

# CICLOS FORMATIVOS DE GRADO SUPERIOR EN DESARROLLO DE APLICACIONES WEB Y DESARROLLO DE APLICACIONES MULTIPLATAFORMA



Lenguajes de marcas y sistemas de  
gestión de información

Quedan rigurosamente prohibidas, sin la autorización escrita de los titulares de «Copyright», bajo las sanciones establecidas en las leyes, la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático, y la distribución de ejemplares de ella mediante alquiler o préstamo públicos. Diríjase a CEDRO (Centro Español de Derechos Reprográficos, <http://www.cedro.org>) si necesita fotocopiar o escanear algún fragmento de esta obra.

## INICIATIVA Y COORDINACIÓN

**IFP** Innovación en Formación Profesional

*Supervisión editorial y metodológica:*

Departamento de Producto de Planeta Formación

*Supervisión técnica y pedagógica:*

Departamento de Enseñanza de **IFP** Innovación en Formación Profesional

Módulo: Lenguajes de marcas y sistemas de gestión de información / Desarrollo de aplicaciones web

© Planeta DeAgostini Formación, S.L.U.

Barcelona (España), 2017

# INTRODUCCIÓN AL MÓDULO

---

En este módulo estudiaremos los **lenguajes de marcas** y sus características comunes, así como también su clasificación, estructura y sintaxis.

Analizaremos también el uso de los lenguajes de marcas **HTML** (*HyperText Markup Language*) y **XHTML** (*eXtensible HyperText Markup Language*) en entornos web, así como las principales herramientas de diseño web y hojas de estilo. HTML tiene dos características esenciales: el hipertexto y la universalidad. Hipertexto significa que puede crear un vínculo en una página web que lleve al visitante a cualquier otra página en Internet; universalidad significa que, puesto que los documentos HTML se guardan como archivos de sólo texto, cualquier ordenador puede leer una página web sin que importe su sistema operativo. XHTML es una importante mejora sobre el HTML. Es más sólido, más flexible, más potente y puede ampliarse para ajustarse a cualquier necesidad.

Profundizaremos también en el **lenguaje XML** (*eXtensible Markup Language*). Aparentemente, el lenguaje XML es muy similar a XHTML, con etiquetas, atributos y valores. Sin embargo, no solamente sirve para crear páginas web, sino que se trata de un lenguaje que sirve para crear otros lenguajes. Podemos utilizar XML para diseñar nuestro propio lenguaje de marcas, del que nos valdremos para dar formato a nuestros documentos. Nuestro lenguaje personalizado de marcas contendrá etiquetas que describen realmente los datos que contienen.

Por último, veremos cuáles son los sistemas de gestión empresarial, identificaremos los flujos de información e integraremos módulos e informes con aplicaciones ofimáticas que nos permitan exportar la información al formato deseado.



# UNIDAD FORMATIVA 1

- Características de los lenguajes de marcas
- Utilización de lenguajes de marcas en entorno web

# 1. CARACTERÍSTICAS DE LOS LENGUAJES DE MARCAS

El **lenguaje de marcas** puede definirse como la forma de codificar un documento al que, junto con el texto, se incorporan una serie de etiquetas o marcas que contienen información adicional acerca de la estructura del texto o de su presentación.

A la hora de escribir un texto cualquiera ya estamos utilizando marcas. Por ejemplo, los espacios indican límites entre las palabras, las comas establecen separaciones entre frases y los puntos marcan el final de una frase. Aunque no seamos conscientes de ello, estamos utilizando aquellas marcas que nuestro sistema de escritura requiere.

Las marcas no son parte del texto en sí mismo, ya que si lo leemos en voz alta no las pronunciamos. Sin embargo, interpretamos estas marcas y gesticulamos para transmitir la información adecuadamente. Por ejemplo, un signo de interrogación provocará que cambiemos la entonación.

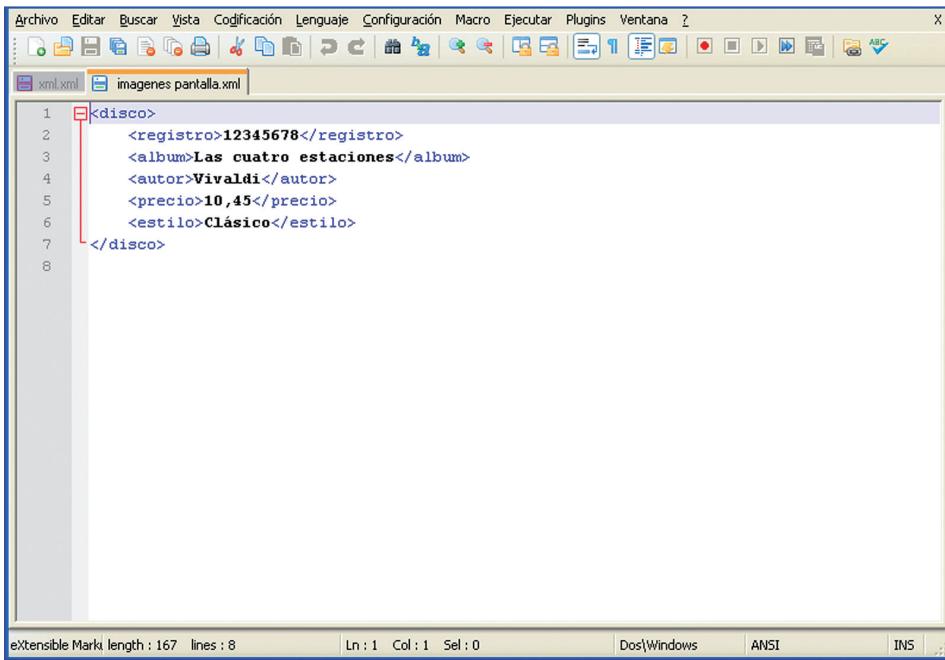
En esta unidad analizaremos, además del lenguaje de marcas propiamente dicho, las características del lenguaje XML (*eXtensible Markup Language*) para crear y modificar documentos XML aplicables a diferentes ámbitos: web, sindicación de contenidos, ficheros de configuración, etc.

## 1.1 Características comunes

Un **lenguaje de marcas** se caracteriza por ser un sistema que permite incluir anotaciones, llamadas **marcas** o **etiquetas**, en un texto. Éstas sirven, por ejemplo, para indicar cómo se quiere presentar el texto cuando se visualice en un navegador web. De modo que, el lenguaje HTML (HyperText Markup Language) utiliza la etiqueta `<b> . . . </b>` para indicar que el texto se debe representar en negrita:

`<b> Esto se vería en negrita </b>`

Otro uso habitual de los lenguajes de marcas es la estructuración de datos, que permite su procesamiento automático por parte de un programa informático. Veamos a continuación un posible formato de datos estructurados para un sistema de gestión de discos de música. Si estos datos no estuvieran organizados, su procesado informático sería mucho más complicado (Figura 1.1).



```

<?xml version="1.0" encoding="UTF-8"?>
<disco>
    <registro>12345678</registro>
    <album>Las cuatro estaciones</album>
    <autor>Vivaldi</autor>
    <precio>10,45</precio>
    <estilo>Clásico</estilo>
</disco>

```

The screenshot shows a software interface for editing XML files. The title bar says 'xml.xml'. The main window displays the XML code for a music album. The code includes elements like 'registro', 'album', 'autor', 'precio', and 'estilo'. The software has a toolbar at the top with various icons for file operations, and a status bar at the bottom showing file length, line count, column count, selection, and encoding information.

**Figura 1.1**

Ejemplo de lenguaje de marcas.

