TEMA 2: LA DECLARACIÓN DEL TIPO DOCUMENTO.

1.- INTRODUCCIÓN.

XML brinda al autor del documento la posibilidad de definir valores por defecto para los atributos así como ampliar la funcionalidad de las entidades, que hasta ahora se han utilizado únicamente para escapar caracteres, pero que pueden utilizarse en más situaciones, siendo el propio usuario de XML el que defina su valor.

Los documentos XML con los que hemos trabajado en capítulos anteriores eran esencialmente libres en sus formas y contenidos, es posible sin embargo, declarar ciertas restricciones adicionales a las ya conocidas de buena formación, útiles en algunos contextos y especificar, por ejemplo, qué elementos y qué atributos están permitidos en el documento y cuáles son los contenidos que éstos pueden tomar.

¿QUÉ ES Y PARA QUÉ SIRVE UN TIPO?

Intuitivamente todos conocemos lo que significa un tipo, o una clase: un conjunto de especimenes diferenciados del resto por alguna característica. Si nos ceñimos a los documentos la idea es la misma, podríamos pensar en cartas, currículo, recibos, etc. cada una de estas clases de documento con una estructura, un contenido o unas propiedades únicas que permiten agruparlos y hablar así del tipo carta, el tipo de documento currículo o el tipo recibo.

En la introducción se comentó que XML permite definir para nuestros documentos ciertas propiedades, que estudiaremos, y que los convierten en especiales, diferenciándolos de otros, se dice entonces que aparece un tipo de documento XML. Es decir, todas esas restricciones y características, que el autor de un documento puede definir, se engloban bajo un nombre que las representa, el nombre del tipo.

Para que sirve un tipo

Como sabemos gracias a la declaración de tipo es posible restringir el contenido de los elementos y los atributos, para, principalmente realizar un filtrado de los documentos que el procesador XML debe admitir, esto asegura que cumplen una serie de requisitos o cualidades y, en definitiva, certifica la calidad de los documentos XML. Esto conlleva resultados más fiables y mayor sencillez a la hora de programar las aplicaciones que procesarán los

documentos, ya que se trabaja a partir de una base de seguridad y no será necesario contemplar ciertas situaciones de error.

Una declaración de tipo para un documento también es útil para detectar y corregir errores en la fase de elaboración del documento, posiblemente cuando aún se está a tiempo de solucionar la falta con poco esfuerzo. Esta utilidad tiene su sentido sobre todo en entornos manuales en los que es fácil que el operador se confunda y, por ejemplo, olvide rellenar un campo. ¿Qué haríamos con un pedido en el que no se ha especificado el destinatario?

Ya conocemos de capítulos anteriores lo que son las entidades, hasta el momento se han utilizado únicamente para escapar caracteres. Estas entidades tienen una funcionalidad limitada ya que su contenido viene dado y no se permite alterarlo. Sin embargo, es posible a través de una declaración de tipo del documento definir nuestras propias entidades, para por ejemplo, ahorrar escritura, facilitar la actualización de documentos (al poder modificar en un único sitio múltiples ocurrencias de un texto, o para incluir imágenes, sonidos, etc., como parte del documento).

Definir valores por defecto para los atributos puede llegar a ser bastante útil, por ejemplo, para ahorrar algo de tiempo en la creación de los documentos o para evitar que un atributo se quede sin asignar un valor.

Se puede comprobar hasta qué punto puede ser importante una declaración de tipo en los ejemplos siguientes. Imaginemos que trabajamos en una oficina que recibe pedidos y nuestra labor consiste en enviar los pedidos que nos llegan, formateados en XML, a los repartidores encargados de entregar los pedidos.

¿Qué haríamos con unos documentos como los siguientes sin declaración de tipo?

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<Pedidos>
  <Pedido>
          <Articulo>
            <Nombre>bicicleta de montaña</Nombre>
            <Referencia>22356A</Referencia>
            <Precio moneda="euro">180</Precio>
            <Almacen>5A<Almacen>
            <Fecha_entrega>16-5-2000</Fecha_entrega>
          </Articulo>
  </Pedido>
...
</Pedidos>
```

Efectivamente no tiene Destinatario, como consecuencia es posible que este envío se pueda perder, lo cual acarrea una serie de problemas en nuestra aplicación. Continuemos con el siguiente documento XML.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<Pedidos>
  <Pedido>
        <Destino>
              <Destinatario>Federico...</Destinatario>
              <Direccion>Calle...</Direccion>
              <Telefono>984567889</Telefono>
        </Destino>
        <Articulo>
          <Nombre>bicicleta de montaña</Nombre>
          <Precio>180</Precio>
          <Almacen>5A<Almacen>
          <Fecha_entrega>16-5-2000</Fecha_entrega>
        </Articulo>
  </Pedido>
...
</Pedidos>
```

En este caso, hay un par de detalles que faltan, se ha eliminado el elemento Referencia que identificaba al artículo, puede que no sea imprescindible, pero en cualquier caso representa un trabajo extra, habrá que buscar en el catálogo un artículo con semejante nombre o contactar con el destinatario y pedirle que especifique el producto que desea. También se ha omitido el atributo moneda en el elemento Precio que indicaba la moneda en la que se expresaba el valor numérico, ¿en qué moneda se le cobra?, habrá que consultar de nuevo el catálogo, con la consiguiente pérdida de tiempo.

Esperemos que no existan muchos documentos como el anterior, aunque lo ideal es no tener que pensar en la suerte y especificar ciertas restricciones en el documento XML: elementos obligatorios, orden de los mismos, atributos también obligatorios, así como los valores que admiten, etc.

2.- LA DECLARACIÓN DEL TIPO DEL DOCUMENTO.

Como es lógico, es necesario expresar con una notación que XML entienda cuál es el tipo de un documento y qué es lo que implica un tipo. Esta formalización se realiza a través de lo que se conoce como "la declaración del tipo del documento".

Si recordamos, en el capítulo **XML Básico** se comentaba que un documento XML puede constar de varias partes, **un prólogo** y **un ejemplar**. Decíamos entonces que el prólogo a su vez puede componerse de otras dos partes: una **"declaración XML"**, ya estudiada, y una **"declaración de tipo"**.

Conceptualmente la declaración de tipo consta de dos partes que serán objeto de estudio en los próximos apartados:

- La declaración de tipo del documento propiamente dicha.
- La definición del tipo del documento.

A modo de resumen, recalcar que la declaración del tipo de un documento es una parte del prólogo, y que es la parte del documento XML que nos ofrece la posibilidad adicional de marcar restricciones sobre el documento. Sobre ella versará el contenido de este capítulo.

La Declaración del Tipo del Documento

Todas las declaraciones de tipo comienzan con lo que es la declaración propiamente dicha, es decir, el texto concreto que especifica el nombre del tipo. Para ello se emplea la cadena "<!DOCTYPE" seguida del **nombre del tipo**. El siguiente ejemplo muestra una declaración de tipo, y aunque está incompleta, podemos ver su estructura:

<**!DOCTYPE** NombreXML ... >

Un tipo de documento no tendría sentido si no se pudiera asociar con sus documentos XML, este enlace se realiza en cada documento a través del elemento raíz, que debe tener el mismo nombre que el tipo, como muestran las siguientes líneas:

<!DOCTYPE **NombreXML** ... >
<**NombreXML**>
    ...
</**NombreXML**>

Esta declaración, se sitúa entre la declaración XML (en el caso de que esté presente) y la etiqueta de inicio del primer elemento (el elemento documento o raíz).

Veamos esta estructura en el siguiente texto:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE Casas_Rurales ... >
<Casas_Rurales>
        ...
</Casas_Rurales>
```

Como se puede observar a este tipo de elemento le falta algo, está incompleto, lo único que se ha hecho hasta ahora es declarar el documento como de tipo **"Casas_Rurales"**, esto es, se ha asociado un tipo al documento actual. Pero, ¿no le falta algo?, ¿qué tienen de especial los documentos de un tipo?, ¿qué implicaciones tiene pertenecer a este tipo?.

La Definición del Tipo del Documento

Toda declaración de tipo propiamente dicha para un documento se complementa con una **definición del tipo** (abreviada como **DTD**) que asocia al tipo las cualidades que posee. La DTD define los tipos de elementos, atributos, entidades y notaciones que se podrán utilizar en el documento, así como ciertas restricciones estructurales y de contendido, valores por defecto, etc.

Para formalizar todo ello XML provee ciertas estructuras especiales, las llamadas **declaraciones de marcado** y que pertenecen a alguno de los siguientes tipos:

- Declaraciones de tipos de elementos.
- Declaraciones de listas de atributos para los tipos de elementos.
- Declaraciones de entidades.
- Declaraciones de notación.

El estudio de cada tipo de declaración se realiza en alguno de los siguientes capítulos, de momento y para ir tomando contacto con la DTD (la definición del tipo del documento), al documento anterior añadiremos algunas **"declaraciones de tipos de elementos"** que forman la "definición del tipo" y que dan sentido a la "declaración del tipo".

No se preocupe por entender las declaraciones, únicamente comprenda la idea de fondo que quiere mostrar el ejemplo, y que se refiere a la capacidad de definir un tipo para un documento XML y cómo el documento se debe ajustar a esa definición (si se desea que sea de tipo válido).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE Casas_Rurales [
        <!ELEMENT Casas_Rurales (Casa)*>
        <!ELEMENT Casa (Dirección, Descripción, Estado, Tamaño)>
        <!ELEMENT Dirección (#PCDATA) >
        <!ELEMENT Descripción (#PCDATA) >
        <!ELEMENT Estado (#PCDATA) >
        <!ELEMENT Tamaño (#PCDATA) >
]>
```

```
<Casas_Rurales>
        ...
</Casas_Rurales>
```

La definición del tipo anterior especifica que el tipo de documento Casas_Rurales está formado por elementos de tipo Casa. Los elementos de tipo Casa contienen a su vez elementos de tipo Dirección, Descripción, Estado y Tamaño, en este orden y sin faltar ninguno. El contenido de estos elementos está formado exclusivamente por datos carácter (#PCDATA). El lector podría, a partir del ejemplo, pensar que todas las definiciones del tipo de un documento se parecen en mayor o menor medida a la anterior, la verdad es que admiten un gran número de posibilidades.

Para acabar de completar el documento XML del ejemplo, incluiremos el contenido del ejemplar del documento. Se puede comprobar cómo el ejemplar cumple con las restricciones de tipo, estructura y contenido especificadas.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE Casas_Rurales [
        <!ELEMENT Casas_Rurales (Casa)*>
        <!ELEMENT Casa (Dirección, Descripción, Estado, Tamaño)>
        <!ELEMENT Dirección (#PCDATA) >
        <!ELEMENT Descripción (#PCDATA) >
        <!ELEMENT Estado (#PCDATA) >
        <!ELEMENT Tamaño (#PCDATA) >
]>
<Casas_Rurales>
        <Casa>
         <Dirección>Calle Las Palmas 23. La Sagra. Toledo </Dirección>
         <Descripción>Se trata de una casa del siglo XVII, ...</Descripción>
         <Estado>El estado de conservación es magnífico, ...</Estado>
         <Tamaño>La casa tiene 259 metros cuadrados, ...</Tamaño>
        </Casa>
        <Casa>
         <Dirección>Calle Or 5, Fregenal de la Sierra. Badajoz </Dirección>
         <Descripción>De arquitectura espectacular la casa ...</Descripción>
         <Estado>Recientemente restaurada su estado actual es ...</Estado>
         <Tamaño>La superficie habitable es de ...</Tamaño>
        </Casa>
</Casas_Rurales>
```

Volvemos a recordar en este punto que las *palabras clave "xml"* como **"ELEMENT"** y **"DOCTYPE"** deben escribirse tal y como se muestran en la referencia, de otro modo generarán errores fatales, de buena formación.

Las declaraciones de marcado incluidas en la DTD pueden ser internas o externas, dependiendo de si se encuentran dentro del propio documento o no, es decir dentro de la entidad documento que se procesa. A partir de esta diferencia surge lo que se denomina el subconjunto interno y el subconjunto externo, como es de suponer el subconjunto interno lo forman todas las declaraciones de marcado que se encuentran dentro del documento y el subconjunto externo lo forman todas las declaraciones que se encuentran fuera. Normalmente un documento XML se compone de una mezcla de declaraciones de marcado internas y externas.

Como es de suponer, debe existir alguna manera de expresar dónde se encuentran las declaraciones externas. A partir de este punto la explicación se desglosa en dos, una para el subconjunto interno y otra para el externo.

Subconjunto interno

Como ya sabemos, si las declaraciones están incluidas dentro del documento de partida forman el llamado subconjunto interno y están localizadas dentro de unos corchetes que siguen a la declaración del tipo del documento. Para el subconjunto interno la declaración del tipo del documento toma básicamente la forma ya vista en los ejemplos anteriores.

