LENGUAJES DE MARCAS

Unidad 3. DTD (Document Type Definition) y validación.

Índice de contenido

1. Introducción

Hasta ahora hemos visto cómo crear documentos XML bien formados, en los que partiendo de un único elemento raíz comprobábamos que todos sus elementos estaban anidados adecuadamente.

Pero cuando un documento XML no contiene un DTD (que pronto veremos lo que es), cualquier etiqueta que aparezca en el mismo se considerará válida. De manera que el parser o analizador sólo podrá comprobar que el documento está bien formado. La existencia del DTD permite asegurar que los documentos siguen las reglas del lenguaje.

Por lo tanto es imprescindible la especificación de un DTD que defina formalmente el lenguaje de etiquetado requerido. Este debería ser el primer paso antes de escribir cualquier documento XML.

Podemos hacer una comparación con el procedimiento que se usa trabajando con una base de datos: primero se define la estructura o esquema y luego ya se puede trabajar con los datos correspondientes.

Un documento XML válido es un documento “bien formado” que, además, se ajusta a las reglas de un DTD (Document Type Definition).

O sea que:

- Un XML con la sintaxis correcta es un documento “bien formado”.
- Un XML validado contra un DTD es un XML “válido”.

Mediante el uso de los DTD podremos ***validar*** documentos XML. La validación de documentos consiste en comprobar que, además de ser bien formados, se corresponden con la estructura prevista para el contenido que aportan.

Si ponemos como ejemplo el ejercicio de los coches del tema anterior, y suponemos que cada tienda o concesionario tiene que enviar su información a un sitio web donde se publicarán las ofertas, entendemos que no puede ser viable que cada una de esas empresas implemente su propia versión de documento XML. Por el contrario, lo adecuado es que el sitio web defina el formato exacto que deben seguir los documentos se van a recibir, y que así se asegure que serán documento XML ***válidos*** y todos con la misma estructura.

2. Declaración/definición del DTD

Una definición de tipo de documento (DTD) es una descripción de la estructura y sintaxis de un documento XML.

Una DTD define las reglas que debe cumplir la información contenida en un documento XML, para que el documento sea válido. Cuando creamos una definición de tipo de documento lo que estamos haciendo es crear nuestro propio lenguaje de marcas para nuestra aplicación concreta, de forma que el documento XML que se ajuste a esa DTD se pueda considerar **válido**.

En un DTD se describen los elementos (los nombres de los elementos, los atributos que pueden tener, los tipos de datos que pueden contener, etc.) que podrá contener el documento así como su estructura y posibilidades de anidamiento.

La DTD puede ser un fichero externo, con extensión .dtd, aunque también puede estar contenida en el propio documento XML, incluido en la declaración de tipo de documento, que como vimos en el tema anterior, forma parte del prólogo del documento.

Por tanto no hay que confundir ambas cosas, ya que:

- la ***Definición*** (DTD) contendrá las especificaciones necesarias sobre los elementos, pudiendo estar incluido o no en el propio documento
- la D**eclaración** siempre estará en el prólogo del documento (incluyendo la definición del etiquetado o simplemente haciendo referencia a su ubicación exterior).

Esta declaración aunque es de carácter opcional en el documento XML, será necesaria para poder validar los datos que contiene. Estará situada en el prólogo del documento justo a continuación de la declaración XML, en la segunda línea, mediante la declaración **DOCTYPE** y deberá contener **siempre** la **espe**cificación de elemento raíz del documento.

2.1. *Definición interna*

El formato de declaración, cuando incluya la definición en el propio documento, podría ser:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE elemento_raiz [
        Declaraciones internas
]>
```

```
<elemento_raiz>
  ……..
</elemento_raiz>
```

Es decir, estará incluida dentro de la declaración DOCTYPE, después del elemento_raiz y comprendida entre **"["** y **"]"**

Ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE nota
[
        <!ELEMENT nota (destinatario,remitente,cabecera,cuerpo)>
        <!ELEMENT destinatario (#PCDATA)>
        <!ELEMENT remitente (#PCDATA)>
        <!ELEMENT cabecera (#PCDATA)>
        <!ELEMENT cuerpo (#PCDATA)>
]>
<nota>
        <destinatario>Tove</destinatario>
        <remitente>Jani</remitente>
        <cabecera>Recordatorio</cabecera>
        <cuerpo>Llámame!</cuerpo>
</nota>
```

