

## TEMA 7 (I)

### XML. CREACIÓ DTD

LM  
CFGs DAW

Autor: Pascual Ligeró

Revisado por:

Fco. Javier Valero – [franciscojavier.valero@ceedcv.es](mailto:franciscojavier.valero@ceedcv.es)

2019/2020

Versión:191126.2036

## Licencia



**CC BY-NC-SA 3.0 ES Reconocimiento - NoComercial - CompartirIgual (by-nc-sa)**


No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

**NOTA: Esta es una obra derivada de la original realizada por Pascual Ligeró.**

## Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

# ÍNDICE DE CONTENIDO

<b>1. DTD.....</b>	<b>4</b>
<b>2. DECLARACIÓN DEL DTD.....</b>	<b>5</b>
<b>3. DECLARACIÓN DE TIPOS DE ELEMENTOS.....</b>	<b>7</b>
3.1 Elementos que sólo contienen elementos.....	10
3.2 Elementos que solo contienen datos.....	11
3.3 Elementos vacíos.....	11
3.4 Elementos mixtos.....	12
<b>4. DECLARACIÓN DE TIPOS DE ATRIBUTOS.....</b>	<b>12</b>
4.1 Tipo de atributo CDATA, NMTOKEN Y NMTOKENS.....	14
4.2 Tipos de atributos enumerados.....	15
4.3 Tipos de atributos ID.....	15
4.4 Predeterminación de atributos.....	17
<b>5. DECLARACIÓN DE ENTIDADES.....</b>	<b>17</b>
<b>6. VALIDACIÓN DE DOCUMENTOS XML FRENTE AL DTD.....</b>	<b>18</b>
<b>7. Bibliografía.....</b>	<b>18</b>

## UD07. XML. CREACIÓN DTD

### 1. DTD

Cuando un documento XML no contiene un DTD, cualquier etiqueta que aparezca en el mismo se considerará válida. De manera que el analizador sólo podrá comprobar que el documento está bien formado. La existencia del DTD permite asegurar que los documentos siguen las reglas del lenguaje.

Por lo tanto es imprescindible la especificación de un DTD que defina formalmente el lenguaje de etiquetado requerido. Este debería ser el primer paso antes de escribir cualquier documento XML.

Podemos hacer una comparación con el procedimiento que se usa trabajando con una base de datos: primero se define la estructura o esquema y luego ya se puede trabajar con los datos correspondientes.

**Un documento XML válido es un documento “bien formado” que, además, se ajusta a las reglas de un DTD (Document Type Definition).**

- Un XML con la sintaxis correcta es un documento “bien formado”.
- Un XML validado contra un DTD es un XML “válido”.

Mediante el uso de los DTD podremos validar documentos XML. La validación de documentos consiste en comprobar que, además de ser bien formados, se corresponden con la estructura prevista para el contenido que aportan.

Si, por ejemplo, suponemos que cada tienda o concesionario que quiere vender coches tiene que enviar información sobre ellos a un sitio web donde se publicarán las ofertas, entendemos que no puede ser viable que cada una de esas empresas implemente su propia versión de documento XML.

Por el contrario, lo adecuado es que el sitio web defina el formato exacto que deben seguir los documentos se van a recibir, y que así se asegure que serán documento XML *válidos* y todos con la misma estructura.

## 2. DECLARACIÓN DEL DTD

Una definición de tipo de documento (DTD) es una descripción de la estructura y sintaxis de un documento XML.

**Una DTD define las reglas que debe cumplir la información contenida en un documento XML,** para que el documento sea válido. Cuando creamos una definición de tipo de documento lo que estamos haciendo es crear nuestro propio lenguaje de marcas para nuestra aplicación concreta, de forma que el documento XML que se ajuste a esa DTD se pueda considerar **válido**.

En un DTD se describen los elementos (los nombres de los elementos, los atributos que pueden tener, los tipos de datos que pueden contener, etc.) que podrá contener el documento así como su estructura y posibilidades de anidamiento.