```
<!DOCTYPE NombreXML [
        ...
]>
```

El subconjunto interno contiene declaraciones que pertenecen únicamente a un documento, que son específicas para él y que no es posible compartir.

Subconjunto externo

Las declaraciones de marcado también pueden encontrarse fuera del documento XML, es decir, pueden no estar incluidas dentro de la entidad documento de partida. Estas declaraciones de marcado externas pueden referenciarse de varias maneras:

- Mediante una declaración explícita de subconjunto externo.
- Mediante entidades parámetro externas.

Con los conocimientos disponibles no estamos todavía capacitados para entender la segunda forma, se estudiará más adelante en el capítulo dedicado a las entidades y a las declaraciones de entidades.

Normalmente el subconjunto externo está formado por declaraciones comunes que se comparten entre múltiples documentos XML que pertenecen al mismo tipo. Se escriben y modifican una vez en lugar de tener que repetirla y modificarla muchas veces.

Ahora los corchetes pierden su sentido y a cambio se impone utilizar algún sistema de referencia que permita localizar las declaraciones de marcado externas. Para el subconjunto externo la declaración del tipo del documento puede tomar alguna de las siguientes formas:

```
<!DOCTYPE NombreXML SYSTEM "URI" >
<!DOCTYPE NombreXML PUBLIC "id_publico" "URI" >
```

En la primera, detrás de la palabra SYSTEM se especifica un URI donde se pueden encontrar las declaraciones. En la segunda forma se especifica también un identificador, que puede ser utilizado por el procesador XML para intentar generar un URI alternativo, posiblemente basado en alguna tabla. Hay que notar que también es necesario (obligatorio) especificar un URI.

Podemos transformar el ejemplo de las casas rurales para que todas las declaraciones de los elementos sean externas al documento. El URI en este ejemplo es absoluto pero podría haberse empleado uno relativo, si se desea probar el ejemplo se puede cambiar el URI a uno relativo que incluyera únicamente el nombre del fichero **casasrurales.dtd**, situando ambos ficheros en el mismo directorio.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE Casas_Rurales SYSTEM "http://www.casasrurales.com/casasrurales.dtd">
<Casas_Rurales>
  ...
</Casas_Rurales>
```

Siendo el contenido del fichero **casasrurales.dtd**:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT Casas_Rurales (Casa)*>
<!ELEMENT Casa (Dirección, Descripción, Estado, Tamaño)>
<!ELEMENT Dirección (#PCDATA) >
<!ELEMENT Descripción (#PCDATA) >
<!ELEMENT Estado (#PCDATA) >
<!ELEMENT Tamaño (#PCDATA) >
```

La primera línea del fichero es una declaración de texto, enteramente opcional y que especifica que las siguientes líneas son XML y la versión con la que cumplen. Incluye también una declaración de codificación. Las declaraciones de texto no pueden incluir una declaración de documento autónomo, no tendría sentido.

Destacar que en el contenido no puede aparecer la declaración de un nuevo DTD

La extensión "**.dtd**" del archivo es sólo una convención, no hay ninguna obligación en que así sea.

Declaración de documento autónomo

Si recordamos, en la declaración XML podíamos especificar que nuestro documento XML era independiente de otros contenidos externos para su interpretación, podíamos declarar nuestro documento como "autónomo".

Ahora, al incluir un subconjunto externo el documento ya no puede ser denominado como autónomo, ya que estas declaraciones son necesarias para realizar una interpretación correcta del documento.

Declarar un documento como autónomo ("standalone") puede acelerar el procesamiento de un documento, ya que su procesado comenzará antes de obtener todas las entidades externas. Un documento que no es autónomo tiene que esperar a que el procesador XML obtenga todas las entidades para comenzar a ser procesado.

Orden de evaluación de las DTD's.

Normalmente la DTD se compone, como muestra el ejemplo, de una mezcla de definiciones internas y externas:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE Casas_Rurales SYSTEM "dtd/casasrurales2.dtd" [
        <!ELEMENT Casas_Rurales (Casa)*>
        <!ELEMENT Casa (Dirección, Estado, Tipo)>
        <!ELEMENT Dirección (#PCDATA) >
        <!ELEMENT Estado (#PCDATA) >
        <!ELEMENT Tipo (#PCDATA) >
]>
<Casas_Rurales>...</Casas_Rurales>
```

De momento, mientras no se conozcan las entidades parámetro externas, podemos pensar que el orden de procesamiento es el siguiente: se procesa primeramente el subconjunto interno (junto con el subconjunto externo implícito) y posteriormente el subconjunto externo explícito (SYSTEM).

Este orden permite sobrescribir declaraciones externas, compartidas entre varios documentos y ajustar la DTD a un documento específico. En los siguientes capítulos en los que estudiaremos más detalladamente los distintos tipos de declaraciones se incluyen ejemplos concretos que muestran la forma de proceder cuando existen declaraciones internas y externas.

Documentos de Tipo Válido.

Los documentos que tienen declarado un tipo y además cumplen con él se denominan **válidos** o de **tipo válido**, los que tienen declarado un tipo pero no cumplen con él se denominan **no válidos** o de **tipo no válido**. Es preferible emplear las segundas denominaciones ya que las primeras (válido o no válido) parecen indicar que el documento "no sirve". Hasta el momento se ha trabajado con documentos que no podían denominarse válidos, ya que carecían de declaración de tipo, pero que nos eran muy útiles. Las expresiones de tipo válido o de tipo no válido dejan más claro este matiz.

Aquellos documentos XML que no incluyen una DTD no pueden clasificarse ni como de tipo válido ni como de tipo no válido, ya que no se puede decir que cumplan o que no cumplan con alguna DTD. Hay que destacar que un documento sea de tipo "no válido" no implica que esté mal formado, de hecho un documento XML para que sea considerado como válido debe estar primeramente bien formado.

No todos los procesadores XML comprueban si un documento es válido o no, sin embargo, todos comprueban que el documento esté bien formado, ya que de otra manera no podrían realizar un procesamiento fiable de los documentos XML. Que un procesador XML no compruebe la validez del documento no significa que la DTD para ese documento quede inservible, existen características de la DTD como la declaración de entidades y valores por defecto de los atributos de las que todavía se puede sacar provecho. A este respecto se añadirá algo más de información en capítulos posteriores, cuando se conozcan las declaraciones de marcado.

3.- DECLARACIONES DE TIPOS DE ELEMENTOS.

Esta pregunta es un complemento de la pregunta anterior donde se explicaba que era la declaración y la definición del tipo de un documento y donde se apuntó brevemente que la definición del tipo de un documento podía incluir los siguientes tipos de declaraciones de marcado:

1. Declaraciones de tipos de elementos.
2. Declaraciones de listas de atributos para los tipos de elementos.
3. Declaraciones de entidades.
4. Declaraciones de notación.

El estudio de estos tipos de declaraciones se dejó para capítulos posteriores. Pues bien, el contenido del presente está orientado para que al terminar el lector conozca y sea capaz de incluir declaraciones de tipos de elementos en la definición del tipo de un documento.

Por lo dicho anteriormente, suponemos que ya se conoce lo que es la declaración del tipo de un documento y, para ilustrar la teoría de este capítulo, partimos de un documento con una declaración de tipo como la siguiente (ahora incompleta):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE listado_discos  ...
<listado_discos>...</listado_discos>
```

Este fragmento de documento XML se irá completando a lo largo del capítulo e irá creciendo con los conocimientos adquiridos.

¿QUÉ SON LAS DECLARACIONES DE TIPOS DE ELEMENTOS?

Las "**declaraciones de tipos de elementos**", como dice la propia expresión, permiten definir tipos de elementos. A través de ellas podremos comunicar a un procesador XML, encargado de validar el documento, qué elementos están permitidos y cuál es su contenido. Con otras palabras, en lo que respecta a los elementos, para que un documento XML sea de tipo válido se debe cumplir que:

- Todos los elementos estén reconocidos mediante "declaraciones de tipos", o lo que es lo mismo, todos los elementos deben pertenecen a un tipo declarado.
- El contenido de cada elemento se ajusta a lo declarado en su tipo.

Así, las **Declaraciones de Tipos de Elementos** permitirán la detección de ciertos errores: inclusión de elementos no declarados, contenidos no válidos en un elemento, elementos obligatorios olvidados, elementos repetidos, etc.

INCLUSIÓN DE DECLARACIONES DE TIPOS DE ELEMENTOS

Todas las declaraciones de tipos de elementos deben estar contenidas dentro de una definición de tipo de un documento (DTD), ya sea interna o externa. La Recomendación de XML recoge en sus reglas [45] y [46] la forma que debe tener una Declaración de Tipo de Elemento:

```
[45]  elementdecl ::=  '<!ELEMENT' S Nombre S contentspec S? '>'
[46]  contentspec ::=  'EMPTY' | 'ANY' | Mixed | children
```

Todas las Declaraciones de Tipo de Elementos se componen de una cadena [**<!ELEMENT**] seguida de uno o más espacios en blanco, un nombre XML, espacios en blanco, la especificación del contenido del elemento, espacios en blanco opcionales y el carácter de cierre [**>**].

En el párrafo anterior lo único que queda por determinar es cómo se realiza la especificación del contenido de un elemento. Sobre los espacios en blanco hay que notar que no existe ninguna separación entre **"<!"** y **"ELEMENT"**, lo contrario implica un error de formación.

Sobre las declaraciones de tipos de elementos hay que comentar también que los tipos de elementos se declaran de uno en uno, es decir, **las Declaraciones de Tipos de Elementos son individuales**. No es posible tampoco declarar un tipo de elemento dos veces, la Recomendación de XML considera este hecho como un error de buena formación.

Así, en el ejemplo siguiente, la única declaración que sería válida, suponiendo que estuviera completa, sería la primera. En la segunda existen dos tipos de elementos declarados conjuntamente y en la tercera se declara un tipo de elemento ya existente:

```
<!DOCTYPE listado_discos [
        ...
        <!ELEMENT cinta ...>
        ...
        <!ELEMENT canción disco ...>
        ...         ...
        <!ELEMENT cinta ...>
        ...
]>
```

Las declaraciones de elementos anteriores están sin acabar, falta determinar cuál es el contenido válido para el tipo que se declara. La regla de producción [**46**] se refiere a este aspecto y concreta cuatro modelos posibles para describir el contenido de los elementos que cumplen con un tipo.

```
[46] contentspec ::= 'EMPTY' | 'ANY' | Mixed | children
```

Es decir, un elemento puede no tener contenido, ser vacío (**EMPTY**), tener cualquier contenido (**ANY**), una mezcla de valores (**Mixed**) o un conjunto de elementos hijos (**chidren**).

El tipo ANY

Este tipo no impone ninguna restricción. Una declaración como la siguiente crea un nuevo tipo de elemento *"nombre_de_elemento"* cuyos elementos podrán contener cualquier mezcla de datos carácter y elementos (declarados) sin restricciones:

```
<!ELEMENT nombre_de_elemento ANY>
```

Continuando con el ejemplo anterior, podemos declarar el elemento raíz como de tipo **ANY** (el elemento raíz es un elemento como cualquier otro, también hay que declararlo), **listado_discos** podrá contener cualquier combinación de elementos de cualquier tipo y datos carácter.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE listado_discos [
      <!ELEMENT listado_discos ANY>
]>
<listado_discos>El disco ...<listado_discos/>
```

El tipo EMPTY

Para especificar que un elemento no puede tener contenido se le declara como **EMPTY**. El formato general de una declaración de este tipo es el siguiente:

```
<!ELEMENT nombre_de_elemento EMPTY>
```

Continuando con el ejemplo de partida incluiremos una declaración de elemento vacío.

En el ejemplo anterior se declara el documento actual como de tipo **listado_discos**, y se define el elemento raíz como de tipo vacío.

Como el lector podrá observar el documento anterior, tal y como está ahora, no tiene mucho sentido, generalmente no será de mucha utilidad un elemento raíz de tipo vacío. De cualquier manera, normalmente un elemento sin contenido tendrá atributos que aporten algún tipo de información y que den sentido al elemento. Una comparación con HTML, nos haría pensar en el elemento **<IMG>** como un elemento vacío en el que los atributos contienen valores que hacen útil al elemento.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE listado_discos [
        <!ELEMENT listado_discos ANY>
        <!ELEMENT disco EMPTY>
]>
<listado_discos>
        <disco/>
        <disco/>
        ...
</listado_discos>
```

Para que el ejemplo tenga un poco más de sentido se han añadido atributos, al elemento de tipo **disco**, que lo complementan. Aunque se explicará en el tema siguiente, hay que decir que para el documento siguiente fuera de tipo válido, habría que declarar también los atributos que contiene el tipo de elemento **disco** (si estaría bien formado).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE listado_discos [
        <!ELEMENT listado_discos ANY>
        <!ELEMENT disco EMPTY>
]>
<listado_discos>
        <disco título="Maravillas" autor="Amaya Diosdado Sois" año="1974" />
        <disco título="Sin razón" autor="Federico Martín Martín" año="1988" />
        <disco título="Esperándote" autor="Andrea Sonjuan" año="2000" />
        ...
</listado_discos>
```

El tipo MIXED

Ha pasado algún tiempo y necesitamos ampliar la información contenida en nuestro inventario de discos, necesitamos incluir texto que no pueda contener elementos. Es posible, y necesario, incluir declaraciones de tipos de elementos que especifiquen que los elementos de dicho tipo contendrán **únicamente datos carácter**. Las formas siguientes son completamente equivalentes:

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

Los elementos declarados como de tipo **PCDATA no pueden contener** los caracteres siguientes:

El caracter [**<**], porque se interpretará como el comienzo de un nuevo elemento y el tipo **PCDATA** no puede contener etiquetas, ni elementos.
El caracter [**&**], porque se interpretará siempre como el comienzo de una entidad.
La cadena [**]]>**], porque marca el final de una sección **CDATA**, estudiada más adelante en el curso.

Si se desean incluir estos caracteres, para que no se interpreten erróneamente, es necesario escaparlos, como ya se explicó. A parte de estas restricciones obligatorias no existe forma de limitar el valor de un elemento **PCDATA** a un determinado grupo de caracteres.

