha>
    <asunto> Vendo bicicleta </asunto>
    <texto> 2 años, cambios en buen estado</texto>
    <precio moneda_pago="eur"> 500 euros</precio>
    <contacto visible="si"> 95 456 456 </contacto>
  </Anuncio>
  <anuncio codigo=A001 vendido="no">
    <fecha> 22/09/2011 </fecha>
    <asunto> Vendo moto
    <texto> buena, bonita, barata</texto>
    <precio moneda_pago="eur"> 500 euros</precio>
    <contacto visible="si"> 666 123 123 </contacto>
  </anuncio>
```

Ante este fichero XML, el resultado en el navegador es:



7.3 Bien formado y válido

Por lo tanto, interesa comprobar el documento. Recuerde que en XML puede comprobar si el fichero xml es sintácticamente correcto (“well-formed”), y si es “válido”, es decir, si se corresponde con una estructura, indicada normalmente en otro fichero (en un DTD o en un *schema*, tecnologías que quedan fuera del alcance de esta asignatura).

Podemos comprobar si el fichero XML es sintácticamente correcto intentando mostrarlo en el navegador. Pero el navegador no comprueba si el fichero es “válido”. Para ello podemos usar un IDE que lo haga, por ejemplo Eclipse, emacs, xmlCopyEditor, etc. O usar alguna herramienta en línea de comandos, como `xmllint`. Debe consultar el manual, pero le adelantamos su uso básico.

Para comprobar si el fichero está bien formado:

```
xmllint fichero.xml
```

Si el fichero XML indica la DTD que debe verificar: mediante una sentencia DOCTYPE (`<!DOCTYPE fichero SYSTEM "fichero.dtd">`)

```
xmllint --valid fichero.xml
```

Y si no incluye la DTD la podemos especificar en la comprobación:

```
xmllint --dtdvalid fichero.dtd fichero.xml
```

El parámetro `--noout` hace que el comando no presente a la salida el árbol del documento.

TAREAS

- 1º Utilice los ficheros XML de los apartados anteriores, el fichero `ventas.xml` y el fichero `tablon01_mal_formado.xml` y compruebe si están bien formados usando la herramienta `xmllint`.
- 2º Estudie otras opciones de la herramienta `xmllint` (`man xmllint`), por ejemplo aquellas que permite que la salida del proceso sea el árbol del documento formateado correctamente, y como almacenar dicha salida en un fichero. Utilice para ello el fichero `ventas.xml`, cuyo código no está correctamente tabulado, y tiene errores sintácticos que debe corregir.

Los editores suelen incluir alguna opción para comprobar la sintaxis durante la edición. Por ejemplo, emacs al abrir ese fichero muestra en la parte inferior que el fichero no es correcto, y resalta en rojo los errores:

```

emacs@debian
File Edit Options Buffers Tools XML Help

curso.xml tablon01_err.xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- DTD asociada <!DOCTYPE tablon SYSTEM "tablon01.dtd" --> -->
<!-- Hoja de estilos asociada <?xml-stylesheet type="text/css" href=
s'
>
<tablon>
  <Anuncio codigo="A000" vendido="no">
    <fecha> 22/09/2014</fecha>
    <asunto>Vendo bicicleta azul</asunto>
    <texto> 2 años, cambios en buen estado</texto>
    <precio moneda_pago="eur"> 500 euros</precio>
    <contacto visible="si"> 95 456 456 </contacto>
    <fecha_limite>21707/2013</fecha_limite>
  </anuncio>
  <anuncio codigo="A000" vendido="no">
    <fecha> 22/09/2011 </fecha>
    <texto> buena, bonita, barata</texto>
    <asunto> Vendo moto </asunto>
    <precio moneda_pago="eur"> 500 euros</precio:
    <contacto visible="depende"> 666 123 123 </
  </anuncio>
  <anuncio codigo="A0022" vendido="si">
    <fecha> 24/09/2011 </fecha>
    <asunto>
    <texto> muy bonita, con candado</texto>
    <precio> 90 euros</precio>
    <contacto> a@ese.com </contacto>
  </anuncio>
</tablon>