Los lenguajes de marcas suelen confundirse con lenguajes de programación. Sin embargo, no son lo mismo, ya que el lenguaje de marcas no tiene funciones aritméticas o variables, tal como poseen los lenguajes de programación. A los lenguajes de marcas también se les llama lenguajes de etiquetaje o marcado, y entre sus características principales hay que destacar las siguientes:

- **Separan el contenido de su presentación.** Por un lado, presentan el contenido del documento y por otro lado cómo queremos que se muestre.
- **Son archivos de texto plano.** Los archivos de texto plano son aquellos que están compuestos únicamente por texto sin formato, sólo caracteres. Estos caracteres se pueden codificar de distintos modos dependiendo del lenguaje usado. Una de las principales ventajas del lenguaje de marcas es que puede ser interpretado directamente, dado que son archivos de texto plano. Esto es una ventaja evidente con respecto a los sistemas de archivos binarios, que requieren siempre de un programa intermediario que lo interprete. Un documento escrito con lenguaje de marcas puede ser editado por un usuario con un sencillo editor de textos, sin perjuicio de que se puedan utilizar programas más sofisticados que faciliten el trabajo. Al tratarse solamente de texto, los documentos son independientes de la plataforma, del sistema operativo o del programa con el que se crearon.
- **Facilitan la interoperabilidad.** Una de sus mayores ventajas es la posibilidad de intercambiar información mediante archivos de texto plano entre programas y sistemas operativos distintos.

- **Uso de etiquetas o marcas.** Las etiquetas (también llamadas *tags*) permiten marcar el texto y, normalmente, van mezcladas con el propio documento sobre el que se trabaja.
- **Flexibilidad.** Pueden utilizarse varios lenguajes de marcas diferentes en un único archivo, debido a que se han empezado a utilizar en áreas como gráficos vectoriales, servicios web, sindicación web o interfaces de usuario.
- **Compactación.** Hay un único flujo de datos en donde aparecen los contenidos y las instrucciones. Éste es un ejemplo de marcado de texto en lenguaje HTML (Figura 1.2).

**Figura 1.2**  
Ejemplo de lenguaje HTML.

Ejemplos	HTML
Título	<h1>Título</h1>
Lista	<ul> <li>Punto 1</li> <li>Punto 2</li> <li>Punto 3</li> </ul>
texto en <b>negrita</b>	<b>texto</b>
texto en <i>cursiva</i>	<i>texto</i>

- **Facilidad de procesamiento.** Esta característica ha permitido cambiar los hábitos de comportamiento de muchas empresas, pues ha conseguido que sus manuales se editen únicamente en versión electrónica, y a partir de ésta se realicen las versiones impresas, en línea o en CD. Un ejemplo notable fue el caso de Sun Microsystems, una empresa que optó por escribir la documentación de sus productos en SGML, ahorrándose de este modo costes considerables.

## 1.2 Identificación de ámbitos de aplicación

Actualmente, los lenguajes de marcas tienen una amplia aplicación en diferentes ámbitos. Se utilizan para cualquier tipo de documento (textos, presentaciones, gráficos, tecnologías de Internet, matemáticas, música, multimedia, etc.). A continuación, indicamos algunos de ellos:

- **Marcado de documentos de tipo general.** La sencillez y compactación del lenguaje de marcas permite que pueda utilizarse en documentos de todo tipo. Uno de estos usos es el lenguaje de marcación para wikis, denominado *Wikitexto*.

- **Tecnologías de Internet.** El ámbito de aplicación más utilizado. La evolución de la World Wide Web se ha producido gracias a los lenguajes de marcación como SGML, HTML, XML, XHTML etc. (Figura 1.3)

World Wide Web	Interfaz de usuario	Sindicación	Servicios web
HTML	GladeXML	Atom	WSDL
XHTML	MXML (Macromedia)	RSS	XINS
Wireless ML	User Interface ML	ICE	WSCL
Handheld ML	XAML and MyXaml	OPML y OML	WSFL
RDF	XForms	SyncML	XML-RPC
Meta Content Framework	XUL / XBL		Webml

**Figura 1.3**  
Ejemplos de lenguajes de marcas según su aplicación específica.

- **Business-to-business.** Es la transmisión de información, referente a transacciones comerciales electrónicas, que permite establecer relaciones entre empresas para compartir información que les permita establecer negocios comunes en la red.
- **Lenguajes especializados de marcación.** También podemos encontrar la aplicación de los lenguajes de marcas a ámbitos específicos, como pueden ser los videojuegos y las matemáticas (Figura 1.4).

```
#VRML V2.0 utf8
#Ejemplo de agrupación de una caja y un cono,
#haciendo uso de los comandos DEF y USE.

Group {
    children [
        Shape {
            appearance DEF PorDefecto Appearance {
                material Material { }
            }
            geometry Box {
                size 2.0 0.5 3.0
            }
        },
        Shape {
            appearance USE PorDefecto
            geometry Cone {
                height 3.0
                bottomRadius 0.75
            }
        }
    ]
}
```

**Figura 1.4**  
Ejemplo de lenguaje OpenMath, utilizado en el campo de las matemáticas.

## 1.3 Clasificación

Hay tres clases de lenguajes de marcas, aunque en la práctica pueden combinarse varias clases en un mismo documento. Los lenguajes de marcado pueden clasificarse en:

- **De presentación.** Muestra la información necesaria para dar formato al texto. Se utiliza generalmente para maquetar documentos, pero no es aplicable al procesamiento automático de la información porque resulta poco flexible y difícil de mantener. El lenguaje de marcas de presentación resulta más fácil de elaborar, sobre todo para cantidades pequeñas de información. Sin embargo, es complicado de mantener o modificar, por lo que su uso se ha ido reduciendo a proyectos grandes en favor de otros tipos de marcado más estructurados. Se puede tratar de averiguar la estructura de un documento de esta clase buscando pistas en el texto. Por ejemplo, el título puede ir precedido de varios saltos de línea, y estar centrado en la página. Varios programas pueden deducir la estructura del texto basándose en esta clase de datos, aunque el resultado suele ser bastante imperfecto.
- **De procedimientos.** Se trata de lenguajes enfocados a la presentación del texto; sin embargo, las etiquetas pueden incluir la ejecución de algún comando o rutina que indica cómo debe tratarse el texto (Figura 1.5).

```
<html>
<head> <title>COBACH</title> </head>
```

```
<body bgcolor="#C0D9D9" text="#000000">
```

```
    <h1 align="center">
```

```
        COLEGIO DE BACHILLERES DEL ESTADO DE BAJA CALIFORNIA</h1><br>
```

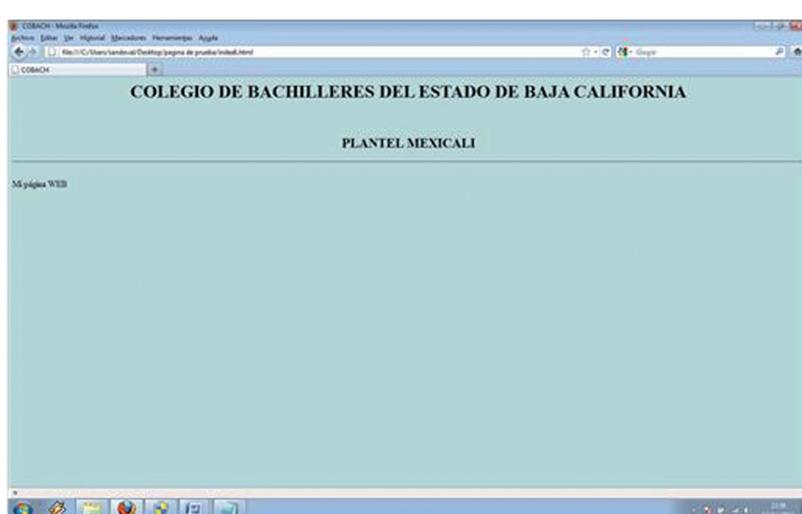
```
    <h2 align="center">PLANTEL MEXICALI </h2>
```

```
    <hr align="center" width=100%><br>
```

```
    Mi p&aacute;gina WEB
```

```
    </body>
```

```
</html>
```



**Figura 1.5**

Ejemplo de lenguaje de procedimientos HTML.

