UT4. Definición de esquemas y vocabularios en XML

ÍNDICE

1 ¿Qué es XML?

XML (eXtensible Markup Language, Lenguaje de Marcado eXtensible) es un lenguaje desarrollado por W3C (World Wide Web Consortium) que está basado en SGML (Standard Generalized Markup Language, Lenguaje de Marcado Generalizado Estándar).

XML es un lenguaje utilizado para el almacenamiento e intercambio de datos estructurados entre distintas plataformas.

XML es un metalenguaje, es decir, puede ser empleado para definir otros lenguajes, llamados dialectos XML. Por ejemplo, algunos lenguajes basados en XML son:

- GML (Geography Markup Language, Lenguaje de Marcado Geográfico).
- MathML (Mathematical Markup Language, Lenguaje de Marcado Matemático).
- RSS (Really Simple Syndication, Sindicación Realmente Simple).
- SVG (Scalable Vector Graphics, Gráficos Vectoriales Escalables).
- XHTML (eXtensible HyperText Markup Language, Lenguaje de Marcado de Hipertexto eXtensible).

1.1 Elementos en XML

Los documentos XML están formados por texto plano (sin formato) y contienen marcas (etiquetas) definidas por el desarrollador. Dichas marcas, es recomendable que sean lo más descriptivas posible y, para escribirlas, se utilizan los caracteres menor que "**<**", mayor que "**>**" y barra inclinada "**/**".

EJEMPLO Si en un documento XML se quiere guardar el nombre **Javier**, se puede escribir:

```
<nombre>Javier</nombre>
```

La sintaxis utilizada en el ejemplo es la básica para escribir un elemento en XML:

```
<etiqueta>valor</etiqueta >
```

Obsérvese que, entre la etiqueta de inicio (**<nombre>**) y la etiqueta de fin (**</nombre>**) se ha escrito el dato (**valor**) que se quiere almacenar. En este caso **Javier**.

1.1.1 Elementos vacíos

En un documento XML, un **elemento puede no contener ningún valor**. En tal caso hay que escribir:

```
<etiqueta></etiqueta>
```

Se puede expresar lo mismo escribiendo:

```
<etiqueta/>
```

EJEMPLO Para escribir el elemento "nombre" vacío, se puede escribir:

```
<nombre></nombre>
```

O también:

```
<nombre/>
```

1.1.2 Relaciones padre-hijo entre elementos

EJEMPLO Por otra parte, un elemento (padre) puede contener a otro u otros elementos (hijos):

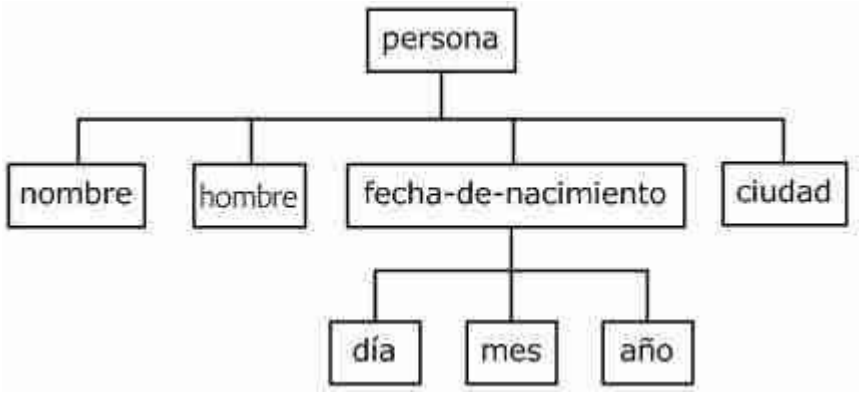
```
<persona>
   <nombre>Javier</nombre>
   <hombre/>
   <fecha-de-nacimiento>
      <día>6</día>
      <mes>2</mes>
      <año>1987</año>
   </fecha-de-nacimiento>
   <ciudad>Mérida</ciudad>
</persona>
```

En este ejemplo, el elemento "persona" contiene cuatro elementos (hijos): "nombre", "mujer", "fecha de nacimiento" y "ciudad". A su vez, el elemento "fecha de nacimiento" contiene otros tres elementos (hijos): "día", "mes" y "año".

Véase que, de todos los elementos que aparecen en este ejemplo, sólo el elemento "hombre" está vacío.

1.1.3 Elemento raíz de un documento XML

Todo documento XML tiene que tener un único elemento raíz (padre) del que desciendan todos los demás. En este caso, el elemento raíz es "persona". Gráficamente, la estructura de elementos de este documento se puede representar como se muestra a continuación:



De esta forma, la estructura de cualquier documento XML se puede representar como un árbol invertido de elementos. Se dice que los elementos son los que dan estructura semántica al documento.

1.1.4 Elementos con contenido mixto

EJEMPLO Un elemento puede contener contenido mixto, es decir, texto y otros elementos:

```
<persona>
   <nombre>Javier</nombre> vive en <ciudad>Mérida</ciudad>.
</persona>
```

En este ejemplo, el elemento "persona" contiene los elementos "nombre" y "ciudad", además de los textos " **vive en** " y "**.**".

1.2 Normas de sintaxis básicas en XML

En un documento XML, todos los nombres de **los elementos son case sensitive**, es decir, **sensibles a letras minúsculas y mayúsculas**, teniendo que cumplir las siguientes normas:

- Pueden contener letras minúsculas, letras mayúsculas, números, puntos ".", guiones medios "-" y guiones bajos "_".
- Asimismo, pueden contener el carácter dos puntos ":". **No obstante, su uso se reserva para cuando se definan espacios de nombres.**
- **El primer carácter tiene que ser una letra o un guion bajo "_".**

Por otra parte, hay que tener en cuenta que, detrás del nombre de una etiqueta se permite escribir un espacio en blanco o un salto de línea. Por ejemplo, sintácticamente es correcto escribir:

```
<ciudad >Pamplona</ciudad
>
```

Ahora bien, **no puede haber un salto de línea o un espacio en blanco antes del nombre de una etiqueta:**

```
<
```

```
ciudad>Pamplona</ ciudad>
```

Los siguientes elementos no están escritos correctamente por incumplir alguna regla de sintaxis:

```
<Ciudad>Mérida</ciudad>

<día>6</dia>

<mes>2<mes/>

<ciudad>PamplonMérida</finciudad>

<_rojo>

<2colores>Rojo y Naranja</2colores>

< Aficiones >Cine, Bailar, Nadar</ Aficiones >

<persona><nombre>Javier</persona></nombre>

<color favorito>azul</color favorito>
```

En su lugar, sería correcto escribir:

```
<Ciudad>Mérida</Ciudad>

<día>6</día>

<mes>2</mes>

<ciudad>Mérida</ciudad>

<_rojo/>

<colores2>Rojo y Naranja</colores2>

<Aficiones >Cine, Bailar, Nadar</Aficiones >

<persona><nombre>Javier</nombre></persona>

<color.favorito>azul</color.favorito>

<color-favorito>azul</color-favorito>

<color_favorito>azul</color_favorito>
```

Las letras no inglesas (á, Á, ñ, Ñ...) están permitidas. Sin embargo, **es recomendable no utilizarlas** para reducir posibles incompatibilidades con programas que puedan no reconocerlas.

En cuanto al carácter guion medio "-" y al punto ".", aunque también están

permitidos para nombrar etiquetas, igualmente **se aconseja evitar su uso**; el guion medio porque podría confundirse con el signo menos, y el punto porque, por ejemplo al escribir color.favorito, podría interpretarse que favorito es una propiedad del objeto color.

1.3 Atributos en XML

Los elementos de un documento XML pueden tener atributos definidos en la etiqueta de inicio. **Un atributo sirve para proporcionar información extra sobre el elemento que lo contiene**.

EJEMPLO Dados los siguientes datos de un producto:

- Código: **G45**
- Nombre: **Gorro de lana**
- Color: **negro**
- Precio: **12.56**

Su representación en un documento XML podría ser, por ejemplo:

```
<producto codigo="G45">
   <nombre color="negro" precio="12.56">Gorro de lana</nombre>
</producto>
```

En este ejemplo se han escrito tres atributos: **codigo**, **color** y **precio**. Obsérvese que, sus valores ("G45", "negro" y "12.56") se han escrito entre comillas dobles ("). No obstante, también pueden ir entre comillas simples (').