De esta manera, se podría declarar un tipo de elemento **disco** que recogiera una descripción textual para cada disco.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE listado_discos [
        <!ELEMENT listado_discos ANY>
        <!ELEMENT disco (#PCDATA)>
]>
<listado_discos>
  <disco>
   El título de este disco es "Maravillas", su autor es Amaya Pan y se compone
   de 16 canciones, todas con una melodía bastante pegadiza. La primera canción
   Ave María pretende ser un canto a la vida y está llena de notas y sonidos que
   evocan una tarde de verano. La segunda canción, Alegre Primavera, recibió muy
   buenas críticas,...
  </disco>
  <disco>Disco titulado "América", cuyo autor es... </disco>
  <disco>... </disco>
...
</listado_discos>
```

En el ejemplo anterior se declara el documento actual como de tipo **listado_discos**, cuya definición podría decir lo siguiente: un documento de éste tipo contiene únicamente elementos de tipo **disco** y datos carácter en cualquier combinación (tipo **ANY**), a su vez cada elemento de tipo **disco** se compone de datos carácter (**#PCDATA**), no admitiendo otros elementos en su interior (no se podría, por ejemplo, anidar elementos de tipo **disco**).

Parte de la riqueza de XML está formada por esa mezcla entre datos carácter y elementos, y, como era de esperar, veremos que existen declaraciones que especifican que el contenido de un elemento puede estar compuesto por datos carácter y elementos. Por ejemplo:

```
<!ELEMENT nombre_de_elemento (#PCDATA | Elemento1 | Elemento2 | .... )*>
```

La aparición de los distintos elementos o de los datos carácter en un elemento concreto no es obligatoria y podrían no estar presentes. No se pueden establecer restricciones adicionales sobre el orden o sobre el número. Este modelo de contenido simplemente define una mezcla entre datos carácter y los elementos que se declararan. Por este motivo es un error especificar dos veces el mismo elemento en la declaración.

El asterisco que sigue al paréntesis es necesario. Igualmente, sería un error incluir algún elemento de repetición [**+**] o [**?**] distinto del necesario asterisco final [*****].

En este punto es posible redefinir el elemento **disco** del ejemplo anterior para que pueda contener ciertos elementos que nos permiten acceder a las cadenas que lo componen (todos son de tipo **"#PCDATA"**), por ejemplo, buscar un autor o un título en concreto.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE listado_discos [
        <!ELEMENT listado_discos ANY>
        <!ELEMENT disco (#PCDATA | autor | título | canción)*>
        <!ELEMENT autor (#PCDATA)>
        <!ELEMENT título (#PCDATA)>
        <!ELEMENT canción (#PCDATA)>
]>
<listado_discos>
        Listado de discos de Federico Garcés.
        <disco>Este disco titulado <título>Maravillas</título> y cuyo autor
        es <autor>Amaya Diosdado</autor> se compone de varias canciones,
        todas con una melodía bastante pegadiza.
        La primera canción <canción>Ave María</canción> pretende ser un canto
        a la vida y está llena de notas y sonidos que evocan una tarde de verano.
        La tercera canción <canción>Alegre Primavera</canción> recibió muy
        buenas críticas, ...</disco>
        <disco><título>América</título></disco>
</listado_discos>
```

La definición del tipo anterior especifica que el tipo **listado_discos**, puede contener únicamente elementos de tipo **disco** entremezclados con datos carácter. Los elementos de tipo disco a su vez pueden reunir datos carácter mezclados con elementos de tipo **autor**, **título**, **canción**, en cualquier combinación, pudiendo contener éstos últimos únicamente datos carácter.

Podemos fijarnos un instante en el último elemento de tipo **disco**, contiene únicamente un elemento de tipo **título**. La declaración para el elemento **disco** no obliga a que aparezcan elementos o datos carácter e incluso podría estar vacío.

El tipo *CHILDREN*

Para finalizar veremos también que existen tipos de elementos que solamente pueden contener otros elementos y que no están autorizados para incluir datos carácter. Este tipo se utiliza fundamentalmente para agrupar otros elementos y formalizar estructuras. En la declaración del tipo se incluye un patrón que deben seguir los elementos de este tipo y que puede tomar varias formas que se explican a continuación.

Secuencia de elementos

Para especificar que el modelo se compone de una secuencia de elementos, después del nombre del tipo de elemento que se declara se añade una lista de los elementos que puede contener separados por comas. Como ilustra el siguiente ejemplo:

```
<!ELEMENT nombre_de_elemento (elemento1, ... , elementoX)>
```

Alternativa de elementos

Si lo que se desea es especificar una alternativa de elementos, en lugar de comas se utiliza como separador la barra vertical [|]. El ejemplo siguiente ejemplo se especifica que únicamente un elemento de la lista puede formar parte del contenido en cada realización del tipo de elemento

```
<!ELEMENT nombre_de_elemento (elemento1 | ... | elementoX)>
```

Combinación de modelos

Se pueden utilizar paréntesis para agrupar secuencias y alternativas de modelos. Como muestra el ejemplo:

```
<!ELEMENT nombre_de_elemento (elemento1 | ( elemento2, elemento3))>
```

Especificando una frecuencia

Es posible además especificar una frecuencia de repetición adjuntando a un elemento o a un paréntesis de cierre uno de los siguientes caracteres:

El caracter [**?**] indica opción, el elemento o grupo se puede repetir **cero o una** vez
El caracter [**+**] indica **una o más** repeticiones. Asegura una ocurrencia como mínimo.
El caracter [*****] indica **cero o más** repeticiones.

No es posible especificar una frecuencia determinada, 3, 5, 6 veces. Si no se especifica ninguno de estos caracteres el significado es de uno y como máximo uno.

Para finalizar este apartado se acompañará la teoría con un ejemplo. El listado de discos anterior carecía prácticamente de estructura, era una mezcla indiscriminada de elementos y datos carácter. Queremos seguir añadiendo más información, para ello se impone reorganizar la estructura de alguna manera que permita absorber los nuevos contenidos. En concreto se quiere añadir para cada disco un apartado con datos específicos sobre el disco, una descripción textual y comentarios.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE listado_discos [
        <!ELEMENT listado_discos (disco)*>
        <!ELEMENT disco (datos, descripción, comentarios? )>
        <!-- Declaración del tipo datos y todos los tipos que contiene -->
        <!ELEMENT datos ((solista | grupo), título, año?)>
        <!ELEMENT solista (#PCDATA)>
        <!ELEMENT grupo (#PCDATA)>
        <!ELEMENT título (#PCDATA)>
        <!ELEMENT año (#PCDATA)>
        <!-- Declaración del tipo descripción y todos los tipos que
contiene -->
        <!ELEMENT descripción (canción)+>
        <!ELEMENT canción (#PCDATA)>
        <!-- Declaración del tipo de elemento comentarios -->
        <!ELEMENT comentarios (#PCDATA)>
]>
<listado_discos>
  <disco>
    <datos>
      <grupo>IO</grupo>
      <título>Sensación</título>
    </datos>
    <descripción>
      <canción>Luna de Papel</canción>
      <canción>Versos</canción>
      <canción>Naufragio</canción>
    </descripción>
    <comentarios>Este disco es de los que más me gustan.</comentarios>
```

```
  </disco>
  <disco>
   <datos>
     <solista>Raúl</solista>
     <título>Tu Barco</título>
     <año>1974</año>
    </datos>
    <descripción>
     <canción>Marina</canción>
    </descripción>
  </disco>
</listado_discos>
```

Este ejemplo es más complejo que los anteriores, a continuación lo explicaremos para que sea comprensible: Los documentos XML de tipo **listado_discos** se componen de elementos de tipo **disco** (cero o más). Cada elemento de tipo **disco** estan formados a su vez por un elemento de tipo **datos**, seguido de un elemento de tipo **descripción**, ambos obligatorios, seguidos de un elemento de tipo **comentarios** opcional (se puede comprobar que el último disco no tiene ningún comentario).

Los elementos de tipo **datos** se componen de tres elementos ordenados, un elemento de tipo **grupo** o de tipo **solista** (uno de los dos, pero no los dos o ninguno de ellos), un elemento de tipo **título** y un elemento de tipo **año** opcional (puede ocurrir una o cero veces).

Los elementos de tipo **descripción** pueden contener elementos de tipo **canción** (uno como mínimo). El resto de los elementos pueden contener únicamente caracteres según se regula en su declaración de tipo.

4.- DECLARACIONES DE LISTAS DE ATRIBUTOS.

En la pregunta anterior se estudió cómo incluir en un documento declaraciones de tipos de elementos; continuaremos estudiando declaraciones de marcado, aprenderemos a definir atributos asociados a los elementos mediante **Declaraciones de Listas de Atributos**.

Partiremos de un documento como el siguiente con algunas declaraciones de elementos y que posiblemente cumple sus funciones en una tienda de animales especializada en perros:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE tienda_animales [
        <!ELEMENT tienda_animales (perro)*>
        <!ELEMENT perro (#PCDATA | nombre | raza)*>
        <!ELEMENT nombre (#PCDATA)>
        <!ELEMENT raza (#PCDATA)>
]>
<tienda_animales>
        <perro>...</perro>
        <perro>Este es un estupendo ejemplar de perro
        <raza>labrador</raza>, en la tienda le conocemos
        como <nombre>Chispa</nombre> por la viveza de sus ojos.
        Su piel es de color canela...</perro>
        <perro>...</perro>
</tienda_animales>
```

¿QUÉ SON LAS DECLARACIONES DE LISTAS DE ATRIBUTOS?

Las Declaraciones de Listas de Atributos, definen:

- ✓ El conjunto de atributos que pertenecen a cada tipo de elemento (cada atributo está ligado siempre a un elemento).
- ✓ Establecen restricciones en el contenido de estos atributos, a través de tipos predefinidos a los que se debe ajustarse el atributo.
- ✓ Aportan información sobre la naturaleza del atributo (obligatorio, voluntario, de valor fijo) así como la posibilidad de especificar un valor por defecto para el mismo.

Ya sabemos que los elementos pueden contener atributos, pues bien, en un documento XML de tipo válido, todos los atributos deben declararse mediante **Declaraciones de Listas de Atributos**, incluidas en la DTD del documento, además, el valor de cada atributo debe estar acorde con lo declarado.

Sin las DTDs se están asignando valores arbitrariamente a los atributos, no existe ningún control a este respecto. Ni valores por defecto, ni

restricciones de ningún tipo, si el documento es pequeño esto puede no ser ningún problema, pero con documentos grandes, las ventajas que ofrecen las DTDs se vuelven muy valiosas.

Sin una DTD tampoco se puede añadir objetos binarios al documento ya que, éstos se añaden a través de atributos.

INCLUSIÓN DE DECLARACIONES DE LISTAS DE ATRIBUTOS

La forma que debe tener una declaración de lista de atributos queda recogida en las reglas número [**52**] y [**53**] de la Recomendación de XML. **Estas declaraciones se incluyen siempre dentro de una DTD**.

```
[52]  AttlistDecl ::=  '<!ATTLIST' S Nombre AttDef* S? '>'
[53]  AttDef ::=  S Nombre S AttType S DefaultDecl
```

Todas las declaraciones de listas de atributos comienzan con la cadena [**<!ATTLIST**], a continuación, y separados por al menos un espacio, el nombre del elemento al que pertenecen los atributos. Seguidamente el nombre del atributo que se está declarando, espacios, el tipo del atributo y su declaración por defecto. Es posible declarar varios atributos para un elemento, las declaraciones de atributos pueden tomar dos formas. La de una declaración de un único atributo en la lista o la de una declaración de varios atributos conjuntamente:

```
<!ATTLIST nombre_de_elemento nombre_de_atributo tipo declaración_por_defecto>

<!ATTLIST nombre_de_elemento nombre_de_atributo tipo declaración_por_defecto
                                    ...
                    nombre_de_atributo tipo declaración_por_defecto
                                    ...
>
```

Las declaraciones simples y múltiples se pueden combinar como se quiera, en otras palabras, algunos atributos se pueden declarar mediante declaraciones de listas simples y otros mediante listas múltiples a gusto del diseñador de la DTD, el efecto para XML es el mismo que si se hubiesen declarado en una única declaración todos los atributos de un mismo elemento. A este respecto, se comentarán algunos detalles adicionales en un apartado de este capítulo.

Un ejemplo concreto podría ser el siguiente, creado a partir del ejemplo de partida, que contendrán como atributos la fecha de nacimiento y la fecha de venta de un perro. De momento, olvídese de los detalles de la declaración de los atributos y fíjese como siguen la forma general declarada arriba.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE tienda_animales [
        <!ELEMENT tienda_animales (perro)*>
        <!ELEMENT perro (#PCDATA)>
        <!ATTLIST perro fecha_nacimiento CDATA "">
        <!ATTLIST perro fecha_venta CDATA "">
]>
<tienda_animales>
  <perro>...</perro>
  <perro fecha_nacimiento="14/7/1999">
     Este es un estupendo ejemplar de perro labrador, en la tienda le
conocemos como
     Chispa por la viveza de sus ojos. Su piel es de color canela...
  </perro>
  <perro>...</perro>
</tienda_animales>
```

XML no tiene demasiadas reglas de sintaxis, los atributos son sencillos de declarar y de utilizar, no obstante es necesario observar algunos pequeños detalles:

1. Todas las declaraciones de listas de atributos comienzan con la cadena **"<!ATTLIST"**, tal y como está aquí escrita, hay que notar que entre **"<!"** y **"ATTLIST"** no hay ninguna separación.
2. El orden de las declaraciones es como se especifica. Cualquier alteración en este sentido será causa de errores. Es decir, el elemento (*nombre_de_elemento*) al que se aplica la definición siempre se especifica antes que los atributos definidos, el tipo de cada atributo (*tipo*) se concreta después del atributo (*nombre_de_atributo*) y a continuación la declaración (*declaración_por_defecto*).
3. Los valores de los atributos por defecto deben delimitarse por comillas dobles o simples.
4. Hay que seguir la misma consistencia con las mayúsculas y las minúsculas que llevamos hasta ahora. Los nombres de los atributos empleados en la declaración deben ser los mismos que luego reciban los valores en los elementos.
5. El número de atributos que se declaran para un elemento puede ser tan grande como se desee. XML no impone ninguna restricción en este sentido.
6. El orden en que se especifican los atributos para un elemento en concreto es irrelevante y no viene impuesto por el orden definido en la declaración ni por ningún otro factor.

Nombre del elemento

El **nombre_de_elemento** que aparece en la especificación se refiere al nombre del tipo de elemento al que se asociará el atributo. Como ya sabemos, tiene que cumplir las restricciones impuestas a los nombres XML. En la siguiente declaración de lista de atributos, **perro** es el nombre del tipo de elemento al que se asocian los atributos **fecha_nacimiento** y **fecha_venta**.

