

# Lenguaje HTML

---

### 3.1. Introducción

HTML (*HyperText Markup Language*) es el lenguaje utilizado en Internet para definir las páginas del *World Wide Web*. Los ficheros HTML son ficheros de texto puramente ASCII, que pueden ser escritos con cualquier editor básico, tal como *Notepad* en *Windows* o *vi* en *Unix*. También se pueden utilizar procesadores de texto más complicados como *Microsoft Word*, pero en este caso hay que asegurarse que el fichero es guardado en disco como "text only". En este fichero de texto se introducen unas marcas o caracteres de control llamadas TAGs (en esto, HTML se parece a los primeros procesadores de texto), que son interpretadas por el browser. Cuando éste lee un fichero ASCII con extensión *\*.htm* o *\*.html* interpreta estas TAGs y formatea el texto de acuerdo con ellas.

En general puede decirse que HTML es un lenguaje sencillo y eficiente. Aunque no puede competir con los procesadores de texto en capacidades de formato, es universal, es hipertexto e hipermedia, es muy accesible, sus ficheros ocupan poco espacio en disco; por otra parte es fácil de interpretar y de enviar a través de las redes. De hecho, es uno de los estándares en los cuales las empresas están basando sus *Intranets* y sus servicios de información interna.

En la página web de la asignatura está disponible la especificación HTML 4.01 (última versión).

### 3.2. Tags generales

El formato con el que imprime o visualiza un fichero HTML depende de unas marcas especiales llamadas TAGs. Todas las TAGs de HTML van encerradas entre los caracteres "<" y ">", como por ejemplo <HTML>. Además, la mayor parte de ellas son **dobles**: hay una TAG de comienzo y otra de final; entre ambas hay un **contenido** o texto afectado por dichas TAGs. La TAG de final es como la de comienzo, pero incluyendo una barra (/) antes del nombre, por ejemplo </HTML>. Normalmente las TAGs se escriben en letra mayúscula para hacer más legible el cuerpo del documento (distinguiéndolas del texto ordinario), pero de hecho también se pueden escribir en minúsculas. La forma general de estas TAGs dobles es la siguiente:

```
<COMANDO>Texto afectado</COMANDO>
```

Existen también TAGs **simples** (sin TAG de cierre), que no tienen un texto contenido al que se aplican. Por ejemplo, la marca de comienzo de párrafo <P> era un comando simple hasta la versión 3.0 de HTML, pues cada vez que empieza un párrafo se sobreentiende que ha terminado el anterior. La versión 3.0 de HTML ha introducido la TAG </P>, aunque su uso es opcional.

Una TAG, tanto si es **doble** o **simple**, puede tener uno o más **atributos**, que son parámetros que definen su forma de actuar. Los atributos se incluyen después del nombre de la TAG, antes del carácter ">", separados por uno o más blancos. Si la TAG es doble, los atributos se incluyen siempre en la TAG de apertura.

Los atributos que aún se soportan pero pueden quedar obsoletos en futuras versiones se indicarán como **[deprecated]**. Estos atributos han sido sustituidos por otras alternativas, generalmente utilizando hojas de estilo (*style sheets*), que no se verán en esta asignatura.

Se puede anidar unas TAGs dentro de otras. Se podría decir que en este caso los efectos son acumulativos, como en el siguiente ejemplo:

```

<TAG1>
  Texto afectado por el comando 1
  <TAG2>
    Texto afectado por las TAGs 1 y 2
  </TAG2>
</TAG1>

```

**Advertencia:** La *anidación* de TAGs debe hacerse de forma coherente, pues de lo contrario puede dar unos resultados que no son los esperados: la última TAG en abrirse debe ser la primera en ser cerrada, es decir que no se pueden entrecruzar los dominios de actuación de las TAGs.

Todos los ficheros HTML tienen una estructura similar a la siguiente:

```

<HTML>
  <HEAD>
    <TITLE>Título de la página</TITLE>
    ...
  </HEAD>
  <BODY>
    ...
  </BODY>
</HTML>

```

donde las marcas mostradas indican lo siguiente:

- <HTML>...</HTML>: Indica el comienzo y el fin de un documento HTML. Es necesario para que el browser sepa que tiene que interpretar un fichero en lenguaje HTML.
- <HEAD>...</HEAD>: Indica la cabecera del documento donde se guarda diversa información sobre el mismo, como por ejemplo el título de la ventana en la que aparecerá (<TITLE>Este es el título</TITLE>).
- <BODY>...</BODY>: En él aparece el cuerpo del documento Web propiamente dicho, que contiene el texto, las imágenes, los enlaces o *links* a otras páginas, etc.

### 3.3. Formato de texto

En este apartado se verá cómo organizar y formatear el texto de un documento.

#### 3.3.1. TAGs generales de un documento.

Existen una serie de atributos para la TAG BODY, que afectarán a todo el documento, dado que todo el código de las páginas web se escribe dentro de la TAG doble BODY. Entre otros, los más utilizados son éstos:

- *[deprecated]* BGCOLOR="nombre\_color" ó BGCOLOR="#00FF00", donde *nombre\_color* es uno de los nombres reconocidos por los browsers (existe una tabla definida por Netscape, mayoritariamente aceptada), y "#00FF00" es el código RGB del color deseado (dos dígitos hexadecimales por cada color fundamental rojo, verde y azul, para representar hasta 256 intensidades de cada color, y unos 16 millones de colores distintos). Esta segunda opción es la más empleada, por ser la estándar ( y por ello no da errores).
- *[deprecated]* BACKGROUND="imagen.ext": sirve para poner un dibujo como fondo de documento, siendo *imagen.ext* el nombre del fichero y la extensión. Si el dibujo no ocupa la pantalla entera, se dispondrá en mosaico hasta llenarla.

```

<BODY BACKGROUND="imagen.ext">
...
</BODY>

```

- *[deprecated]* TEXT="código de color": define el color del texto del documento.
- *[deprecated]* LINK="código de color": define el color del texto definido como *link*.
- *[deprecated]* VLINK="código de color": define el color de los links visitados (es decir, los links sobre los que ya se ha clicado en alguna ocasión).

### 3.3.2. Comentarios en HTML

El lenguaje HTML permite introducir comentarios, con el fin de que un documento sea más claro y comprensible cuando se analiza su código. Lógicamente, tales comentarios no aparecerán en el browser. La sintaxis de los comentarios, que pueden ocupar tantas líneas como se desee, es la que sigue:

```
<!-- Comentario -->
```

### 3.3.3. Caracteres de separación en HTML

Se consideran caracteres de separación el espacio en blanco, el tabulador y el salto de línea. Un punto importante a considerar es que, en principio, HTML considera varios caracteres de separación consecutivos como un único carácter de separación. Además, les otorga a todos ellos la misma categoría, es decir, considera todos como si fuesen espacios en blanco. Esto quiere decir que el aspecto del fichero ASCII que contiene el código HTML y el aspecto del documento visto en el browser pueden diferir notablemente. De ahí que al escribir un fichero luego haya que comprobarlo en el browser para ver si el resultado final es el esperado. Por ejemplo, introducir un salto de línea en un fichero HTML no implica, en principio, un salto de línea en la página Web.

Así pues, para organizar el texto HTML no se basa en los caracteres de separación citados sino en TAGs especiales para ello. Esas TAGs se considerarán en el apartado siguiente. Los puntos suspensivos (...) entre las TAGs de comienzo y final representan un contenido a concretar

### 3.3.4. TAGs de organización o partición de un documento

Las TAGs más importantes de organización del contenido de un documento son las siguientes:

- **Tag <P>...</P> (de *paragraph*).** Cada párrafo deberá ir incluido entre estas TAGs. Entre párrafo y párrafo se deja el espacio correspondiente a una línea en blanco (aproximadamente).
- Se puede añadir a <P> el atributo ALIGN [*deprecated*] para especificar cómo debe ser alineado un párrafo. Este atributo puede tomar los valores "LEFT", "CENTER" y "RIGHT".
- **Tag <BR> (de *break*).** Salta de línea sin dejar una línea en blanco a continuación. Es un comando simple.
- **Tag <HR> (de *hard rule*).** Este comando simple introduce una *línea horizontal* que deja espacios antes y después, por lo que no hace falta añadir <P> o <BR> suplementarios.