Si, por ejemplo, el atributo codigo se quisiera representar como un elemento, se podría escribir:

```
<producto>
   <codigo>G45</codigo>
   <nombre color="negro" precio="12.56">Gorro de lana</nombre>
</producto>
```

Como se puede apreciar, ahora el valor del código no se ha escrito entre comillas dobles.

1.3.1 Normas de sintaxis

EJEMPLO Los nombres de los atributos deben cumplir las mismas normas de sintaxis que los nombres de los elementos. Además, **todos los atributos de un elemento tienen que ser únicos**. Por ejemplo**, es incorrecto escribir**:

```
<datos x="3" x="4" y="5"/>
```

Sin embargo, sí es correcto escribir:

```
<datos x="3" X="4" y="5"/>
```

Los atributos contenidos en un elemento, como en este caso x, X e y, deben separarse con espacios en blanco, no siendo significativo su orden.

1.4 Declaración XML

La declaración XML que se puede escribir al principio de un documento XML, empieza con los caracteres "**<?**" y termina con "**?>**" al igual que las instrucciones de procesamiento. Sin embargo, la declaración XML no es una instrucción de procesamiento (o proceso).

1.4.1 Atributos version y encoding

EJEMPLO Un documento XML podría contener la siguiente declaración XML:

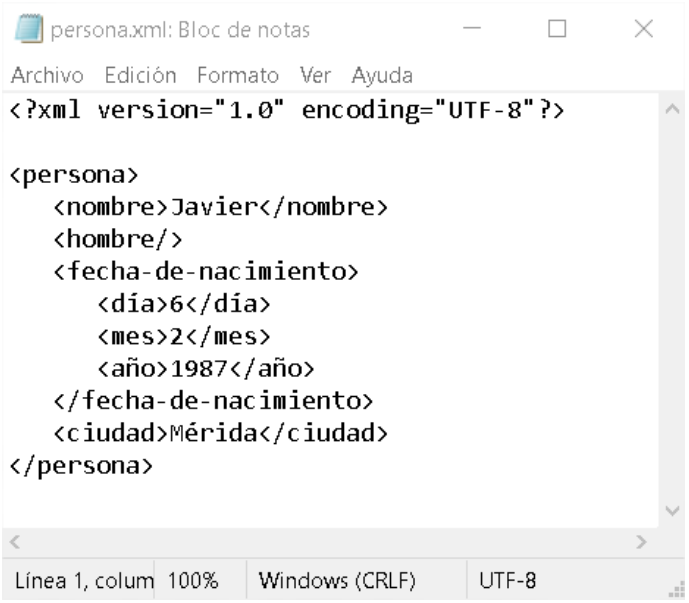
```
<?xml version="1.0" encoding="UTF-8"?>
```

En esta declaración XML, se está indicando que 1.0 es la versión de XML utilizada en el documento y UTF-8 (8-bit Unicode Transformation Format, Formato de Transformación Unicode de 8 bits) es la codificación de caracteres empleada.

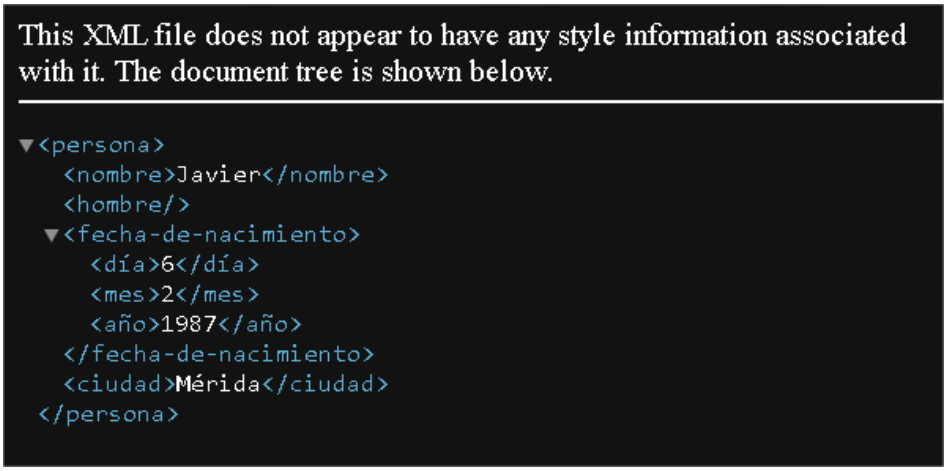
En un documento XML no es obligatorio que aparezca la declaración XML. Ahora bien, si se incluye, tiene que aparecer en la primera línea del documento, y el carácter "**<**" debe ser el primero de dicha línea, es decir, **antes no pueden aparecer espacios en blanco**.

1.4.2 Cómo crear un documento XML

EJEMPLO Si, por ejemplo, en el Bloc de notas de Microsoft Windows escribimos y guardamos (codificado en UTF-8) un archivo llamado "persona.xml" como se muestra en la siguiente imagen:



Al visualizar dicho archivo en un navegador web, como por ejemplo Google Chrome, se podrá ver algo parecido a:



1.5 Instrucciones de procesamiento en XML

En un documento XML, **una instrucción de procesamiento (*processing instruction*) sirve para indicar que se procese un documento.** Las instrucciones de proceso se escriben empezando con la pareja de caracteres "**<?**" y finalizando con "**?>**".

1.5.1 Cómo asociar un archivo CSS a un documento XML

EJEMPLO En un documento XML podría escribirse, por ejemplo, la siguiente instrucción de procesamiento:

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

Esta instrucción sirve para asociar el archivo **CSS** (*Cascading Style Sheets*, Hojas de Estilo en Cascada) ***"estilo-animales.css"*** al documento XML. Dicho archivo podría contener, por ejemplo, el siguiente código:

```
nombre{color:blue;font-size:40px}
patas{color:red;font-size:22px}
```

De forma que, dado por ejemplo el archivo ***"animales.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="estilo-animales.css"?>
<animales>
   <animal>
      <nombre>perro</nombre>
      <patas>4</patas>
   </animal>
   <animal>
      <nombre>pato</nombre>
      <patas>2</patas>
   </animal>
   <animal>
      <nombre>ballena</nombre>
      <patas>0</patas>
```

```
    </animal>
</animales>
```

En un navegador web se verá algo parecido a:

En un documento XML, **no es obligatorio que aparezcan instrucciones de procesamiento**.

1.6 Referencias a entidades en XML

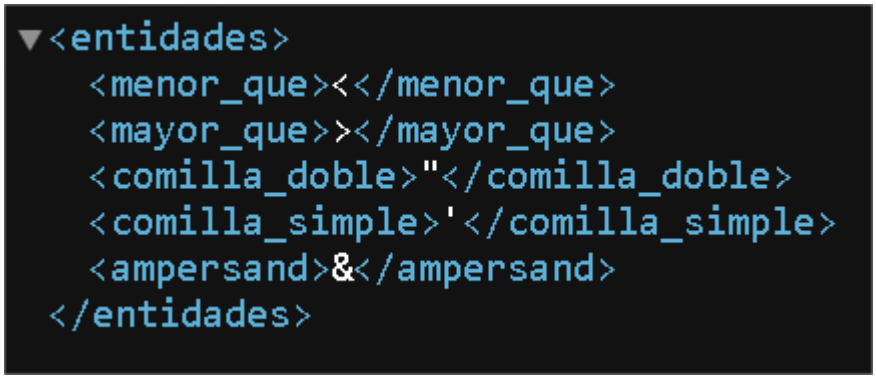
En XML existen algunos caracteres que son especiales por su significado y, para escribirlos en un documento XML, se pueden utilizar las referencias a entidades mostradas en la siguiente tabla:

Carácter	***Entidad***	***Referencia a entidad***
< (*menor que*)	**lt** (*less than*)	**<**
> (*mayor que*)	**gt** (*greater than*)	**>**
" (*comilla doble*)	**quot** (*quotation mark*)	**"**
' (*comilla simple*)	**apos** (*apostrophe*)	**'**
& (*ampersand*)	**amp** (*ampersand*)	**&**

EJEMPLO Dado el archivo "entidades.xml":

