Validación de Documentos XML - DTDs

1 INTRODUCCION

Una de las características más potentes de XML es que puede ser utilizado como metalenguaje para definir nuevos lenguajes de marcas conocidos como *tipos de documento*, *aplicaciones o dialectos XML*. Cuando se utiliza un dialecto XML para la creación de un documento, la comprobación de que un documento XML esté bien formado deja de ser suficiente y se hace necesario un mecanismo de validación que permita comprobar que su estructura se ajusta a la definida para dicho dialecto. Dicho mecanismo de validación se apoyará en la definición de una gramática formal para cada dialecto mediante otros tipos especiales de documentos denominados *esquemas XML*.

Un esquema XML es un documento publicable que proporciona una descripción formal de un dialecto XML, normalmente en términos de restricciones acerca de la estructura y contenido de los documentos de ese tipo, que va más allá de las limitaciones sintácticas básicas que impone el propio lenguaje XML. Estas restricciones se expresan generalmente mediante una combinación de reglas gramaticales que rigen el orden de los elementos, las condiciones que han de cumplir los contenidos, los tipos de datos del contenido de los elementos y del valor de sus atributos, y normas más especializadas, tales como unicidad de los elementos y restricciones de integridad referencial.

La validación de documentos XML asegura que los documentos cumplen con un conjunto mínimo de requisitos, buscando aquellos defectos que podrían dificultar o impedir su procesamiento. En términos generales, la validación debería de llevarse a cabo al menos en cuatro niveles:

- Validación estructural: tiene que ver con el uso y colocación de elementos de marcado y atributos. Es la más importante y para la que mejor preparados están los esquemas.
- Comprobación de tipos de datos: es útil a menudo, sobre todo en los documentos orientados a datos, pero no está ampliamente soportada.
- Comprobación de integridad: relacionada con el estado de los enlaces entre los nodos y los recursos, es menos común y un tanto problemática de definir.
- Comprobación de las reglas de negocio: implica la realización de diversas comprobaciones, tales como control de ortografía, sumas de control, etc. Las reglas de negocio son a menudo controladas por las aplicaciones.

Existen dos tipos de esquemas XML, los *DTD* y los *XML Schema*. En este documento se estudia el primero de ellos.

2 DTD

Un DTD usa una gramática formal para especificar el conjunto de reglas que definen la estructura de un determinado tipo de documento XML. Lo hace de la forma siguiente:

- Declarando un conjunto de elementos permitidos. Esto se puede ver como el *vocabulario*.
- Definiendo una especificación de contenido para cada elemento, que consiste en un patrón que indica qué elementos o datos puede contener, en qué orden, en qué cantidad, y si son obligatorios u opcionales. Esto se puede ver como la *gramática*.
- Declarando un conjunto de atributos permitidos para cada elemento. Para cada uno de ellos se define su nombre, tipo de datos, valor por defecto (si existe), y el comportamiento (por ejemplo, si es necesario u opcional).
- Proporcionando una variedad de mecanismos para hacer más fácil la gestión del modelo, por ejemplo, el uso de entidades de parámetro y la capacidad de importar partes del modelo a partir de archivos externos.

Cuando se valida un documento XML contra un DTD, éste se considera válido si satisfice los cuatro criterios siguientes:

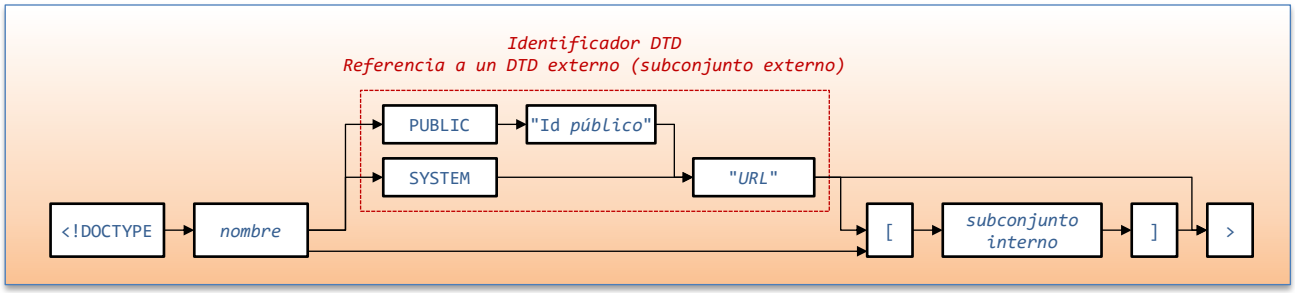
1. Es un documento XML bien formado.
2. Incluye una declaración de tipo de documento.
3. El elemento raíz coincide con el especificado en la declaración de tipo de documento.
4. Satisface todas las restricciones que se definen en el DTD especificado en la declaración de tipo de documento.

