Capítulo 2

DTD: Definiendo la estructura del documento XML

En este capítulo aprenderá:

- Qué es un documento DTD.
- A crear documentos DTD.

Los documentos DTD

Siguiendo con el ejemplo del documento XML del catálogo de libros, si estuviésemos en el lugar de las librerías, ¿verdad que no nos gustaría recibir dicho documento con errores? Si lo recibiésemos con errores, no nos serviría, no podríamos actualizar nuestro propio listado de libros, tendríamos que ponernos en contacto con la editorial para informarles de lo ocurrido y perderíamos mucho tiempo. Aunque los documentos XML tienes una estructura sencilla, esto no evita que se comentan errores, así que necesitamos algún mecanismo que nos permita definir cómo debe ser la estructura del documento XML y así poder comparar si está correcta.
La idea es la siguiente. En primer lugar, vamos a crear un nuevo documento donde definiremos cómo debe ser la estructura del documento XML. A este nuevo documento nos referiremos como documento DTD (Definición de Tipos de Documento, Document Type Definition). En segundo lugar, en el documento XML indicaremos que éste debe mantener el formato especificado en el documento DTD que indiquemos.
Posteriormente, siempre podremos comparar el contenido del documento XML con la estructura definida en el documento DTD para saber si está correcto. A esta tarea la denominamos validación del documento XML o validar el documento XML, que trataremos cerca del final de este capítulo.

Creando nuestro primer documento DTD

Supongamos que tenemos una empresa de productos de alimentación que vende a cadenas de hoteles y grandes superficies. La información de algunos productos en venta se hallan en un documento XML como el siguiente, así podemos compartir fácilmente los datos de los productos con nuestros clientes.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-001.xml -->
<listadeproductos>
   <producto>
      <nombre>Galletas integrales</nombre>
      <codigo>8402973812000</codigo>
      <precio>6.45</precio>
   </producto>
   <producto>
      <nombre>Cereales con chocolate</nombre>
      <codigo>6482947100032</codigo>
      <precio>4.75</precio>
   </producto>
</listadeproductos>
```

Pero, ¿qué ocurriría si este documento lo hubiésemos escrito con los errores que vemos a continuación?

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-002.xml -->
<listadeproductos>
   <producto>
      <nombre>Galletas integrales</nombre>
      <codigo>8402973812000</codigo>
   </producto>
   <productos>
      <nombre>Cereales con chocolate</nombre>
      <codigo>6482947100032</codigo>
      <precio>4.75</precio>
   </productos>
</listadeproductos>
```

Observemos que el primer producto no tiene el elemento `<precio></precio>` y que el segundo elemento `<productos></productos>` está en plural, mientras que el anterior está en singular. Aunque este documento XML está escrito correctamente, ¿lo productos pueden no tener precio? ¿el elemento productos se escribe en singular o en plural?
Para evitar problemas y dudas, debemos asegurarnos que este documento siempre tiene el formato que nosotros deseamos. Para ello, vamos a crear un documento DTD donde definamos la estructura.
Empezaremos definiendo cada elemento del documento XML siguiendo la siguiente sintaxis:

```
<!ELEMENT nombre-elemento (expresión-regular)>
```

Donde `nombre-elemento` es el nombre del elemento que hay que definir y `expresión-regular` describe el elemento.

Definiendo elementos anidados

Si observamos el documento XML, el elemento principal `listadeproductos` contiene a varios elementos `producto`. Esto se define de la forma:

```
<!ELEMENT listadeproductos (producto)>
```

Y el elemento `producto` contiene otros tres elementos: `nombre`, `codigo` y `precio`. Esto lo definimos así:

```
<!ELEMENT producto (nombre, codigo, precio)>
```

Hasta el momento, nuestro documento DTD tiene el siguiente contenido:

```
<!ELEMENT listadeproductos (producto)>
<!ELEMENT producto (nombre, codigo, precio)>
```

Así estamos definiendo que el documento XML tiene un elemento principal llamado `listadeproductos` que contiene otros elementos llamados `producto`. A su vez, los elementos `producto` contienen otros tres elementos que aparecen por ese mismo orden: `nombre`, `codigo` y `precio`.
Es muy importante tener en cuenta que en el DTD estamos definiendo exactamente el nombre de los elementos y el orden en el que deben aparecer en el documento XML.