```
...
<!ATTLIST perro fecha_nacimiento CDATA "">
<!ATTLIST perro fecha_venta CDATA "">
<!ATTLIST perro fecha_nacimiento CDATA ""
              sexo (macho | hembra) #REQUIRED
              ID ID #REQUIRED
>
...
```

Las declaraciones de listas de atributos para un tipo de elemento pueden especificarse en cualquier lugar dentro de la DTD, pero si los atributos se declaran antes que su tipo de elemento, un analizador que valide podría mostrar un aviso al usuario y continuar el proceso, aunque no es un error. También puede ocurrir que el analizador lea toda la DTD y compruebe que no falta ninguna declaración, en cuyo caso no mostraría ningún mensaje.

Nombre del atributo

El **nombre_de_atributo** que aparece en la especificación se refiere al nombre del atributo que se va a definir y, al igual que ocurría con los nombres de los elementos, también debe ser un nombre XML y cumplir las restricciones asociadas. En el ejemplo siguiente, **fecha_nacimiento** y **sexo** son atributos de perro.

```
...
        <!ATTLIST perro fecha_nacimiento CDATA "" >
        <!ATTLIST perro sexo CDATA #REQUIRED>
        <!ATTLIST perro fecha_nacimiento CDATA #REQUIRED
                        sexo (macho | hembra) #REQUIRED
                        ID ID #REQUIRED
        >
...
```

Tipos de atributos.

El tipo en cada declaración de atributos especifica cómo puede ser el contenido del atributo limitando los valores que puede tomar. Los tipos están predefinidos y son los siguientes: tipos cadena (CDATA), tipos enumerados y tipos específicos (ENTITY, ENTITIES , ID, IDREF, IDREFS, NMTOKEN y NMTOKENS)

En la Recomendación las reglas de producción que definen los tipos de atributo son las [54], [55], [56] y [57].

Tipos cadena.

Sólo existe un tipo dentro de esta categoría, el tipo **CDATA**, que significa "*Character DATA*", es decir, datos carácter. Es el tipo más general. Cualquier cadena de texto (que no incluya los caracteres abajo indicados), es un valor válido para un atributo CDATA. Este tipo de atributo es más permisivo de todos los tipos en cuanto a los caracteres que admite; si se desea almacenar una cadena de texto "cualquiera", éste es el tipo adecuado.

Los caracteres que **no se pueden incluir** (para utilizarlos habrá que escaparlos) dentro del valor de un atributo CDATA son los siguientes:

1. El caracter [**<**], porque se interpretará como el comienzo de un nuevo elemento, el tipo CDATA no puede contener etiquetas, ni elementos.
2. El caracter [**&**], salvo que se utilice en una referencia a una entidad.
3. El caracter de comilla [**'**] o [**"**] que se utilice para delimitar el valor del atributo, ya que se confundiría con el final del atributo.

Para declarar un atributo de tipo cadena, lo único que hay que hacer es añadir la palabra CDATA a continuación de su nombre en la declaración de una lista de atributos.

```
<!ATTLIST nom_elemento nom_atributo CDATA valor_por_defecto>
<!ATTLIST perro fecha_nacimiento CDATA "">
```

En el ejemplo precedente se ha utilizado el tipo CDATA para almacenar la fecha de nacimiento del animal ya que la fecha contiene un carácter [****] para separar los días de los meses y los años de los meses, carácter prohibido en el resto de los tipos.

Tipos enumerados.

Recordamos que dentro de esta categoría existen dos tipos, los explicaremos:

1. **Tipo enumerado**: los atributos **ENUMERATION** ofrecen una lista de posibles valores que puede tomar el atributo, todos estos valores son nombres de tipo *token*.
2. **Tipo notación**: este atributo indica que el contenido del elemento debe ser procesado mediante la herramienta que indica la notación. Los atributos **NOTATION** contienen una notación declarada en la DTD. Sobre las notaciones y su declaración se dedica un capítulo más adelante, decir de momento, que las notaciones identifican aplicaciones que realizarán algún proceso.

Existen ciertos detalles a tener en cuenta:

Los nombres de tipo *token* tienen ciertas limitaciones en cuanto a los caracteres que pueden formar parte de ellos. Entre estos caracteres no permitidos destacan los siguientes: espacios en blanco, [**,**], [**!**], [**;**], [**/**] o [****]. Para una explicación más detallada sobre los caracteres que admite un nombre token se puede consultar la Recomendación de XML, su regla de producción es la número [7].

Si se desea declarar un atributo de tipo **ENUMERATION** hay que añadir, en la declaración de lista de atributos, a continuación del nombre del atributo una lista con los posibles valores que puede tomar el atributo. Los valores se presentan separados entre sí por un carácter [**|**] y encerrados entre paréntesis:

```
<!ATTLIST nom_elemento atributo (valor1 | valor2 |...) valor_por_defecto>
```

Para definir un atributo como de tipo **NOTATION**, hay que añadir la palabra "**NOTATION**" seguida de una lista de notaciones delimitada por paréntesis (notaciones declaradas previamente). Los nombres de las notaciones que puede tomar el atributo se deben separar por caracteres de barra vertical [**|**].

```
<!ATTLIST nom_elemento nom_atributo NOTATION (notación1 | notación2 |...)
valor_por_defecto >
```

Ejemplos concretos de tipos enumerados podrían ser los siguientes, el elemento **foto** podría contener una foto en formato *gif* del animal codificada en *base64*.

```
...
<!ATTLIST perro sexo (macho | hembra) #REQUIRED>
<!ATTLIST foto formato_foto NOTATION (gif | jpg) "gif">
<!ATTLIST foto codificación_foto NOTATION (base64) "base64">
...
<foto formato_foto="gif" codificación_foto="base64">
   MTAAEBAQIQEBAQMTAABQ7GBoaMUF ...
</foto>
```

El **sexo** del animal no puede tomar más que dos valores, al declarar un tipo enumerado, nos aseguramos que el valor que se le da al atributo está dentro del rango.

El atributo **formato_foto** y **codificación_foto** se utilizan para incluir las fotos de los animales como parte del documento XML (dentro del propio documento), como se muestra en el ejemplo.

Existen ciertos detalles a tener en cuenta:

Los valores posibles de un tipo numerado no se especifican entre comillas. Sin embargo, si se desea especificar un valor por defecto, éste debe ir entre comillas.
Los caracteres de barras verticales y los paréntesis de los tipos enumerados (ENUMERATION y NOTATION) admiten a ambos de sus lados los espacios en blanco que se desee, incluyendo ninguno. El significado de "espacios en blanco" es el de siempre en XML.
La regla de producción de los tipos enumerados es la número [57]. La de los tipos notación la [58] y la de los tipos Enumeration la [59].

Atributos tipo *token*

Los atributos de tipo *token* se utilizan para definir nombres, se parecen a los atributos declarados como CDATA, con la diferencia que estos tipos son más restrictivos en cuantos a los caracteres que permiten.

A diferencia de los atributos de tipo CDATA que puede contener cualquier caracter si éste se atiene a las reglas de formación, los atributos de tipo NMTOKEN sólo puede contener letras, dígitos, punto [.], guión [-], subrayado [_] y dos puntos [:]. Los del tipo NMTOKENS pueden contener los mismos caracteres que NMTOKEN más espacios en blanco. Un espacio en blanco consiste en uno o más espacios, retornos de carro o tabuladores.

ID: se utilizan para dar un nombre a los elementos, identificándolos de manera única. Los atributos ID deben cumplir las siguientes restricciones:

Los valores que pueden tomar son nombres XML. Conocer esta característica es importante y puede ahorrarnos trabajo.

El valor del atributo ID deberá ser único dentro del documento XML, es decir dos atributos ID no pueden tener el mismo valor, aún en el caso en el que los atributos tengan distintos nombres y se refieran a tipos de elementos diferentes.

Cualquier elemento puede tener un atributo de tipo ID, pero tendrá como máximo uno, una declaración de una lista de atributos con dos atributos de tipo ID para el mismo tipo de elemento implica una declaración no válida.

IDREF, IDREFS: estos tipos de atributos se utilizan para referenciar elementos identificados con un atributo ID, (el valor de éstos atributos contiene el valor de otro atributo ID). Sobre este tipo de atributo debemos saber que:

Al igual que ocurría con los atributos de tipo ID, es lógico pensar que los valores que pueden tomar son únicamente nombres XML.

Cada referencia de un tipo IDREF debe estar recogida en un atributo de tipo ID, estas referencias deben existir, es decir, los nombres que tomen como valor estos atributos deben ser referencias a valores de atributos ID, si esto no se cumple el documento no será de tipo válido.

Los atributos IDREFS son análogos a los IDREF con la diferencia de que pueden tomar varios valores, separados por espacios.

ENTITY, ENTITIES: los objetos (ficheros de imágenes, sonido, etc.) en un documento XML se gestionan a través de atributos entidad. Estas entidades (se estudiarán en el siguiente capítulo), son especiales e incluyen en su declaración el nombre del fichero que contiene los datos y una notación que identifica la aplicación que es capaz de interpretarlos. No se pueden utilizar entidades si no se declaran en una DTD o Esquema. Los nombres de entidad son nombres XML y deben cumplir las restricciones impuestas.

NMTOKEN, NMTOKENS: son análogos a los de tipo CDATA, ya conocidos, pero con la diferencia de que sus valores válidos sólo pueden ser nombres tipo *token*. Los atributos de tipo NMTOKENS son idénticos a los de tipo NMTOKEN con la diferencia de que aceptan varios valores al mismo tiempo, al igual que ocurría con los atributos de tipo IDREFS, los distintos valores se separan utilizando espacios.

La declaración de estos tipos de atributos es muy sencilla, lo único que se debe hacer es añadir la palabra, ID, IDREF, IDREFS, ENTITY, ENTITIES, NMTOKEN, NMTOKENS después del nombre del atributo.

De todos estos tipos de atributos "*token*" los más útiles son ID, IDREF, IDREFS y ENTITY. La utilidad de los atributos NMTOKEN y NMTOKENS es limitada, por su parecido con el tipo CDATA, aunque NMTOKENS tiene la ventaja de que admite una lista de valores.

La regla de producción de los tipos *token* es la número [56], la de los *nombres* es la [5] y la de los *nombres token* es la [7]

Mostremos la teoría explicando el ejemplo de la tienda:

```
...
        <!ATTLIST perro ID ID #REQUIRED>
```

```
        <!ATTLIST perro madre IDREF #IMPLIED>
        <!ATTLIST perro padre IDREF #IMPLIED>
        <!ATTLIST perro otrosnombres NMTOKENS "">
        <!ATTLIST perro fotos ENTITIES "">
...
```

Cada **perro** lleva un atributo **ID** que lo identifica, además se guardará la referencia ID de su **madre** y de su **padre**, siempre y cuando se tenga información de los mismos. Los niños que visitan la tienda pueden, si lo desean, sugerir un nombre para los perros, los más bonitos se almacenan en este atributo **otrosnombres**. Para finalizar el atributo **foto** que hará referencia ficheros con fotos del animal.

Declaraciones por defecto

La Declaración por defecto es la última parte de la declaración de un atributo. Éstos son los posibles valores para esta declaración:

#REQUIRED: si un atributo se declara de este modo, en el ejemplar del documento se debe dar valor a todos los atributos de este tipo. No hay valor por defecto.
#IMPLIED: los atributos que se declaran así pueden ser, opcionalmente, especificados en los elementos del tipo para el que se declara el atributo. No hay valor por defecto. Se sobreentiende un valor, de manera que el analizador de XML pasará un valor en blanco a la aplicación que puede poner su propio valor por defecto.
"valor del atributo": los atributos pueden incluir en su declaración valores por defecto, de forma que el atributo queda inicializado a ese valor por defecto, y si no se especifica ningún otro valor en el ejemplar del documento, el atributo contendrá ese valor. Como es obvio, el valor por defecto debe ser un valor permitido. Las comillas pueden ser simples o dobles.
#FIXED "valor del atributo": estos atributos no cambian su valor, todos los elementos de este tipo tienen un atributo con este valor fijo, no es posible modificar el valor de un atributo FIXED.

Los atributos ID sólo pueden tener como valor por defecto #IMPLIED y #REQUIRED, no se permiten otros valores por defecto, hay que notar que no tendría sentido un atributo ID declarado con un valor por defecto o con un valor fijo, ya que sería fácil que dos elementos acabaran con el mismo ID (lo que haría que el documento fuese de tipo no válido).

Para ilustrar la teoría sobre declaraciones por defecto, podemos terminar las declaraciones de los atributos de ejemplos anteriores, que ahora están, por fin, completas.

