Introducción

XML es, junto con HTML, uno de los lenguajes de marcas más característicos. En realidad, XML es un metalenguaje y la base para la construcción de otros lenguajes más específicos.

XML define las reglas de construcción de los documentos, pero no se especifican las etiquetas tal y como se hace en HTML. En XML, en lugar de haber elementos o atributos predefinidos, hay mecanismos para definirlos.

Al tratarse de un estándar, alrededor de XML hay un buen número de tecnologías y herramientas: mecanismos de validación, editores, programas de visualización o librerías de programación para integrarlo en desarrollos de software propios.

4.1. Introducción, evolución y estado actual

Cuando se habla de lenguajes de marcas, los dos ejemplos más citados son HTML y XML. HTML es, como se explica en este mismo libro, el lenguaje de la web: un conjunto de etiquetas comprensibles e interpretables por el navegador que posibilitan representar cualquier información para que la puedan consultar los usuarios.

El objetivo de XML es, en cambio, menos concreto. No está orientado a la web, pero puede utilizarse para presentar datos en un navegador como HTML. Tampoco está orientado a la representación de interfaces gráficas de usuario, pero se utiliza en un buen número de tecnologías para ello. También se puede utilizar para intercambiar información entre sistemas, guardar la configuración de una aplicación o difundir información mediante RSS.

El lenguaje de marcado XML (*eXtensible Markup Language*, «lenguaje de marcas extensible», en castellano) es, en realidad, un metalenguaje de propósito general desarrollado por el W3C (World Wide Consortium) y se utiliza en multitud de contextos.

Tabla 4.1. Evolución de las versiones de XML

Año	Versión	Observaciones
1998	XML 1.0	Primera edición
2000	XML 1.0	Segunda edición
2004	XML 1.0	Tercera edición
2004	XML 1.1	Primera edición
2006	XML 1.0	Cuarta edición
2006	XML 1.1	**Segunda edición**
2008	XML 1.0	**Quinta edición**

XML proviene del lenguaje GML *(Generalized Markup Language)* y de SGML *(Standard Generalized Markup Language)*. Fue **lanzado en 1998** y se encuentra en la **versión 1.1.** Es un **formato abierto**, lo que implica que se puede utilizar de manera libre y sin coste. La extensión de los ficheros XML es **.xml** y el tipo MIME es **application/text** y **text/xml.**

El W3C anima a crear los documentos en XML 1.0 salvo que se necesiten las nuevas funciones añadidas en XML 1.1. En la gran mayoría de los casos estas funciones no son necesarias, por lo que habitualmente los documentos XML se encuentran en versión 1.0.

A continuación, se muestra un ejemplo de un documento XML 1.0:

```
<?xml version="1.0" encoding="UTF-8"?>
<animal>
  <nombre>Tiburón blanco</nombre>
  <nombrecientifico>Carcharodon carcharias</nombrecientifico>
  <familia>Lamnidae</familia>
  <estadodeconservacion>Vulnerable</estadodeconservacion>
</animal>
```

Sin entrar en detalles (se tratarán en el siguiente apartado), se puede observar que el formato es sencillo: texto plano, etiquetas de apertura y cierre o estructura jerárquica son algunas de las características de este tipo de documentos.

XML tiene fundamentos robustos y una sintaxis muy simple, lo que lo convierte en una herramienta muy versátil y que goza de una gran aceptación. XML y sus dialectos se utilizan en múltiples contextos; creación de ficheros de configuración, diseño de interfaces gráficas, publicación de contenidos, intercambio de información entre aplicaciones o diseño de páginas web son solo algunas de las áreas de aplicación en las que se emplea.

A pesar de la sencillez de XML, existe un importante conjunto de tecnologías alrededor que convierten el lenguaje es una herramienta poderosa: validadores, transformadores, editores... La estandarización a la que se somete permite que el ecosistema de aplicaciones relacionadas con el lenguaje sea uno de sus activos más importantes.

Nota técnica

Un metalenguaje es una base para crear lenguajes. No tiene etiquetas concretas ni un ámbito de aplicación definido. Un metalenguaje contiene reglas para ser la base de la creación de lenguajes robustos. XHTML, por ejemplo, es un lenguaje basado en el metalenguaje XML.

4.2. Estructura y sintaxis de XML

Los documentos XML se estructuran en dos partes principales: **prólogo** y **cuerpo.** El prólogo contiene información relativa al conjunto del documento y el cuerpo recoge los elementos con la información propiamente dicha. También permite incluir comentarios que no serán procesados.

4.2.1. Declaración

La declaración del documento XML se realiza en el prólogo, siendo este opcional. Si existe, debe ser la primera línea del documento. En él se indica que el documento es XML, la versión y la codificación utilizada mediante los siguientes atributos:

- versión. Versión de XML.
- encoding. Codificación utilizada en el documento.
- standalone. Indica si el documento es independiente o si existe un DTD externo para realizar la validación. Admite los valores **yes** (es independiente) y **no** (está asociado a un DTD de validación). Este atributo es opcional.

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

Nota técnica

Un DTD *(Document Type Definition)* contiene reglas que determinan que un documento XML es válido en un contexto determinado. En el Apartado 4.3. se explica en detalle qué son y cómo se construyen las validaciones.

4.2.2. Estructura, sintaxis, elementos y atributos

El cuerpo del documento XML está compuesto por elementos. Los elementos son el componente fundamental de XML. Se parecen mucho a los elementos de HTML, pues tienen una etiqueta de apertura y otra de cierre, que en el caso de XML es obligatoria.

Las etiquetas de apertura tienen un nombre y, opcionalmente, uno o más atributos separados por espacios. La etiqueta de cierre debe tener el mismo nombre que la de apertura, pero este estará precedido por el símbolo /.

```
<nombre-elemento nombre-atributo-1="valor1" nombre-atributo-2="valor2" >
</nombre-elemento>
```

Las reglas para la asignación de nombres de elementos son las siguientes:

- Diferencian entre mayúsculas y minúsculas.
- Deben comenzar con una letra o un guion bajo.
- Los nombres de elementos tienen que ser idénticos en las etiquetas de apertura y de cierre.
- Los nombres pueden estar formados por caracteres alfanuméricos, guiones, guiones bajos y puntos.
- Los nombres no pueden contener espacios (interpreta como nombre del elemento la primera palabra y el resto como parámetros).

