

# Introducción

En este primer capítulo haremos un recorrido por las posibilidades que brinda XML y en qué casos es conveniente utilizarlo.

Además, plantearemos los principios mínimos indispensables para comenzar a conocer e implementar nuestros propios documentos XML.

<b>Qué es XML</b>	<b>14</b>
El aporte de XML	17
Ventajas de XML	18
Jerarquías de la información	19
XML es autodescriptivo	19
<b>Escribir nuestros propios documentos</b>	<b>21</b>
Visualizar documentos	22
<b>Fundamentos de la sintaxis</b>	<b>24</b>
Elementos y atributos	25
Comentarios	35
Prólogo de un documento	38
Tipos de documento	40
<b>Espacios de nombres</b>	<b>45</b>
<b>Resumen</b>	<b>49</b>
<b>Actividades</b>	<b>50</b>

# QUÉ ES XML

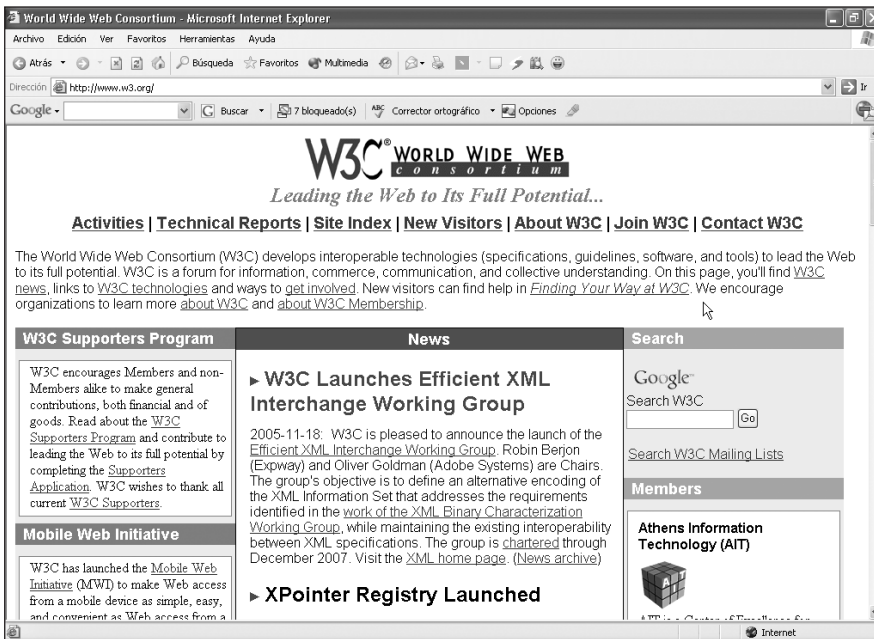
Aunque sus siglas (*eXtensive Markup Language*, Lenguaje de Marcado Extensible) nos hagan suponer lo contrario, XML no es sólo un lenguaje, sino un conjunto de **reglas** y **definiciones** para crearlos.

Por esta razón, XML es considerado como un metalenguaje.

Dichas reglas –que serán explicadas a continuación en este mismo capítulo– se caracterizan por ser simples y estrictas.

XML no está orientado a almacenar algún tipo de dato en particular, sus fines son generales y puede ser utilizado por cualquier tipo de organización.

XML fue creado y es mantenido por la W3C (*World Wide Web Consortium*) con el objeto de superar las limitaciones de otro lenguaje de marcado, HTML.



**Figura 1.** El sitio oficial de la W3C ([www.w3.org](http://www.w3.org)) incorpora referencias y tutoriales imprescindibles acerca de XML.

XML deriva a su vez de SGML (*Standard Generalized Markup Language*), que, similar en cuanto a prestaciones y principios, perdió la chance de convertirse en un estándar popular debido a su gran cantidad de opciones disponibles, lo que se tradujo en complejidad –en ocasiones, innecesaria– para los desarrolladores.

Suele decirse que XML es algo así como SGML simplificado, de manera que una aplicación no necesita comprender SGML por completo para interpretar un documento, sino sólo un subconjunto de las reglas que éste define. Los editores y los desarrolladores SGML, entonces, pueden comprender XML.

Volviendo a HTML: alguien dijo que su problema principal es que fue diseñado pensando en humanos; un individuo eventualmente podría interpretar y diferenciar las partes y los sentidos de un código escrito en este lenguaje con tan sólo verlo, pero ¿podría hacerlo una máquina?

Veamos el siguiente ejemplo:

```
<HTML>
<HEAD><TITLE> grandes peliculas </TITLE></HEAD>
<BODY>

<h1>psicosis</h1>

<br>

<p><b>alfred hitchcock</b>
<p>anthony perkins
<p>janet leight
<p>121 minutos

</BODY>
</HTML>
```

Una persona podría darse cuenta quizá del contenido del archivo. Pero dicho archivo podría ser utilizado para almacenar otro tipo de información:



## EL CONSORCIO DE LA TRIPLE W

La W3C (*World Wide Web Consortium*, [www.w3.org](http://www.w3.org)) se encarga de mantener y revisar en forma constante una gran cantidad de estándares como, por ejemplo, XML, HTML, CSS, entre muchos otros. Si necesitamos la descripción oficial de alguna de estas tecnologías, no dudemos en visitar el sitio de la W3C.

```
<HTML>
<HEAD><TITLE> grandes recetas </TITLE></HEAD>
<BODY>

<h1>pato a la naranja</h1>

<br>

<p><b>preparacion</b>
<p>vierta el contenido del pato en ...
<p>cocine a fuego lento por ...
<p>servase con ...

</BODY>
</HTML>
```

Es decir que, contestando a la pregunta original, no podría hacerlo, necesitaría diferenciar de alguna manera las estructuras de la información.

Otro ejemplo:

```
<HTML>
<HEAD><TITLE> remitente </TITLE></HEAD>
<BODY>

<p>homero simpson
<p>avenida siempre viva 742
<p>springfield

</BODY>
</HTML>
```

```
<HTML>
<HEAD><TITLE> remitente </TITLE></HEAD>
<BODY>

<p>avenida siempre viva 742
<p>springfield
```

```
<p>homero simpson  
</BODY>  
</HTML>
```

¿Cómo podría una aplicación saber cuál es la dirección y cuál el es nombre de la persona? XML puede dar respuesta a estas preguntas.

## El aporte de XML

Si se presta la suficiente atención, podríamos observar que HTML cuenta con, entre otras, las siguientes características:

- **Forma y contenido:** las etiquetas no definen la estructura de los datos contenidos, sino su formato.
- **Límite:** las etiquetas son limitadas en número y, como consecuencia, el lenguaje es limitado. Luego las etiquetas suelen repetirse, y como no hay forma de ligar un dato a una etiqueta, encontrar información en particular dentro de un documento HTML es virtualmente imposible.
- **Reglas:** la sintaxis puede no someterse a reglas estrictas, puesto que los navegadores interpretan de una forma u otra lo que se quiso especificar. Un caso clásico es el de las etiquetas de cierre: aunque no se incluyan explícitamente, ante la apertura de un nuevo Tag (en español, “etiqueta”) se supone que el anterior se cerró.

En resumidas cuentas, HTML puede definir cómo mostrar la información, pero no puede decir lo que la información significa.

Esta libertad y esta orientación al formato más que a las estructuras de datos hicieron que se buscara una tecnología capaz de ocupar el lugar vacío.

Ninguno de los puntos anteriores implica que HTML sea un lenguaje desdeñable, sino que simplemente no puede resolver algunas cuestiones que con insistencia se fueron planteando como necesidad por parte de los desarrolladores.