La DTD puede ser un fichero externo, con extensión .dtd, aunque también puede estar contenida en el propio documento XML, incluido en la declaración de tipo de documento.

Por tanto no hay que confundir ambas cosas, ya que:

La **Definición** (DTD) contendrá las especificaciones necesarias sobre los elementos, pudiendo estar incluido o no en el propio documento.

La **Declaración** siempre estará en el prólogo del documento (incluyendo la definición del etiquetado o simplemente haciendo referencia a su ubicación exterior).

Esta declaración aunque es de carácter opcional en el documento XML, será necesaria para poder validar los datos que contiene.

Estará situada en el prólogo del documento justo a continuación de la declaración XML, en la segunda línea, mediante la declaración **DOCTYPE** y deberá contener **siempre** la especificación de elemento raíz del documento.

- A) **El formato de declaración, cuando incluye la definición en el propio documento, podría ser:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE elemento_raiz [
Declaraciones internas
]>
<elemento_raiz>
.....
</elemento_raiz>
```

Es decir, estará incluida dentro de la declaración DOCTYPE, después del elemento\_raiz y comprendida entre “[” y “]”

Para definir el contenido del elemento se pueden utilizar los términos EMPTY, (#PCDATA) o ANY o escribir expresiones más complejas:

- **EMPTY**: significa que el elemento es vacío, es decir, que no puede tener contenido.  
Los elementos vacíos pueden escribirse con etiquetas de apertura y cierre sin nada entre ellos, ni siquiera espacios, o con una etiqueta vacía. EMPTY debe escribirse sin paréntesis.
- **(#PCDATA)**: significa que el elemento puede contener texto. #PCDATA debe escribirse entre paréntesis.
- **ANY**: significa que el elemento puede contener cualquier cosa (texto y otros elementos). ANY debe escribirse sin paréntesis.

**Ejemplo:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE nota [
  <!ELEMENT nota (destinatario,remitante,cabecera,cuerpo)>
  <!ELEMENT destinatario (#PCDATA)>
  <!ELEMENT remitente (#PCDATA)>
  <!ELEMENT cabecera (#PCDATA)>
  <!ELEMENT cuerpo (#PCDATA)>
]>
<nota>
  <destinatario>Tove</destinatario>
  <remitente>Jani</remitente>
  <cabecera>Recordatorio</cabecera>
  <cuerpo>Llámame!</cuerpo>
</nota>
```

El DTD anterior tiene el siguiente significado:

- **!DOCTYPE nota**, indica que el elemento raíz de este documento es nota.
- **!ELEMENT nota**, indica que el elemento nota contiene cuatro elementos: destinatario, remitente, cabecera y cuerpo.
- **!ELEMENT destinatario**, indica que el elemento destinatario es de tipo #PCDATA, es decir, texto.
- **!ELEMENT remitente**, indica que el elemento remitente es de tipo #PCDATA.
- **!ELEMENT cabecera**, indica que el elemento cabecera es de tipo #PCDATA.
- **!ELEMENT cuerpo**, indica que el elemento cuerpo es de tipo #PCDATA.

B) *Normalmente un DTD se utiliza para validar un gran número de documentos XML. La mayoría de las veces tiene poco sentido que el DTD esté incluido dentro del documento XML ya que se tendría que repetir en todos los documentos XML pertenecientes a un mismo lenguaje (DTD).*

*Teniendo esto en cuenta se puede distinguir entre dos tipos de referencias externas:*

Un documento DTD aún no publicado. Se especifica con la palabra SYSTEM seguida de la URL con la ubicación del documento: `<!DOCTYPE elemento_raiz SYSTEM "archivo_declaraciones.dtd">`

En caso de utilizar un DTD que ha sido publicado utilizaremos la palabra PUBLIC seguida por el identificador público asociado a este DTD. Sigue siendo necesario incluir la URL al fichero DTD que solo será utilizado en caso de fallar la localización del fichero. Usando el identificador público: `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`.

