Tema 9

Validación de documentos. DTD

Objetivos

- Reconocer la necesidad de describir la información contenida en los documentos XML y sus reglas de validación.
- Identificar las tecnologías relacionadas con la definición de los documentos XML.
- Analizar la estructura y sintaxis específica utilizada en la descripción.
- Crear descripciones de documentos XML.
- Utilizar descripciones para la elaboración y validación de documentos XML.
- Asociar descripciones de la información con los documentos XML.
- Utilizar herramientas específicas de descripción y validación de documentos XML.

Introducción

En el tema anterior se ha estudiado cuál es la estructura de un documento XML, qué componentes lo forman (elementos, atributos, etc.) y qué reglas deben cumplirse cuando se crea un documento XML para que sea correcto sintácticamente, es decir, para que esté bien formado. En este tema se estudian de las reglas que debe cumplir un documento XML para que sea válido, es decir, para que tenga significado. Las reglas de validación de un documento XML se relacionan con definir un lenguaje concreto basado en XML, con su propio vocabulario y reglas semánticas. Estas reglas pueden ser, por ejemplo: qué elementos o atributos pueden aparecen en el documento, en qué orden, cuáles son los posibles valores que pueden tomar, etc. La tecnología de validación que se estudia en este tema es DTD *(Document Type Definition)*.

Índice

9.1 DOCUMENTOS XML VÁLIDOS

Se dice que un documento XML es válido, o está validado, si se han definido unas reglas de validación asociadas a él y, además, el documento las cumple. Las reglas de validación especifican, por tanto, la estructura gramatical que debe tener un documento XML.

La función básica de la validación de documentos XML es la descripción de la estructura de datos que deben contener. Todos los documentos que utilicen las mismas reglas de validación tendrán la misma estructura y mantendrán la consistencia de la información que contienen. Además, estos documentos podrán ser validados, porque se conocen sus reglas de validación. Las reglas de validación definen la estructura de los elementos y la descripción de la información que contienen los documentos XML. **La validación de un documento XML consiste en comprobar si la información que contiene cumple con las reglas específicas que se han definido para ese conjunto de información en particular**. Es decir, consiste en comprobar si la información que contiene se corresponde con la estructura prevista para el contenido que proporciona.

Ahora bien, es diferente que un documento XML esté bien formado a que, además, sea válido.

- Un documento XML está **bien formado**, si su sintaxis es correcta. Es decir, si cumple con las reglas sintácticas establecidas por la especificación XML.

- Un documento XML es **válido**, si cumple con unas reglas de validación específicas asociadas a él.

Habitualmente, se suele considerar que si un documento es válido estará, también, bien formado. Sin embargo, un documento bien formado no tiene por qué estar validado. Los validadores de documentos XML suelen comprobar, en primer lugar, si el documento XML es correcto sintácticamente y, en segundo lugar, comprueban si es válido.

Para explicar la diferencia entre documento bien formado y documento válido, se puede hacer una comparación con dos frases en castellano. Por ejemplo, la primera frase: “montaña La nevada está”. Y, la segunda frase: “La montaña está nevada”. Desde un punto de vista sintáctico, ambas frases son correctas. Sin embargo, la primera frase tiene una estructura gramatical que no es válida y, por este motivo, carece de significado.

Tecnologías de validación de documentos XML

El uso de las técnicas de validación permite comprobar si un documento XML es válido, es decir, si cumple ciertas reglas de validación. Existen diversos métodos de validación de documentos XML, como son: DTD *(Document Type Validation)*, XML Schema, RELAX NG, Schematron, etc.

En este tema se aborda el estudio de la tecnología de validación DTD. Se trata de un mecanismo de validación sencillo que continúa utilizándose debido a su gran implantación. Actualmente, también se utilizan otras tecnologías de validación más modernas y complejas, como por ejemplo XML Schema, también conocida como Esquemas XML o XSD *(XML Schema Document)*.

9.2 VALIDACIÓN DE DOCUMENTOS XML MEDIANTE DTD

El método de validación DTD *(Document Type Validation)* es un estándar que permite definir una gramática que deben cumplir los documentos XML para que puedan ser considerados válidos. Una definición DTD para un determinado documento XML describe:

- Qué elementos (etiquetas) pueden existir en un documento XML y el tipo de su contenido.
- Qué atributos pueden tener los elementos.
- Qué elementos pueden o deben aparecer dentro de otros elementos.
- En qué orden deben aparecer los elementos.

