Tema 5. DTD (Document Type Definition)

Tema 5. DTD (Document Type Definition)

1. Introducción.

Ya sabemos qué son los documentos XML bien formados: aquellos que siguen las reglas sintácticas de cómo se estructura XML. Ahora nos interesa crear documentos válidos, es decir, documentos que además de estar bien formados, cumplen unas normas que les dan un significado concreto (semántica).

2. Validación XML.

En primer lugar, los documentos XML deben basarse en la sintaxis definida en la especificación XML para ser correctos (documentos bien formados). Esta sintaxis impone reglas como la coincidencia de mayúsculas/minúsculas en los nombres de etiqueta, comillas obligatorias para los valores de atributo, etc. Sin embargo, para tener un control más preciso sobre el contenido de los documentos es necesario un proceso de análisis más exhaustivo.

La validación es la parte más importante dentro de este análisis, ya que determina si un documento XML creado se ciñe a las restricciones descritas en el esquema utilizado para su construcción.

Un esquema lo constituye un conjunto de definiciones que describe una estructura de documento XML, declarando y definiendo todos los tipos de los elementos del documento, el orden de cada tipo de elemento y cualquier atributo, entidad, anotación, comentario o referencia incluida en el mismo.

Controlar el diseño de documentos a través de esquemas aumenta su grado de fiabilidad, consistencia y precisión, facilitando su intercambio entre aplicaciones y usuarios. Cuando creamos documentos XML válidos aumentamos su funcionalidad y utilidad.

La validación se encarga de verificar:

- **La corrección de los datos:** aunque validar contra un esquema no garantiza al 100% que los datos sean correctos, nos permite detectar formatos nulos o valores fuera de rango y por tanto incorrectos.
- **La integridad de los datos:** al validar, se comprueba que toda la información obligatoria está presente en el documento.
- **El entendimiento compartido de los datos:** a través de la validación se comprueba que el emisor y receptor entiendan el documento de la misma manera, que lo interpretan igual.

Por lo tanto un documento válido debe respetar las normas dictadas por el esquema utilizado para su construcción. Dependiendo del tipo de esquema existen varios métodos para validar los documentos XML. Los métodos más usados son la **DTD** de XML, el **XML Schema** de W3C y **RELAX NG** de Oasis.

En este tema veremos los esquemas de formato DTD, dejando para otro tema el formato XML Schema.

3. DTD (Document Type Definitions).

Un DTD permite definir un tipo de documento XML mediante una serie de reglas. Aquellos documentos XML que cumplan dichas reglas serán válidos según dicho DTD. Las reglas expresadas en un DTD indican qué etiquetas se pueden usar en un documento (obligatorias, opcionales), en qué orden deben aparecer, cuáles pueden aparecer dentro de otras, cuáles tienen atributos, valores por defecto, etc.

Crear una definición del tipo de documento (DTD) es como crear nuestro propio lenguaje de marcado para una aplicación específica. Por ejemplo, podríamos crear un DTD que defina una *tarjeta de visita*. A partir de ese DTD, podríamos crear documentos XML que nos permitirían definir tarjetas de visita válidas frente a dicho DTD.

Originalmente los DTD se desarrollaron para ser utilizados con documentos SGML antes de que apareciera XML, por lo que su sintaxis no es tipo XML. Los DTD pueden integrarse en el propio documento XML, pero se suelen colocar en archivos externos para utilizarlos de forma común de manera que el mismo DTD pueda validar varios documentos. La principal pega que se les achaca es que no siguen una sintaxis XML, sino una propia basada en SGML, ni que tampoco permiten definir tipos de datos, aspecto muy importante para el intercambio de datos, bases de datos, etc.

Puesto que XML es un sistema para definir lenguajes, cualquier documentos XML no puede tener un único DTD universal (al contrario de HTML o XHTML que tienen un conjunto de etiquetas fijas). En lugar de eso, quien necesite usar XML para intercambio de datos deberá definir su propio DTD.