### 3. DECLARACIÓN DE TIPOS DE ELEMENTOS

Los elementos son la base de las marcas XML. Indican qué etiquetas serán permitidas en el documento. El DTD tiene que declarar cada uno de ellos, y las declaraciones de tipo de elemento deben empezar con "**<!ELEMENT**" seguidas por el identificador genérico del elemento que se declara.

**<!ELEMENT** *nombre tipo\_contenido* **>**

El nombre del elemento debe ser un nombre XML válido y sólo puede haber una declaración por elemento. No podrá repetirse.

Veremos un pequeño caso que iremos analizando. Teníamos el siguiente documento:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE identificacion SYSTEM "identificacion.dtd">

<identificacion>
  <nombre_completo>
    <nombre>
      Pepe
    </nombre>

    <apellido1>
      Gonzalez
    </apellido1>

    <apellido2>
      Ribera
    </apellido2>
  </nombre_completo>

  <apodo>
    Pepito Grillo
  </apodo>
</identificacion>
```

Al cual le podría corresponder un DTD como el siguiente, que estaría contenido en el fichero identificacion.dtd:

```
<!ELEMENT identificacion (nombre_completo, apodo)>
  <!ELEMENT nombre_completo (nombre, apellido1,apellido2)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT apellido1 (#PCDATA)>
  <!ELEMENT apellido2 (#PCDATA)>
  <!ELEMENT apodo (#PCDATA)>
```

Vemos como en el DTD lo primero que hacemos es definir el “**elemento raíz**” que llamamos “identificacion” y está formado, a su vez, por dos elementos: “nombre\_completo” y “apodo”. Por otro lado, “nombre\_completo” está formado, a su vez por otros tres elementos: “nombre”, “apellido1” y “apellido2”.

También podemos observar cómo los elementos que no contienen a otros elementos, como “nombre”, “apellido1”, “apellido2” y “apodo” definen que el tipo de datos de esos elementos es #PCDATA, que más tarde veremos que es texto plano.

Comprueba también que el fichero xml es válido, es decir, sigue la estructura definida por el DTD.



Vamos ahora a ampliarlo con algunos elementos que explicaremos en cada caso: El nuevo DTD podría validar documentos con contenidos similares al siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE identificacion SYSTEM "identificacion_ampliado.dtd">
<identificacion>
  <situacion>
    <estudiante>
      Universitat Jaume I Castelló
    </estudiante>
    <!-- esto es un comentario:
          estudiante y trabajador son alternativos,
          si lleva información de uno no tendrá del otro -->
  </situacion>

  <nombre_completo>
    <nombre>
      Pepe
    </nombre>

    <apellido1>
      Gonzalez
    </apellido1>

    <apellido2>
      Ribera
    </apellido2>
  </nombre_completo>

  <apodo>
    Pepito Grillo
  </apodo>

  <mai>
    elgrillito@correobasura.com
  </mai>
</identificacion>
```

Ampliamos el DTD con más elementos que explicaremos a continuación:

```

<!ELEMENT identificacion (situacion?,nombre_completo, apodo*, mail?)>
  <!ELEMENT situacion (estudiante|trabajador)>
  <!ELEMENT estudiante (#PCDATA)>
  <!ELEMENT trabajador (#PCDATA)>
  <!ELEMENT nombre_completo (nombre+, apellido1,apellido2)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT apellido1 (#PCDATA)>
    <!ELEMENT apellido2 (#PCDATA)>
  <!ELEMENT apodo (#PCDATA)>
  <!ELEMENT mail (#PCDATA)>

```

Según el contenido de los elementos podemos tener diferentes tipos:

### 3.1 Elementos que sólo contienen elementos

En este caso tendremos que especificar, entre paréntesis, el identificador de cada uno de los elementos que anidará.