El mecanismo de validación DTD ya existía incluso antes de la aparición de XML. Se usaba para validar documentos SGML Cuando apareció el lenguaje de marcas XML, se integró la validación DTD como modelo de validación gramatical de los documentos XML.

Una validación DTD es una descripción de la estructura que deben tener los documentos XML a los que se les asocie esa validación. Hay que tener en cuenta que una definición DTD puede asociarse a uno o varios documentos XML. Cuando se emplea la especificación DTD se pueden declarar diversos tipos de componentes que forman parte de la descripción de validación:

- Declaración DTD.
- Declaración de elementos.
- Declaración de atributos.
- Declaración de entidades.
- Declaración de notaciones.

En los siguientes apartados de este tema se aborda el estudio de cada uno de estos componentes.

9.3 DECLARACIÓN DTD

La declaración DTD define qué descripción de validación DTD se utilizará para validar el documento XML. Se utiliza la declaración <!DOCTYPE> para especificar la declaración DTD que debe aparecer en el prólogo del documento XML. La sintaxis de la declaración DTD es diferente según las características de la validación que se vayan a aplicar. Estas características se refieren a la ubicación y el tipo de uso de las reglas de validación.

- Ubicación. Define dónde se localizan las reglas de validación DTD. Puede utilizarse:

  - Validación interna. Las reglas de validación aparecen en el propio documento XML.
  - Validación externa. Las reglas de validación se especifican en un archivo externo.
  - Validación mixta. Las reglas de validación se especifican en el propio documento XML y, también, en un archivo externo. En este caso las reglas de validación internas tienen prioridad sobre las externas.

- Tipo de uso. Define el ámbito del uso de la declaración de validación DTD. Puede ser de:
  - Uso privado. Se identifica mediante la palabra SYSTEM.
  - Uso público. Se identifica mediante la palabra PUBLIC. Debe ir acompañada de la etiqueta FPI *(Formal Public Identifier)*, o Identificador Formal Público, que permite identificar a la descripción DTD correspondiente de manera universal.

Pueden darse las distintas combinaciones en cuanto a la ubicación y tipo de uso. Algunas de las posibles combinaciones pueden no tener sentido, Así, por ejemplo, una declaración DTD de uso público fuerza a que deba especificarse en un archivo independiente y, por tanto, deberá ser externa. En la siguiente tabla, se puede apreciar la sintaxis de las combinaciones más comunes:

Sintaxis	**Características DTD**
<!DOCTYPE elemento_raíz [reglas_de_validación]>	Interna y uso privado
<!DOCTYPE elemento_raíz SYSTEM uri>	Externa y uso privado
<!DOCTYPE elemento_raíz SYSTEM uri [reglas_de_validación]>	Mixta y uso privado
<!DOCTYPE elemento_raíz PUBLIC FPI uri>	Externa y uso público
<!DOCTYPE elemento_raíz PUBLIC FPI url [reglas_de_validación]>	Mixta y uso público

La sintaxis de la declaración DTD comienza, en todos los casos, con la palabra DOCTYPE y, a continuación, se especifica el nombre del elemento o nodo raíz del documento XML. Seguidamente, aparece el tipo de uso: SYSTEM o PUBLIC. Si se trata de una declaración de validación pública, además se deberá incluir el FPI. Si se trata de una declaración de validación externa, a continuación, se incluye la ruta y el nombre del archivo externo mediante formato URI. La ruta del archivo externo puede especificarse de manera absoluta o relativa. Se recomienda utilizar siempre rutas relativas. Y, finalmente, se pueden incluir las reglas de validación interna entre corchetes.

Ejemplo:

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

En ejemplo anterior, incluye solo validación DTD interna. Puede apreciarse que las reglas de validación han quedado especificadas dentro del propio documento XML. La definición de validación especificada en el ejemplo tiene el siguiente significado:

- <!DOCTYPE nota >. Especifica que la declaración DTD del documento XML es interna y de uso privado, así como que el elemento raíz del documento XML es el elemento nota.

- <!ELEMENT nota(destinatario,remitente,cabecera,cuerpo)>. Especifica que el elemento nota debe contener cuatro elementos: destinatario, remitente, cabecera y cuerpo. Y que, además, deberán aparecer en ese orden.

- <!ELEMENT destinatario (#PCDATA)>. Especifica que el elemento destinatario es de tipo #PCDATA, es decir, de tipo texto. Los demás elementos se especifican del mismo tipo.