Normalmente las etiquetas son visibles para el usuario que edita el texto. Ejemplos de estos lenguajes son el HTML (Figura 1.6) o el TeX. El lenguaje de marcas de procedimiento debe utilizar un programa que representa el documento, que debe interpretar el código en el mismo orden en que aparece. Por ejemplo, para formatear un título, debe haber antes del texto en cuestión una serie de directrices que indiquen al software cómo centrar, aumentar el tamaño de la fuente o cambiar a negrita. Inmediatamente después del título, deberá haber etiquetas inversas que reviertan estos efectos. En sistemas más avanzados, se utilizan macros o pilas que facilitan el trabajo. Este tipo de marcado se ha utilizado extensamente en aplicaciones de edición profesional, manipulados por tipógrafos cualificados, puesto que puede llegar a ser extremadamente complejo.



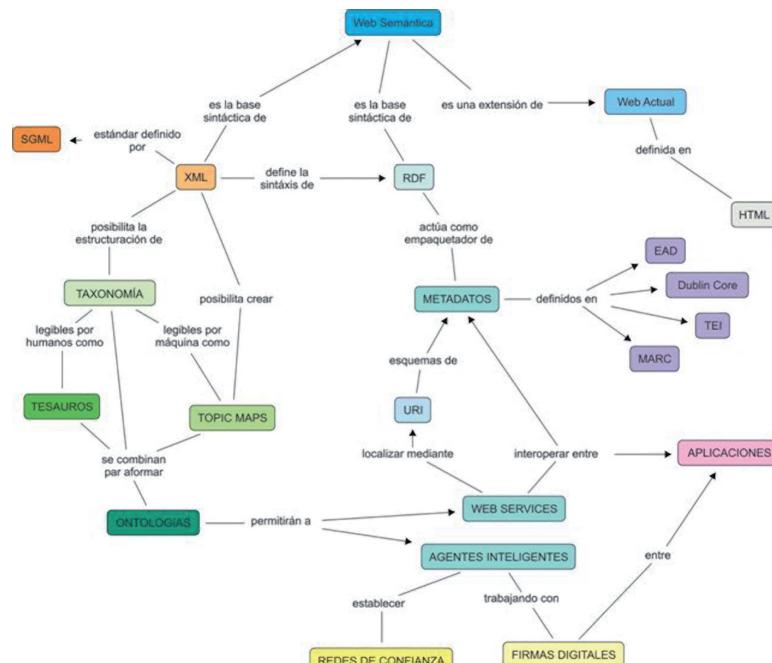
**Figura 1.6**

Ejemplo de lenguaje de marcas, en este caso el lenguaje HTML.

- **Descriptivo o semántico.** Describe las diferentes partes en las que se estructura el documento, pero sin especificar cómo debe representarse. Utiliza etiquetas para describir los fragmentos de texto, pero sin detallar cómo deben ser representados o en qué orden. Los lenguajes expresamente diseñados para generar marcado descriptivo son el SGML y el XML. Las etiquetas pueden utilizarse para añadir cualquier clase de metadatos al contenido. Por ejemplo, el estándar Atom, un lenguaje de gestión, proporciona un método para marcar la hora “actualizada”, que es el dato facilitado por el editor de cuándo ha sido modificada por última vez cierta información. El estándar no especifica cómo debe representarse, o siquiera si se debe representar. El software puede emplear este dato de múltiples maneras, e incluir algunas no previstas por los diseñadores del estándar.

Una de las ventajas del marcado descriptivo es su flexibilidad: los fragmentos de texto se etiquetan tal como son, y no tal como deben aparecer. Estos fragmentos pueden utilizarse para más usos de los previstos inicialmente. Por ejemplo, los

hiperenlaces fueron diseñados, en un principio, para que un usuario que leyese el texto los pulsase. Sin embargo, hoy en día, los buscadores los emplean para localizar nuevas páginas con información relacionada, o para evaluar la popularidad de determinado sitio web. El marcado descriptivo también simplifica la tarea de reformatear un texto, debido a que la información del formato está separada del propio contenido. Por ejemplo, un fragmento indicado como cursiva (<i>texto</i>) puede emplearse para marcar énfasis o para señalar palabras en otro idioma. El lenguaje de marcas descriptivo está evolucionando hacia el marcado genérico. Los nuevos sistemas de marcado descriptivo estructuran los documentos en árbol, con la posibilidad de añadir referencias cruzadas. Esto permite tratarlos como bases de datos, en las que el propio almacenamiento tiene en cuenta la estructura, y no como los grandes objetos binarios del pasado. Estos sistemas no tienen un esquema estricto como las bases relacionales, por lo que a menudo se les considera bases semiestructuradas. Dentro de la categoría descriptiva o semántica debemos destacar los lenguajes XML (eXtensible Markup Language) y SGML (Figura 1.7). Ambos son estándares del W3C, que es el organismo encargado de la estandarización de las tecnologías sobre las que se sustenta la Web. Estos lenguajes no muestran qué hacer con el texto, sino qué es el texto. Como su propio nombre indica, lo describen. La idea clave es que las marcas no determinan el procesamiento del documento de manera fija (como los procedimentales), ya que dicho procesamiento se determina a partir de necesidades concretas, y se beneficia de la estructura lógica del documento caracterizada a través de sus marcas.



**Figura 1.7**

SGML y XML son lenguajes diseñados expresamente para generar marcado descriptivo.

## 1.4 XML: Estructura y sintaxis

El **lenguaje XML** (*eXtensible Markup Language*) se utiliza para transportar y almacenar datos. Aunque pueda parecerse al HTML, sus aplicaciones son distintas: el XML se centra en lo que son los datos (hay que recordar que es un lenguaje descriptivo), mientras que el HTML se encarga de cómo tienen que visualizarse esos datos. Por tanto, XML no es un sustituto del HTML, sino un complemento.

En XML no existen etiquetas predefinidas, sino que es el usuario quien las crea (Figura 1.8).