```
<?xml version="1.0" encoding="UTF-8"?>
<entidades>
   <menor_que>&lt;</menor_que>
   <mayor_que>&gt;</mayor_que>
   <comilla_doble>&quot;</comilla_doble>
   <comilla_simple>&apos;</comilla_simple>
   <ampersand>&amp;</ampersand>
</entidades>
```

Al abrirlo en Google Chrome se podrá visualizar:

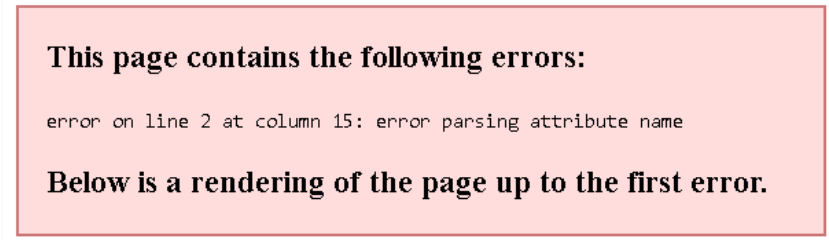


En el navegador web, se puede ver que donde se han escrito las referencias a entidades en el documento XML (por ejemplo **<**), se muestran los caracteres correspondientes (por ejemplo **<**).

1.6.1 Caracteres problemáticos en XML: menor que (<) y ampersand (&)

En un documento XML, **el carácter "<" es problemático porque indica el comienzo de una etiqueta.** Por tanto, en vez de escribir, por ejemplo:

```
<condicion>a<b</condicion>
```



Habría que utilizar la referencia a entidad **<** escribiendo:

```
<condicion>a&lt;b</condicion>
```

El carácter "**>**" sí puede utilizarse en el texto contenido en un elemento, no siendo incorrecto escribir, por ejemplo:

```
<condicion>a>b</condicion>
```

Ahora bien, **se recomienda hacer uso de su referencia a entidad** (**>**).

En un documento XML, **el carácter ampersand "&" también es problemático, ya que se utiliza para indicar el comienzo de una referencia a entidad.** Por ejemplo, no es correcto escribir:

```
<condicion>a==1 && b==2</condicion>
```

En su lugar se debe escribir lo siguiente:

```
<condicion>a==1 &amp;&amp; b==2</condicion>
```

1.6.2 Uso de la comilla doble (") y de la comilla simple (') en atributos

Si el valor de un atributo se escribe entre comillas dobles (**"**), dicho valor no podrá contener dicho carácter. Por ejemplo, no es correcto escribir:

```
<dato caracter="comilla doble(")"/>
```

Para ello, hay que utilizar la referencia a entidad **"** como se muestra a continuación:

<dato caracter="comilla doble(**"**)"/>

De igual modo ocurre con la comilla simple (**'**), siendo incorrecto escribir, por ejemplo:

```
<dato caracter="comilla doble(&quot;)"/>
```

Por lo que, en este caso, habría que usar **'** como se muestra seguidamente:

```
<dato caracter='comilla simple(&apos;)'/>
```

Por otro lado, **los valores de atributos escritos entre comillas dobles (") sí pueden contener al carácter comilla simple (') y a la inversa**. Por ejemplo, es correcto escribir:

```
<dato caracter="comilla simple(')"/>
<dato caracter='comilla doble(")'/>
```

En estos casos, no es obligatorio usar las referencias a entidades, pero sí recomendable.

1.7 Comentarios en XML

Para escribir comentarios en un documento XML, estos deben escribirse entre los caracteres "**<!--**" y "**-->**". Por ejemplo:

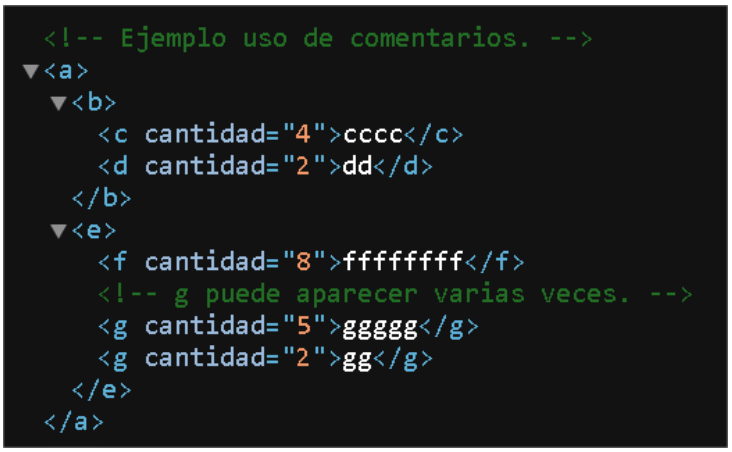
```
<!-- Esto es un comentario escrito en un documento XML -->
```

EJEMPLO Dado el archivo XML ***"letras.xml"***:

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

<!--Ejemplo uso de comentarios.-->

<a>
   <b>
      <c cantidad="4">cccc</c>
      <d cantidad="2">dd</d>
   </b>
   <e>
      <f cantidad="8">ffffffff</f>

      <!--g puede aparecer varias veces.-->
```

```
      <g cantidad="5">ggggg</g>
      <g cantidad="2">gg</g>
   </e>
</a>
```

En un navegador se verá:



Obsérvese que los comentarios son visibles.

En un documento XML, **no se pueden escribir comentarios dentro de las etiquetas**. Por ejemplo, **no es correcto escribir**:

```
<mujer <!-- elemento vacío --> />
```

Por otro lado, hay que tener en cuenta **que en los comentarios de un documento XML no está permitido usar dos guiones seguidos**:

```
<!-- Dos guiones seguidos -- en un comentario da error -->
```

De forma que, no es posible anidar comentarios en un documento XML.

1.8 Secciones CDATA en XML

Un documento XML puede contener secciones **CDATA** (*Character DATA*) para escribir texto que no se desea que sea analizado. Por ejemplo, esto puede ser útil cuando se quiere escribir texto que contenga alguno de los caracteres problemáticos: *menor que* "**<**" o *ampersand* "**&**".

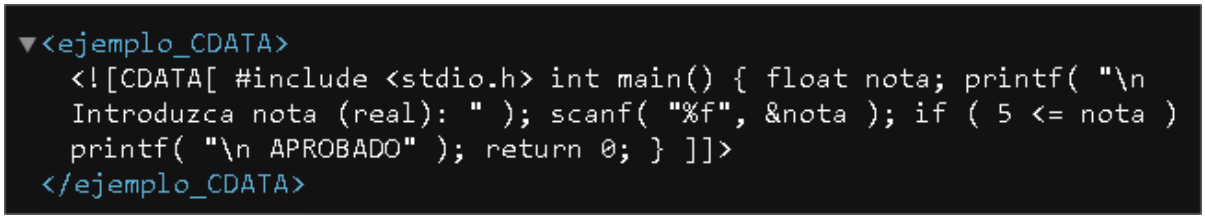
En un documento XML, para incluir una sección CDATA, esta se escribe comenzando con la cadena de caracteres "**<![CDATA[**" y terminando con los caracteres "**]]>**".

Una sección CDATA puede contener, por ejemplo, el código fuente de un programa escrito en lenguaje C:

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo_CDATA>
<![CDATA[
```

```
#include <stdio.h>
int main()
{
   float nota;
   printf( "\n   Introduzca nota (real): " );
   scanf( "%f", &nota );
   if ( 5 <= nota )
      printf( "\n   APROBADO" );
   return 0;
}
]]>
</ejemplo_CDATA>
```

En un navegador web se visualizará algo parecido a:



Dentro de una sección CDATA no se puede escribir la cadena "**]]>**". En consecuencia, **no se pueden anidar secciones CDATA.**

Por otra parte, no está permitido escribir espacios en blanco o saltos de línea en las cadenas de inicio "**<![CDATA[**" o fin "**]]>**" de una sección CDATA.

1.9 Espacios de nombres en XML

Varios documentos XML se pueden combinar entre sí, pudiendo en estos casos coincidir el nombre de algunos elementos.

Dos documentos XML podrían contener un elemento llamado "carta", pero con significados distintos.