Sin embargo, no todos los dialectos XML pueden ser descritos con precisión a través de un DTD, debido limitaciones propias de estos últimos. Estas limitaciones son:

- No son documentos XML.
- No dan soporte a los espacios de nombres.
- No es posible especificar tipos de datos para los valores de los atributos ni para el contenido de los elementos.
- Las posibilidades para definir especificaciones de contenido son muy limitadas: poca flexibilidad para elementos con contenido mixto o limitaciones para establecer cardinalidades.

2.1 Declaración de Tipo de Documento.

La declaración de tipo de documento o *DOCTYPE declaration* asocia un documento XML con un DTD. Esta declaración se incluye dentro del prólogo del documento, después de la declaración XML, y su sintaxis es la siguiente:



El *nombre* de la declaración DOCTYPE tiene que ser el mismo que el nombre del elemento raíz del documento.

El DTD que se asocia al documento puede estar formado por:

- Un conjunto de declaraciones que recibe el nombre de *subconjunto externo* y que se guardan en un recurso externo (un fichero, por ejemplo).
- Un conjunto de declaraciones que recibe el nombre de *subconjunto interno* y que forman parte de la propia declaración DOCTYPE.
- Por ambos subconjuntos.

Esto da lugar a tres tipos de DTDs: externo, interno y mixto.

2.1.1 DTD Interno

Es la forma más simple de DTD ya que forma parte de la propia declaración DOCTYPE.

La desventaja obvia de este enfoque es que el DTD solo se puede utilizar para validar el documento en el que se incluye, es decir, no se puede volver a utilizar para validar otras instancias del tipo de documento que describe.

Sin embargo, hay situaciones en las que puede resultar útil. Por ejemplo, durante el desarrollo de un nuevo DTD, su declaración interna facilita su sincronización con el documento de prueba.

Ejemplo:

2.1.2 DTD Externo

La declaración DOCTYPE hace referencia a DTD externo a través del denominado *identificador DTD*, que puede ser de dos tipos: SYTEM y PUBLIC.

Identificador SYSTEM

Está formado por la palabra SYSTEM seguida de la URL del recurso que contiene el DTD (p.e. un fichero de texto). Este tipo de identificador no debería de ser usado en documentos que vayan a ser intercambiados entre diferentes sistemas, ya que podrían perder su significado en el momento en el que el documento abandone el sistema de origen (por ejemplo, si utilizamos una *URL file*).

Ejemplos:

```
<?xml version="1.0"?>
<!DOCTYPE mensaje SYSTEM "file://c|/mensaje.dtd">
```

```
<?xml version="1.0"?>
<!DOCTYPE mensaje SYSTEM "http://www.misitio.com/mensaje.dtd">
```

Identificador PUBLIC

Está formado por la palabra PUBLIC seguida de un identificador público seguido de la URL del recurso que contiene el DTD. El identificador público es un identificador único para el DTD que se puede proporcionar de tres formas:

- Mediante un FPI (Formal Public Identifier).
- Mediante un URI (Uniform Resource Identifier).
- Mediante un UUID (Universally Unique Identifier).

Este identificador único puede ser usado por un procesador XML en combinación con un catálogo XML para intentar recuperar el DTD por sí mismo. Si esto no es posible, tendría que hacer uso de la URL para intentar recuperarlo.

Un ejemplo de uso de este identificador se encuentra en las declaraciones DOCTYPE de los documentos XHTML:

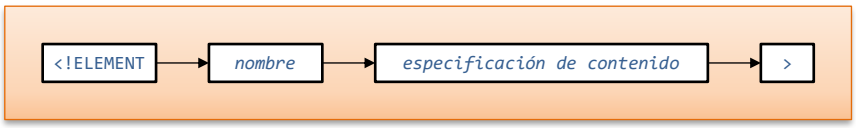
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

2.1.3 DTD Mixto

La declaración DOCTYPE combina una referencia SYSTEM o PUBLIC a un DTD externo con un DTD interno.