## VENTAJAS DE LOS EDITORES

El uso de editores especializados en XML nos es obligatorio, pero es indudable que brinda ciertas facilidades que agradan a los desarrolladores: autocomplemento de elementos, resaltado de palabras clave, validación de documentos, plantillas de ejemplo, entre otras.

## Ventajas de XML

Además de la facilidad de aprendizaje en base a sus reglas claras, XML ofrece ciertas características que fueron la base de su desarrollo:

### Intercambio de datos

La comunicación entre los sistemas de información existentes, e incluso entre partes de un mismo sistema, suele convertirse en un problema, puesto que generalmente coexisten aplicaciones desarrolladas en lenguajes, bases de datos, plataformas e, incluso, protocolos de comunicación diferentes. Allí surge la necesidad de unificar el formato de envío y recepción de información: XML proporciona reglas claras para estructurar documentos. Ahora lo único que queda es que las aplicaciones existentes transformen sus datos desde y hacia XML.

Con relación a lo anterior, cuando los documentos XML viajan a través de una red, lo que estamos haciendo es transportar sólo las estructuras y los datos que ellas contienen. No es necesario intercambiar herramientas para que interpreten dicha información: no son difíciles de desarrollar e, incluso, muchos lenguajes traen incorporadas las suyas propias.

### Una estructura, múltiples presentaciones

XML y sus tecnologías asociadas permiten estructurar la información y luego aplicarle fácilmente transformaciones para su presentación. Es normal hoy en día que la información esté almacenada en una base de datos, se convierta a XML y luego se transforme en otra clase de documento para servirlo al cliente (mediante, justamente, el uso de las transformaciones, que veremos en los próximos capítulos).

### Búsquedas más específicas

Al existir la posibilidad de estructurar la información según nuestras necesidades, las búsquedas sobre los documentos serán más específicas y consistentes: mientras que en HTML se realizan analizando el documento completo, en XML existe la posibilidad de buscar sólo en algunas regiones, o sólo la información contenida en algún elemento en particular. Esto se reduce a búsquedas más rápidas y específicas.



## XML TODO TERRENO

XML se ha convertido en una de esas tecnologías que se ven en pequeños, medianos y grandes proyectos: esto prueba lo importante que es conocer sus principios fundamentales y su funcionalidad, el nivel de complejidad de la aplicación en la que se implemente no influye.

## Programación en capas

La programación en capas viene siendo un desafío, si no un punto de partida, para cualquier desarrollo. La idea es separar y delimitar de cierta forma las reglas de negocio, el diseño y los datos. Mientras que algunas tecnologías dejan estos asuntos librados a las intenciones del desarrollador, en XML es casi una obligación formal tanto para los pequeños proyectos como para los grandes. En este capítulo obtendremos una aproximación a cómo estructurar los datos, y más adelante analizaremos cómo manipular el diseño y validar la información dentro de un documento.

## Jerarquías de la información

XML estructura la información de tal manera que, respetando determinadas reglas, una aplicación podría llegar a identificar y recuperar datos de manera inequívoca.

Incluso a simple vista, si el documento no es demasiado complejo, podremos elaborar en nuestra mente una idea de lo que el autor trató de definir.

Normalmente esta imagen generada a partir de la lectura de un documento tiene una estructura jerárquica de la información.

## XML es autodescriptivo

Suele decirse que un documento XML es autodescriptivo, puesto que los autores de estos documentos pueden implementar sus propias etiquetas provocando un nivel de detalle que no da lugar a confusiones.

Como se dijo anteriormente, la estructura de un documento XML puede ser interpretada por máquinas, pero también por humanos: conocer sus reglas básicas garantiza entender el fin de un documento, siempre y cuando éste este modelado y diseñado correctamente.

Al tener un elemento raíz –una etiqueta que engloba a todas las demás; profundizaremos en estos temas luego-, podemos visualizar un documento XML como una estructura tipo árbol.

Veamos a tal fin el siguiente ejemplo (ejemplo 1):

```
<?xml version="1.0"?>

<stock>
  <autor>
```

```
<nombre>Ernest Hemingway</nombre>
<obra>
  <titulo>Los Asesinos</titulo>
  <editorial>Noguer Y Caralt</editorial>

  <precio>53</precio>
</obra>
<obra>
  <titulo>Islas en el Golfo</titulo>
  <editorial>Emece</editorial>
  <precio>16</precio>
</obra>
</autor>
<autor>
  <nombre>H.G.Wells</nombre>
  <obra>
    <titulo>La guerra de los mundos</titulo>
    <editorial>Edaf</editorial>
    <precio>14</precio>
  </obra>
</autor>
<autor>
  <nombre>Leon Tolstoi</nombre>
  <obra>
    <titulo>Los Cosacos</titulo>
    <editorial>Coleccion Austral</editorial>
    <precio>16</precio>
  </obra>
</autor>
</stock>
```

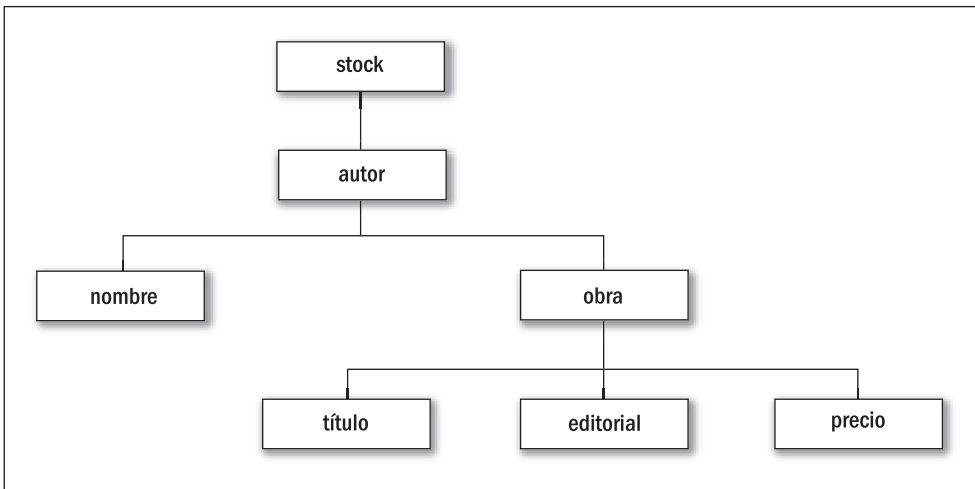


## BASES DE DATOS RELACIONALES

El diseño de bases de datos relacionales y el diseño de documentos XML son dos tareas distintas que tienen mucho en común. Hay una gran cantidad de material informativo acerca del primer punto y se recomienda acceder a él para obtener e incorporar conceptos que indudablemente nos serán de importancia al trabajar con XML.

**Importante:** en caso de ser introducida la primera línea de un documento XML, deberá ser la que se utiliza para incluir el prólogo del documento (veremos esto en detalle más adelante, pero en el ejemplo la línea es `<?xml version="1.0"?>`). Esta línea, además, no podrá ser precedida por espacios. Este comentario se aplica a todos los demás ejemplos.

Dejando de lado la sintaxis utilizada –cuyos principios serán explicados a continuación– podemos observar cómo las demás etiquetas también encierran o contienen otras. Siguiendo este principio, podemos pensar el siguiente diagrama:



**Figura 2.** Los documentos XML pueden visualizarse gráficamente mediante estructuras jerárquicas; es una forma intuitiva de apreciar sus características.

## ESCRIBIR NUESTROS PROPIOS DOCUMENTOS

XML es un estándar abierto, por lo que no hay una empresa comercial propietaria y tampoco hay un único entorno de desarrollo. Para escribir documentos XML nece-



### RECOMENDACIÓN

Tengamos en cuenta que para obtener la recomendación oficial por parte de la W3C de la última versión disponible de XML, debemos visitar la dirección [www.w3.org/XML](http://www.w3.org/XML). Si deseamos obtener la traducción al español, visitemos [www.w3.org/TR/REC-xml](http://www.w3.org/TR/REC-xml). En ambas encontraremos una gran cantidad de información y enlaces a otras tecnologías basadas en XML.