Por ejemplo, el siguiente elemento generará un error al validarlo con un navegador web:

```
<1color>Azul</1color>
```

Figura 4.1. *Con XML se garantiza que los documentos estén bien formados.*

También serán erróneos los siguientes elementos:

```
<color>Azul</Color>
```

```
<-color>Azul</-color>
```

```
<.color>Azul</.color>
```

```
<el color>Azul</el color>
```

En cambio, los siguientes elementos serán procesados correctamente:

```
<color1>Azul</color1>
```

```
<_color>Azul</_color>
```

```
<COLOR>Azul</COLOR>
```

```
<el_color>Azul</el_color>
```

Las reglas de creación de atributos se muestran a continuación:

- Deben tener asignado un valor.
- Los valores siempre van entrecomillados, admitiendo comillas simples y dobles (deben coincidir el tipo de comilla de cierre con el de apertura).
- Diferencian entre mayúsculas y minúsculas.

- Deben comenzar con una letra o un guion bajo.
- Pueden estar formados por caracteres alfanuméricos, guiones, guiones bajos y puntos.

Ejemplo de atributo válido:

```
<nombre familia="Lamnidae">Tiburón blanco</nombre>
```

En XML se puede caracterizar un elemento mediante atributos o elementos hijos. No existe una regla para determinar en qué caso se deben utilizar uno u otro, pero sí casuísticas en la que los atributos no son suficientes para crear la estructura de datos que se desea.

Los elementos pueden contener texto u otros elementos. En el segundo caso, la relación entre los elementos es jerárquica: un elemento puede contener a otros elementos y estos, a su vez, ser contenedores de otros más. Esta estructura se denomina «de árbol» y se caracteriza por tener un único elemento en la parte alta de la jerarquía (denominado *raíz*) que se ramifica hasta llegar a los nodos finales u hojas.

Figura 4.2. *Los documentos XML son estructuras jerárquicas en forma de árbol.*

Las relaciones entre los elementos de un documento XML se denominan de la siguiente manera:

- Existe un único elemento raíz (**root**).
- Cada uno de los elementos descendientes directos de un elemento se llama *hijo* (**children**).
- El elemento ascendiente directo de un elemento se llama *padre* (**parent**).
- Los elementos que tienen un padre común se denominan *hermanos* (**sibling**).

En el ejemplo que se muestra a continuación las relaciones entre los elementos son las siguientes:

- **<animales>** es el elemento raíz.
- Los elementos **<animal>** son hijos (**children**) del elemento **<animales>**.
- El elemento **<animales>** es padre (**parent**) de los elementos **<animal>**.

- Los elementos **<animal>** son hermanos (**sibling**) entre sí.
- Los elementos **<nombre>** y **<nombrecientifico>** son hijos del elemento **<animal>**.

```
<?xml version="1.0" encoding="UTF-8"?>
<animales>
  <animal id="1">
    <nombre>Tiburón blanco</nombre>
    <nombrecientifico>Carcharodon carcharias</nombrecientifico>
  </animal>
  <animal id="2">
    <nombre>Atún rojo</nombre>
    <nombrecientifico>Thunnus thynnus</nombrecientifico>
extinción</estadodeconservacion>
  </animal>
</animales>
```

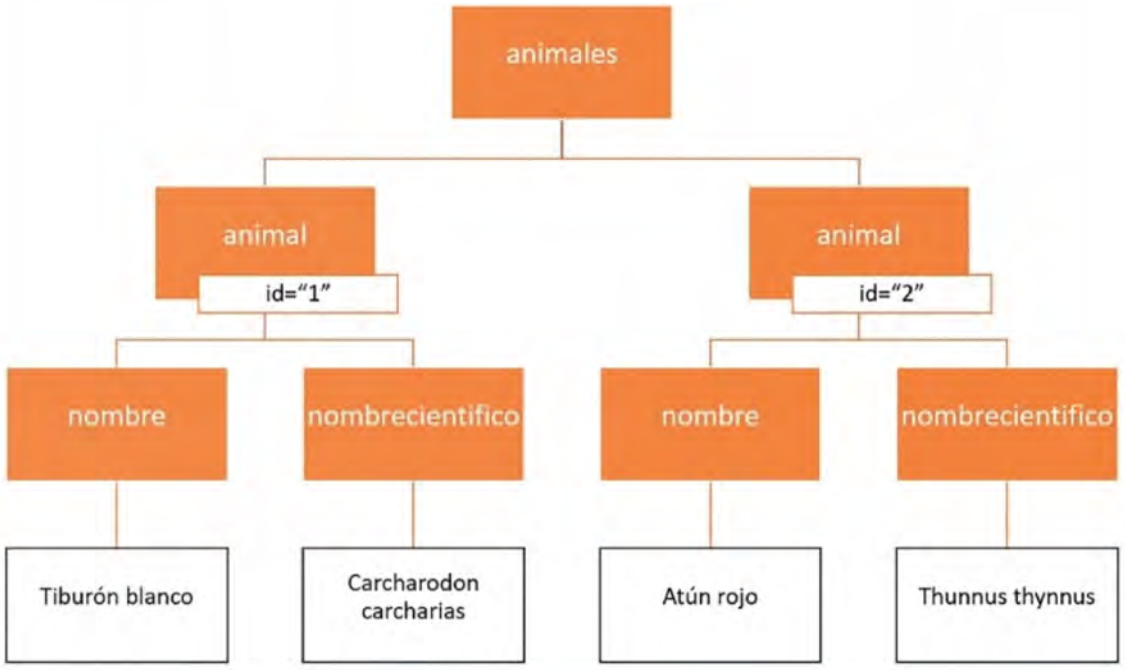

Figura 4.3. *Las relaciones entre los elementos de un documento XML se asimilan con relaciones parentales.*

Actividad resuelta 4.1

Representación de un *curriculum vitae*

Representar en un documento XML un *curriculum vitae*. Los datos que debe contener son los siguientes:

- Nombre y apellidos.
- Año de nacimiento.
- Residencia:
  - Ciudad.
  - País.