2.2 Declaración de elementos: modelos de contenido

Los elementos forman la estructura principal de un documento XML. Cada elemento utilizado en un documento XML válido debe ser declarado en el DTD correspondiente mediante una ***declaración de elemento*** con la sintaxis:



donde *nombre* representa un nombre XML válido para el elemento que se declara y *especificación de contenido* es una expresión que utiliza una gramática simple para especificar con precisión qué está y qué no está permitido como contenido del elemento que se declara (cosas como qué hijos puede tener, en qué orden o cuántos). La tabla siguiente muestra varias especificaciones de contenido básicas:

Sintaxis	Descripción del contenido
EMPTY	Elemento vacío.
(#PCDATA)	Texto analizado.
(*hijo*)	Un único elemento hijo.
(*hijo1*,*hijo2*,…)	Secuencia de elementos hijo respetando el orden especificado.
(*hijo1*|*hijo2*|…)	Alternativa: solamente uno de los elementos hijo de la lista.
(#PCDATA|*hijo1*|*hijo2*|…)*	Contenido mixto: combinación de texto analizado y elementos hijo.
ANY	Elemento sin restricción de contenidos.

2.2.1 EMPTY

El elemento más simple que se puede declarar es el elemento vacío. Su especificación de contenido consta únicamente de la palabra `EMPY`. Ejemplo:

```
<!-- declaración -->
<!ELEMENT a EMPTY>

<!-- ejemplos válidos según especificación de contenido -->
<a/>
<a></a>
```

2.2.2 #PCDATA

El contenido `#PCDATA` (*Parsed Character DATA*) está formado por texto que será analizado por el parser llevando a cabo la expansión de las entidades que incluya. Precisamente por tratarse de texto analizado, el carácter `&` solamente podrá formar parte de este tipo de contenido si se incluye mediante la entidad `&` o dentro de una sección `CDATA`. Por la misma razón, los caracteres `<` y `>` solamente podrán formar parte de este tipo de contenido si se incluyen mediante las entidades `<` y `>` respectivamente, o dentro de una sección `CDATA`.

```
<!-- declaración -->
<!ELEMENT titulo (#PCDATA)>

<!-- ejemplos válidos según especificación de contenido -->
<titulo>XML y DTDs</titulo>
<titulo>XML &amp;amp; DTDs</titulo>
<titulo><![CDATA[XML & DTDs]]></titulo>

<!-- ejemplo no válido -->
<titulo>XML & DTDs</titulo>
```

2.2.3 Modelos de Contenido para elementos hijo

La especificación del lenguaje XML utiliza el término **modelo de contenido** para referirse a la forma en que se describe qué elementos hijo puede contener un elemento padre, en qué orden y cuántas ocurrencias de cada uno.

Un modelo de contenido describe la forma en la que elementos hijo se estructuran como contenido de un elemento padre a través de una sintaxis que emplea un conjunto específico de símbolos. El significado de estos símbolos se describe en la tabla siguiente:

NOTA: con objeto de simplificar los ejemplos que se muestran en este apartado, en todos ellos se omite la declaración de los elementos hijo suponiendo que son elementos EMPTY.

Símbolos	Función	Ejemplos de Declaración	Ejemplos válidos
,	Declarar secuencias de elementos hijo: los elementos hijo estarán contenidos en el elemento padre en el orden que especifica la secuencia.	`<!ELEMENT a (b,c,d)>`	`<a><b /><c /><d /></a>`

|	Declarar alternativas. Solamente se incluye uno de los elementos de la lista.	`<!ELEMENT a (b|c|d)>`	`<a><b/></a>`
			`<a><c/></a>`
			`<a><d/></a>`

?	Asignar cardinalidad 0 o 1 a un elemento hijo, secuencia o alternativa.	`<!ELEMENT a (b?,c)>`
		
