DTD- ATRIBUTOS-ENTIDADES

DTD (Document Type Definition)

INTRODUCCIÓN

Los documentos XML con los que hemos trabajado hasta ahora 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 especímenes 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.

Ya 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, en definitiva, certifican 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?

- Recordemos que 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>16-5-2010</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>16-5-2010</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.

EL PRÓLOGO. 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".

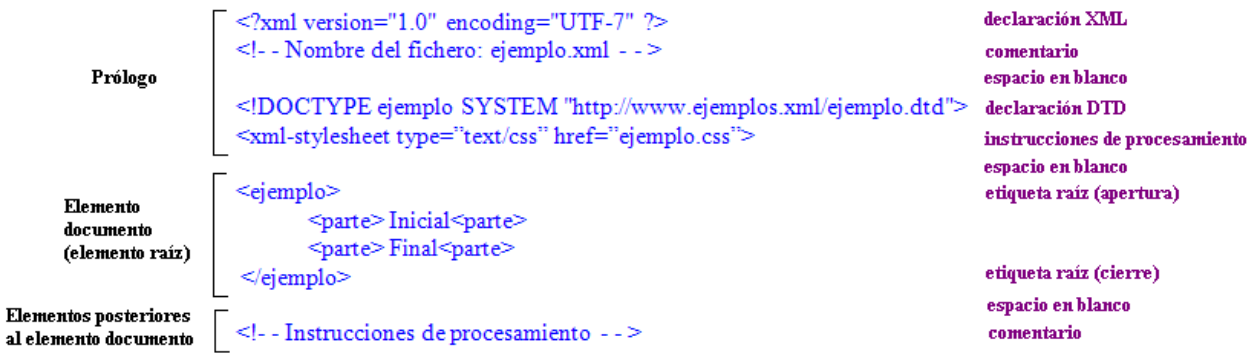
Un documento XML puede constar de dos partes, **un prólogo** y **un ejemplar**. El prólogo a su vez puede componerse de otras dos partes:

- Una **"declaración XML"**, ya estudiada.
- Una **"declaración de tipo"**.

Conceptualmente la **declaración de tipo consta de dos partes**:

- 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



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>
```

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 (Element type).
- Declaraciones de listas de atributos para los tipos de elementos (attribute list).
- Declaraciones de entidades (entity).
- Declaraciones de notación.

- El formato general de una definición elemental en una DTD es:
  <!clase parámetros ...>

donde clase será ELEMENT, ATTLIST, ENTITY o NOTATION, y los parámetros dependerán de la clase de definición.

- Elementos:
  - Los elementos configuran la estructura general de un documento DTD
  - Se anidan unos dentro de otros formando un árbol
  - Para cada elemento se define su nombre y una regla.
- Atributos:
  - Los atributos son fragmentos de información asociados a un elemento
  - Tienen nombre y su contenido es siempre texto. No pueden anidarse.
- Entidades:
  - Las entidades son similares a las macros de ciertos lenguajes de programación
  - Son fragmentos de texto constantes a los que se puede hacer referencia mediante un nombre
  - Sirven para simplificar la escritura de documentos y DTD's en los que aparecen repetidamente ciertos fragmentos de texto
- Notaciones:
  - Las notaciones permiten delimitar contenido no XML dentro de un documento XML.

Para ir tomando contacto con la DTD (la definición del tipo del documento), añadiremos algunas **"declaraciones de tipos de elementos"** que forman la "definición del tipo" y que dan sentido a la "declaración del tipo".

```
<?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). 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.

UTILIZACIÓN DE LA DTD

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

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 siguiente forma:

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

```
<!DOCTYPE Nombre_Elemento_Raiz[declaraciones          ...]>

Nombre_Elemento_Raiz: contienen el nombre del elemento raíz.

Declaraciones: muestra una lista de declaraciones de los elementos y atributos
```

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

```
<?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>
```

EJEMPLO RESUMEN: DTD INTERNO

Se quiere almacenar los mensajes de móviles que se envían a un servidor. Los datos que se guardarán son los siguientes:

- Número de teléfono del usuario
- Fecha de envío
- Hora de envío
- Contenido del mensaje

Crear primero el XML que permita almacenar dicha información y posteriormenteuna DTD que permita establecer el formato de intercambio de estos mensajes.

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

```
<!DOCTYPE BDsms [
  <!ELEMENT BDsms (sms*)>
  <!ELEMENT sms (telefono,fecha,hora,mensaje ) >
  <!ELEMENT telefono (#PCDATA)>
  <!ELEMENT   fecha (#PCDATA)>
```

```
  <!ELEMENT hora (#PCDATA)>
  <!ELEMENT mensaje (#PCDATA)>
]>
<BDsms>
 <sms>
  <telefono>23456756</telefono>
  <fecha>1/3/2012</fecha>
  <hora>11:30</hora>
  <mensaje>Juego : parchis</mensaje>
 </sms>

 <sms>
  <telefono>1213231</telefono>
  <fecha>1/3/2012</fecha>
  <hora>11:30</hora>
  <mensaje>Juego : oca</mensaje>
 </sms>
 <sms>
  <telefono>435465366</telefono>
  <fecha>1/3/2012</fecha>
  <hora>11:30</hora>
  <mensaje>Juego : quien es quien</mensaje>
 </sms>
 </BDsms>
```

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.

Sólo se puede añadir una DTD externa en un documento XML.

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.

- **En un documento externo privado**

  En este caso la validación se crea en un documento-plantilla externa. De modo que cuando un documento debe cumplir las reglas de la plantilla DTD, se debe indicar la ruta (sea relativa o absoluta) a la misma.

- **La extensión de los archivos DTD suele ser precisamente esa, dtd.**

La sintaxis de la etiqueta DOCTYPE que permite asignar un DTD privado a un documento XML es:

<!DOCTYPE raíz SYSTEM "rutaURLalDTD">

Salvo que se desee crear un único documento con una validación DTD, lo lógico es utilizar la forma de DTD externa ya que de esa forma se pueden validar varios documentos a la vez.

- La ruta puede ser absoluta y entonces se indica su URL:

  <!DOCTYPE raíz **SYSTEM** "http://www.empresa.com/docs.dtd"**>**

- Pero puede ser relativa:

  <!DOCTYPE raíz **SYSTEM** " docs.dtd"**>**

Entonces se busca al archivo DTD desde el directorio donde se encuentra el archivo XML que queremos validar (en el ejemplo, el archivo **docs.dtd** debe encontrarse en el mismo directorio que el archivo que contiene ese código **DOCTYPE**).

En ambos casos se puede añadir código DTD para en ese documento concreto añadir instrucciones de validación. Ejemplo:

<!DOCTYPE raíz **SYSTEM** "http://www.empresa.com/docs.dtd" **[**
<!ELEMENT nombre (#PCDATA)**>**
] >

- **DTD externo de tipo PUBLIC**

Si planeamos proporcionar esa DTD de forma pública deberíamos utilizar otra declaración, más correcta:

<!DOCTYPE raiz **PUBLIC "docs"** http://www.empresa.com/docs.dtd>

A continuación de PUBLIC y antes de la URL en la que está ubicada la DTD debemos situar el nombre que recibe**, docs.**

Nota:

En el caso de una DTD externa, el atributo **standalone** de la declaración del documento debería tomar el valor **no**, puesto que es necesaria la intervención de un elemento externo, del que depende. Y**es** En caso de que el documento XML no utilice DTD externa

- **El tipo PUBLIC se utiliza cuando la DTD es compartida por múltiples usuarios de organizaciones distintas. Se utiliza para ello un identificador público y se añade una URI alternativo por si el público da problemas.**

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

En resumen, las posibilidades para la Definición del Tipo de Documento (DTD o Document Type Declaration) son:

- <!DOCTYPE rootname [DTD]>
- <!DOCTYPE rootname SYSTEM URI>
- <!DOCTYPE rootname SYSTEM URI [DTD]>
- <!DOCTYPE rootname PUBLIC identifier URI>
- <!DOCTYPE rootname PUBLIC identifier URI [DTD]>

Donde "rootname" es el elemento raíz del documento, "URI" es la dirección o URL en la que se ubica el documento externo DTD, "identifier" es el identificador de la URI en el caso de que usemos un DTD público y "[DTD]" es el conjunto de declaraciones internas que podemos hacer de forma directa con independencia de que haya o no un DTD en un archivo externo tanto si se encuentra en el sistema como si es público.