- Datos de contacto:
  - Teléfono.
  - Correo electrónico.
  - Perfil LinkedIn.
- Breve descripción general.
- Listado de competencias, indicando el nombre y el nivel.
- Formación académica, con la titulación, el centro de formación y el año de titulación.
- Experiencia profesional, especificando el puesto, la empresa, el año de inicio y el año de finalización.

Solución

```
<?xml version="1.0" encoding="UTF-8"?>
<cv>
    <nombre>Shigeru Miyamoto</nombre>
    <anyo-nacimiento>1952</anyo-nacimiento>
    <residencia>
        <ciudad>Kioto</ciudad>
        <pais>Japón</pais>
    </residencia>
    <contacto>
        <telefono>+81-555-11-33-66</telefono>
        <email>shigeru@servidordecorreo.jp</email>
        <linkedin>http://www.linkedin.com/miyamoto</linkedin>
    </contacto>
    <descripcion>
    Gran capacidad creativa. Interesado en el diseño de videojuegos.     </
descripcion>
    <competencias>
        <competencia nombre="Diseño" nivel="Alto"/>
        <competencia nombre="Trabajo en equipo" nivel="Medio"/>
        <competencia nombre="Música" nivel="Alto"/>
    </competencias>
    <formacion>
        <titulacion nombre="Diseño industrial" centro="Colegio Municipal de Artes
Industriales de Kanazawa" anyo="1975"/>
    </formacion>
    <experiencia-profesional>
        <experiencia>
            <puesto>Diseñador de juegos</puesto>
            <empresa>Nitendo</empresa>
            <inicio>1977</inicio>
            <fin></fin>
        </experiencia>
    </experiencia-profesional>
</cv>
```

Actividad propuesta 4.1

Representación de un viaje turístico

Representa en un documento XML un viaje turístico. Los datos que debe contener son los siguientes:

- Nombre y apellidos de los clientes.
- Fecha de inicio.
- Fecha de fin.
- Origen del viaje.
- Una relación de destinos, cada uno de los cuales contendrá la siguiente información:
  - Nombre de la ciudad.
  - Nombre del hotel.
  - Número de noches.
- Precio del viaje.

4.2.3. Comentarios

XML admite incluir comentarios en el interior de los documentos. Están formados por textos delimitados por los caracteres de apertura **<!--** y de cierre **-->**.

Los comentarios se escriben para ser leídos por personas, por lo que el contenido no tiene que cumplir con ninguna sintaxis ni regla; contiene un texto en lenguaje natural que explica, aclara o informa de algo relevante para la comprensión del documento por parte de la persona que lo está leyendo. No son procesados, ni validados, ni participan de la estructura de datos que representa el documento.

```
<!-- Este texto es un comentario -->
```

4.2.4. Espacios de nombres

El **espacio de nombres** o **namespace** es un identificador que se utiliza para resolver las ambigüedades que podrían surgir en caso de que haya elementos o atributos con el mismo nombre.

Por ejemplo, dos vocabularios de XML podrían utilizar el elemento <autor>. Si esos dos vocabularios se utilizan en el mismo documento, surge la pregunta siguiente: ¿cómo saber cuando se encuentra un elemento <autor> a qué vocabulario de XML corresponde?

Es para problemas como este cuando el espacio de nombres surge y tiene utilidad, ya que permitiría relacionar cada uno de los elementos <autor> con un «cualificador» distinto (el espacio de nombres) que resolvería de forma inequívoca la confusión al referenciar a dicho elemento.

La declaración de un espacio de nombres se realiza utilizando el atributo reservado **xmlns**, seguido del nombre del espacio de nombres y una URL:

```
xmlns:nombre-espacio-nombres=URL
```

En el ejemplo referenciado anteriormente, se podrían declarar dos espacios de nombres para los dos vocabularios utilizados:

```
xmlns:autorestecnicos='http://unaurlcualquiera';
xmlns:autoresliterarios='http://otraurlcualquiera';
```

La ambigüedad se resolvería referenciando al elemento <autor> incluyendo su **namespace** de la siguiente manera: <autorestecnicos:autor>.

4.2.5. Entidades

Las entidades XML son un mecanismo para representar información dentro de un documento haciendo referencia a ella en lugar de incluirla directamente. Esta información puede ser desde un carácter predefinido hasta un fichero externo. Gracias a las entidades se puede repartir el contenido de los documentos en varias fuentes.

Esta fragmentación aporta una mejor estructura, facilita el trabajo en grupo y permite la reutilización, ya que las entidades pueden ser referenciadas desde diversos documentos principales. De cara al proceso no tiene consecuencias, pues el programa que procese el XML lo hará sobre el conjunto total de información, como si fuese una única unidad.

Las entidades se definen en el DTD. Al declarar una entidad se asigna un alias a un bloque de texto. Ese alias se puede adjuntar a un documento XML para incrustar el bloque correspondiente.

Por ejemplo, es posible crear una entidad que almacene la ficha de una empresa (nombre, dirección y CIF) e incrustarla en varios documentos sin necesidad de escribir el mismo texto una y otra vez. En caso de que la empresa cambie de dirección, solo habrá que cambiarla en un punto.

Existen categorías principales de entidades en XML: las **entidades generales** y las **entidades de parámetros.** La entidad general es un alias que se convierte en parte del documento XML. La entidad de parámetros se convierte, en cambio, en parte del DTD.

Las entidades generales se dividen, a su vez, en dos tipos: **internas** y **externas.** Las entidades internas mantienen sus valores en la propia declaración de la entidad. Las entidades externas mantienen los valores en un archivo externo.