		`<!ELEMENT a (b,c)?>`
			`<a><c/></a>`
			`<a><b/><c/></a>`
			
			`<a></a>`
			`<a><b/><c/></a>`

*	Asignar cardinalidad 0 o varios a un elemento hijo, secuencia o alternativa.	`<!ELEMENT a (b*,c)>`
		
		`<!ELEMENT a (b,c)*>`
			`<a><c/></a>`
			`<a><b/><c/></a>`
			`<a><b/><b/><c/></a>`
			
			`<a></a>`
			`<a><b/><c/></a>`
			`<a><b/><c/><b/><c/></a>`

+	Asignar cardinalidad 1 o varios a un elemento hijo, secuencia o alternativa.	`<!ELEMENT a (b+,c)>`
		
		`<!ELEMENT a (b,c)+>`
			`<a><b/><c/></a>`
			`<a><b/><b/><c/></a>`
			
			`<a><b/><c/></a>`
			`<a><b/><c/><b/><c/></a>`

La ausencia de símbolo de cardinalidad se puede interpretar como la obligatoriedad de incluir exactamente una ocurrencia del elemento hijo o del grupo (secuencia o alternativa).

En modelo de contenido más simple será aquel que define un único elemento hijo obligatorio:

```
<!-- declaración -->
<!ELEMENT padre (hijo)>

<!-- ejemplo válido -->
<padre>
    <hijo/>
</padre>
```

Sin embargo, es posible declarar modelos de contenido de diferente complejidad mediante la especificación de secuencias o alternativas, su anidación en varios niveles de profundidad y la especificación de cardinalidades en elementos, secuencias o alternativas, tal y como se demuestra en los ejemplos siguientes:

Ejemplo 1: cardinalidad de elementos individuales.

```
<!-- declaración -->
<!ELEMENT padre (a, b?, c+)>

<!—ejemplos válidos -->
<padre>
    <a/><c/>
</padre>
<padre>
    <a/><c/><c/>
</padre>
<padre>
    <a/><b/><c/>
</padre>
<padre>
    <a/><b/><c/><c/>
</padre>

<!—ejemplos no válidos -->
<padre>
    <a/><b/>
</padre>
<padre>
    <a/><b/><b/><c/>
</padre>
```

Ejemplo 2: anidación de una alternativa dentro de una secuencia.

```
<!-- declaración -->
<!ELEMENT padre (a,(b|c),d)>

<!-- ejemplos válidos -->
<padre>
    <a/><b/><d/>
</padre>
<padre>
    <a/><c/><d/>
</padre>
```

Ejemplo 3: anidación de una secuencia dentro de una alternativa.

```
<!-- declaración -->
<!ELEMENT padre (a|(b,c)|d)>

<!-- ejemplos válidos -->
<padre>
    <a/>
</padre>
<padre>
    <b/><c/>
</padre>
<padre>
    <d/>
</padre>
```

Ejemplo 4: anidación con tres niveles de profundidad.

```
<!-- declaración -->
<!ELEMENT padre (a|(b,(c|d))|e)>

<!-- ejemplos válidos -->
<padre>
   <a/>
</padre>
<padre>
   <b/><c/>
</padre>
<padre>
   <b/><d/>
</padre>

<padre>
   <e/>
</padre>
```

Ejemplo 5: anidación con tres niveles de profundidad y cardinalidad a nivel de elemento y grupo.

```
<!-- declaración -->
<!ELEMENT padre (a?,(b|(c,d)+),e)*>

<!-- ejemplos válidos -->
<padre></padre>
<padre>
   <a/><b/><e/>
   <a/><c/><d/><c/><d/><e/>
</padre>
<padre>
   <c/><d/><e/>
   <b/><e/>
   <a/><c/><d><e/>
   <c/><d/><c/><d/><c/><d/><e/>
</padre>

<!-- ejemplo no válido -->
<padre>
   <a/><e/>
   <b/><c/><d/><c/><d/><e/>
</padre>
```

2.2.4 Modelo de contenido mixto

Este modelo de contenido es el que utilizaremos cuando un elemento pueda contener tanto texto como otros elementos hijo. No obstante, está sujeto a restricciones que limitan la flexibilidad a la hora de utilizar este modelo de contenido.