Algunos atributos de este comando son:

- [*deprecated*] WIDTH: define la longitud de la línea horizontal en la pantalla, que puede especificarse usando un valor absoluto (número de pixels) o bien indicando un porcentaje (opción recomendada, ya que no es posible conocer a priori la anchura de la ventana del cliente Web), con lo que el resultado final podría no ser el esperado. El valor por defecto es del 100%.

```
<HR WIDTH=75%>
```

- [*deprecated*] ALIGN: alinea la línea donde se desee. Puede tomar los valores LEFT, RIGHT y CENTER

```
<HR ALIGN=RIGHT>
```

- [*deprecated*] SIZE: especifica la anchura o grosor de la línea horizontal en número de pixels. El valor por defecto es 1 pixel

```
<HR SIZE=4>
```

- [*deprecated*] NOSHADE: Por defecto, estas líneas aparecen sombreadas, con un cierto efecto 3-D. Con este comando, tal efecto desaparecerá.

Puede observarse que no hay ningún atributo que haga referencia al color de la línea, que siempre se dibuja en negro. Conviene tener esto en cuenta a la hora de elegir colores de fondo.

### 3.3.5. Formateo de textos sin particiones

- **Tag <PRE>...</PRE> (de *preformatted*).** Como ya se ha mencionado, el browser ignorará de ordinario todos los espaciados de línea, múltiples espacios consecutivos, tabuladores, etc. Por lo general esta práctica es muy adecuada, pero pueden presentarse ocasiones en las que se desee un formato específico de texto, que aparezca en la pantalla tal y como se ha escrito (un caso típico son los listados de programas de ordenador, poesías, letras de canciones, etc.). Este objetivo se logra

encerrando el texto entre el comando doble `<PRE> texto </PRE>`, tal como puede verse en la Figura 3.1.

```

Por ejemplo,
    si quisiera
      crear alguna
        figura con
          palabras,
            Necesitaría
              usar
                el TAG
                  PRE
                    para
                      hacerlo.
  
```

Figura 3.1

- **Tag `<BLOCKQUOTE>...</BLOCKQUOTE>` (de *block quote*).** Esta TAG de formato de párrafos es muy útil. Inserta un salto de párrafo e indenta todo el texto que viene a continuación, hasta llegar a su TAG emparejada que, a su vez, insertará otro salto de párrafo y suprimirá la indentación anterior. Así la TAG `BLOCKQUOTE` debe emplearse para indentar bloques determinados de textos, pero en ningún caso con el único objetivo de mover el margen, ya que el comportamiento del texto afectado por este comando varía según los browsers.

### 3.3.6. Centrado de párrafos y cabeceras con `<CENTER>`

- **Tag `<CENTER>...</CENTER>`.** Esta TAG se emplea para centrar todo tipo de texto, figuras, tablas y otros elementos, como por ejemplo:

```
<CENTER> Texto afectado por el centrado </CENTER>
```

### 3.3.7. Efectos de formato en texto

- **Tag `<H>...</H>` (de *heading*).** Es una tag doble que toma necesariamente un argumento, un número comprendido entre 1 y 6, de tal modo que `<H1>` es un título de primer nivel (normalmente el más grande) y `<H6>` es un título de sexto nivel (de ordinario el más pequeño, aunque el hecho de que sea más grande o pequeño que el estándar depende del browser). Inserta automáticamente un salto de párrafo, antes y después. Puede tomar como atributo `ALIGN` [*deprecated*], utilizado de la misma forma que en `<P>`
- **Tags `<B>... </B>`, `<I>... </I>` y `<U>... </U>` (de *bold*, *italic* y *underlined* respectivamente).** `<B>` hace que el texto afectado esté en negrita (bold), `<I>` aplica el estilo cursiva y `<U>` [*deprecated*] lo subraya.
- **Tag `<TT>...</TT>` (de *teletype*).** Esta TAG hace que la letra que se imprima sea de estilo teletipo, es decir, que su apariencia sea similar a la que de una máquina de escribir (fuente Courier o similar).
- **Tag `<FONT>...</FONT>` [*deprecated*].** Esta TAG sirve para controlar, por medio de sus atributos, el tamaño, el color y el tipo de letra con el que se representará el texto (si el browser es capaz de ello). Los atributos de esta TAG son los siguientes:
  - [*deprecated*] `SIZE=n` (siendo *n* un entero entre 1 y 7). Con este atributo se puede cambiar el tamaño de la letra. El tamaño base o tamaño por defecto es el 3.
  - [*deprecated*] `SIZE=+n` ó `SIZE=-n`. Con esta TAG se cambia el tamaño de la letra respecto al tamaño base, aumentando o reduciéndolo según se aplique un incremento positivo o negativo. Los tamaños deben estar comprendidos en la escala de 1 a 7.
  - [*deprecated*] `COLOR="nombre_color"` ó `COLOR="#00FF00"`, donde *nombre\_color* es uno de los nombres reconocidos por los browsers y `#00FF00` es el código RGB del color deseado.
  - [*deprecated*] `FACE="tipo_letra"`, donde *tipo\_letra* puede ser "Arial", "Times New Roman", etc.

Ejemplo: `<FONT SIZE=4 COLOR="Blue" FACE="Arial">Texto a representar</FONT>`

- **Tag `<BASEFONT SIZE=n>` [*deprecated*].** Establece el tamaño de letra base o por defecto. Es un TAG simple.
- **Tags `<SUP>...</SUP>` y `<SUB>...</SUB>` (de *superscript* y *subscript*).** `<sup>` y `<sub>` crean los estilos de superíndice o subíndice, respectivamente.

- **Tags <BIG>...</BIG> y <SMALL>...</SMALL>.** Las TAGs <BIG> y <SMALL> hacen al texto más grande y más pequeño respectivamente.

### 3.3.8. Caracteres especiales

Existen algunos caracteres, denominados especiales, que merecen una pequeña mención. Los que tienen más utilidad para el castellano son:

Las vocales acentuadas se escriben en HTML como sigue:

- Si son mayúsculas, &(LETRAMAYÚSC)*acute*. Por ejemplo, Á se escribe &Aacute.
- Si son minúsculas, &(letramin)*acute*. Por ejemplo, é se escribe &eacute.
- Ñ es &Ntilde y ñ es &ntilde.

## 3.4. Listas

### 3.4.1. Listas desordenadas

Una lista desordenada es una lista de elementos en líneas separadas precedidas de un guión o similar (en Inglés *bullet*).

- **Tag <UL>...</UL> (de *unordered list*):** comienzo de una lista no ordenada

Un atributo de esta Tag es TYPE [*deprecated*], que define la forma del *bullet* y puede tomar los valores: DISC(□), CIRCLE(●), que es el valor por defecto y SQUARE(■).

- **Tag <LI> (de *line*):** nuevo elemento de la lista.

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    Esto es una lista no ordenada:
    <UL>
      <LI>Elemento 1
      <LI>Elemento 2
      <LI>Elemento 3
    </UL>
  </BODY>
</HTML>
```

Esto es una lista no ordenada:

- Elemento 1
- Elemento 2
- Elemento 3

### 3.4.2. Listas ordenadas

En este tipo de listas se emplean números en lugar de *bullets*.

- **Tag <OL>...</OL> (de *ordered list*):** comienzo de una lista ordenada.

Su atributo TYPE [*deprecated*] permite definir el tipo de numeración. Puede tomar los valores: A (para numerar A,B,C...), a (a,b,c...), I (I,II,III...), i (i,ii,iii), 1 (1,2,3...) que es el valor por defecto.

- **Tag <LI> (de *line*):** nuevo elemento de la lista .

Su atributo VALUE [*deprecated*] permite definir a partir de qué valor se empieza a numerar. Obviamente, por defecto comienza a numerarse por el principio. Por ejemplo, <LI VALUE=iii> numerará iii,iv,v... Cada vez que se utilice este atributo VALUE dentro de una misma lista ordenada se rompe con la numeración anterior.