El DTD anterior tiene el siguiente significado:

- **!DOCTYPE nota**, indica que el elemento raíz de este documento es nota.
- **!ELEMENT nota**, indica que el elemento nota contiene cuatro elementos: destinatario, remitente, cabecera y cuerpo.
- **!ELEMENT destinatario**, indica que el elemento destinatario es de tipo #PCDATA, es decir, texto.
- **!ELEMENT remitente**, indica que el elemento remitente es de tipo #PCDATA.
- **!ELEMENT cabecera**, indica que el elemento cabecera es de tipo #PCDATA.
- **!ELEMENT cuerpo**, indica que el elemento cuerpo es de tipo #PCDATA.

2.2. Definición externa

Pero un DTD normalmente se utiliza para validar un gran número de documentos XML por lo que la mayoría de las veces tiene poco sentido que esté incluido dentro del propio documento ya que se tendría que repetir en todos ellos.

Lo habitual será que se separen el documento XML del DTD, por lo que cada uno de ellos estará situado en un fichero distinto.

Teniendo en cuenta esto, se puede distinguir entre dos tipos de referencias externas:

2.2.1. Un documento DTD que no es público.

En este caso nosotros mismos hemos creado el archivo .dtd y lo tenemos en una máquina local o un servidor de nuestra red, pero no es público en Internet.

En este caso se especifica con la palabra **SYSTEM**. Deberemos indicar en la declaración la ubicación del .dtd, bien con la ruta al fichero o la URL al mismo.

Ejemplos de declaraciones podrían ser:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE elemento_raiz SYSTEM "fichero.dtd">
<elemento_raiz>
  ……..
</elemento_raiz>
```

Otras declaraciones válidas podrían ser:

<!DOCTYPE Agenda SYSTEM "http://www.mi-sitio.com/dtd/mi_archivo.dtd">

<!DOCTYPE Agenda SYSTEM "dtd/mi_archivo.dtd">

En el primer ejemplo, “fichero.dtd” debería estar ubicado en el mismo directorio que el fichero .xml. En el segundo, accederíamos a él a través de su URL. Y en el tercero estaría dentro del directorio dtd que se encuentra al mismo nivel que el fichero xml.

Siguiendo el ejemplo de la “nota” del punto anterior, si en lugar de realizar una definición interna en el propio documento xml la hiciéramos en un dtd externo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE nota SYSTEM “nota.dtd”>
<nota>
        <destinatario>Tove</destinatario>
        <remitente>Jani</remitente>
        <cabecera>Recordatorio</cabecera>
        <cuerpo>Llámame!</cuerpo>
</nota>
```

La declaración DOCTYPE es una referencia a un archivo DTD externo, llamado nota.dtd (que se encontrará en el mismo directorio que el documento xml), junto con el nombre del elemento raíz del documento.

Si en lugar de usar la ruta al documento usáramos una URL podríamos sustituír la segunda línea por:

```
<!DOCTYPE nota SYSTEM "http://www.servidor.org/dtd/nota.dtd">
```

Independientemente de su ubicación el archivo “nota.dtd” contendrá las siguiente información:

```
<!ELEMENT nota (destinatario,remitente,cabecera,cuerpo)>
<!ELEMENT destinatario (#PCDATA)>
<!ELEMENT remitente (#PCDATA)>
<!ELEMENT cabecera (#PCDATA)>
<!ELEMENT cuerpo (#PCDATA)>
```

2.2.2. Un DTD que ha sido publicado.

En este caso el DTD es público. Se sustituirá la palabra **SYSTEM** por **PUBLIC**, seguido del identificador público asociado a este DTD.