Definiendo el tipo de elementos

Nuestro documento XML del ejemplo también contiene una serie de elementos con unos valores: `nombre`, `codigo` y `precio`. Para especificar que estos elementos ya no contienen más elementos, sino que contienen datos o valores, los definiremos como #PCDATA (*Parsed Character Data*, Datos Carácter Analizados) de la siguiente manera:

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

Nuestro documento DTD tendría ahora el siguiente contenido:

```
<!ELEMENT listadeproductos (producto)>
<!ELEMENT producto (nombre, codigo, precio)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
```

Recordemos el significado. Así estamos definiendo que el documento XML tiene un elemento principal llamado `listadeproductos` que contiene otros elementos llamados `producto`.
A su vez, los elementos `producto` contienen otros tres elementos que aparecen por ese mismo orden: `nombre`, `codigo` y `precio`. Estos tres elementos (`nombre`, `codigo` y `precio`) contienen datos.

Definiendo el número de veces que puede aparecer un elemento

La definición que tenemos hasta ahora realmente dice que sólo debe existir un único elemento `listadeproductos`, lo cual es correcto, pues es el elemento raíz. Pero la definición actual también especifica que sólo contiene un único elemento `producto`. Sin embargo, nosotros deseamos que `listadeproductos` contenga varios. La cuestión es que si definimos los elementos

simplemente como lo hemos hecho hasta ahora, estaremos indicando que cada elemento debe aparecer una única vez, ni más veces, ni menos veces. Y ¿cómo solucionamos esto? Pues utilizando una serie de operadores.

Operador +

El operador + permite definir que en el documento XML puede aparecer una o más veces un elemento, debiendo existir siempre al menos uno. Debe situarse después del nombre del elemento cuando se define la anidación de elementos.
Así que ya tenemos solución para el problema que acabábamos de comentar. El documento DTD quedaría de la siguiente forma:

```
<!ELEMENT listadeproductos (producto+)>
<!ELEMENT producto (nombre, codigo, precio)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
```

Observemos cómo en la primera línea hemos incluido el operador + tras el nombre del elemento `producto`. Así hemos definido que el documento XML debe tener un elemento principal llamado `listadeproductos` que a su vez debe contener uno o más elementos `producto`.

Operador *

Si por alguna circunstancia consideramos que un elemento puede aparecer cero o más veces, por lo que puede ser que no aparezca en el documento XML, utilizaremos el operador * de igual forma, escribiéndolo después del nombre del elemento cuando se define la anidación de elementos.
Supongamos que el documento XML del ejemplo que seguimos pudiese no contener ningún elemento `producto`, entonces, el documento DTD sería:

```
<!ELEMENT listadeproductos (producto*)>
<!ELEMENT producto (nombre, codigo, precio)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
```

Operador ?

Anteriormente planteamos el caso de que se omitiese por error el elemento `precio` en uno de los productos. Pero ¿y si realmente algunos productos pudiesen no tener precio porque aún no se ha fijado y debe consultarse a la

empresa? En casos como este podemos utilizar el operador ? para indicar que un elemento es opcional, por lo que puede aparacer una vez o ninguna. Escribiremos este operador después del nombre del elemento cuando se define la anidación.
Siguiendo este ejemplo vemos el resultado en el documento DTD:

```
<!ELEMENT listadeproductos (producto+)>
<!ELEMENT producto (nombre, codigo, precio?)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
```

Observemos cómo en la primera línea hemos incluido nuevamente el operador + tras el nombre del elemento `producto` para definir que el documento XML debe tener un elemento principal llamado `listadeproductos` que a su vez debe contener uno o más elementos `producto`. En la segunda línea, hemos definido que el elemento `producto` contiene otros tres elementos: `nombre`, `codigo` y `precio`. En este caso, mediante el operador ?, hemos definido que el elemento `precio` puede aparecer una vez o ninguna para un determinado producto dentro del documento XML.