Sólo nos queda indicar cuál es el formato que debemos usar para el **"identificador"** en el caso de que sea público (Formal Public Identifier o FPI – Identificador Publico formal):

- Primer campo: **"-"** no standard, **"+"** standard DTD . Este carácter (-) indica que la organización no está registrada por ISO, como de hecho ocurre con el W3C. Si la organización sí está registrada, se sustituye por (+).
- Segundo campo: nombre de grupo o persona responsable del DTD
- Tercer campo: tipo de documento y número de versión
- Cuarto campo: lenguaje (2 letras)
- Los campos van separados por //

Por ejemplo:

<!DOCTYPE LIBRO PUBLIC "-//Pilarl //Libros Version 1.3//ES"
"http://www.biblioteca.org/libro.dtd">

- El signo "-" significa que la DTD no ha sido aprobado por una norma formal.
- Segundo campo es el nombre o grupo de personas responsable de la DTD
- Tercer campo , el tipo de documento y versión
- Cuarto :idioma

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

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

EJEMPLO RESUMEN: dtd externo

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

<!DOCTYPE BDsms SYSTEM “BDsms.dtd”>
<BDsms>
 <sms>
  <telefono>23456756</telefono>
```

```
  <fecha>1/3/2012</fecha>
  <hora>11:30</hora>
  <mensaje>Juego : parchis</mensaje>
</sms>

 <sms>
  <telefono>1213231</telefono>
  <fecha>1/3/2012</fecha>
  <hora>11:30</hora>
  <mensaje>Juego : oca</mensaje>
</sms>
 <sms>
  <telefono>435465366</telefono>
  <fecha>1/3/2012</fecha>
  <hora>11:30</hora>
  <mensaje>Juego : quien es quien</mensaje>
</sms>
</BDsms>
```

BDsms.dtd

```
  <!ELEMENT BDsms (sms*)>
  <!ELEMENT sms (telefono,fecha,hora,mensaje ) >
  <!ELEMENT telefono (#PCDATA)>
  <!ELEMENT   fecha (#PCDATA)>
  <!ELEMENT hora (#PCDATA)>
  <!ELEMENT mensaje (#PCDATA)>
```

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. Mas tarde 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.

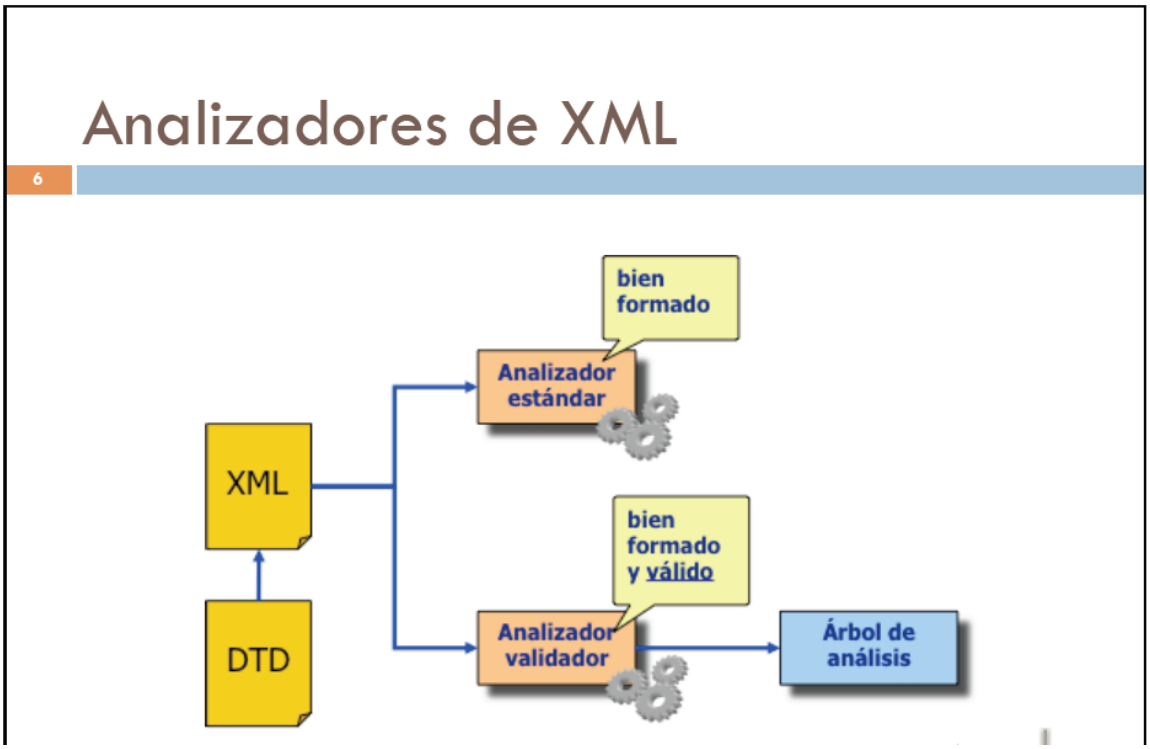
No todos los procesadores XML comprueban si un documento es válido o no, sin embargo, todos comprueban que el documento esté bien formado, ya que de otra manera no podrían realizar un procesamiento fiable de los documentos XML.

Un parser o procesador **o analizador sintáctico** lee el documento XML y verifica que es XML bien formado, algunos también comprueban que el código XML sea válido.

El parser o procesador de XML es la herramienta principal de cualquier aplicación XML. Mediante el parser no solamente podemos comprobar si nuestros documentos son bien formados o válidos, sino que también podemos incorporarlos a nuestras

aplicaciones, de manera que estas puedan manipular y trabajar con documentos XML. Podemos dividir los parsers XML en dos grupos principales:

- Sin validación: el parser no valida el documento utilizando un DTD, sino que sólo chequea que el documento esté bien formado de acuerdo a las reglas de sintaxis de XML (sólo hay una etiqueta raíz, las etiquetas están cerradas, etc).
- Con validación: además de comprobar que el documento está bien formado según las reglas anteriores, comprueba el documento utilizando un DTD (ya sea interno o externo).



- Analizadores sintácticos *no validadores*: solamente verifican que el documento XML está bien formado. Los últimos navegadores integran estos analizadores como parte integral de sus componentes software.
  - Expat, Lark, XP, TclXML, XML Testbed, RUWF…
- Analizadores sintácticos *validadores*: realizan las tareas adicionales de verificar el contenido de un documento XML respecto a una DTD.
  - LT XML, XML4Java, MS XML, MXL Checker…

REALIZAR EL EJERCICIO: CREAR UN DOCUMENTO DTD EXTERNO Y OTRO INTERNO SOBRE EL DOCUMENTO ALUMNOS.XML.

1. Crear un documento xml “alumnos.xml” bien formado y valido para describir elementos relativos a los alumnos de una clase, incluido su número de teléfono móvil, se debe especificar si es fijo o es móvil. Insertar varios alumnos.

    1- Declaración del tipo de documento
    2- Insertamos un comentario para explicar que información tendrá el documento.
    3- Declaramos el elemento raíz alumnos
    4- Creamos el primer item, alumno. Que tiene nombre, apellidos1, apellidos2, teléfono con un atributo tipo que puede ser fijo o móvil.

    REALIZAR UN DOCUMENTO XML CON SU CORRESPONDIENTE DTD Y CSS PARA QUE MUESTRE LA SIGUIENTE IMAGEN

DTD- TIPOS DE ELEMENTOS ELEMENT

1. DECLARACIONES DE TIPOS DE ELEMENTOS

Las Declaraciones de Tipo de Elementos se componen de una cadena que contiene:

<!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 [>].

Ejemplo:

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

En este ejemplo, el elemento <receta> puede contener dentro elementos <titulo>, <ingredientes> y <procedimiento>, que, a su vez, estarán definidos también en la DTD y podrán contener más elementos.

Siguiendo la definición de elemento anterior, este ejemplo de documento XML sería válido:

```
<receta>
<titulo>...</titulo>
<ingredientes>...</ingredientes>
<procedimiento>...</procedimiento>
</receta>
```

El siguiente ejemplo no sería válido:

```
<receta>
<parrafo>Esto es un párrafo</parrafo>
<titulo>...</titulo>
<ingredientes>...</ingredientes>
<procedimiento>...</procedimiento>
</receta>
```

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.

IMPORTANTE:

- No existe ninguna separación entre **"<!"** y **"ELEMENT"**, lo contrario implica un error de formación.

- 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.

2. ESPECIFICACIÓN DEL CONTENIDO

Las declaraciones de elementos anteriores están sin acabar, falta determinar cuál es el contenido válido para el tipo que se declara.

Es decir, un elemento puede:

- No tener contenido, ser vacío (EMPTY),
- Tener cualquier contenido (ANY),
- Una mezcla de valores (Mixed)
- Un conjunto de elementos hijos (chidren).

Ejemplo en HTML la etiqueta
</br> permite establecer un salto de línea. Si se declara como EMPTY, entonces se podría simplificar el salto de línea utilizando <br/>. <!ELEMENT saltoLinea EMPTY> <saltoLinea/>.

- Los parámetros de una definición de elemento son su nombre y una regla
- El formato de la definición pueden ser los siguientes:

  <!ELEMENT nombre ANY >

  Define un elemento cuyo contenido puede ser cualquiera. No se suelen utilizar, ya que conviene estructurar los documentos XML. Ejemplo: <!ELEMENT batiburrillo ANY>

  <!ELEMENT nombre EMPTY >

  Define un elemento sin contenido. Suele usarse para los atributos.

  Ejemplo: <!ELEMENT foto EMPTY>

  <!ELEMENT nombre (expresión regular)>

  <!ELEMENT nombre (expresión regular)repetición >

  - Definen elementos compuestos que contienen otros elementos y cuya estructura debe ajustarse a la expresión regular que se indica
  - La expresión regular debe estar formada por nombres de elementos y los metacaracteres: de agrupación "(" ")", de secuencia ",", de alternativa"|" y de repetición "+" [1..n], "?" [0..1], "*" [0..n]
  - Siempre es necesario un nivel externo de paréntesis
  - Estas formas de expresión no pueden contener el símbolo #PCDATA

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>

Puesto que el sentido de utilizar DTD es definir la estructura de un documento, ANY debe evitarse siempre que se pueda.

Por ejemplo 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/>
```

No se suele usar ya que la DTD sirve para lo contrario precisamente.

EL TIPO EMPTY

Para especificar que un elemento no puede tener contenido se le declara como EMPTY:

<!ELEMENT nombre_de_elemento EMPTY>

En XML, un elemento vacío puede no tener ningún contenido entre la etiqueta inicial y la etiqueta de cierre.

Normalmente un elemento sin contenido tendrá atributos que aporten algún tipo de información y que den sentido al elemento. Por ejemplo en HTML, 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.

```
<?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

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)*>

<!ELEMENT nombre (#PCDATA) >
<!ELEMENT nombre (#PCDATA | nombre | nombre ...)* >

- Definen elementos con lo que se denomina mixed content.
- Están formados por texto solo o entremezclado con otros elementos
- El nombre especial #PCDATA indica contenido de texto
- Debe aparecer siempre al principio de la expresión
- Puede contener sólo ese término o será una repetición de una alternativa simple.
- Ejemplos:

```
<!ELEMENT alumnos (alumno+)>
<!ELEMENT alumno (nb, ape1,ape2, foto, telefono*)>
<!ELEMENT nb (#PCDATA)>
<!ELEMENT ape1 (#PCDATA)>
<!ELEMENT ape2 (#PCDATA)>
<!ELEMENT foto  EMPTY>
<!ELEMENT telefono (#PCDATA)>
```

PCDATA (Parsed Character Data). Indica que entre la etiqueta de apertura y cierre de ese elemento, se almacenaran caracteres como texto y serán analizados por el parser.

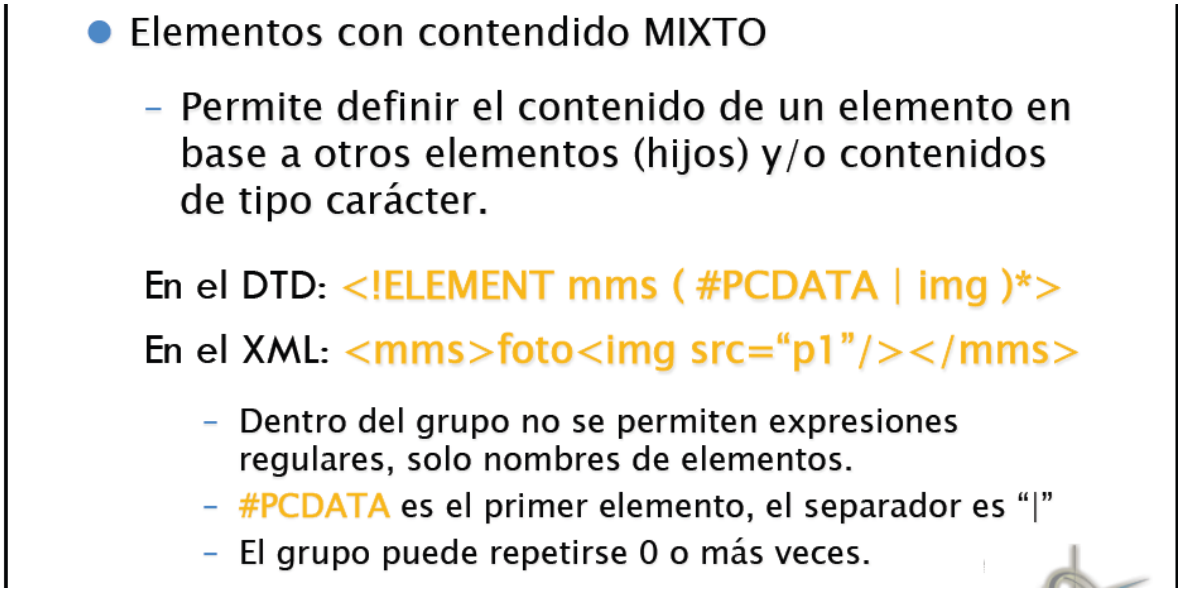
Los elementos declarados como de tipo **PCDATA no pueden contener** los caracteres siguientes:

- 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**.

Si se desean incluir estos caracteres, para que no se interpreten erróneamente, es necesario escaparlos. 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 </disco>
 <disco> Disco titulado "América", cuyo autor es... </disco>
 <disco>... </disco>
...
</listado_discos>
```



NOTA: No se puede usar el cuantificador + con un contenido mixto

ELEMENTOS

A partir del siguiente ejemplo se describirán veremos como se pueden agrupar los elementos:

```
<pelicula>

  <titulo>Brazil</titulo>
```

```
  <reparto>
    <interprete>Jonathan Pryce</interprete>
    <interprete>Robert De Niro</interprete>
    <interprete>Kim Greist</interprete>
  </reparto>
</pelicula>
```

¿Con qué DTD podríamos describirlo? Quizá con ésta:

```
<!ELEMENT pelicula (titulo, reparto)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT reparto (interprete, interprete, interprete)>
<!ELEMENT interprete (#PCDATA)>
```

En una DTD, los elementos a los que se hace referencia en las reglas pueden ser agrupados de diferentes formas.

Carácter	Significado
()	Conjunto de elementos
,	Separador de elementos. El orden debe respetarse.
|	Separador de alternativas: uno u otro

Veamos cada uno de ellos:

- **Paréntesis:** Nos permiten agrupar elementos. Podemos separar estos elementos entre sí con caracteres de agrupamiento. En el ejemplo anterior los hemos utilizado para agrupar los elementos que forman una película y también los que componen el reparto.
- **Coma**: Todos los elementos que separa deben aparecer, y además en el orden indicado. Es el caso del reparto, por ejemplo.
- **Barra**: Los elementos que separa son alternativas, uno u otro debe aparecer.

Para incluir declaraciones que indiquen que un elemento contiene elementos:

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

Se puede mezclar también con PCDATA:

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

Un tipo elemento puede contener otros elementos hijos. En este caso no podrá contener ningún texto sino tan solo elementos separados, opcionalmente, por espacios en blanco.