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<cita>
    <dia>lunes</dia>
    <lugar>dentista</lugar>
    <nota>nuevo empaste</nota>
</cita>

```

### Recuerda

**XML es un lenguaje de marcas independiente del sistema operativo y está indicado para almacenar y transportar datos entre aplicaciones; de ahí que simplifique el compartir datos entre programas y la realización de cambios en la configuración de los sistemas. Permite también crear otros lenguajes a partir de él, como el XHTML, por lo que se dice que es un metalenguaje.**

**Figura 1.8**

Ejemplo de un documento XML en el que se definen sus etiquetas personalizadas.

Los archivos XML, en sí mismos, no hacen nada más que describir los datos. Son archivos de texto plano editables con cualquier editor.

Desde que, en febrero de 1998, el XML se convirtió en una recomendación del W3C, su popularidad fue aumentando hasta ser, en la actualidad, uno de los pilares de la web. Asimismo, su uso fuera de las tecnologías web está muy extendido.

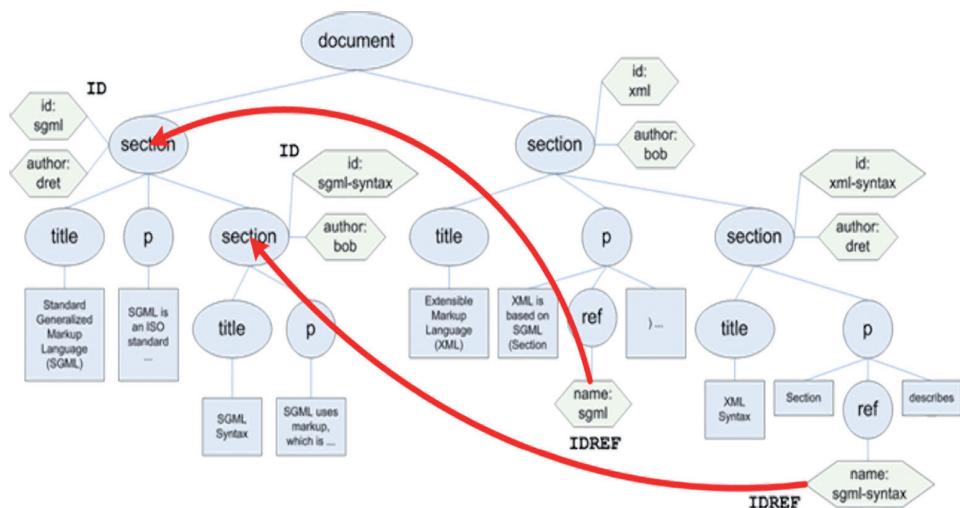
### 1.4.1 Estructura

La **estructura** de los documentos XML se compone del **prólogo** y el **elemento raíz**. El prólogo es la primera zona del documento, y sirve para describir de qué tipo de documento se trata. Puede contener la declaración del documento, que permite indicar el tipo de documento XML que es, las instrucciones para el procesado del documento, comentarios, la indicación del documento DTD o el esquema para la validación. Dentro del prólogo, en la primera línea del lenguaje, se indica la declara-

ción XML, en la que se define la versión y la codificación a utilizar; por ejemplo, el código *ISO-8895-1 = Latin-1* indica el conjunto de caracteres de Europa Occidental. En realidad es opcional, pero es muy recomendable. La expresión `<?xml version="1.0" encoding="UTF-8"?>` indica la versión XML del documento y la codificación utf-8 es la habitual.

Todo documento de un lenguaje de marcas tiene una gramática que define el marcado permitido, el marcado requerido y cómo debe ser utilizado dicho marcado en la instancia del documento.

El estándar define esta gramática mediante la DTD (Definición de Tipo de Documento) (Figura 1.9), que establece las reglas de formación del lenguaje formal, es decir, qué combinaciones de símbolos elementales son sintácticamente correctas.



**Figura 1.9**

Estructura de DTD (*Document Type Definition*).

En la DTD se identifica la estructura del documento, es decir, aquellos elementos que son necesarios para la elaboración de un documento o un grupo de documentos estructurados de manera similar. Contiene, además, las reglas de dichos elementos: los nombres, su significado, dónde pueden ser utilizados y qué pueden contener. La especificación del W3C para HTML 4.0 contempla tres DTD:

- **DTD estricta (HTML 4.0 Strict DTD).** Incluye todos los elementos y atributos que no han sido declarados “desaprobados” (*deprecated*), si tenemos en cuenta la expresión en el sentido de que no se recomienda ya su uso y se proponen nuevos y mejores recursos para hacer lo mismo.
- **DTD transicional o flexible -loose-(HTML 4.0 Transitional DTD).** Incluye todo lo de la anterior, además de los elementos y atributos desaprobados (*deprecated*).

- **DTD para documentos con marcos (HTML 4.0 Frameset DTD).** Engloba todo lo incluido en la transicional, además de lo relativo a la creación de documentos con marcos (*frames*).

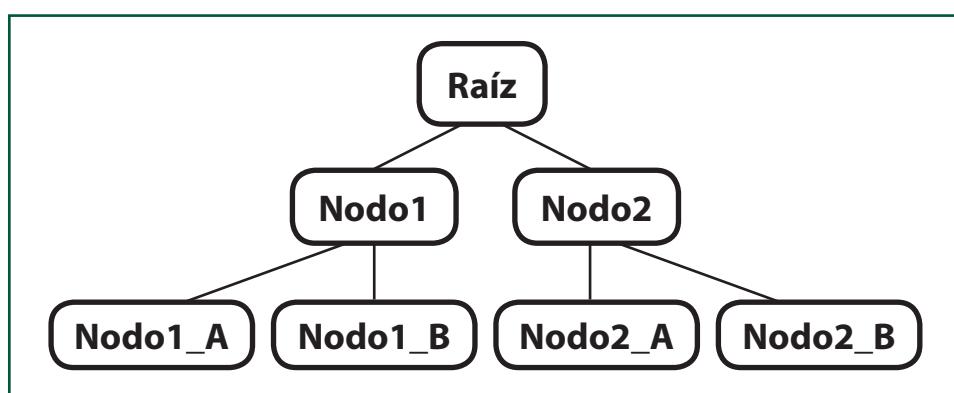
La DTD es el formato de esquema nativo (y el más antiguo) para validar documentos XML, heredado de SGML. Utiliza una sintaxis no-XML para definir la estructura o modelo de contenido de un documento XML válido, es decir, define todos los elementos, las relaciones entre los distintos elementos, proporciona información adicional que puede ser incluida en el documento (atributos, entidades, notaciones) y también aporta comentarios e instrucciones para su procesamiento y representación de los formatos de datos. Es el método más sencillo que se utiliza para validar y, por esta razón, presenta varias limitaciones, ya que no soporta nuevas ampliaciones de XML y no es capaz de describir ciertos aspectos formales de un documento a nivel expresivo.

### Recuerda

**Los documentos XML forman una estructura en árbol que comienza en la raíz y se bifurca hacia las hojas.**

Las DTD pueden ser internas o externas a un documento, o ambas cosas a la vez. El prólogo también puede incluir **instrucciones de procesamiento**. Un documento XML puede incluir instrucciones de este tipo para indicar un documento para validar el XML, darle formato, etc., u otras funciones. Por ejemplo: <?xmlstylesheet type="text/xsl" href="stylesheet.xsl"?> Esta instrucción asocia un documento XSL al documento XML para poder darle un formato de salida, por ejemplo para especificar la forma en la que los datos se muestran por pantalla.

Todo el contenido del documento debe estar incluido en el llamado **elemento raíz** (Figura 1.10). Se trata de un elemento obligatorio que se abre tras el prólogo y debe cerrarse justo al final. De este modo, cualquier elemento se halla dentro del elemento raíz. Éste contiene más elementos hijos, atributos, texto normal, entidades y comentarios. La segunda línea describe el elemento raíz del documento, y las siguientes líneas mostrarán los elementos hijos del elemento raíz. En la última línea se define dónde finaliza el elemento raíz y se cierra la etiqueta.



**Figura 1.10**

Los documentos XML tienen estructura de árbol, que se bifurca en distintas ramas.

### 1.4.2 Atributos

Los elementos son la base del documento XML. Sirven para dar significado al texto o a otros elementos, y también para definir relaciones entre distintos elementos y datos. Un elemento puede contener simplemente texto u otros elementos, o bien ambas cosas. Los elementos deben abrirse y cerrarse con la etiqueta que sirve para definir el elemento; siempre debe cerrarse el último elemento que se abrió, e incluso puede haber elementos vacíos.

Los **atributos** se definen dentro de las etiquetas de apertura de los elementos. Se indica su nombre seguido del signo = y del valor (entre comillas) que se le da al atributo. Un elemento puede contener varios atributos. Los atributos son definiciones de características que proporcionan información adicional sobre un elemento:

- Los atributos no pueden contener múltiples valores (los elementos, sí).
- Los atributos no tienen estructura de árbol (los elementos, sí).
- Los atributos son difíciles de mantener (para cambios futuros en el archivo XML).

Un atributo también puede representarse como un elemento hijo.

No existen reglas sobre cuándo hay que usar atributos o elementos; pero, en general, es preferible utilizar elementos. De todas formas, existe una excepción en la que sí es aconsejable el uso de atributos, es decir, cuando se emplean para representar metadatos. Por ejemplo, en ocasiones hay que asignar un identificador único (ID) a un elemento (Figura 1.11).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<contactos>
<contacto id="501">
<nombre>Juan Pérez</nombre>
</contacto>
<contacto id="502">
<nombre>Manolo Martínez</nombre>
</contacto>
</contactos>
```

Figura 1.11

Ejemplo del uso de atributos para la representación de metadatos.