Operador |

Podemos contemplar más casos, como el que nos ofrece el operador de barra vertical (|), que permite definir que en el documento XML puede aparecer sólo uno de varios elementos separados por este operador, debiendo aparecer obligatoriamente sólo uno de ellos.
Supongamos que cada producto puede contener un elemento `precio` o un elemento `preciooferta`, pero sólo debe existir uno de ellos en cada producto. En este caso, el documento DTD será así:

```
<!ELEMENT listadeproductos (producto+)>
<!ELEMENT producto (nombre, codigo, precio | preciooferta)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT preciooferta (#PCDATA)>
```

Observemos que se ha incluido una línea más al final para definir el tipo de elemento `preciooferta`. El siguiente documento XML muestra un ejemplo de este caso:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-003.xml -->
<listadeproductos>
   <producto>
      <nombre>Galletas integrales</nombre>
      <codigo>8402973812000</codigo>
```

```
      <precio>6.45</precio>
   </producto>
   <producto>
      <nombre>Cereales con chocolate</nombre>
      <codigo>6482947100032</codigo>
      <preciooferta>2.75</preciooferta>
   </producto>
</listadeproductos>
```

En este punto, podríamos plantearnos la siguiente cuestión. Los productos en oferta siempre deben ir acompañados por su precio anterior, así que sería conveniente incluir los dos precios, aunque debemos tener en cuenta que el precio de oferta puede no existir si el producto no está en oferta. El documento DTD que resuelve este problema sería el siguiente:

```
<!ELEMENT listadeproductos (producto+)>
<!ELEMENT producto (nombre, codigo, precio, preciooferta?)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT preciooferta (#PCDATA)>
```

El siguiente documento XML muestra un ejemplo para este caso:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo:02-004.xml -->
<listadeproductos>
   <producto>
      <nombre>Galletas integrales</nombre>
      <codigo>8402973812000</codigo>
      <precio>6.45</precio>
   </producto>
   <producto>
      <nombre>Cereales con chocolate</nombre>
      <codigo>6482947100032</codigo>
      <precio>3.95</precio>
      <preciooferta>2.75</preciooferta>
   </producto>
</listadeproductos>
```

Ahora hemos definido que cada elemento `producto` debe contener obligatoriamente un elemento `nombre`, `codigo` y `precio`. Además, es posible que contenga un elemento `preciooferta`, aunque es opcional.

Definiendo elementos vacíos

En algunos casos necesitaremos que un elemento deba aparecer en el documento XML, pero sin valor alguno, vacío. Para definir un elemento vacío utilizamos la etiqueta `EMPTY` de la siguiente forma:

```
<!ELEMENT nombre-elemento EMPTY>
```

Vamos a ver un ejemplo en el que hemos definido un elemento vacío llamado tipo.

```
<!ELEMENT listadeproductos (producto+)>
<!ELEMENT producto (nombre, codigo, precio, preciooferta?, tipo)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT preciooferta (#PCDATA)>
<!ELEMENT tipo EMPTY>
```

El siguiente documento XML muestra un ejemplo de este caso:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-005.xml -->
<listadeproductos>
   <producto>
      <nombre>Galletas integrales</nombre>
      <codigo>8402973812000</codigo>
      <precio>6.45</precio>
      <tipo />
   </producto>
   <producto>
      <nombre>Cereales con chocolate</nombre>
      <codigo>6482947100032</codigo>
      <precio>3.95</precio>
      <preciooferta>2.75</preciooferta>
      <tipo />
   </producto>
</listadeproductos>
```

Como podemos observar, el elemento `tipo` está vacío, sin valor, y lo escribimos de la forma:

```
<tipo />
```

Los elementos vacíos pueden resultar útiles cuando lo único que nos interesa de ellos son sus atributos, tal y como veremos en algunos ejemplos a continuación.

Definiendo atributos

Teniendo en cuenta que los elementos pueden tener atributos para definir propiedades de éstos, los atributos también pueden definirse en el documento DTD siguiendo la sintaxis:

```
<!ATTLIST nombre-elemento nombre-atributo especificaciones>
```

Debemos tener en cuenta que, mientras que en los elementos definíamos el tipo de dato como #PCDATA, en los atributos lo definimos como CDATA para indicar que contiene un valor formado por caracteres.

Atributos obligatorios

Cuando queremos definir que la presencia de un atributo es obligatoria, empleamos la palabra clave #REQUIRED. En el siguiente ejemplo, indicamos que el elemento tipo tiene un atributo llamado es-dietetico que almacenará un valor cualquiera (CDATA) y que este atributo debe aparecer obligatoriamente (#REQUIRED).