```

<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    Esto es una lista ordenada:
    <OL>
      <LI>Elemento 1
      <LI>Elemento 2
      <LI>Elemento 3
    </OL>
  </BODY>
</HTML>

```

Esto es una lista ordenada:

1. Elemento 1
2. Elemento 2
3. Elemento 3

### 3.4.3. Listas de definiciones

Este tipo de lista es útil para obtener resultados como el que se ve a continuación:

```

Término 1
  Definición del término 1
Término 2
  Definición del término 2
...

```

- Tag <DL>...</DL> (de *definition list*). comienzo de la definición de la lista.
- Tag <DT> (de *definition term*) - nueva definición.
- Tag <DD> (de *definition description*) - cuerpo o descripción de la definición.

## 3.5. Imágenes

El lenguaje HTML además de hipertexto, es hipermedia, es decir, que por lo tanto permite incluir información de tipo gráfico. Las imágenes se tratan de forma separada al texto, y, debido al tamaño que éstas suelen tener en general, se tarda más tiempo tanto en generar como en visualizar páginas HTML con imágenes.

- Tag **IMG** (de *image*): Es un comando simple que se utiliza para insertar una imagen en el documento. Algunos atributos de esta tag son:

- SRC="imagen.ext": es requerido y sirve para indicar dónde se va a encontrar la imagen, siendo *imagen.ext* el nombre del fichero y la extensión.

```
<IMG SRC="imagen.ext">
```

- ALT="texto" donde *texto* es el texto que se verá en lugar de la imagen, bien porque se han desactivado los gráficos para navegar más rápidamente por Internet, bien porque el browser es "text only". Es requerido para evitar problemas en dichos casos.
- WIDTH y HEIGHT: controlan el tamaño de la ventana, que puede especificarse usando un valor absoluto (número de pixels) o bien indicando un porcentaje (opción recomendada, ya que no es posible conocer a priori la anchura de la ventana del cliente Web).

```
<IMG WIDTH=33% HEIGHT=60% SRC="imagen.ext">
```

- ALIGN: alinea el texto donde se desee respecto a la imagen. Puede tomar los valores TOP, BOTTOM, MIDDLE, LEFT, RIGHT, TEXTTOP, ABSMIDDLE, BASELINE, ABSBOTTON.
- BORDER=n. Permite añadir un borde o marco a la imagen, siendo *n* la anchura de éste en pixels. BORDER=0 implica la supresión del borde.
- HSPACE=n y VSPACE=n. Estos atributos dejan un espacio horizontal y vertical respectivamente alrededor de la imagen, siendo *n* el valor en pixels.
- LOWSRC=URL carga primero una imagen de baja resolución y cuando termina de cargar todo el documento la sustituye por la versión de alta resolución.

## 3.6. Links

Encerrando un texto o un gráfico entre las TAGs `<A> ... </A>` se consigue convertir dicho texto en hipertexto. Este link puede hacerse, entre otras posibilidades, a:

- Otra página Web.
- El correo electrónico de otra persona (se emplea el link para enviarles correo).
- Un archivo local (¡práctica a evitar si se pretende que los demás puedan acceder a ese link!).
- Una dirección relativa a documentos en directorios superiores o inferiores al directorio en que se está en ese momento o en el directorio actual.

Por lo general, los links tienen la forma `<A HREF="URL">texto</A>`. A su vez, los URLs tienen una estructura del tipo siguiente:

```
método://localización/archivo/
```

donde *método* hace referencia al modo de acceder al fichero (http, ftp, gopher, news, mailto), *localización* es el lugar donde se encuentra el fichero actualmente y *archivo* es el nombre completo del *archivo* requerido en el sistema especificado. Veamos algunos ejemplos:

```
<A HREF="http://www.ceit.es/">WEB CEIT</A>
```

lleva al web del CEIT al clicar sobre las palabras WEB CEIT, que aparecerán subrayadas y en otro color (hiperlink).

```
<A HREF="mailto://president@whitehouse.gov">mail</A>
```

mandará un e-mail al Presidente de los EE.UU.

En el caso en que se quiera "saltar" a otro punto del propio documento, se procederá como se ve en el siguiente ejemplo: Establecemos el link `<A NAME="aliniociodeldocumento">` (Atención: si es posible evitar los espacios, es mejor hacerlo, pues pueden originar problemas). A este tipo de links se les denomina anclas o "anchors" puesto definen zonas fijas de un documento a las que se va a hacer referencia. Una vez efectuada esta operación, si ahora hacemos `<A HREF="#aliniociodeldocumento"> texto </A>`, al clicar sobre la palabra *texto* iremos al lugar donde se ha definido el link.

## 3.7. Tablas

- Las tablas se definen empleando los códigos pareados `<TABLE>` y `</TABLE>`.
- Las celdas se agrupan en filas, que se definen con los códigos pareados `<TR>` y `</TR>` (de *Table Row*).
- Una tabla se compone de celdas de datos. Una celda se define usando los códigos pareados `<TD>` y `</TD>` (de *Table Data*).
- Las celdas pueden contener cualquier elemento HTML: texto, imágenes, enlaces e incluso otras tablas anidadas.
- `<TH>...</TH>` (de *Table Header*): Para crear celdas cuyo texto sea resaltado (por ejemplo, en los encabezamientos) se sustituye el TD habitual por TH.

Algunos atributos que pueden añadirse a TABLE son:

- WIDTH: especifica la anchura de la tabla. Puede darse un valor absoluto en pixels o un valor relativo en porcentaje. Si no se especifica la anchura por defecto dependerá del browser.
- BORDER: Si se quiere que la tabla posea bordes: `<TABLE BORDER(=n)> ... </TABLE>`
- siendo n el grosor del borde en pixels. Por ser opcional (toma un valor por defecto) se ha denotado entre paréntesis.
- *[deprecated]* ALIGN: Posiciona la tabla respecto al documento. POSIC puede tomar los valores LEFT, CENTER y RIGHT.
- CELLSPACING: espaciado entre celdillas: `<TABLE CELLSPACING(=n)> ... </TABLE>`
- CELLPADDING(=n): permite especificar el ancho que debe existir desde los bordes de cada celdilla a los elementos en ella incluidos.

- *[deprecated]* BGCOLOR: indica el color de fondo de la tabla.

Algunos atributos de TR son:

- ALIGN: se utiliza para determinar la alineación del contenido de las celdas de la fila.
- VALIGN: se utiliza para determinar la posición vertical del contenido de las celdas de la fila.

- *[deprecated]* BGCOLOR: indica el color de fondo de la fila.

Algunos atributos que pueden añadirse a TH y TD son:

- ROWSPAN(=n): indica el número de filas que debe abarcar la celda actual.
- COLSPAN (=n): indica el número de columnas de la fila que abarcará la celda actual.
- ALIGN: se utiliza para determinar la alineación del contenido de la celda.
- VALIGN: se utiliza para determinar la posición vertical del contenido de la celda.
- WIDTH: especifica la anchura de la celda, en pixels o en porcentaje.
- HEIGHT: especifica la altura de la celda, en pixels o en porcentaje.
- *[deprecated]* BGCOLOR: indica el color de fondo de la celda.

Un ejemplo de tabla con varios de los elementos vistos sería la de la Figura 1.4., que corresponde al siguiente código:

```
<HTML>
<HEAD><TITLE>Ejemplo de tabla</TITLE></HEAD>
<BODY>
<TABLE BORDER=3>
  <CAPTION><STRONG>Ejemplo</STRONG></CAPTION>
  <TR>
    <TD>Elemento 1</TD>
    <TD>Elemento 2</TD>
    <TD ROWSPAN=3>Elemento</TD>
    <TD>Total</TD>
  </TR>
  <TR>
    <TD>Elemento 3</TD>
    <TD>Elemento 4</TD>
  </TR>
  <TR>
    <TD>Elemento 5</TD>
    <TD>Elemento 6</TD>
  </TR>
  <TR>
    <TD colspan=3><CENTER>Elemento</CENTER></TD>
  </TR>
</TABLE>
</BODY>
</HTML>
```

Ejemplo			
Elemento 1	Elemento 2	Elemento	Total
Elemento 3	Elemento 4		
Elemento 5	Elemento 6		
Elemento			

## 3.8. Frames

### 3.8.1. Introducción a los frames

Una ventana HTML puede ser dividida en *subventanas* o **frames**, en cada una de las cuales se puede mostrar un documento o un URL distinto. Cada **frame** puede tener un *nombre* al que se puede dirigir el resultado (mediante el atributo TARGET, *objetivo*) de una acción (por ejemplo clicar un hiperlink). El tamaño de los **frames** se puede cambiar interactivamente con el ratón (salvo que se haya eliminado ésta posibilidad explícitamente, al definirlos.).

Puede haber **frames** de contenido fijo o estático (por ejemplo un logo de una empresa, un anuncio) y otros cuyo contenido vaya variando. Un **frame** puede contener un índice y otro (más grande) va mostrando los apartados a los que se hace referencia en el índice.

Se llama **frameset** a una ventana con **frames**. Un **frameset** puede contener otros **framesets**.



Dentro del TAG doble `<FRAMESET>... </FRAMESET>` sólo puede haber los siguientes contenidos:

1. Otro `<FRAMESET>...</FRAMESET>`
2. Definición de *frames* con el TAG simple `<FRAME>`

Para crear *frames* es necesario un fichero HTML con una estructura semejante a la siguiente (obsérvese que la TAG `<FRAMESET>` sustituye a `<BODY>`):

```
<HTML>
  <HEAD><TITLE>Título</TITLE></HEAD>
  <FRAMESET>
    Definición de marcos o frames
  </FRAMESET>
</HTML>
```

La TAG `<FRAMESET>` tiene dos atributos: `COLS` y `ROWS`. Ambos atributos admiten una lista de valores separados por comas; estos pueden ser dados en valores absolutos (nº de pixels) o en porcentajes (muy recomendable). En este contexto el asterisco (\*) representa el resto del tamaño disponible.

A continuación se exponen algunos ejemplos:

- Para dividir verticalmente la ventana en dos partes separadas por una línea vertical:
 

```
<FRAMESET COLS="20%, 80%">
```
- Si se desea reservar dos áreas horizontales de 80 y 160 pixels y una tercera con el resto de la ventana:
 

```
<FRAMESET ROWS="80, 160, *">
```
- Para crear una rejilla de 2x3 ventanas:
 

```
<FRAMESET rows="30%,70%" cols="33%,34%,33%">
```

Como no se sabe la resolución de la pantalla en la que se ejecutará el browser, conviene siempre utilizar tamaños relativos, o si se utilizan tamaños absolutos, disponer de una partición cuyo tamaño se determina con el asterisco (\*). Hay que tener en cuenta que si se utilizan porcentajes relativos que no suman 100, a dichos porcentajes se les aplica un factor de escala de modo que representen correctamente los tamaños relativos de los distintos *frames*.

La TAG `<FRAME>` es simple y define las propiedades de cada *frame* en el *frameset*. Algunos de sus atributos son:

- `<FRAME SRC="URL">` Define el contenido del *frame*, es decir el URL cuyo contenido se mostrará en él. Sin este atributo aparecerá en blanco.
 

```
<FRAMESET COLS="20%,60%,20%">
  <FRAME SRC="frame1.html">
  <FRAME SRC="frame2.html">
  <FRAME SRC="frame3.html">
</FRAMESET>
```

En este ejemplo se crea una división vertical de la ventana del browser en tres frames, en los que aparecen las páginas: *frame1.html*, *frame2.html* y *frame3.html*.

- `NAME="nombre"`: Define un "nombre" para poder dirigir a ese *frame* contenidos poniendo ese nombre en el atributo `TARGET` de un hiperlink.
- `MARGINWIDTH="n_pixels"`: es opcional, y determina la distancia horizontal del contenido a los márgenes de los *frames*.
- `MARGINHEIGHT="n_pixels"`: Similar al anterior para la distancia vertical.
- `SCROLLING="YES/NO/AUTO"`: Para que haya o no barras de desplazamiento en el *frame*. Con `AUTO` las habrá o no según quepa o no el contenido.
- `NORESIZE`: El usuario no podrá cambiar el tamaño del *frame* con el ratón.

Véase el siguiente ejemplo, correspondiente a la Figura 3.2 con frames anidados:

```
<HTML><HEAD><TITLE>Ejemplo de frames</TITLE></HEAD>
<FRAMESET ROWS="30%,30%,40%">
  <FRAME SRC="PRAC5.HTM" NORESIZE>
  <FRAMESET cols="25%,25%,50%">
    <FRAME SRC="prac1.htm">
    <FRAME SRC="prac2.htm">
    <FRAME SRC="prac3.htm">
  </FRAMESET>
  <FRAME SRC="PRAC4.HTM">
</FRAMESET>
</HTML>
```

Para los browsers que no admiten frames se puede emplear la TAG doble <NOFRAMES> y </NOFRAMES>, pudiendo visualizarse el documento sin problemas de forma alternativa:

```
<HTML><HEAD><TITLE> Ejemplo de FRAMES</TITLE></HEAD>
<FRAMESET COLS="80%, 20%">
  <FRAME SRC="Indice.htm">
  <FRAME SRC="Tema1.htm">
</FRAMESET>
<NOFRAMES>
  <A HREF="Tema1.htm">Tema 1</A><BR>
</NOFRAMES>
</HTML>
```

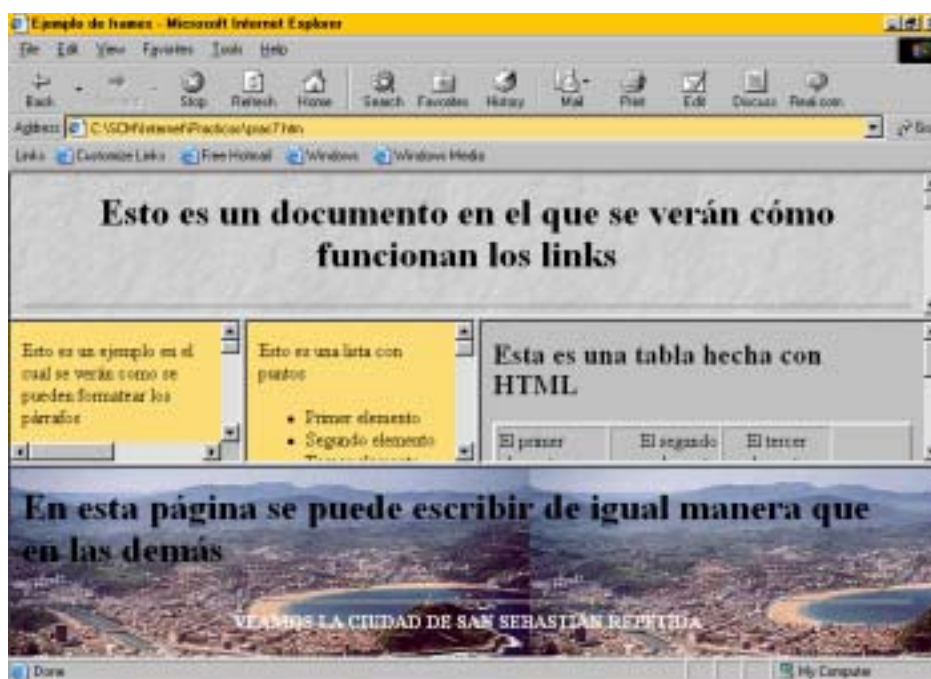


Figura 3.2.

### 3.8.2. Salida a ventanas o frames concretas de un browser.

Sin *frames*, al clicar en un *link* el nuevo documento aparecía en la misma ventana en que se estaba (o en una ventana nueva si así se había establecido en el browser).

Cuando se utilizan *frames* se puede especificar dónde se desea que aparezca el resultado cuando se clique en un *hiperlink*. Para ello se pueden asignar nombres (mediante el atributo NAME) a las ventanas o *frames* donde se mostrarán los documentos.

A su vez, hay que asignar un atributo TARGET que haga referencia al NAME de un frame o ventana. Si no existe, se creará una ventana con dicho nombre.

- TARGET en la TAG <A>...</A>: <A HREF="URL" TARGET="nom\_ventana">Texto</A>  
donde el documento indicado en el URL aparecerá en la ventana cuyo nombre sea "nom\_ventana".

- TARGET con la TAG <BASE>, para hacer que todos los *links* de un documento se dirijan al mismo TARGET. Así se establece un TARGET por defecto para todo el documento, que puede cambiarse con TARGETs específicos en TAGs tipo <A>. La TAG <BASE> debe incluirse en el <HEAD> del documento.

