XML

Introducción

XML es el acrónimo de *Extensible Markup Language* (Lenguaje extensible de marcas) un estándar desarrollado por el *World Wide Web Consortium* (W3C). XML es un subconjunto del *Standard Generalized Markup Language*[1] (SGML). Las simplificaciones no redundan en perjuicio de la extensibilidad del mismo. Solamente se trata de facilitar la tarea de desarrollar archivos XML válidos.

¿Qué diferencias existen entre XML y HTML? En HTML, tanto la semántica de las etiquetas como el conjunto de etiquetas está fijado. La etiqueta `<h1>` siempre fija un encabezado de primer nivel, mientras que la etiqueta `<cua>` no tiene sentido porque no se encuentra definida. El W3C junto con los fabricantes de navegadores y otros miembros de la WWW, están constantemente trabajando en ampliar la definición de HTML para incorporar nuevas etiquetas y atributos.

XML no especifica ni la semántica ni el conjunto de etiquetas. En realidad, XML es un metalenguaje: un lenguaje para describir (definir) lenguajes. XML proporciona una serie de herramientas que permiten definir etiquetas y las relaciones estructurales que guardan las etiquetas entre sí.

[1] SGML es un sistema para organizar y etiquetar los elementos de un documento. SGML fue desarrollado y estandarizado por la *International Organization for Standards* (ISO) en 1986 con el número ISO 8879. A su vez, éste se basa en el *Generalized Markup Language* (GML), inventado por IBM en 1969. SGML no especifica ningún tipo de formateo concreto: más bien, especifica las reglas que deben de cumplir las etiquetas. SGML se usa extensamente para manejar grandes documentos que sufren frecuentes revisiones y necesitan imprimirse con diferentes formatos. Debido a que se trata de un sistema amplio y complejo, no se usa mucho en los ordenadores personales. Sin embargo, debido a que HTML define e interpreta las etiquetas de acuerdo a SGML, existe un renovado interés hacia este sistema.

Ventajas

Las principales ventajas que ofrece XML son:

- Mejora la precisión de las búsquedas, ya que cuando se utilizan metadatos[2] la efectividad de los motores de búsqueda se incrementa.
- Facilita el intercambio de información entre distintas aplicaciones ya que se basa en estándares aceptados.
- Proporciona una visión estructurada de la información, lo que permite su posterior tratamiento de forma local.
- Permite integrar información procedente de diferentes fuentes.
- Permite actualizaciones granulares de la información.

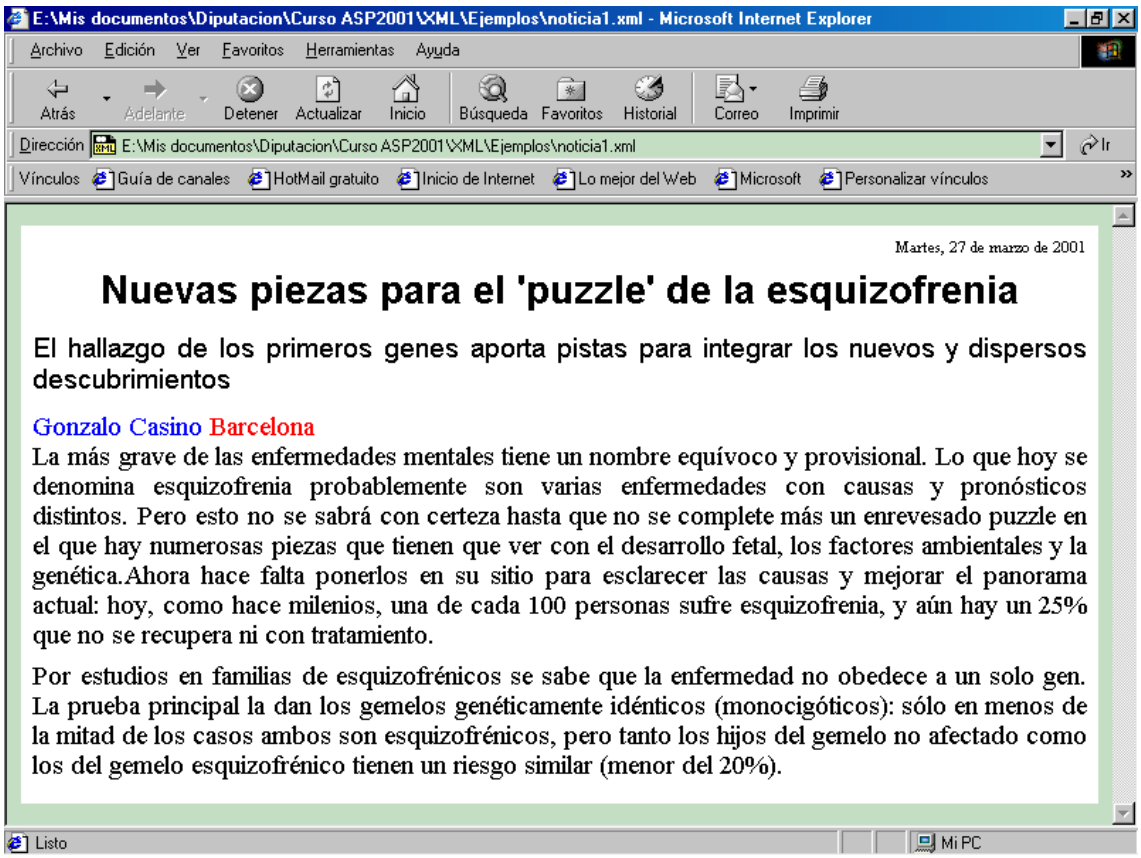

Figura 1

Sin embargo, la principal ventaja de XML es que mantiene una separación entre los datos y su presentación. Esta característica facilita el mantenimiento de la información y permite ofrecer múltiples puntos de vista de una misma información. Por

[2] Información sobre los datos.

ejemplo, la Figura 1 y la Figura 2 muestran en un navegador el mismo documento XML pero con distinto aspecto visual, ya que en cada caso se emplea una plantilla de presentación distinta.

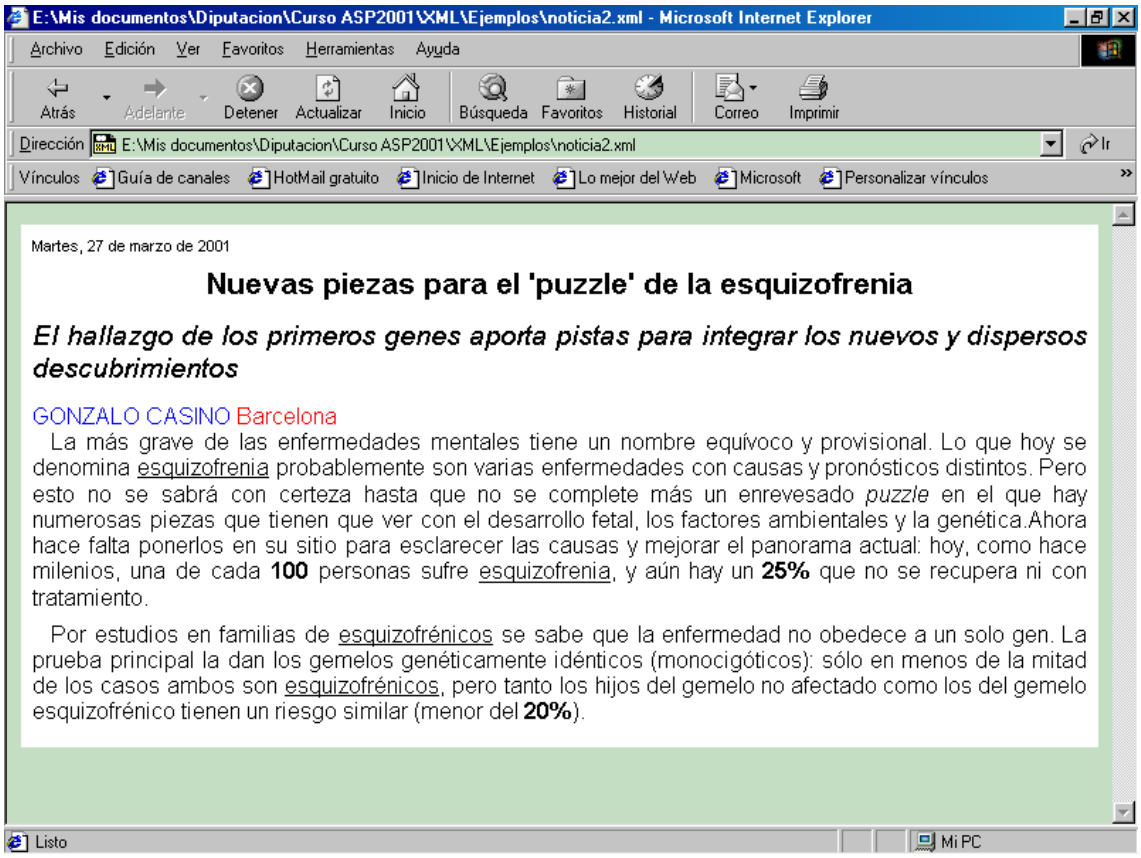

Figura 2

XHTML

Extensible Hypertext Markup Language (XHTML) es un híbrido entre HTML y XML diseñado específicamente para ser usado en Internet (se supone que será el sustituto de HTML). XHTML es un lenguaje de marcas escrito en XML: es una aplicación concreta de XML. Por lo tanto, no se deben confundir XML y XHTML.

Objetivos de diseño

Al inicio del proceso que condujo al desarrollo de XML, el W3C decidió definir una serie de objetivos (principios o metas) que guiasen el proceso y asegurasen que la

especificación final obtenida cubriese las necesidades que originaron el desarrollo de un nuevo lenguaje. En total se definieron diez objetivos:

1. XML debe de poderse usar directamente en Internet. XML ha adoptado muchas de las convenciones de HTML para facilitar el cambio hacia XML.
2. XML debe de soportar una amplia variedad de aplicaciones. XML permite describir una amplia variedad de tipos de información, desde altamente estructurada hasta semiestructurada, desde textual hasta multimedia.
3. XML debe de ser compatible con SGML. XML se define como un subconjunto de SGML, de modo que se puede trabajar con las herramientas para SGML.
4. Debe de ser sencillo escribir programas que procesen documentos escritos en XML. Un procesador de XML es más sencillo de desarrollar que una herramienta similar para SGML, ya que XML posee menos opciones y variaciones.
5. El número de características opcionales en XML debe de ser el mínimo posible, idealmente cero. Por una parte, casi todas las características de XML son opcionales. Pero por otra, raramente hay dos o más formas de conseguir un mismo resultado.
6. Los documentos basados en XML deben de poder leerse por un humano y poseer una estructura clara. XML se basa en SGML, un lenguaje de etiquetado que emplea textos en formato ASCII y que puede ser leído por humanos.
7. El diseño de XML debe de ser rápido. El desarrollo de XML comenzó en el verano de 1996 y la versión final fue publicada en febrero de 1998.
8. El diseño de XML debe de ser formal y conciso. XML se especifica mediante un conjunto de reglas que definen una gramática.
9. Los documentos basados en XML deben de ser fáciles de crear. Con un simple editor de textos ASCII se puede crear un documento basado en XML.
10. La concisión en las etiquetas de XML es de mínima importancia. SGML incluye muchas características opcionales que permiten que se omitan etiquetas o partes de etiquetas cuando el contexto permita inferir la presencia de las etiquetas que faltan. Incluso en HTML es posible omitir algunas etiquetas finales, como `</p>` u `</option>`, sin que se introduzca ambigüedad. En XML, no es posible esto.

Qué necesito para usar XML

No es necesario un servidor web, un proveedor web o tener una conexión a Internet para empezar a escribir documentos XML. Todo lo que se necesita es un editor (como el *Bloc de notas* de Windows) para los archivos y un navegador para verlos. Podemos crear, vincular y probar documentos XML completos en nuestro ordenador, aunque no esté conectado a alguna red. Aunque se asocia XML con Internet, existen muchas aplicaciones de XML que no tienen nada que ver con Internet.

No obstante, es cierto que a medida que los documentos crecen y los proyectos se hacen más sofisticados, la utilización de herramientas específicas contribuyen a facilitar las labores de edición y mantenimiento. Las herramientas que vamos a emplear son:

- Microsoft Internet Explorer 5.x o superior con IE XML/XSL Viewer Tools[3]. Es gratuito.
- Microsoft XML Notepad Beta 1.5: sencillo editor de documentos XML. Representa gráficamente la estructura arborescente de un documento. No permite validar documentos. Es gratuito.
- ezDTD 1.5[4]: permite escribir DTD (tanto para XML como para SGML) y convertirlo a HTML con enlaces internos. Es gratuito.
- Altova XML Spy[5] (múltiples versiones): entorno de desarrollo integrado para XML. Permite editar y validar documentos XML, editar y validar DTDs y esquemas de XML y editar XSL y transformar documentos. Es un programa de pago, pero dispone de versión de demostración.

Microsoft Internet Explorer 5.x y Netscape 6.x soportan XML, pero no completamente, ya que para mostrar los documentos XML se basan en CSS y no en XSL.

---

[3] Actualización que se puede descargar de la web de Microsoft. Actualiza el navegador con dos nuevas opciones: *Validate XML* y *View XSL Output*.

[4] Disponible en `http://www.geocities.com/SiliconValley/Haven/2638/-ezDTD.htm`.

[5] Disponible en `http://www.xmlspy.com/`.

La extensión de un archivo XML suele ser `.xml`. Se deben emplear nombres cortos y sencillos. Hay que evitar el uso de espacios o de caracteres especiales en el nombre del archivo y también controlar el uso de mayúsculas y minúsculas puesto que en Internet existen multitud de sistemas operativos, que no pueden aceptar los mismos nombres de archivo que acepta el nuestro. Por ejemplo, hay sistemas operativos en los que las mayúsculas y minúsculas se distinguen y otros donde no.