Hay dos tipos de relación entre los elementos hijos:

- **Secuenciales**, referenciándolos por su nombre, separados por comas:

```
<!ELEMENT nombre_elemento (elemento1,elemento2,elemento3...)>
```

En nuestro ejemplo, el elemento raíz (que siempre incorporaremos en primer lugar), hemos visto que lo especificamos como:

```
<!ELEMENT identificacion (situacion?,nombre_completo, apodo*,mail?)>
```

- **Alternativos**: Cuando el elemento contiene uno y solo uno de los elementos hijos especificados, en cuyo caso los separaremos mediante “|”:

```
<!ELEMENT nombre_elemento (elemento1 | elemento2 | elemento3...)>
```

En nuestro ejemplo tenemos:

```
<!ELEMENT situacion (estudiante|trabajador)>
```

En este ejemplo el elemento situación podrá tomar uno de los dos valores especificados: estudiante o trabajador, pero sólo uno de ellos.

Además de especificar qué elementos hijos puede contener el elemento y en qué orden, se puede establecer cuántas veces aparece cada uno de ellos, con un carácter que indique el factor de repetición, o un **indicador de frecuencia**:

- El carácter ‘\*’: el elemento o grupo de elementos puede repetirse 0 o más veces.

- El carácter '?': el elemento o grupo de elementos puede aparecer 0 o 1 veces.
- El carácter '+': el elemento o grupo de elementos puede repetirse 1 o más veces.
- Por defecto, si no ponemos nada, el elemento debe aparecer una vez.
- (): permite agrupar expresiones: `<!ELEMENT ejemplo (a, (a|b))>`

Siguiendo nuestro ejemplo:

**`<!ELEMENT identificacion (situacion?,nombre_completo, apodo*,mail?)>`**

Si consideramos que nuestro elemento raíz (identificacion) está formado por los elementos "situación" y "mail" que son opcionales, pueden aparecer una vez o no aparecer; "nombre\_completo" siempre debe aparecer, aunque solo una vez ya que aparece en la lista sin ningún indicador de frecuencia, mientras que el apodo puede no estar presente o aparecer repetidamente, ya que una persona puede no tener ningún apodo o varios.

**`<!ELEMENT nombre_completo (nombre+, apellido1,apellido2)>`**

Así mismo, anidado en nombre\_completo, tenemos el nombre que podemos considerar que es necesario una vez, es decir, todas las personas tienen como mínimo un nombre, aunque pueden tener más, mientras tanto el apellido1 como el apellido2 serán únicos.

### 3.2 Elementos que solo contienen datos

En la declaración se especifican mediante (**`#PCDATA`**) e indica que pueden contener datos de tipo carácter (**`Parser Character Data`**). Debemos tener cuidado de que entre el identificador del elemento y el símbolo inicial del paréntesis haya un espacio de separación.

En nuestro ejemplo los siguientes elementos son de tipo texto:

**`<!ELEMENT nombre (#PCDATA)>`**

**`<!ELEMENT apellido1 (#PCDATA)>`**

**`<!ELEMENT apellido2 (#PCDATA)>`**

**`<!ELEMENT apodo (#PCDATA)>`**

**`<!ELEMENT mail (#PCDATA)>`**

### 3.3 Elementos vacíos

Aunque no es usual, los elementos pueden no tener ningún contenido pero pueden utilizarse para insertar los atributos. Se declaran especificando la palabra **`EMPTY`**.

**`<!ELEMENT nombre_elemento EMPTY>`**

Un ejemplo es la declaración `<br/>` de XHTML.

**`<!ELEMENT br EMPTY>`**

### 3.4 Elementos mixtos

Son aquellos que indican que un elemento puede contener texto y otros elementos.

No suelen utilizarse en XML ya que se puede especificar qué elementos hijos podrán aparecer, pero no dan indicación de frecuencia o si forman parte de una secuencia alternativa. Su formato es muy rígido, siempre en primer lugar PCDATA, con una lista alternativa como un grupo.

***No se puede aplicar caracteres de repetición a los elementos hijos, solo la posibilidad del carácter de repetición \* para el conjunto. (Debe especificarse obligatoriamente el carácter de repetición '\*' a todo el grupo).***

Se desaconseja su uso.