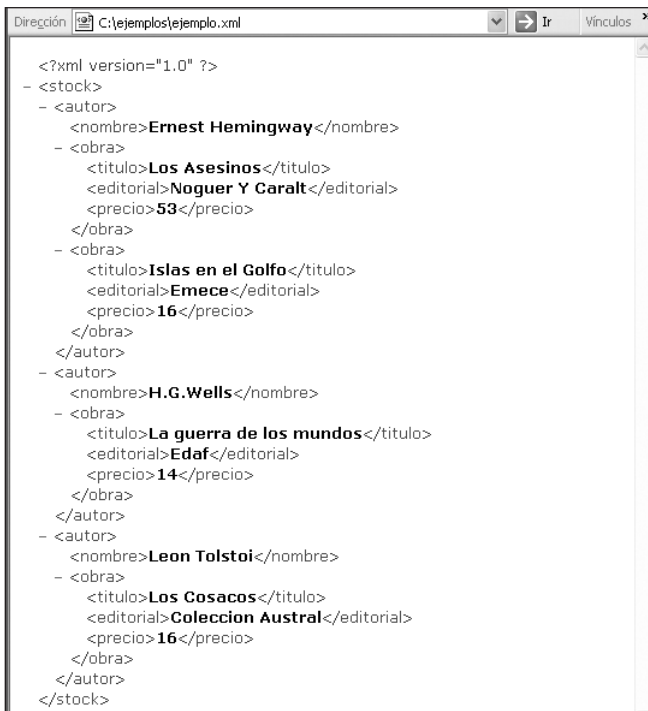
sitamos un editor de textos que nos ofrezca la posibilidad de salvar nuestros documentos en formato de texto plano. **Emacs**, **WordPad**, **NotePad**, **Vim** son sólo algunas posibilidades. A medida que los documentos XML se vuelvan complejos y/o extensos, posiblemente nos veamos en la necesidad de utilizar algún entorno de desarrollo orientado específicamente a XML para hacer más fácil la escritura de código.

Un editor específico para XML entiende las particularidades del lenguaje, “ayudando” al desarrollador en sus tareas más rutinarias.

Pero, como se dijo, si no contamos con uno, simplemente salvamos los documentos dándoles una extensión `.XML` y los abrimos con algún navegador web, como ser MS Internet Explorer (se recomienda la versión 5 en adelante).

## Visualizar documentos

Si guardamos alguno de los ejemplos planteados en este capítulo como un archivo de texto plano con extensión `.XML` e intentamos visualizarlo mediante un navegador, probablemente no obtengamos en la salida un diseño acorde a lo deseado.



```
<?xml version="1.0" ?>
- <stock>
- <autor>
  <nombre>Ernest Hemingway</nombre>
- <obra>
  <titulo>Los Asesinos</titulo>
  <editorial>Noguer Y Caralt</editorial>
  <precio>53</precio>
</obra>
- <obra>
  <titulo>Islas en el Golfo</titulo>
  <editorial>Emece</editorial>
  <precio>16</precio>
</obra>
</autor>
- <autor>
  <nombre>H.G.Wells</nombre>
- <obra>
  <titulo>La guerra de los mundos</titulo>
  <editorial>Edaf</editorial>
  <precio>14</precio>
</obra>
</autor>
- <autor>
  <nombre>Leon Tolstoi</nombre>
- <obra>
  <titulo>Los Cosacos</titulo>
  <editorial>Coleccion Austral</editorial>
  <precio>16</precio>
</obra>
</autor>
</stock>
```

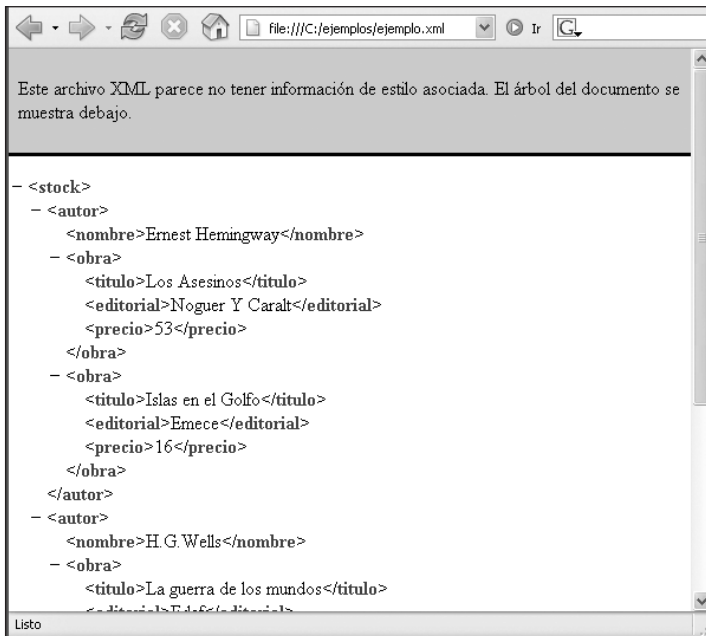
**Figura 3.** Los navegadores web visualizan XML de distintas maneras. MS Internet Explorer ([www.microsoft.com](http://www.microsoft.com)) diagrama y muestra un documento XML válido mediante una estructura de árbol.

Hoy por hoy –y desde hace mucho tiempo– hay una amplia oferta de navegadores disponibles para múltiples sistemas operativos y plataformas. Una muestra clara de la importancia y la posición que toma XML en la industria es que el soporte que los navegadores le dan se ha convertido en un ítem importante a la hora de realizar la elección.



**Figura 4.** Opera ([www.opera.com](http://www.opera.com)) es un navegador web cada vez más utilizado cuyo soporte para XML viene evolucionando versión a versión. En su sitio destina una sección a las tecnologías XML.

Como se dijo, en XML las funciones se modularizan, y distintas tecnologías se necesitan unir para obtener un resultado final que satisfaga tanto a los usuarios como a los desarrolladores de un documento en cuestión.



**Figura 5.** Mozilla Firefox ([www.mozilla.org](http://www.mozilla.org)), para muchos la alternativa a Internet Explorer, brinda en sus últimas versiones soporte XML.

Hasta aquí hemos dado sólo el primer paso: conocer y aplicar las reglas que el estándar XML nos ofrece para estructurar nuestros propios documentos, mientras que en los siguientes capítulos veremos cómo validar los datos que ellos contienen, darles un formato de salida elegante, extraer los datos que necesitamos, etc.

En comparación con HTML, es una situación diferente: allí el diseño y los datos están posiblemente en el mismo documento, mezclados, y al cargarse un documento el navegador los interpreta y muestra la salida correspondiente.

## FUNDAMENTOS DE LA SINTAXIS

Cuando trabajamos con XML, lo que en realidad hacemos es crear documentos basados en el estándar XML, es decir, respetar las reglas impuestas y tratar de describir lo más fielmente posible la estructura de la información que deseamos almacenar.

Cuando un documento XML cumple con las reglas impuestas por el estándar, suele decirse que ese documento está “bien formado”.

Cuando creamos documentos basados en el estándar XML, no tenemos etiquetas predefinidas como en HTML, aquí debemos crearlas nosotros mismos, lo que nos permite estructurar la información según las necesidades existentes y obtener un nivel de detalle muy profundo.