Este sistema se usa cuando queremos validar nuestro archivo .xml contra “vocabularios públicos”, es decir, formatos de ficheros xml públicamente conocidos. En estos casos suele indicarse la ruta al fichero y también la URL, que se usa sólo en caso de que no se localice el fichero a través de la ruta. Un ejemplo podría ser el dtd utilizado para validar documentos XHTML estrictos. En este ejemplo la declaración sería:

<!DOCTYPE HTML **PUBLIC** "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

3. Declaración de tipos de elementos

Los elementos son la base de las marcas XML. Indican qué etiquetas serán permitidas en el documento. El DTD tiene que declarar cada uno de ellos, y las declaraciones de tipo de elemento deben empezar con "**<!ELEMENT**" seguidas por el identificador genérico del elemento que se declara.

<!ELEMENT *nombre* *tipo_contenido***>**

El nombre del elemento debe ser un nombre XML válido y sólo puede haber una declaración por elemento. No podrá repetirse.

Veremos un pequeño caso que iremos analizando, basado en el ejemplo “identificacion” del tema anterior.

Teníamos el siguiente documento:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE identificacion SYSTEM "identificacion.dtd">

<identificacion>
        <nombre_completo>
                <nombre>
                        Pepe
                </nombre>

                <apellido1>
                        Gonzalez
                </apellido1>

                <apellido2>
                        Ribera
                </apellido2>
        </nombre_completo>

        <apodo>
                Pepito Grillo
        </apodo>
        </identificacion>
```

Al cual le podría corresponder un DTD como el siguiente, que estaría contenido en el fichero identificacion.dtd :

```
<!ELEMENT identificacion (nombre_completo, apodo)>
    <!ELEMENT nombre_completo (nombre, apellido1,apellido2)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT apellido1 (#PCDATA)>
    <!ELEMENT apellido2 (#PCDATA)>
<!ELEMENT apodo (#PCDATA)>
```

Vemos como en el dtd lo primero que hacemos es definir el “elemento raíz” que llamamos “identificacion” y está formado, a su vez, por dos elementos: “nombre_completo” y “apodo”. Por otro lado, “nombre_completo” está formado, a su vez por otros tres elementos: “nombre”, “apellido1” y “apellido2”.

También podemos observar cómo los elementos que no continen a otros elementos, como “nombre”, “apellido1”, “apellido2” y “apodo” definen que el tipo de datos de esos elementos es #PCDATA, que más tarde veremos que es texto plano.

Comprueba también que el fichero xml es válido, es decir, sigue la estructura definida por el dtd.

Vamos ahora a ampliarlo con algunos elementos que explicaremos en cada caso:

El nuevo DTD podría validar documentos con contenidos similares al siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE identificacion SYSTEM "identificacion_ampliado.dtd">
<identificacion>
    <situacion>
        <estudiante>
          Universiitat Jaume I Castelló
        </estudiante>
         <!– esto es un comentario:
                estudiante y trabajador son alternativos,
                si lleva información de uno no tendrá del otro -->
    </situacion>

    <nombre_completo>
          <nombre>
            Pepe
          </nombre>

          <apellido1>
            Gonzalez
          </apellido1>

          <apellido2>
            Ribera
          </apellido2>
    </nombre_completo>

    <apodo>
      Pepito Grillo
    </apodo>

    <mail>
          elgrillito@correobasura.com
    </mail>
</identificacion>
```

Ampliamos el DTD con más elementos que explicaremos a continuación

```
<!ELEMENT identificacion (situacion?,nombre_completo, apodo*, mail?)>
   <!ELEMENT situacion (estudiante|trabajador)>
   <!ELEMENT estudiante (#PCDATA)>
   <!ELEMENT trabajador (#PCDATA)>
   <!ELEMENT nombre_completo (nombre+, apellido1,apellido2)>
      <!ELEMENT nombre (#PCDATA)>
      <!ELEMENT apellido1 (#PCDATA)>
      <!ELEMENT apellido2 (#PCDATA)>
   <!ELEMENT apodo (#PCDATA)>
   <!ELEMENT mail (#PCDATA)>
```

Según el contenido de los elementos podemos tener diferentes tipos:

3.1. Elementos que sólo contienen elementos

En este caso tendremos que especificar, entre paréntesis, el identificador de cada unos de los elementos que anidará.