La declaración sería:

***<!ELEMENT nombre elemento (#PCDATA|elem1|elem2|elem3)\*>***

```
<!ELEMENT comentario (#PCDATA | enlace | persona)* >
<comentario>
  Este texto está relacionado con el
  <enlace url="http://www.quijote.com">Quijote</enlace>
  y ha sido realizado por
  <persona>Leopoldo Alas Clarín</persona>
</comentario>
```

## 4. DECLARACIÓN DE TIPOS DE ATRIBUTOS

Los atributos permiten añadir información adicional a los elementos de un documento. Las diferencias entre los elementos y los atributos son:

Los atributos no pueden contener sub-atributos y que los usamos para añadir información corta, sencilla y desestructurada.

***Cada uno de los atributos sólo se puede especificar una vez, y en cualquier orden.***

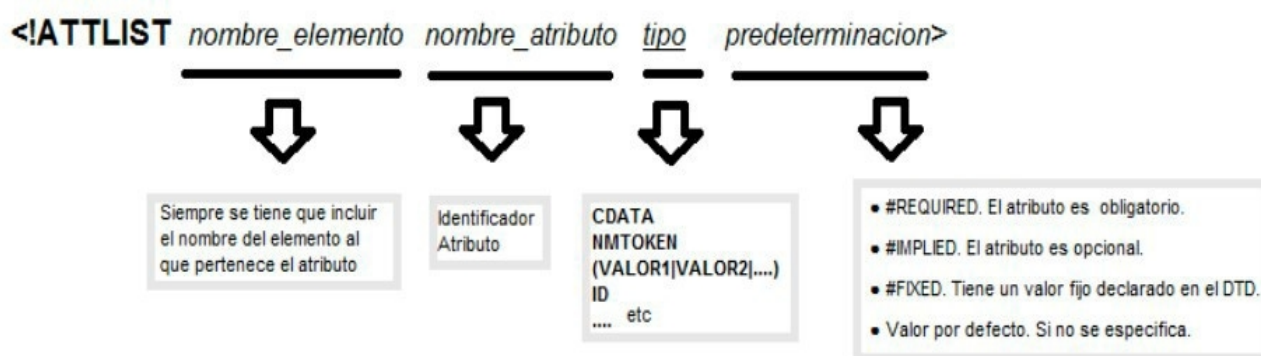
```
<mensaje prioridad="urgente">
  <de>Alfredo Reino</de>
  <a>Hans van Parijs</a>
  <texto idioma="holandés"> Hallo Hans, hoe gaat het?
  ...
</texto>
</mensaje>
```

Igual que ocurre con los elementos, cada uno de los distintos atributos identificados en la fase de diseño, debe declararse previamente en el DTD.

Pueden agruparse en una lista correspondiente para cada elemento.

Para cada atributo podremos tener también la especificación de su tipo y su valor por defecto o predeterminación. Puede haber múltiples definiciones de listas de atributos para un mismo elemento. Pero si se declara varias veces el mismo atributo sólo prevalece el primero.

Su sintaxis será:



En el ejemplo anterior, para declarar la lista de atributos del elemento <mensaje> podríamos definir y utilizar la siguiente definición de atributos:

**<!ELEMENT mensaje (de, a, texto)>**

**<!ATTLIST mensaje prioridad (normal | urgente)>**

Así pues, definimos para el elemento “mensaje” un atributo que llamamos “prioridad” y que puede tomar los valores “normal” o “urgente”.

Veamos a continuación cada parte del atributo con más detalle. El tipo del atributo podrá tomar los siguientes valores:

- CDATA
- NMTOKEN
- NMTOKENS
- Tipos de atributos enumerados
- Tipos de atributos **ID**, **IDREF**, **IDREFS**

#### 4.1 Tipo de atributo CDATA, NMTOKEN Y NMTOKENS.

Si el valor del atributo está formado por un texto que puede incluir cualquier carácter imprimible, a excepción de los caracteres especiales, incluidos los espacios en blanco, entonces el tipo será **CDATA**.