```
<HEAD>
  <TITLE>Our Products</TITLE>
  <BASE href="http://www.aviary.com/intro.html" TARGET="ventana">
</HEAD>
```

- TARGET con la TAG <FORM>, de modo que los resultados de un programa CGI (salida estándar) se dirijan a la ventana indicada. Los formularios <FORM> y los CGI se estudiarán más adelante.

```
<FORM ACTION="URL" TARGET="nom_ventana">
```

### 3.8.3. Nombres de TARGET

El atributo TARGET puede contener nombres de ventana. Los nombres válidos son los siguientes:

1. Cualquier nombre que empiece por carácter alfanumérico.
2. Los nombres que empiezan por el carácter (\_) son especiales:
  - TARGET="\_blank": Salida dirigida a una ventana nueva en blanco y sin nombre.
  - TARGET="\_self". Salida dirigida a la propia ventana del *hiperlink*.
  - TARGET="\_parent". Salida dirigida al *frameset* padre del documento actual. Si no hay padre el resultado es como el de \_self.
  - TARGET="\_Top". Destruye todas las FRAMES que haya y la salida se dirige a la ventana principal del browser.

**Advertencia:** No se admiten *framesets* recursivos (cuando en un *frame* de un *frameset*, se carga el propio *frameset*); si se intenta actúa como FRAME="\_blank".

## 3.9. Mapas de imágenes o imágenes sensibles

### 3.9.1. Cómo funciona un mapa de imágenes

Los *mapas de imágenes* son imágenes clicables que permiten al usuario acceder a un URL o a otro dependiendo de dónde se clica sobre dicha imagen. Inicialmente las imágenes sensibles eran procesadas en el servidor remoto http, por lo que se perdía mucho tiempo y además no se indicaba la dirección a la que se iba a acceder. La solución a estos problemas fue procesar las imágenes sensibles en el browser, acelerando el proceso en gran medida y con la ventaja adicional de que el browser informa en la barra de estado acerca de la dirección URL a la que se va a acceder si se clica en ese punto.

### 3.9.2. Mapas de imágenes procesados por el browser

Un mapa de imágenes se elabora mediante varias TAGs simples <AREA> entre las TAG dobles <MAP>...</MAP>. La TAG <MAP> debe tener un atributo NAME para identificar el mapa de imágenes.

- Las TAGs simples <AREA> definen tantas zonas geométricas activas sobre una imagen como se desee, y pueden tener varios atributos:
- HREF="URL". Define el URL de destino del área.
- ALT="texto". Contiene una descripción textual alternativa del área (como en <IMG>). Se requiere para que los browsers que no tengan gráficos disponibles puedan utilizar satisfactoriamente el Web.
- SHAPE="forma". Indica la forma de la zona activa; puede tomar los valores: POLY, CIRCLE, RECT (siendo este último el valor por defecto).
- COORDS="x1,y1,x2,y2,..." Este atributo define una lista de coordenadas (en pixels) separadas por comas que determinan las zonas activas. Las coordenadas son relativas a la esquina superior izquierda de la imagen. El orden y el valor de estas coordenadas depende del atributo SHAPE:
  - En el caso de RECT: X izda., Y sup. X dcha., Y inf.
  - En el caso de CIRCLE: X del centro, Y del centro, radio.
  - En el caso de POLY: X e Y de cada uno de los vértices, siendo el último par de coordenadas coincidente con el primero para cerrar el polígono.

- NOHREF. Este es un atributo opcional que indica que no se debe realizar ninguna acción si el usuario clicla en la zona correspondiente. En todo caso, las zonas de la imagen no incluidas en ninguna zona activa definida se consideran de tipo NOHREF.

Para hacer referencia a un determinado mapa de imágenes se utiliza el atributo USEMAP a la TAG <IMG>, indicando el NAME del mapa deseado como valor del atributo USEMAP. A continuación se expone un ejemplo:

```
<HTML><BODY>
<IMG SRC="concha.gif" USEMAP="#FOTO" ALT="Bahía de San Sebastián">
<MAP NAME="FOTO">
<AREA SHAPE="RECT" COORDS=20,25,155,83 HREF="historia.html" ALT="Historia">
<AREA SHAPE="CIRCLE" COORDS=60,150,100,150 HREF="plano.html" ALT="Planos">
<AREA SHAPE="POLY" COORDS=106,100,126,170,254,170,254,49,222,100
HREF="fotos.html" ALT="Fotos">
<AREA SHAPE="POLY" COORDS=169,26,169,93,267,26 NOHREF ALT="Idioma">
</MAP>
</BODY></HTML>
```

Para que funcione este código se ha creado el gráfico *concha.gif* incluyendo el dibujo de los contornos de las futuras zonas activas. Se han obtenido las coordenadas de las zonas activas y se han escrito en el atributo COORDS tal y como se ha explicado. Se han creado también los archivos *plano.html*, *historia.html* y *fotos.html*, a los que se accederá tras cliclar sobre esas áreas.

En este ejemplo, se supone que la opción del idioma no está aún preparada. Por ello, la TAG AREA no hace referencia a ningún archivo o link. Se ha empleado el atributo NOHREF para indicar expresamente que no se acceda a ningún URL al cliclar en esta zona. De este modo, cuando esté ya preparado el archivo *idiomas.html*, no habrá más que cambiar esta parte de código. Finalmente, podrá decirse que la imagen sensible funciona si al desplazar el ratón sin cliclar sobre un área activa aparece en la barra de estado la dirección a la que se accederá si se clicla sobre ella. Obviamente, esto último no se cumple para las áreas definidas con NOHREF.



Figura 3.3.

### 3.10. Editores y conversores HTML

Con un simple editor de texto como *Notepad* se pueden crear los ficheros HTML más sofisticados. Sin embargo, hay que señalar que la tarea se simplifica mucho si se dispone de un buen *editor* o *conversor* especialmente preparado para producir este tipo de ficheros.

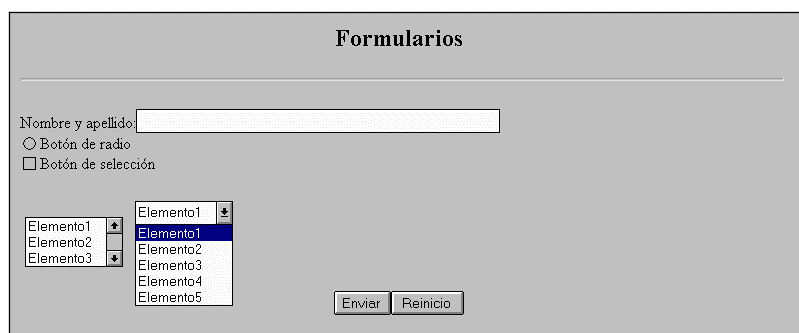
Los *editores* permiten introducir de modo automático o semiautomático las TAGs de HTML a medida que se va tecleando el texto. Los hay de dos tipos: editores automáticos *WYSIWYG* (*What You See Is What You Get*) en los que al introducir el texto se muestra tal como se verá en el browser, y editores semiautomáticos más sencillos que simplemente proporcionan ayuda para ir introduciendo los TAGs; de ordinario estos editores permiten llamar al browser preferido pulsando un simple botón, para comprobar cómo se ve el fichero que se está creando. Son editores semiautomáticos *CMED* y *HotDog* y automáticos los editores *Netscape Navigator Gold* y *FrontPage HTML Editor*.

Los *conversores* son programas que convierten los formatos propios de una aplicación (por ejemplo *Word*, *Excel* y *Powerpoint*) a formato HTML. Es posible que en la conversión se pierdan o cambien algunas características de formato, pero eso no resta utilidad a los conversores. Cada vez más, los conversores son programas integrados, es decir, es la propia aplicación original la que tiene la posibilidad de crear ficheros *\*.html*. *Internet Assistant for Word*, *Excel* y *Powerpoint* son conversores integrados en los programas de *Microsoft Office*.

# Formularios y CGIs

## 4.1. Formularios (Forms)

Hasta el momento se ha visto cómo el servidor puede crear páginas Web que son visualizadas por medio de un browser en el ordenador del usuario remoto, y cómo se mantiene cierta interactividad mediante el carácter de *hipertexto*, mediante el cual el usuario va solicitando distintos contenidos según sus preferencias. Sin embargo, en lo visto hasta ahora se echa en falta la posibilidad de que el usuario envíe datos al servidor, tales como sugerencias, nombres, datos personales, etc. Esta posibilidad se consigue por medio de los *formularios* (FORMs) que permiten el envío de datos mediante diversos tipos de botones, cuadros de diálogo y ventanas de texto. En este apartado los formularios y sus distintos elementos se estudiarán con un cierto detenimiento. En la Figura 4.1. se presenta un *formulario* con distintos *elementos*: cajas de texto, botones de selección o de opción, listas desplegadas, etc.



The image shows a web form titled "Formularios". It contains the following elements: a text input field labeled "Nombre y apellido:"; a radio button labeled "Botón de radio"; a checkbox labeled "Botón de selección"; a vertical list box with three items: "Elemento1", "Elemento2", and "Elemento3"; a horizontal list box with five items: "Elemento1", "Elemento2", "Elemento3", "Elemento4", and "Elemento5"; and two buttons at the bottom right labeled "Enviar" and "Reinicio".

Figura 4.1.

Los formularios tienen un modo propio de funcionar que conviene entender correctamente, primero de un modo más general y luego más en detalle. Cada uno de los *elementos* del formulario tiene un *nombre*, determinado por la persona que lo ha programado. Ese nombre se asociará posteriormente a la opción elegida o al texto introducido por el usuario. Un elemento particularmente importante (por lo decisivo, más que por su complejidad) es el botón *Enviar* (o *Submit*, *Send*, *O.K.*, o como se le haya querido llamar), cuya misión es dar por concluida la recogida de datos en el browser y enviar al servidor toda la información que el usuario haya introducido. Cuando se pulsa este botón se envía al servidor (en la forma que más adelante se verá) una larga cadena de caracteres que contiene los nombres de todos los elementos del formulario junto con la información que el usuario ha introducido en cada uno de ellos.

¿Qué hace el servidor con toda la información que le llega de un formulario debidamente cumplimentado? Pues algo muy sencillo: arrancar un programa cuyo nombre está especificado en el propio formulario, pasarle toda la información que ha recibido, recoger la información que dicho programa le envía como respuesta, y enviarla al cliente o browser que rellenó el formulario. Además de responder al cliente, este programa podrá realizar cualquier otra acción, como por ejemplo guardar la información recibida en un fichero o en una base de datos. Ya se ve que en todo este proceso el servidor actúa como una especie de intermediario entre el browser y el programa que recibe la información del formulario y la procesa. La forma en la que el servidor se entiende con el programa al que llama para que procese los datos del formulario recibe el nombre de CGI (*Common Gateway Interface*). La Figura 4.2. representa de forma simbólica todo este proceso.

Así pues todo formulario se define mediante la doble TAG `<FORM>...</FORM>`, dentro de las cuáles se definirán los distintos elementos. Al crear un formulario se debe definir el *método de envío de datos* (GET o POST) que el servidor utilizará para enviar al programa CGI los datos que debe procesar. También se debe especificar la *acción* que el servidor deberá emprender cuando reciba los datos, y que será

arrancar dicho programa CGI cuyo nombre debe conocer. Estas dos tareas se definen, respectivamente, con los atributos METHOD y ACTION.

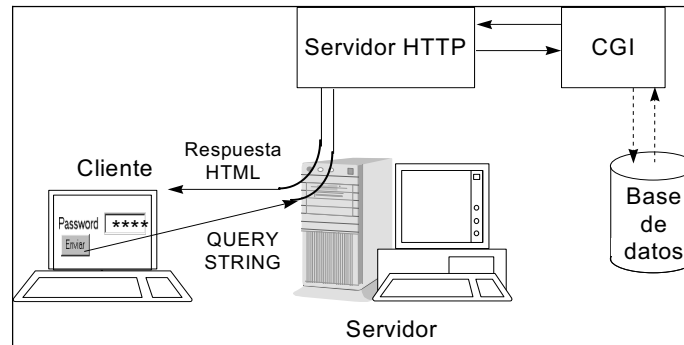


Figura 4.2.

A continuación se presenta un ejemplo simple que permite ver cómo se define un formulario y cómo aparece en el browser. El código HTML es el siguiente:

```
<FORM ACTION="/cgi-bin/form1.exe" METHOD="GET">
  Introduzca su nombre:
  <INPUT TYPE="text" NAME="nombre"><br>
  <INPUT TYPE="SUBMIT" VALUE="Enviar">
</FORM>
```

En la Figura 4.3. aparece representado el formulario correspondiente, que sólo tiene dos elementos: un *cuadro de texto* para que el usuario teclee el nombre y un *botón* para enviar los datos al servidor cuando el nombre haya sido ya introducido. Todos los elementos de un formulario que recogen información se definen con la TAG simple `<INPUT>`. El atributo TYPE define el tipo de elemento de que se trata (si se omite, por defecto se supone que el tipo es *text*). A cada elemento (excepto a los singulares -que no se pueden repetir- como el botón *Enviar*) hay que añadirle un atributo NAME que defina el nombre con el que se identificará luego su contenido.

La imagen muestra una interfaz de usuario de un navegador web. En la parte superior hay una barra de navegación con los botones 'What's New?', 'What's Cool?', 'Destinations' y 'Net Search'. Debajo de esta barra, se encuentra un formulario con el texto 'Introduzca su nombre:' seguido de un cuadro de texto vacío. Debajo del cuadro de texto hay un botón etiquetado como 'Enviar'.

Figura 4.3.

En el ejemplo anterior, el cuadro de texto se ha definido con la TAG INPUT; en esa TAG se ha introducido un parámetro NAME definiendo que ese cuadro se llama "*nombre*".

Se ha introducido el atributo TYPE="SUBMIT" para el segundo elemento del formulario. Este elemento es un botón cuya función es enviar los datos de los elementos del formulario al programa CGI que lo va a procesar. Dicho programa se especifica mediante el atributo ACTION, y resulta ser "`/cgi-bin/form1.exe`". En general *cgi-bin* es un directorio localizado en el servidor, que contiene los programas CGI.

También se puede crear un botón (llamado en inglés RESET) que reinicie el formulario a su estado inicial, anulando todos los cambios que hubiera podido introducir el usuario. La manera de crearlo se verá en el siguiente ejemplo, que contiene algunos elementos y atributos nuevos con los que se trata de recoger los datos del domicilio de una persona. La Figura 4.4. muestra como se ve en el browser el formulario definido. Gracias a la TAG `<PRE>...</PRE>` que impone paso constante y respeta los espacios en blanco se consigue que los cuadros de texto queden alineados. El atributo NAME permite definir un nombre para cada elemento. El atributo SIZE determina la longitud del cuadro de texto, mientras que MAXLENGTH limita el número de caracteres que se pueden introducir en cada cuadro. Inicialmente todos los cuadros de texto están en blanco, tal como se ve en la figura 4.4. Si el usuario introduce datos y pulsa luego el botón *Borrar*, el formulario volverá a la condición inicial. Los botones SUBMIT y BORRAR no necesitan NAME porque quedan identificados por su atributo TYPE. Por su parte los cuadros de texto no necesitan atributo TYPE porque el tipo por defecto es TEXT.