Hay dos tipos de relación entre los elementos hijos:

•**Secuenciales**, referenciándolos por su nombre, separados por comas:

<!ELEMENT nombre_elemento (elemento1,elemento2,elemento3....)>

En nuestro ejemplo, el elemento raíz (que siempre incorporaremos en primer lugar), hemos visto que lo especificamos como:

<!ELEMENT identificacion (situacion?,nombre_completo, apodo*,mail?)>

•**Alternativos:** Cuando el elemento contiene uno y solo uno de los elementos hijos especificados, en cuyo caso los separaremos mediante "|":

<!ELEMENT nombre_elemento (elemento1 | elemento2|elemento3....)>

En nuestro ejemplo tenemos:

<!ELEMENT situacion (estudiante|trabajador)>

En este ejemplo el elemento situación podrá tomar uno de los dos valores especificados: estudiante o trabajador, pero sólo uno de ellos.

Además de especificar qué elementos hijos puede contener el elemento y en qué orden, se puede establecer cuántas veces aparece cada uno de ellos, con un carácter que indique el factor de repetición, o un **indicador de frecuencia:**

Indicadores de frecuencia

INDICADOR	TIPO	FRECUENCIA
?	Opcional	0 o 1 vez
*****	Opcional y repetible	0 o más veces
+	Necesario y repetible	1 o más veces
Si no ponemos nada	Necesario y NO repetible	Debe aparecer 1 vez

Siguiendo nuestro ejemplo:

<!ELEMENT identificacion (situacion?,nombre_completo, apodo*,mail?)>

Si consideramos que nuestro elemento raiz (identificacion) está formado por los elementos "situación" y "mail" que son opcionales, pueden aparecer una vez o no aparecer; "nombre_completo" siempre debe aparecer, aunque solo una vez ya que aparece en la lista sin ningún indicador de frecuencia, mientas que el apodo puede no estar presente o aparecer repetidamente, ya que una persona puede no tener ningún apodo o varios.

<!ELEMENT nombre_completo (nombre+, apellido1,apellido2)>

Así mismo, anidado en nombre_completo, tenemos el nombre que podemos considerar que es necesario una vez, es decir, todas las personas tienen como mínimo un nombre, aunque pueden tener más, mientras tanto el apellido1 como el apellido2 serán únicos.

3.2. ELEMENTOS QUE SOLO CONTIENEN DATOS

En la declaración se especifican mediante **(#PCDATA)** e indica que pueden contener datos de tipo carácter (**Parser Character Data**). Debemos tener cuidado de que entre el identificador del elemento y el símbolo inicial del paréntesis haya un espacio de separación.

En nuestro ejemplo los siguientes elementos son de tipo texto:

<!ELEMENT nombre (#PCDATA)>

<!ELEMENT apellido1 (#PCDATA)>

<!ELEMENT apellido2 (#PCDATA)>

<!ELEMENT apodo (#PCDATA)>

<!ELEMENT mail (#PCDATA)>

3.3. Elementos vacíos

Aunque no es usual, los elementos pueden no tener ningún contenido pero pueden utilizarse para insertar los atributos (que veremos su significado posteriormente).

Se declaran especificando la palabra **EMPTY**.

<!ELEMENT nombre_elemento EMPTY>

Un ejemplo es la declaración
 de XHTML.

<!ELEMENT br EMPTY>

3.4. Elementos mixtos

No suelen utilizarse en XML ya que se puede especificar qué elementos hijos podrán aparecer, pero no dan indicación de frecuencia o si forman parte de una secuencia alternativa. Su formato es muy rígido, siempre en primer lugar PCDATA, con una lista alternativa como un grupo.

No se puede aplicar caracteres de repetición a los elementos hijos, solo la posibilidad del carácter de repetición * para el conjunto. (Debe especificarse obligatoriamente el carácter de repetición '*' a todo el grupo).

Se desaconseja su uso.

La declaración sería:

<!ELEMENT nombre elemento (#PCDATA|elem1|elem2|elem3)*>

4. Declaración de tipos de atributos

Ya hablamos de que los atributos permiten añadir información adicional a los elementos de un documento. Y que las diferencias entre los elementos y los atributos son:

Los atributos no pueden contener sub-atributos y que los usamos para añadir información corta, sencilla y desestructurada.