Estructura lógica de un documento XML

Una de las principales características de XML es que permite crear una plantilla para las etiquetas de un documento. De este modo, la posición de las etiquetas y sus atributos se pueden controlar y se puede verificar la corrección de un documento.

La estructura de un documento se define mediante un *Document Type Definition* (DTD). Un DTD es un conjunto de reglas que definen la estructura de un documento. Se trata de una característica opcional, ya que para crear un documento XML no es necesario emplear un DTD.

Un DTD establece la estructura formal de un documento: define los elementos que se pueden emplear e indica dónde se pueden usar en relación con los demás. Por tanto, especifica la jerarquía y la granularidad del documento.

La jerarquía (organización) describe las relaciones que existen entre los distintos elementos. La jerarquía se suele representar gráficamente en forma de árboles. Por ejemplo, un Libro se compone de Capítulos; un Capítulo se compone de Secciones; una Sección se compone de Párrafos.

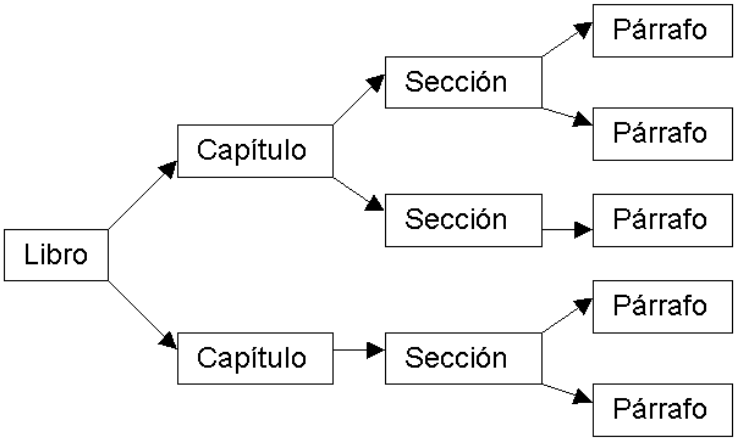

Figura 3

La granularidad indica el grado de división de un elemento en sus partes más pequeñas. Una jerarquía compleja implica una granularidad fina. Estructuras simples con pocos niveles indican una granularidad gruesa. Por ejemplo, un elemento Nombre puede contener directamente el nombre completo de un persona (granularidad gruesa) o puede estar dividido en partes más pequeñas que separan el nombre de los apellidos (granularidad fina).

Para qué sirve un DTD

Algunos procesadores de XML son capaces de leer un DTD y usarlo para construir en memoria el modelo del documento. Este modelo se compara con el contenido de un documento concreto y se pueden localizar los posibles errores que contenga el documento. Los procesadores de este tipo se dice que poseen la capacidad de **validación** e incluyen un modulo de **análisis/validación**.

Por ejemplo, si un DTD afirma que un título debe de aparecer al principio de cada capítulo, pero en un capítulo no aparece, al validar el documento se producirá un error.

Documento "bien formado" y documento válido

Un documento XML es **válido** si su contenido cumple las reglas de su DTD asociado. Esta cualidad –la validez de un documento XML– sirve para que las aplicaciones puedan estar seguras de que los datos XML están completos, correctamente formateados y además los valores de sus atributos son los adecuados.

Cuando un documento XML no hace referencia a ningún DTD no podemos decir nada acerca de su validez. En esos casos, el analizador solamente puede informarnos acerca de si el documento está **"bien formado"** (es decir, respeta la sintaxis del lenguaje XML).

Mientras que todos los documentos XML han de estar bien formados, no tienen por qué ser válidos. Por definición, si un documento no está bien formado, entonces no es un documento XML.

Estructura de un documento XML

Vamos a ver de qué se compone un documento XML y cuáles son las reglas que rigen su construcción.

Etiquetas

En XML, es el usuario el que se crea sus propias etiquetas (o usa las que ha creado otra persona)[6]. Todas las etiquetas comienzan con el símbolo < (menor que) y terminan con el símbolo > (mayor que). Entre estos dos símbolos aparece el nombre de la etiqueta y sus atributos. Por ejemplo, `<LIBRO>` es una etiqueta válida, pero `<LIBRO` o `<LIBRO<` no lo son.

Todas las etiquetas iniciales deben tener su correspondiente etiqueta final. Dentro de XML, un par que etiquetas, llamadas **inicial** y **final**, se denomina un **elemento**. La etiqueta final tiene el mismo nombre que la inicial, pero va precedida del símbolo / (barra inclinada). Por ejemplo, la etiqueta final correspondiente a `<LIBRO>` es `</LIBRO>`.

Etiquetas vacías

XML permite incluir etiquetas vacías, es decir, que son en sí mismas inicial y final y no contienen nada. Estas etiquetas se identifican por terminar con el carácter /. Por ejemplo, la siguiente etiqueta es vacía:

[6] En HTML se emplean las etiquetas establecidas en el estándar correspondiente.

```
<ALUMNO NOMBRE=”Sergio” APELLIDOS=”Luján Mora” />
```

El lenguaje HTML también permite este tipo de etiquetas, pero no es necesaria la aparición del carácter / al final de las etiquetas. Un ejemplo de esto son las etiquetas de salto de línea (<BR>), línea horizontal (<HR>) o inserción de una imagen (<IMG>):

```
<HR>Aquí viene una imagen <IMG SRC=”imagen.gif”><HR>
```

Estructura jerárquica de los elementos

Los documentos XML deben tener una estructura de etiquetas jerarquizada: cualquier elemento debe estar adecuadamente anidado dentro de otro. Por ejemplo, la siguiente tabla muestra dos documentos XML: uno correcto y el otro incorrecto. XML no permite que dos elementos estén entremezclados: o el elemento LIBRO está contenido dentro del elemento CAPITULO o al revés, pero nunca la situación que se muestra como incorrecta.

CORRECTO	INCORRECTO
<LIBRO>	<LIBRO>
<CAPITULO>	<CAPITULO>
</CAPITULO>	</LIBRO>
</LIBRO>	</CAPITULO>

El lenguaje HTML no es tan estricto a la hora de considerar que un documento esté bien estructurado, ya que se pueden dar situaciones como la siguiente:

```
<FONT COLOR=”red”>Hola a <B><I>todos</B></FONT></I>
```

Elemento raíz único

Los documentos XML sólo permiten un **elemento raíz** (también llamado **elemento documento**): todas las etiquetas tienen que estar englobadas dentro de una, que es exterior a todas ellas[7]. Esta restricción sirve para verificar que el documento está completo. Por ejemplo, si tenemos información sobre varios libros:

```
<LIBRO>
```

[7] En HTML la raíz suele ser la etiqueta <HTML>: dentro de ella tienen que aparecer el resto de etiquetas. Los navegadores actuales son muy permisivos y no obligan a seguir esta regla.

```
  <TITULO>Los secretos de XML</TITULO>
  <AUTOR>Sergio Luján Mora</AUTOR>
</LIBRO>
<LIBRO>
  <TITULO>XML en 35 segundos</TITULO>
  <AUTOR>Sergio Luján Mora</AUTOR>
  <AUTOR>Marisa Zayas Fornieles </AUTOR>
</LIBRO>
```

la sintaxis de XML nos obliga a agrupar todos los elementos (etiquetas) dentro de un único elemento:

```
<BIBLIOTECA>
<LIBRO>
  <TITULO>Los secretos de XML</TITULO>
  <AUTOR>Sergio Luján Mora</AUTOR>
</LIBRO>
<LIBRO>
  <TITULO>XML en 35 segundos</TITULO>
  <AUTOR>Sergio Luján Mora</AUTOR>
  <AUTOR>Marisa Zayas Fornieles </AUTOR>
</LIBRO>
</BIBLIOTECA>
```

Distinción entre mayúsculas y minúsculas

Las etiquetas XML distinguen entre mayúsculas y minúsculas[8]. Por este motivo, resulta recomendable escribir las etiquetas XML desde el principio todas en mayúsculas o en minúsculas.

Atributos

Las etiquetas XML pueden tener atributos definidos por el usuario. Un atributo proporciona información adicional sobre el elemento que acompaña. Un atributo está

[8] En HTML no se distinguen las mayúsculas y minúsculas. Por ejemplo, `<br>`, `<Br>` y `<BR>` producen el mismo efecto.

formado por un nombre y un valor[9]. El nombre precede a su valor y ambos se separan por el signo igual. Los valores de los atributos deben escribirse siempre entre comillas (simples o dobles)[10], porque puede contener espacios en blanco y de otra forma sería imposible detectar el final de un valor y el inicio de otro atributo.

Tiene que existir al menos un espacio en blanco entre el nombre del elemento y el primer atributo, y también pueden aparecer espacios en blanco alrededor del signo igual. Por ejemplo:

```
<LIBRO TITULO = "XML"    AUTOR    = "Sergio Luján Mora" />
```

La posibilidad de emplear comillas simples o dobles permite encerrar un valor que contiene comillas del otro tipo. Por ejemplo:

```
<LIBRO TITULO="El futuro de Internet: 'XML'" />
```

Si ambos tipos de comilla se tienen que emplear en un valor, entonces es necesario codificar una de las comillas: `&apos;` para la comilla simple (apóstrofe) y `&quot;` para la comilla doble.

Atributos reservados

Sólo hay dos atributos reservados en el estándar de XML. Para evitar conflictos con los atributos definidos por el usuario, el prefijo `xml:` se emplea en el nombre del atributo.

xml:lang

Algunas veces, puede ser interesante identificar el idioma usado en el texto contenido en un elemento concreto. El atributo xml:lang permite almacenar tanto el idioma como el país (si hace falta, porque el mismo idioma cambia entre distintos países). Los valores que puede tomar este atributo vienen recogidos en el estándar RFC 1766[11]. Por ejemplo:

[9] En HTML existen atributos que no tienen valor. Por ejemplo, `<FRAME NORESIZE>` o `<HR NOSHADE>`.

[10] El lenguaje HTML permite atributos sin entrecomillar. Por ejemplo, `<FONT SIZE=5>`.

[11] *Tags for the Identification of Languages*, Marzo 1995.

```
<parrafo xml:lang=”en”>This text is in English</parrafo>
<parrafo xml:lang=”es”>Este texto está en Español</parrafo>
```

xml:space

Normalmente, los espacios en blanco (ver apartado siguiente) no se tienen en cuenta (espacios en blanco no significativos). Sin embargo, si en algún caso se desea que se conserven los espacios en blanco (espacios en blanco significativos), se puede indicar con el atributo `xml:space`, que puede tomar dos valores: `default` (el valor por defecto si no aparece este atributo) y `preserve` (conserva espacios en blanco).

Espacios en blanco

El término **espacio en blanco** comprende una serie de caracteres que no tienen apariencia visual, pero que afectan de alguna forma al formato de un documento. Los espacios en blanco aceptados por XML es cualquier combinación de los siguientes caracteres:

Nombre		Código ASCII
Tabulador	Tab	9
Avance de línea	Line Feed (LF)	10
Retorno de carro	Carriage Return (CR)	13
Espacio en blanco	Space	32

Los espacios en blanco dentro de las etiquetas, entre los atributos, son irrelevantes. Por ejemplo, entre estas dos etiquetas no hay ninguna diferencia:

```
<ALUMNO NOMBRE=”Sergio”   APELLIDOS=”Luján Mora” />
<ALUMNO
NOMBRE=”Sergio”
APELLIDOS  =          ”Luján Mora” />
```

Dentro de las propias etiquetas, en los valores de los atributos, los espacios en blanco se eliminan o normalizan: se eliminan los espacios en blanco iniciales y finales. Esto es muy importante tenerlo en cuenta, ya que puede inducir fácilmente a errores. Por ejemplo, estas dos etiquetas poseen los mismos valores en sus atributos:

```
<PERSONA NOMBRE=”Sergio“ APELLIDOS=”Luján Mora“ />
<PERSONA NOMBRE=”Sergio   “  APELLIDOS=”  Luján    Mora   “ />
```

Normalización de los caracteres de final de línea

El estándar ASCII incluye dos caracteres que se pueden interpretar como final de línea: *Carriage Return* (CR) y *Line Feed* (LF). Entre distintos sistemas operativos existen diferencias a la hora de su uso:

Sistema operativo	Final de línea
Macintosh	CR
MS-DOS / Windows	CR y LF
Unix	LF

Cuando se procesa un documento XML, los caracteres de final de línea se normalizan del siguiente modo:

- Si se encuentra LF, se deja.
- Si se encuentra CR, se convierte en LF.
- Si se encuentra CR y LF, se elimina CR (se deja LF).

Caracteres especiales

Hay varios caracteres que forman parte del lenguaje XML y no son interpretados como simples caracteres si se sitúan dentro del documento. La siguiente tabla muestra algunos de los caracteres reservados y su codificación correspondiente:

Carácter	Codificación
<	&lt;
>	&gt;
&	&amp;
“	&quot;
‘	&apos;

Comentarios

Se pueden incluir comentarios en cualquier punto de un documento XML. Los comentarios no se consideran parte del documento. El comienzo de un comentario se

indica con los caracteres <!-- y finaliza con los caracteres -->[12]. Los caracteres -- no pueden aparecer dentro de un comentario.

Cabecera de un documento XML

Al principio de todo documento XML debe aparecer una etiqueta especial[13] que contiene información importante sobre el documento. Esta etiqueta, llamada **declaración XML**, puede contener tres atributos que indican:

- La versión de XML empleada. Por ahora sólo existe una versión, la 1.0.
- El juego de caracteres empleado.
- La importancia o no de declaraciones definidas externamente.