```
<!ATTLIST tipo es-dietetico CDATA #REQUIRED>
```

Veamos un ejemplo de documento XML donde el elemento `tipo` tiene el atributo `es-dietetico`.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-006.xml -->
<listadeproductos>
   <producto>
      <nombre>Galletas integrales</nombre>
      <codigo>8402973812000</codigo>
      <precio>6.45</precio>
      <tipo es-dietetico="si" />
   </producto>
   <producto>
      <nombre>Cereales con chocolate</nombre>
      <codigo>6482947100032</codigo>
      <precio>3.95</precio>
      <preciooferta>2.75</preciooferta>
      <tipo es-dietetico="no" />
   </producto>
</listadeproductos>
```

Atributos opcionales

Cuando queremos definir que la presencia de un atributo es opcional, empleamos la palabra clave #IMPLIED. En el siguiente ejemplo, el elemento `tipo` tiene un atributo llamado `es-dietetico` cuya presencia es opcional (#IMPLIED).

```
<!ATTLIST tipo es-dietetico CDATA #IMPLIED>
```

Atributos constantes

También podemos definir un atributo con un valor fijo mediante la palabra clave #FIXED seguida del valor constante entre comillas dobles, como en el siguiente ejemplo, en el que el elemento `tipo` tiene un atributo que se llama `es-dietetico` cuyo valor constante (#FIXED) es "desconocido".

```
<!ATTLIST tipo es-dietetico CDATA #FIXED "desconocido">
```

Al ser un atributo definido con un valor constante o bien fijo, éste no puede variar.

Atributos con valor predefinido

El documento DTD nos permite definir un valor predeterminado para los atributos simplemente escribiendo este valor entre comillas dobles después del nombre del atributo. En este caso no se especifica la palabra clave CDATA.
En el siguiente ejemplo, vamos a indicar que el valor por defecto del atributo `es-dietetico`, del elemento `tipo`, es "no".

```
<!ATTLIST tipo es-dietetico "no">
```

Atributos con valor de tipo enumerado

Es posible que nos interese definir una serie de valores que puede tomar un atributo.
Para ello, después del nombre del atributo, escribiremos dentro de unos paréntesis los posibles valores separados por una barra vertical (`|`), que representa el operador `OR` u "o lógico".

```
<!ATTLIST tipo es-dietetico (si | no)>
```

En el ejemplo anterior hemos definido el atributo `es-directo` de forma que puede tomar el valor `si` o el valor `no`.
Si además deseamos definir un valor por defecto, simplemente escribimos tras los paréntesis y entre comillas dobles dicho valor, tal como se muestra a continuación.

```
<!ATTLIST tipo es-dietetico (si | no) "no">
```

Si no indicamos ningún valor por defecto y el atributo es obligatorio, recurriremos a la palabra clave #REQUIRED:

```
<!ATTLIST tipo es-dietetico (si | no) #REQUIRED>
```

Vincular el documento DTD con el documento XML

Hasta este momento hemos aprendido a crear documentos DTD. Ahora explicaremos de qué formas se puede vincular un documento DTD a un documento XML.

Documento DTD en documento XML

El documento DTD puede ser incluido por completo en el mismo documento XML, concretamente en el prólogo, tras la declaración XML, con la sixtaxis:

```
<!DOCTYPE nombre-elemento-raiz [ definiciones-DTD ]>
```

Donde `nombre-elemento-raiz` es el nombre del elemento principal del documento XML y `definiciones-DTD` son las definiciones del documento DTD. Además, en la declaración XML incluiremos el atributo:

```
standalone="yes"
```

Así indicaremos que el documento DTD está incluido y el documento XML no depende de otros documentos externos.
Siguiendo los ejemplos anteriores, el resultado sería el siguiente:

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<!-- Ejemplo: 02-007.xml -->
<!DOCTYPE listadeproductos [
   <!ELEMENT listadeproductos (producto+)>
   <!ELEMENT producto (nombre, codigo, precio, preciooferta?, tipo)>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT codigo (#PCDATA)>
   <!ELEMENT precio (#PCDATA)>
   <!ELEMENT preciooferta (#PCDATA)>
   <!ELEMENT tipo EMPTY>
   <!ATTLIST tipo es-dietetico (si | no) "no">
]>
<listadeproductos>
   <producto>
      <nombre>Galletas integrales</nombre>
      <codigo>8402973812000</codigo>
      <precio>6.45</precio>
      <tipo es-dietetico="si" />
   </producto>
   <producto>
      <nombre>Cereales con chocolate</nombre>
      <codigo>6482947100032</codigo>
      <precio>3.95</precio>
      <preciooferta>2.75</preciooferta>
      <tipo es-dietetico="no" />
   </producto>
</listadeproductos>
```

Documento DTD de sistema

Si el documento DTD se encuentra en otro archivo almacenado en nuestro propio sistema, por ejemplo con el nombre "misproductos.dtd", podemos referenciarlo desde el prólogo del documento XML mediante la sintaxis:

```
<!DOCTYPE nombre-elemento-raiz SYSTEM "nombre-archivo.dtd">
```

Donde `nombre-elemento-raiz` es el nombre del elemento principal del documento XML, `SYSTEM` es la palabra clave que nos indica que el documento DTD está almacenado en el propio sistema y "`nombre-archivo.dtd`" es el nombre del archivo con las definiciones DTD.
En este caso, el documento XML sería:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-008.xml -->
<!DOCTYPE listadeproductos SYSTEM "02-008.dtd">
<listadeproductos>
   <producto>
      <nombre>Galletas integrales</nombre>
      <codigo>8402973812000</codigo>
      <precio>6.45</precio>
      <tipo es-dietetico="si" />
   </producto>
   <producto>
      <nombre>Cereales con chocolate</nombre>
      <codigo>6482947100032</codigo>
      <precio>3.95</precio>
      <preciooferta>2.75</preciooferta>
      <tipo es-dietetico="no" />
   </producto>
</listadeproductos>
```

Vemos que en la declaración XML ya no está el atributo `standalone`. El documento DTD, almacenado en el archivo "02-008.dtd" tendría el siguiente contenido:

```
<!ELEMENT listadeproductos (producto+)>
<!ELEMENT producto (nombre, codigo, precio, preciooferta?, tipo)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT preciooferta (#PCDATA)>
<!ELEMENT tipo EMPTY>
<!ATTLIST tipo es-dietetico (si | no) "no">
```

Recordemos que en el caso de vincular el documento XML con el documento DTD utilizando SYSTEM, ambos documentos deben estar almacenados en la misma carpeta, o bien, debe indicarse la ruta del documento DTD dentro del propio sistema.

Documento DTD público

El hecho de que el documento DTD se almacene en nuestro propio sistema, como acabamos de ver, impide que otras personas puedan acceder a él. Sin embargo, es posible que nos interese que otras personas tengan acceso al

documento DTD para que puedan utilizarlo cuando lo necesiten para validar el documento XML que le enviamos, o bien, validar un documento XML que hayan hecho para nosotros. Esta opción es la que nos puede resultar más interesante en un principio.
En este caso, el archivo con las definiciones DTD se almacenará en algún sitio accesible a través de Internet y seguiremos la sintaxis:

```
<!DOCTYPE nombre-elemento-raiz PUBLIC "nombre-dtd" "ruta/nombre-archivo.dtd">
```