-U:--- tablon01_err.xml Top (24,0) (nXML Invalid)-----

```

TAREAS

- 1º Analice el fichero `tablon01_mal_formado.xml` y observe la presentación que hace el navegador: hay errores y no se muestra el documento.
- 2º Compruebe con `xmllint` si el mismo fichero está bien formado.
- 3º Edite el fichero y corrija los errores: raíz duplicada, etiquetas no cerradas, etiquetas mal anidadas, atributos no entrecomillados o no correspondencia mayúsculas y minúsculas en los nombres de las etiquetas. Corregidos los errores, compruebe que el fichero está bien formado con `xmllint` y en el navegador.
- 4º El navegador muestra la estructura del documento, con las etiquetas. Además, colorea la sintaxis y permite expandir o colapsar los elementos usando los símbolos “+” y “-” a la izquierda de los elementos. *Nota: la presentación en el navegador puede depender del navegador, versión, etc.*

Aunque no es parte de los contenidos evaluables el cómo se define una DTD, sí debe conocer cómo validar un documento xml si nos proporcionan la DTD.

TAREAS

- 1º Analice el fichero `tablon01.dtd` para tener una impresión de que aspecto tiene una DTD.
- 2º Compruebe con `xmllint` que el fichero `tablon01_correcto.xml` está bien formado y además es válido respecto a la DTD `tablon01.dtd`.
- 3º Compruebe que el fichero `tablon02_mal_formado.xml` es sintácticamente incorrecto. Corrija los errores y compruebe que es válido.
- 4º Compruebe que el fichero `tablon01_no_valido.xml` es sintácticamente correcto, pero no es válido. Corrija los errores: en el primer anuncio fecha debe aparecer antes de asunto, en el segundo anuncio falta el atributo código y en el tercero el valor visible del elemento `mail` sólo puede tener los valores “si” o “no”. Compruebe tras corregirlos que es válido.

A la hora de presentar un documento XML se suelen usar hojas de estilo en cascada (CSS), o el lenguaje de transformación XSL (*Extensible Stylesheet Language*). Esta asignatura no aborda la presentación de ficheros XML usando estas tecnologías.

Como tarea opcional no evaluable, puede comprobar el uso de XML y CSS en la siguiente tarea:

TAREAS

Tarea opcional no evaluable

- 1º Acceda al fichero `tablon01_correcto_con_css.xml`. Muestre el fichero en el navegador y compruebe que la presentación ha cambiado en este caso debido a la acción del fichero `tablon01.css`.
- 2º Analice el fichero `tablon01_correcto_con_css.xml`. Observe cómo se hace referencia al fichero `tablon01.css`. Edite ese fichero y elimine los comentarios (`/* ...*/`). Observe cómo afecta a la presentación.

7.4 Espacios de nombres

En XML, los nombres de elementos se definen por el programador. Esto a menudo resulta en un conflicto cuando se trata de mezclar documentos XML a partir de diferentes aplicaciones XML. Observe un ejemplo extraído de w3schools que almacena información de una mesa (table) y una tabla HTML, que también usa el elemento `table`:

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Si se suman estos fragmentos XML, habría un conflicto de nombres. Ambos contienen un elemento `<table>`, pero los elementos tienen diferente contenido y significado. Un usuario o una aplicación XML no saben cómo manejar estas diferencias.

Para evitarlo, asignaremos a cada tipo de información un espacio de nombres (namespace), y lo identificaremos mediante un prefijo. La forma de hacerlo se muestra en el siguiente fichero `espacios.xml`, que ya engloba la información de las dos fuentes.

Para definir el espacio de nombre se usa el atributo `xmlns`, y se le asocia un prefijo y un identificador único, un URI.

```
xmlns:prefijo="URI"
```

En nuestro ejemplo usamos los prefijos “h” y “f”:

```
xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3schools.com/furniture"
```

espacios.xml

espacios.xml

```
<root xmlns:h="http://www.w3.org/TR/html4/"
      xmlns:f="http://www.w3schools.com/furniture">

  <h:table>
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>

  <f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>

</root>
```

Aunque el espacio de nombres se puede definir sobre el elemento implicado directamente, lo habitual es hacerlo en el elemento raíz.

Por otra parte, si hay varios espacios de nombres, uno de ellos puede ser considerado el espacio de nombres por defecto, o principal. En ese caso la notación se puede simplificar, y obviar el prefijo en la definición y en los elementos del espacio de nombre por defecto. De esta forma, todos los elementos

que no tengan prefijo se suponen del espacio de nombres por defecto. En ese caso, el ejemplo anterior queda:

espacios01.xml

```
<root xmlns="http://www.w3.org/TR/html4/"
      xmlns:f="http://www.w3schools.com/furniture">
  <table>
    <tr>
      <td>Apples</td>
      <td>Bananas</td>
    </tr>
  </table>