Por ejemplo, el *Wall Street Journal Interactive Edition* tiene un DTD para especificar cada edición, con información relativa a las páginas, sumarios, cabeceras, subcabeceras, párrafos, etc.

Más estrictamente definimos DTD como un documento en el cual se declaran las entidades, notaciones y elementos permitidos en aquellos documentos que las utilicen. En concreto, para los elementos se definen sus etiquetas, estableciendo al mismo tiempo cuáles serán sus contenidos y la lista de atributos que pueden utilizar.

El DTD que usará un documento XML se define con una línea de *declaración de tipo de documento* que tiene la forma: <!DOCTYPE>, y se escribe antes que el contenedor principal en el archivo XML.

La declaración <!DOCTYPE> especifica si el DTD está en el propio documento XML (cuando se va a utilizar sólo para dicho documento XML), o en un archivo externo (con la intención de que se pueda usar para varios documentos XML).

La estructura de la línea de declaración es la siguiente:

```
<!DOCTYPE elementoRaiz declaración>
```

en donde:

elementoRaiz: es el nombre del elemento contenedor principal del archivo XML (elemento raíz).
declaración: indica donde están descritas las declaraciones.

Puede ser que las declaraciones estén en el propio documento o en un archivo externo.

a) Si las declaraciones están en el propio documento (DTD interna) la declaración es de la forma:

```
<!DOCTYPE elementoRaiz  [
…declaraciones de elementos…
]>
```

b) Si las declaraciones están en un documento externo (archivos con extensión ".dtd"), puede presentarse dos casos:
   - Que el archivo DTD sea privado y ubicado en el propio ordenador, en una red local o en una intranet. En cuyo caso se usa la palabra SYSTEM acompañado de la URL del archivo DTD.

     ```
     <!DOCTYPE elementoRaiz SYSTEM  "URL">
     ```

   - Que sea público y esté ubicado en una red pública. En este caso se usa la palabra PUBLIC, seguido de un FPI (*Formal Public Identifier*) y termina con la URL del archivo DTD.

     ```
     <!DOCTYPE elementoRaiz PUBLIC "FPI" "URL">
     ```

Un **Identificador Público Formal** (FPI) es una cadena de caracteres con un formato especial que se usa para identificar de forma única un producto, una especificación o un documento.

Está formado por cuatro campos separados por los caracteres //.

1. El primero campo indica: con un signo (-) que la organización no está registrada por la ISO, como de hecho ocurre con el W3C. Si la organización sí está registrada, se sustituye por el signo (+).
2. El segundo campo indica el nombre del organismo responsable del DTD.
3. El tercer campo especifica la clase de documento que se describe junto a la descripción.
4. El cuarto campo indica el idioma que se expresa el DTD.

Por ejemplo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Veamos un ejemplo de uso de un DTD. Supongamos un documento XML que almacena recetas de cocina:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE recetario SYSTEM "recetario.dtd">
<recetario>
    <receta>
        <titulo>Patatas con arroz</titulo>
        <ingredientes>patatas, arroz, agua, ajo, aceite, sal</ingredientes>
        <procedimiento>Pelar, lavar y trocear las patatas… </procedimiento>
    </receta>
    <receta>
        <titulo>Picatostes</titulo>
        <ingredientes>pan, aceite (azúcar opcional)</ingredientes>
        <procedimiento> Con el pan del día anterior, hacer rebanadas … </procedimiento>
    </receta>
    <receta>
        <titulo>Salmorejo</titulo>
        <ingredientes>tomates, ajo, pan, aceite, sal y vinagre </ingredientes>
        <procedimiento>Lavados los tomates, pelarlos junto al ajo y  … </procedimiento>
    </receta>
<recetario>
```

Un posible DTD (archivo *recetario.dtd*) que permite establecer el formato de este documento XML de recetas podría ser el siguiente:

```
<!ELEMENT recetario (receta)*>
<!ELEMENT receta (titulo, ingredientes, procedimiento)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT ingredientes (#PCDATA)>
<!ELEMENT procedimiento (#PCDATA)>
```

4. Declaraciones en los DTD's.

Las posibles declaraciones en un DTD son solamente de cuatro tipos y permiten declarar todos los objetos que pueden aparecer en un documento y sus características.

Las declaraciones se pueden clasificar en dos grupos dependiendo de que se refieran a la estructura lógica o a la estructura física del documento.

Con respecto a la estructura lógica las declaraciones son:

- Declaración de elemento <!ELEMENT>
- Declaración de atributos <!ATTLIST>

Con respecto a la estructura física son:

- Declaración de entidad <!ENTITY>
- Declaración de notación <!NOTATION>

4.1. Estructura lógica.

La estructura lógica de un documento está definida por la declaración de los elementos que pueden utilizarse y por el tipo de sus contenidos; es lo que se domina el *modelo de contenido*. Se utiliza la marca <!ELEMENT>, y esta información se puede completar con declaraciones para las listas de atributos y sus posibles valores mediante la marca <!ATTLIST>. Los atributos se usan para añadir información corta, sencilla y desestructurada a los elementos.

4.1.1. Declaraciones de tipo de elemento.

<!ELEMENT nombre (modelo de contenido)>

En donde:

nombre es el nombre del elemento y se corresponde con el nombre de la etiqueta.

(modelo de contenido) se utiliza para indicar el tipo de contenido para esa etiqueta. Puede ser el nombre de otros elementos, que también se definirán en el TDT y que son elementos hijos, (se pueden poner incluso con sus propios contenidos pero es mejor definirlos aparte) y/o una de las siguientes palabras reservadas #PCDATA, EMPTY o ANY.

Ejemplo 1:

```
<!ELEMENT receta (titulo, ingredientes, procedimiento)>
```

En este ejemplo, el elemento <receta> sólo puede contener los elementos <titulo>, <ingredientes> y <procedimiento>, en este orden. Estos tres elementos a su vez estarán definidos también en la DTD y podrán contener otros elementos.

Siguiendo la definición de elemento anterior, el ejemplo siguiente de documento XML NO sería válido:

```
<receta>
    <tiempo>El tiempo estimado de preparación es de ...</tiempo>
    <titulo>...</titulo>
    <ingredientes>...</ingredientes>
    <procedimiento>...</procedimiento>
</receta>
```

Ejemplo 2:

```
<!ELEMENT salto EMPTY>
```

El elemento <salto> no tiene contenido, por lo tanto la etiqueta tendría el cierre en ella misma: <salto/>

Ejemplo 3:

```
<!ELEMENT cualquiera ANY>
```

El elemento <cualquiera> puede tener cualquier contenido. No se suele utilizar, ya que es conveniente estructurar adecuadamente nuestros documentos XML.

Ejemplo 4:

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

El elemento <nombre> tiene como contenido datos (caracteres) que no contengan marcas. El significado de PCDATA es (*Parsed Character DATA*), texto o datos de tipo carácter que serán analizados por el *parser*. (Por lo tanto no se podrán usar directamente los símbolos &, <, “, etc. que deberán ser sustituidos por sus entidades correspondientes). Debe aparecer entre paréntesis.

Ejemplo 5:

```
<!ELEMENT parrafo (#PCDATA|nota)*>

<parrafo>Hola Pepe:</parrafo>

<parrafo>Espero que cuando recibas la presente <nota>me refiero a la carta</nota> estés bien.</parrafo>
```

El elemento <párrafo> es mixto porque puede contener más de un tipo: datos de tipo carácter que serán analizados y/o puede tener un subelemento <nota> que también deberá ser definido.

Los ejemplos muestran que podemos establecer diversos modelos de contenido. Entendiendo por modelo de contenido un patrón que establece los subelementos aceptados, y el orden en que se aceptan.

Por ejemplo: un modelo sencillo que indique que sólo puede tener un solo tipo de subelemento sería:

```
<!ELEMENT aviso (parrafo)>
```

La coma denota una secuencia. Es decir, en el ejemplo siguiente, el elemento <aviso> debe contener un <titulo> seguido de un <parrafo>.