En donde `nombre-elemento-raiz` es el nombre del elemento principal del documento XML, `PUBLIC` es la palabra clave que indica que el documento DTD es de acceso público, `"nombre-dtd"` es el nombre que damos a la DTD y `"ruta/nombre-archivo.dtd"` es la dirección de Internet que incluye el archivo con las definiciones DTD. Ejemplo del documento XML:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-009.xml -->
<!DOCTYPE listadeproductos PUBLIC "misproductos"
      "http://dominio.com/02-009.dtd">
<listadeproductos>
   <producto>
      <nombre>Galletas integrales</nombre>
      <codigo>8402973812000</codigo>
      <precio>6.45</precio>
      <tipo es-dietetico="si" />
   </producto>
   <producto>
      <nombre>Cereales con chocolate</nombre>
      <codigo>6482947100032</codigo>
      <preciooferta>2.75</preciooferta>
      <tipo es-dietetico="no" />
   </producto>
</listadeproductos>
```

En este caso, el documento DTD "02-009.dtd" se encontraría almacenado en `http://dominio.com/` y tendría el siguiente contenido:

```
<!ELEMENT listadeproductos (producto+)>
<!ELEMENT producto (nombre, codigo, precio, preciooferta?, tipo)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT codigo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT preciooferta (#PCDATA)>
<!ELEMENT tipo EMPTY>
<!ATTLIST tipo es-dietetico (si | no) "no">
```

En este libro vincularemos por lo general el documento XML con el documento DTD usando SYSTEM, de esta forma podrá probar todos los ejemplos sin necesidad de tener un sitio Web, aunque como ya hemos explicado, lo ideal es que el documento DTD sea público.

Nuestro segundo documento DTD

Ya sabemos crear documentos DTD y vincularlos al documento XML, así que vamos a plantear un segundo ejemplo diferente y completo.
Partiremos del siguiente documento XML. Vamos a analizarlo, plantear algunos problemas y ver cómo lo solucionamos y creamos el documento DTD.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-010.xml -->
<catalogo>
   <producto>
      <calidad>10</calidad>
      <nombre>Mesa THYO</nombre>
      <referencia>3948</referencia>
      <stock>3</stock>
      <precio>60</precio>
   </producto>
   <producto>
      <calidad>7</calidad>
      <nombre>Silla THYO</nombre>
      <referencia>5343</referencia>
      <stock>8</stock>
      <precio>90</precio>
   </producto>
</catalogo>
```

Como podemos observar, tenemos un catálogo de productos, concretamente diversos muebles. Cada mueble tiene una calidad, un nombre, una referencia, un stock o número de unidades disponibles y un precio.
En un primer momento, podríamos crear un documento DTD como el que se muestra a continuación:

```
<!ELEMENT catalogo (producto+)>
<!ELEMENT producto (calidad, nombre, referencia, stock, precio)>
<!ELEMENT calidad (#PCDATA)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT referencia (#PCDATA)>
<!ELEMENT stock (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
```

Recordemos que el elemento `catalogo` pude incluir uno o más elementos `producto`, por lo que hemos escrito el operador + después del nombre del elemento `producto` en la primera línea.
Pero fijémonos en el elemento `calidad`. Si este elemento puede tomar un valor entre 1 y 10, no tenemos forma alguna de definir que un elemento puede tomar estos valores, pero sería interesante poder controlar esto. ¿Disponemos de algún mecanismo que nos permita resolver este problema? Sí, los atributos.

Podemos crear el elemento `calidad` como vacío, incorporarle un atributo llamado `valor` y definir para éste un conjunto de valores posibles entre 1 y 10.
El resultado del documento XML, ya vinculado al documento DTD, podría ser similar a:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-011.xml -->
<!DOCTYPE catalogo SYSTEM "02-011.dtd">
<catalogo>
   <producto>
      <calidad valor="10" />
      <nombre>Mesa THYO</nombre>
      <referencia>3948</referencia>
      <stock>3</stock>
      <precio>60</precio>
   </producto>
   <producto>
      <calidad valor="7" />
      <nombre>Silla THYO</nombre>
      <referencia>5343</referencia>
      <stock>8</stock>
      <precio>90</precio>
   </producto>
</catalogo>
```

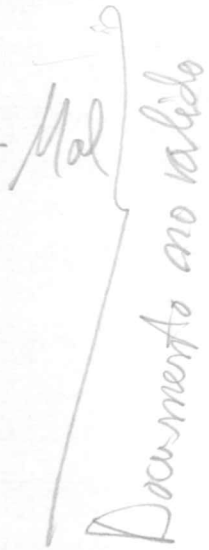
Y el documento DTD, con el nombre de archivo "02-011.dtd", almacenado en la misma carpeta que el documento XML, recordemos que hemos especificado SYSTEM, sería:

```
<!ELEMENT catalogo (producto+)>
<!ELEMENT producto (calidad, nombre, referencia, stock, precio)>
<!ELEMENT calidad EMPTY>
<!ATTLIST calidad valor (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) #REQUIRED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT referencia (#PCDATA)>
<!ELEMENT stock (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
```

Puesto que no indicamos ningún valor por defecto para el atributo `valor`, del elemento `calidad`, y es obligatorio especificarlo, recurriremos a la palabra clave #REQUIRED para indicar que es necesario incluir el atributo y darle un valor.
El documento XML anterior es válido considerando este documento DTD, pero el siguiente documento XML no es válido, pues el atributo `valor` del elemento `calidad` del primer producto no incluye uno de los valores posibles definidos.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-012.xml -->
<!DOCTYPE catalogo SYSTEM "02-011.dtd">
```

```
<catalogo>
   <producto>
      <calidad valor="desconocido" />
      <nombre>Mesa THYO</nombre>
      <referencia>3948</referencia>
      <stock>3</stock>
      <precio>60</precio>
   </producto>
   <producto>
      <calidad valor="7" />
      <nombre>Silla THYO</nombre>
      <referencia>5343</referencia>
      <stock>8</stock>
      <precio>90</precio>
   </producto>
</catalogo>
```



Nuestro tercer documento DTD

Analicemos otro documento XML que contiene un catálogo de libros.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Ejemplo: 02-013.xml -->
<!DOCTYPE catalogo SYSTEM "02-013.dtd">
<catalogo>
   <libro>
      <titulo>Manual Imprescindible de C/C++</titulo>
      <isbn>9788441526143</isbn>
      <autor>Miguel Angel Acera Garcia</autor>
      <paginas>416</paginas>
      <editorial>Anaya Multimedia</editorial>
   </libro>
   <libro>
      <titulo>Redes locales</titulo>
      <isbn>9788441519800</isbn>
      <autor>Jim Doherty</autor>
      <autor>Neil Anderson</autor>
      <paginas>544</paginas>
      <editorial>Anaya Multimedia</editorial>
   </libro>
</catalogo>
```

En este caso, encontramos una particularidad: todos los libros tienen al menos un autor, pero pueden tener varios. ¿Cuál sería el contenido del documento DTD "02-013.dtd" de este ejemplo? Solución:

```
<!ELEMENT catalogo (libro+)>
<!ELEMENT libro (titulo, isbn, autor+, paginas, editorial)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT paginas (#PCDATA)>
<!ELEMENT editorial (#PCDATA)>
```

Puesto que el elemento `catalogo` puede incluir uno o más elementos `libro`, hemos escrito el operador + después del nombre del elemento `libro` en la primera línea.
La clave está en la segunda línea, después del nombre del elemento `autor`, donde también se ha incluido el operador + para indicar que podemos encontrar uno o más elementos `autor` por cada `libro`.

Validación de documentos XML

Ya sabemos crear documentos XML y documentos DTD, pero ¿cómo validamos los documentos XML? Existen diversas herramientas y al final del libro encontrará un apéndice dedicado a éstas. Consúltelo para aprender a validar documentos XML.

Ejercicios resueltos

Intente realizar los siguientes ejercicios. Encontrará la solución en los apéndices de este libro.

1. Complete el DTD del siguiente documento XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Ejercicio: A02-001.xml -->
<!DOCTYPE listatickets [

]>
<listatickets>
  <ticket>
    <numero>7483</numero>
    <fecha>25/08/2016</fecha>
    <hora>22:00</hora>
    <producto>
      <nombre>Agua</nombre>
      <precio>1</precio>
    </producto>
    <producto>
      <nombre>Refresco</nombre>
      <precio>1.50</precio>
    </producto>
    <total>2.5</total>
  </ticket>
  <ticket>
    <numero>7484</numero>
    <fecha>25/08/2016</fecha>
    <hora>22:05</hora>
```

```
        <producto>
           <nombre>Refresco</nombre>
           <precio>1.5</precio>
        </producto>
        <producto>
           <nombre>Refresco</nombre>
           <precio>1.5</precio>
        </producto>
        <total>3</total>
    </ticket>
</listatickets>
```