### 1.4.3 Sintaxis

La **sintaxis** de un documento XML es muy sencilla, ya que se basa en un documento de texto ASCII, una cabecera y un conjunto extendido de etiquetas. Sólo hay que tener en cuenta una serie de reglas sintácticas a la hora de escribir archivos en XML. Veamos cuáles son:

- Todos los elementos XML deben tener una etiqueta de cierre.
- El XML distingue entre mayúsculas y minúsculas.
- Los elementos XML tienen que estar correctamente anidados.
- Los documentos XML deben tener siempre un único elemento raíz.
- Los atributos XML deben ir entre comillas, simples o dobles.
- Existen caracteres especiales que tienen un determinado significado para el XML. Por ejemplo, el carácter < puede confundirse con el principio de una etiqueta. Para evitarlo se usan entidades de referencia (más adelante veremos cómo crearlas).
- La sintaxis para escribir un comentario dentro de un archivo XML es igual que en HTML (entre los signos <!-- -->).

### 1.5 Etiquetas

Los documentos XML son, en definitiva, documentos de lenguajes de marcas, en los que hay texto normal, y etiquetas (marcas) que permiten clasificar dicho texto al indicar su significado. Las **etiquetas** en XML se deciden según nos interese, pues no hay una serie de etiquetas que puedan utilizarse; de hecho, la función de XML es definir tipos de documentos etiquetados (Figura 1.12).