Existen cinco entidades internas predefinidas de uso común. Representan caracteres que provocarían ambigüedad en el procesado del XML, ya que tienen un significado estructural. Se muestran en la Tabla 4.2.

Tabla 4.2. Entidades predefinidas

Entidad	Significado	Carácter
<	Símbolo de «menor que»	<
>	Símbolo de «mayor que»	>

Entidad	Significado	Carácter
&amp;	Símbolo del *ampersand* (en español, *et*)	&
&apos;	Comilla simple	'
&quot;	Comilla doble	"

El siguiente documento XML de ejemplo contiene dos errores:

- El símbolo & en el valor del atributo **nombre** se interpreta como el comienzo de una entidad.
- Las comillas que rodean la palabra **sueño** en el valor del atributo **mensaje** se confunden con las comillas que delimitan la cadena de caracteres.

```
<?xml version="1.0" encoding="UTF-8"?>
<ficha>
    <empresa nombre="Toledo & Kurylenko S.L"></empresa>
    <eslogan mensaje="Vendemos "sueños" imposibles"></eslogan>
</ficha>
```

Al tratar de abrirlo desde un navegador, se detendrá al detectar el primero de los errores (los procesadores de XML de los navegadores suelen parar en el momento en el que encuentran el primer error durante el análisis del documento) (Figura 4.4).

Figura 4.4. *Los navegadores detectan si el documento XML está mal formado.*

En ambos casos, la solución consiste en sustituir los caracteres problemáticos por las entidades correspondientes:

```
<?xml version="1.0" encoding="UTF-8"?>
<ficha>
    <empresa nombre="Toledo &amp;amp; Kurylenko S.L"></empresa>
    <eslogan mensaje="Vendemos &amp;quot;sueños&amp;quot; imposibles"></eslogan>
</ficha>
```

```
-<ficha>
    <empresa nombre="Toledo & Kurylenko S.L"/>
    <eslogan mensaje="Vendemos "sueños" imposibles"/>
  </ficha>
```

Figura 4.5. *Las entidades permiten representar caracteres problemáticos.*

4.2.6. CDATA

En XML una sección CDATA contiene un conjunto de caracteres que no debe ser tratado por el analizador. Las secciones CDATA se utilizan habitualmente para almacenar código XML o HTML que, de otra forma, impediría que el documento XML contenedor no estuviese bien formado.

Nota técnica

Se dice que un documento XML está bien formado cuando es conforme a la sintaxis de XML. Este y otros aspectos formales relacionados con la validación de documentos XML se tratan en el Apartado 4.3.

En una sección CDATA no es necesario sustituir los caracteres conflictivos por entidades, ya que son ignorados por el analizador. Si se incluyen etiquetas XML dentro de una sección CDATA, no serán tratadas como tal, sino como una sucesión de caracteres sin ningún sentido específico.

Las secciones CDATA comienzan por la secuencia de caracteres:

```
<![CDATA[
```

Terminan por la secuencia:

```
]]>
```

En el siguiente código de ejemplo se muestra un documento XML que incluye código HTML. Dentro del código HTML hay un elemento <br> que no cumple con las reglas de XML, pues no tiene etiqueta de cierre. Por lo tanto, es necesario incluir el código HTML como una sección CDATA para que el documento XML esté bien formado.

```
<?xml version="1.0" encoding="UTF-8"?>
<ficha>
    <nombre>Lenguaje C</nombre>
    <info-html>
        <![CDATA[
            <h1>LENGUAJES DE PROGRAMACIÓN<h1>
            <hr>
            <a href="url-enlace">C</a>
        ]]>
    </info-html>
</ficha>
```

Como se puede suponer, la información contenida en secciones CDATA no forma parte de la estructura jerárquica del documento, aunque se trate de elementos XML. El elemento CDATA se tratará como tal, considerándose una única unidad de la estructura.

4.3. Validación de XML

XML es un lenguaje con una sintaxis muy simple pero dotado de un importante y muy potente conjunto de herramientas. Entre las más destacadas están las destinadas a garantizar la correcta estructura de los documentos. XML permite determinar las reglas que debe cumplir un determinado documento escrito en este lenguaje. Estas reglas, junto con los correspondientes procesadores de XML, permiten garantizar que un documento y los datos almacenados en él están correctamente construidos.

4.3.1. XML válido y XML bien formado

Una de las características de un metalenguaje como XML es que no tiene etiquetas predeterminadas. Esto significa que, en lugar de listado de etiquetas con sus significados XML, proporciona una infraestructura sobre la que crear etiquetas y dotarlas de significado. No obstante, hay ciertas reglas generales de diseño que hay que cumplir y que dotan de robustez a los lenguajes creados con XML:

- El documento tiene que estar **bien formado** o, lo que es lo mismo, es **conforme la sintaxis** de XML.
- El documento tiene que ser **válido** o, lo que es lo mismo, **conforme a las reglas** definidas en su DTD *(Document Type Definition)* o XML Schema, aunque estas definiciones de reglas semánticas son opcionales.

Para que un documento esté **bien formado**, tiene que cumplir con las siguientes reglas:

- Debe haber uno y solo un elemento raíz.
- Todos los elementos deben estar cerrados.
- Los elementos tienen que estar anidados correctamente: no se pueden intercalar aperturas y cierres de elementos distintos.
- Todos los valores de los atributos están entrecomillados.
- Los nombres de elementos y atributos han de cumplir con sus respectivas reglas.

Para que un documento sea **válido** tiene que existir una definición que contenga las reglas semánticas que validar. Estas reglas se definen utilizando DTD o XML Schema y se explicarán posteriormente.

Los navegadores y la mayoría de los programas que permiten procesar documentos XML validan que estén bien formados y sean válidos. Si el documento contiene algún error, muestran un mensaje que lo indica y señala su ubicación.