EJEMPLO

El elemento raíz XXX debe contener únicamente un elemento AAA seguido de otro elemento BBB. Los elementos AAA y BBB pueden contener texto pero no otros elementos.

```
<!ELEMENT XXX (AAA , BBB)>
<!ELEMENT AAA (#PCDATA)>
<!ELEMENT BBB (#PCDATA)>
```

Un documento válido que contiene algún texto

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX>
  <AAA>Comienzo</AAA>
  <BBB>Fin</BBB>
</XXX>
```

Este documento también es válido

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX> <AAA/> <BBB/> </XXX>
```

Ejemplo:

```
<?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>
                La presentación del disco es la siguiente
                <título>Maravillas</título>
                <autor>Amaya Diosdado</autor>
                <canción>Ave María</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 entremezclado 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.

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)>

De esta forma indicamos que debe haber cada uno de los elementos que aparecen y en ese orden.
EJEMPLO:

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

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

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

<!DOCTYPE persona [

  <!ELEMENT persona (nombre, apellidos, edad)>

  <!ELEMENT nombre (#PCDATA)>

  <!ELEMENT apellidos (#PCDATA)>

  <!ELEMENT edad (#PCDATA)>

]>

<persona>

   <nombre>Antonio</nombre>

   <apellidos>Pérez</apellidos>

   <edad>35</edad>

</persona>
```

Indica que el elemento contendrá la lista de elementos indicada, la cual deberá estar en el mismo orden de la secuencia. Es decir en el ejemplo los ***apellidos*** no se podrían poner delante del ***nombre***

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 [|]. En el 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)>

<!ELEMENT aviso (parrafo | grafico)> Uno u otro.

La barra vertical "|"indica una opción. Es decir, <aviso> puede contener o bien un <parrafo> o bien un <grafico>.

Con el carácter [|] se puede seleccionar uno de entre varios elementos.

Veamos la siguiente DTD:

<!ELEMENT película (titulo | reparto)>

<! ELEMENT titulo (#PCDATA)>

<!ELEMENT reparto (interprete, interprete, interprete)>

<!ELEMENT interprete (#PCDATA)>

La primera línea nos dice que el elemento película puede estar compuesto de un elemento titulo o de un elemento reparto. El siguiente documento según dicha DTD sería válido.

```
<pelicula>
   <titulo>Brazil</titulo>
 </pelicula>
```

Al igual que este otro:

```
<pelicula>
  <reparto>
    <interprete>Jonathan Pryce</interprete>
    <interprete>Robert De Niro</interprete>
    <interprete>Kim Greist</interprete>
  </reparto>
</pelicula>
```

EJEMPLO

El elemento raíz XXX debe contener un elemento AAA seguido de un elemento BBB. El elemento AAA tiene que contener un elemento CCC seguido de un elemento DDD. El elemento BBB tiene que contener bien un elemento CCC o bien un elemento DDD:

```
<!ELEMENT XXX (AAA , BBB)>
<!ELEMENT AAA (CCC , DDD)>
<!ELEMENT BBB (CCC | DDD)>
<!ELEMENT CCC (#PCDATA)>
<!ELEMENT DDD (#PCDATA)>
```

Un documento válido:

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX>
  <AAA>
    <CCC/> <DDD/>
  </AAA>
  <BBB>
    <CCC/>
  </BBB>
</XXX>
```

Otro documento válido:

```
<!DOCTYPE XXX SYSTEM "ejemplol.dtd">
<XXX>
  <AAA>
    <CCC/> <DDD/>
  </AAA>
  <BBB>
    <DDD/>
  </BBB>
</XXX>
```

EJEMPLO INTERCALADO DE NODOS Y TEXTO EN UN DOCUMENTO.

EL TEXTO PUEDE SER INTERCALADO CON ELEMENTOS.

El elemento AAA puede contener o bien BBB o CCC. Por otro lado el elemento BBB puede contener cualquier combinación de texto y elementos CCC.:

```
<!ELEMENT XXX (AAA+ , BBB+)>
<!ELEMENT AAA (BBB | CCC )>
<!ELEMENT BBB (#PCDATA | CCC )*>
<!ELEMENT CCC (#PCDATA)>
```

Un documento válido que explora varias posibilidades:

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX>
  <AAA>
    <CCC>Exactamente un elemento.</CCC>
  </AAA>
  <AAA>
    <BBB>
      <CCC/>
```

```
        <CCC/>
        <CCC/>
      </BBB>
    </AAA>
    <BBB/>
    <BBB>
      Esta es <CCC/> una combinacion <CCC/> de <CCC> elementos CCC </CCC> y texto <CCC/>.
    </BBB>
    <BBB>
      Sólo texto.
    </BBB>
  </XXX>
```

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))>

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

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

Por supuesto puede haber combinaciones, si tenemos un documento DTD llamado ***coordenada.dtd*** con este contenido:

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

<!ELEMENT coordenada ((longitud, latitud) | coordUniversal)>

<!ELEMENT longitud (#PCDATA)>

<!ELEMENT latitud (#PCDATA)>

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

Sería válido este documento:

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

<!DOCTYPE coordenada SYSTEM "coordenada.dtd">

<coordenada>

   <longitud>234</longitud>

   <latitud>-23</latitud>

</coordenada>
```

Sería válido también:

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

 <!DOCTYPE coordenada SYSTEM "coordenada.dtd">

<coordenada>

    <coordUniversal>1232332</coordUniversal>

</coordenada>
```

Se puede mezclar también con PCDATA:

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

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:

- **a+** - Una o más ocurrencias de a
  <!ELEMENT libro (capitulo)+>
- **a*** - Cero o más ocurrencias de a
  <!ELEMENT lista (objeto)*>
- **a?** - a o nada
  <!ELEMENT mesa (plato)?>
- **a, b** - a seguido de b
  <!ELEMENT resta (op1, op2)>
- **a | b** - a o b pero no ambos
  <!ELEMENT precio (euros | pesetas)>
- **(expression)** - expresiones que pueden ser tomadas como una sola
  <!ELEMENT capitulo (introduccion, (parrafo| comentario | nota)*, seccion*)>

- **El carácter [?] indica opción, el elemento o grupo se puede repetir cero o una vez.**

EJEMPLO

Si el nombre de un elemento en la DTD está seguido por un signo de interrogación [?], este elemento puede aparecer ninguna o una vez.

El elemento raíz XXX puede contener un elemento AAA seguido de exactamente un elemento BBB. El elemento BBB tiene que estar siempre presente:

```
<!ELEMENT XXX (AAA? , BBB)>
<!ELEMENT AAA (#PCDATA)>
<!ELEMENT BBB (#PCDATA)>
```

Un documento válido:

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX> <AAA/> <BBB/> </XXX>
```

El elemento AAA no es obligatorio:

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX> <BBB/> </XXX>
```

- **El carácter [+] indica una o más repeticiones. Asegura una ocurrencia como mínimo.**

EJEMPLO

Si el nombre de un elemento en una DTD está seguido por el carácter más [+], este elemento tiene que aparecer una o más veces.

El elemento raíz XXX debe contener uno o más elementos AAA seguidos de exactamente un elemento BBB. El elemento BBB tiene que estar siempre presente.

```
<!ELEMENT XXX (AAA+ , BBB)>
<!ELEMENT AAA (#PCDATA)>
<!ELEMENT BBB (#PCDATA)>
```

Un documento válido

```
<!DOCTYPE XXX SYSTEM "ejemplol.dtd">
<XXX> <AAA/> <BBB/> </XXX>
```

Pueden aparecer varios elementos AAA en el documento

```
<!DOCTYPE XXX SYSTEM "tutorial.dtd">
<XXX> <AAA/> <AAA/> <AAA/> <AAA/> <AAA/>
<AAA/> <AAA/> <BBB/> </XXX>
```

- **El carácter [*] indica cero o más repeticiones.**

EJEMPLO

El elemento raíz XXX puede contener ninguno, uno o varios elementos AAA seguido de exactamente un elemento BBB. El elemento BBB tiene que estar siempre presente.

```
<!ELEMENT XXX (AAA* , BBB)>
<!ELEMENT AAA (#PCDATA)>
<!ELEMENT BBB (#PCDATA)>
```

Un documento válido

```
<!DOCTYPE XXX SYSTEM "tutorial.dtd">
<XXX> <AAA/> <BBB/> </XXX>
```

Otro documento válido. El elemento AAA no es obligatorio

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX> <BBB/> </XXX>
```

Más de un elemento AAA puede aparecer dentro del documento

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX> <AAA/> <AAA/> <AAA/> <AAA/> <AAA/> <AAA/> <AAA/> <BBB/> </XXX>
```

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.

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

En este caso, <aviso> puede tener <titulo> o no (pero sólo uno), y puede tener cero o más conjuntos <parrafo><grafico>, <parrafo><parrafo><grafico>, etc.

EJEMPLO DE COMBINACIÓN DE LOS OPERADORES +, * Y?

ESTE EJEMPLO USA UNA COMBINACIÓN DE [+ * ?]