```
...
        <!ATTLIST perro fecha_nacimiento CDATA #REQUIRED>
        <!ATTLIST perro sexo (macho | hembra) #REQUIRED>
        <!ATTLIST perro ID ID #REQUIRED>
        <!ATTLIST perro madre IDREF #IMPLIED>
        <!ATTLIST perro padre IDREF #IMPLIED>
        <!ATTLIST perro otrosnombres NMTOKENS "">
        <!ATTLIST perro fotos ENTITIES "">
        <!ATTLIST perro veterinario CDATA #FIXED "Felix Marquez Sanz">
...
```

Los atributos **fecha_nacimiento**, **sexo**, **ID** se han declarado como obligatorios, no puede haber ningún **perro** que no tenga definido, como mínimo, estos tres valores, se entiende que son fundamentales. Al añadir nuevos animales se comprobará, mediante una herramienta que valide, que se incluye en su elemento asociado un atributo con su sexo, otro con su fecha de nacimiento y otro con su ID.

Sin embargo, **madre** y **padre** de tipo IDREF, son de especificación opcional. Conocer cuáles fueron los padres de un perro no es una información imprescindible, puede que esta información se almacene en otro lugar, o que incluso no esté disponible.

Los atributos **otrosnombres** y **fotos**, se han inicializado a unos valores que tendrán por defecto todas las realizaciones concretas de los mismos.

Existe un único tipo de datos fijo, **veterinario**, cada animal que está en la tienda tiene asignado uno, todos los perros comparten el mismo. Sólo interesa información actual, es decir, si en un momento dado el veterinario cambia, al cambiar el valor de este atributo se actualizarán automáticamente todas las ocurrencias del mismo.

Ahora que ya disponemos de los conocimientos suficientes podemos mostrar el ejemplo de partida completo, con los atributos.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE tienda_animales [
        <!-- Esta parte del documento no se ha estudiado todavía -->
        <!NOTATION jpg SYSTEM "programadegraficos.exe">
        <!NOTATION gif SYSTEM "programadegraficos.exe">
        <!ENTITY P250-01 SYSTEM "fotos/250-01.JPG" NDATA jpg>
        <!ENTITY P250-02 SYSTEM "fotos/P250-02.GIF" NDATA gif>
        <!ENTITY P250-03 SYSTEM "fotos/P250-03.JPG" NDATA jpg>
        <!-- Esta parte ya es familiar -->
        <!ELEMENT tienda_animales (perro)*>
        <!ELEMENT perro (#PCDATA | nombre | raza)*>
        <!ELEMENT nombre (#PCDATA)>
        <!ELEMENT raza (#PCDATA)>
        <!ATTLIST perro fecha_nacimiento CDATA #REQUIRED>
        <!ATTLIST perro sexo (macho | hembra) #REQUIRED>
        <!ATTLIST perro ID ID #REQUIRED>
```

```
        <!ATTLIST perro madre IDREF #IMPLIED>
        <!ATTLIST perro padre IDREF #IMPLIED>
        <!ATTLIST perro otrosnombres NMTOKENS #IMPLIED>
        <!ATTLIST perro fotos ENTITIES #IMPLIED>
        <!ATTLIST perro veterinario CDATA #FIXED "Felix Marquez Sanz">
]>
<tienda_animales>
        <perro fecha_nacimiento="1/4/1994" sexo="hembra" ID="P040">
          Extraordinaria hembra <raza>labrador</raza>, se trajo a la tienda
          con una herida en una pata trasera que ...
        </perro>
        ...
        <perro fecha_nacimiento="30/5/1996" sexo="macho" ID="P050">
          Macho <raza>labrador</raza> con un pelaje fantástico
          y una gran fortaleza,...
        </perro>
        ...
        <perro fecha_nacimiento="19/7/2000" sexo="macho" ID="P250"
              madre="P040" padre="P050" otrosnombres="Barullo Canela"
              fotos="P250-01 P250-02 P250-03">
          Este es un estupendo ejemplar de perro <raza>labrador</raza>, en
          la tienda le conocemos como <nombre>Chispa</nombre> por la
viveza
          de sus ojos. Su piel es de color canela...
        </perro>
        ...
</tienda_animales>
```

Se puede comprobar cómo los valores que finalmente toma cada ocurrencia de los atributos concuerdan con lo declarado. Así, **sexo** toma el valor de "hembra" y "macho". **ID** toma el valor de "P040", "P050" y "P250" (fíjese el lector que no podría tomar como valor una cadena que comenzara por un dígito como "250" por las limitaciones del tipo ID), así mismo, los valores que tomen **padre** y **madre** tienen que estar definidos en un atributo ID en el documento, como ocurre para el caso de *Chispa*, perro con la ID "P250". También podemos fijarnos en que, como mínimo, siempre están definidos los valores para los tres atributos requeridos.

Vemos en el ejemplo que los atributos otrosnombres y fotos admiten varios valores, por ser de tipo NMTOKENS y ENTITIES, todos estos valores son nombres token separados entre sí por espacios.

En teoría habría muchos más perros que formarían parte del documento, esto se indica mediante los puntos suspensivos colocados entre elementos de tipo **perro**, hay que darse cuenta de que un validador indicaría error, ya que estos puntos se consideran texto y los elementos **tienda_animales** sólo pueden contener elementos de tipo **perro**. Si se desea validar el ejemplo anterior hay que eliminar estos puntos suspensivos.

La parte que todavía no se ha estudiado declara las entidades (las fotos de los animales) que utiliza el documento, su localización y sus notaciones. Las notaciones deben estar declaradas e identifican la aplicación que puede gestionar estas entidades.

Por último cuando visualizamos el documento podemos comprobar cómo el procesador XML ha incluido atributos con valores por defecto dentro de cada elemento, como veterinario.

NORMALIZACIÓN DE LOS VALORES DE UN ATRIBUTO

Antes de utilizarse, los valores de los atributos se normalizan y se comprueba si su valor concuerda con el tipo con el que se declaró. Para normalizar el valor de un atributo se siguen los siguientes pasos:

> Las referencias carácter se sustituyen por el carácter correspondiente.
> Las entidades se sustituyen recursivamente si es necesario hasta que todas las referencias a entidades se han reemplazado.
> Los caracteres de espacio en blanco [**espacio** (Unicode/ASCII 32), **tabulador** (Unicode/ASCII 9), **retorno de carro** (Unicode/ASCII 13), y **salto de línea** (Unicode/ASCII 10)], se cambian, cada uno, por un espacio. La secuencia de caracteres retorno de carro y fin de línea se sustituyen por un único espacio.
> Si el atributo es de tipo CDATA, la normalización termina en el paso anterior; en otro caso se efectúan los siguientes pasos adicionales: se eliminan todos los espacios por delante y por detrás. y cada secuencia de espacios se reemplaza por un único espacio.

Los analizadores validadores normalizan los valores de los atributos sin ningún problema, conocen todos los tipos de los atributos, sin embargo los analizadores que no validan pueden no conocer todos los tipos, en ese caso normalizarán el valor del atributo como si fuera de tipo CDATA.

ATRIBUTOS PREDEFINIDOS

Existen una serie de atributos que están reservados para uso con XML. **Estos atributos solamente tienen sentido cuando se declaran en la DTD**, de otra manera su inclusión genera un error de buena formación del documento.

Atributo predefinido *xml:lang*

Uno de estos atributos que están reservados para el uso de XML, es el atributo **xml:lang**. Se utiliza para identificar el lenguaje en el que se ha escrito el contenido de un elemento y los valores de sus atributos (si los tiene). Los

posibles valores del atributo, que deben ser un nombre de tipo *token*, están especificados en la [RFC 1766].

Normalmente cada valor de este atributo se divide en dos partes, una primera que identifica el lenguaje y una segunda prescindible que identifica una variante de un país. Los códigos para los idiomas se encuentran recogidos en la [ISO 639] y los códigos para las variantes de cada país se especifican en la [ISO 3166].

Algunos ejemplos de códigos [ISO 639] para los lenguajes son los siguientes: ca (catalán), de (alemán), el (griego), en (inglés), es (español), eu (vasco), fr (francés), gl (gallego), it (italiano), y pt (portugués).

Como ejemplos de códigos para los países [ISO 3166] podrían listarse los siguientes: AR (Argentina), CA (Canadá), CU (Cuba), ES (España), FR (Francia), GB (Gran Bretaña), GR (Grecia) y MX (Méjico).

A pesar de que, a diferencia de otros valores para los atributos, no se distinguen las mayúsculas y las minúsculas de una misma letra (por motivos de compatibilidad), los códigos de lenguajes se escriben en minúsculas y los códigos para los países en mayúsculas. Así, posibles valores para el atributo **xml:lang** son los siguientes: **es-AR**, **es-CU**, **es-ES** y **es-MX**.

Como decíamos en la introducción es necesario declarar este atributo en la DTD, esto se realiza, como es obvio, para un elemento concreto. Se puede observar que el atributo **xml:lang** es de tipo NMTOKEN.

```
<!ATTLIST nombre_elemento xml:lang NMTOKEN valor_por_defecto>
```

Podemos mostrar un ejemplo de documento XML que podría ser útil en una librería, los títulos de los libros se guardan en atributos, su contenido puede estar en varios en varios idiomas que se especificarán mediante un atributo xml:lang:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE libros [
        <!ELEMENT libros (libro)*>
        <!ELEMENT libro EMPTY>
        <!ATTLIST libro título CDATA "">
        <!ATTLIST libro xml:lang NMTOKEN "es-CU">
]>

<libros>
 <libro xml:lang="fr" título="Tous les hommes sont mortels"/>
 <libro xml:lang="es-ES" título="Todos los hombres son mortales"/>
</libros>
```

Atributo predefinido *xml:space*

Los espacios en blanco pueden ser significativos o no dentro del marcado, pueden utilizarse para estructurar el texto o ser realmente una parte importante del texto. Para el procesador XML, dentro de los elementos que pueden tomar en sus valores datos carácter, los espacios en blanco son significativos y se tratan como cualquier otro carácter que forme parte de los datos carácter, no obstante, la aplicación que maneja los datos puede decidir eliminar estos espacios.

Con el objetivo de indicar a la aplicación que gestiona los datos qué debe hacer con los espacios en blanco, aparece otro atributo predefinido **xml:space**, que puede tomar los valores "defautl" o "preserve". El primero le indica a la aplicación que se comporte de manera predeterminada y el segundo le indica que conserve los espacios en blanco.

La forma general del atributo **xml:space** es la que sigue. Donde **nombre_elemento** es el nombre del elemento cuyos espacios en blanco se quieren preservar y el **valor_por_defecto** toma "**default**" o "**preserve**".

```
<!ATTLIST nombre_elemento xml:space (default | preserve) valor_por_defecto>
```

Se puede ver un ejemplo del atributo **xml:space** en el siguiente documento, para ello se ha rescatado un poema de un capítulo anterior:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Poema [
    <!ELEMENT Poema (#PCDATA)>
    <!ATTLIST Poema xml:lang NMTOKEN "es-ES">
    <!ATTLIST Poema xml:space (default | preserve) "preserve">
]>

<Poema xml:space="preserve">
    El Gallo (Popular)

    Gallo montero,
    gentil caballero,
    vestido de plumas
    como un coracero.

</Poema>
```

El atributo **xml:space** tiene un efecto heredado, es decir todos los hijos del elemento con este atributo preservarán los espacios en blanco, salvo que exista otro **xml:space** de nivel inferior en la jerarquía que determine otro modo de tratar los espacios.

Entidades y Declaraciones de Entidades.

Nuestro conocimiento sobre XML está alcanzando un buen nivel, uno de los temas pendientes que nos queda son las entidades. Ellas nos van a capacitar para realizar algunas cosas como las siguientes:

- Las entidades ya nos son familiares, a lo largo del curso se han utilizado para escapar caracteres, gracias a ellas es posible incluir caracteres de marcado [<],[<],[&], ["],['] sin que se interpreten como tales.

- A través de entidades es posible dar un nombre a un texto o a una cadena, y mediante referencias a la entidad, incluir esta cadena en otros lugares del documento. Útil en casos como los siguientes.
    - Puede ocurrir que en un documento XML una cadena se repita frecuentemente, utilizando entidades se evita tener que escribirla cada vez. Esta aplicación de las entidades reduce el texto del documento XML y en algunos casos puede ayudar a que el código fuente sea más legible.
    - Las entidades también nos pueden ayudar a realizar más eficientemente las actualizaciones de los documentos XML por dos razones, las cadenas a cambiar se recogen al principio del documento con lo que luego no hay que preocuparse de buscarlas, y porque, si existen varias ocurrencias de la misma cadena sólo se cambia en un sitio actualizándose todas sus ocurrencias "automáticamente".

- Otra utilidad que nos proporcionan las entidades es la de poder gestionar ficheros que no son nativos de XML, como imágenes, sonidos, etc.

El capítulo prosigue con una descripción de los distintos tipos de entidades existentes, su funcionalidad y las declaraciones implicadas, el curso continuará con su tónica general y completando las explicaciones se incluirán ejemplos.

¿QUÉ SON LAS ENTIDADES?

XML denomina a las unidades físicas que componen un documento, piezas de texto, archivos o recursos accesibles vía URL como **entidades** (las entidades no son siempre ficheros). Las entidades nos ayudarán a organizar un documento, independientemente de su estructura lógica. Hasta el momento se ha trabajado con, al menos, una entidad, la entidad que comprende el fichero XML de partida, y que se denomina **entidad documento**.

Los documentos XML pueden estar compuestos por varias unidades de almacenamiento. Como ya se ha visto, es posible incluir declaraciones de

marcado fuera del documento, o también incluir dentro del documento texto o imágenes que no están en el documento actual. Estas unidades son también entidades.

TIPOS DE ENTIDADES

Aunque la clasificación de las entidades es un asunto que se puede abordar de varias maneras, la *recomendación* deja muy claro que existen dos tipos principales de entidades, unas las llamadas "entidades generales" y otras las llamadas "entidades parámetro".

La recomendación comienza a hablar sobre las entidades en su apartado 4, *Estructuras Físicas*, y recoge en su regla de producción [70] la forma general de una declaración de una entidad.

Las entidades generales básicamente se utilizan para nombrar cadenas de texto que más tarde se utilizarán dentro del ejemplar del documento: valores de atributos, el contenido de algún elemento, elementos completos... También se utilizan para incluir datos que, en principio no son XML, como imágenes o sonidos, como parte del documento XML actual.

Las entidades parámetro, también se utilizan para poner un nombre a una cadena de texto, pero se utilizan exclusivamente dentro de la DTD del documento XML y contienen texto con sentido en este entorno, fundamentalmente declaraciones de marcado, aunque pueden contener datos carácter y marcado.

Además de estas limitaciones impuestas por el sentido que tiene la utilización de cada tipo de entidad, los contenidos de las entidades deben ser coherentes con el sitio en el que se reemplazarán, de otro modo el documento puede resultar que no esté bien formado o que no sea válido.

Podría decirse que existe un tercer tipo de entidad, las "entidades predefinidas", que se utilizan para escapar caracteres y que, a diferencia de los dos tipos anteriores, no hay que declarar en la DTD. De sobra conocidas, se hablará brevemente de ellas en este capítulo.

CARACTERÍSTICAS COMUNES

A pesar de sus diferencias las entidades generales y parámetro tienen en común algunas características:

- Los nombres de las entidades son nombres XML, deben cumplir sus restricciones. Los nombres de las entidades generales y de las entidades parámetro no entran en conflicto, ya que la manera de referenciarlas y de declararlas es diferente y no hay confusión posible.

- No se permiten referencias recursivas. Dentro del valor de una entidad no puede haber una referencia a sí misma.
- Ambos tipos de entidades se deben declarar en la DTD.
- Si una entidad se declara varias veces, el procesador XML toma como válida la primera e ignora el resto.
- No admiten entre sus valores los siguientes caracteres (salvo que se escapen apropiadamente):
    - El carácter [&] a menos que se utilice como comienzo de una referencia a una entidad general.
    - El carácter [%] a menos que se utilice como comienzo de referencia a una entidad parámetro.
    - El carácter de comilla [" o '] que cierra el valor literal de la entidad.