Actividad propuesta 4.2

Documento bien formado

Haz los cambios necesarios en el siguiente documento XML para que esté bien formado:

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

```
    <titulo anyo="2017>Fortnite</titulo>
    <empresa>Epic Games</empresas>
    <motor>Unreal Engine</Motor>
</videojuego>
```

4.3.2. DTD

Se dice que un documento es **válido** si, además de estar **bien formado**, cumple con todas las reglas descritas en un DTD determinado.

Como se ha comentado anteriormente, DTD son las siglas de *Document Type Definition* («definición de tipo de documento», en castellano). Un DTD contiene una serie de reglas que sirven para validar un documento XML. En ningún caso es obligatorio utilizar un DTD para trabajar con documentos XML, pero es recomendable en muchas ocasiones.

Las reglas contenidas en un DTD tienen que ver con la estructura del documento y los elementos y atributos que puede contener. Es un mecanismo relativamente sencillo de usar, aunque tiene algunas limitaciones. Estas limitaciones se pueden superar utilizando XML Schema, otra herramienta de validación de documentos XML.

DTD interna y externa

Existen dos maneras de asociar un DTD a un documento XML: de forma **interna** o **externa.**

En el DTD interno, las reglas se definen dentro del propio documento XML. En el DTD externo, las reglas se definen en un documento externo enlazado con el documento XML.

El DTD interno tiene como ventaja que tanto los datos como las reglas de validación se encuentran en un único documento, lo que en cierta medida puede resultar una alternativa más sencilla y compacta.

El DTD interno se crea incluyendo un elemento de tipo DOCTYPE inmediatamente después de la declaración del documento XML. Este elemento tiene la siguiente sintaxis:

```
<!DOCTYPE nombre-elemento-raíz[
        Elementos-y-sus-relaciones
]>
```

Siendo **nombre-elemento-raíz** el mismo nombre que el del elemento raíz del documento y **Elementos-y-sus-relaciones** el conjunto de elementos y cómo se estructuran y anidan, construyendo así una declaración formal de cómo debe ser un documento XML para ser acorde a un DTD y, por lo tanto, válido.

En el siguiente código se puede observar un DTD interno:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE aviso [
```

```
  <!ELEMENT aviso (de,para,mensaje)>
  <!ELEMENT de (#PCDATA)>
  <!ELEMENT para (#PCDATA)>
  <!ELEMENT mensaje (#PCDATA)>
]>
<aviso>
    <de>David</de>
    <para>Rosalía</para>
    <mensaje>Mañana nos vemos en el estudio a las 10.</mensaje>
</aviso>
```

Por su parte, el DTD externo facilita la reutilización y permite compartir las reglas de validación para utilizarlas, por ejemplo, en procesos de intercambio de información entre sistemas. De manera general se recomienda, por lo tanto, utilizar DTD externos.

La vinculación de un DTD a un documento XML se realiza añadiendo una referencia al fichero que contiene el DTD en este último. La sintaxis es la siguiente:

```
<!DOCTYPE nombre-elemento-raíz SYSTEM "URI">
```

siendo **nombre-elemento-raíz** el mismo nombre que el del elemento raíz del documento y **URI** la dirección en la que se encuentra el fichero con el DTD.

En el siguiente código se muestra el documento XML en el que se referencia a un DTD externo almacenado en la misma ubicación que el fichero XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE aviso SYSTEM "dtdExterno.dtd">
<aviso>
    <de>David</de>
    <para>Rosalía</para>
    <mensaje>Mañana nos vemos en el estudio a las 10.</mensaje>
</aviso>
```

El documento referenciado, de nombre **dtdExterno.dtd**, contiene los elementos que definen la estructura que debe tener el documento XML:

```
<!ELEMENT aviso (de,para,mensaje)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT mensaje (#PCDATA)>
```

Como se puede observar, los elementos de definición del DTD son idénticos en la declaración interna y en la externa. Únicamente cambia la ubicación de dichos elementos y, por consecuencia, la forma en la que se hace uso de estos.

Entidades en los DTD

En un DTD, una entidad es el equivalente a una constante en programación. Cuando se define una entidad, se le asigna un nombre y un valor que sustituirá a las referencias que se hagan a dicho nombre. Una de las categorizaciones divide a las entidades en **internas** y **externas.**

La sintaxis de la declaración de una **entidad interna** es la siguiente:

```
<!ENTITY nombre-entidad "texto de reemplazo">
```

- nombre-entidad: es el alias que se asigna a la entidad.
- «texto de reemplazo»: es cualquier texto (entrecomillado) que sustituirá a las referencias a la entidad.

La sintaxis para referenciar a una entidad interna en el documento XML es la siguiente:

```
&nombre-entidad;
```

En el ejemplo que se muestra a continuación se define una entidad población con texto asociado. Además, se hacen dos referencias en el cuerpo del documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fichas [
  <!ENTITY poblacion "Tejeda de Tiétar, Cáceres">
]>
<fichas>
    <ficha>
        <empresa nombre="Tractores Toledo"></empresa>
        <direccion>&poblacion;</direccion>
    </ficha>
    <ficha>
        <empresa nombre="Congelados Kurylenko"></empresa>
        <direccion>&poblacion;</direccion>
    </ficha>
</fichas>
```

Recuerda

Existen entidades predefinidas que se utilizan para representar caracteres que tienen significado propio en XML y que pueden generar confusión en el análisis.

El resultado al abrir el documento en el navegador muestra la sustitución de las referencias a la entidad por el texto asociado:

```
−<fichas>
 −<ficha>
    <empresa nombre="Tractores Toledo"/>
    <direccion>Tejeda de Tiétar, Cáceres</direccion>
  </ficha>
 −<ficha>
    <empresa nombre="Congelados Kurylenko"/>
    <direccion>Tejeda de Tiétar, Cáceres</direccion>
  </ficha>
</fichas>
```

Figura 4.6. *Las entidades permiten la reutilización de los datos.*

Por su parte, las **entidades externas** se definen fuera del DTD. Se dividen en **públicas** y **privadas.**

Las **entidades externas públicas** tienen la siguiente sintaxis:

```
<!ENTITY nombre-entidad PUBLIC "id-publico" "URI/URL">
```

- **nombre-entidad:** es el alias que se asigna a la entidad.
- **PUBLIC:** palabra clave que indica que la entidad es pública.
- **id-publico:** es el identificador público.
- **URI/URL:** es el identificador o la ubicación del recurso.

Las **entidades externas privadas** o de sistema, por su parte, tienen la siguiente sintaxis:

```
<!ENTITY nombre-entidad SYSTEM "URI/URL">
```

- **nombre-entidad:** es el alias que se asigna a la entidad.
- **SYSTEM:** palabra clave que indica que la entidad es privada.
- **URI/URL:** es el identificador o la ubicación del recurso.

En el siguiente ejemplo se muestra la aplicación de entidades internas y externas:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE fichas [
  <!ENTITY poblacion "Tejeda de Tiétar">
  <!ENTITY provincia SYSTEM "provincia.txt">
  <!ENTITY detalle SYSTEM "ficha.xml">
]>
<grupo>
  &detalle;
  <fichas>
    <ficha>
      <empresa nombre="Tractores Toledo"></empresa>
     <poblacion>&poblacion;</poblacion>
```

```
      <provincia>&provincia;</provincia>
    </ficha>
    <ficha>
      <empresa nombre="Congelados Kurylenko"></empresa>
      <poblacion>&poblacion;</poblacion>
      <provincia>&provincia;</provincia>
    </ficha>
  </fichas>
</grupo>
```

Las entidades externas tienen referencias a sendos ficheros escritos en texto sin formato (provincia.txt) y en formato XML (ficha.xml). Son los siguientes:

provincia.txt:

```
Cáceres
```

ficha.xml:

```
<distribuidores>
    <codigo>10834</codigo>
    <descripcion>Distribuidores oficiales</descripcion>
</distribuidores>
```

El resultado al abrir el fichero en un navegador es:

```
▼<grupo>
  ▼<fichas>
    ▼<ficha>
       <empresa nombre="Tractores Toledo"/>
       <poblacion>Tejeda de Tiétar</poblacion>
       <provincia/>
     </ficha>
    ▼<ficha>
       <empresa nombre="Congelados Kurylenko"/>
       <poblacion>Tejeda de Tiétar</poblacion>
       <provincia/>
     </ficha>
   </fichas>
 </grupo>
```

Figura 4.7. *Las entidades externas admiten múltiples formatos.*

Como se puede observar, la inclusión de las entidades externas no se ha realizado por restricciones incorporadas a los navegadores por razones de seguridad.

En cambio, al abrir el fichero desde el programa XML Notepad, se puede observar que las entidades externas se han incluido en el fichero.

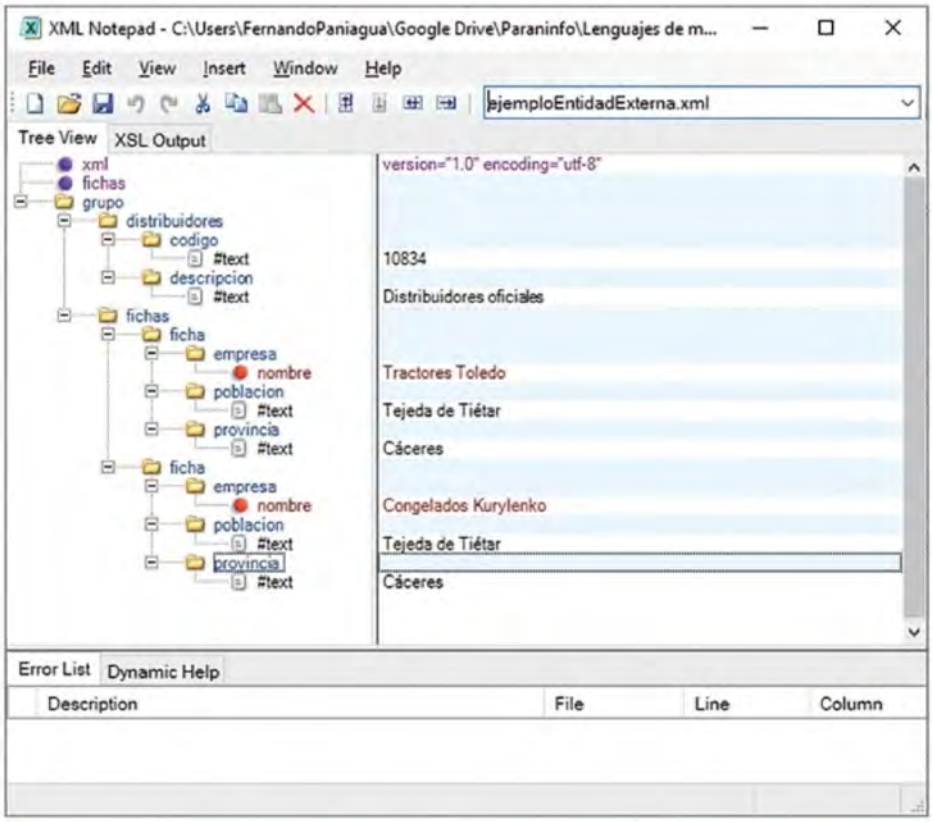

Figura 4.8. *Solo algunos programas procesan las entidades externas.*

Nota técnica

Existe un tipo de ataque informático llamado «ataque de entidad externa XML», que se realiza sobre la aplicación que analiza un fichero XML que contiene una referencia a una entidad externa.

Anotaciones

Las anotaciones (**notation**) se utilizan para identificar el formato de entidades que no son XML y que, por lo tanto, no se van a procesar, o como valor válido para un atributo. Pueden ser públicas o privadas.

La sintaxis general es cualquiera de las siguientes:

<!NOTATION nombre SYSTEM "URI">

<!NOTATION nombre PUBLIC "id-público" >

<!NOTATION nombre PUBLIC "id-público" "URI">

Ejemplo:

```
<!NOTATION gif PUBLIC "GIF">
```

Elementos

Al incluir un elemento en un DTD, se está indicando en qué condiciones debe encontrarse dicho elemento en el documento XML que se está validando.