El elemento raíz XXX puede contener un elemento AAA seguido de uno o más elementos BBB. El elemento AAA puede contener un elemento CCC y varios elementos DDD. El elemento BBB tiene que contener, exactamente, un elemento CCC y un elemento DDD:

```
<!ELEMENT XXX (AAA? , BBB+)>
<!ELEMENT AAA (CCC? , DDD*)>
<!ELEMENT BBB (CCC , DDD)>
<!ELEMENT CCC (#PCDATA)>
<!ELEMENT DDD (#PCDATA)>
```

Un documento válido:

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX>
```

```
        <AAA>
          <CCC/><DDD/>
        </AAA>
        <BBB>
          <CCC/><DDD/>
        </BBB>
      </XXX>
```

Los elementos en AAA no son obligatorios:

```
      <!DOCTYPE XXX SYSTEM "ejemplo.dtd">
      <XXX>
        <AAA/>
        <BBB>
          <CCC/><DDD/>
        </BBB>
      </XXX>
```

El elemento AAA puede ser omitido:

```
      <!DOCTYPE XXX SYSTEM "ejemplo.dtd">
      <XXX>
        <BBB>
          <CCC/><DDD/>
        </BBB>
      </XXX>
```

EJEMPLO:

```
<?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>
```

Los documentos XML de tipo listado_discos se componen de elementos de tipo disco (cero o más). Cada elemento de tipo disco está formado 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.

EJERCICIOS:

Dado el siguiente documento XML, crea una DTD que lo haga válido.

```
<iddispositivo>
<model>320</model>
<versionfirware>V0.640</versionfirware>
<serial>CN125678FB</serial>
<indicacion>#Hewlett-Packard#Cam320#CN125678FB</indicacion>
<language>en</language>
</iddispositivo>
```

ATRIBUTOS

1. DECLARACIONES DE LISTAS DE ATRIBUTOS

```
<!ATTLIST elemento
        nombre tipo tratamiento_por_defecto
        nombre tipo tratamiento_por_defecto
        ...
>
```

- elemento: es el nombre del elemento al que corresponden los atributos
- nombre: es el nombre del atributo
- tipo: CDATA, (valor | valor | ...), ID, IDREF, IDREFS, NMTOKEN, NMTOKENS
- tratamiento_por_defecto: #REQUIRED, #IMPLIED, #FIXED valor_por_defecto, valor_por_defecto

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.

La declaración de lista de atributos se indica mediante:

<!ATTLIST, 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.

<!ATTLIST nombre_de_elemento nombre_de_atributo tipo 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
- La de una declaración de varios atributos conjuntamente:

<!ATTLIST nombre_de_elemento nombre_de_atributo tipo declaración_por_defecto

...
nombre_de_atributo tipo declaración_por_defecto
>

EJEMPLO:

```
<!ELEMENT comprador EMPTY>
<!ATTLIST comprador apellido CDATA
            nombre CDATA fecha CDATA
              compra CDATA
            precio CDATA
            identificador CDATA>

<comprador apellido="Martinez" nombre="Manuel"
    fecha="21/03/2011" compra="Manzanas"
    precio="3" identificador="T225" />
```

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

- ¡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.**
- **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*).**
- **Los valores de los atributos por defecto deben delimitarse por comillas dobles o simples**.
- 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.

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.

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.

2. TIPOS DE ATRIBUTOS

El tipo en cada declaración de atributos especifica cómo puede ser el contenido del atributo limitando los valores que puede tomar.

- tipo: CDATA, (valor | valor | ...), ID, IDREF, IDREFS, NMTOKEN, NMTOKENS

Los tipos están predefinidos y son los siguientes:

- **Tipos cadena (CDATA),**
- **Tipos enumerados**
- **Tipos específicos (ENTITY, ENTITIES , ID, IDREF, IDREFS, NMTOKEN y NMTOKENS)**

A. 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.

- Un valor del tipo CDATA corresponde a un valor de texto, en general

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

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

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

Ejemplo en el que definimos un único atributo *lenguaje* (cuyo valor por defecto es "Español") al elemento *mensaje*:

```
<!ELEMENT mensaje (#PCDATA)>
<!ATTLIST mensaje lenguaje CDATA "Español">
< mensaje lenguaje ="Español">
        ¡Hola!
</ mensaje>
```

<!ATTLIST apartamento tipo CDATA #IMPLIED>

EJERCICIO: DECLARA UN ATRIBUTO PARA EL ELEMENTO CIRCULO CUYO VALOR SEA 40PX

B. TIPOS ENUMERADOS

- Un valor del tipo enumerado (valor | valor | ...) especifica uno entre varios posibles nmtoken

Dos tipos:

- **Tipo enumerado**: los atributos ENUMERATION ofrecen una lista de posibles valores que puede tomar el atributo, todos estos valores son nombres de tipo ***token*.**
- **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.

  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 [\].

<!ATTLIST apartamento tipo (ubicación | venta) #IMPLIED>

EJEMPLO: CLINICA.XML

```
<!DOCTYPE clinica SYSTEM "clinica.dtd">
 <clinica nombre="vista" direccion="ssssdsdsl">
 <cita fecha="10/10/2010">
      <medico nif="d16345665465" nombre="pepe" apellidos="moreno">
            <especialidad>traumatologia</especialidad>
      </medico>
      <paciente nif="d00000001A" nombre="Drenthe">
            <diagnostico>ulcera de estómago</diagnostico>
       </paciente>
   </cita>
 <cita fecha="12/10/2010">
   <medico nif="d5258020000" nombre="luis" apellidos="moreno">
            <especialidad>traumatologia</especialidad>
   </medico>
   <paciente nif="d00000002A" nombre="Mujer de Drenthe">
            <diagnostico>parto</diagnostico>
  </paciente>
```

```
</cita>
</clinica>
```

CLINICA.DTD

```
<!ELEMENT clinica (cita+)>
<!ELEMENT cita (medico, paciente)>
<!ELEMENT medico (especialidad+)>
<!ELEMENT paciente (diagnostico*)>
<!ELEMENT diagnostico (#PCDATA)>
<!ELEMENT especialidad (#PCDATA)>
<!ATTLIST clinica
            nombre CDATA #REQUIRED
            direccion CDATA #IMPLIED
>
<!ATTLIST medico
            nif ID #REQUIRED
           nombre CDATA #REQUIRED
           apellidos CDATA #IMPLIED
           sexo (hombre | mujer) #IMPLIED
>
<!ATTLIST paciente
            nif ID #REQUIRED
            nombre CDATA #REQUIRED
           apellidos CDATA #IMPLIED
            sexo (hombre | mujer) #IMPLIED
>
<!ATTLIST cita
           fecha CDATA #REQUIRED
>
```

C. ATRIBUTOS TIPO *ESPECIFICOS (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 carácter 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 [:], siendo útiles para describir números.**

<! ELEMENT Nombre (#PCDATA) >
<!ATTLIST Nombre contraseña NMTOKEN #REQUIRED>

<Nombre contraseña=".-92AB .67-35·">Ramon xxxx</Nombre>

EJEMPLO

```
<!ELEMENT nombre_estudiante (#PCDATA)>
  <!ATTLIST nombre_estudiante num_estudiante NMTOKEN #REQUIRED>
  < nombre_estudiante num_estudiante ="12345">
          Albert Einstein
  </ nombre_estudiante >
```

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.
    - ID, permite declarar un atributo con un nombre identificativo *(IDentification)* particular para ser utilizado en cualquier momento como referencia nominal. Un tipo de elemento puede tener sólo un atributo ID.
    - **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.

ID

- **Identificador único del atributo para cada elemento**
- **De tipo #REQUIRED o #IMPLIED**
- **No debe aparecer más de una sola vez en el documento XML**
- **El primer carácter debe ser una letra o '_', o ':'**
- **Los demás caracteres no pueden ser espacios.**

EJEMPLO:

```
<!ELEMENT nombre_estudiante (#PCDATA)>
 <!ATTLIST nombre_estudiante estudiante _id ID #REQUIRED>
<nombre_estudiante estudiante _id="E50716">
  Pepito Grillo
</ nombre_estudiante>
```

▪ Un valor del tipo ID es un nombre que debe ser único en todo el documento, y que sirve para identificar el elemento

- **IDREF, IDREFS**: estos tipos de atributos se utilizan para **referenciar elementos identificados con un atributo ID**, (el valor de estos atributos contiene el valor de otro atributo ID).
    - 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.

EJEMPLO:

```
<?xml version="1.0" standalone="yes"?>
  <!DOCTYPE grupo [
   <!ELEMENT grupo (nombre_estudiante)*>
    <!ELEMENT nombre_estudiante (#PCDATA)>
    <!ATTLIST nombre_estudiante estudiante _id ID #REQUIRED
     suplente IDREF #IMPLIED>
    ]>