ENTIDADES GENERALES

Ya conocemos algunas de las utilidades de las entidades generales. En este apartado se van a estudiar con más detenimiento, en particular, veremos que es posible aplicarlas ciertos adjetivos, y que una entidad general puede ser interna o externa, analizable o no analizable. Antes de nada vamos a explicar qué significan estas palabras:

- Una **entidad analizada** (*parsed*) contiene marcado o datos carácter, información que el procesador XML entiende y que es capaz de tratar. Estas entidades se declaran y se utilizan de manera que puedan ser empleadas dentro del documento.
- Una **entidad no analizada** (*unparsed*) contiene información, que para el procesador XML no es XML, aunque podría. Generalmente, son ficheros en un formato no nativo de XML, imágenes, sonidos, etc. Estas entidades se deben incluir a través de atributos especiales y no mediante referencias, como ocurre con el resto de las entidades.
- Una **entidad es interna** si el contenido de estas entidades se ha definido totalmente dentro del documento XML de partida.
- Una **entidad es externa** si, al contrario de lo que ocurre con las entidades internas, el contenido de éstas se define total o parcialmente fuera de la entidad documento.

Así, y aunque las posibles combinaciones son cuatro, sólo existen tres tipos de entidades generales en función de los adjetivos que se les aplica:

- Entidades generales internas analizables: se utilizan a modo de comodín. Representan una cadena de texto que se encuentra definida completamente dentro del documento XML de partida. Se utilizan a través de referencias, que el procesador sustituirá por un texto. Su contenido es texto analizable por el procesador XML y que tiene sentido en el lugar donde se enlaza la referencia.
- Entidades generales internas no analizables: este tipo no se puede dar, ya que, como sabemos en un documento XML no se puede incluir directamente código binario, es decir código no analizable.

- Entidades generales externas analizables: son análogas a las entidades generales internas, con la diferencia de que están definidas (total o parcialmente) fuera de la entidad documento. Se utilizan, de igual manera a través de referencias.
- Entidades generales externas no analizables: este tipo es algo especial, puede contener ficheros de sonido, imágenes, documentos generados con un procesador de textos, hojas de cálculo, presentaciones, etc. Ya sabemos que en un documento XML no es posible la inclusión de ciertos caracteres, con lo que estos contenidos tienen que ser obligatoriamente externos. La utilización de este tipo implica algunas declaraciones de marcado adicionales aparte de la de la propia entidad. Su contenido se referencia a través de tipos de atributos especiales en los elementos.

Todas las declaraciones de entidades generales comienzan con la cadena [<!ENTITY] dentro de la DTD, seguidas de uno o más espacios, de un nombre XML válido (el nombre de la entidad) y de nuevo uno o más espacios (espacios XML). Para finalizar la definición del valor de la entidad dependiente de cada tipo de entidad, y cero o más espacios seguidos del carácter [>], que cierra la declaración.

A continuación se detallará cada tipo de entidad general y su declaración.

Entidades Generales Internas

Para centrarnos vamos a repetir que las entidades generales internas representan cadenas que tienen sentido dentro del ejemplar del documento o en valores de atributos, es decir, contienen texto XML que el procesador es capaz de interpretar, ya sea como marcado o como datos carácter, o una combinación (no pueden contener declaraciones de marcado), esto implica que las entidades generales internas son siempre analizables.

Las referencias a estas entidades serán sustituidas por sus correspondientes cadenas. Podríamos considerarlas como abreviaturas de texto. Cuando el procesador XML necesite conocer el contenido de un elemento o el valor de un atributo. Más adelante, en este apartado, veremos que este texto, a parte de ser analizable, debe, además, cumplir ciertas restricciones y ser coherente en el lugar que se sustituye.

Declaración y referencias

Dentro de la DTD, la forma que toman las declaraciones de entidades generales internas es la siguiente:

<!ENTITY nombre_de_la_entidad "valor de la entidad" >

Donde nombre_de_la_entidad es el nombre de la entidad que se declara, a partir de este momento el nombre de la entidad queda asociado al valor especificado entre comillas (dobles o simples). Las referencias a estas entidades tienen la forma:

&nombre_de_la_entidad**;**

Donde nombre_de_la_entidad es el nombre de la entidad declarada que el procesador sustituirá, cuando se necesite, por el texto definido entre las comillas presente en la declaración.

¿Dónde se admiten estas referencias?

Estas referencias no pueden ser incluidas en todos los sitios, puede utilizarse únicamente:

- Dentro del contenido de un elemento. La entidad se reemplaza y se procesa como si no hubiera estado nunca.
- Incluidas dentro del valor de un atributo (no se admiten como el valor de un atributo), la referencia se sustituye y se incluye dentro del literal. Los posibles caracteres de comilla simple o doble incluidos dentro de la entidad de reemplazo no finalizan el literal del atributo.
- Se pueden incluir dentro del valor de una entidad, la referencia permanecerá sin sustituir hasta que se necesite.

En particular, no se pueden utilizar en un DTD fuera del valor de una entidad, de un atributo, una instrucción de proceso, un comentario, o el literal de un identificador público o de sistema.

Ejemplos de aplicación y errores

A continuación se muestran algunos de los ejemplos de contenidos que admiten este tipo de entidades, otros que no y algunos errores.

En este primer ejemplo la entidad contiene sólo texto. Podemos comprobar cómo la entidad se encuentra definida completamente dentro de la entidad documento y que contiene texto XML bien formado (tiene sentido como valores de atributos o como contenido de algún elemento).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ejemplo_entidades [
        <!ELEMENT ejemplo_entidades (#PCDATA)>
        <!ENTITY Datos_caracter "Esta entidad sólo contiene texto" >
]>
<ejemplo_entidades> &Datos_caracter; </ejemplo_entidades>
```

Al visualizar el ejemplo en el navegador se puede comprobar cómo se ha sustituido la entidad por su correspondiente valor.

En el ejemplo que sigue las entidades sólo contienen elementos, se puede ver también un ejemplo de anidamiento de entidades generales internas, en la definición del valor de solo_elementos2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ejemplo_entidades [
        <!ELEMENT ejemplo_entidades (elemento)*>
        <!ELEMENT elemento EMPTY>
        <!ENTITY solo_elementos "<elemento></elemento>" >
        <!ENTITY solo_elementos2 "&solo_elementos;" >

]>
<ejemplo_entidades>&solo_elementos;</ejemplo_entidades>
```

Como podemos observar se admite cualquier combinación de elementos y datos carácter.

Las entidades generales internas son las únicas que se admiten dentro de los valores de los atributos, así los siguientes ejemplos serían posibles, en ellos se utilizan las entidades para dar un valor por defecto a un atributo en su declaración, así como para dar otro valor en el ejemplar.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ejemplo_entidades [
        <!ELEMENT ejemplo_entidades (#PCDATA)>
        <!ENTITY atributo "valor de un atributo" >
        <!ENTITY atributo2 "otro valor para un atributo" >
        <!ATTLIST ejemplo_entidades atributo CDATA "&atributo;">
]>
<ejemplo_entidades atributo="&atributo2;"></ejemplo_entidades>
```

Las entidades generales solamente pueden contener texto que esté bien formado, es decir que sea visto como bien formado desde fuera, independientemente del uso que se le pretenda dar, esto es, no pueden contener trozos de marcado. Declaraciones como las siguientes no se permiten.

```
<!ENTITY  fec "<fec" >
<!ENTITY  ha "ha>" >
<!ENTITY  fecha  "<fecha>" >
```

Tampoco admiten declaraciones de marcado.

```
 <!ENTITY  Declaración "<!ELEMENT Noticia (#PCDATA)>" >
```

Después de sustituir la referencia a la entidad, el texto resultante debe estar bien formado y si se desea que el documento sea de tipo válido, el texto de reemplazo de las entidades debe "casar" en el sitio en el que se sustituya, es decir, el texto de reemplazo debe estar de acuerdo con lo que se ha declarado, tanto para los elementos como para los atributos. No es posible, por ejemplo,

incluir dentro de un atributo de tipo ID una cadena de texto cualquiera, o dentro de un elemento, declarado para contener solamente otros elementos, no se admiten datos carácter.

Un ejemplo de lo anterior puede ser el siguiente párrafo. El procesador XML avisará de que no se admite el carácter [<] dentro del valor de un atributo:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE Noticia [
        <!ELEMENT Noticia (#PCDATA)>
        <!ENTITY Título_por_defecto "<nombre></nombre>" >
        <!ATTLIST Noticia Título CDATA "&Título_por_defecto;">
]>
<!-- Ejemplo de mala formación -->
<Noticia Título= '&Título_por_defecto;'></Noticia>
```

Entidades Generales Externas Analizables

Las entidades generales externas analizables, se utilizan para referenciar, de una manera sencilla, texto XML que no está incluido en el documento XML inicial. Comparten con las entidades generales internas la lógica del texto que puede contener: marcado y datos carácter; por este motivo tampoco admiten declaraciones de marcado, y deben cumplir las mismas restricciones de buena formación, y si el documento se valida, además deben cumplir las restricciones de la DTD en cuanto al contenido de elementos y atributos. **A diferencia de las entidades generales internas, las externas no se admiten dentro de atributos, ya sea en el ejemplar del documento o en la declaración de una lista de atributos.**

Declaración y referencias

Las entidades generales pueden declararse de dos maneras, siempre dentro de la DTD (en el subconjunto interno o externo):

Mediante un identificador de sistema. Como se muestra en la siguiente línea:

```
<!ENTITY  nombre_de_la_entidad SYSTEM "uri" >
```

O utilizando un identificador público alternativo junto con un identificador de sistema:

```
 <!ENTITY  nombre_de_la_entidad  PUBLIC "id_pública" "uri" >
```

Donde nombre_de_la_entidad es obviamente, el nombre de la entidad que se está declarando y "uri" es la localización del recurso (como un URL), e "id_pública" es un identificador público para el recurso. Los identificadores públicos son herencia de SGML y ofrecen una manera de referenciar indirectamente recursos externos al procesador XML. La palabra SYSTEM indica que lo que sigue no es un valor literal, si no un URI.

XML nos permite abstraernos de donde está el contenido de la entidad, si es interno o externo y las referencias a las entidades generales externas son iguales que las referencias a entidades generales internas:

&nombre_de_la_entidad;

Donde nombre_de_la_entidad es el nombre de la entidad declarada que el procesador sustituirá, cuando se necesite, por el texto definido entre las comillas en la declaración.

¿Dónde se admiten estas referencias?

Estas referencias no se pueden incluir en cualquier sitio, concretamente estos son los lugares donde se permiten y donde no:

- Dentro del contenido de un elemento, es decir, entre la etiqueta de inicio y de fin de un elemento. Estas entidades serán reemplazadas solamente si el procesador XML decidió obtener el valor de la entidad. Para los procesadores que validan el documento obtener estas entidades es obligatorio.
- No se admiten dentro del valor de un atributo, ni como el valor de un atributo en sí.
- Las referencias a entidades generales analizables son ignoradas dentro del valor de otra entidad, se dejan tal cual.
- En una DTD no se admiten fuera de valores de atributos, entidades, instrucciones de proceso, comentarios, o literales de identificador de sistema o público.

Ejemplos de aplicación y errores

Se continuará con algunos ejemplos que consolidarán la teoría, para empezar simplemente se mostrará una entidad general externa que contiene texto y algunos elementos.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE Noticia [
        <!ELEMENT Noticia (#PCDATA | Título | Fecha)*>
        <!ELEMENT Título (#PCDATA)>
        <!ELEMENT Fecha (#PCDATA)>
        <!ENTITY Título "Este es el título de la noticia" >
        <!ENTITY Texto_Noticia SYSTEM "47autobuses.txt" >
]>
<Noticia> &Texto_Noticia; </Noticia>
```

El contenido del archivo "**47autobuses.txt**" podría ser el siguiente:

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

<Título> &Título; </Título>
```

...En Madrid, 47 autobuses y 250 agricultores
...el hecho ocurrió el día
<Fecha>10/10/2000</Fecha>,...

Este ejemplo muestra dos de los lugares donde es posible incluir las referencias a estas entidades, dentro del contenido de un elemento y dentro del contenido de otra entidad. También se puede comprobar como existe un elemento de tipo Título que contiene una referencia a una entidad general interna declarada en la entidad documento de partida.

Entidades Generales Externas No Analizables

Las entidades generales externas no analizables se utilizan para relacionar con el documento XML contenidos que a efectos del procesador XML no serán XML. Estas entidades se relacionan con ficheros de imágenes, sonido y, en general, cualquier fichero que no sea XML.

Declaración y referencias

Las declaraciones de las entidades generales externas no analizables toman la forma siguiente:

<!ENTITY nombre_de_la_entidad SYSTEM "uri" NDATA nombre_de_notación ?>

Donde nombre_de_la_entidad se refiere igual que siempre, al nombre de la entidad que se está declarando. A continuación, detrás de la cadena SYSTEM se especifica un URI al recurso que se quiere asignar a la entidad. La cadena NDATA indica al procesador XML que detrás viene una notación (nombre_de_notación), a través del nombre de la notación el procesador XML puede relacionar la entidad con una aplicación que será capaz de tratarla. En un documento XML de tipo válido todas las notaciones se deben declarar.

Estas entidades únicamente se pueden referenciar a través de atributos especiales, atributos ENTITY o ENTITIES ya estudiados, es decir, estas referencias se admiten únicamente en atributos declarados especialmente a tal efecto y toman la siguiente forma:

<Elemento Atributo="nombre_de_entidad"> ... </Elemento>

Asociados a estas entidades existen pues:

- **La declaración de la entidad**: es necesario claro está especificar dónde se localiza esta entidad y darla un nombre de referencia, esto se realiza declarando la entidad.
- **Un atributo declarado como de tipo ENTITY o ENTITIES** que contendrá el nombre de la entidad.