2. El siguiente documento XML, que incluye un DTD, almacena información sobre dónde se almacenan los productos de una tienda de muebles dentro del gran almacén. Válidelo y, en caso de que hubiese algún error en la estructura, corríjalo.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Ejercicio: A02-002.xml -->
<!DOCTYPE almacen [
   <!ELEMENT almacen (producto+)>
   <!ELEMENT producto (referencia, seccion, pasillo, descripcion)>
   <!ELEMENT referencia (#PCDATA)>
   <!ELEMENT seccion (#PCDATA)>
   <!ELEMENT pasillo (#PCDATA)>
   <!ELEMENT descripcion (#PCDATA)>
]>
<almacen>
   <producto>
      <referencia>27.443.23</referencia>
      <seccion>18</seccion>
      <pasillo>3</pasillo>
      <descripcion>Silla SHYO<descripcion>
   </producto>
   <producto>
      <referencia>191.67.689</referencia>
      <seccion>7</seccion>
      <seccion>2</seccion>
      <descripcion>Mesa MHYO<descripcion>
   </producto>
   <producto>
      <referencia>91.67.689</referencia>
      <seccion>7</seccion>
      <pasillo>5</pasillo>
      <descripcion>Taburete THYO<descripcion>
   </producto>
</almacen>
```

3. El siguiente documento XML incluye un DTD. Nos han confirmado que la estructura y los datos sí son correcto, pero que el DTD no está bien escrito. Revise el DTD y realice en él las modificaciones necesarias para que el documento sea válido. Recuerde: sólo debe modificar el DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Ejercicio: A02-003.xml -->
<!DOCTYPE libro [
   <!ELEMENT libro (capitulo)>
   <!ELEMENT capitulo (titulo, paginas, autor)>
   <!ELEMENT titulo (#PCDATA)>
   <!ELEMENT autor (#PCDATA)>
]>
<libro>
   <capitulo>
      <titulo>El amanecer</titulo>
      <paginas>18</paginas>
      <autor>Carmen</autor>
   </capitulo>
   <capitulo>
      <titulo>Una tarde apacible</titulo>
      <paginas>34</paginas>
      <autor>Antonio</autor>
      <autor>Carmen</autor>
   </capitulo>
   <capitulo>
      <titulo>Conociendo el entorno</titulo>
      <paginas>27</paginas>
      <autor>Antonio</autor>
   </capitulo>
</libro>
```

Resumen

Los documentos DTD nos permiten definir normas que nos dicen cómo debe ser el contenido de un determinado documento XML. Sin estas definiciones, un documento XML podría ser un auténtico caos y no habría forma de saber qué está bien y qué está mal.

El documento DTD puede estar incluido en el propio documento XML, en el propio sistema o en un lugar con acceso público a través de Internet, como un sitio Web. Esta tercera opción es idónea cuando deseamos mejorar la comunicación con otras personas o empresas, puesto que así permitimos que ellos mismos puedan validar los documentos XML relacionados con ese documento DTD.

¿Recuerda el ejemplo de la editorial y las librerías que presentamos en el capítulo anterior? Si la editorial pusiese a disposición de la librerías el documento DTD que define cómo debe ser el documento XML que les envía, las librerías también podrían comprobar si está correcto cuando lo reciben. Además, las librerías podrían crear y enviar a la editorial otros documento XML, por ejemplo con una relación de libros que necesitan, manteniendo la misma estructura y empleando para su validación el documento DTD de

la editorial. Este ejemplo sencillo nos permite mostrar la utilidad que tiene que los documentos DTD sean públicos para que todos los participantes en una tarea puedan usarlo y validar sus documentos XML cuyas estructuras son iguales.

Lo cierto es que los documentos DTD tienen la ventaja de que son muy sencillos, lo cual es genial para muchos casos, pero para otros casos esto es a la vez una desventaja, pues no nos permiten crear reglas más estrictas cuando lo necesitamos. Este problema lo resolveremos en el siguiente capítulo.