Por ejemplo, si en un DTD se incluye la siguiente declaración:

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

Se está indicando que el documento XML debe tener un elemento <de> que contiene una cadena de caracteres.

La declaración de elementos en un DTD se realiza según la siguiente sintaxis:

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

siendo **nombre-elemento** el nombre del elemento referenciado, y **contenido** la especificación del tipo contenido válido para dicho elemento.

Los posibles tipos de **contenido** son los que se detallan en la Tabla 4.3.

Tabla 4.3. Posibles tipos de contenido de los elementos

Tipo de contenido	Descripción
EMPTY	Indica que el elemento referenciado debe estar vacío.
ANY	Señala que el elemento referenciado puede contener cualquier contenido.
(#PCDATA)	Informa de que el elemento referenciado puede tener datos de tipo carácter *(Parsed Character Data).*
(*nombreElemento*)	Indica que el elemento referenciado puede contener al elemento indicado.
(*nombreElemento1, nombreElemento2, ...*)	Señala que el elemento referenciado puede contener una sucesión de elementos indicados como una lista separada por comas.

En las relaciones entre elementos se pueden especificar cardinalidades para determinar cuántos elementos pueden estar contenidos dentro del elemento referenciado. Las opciones son las que se recogen en la Tabla 4.4.

Tabla 4.4. Cardinalidades en las relaciones

Notación	Descripción	Ejemplo
(*nombreElemento*)	Una única ocurrencia del elemento.	<!ELEMENT aviso (de)>
(*nombreElemento?*)	Cero o una única ocurrencia del elemento.	<!ELEMENT aviso (de?)>

Notación	Descripción	Ejemplo
(*nombreElemento+*)	Una o más ocurrencias del elemento.	<!ELEMENT aviso (mensaje+)>
(*nombreElemento**)	Cero o más ocurrencias del elemento.	<!ELEMENT aviso (mensaje*)>
(*nombreElemento1, nombreElemento2,...*)	Debe contener todos los elementos de la lista.	<!ELEMENT aviso (de, para, mensaje)>
(nombreElemento1 | nombreElemento2)	Debe contener uno u otro elemento.	<!ELEMENTO aviso (de | para, mensaje)>

Estas opciones para la declaración de relaciones entre elementos se pueden combinar y agrupar con paréntesis para asignar las cardinalidades. Por ejemplo, se puede declarar un elemento de la siguiente manera:

```
<!ELEMENT aviso (#PCTADA | mensaje | nota | correo | llamada)*>
```

El significado para el ejemplo anterior es el siguiente:

Un aviso puede tener cero o más ocurrencias del tipo texto libre, elemento mensaje, elemento nota, elemento correo o elemento llamada.

A continuación, se expone un ejemplo de un DTD interno en un documento XML que es válido respecto a dicho DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cv [
  <!ELEMENT cv (nombre, direccion, telefono, fax?, email+, idiomas)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT direccion (#PCDATA)>
  <!ELEMENT telefono (#PCDATA)>
  <!ELEMENT fax (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
  <!ELEMENT idiomas (idioma*)>
  <!ELEMENT idioma (#PCDATA)>
]>
<cv>
    <nombre>Desmond Miles</nombre>
    <direccion>Black Hills, Dakota del Sur, Estados Unidos</direccion>
    <telefono>+1-555-73-66-58</telefono>
    <email>desmond.miles@servidordeprueba.org</email>
    <email>desmond.miles@otroservidor.fr</email>
    <idiomas>
        <idioma>Francés</idioma>
        <idioma>Inglés</idioma>
        <idioma>Italiano</idioma>
    </idiomas>
</cv>
```

La explicación de las condiciones de validación del DTD anterior se muestran en la Tabla 4.5.

Tabla 4.5. Significado de las reglas del DTD de ejemplo

Definición	Descripción
<!DOCTYPE cv	Indica que **cv** es el elemento raíz del documento.
<!ELEMENT cv (nombre, direccion, telefono, fax?, email+, idiomas)>	Señala que el elemento **cv** tiene que contener los siguientes elementos: un **nombre,** una **dirección,** un **teléfono,** opcionalmente un **fax,** uno o más **email** y un **idiomas.**
<!ELEMENT nombre (#PCDATA)>	Informa de que el elemento **nombre** puede contener texto.
<!ELEMENT direccion (#PCDATA)>	Indica que el elemento **direccion** puede contener texto.
<!ELEMENT telefono (#PCDATA)>	Señala que el elemento **telefono** puede contener texto.
<!ELEMENT fax (#PCDATA)>	Indica que el elemento **fax** puede contener texto.
<!ELEMENT email (#PCDATA)>	Informa de que el elemento **email** puede contener texto.
<!ELEMENT idiomas (idioma*)>	Señala que el elemento **idiomas** opcionalmente (podría no tener ninguno) puede tener varios elementos **idioma.**
<!ELEMENT idioma (#PCDATA)>	Indica que el elemento **idioma** puede contener texto.

Respecto a los atributos, los DTD también permiten incluir reglas y condiciones que determinen si un documento XML es válido. Se declaran a nivel de elemento. La sintaxis es la siguiente:

```
<!ATTLIST nombre-elemento nombre-atributo tipo-atributo valor-atributo>
```

Ejemplo:

```
<!ATTLIST direccion calle CDATA "Domicilio desconocido">
```

La lista de los tipos de atributos más utilizados se recoge en la Tabla 4.6.

Tabla 4.6. Tipos de atributos

Tipo	Descripción
CDATA	Cadena de caracteres.
(*valor1* | *valor2* | ...)	Una lista de posibles valores.
ID	Un identificador único.
IDREF	Una referencia a un identificador único de otro elemento.

Tipo	Descripción
IDREFS	Una lista de referencias separadas por espacios a identificadores de otros elementos.
NMTOKEN	Un nombre XML válido.
NMTOKENS	Una lista de nombres XML válidos separados por espacios.
ENTITY	Una referencia a una entidad.
ENTITIES	Una referencia a un conjunto de entidades.
NOTATION	Un nombre de una notación.
xml:lang	Indica el idioma del contenido.
xml:space	Señala si se han de respetar los espacios, las tabulaciones y los retornos de carro múltiples (valor «preserve») o eliminarlos (valor «default»).

La lista de valores posibles se puede consultar en la Tabla 4.7.

Tabla 4.7. Valores posibles para los atributos

Valor	Descripción
valor	El valor por defecto del atributo.
#REQUIRED	Indica que el atributo es obligatorio.
#IMPLIED	Señala que el atributo es opcional.
#FIXED *valor*	El valor del atributo.

Por ejemplo, la siguiente declaración de atributos indica que el elemento **idioma** tiene un atributo **nivel** que puede tomar los valores **bajo**, **medio** y **alto**, asignándosele por defecto el valor **medio**.