La sintaxis genérica para la declaración de elementos con contenido mixto es la siguiente:

```
<!ELEMENT elemento (#PCDATA | hijo1 | hijo2 | …)*>
```

El contenido #PCDATA deberá aparecer el primero en la lista y no será posible especificar ningún tipo de cardinalidad para los elementos hijo de forma individual ni establecer el orden en el que deben aparecer.

2.2.5 ANY

El elemento que se declara podrá contener cualquier combinación bien formada de texto y elementos sin más restricción que la de que los elementos que vayan a formar parte del contenido tengan que estar declarados en el DTD.

Se suele utilizar de forma provisional en la fase de desarrollo de los DTD cuando aún no se tienen claras las restricciones que se han de definir para el contenido de un determinado elemento. Sin embargo, es muy inusual, considerándose incluso mala práctica, que este tipo de contenido aparezca en DTDs ya finalizados.

2.3 Declaración de atributos

Los atributos proporcionan información acerca de los elementos definiendo propiedades de los mismos. Dentro de los elementos XML se puede especificar cualquier número de atributos en forma de pares nombre-valor siempre que el nombre no se repita.

La declaración de atributos para los elementos XML dentro de un DTD se realiza mediante declaraciones <!ATTLIST …>. En cada declaración <!ATTLIST …> se declara una lista de uno o más atributos para un determinado elemento XML. La sintaxis básica es:

```
<!ATTLIST nombre_elemento
   nombre_atributo_1 tipo_atributo valor
   nombre_atributo_2 tipo_atributo valor
      ⋮
   nombre_atributo_n tipo_atributo valor
>
```

2.3.1 Tipos de atributos

Los tipos de atributo que se pueden declarar en un DTD pertenecen a una de las tres categorías básicas siguientes:

- String.
- Tokenized.
- Enumerated.

Cada categoría determina como debería de ser procesado el valor asignado al atributo.

La tabla siguiente enumera los tipos de atributo que se pueden especificar en la declaración de una lista de atributos, su descripción y la categoría a la que pertenecen:

TIPO	DESCRIPCIÓN	CATEGORÍA
CDATA	El valor del atributo es una cadena de caracteres en la que no se identifica un tipo de datos específico.	STRING
ID	El valor asignado a un atributo de este tipo es un identificador único que ha de comenzar con un carácter alfabético, el guion bajo '_' o el carácter dos puntos ':'.	TOKENIZED
IDREF	El valor asignado a un atributo de este tipo en un elemento determinado tiene que hacer referencia al valor asignado a un atributo de tipo ID en el mismo elemento o en otro elemento. El atributo que referencia y el referenciado no tienen que tener el mismo nombre.	TOKENIZED
IDREFS	Uno o varios valores IDREF separados por espacios en blanco.	TOKENIZED
ENTITY	El nombre de una entidad definida en el DTD.	TOKENIZED
ENTITIES	Uno o varios nombres de entidades separados por espacios en blanco.	TOKENIZED

NMTOKEN	Para asignar un valor a los atributos de este tipo de atributo solo se pueden utilizar los caracteres siguientes: letras minúsculas, letras mayúsculas, números, punto ".", guion medio "-", guion bajo "_" y el carácter dos puntos ":".	TOKENIZED
NMTOKENS	Uno o varios valores NMTOKEN separados por espacios en blanco.	TOKENIZED
NOTATION	El nombre de una notación.	ENUMERATED
(*valor1*|*valor2*|…)	El valor de la lista tiene que ser uno de los especificados en la lista de valores entre paréntesis separados por el carácter "|".	ENUMERATED

2.3.2 Valor

Con la última parte en la declaración de cada atributo se pueden especificar cuatro opciones diferentes con respecto a su valor:

- **"*valor*"**: el atributo se puede omitir, en cuyo caso, se le asignará el *valor* escrito entre comillas.
- **#REQUIRED**: el atributo es obligatorio. Si se omite el documento XML no será válido.
- **#FIXED "*valor*"**: el atributo es obligatorio y su valor ha de ser el especificado. En caso contrario el documento no será válido.
- **#IMPLIED**: el atributo se puede omitir y no tendrá ningún valor por defecto.