Luego, las aplicaciones que se encarguen de leer y manipular documentos XML podrán delimitar las regiones del mismo a través de ellas para identificar y separar las partes unas de otras. Lo que las aplicaciones o “parsers” hagan luego con cada parte del documento no nos interesa, o por lo menos está fuera de lo que sería el diseño o modelado de un documento. Si tenemos experiencia en el manejo de bases de datos, notaremos que el principio es el mismo, en el sentido que la información se almacena de forma organizada para que las aplicaciones accedan y obtengan lo que buscan



### XML VS. HTML

Los documentos XML no fueron creados con la pretensión de reemplazar a los documentos HTML. Aunque es posible hacerlo, estas tecnologías tienen objetivos diferentes, mientras que XML apunta a estructurar la información, HTML pone la vista en darle formato para presentarla.

(finalmente XML también nos permite guardar información de forma estructurada para que otros accedan a ella). En síntesis, XML nos da la libertad de generar estructuras a nuestra medida, pero nos ata a respetar de forma precisa algunas reglas.

## Elementos y atributos

Veamos el siguiente documento XML:

```
<?xml version="1.0"?>

<videoteca>

  <pelicula>
    <titulo>El cuchillo bajo el agua</titulo>
    <director>Roman Polanski</director>
    <estreno>1956</estreno>

    <imagen nombre="cuchillo.png" />
  </pelicula>

  <pelicula>
    <titulo>Blue</titulo>
    <director>Cristov Kieslowski</director>
    <estreno>1984</estreno>
    <imagen nombre="blue.jpg" />
  </pelicula>

  <pelicula>
    <titulo>la dama de honor</titulo>
    <director>Claude Chabrol</director>
```



## PARSERS XML

Hemos utilizado el término “parser” a lo largo de este capítulo y lo utilizaremos en los que siguen. Podríamos decir que un parser es una pieza de código que intenta leer un documento, recorrerlo e interpretar su contenido. Cualquier código que realiza estas acciones sobre un documento XML se considera justamente un parser XML.

```
<estreno>2005</estreno>
  <imagen nombre="dama.png"/>
</pelicula>

</videoteca>
```

La letra M de XML (Markup - Marcado) nos indica que la estructura de un documento va estar basada en marcas (tags, etiquetas).

Cada marca está delimitada por la presencia de los signos menor que (<) y mayor que (>). Existen marcas que indican el inicio (<nombre>) y marcas que indican el final (</nombre>).

En XML las marcas se denominan **elementos**. Veamos algunas de sus características:

- **Inicio y fin:** cada elemento debe tener una apertura y un cierre. Así, el siguiente ejemplo es incorrecto:

```
<?xml version="1.0"?>

<pelicula>
  <titulo>El cuchillo bajo el agua
  <director>Roman Polanski
</pelicula>
```

- **Nombres:** los nombres de los elementos no pueden contener espacios. El siguiente ejemplo sería incorrecto:

```
<?xml version="1.0"?>

<pelicula de suspenso>
  <titulo>El cuchillo bajo el agua</titulo>
  <director>Roman Polanski</director>
</pelicula de suspenso>
```

- **Anidamiento permitido:** se pueden anidar elementos, pero deben cerrarse en orden (se dice que los elementos no pueden solaparse).

Entonces este ejemplo es incorrecto:

```
<?xml version="1.0"?>

<pais>venezuela<capital>caracas</pais></capital>
```

Si comienza un elemento **<capital>** dentro de un elemento **<pais>**, también se deberá terminarlo dentro de éste. Lo correcto hubiera sido:

```
<?xml version="1.0"?>

<pais>venezuela<capital>caracas</capital></pais>
```

Notemos que un intérprete XML aceptará solamente este último marcado, mientras que los intérpretes HTML de la mayoría de los navegadores aceptarán ambos.

Como se dijo anteriormente, los elementos pueden contener otros elementos. Veamos el siguiente ejemplo:

```
<?xml version="1.0"?>
<listado>
  <persona>
    <nombre>Juan</nombre>
    <mascota>Gato</mascota>
  </persona>

  <persona>
    <nombre>Pedro</nombre>
    <mascota>Gato</mascota>
  </persona>

  <persona>
    <nombre>Ariel</nombre>
    <mascota>Perro</mascota>
  </persona>
</listado>
```

Aquí, el elemento **<persona>** consta de los elementos **<nombre>** y **<mascota>**. Suele decirse que los elementos **<nombre>** y **<mascota>** son hijos del elemento **<persona>**.

- **Elementos vacíos:** los elementos pueden constar de una sola etiqueta, que hará las veces de apertura y cierre. El siguiente ejemplo es incorrecto:

```
<imagen nombre="cuchillo.png">
```

Lo correcto hubiera sido:

```
<imagen nombre="cuchillo.png"/>
```

Otra opción correcta:

```
<imagen nombre="cuchillo.png"></imagen>
```

Los elementos anteriores carecen de contenido, y por eso suele llamárselos **elementos vacíos**. Los elementos `<br>` y `<img>` –propios de HTML– son dos ejemplos.

Todos los elementos deben tener obligatoriamente una etiqueta de fin, incluso aquellos que consten de una sola etiqueta (veamos ejemplos anteriores). Esto en ocasiones no es obligatorio en HTML.

- **Atributos:** los elementos pueden tener atributos, también llamados **modificadores**. Un atributo suele representar una característica de un elemento. Así, en el ejemplo anterior, **nombre** es un atributo del elemento **imagen**. Puede haber más de un atributo por elemento, puede haber sólo uno o puede no haber ninguno.

```
<?xml version="1.0"?>
```

```
<alumno edad="15" estatura="170 centimetros">
```



## XML PUNTO COM

La editorial O'reilly mantiene un sitio especialmente dedicado a XML en donde publicita sus libros e incorpora valiosos artículos técnicos de primer nivel. Pueden visitar el sitio accediendo a la dirección [www.xml.com](http://www.xml.com).

```
Juan Marquez
</alumno>
```

Se dice que un atributo es un par nombre-valor. Veamos la sintaxis de los atributos:

**<nombre-del-elemento nombre-del-atributo=valor-del-atributo>**

El valor del atributo debe encerrarse entre comillas simples o dobles. Los atributos deben contener valores. Así, el siguiente ejemplo es inválido:

```
<?xml version="1.0"?>

<bebida>
  <cafe>
    <capuchino azucar>
  </cafe>
</bebida>
```

Mientras que el siguiente es válido:

```
<?xml version="1.0"?>

<bebida>
  <cafe>
    <negro azucar="si">

    <irlandes azucar="">
  </cafe>
</bebida>
```



## IMPORTANCIA DEL RELEVAMIENTO

Es recomendable realizar un fino trabajo de recolección de información acerca del sistema que queramos desarrollar mediante XML. Esto nos servirá para saber ciertamente qué elementos intervienen, de qué tipo son, y cómo se relacionan entre ellos, es decir, información crítica al momento de desarrollar DTDs.

Podemos usar comillas simples o dobles siempre y cuando la situación no se preste a confusión. Si el valor del atributo contiene algún tipo de comillas, puede usarse el otro tipo para rodear al valor. El siguiente ejemplo es válido:

```
<?xml version="1.0"?>
<disco nombre="please please me">
  <cancion nombre="baby it's you">
    <interprete>the beatles</interprete>
  </cancion>

  <cancion nombre='love me do'>
    <interprete>the beatles</interprete>
  </cancion>

</disco>
```

Mientras que el siguiente no lo es:

```
<?xml version="1.0"?>

<disco nombre="please please me">

  <cancion nombre='baby it's you'>
    <interprete>the beatles</interprete>
  </cancion>

</disco>
```

Otra opción es usar las entidades equivalentes:

COMILLAS	CARÁCTER	SÍMBOLO
Simples	'	&apos;
Dobles	"	&quot;

**Tabla 1.** Definición de entidades equivalentes en XML.

```
<?xml version="1.0"?>
<disco nombre="please please me">
  <cancion nombre='baby it&apos;s you'>
```

```

<interprete>the beatles</interprete>
</cancion>

</disco>

```

Otras equivalencias que pueden servirnos:

CARÁCTER	SÍMBOLO
>	&gt;
<	&lt;
&	&amp;

**Tabla 2.** Más entidades equivalentes.

Luego, el parser de XML se encargará de reemplazar cada uno de los símbolos por el carácter que le corresponda.