```
<!ATTLIST idioma nivel (bajo | medio | alto) "medio">
```

Aplicado al siguiente fragmento de XML:

```
<idiomas>
    <idioma nivel="bajo">Francés</idioma>
    <idioma>Inglés</idioma>
    <idioma>Italiano</idioma>
</idiomas>
```

genera la salida en el navegador que se recoge en la Figura 4.9.

```
-<cv>
   <nombre>Desmond Miles</nombre>
   <direccion>Black Hills, Dakota del Sur, Estados Unidos</direccion>
   <telefono>+1-555-73-66-58</telefono>
   <email>desmond.miles@servidordeprueba.org</email>
   <email>desmond.miles@otroservicor.fr</email>
  -<idiomas>
     <idioma nivel="bajo">Francés</idioma>
     <idioma nivel="medio">Inglés</idioma>
     <idioma nivel="medio">Italiano</idioma>
   </idiomas>
 </cv>
```

Figura 4.9. *Se puede definir un valor por defecto para los atributos.*

Como se puede observar en la Figura 4.9, aquellos elementos **idioma** a los que no se les ha asignado atributos **nivel** se les asigna de manera automática, al estar indicado un valor por defecto en la especificación del atributo en el DTD.

Si se modifica la declaración del atributo en el DTD sustituyendo el valor por defecto por #REQUIRED, cuando se realice una validación del documento se mostrarán tantos errores como elementos **idioma** carezcan del atributo **nivel** (Figura 4.10).

```
<!ATTLIST idioma nivel (bajo | medio | alto) #REQUIRED>
```

```
19 <idiomas>
20     <idioma nivel="bajo">Francés</idioma>
   Attribute "nivel" is required and must be specified for element type "idioma".
21     <idioma>Inglés</idioma>
22     <idioma>Italiano</idioma>
23 </idiomas>
```

Figura 4.10. *Si el atributo obligatorio no existe, el procesador muestra un error.*

En paralelo, además de validar que los elementos y atributos existen y están en el orden preciso, se valida que no hay elementos o atributos no declarados en el DTD.

En la Figura 4.11 se muestra un documento XML con el DTD interno. El editor utilizado, al realizar la validación, marca errores en las líneas 9, 11, 13, 14 y 15.

Los errores de la línea 9 y 11 tienen el mismo origen: el elemento **categoria** no está declarado como elemento hijo del elemento **cv**. Esto provoca por un lado que el elemento **cv** no esté bien construido, provocando un error. Por otra parte, el elemento **categoria** no aparece como declarado en ningún lugar, por lo que provoca otro error.

Los errores de las líneas 13, 14 y 15 tienen el mismo origen: el atributo **código** no está definido como atributo del elemento **idioma**.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE cv [
3   <!ELEMENT cv (nombre, idiomas)>
4   <!ELEMENT nombre (#PCDATA)>
5   <!ELEMENT idiomas (idioma*)>
6   <!ELEMENT idioma (#PCDATA)>
7   <!ATTLIST idioma nivel (bajo | medio | alto) "medio">
8 ]>
9 <cv>
10     <nombre>Desmond Miles</nombre>
11     <categoria>Senior</categoria>
12     <idiomas>
13         <idioma codigo="fr" nivel="bajo">Francés</idioma>
14         <idioma codigo="en">Inglés</idioma>
15         <idioma codigo="it">Italiano</idioma>
16     </idiomas>
17 </cv>
```

Figura 4.11. *En caso de no declarar un atributo en DTD, su existencia provocará errores.*

En el siguiente ejemplo se muestra un uso de los tipos de atributo **ID** e **IDREFS**. Se declara un atributo **identificador** en el elemento **lengua** y se le asigna el tipo **ID**, convirtiendo al atributo en un identificador de entidad (los valores no se pueden repetir). Por otra parte, se declara el atributo **codigoPais** del elemento **idioma** como una referencia a un identificador de otra entidad (sin especificar cuál) mediante el tipo **IDREF**. Como resultado de esta declaración, para que el documento XML sea válido, los valores de los atributos **codigoPais** deben coincidir con los valores de atributos que sean identificadores de otra entidad.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cv [
  <!ELEMENT cv (nombre, idiomas, lenguas)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT idiomas (idioma*)>
  <!ELEMENT idioma (#PCDATA)>
  <!ELEMENT lenguas (lengua*)>
  <!ELEMENT lengua (#PCDATA)>
  <!ATTLIST lengua identificador ID #REQUIRED>
  <!ATTLIST idioma nivel (bajo | medio | alto) "medio">
  <!ATTLIST idioma codigoPais IDREF #REQUIRED>
]>
<cv>
    <nombre>Desmond Miles</nombre>
    <idiomas>
        <idioma codigoPais="fr" nivel="bajo">Francés</idioma>
        <idioma codigoPais="en">Inglés</idioma>
        <idioma codigoPais="it">Italiano</idioma>
    </idiomas>
    <lenguas>
        <lengua identificador="fr">Francés</lengua>
        <lengua identificador="en">Inglés</lengua>
        <lengua identificador="it">Italiano</lengua>
    </lenguas>
</cv>
```