  <f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</root>
```

Nota: El URI que identifica el espacio de nombres no es utilizado por el analizador para buscar información. El objetivo es dar al espacio de nombres un nombre único. Sin embargo, a menudo se utiliza el espacio de nombres como un puntero a una página web que contiene información del espacio de nombres.

Puede comprobar el uso de espacios de nombres en la siguiente tarea:

TAREAS

- 1º Acceda a la página web que se indica a continuación. Encontrará un ejemplo de uso conjunto de dos espacios de nombres, XHTML y SVG:

https://developer.mozilla.org/en-US/docs/Web/SVG/Namespace_Crash_Course/Example

- 2º Busque en el fichero los dos espacios de nombres definidos, uno con prefijo svg y otro usa el espacio de nombres por defecto.
- 3º Busque en el código qué elementos pertenecen al espacio de nombres con prefijo svg.

APAGUE EL EQUIPO

Cuando finalice la práctica no olvide apagar el equipo.

8 Anexo (no evaluable): Validación XML con XML Schema (XSD)

Aunque ya se ha visto que se puede validar un documento XML con DTD, hay un mecanismo mucho más potente llamado esquema o “XML Schema”, que no es más que un documento XML que describe la estructura de otro documento XML.

Observe el aspecto de un esquema XML:

tablon_simp.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="tablon">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="anuncio">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="fecha" type="xs:date" />
              <xs:element name="asunto" type="xs:string" />
              <xs:element name="texto" type="xs:string" />
              <xs:element name="precio" type="xs:decimal" />
              <xs:element name="contacto" type="xs:string" />
            </xs:sequence>
            <xs:anyAttribute/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

No es el objetivo que conozca en profundidad cómo construir un esquema pero sí que identifique sus partes principales, estructura, etc.

El fichero anterior es un fichero XML (aunque tenga la extensión .xsd para indicar que es un esquema) que utiliza un espacio de nombres normalizado por el W3C para los esquemas.

<http://www.w3.org/2001/XMLSchema>

Desde el punto de vista de los esquemas, los elementos de un documento XML pueden ser complejos (pueden contener otros elementos) o simples (no pueden).

Observe en el ejemplo previo cómo el primer elemento (element) que aparece tiene de nombre `tablon` (el atributo `name` con el valor “`tablon`”), es de tipo complejo (complexType), y está compuesto de una secuencia (sequence) de un número indeterminado (`maxOccurs="unbounded"`) de elementos de nombre `anuncio`. A su vez `anuncio` es un elemento complejo que es una secuencia de elementos simples de nombre `fecha`, `asunto`, `texto`, `precio` y `contacto`, cada uno de ellos con un tipo de datos: `xs:date`, `xs:decimal` y `xs:string`. Una de las grandes ventajas de XSD es que soporta tipos de datos, y que además es muy flexible y sencillo extenderlos.

Para indicar cómo se especifica en el fichero XML con qué esquema hay que hacer la validación, una forma simple de hacerlo, es añadiendo el atributo `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` en el elemento raíz del documento. Dado que ese atributo está relacionado con los esquemas y pertenece a otro espacio de

nombres, hay que indicar el espacio de nombres normalizado por el W3C para las instancias de los esquemas:

```
http://www.w3.org/2001/XMLSchema-instance
```

y definir una etiqueta para usarla como prefijo, en este caso `xsi`. A continuación se muestra un ejemplo donde se indica que el fichero de esquema asociado es `tablon_simp.xsd`, y deberá estar en el mismo directorio que el fichero a validar, ya que no se especifica ningún camino.

tablon_simp.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<tablon
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="tablon_simp.xsd">

  <anuncio>
    <fecha>2014-09-22</fecha>
    <asunto> Vendo bicicleta </asunto>
    <texto> 2 años, cambios en buen estado</texto>
    <precio>500</precio>
    <contacto> 95 456 456 </contacto>
  </anuncio>

  <anuncio>
    <fecha>2011-09-22</fecha>
    <asunto> Vendo moto </asunto>
    <texto> buena, bonita, barata</texto>
    <precio> 1000</precio>
    <contacto> 666 123 123 </contacto>
  </anuncio>

  <anuncio>
    <fecha>2011-09-24</fecha>
    <asunto> Vendo bici</asunto>
    <texto> muy bonita, con candado</texto>
    <precio> 90 </precio>
    <contacto> a@ese.com </contacto>
  </anuncio>

</tablon>
```

TAREAS

- 1º Descargue los ficheros `tablon_simp.xml` y `tablon_simp.xsd`
- 2º Use la herramienta `xmllint` para validar el fichero xml, y compruebe que es válido respecto al esquema especificado (si tiene dudas consulte man `xmllint`)


```
xmllint --noout --schema tablon_simp.xsd tablon_simp.xml
```
- 3º Edite el fichero `tablon_simp.xml` y cambie el valor de alguna fecha, por ejemplo sustituya `2011-09-24` por `24/09/2011`. Vuelva a validar el fichero ¿es ahora válido?
- 4º Edite el fichero `tablon_simp.xml` y añada un atributo a algún anuncio, por ejemplo modifique la etiqueta de apertura de un anuncio para que quede `<anuncio vendido="no">`. Vuelva a validar el fichero ¿es ahora válido?