**Figura 6.** El uso de entidades predefinidas puede implementarse opcionalmente en HTML y es de gran importancia en XML.

- **Escritura sensible:** los elementos son sensibles a mayúsculas, se escriben en letra minúscula por convención, pero esto no es obligatorio. Lo que sí es obligatorio es escribir de igual manera los tags de apertura y cierre correspondientes a un mismo elemento, en otras palabras, XML es tipado:

## \* EL ELEMENTO HTML

Si prestamos atención, podremos observar cómo en HTML también hay un elemento que contiene a todos los demás, y ese elemento es justamente **<HTML>**. Ocurre que los navegadores no requieren su inclusión en la mayoría de los casos y, por lo tanto, no es obligatoria.

Correcto:

```
<libro>
<titulo>la senda del perdedor</titulo>
</libro>
```

Incorrecto:

```
<Libro>
<titulo>la senda del perdedor</titulo>
</libro>
```

Correcto:

```
<Libro>
<titulo>la senda del perdedor</titulo>
</Libro>
```

Correcto:

```
<abc>
<ABC>texto</ABC>
</abc>
```

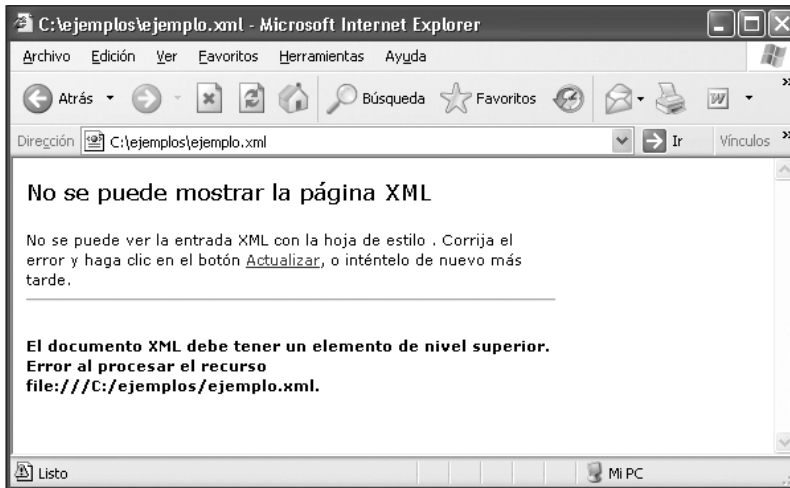
En HTML los elementos no son sensibles a mayúsculas.

- **Raíz:** existe un elemento especial llamado **elemento raíz**. En un documento XML es obligatorio que haya un elemento que encierre a todos los demás.

## FUNDACIÓN APACHE

XML se ha convertido en uno de los puntos fuertes sobre los que se sustenta el llamado proyecto Apache, cuya base es el servidor web que lleva ese nombre. En el sitio dedicado a XML (<http://xml.apache.org>) se listan enlaces a las herramientas desarrolladas para trabajar con este lenguaje [utilizaremos algunas de ellas a lo largo del libro].

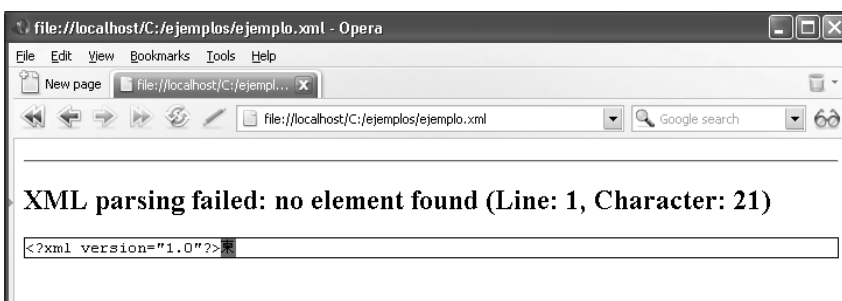
- **Documentos no vacíos:** un documento XML debe contener uno o más elementos.



**Figura 7.** La información y el detalle acerca de los errores en XML son fundamentales. MSIE emite mensajes de error ante un documento XML que no contiene elementos.

Haciendo una analogía entre un documento y un elemento, puede afirmarse que un documento debe contener elementos (es decir, no puede estar vacío) pero un elemento sí puede ser vacío.

La función básica de un documento XML es almacenar información de forma estructurada, es decir, con algún significado que nos permita luego catalogar o diferenciar el contenido. Por esa razón, y basándonos en esta premisa, sería ilógico permitir crear documentos vacíos.



**Figura 8.** La visualización de errores en Opera difiere de la de MSIE: ante el mismo error, el mensaje textual es idéntico, pero el formato de presentación usualmente difiere.

- Cuando un documento cumple con todas las reglas anteriores, suele decirse que se trata de un documento bien formado.

Un dato acerca del diseño de las estructuras de un documento: a medida que se comienzan a escribir archivos XML surge la duda de cuándo un dato debe ser un atributo de un elemento y cuándo un elemento en sí mismo. La respuesta a esto no es precisa, el sentido común y lo que nos sugiera la circunstancia y la experiencia deben prevalecer, pero hay algunas reglas:

- Cuando un dato depende de otro, puede llegar a ser un atributo. De lo contrario, no. Supongamos el siguiente ejemplo:

```
<?xml version="1.0"?>

<persona>
  <nombre>Juan</nombre>
  <mascota>Gato</mascota>
</persona>
```

Si quisiéramos almacenar en el documento la edad de la persona, habría tres opciones:

- Ubicarla como atributo o elemento de **<persona>**.
- Ubicarla como atributo o elemento de **<nombre>**.
- Ubicarla como atributo o elemento de **<mascota>**.

Como la edad de una persona no se relaciona o depende de ningún modo ni de la mascota ni del nombre, pero sí de la persona, se tomaría la primera opción. La edad es una característica de la persona, como lo son el nombre y la mascota; en este caso específico, daría lo mismo ubicarla como atributo o elemento dentro de **<persona>**.

- El contenido de un atributo no puede resultar demasiado extenso. Esto tiene más que ver con tratar de mantener la claridad del documento XML que con otra cosa.
- Cuando un dato puede tomar valores predefinidos, es un atributo. Trataremos esto con más claridad en los próximos capítulos, pero veamos un ejemplo:



## VIDA REAL

Se pueden realizar prácticas para mejorar el manejo de XML mediante la implementación de casi cualquier tipo de situación: utilicemos XML para catalogar nuestros discos, agendar contactos, crear una agenda de actividades, etc.; en definitiva, cualquier evento puede representarse según las reglas impuestas por XML.

```

<?xml version="1.0"?>

<prenda>
  <camisa>
    <tipo talle="chico">Manga Larga</tipo>
    <precio>$ 123</precio>
  </camisa>

  <camisa>
    <tipo talle="grande">Manga Larga</tipo>
    <precio>$ 223</precio>
  </camisa>

  <camisa>
    <tipo talle="chico">Manga Corta</tipo>
    <precio>$ 123</precio>
  </camisa>

  <pantalon>
    <tipo talle="chico">Pinzado</tipo>
    <precio>$ 123</precio>
  </pantalon>
</prenda>

```

Como vemos, el atributo **talle** puede tomar valores predefinidos, supongamos “chico”, “mediano” y “grande”.