   <grupo>
   <nombre_estudiante estudiante _id ="E1123">Ana
Garcia</nombre_estudiante>
   <nombre_estudiante estudiante _id ="E1124">Isabel
Lopez</nombre_estudiante>
   < nombre_estudiante estudiante _id ="E1125"
                suplente="E1123">Maria Sanchez</nombre_estudiante >
   </grupo>
```

▪ Un valor del tipo IDREF es un nombre que debe aparecer como valor de un atributo ID en algún elemento del documento

▪ Un valor del tipo IDREFS es una lista de IDREF separados por espacio en blanco

EJEMPLO:

La idea es poder relacionar elementos a través de atributos de tipo ID e IDREF. Ejemplo de uso:

```
<?xml version="1.0" encoding="UTF-8"?>
 <!-- Archivo directorio.dtd -->
<!ELEMENT directorio (persona)+ >
<!ELEMENT persona (#PCDATA) >
 <!ATTLIST persona id ID #REQUIRED
                   madre IDREF #IMPLIED
                   padre IDREF #IMPLIED>
<?xml version="1.0" encoding="UTF-8"?>
<!-- Archivo directorio1.xml-->
<!DOCTYPE directorio SYSTEM "directorio.dtd">
 <directorio>
        <persona id="p1">Pedro</persona>
    <persona id="p2">Marisa</persona>
    <persona id="p3" madre="p2" padre="p1">Carmen</persona>
</directorio>
```

Carmen es la hija de Pedro y Marisa, según el código anterior.

- **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 apartado), 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.

- Un valor del tipo NMTOKEN es similar a CDATA. Solo admite letras, nº, puntos, guiones, subrayados y los dos puntos
- Un valor del tipo NMTOKENS es una lista de NMTOKEN separados por espacio en blanco

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.

Ejemplo:

El tipo ID permite que un tipo determinado tenga un nombre único que podrá ser referenciado por un atributo de otro elemento que sea de tipo IDREF. Por ejemplo, para implementar un sencillo sistema de hipervínculos en un documento:

```
<!ELEMENT enlace EMPTY>
<!ATTLIST enlace destino IDREF #REQUIRED>
<!ELEMENT capitulo (parrafo)*>
<!ATTLIST capitulo referencia ID #IMPLIED>
```

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

```
…
        <!ATTLIST perro nomid ID #REQUIRED>
        <!ATTLIST perro madre IDREF #IMPLIED>
        <!ATTLIST perro padre IDREF #IMPLIED>
        <!ATTLIST perro otrosnombres NMTOKENS "">
        <!ATTLIST perro fotos ENTITIES "">
…
```

En una tienda de animales 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.

EJEMPLO:

```
<!ELEMENT XXX (AAA+ , BBB+ , CCC+)>
<!ELEMENT AAA (#PCDATA)>
```

```
<!ELEMENT BBB (#PCDATA)>
<!ELEMENT CCC (#PCDATA)>
<!ATTLIST AAA id ID #REQUIRED>
<!ATTLIST BBB code ID #IMPLIED list NMTOKEN #IMPLIED>
<!ATTLIST CCC X ID #REQUIRED Y NMTOKEN #IMPLIED>
```

Todos los valores ID son únicos:

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX>
<AAA id="a1"/>
<AAA id="a2"/>
<AAA id="a3"/>
<BBB code="QWQ-123-14-6" list="14:5"/>
<CCC X="zero" Y="16" />
</XXX>
```

Los atributos list y Y son del tipo NMTOKEN no ID. Éstos pueden tener, por lo tanto, el mismo valor que los atributos ID o tener el mismo valor en varios elementos:

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX>
<AAA id="L12"/>
<BBB code="QW" list="L12"/>
<CCC X="x-0" Y="QW" />
<CCC X="x-1" Y="QW" />
</XXX>
```

Tipo de atributo IDREF y IDREFS.- El valor de un atributo IDREF tiene que corresponder con el valor de algún atributo ID del documento. El valor del atributo IDREFS puede contener varias referencias a elementos con atributos ID separados por espacios en blanco.

Ejemplo

Los atributos id y mark determinan inequívocamente su elemento. Los atributos ref hacen referencia a estos elementos:

```
<!ELEMENT XXX (AAA+ , BBB+, CCC+, DDD+)>
<!ELEMENT AAA (#PCDATA)>
<!ELEMENT BBB (#PCDATA)>
<!ELEMENT CCC (#PCDATA)>
<!ELEMENT DDD (#PCDATA)>
<!ATTLIST AAA mark ID #REQUIRED>
<!ATTLIST BBB id ID #REQUIRED>
<!ATTLIST CCC ref IDREF #REQUIRED>
<!ATTLIST DDD ref IDREFS #REQUIRED>
```

Todos los valores ID son únicos y todos los valores IDREF e IDREFS apuntan a elementos con IDs relevantes:

```
<!DOCTYPE XXX SYSTEM "ejemplo.dtd">
<XXX>
<AAA mark="a1"/>
<AAA mark="a2"/>
<AAA mark="a3"/>
<BBB id="b001" />
<CCC ref="a3" />
<DDD ref="a1 b001 a2"/>
</XXX>
```

3. 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**(obligatorio): 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.

```
<!ELEMENT img EMPTY>
 <!ATTLIST img
        alt CDATA #REQUIRED
        src CDATA #REQUIRED>
<img src="imagen.jpg" alt="imagen"/>
```

- **#IMPLIED**(optativo): 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.

```
<!ELEMENT img EMPTY>
<!ATTLIST img
            alt CDATA #REQUIRED
      src CDATA #REQUIRED
      width CDATA #IMPLIED
      height CDATA #IMPLIED>
 <img src="imagen.jpg" alt="imagen" width="500"/>
```

- **#FIXED(fijos)** "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.

<!ELEMENT direccion (#PCDATA)>
<!ATTLIST direccion ciudad CDATA #FIXED "Madrid">
< direccion ciudad="Madrid">Calle General Ricardos 111.</direccion>

- **Se puede indicar un valor predeterminado para el atributo**

  Además, existe la posibilidad de indicar una lista de posibles valores para el atributo, entre los que los usuarios de la DTD deben escoger.

Por ejemplo, si quisiésemos incluir un atributo con una calificación para la película, del 1 al 5, podríamos hacerlo de la siguiente forma:
<!ATTLIST película nota (1 | 2 | 3 | 4 | 5) "3">
Si no se indicase valor alguno para el atributo, el predeterminado es el 3. Así, este documento sería válido:

<pelicula titulo="Brasil"/>
Éste también lo sería:
<pelicula titulo="Brasil nota="5"/>
Sin embargo, éste no:
<pelicula titulo="Brasil" nota="7"/>
El documento XML completo, con la descripción del elemento película y sus atributos, es el siguiente:

```
<?xml version="1.0"?>
<!D0CTYPE pelicula [
<!ELEMENT pelicula EMPTY>
<!ATTLIST pelicula titulo CDATA "">
<!ATTLIST pelicula nota (1 | 2 | 3 | 4 | 5) "3">
] >
<pelicula titulo="Brasil" nota="5"/>
```

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).

Con el siguiente ejemplo podemos ver las declaraciones de los atributos 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.

EJEMPLO COMPLETO TIENDA DE ANIMALES

```
<?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)>

        <!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="1/1/2012" 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="30/12/2011" sexo="macho" ID="P050">

          Macho <raza>labrador</raza> con un pelaje fantástico

          y una gran fortaleza,...

        </perro>

        <perro fecha_nacimiento="19/7/2010" 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" (no puede 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, tienda_animales sólo pueden contener elementos de tipo perro. 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.

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.

EJERCICIO RESUMEN LIBRERÍA

. La librería posee una lista de libros ordenados por categorías y los libros se pueden describir como la asociación de los siguientes elementos: - autor;- título;- precio;- editor;- cantidad. Además, un libro debe asociarse a una categoría, identificada por un código. Una categoría posee un título y, a veces, un comentario.
Teniendo en cuenta esta información, vamos a ver cómo componer la DTD de este ejemplo.