Por ejemplo, la siguiente declaración indica que se va a emplear la codificación del estándar ISO 8859/1[14] y no se quiere procesar las declaraciones externas:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
```

Si esta declaración no se incluye, se supone que la versión de XML que se está usando es la 1.0, la codificación de caracteres UTF-8 y sí se desea procesar las posibles declaraciones externas.

Ejemplo biblioteca

El siguiente documento XML almacena información sobre los libros de una biblioteca. Cada libro tiene un código (obligatorio) y está compuesto por cero o más autores (puede ser anónimo), un título, el año de publicación y la editorial (ambos opcionales). En la Figura 4 vemos como se representa en el navegador Microsoft Internet Explorer 5.5.

[12] Igual que en HTML.

[13] Realmente no es una etiqueta: es una instrucción de procesamiento. Estas instrucciones, encerradas entre los caracteres <? y ?>, se suelen emplear para indicar información requerida por una aplicación concreta.

[14] También conocido como ISO Latin-1. Conjunto de 255 caracteres: los caracteres ASCII, más caracteres para los idiomas Danés, Holandés, Inglés, Faroe, Finés, Alemán, Islandés, Irlandés, Italiano, Noruego, Portugués, Español y Sueco.

biblioteca.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<BIBLIOTECA>
   <LIBRO COD="1">
      <TITULO>XML para todos</TITULO>
      <AUTOR>Sergio Lujan Mora</AUTOR>
      <ANYO>2001</ANYO>
      <EDITORIAL>UA Prensa</EDITORIAL>
   </LIBRO>
   <LIBRO COD="11">
      <TITULO>Como aprobar una oposición</TITULO>
      <AUTOR>Marisa Zayas Fornieles</AUTOR>
      <AUTOR>Sergio Lujan Mora</AUTOR>
      <ANYO>1999</ANYO>
      <EDITORIAL>Prensa Editorial</EDITORIAL>
   </LIBRO>
</BIBLIOTECA>
```

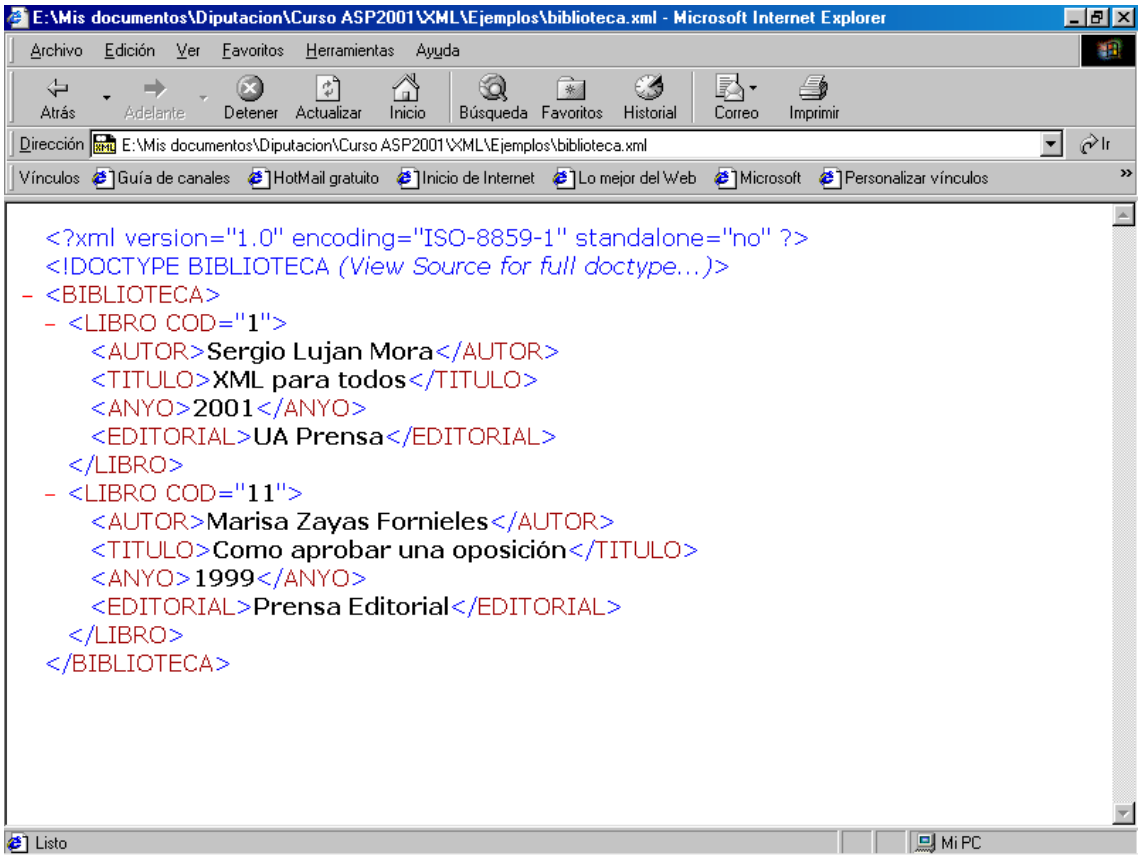

Figura 4

Ejemplo BOE

El siguiente código representa un documento XML que almacena el sumario de un Boletín Oficial del Estado. Un sumario está compuesto de secciones, cada sección tiene un título y varios apartados, cada apartado pertenece a un organismo y está compuesto de varios párrafos, que a su vez poseen un texto y una o varias referencias a las páginas que ocupa el texto. En la Figura 5 se muestra como se ve cuando se visualiza con Microsoft Internet Explorer 5.5.

boe.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BOE>
  <TITULO>BOLETIN OFICIAL DEL ESTADO</TITULO>
  <FECHA><AAAA>2001</AAAA><MM>03</MM><DD>27</DD></FECHA>
  <SUMARIO>
    <SECCION>
      <TIT-SECCION>I. Disposiciones Generales</TIT-SECCION>
      <APARTADO>
        <ORGANISMO>MINISTERIO DE ASUNTOS EXTERIORES</ORGANISMO>
        <PARRAFO>
          <TEXTO>
            <LEY>Ley 1/2001 de 5 de marzo</LEY>, de la cesión gratuita
            a las Comunidades Autónomas el Impuesto sobre Animales y
            Cosas establecido por el <RD>REAL DECRETO 13/1991</RD>.
          </TEXTO>
          <PAGINA>11290</PAGINA>
        </PARRAFO>
        <PARRAFO>
          <TEXTO>
            <ORDEN>ORDEN de 28 de febrero de 2001</ORDEN> por la ...
          </TEXTO>
          <PAGINA>11290</PAGINA>
          <PAGINA>11291</PAGINA>
        </PARRAFO>
      </APARTADO>
      <APARTADO>
        <ORGANISMO>MINISTERIO DE HACIENDA</ORGANISMO>
        <PARRAFO>
          <TEXTO>
            <RD>REAL DECRETO 285/2001</RD>, de 19 de marzo, sobre ...
```

```
      </TEXTO>
      <PAGINA>11291</PAGINA>
      <PAGINA>11292</PAGINA>
      <PAGINA>11293</PAGINA>
      <PAGINA>11294</PAGINA>
    </PARRAFO>
  </APARTADO>
  <APARTADO>
    <ORGANISMO>MINISTERIO DE FOMENTO</ORGANISMO>
    <PARRAFO>
      <TEXTO>
        CORRECCIÓN de erratas de la Orden ...
      </TEXTO>
      <PAGINA>11291</PAGINA>
      <PAGINA>11292</PAGINA>
      <PAGINA>11293</PAGINA>
      <PAGINA>11294</PAGINA>
    </PARRAFO>
  </APARTADO>
</SECCION>
<SECCION>
  <TIT-SECCION>II. Autoridades y Personal</TIT-SECCION>
  <APARTADO>
    <ORGANISMO>CORTES GENERALES</ORGANISMO>
    <PARRAFO>
      <TEXTO>
        <RES>RESOLUCIÓN de 19 de marzo de 2001</RES>, de la ...
      </TEXTO>
      <PAGINA>11294</PAGINA>
      <PAGINA>11295</PAGINA>
    </PARRAFO>
  </APARTADO>
  <APARTADO>
    <ORGANISMO>MINISTERIO DE ASUNTOS EXTERIORES</ORGANISMO>
    <PARRAFO>
      <TEXTO>
        <RD>REAL DECRETO 314/2001</RD>, de 24 de marzo, por el ...
      </TEXTO>
      <PAGINA>11291</PAGINA>
      <PAGINA>11292</PAGINA>
      <PAGINA>11293</PAGINA>
```