A continuación, se realiza este mismo ejemplo, pero utilizando una declaración DTD externa. El resultado final será idéntico, aunque en este caso se utilizarán dos archivos:

Archivo: *Ejemplo1-Externa.xml*

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE nota SYSTEM "Ejemplo1-Externa.dtd">
<nota>
  <destinatario>Tove</destinatario>
  <remitente>Jani</remitente>
  <cabecera>Recordatorio</cabecera>
  <cuerpo>Llámame!</cuerpo>
</nota>
```

Archivo: *Ejemplo1-Externa.dtd*

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

Puede apreciarse que la declaración <DOCTYPE> del documento XML incluye la dirección URI que referencia al archivo externo de validación, de modo que el documento XML queda vinculado con la declaración DTD externa. En este caso, ambos archivos se encuentran en la misma carpeta.

En los siguientes apartados del tema se estudia con más detalle cómo se puede describir la validación de los elementos, los atributos, las entidades y las notaciones.

9.4 DECLARACIÓN DE ELEMENTOS

Se utiliza la declaración <!ELEMENT> para especificar cada uno de los elementos que forman parte del documento XML. Una declaración de elemento tiene la sintaxis siguiente:

<!ELEMENT nombre_elemento modelo_contenido>

Donde, *nombre_elemento* es el nombre del elemento que se está especificando y que debe coincidir con el nombre de la etiqueta correspondiente en el documento XML, aunque sin los signos menor que (<) y mayor que (>). Y donde, *modelo_contenido* es el modelo del contenido que se utiliza para especificar diversos aspectos de la declaración del elemento. Por ejemplo:

```
<!ELEMENT Receta (Nombre, Descripcion?, Ingredientes?, Instrucciones?)>
```

El código anterior define la etiqueta Receta. Además, especifica qué este elemento deberá contener, y en ese orden, las subetiquetas: Nombre, Descripción, Ingredientes e Instrucciones. Y, finalmente, se especifica que los tres últimos subelementos son opcionales, como indica el carácter ?.

A continuación, se estudian las opciones más importantes que pueden incluirse en la declaración de un elemento para describir sus características de validación.

Elementos vacíos

Se declaran especificando la palabra EMPTY. Por ejemplo:

```
<!ELEMENT nombre EMPTY>
```

En el caso del ejemplo anterior, el elemento nombre se usaría como un elemento vacío, es decir, de este modo: <nombre />.

Elementos que solo contienen información

Los elementos que contendrán información y que, por tanto, no podrán contener a otros elementos, se describen como #PCDATA *(Parsed Character Data)*. De modo que toda información contenida en un elemento será de tipo texto y representará valores alfanuméricos o de cualquier otro tipo. La descripción #PCDATA debe escribirse entre paréntesis. Por ejemplo:

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

La declaración anterior especifica que el elemento email contendrá información textual.

Elementos que solo contienen a otros elementos

La descripción de los elementos descendientes o subelementos debe aparecer entre paréntesis. El uso de diversas reglas permite diferenciar el tipo de relación que puede tener un elemento con sus elementos descendientes o hijos.

- **Cardinalidad de los elementos.** Para especificar cuántas veces puede aparecer cada uno de los elementos hijo de un elemento, se utiliza un carácter que indica el factor de repetición:

Símbolo	Significado
*	El elemento, o grupo de elementos, puede aparecer 0 o más veces
?	el elemento, o grupo de elementos, puede aparecer 0 o 1 veces
+	el elemento, o grupo de elementos, puede aparecer 1 o más veces
	Por defecto, si no se pone nada, el elemento debe aparecer una vez

La siguiente declaración indica que el elemento hijo sólo puede aparecer una vez dentro del elemento nombre: <!ELEMENT nombre (hijo)>. Para hacer que el elemento hijo pueda aparecer más de una vez y que como mínimo deba aparecer una vez: <!ELEMENT nombre (hijo+)>. Si se desea que el elemento hijo pueda aparecer cualquier número de veces, incluida ninguna vez: <!ELEMENT nombre (hijo*)>. Si se desea que el elemento hijo pueda aparecer solo una vez, pero que no sea obligatorio: <!ELEMENT nombre (hijo?)>.

- **Secuencias de elementos.** Los elementos que tendrán uno o más hijos se definen con el nombre de los elementos hijo entre paréntesis. En las secuencias de elementos hijo se pueden utilizar dos símbolos para indicar el orden en que un elemento debe aparecer, o bien, si debe aparecer como alternativa a otro.

Símbolo	Significado
A,B	Índica que el elemento A aparecerá inmediatamente antes del elemento B. Una secuencia de nodos hijo separados por comas índica el orden de aparición de los elementos en el documento XML
A|B	La barra vertical índica que los elementos hijo son alternativos, es decir, deberá aparecer el elemento A o el elemento B. El elemento contendrá uno y solo uno de los elementos hijo alternativos

Los elementos hijo que se declaran como una secuencia de elementos, separados por comas, deben aparecer en el mismo orden en el documento. Ejemplo:

```
<!ELEMENT mensaje (remitente, destinatario, asunto, cuerpo)>
<!ELEMENT coche (marca, matricula, color)>
```

Los elementos hijo de una secuencia de elementos también deben declararse en el mismo documento DTD. Estos elementos hijo, a su vez, pueden tener otros elementos hijo. Por ejemplo, la declaración completa del elemento coche sería:

```
<!ELEMENT coche (marca, matricula, color)>
<!ELEMENT marca (#PCDATA)>
<!ELEMENT matricula (#PCDATA)>
<!ELEMENT color (#PCDATA)>
```

Se pueden combinar el uso de símbolos de cardinalidad de los elementos con los símbolos de secuencias de elementos. A continuación, se incluyen algunos ejemplos:

Ejemplo. Un correo electrónico se describe como una secuencia de elementos, algunos opcionales:

```
<!ELEMENT email (para,cc?,cco?,asunto,cuerpo)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT cc (#PCDATA)>
<!ELEMENT cco (#PCDATA)>
<!ELEMENT asunto (#PCDATA)>
<!ELEMENT cuerpo (#PCDATA)>
```

Ejemplo. Un contrato incluye una lista de cláusulas. Cada una de las cláusulas consta de varios epígrafes y sus desarrollos correspondientes. Además, un contrato finaliza con un epílogo:

```
<!ELEMENT contrato (clausulas, epilogo)>
<!ELEMENT clausulas (clausula+)>
<!ELEMENT clausula ((epigrafe,desarrollo)+)>
<!ELEMENT epigrafe (#PCDATA)>
<!ELEMENT desarrollo (#PCDATA)>
<!ELEMENT epilogo (#PCDATA)>
```

Ejemplo. Sea la siguiente declaración combinada de subelementos:

```
<!ELEMENT elem (a, (b | c)*, d+, e?>
<!ELEMENT a (#PCDATA)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>
<!ELEMENT d (#PCDATA)>
<!ELEMENT e (#PCDATA)>
```

Todas las siguientes estructuras serían válidas para el elemento elem en un documento XML:

```
<elem>
   <a>Texto</a>
   <d>Texto</d>
</elem>
```

```
<elem>
   <a>Texto</a>
   <b>Texto</b>
   <d>Texto</d>
</elem>
```

```
<elem>
   <a>Texto</a>
   <d>Texto</d>
   <d>Texto</d>
   <e>Texto</e>
</elem>
```

```
<elem>
    <a>Texto</a>
    <c>Texto</c>
```

```
    <d>Texto</d>
    <d>Texto</d>
    <e>Texto</e>
</elem>
```

Elementos de contenido mixto

Se trata de elementos que pueden contener información y otros elementos. El uso de este caso es poco habitual. El formato de esta declaración es muy rígido. Siempre, debe aparecer (#PCDATA) en primer lugar y, después, una lista alternativa de posibles elementos hijo. En este caso, no se pueden aplicar símbolos de cardinalidad a los elementos hijo. Obligatoriamente, debe especificarse el símbolo de cardinalidad '*' a todo el grupo. Por ejemplo:

```
<!ELEMENT nombre (#PCDATA | a | b | c)*>
```

9.5 DECLARACIÓN DE ATRIBUTOS

Se utiliza la declaración <!ATTLIST> para especificar los atributos de un elemento en el documento XML. Una declaración de atributo tiene la sintaxis siguiente:

<!ATTLIST nombre_elemento nombre_atributo tipo_atributo valor_atributo>

Se suele utilizar una declaración <!ATTLIST> para declarar cada uno de los atributos de un elemento, aunque se puede utilizar una única declaración <!ATTLIST> para declarar todos los atributos de un elemento. Si se declara varias veces el mismo atributo, entonces prevalece la primera declaración.

Ejemplo. Declaración de un atributo de un elemento:

```
<!ATTLIST pago metodo CDATA "cheque">
```

En el ejemplo anterior, se declara el atributo metodo para el elemento pago. Este atributo tendrá como valor por defecto "cheque". En el documento XML asociado a la declaración del ejemplo anterior podría aparecer la etiqueta del elemento pago de la siguiente forma:

```
<pago metodo="cheque" />
```

Tipo de atributo

El tipo de atributo, que aparece como *tipo_atributo* en la sintaxis anterior, puede ser una de las siguientes opciones:

Tipo de atributo	Descripción
CDATA	El valor es texto alfanumérico
(v1 | v2 | v3 | ...)	El valor será uno de los que aparecen de la lista especificada
ID	El valor será un identificador único

Tipo de atributo	Descripción
IDREF	El valor es el ID de otro elemento
IDREFS	El valor es una lista de ID otros elementos
NMTOKEN	El valor contendrá letras, dígitos, puntos, guiones y subrayados
NMTOKENS	El valor es una lista de cadenas que contendrán letras, dígitos, puntos, guiones y subrayados
ENTITY	El valor es una entidad
ENTITIES	El valor es una lista de entidades
NOTATION	El valor es el nombre de una notación
xml:	El valor es un valor XML predefinido

Las opciones más habituales para describir el tipo de atributo son las dos primeras. El tipo de atributo CDATA indica que el atributo contendrá un valor de tipo texto que puede contener cualquier carácter a excepción de caracteres especiales, incluidos espacios en blanco. En el siguiente ejemplo, el atributo color del elemento coche puede tomar cualquier valor de tipo texto.

```
<!ATTLIST coche color CDATA>
```

Los atributos enumerados se usan cuando el valor del atributo está restringido a un conjunto de valores. De modo que, se declara una lista de posibles valores, de entre los cuales el atributo deberá tomar uno de ellos. Se usa el carácter '|' para separar los posibles valores. Por ejemplo:

```
<!ELEMENT semaforo EMPTY>
<!ATTLIST semaforo color (rojo | amarillo | verde) "verde">
```

En el ejemplo anterior, se describe una regla de validación para el elemento semáforo que es de contenido vacío. Se especifica que solo tendrá un atributo, denominado color, cuyos posibles valores son rojo, amarillo y verde. Y, además, el valor por defecto del atributo color será verde.

El tipo NMTOKEN se utiliza para limitar los caracteres que pueden aparecer como valor en un atributo. De este modo, solo se permite que aparezcan: letras, dígitos, puntos, guiones y subrayados. Además, los valores del atributo deberán comenzar por una letra, no podrán contener espacios en blanco y los espacios en blanco anteriores y posteriores se ignorarán. Por ejemplo:

```
<!ATTLIST coche color NMTOKEN>
```

El tipo NMTOKENS indica que el atributo contendrá una lista de cadenas formas por letras, dígitos, puntos, guiones y subrayados sin espacios en blanco. Por ejemplo:

```
<ATTLIST coche color NMTOKENS>
```

En el documento XML correspondiente al ejemplo anterior, la propiedad color será una lista valores de tipo NMTOKENS separadas por un espacio en blanco. Por ejemplo:

```
<coche color="blanco negro gris">
```

Es frecuente que algunos elementos tengan algún valor que los identifica de forma unívoca. Para ello, se utiliza el tipo de atributo ID. Por ejemplo:

```
<!ATTLIST coche matricula ID>
```

De esta forma se asegura que, en el documento XML, no habrá ningún elemento con un identificador igual, incluso en otros elementos. El valor del atributo debe seguir las mismas reglas que los nombres de atributos y elementos. Además, como se trata de un identificador único, entonces se permite que otros elementos puedan referenciarlo. Para ello, se utiliza el tipo IDREF. Los valores que puede tomar un atributo IDREF son el conjunto de identificadores declarados en el documento. Por ejemplo:

```
...
<!ATTLIST coche matricula ID>
<!ATTLIST multa matricula IDREF>
...
```

También se puede utilizar el tipo IDREFS que permite extender la definición de identificadores únicos de otros elementos a una lista de valores.

Valores para los atributos

Además del nombre y el tipo del atributo la declaración de atributos también permite especificar las características de la validación del atributo con relación a sus posibles valores. El valor del atributo, que aparece como *valor_atributo* en la sintaxis anterior, puede ser:

Valor de atributo	Descripción
"valor"	El valor que aparece entre comillas representa el valor por defecto del atributo
#REQUIRED	El atributo es obligatorio. No se le asigna ningún valor por defecto
#IMPLIED	El atributo es opcional. No se le asigna ningún valor por defecto
#FIXED "valor"	El atributo es obligatorio y se le asigna el valor especificado que, además, es el único valor que puede tomar

De manera predeterminada, si no se especifica ninguna de las opciones anteriores, el atributo es opcional. Existen casos especiales, por ejemplo, los atributos de tipo ID que pueden ser opcionales u obligatorios, pero no pueden tomar valores por defecto ni ser fijos.

Ejemplo. Sea la siguiente declaración del atributo método del elemento pago:

```
<!ELEMENT pago EMPTY>
<!ATTLIST pago metodo CDATA "contra-reembolso" >
```

Según la declaración anterior, el siguiente contenido de un documento XML sería válido.