2.4 ENTIDADES

En XML las entidades constituyen una potente herramienta de uso común que facilita tanto la creación de documentos XML como la definición de DTDs. Entre los beneficios que proporciona el uso de entidades podemos destacar los siguientes:

- Hacen posible que caracteres con significado especial formen parte del contenido #PCDATA de los elementos o del valor de los atributos, ya que la especificación del lenguaje XML prohíbe su uso dentro de este tipo de contenido si empleamos su representación normal.
- Facilitan la utilización de caracteres de uso poco frecuente que resulten difíciles o imposibles de obtener a través del teclado.
- Proporcionan una representación abreviada para cadenas de caracteres y bloques de texto que se usen con frecuencia.
- Facilitan la inclusión de datos no textuales.
- Ofrecen un modo de organizar y mantener tanto documentos XML como DTDs de un modo eficiente.
- Permiten el ahorro de código y facilitan el mantenimiento de los documentos.

Podemos diferenciar cuatro tipos de entidades:

- Entidades predefinidas (built-in).
- Entidades carácter.
- Entidades generales.
- Entidades paramétricas.

2.4.1 Entidades predefinidas

Los caracteres &, <, >, ", ', no están permitidos dentro del contenido #PCDATA de los elementos ni dentro del valor de los atributos en su representación normal. Para que puedan formar parte de este tipo de contenido es necesario referenciarlos a través de la

correspondiente entidad predefinida. La tabla siguiente muestra la correspondencia entre estos caracteres y las entidades correspondientes:

CARÁCTER	ENTIDAD
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&apos;

2.4.2 Entidades carácter

Las entidades carácter facilitan la utilización de caracteres de uso poco frecuente que resultan difíciles de introducir a través del teclado. Algunos editores de texto proporcionan diferentes métodos de introducción de tales caracteres. Por ejemplo, si tuviésemos que usar la letra griega omega (Ω) en un documento creado con Microsoft Word podríamos escribir la secuencia de caracteres 3a9 y pulsar a continuación la combinación de teclas Alt+X. Con esto, la secuencia se sustituye inmediatamente por la letra griega. Sin embargo, en los documentos XML no es necesario recurrir a estos métodos de introducción de caracteres de este tipo, sino que podemos hacer uso de las entidades carácter, ya que con ellas podemos referenciar a cualquier carácter.

La escritura de este tipo de entidades se rige por una sintaxis similar a las anteriores:

```
&#número;
```

Donde *número* es una secuencia de dígitos decimales o hexadecimales que representan el código Unicode (Unicode code point) del carácter referenciado. Los números hexadecimales irán precedidos del carácter `X`. Así, por ejemplo, para referenciar a la letra griega omega podemos utilizar la entidad `&#937;` o la entidad `&#X03A9`.

2.4.3 Entidades generales

Las entidades generales deben declararse en un DTD antes de que se puedan ser utilizadas en un documento XML y según la ubicación de los datos referenciados pueden ser internas o externas.

Según la naturaleza de los datos a los que hacen referencia podemos diferenciar dos tipos de entidades generales:

- **Analizadas:** proporcionan una representación abreviada de bloques de texto de uso frecuente dentro de un documento XML. Estos bloques de texto reciben el nombre de texto de sustitución y su longitud no está limitada. Cuando un parser XML analiza el documento XML las sustituye por el texto correspondiente, pasando a formar parte del contenido XML que se será sometido a análisis sintáctico.
- **No analizadas:** representan datos binarios que no son analizados por el parser XML, sino que será la aplicación que lea el documento XML la encargada de procesarlos.

2.4.3.1 Entidades analizadas internas

El texto referenciado forma parte de la declaración de la entidad en el DTD, cuya sintaxis es:

```
<!ENTITY nombre "texto de sustitución">
```

Dentro del documento XML utilizaremos las entidades para hacer referencia al texto de sustitución con la sintaxis habitual: `&nombre;`.