Si pretendemos limitar el tipo de caracteres que pueden aparecer como valor en el atributo, debemos utilizar el tipo **NMTOKEN**. Sólo permite que aparezcan los mismos caracteres que utilizamos para definir elementos y atributos.

Existe también la posibilidad de utilizar el tipo **NMTOKENS**. Esto indica que el atributo contendrá una lista de cadenas de tipo **NMTOKEN**.

Ejemplos (aunque los puntos .... indica que la definición del atributo está incompleta, según veremos posteriormente:

**<!ATTLIST coche color CDATA ... >**

Significa que el atributo color puede tomar cualquier valor, por ejemplo: blanco-rojo, rojo, beige.claro, azul\_celeste ...

**<!ATTLIST coche color NMTOKEN...>.**

Significa que la propiedad color puede tomar solo valores que contengan letras, dígitos, puntos, guiones y subrayados. Deben comenzar por letra y no pueden contener espacios en blanco.

**<ATTLIST coche color NMTOKENS...>**

La propiedad color será una lista de NMTOKENS. Por ejemplo: <coche color="blanco negro gris">

ejemplo:

**<!ATTLIST coche color CDATA #IMPLIED>**

<coche color="az/uul"> --> VÁLIDO

<coche color="azul turquesa"> --> VÁLIDO

**<!ATTLIST coche color NMTOKEN #IMPLIED>**

<coche color="az/uul"> --> INVÁLIDO

<coche color="azul turquesa"> --> INVÁLIDO

**NMTOKEN** no admite caracteres que no sean letras!

## 4.2 Tipos de atributos enumerados

Se usan cuando el valor del atributo está restringido a un conjunto de valores. En la declaración se usa el carácter '|' para separar los valores. Al final hay que indicar uno por defecto.

```
<!ATTLIST coche color (blanco | negro | gris) "negro">
```

De esta forma, la propiedad color solo puede tomar los valores "blanco", "negro" o "gris", y sólo uno de ellos. Cualquier otro valor hará que la validación del documento XML falle.

## 4.3 Tipos de atributos ID

Es frecuente que algunos elementos tengan algún valor que los identifica de forma unívoca. Cuando un elemento contiene una propiedad de este tipo hay que asegurarse que ésta no se repite en otro elemento.

Incluso con elementos diferentes. **Seguirá la regla de valores NMTOKEN.** Podemos poner:

```
<!ATTLIST libro codigo ID #REQUIRED>
```

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
</ejemplo>

<ejemplo>
  <libro codigo="1">Poema de Gilgamesh</libro>
  <!-- ERROR: el valor de un atributo de tipo ID no puede
empezar con un número. NMTOKEN-->
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
</ejemplo>

<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L1">Los preceptos de Ptah-Hotep</libro>
<!-- ERROR: no se puede repetir un atributo de tipo ID --
>
</ejemplo>
```

- **IDREF:** el valor del atributo debe coincidir con el valor del atributo ID de otro elemento.

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo ((libro|prestamo)*)>
  <!ELEMENT libro (#PCDATA) >
  <!ATTLIST libro codigo ID #REQUIRED>
  <!ELEMENT prestamo (#PCDATA) >
  <!ATTLIST prestamo libro IDREF #REQUIRED>
]>
```

**OK:**

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <prestamo libro="L1">Numa Nigerio</prestamo>
</ejemplo>
```

**ERRONEO:**

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <prestamo libro="L2">Numa Nigerio</prestamo>
<!-- ERROR: el valor "L2" no es ID de ningún elemento -->
</ejemplo>
```

- **IDREFS:** el valor del atributo es una serie de valores separados por espacios que coinciden con el valor del atributo ID de otros elementos.