```
<carta>
   <palo>Corazones</palo>
   <numero>7</numero>
</carta>
```

```
<carta>
   <carnes>
      <filete_de_ternera precio="12.95"/>
      <solomillo_a_la_pimienta precio="13.60"/>
   </carnes>
   <pescados>
      <lenguado_al_horno precio="16.20"/>
      <merluza_en_salsa_verde precio="15.85"/>
   </pescados>
</carta>
```

1.9.1 Uso de espacios de nombres

De forma que, si se incluyen ambos elementos **<carta>** en un documento XML, se origina un conflicto de nombres. Para resolverlo, se pueden utilizar espacios de nombres (*XML Namespaces*). Por ejemplo, escribiendo:

```
<?xml version="1.0" encoding="UTF-8"?>
<cartas
xmlns:e1="http://www.example.com/ejemplo1"
xmlns:e2="http://www.example.com/ejemplo2">

   <e1:carta>
      <e1:palo>Corazones</e1:palo>
      <e1:numero>7</e1:numero>
   </e1:carta>

   <e2:carta>
      <e2:carnes>
         <e2:filete_de_ternera precio="12.95"/>
         <e2:solomillo_a_la_pimienta precio="13.60"/>
      </e2:carnes>
      <e2:pescados>
         <e2:lenguado_al_horno precio="16.20"/>
         <e2:merluza_en_salsa_verde precio="15.85"/>
      </e2:pescados>
   </e2:carta>

</cartas>
```

1.9.2 Sintaxis para definir un espacio de nombres

Para definir un espacio de nombres se utiliza la siguiente sintaxis:

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

En el ejemplo, obsérvese que, **xmlns** es un atributo que se ha utilizado en la etiqueta de inicio del elemento **<cartas>** y, en este caso, se han definido dos espacios de nombres que hacen referencia a los siguientes **URI** (*Uniform Resource Identifier*, Identificador Uniforme de Recurso):

- **http://www.example.com/ejemplo1**
- **http://www.example.com/ejemplo2**

Los prefijos definidos son **e1** y **e2**, respectivamente. Véase que, se han añadido dichos prefijos a las etiquetas que aparecen en el documento:**<e1:carta>**, **<e2:carta>**, **<e1:palo>**, etc.

Los URI especificados en un documento XML no tienen por qué contener nada, su función es ser únicos. No obstante, en un URI se puede mostrar información si se considera oportuno. Véase, por ejemplo:

1.10 Espacios en blanco en XML

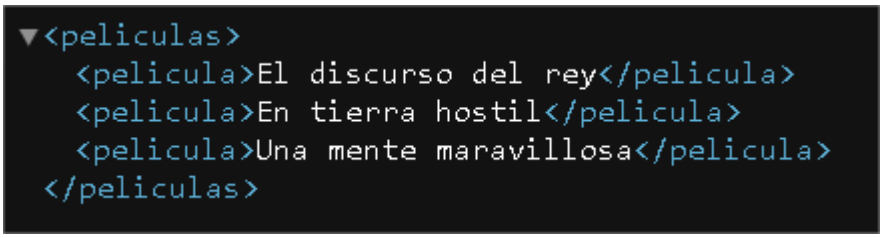
En un documento XML, los espacios en blanco, las tabulaciones y los retornos de carro pueden ser tratados de un modo especial.

Espacios en blanco en el contenido (texto) de un elemento

Dado el archivo XML ***"peliculas.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<peliculas>
   <pelicula>El discurso del rey</pelicula>
   <pelicula>En   tierra          hostil</pelicula>
   <pelicula>Una
      mente

maravillosa</pelicula>
</peliculas>
```

Al visualizar dicho archivo en *Google Chrome*, se verá algo parecido a:



Esto es debido a que, las tabulaciones, los retornos de carro y varios espacios en blanco contenidos en el texto de los elementos del documento, han sido representados como un único espacio en blanco.

1.11 Documentos XML bien formados y válidos

Se dice que **un documento XML está bien formado cuando no tiene errores de sintaxis.** Esto incluye los siguientes aspectos:

- Los nombres de los elementos y sus atributos deben estar escritos correctamente.
- Los valores de los atributos deben estar escritos entre comillas dobles o simples.
- Los atributos de un elemento deben separarse con espacios en blanco.
- Se tienen que utilizar referencias a entidades donde sea necesario.
- Tiene que existir un único elemento raíz.
- Todo elemento debe tener un elemento padre, excepto el elemento raíz.
- Todos los elementos deben tener una etiqueta de apertura y otra de cierre.
- Las etiquetas deben estar correctamente anidadas.
- Las instrucciones de proceso deben estar escritas de forma correcta.
- La declaración XML debe estar en la primera línea escrita correctamente.

- Las secciones CDATA y los comentarios deben estar correctamente escritos.

Por otro lado, se dice que **un documento XML es válido (*valid*) cuando, además de no tener errores de sintaxis, no incumple ninguna de las normas establecidas en su estructura.** Dicha estructura se puede definir utilizando distintos métodos, tales como:

- **DTD** (*Document Type Definition*, Definición de Tipo de Documento).
- **XML Schema**.
- **RELAX NG** (*REgular LAnguage for XML Next Generation*).

2 Qué es DTD

DTD (*Document Type Definition*, Definición de Tipo de Documento) sirve para definir la estructura de un documento SGML o XML, permitiendo su validación.

Un documento XML es válido cuando, además de estar bien formado, no incumple ninguna de las normas establecidas en su estructura.

Existen otros métodos que también permiten validar documentos XML, como por ejemplo ***XML Schema*** o *RELAX NG*.

2.1 Declaración de tipo de documento

Una DTD se puede escribir tanto **interna** como **externamente** a un archivo XML. Ahora bien, en ambos casos hay que escribir una definición **DOCTYPE** (*Document Type Declaration*, Declaración de Tipo de Documento) para asociar el documento XML a la DTD. Asimismo, un archivo XML se puede asociar simultáneamente a una DTD interna y externa.

2.1.1 Documento XML asociado a una DTD interna

La sintaxis para escribir una DTD interna es:

```
<!DOCTYPE elemento-raíz [ declaraciones ]>
```

En un documento XML se quiere guardar una lista de marcadores de páginas web, almacenando de cada uno de ellos su nombre, una descripción y su URL. Para ello, se puede escribir, por ejemplo, el archivo ***"marcadores-con-dtd-interna.xml"*** siguiente, que contiene una DTD interna:

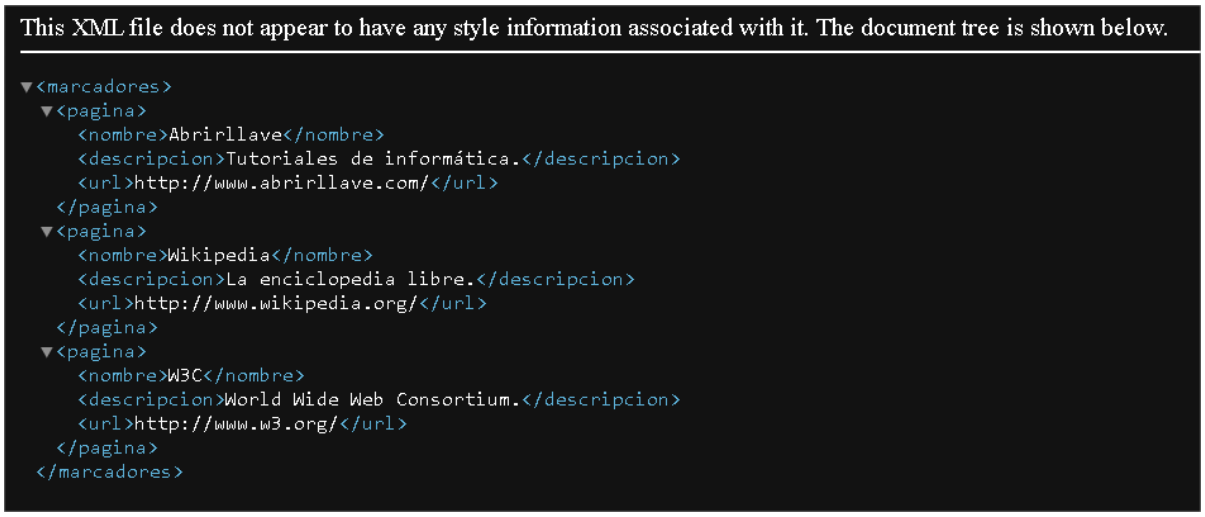
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE marcadores [
   <!ELEMENT marcadores (pagina)*>
   <!ELEMENT pagina (nombre, descripcion, url)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT descripcion (#PCDATA)>
   <!ELEMENT url (#PCDATA)>
]>

<marcadores>
   <pagina>
      <nombre>Abrirllave</nombre>
      <descripcion>Tutoriales de informática.</descripcion>
      <url>http://www.abrirllave.com/</url>
   </pagina>
   <pagina>
      <nombre>Wikipedia</nombre>
      <descripcion>La enciclopedia libre.</descripcion>
      <url>http://www.wikipedia.org/</url>
   </pagina>
   <pagina>
      <nombre>W3C</nombre>
      <descripcion>World Wide Web Consortium.</descripcion>
      <url>http://www.w3.org/</url>
```