```
<pago />
```

Además, en el caso anterior, como no se especifica ningún valor para el atributo metodo, entonces el valor del atributo sería "contra-reembolso". Sin embargo, en el ejemplo siguiente el valor del atributo método será "tarjeta", puesto que se ha especificado ese valor concreto.

```
<pago metodo="tarjeta" />
```

Ejemplo. La siguiente declaración de ejemplo:

```
<!ELEMENT pago EMPTY>
<!ATTLIST pago metodo CDATA #IMPLIED>
```

validará correctamente el siguiente contenido XML, porque el atributo es opcional:

```
<pago />
```

Y también validará correctamente el siguiente contenido XML:

```
<pago metodo="tarjeta" />
```

Se utilizará la opción #IMPLIED cuando no se desea forzar al usuario a tener que definir un atributo, ni tampoco se desea que el atributo tenga un valor por defecto.

Se utiliza la opción #REQUIRED en aquellos casos en los que no se proporciona un valor por defecto, pero se desea que el atributo aparezca en el elemento y se le asigne algún valor. Por ejemplo:

```
<!ATTLIST coche matricula ID #REQUIRED>
```

El ejemplo anterior validaría correctamente el siguiente contenido XML:

```
<coche matricula="ABC1234">
```

Se utiliza la opción #FIXED para que un atributo tome, necesariamente, el valor especificado.

```
<!ATTLIST coche marca CDATA #FIXED "Ford">
```

En ejemplo anterior validaría correctamente el siguiente contenido XML:

```
<coche marca="Ford">
```

A continuación, pueden analizarse los siguientes ejemplos completos.

Ejemplo 1. Archivo: *Agenda.xml*

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE agenda [
```

```
  <!ELEMENT agenda (nombre,apellido+,direccion,telefono+,email)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT apellido (#PCDATA)>
  <!ELEMENT direccion (domicilio,cp,ciudad)>
  <!ELEMENT telefono (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
  <!ELEMENT domicilio (#PCDATA)>
  <!ELEMENT cp (#PCDATA)>
  <!ELEMENT ciudad (#PCDATA)>
  <!ATTLIST telefono tipo CDATA #REQUIRED>
  ]>
<agenda>
  <nombre>Jose</nombre>
  <apellido>Perez</apellido>
  <apellido>Juan</apellido>
  <direccion>
    <domicilio>C/ La paz, 2</domicilio>
    <cp>Alicante</cp>
    <ciudad>Alicante</ciudad>
  </direccion>
  <telefono tipo="móvil">111</telefono>
  <telefono tipo="fijo">111</telefono>
  <email>jose.perez@correo.com</email>
</agenda>
```

Ejemplo 2. Archivo: *Receta.xml*

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE Receta SYSTEM "Receta.dtd">
<Receta>
  <Nombre>Tortilla de patatas</Nombre>
  <Descripcion>
    La tradicional y típica tortilla de patatas.
  </Descripcion>
  <Ingredientes>
    <Ingrediente>
      <Cantidad unidad="pieza">3</Cantidad>
      <Item>Patata</Item>
    </Ingrediente>
    <Ingrediente>
      <Cantidad unidad="pieza">2</Cantidad>
      <Item>Huevos</Item>
    </Ingrediente>
    <Ingrediente>
      <Cantidad unidad="litro">0.1</Cantidad>
      <Item>Aceite</Item>
    </Ingrediente>
  </Ingredientes>
  <Instrucciones>
    <Paso>
      Pelar y cortar las patatas en rodajas
    </Paso>
    <Paso>
      Batir los huevos y mezclar con las patatas
    </Paso>
    <Paso>
      Calentar aceite en una sartén
    </Paso>
    <Paso>
```

```
      Echar la mezcla en la sartén y cuajar la mezcla por ambos lados
    </Paso>
  </Instrucciones>
</Receta>
```

Archivo: *Receta.dtd*