- **Una notación declarada**: a través de la notación se identificará a la aplicación que es capaz de procesar la entidad general externa que se quiere incluir. Las declaraciones de notación se estudiarán en un capítulo próximo.
- **Un elemento que recoge el atributo**: para que un atributo recoja el nombre de la entidad es necesario el elemento asociado al atributo.

Ejemplos de aplicación y errores

Como de costumbre, a continuación se muestran algunos ejemplos que clarifican las explicaciones, en este caso se desea incluir como parte del documento XML varias fotografías.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE Noticia [
        <!ELEMENT Noticia (#PCDATA)>
        <!NOTATION gif SYSTEM "aplicacióngraficos.exe">
        <!NOTATION jpg SYSTEM "aplicacióngraficos.exe">
        <!ENTITY Foto_Noticia_1 SYSTEM "47autobuses.gif" NDATA gif >
        <!ENTITY Foto_Noticia_2 SYSTEM "47autobuses.jpg" NDATA
jpg >
        <!ATTLIST Noticia Fotos ENTITIES "">
]>
<Noticia Fotos="Foto_Noticia_1 Foto_Noticia_2">
...En Madrid, 47 autobuses y 250 agricultores ... el hecho ocurrió el día ...
</Noticia>
```

Se puede ver en el ejemplo la existencia de las diversas partes mencionadas y cómo se relacionan entre sí.

ENTIDADES PARÁMETRO

Al comenzar se presentó una ligera idea de la función de las entidades parámetro, es hora de estudiarlas con profundidad. Las entidades parámetro se utilizan a modo de comodines, exclusivamente dentro de la DTD (subconjunto interno y externo), y al igual que ocurría con las entidades generales, a las entidades parámetro, también se les puede aplicar ciertos adjetivos, ya conocidos.

Las entidades parámetro pueden ser internas o externas; además, son siempre analizables, este adjetivo se sobreentiende ya que no tendría sentido hablar de entidades parámetro no analizables puesto que su objetivo es ayudar en la redacción de las DTD y contienen cadenas útiles en este contexto.

Las entidades parámetro (internas o externas) se declaran de manera diferente, como era de esperar, muy parecida a las declaraciones de las entidades generales. Las declaraciones de entidades parámetro empiezan con la cadena [<!ENTITY], seguida de espacios en blanco, un carácter de porcentaje [%], más espacios en blanco, el nombre de la entidad (un nombre XML) y más espacios

en blanco. A continuación, viene una parte que depende del tipo de entidad que se declara, interna o externa. Para finalizar un carácter [>].

Si nos fijamos, esta declaración no se diferencia más que en el carácter [%] de las declaraciones de entidades generales. Hay que notar también que el carácter de [%] debe estar separado por espacios, lo contrario supone un error.

El carácter [%], también se emplea en las referencias a este tipo de entidades, que son de la forma:

%nombre_de_entidad;

Dentro del ejemplar del documento el carácter [%] pierde su sentido como iniciador de una referencia de entidad parámetro y es interpretado como cualquier carácter.

Características comunes a ambos tipos de entidades parámetro:

- Las entidades parámetro deben ser declaradas antes de poder ser utilizadas.
- Cuando una entidad parámetro (interna o externa) se utiliza en el subconjunto interno de una DTD solamente puede contener declaraciones de marcado completas. Esto implica que no se pueden utilizar referencias a entidades parámetro dentro de una declaración de marcado, salvo en la definición de otra entidad parámetro. Esta restricción tiene su explicación, procesadores de XML que no validan el contenido pueden decidir no procesar el contenido de referencias externas a entidades parámetro, con lo que si no se impusieran ciertas restricciones al contenido de estas entidades, el documento podría parecer a vista de estos procesadores como que no está bien formado. Estos procesadores aunque no validen tienen la obligación de comprobar que el documento XML está bien formado.
- El contenido de las entidades parámetro debe estar bien formado, es un requisito, al igual que lo fue para las entidades generales analizables.

¿Dónde se admiten estas referencias?

Al igual que se hizo con los otros tipos de entidades se ofrece una relación que muestra donde es posible incluir referencias a entidades parámetro. Hay que saber que:

- Las referencias a estas entidades dentro del contenido de un elemento, dentro del valor de un atributo, o como el valor de un atributo, son ignoradas.
- Las referencias a estas entidades incluidas dentro de otra entidad son incluidas dentro del literal. Es posible que existan comillas, dentro del texto de reemplazo, estas comillas no truncarán el valor de la entidad en la que se sustituye la referencia.

- Las referencias a estas entidades incluidas en la DTD y fuera del valor de una entidad, de un atributo, de una instrucción de proceso, de un comentario o fuera de los literales de identificadores públicos o de sistema, son sustituidas. En el caso de las referencias a entidades parámetro externas está en la mano del analizador obtener o no el texto de reemplazo. Para los validadores obtener el valor de estas entidades es obligatorio. En cualquier caso, si el texto de reemplazo se cambia por la referencia a la entidad, se añaden un espacio al principio y otro al final, con el objetivo de garantizar que estas entidades contienen unidades (mínimas o no) gramaticales completas (*gramatical token*) . Es decir, este tipo de entidades no pueden contener trozos de declaraciones que no tengan un sentido unitario. Un ejemplo de sentido unitario puede ser un nombre XML, lo anterior implica que no se pueden unir dos entidades parámetro para formar un nombre.

Entidades Parámetro Internas (Analizables)

Sabemos que las entidades parámetro se utilizan, al igual que las entidades generales, para nombrar cadenas. El contexto de utilización es dentro de la DTD, con la restricción de que estas entidades no pueden utilizarse en declaraciones de marcado. Su contenido está limitado pues a declaraciones de marcado, que deben ser enteras, deben comenzar y acabar dentro de la definición de la entidad.

Declaración y referencias

La forma de la declaración de las entidades parámetro internas es la siguiente:

<!ENTITY % nombre_de_la_entidad "valor de la entidad">

Es casi calcada a las declaraciones de las entidades generales internas salvo por el carácter [%]. De nuevo, nombre_de_la_entidad, es el nombre de la entidad que se declara, un nombre XML que cumple las restricciones. El valor de la entidad entre comillas, dobles o simples, como siempre.

Las referencias a las entidades parámetro internas se realizan como se muestra a continuación:

%nombre_de_la_entidad;

Estas referencias se reconocen únicamente dentro de la DTD y deben declararse antes de ser utilizadas. Este último requisito posibilita que la DTD sea construida en un solo paso, en una sola lectura.

Ejemplos de aplicación y errores

El siguiente listado muestra un documento sencillo que utiliza dos entidades parámetro internas. Su referencia se produce después que su declaración.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE Ciudades [
        <!ELEMENT Ciudades (#PCDATA | Ciudad | País)*>
        <!ENTITY % Ciudad "<!ELEMENT Ciudad (#PCDATA)>">
        <!ENTITY % País  "<!ELEMENT País (#PCDATA)>">
        %Ciudad;
        %País;
]>
<Ciudades>
        <País>Italia</País>
        <Ciudad>Pisa</Ciudad>
</Ciudades>
```

Seguidamente un ejemplo en el que una entidad parámetro interna contiene una declaración de un atributo que incluye una entidad general para definir su valor.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE Ciudades [
        <!ELEMENT Ciudades (#PCDATA | Ciudad | País)*>
        <!ENTITY % Ciudad "<!ELEMENT Ciudad (#PCDATA)>">
        <!ENTITY % País  "<!ELEMENT País (#PCDATA)>">
        <!ENTITY Habitantes "23.000.000">
        <!ENTITY % Atributo "<!ATTLIST País Habitantes CDATA
'&Habitantes;'>">
        %Ciudad;
        %País;
        %Atributo;
]>
<Ciudades>
        <País></País>
        <Ciudad></Ciudad>
</Ciudades>
```

En especial, dentro de subconjunto interno, estas entidades no pueden participar en ninguna declaración. Este comportamiento tiene su razón de ser para evitar que los procesadores XML que no gestionen entidades externas vean como mal formado el documento XML.

El siguiente ejemplo sería erróneo por que la declaración de la entidad "Todo" contiene referencias a entidades parámetro:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE Cuidades [
        <!ELEMENT Ciudades (#PCDATA | Ciudad | País)*>
        <!ENTITY % Ciudad "<!ELEMENT Nombre (#PCDATA)>">
        <!ENTITY % País  "<!ELEMENT País (#PCDATA)>">
        <!ENTITY % Todo  "%Ciudad; %País;">
        %Todo;
```

```
]>
<Ciudades>
        <País>Italia</País>
        <Ciudad>Pisa</Ciudad>
</Ciudades>
```

Entidades Parámetro Externas (Analizables)

Repasemos lo que ya sabemos sobre las entidades parámetro externas, para centrarnos, cumplen su función dentro de la DTD, y se utilizan para hacer más fácil la elaboración de la DTD, aunque sobre todo permiten que se compartan declaraciones de marcado entre varios documentos.

Las declaraciones de marcado que se incluyen en un documento a través de entidades parámetro externas, se consideran parte del subconjunto externo, ya que están definidas fuera de la entidad documento de partida, ahora bien a la hora de procesar la DTD, un procesador XML que procese también las entidades externas deberá procesar las entidades parámetro externas como si pertenecieran al subconjunto interno, es decir, las procesará en el momento que se las encuentre.

Un procesador XML que no procese entidades externas, debe parar de procesar la DTD cuando encuentre una referencia a una entidad externa que no puede procesar y que puede afectar a la interpretación del documento.

Declaración y referencias

Las entidades parámetro externas pueden declararse de dos maneras:

```
<!ENTITY %  nombre_de_la_entidad  SYSTEM "uri" >
```

o con identificador de sistema:

```
  <!ENTITY  % nombre_de_la_entidad  PUBLIC "id_pública" "uri" >
```

Donde nombre_de_la_entidad es obviamente, el nombre de la entidad que se está declarando y "uri" es la localización del recurso (como un URL), e "id_pública" es un identificador público para el recurso. Los identificadores públicos son herencia de SGML y dan una manera de referenciar indirectamente recursos externos al procesador XML(posiblemente a través de un catálogo). Son casi igual que las generales externas analizables. **Estas referencias se reconocen únicamente dentro de la DTD**.

Las referencias a las entidades parámetro internas se realizan de la siguiente forma:

```
%nombre_de_la_entidad;
```

Ejemplos de aplicación y errores

Estas entidades pueden contener partes de declaraciones de marcado, pero lo mínimo debe ser una unidad lógica completa: un nombre, un tipo, un valor por defecto, etc. El procesador XML añadirá un espacio en blanco delante y detrás del texto de reemplazo de la entidad.

Datos carácter que pueden formar parte del valor de un atributo (a través de una declaración de marcado) o de otra entidad. Indirectamente a través de una entidad general pueden definir el contenido de elementos y atributos en el ejemplar del documento.

Texto que puede formar parte de una entidad.

Elementos:

```
<Elemento nombre="" />
```

Datos carácter y elementos:

```
<Elemento>Datos caracter</Elemento>
```

Declaraciones de marcado:

```
<!ENTITY %  Nombre_de_la_entidad  SYSTEM "uri" >
```

Partes lógicas de una declaración de marcado:

```
<!ENTITY %  Nombre_de_la_entidad  SYSTEM "uri" >
```

ORDEN DE PROCESO. SOBREESCRITURA

Ahora que se conocen las entidades parámetro externas, es necesario completar la explicación relativa al orden de procesamiento del subconjunto interno y externo, ya que, conocer este orden es necesario para entender cómo se procesará el documento y cómo afectará una declaración.

Hay dos modelos de funcionamiento:

- Si el procesador solamente verificará que el documento esté bien formado puede elegir no reemplazar entidades externas. En este caso, comenzará a extraer los valores de los atributos y de las entidades hasta que encuentre una entidad parámetro externa que no procesará y que puede alterar el valor de sucesivos atributo o entidades.
- Si el procesador va a obtener el valor de las entidades externas, la extracción de los valores de los atributos y entidades comienza por el subconjunto interno de la DTD, si encuentra una entidad parámetro externa la procesará (subconjunto externo implícito) y la tratará como si fuera parte del subconjunto interno. Finalmente, y una vez terminado el procesado de los anteriores, se procesará el subconjunto externo explícito.

Durante este proceso puede ocurrir que una determinada declaración, de elemento, atributo o entidad esté duplicada, en estos casos puede ocurrir lo siguiente:

- Si se ha declarado un elemento mas de una vez, se genera un error fatal, un error de buena formación y se detiene el proceso. El procesador podrá continuar a su elección leyendo el documento y buscando nuevos errores pero nunca podrá continuar un procesado normal del documento.
- Sucesivas declaraciones de atributos son ignoradas y solamente tiene efecto la primera.
- Las declaraciones repetidas de entidades también son ignoradas.

Este orden en el procesado de la DTD, subconjunto interno y subconjunto externo implícito primeramente y subconjunto externo explícito por último, permite al usuario de XML ignorar ciertas declaraciones y personalizar su documento con declaraciones internas/externas respecto a un subconjunto externo explícito y posiblemente compartido (genérico).

SECCIONES CONDICIONALES

El subconjunto externo (que no el interno), permite la utilización de las secciones condicionales, un mecanismo para permitir la inclusión opcional de las declaraciones de marcado incluidas entre sus corchetes.

```
<![INCLUDE [
...
]]>

<![IGNORE [
        ...
]]>
```

Estas secciones combinadas con entidades parámetro pueden hacer que unas declaraciones se incluyan o no.

```
<!ENTITY % SioNo "INCLUDE" >
<![%SioNo; [
<!ELEMENT ... >
<!ATTLIST ...>
]]>
```

Se pueden anidar. Cuidado con la cadena []]>] podría terminar anticipadamente la sección condicional.

ENTIDADES PREDEFINIDAS

Para poder utilizar cualquier entidad es necesario declararla en una DTD o en un Esquema, existen, sin embargo, un conjunto de entidades que ya están predefinidas, son las entidades que ya conocemos y que se utilizan para

escapar un carácter que no queremos que se interprete como marcado. Son las siguientes.