```
            <PAGINA>11294</PAGINA>
          </PARRAFO>
        </APARTADO>
      </SECCION>
    </SUMARIO>
</BOE>
```

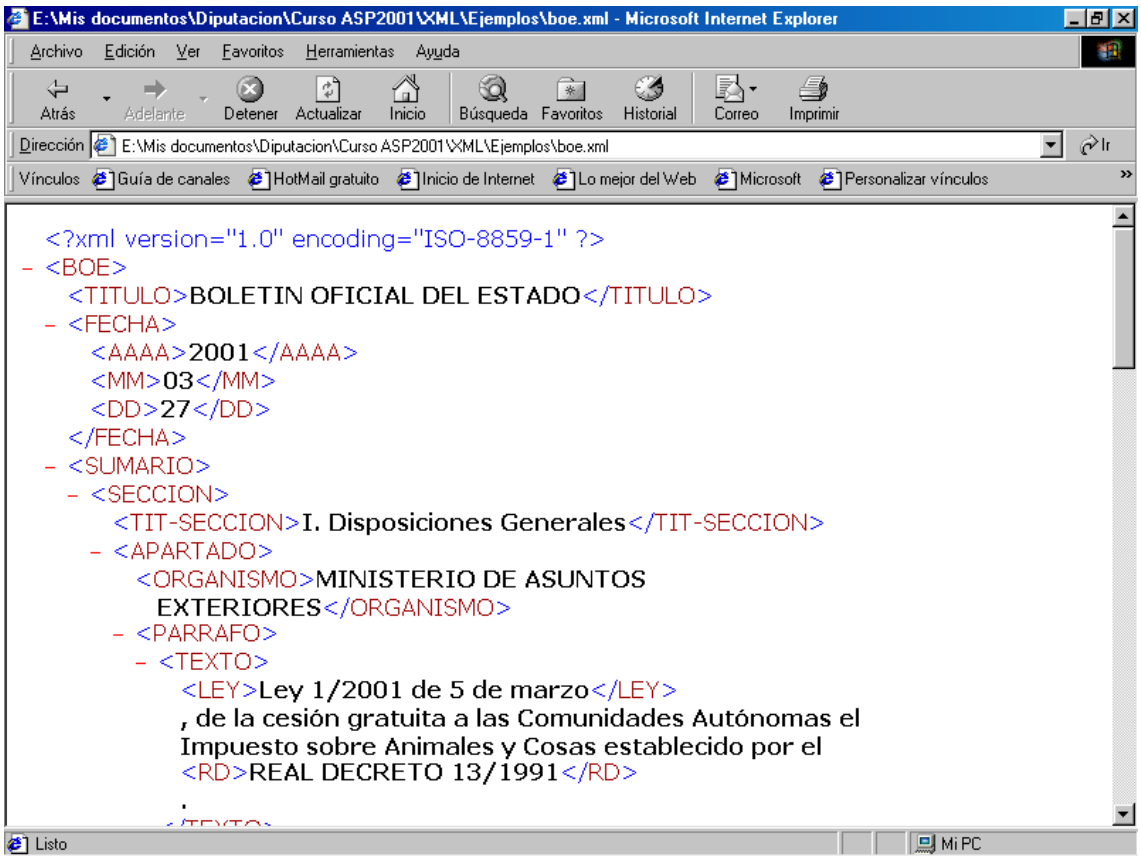

Figura 5

Microsoft XML Notepad 1.5

Esta herramienta gratuita de Microsoft[15] es un sencillo editor de documentos XML. Su objetivo es ayudar a construir prototipos de documentos XML. Ofrece un interfaz de usuario intuitivo y sencillo que representa gráficamente la estructura de un documento XML.

En la Figura 6 se puede observar el aspecto que presenta esta herramienta cuando se ejecuta por primera vez. Se puede distinguir la barra de botones, el marco

[15] Microsoft ya no soporta esta herramienta.

izquierdo donde se representa el documento en forma de árbol y la parte derecha donde se asigna valor a los atributos.

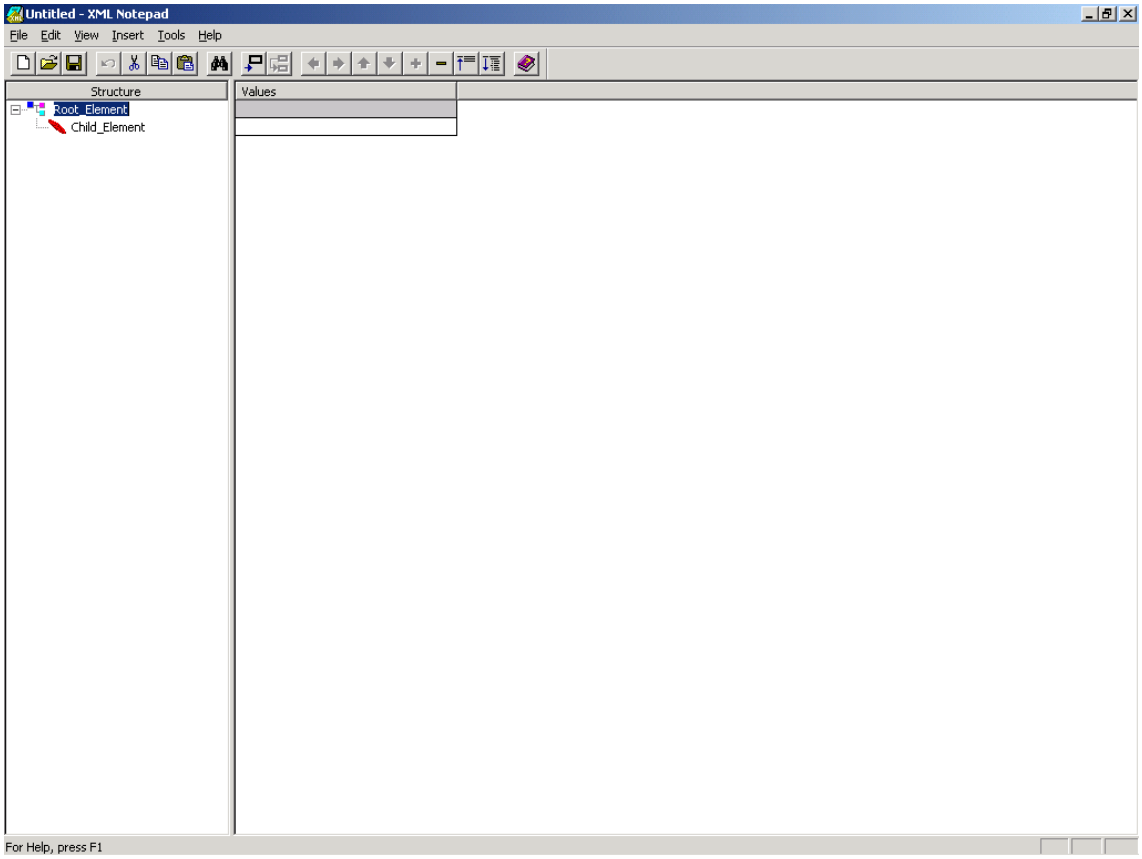

Figura 6

En la Figura 7 se muestra un documento XML visualizado con esta herramienta. En este documento, el elemento raíz es `BIBLIOTECA`, que a su vez contiene dos elementos de tipo `LIBRO`, que contienen un atributo llamado `COD` y un conjunto de elementos (`TITULO`, `AUTOR`, `ANYO`, `EDITORIAL`). Se puede observar como en cada caso, según sea el tipo del nodo del documento XML (elemento no vacío, atributo o elemento vacío) se muestra un icono específico.

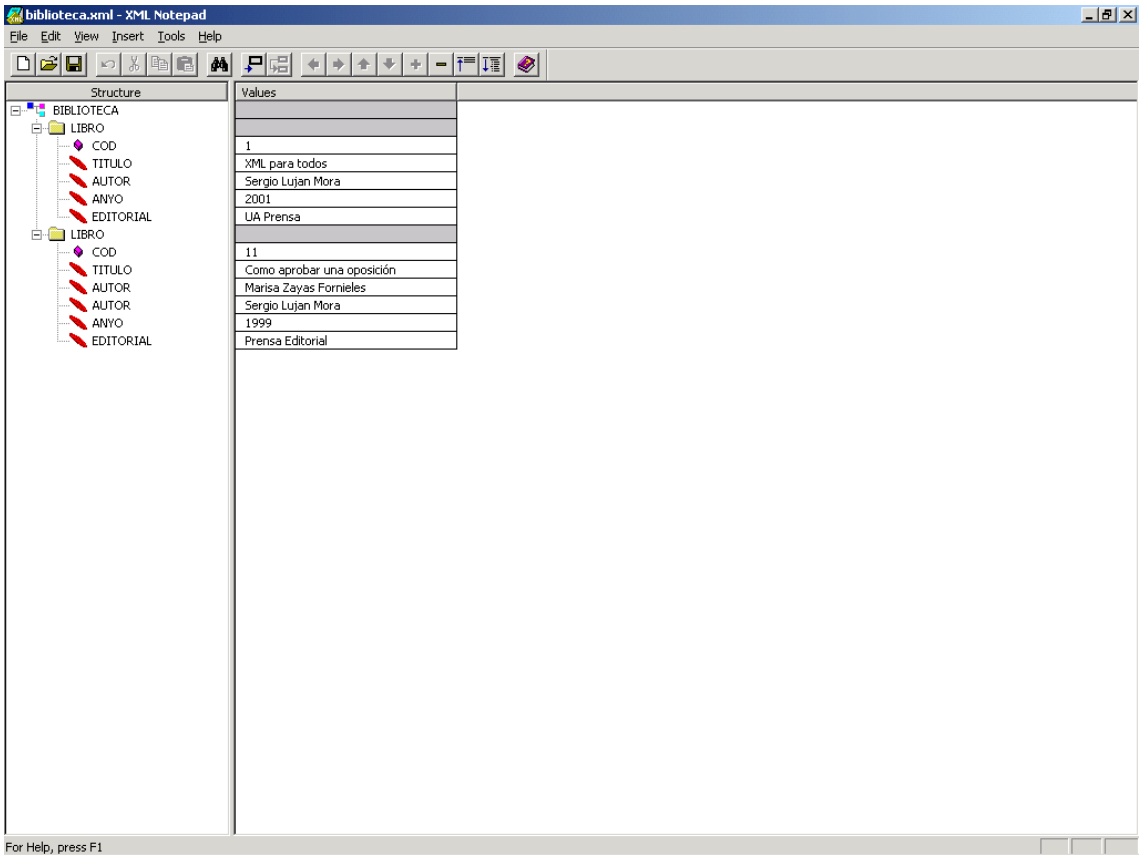

Figura 7

En la Figura 8 se muestra el código XML correspondiente del documento. Para visualizar el código XML, en el menú View existe la opción Source. Como se puede observar, el anidamiento de las etiquetas se muestra mediante un aumento de la sangría izquierda conforme aumenta el nivel de anidamiento.

Añadir nuevas etiquetas o atributos a un documento XML es muy sencillo. Si sobre el árbol del marco izquierdo se pulsa el botón derecho del ratón, aparece un menú contextual con dos opciones importantes:

- Insert: permite añadir un nuevo elemento en el mismo nivel (Element), un nuevo elemento anidado (Child Element), un atributo al elemento seleccionado (Attribute), un texto dentro del elemento seleccionado (Text) o un comentario (Comment).
- Change To: permite modificar el tipo de un nodo del documento. Las posibles conversiones son a elemento (Element), atributo (Attribute) o texto (Text).

Las opciones del menú contextual se activan en aquellos casos donde sea posible. Por ejemplo, si se tiene seleccionado un atributo, no es posible crear un elemento anidado (Child Element).

Otra opción importante es Duplicate, que permite duplicar fácilmente un parte de un documento XML. Esta opción únicamente duplica la estructura, ya que los valores originales no se duplican en la copia.

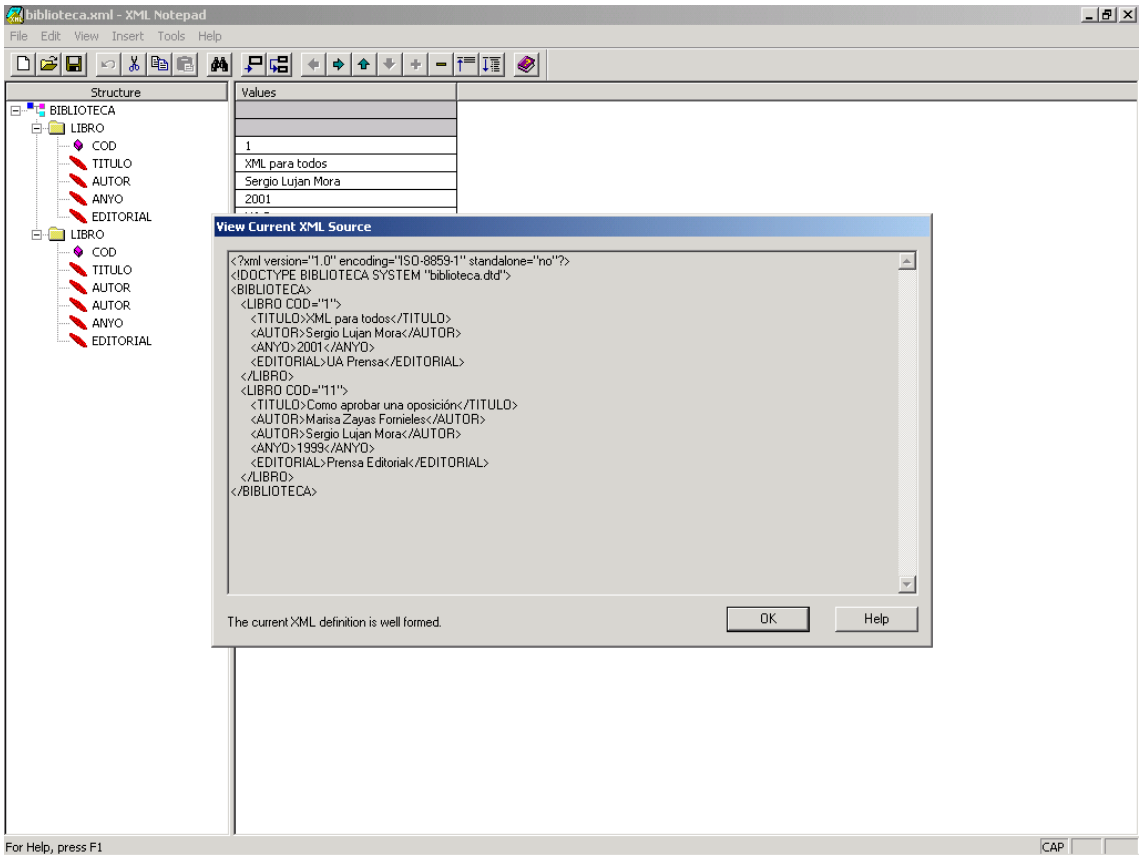

Figura 8

Estructura de un DTD

La principal característica de XML es su carácter de extensibilidad: cada usuario puede crear sus propias etiquetas. Un archivo DTD representa una gramática que describe qué etiquetas y atributos son válidos dentro de un documento XML, y en qué contexto son válidos[16]. Un DTD se compone de una serie de declaraciones. Cada

[16] Es preciso definir qué sintaxis es válida para un documento, ya que si se desarrolla un aplicación que trabaje con un tipo de documento, será necesario conocer cual es la estructura que posee.

declaración va encerrada entre los caracteres <! y >, y se clasifican en cuatro tipos de declaraciones: elemento (ELEMENT), atributo (ATTLIST), entidad (ENTITY) y notación (NOTATION).

Declaración de elemento

Una declaración de elemento se emplea para definir un elemento nuevo (definición de etiquetas) y especificar su contenido válido. La sintaxis de esta declaración es:

```
<!ELEMENT nombre contenido >
```

El nombre de un elemento debe comenzar con una letra, el carácter de subrayado '_' o dos puntos ':', seguido de uno de esos caracteres o también dígitos u otros caracteres de puntuación ('.' y '-'). No hay límites a la longitud del nombre.

El contenido de un elemento puede ser:

- Nada. Se dice que es un elemento vacío. Se indica poniendo en el contenido la palabra clave `EMPTY`. Por ejemplo:
  `<!ELEMENT imagen EMPTY>`
- Sólo otros elementos hijo. Si puede contener cualquier otro elemento existente en el DTD, se indica con la palabra clave `ANY`, aunque lo normal es indicar un conjunto de elementos con un **modelo de grupo**. Por ejemplo:
  `<!ELEMENT p ANY>`
- Sólo texto.
- Una mezcla de elementos hijo y texto.

Un elemento que puede contener elementos hijo, texto, o ambas cosas, puede ser que alguna vez no tenga ningún contenido. En ese caso, es correcto emplear tanto la etiqueta inicial y final sin contenido (`<titulo></titulo>`) o una etiqueta vacía (`<titulo/>`).

Modelo de grupo

Un modelo de grupo se emplea para describir los elementos y el texto que puede contener un elemento. Sin embargo, no se puede indicar el orden en que el texto y los elementos se tienen que entremezclar.

Un modelo de grupo va encerrado entre paréntesis y al menos debe contener el nombre de otro elemento. Se pueden controlar dos aspectos del modelo de grupo: la secuencia de aparición de los elementos y la secuencia de cantidad (cuánto se repite cada uno de ellos).

La **secuencia de aparición** se controla con los conectores ',' (**conector de secuencia**) y '|' (**conector de elección**). La secuencia `(a, b, c)` indica que al elemento `a` le sigue el elemento `b`, y a éste el elemento `c`. Por ejemplo, si queremos indicar que un libro está formado por la portada, el contenido y la contraportada:

```
<!ELEMENT LIBRO (PORTADA, CONTENIDO, CONTRAPORTADA)>
```

Un documento XML válido según el DTD anterior sería:

```
<LIBRO><PORTADA/><CONTENIDO/><CONTRAPORTADA/></LIBRO>
```

La secuencia `(a | b | c)` indica que se puede elegir entre los tres elementos y sólo se puede seleccionar uno. Por ejemplo, si queremos indicar que una jaula puede contener un canario, un loro o un periquito:

```
<!ELEMENT JAULA (CANARIO, LORO, PERIQUITO)>
```

Se pueden construir expresiones más complejas combinando varias veces estos dos operadores. Por ejemplo, la representación de un menú puede ser:

```
<!ELEMENT MENU ((Entremeses | Ensalada), (Paella | Sopa),
(Emperador | Pollo), ((Agua | Vino) | (Helado | Flan)))>
```

La **secuencia de cantidad** permite indicar cuantas veces puede aparecer un elemento en cada sitio. Para ello se emplean los **indicadores de cantidad**:

Indicador	Representa	Cardinalidad
Nada	El elemento es requerido	1

?	El elemento puede o no puede aparecer (es opcional)	0 ó 1
+	El elemento es requerido, pero puede repetirse	1 a N
*	El elemento puede no aparecer o aparecer varias veces (es opcional pero también se puede repetir)	0 a N

Los indicadores de cantidad se pueden aplicar también a un modelo de grupo. Por ejemplo:

DTD	Descripción
`<!ELEMENT prueba (a, b+)>`	Después del elemento `a` tiene que aparecer el `b` al menos una vez
`<!ELEMENT prueba (a, a+)>`	El elemento `a` tiene que aparecer dos o más veces
`<!ELEMENT prueba (a+, b+)>`	Una secuencia de una o más veces el elemento `a` seguida de una secuencia de una o más veces el elemento `b`. Por ejemplo: `<a/><a/><a/><b/><b/>`
`<!ELEMENT prueba (a, b)+>`	Una secuencia de una o más veces los elementos `a` y `b`. Por ejemplo: `<a/><b/><a/><b/><a/><b/>` No confundir con el anterior modelo

Todos estos conectores e indicadores se pueden combinar entre sí. Pero si no se hace con cuidado, se pueden originar ambigüedades, lo que impedirá que un analizador de XML lo pueda procesar. Por ejemplo, en la siguiente tabla aparecen dos casos de ambigüedad y su posible solución:

Ambiguo	Correcto
`(item?, item)`	`(item, item?)`
`((apellido, empleado) | (apellido, cliente))`	`(apellido, (empleado | cliente))`

Texto

La posición donde puede aparecer texto se indica con la palabra clave **PCDATA** (*Parsable Character Data*), y que tiene que aparecer precedida con el **indicador de nombre reservado** '**#**', para evitar confusiones con un elemento que posea el mismo nombre. Esta palabra clave representa cero o más caracteres.

Un elemento que sólo contenga texto[17], se define así:

```
<!ELEMENT TITULO (#PCDATA)>
```

Cuando se quiere que un elemento contenga tanto texto como elementos hijo, hay que seguir una serie de reglas estrictas (definen un **modelo de contenido mixto**):

- La palabra `PCDATA` debe ser el primer elemento en el modelo de grupo.
- Los elementos del grupo deben de estar unidos mediante conectores de elección.
- El grupo tiene que ser opcional y repetible.

Por ejemplo, las siguientes definiciones son correctas:

```
<!ELEMENT TITULO (#PCDATA | sub | sup)*>
<!ELEMENT sub (#PCDATA)>
<!ELEMENT sup (#PCDATA)>
```

Un documento XML válido según el DTD anterior podría ser:

```
<TITULO>El <sup>título</sup> del <sub>documento</sub></TITULO>
```

Declaración de atributo

Los atributos de los elementos se declaran separadamente del elemento, en la declaración de atributos. Normalmente, todos los atributos asociados con un elemento se declaran todos juntos en una única declaración; cuando haya más de una declaración, se combinan todas ellas en una sola. Cuando existan varias declaraciones del mismo atributo, la primera de las definiciones tiene precedencia sobre las demás.

[17] Prestar mucha atención a los paréntesis: siempre hay que ponerlos, aunque sólo encierren un elemento.

Una declaración de atributos se identifica por la palabra reservada ATTLIST, seguida del nombre del elemento al que hace referencia, más una serie de parejas o tríos que indican el nombre del atributo, el tipo de atributo y el valor por defecto (opcional):

```
<!ATTLIST elemento
  atributo1 contenido1 defecto1
  atributo2 contenido2 defecto2 ...>
```

Una posible declaración de tres atributos para el elemento `img` podría ser:

```
<!ATTLIST img
  src CDATA
  align (left | center | right)
  size NMTOKEN #REQUIRED>
```

El nombre del atributo sigue las mismas reglas que los nombres de los elementos. El tipo de atributo restringe el rango de valores que puede contener. Los tipos de atributos existentes son:

- CDATA
  Una cadena de texto (no impone ninguna restricción). Por ejemplo:
  `<alumno nombre=”Sergio Luján Mora”>`
- NMTOKEN
  Un token (palabra), con las mismas limitaciones que los nombres de los elementos y los atributos, excepto que no existen restricciones con el primer carácter (puede comenzar por un dígito). Por ejemplo:
  `<doc size=”A4”>`
- NMTOKENS
  Una serie de tokens separados por espacios en blanco. Por ejemplo:
  `<figura limites=”5 10 40 60”>`
- ENTITY
  El valor del atributo es una referencia a una entidad.
- ENTITIES
  Una serie de referencias a entidades separadas por espacios en blanco.

- ID

  Sirve para identificar a un elemento. Todos los atributos ID de un documento han de contener valores diferentes de modo que sea posible identificar cada elemento por su ID. Cada elemento sólo puede tener un atributo de este tipo. Por ejemplo:

  ```
  <!ATTLIST alumno
    nombre CDATA
    matricula ID>
  ```

- IDREF

  Empleado para facilitar los enlaces a hipertexto: sirve para crear referencias otros elementos a partir de su ID.

- IDREFS

  Una serie de referencias a valores ID separadas por espacios en blanco.

- NOTATION

  Indica que tipo de datos embebidos pueden aparecer dentro de un elemento.

- Grupo de tokens

  Restringe el valor a un conjunto finito de valores (lista de enumerados). Por ejemplo, la definición `(left | center | right)` especifica que el valor debe ser uno de los indicados:

  ```
  <table align="center">
  ```

El último parámetro, que es opcional, indica si existe un valor por defecto cuando no se indica un valor para el atributo o si el atributo es obligatorio u opcional.

Los atributos obligatorios se denominan **atributos requeridos** y se indican con la palabra reservada `#REQUIRED`.

Los atributos opcionales se denominan **atributos implícitos** y se indican con la palabra reservada `#IMPLIED`.

Los atributos también pueden tener un **valor por defecto**: cuando no se indica un valor, se emplea el valor por defecto definido. No se puede definir un valor por defecto con los atributos requeridos o implícitos. Por ejemplo, el siguiente fragmento de DTD:

```
<!ATTLIST lista
  separador NMTOKEN #REQUIRED
  desplaza NMTOKEN #IMPLIED
  tipo (alpha | num) "num">
```

permite definir el siguiente documento XML:

```
<lista separador="*">…</separador>
<lista separador="*" desplaza="5mm">…</separador>
<lista separador="*" tipo="num">…</separador>
<lista separador="*" desplaza="5mm "tipo="alpha">…</separador>
```

Por último, si un valor por defecto va precedido de la palabra reservada `#FIXED`, entonces el valor por defecto es el único valor que puede tomar el atributo[18].

Declaración de entidad

Las declaraciones de entidad permiten asociar un nombre con un fragmento de texto, de modo que cuando se haga referencia al nombre, se sustituirá por el fragmento de texto. Las ventajas que nos ofrece esta característica son:

- Facilita la escritura de DTDs.
- Disminuye los posibles errores.
- Clarifica la estructura de los DTDs (aunque su abuso puede producir el efecto contrario).
- Los DTDs se pueden modificar más fácilmente, ya que modificando el valor en la declaración de la entidad, lo modificamos en todo el documento.

Por ejemplo, si tenemos estas dos declaraciones de elementos:

```
<!ELEMENT tit (#PCDATA | parrafo | lista | definicion | nota)*>
<!ELEMENT sub (#PCDATA | parrafo | lista | definicion)*>
<!ELEMENT sup (#PCDATA | parrafo | lista | definicion)*>
```

podemos sustituirla por las siguientes líneas:

[18] Puede parecer raro el empleo de #FIXED, pero tiene su sentido cuando se usa con otros estándares de XML como XLink.

```
<!ENTITY % sub "#PCDATA | parrafo | lista | definicion">
<!ELEMENT tit (%texto; | nota)*>
<!ELEMENT sub (%texto;)*>
<!ELEMENT sup (%texto;)*>
```

Dos cuestiones importantes a tener en cuenta son:

- Una referencia a una entidad no puede estar modificada directamente por un indicador de cantidad (?, +, *). Si se quiere modificar, tiene que ir encerrado entre paréntesis.
- Al principio y al final del texto que se sustituye se añade un espacio en blanco para asegurar que la entidad contiene únicamente tokens completos.

Declaración de notación

Las declaraciones de notación son utilizadas para especificar el tipo de los ficheros binarios referenciados mediante las referencias a entidades externas no textuales. Esta información se pasa directamente sin analizar y es por tanto la aplicación la que debe decidir que hacer con esta información. Una declaración típica de este tipo es:

```
<!NOTATION GIF87A SYSTEM "GIF">
```

Colocación de las declaraciones dentro de un documento

Hay tres formas de indicar a que DTD pertenece un determinado documento XML[19]:

1. Incluir todo el DTD en el documento XML.

```
<!DOCTYPE libro [
<!-- Aquí comienza el DTD -->
<!ELEMENT libro (portada, cuerpo, contraportada)>
<!ELEMENT portada (titulo, edicion, autor, editor)>
<!ELEMENT titulo (#PCDATA)>
```

[19] Aunque no es obligatorio, es lo primero que deberíamos indicar en el documento.

```
<!ELEMENT edicion (#PCDATA)>
<!ELEMENT autor (nombre, apellidos, email?)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT email (#PCDATA)>
...
<!-- Aquí termina el DTD -->
]>
```

2. Incluir una referencia al fichero donde se encuentra el DTD. En este caso, puede ser de dos tipos:
   - PUBLIC: referencia a un DTD definido por organismos o estándares públicos.

     ```
     <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
     "http://www.wapforum.org/DTD/wml_1.1.xml">
     ```

   - SYSTEM: definido por el usuario.

     ```
     <!DOCTYPE libro SYSTEM "../DTDS/libro.dtd">
     ```

3. La combinación de las dos anteriores.

```
<!DOCTYPE libro SYSTEM "libro.dtd" [
<!-- Aquí comienza el DTD -->
<!ELEMENT autor (nombre, apellidos, email?)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!-- Aquí termina el DTD -->
]>
```

Orden de procesamiento de los DTDs

Cuando un documento XML contiene tanto un DTD interno como una referencia a un DTD externo, el interno se procesa antes que el externo. Si se declara un elemento, atributo o entidad con el mismo nombre varias veces, la primera declaración prima frente a las sucesivas. Por tanto, las declaraciones del DTD interno se pueden usar para modificar el DTD externo.

Además, se puede emplear el atributo `standalone` en la declaración XML[20] para indicar si se tienen que leer las declaraciones externas. En la Figura 9 se resume el procesamiento de los DTDs internos y externos.

[20] Ver el punto *Cabecera de un documento XML*.

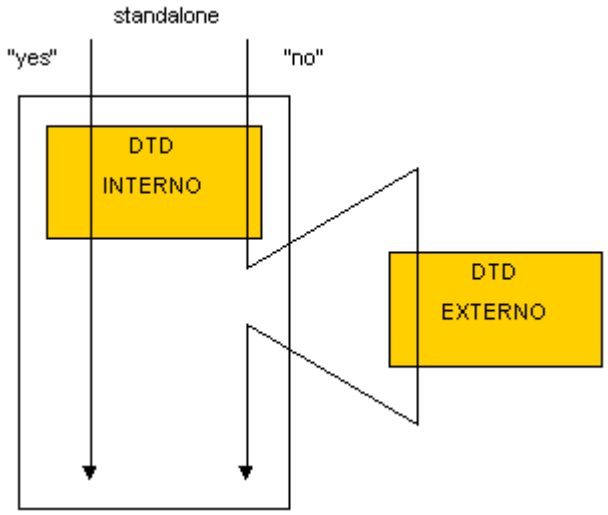

Figura 9

DTDs ya definidos

Existen diversos DTDs destinados a cubrir necesidades muy específicas y que permiten describir la estructura de documentos concretos:

DTD	Descripción
Channel Definition Format (CDF)	Define canales en los que el servidor envía periódicamente información a los usuarios
Chemical Markup Language (CML)	Describe fórmulas y datos químicos
Genealogical Data in XML (GedML)	Proporciona un formato de descripción para datos genealógicos
Mathematical Markup Language (MathML)	Describe datos matemáticos
Open Software Description (OSD)	Describe paquetes software que permiten instalar de modo remoto software y componentes de una intranet o Internet
Synchronized Multimedia Language (SMIL)	Permite definir presentaciones para la web con objetos multimedia sincronizados

Resource Description Framework (RDF)	Describe recursos de todo tipo para su mejor catalogación, búsqueda y referencia

Ejemplo biblioteca

El siguiente DTD representa la estructura del ejemplo de biblioteca que se ha visto anteriormente. El fichero `biblioteca.xml` hay que modificarlo para que incluya el DTD:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE BIBLIOTECA SYSTEM "biblioteca.dtd">
```

biblioteca.dtd:

```
<!ELEMENT BIBLIOTECA (LIBRO+)>
<!ELEMENT LIBRO (TITULO, AUTOR*, ANYO?, EDITORIAL?)>
<!ATTLIST LIBRO COD CDATA #REQUIRED>
<!ELEMENT AUTOR (#PCDATA)>
<!ELEMENT TITULO (#PCDATA)>
<!ELEMENT EDITORIAL (#PCDATA)>
<!ELEMENT ANYO (#PCDATA)>
```

Ejemplo BOE

El siguiente código es un DTD para el ejemplo del Boletín Oficial del Estado. El fichero boe.xml hay que modificarlo para que incluya el DTD:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE BIBLIOTECA SYSTEM "boe.dtd">
```

boe.dtd:

```
<!ELEMENT BOE (TITULO, FECHA, SUMARIO)>
<!ELEMENT TITULO (#PCDATA)>
<!ELEMENT FECHA (AAAA, MM, DD)>
<!ELEMENT AAAA (#PCDATA)>
<!ELEMENT MM (#PCDATA)>
<!ELEMENT DD (#PCDATA)>
```