```
<!ELEMENT Receta (Nombre, Descripcion?, Ingredientes?, Instrucciones?)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Descripcion (#PCDATA)>
<!ELEMENT Ingredientes (Ingrediente*)>
<!ELEMENT Ingrediente (Cantidad, Item)>
<!ELEMENT Cantidad (#PCDATA)>
<!ATTLIST Cantidad unidad CDATA #REQUIRED>
<!ELEMENT Item (#PCDATA)>
<!ATTLIST Item opcional CDATA "0" vegetariano CDATA "si">
<!ELEMENT Instrucciones (Paso+)>
<!ELEMENT Paso (#PCDATA)>
```

9.6 DECLARACIÓN DE ENTIDADES

En general, el concepto de entidad en una validación DTD se refiere a un elemento usado para guardar información.

En el siguiente ejemplo, se define una entidad interna cuya referencia será sustituida por su valor en el documento XML. El analizador o *parser* de XML será el encargado de realizar esta tarea de sustitución. Se escribirá *&usa;* para obtener el valor de la entidad en el documento XML:

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE pais
[
  <!ELEMENT pais (nombre,superficie,nhabitantes,fgobierno)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT superficie (#PCDATA)>
  <!ELEMENT nhabitantes (#PCDATA)>
  <!ELEMENT fgobierno (#PCDATA)>
  <!ENTITY usa "Estados Unidos">
]>
<pais>
  <nombre>&usa;</nombre>
  <superficie>9,83 millones de km2</superficie>
  <nhabitantes>Más de 324 millones de habitantes</nhabitantes>
  <fgobierno>República federal</fgobierno>
</pais>
```

Pueden definirse también entidades externas que permiten hacer referencia a un contenido externo mediante una URI precedida de la palabra SYSTEM o PUBLIC, según proceda. Estas entidades externas permiten guardar cierto contenido de información en un archivo externo que puede ser utilizado tantas veces como sea necesario. De este modo, se puede descomponer un documento extenso en subconjuntos o partes de contenido más pequeños y manejables.

9.5 DECLARACIÓN DE NOTACIONES

Este tipo de declaración permite utilizar un valor que ha sido declarado como una notación en la DTD. Se utilizan para especificar el formato de información que no es contenido XML. Desde un punto de vista práctico, las notaciones se suelen utilizar para describir el tipo de contenido, utilizando para ello el tipo MIME correspondiente, por ejemplo: “imagen/jpeg”, “text/plain”, etc.

```
<!ATTLIST rio
  fotografia ENTITY #IMPLIED
  tipo_fotografia NOTATION (GIF | JPG | PNG) #IMPLIED>
  <!ENTITY imagen1 SYSTEM "imagen1.jpg">
```

En la declaración DTD anterior, se ha especificado que el valor del atributo tipo_fotografia puede ser uno de los tres valores posibles. El siguiente código XML sería válido, ya que se ajusta a la declaración DTD anterior:

```
<rio fotografia="imagen1" tipo_fotografia="JPG">
    <nombre>Ebro</nombre>
  </rio>
```

Tanto las entidades como las notaciones no suelen utilizarse demasiado en declaraciones DTD.

9.6 HERRAMIENTAS DE AYUDA

Existen distintas herramientas informáticas que facilitan la edición de documentos XML mediante el resaltado de las etiquetas, la incorporación de sangrías, la utilización de colores, etc. Algunos de estos editores de código XML, además, pueden comprobar que el documento está bien formado, ya sea una vez finalizada la edición del documento, o bien, durante la edición al mismo tiempo que se escribe el código. Igualmente, algunos programas de edición de código XML también incorporan estas mismas funcionalidades cuando se escribe el código que especifica las declaraciones DTD.

Además de las funcionalidades anteriores, existen programas de edición que permiten comprobar que un documento XML es válido con respecto a una declaración DTD. En algunos casos, esta comprobación se va haciendo, incluso, durante la edición del código del documento XML y de la declaración DTD, lo cual es muy útil. En otros casos, la comprobación de la validación DTD se realiza una vez finalizada la edición de ambos documentos. Aunque existe una gran variedad de herramientas que permiten comprobar de la validación DTD de un documento XML, se mencionan las siguientes por su sencillez y facilidad de uso: **XML Copy Editor**, **CookTop** y **Visual Studio**.

También pueden utilizarse sitios Web especializados para validación on-line de documentos XML, como pueden ser los siguientes:

- https://www.xmlvalidation.com/
- http://xmlvalidator.new-studio.org/

9.7 PANORÁMICA ACTUAL DE TECNOLOGÍAS DE VALIDACIÓN

La tecnología de validación de documentos XML más sencilla, extendida y utilizada es la tecnología de validación DTD. Sin embargo, el uso de esta técnica de validación tiene algunas limitaciones, ente las cuales se pueden destacar las siguientes:

- Una declaración DTD no es un documento XML. Es decir, no se emplea el lenguaje de marcas XML para definir las reglas de validación.
- No se pueden definir restricciones sobre los valores de elementos y atributos, en cuanto a tipo de datos, tamaño, etc.
- Solo se pueden enumerar los valores de los atributos, pero no para los elementos.
- Únicamente pueden definirse valores por defecto para los atributos, pero no para los elementos.
- Existe un control limitado en cuanto a las cardinalidades de los elementos para especificar el número de veces que pueden aparecer.
- No soporta espacios de nombres.