& &
' '
> >
< <
" "

Estas son las entidades más fáciles de utilizar, ya que no requieren ningún conocimiento más que el de su uso.

```
<?xml version="1.0"?>
<Ejemplo_de_uso_de_entidades_predefinidas>
  Existen ciertos caracteres que se interpretarán como
  marcado a menos de que se escapen, sabemos los que son:
        &, &apos;, >, < y "
</Ejemplo_de_uso_de_entidades_predefinidas>
```

Más apuntes sobre entidades

Para finalizar el capítulo vamos a ver algunos conceptos finales sobre entidades que combiene conocer:

Restricciones de buena formación

Todas las entidades internas/externas analizables deben contener XML bien formado y deben verse como bien formadas desde fuera de la entidad. En general, no pueden contener elementos parciales (con etiqueta de inicio pero sin fin), o marcado parcial (trozos de etiquetas).

Declaración de texto

Las entidades externas analizables (generales y parámetro) pueden contener una cabecera que declare la versión de XML y la codificación utilizada en el documento. Esta cabecera recibe el nombre de declaración de texto y a diferencia de las declaraciones de documento XML no pueden contener una declaración "standalone". Cuando el procesador XML obtiene el contenido de estos ficheros, lo transformará para que todos los caracteres estén codificados internamente de la misma manera.

Entidades y texto de reemplazo

Las entidades pueden incluir dentro de su valor otras referencias a entidades, por lo que el valor literal de la entidad no tiene por qué coincidir con el texto de reemplazo.

Antes de sustituir el valor de una entidad, parámetro o general, ocurren los siguientes pasos:

- Las referencias a las entidades parámetro se sustituyen por el correspondiente texto de reemplazo. Este proceso se continúa hasta que todas las posibles referencias a entidades parámetro dentro del valor literal de la entidad se han sustituido. Este proceso es necesario ya que pueden variar la interpretación que se haga del documento.
- Las referencias a carácter son sustituidas por el carácter correspondiente.

Las entidades generales se dejan tal cual y se sustituyen cuando se utilizan, es decir, cuando es requerido el valor del elemento o del atributo que las contiene.

El proceso anterior, es lógico si pensamos que las referencias a carácter y las entidades parámetro se necesitan para interpretar la gramática del documento y analizarla. Las entidades generales son irrelevantes en este sentido y se pueden expandir cuando se necesiten para especificar el valor de un atributo o el de otra entidad general o el de un elemento.

El análisis se hace después de que las entidades se sustituyan. Hay que tener cuidado cuando se escapa marcado dentro de una entidad parámetro o general, en especial con los atributos que definen sus valores con entidades y dentro también.

Otros elementos de marcado

En este capítulo se explica el concepto de notación (**NOTATION**), a pesar de que ya se ha hablado sobre ellas en secciones anteriores es aquí donde entenderá sus funciones.

También nos muestra la utilización de las instrucciones de proceso, que son un mecanismo que proporciona XML para poder enviar instrucciones a las aplicaciones que procesarán un documento XML.

Finalmente, se estudiarán las secciones **CDATA** que se emplean para escapar textos y los espacios de nombres (**NameSpaces**) utilizados para especificar un contexto de aplicación de los nombres presentes en los documentos.

NOTACIONES

Las notaciones, básicamente, identifican un nombre XML con una aplicación auxiliar capaz de procesar ciertos datos y que podrá ser empleada por el procesador XML, o por la aplicación a la que sirve, para realizar alguna tarea. Las notaciones también se utilizan para realizar negociación de contenidos.

Las declaraciones toman básicamente alguna de las dos formas siguientes:

```
<!NOTATION nombre_de_la_notación SYSTEM "uri_aplicación">
```

o bien

```
<!NOTATION nombre_de_la_notación PUBLIC "id_pública" "uri_aplicación">
```

Donde **nombre_de_la_notación** se refiere al nombre de la notación y debe ser un nombre XML, "**uri_aplicación**" identifica un URI que localiza a la aplicación que es capaz de procesar los datos. En el caso de la segunda forma, "**id_publica**" es un identificador público para el recurso que podrá ser utilizado para localizar alternativamente a la aplicación. Estas declaraciones se sitúan dentro de la DTD, en el subconjunto interno o externo y en un documento XML de tipo válido todas las notaciones deben declararse.

Las notaciones se utilizan básicamente para:

Indicar a través de un atributo notación cómo interpretar el contenido de un elemento.
En la declaración de entidades generales externas no analizables, relacionadas con los elementos a través de atributos ENTITY y ENTITIES.
En las instrucciones de proceso, para identificar la aplicación destino de la instrucción.

Veamos algún ejemplo, éste se utilizaba en un capítulo anterior para incluir fotos sobre animales en una tienda de animales, se puede comprobar cómo las notaciones están declaradas y se utilizan para declarar ciertas entidades externas no analizables:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE tienda_animales [
        <!NOTATION jpg SYSTEM "programadegraficos.exe">
        <!NOTATION gif SYSTEM "programadegraficos.exe">
        <!ENTITY P250-01 SYSTEM "fotos/250-01.JPG" NDATA jpg>
        <!ENTITY P250-02 SYSTEM "fotos/P250-02.GIF" NDATA gif>
        <!ENTITY P250-03 SYSTEM "fotos/P250-03.JPG" NDATA jpg>

        <!ELEMENT tienda_animales (perro)*>
        <!ELEMENT perro (#PCDATA)*>
        <!ATTLIST perro fotos ENTITIES "">
]>
<tienda_animales>
        <perro fotos=" P250-03 P250-02">Este perro ...</perro>
</tienda_animales>
```

El ejemplo anterior se puede modificar para que el procesador XML sea capaz de, a la hora de obtener una entidad externa (suponiendo que decida

realizarlo), negociar un formato u otro, para ello, se ha declarado una notación (varios) que recoge una secuencia de tipos MIME:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE tienda_animales [
   <!NOTATION varios SYSTEM "image/gif:image/jpg">
   <!NOTATION jpg SYSTEM "programadegraficos.exe">
   <!NOTATION gif SYSTEM "programadegraficos.exe">
   <!ENTITY P250-01 SYSTEM "fotos/250-01.JPG" NDATA varios>
   <!ENTITY P250-02 SYSTEM "fotos/P250-02.GIF" NDATA varios>
   <!ENTITY P250-03 SYSTEM "fotos/P250-03.JPG" NDATA varios>

   <!ELEMENT tienda_animales (perro)*>
   <!ELEMENT perro (#PCDATA)*>
   <!ATTLIST perro fotos ENTITIES "">
]>
<tienda_animales>
        <perro fotos=" P250-03 P250-02">Este perro ...</perro>
</tienda_animales>
```

INSTRUCCIONES DE PROCESO

Ya conocemos alguna instrucción de proceso, sin ir más lejos, la declaración XML que se ha utilizado durante todo el curso es una instrucción de proceso, destinada a informar que el contenido del documento es XML, la versión, la codificación utilizada y si es autónomo o no.

El formato general de una instrucción de proceso es el siguiente:

```
<?aplicación_objetivo cadena ?>
```

Las instrucciones pueden hacen uso de notaciones para declarar cuál es la aplicación a la que se envía los datos, aunque no es un requisito. En el formato presentado "**aplicación_objetivo**" representa un hipotético nombre de notación. En cualquier caso, objetivo es un nombre XML. Lo que continúa, **cadena**, es el contenido de la instrucción es decir, el comando que se enviará a la aplicación declarada en la notación. La única restricción en cuanto al contenido de la cadena, es que, como es lógico, no puede contener los caracteres que cierran la instrucción [**?>**].

El procesador XML debe pasar las instrucciones de proceso a la aplicación para la que el procesador XML está trabajando.

Las instrucciones de proceso pueden aparecer en gran número de lugares:

En el prólogo dentro de la DTD
Entre el prólogo y el ejemplar del documento.
Dentro del ejemplar del documento.
Después del ejemplar del documento, después de la etiqueta de cierre del elemento raíz.

Un ejemplo ilustrativo de una instrucción de proceso podría ser el siguiente. En él se envía a un navegador una hoja de estilos en cascada que definirá cómo debe visualizarse.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet type="text/css" ?>
<!DOCTYPE Poema [
	<!ELEMENT Poema (Título , Cuerpo)>
	<!ELEMENT Título (#PCDATA)>
	<!ELEMENT Cuerpo (#PCDATA)>
]>
<Poema xmlns:HTML="http://www.w3.org/Profiles/XHTML-transitional">
	<HTML:STYLE>
		Título {
			display:block;
			color: #F00FDD;
			font-style: italic:
			text-align: center;
		}
		Cuerpo {
			display:block;
		}
	</HTML:STYLE>
	<Título>El Gallo (Popular)</Título>
	<Cuerpo>
		Gallo montero,
		gentil caballero,
		vestido de plumas,
		como un coracero.
	</Cuerpo>
</Poema>
```

La instrucción de proceso en cuestión está en la segunda línea, indica al navegador que el formato visual del documento se haya definido mediante una hoja de estilos en cascada (**text/css**).

Esta instrucción de proceso no se puede considerar típica ya que necesita algunos pasos adicionales, un espacio de nombres, un atributo **xmlns** y un elemento especial (**HTML:STYLE**), requerimientos que no son necesarios en otras instrucciones de proceso. A cambio, ilustra perfectamente una utilidad de las instrucciones de proceso. Las hojas de estilo en cascada son motivo de estudio posterior en el curso.

SECCIONES CDATA

Hasta ahora se han aprendido maneras para escapar caracteres, las secciones **CDATA** son una forma cómoda de escapar textos, o de especificar al procesador XML que un determinado texto no debe ser interpretado como marcado. Las secciones CDATA se consideran como información que debe

pasarse a las aplicaciones, a diferencia de los comentarios que no se consideran datos carácter, y el procesador XML no tiene la obligación de pasarlos.

Las secciones CDATA comienzan por la cadena [**<![CDATA[**] y terminan con la cadena [**]]>**]. Como muestra el siguiente ejemplo:

```
<![CDATA[ Este es un texto que puede ser <HTML>  ]]>
```

Estas cadenas se deben escribir exactamente así, sin ningún carácter adicional, de otra manera se generará un error.

Las secciones CDATA no pueden contener datos binarios, ya que como sabemos un documento XML no puede contener ciertos caracteres. Los únicos caracteres que no pueden incluirse en una sección **CDATA**, son obviamente los que la cierran, [**]]>**], esto evidencia la imposibilidad de anidar secciones **CDATA**.

Un ejemplo más extenso, integrado dentro de documento, de una sección CDATA podría ser el siguiente listado:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<tutorial_HTML>
      Una página sencilla de prueba, con todas las partes
      básicas de un documento HTML:
      <![CDATA[
      <HTML>
       <HEAD>
        <TITLE>prueba</TITLE>
        <META HTTP-EQUIV='Content-Type'
            CONTENT='text/html; charset=iso-8859-1'>
       </HEAD>
       <BODY BGCOLOR='yellow'>
       <H1>prueba</H1>
       </BODY>
      </HTML>
      ]]>
      ...
</tutorial_HTML>
```

En general las secciones CDATA tienen una utilidad bastante reducida y quizás una de sus aplicaciones más comunes sea la de permitir incluir código *HTML*, *Java*, *JavaScript*, etc. sin necesidad de escaparlo. Aunque para incluir código en un documento XML utilícense mejor las instrucciones de proceso, creadas para tal fin.

ESPACIOS DE NOMBRES

Los espacios de nombres se utilizan para incluir a los nombres de los elementos y atributos un contexto de aplicación y de esta forma evitar confusiones con otros componentes de igual nombre pero referidos a distintos entornos.

Los espacios de nombres están asociados a URI que los identifica, se supone que únicamente.

Los espacios de nombres son útiles por ejemplo, en casos en los que se van a crear nuevos documentos a partir de otros. ¿Qué ocurriría si se mezclan los dos documentos siguientes en uno sólo?

```
<proveedores>
        <nombre id="2">Productos Ecológicos S.A.</nombre>
<proveedores>
```

Y un fichero con los productos que ofrece un determinado proveedor.

```
 <productos>
        <nombre id="2">Palillos Redondos</nombre>
        <nombre id="122">Palillos Planos</nombre>
        <nombre id="213">Palillos extra largos</nombre>
 <productos>
```

Una mezcla, bastante poco hábil podría llevar a un resultado como el siguiente, en el que los nombres no se distinguen.

```
 <proveedores_productos>
        <nombre id="2">Productos Ecológicos S.A.</nombre>
        <nombre id="2">Palillos Redondos</nombre>
        <nombre id="122">Palillos Planos</nombre>
        <nombre id="213">Palillos extra largos</nombre>
 </proveedores_productos>
```

La inclusión de un espacio de nombres puede solucionar el problema.

```
 <?xml version="1.0" encoding="ISO-8859-1"?>
 <productos xmlns:productos="http://esmeralda/productos">
        <productos:nombre productos:id="2">Palillos Redondos</nombre>
        <productos:nombre productos:id="122">Palillos Planos</nombre>
        <productos:nombre productos:id="213">Palillos extra
largos</nombre>
 <productos>
```

Ciertamente la cosa cambia. Para incluir un espacio de nombres es necesario declararlo, es decir asociar un índice con el URI asignado al espacio, mediante un atributo especial **xmlns**, tal y como se muestra en el ejemplo anterior.

Los espacios de nombres (*Namespaces*) tienen una recomendación, en XML: http://www.w3.org/TR/REC-xml-names/.

Los nombres de los atributos y elementos pueden aparecer con un prefijo referido al espacio de nombres al que se refieren. En este caso el prefijo recibe el nombre de "**el prefijo del espacio de nombres**", a continuación del prefijo aparece un caracter de dos puntos y por último la parte local, o el nombre del elemento. Todo este conjunto se denomina "**nombre cualificado**". El nombre cualificado también puede estar formado por la parte local exclusivamente, en casos en los que no se especifique el prefijo.

Los URI que definen un espacio de nombres pueden contener caracteres no permitidos en los nombres, por ello no se pueden utilizar directamente y es necesario emplear el prefijo.

El espacio de nombres en un elemento se hereda a sus hijos, que pueden si lo desean sobreescribir este espacio.