Definición de la gramática de la lista de libros
Desde el elemento más global hasta los elementos terminales, este es el orden de descripción de los elementos:
- **Librería:** El elemento raíz de nuestro ejemplo, compuesto por una lista de productos (el elemento Libros) y el conjunto de categorías (el elemento Categorías);
    - **Libros:** lista de productos de la librería;
    - **Categorías**: lista de categorías de libros;
    - **Producto**: un libro;
    - **Categoría:** categoría de libro;
    - **Precio**: precio de un libro;
    - **Autor**: autor del libro;
    - **Titulo**: título del libro;
    - **Editor**: editor del libro;
    - **Cantidad:** número de productos disponibles a la venta.

Así pues, elemento por elemento, esta sería la gramática DTD de nuestra librería:

- El elemento librería:

```
<!ELEMENT LIBRERIA ( LIBROS, CATEGORIAS ) >
```

La librería se compone de libros (de productos) y de categorías. Los elementos Libros y producto:

```
<!ELEMENT LIBROS ( PRODUCTO+ ) >
<!ELEMENT PRODUCTO ( AUTOR | PRECIO | EDITOR | CANTIDAD |TITULO )* >
<!ATTLIST PRODUCTO CAT IDREF #REQUIRED >
```

Tenga en cuenta que los productos poseen el atributo CAT como referencia a una categoría asociada. La palabra clave #REQUIRED indica que esta información es obligatoria.

- Los elementos Categorías y Categoría:

```
<!ELEMENT CATEGORIAS ( #PCDATA | CATEGORIA )* >
<!ATTLIST CATEGORIAS DESC CDATA #REQUIRED >
<!ELEMENT CATEGORIA EMPTY >
<!ATTLIST CATEGORIA CODE ID #REQUIRED >
<!ATTLIST CATEGORIA DESC NMTOKEN #REQUIRED >
<!ATTLIST CATEGORIA NOTE CDATA #IMPLIED >
```

El elemento Categorías se compone de datos textuales y/o de categoría atributo obligatorio DESC puede contener una descripción. El elemento categoría tiene únicamente los atributos CODE (identificador de la categoría y DESC (descripción de la categoría). Estos dos atributos

son obligatorios. Sin embargo, la palabra clave #IMPLIED significa que el atributo NOTE (observación sobre la categoría) es facultativo.

- Los elementos Autor, Precio, Editor, Cantidad y Titulo:

```
<!ELEMENT AUTOR ( #PCDATA ) >
<!ELEMENT PRECIO ( #PCDATA ) >
<!ELEMENT EDITOR ( #PCDATA ) >
<¡ELEMENT CANTIDAD ( #PCDATA ) >
<!ELEMENT TITULO ( #PCDATA ) >
```

Estos cinco elementos son terminales. Uniendo estos distintos trozos de código obtendrá la DTD completa de nuestro ejemplo .Guarde este archivo, por ejemplo, con el nombre librería.dtd:

A continuación se muestra un ejemplo de archivo XML válido para esta gramática:

```
<?xml version="1.0" encoding="ISO-8859-l" standalone="no" ?>
<!DOCTYPE LIBRERIA SYSTEM "librería.dtd">
<LIBRERIA>
<LIBROS>
<PRODUCTO CAT="S">
<TITULO>Number, the Language of Science</TITULO>
<AUTOR>Danzig</AUTOR>
<PRECIO>5,95</PRECIO>
<CANTIDAD>3</CANTIDAD>
</PRODUCTO>
<PRODUCTO CAT="P">
<TITULO>Escritos de Morrison</TITULO>
<EDITOR>Christian Bourgois Editeur</EDITOR>
<AUTOR>Jim Morrison</AUTOR>
<PRECIO>43</PRECIO>
<CANTIDAD> 5</CANTIDAD>
</PRODUCTO>
<PRODUCTO CAT="I">
<TITULO>WI-FI 6 soluciones para su empresa</TITULO>
<AUTOR>Laurence Soyer</AUTOR>
<PRECIO>NC</PRECIO>
<CANTIDAD>15</CANTIDAD>
</PRODUCTO
<PRODUCTO CAT="R">
<TITULO>¡A la cama!</TITULO>
<AUTOR>David Baddiel</AUTOR>
<PRECIO>7,50</PRECIO>
<CANTIDAD>2</CANTIDAD>
</PRODUCTO>
</LIBROS>
<CATEGORIAS DESC="Las categorías">
<CATEGORIA CODE="S" DESC="Ciencia"/>
<CATEGORIA CODE="P" DESC="Poesía" NOTA="Edición Limitada"/>
<CATEGORIA CODE="I" DESC="Informática"/>
<CATEGORIA CODE="D" DESC="Derecho"/>
<CATEGORIA CODE="R" DESC="Novela"/>
</CATEGORIAS>
</LIBRERIA>
```

EJERCICIO

Dada la siguiente DTD, crea un documento XML que sea valido con respecto a ella:

```
<!ELEMENT noticias (noticia*)>
<!ELEMENT noticia (titulo, autor, url, texto?, foto?)>
    <!ATTLIST noticia dia CDATA #REQUIRED>
    <!ATTLIST noticia mes CDATA #REQUIRED>
    <!ATTLIST noticia anyo CDATA #REQUIRED>
<!ELEMENT titulo  (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT url  (#PCDATA)>
<!ATTLIST url direccion CDATA #REQUIRED>
<!ELEMENT texto (#PCDATA)>
<!ELEMENT foto (#PCDATA)>
```

ENTIDADES

- Las entidades son valores constantes a los que se puede hacer referencia mediante un nombre
- Hay dos clases de entidades: General entities (para ser usadas en el contenido del documento) y Parameter entities (para ser usadas en la DTD)
- Una general entity se define como:
  <!ENTITY nombre valor_de_sustitución >
  - El valor_de_sustitución puede ser un texto literal, que a su vez puede contener referencias a otras entidades
  - Se hace referencia a ella con &nombre; en el contenido del documento
- Una parameter entity se define como:
  <!ENTITY % nombre valor_de_sustitución >
  - El valor_de_sustitución puede ser un texto literal, que a su vez puede contener referencias a otras entidades
  - A la entidad se hace referencia con %nombre; en la DTD

Las entidades permiten definir las constantes de los documentos XML. Existen dos tipos de entidades: internas y externas.

- **Entidades internas**: pueden utilizarse únicamente dentro del documento en el que han sido declaradas.

Ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE tarjetavisita [<!ENTITY ca "Calle Caoba">]>
<tarjetavisita>
 <apellido>GRILLO</apellido>
```

```
<nombre>PEPITO</nombre>
<profesion>CARPINTERO</profesion>
<direccion>
 <numero>123</numero>
 <calle>&ca;</calle>
 <codigopostal>24342</codigopostal>
 <poblacion>PPPPPPPPPPP</poblacion>
</direccion>
</tarjetavisita>
```

Para poner el nombre de la entidad : **&nombreentidad;**

```
<?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>
```

- **Entidades externas** permiten vincular el documento XML a otro documento a través de una URL. Este documento puede ser de tipo XML o de cualquier otro. <!ENTITY otrodoc SYSTEM "http://localhost/docxml/otrodoc.xml>.

Cuando incluimos dentro del texto de un nodo una entidad, el analizador la sustituye por el valor con el que se corresponda. Todas las entidades están delimitadas por el ampersand (&) y el punto y coma (;), es decir, deben aparecer entre esos dos caracteres. Ya vimos las cinco entidades predeterminadas.

- **Entidades predeterminadas**

Nombre	**Carácter**	**Alias**
ampersand	&	&amp;
menor que	<	<
mayor que	>	>
comillas simples	‘	'
comillas dobles	“	"

EJEMPLO: documento XML:

```
<?xml version=”1.0”?>
<pelicula>Tango &amp; Cash</pelicula>
```

La figura muestra el aspecto que presenta ese documento en un cliente Web. Como puede verse, la entidad ha sido sustituida por el valor correspondiente.

No tenemos que ceñirnos a las **entidades predeterminadas**, podemos crear las nuestras propias. Gracias a las entidades podemos realizar algunas operaciones que pueden sernos de gran utilidad, como:

1. La inserción de caracteres especiales.
2. La simplificación de textos largos y repetitivos.
3. La división modular de los documentos.

CARACTERES ESPECIALES

Estas entidades se declaran de forma similar a los elementos o a los atributo

<!ENTITY nombre sustitución>

Suponga que uno de los valores que vamos a almacenar en los documentos XML de nuestra filmoteca será el precio en euros de una copia en formato DVD. De entre todas las opciones disponibles para incluir el símbolo del euro en un documento XML, ésta es una de las posibles:

```
<precio>19,95 &#x20ac;</precio>
```

Con esa secuencia de números, difícil de recordar, estamos indicando el código del carácter que queremos introducir. Podemos simplificar un poco la tarea definiendo una entidad, de la siguiente forma:

```
<!ENTITY euro "&#x2 0ac;">
```

A partir de ese momento, podremos incluir el símbolo del euro así:

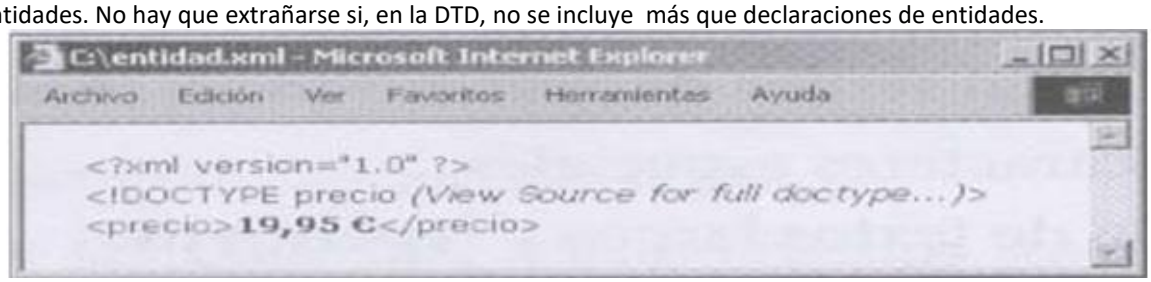
```
<precio>19,95 &euro;</precio>
```

El documento completo sería el siguiente:

```
<?xml version="1.0"?>
<!DOCTYPE precio [
<!ENTITY euro "&#x20ac;">
]>
<precio>19,95 &euro; </precio>
```

La figura muestra el aspecto de este documento XML en un cliente Web.

Nota: Estos documentos no son válidos, sólo se pretende demostrar el funcionamiento de las entidades. No hay que extrañarse si, en la DTD, no se incluye más que declaraciones de entidades.



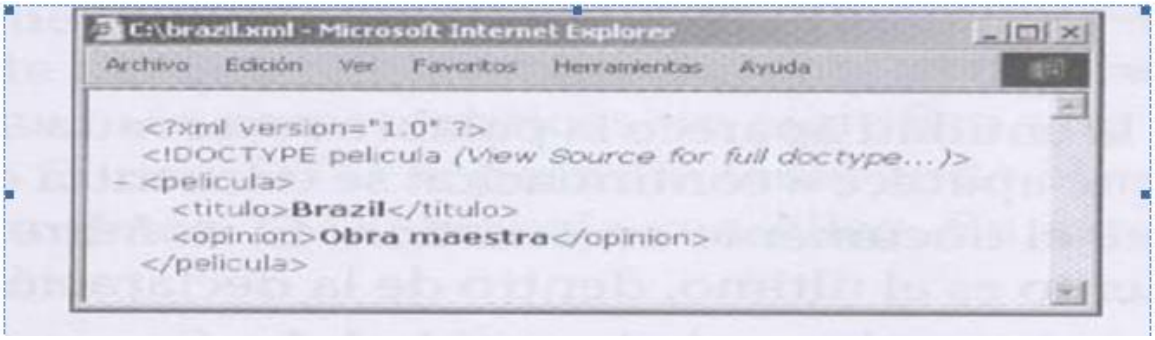
TEXTOS QUE SE REPITEN

La misma técnica utilizada en la sección anterior la podemos utilizar para hacer nuestro trabajo un poco más simple, ahorrándonos texto que escribir. Por ejemplo, suponga que queremos asignar a cada película una frase de recomendación, como "Obra maestra" o "Bodrio infumable". Sólo tenemos que seleccionar un atajo adecuado y crear las entidades correspondientes:

<!ENTITY om "Obra maestra">
<!ENTITY bi "Bodrio infumable">
Un documento que pueda utilizar esas entidades podría ser parecido al siguiente:

```
<?xml versión»"1.0"?>
<!DOCTYPE pelicula [
<!ENTITY om "Obra maestra">
<!ENTITY bi "Bodrio infumable">
]>
<pelicula>
<titulo>Brazil</titulo>
<opinion>&om;</opinion>
</pelicula>
```

El resultado de ver ese documento XML en un cliente Web es similar al que ofrece la figura.



DIVISIÓN MODULAR

Las entidades nos permiten mucho más que ahorrar tiempo de escritura. De forma similar a como definimos una entidad para un determinado carácter o una cadena de texto, podemos incluir el contenido de un documento XML dentro de otro. En el ejemplo anterior añadíamos una opinión de carácter personal sobre una determinada película. Para dejar claro que esas opiniones están vertidas por los autores del documento de cada película, queremos añadir una serie de notas al final de todos los documentos. Estas notas (por ahora sólo Una) están recogidas en el siguiente documento XML:

```
<?xml version="1 . 0"?>
<notas>
        <nota>
        Las opiniones son responsabilidad de sus autores.
        </nota>
</notas>
```

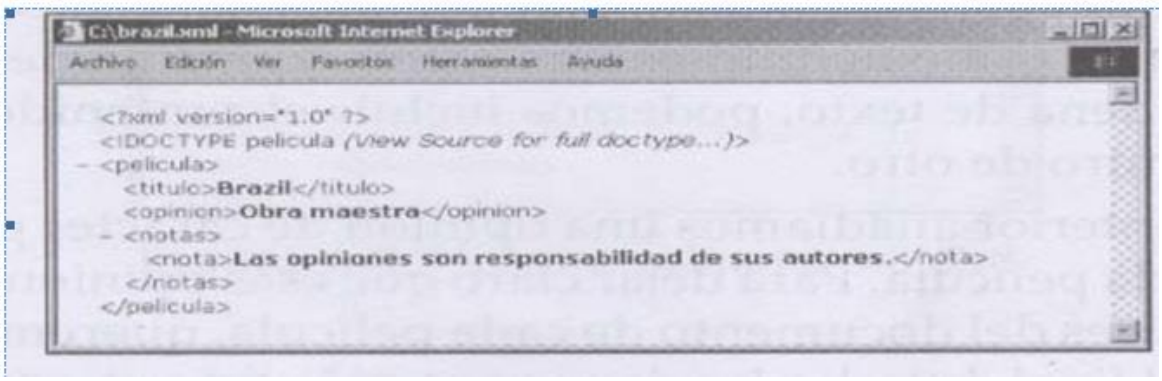
Podemos guardar este documento XML con el nombre notas.xml. Lo usaremos ahora mismo. Incluir esas notas al final del documento de una película es tan sencillo como esto:

```
<?xml versión="1.0" ?>
<!DOCTYPE película [
<!ENTITY om "Obra maestra">
<!ENTITY bi "Bodrio infumable">
<!ENTITY notas SYSTEM "notas.xml">
]>
<pelicula>
<titulo>Brazil</titulo
<opinion>&om;</opinion>
&notas;
</pelicula>
```

El documento anterior es básicamente igual que aquel en el que añadimos una opinión sobre la película, con dos salvedades. La primera, que hemos añadido una nueva definición de entidad: **<!ENTITY notas SYSTEM "notas.xml">**

Tras el nombre de la entidad aparece la palabra reservada **SYSTEM,** que indica que el recurso que aparece a continuación se encuentra en el mismo sistema de archivos que el documento en cuestión. El nombre del archivo en el que está el documento es el último, dentro de la declaración de la entidad. La segunda diferencia es el uso de la entidad.

La figura muestra el resultado de cargar el documento anterior en un cliente Web. Como puede comprobarse, **&notas**,- ha sido reemplazado por el contenido del documento XML indicado en la declaración de la entidad.



EJEMPLO2 CON ELEMENT

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

EJEMPLO3 CON ATRIBUTOS

```
<?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>
<!--Probar con <ejemplo_entidades></ejemplo_entidades> para que inserte el valor
por defecto-->
```

EJEMPLO4: INSTRUCCIONES DE PROCESO IP

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet type="text/css"href="ipv1.css" ?>
<!DOCTYPE Poema [
        <!ELEMENT Poema (Título , Cuerpo)>
        <!ELEMENT Título (#PCDATA)>
        <!ELEMENT Cuerpo (#PCDATA)>
]>
<Poema >
                <Título>El Gallo  Popular </Título>
          <Cuerpo>  Gallo montero,
                    gentil caballero,
                    vestido de plumas,
                     como un coracero.
    </Cuerpo>
</Poema>
```

IPV1.CSS

```
Titulo {display:block;  }
Cuerpo {display:block; color:blue;}
```