Cada uno de los atributos sólo se puede especificar una vez, y en cualquier orden.

<mensaje prioridad="urgente">

<de>Alfredo Reino</de>

<a>Hans van Parijs

<texto idioma="holandés">

Hallo Hans, hoe gaat het?

...

</texto>

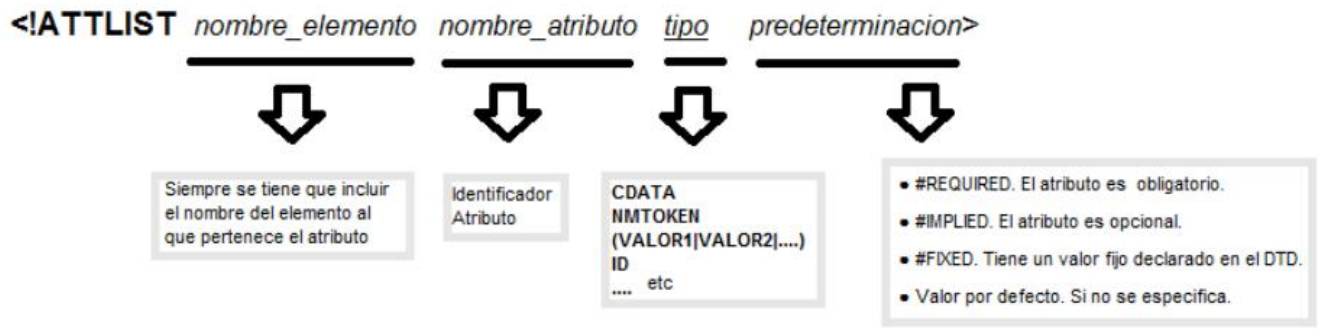
</mensaje>

Igual que ocurre con los elementos, cada uno de los distintos atributos identificados en la fase de diseño, debe declararse previamente en el DTD.

Pueden agruparse en una lista correspondiente para cada elemento.

Para cada atributo podremos tener también la especificación de su tipo y su valor por defecto o predeterminación. Puede haber múltiples definiciones de listas de atributos para un mismo elemento. Pero si se declara varias veces el mismo atributo sólo prevalece el primero.

Su sintaxis será:



En el ejemplo anterior, para declarar la lista de atributos del elemento <mensaje> podríamos definir utilizar la siguiente definición de atributos:

<!ELEMENT mensaje (de, a, texto)>

<!ATTLIST mensaje prioridad (normal | urgente)>

Así pues, definimos para el elemento “mensaje” un atributo que llamamos “prioridad” y que puede tomar los valores “normal” o “urgente”.

Veamos a continuación cada parte del atributo con más detalle.

4.1. Tipo del atributo

El tipo del atributo podrá tomar los siguientes valores:

- CDATA
- NMTOKEN
- NMTOKENS
- Tipos de atributos enumerados
- Tipos de atributos ID

4.1.1. Tipo de atributo CDATA, NMTOKEN y NMTOKENS.

Si el valor del atributo está formado por un texto que puede incluir cualquier carácter imprimible, a excepción de los caracteres especiales, incluidos los espacios en blanco, entonces el tipo sera **CDATA.**

Si pretendemos limitar el tipo de caracteres que pueden aparecer como valor en el atributo, debemos utilizar el tipo **NMTOKEN**. Sólo permite que aparezcan los mismos caracteres que utilizamos para definir elementos y atributos.

Existe también la posibilidad de utilizar el tipo **NMTOKENS.** Esto indica que el atributo contendrá una lista de cadenas de tipo **NMTOKEN**.

Ejemplos (aunque los puntos …. indica que la definición del atributo está incompleta, según veremos posteriormente:

<!ATTLIST coche color CDATA … >

Significa que el atributo color puede tomar cualquier valor: blanco-rojo, rojo, beige.claro, azul_celeste? ...

<!ATTLIST coche color NMTOKEN...>.