```
    </pagina>
</marcadores>
```

- Obsérvese que, en la DTD se ha indicado que **marcadores** es el elemento raíz del documento XML, el cual puede contener cero o más páginas. Para indicar esto último, se ha escrito: **(pagina)***
- Escribiendo **pagina (nombre, descripcion, url)** se especifica que, cada elemento "pagina" tiene que contener tres elementos (hijos): "nombre", "descripcion" y "url".
- Con **#PCDATA** (*Parsed Character Data*) escrito entre paréntesis "**()**" se indica que los elementos "nombre", "descripcion" y "url" pueden contener texto (cadenas de caracteres) analizable por un procesador XML.

Nota: al nombrar a los elementos "pagina" y "descripcion" no se han utilizado los caracteres (á) y (ó), respectivamente, para evitar posibles incompatibilidades con programas que puedan no reconocerlos.

Resultado:



Como se puede observar, la DTD no se muestra en el navegador.

2.1.2 Documento XML asociado a una DTD externa

Existen dos tipos de DTD externa: privada y pública. Para las privadas se utiliza **SYSTEM** y para las públicas **PUBLIC**. La sintaxis en cada caso es:

```
<!DOCTYPE elemento-raíz SYSTEM "URI">
```

```
<!DOCTYPE elemento-raíz PUBLIC "identificador-público" "URI">
```

DTD externa privada - SYSTEM

Si en un archivo llamado ***"marcadores.dtd"*** se escribiese la siguiente DTD:

```
<!ELEMENT marcadores (pagina)*>
<!ELEMENT pagina (nombre, descripcion, url)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT descripcion (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

El siguiente documento XML llamado ***"marcadores-con-dtd-externa.xml"***, sería válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE marcadores SYSTEM "marcadores.dtd">

<marcadores>
   <pagina>
      <nombre>Abrirllave</nombre>
      <descripcion>Tutoriales de informática.</descripcion>
      <url>http://www.abrirllave.com/</url>
   </pagina>
   <pagina>
      <nombre>Wikipedia</nombre>
      <descripcion>La enciclopedia libre.</descripcion>
      <url>http://www.wikipedia.org/</url>
   </pagina>
   <pagina>
      <nombre>W3C</nombre>
      <descripcion>World Wide Web Consortium.</descripcion>
      <url>http://www.w3.org/</url>
   </pagina>
</marcadores>
```

- En este documento XML, haciendo uso de una DTD externa privada, se ha escrito una lista de marcadores de páginas web, guardando de cada uno de ellos su nombre, una descripción y su URL.

DTD externa pública - PUBLIC

El siguiente documento XML está asociado a una DTD externa pública:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
   <head>
      <title>Título</title>
   </head>
   <body>
      <p>Párrafo</p>
   </body>
</html>
```

- **-//W3C//DTD XHTML 1.0 Strict//EN** es un FPI (*Formal Public Identifier*, Identificador Público Formal).

¿Cuándo utilizar una DTD interna o una DTD externa?

Para validar más de un documento XML con la misma DTD, escribir esta en un archivo externo proporciona la ventaja de no tener que repetir la DTD internamente en cada documento XML.

En el caso de que la DTD solo se utilice para validar un único documento XML, la DTD es habitual escribirla internamente.

2.2 Elementos - Contenido

Para declarar un elemento en una DTD se utiliza la siguiente sintaxis:

```
<!ELEMENT nombre-del-elemento tipo-de-contenido>
```

En el tipo de contenido se especifica el contenido permitido en el elemento, pudiendo ser:

- Texto, (#PCDATA).
- Otros elementos (hijos).
- Estar vacío, EMPTY.
- Mixto (texto y otros elementos), ANY

2.2.1 Texto - (#PCDATA)

En el siguiente documento XML, el elemento "ciudad" puede contener cualquier texto (cadena de caracteres):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudad [
   <!ELEMENT ciudad (#PCDATA)>
]>

<ciudad>Roma</ciudad>
```

- Escribiendo **#PCDATA** (*Parsed Character Data*) entre paréntesis "**()**", se ha indicado que el elemento "ciudad" puede contener una cadena de caracteres analizable.

2.2.2 Otros elementos

En el siguiente ejemplo, el elemento "ciudad" contiene a los elementos "nombre" y "pais":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudad [
   <!ELEMENT ciudad (nombre, pais)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT pais (#PCDATA)>
```

```
]>

<ciudad>
  <nombre>Roma</nombre>
  <pais>Italia</pais>
</ciudad>
```

2.2.3 Vacío - EMPTY

En la DTD interna del siguiente documento XML, se ha declarado el elemento "mayor_de_edad" como vacío, **EMPTY**. Por tanto, debe escribirse **<mayor_de_edad/>**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, mayor_de_edad, ciudad)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT mayor_de_edad EMPTY>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Elsa</nombre>
   <mayor_de_edad/>
   <ciudad>Pamplona</ciudad>
</persona>
```

O también:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, mayor_de_edad, ciudad)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT mayor_de_edad EMPTY>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Elsa</nombre>
   <mayor_de_edad></mayor_de_edad>
   <ciudad>Pamplona</ciudad>
</persona>
```

Nota: los elementos vacíos no pueden tener contenido, pero sí pueden tener atributos.

2.2.4 Mixto - ANY

En la DTD interna del siguiente documento XML, se ha indicado que el elemento "persona" puede contener texto y otros elementos, es decir, contenido mixto, **ANY**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
```

```
   <!ELEMENT persona ANY>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Elsa</nombre> vive en <ciudad>Pamplona</ciudad>.
</persona>
```

Obsérvese que, por ejemplo, también sería válido el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona ANY>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Elsa</nombre> vive en Pamplona.
</persona>
```

O el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona ANY>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Elsa</nombre>
</persona>
```

Incluso, si el elemento "persona" estuviese vacío, el documento también sería válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona ANY>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona/>
```

2.3 Elementos - Secuencias

En una DTD, **un elemento (padre) puede ser declarado para contener a otro u otros elementos (hijos)**. En la sintaxis, los hijos –también llamados sucesores– tienen que escribirse entre paréntesis "**()**" y separados por comas "**,**".

Para declarar un elemento (padre) que contenga tres elementos (hijos), se puede escribir:

```
<!ELEMENT padre (hijo1, hijo2, hijo3)>
```

En el siguiente documento XML, el elemento "persona" contiene a los elementos "nombre", "fecha_de_nacimiento" y "ciudad":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT fecha_de_nacimiento (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Iker</nombre>
   <fecha_de_nacimiento>26-12-1997</fecha_de_nacimiento>
   <ciudad>Valencia</ciudad>
</persona>
```

A su vez, los hijos también pueden tener sus propios hijos. Así, el elemento "fecha_de_nacimiento" puede contener, por ejemplo, a los elementos "dia", "mes" y "anio":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT fecha_de_nacimiento (dia, mes, anio)>
   <!ELEMENT dia (#PCDATA)>
   <!ELEMENT mes (#PCDATA)>
   <!ELEMENT anio (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Iker</nombre>
   <fecha_de_nacimiento>
      <dia>26</dia>
      <mes>12</mes>
      <anio>1997</anio>
   </fecha_de_nacimiento>
   <ciudad>Valencia</ciudad>
</persona>
```

Nota: En un documento XML**, los elementos (hijos) de un elemento (padre), deben escribirse en el mismo orden en el que han sido declarados en la DTD.**

El siguiente documento **XML no es válido:**

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

```
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT fecha_de_nacimiento (dia, mes, anio)>
   <!ELEMENT dia (#PCDATA)>
   <!ELEMENT mes (#PCDATA)>
   <!ELEMENT anio (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
   <nombre>Iker</nombre>
   <fecha_de_nacimiento>
      <anio>1997</anio>
      <mes>12</mes>
      <dia>26</dia>
   </fecha_de_nacimiento>
   <ciudad>Valencia</ciudad>
</persona>
```

- El documento no es válido porque los elementos sucesores (hijos) del elemento "fecha_de_nacimiento" no se han escrito en el mismo orden que en la DTD.

2.4 Elementos - Cardinalidad

En una DTD, para definir el número de veces que pueden aparecer los elementos de un documento XML, se pueden utilizar los *operadores de cardinalidad* mostrados en la siguiente tabla:

Operadores de cardinalidad en DTD

Operador	***Cardinalidad***	***Significado***
? (*interrogación*)	0-1	El elemento es **opcional**, pudiendo aparecer una sola vez o ninguna.
***** (*asterisco*)	0-n	El elemento **puede aparecer** cero, una o más veces.
+ (*signo más*)	1-n	El elemento tiene que aparecer, **obligatoriamente**, una o más veces.