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo ((libro|prestamo)*)>
  <!ELEMENT libro (#PCDATA) >
  <!ATTLIST libro codigo ID #REQUIRED>
  <!ELEMENT prestamo (#PCDATA) >
  <!ATTLIST prestamo libro IDREFS #REQUIRED>
]>
```

**OK:**

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
  <prestamo libro="L1 L2">Numa Nigerio</prestamo>
</ejemplo>
```

**OK:**

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
  <prestamo libro="L1">Numa Nigerio</prestamo>
</ejemplo>
```

**ERRONEO:**

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
  <prestamo libro="L3">Numa Nigerio</prestamo>
<!-- ERROR: el valor "L3" no es ID de ningún elemento -->
</ejemplo>
```



## 4.4 Predeterminación de atributos

A continuación del nombre y el tipo del atributo debemos especificar si se requiere o no la presencia de un atributo, y el modo de gestionarlo en ese caso.

Existen cuatro posibles alternativas:

- **#REQUIRED**. El atributo es obligatorio.
- **#IMPLIED**. El atributo es opcional.
- **#FIXED**. Tiene un valor fijo declarado en el DTD.
- Valor por defecto si no se especifica.

Si no se define ninguna de estas alternativas el atributo será por defecto opcional. Siguiendo y completando nuestros ejemplos de los coches, podríamos tener:

```
<!ATTLIST coche color CDATA #IMPLIED>
```

```
<!ATTLIST coche matricula ID #REQUIRED>
```

```
<!ATTLIST coche marca CDATA FIXED "Seat">
```

```
<!ATTLIST coche color CDATA "rojo">
```

## 5. DECLARACIÓN DE ENTIDADES

Una entidad es una referencia a un objeto (texto, ficheros, páginas web, etc.,) que serán sustituidas por el contenido al que se refieren.

Permite guardar contenido que puede ser utilizado muchas veces y poder descomponer un documento grande en subconjuntos más manejables.

En ocasiones se emplean para descomponer un documento grande en otros más pequeños, y en otros casos se usan para representar caracteres que no pueden incluirse como texto, como el caso de caracteres especiales.

Su sintaxis general sería:

```
<!ENTITY identificador "valor">
```

Puede ser una **entidad interna**. Es la más sencilla. Consiste en abreviaturas definidas en el DTD. Ejemplo:

```
<ENTITY tema "Introducción a XML">
```

Una vez definida en el DTD, en el documento XML correspondiente podemos utilizarla insertando

```
&tema;
```

Es decir, con el identificador precedido de & y acabado en ";". El parser cambiará la entidad por el valor asignado.

Existen también las **entidades externas**. Aquí no tenemos el contenido dentro del DTD sino en cualquier otro sitio del sistema. Se hace referencia a su contenido mediante una URL precedida de la palabra SYSTEM o PUBLIC según proceda, y de esa forma podemos incluir parte de archivos para poder descomponerlos en pequeñas partes. La sintaxis es

**<!ENTITY nombre SYSTEM "URL">**

Por ejemplo:

<!ENTITY tema SYSTEM <http://www.misapuntes.com/tema3.xml>> o

<!ENTITY intro SYSTEM <http://ww.miservidor.com/intro.xml>>

## 6. VALIDACIÓN DE DOCUMENTOS XML FRENTE AL DTD

Existen distintos programas que además de ayudarnos en la edición del fichero XML y el DTD mediante el resaltado de las etiquetas, incorporación de sangrías, utilización de colores, etc., también nos comprueban que el documento esté bien formado.

Aunque se podría utilizar cualquiera de ellos, en este curso utilizaremos el [XML Copy Editor](#) , que es Software libre y está disponible para su descarga. Se trata de un editor muy sencillo e intuitivo.

Para validar un xml y su dtd podemos usar XML Copy Editor pulsando F5, si no podemos ejecutar este programa podemos usar la página web <http://xmlvalidator.new-studio.org/> subiendo el fichero y usando el navegador.

## 7. BIBLIOGRAFÍA

[1] [http://ieselcaminas.edu.gva.es/informatica/moodle\\_1112/course/view.php?id=143](http://ieselcaminas.edu.gva.es/informatica/moodle_1112/course/view.php?id=143)

[2] [http://www.mclibre.org/consultar/xml/lecciones/xml\\_dtd.html](http://www.mclibre.org/consultar/xml/lecciones/xml_dtd.html)