Incluso, hasta podríamos definir dos listados diferentes de talles: un listado especial de talles de camisas y otro distinto para los talles de los pantalones (veremos esto con mayor profundidad más adelante).

## Comentarios

Es posible incorporar comentarios en los documentos XML. La sintaxis de un comentario es la siguiente:

```
<!-- contenido del comentario -->
```

Los comentarios pueden incluirse en cualquier parte de un documento, siempre y cuando estén debajo de la declaración de inicio de XML y no dentro de un elemento. Veamos un ejemplo:

```
<?xml version="1.0"?>
<!--este es un comentario -->

<saludo>

  <ingles>
    Hello
  </ingles>

  <frances>
    Bonjour
  </frances>

  <espanol>
    Hola      <!--este es otro comentario -->
  </espanol>

</saludo>
<?xml version="1.0"?>
```

```
<saludo>

  <ingles <!-- esto mostraria un error -->
    Hello
  </ingles>

</saludo>
```

El texto de un comentario no puede contener un guión doble (– –). Hecha esta excepción, un comentario puede contener cualquier conjunto de caracteres.

Los comentarios pueden incluir múltiples líneas:

```
<?xml version="1.0"?>

<!--
este es un comentario multilinea
-->
```

```
<saludo>

  <ingles>
    Hello
  </ingles>

  <frances>
    Bonjour
  </frances>

  <espanol>
    Hola          <!--este no -->

  </espanol>
</saludo>
```

Si incluimos un elemento dentro de un **comentario**, éste será ignorado. Si se quiere eliminar una gran sección de un documento XML, simplemente introdúzcala en un comentario. Para restaurar la sección comentada, simplemente eliminamos las etiquetas de **comentario**:

```
<?xml version="1.0"?>

<saludo>

  <ingles>
    Hello
  </ingles>

  <frances>
    Bonjour
  </frances>

  <!-- comienzo del comentario

  <espanol>
    Hola
  </espanol>
```

```
fin del comentario -->  
  
</saludo>
```

## Prólogo de un documento

Un documento XML consta de información estructurada y delimitada mediante el uso de elementos y atributos, como acabamos de ver, pero además podemos indicar algunos datos extra con información formal acerca de la versión de XML utilizada, el juego de caracteres del que se dispondrá y la indicación acerca de si se van a utilizar documentos externos o no.

Todos estos datos se incluyen en una parte del documento llamada **prólogo** y tienen como fin determinar formalmente un documento. Hoy en día existe una gran cantidad de formatos disponibles –más allá de XML, claro– y de uso extendido, por lo que nos será preciso identificar nuestros documentos XML como tales ante las aplicaciones que deseen hacer uso de ellos.

Esta declaración está recomendada, pero no es obligatoria. Si existe una, debe ser lo primero que aparezca en el documento y deberá contener de forma obligatoria sólo la primera propiedad, mientras que las demás pueden ser omitidas, en cuyo caso tomarán valores por defecto.

Veamos la sintaxis y el detalle de cada una mediante un ejemplo:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>  
  
<destino>  
  <direccion>  
    <nombre>eduardo</nombre>
```



## LA VERSIÓN DE XML

La línea que contenga la versión de XML (`<?xml version="1.0"?>`), en caso de ser incluida, deberá figurar en primer lugar en nuestros documentos XML, ni siquiera podemos dejar líneas en blanco ni espacios antes de esta declaración.

```

    <apellidos>mars</apellidos>
    <calle>calle 1401 </calle>
    <ciudad>berazategui</ciudad>
    <codigo-postal>34829</codigo-postal>
  </direccion>

  <direccion>
    <nombre>maria</nombre>
    <apellidos>fritz</apellidos>
    <calle>calle 1402 </calle>
    <ciudad>monte grande</ciudad>
    <codigo-postal>34830</codigo-postal>
  </direccion>
</destino>

```

## Versiones

XML es una tecnología reciente y en continua revisión, por lo que se recomienda prestar atención a las últimas versiones y utilizar la que satisfaga nuestras necesidades. Como la versión de XML actual aprobada por la W3C es justamente la 1.0, la utilizaremos a lo largo de los ejemplos de este libro. A medida que vayan surgiendo otras, deberemos tener en cuenta cuáles son las mejoras introducidas y cuál es la compatibilidad con la utilizada por nosotros hasta ese momento en nuestros documentos. Si actualizamos a una versión distinta de la que estamos utilizando, deberemos eventualmente revisar nuestros documentos y actualizarlos según la última especificación.

## Juego de caracteres

Si no se especifica encoding, el parser XML asume que los caracteres pertenecen al conjunto UTF-8, un estándar Unicode que soporta virtualmente cada carácter e ideograma de cualquier lenguaje del mundo.



## ENCODING

Cada carácter tiene asociado un código numérico. ISO-8859-1 define la codificación del alfabeto latino, con caracteres acentuados, ñ, ç y ž. Estos caracteres de ISO-8859-1 son los primeros 256 del estándar ISO 10646 ([www.unicode.org](http://www.unicode.org)). Se subdivide en UTF-8, UTF-16, y UTF-32, que sólo difieren en el número de bits que utilizan. En XML, se utilizará por defecto UTF-8 si no declaramos otro.

## Documentos externos

Un documento XML puede hacer uso de otros documentos que pueden o no ser XML. Para ello se utiliza la palabra **standalone**. Esta propiedad puede tomar uno de dos valores: **yes** si no se van a utilizar documentos externos, y **no** si se van a utilizar documentos externos. Si se omite, asumimos que existe la posibilidad de que se vayan a utilizar documentos externos, sin importar si luego se utilizan o no.

Cada uno de los valores de estas tres propiedades puede encerrarse entre comillas, tanto simples como dobles.

## Tipos de documento

Pero además de esta declaración, podemos adjuntar más información dentro del prólogo de un documento. Veamos el siguiente ejemplo:

```
<?xml version="1.0"?>

<!DOCTYPE estreno "estrenos.dtd"
[
<!ENTITY titulo_inicio "Catalogo del mes">
<!ENTITY pie " - 2006 - ">
]>

<estreno>

    <titulo>&titulo_inicio;</titulo>

    <pelicula>
        <nombre>El hijo de la novia</nombre>
        <duracion>126</duracion>
        <origen>Argentina</origen>
```



## ASOCIANDO CONTENIDOS MEDIANTE ENTIDADES

La ventaja del uso de entidades es hacer más fácil la escritura, la lectura y el mantenimiento de los documentos. Suele ocurrir que los documentos XML contengan más y más información a lo largo del tiempo, y allí el uso de entidades se vuelve útil y necesario. Practiquemos la utilización de entidades, tarde o temprano las necesitaremos.