```
<!ELEMENT SUMARIO (SECCION+)>
<!ELEMENT SECCION (TIT-SECCION, APARTADO+)>
<!ELEMENT TIT-SECCION (#PCDATA)>
<!ELEMENT APARTADO (ORGANISMO, PARRAFO+)>
<!ELEMENT ORGANISMO (#PCDATA)>
<!ELEMENT PARRAFO (TEXTO, PAGINA+)>
<!ELEMENT TEXTO (#PCDATA | LEY | ORDEN | RD | RES)*>
<!ELEMENT PAGINA (#PCDATA)>
<!ELEMENT LEY (#PCDATA)>
<!ELEMENT RD (#PCDATA)>
<!ELEMENT ORDEN (#PCDATA)>
<!ELEMENT RES (#PCDATA)>
```

Validación de un documento XML contra su DTD

Existen dos tipos de validaciones:

- Comprobar que un documento está bien formado: se ajusta a las normas que marca XML (todas las etiquetas están cerradas, los valores de los atributos aparecen entre comillas, etc.).
- Comprobar que un documento es válido respecto a su DTD: verificar que su estructura y contenido se ajusta a lo indicado en su DTD.

Existen diversas herramientas que realizan las dos funciones. Microsoft Internet Explorer por defecto sólo realiza la primera, pero si se instala Micrososft IE XML/XSL Viewer Tools[21], se incorpora al navegador la opción *Validate XML*.

Documento bien formado

Cada vez que se visualiza un documento XML, Microsoft Internet Explorer verifica que está bien formado. En caso contrario, muestra un mensaje de error donde se explica la situación del error y la causa del error.

[21] Disponible en `http://www.microsoft.com/downloads/details.aspx?FamilyId=D23C1D2C-1571-4D61-BDA8-ADF9F6849DF9&displaylang=en`.

Por ejemplo, el siguiente documento XML contiene dos errores (marcados en negrita): una etiqueta `TITULO` no se ha cerrado y el valor de un atributo `DNI` no aparece entre comillas.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<CURSO>
   <TITULO>XML en 12 horas
   <FECHA-INICIO>
      <DIA>12</DIA>
      <MES>4</MES>
      <ANYO>2001</ANYO>
   </FECHA-INICIO>
   <FECHA-FIN>
      <DIA>14</DIA>
      <MES>4</MES>
      <ANYO>2001</ANYO>
   </FECHA-FIN>
   <ALUMNOS>
      <ALUMNO DNI=21513360>
         <NOMBRE>Sergio</NOMBRE>
         <APELLIDOS>Luján Mora</APELLIDOS>
      </ALUMNO>
      <ALUMNO DNI="22333444">
         <NOMBRE>Juan</NOMBRE>
         <APELLIDOS>López Pérez</APELLIDOS>
      </ALUMNO>
      <ALUMNO DNI="11222333">
         <NOMBRE>María</NOMBRE>
         <APELLIDOS>Gómez Gómez</APELLIDOS>
      </ALUMNO>
   </ALUMNOS>
</CURSO>
```

En la Figura 10 el mensaje de error que muestra Internet Explorer al intentar visualizar este documento XML. El mensaje de error indica que la etiqueta de fin `CURSO` no coincide con la etiqueta de inicio `TITULO`: se ha llegado al final del documento y no se ha encontrado la etiqueta de cierre de `TITULO`.

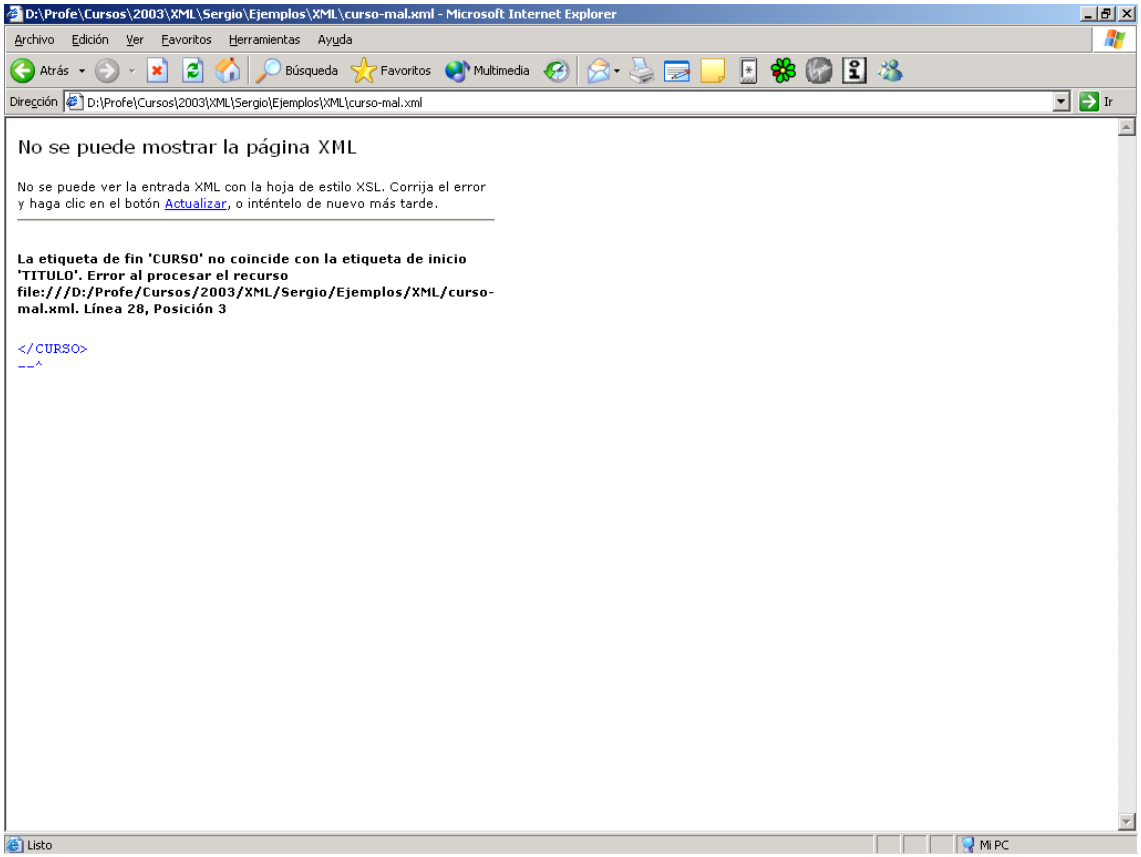

Figura 10

Si se arregla el error anterior y se vuelve a cargar el documento XML, Internet Explorer muestra un mensaje que detecta el segundo error (Figura 11): no se han encontrado las comillas de apertura.

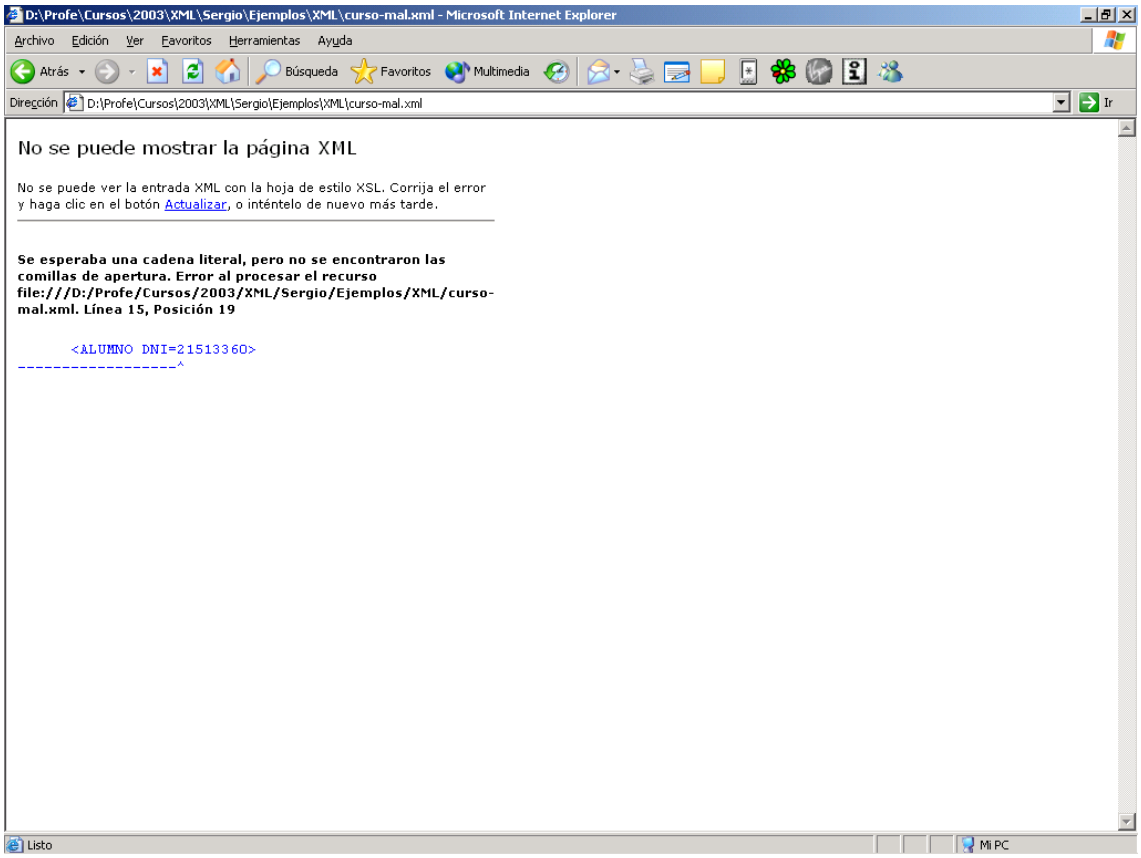

Figura 11

Documento válido

Para validar un documento XML es necesario disponer de su DTD: si el DTD no existe o no está disponible, un documento XML bien formado siempre es válido. En la Figura 12 se muestra el menú contextual de Internet Explorer; se pueden apreciar las dos opciones que aparecen cuando se instala Micrososft IE XML/XSL Viewer Tools: `Validate XML` y `View XSL Output`.

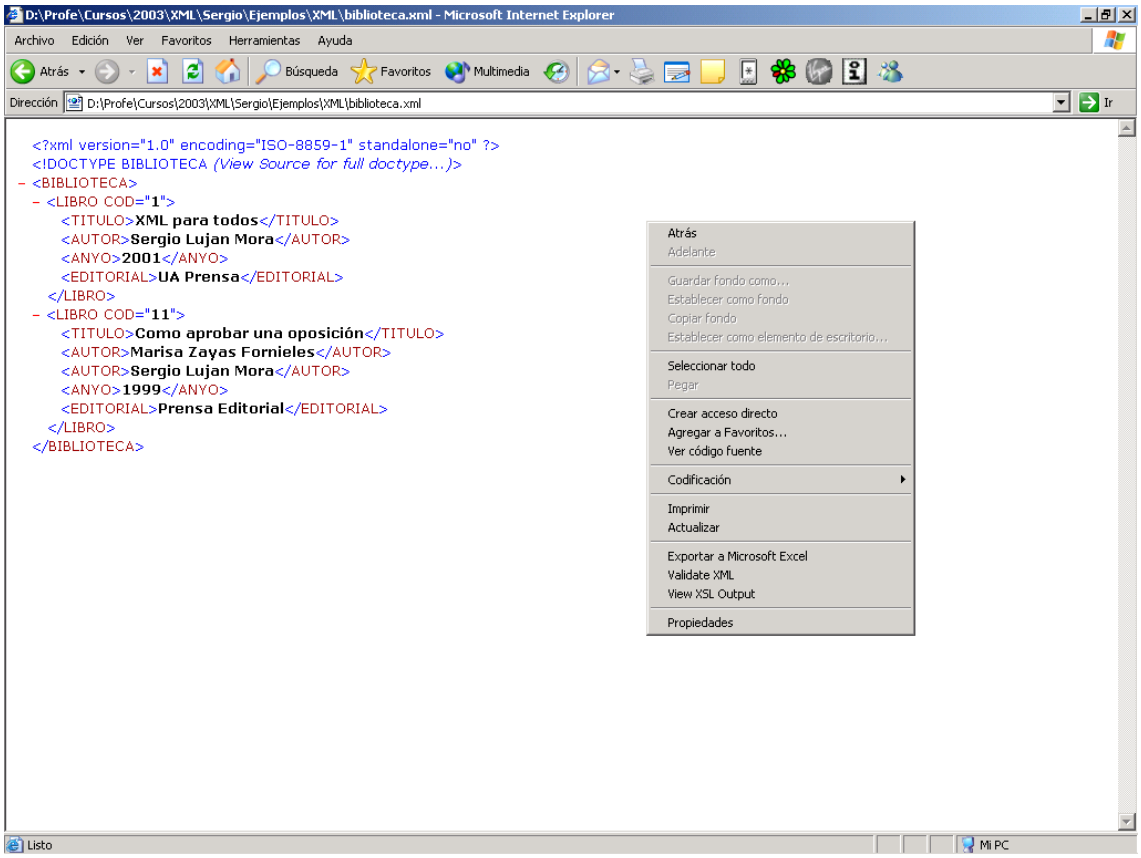

Figura 12

En la Figura 13 se muestra la validación de un documento XML bien formado y válido con respecto a su DTD: al pulsar la opción *Validate XML*, se ejecuta el proceso de validación y se informa al usuario del resultado del mismo.