```
<!ELEMENT aviso (titulo, parrafo)>
```

La barra vertical "|" indica una opción. En el ejemplo siguiente, el elemento <aviso> puede contener o bien un elemento <parrafo> o bien un elemento <grafico>, pero no los dos.

```
<!ELEMENT aviso (parrafo | grafico)>
```

El número de opciones no está limitado a dos, y se pueden agrupar usando paréntesis.

En el siguiente caso, el elemento <aviso> debe contener un elemento <titulo> seguido de un elemento <parrafo> o de un elemento <grafico>.

```
<!ELEMENT aviso (titulo, (parrafo | grafico))>
```

Además, cada partícula de contenido puede llevar un *indicador de frecuencia*, que sigue directamente a un identificador general, una secuencia o una opción, y no pueden ir precedidos por espacios en blanco.

Indicadores de frecuencia	
?	Opcional (0 ó 1 vez)
*	Opcional y repetible (0 ó más veces)
+	Necesario y repetible (1 ó más veces)

(Nota: Si no se señala indicador de frecuencia es que debe aparecer obligatoriamente y sólo una vez.)

Ejemplo 6: Elemento que defina un *aviso*

```
<!ELEMENT aviso (titulo?, (parrafo+, grafico)*)>
```

En este caso, <aviso> puede tener <titulo>, o no (pero sólo uno), y puede tener cero o más conjuntos de (<parrafo>+,<grafico>) donde los párrafos pueden ser más de uno.

4.1.2. Declaraciones de atributo.

<!ATTLIST> permite la declaración de atributos. En general se utiliza un solo <!ATTLIST> para declarar todos los atributos de un mismo elemento, aunque para mayor claridad se puede usar una declaración <!ATTLIST> por cada uno de los atributos del elemento. La sintaxis es:

```
<!ATTLIST nombre-elemento  nombre-atributo  tipo-atributo  valor-por-defecto>
```

En donde:

- nombre-elemento es el nombre del elemento para el cual se define la lista de atributos.
- nombre-atributo es el nombre del atributo. Debe ser un nombre XML válido.
- tipo-atributo indica el tipo de dato utilizado por el atributo. Se puede especificar o el tipo de valor del atributo, o bien una lista de posibles valores. Lo veremos más adelante.
- valor-por-defecto indica un valor por defecto para el atributo, así como su grado de obligatoriedad.

Por ejemplo:

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

En el ejemplo anterior, para declarar la lista de atributos de los elementos <mensaje> y <texto> haríamos lo siguiente:

```
<!ELEMENT mensaje (de, a, texto)>
<!ATTLIST mensaje prioridad (normal | urgente) "normal">
<!ELEMENT texto (#PCDATA)>
<!ATTLIST texto idioma CDATA #REQUIRED>
```

En donde estamos diciendo que el elemento <mensaje> tiene un atributo que se llama prioridad con dos posibles valores "normal o urgente" con valor por defecto "normal". Y el elemento <texto> tiene un atributo que se llama idioma cuyo tipo debe ser una cadena de caracteres, la cual que hay que cumplimentar obligatoriamente pues no hay valor por defecto.

Antes de ver los tipos de atributos y para comprender mejor los ejemplos, comentaremos la declaración valor-por-defecto del atributo, es decir la última parte de la declaración de un atributo. Se puede indicar: un valor por defecto para el atributo, o bien si éste es opcional. Los posibles valores son:

- un valor textual entre comillas (un literal) que representa un valor por defecto para el atributo.
- #IMPLIED. Indica que el atributo es opcional y no se le asigna ningún valor por defecto.
- #REQUIRED. El atributo es obligatorio y no se le asigna ningún valor por defecto.
- #FIXED "valor". El atributo es opcional, pero si se usa deberá tomar obligatoriamente dicho valor concreto.

- Tipos de atributo.

Los tipos de atributos son cuatro: *cadena*, *enumerados*, *notaciones*, y *tokenizados*.

- Tipo cadena, CDATA (datos carácter): representan datos de tipo carácter que no contengan otras etiquetas. Son los más utilizados pues los datos de los documentos XML son de tipo textual.

Ejemplo:

```
<!ATTLIST mensaje fecha CDATA #REQUIRED>
...
<mensaje fecha="15 de Diciembre de 1943">
```

- **Tipos enumerados y notaciones:** Los atributos enumerados son aquellos que sólo pueden contener un valor de entre un número fijo de opciones. Opcionalmente se puede especificar a continuación un valor por defecto. Los valores de la enumeración deben aparecer entre paréntesis y separado por el símbolo |.

Ejemplo:

```
<!ATTLIST mensaje prioridad (normal | urgente) "normal">
```

Existe otro tipo de atributo parecido a la enumeración, llamado de notación (NOTATION). Este tipo de atributo permite declarar que su valor debe ser una notación previamente declarada en el DTD. Lo veremos...

- **Tipos tokenizados**: Sirven para representar palabras reservadas y por ello tienen restricciones léxicas y semánticas. Los posibles tipos de datos utilizados para estos atributos son: ID, IDREF, IDREFS, ENTITY, ENTITIES, NMTOKEN y NMTOKENS. Los comentaremos detalladamente en la siguiente sección.

- Tipos de datos tokenizados en los atributos.

a) Atributos de tipo ID, IDREF e IDREFS.

Los atributos de tipo ID permiten identificar de forma única un elemento dentro de un documento XML. Una vez que el elemento ha sido identificado a través del valor del atributo ID, podemos posteriormente usar atributos de tipo IDREF o IDREFS para referirnos a dicho elemento.

Cuando se usen atributos de tipo ID deben seguirse las siguientes reglas:

- El valor del atributo de tipo ID debe ser único en todo el documento XML.
- Un elemento sólo puede tener un único atributo de tipo ID.
- El valor del atributo ID debe ser #IMPLIED (opcional) o #REQUIRED (obligatorio).

Por ejemplo, para implementar un sencillo sistema de hipervínculos en un documento:

```
<!ELEMENT capitulo (parrafo)*>
<!ATTLIST capitulo referencia ID #IMPLIED>
```

Cada capítulo podría (es opcional…) tener un atributo referencia cuyo valor debe ser único dentro del documento.

Los atributos de tipo IDREF sólo pueden tomar como valores los que ya tuviera otro atributo de tipo ID. Por ejemplo, si ahora definimos el siguiente elemento:

```
<!ELEMENT enlace EMPTY>
<!ATTLIST enlace destino IDREF #REQUIRED>
```

Estamos indicando que el atributo destino del elemento <enlace> es obligatorio y deberá tomar como valor necesariamente alguno de los posibles valores de los atributos ID existentes en el documento.

En este caso, la etiqueta <enlace destino="seccion-3"/> haría referencia a un <capitulo referencia="seccion-3">, de forma que el procesador XML lo podría convertir en un hipervínculo, u otra cosa.

Los atributos de tipo IDREFS indica que podrán tomar como valores una lista de valores de atributos ID existentes en el documento. En la lista se separan los valores mediante espacios. Por ejemplo:

```
<enlace destino="seccion-3 seccion-6 seccion-14"/>
```

b) Atributos de tipo ENTITY y ENTITIES

ENTITY indica que el valor del atributo debe ser una **entidad general externa no analizada** que haya sido declarada en el DTD. Como luego veremos, las entidades externas no analizadas permiten hacer referencia a los tipos de los objetos que no son documentos XML tales como imágenes, sonidos, etc. que se encuentren en archivos externos. Lo veremos...

c) Atributos NMTOKEN y NMTOKENS

El tipo NMTOKEN permite cadenas de caracteres pero sin espacios en blanco. En la práctica este tipo se usa cuando se desea que el valor del atributo sea una sola palabra (sin espacios). Por ejemplo:

```
<!ELEMENT Portugal EMPTY>
<!ATTLIST Portugal idioma NMTOKEN "pt">
    . . .
<España idioma = "pt"/>
```

En el caso de NMTOKENS se refiere a que el atributo puede contener una lista de datos de tipo NMTOKEN separados por espacios en blanco. Por ejemplo:

```
<!ATTLIST Europa idiomas NMTOKENS #IMPLIED>
   . . .
<Europa idiomas = "es fr it en pt de fl"/>
```

▪ Valores de los atributos

Respecto a los valores por defecto que pueda tener un atributo, ya comentamos que pueden usarse literales o bien palabras reservadas. En el caso de un literal, éste deberá aparecer entrecomillado y su texto podrá ser cualquiera o bien un texto de entre los que aparezcan previamente en una lista de enumeración. Por ejemplo:

```
<!ATTLIST figura color CDATA "naranja">
<!ATTLIST semáforo color (rojo | verde | naranja) "naranja">
```

Si se utiliza una palabra reservada, podrá ser una de las tres siguientes:

- #REQUIRED significa que no tiene valor por defecto, ya que es obligatorio especificar este atributo. Por ejemplo:

  ```
  <!ATTLIST usuario correo CDATA #REQUIRED>
  ```

  Es decir, hay que poner necesariamente la dirección de correo del usuario.

- #IMPLIED significa que es opcional y por lo tanto se puede omitir el atributo sin que se adopte automáticamente un valor por defecto. Por ejemplo:

  ```
  <!ATTLIST usuario web CDATA #IMPLIED>
  ```

  Por lo tanto no es necesario poner la web del usuario: puede no tenerla o ser desconocida.

- #FIXED "valor predeterminado". En ese caso el atributo es opcional, pero si se usa debe coincidir con el valor por defecto suministrado. Por ejemplo:

  ```
  <!ATTLIST Europa moneda NMTOKEN #FIXED "euro">
  ```

4.2. Estructura física.

4.2.1. Declaración de entidades.

Las entidades se declaran de diferente forma según su funcionalidad y para ver la forma de describirlas necesitamos primero hacer una clasificación de ellas. Existen tres criterios para realizar dicha clasificación:

- Según el nivel de acceso a ellas, se clasifican en **generales** y de **parámetro**.
- Según si el procesador de XML deba analizarlas o no, se clasifican en **analizadas** y **no analizadas.**
- Que se encuentren dentro del propio documento XML o en un archivo DTD (**internas)**; o en archivos externos ubicados en el mismo ordenador o accesible por red (**externas)**.

- Entidades generales

Las entidades generales se utilizan como mecanismo de sustitución o reemplazo con el fin de simplificar y optimizar los documentos XML. Al ser **siempre analizadas**, deben incluir texto XML bien formado.

El contenido de reemplazo o valor a reemplazar puede indicarse en la propia definición de la entidad (entidad interna) o indicando el archivo que contiene dicho valor (entidad externa)

La **entidades generales internas** se definen como

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

posteriormente, la forma de hacer referencia a ellas es mediante la referencia de identidad *&identificador;*

Por ejemplo, supongamos el DTD *libros.dtd* donde se definen dos entidades

```
<!ENTITY edita "<editorial>Nucama</editorial>">
<!ENTITY texto_copyleft "derecho de copyleft: Dpto. Informatica">

<!ELEMENT libro (autor, editorial, observaciones)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT editorial (#PCDATA)>
<!ELEMENT observaciones (#PCDATA)>
```

Podríamos construir el siguiente documento XML

```
<?xml version="1.0"?>
<!DOCTYPE libro SYSTEM "libro.dtd">
<libro>
    <autor>varios autores</autor>
    &edita;
    <observaciones>&texto_copyleft;</observaciones>
</libro>
```

Cuando se procese el documento XML, el analizador realizará la sustitución:

```
<libro>
  <autor>autores varios</autor>
  <editorial>Nucama</editorial>
  <observaciones>derechos de copyleft para Dpto. Informatica</observaciones>
</libro>
```

Vemos que su uso es como el de una macro de sustitución, de forma que el procesador XML las expande.

La **entidades generales externas** se definen indicando cual es el archivo con el contenido a reemplazar. Puede presentarse dos casos:

- Que el archivo sea privado y esté ubicado en el propio ordenador o en una red local o intranet. En cuyo caso se usa la palabra SYSTEM acompañado de la URL del archivo. Su formato es:

```
<!ENTITY identificador SYSTEM "URL">
```

- Que el archivo sea público y esté ubicado en una red pública. En este caso se usa la palabra PUBLIC, seguida de un FPI (*Formal Public Identifier*) y termina con la URL del archivo. Su formato es:

```
<!ENTITY identificador PUBLIC "FPI" "URL">
```

En donde

identificador es el nombre dado a la entidad.

SYSTEM o PUBLIC indica el tipo de uso: privado o público.

FPI es el identificador público formal.

URL es la información de localización de la entidad que se utiliza.

Como ejemplo para este caso podemos modificar el último ejemplo de la página anterior de forma que la entidad interna copyleft_texto la convertiremos en externa de ámbito privado.

```
<!ENTITY texto_copyleft SYSTEM "derechos.txt">
```

En este caso el archivo *derechos.txt* contiene el texto *derecho de copyleft: Dpto. Informatica*; y será analizada. Ahora, al igual que antes, cuando aparezca en el documento XML la referencia de identidad

<observaciones>**&texto_copyleft;**</observaciones>

El *parser* los sustituye por su contenido

<observaciones>**derechos de copyleft para Dpto. Informatica**</observaciones>

- Entidades de parámetro.

Se usan como mecanismo de sustitución, pero ahora dentro del propio DTD. Se suelen usar para definir determinados datos que se tienen que utilizar repetidamente en el DTD para no tener que escribirlos varias veces. Su declaración es la siguiente:

```
<!ENTITY % parametro "definición">
```

La forma de referenciar a la entidad NO es la habitual de *&parámetro;* sino *%parametro;*

Es importante tener en cuenta que **solo se puede hacer referencia a ellas en la propia DTD**, nunca en el documento XML propiamente dicho.

Ejemplo:

```
<!ENTITY % gama_colores "(amarillo | naranja | rojo | azul |verde | granate | negro)">
<!ELEMENT silla (modelo, material)>
<!ATTLIST silla color %gama_colores; "negro">
```

En este caso la sustitución se haría en el propio DTD quedando:

```
<!ATTLIST silla color (amarillo |naranja | rojo | azul | verde | granate | negro) "negro">
```

- Entidades no analizadas.

A veces se necesita incluir en los documentos XML, referencias a datos externos que no de tipo texto, como imágenes, videos, hojas de cálculo u cualquier otro formato de datos binarios. Esta información no debe ser analizada por el procesador XML, ya que no son datos textuales, pero podemos indicar de alguna manera, cual es el formato de dichos datos para que una aplicación externa pueda conocer dichos tipos.

Estas entidades son siempre externas (hacen referencia a datos no textuales ubicados en archivos). La declaración de las entidades no analizadas pueden ser privadas o públicas:

```
<!ENTITY identificador SYSTEM  "URL"  NDATA formato>
<!ENTITY identificador PUBLIC "FPI" "URL" NDATA formato>
```

El valor de formato es una notación para indicar el tipo o el formato del objeto. Por ejemplo si es una imagen, se pone el formato que tiene dicha imagen:

```
<!ENTITY logo SYSTEM  "./logos/empresa.gif"  NDATA gif>
```

Importante: Las entidades no analizadas, como la del ejemplo anterior, pueden usarse como tipo de atributo de un elemento. (Recordar el apartado de atributos…), pero necesitan de una declaración de notación que veremos en el apartado siguiente.

4.2.2. Declaraciones de notación.

Las aplicaciones que manipulan o usan documentos XML, necesitan de una notación para gestionar aquellos datos que no sean de tipo XML. Con las notaciones se permite especificar dos aspectos: bien la aplicación que deberá procesar dichos datos que no son de tipo XML, o bien el tipo MIME del archivo que contiene dichos datos.

La sintaxis básica de la declaración <!NOTATION> es:

```
<!NOTATION nombre SYSTEM  "identificador_externo">
```

```
<!NOTATION nombre PUBLIC "FPI" "identificador_externo">
```

Mediante SYSTEM o PUBLIC se establece si la localización de la aplicación auxiliar que debe procesar dichos datos, o el tipo MIME de dichos datos, es de un dominio privado o público.

El identificador_externo indica bien la URL de la aplicación auxiliar o bien el tipo MIME del archivo.

Ejemplos:

```
<!NOTATION dibujopng SYSTEM  "c:\programas\visordepngs.exe">
<!NOTATION dibujopng SYSTEM  "img/png">
```

En el caso de PUBLIC debe ir seguido de un FPI y después el identificador_externo. Por ejemplo:

```
<!NOTATION dibujogif PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format 89a//EN"
"c:\programas\visordegifs.exe">
```

- Uso de las notaciones.

Un primer caso de uso es cuando mediante ATTLIST declaramos atributos de tipo NOTATION. En este caso la notación debe estar previamente declarada. Por ejemplo:

```
<!NOTATION png SYSTEM " img/png">
<!NOTATION gif SYSTEM " img/gif">
<!NOTATION jpg SYSTEM " img/jpg">

<!ELEMENT imagen EMPTY>
<!ATTLIST imagen archivo CDATA #REQUIRED>
<!ATTLIST imagen formato NOTATION (png | gif) #IMPLIED>
…
<imagen archivo = "sunrise.png" formato="png" />
```

El atributo formato se declara como NOTATION y sólo podrá tomar si se usa, uno de los dos valores de las notaciones de la lista previamente declaradas. Importante: "los valores de los atributos que son de tipo NOTATION han tenido que ser previamente declarados".

El segundo caso donde se usan las notaciones surge de una forma indirecta cuando mediante ATTLIST declaramos atributos de tipo ENTITY o ENTITIES.

Estas entidades sólo pueden ser entidades externas no analizadas. Recordemos que en la definición de estas entidades, el parámetro NDATA formato indica el tipo o formato de dichos datos que no son de tipo XML (imágenes, vídeos, etc.). Pero seguidamente también debemos informar mediante una notación, de cuáles son las aplicaciones, o el tipo MIME, que deben manipular estos datos que están en dicho formato.

Supongamos que tenemos declarada en nuestro DTD la entidad externa no analizada siguiente junto a la notación correspondiente:

```
<!ENTITY FotoManolete SYSTEM "Manolete.png" NDATA png>
<!NOTATION png SYSTEM "c:\programas\visordepngs.exe">
```

Y que el DTD contiene la siguiente declaración de atributo:

```
<!ATTLIST contacto foto ENTITY #IMPLIED>
```

Esto indica que el atributo foto debe tener como valor un entidad externa no analizada previamente declarada. En nuestro caso nos permitiría que tomara el valor de la entidad FotoManolete.

```
<contacto foto="FotoManolete">
```

Si necesitamos que un atributo pueda tomar como valor no solo el de una única entidad, sino los valores de una o varias entidades, se indicará con el tipo ENTITIES. De esta forma se referencia a una lista de entidades previamente definidas. Por ejemplo, si declaramos:

```
<!NOTATION png SYSTEM "c:\programas\visordepngs.exe">
<!ENTITY FotoManolete-chica SYSTEM "ManoleteCh.png" NDATA png>
<!ENTITY FotoManolete-grande SYSTEM "ManoleteGr.png" NDATA png>
<!ATTLIST contacto fotillos ENTITIES #IMPLIED>
```

Estamos indicando que el atributo fotillos puede tener como valores un conjunto de algunas de las entidades anteriormente definidas (FotoManolete-chica, FotoManolete-grande). Por lo tanto un documento XML podría tener:

```
<contacto fotillos="FotoManolete-chica FotoManolete-grande">
```

Vemos como al atributo fotillos se le ha asignado varias entidades (las cuales deben separarse por espacios en blanco). El resultado es que podemos tener varias fotos para un contacto.