```

</pelicula>

<pelicula>
  <nombre>Amores perros</nombre>
  <duracion>134</duracion>
  <origen>Mejico</origen>
</pelicula>

<pie_de_pagina>&pie;</pie_de_pagina>

</estreno>

```

La primera línea ya nos es familiar: declaramos la versión de XML utilizada implícitamente, que utilizaremos el juego de caracteres UTF-8, y que existe la posibilidad de que requiramos archivos externos.

Luego nos encontramos con un cierto contenido ubicado entre **<!DOCTYPE y >**:

```

<!DOCTYPE
  ...
  ...
  ...
>

```

Este contenido es la declaración del tipo de documento y nos servirá para definir datos tales como entidades, ubicación del DTD y el elemento raíz del documento.

## Entidades

Las entidades sirven para declarar una asociación entre una palabra y un contenido. Dicha palabra podrá ser utilizada todas las veces que se desee a lo largo del documento, y el parser se encargará luego de reemplazarla por el contenido, todas y cada una de las veces que la encuentre.

Vale decir, se escribe cierto contenido una vez y se utiliza muchas.

Para hacer uso de una entidad, debemos seguir dos puntos básicos: realizar la declaración y llamarla desde la parte principal del documento. Para llamar a una constante debemos incluir la secuencia `&` + nombre de la entidad + punto y coma, tal como puede verse en el ejemplo.

Una entidad puede tener como contenido elementos:

```
<!ENTITY nom "<nombre>joaquin</nombre>">
```

## Ubicación del DTD

Veremos en profundidad qué es y para qué puede llegar a servirnos un documento DTD en el próximo capítulo, por ahora diremos que ligar un documento XML a un DTD hará que podamos definir la estructura de nuestros documentos de forma estricta y detallada. Por ejemplo, podríamos llegar a definir qué elementos pueden estar dentro de otros, cuáles no, en qué orden deben figurar, qué tipo de datos podrán almacenar, qué atributos contendrán, etc.

En caso de que el archivo DTD esté en nuestro sistema y queramos acceder a él sin la utilización de un servidor, observemos los siguientes ejemplos:

```
<!DOCTYPE texto SYSTEM "C:\WINDOWS\archivo.dtd">
```

```
<!DOCTYPE texto SYSTEM "archivo.dtd">
```

```
<!DOCTYPE texto SYSTEM "./archivo.dtd">
```

```
<!DOCTYPE texto SYSTEM "../documentos/archivo.dtd">
```

```
<!DOCTYPE texto SYSTEM "../dtds/archivo.dtd">
```

Esto también es válido para las entidades:

**doc1.xml:**

```
<pelicula>
  <nombre>Z</nombre>
  <duracion>88</duracion>
  <origen>Estados Unidos</origen>
</pelicula>
```

**doc2.xml:**

```

<pelicula>
  <nombre>Adios a las vegas</nombre>
  <duracion>95</duracion>
  <origen>Estados Unidos</origen>
</pelicula>

```

**doc3.xml:**

```

<pelicula>
  <nombre>21 gramos</nombre>
  <duracion>126</duracion>
  <origen>Estados Unidos</origen>
</pelicula>

<pelicula>
  <nombre>el rio</nombre>
  <duracion>118</duracion>
  <origen>Japon</origen>
</pelicula>

```

**estrenos.xml:**

```

<?xml version="1.0"?>

<!DOCTYPE estreno SYSTEM "estrenos.dtd"
[
<!ENTITY partel SYSTEM "doc1.xml">

```

**MSDN**

Microsoft es una de las empresas que apuntalan XML y, por supuesto, en su sitio web (<http://msdn.microsoft.com/xml>) incluye toda la información necesaria para incorporar y vincular XML a sus propias herramientas de desarrollo. Además, contiene enlaces a otros sitios, artículos y códigos de ejemplo.

```

<!ENTITY parte2 SYSTEM "doc2.xml">
<!ENTITY parte3 SYSTEM "doc3.xml">
]>

<estreno>

&parte1;
&parte2;
&parte3;

</estreno>

```

En el caso que un DTD resida en un servidor:

```

<!DOCTYPE texto SYSTEM "http://www.dtds.com/dtds/archivo.dtd">

```

Otra forma bastante vista de acceder a un DTD viene dada por la utilización del formato FPI. Veamos un ejemplo:

```

<!DOCTYPE inicio PUBLIC "-//MPE//DTD Ejemplo Version 1.0//ES"
"http://www.onweb.com/topics/xml/dtds/ini.dtd">

```

La parte "**-//MPE//DTD Ejemplo Version 1.0//ES**" es conocida como FPI (*Formal Public Identifier*, Identificador Público Formal) y está compuesta por las siguientes partes:

#### **reconocido // dueño // descripción // idioma**

- **reconocido:** puede tomar los valores + (el DTD es reconocido por alguna entidad oficial) y - (el DTD no es reconocido por ninguna entidad oficial).
- **dueño:** palabra clave que identifica a la institución responsable que se encarga de mantener este documento.
- **descripción:** indicativo del tipo de documento (DTD) y su nombre identificador.
- **idioma:** acrónimo del lenguaje por defecto (ES: Español).

En algunos casos se agrega un dato más a la sintaxis utilizada:

**reconocido // dueño // descripción // idioma // sintaxis**

```
<!DOCTYPE inicio PUBLIC "-//MPE//DTD Ejemplo Version 1.0//ES//XML"
"http://www.onweb.com/topics/xml/dtds/ini.dtd">
```

En nuestro caso, XML.

### Elemento raíz

El elemento raíz es aquel que encierra o contiene a todos los demás. Aquí sólo incluimos el nombre del elemento, es decir, sin los signos < y >.

## ESPACIOS DE NOMBRES

Como se dijo anteriormente, los autores de documentos XML tienen absoluta libertad de decisión en cuanto a la creación de elementos y atributos propios para definir sus estructuras de datos.

También se dijo que está previsto que diversos documentos XML sean utilizados por diferentes módulos de software. Supongamos el caso de un documento que almacena los códigos postales de alguna región y otro que guarda la definición de complejas fórmulas matemáticas: es lógico y comprensible que si alguien requiere para sus propios proyectos la utilización de tales servicios, pueda simplemente incluirlos.

Ahora bien, cada uno de dichos documentos contendrá sus propias marcaciones, que podrán interferir eventualmente con las de los otros, en el sentido que podrán incluir nombres de elementos coincidentes. Veamos el siguiente ejemplo en donde el elemento **titulo** –que tiene diferente significado según la circunstancia en que se utilice– aparece en dos estructuras diferentes:

### JAVA

El lenguaje de programación Java incluye innumerables herramientas para trabajar con XML, la mayoría de las cuales están disponibles de forma gratuita para los desarrolladores interesados. Más información, en <http://java.sun.com/xml>.

```
<?xml version="1.0"?>

<pelicula-favorita>
  <titulo>la ciudad del pecado</titulo>
  <director>robert rodriguez</director>
  <duracion>112</duracion>
</pelicula-favorita>
```

```
<?xml version="1.0"?>
<postulante>

  <titulo>Ingeniero</titulo>
  <experiencia>Desde 2001 en Mars SA</experiencia>
  <idioma>Ingles</idioma>
  <residencia>Los Angeles CA</residencia>