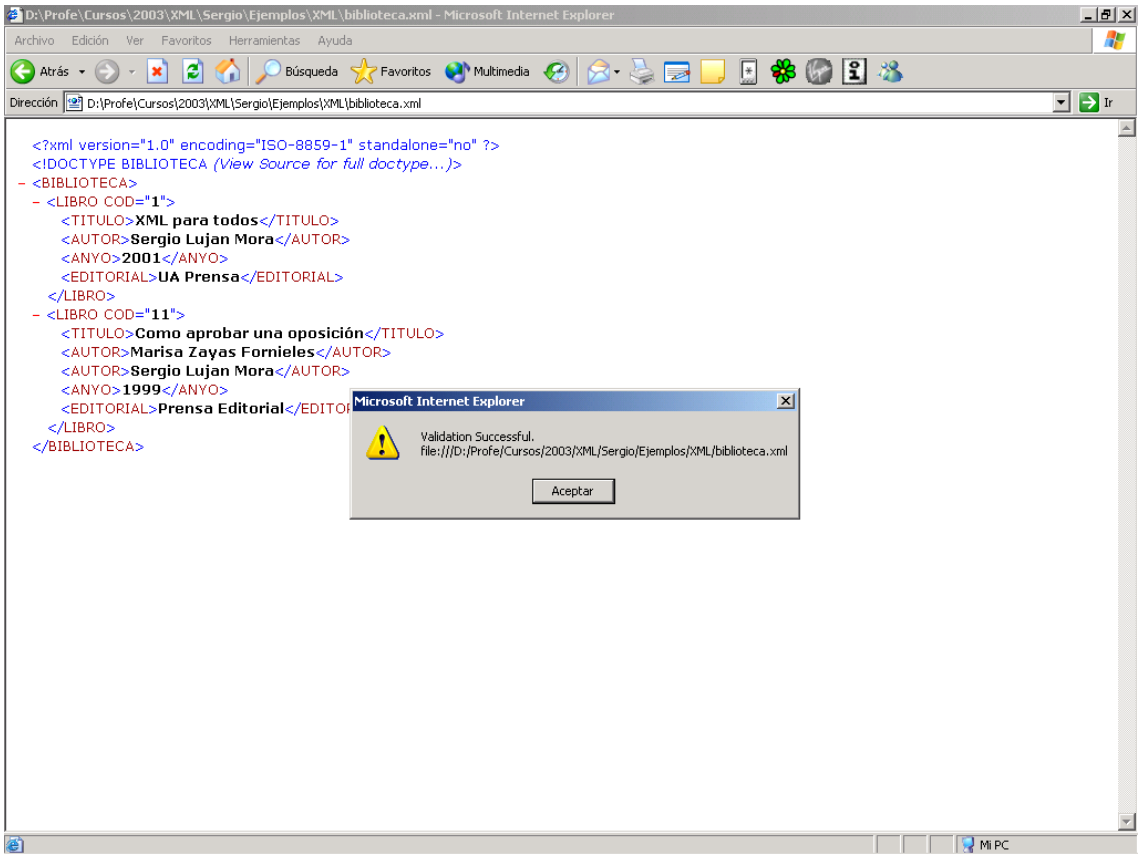

Figura 13

El siguiente documento XML está bien formado pero no es válido respecto a su DTD: en el primer libro aparece dos veces la etiqueta `TITULO`, pero eso no está permitido por el DTD correspondiente. Cómo el documento está bien formado, cuando se carga en Internet Explorer no se produce ningún mensaje de error, pero cuando se valida (Figura 14) se informa al usuario del tipo de error.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE BIBLIOTECA SYSTEM "biblioteca.dtd">
<BIBLIOTECA>
   <LIBRO COD="1">
      <TITULO>XML para todos</TITULO>
      <TITULO>Un enfoque práctico</TITULO>
      <AUTOR>Sergio Lujan Mora</AUTOR>
      <ANYO>2001</ANYO>
      <EDITORIAL>UA Prensa</EDITORIAL>
   </LIBRO>
   <LIBRO COD="11">
```