```
<FORM ACTION="/cgi-bin/form3" METHOD="POST">
  <PRE>
  Nombre:      <INPUT NAME="nombre" SIZE=30>
  Calle:       <INPUT NAME="calle" SIZE=40>
  Localidad:   <INPUT NAME="localid" SIZE=35>
  Provincia:   <INPUT NAME="prov" SIZE=25>
  Cód. Postal: <INPUT NAME="cp" SIZE=5 MAXLENGTH=5>
  </PRE>
  <INPUT TYPE="SUBMIT" VALUE="Enviar">
  <INPUT TYPE="RESET" VALUE="Borrar">
</FORM>
```

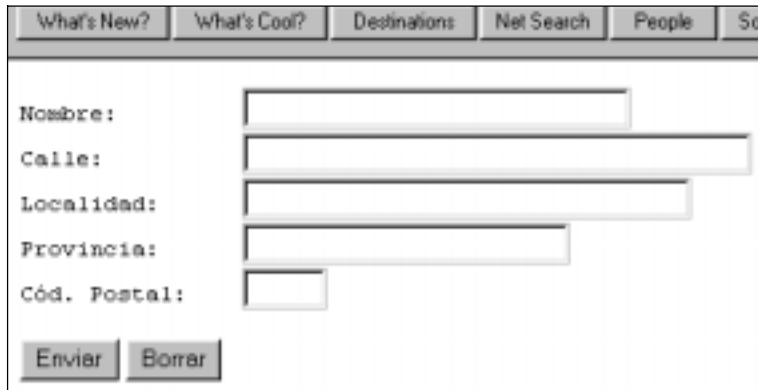


Figura 4.4.

Considérese un nuevo ejemplo correspondiente al formulario que se muestra en la Figura 4.5. En él se ve que también se pueden crear **áreas de texto** en las cuales no se limita la cantidad de texto a introducir, y que pueden servir para pedir sugerencias u opiniones. La forma de crearlas es con la TAG doble <TEXTAREA>...</TEXTAREA>. Los parámetros que se le envían son el nombre (atributo NAME) y los números de filas y de columnas de la ventana (atributos ROWS=n y COLS=m). En este caso la TAG doble <TEXTAREA> sustituye a la TAG simple <INPUT>. Más adelante se verán otros TAGs alternativas para crear distintos elementos del formulario. A continuación se muestra un ejemplo (Figura 4.5).

```
<H2>Sugerencias y críticas</H2>
<P><FORM ACTION="/cgi-bin/form3" METHOD="POST">
  Por favor, introduzca cualquier sugerencia o crítica positiva:<BR>
  <TEXTAREA ROWS=5 COLS=30 NAME="positivo">Me encanta este Web porque...
  </TEXTAREA></P>
  <P>Introduzca los comentarios negativos:<BR>
  <TEXTAREA ROWS=2 COLS=15 NAME="negativo"></TEXTAREA></P>
  <P><INPUT TYPE="SUBMIT" VALUE="Enviar">
</FORM></P>
```

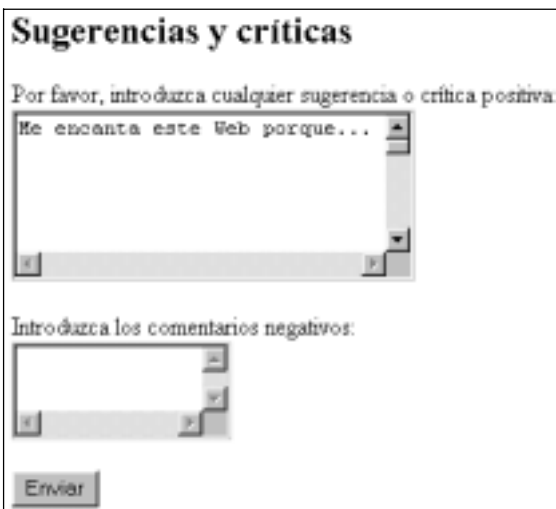


Figura 4.5.

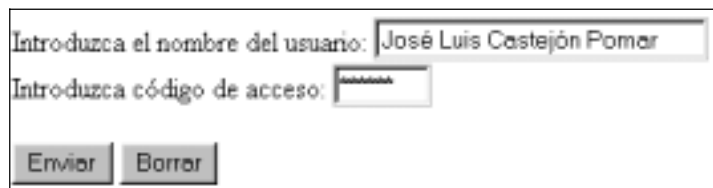


Figura 4.6.

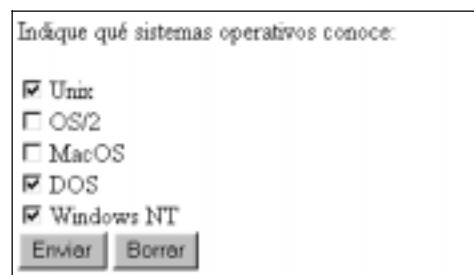


Figura 4.7.

En el siguiente ejemplo se va a ver que es posible crear cuadros de texto en los que lo que se escribe no aparezca (que aparezca como caracteres \*). Estos elementos se emplean para que el usuario introduzca un **password**. En la Figura 4.6. se muestra el resultado del código HTML que sigue:

```
<FORM>
  Introduzca el nombre del usuario: <INPUT NAME="login" SIZE=25><BR>
  Introduzca código de acceso:
  <INPUT TYPE="PASSWORD" NAME="acceso" SIZE=6 MAXLENGTH=10><P>
  <INPUT TYPE="SUBMIT" VALUE="Enviar">
  <INPUT TYPE="RESET" VALUE="Borrar">
</FORM>
```

Como se puede ver, el modo de crear este tipo de cuadros de texto es análogo a los anteriores: únicamente hay que introducir el atributo `TYPE="password"`.

Una de las opciones más interesantes de los formularios es la de poder crear **casillas de verificación**. En la Figura 4.7 aparece un ejemplo de este tipo de elemento. Para poder crear estas casillas se debe utilizar la TAG `INPUT`, poniendo el atributo `TYPE="CHECKBOX"` y el mismo `NAME` para todas las casillas del grupo. La característica de estas casillas es que se pueden seleccionar varias a la vez. Si al definir las se introduce además el atributo `CHECKED`, aparecerán inicialmente seleccionadas (Pese a que luego puedan no seleccionarse). El código HTML necesario para crear este formulario es el que sigue a continuación:

```
<FORM ACTION="/cgi-bin/form" METHOD="POST">
  Indique qué sistemas operativos conoce:<P>
  <INPUT TYPE="checkbox" NAME="sisoper" VALUE="Unix" CHECKED>Unix<BR>
  <INPUT TYPE="checkbox" NAME="sisoper" VALUE="OS/2">OS/2<BR>
  <INPUT TYPE="checkbox" NAME="sisoper" VALUE="MacOS">MacOS<BR>
  <INPUT TYPE="checkbox" NAME="sisoper" VALUE="DOS" CHECKED>DOS<BR>
  <INPUT TYPE="checkbox" NAME="sisoper" VALUE="WinNT" CHECKED>Windows NT<BR>
  <INPUT TYPE="SUBMIT" VALUE="Enviar">
  <INPUT TYPE="RESET" VALUE="Borrar">
</FORM>
```

En el siguiente ejemplo se va a ver que también se pueden crear **botones de radio**, que son similares a los `CHECKBOX`, pero con la condición de que sólo se puede seleccionar uno de ellos a la vez. La Figura 4.8. muestra el resultado en el browser del código que se incluye a continuación:

```
<FORM ACTION="/cgi-bin/form" METHOD="POST">
  Indique su estado civil:<P>
  <INPUT TYPE="radio" NAME="estado" VALUE="Soltero">Soltero/a <BR>
  <INPUT TYPE="radio" NAME="estado" VALUE="Casado">Casado/a <BR>
  <INPUT TYPE="radio" NAME="estado" VALUE="Viudo">Viudo/a <BR>
  <INPUT TYPE="radio" NAME="estado" VALUE="Otros">Otros <BR>
  <INPUT TYPE="SUBMIT" VALUE="Enviar">
  <INPUT TYPE="RESET" VALUE="Borrar">
</FORM>
```

Finalmente se presentan ejemplos de otra de las opciones que permiten los formularios: las **ventanas de selección desplegables** y las **ventanas de scroll** (o ventanas con barras de deslizamiento). Las primeras están basadas en una TAG doble llamado `<SELECT>...</SELECT>`. En el contenido de esta TAG se pueden incluir tantos TAGs simples `<OPTION>` como se desee. El TAG `<SELECT>` tiene el atributo `NAME`, lo que no ocurre con las TAGs `<OPTION>`. Cada TAG `<OPTION>` contiene un atributo `VALUE` que describe el texto que aparecerá en la ventana.

Figura 4.8.

Figura 4.9.



El aspecto visual del primero de los ejemplos se muestra en la Figura 4.9 y su código HTML:

```
<FORM>
  Indique cuál es el medio de<BR>
  transporte que usa con mayor<BR>
  frecuencia para ir al trabajo:<P>
  <SELECT NAME="Transp">
    <OPTION VALUE="coche"> Coche particular </OPTION>
    <OPTION VALUE="bus"> Autobús </OPTION>
    <OPTION VALUE="taxi"> Taxi </OPTION>
    <OPTION VALUE="tren"> Tren </OPTION>
    <OPTION VALUE="metro"> Metro </OPTION>
    <OPTION VALUE="bici"> Bicicleta </OPTION>
  </SELECT>
  <INPUT TYPE="SUBMIT" VALUE="Enviar">
</FORM>
```

Como se puede ver cada una de las opciones de la ventana se introduce con la TAG simple <OPTION VALUE="nombre">, y es necesario que todas las opciones estén entre las TAGs <SELECT> y </SELECT>

En el siguiente ejemplo se va a utilizar, en vez de una *ventana desplegable*, una ventana con barras de deslizamiento o *ventana de scroll*. En la Figura 4.10 se muestra el resultado del código que sigue:

```
<FORM>
  Indique cuál es el medio de<BR>
  transporte que usa con mayor<BR>
  frecuencia para ir al trabajo:<P>
  <SELECT NAME="Transp" SIZE=3>
    <OPTION VALUE="coche"> Coche particular </OPTION>
    <OPTION VALUE="bus"> Autobús </OPTION>
    <OPTION VALUE="taxi"> Taxi </OPTION>
    <OPTION VALUE="tren"> Tren </OPTION>
    <OPTION VALUE="metro"> Metro </OPTION>
    <OPTION VALUE="bici"> Bicicleta </OPTION>
  <SELECT>
  <INPUT TYPE="SUBMIT" VALUE="Enviar">
</FORM>
```

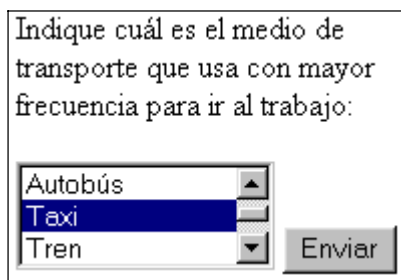


Figura 4.10.

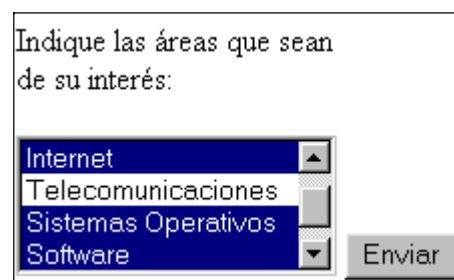


Figura 4.11.

La única diferencia respecto al ejemplo anterior es el atributo SIZE que define el tamaño vertical (número de líneas) de la ventana. Si SIZE=1 la ventana será desplegable, y en caso contrario aparecerá una ventana con barras de deslizamiento.

En los dos ejemplos anteriores sólo se podía seleccionar uno de las opciones de la ventana. En el siguiente ejemplo se va a crear una *ventana de selección múltiple*. El resultado se muestra en la Figura 4.11. y el código HTML es como sigue:

```

<FORM>
  Indique las áreas que sean<BR>
  de su interés:<P>
  <SELECT NAME="Areas" SIZE=4 MULTIPLE>
    <OPTION VALUE="lan"> Redes de Área Local
    <OPTION VALUE="inet"> Internet
    <OPTION VALUE="teleco"> Telecomunicaciones
    <OPTION VALUE="sop"> Sistemas Operativos
    <OPTION VALUE="soft"> Software
    <OPTION VALUE="hard"> Hardware
  </SELECT>
  <INPUT TYPE="SUBMIT" VALUE="Enviar">
</FORM>

```

Ni que decir tiene que también es posible crear ventanas desplegadas de selección múltiple: pruébese copiando éste mismo código eliminando el parámetro SIZE=4 o, simplemente, haciendo SIZE= 1) En los ejemplos anteriores se han presentado las posibilidades abiertas por los formularios. En el apartado siguiente se considerará cómo se debe procesar esta información cuando llega al servidor.

## 4.2. Programas CGI

Ya se ha visto en el apartado anterior cómo crear formularios con los que recoger datos de diverso tipo para enviarlos al servidor, pero ¿cómo se envían y cómo se procesan estos datos? La respuesta está en que los datos se envían como cadena de caracteres añadida a un URL como *query string*, precedida con un carácter (?); para procesarlos se utilizan los programas CGI (*Common Gateway Interface*), que el servidor arranca y a los que pasa los datos recibidos del browser.

Los CGI son programas hechos por lo general en *Perl* (lenguaje interpretado muy popular en el entorno UNIX, pero que también existe en *Windows NT*) o en *C/C++*.

A grandes rasgos un programa CGI debe realizar las siguientes tareas:

- Recibir los datos que son enviados por el browser e interpretarlos, separando la entrada correspondiente a cada elemento del formulario.
- Realizar las tareas necesarias para procesar esos datos, que de ordinario consistirán en almacenarlos en algún archivo o base de datos y enviar una respuesta al browser donde se encuentra el usuario. Esta respuesta debe tener la forma de página HTML incluyendo todos los elementos necesarios (<HEAD>, <BODY>, etc.).

Para que un programa CGI funcione correctamente debe estar en un directorio /cgi-bin/ del servidor HTTP (servidor Web). En caso que se utilice Visual C++ (u otro compilador compatible) es importante, para que el ejecutable CGI funcione correctamente, el compilarlo para que funcione en modo MS-DOS (o modo Consola). Para un programa de este tipo la *entrada de datos estándar* es el teclado y la *salida de datos estándar* es la pantalla. Para un programa CGI el servidor HTTP asume el papel de entrada y salida de datos estándar.