Nota: los elementos declarados en una DTD sobre los que no actúe ningún operador de cardinalidad, **tendrán que aparecer obligatoriamente una única vez**, en el o los documentos XML a los que se asocie.

2.4.1 Cardinalidad 1-n "+"

En el siguiente documento XML, el elemento "nombre" tiene que aparecer una o

más veces. En este caso, aparece tres veces:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE personas [
   <!ELEMENT personas (nombre+)>
   <!ELEMENT nombre (#PCDATA)>
]>

<personas>
   <nombre>Ana</nombre>
   <nombre>Iker</nombre>
   <nombre>Elsa</nombre>
</personas>
```

- Si sobre **nombre** no actuase el operador (**+**) el documento no sería válido, ya que, el elemento "personas" solo tendría que contener un elemento "nombre".
- En vez de **(nombre+)**, también se puede escribir **(nombre)+**.

2.4.2 Cardinalidad 0-n "*"

En la DTD interna del siguiente documento XML, se ha indicado que el elemento "nombre" tiene que aparecer una única vez. Ahora bien, el elemento "ingrediente" tiene cardinalidad (0-n), por tanto, puede aparecer cero, una o más veces:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE receta_de_cocina [
   <!ELEMENT receta_de_cocina (nombre, ingrediente*)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT ingrediente (#PCDATA)>
]>

<receta_de_cocina>
   <nombre>Tortilla de patatas</nombre>
   <ingrediente>Huevo</ingrediente>
   <ingrediente>Patata</ingrediente>
   <ingrediente>Aceite</ingrediente>
   <ingrediente>Sal</ingrediente>
</receta_de_cocina>
```

2.4.3 Cardinalidad 0-1 "?"

En la DTD del siguiente documento XML, la cardinalidad del elemento "mayor_de_edad" es (0-1), siendo opcional su aparición:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, mayor_de_edad?)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT mayor_de_edad EMPTY>
]>
```

```
<persona>
   <nombre>Iker</nombre>
   <mayor_de_edad/>
</persona>
```

Así pues, el siguiente documento también es válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
   <!ELEMENT persona (nombre, mayor_de_edad?)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT mayor_de_edad EMPTY>
]>

<persona>
   <nombre>Iker</nombre>
</persona>
```

2.5 Elementos - Opcionales

En la DTD asociada a un documento XML, se pueden declarar elementos que contengan elementos opcionales. Para ello, se utiliza el *operador de elección*, representado por una barra vertical (**|**).

En el siguiente documento XML el elemento "articulo" puede contener un elemento "codigo" o un elemento "id"; **obligatoriamente uno de ellos, pero no ambos**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulo [
   <!ELEMENT articulo (codigo | id)>
   <!ELEMENT codigo (#PCDATA)>
   <!ELEMENT id (#PCDATA)>
]>

<articulo>
   <codigo>AF-33</codigo>
</articulo>
```

2.5. Elección "|" y operador "*"

En la DTD del siguiente documento XML se indica que el elemento "articulos" puede contener varios elementos "codigo" e "id":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulos [
   <!ELEMENT articulos (codigo | id)*>
   <!ELEMENT codigo (#PCDATA)>
   <!ELEMENT id (#PCDATA)>
]>

<articulos>
```

```
   <codigo>AF-32</codigo>
   <id>3891</id>
   <codigo>AF-50</codigo>
   <codigo>AF-89</codigo>
</articulos>
```

Obsérvese que, con el operador "*****", en este ejemplo se ha indicado que el contenido del elemento "articulos" tiene cardinalidad (0-n). Por tanto, el elemento "articulos" puede:

- Estar vacío.
- Contener un elemento "codigo".
- Contener un elemento "id".
- Contener un elemento "codigo" y un elemento "id".
- Contener un elemento "codigo" y varios elementos "id".
- Contener un elemento "id" y varios elementos "codigo".
- Contener varios elementos "codigo" y varios elementos "id".

Nótese también que, dentro del elemento "articulos" pueden aparecer elementos "codigo" e "id" en cualquier orden.

2.5. Elección "|" en una secuencia de elementos

En el siguiente documento XML, pueden aparecer cero o más elementos "articulo" que contengan un elemento "codigo" o un elemento "id", y obligatoriamente un elemento "nombre":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulos [
   <!ELEMENT articulos (articulo)*>
   <!ELEMENT articulo ((codigo | id), nombre)>
   <!ELEMENT codigo (#PCDATA)>
   <!ELEMENT id (#PCDATA)>
   <!ELEMENT nombre (#PCDATA)>
]>

<articulos>
   <articulo>
      <codigo>AF-47</codigo>
      <nombre>Martillo</nombre>
   </articulo>
   <articulo>
      <id>2056</id>
      <nombre>Destornillador</nombre>
   </articulo>
</articulos>
```

2.5.3 Secuencia de elementos en una lista de opciones

En la DTD del siguiente documento XML se ha indicado que pueden aparecer cero o más elementos "localidad". En el caso de aparecer, cada uno de ellos contendrá los elementos "pais" y "ciudad", o alternativamente un elemento "codigo_postal":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE localidades [
   <!ELEMENT localidades (localidad)*>
   <!ELEMENT localidad ((pais, ciudad) | codigo_postal)>
   <!ELEMENT pais (#PCDATA)>
   <!ELEMENT ciudad (#PCDATA)>
   <!ELEMENT codigo_postal (#PCDATA)>
]>

<localidades>
   <localidad>
      <pais>España</pais>
      <ciudad>Valencia</ciudad>
   </localidad>
   <localidad>
      <codigo_postal>31015</codigo_postal>
   </localidad>
</localidades>
```

2.5.4 #PCDATA en una lista de opciones - Contenido mixto

Al utilizar el *operador de elección* (**|**) en una DTD, si una de las opciones es **#PCDATA**, **esta debe escribirse en primer lugar**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulos [
   <!ELEMENT articulos (#PCDATA | codigo | id)*>
   <!ELEMENT codigo (#PCDATA)>
   <!ELEMENT id (#PCDATA)>
]>

<articulos>
   <id>8608</id>
   Teclado
   <codigo>AF-18</codigo>
   <codigo>AF-45</codigo>
   Disquetera
   <id>7552</id>
   <id>4602</id>
</articulos>
```

- Fíjese que, el elemento "articulos" de este documento, puede contener contenido mixto, es decir, texto y otros elementos.

Véase, en este último ejemplo, que el elemento "provincia" puede aparecer cero o más veces, pudiendo contener contenido mixto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE localidades [
   <!ELEMENT localidades (provincia*)>
   <!ELEMENT provincia (#PCDATA | ciudad | codigo_postal)*>
   <!ELEMENT ciudad (#PCDATA)>
   <!ELEMENT codigo_postal (#PCDATA)>
]>

<localidades>
   <provincia>
      Navarra
      <ciudad>Estella</ciudad>
      <codigo_postal>31015</codigo_postal>
      <ciudad>Tafalla</ciudad>
   </provincia>
   <provincia>
      Valencia
      <codigo_postal>46520</codigo_postal>
   </provincia>
</localidades>
```

Nota: Un elemento con contenido mixto debe ser definido siempre con la cardinalidad 0-n "*".

2.6 Atributos

La sintaxis básica para declarar un atributo en una DTD es:

<!ATTLIST nombre-del-elemento nombre-del-atributo tipo-de-atributo valor-del-atributo**>**

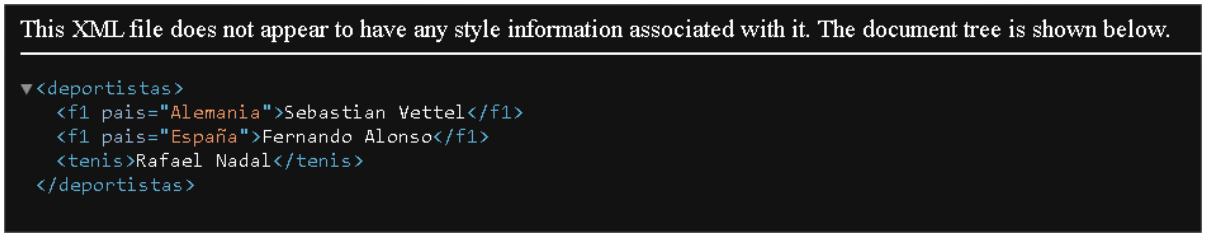
2.6.1 Atributo - Valor por defecto

En la DTD del siguiente documento XML se ha indicado que el elemento "f1" puede tener el atributo "pais":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais CDATA "España">
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="Alemania">Sebastian Vettel</f1>
   <f1>Fernando Alonso</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- Para el elemento "f1", **pais** es un atributo definido de tipo **CDATA** (*Character DATA*), es decir, su valor será una cadena de caracteres.
- Al no indicarse el país de *Fernando Alonso*, por defecto es **"España"**.
- Para *Sebastian Vettel*, al atributo **pais** se le ha asignado **"Alemania"**, que es un valor distinto al **valor-del-atributo**, que por defecto es **"España"**.