Significa que la propiedad color puede tomar solo valores que contengan letras, dígitos, puntos, guiones y subrayados. Deben comenzar por letra y no pueden contener espacios en blanco.

<ATTLIST coche color NMTOKENS...>

La propiedad color será una lista de NMTOKENS. Por ejemplo: <coche color=”blanco negro gris”>

4.1.2. Tipos de atributos enumerados

Se usan cuando el valor del atributo está restringido a un conjunto de valores. En la declaración se usa el carácter '|' para separar los valores.

<!ATTLIST coche color (blanco | negro | gris)>

De esta forma, la propiedad color solo puede tomar los valores “blanco”, “negro” o “gris”, y sólo uno de ellos. Cualquier otro valor hará que la validación del documento XML falle.

4.1.3. Tipos de atributos ID

Es frecuente que algunos elementos tengan algún valor que los identifica de forma unívoca. Cuando un elemento contiene una propiedad de este tipo hay que asegurarse que ésta no se repite en otro elemento. Incluso con elementos diferentes. Podemos poner:

<!ATTLIST coche matricula ID ...>

4.1.4. Predeterminación de los atributos

A continuación del nombre y el tipo del atributo debemos especificar si se requiere o no la presencia de un atributo, y el modo de gestionarlo en ese caso.

Existen cuatro posibles alternativas:

- #REQUIRED. El atributo es obligatorio.
- #IMPLIED. El atributo es opcional.
- #FIXED. Tiene un valor fijo declarado en el DTD.
- Valor por defecto si no se especifica.

Si no se define ninguna de estas alternativas el atributo será por defecto opcional.

Siguiendo y completando nuestros ejemplos de los coches, podríamos tener:

<!ATTLIST coche color CDATA #IMPLIED>

<!ATTLIST coche matricula ID #REQUIRED>

<!ATTLIST coche color CDATA “rojo”>

<!ATTLIST coche marca CDATA FIXED “Seat”>

5. Declaración de Entidades

Una entidad es una referencia a un objeto (texto, ficheros, páginas web, etc.,) que serán sustituidas por el contenido al que se refieren.

Permite guardar contenido que puede ser utilizado muchas veces y poder descomponer un documento grande en subconjuntos más manejables.

En ocasiones se emplean para descomponer un documento grande en otros más pequeños, y en otros casos se usan para representar caracteres que no pueden incluirse como texto, como el caso de caracteres especiales.

Su sintaxis general sería:

<!ENTITY identificador "valor">

Puede ser un **entidad interna**. Es la más sencilla. Consiste en abreviaturas definidas en el DTD. Ejemplo:

<!ENTITY tema “Introducción a XML”>

Una vez definida en el DTD, en el documento XML correspondiente podemos utilizarla insertando

&tema;

Es decir, con el identificador precedido de & y acabado en “;”. El parser cambiará la entidad por el valor asignado.

Existen también las **entidades externas**. Aquí no tenemos el contenido dentro del DTD sino en cualquier otro sitio del sistema. Se hace referencia a su contenido mediante una URL precedida de la palabra SYSTEM o PUBLIC según proceda, y de esa forma podemos incluir parte de archivos para poder descomponerlos en pequeñas partes. La sintaxis es

<!ENTITY nombre SYSTEM “URL”>

Por ejemplo:

<!ENTITY tema SYSTEM http://www.misapuntes.com/tema3.xml>

o

<!ENTITY intro SYSTEM http://ww.miservidor.com/intro.xml>

6. Comprobación de la validación de documentos XML frente al DTD

Existen distintos programas que además de ayudarnos en la edición del fichero XML y el DTD mediante el resaltado de las etiquetas, incorporación de sangrías, utilización de colores, etc., también nos comprueban que el documento esté bien formado (aunque eso ya vimos en el tema anterior que con el propio navegador Firefox podíamos realizar dicha comprobación) y además podemos realizar la validación del documento.

Aunque se podría utilizar cualquiera de ellos, en este curso utilizaremos el XML Copy Editor, que es Software libre y está disponible para su descarga. Se trata de un editor muy sencillo e intuitivo.

ANEXO: Procedimiento aconsejado para crear los DTD.