```
        <TITULO>Como aprobar una oposición</TITULO>
        <AUTOR>Marisa Zayas Fornieles</AUTOR>
        <AUTOR>Sergio Lujan Mora</AUTOR>
        <ANYO>1999</ANYO>
        <EDITORIAL>Prensa Editorial</EDITORIAL>
    </LIBRO>
</BIBLIOTECA>
```

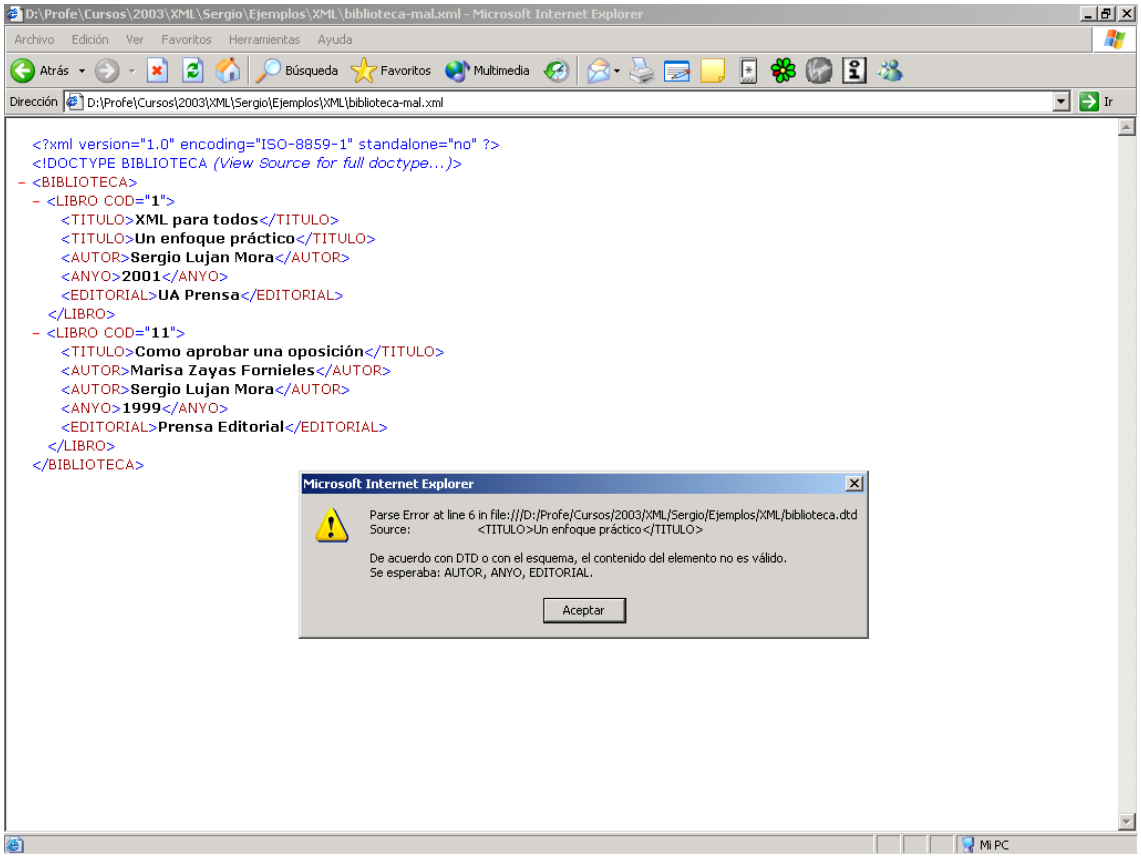

Figura 14

ezDTD 1.5

Esta herramienta gratuita es un editor de DTDs sencillos. Se puede emplear para crear DTDs tanto para SGML como para XML. Una característica útil de esta herramienta es su capacidad de generar una página HTML con el contenido del DTD y con una serie de enlaces que facilitan la lectura de un DTD.

En la Figura 15 se muestra la ventana de la aplicación cuando se ejecuta por primera vez. En la parte superior de la ventana se pueden observar una serie de botones (*Document*, *Heading*, etc.) que permiten definir las distintas partes de un DTD. La parte central de la venta varía según la opción elegida en la barra de botones. En la parte inferior de la ventana se aprecia una pestaña con dos opciones: `Editor` y `Preview`.

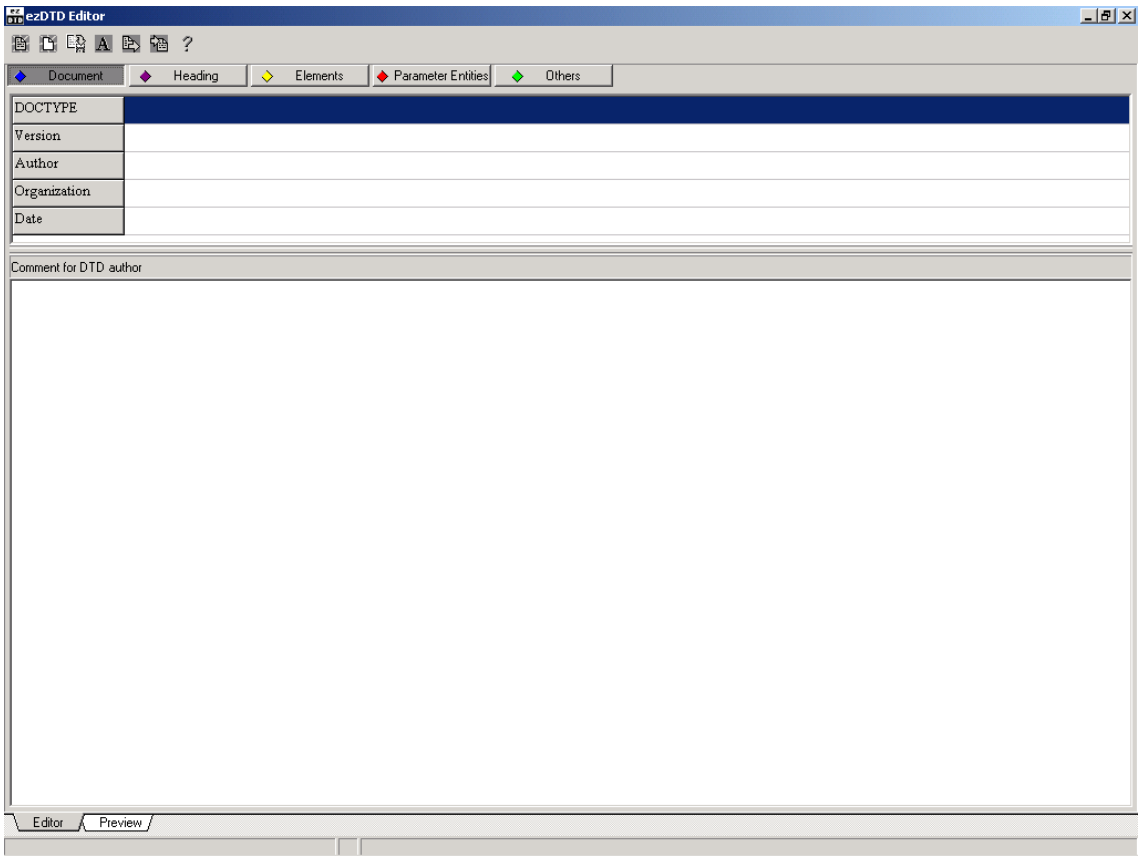

Figura 15

En la Figura 16 se muestra el contenido del DTD `biblioteca.dtd` que se ha definido previamente. En la lista de la izquierda aparecen los elementos definidos en este DTD. Cuando se selecciona un elemento, en la parte derecha de la ventana se pueden modificar sus características: los elementos que contiene y los atributos que posee. En la Figura 16 se puede observar una lista desplegable en el área para definir atributos: esta lista contiene las palabras clave que se emplean para definir un atributo y se puede visualizar pulsando CTRL+Espacio.

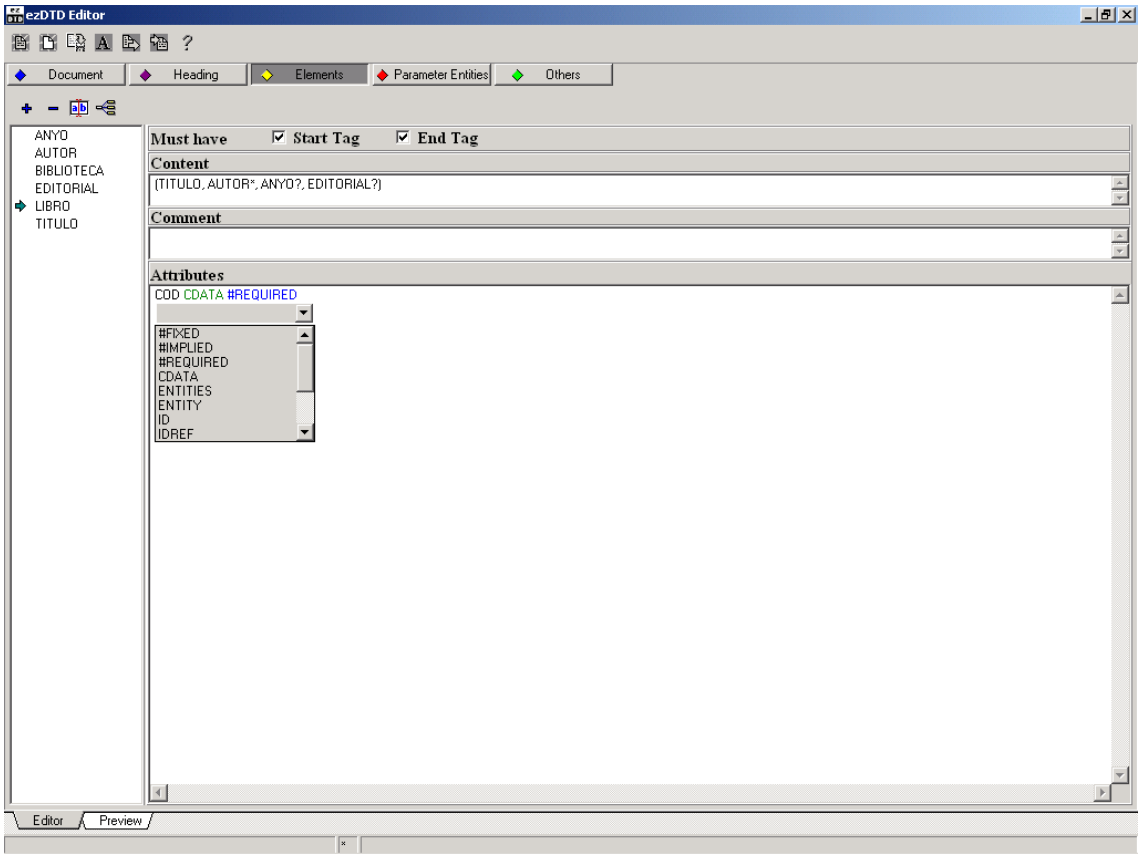

Figura 16

En la Figura 17 se muestra el mismo DTD pero en la opción *Preview*. Esta opción permite visualiza el DTD tal como va a quedar y además permite navegar a través del DTD mediante una serie de enlaces en los elementos del DTD. En cualquier momento, si se pulsa sobre << se vuelve al modo editor para modificar el elemento sobre el que se ha pulsado.

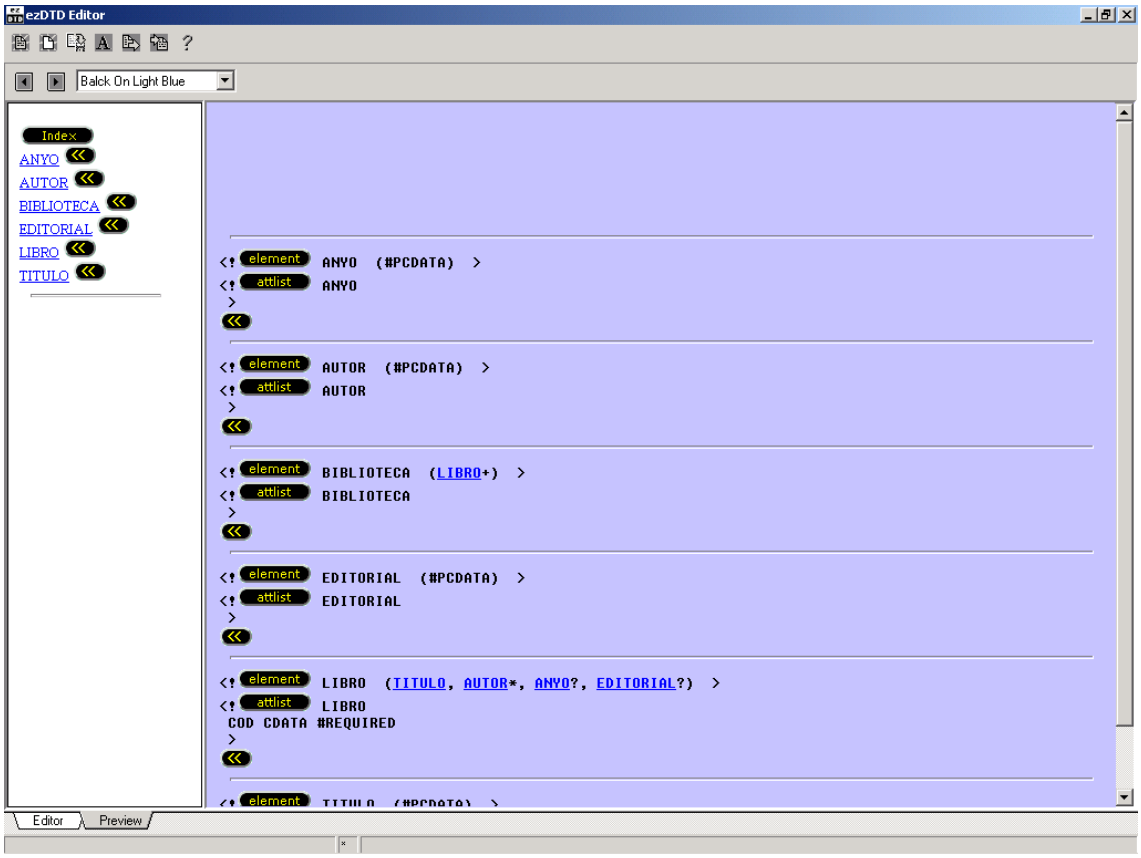

Figura 17

Finalmente, la opción *Generate DTD File*, además de generar el correspondiente DTD, también permite generar una página HTML con enlaces que contiene el DTD. En esta página, el contenido de un elemento que sea a su vez otro elemento se convierte en un enlace, lo que facilita la lectura de un DTD. En la Figura 18 se puede observar la página HTML que se ha generado automáticamente a partir del DTD anterior.

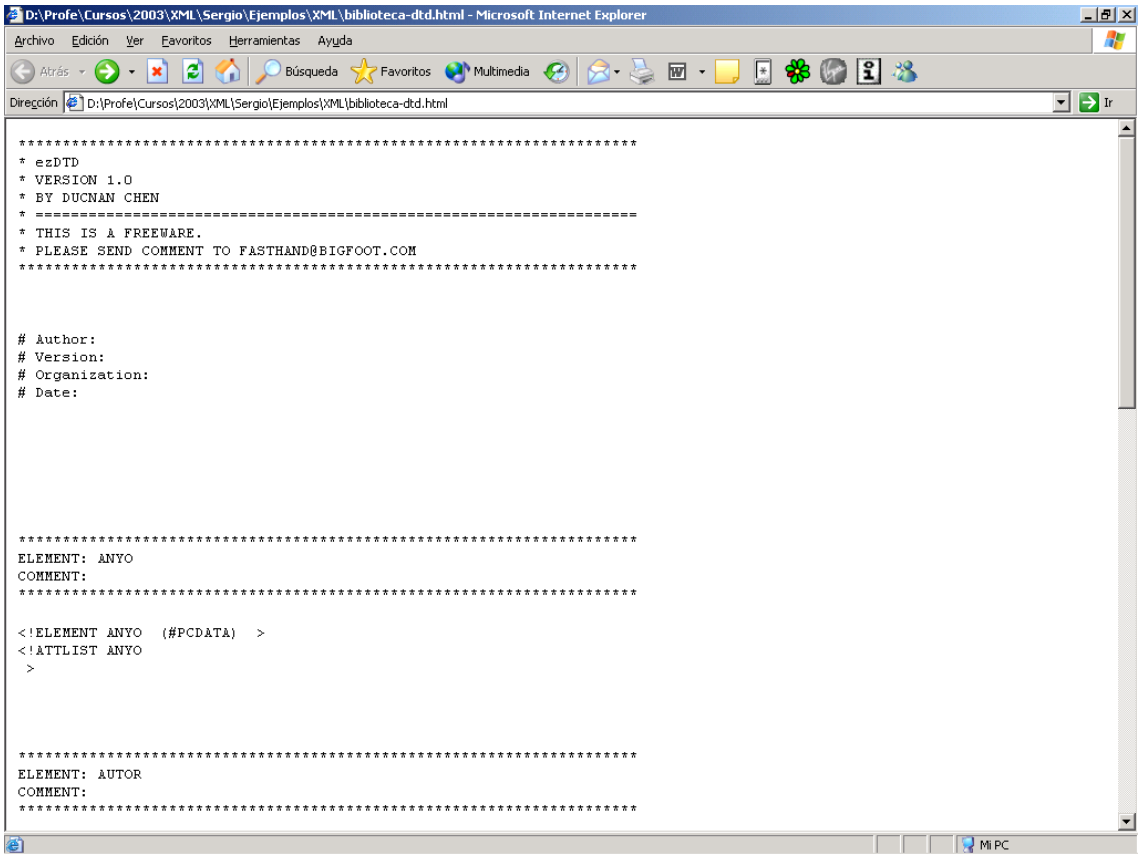

Figura 18

Presentación de los documentos XML

El estándar XML sirve para separar la presentación de los contenidos. Por ello, para visualizar la información contenida en un documento XML en papel o en una pantalla de ordenador es necesario disponer de un mecanismo que indique el estilo de la información[22].

Para describir cómo se deben presentar los documentos XML podemos optar por dos soluciones: las hojas de estilo en cascada (*Cascade Style Sheets*, CSS), que se utilizan en HTML, y el lenguaje de hojas de estilo extensible (XSL, *XML Stylesheet Language*) junto con XSLT (*XSL Transformation*).

[22] Los documentos XML son sencillos de leer, tanto por un humano como por un ordenador, pero es de esperar que la gente que quiera ver un documento no esté interesada en las etiquetas que lo definen.

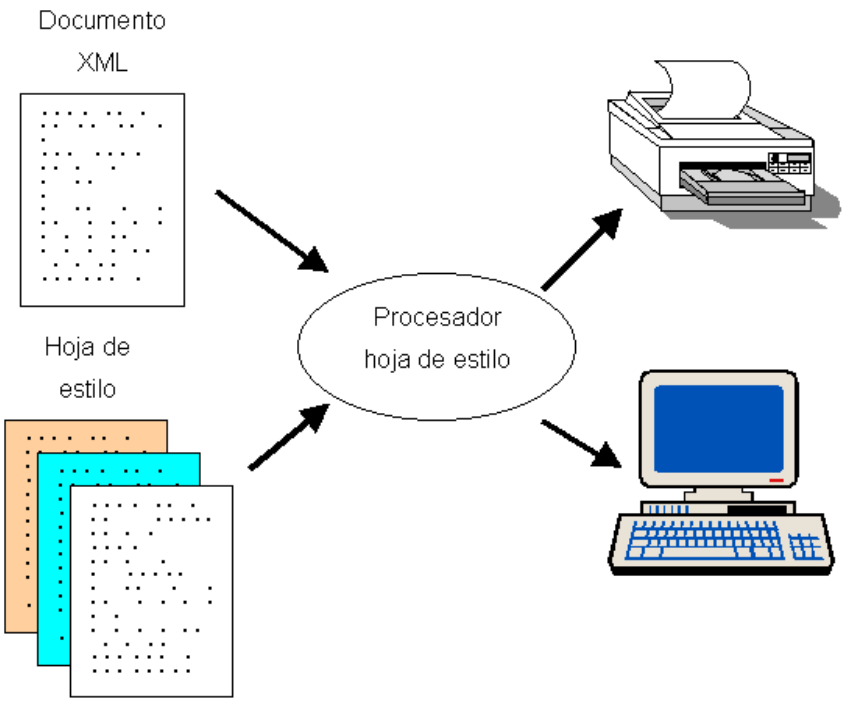

Figura 19

Para documentos en los que los datos tengan una estructura sencilla y regular, puede que las hojas de estilo CSS sean más que suficientes. El lenguaje XSL es más complejo pero pone a nuestra disposición toda la flexibilidad y potencia de un verdadero mecanismo de consulta de datos.

Otros estándares

Existen otros estándares (en desarrollo o finalizados) que complementan a XML:

- DOM (*Document Object Model*): modelo arborescente de un documento XML.
- SAX (*Simple API for XML*): API estándar para el procesamiento de documentos XML.
- XLink y XPointer: definen estándares para representar enlaces entre distintos recursos.

- XMI (*XML Metada Interchange*): representación de modelos UML[23] mediante XML. Permite exportar modelos de una herramienta UML a otra.
- XML-QL: lenguaje de consulta de documentos XML (basado en SQL). Propuesto por AT&T Labs.
- XQL: lenguaje de consulta de documentos XML (basado en XSL). Propuesto por Microsoft, Texcel y WebMethods.
- XML Schemas: un lenguaje para describir el contenido de un documento XML. Sustituye a los DTSs.

Bibliografía

Aladro, Adolfo (1999), "XML. El nuevo estándar de Internet (I)", *Sólo Programadores* 56, Abril, páginas 24-33

Aladro, Adolfo (1999), "XML (II). El DOM de XML", *Sólo Programadores* 57, Mayo, páginas 34-41

Aladro, Adolfo (1999), "XML (III). El lenguaje XSL", *Sólo Programadores* 58, Junio, páginas 38-45

Aladro, Adolfo (1999), "XML (IV). El lenguaje XSL (II)", *Sólo Programadores* 59, Julio, páginas 38-45

Bradley, Neil (2000), *The XML companion*, Addison-Wesley, Segunda edición

Bray, Tim (1998), "The Annotated XML Specification", en http://www.xml.com/axml/testaxml.htm (accedido el 26/3/2001)

Marchal, Benoît (2000), *XML by example*, Que

Natanya, Pitts, *XML*, Anaya Multimedia (Temas Profesionales)

[23] *Unified Modeling Language* (Lenguaje unificado de modelado). Lenguaje para especificar y visualizar diseños orientados a objeto. Unifica los lenguajes de Booch, OMT y OOSE.

Netscape Communications Corporation, “Dynamic HTML in Netscape Communicator”, en http://developer.netscape.com/docs/manuals/communicator/dynhtml/index.htm (accedido el 17/8/1999)

Grosso, Paul; Walsh, Norman (6/7/2000), “XSL Concepts and Practical Use”, en http://www.nwalsh.com/docs/tutorials/xsl/xsl/frames.html (accedido el 26/3/2001)

Walsh, Norman (3/10/1998), “A Technical Introduction to XML”, en http://www.xml.com/pub/a/98/10/guide0.html (accedido el 26/3/2001)

W3C (6/10/2000), “Extensible Markup Language (XML) 1.0 (Second Edition)”, en http://www.w3.org/TR/REC-xml (accedido el 26/3/2001)

W3C (16/11/1999), “XML Path Language (XPath) Version 1.0”, en http://www.w3.org/TR/xpath (accedido el 26/3/2001)

W3C (21/11/2000), “Extensible Stylesheet Language (XSL) Version 1.0”, en http://www.w3.org/TR/xsl/ (accedido el 26/3/2001)

W3C (16/11/1999), “XSL Transformations (XSLT) Version 1.0”, en http://www.w3.org/TR/xslt (accedido el 26/3/2001)

4GuysFromRolla.com, “XML Articles on 4Guys and the Web!”, en http://www.4guysfromrolla.com/webtech/xml.shtml (accedido el 26/3/2001)