Estas limitaciones han hecho que surjan otras tecnologías de validación más modernas, como pueden ser las siguientes:

XML Schema

XML Schema es la evolución de la DTD descrita por el W3C, también llamado de forma más informal XSD *(XML Schema Definition)*. Es un lenguaje de esquema más potente, pero también mucho más complejo. Utiliza sintaxis XML, lo cual permite especificar de forma más detallada. Cuando se utiliza la validación XSD es posible expresar la estructura y contenido de un documento XML en términos del modelo de datos, a partir del esquema de validación definido. Esta funcionalidad, conocida como PSVI *(Post-Schema-Validation Infoset)*, se puede utilizar para transformar el documento en una jerarquía de objetos, a los cuales se puede acceder a través de un lenguaje de programación orientada a objetos. El modelo de datos de XML Schema incluye: el vocabulario (nombres de elemento y atributo), el contenido modelo (relaciones y estructura) y los tipos de datos.

RELAX NG

Es un lenguaje de esquema muy intuitivo y fácil de entender. Por estos motivos, ha crecido tanto su utilización y popularidad en la comunidad de desarrolladores de XML en los últimos años, sobre todo en comparación con la complejidad de la validación XML Schema. Tiene un alto poder expresivo y permite validar elementos intercalados que pueden aparecer en cualquier orden. La validación RELAX NG se ha convertido en un estándar ISO formando parte de la especificación ISO/IEC 19757 denominada DSDL *(Document Schema Definition Language)*.

Schematron

A diferencia de los anteriores lenguajes de validación XML, Schematron se basa en afirmaciones en lugar de basarse en el uso de una gramática. Al basarse en una serie de reglas, utiliza expresiones de acceso en lugar de expresiones gramaticales para definir la estructura de un documento XML. Si el documento XML cumple estas reglas, entonces es válido.

Este método de validación aporta flexibilidad en la descripción de estructuras relacionales, sin embargo, es un lenguaje limitado al especificar la estructura básica del documento. Este inconveniente se puede resolver combinando Schematron con otros lenguajes de esquema, como por ejemplo RELAX NG. La validación Schematron también parte de la especificación ISO/IEC 19757 DSDL *(Document Schema Definition Language)*.

Glosario de términos

Consistencia de la información. Se dice que un conjunto de información es consistente, si soporta un determinado modelo de datos, es decir, si la información cumple con ciertas normas específicas que se establecen para ese conjunto de información en particular.

Metalenguaje. En lingüística, es un lenguaje que se usa para hablar acerca de otro lenguaje o para describirlo. Al lenguaje sobre el cual se está hablando se le denomina lenguaje objeto. El metalenguaje puede ser idéntico al lenguaje objeto, por ejemplo, cuando se habla acerca del español usando el español mismo. Un metalenguaje puede ser, a la vez, el lenguaje objeto de otro metalenguaje de orden superior, y así sucesivamente. Y también, distintos metalenguajes pueden hablar acerca de diferentes aspectos de un mismo lenguaje objeto. En un sentido más general, puede referirse a cualquier terminología o lenguaje usado para hablar con referencia al mismo lenguaje. Por ejemplo, un texto sobre gramática o una discusión acerca del uso del lenguaje.

Parser XML. Es un analizador sintáctico de XML. Es un programa informático que lee un documento XML y verifica que esté bien formado. Algunos *parsers* XML también pueden comprobar que el código XML sea válido. El *parser*, analizador o procesador de XML es la herramienta principal de cualquier aplicación del lenguaje XML. Mediante su uso no solamente se puede comprobar si los documentos XML están son bien formados o son válidos, sino que también pueden ser incorporarlos a otras aplicaciones informáticas, siempre que éstas puedan manipular la información almacenada en documentos XML. Los *parsers* XML se pueden clasificar en dos grupos principales: sin validación, que son aquellos *parsers* que no validan el documento XML, sino que sólo chequean que el documento XML esté bien formado de acuerdo con las reglas de sintaxis de XML; y, con validación, que son aquellos *parsers* que además de comprobar que el documento está bien formado según las reglas, también comprueba que el documento sea válido.