El texto de sustitución puede estar formado por cualquier contenido XML bien formado:

```
<!ENTITY no-gps "<latitud /><longitud />">
```

También puede incluir entidades:

```
<!ENTITY no-dir "La dirección para esta localización es &quot;desconocida&quot;">
```

En cualquier caso, en la declaración de una entidad el texto de sustitución no puede contener ninguna referencia a ella misma. Por tanto, la declaración siguiente es incorrecta:

```
<!ENTITY no-dir "La dirección para esta localización es &no-dir;">
```

2.4.3.2 Entidades analizadas externas

La declaración de entidades generales dentro de un DTD puede llegar a dificultar su legibilidad debido a la ausencia de límites en la longitud de los textos de sustitución. La solución a este problema está en la declaración de entidades externas en las que el texto de sustitución se encuentra en un archivo externo. El enlace al archivo externo se especifica en la declaración de la entidad mediante un identificador SYSTEM o mediante un identificador PUBLIC. La sintaxis para cada caso es la siguiente:

Identificador del sistema

```
<!ENTITY nombre SYSTEM "URI">
```

Identificador público

```
<!ENTITY nombre PUBLIC "identificador_único" "URI">
```

El identificador público es un identificador único que será utilizado en combinación con un catálogo XML para localizar el archivo externo. Junto a este identificador se puede especificar de forma opcional una URI que permita localizar el archivo externo en caso de que no sea posible hacerlo mediante el identificador único.

2.4.3.3 Entidades no analizadas externas

Las entidades no analizadas representan contenido binario que reside en un fichero externo. La declaración de una entidad de este tipo incluye una especificación del formato del contenido binario al que se hace referencia mediante una notación que ha de ser declarada en el DTD con una sintaxis que adopta una de las formas siguientes:

- `<!NOTATION nombre PUBLIC id_público>` declara una notación a partir de un id público que hace referencia a un formato considerado como un estándar. Ejemplos:

```
<!NOTATION jpg PUBLIC "JPG 1.0">
<!NOTATION gif PUBLIC "GIF 1.0">
<!NOTATION png PUBLIC "PNG 1.0">
```

- `<!NOTATION nombre PUBLIC id_público id_sistema>` en este caso el identificador público se interpreta de la misma forma, pero se incluye además un identificador del sistema que debería ayudar a las aplicaciones que lean el documento XML a encontrar los recursos necesarios para procesar los datos binarios. En este sentido, el identificador del sistema puede ser un tipo MIME, la ruta a otra aplicación capaz de procesar los datos o simplemente cualquier identificador reconocible por dichas aplicaciones. Los ejemplos siguientes utilizan un tipo MIME:

```
<!NOTATION jpg PUBLIC "JPG 1.0" "image/jpeg">
<!NOTATION gif PUBLIC "GIF 1.0" "image/gif">
<!NOTATION png PUBLIC "PNG 1.0" "image/png">
```

- `<!NOTATION nombre SYSTEM id_sistema>` en este caso el identificador del sistema es el único podrán utilizar las aplicaciones que lean el documento XML para encontrar los recursos necesarios para procesar los datos binarios. De la misma forma que en el caso anterior, este identificador puede ser un tipo MIME, la ruta a otra aplicación capaz de procesar los datos o simplemente cualquier identificador reconocible por dichas aplicaciones. Los ejemplos siguientes utilizan un tipo MIME:

```
<!NOTATION jpg SYSTEM "image/jpeg">
<!NOTATION gif SYSTEM "image/gif">
<!NOTATION png SYSTEM "image/png">
```

Una vez declaradas las notaciones oportunas, es posible declarar entidades no analizadas según la sintaxis siguiente:

2.4.4 Entidades paramétricas

Al igual que las entidades generales, éstas también se utilizan para proporcionar una representación abreviada de un texto de sustitución. Sin embargo, en este caso no se pueden utilizar dentro del contenido XML, quedando restringido su uso al interior del DTD.

Al contrario que en el los otros tipos de entidades, el texto de sustitución de la declaración de una entidad paramétrica puede contener trozos de código DTD, facilitando la creación de DTDs modulares a partir de múltiples archivos, y por tanto, la compartición y reutilización de código.

Su declaración es bastante similar a la de las entidades generales, con la única diferencia de que incluye el carácter `%` entre la palabra reservada `ENTITY` y el nombre de la entidad:

```
<!ENTITY % nombre "texto_de_sustitución">
```

También es posible declarar entidades paramétricas externas, tanto privadas como públicas:

```
<!ENTITY % nombre SYSTEM "URI">
```

```
<!ENTITY % nombre PUBLIC "identificador_único">
```

```
<!ENTITY % nombre PUBLIC "identificador_único" "URI">
```

Dentro de un DTD se usan las entidades paramétricas mediante la sintaxis *%nombre;*.