El esquema anterior es muy simple, pero es difícil de leer y mantener cuando los documentos son complejos. Es más aconsejable definir clases o tipos, dándoles un nombre con el atributo `name` a los elementos simples y a los elementos complejos, y referenciándolos con el atributo `type` del elemento (es decir, el valor de `name` del referenciado es el valor de `type` del referenciante). A continuación se muestra un esquema que sigue este método y que es totalmente equivalente al anterior.

tablon_simp_tipos.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:complexType name="Tanuncio">
    <xs:sequence>
      <xs:element name="fecha" type="xs:date" />
      <xs:element name="asunto" type="xs:string" />
      <xs:element name="texto" type="xs:string" />
      <xs:element name="precio" type="xs:decimal" />
      <xs:element name="contacto" type="xs:string" />
    </xs:sequence>
    <xs:anyAttribute/>
  </xs:complexType>

  <xs:complexType name="Ttablon">
    <xs:sequence>
      <xs:element name="anuncio" type="Tanuncio"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="tablon" type="Ttablon" />

</xs:schema>
```

Observe en el ejemplo previo cómo en la última línea del esquema se define un elemento (`xs:element`) de nombre (`name`) `tablon`, cuyo tipo (`type`) es `Ttablon`. Previamente se define el tipo `Ttablon` como una secuencia de elementos de nombre `anuncio` y tipo `Tanuncio`. Observe cómo se especifica el número de ocurrencias del elemento `anuncio`, indicando que están ilimitados (`maxOccurs="unbounded"`). Si no se especifica el mínimo se supone que es uno (`minOccurs="1"`). Al inicio del esquema se define el tipo `Tanuncio`, como una secuencia de 4 elementos, cada uno de ellos con un tipo de datos: `xs:date`, `xs:decimal` y `xs:string`. Este esquema especifica que el correspondiente documento XML debe tener un elemento de nombre `tablon`, y especifica cómo debe ser ese elemento (que es totalmente equivalente al esquema anterior). El orden en el que aparecen las definiciones de `Ttablon` o `Tanuncio` no es relevante.

TAREAS

- 1º Descargue el fichero `tablon_simp_tipos.xsd`
- 2º Use la herramienta `xmllint` para validar el fichero xml con este nuevo esquema:


```
xmllint -noout --schema tablon_simp_tipos.xsd tablon_simp.xml
```
- 3º Edite el fichero `tablon_simp_tipos.xsd` y cambie el orden de las definiciones. Vuelva a validar el fichero ¿es equivalente el esquema?

Con las definiciones de tipo anteriores, es fácil modificar el esquema para añadir nuevas especificaciones. Por ejemplo, si se quiere que `precio` esté comprendido entre 0 y 1000, y que `anuncio` admita los atributos `codigo` y `vendido` (del tipo `xs:string`), el fichero `xsd` sería:

tablon_comp.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:simpleType name="Tprecio">
    <xs:restriction base="xs:decimal">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="1000"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="Tanuncio">
    <xs:sequence>
      <xs:element name="fecha" type="xs:date" />
      <xs:element name="asunto" type="xs:string" />
      <xs:element name="texto" type="xs:string" />
      <xs:element name="precio" type="Tprecio" />
      <xs:element name="contacto" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="codigo" type="xs:string"/>
    <xs:attribute name="vendido" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="Ttablon">
    <xs:sequence>
      <xs:element name="anuncio" type="Tanuncio"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="tablon" type="Ttablon" />

</xs:schema>
```

Observe en el ejemplo anterior cómo se ha definido un tipo simple al que se le ha dado el nombre `Tprecio` y que está basado en el tipo `xs:decimal` pero que además tiene dos restricciones. Ese tipo simple es el que se usa para especificar el elemento `precio`.

TAREAS

- 1º Descargue el fichero `tablon_comp.xsd`
- 2º Use la herramienta `xmllint` para validar el fichero `xml` con este nuevo esquema:

```
xmllint -noout --schema tablon_comp.xsd tablon_simp.xml
```
- 3º Edite el fichero `tablon_simp.xml` y cambie el precio de algún elemento para que no cumpla la restricción. Vuelva a validar el fichero ¿es ahora válido?
- 4º Edite el fichero `tablon_simp.xml` y añada un atributo de los permitidos a algún anuncio, por ejemplo modifique la etiqueta de apertura de un anuncio para que quede `<anuncio vendido="si">`. Vuelva a validar el fichero ¿es ahora válido?