Al visualizar el documento XML en un navegador web, se verá algo parecido a:



2.6.2 Atributos - Definición múltiple

En la DTD del siguiente documento XML se ha indicado que el elemento "f1" puede tener tres atributos (**pais**, **fecha_de_nacimiento** y **equipo**):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais CDATA "España">
      <!ATTLIST f1 fecha_de_nacimiento CDATA #IMPLIED>
      <!ATTLIST f1 equipo CDATA #REQUIRED>
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="Alemania" fecha_de_nacimiento="03/07/1987"
equipo="Ferrari">Sebastian Vettel</f1>
   <f1 equipo="McLaren">Fernando Alonso</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- Obsérvese que, en este ejemplo, el atributo **equipo** es obligatorio escribirlo, **#REQUIRED**. Mientras que, el atributo **fecha_de_nacimiento** es opcional, **#IMPLIED**.

En una DTD, cuando se declara más de un atributo para un no es necesario escribir varias veces **<!ATTLIST**, pudiéndose escribir, por ejemplo:

```
   <!ATTLIST f1 pais CDATA "España"
                fecha_de_nacimiento CDATA #IMPLIED
                equipo CDATA #REQUIRED>
```

2.6.3 Atributos - Tipos de declaración

En DTD, existen los siguientes tipos de declaración de atributos:

Tipos de declaración de atributos en DTD

Valor	*Significado*
valor entre comillas dobles (**"**) o simples (**'**).	El atributo tiene un valor por defecto.
#REQUIRED	El atributo es obligatorio escribirlo.
#IMPLIED	El atributo es opcional escribirlo.
#FIXED valor entre comillas dobles (**"**) o simples (**'**).	El valor del atributo es fijo.

En la DTD interna del siguiente documento XML se ha declarado un atributo indicando que es obligatorio, es decir, **#REQUIRED**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais CDATA #REQUIRED>
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="Alemania">Sebastian Vettel</f1>
   <f1>Fernando Alonso</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- En este ejemplo, es obligatorio escribir el atributo **pais** en los elementos "f1". Por tanto, aunque el documento XML está bien formado, habría que indicar el **pais** de *Fernando Alonso* para que fuese válido.

```
<f1 pais="España">Fernando Alonso</f1>
```

- Por otra parte, fíjese que, de *Rafael Nadal* no es obligatorio indicar su país, ni se puede hacer.

En una DTD, para especificar que un atributo es opcional escribirlo o no, hay que

indicarlo mediante **#IMPLIED**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais CDATA #IMPLIED>
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="Alemania">Sebastian Vettel</f1>
   <f1>Fernando Alonso</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

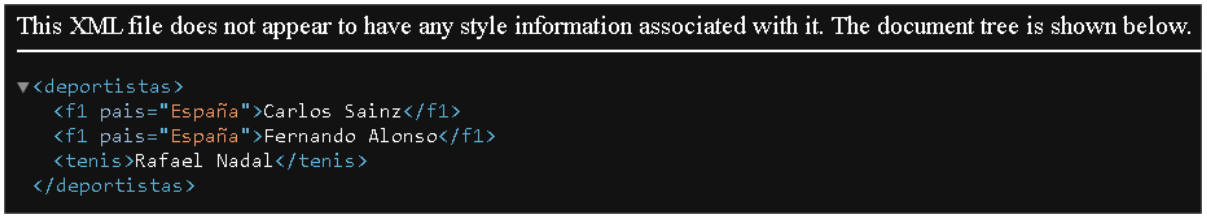
- En este caso, el atributo **pais** es opcional para los elementos "f1" que aparezcan en el documento XML. Así pues, obsérvese que, aunque no se ha indicado el país de *Fernando Alonso*, el documento es válido.

Cuando en una DTD, se quiere declarar un atributo que tome un valor fijo, esto se puede hacer con **#FIXED valor**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais CDATA #FIXED "España">
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="España">Carlos Sainz</f1>
   <f1>Fernando Alonso</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- Según la DTD de este documento XML, todos los elementos “f1” que aparezcan tendrán el atributo **pais** con el valor **"España"**. Por tanto, no es necesario haberlo escrito para *Carlos Sainz*. De hecho, si se hubiese escrito otro valor, el documento no sería válido.

De modo que, para este caso, al visualizar el documento XML en un navegador web, se mostrará algo parecido a:



2.6.4 Atributos - Tipos

En DTD, existen los siguientes tipos de atributos:

Tipos de atributos en DTD	
Tipo	***Descripción***
CDATA	(*Character DATA*) El valor son datos de tipo carácter, es decir, texto.
Enumerado	El valor puede ser uno de los pertenecientes a una lista de valores escritos entre paréntesis "**()**" y separados por el carácter "**|**".
ID	El valor es un identificador único.
IDREF	El valor es un identificador que tiene que existir en otro atributo **ID** del documento XML.
IDREFS	El valor es una lista de valores que existan en otros atributos **ID** del documento XML, separados por espacios en blanco.
NMTOKEN	El valor es una cadena de caracteres, pudiendo contener letras minúsculas, letras mayúsculas, números, puntos "**.**", guiones medios "**-**", guiones bajos "_" o el carácter dos puntos "**:**".
NMTOKENS	El valor puede contener uno o varios valores de tipo **NMTOKEN** separados por espacios en blanco.
NOTATION	El valor es el nombre de una notación.
ENTITY	El valor es el nombre de una entidad.
ENTITIES	El valor puede contener uno o varios valores de tipo **ENTITY** separados por espacios en blanco.
Especiales	Existen dos atributos especiales: **xml:lang** y **xml:space**.

2.6.4.1 Atributos de tipo CDATA en DTD

En una DTD, un atributo de tipo CDATA (Character DATA), es aquel cuyo valor puede ser una cadena de caracteres (texto).

En la DTD del siguiente documento XML, el atributo pais del elemento "ciudad" ha sido declarado de tipo CDATA:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudades [
   <!ELEMENT ciudades (ciudad)*>
   <!ELEMENT ciudad (#PCDATA)>
      <!ATTLIST ciudad pais CDATA #REQUIRED>
]>

<ciudades>
  <ciudad pais="Italia">Roma</ciudad>
  <ciudad pais="Francia">París</ciudad>
  <ciudad pais="Alemania">Berlín</ciudad>
  <ciudad pais="">Viena</ciudad>
</ciudades>
```

- Obsérvese que, el valor del atributo **pais** puede estar vacío.

2.6.4.2 Atributos de tipo enumerado en DTD

En una DTD, se puede declarar un atributo de tipo enumerado indicando que **su valor puede ser uno de los pertenecientes a una lista de valores** escritos entre paréntesis "**()**" y separados por el carácter "**|**".

En la DTD del siguiente documento XML, los valores posibles del atributo **pais** son **"ESP"**, **"FRA"**, **"ITA"** y **"ALE"**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1 | tenis)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 pais (ESP | FRA | ITA | ALE) "ESP">
   <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
   <f1 pais="ALE">Sebastian Vettel</f1>
   <f1>Fernando Alonso</f1>
   <f1 pais="ESP">Carlos Sainz</f1>
   <tenis>Rafael Nadal</tenis>
</deportistas>
```

- Véase que, en este caso, se ha especificado **"ESP"** como valor por defecto, siendo obligatorio que esté en la lista de valores escritos entre paréntesis "**()**".

Al visualizar este documento en un navegador web, en pantalla se verá:

Si se quiere definir el atributo **pais** obligatorio, habría que escribir:

```
<!ATTLIST f1 pais (ESP | FRA | ITA | ALE) #REQUIRED>
```

Por tanto, para Fernando Alonso se tendría que escribir:

```
<f1 pais="ESP">Fernando Alonso</f1>
```

2.6.4.3 Atributos de tipo ID en DTD

En una DTD, los atributos declarados **ID** son aquellos que solo pueden tomar un valor único (identificador) para cada elemento.