Aunque normalmente el DTD se utiliza en una definición externa, es decir, en un fichero separado con extensión .**dtd** , quizás la forma más sencilla para detectar los errores de validación no sea crear los dos ficheros (el .xml y el .dtd) independientes desde el principio. Como se explica en la página 3 se puede hacer una definición interna en el propio documento XML y así, facilitar la mecánica de la validación, de modo que cuando el documento ya esté comprobado podemos separarlos y disponer de un .dtd contra el que se pueda validar otros documentos XML.

Vamos a ver el proceso con el ejemplo más sencillo que hemos visto en el tema:

Paso 1) Realizamos el fichero XML y comprobamos que ***está bien formado***

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<identificacion>
    <nombre_completo>
        <nombre>
          Pepe
        </nombre>
        <apellido1>
          Gonzalez
        </apellido1>
         <apellido2>
           Ribera
         </apellido2>
    </nombre_completo>
     <apodo>
          Pepito Grillo
     </apodo>
</identificacion>
```

Paso 2) A continuación, en el mismo documento incluimos la definición con el formato de la declaración interna:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE identificacion [
    <!ELEMENT identificacion (nombre_completo, apodo)>
       <!ELEMENT nombre_completo (nombre, apellido1,apellido2)>
          <!ELEMENT nombre (#PCDATA)>
          <!ELEMENT apellido1 (#PCDATA)>
          <!ELEMENT apellido2 (#PCDATA)>
       <!ELEMENT apodo (#PCDATA)>
 ]>
<identificacion>
   <nombre_completo>
       <nombre>
         Pepe
       </nombre>
       <apellido1>
         Gonzalez
       </apellido1>
        <apellido2>
          Ribera
        </apellido2>
   </nombre_completo>
    <apodo>
      Pepito Grillo
    </apodo>
</identificacion>
```

Comprobamos que es válido y si nos da algún error en la definición lo corregimos y volvemos a comprobar, hasta que esté correcto.

Paso 3) Seleccionamos la parte de declaración interna (lo que está entre [ ]) y lo cortamos, para abrir un nuevo documento y pegarlo en él:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE identificacion [
    <!ELEMENT identificacion (nombre_completo, apodo)>
        <!ELEMENT nombre_completo (nombre, apellido1,apellido2)>
            <!ELEMENT nombre (#PCDATA)>
            <!ELEMENT apellido1 (#PCDATA)>
            <!ELEMENT apellido2 (#PCDATA)>
        <!ELEMENT apodo (#PCDATA)>
]>
        <identificacion>
    <nombre_completo>
        <nombre>
          Pepe
        </nombre>
        <apellido1>
          Gonzalez
        </apellido1>
         <apellido2>
           Ribera
         </apellido2>
    </nombre_completo>
     <apodo>
       Pepito Grillo
     </apodo>
        </identificacion>
```

Paso 4) Pegamos en un nuevo documento (eligiendo que sea del tipo DTD) que solo contiene la definición, y lo guardamos como .dtd

```
<!ELEMENT identificacion (nombre_completo, apodo)>
    <!ELEMENT nombre_completo (nombre, apellido1,apellido2)>
        <!ELEMENT nombre (#PCDATA)>
        <!ELEMENT apellido1 (#PCDATA)>
        <!ELEMENT apellido2 (#PCDATA)>
    <!ELEMENT apodo (#PCDATA)>
```

Paso 5) En el documento XML la linea de DOCTYPE la rehacemos para que tenga el formato de la llamada a la definición externa, en el que se ha sustituido todo lo que comprendía [----] por SYSTEM “nombre_fichero.dtd”:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE identificacion SYSTEM "identificacion.dtd">

<identificacion>
        <nombre_completo>
                <nombre>
                        Pepe
                </nombre>

                <apellido1>
                        Gonzalez
                </apellido1>

                <apellido2>
                        Ribera
                </apellido2>
        </nombre_completo>

        <apodo>
                Pepito Grillo
        </apodo>
        </identificacion>
```

De esta forma tenemos los dos ficheros por separado, no olvidar que deben residir en la misma carpeta.