</postulante>
```

Entonces, al tener que reconocer dichos elementos, las aplicaciones de software diseñadas y preparadas para recibir un formato de datos específico se encontrarán con estructuras inesperadas, que fueron diseñadas para otras aplicaciones y que, sin embargo, utilizan el mismo nombre de elemento.

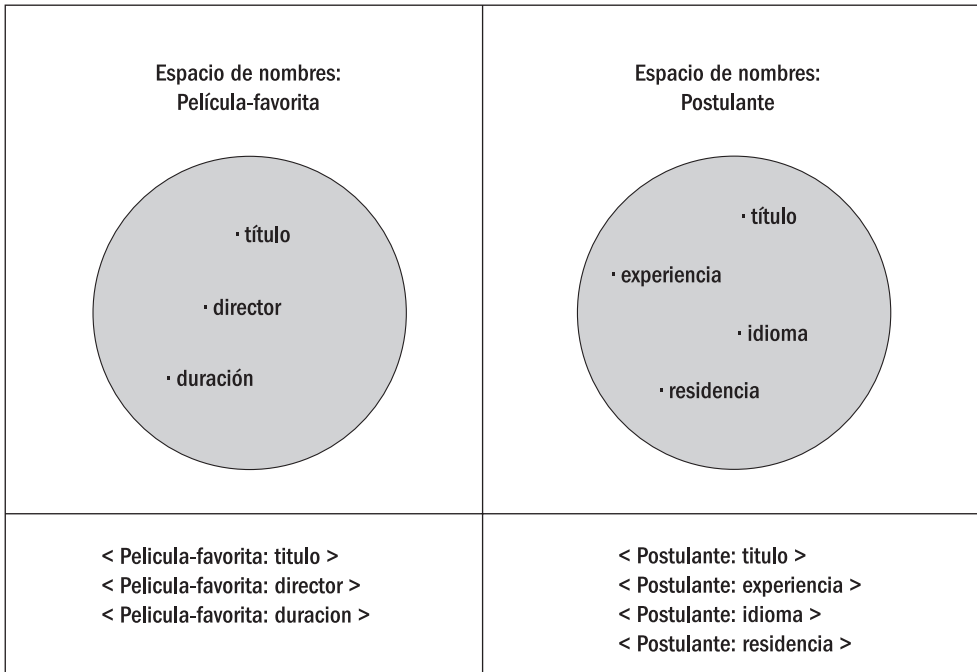
La solución a este problema potencial es que las construcciones de documentos posean nombres universales, es decir, que permitan que sean diferenciables unos de otros, y cuyo ámbito se extienda más allá del documento que las contiene.

Los denominados espacios de nombres (**namespaces**) que son utilizados en XML ofrecen una respuesta que soluciona este inconveniente.



## TUTORIALES EN W3C

La W3C (World Wide Web Consortium) ofrece tutoriales para iniciarse en el uso y la puesta en práctica de los estándares que mantiene, no sólo de XML, sino también de HTML, XHTML, XSLT, XPATH y XQuery, entre otros. La dirección referida a XML es [www.w3schools.com/xml](http://www.w3schools.com/xml).



**Figura 9.** Pensemos en espacios de nombres como si de conjuntos se trataran: pueden tener elementos coincidentes, pero cada uno de éstos pertenecerá únicamente a un conjunto.

Para trabajar con espacios de nombres, en nuestros documentos primeramente debemos declararlos. La declaración utiliza la siguiente sintaxis:

**xmlns:nombre = "url"**

En donde:

- **xmlns:** avisa al parser XML que se está declarando un espacio de nombres.
- **nombre:** es el prefijo que luego podremos utilizar para identificar a qué espacio de nombres corresponde cada elemento de nuestro documento.
- **url:** usualmente debería corresponder a la dirección URL de la entidad que mantiene el espacio de nombres; sin embargo, en ocasiones los espacios de nombres son creados por usuarios y no tienen una entidad oficial relacionada. La URL sólo es necesaria para aportar información adicional al espacio utilizado, y el parser XML no verifica en ningún momento que dicha dirección exista.

La declaración se incluye relacionándola con un elemento. Luego, todos los descendientes de ese elemento serán parte del mismo espacio de nombres, a menos que se especifique lo contrario.

Para definir a qué espacio de nombres corresponde un determinado elemento, utilizamos la siguiente sintaxis:

### **nombre-del-espacio-de-nombres:nombre-del-elemento**

En el siguiente apartado retomamos nuestro primer ejemplo y utilizamos dos espacios de nombres diferentes, **pos** y **pf**:

```
<?xml version="1.0"?>

<pos:postulante xmlns:pos="http://www.esp-p.com/">
  <pos:titulo>Ingeniero</pos:titulo>
  <pos:experiencia>Desde 2001 en Mars SA</pos:experiencia>
  <pos:idioma>Ingles</pos:idioma>
  <pos:residencia>Los Angeles CA</pos:residencia>
  <pf:pelicula-favorita xmlns:pf="http://www.fm.com/">
    <pf:titulo>la ciudad del pecado</pf:titulo>
    <pf:director>robert rodriguez</pf:director>
    <pf:duracion>112</pf:duracion>
  </pf:pelicula-favorita>
</pos:postulante>
```

Existe una variante para trabajar con espacios de nombres que consiste en declarar uno por defecto: en caso de no declarar en un elemento el espacio de nombres al cual pertenece, se asume el espacio por defecto. Para declarar un espacio de nombres por defecto, la sintaxis utilizada es la siguiente:

**xmlns = "url"**

Veamos cómo quedaría nuestro ejemplo utilizando esta opción:

```
<?xml version="1.0"?>

<postulante xmlns="http://www.esp-p.com/">
  <titulo>Ingeniero</titulo>
  <experiencia>Desde 2001 en Mars SA</experiencia>
  <idioma>Ingles</idioma>
  <residencia>Los Angeles CA</residencia>
```

```
<pf:pelicula-favorita xmlns:pf="http://www.fm.com/">
  <pf:titulo>la ciudad del pecado</pf:titulo>
  <pf:director>robert rodriguez</pf:director>
  <pf:duracion>112</pf:duracion>
</pf:pelicula-favorita>
</postulante>
```

Sólo puede haber un espacio de nombres por defecto dentro de un documento; los demás deberán ser declarados y utilizados explícitamente.

La utilización de espacios de nombres no es en absoluto obligatoria, sólo se vuelve necesaria en ciertos casos como, por ejemplo, los enumerados anteriormente. Podremos eventualmente desarrollar documentos XML prescindiendo de ellos.

Veremos en próximos capítulos cómo las tecnologías asociadas a XML normalmente trabajan con espacios de nombres para no interferir en los documentos escritos y desarrollados por particulares.

## RESUMEN

XML es ya un estándar utilizado por desarrolladores de todos los lenguajes. En este primer capítulo vimos cómo puede ayudarnos a resolver situaciones que antes se volvían complejas a falta de una tecnología que fuera independiente de cualquier emprendimiento comercial y, a la vez, pudiera ser utilizada por una amplia gama de lenguajes, plataformas y protocolos de comunicación. Además, aprendimos cómo construir nuestros propios documentos respetando los principios sintácticos impuestos por XML.



### TEST DE AUTOEVALUACIÓN

- 1 Cómo podría definir que es XML?  
\_\_\_\_\_
- 2 Nombre tres ventajas que surjan del uso de XML.  
\_\_\_\_\_
- 3 ¿Cuáles cree que son las principales diferencias entre XML y SGML?  
\_\_\_\_\_
- 4 ¿Cuáles cree que son las principales diferencias entre XML y HTML?  
\_\_\_\_\_
- 5 ¿A qué nos referimos cuando decimos que XML es un lenguaje de marcado?  
\_\_\_\_\_
- 6 ¿Qué es un metalenguaje?  
\_\_\_\_\_
- 7 ¿Cuál es la empresa propietaria de XML?  
\_\_\_\_\_
- 8 ¿Sobre qué sistemas operativos podemos desarrollar documentos XML?  
\_\_\_\_\_
- 9 ¿Qué es un elemento? ¿A qué se denomina elemento raíz de un documento?  
\_\_\_\_\_
- 10 ¿Qué es un atributo?  
\_\_\_\_\_
- 11 ¿Qué es una entidad?  
\_\_\_\_\_

### EJERCICIOS PRÁCTICOS

- ✓ Enumere cinco atributos para el elemento "gato".  
\_\_\_\_\_
- ✓ Enumere cinco elementos que a su vez puedan ser contenidos por el elemento "libro".  
\_\_\_\_\_
- ✓ Trate de describir una estructura para almacenar y administrar una colección de revistas. Ahora escriba un documento XML a tal fin.  
\_\_\_\_\_
- ✓ Trate de describir una estructura para almacenar y administrar una agenda de contactos. Ahora escriba un documento XML a tal fin.  
\_\_\_\_\_
- ✓ Reemplace algún elemento del documento anterior por una entidad.  
\_\_\_\_\_
- ✓ Abra alguno de los documentos anteriores en MS Internet Explorer. ¿Coincide con la estructura que describió originalmente?  
\_\_\_\_\_
- ✓ ¿Utilizaría documentos DTD para validar los ejemplos anteriores? ¿Por qué?  
\_\_\_\_\_