En la DTD del siguiente documento XML, el atributo **codigo** del elemento "f1" ha sido declarado de tipo **ID**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
   <!ELEMENT deportistas (futbol | f1)*>
   <!ELEMENT futbol (#PCDATA)>
   <!ELEMENT f1 (#PCDATA)>
      <!ATTLIST f1 codigo ID #REQUIRED>
]>

<deportistas>
   <f1 codigo="ALO">Fernando Alonso</f1>
   <f1 codigo="VET">Sebastian Vettel</f1>
</deportistas>
```

Hay que tener en cuenta que:

- Cada elemento escrito en un documento XML, solo puede tener un atributo **ID**.
- En un documento XML, no pueden escribirse dos elementos que tengan el mismo valor en un atributo **ID**, aunque dicho atributo sea distinto.
- Todo atributo declarado de tipo **ID** tiene que ser **#IMPLIED** (opcional) o **#REQUIRED** (obligatorio).

2.6.4.4 Atributos de tipo IDREF en DTD

En una DTD, los atributos declarados **IDREF** son aquellos cuyo valor tiene que existir en otro atributo **ID** del documento XML.

En la DTD del siguiente documento XML, se indica que los elementos "pelicula" que se escriban, deben incluir el atributo **direccion**, cuyo valor estará asignado a un

atributo **ID** de otro elemento del documento. En este caso, el valor estará asignado a un atributo **coddir** de un elemento "director":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cine [
   <!ELEMENT cine (directores, peliculas)>
   <!ELEMENT directores (director)*>
      <!ELEMENT director (#PCDATA)>
         <!ATTLIST director coddir ID #REQUIRED>
   <!ELEMENT peliculas (pelicula)*>
      <!ELEMENT pelicula (#PCDATA)>
         <!ATTLIST pelicula direccion IDREF #REQUIRED>
]>

<cine>
   <directores>
      <director coddir="CE">Clint Eastwood</director>
      <director coddir="JC">James Cameron</director>
   </directores>
   <peliculas>
      <pelicula direccion="JC">Avatar</pelicula>
      <pelicula direccion="CE">Mystic River</pelicula>
      <pelicula direccion="JC">Titanic</pelicula>
   </peliculas>
</cine>
```

- Obsérvese que, por ejemplo, para la película *Titanic* se ha indicado en su atributo **direccion** el valor **"JC"**, que es el valor del atributo **coddir** del director *James Cameron*.
- En este documento XML, el atributo de tipo **IDREF** se ha definido obligatorio, **#REQUIRED**. Pero, a un atributo **IDREF** también se le puede especificar un valor por defecto, un valor fijo o que sea opcional escribirlo, **#IMPLIED**.

2.6.4.5 Atributos de tipo IDREFS en DTD

En una DTD, los atributos declarados **IDREFS** son aquellos cuyo valor puede ser una lista de valores que existan en otros atributos **ID** del documento XML.

En la DTD del siguiente documento XML, se indica que el valor del atributo **filmografia** de un elemento "director", puede ser una lista de valores de atributos **ID**. En este caso, una lista de valores escritos en el atributo **codpel** de los elementos "pelicula" que aparezcan en el documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cine [
   <!ELEMENT cine (peliculas, directores)>
   <!ELEMENT peliculas (pelicula)*>
      <!ELEMENT pelicula (#PCDATA)>
         <!ATTLIST pelicula codpel ID #REQUIRED>
   <!ELEMENT directores (director)*>
      <!ELEMENT director (#PCDATA)>
         <!ATTLIST director filmografia IDREFS #REQUIRED>
```

```
]>

<cine>
   <peliculas>
      <pelicula codpel="P1">Avatar</pelicula>
      <pelicula codpel="P2">Mystic River</pelicula>
      <pelicula codpel="P3">The Terminator</pelicula>
      <pelicula codpel="P4">Titanic</pelicula>
   </peliculas>
   <directores>
      <director filmografia="P2">Clint Eastwood</director>
      <director filmografia="P1 P3 P4">James Cameron</director>
   </directores>
</cine>
```

- Obsérvese que, los valores de la lista de valores de un atributo **IDREFS**, se escriben separados por un espacio en blanco.

2.7 Declaración de entidades en una DTD

Una entidad es una unidad de almacenamiento que puede contener cualquier cosa: desde una cadena de caracteres hasta un fichero, un gráfico

En una DTD se pueden declarar entidades generales y paramétricas (de parámetro). Las entidades generales pueden ser:

- **Entidades generales internas analizables**
- Entidades generales externas analizables
- Entidades generales externas no analizables

Por otro lado, las entidades paramétricas pueden ser:

- Entidades paramétricas internas analizables
- Entidades paramétricas externas analizables

Las entidades generales pueden utilizarse en el cuerpo de un documento XML y en su DTD. Sin embargo, las entidades paramétricas solo pueden utilizarse dentro de la DTD.

2.7.1 Entidades generales internas analizables en DTD

Para declarar una entidad general interna analizable (parsed) en una DTD, se utiliza la siguiente sintaxis:

```
<!ENTITY nombre-de-la-entidad "valor-de-la-entidad">
```

En la DTD del siguiente documento XML, se han declarado tres entidades (**escritor**, **obra** y **fecha**):

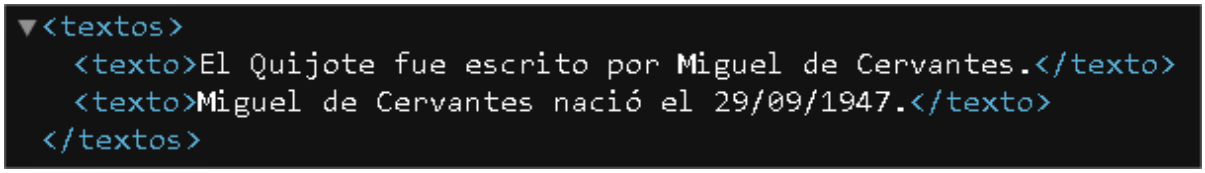
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE textos [
   <!ELEMENT textos (texto)+>
```

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

   <!ENTITY escritor "Miguel de Cervantes">
   <!ENTITY obra "El Quijote">
   <!ENTITY fecha "29/09/1947">
]>

<textos>
   <texto>&obra; fue escrito por &escritor;.</texto>
   <texto>&escritor; nació el &fecha;.</texto>
</textos>
```

- Obsérvese que, para referenciar a las entidades, se ha utilizado la sintaxis:

```
&nombre-de-la-entidad;
```

Si este documento XML se visualizase en un navegador web, se vería algo parecido a:



3 ¿Qué es XSD?

XSD (XML Schema Definition), también llamado simplemente **XML Schema**, es un lenguaje de esquema utilizado para **describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa**, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así una percepción del tipo de documento con un nivel alto de abstracción.

Un ejemplo sería:

Se quiere almacenar una lista de marcadores de páginas web, guardando de cada uno de ellos su nombre, una descripción y su URL. Para ello, se ha escrito el siguiente documento XML (***"marcadores.xml"***) asociado al archivo ***"marcadores.xsd"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<marcadores xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="marcadores.xsd">
   <pagina>
      <nombre>Abrirllave</nombre>
      <descripcion>Tutoriales de informática.</descripcion>
      <url>http://www.abrirllave.com/</url>
   </pagina>
   <pagina>
      <nombre>Wikipedia</nombre>
```

```
      <descripcion>La enciclopedia libre.</descripcion>
      <url>http://www.wikipedia.org/</url>
   </pagina>
   <pagina>
      <nombre>W3C</nombre>
      <descripcion>World Wide Web Consortium.</descripcion>
      <url>http://www.w3.org/</url>
   </pagina>
</marcadores>
```

- Para vincular un esquema a un documento XML, es obligatorio que este último haga referencia al espacio de nombres **http://www.w3.org/2001/XMLSchema-instance**. Para ello, habitualmente se utiliza el prefijo **xsi**.
- El atributo **noNameSchemaLocation** permite referenciar a un archivo con la definición de un esquema que no tiene ningún espacio de nombres asociado. En este caso, dicho archivo es ***"marcadores.xsd"***.

El esquema XML guardado en ***"marcadores.xsd"*** y que permita validar el documento XML ***"marcadores.xml"*** podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="marcadores">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="pagina" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="descripcion" type="xs:string"/>
              <xs:element name="url" type="xs:string"/>
             </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Para estar bien formado, un esquema XML tiene que cumplir las mismas reglas de sintaxis que cualquier otro documento XML.