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <libros>
3
4    <libro>
5      <titulo>Programacion en PHP</titulo>
6      <pginas>921</pginas>
7      <autor>Anonimo</autor>
8      </libro>
9
10   <libro2>
11     <titulo>Programando con Cobol version revisada </titulo>
12     <pginas>11230</pginas>
13     <autor>Anonimo</autor>
14   </libro2>
15
16 </libros>
17

```

**Figura 1.12**

Muestra de etiquetas en XML.

El código es similar a HTML, sólo que las etiquetas se utilizan en función de nuestras necesidades. No obstante, el funcionamiento es el mismo, es decir, las etiquetas tienen un cierre, que se indica con el signo / antes del nombre de la etiqueta y afectan al texto; y otras etiquetas que están entre la apertura y el cierre, y que se indica con los signos <>. Las **etiquetas** son elementos que identifican y estructuran las diferentes partes de un documento XML (Figura 1.13). Existe una serie de reglas que deberemos tener en cuenta a la hora de escribir los nombres de las etiquetas:

- Pueden contener letras, números y otros caracteres.
- No pueden empezar con un número o un carácter de puntuación.
- No pueden empezar con las letras xml (o XML, Xml, etc.); excepto esta salvedad, puede usarse cualquier palabra.
- No pueden contener espacios.
- Pueden escribirse etiquetas sin contenido, pero hay que cerrarlas (<etiqueta></etiqueta>).

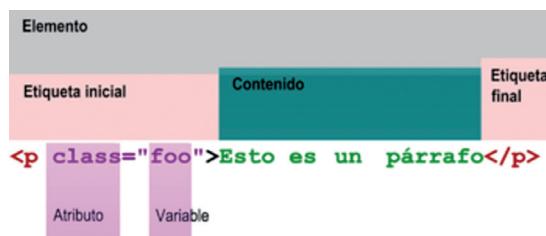


Figura 1.13

Estructura de una etiqueta.

Por otro lado, XML llama elemento a las etiquetas; mejor dicho, un elemento sería tanto la etiqueta como lo que ésta contiene. En cualquier caso las normas son que las etiquetas sirven para indicar elementos. El nombre de la etiqueta se indica entre los símbolos < y >; y las etiquetas se cierran indicando </, seguido del nombre de la etiqueta. XML, a diferencia de HTML, distingue entre mayúsculas y minúsculas, aunque es una buena práctica escribir las etiquetas en minúsculas siguiendo un patrón. Éstas pueden espaciarse y tabularse según se desee. Es conveniente que un elemento interno en otro aparezca en el código con una sangría mayor con el fin de facilitar la interpretación; y, según la W3C, el texto en un documento XML debe de estar codificado en Unicode (normalmente UTF-8).

### 1.5.1 Consejos de nomenclatura

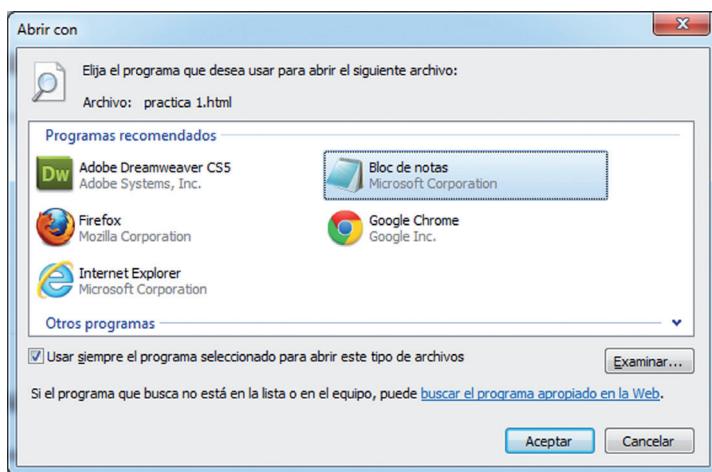
En XML, los elementos, atributos, etc., tienen nombre, el cual debe cumplir estas reglas: en XML se distingue entre mayúsculas y minúsculas, por lo que hay que tener cuidado al utilizar el nombre desde otro punto del documento. Éste debe comenzar, además, por una letra, a la que después le seguirán más letras, números

o el signo de subrayado o guión bajo. Veamos algunos consejos básicos a la hora de escribir los nombres de las etiquetas:

- Utilizar nombres descriptivos, cortos y simples.
- Sustituir los espacios por el guión bajo.
- No usar los caracteres - . :
- Los documentos XML suelen tener su correspondencia en una base de datos, por lo que conviene utilizar las reglas de nomenclatura de la base de datos del documento.
- Los caracteres no ingleses son legales en XML, pero pueden tener problemas de compatibilidad.

## 1.6 Herramientas de edición

En principio, XML puede escribirse desde cualquier editor de texto plano (como el Bloc de notas de Windows o el editor vi de Linux); sin embargo, es más conveniente hacerlo con un editor que reconozca el lenguaje y que, además, marque los errores en el mismo. De hecho, el software necesario es el siguiente: un editor de texto plano (como el Bloc de notas de Windows) (Figura 1.14) o un editor que reconozca el lenguaje XML para facilitar su escritura; un analizador o *parser*, programa capaz de entender y validar el lenguaje XML; y, por último, un motor XML (un navegador) que sea capaz de producir el resultado de los archivos XML en el formato indicado por el creador (mediante CSS o XSL por ejemplo). Hay entornos de trabajo que incluyen todas esas prestaciones dentro del mismo paquete de software. Como los archivos XML son documentos de texto plano, pueden crearse con cualquier programa de edición, por simple que éste sea. De todos modos, es más fácil crear los documentos con editores específicos para este propósito.



**Figura 1.14**

Bloc de notas de Windows.

Las **herramientas de edición** de XML son programas que nos facilitan la edición y validación de los documentos XML con respecto a una DTD o un esquema, lo que facilita la creación de estructuras XML válidas. Un editor XML proporciona las siguientes funcionalidades:

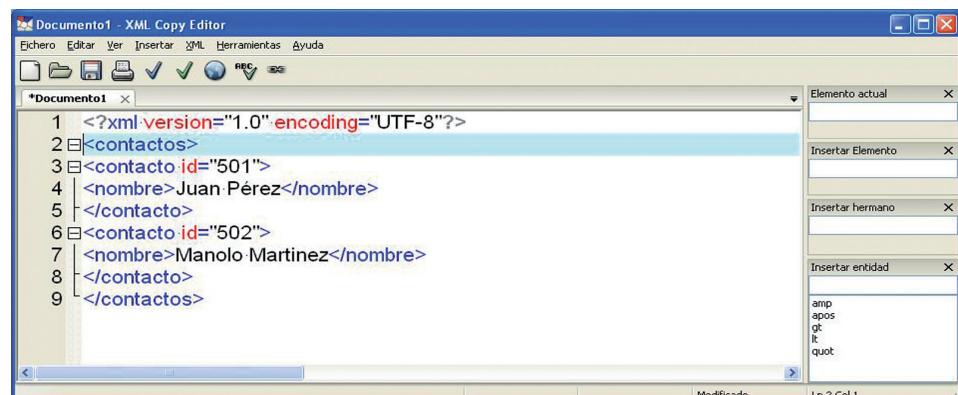
- Añade etiquetas de cierre automáticamente.
- Fuerza la escritura de un documento XML bien formado, es decir, haciendo cumplir las reglas de sintaxis de XML.
- Verifica el XML con respecto a una DTD o esquema.
- Resalta las etiquetas y sintaxis del XML mediante colores.
- También hay editores que proporcionan vistas para crear datos estructurados de manera gráfica.

Entre los editores XML comerciales podemos destacar los siguientes:

- oXygen XML Editor (Windows, Linux, Mac OS X)
- EditiX XML Editor (Windows, Linux, Mac OS X)
- Altova XMLSpy XML Editor (Windows)
- Stylus Studio (Windows)
- XMLmind XML Editor (Windows, Linux, Mac OS X)
- XMLwriter XML Editor (Windows)
- Liquid XML Studio (Windows)
- Serna Enterprise XML Document Editor (Windows, Linux, Mac OS X, Solaris)

También podemos destacar los siguientes editores de código abierto (*Open Source*):

- Serna Free Open Source XML Editor (Windows, Linux, Mac OS X, Solaris)
- XML Copy Editor (Linux) (Figura 1.15)
- Quanta Plus (Linux)
- Bluefish (Windows, Linux, Mac OS X)



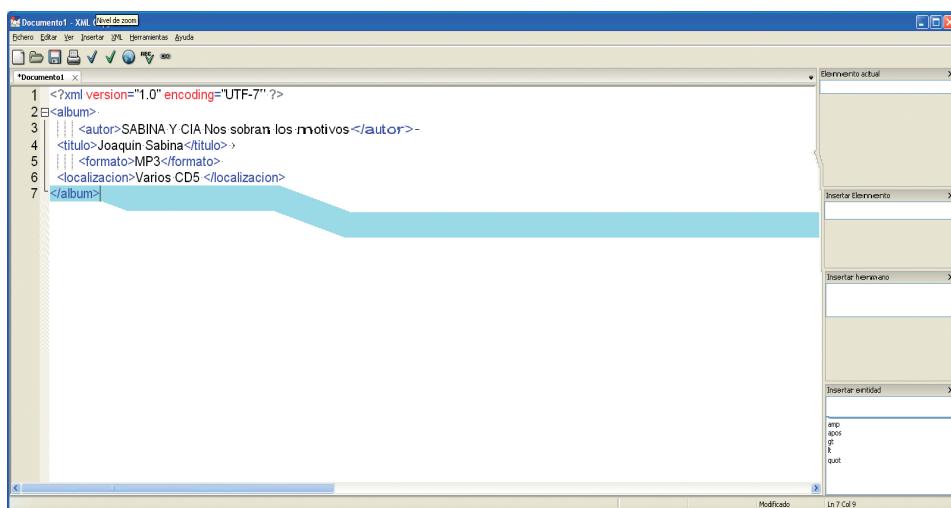
**Figura 1.15**

Ejemplo del editor  
XML Copy Editor.

## 1.7 Elaboración de documentos XML bien formados

Las **sentencias en XML** consisten en una serie de etiquetas, llamadas **elementos**, y una serie de modificadores, llamados **atributos**. Las etiquetas pueden estar anidadas unas dentro de otras, pero cada etiqueta que se abra se tiene que cerrar, y siempre en el mismo orden. Al elemento que no tiene pareja (por no tener ningún contenido dentro), se le denomina **elemento vacío** y se indica con un / al final:

- Todos los documentos XML deben estar bien formados, lo que significa que deben cumplir los siguientes requisitos (Figura 1.16): si no se utiliza DTD, el documento deberá comenzar con una *Declaración de Documento Standalone*.
- Todas las etiquetas deben estar equilibradas: todos los elementos que contengan datos de tipo carácter deben tener etiquetas de principio y fin.
- Todos los valores de los atributos deben ir entrecomillados.
- Cualquier elemento vacío debe terminar con '/>' o debe impedirse que sea vacío, añadiéndole una etiqueta de fin.
- No debe haber etiquetas aisladas en el texto.
- Los elementos deben anidarse dentro de sus propiedades.
- Los nombres de las etiquetas pueden ser alfanuméricos, es decir, comenzar con una letra, e incluir los caracteres - y :.



**Figura 1.16**  
Ejemplo de documento XML.

Según la especificación del W3C (*World Wide Web Consortium*, que es una comunidad internacional que desarrolla estándares que aseguran el crecimiento de la web a largo plazo), un **documento XML** está bien formado si, al analizarlo, cumple la regla denominada *document*; es decir, que está compuesto por uno o más elementos; tiene un elemento llamado *raíz*, del que ninguna parte aparece en el contenido de ningún otro elemento; y el resto de elementos se anidan adecua-

damente. Además, respeta todas las restricciones de buena formación dadas en la especificación inicial, y cada una de las entidades analizadas que se referencian directa o indirectamente en el documento está bien formada.

## 1.8 Utilización de espacios de nombres en XML

Los documentos XML se suelen combinar con otros documentos XML existentes, bien para compartir información entre diferentes usuarios o bien porque es habitual que se desee utilizar uno o varios módulos desarrollados previamente por terceras personas. Esta “modularidad” es una característica esencial de XML y permite al desarrollador poder reutilizar el código existente, que, en muchos casos, ya ha sido depurado y ampliamente probado.

El problema que surge en estos casos es la posible colisión que puede producirse en los nombres de los elementos que conformen los módulos que se quieren utilizar. Para solucionar este problema, XML proporciona un mecanismo denominado **espacio de nombres**, que permite asignar nombres extendidos a los elementos, de forma que puedan evitarse de este modo las colisiones.

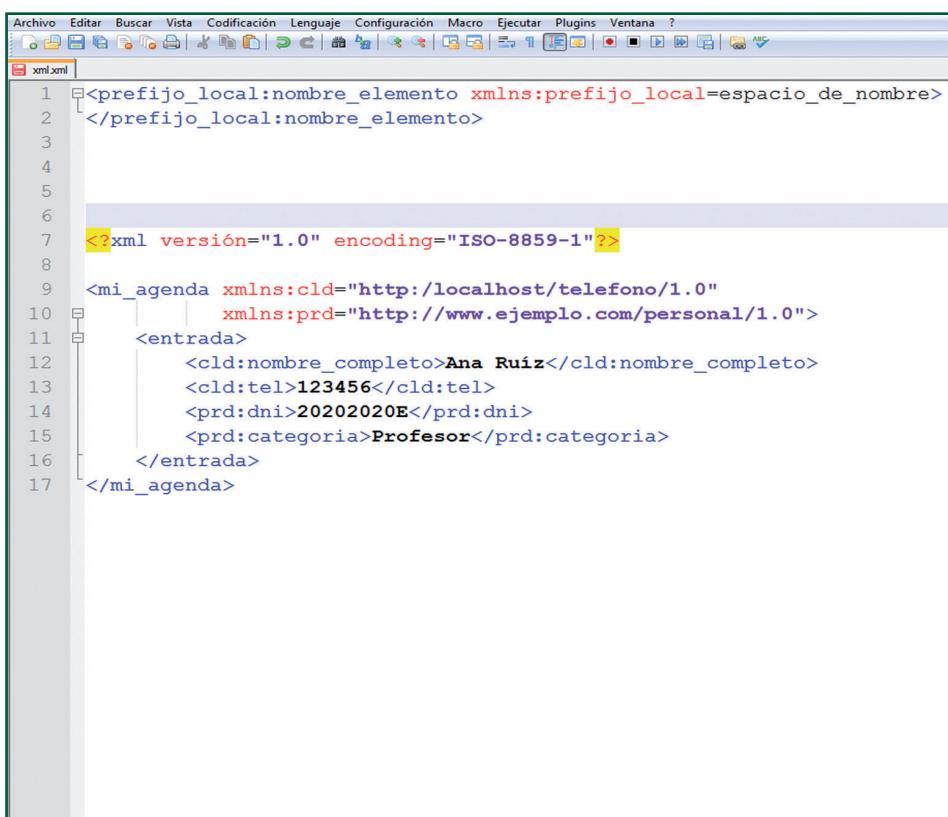
Un espacio de nombres se define como una referencia URI (*Uniform Resource Identifier*), que servirá para identificar los elementos que pertenecen a dicho espacio de nombres. Otra forma de definirlo es que los elementos tendrán un nombre compuesto por dos partes: una con su nombre, y una segunda con el nombre del espacio de nombres. Este nombre compuesto permitirá identificar, de forma unívoca, el elemento en cuestión y, de este modo, saber siempre a qué elemento se está refiriendo el documento.

La construcción de estos nombres extendidos se realiza uniendo el nombre al espacio de nombres y el nombre del elemento o atributo usando como conector el símbolo “:`. Sin embargo, las referencias URI pueden ser largas, lo que dificulta la legibilidad y claridad del documento, lo que propicia, además, que se cometan errores con mayor facilidad. Además, las URL pueden contener caracteres no válidos en XML. Para solucionar este problema, en XML se puede asignar un sinónimo corto al espacio de nombres, de modo que este sinónimo corto sea el que se utilice a lo largo del documento. El sinónimo se asigna utilizando el separador `: y la etiqueta “xmlns”. En realidad, “xmlns” es un atributo reservado (debemos recordar que los atributos no pueden comenzar por “xml” en ninguna combinación de mayúsculas y minúsculas).

Si un espacio de nombres se declara sin su sinónimo correspondiente, esto indicará que todos los elementos (incluido el elemento que declara el espacio de nombres) que contenga pertenecerán a dicho espacio de nombres. Esto será así

siempre que los elementos no tengan el prefijo de otro espacio de nombres; por lo tanto, sería como definir un espacio de nombres por defecto para los elementos que no tengan espacio de nombres asignado.

Otro uso de los espacios de nombres que puede resultar de gran utilidad es dejar su declaración en blanco (`xmlns=""`), lo que indicaría que los elementos y atributos contenidos por defecto no pertenecen a ningún espacio de nombres. Hay dos formas de asignar los espacios de nombres: **Espacios de nombre por defecto**, mediante un **atributo** (a todo un elemento y su contenido) y **espacios de nombre** (Figura 1.17) mediante **prefijos** (a cada elemento particularmente).



The screenshot shows a XML editor interface with a menu bar (Archivo, Editar, Buscar, Vista, Codificación, Lenguaje, Configuración, Macro, Ejecutar, Plugins, Ventana, ?) and a toolbar with various icons. The main window displays the following XML code:

```

1 <prefijo_local:nombre_elemento xmlns:prefijo_local=espacio_de_nombre>
2   </prefijo_local:nombre_elemento>
3
4
5
6
7   <?xml versión="1.0" encoding="ISO-8859-1"?>
8
9   <mi_agenda xmlns:cld="http://localhost/telefono/1.0"
10    xmlns:prd="http://www.ejemplo.com/personal/1.0">
11     <entrada>
12       <cld:nombre_completo>Ana Ruiz</cld:nombre_completo>
13       <cld:tel>123456</cld:tel>
14       <prd:dni>20202020E</prd:dni>
15       <prd:categoría>Profesor</prd:categoría>
16     </entrada>
17   </mi_agenda>

```

**Figura 1.17**

Ejemplo de espacio de nombres en XML.

### 1.8.1 Espacio de nombres por defecto (atributo)

El espacio de nombre por defecto se define mediante un atributo “`xmlns`”. El espacio de nombre se aplicará al elemento en el que se incluye el atributo “`xmlns=”`” y a todo su contenido (siempre que no esté asociado a otro espacio de nombres). La sintaxis es la siguiente:

`xmlns="URI del espacio de nombres"`

### 1.8.2 Espacios de nombres mediante prefijos

Otra forma de especificar los espacios de nombres es con el uso de prefijos en los elementos. En este caso, la sintaxis es la siguiente:

```
xmlns:prefix="URI del espacio de nombres"
```

El prefijo es tan sólo una forma corta para referirse al identificador (la URI), que es el que realmente identifica el espacio de nombres (Figura 1.18).



**Figura 1.18**

Ejemplo del uso de espacios de nombres en documentos XML.

## Resumen

Un lenguaje de marcas es un sistema que permite incluir anotaciones, llamadas marcas o etiquetas, en un texto. Estas etiquetas se utilizan para presentar el texto, por ejemplo cuando éste se visualiza en un navegador web, o para proporcionar información sobre la estructura del mismo.

La aplicación de los lenguajes de marcas es muy diversa y se emplea para cualquier tipo de documento (textos, presentaciones, gráficos, tecnologías de Internet, matemáticas, música, multimedia, etc.).

Los lenguajes de marcas se clasifican en: presentación (muestra la información necesaria para dar formato al texto); procedimientos (lenguajes enfocados a la presentación del texto); descriptivos o semánticos (describen las diferentes partes en las que se estructura el documento pero sin especificar cómo debe representarse), como es el caso de XML (*eXtensible Markup Language*)

El lenguaje XML se basa en lo que son los datos, mientras que el HTML se encarga de cómo tienen que visualizarse dichos datos; por lo tanto XML no es un sustituto del HTML, sino un complemento.

Un documento DTD (Definición de Tipo de Documento) establece las reglas de formación del lenguaje formal, es decir, qué combinaciones de símbolos elementales son sintácticamente correctas en un documento XML.

La sintaxis de un documento XML es muy sencilla, ya que se basa en un documento de texto ASCII, una cabecera y un conjunto extendido de etiquetas. Sin embargo, hay una serie de reglas sintácticas que deben tenerse en cuenta (etiquetas de cierre en todos los elementos, distinción entre mayúsculas y minúsculas, anidación correcta, un único elemento raíz, etc.).

XML proporciona un mecanismo denominado “espacio de nombres”, que permite asignar nombres extendidos a los elementos, de forma que puedan evitarse las colisiones. Un espacio de nombres se define como una referencia URI (Uniform Resource Identifier), que servirá para identificar los elementos que pertenecen a dicho espacio de nombres.

Las herramientas de edición en XML son programas que nos facilitan la edición y validación de los documentos XML con respecto a un DTD o un esquema, por lo que facilitan la creación de estructuras XML válidas.

## Ejercicios de autocomprobación

**Indica si las siguientes afirmaciones son verdaderas (V) o falsas (F):**

1. El lenguaje de marcas puede definirse como la forma de codificar un documento al que, junto con el texto, se incorporan una serie de etiquetas o marcas que contienen información adicional acerca de la estructura del texto o de su presentación.
2. Son características principales de los lenguajes de marcas el uso de etiquetas o marcas, la flexibilidad, la compactación y la facilidad de procesamiento.
3. Las clases de lenguajes de marcas pueden clasificarse en: de presentación, de procedimientos y descriptivo o semántico.
4. El lenguaje XML se utiliza para transportar y almacenar datos. El XML se centra en lo que son los datos (es un lenguaje descriptivo), mientras que el HTML se encarga de cómo tienen que visualizarse esos datos. Por tanto, XML es un sustituto del HTML.
5. La estructura de los documentos XML se compone del prólogo y el elemento raíz. El prólogo es la primera zona del documento, y sirve para describir de qué tipo de documento se trata. El llamado elemento raíz es un elemento obligatorio que se abre tras el prólogo y debe cerrarse justo al final.
6. A la hora de escribir archivos en XML una de las reglas sintácticas es que todos los elementos tengan una etiqueta de cierre.
7. Las etiquetas son elementos que identifican y estructuran las diferentes partes de un documento XML. A la hora de escribir, los nombres de las etiquetas pueden contener espacios.

**Completa las siguientes afirmaciones:**

8. Un lenguaje de marcas se caracteriza por ser un \_\_\_\_\_ que permite incluir \_\_\_\_\_, llamadas marcas o \_\_\_\_\_, en un \_\_\_\_\_. Éstas sirven para indicar cómo se quiere presentar el texto cuando se visualice en un navegador web. De modo que, el lenguaje \_\_\_\_\_ utiliza la etiqueta <b>...</b> para indicar que el texto se debe representar en negrita: <b> Esto se vería en negrita </b>

9. Los elementos son la base del documento \_\_\_\_\_. Sirven para dar \_\_\_\_\_ al texto o a otros elementos, y también para definir relaciones entre distintos elementos y \_\_\_\_\_. Un elemento puede contener simplemente texto u otros elementos, o bien ambas cosas. Los elementos deben abrirse y cerrarse con la \_\_\_\_\_ que sirve para definir el elemento; siempre debe cerrarse el último elemento que se abrió, e incluso puede haber elementos \_\_\_\_\_.
10. \_\_\_\_\_ proporciona un mecanismo denominado “espacio de nombres”, que permite asignar nombres extendidos a los \_\_\_\_\_, de forma que puedan evitarse las colisiones. Un espacio de nombres se define como una referencia \_\_\_\_\_, que servirá para identificar los elementos que pertenecen a dicho espacio de nombres.

Las soluciones a los ejercicios de autocomprobación se encuentran al final de esta Unidad Formativa. En caso de que no los hayas contestado correctamente, repasa la parte de la lección correspondiente.

## 2. UTILIZACIÓN DE LENGUAJES DE MARCAS EN ENTORNO WEB

En esta unidad estudiaremos los elementos básicos para crear la estructura de una página web escrita en **lenguaje HTML**. La estructura de un documento HTML ha de seguir una estructura jerárquica en árbol que parte del elemento raíz (HTML), del que cuelgan todos los demás elementos. Estos elementos hijos deberán estar anidados y cerrados correctamente.

Las fases de un desarrollo web, así como los lenguajes de programación utilizados, son muy extensos y variados. Por ello, necesitaremos herramientas específicas para cada una de estas fases. Conoceremos también las principales herramientas existentes para poder desarrollar fácilmente un proyecto web. En el desarrollo web, todas las herramientas para el diseño que utilicemos (para la maquetación, la programación y la depuración) son muy importantes, razón por la que es preciso elegir aquella que resulte más adecuada a nuestras necesidades y capacidades (Figura 2.1).

The figure consists of two side-by-side screenshots. On the left, a code editor titled 'Html Html' displays the following HTML code:

```
1 <html>
2 <head>
3 <title>¡Hola Mundo!</title>
4 </head>
5 <body>
6 <h1>¡Hola Mundo!</h1>
7 <p>Este es mi primer documento HTML</p>
8 </body>
9 </html>
```

On the right, a screenshot of a Firefox browser window shows the rendered output:

**¡Hola Mundo!**  
Este es mi primer documento HTML

A red box highlights the title '¡Hola Mundo!' in the browser window, with the text 'DOCUMENTO HTML' written above it in red. Another red box highlights the entire browser window, with the text 'PRESENTACIÓN DEL DOCUMENTO EN EL NAVEGADOR' written below it in red.

**Figura 2.1**

Ejemplo de presentación de un documento HTML mediante un navegador web.

### 2.1 HTML: Estructura de una página web

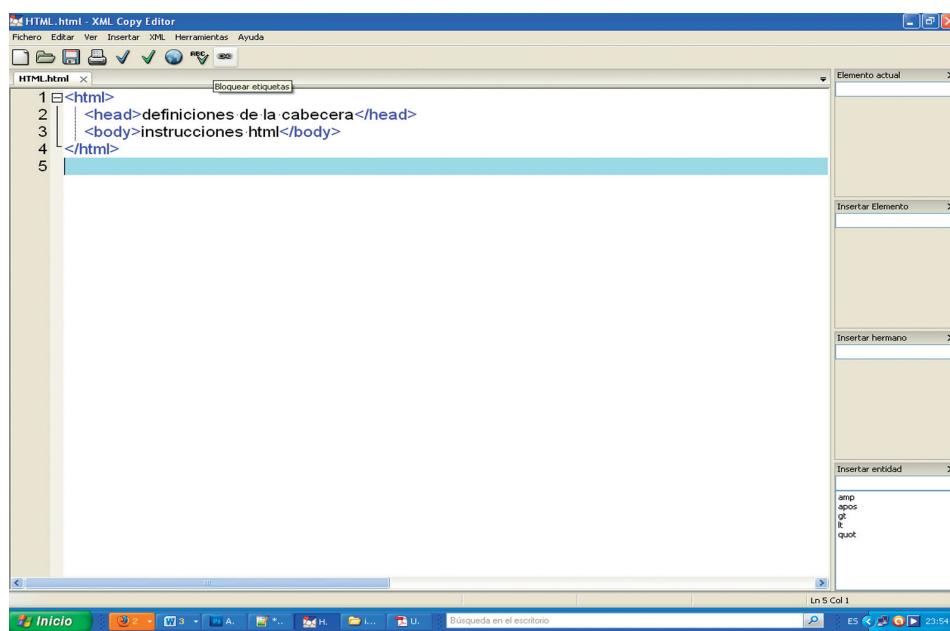
El **HTML** (*HyperText Markup Language* o lenguaje de marcado de hipertexto) es un lenguaje de marcas que basa su sintaxis en un elemento de base al que llamamos **etiqueta**. A través de las etiquetas definiremos los elementos del documento, como enlaces, párrafos, imágenes, etc. Así pues, un documento HTML estará

constituido por texto y un conjunto de etiquetas para definir la forma con la que presentaremos el texto y otros elementos en la página.

HTML es el lenguaje con el que se escriben las páginas web. Las páginas web son vistas por el usuario mediante un navegador, de modo que, por lo tanto, podemos decir que el HTML es el lenguaje utilizado por los navegadores para mostrar las páginas web al usuario, y que, hoy en día, es la interfaz más extendida en la red.

El lenguaje HTML nos permite mostrar textos, sonidos e imágenes y combinarlos a nuestro gusto; nos permite, además, la introducción de referencias a otras páginas por medio de los enlaces hipertexto.

El principio de un documento HTML se indica con la etiqueta <html>, que marca al navegador de Internet que ahí comienza una página web. Esta etiqueta deberá cerrarse al final del documento. Un documento HTML debe estar delimitado por la etiqueta <html> y </html>. La estructura de la página web está dividida en dos partes: la cabecera (*head*) y el cuerpo (*body*), ambas incluidas dentro del archivo HTML (Figura 2.2). La etiqueta <head> indica el encabezado de la página, es decir, el área de la barra de título. La cabecera indica al navegador cómo debe mostrar la información del cuerpo. En la cabecera, definiremos el título de la página, e incluiremos información sobre la página para los motores de búsqueda, estableceremos la ubicación de la página, añadiremos hojas de estilo y escribiremos todos aquellos *scripts* que consideremos necesarios. Todos estos datos no serán visibles para el visitante de la página, salvo el título.



The screenshot shows the XML Copy Editor interface with the file 'HTML.html' open. The code editor displays the following structure:

```

1 <html>
2   | <head>definiciones de la cabecera</head>
3   | <body>instrucciones html</body>
4 </html>
5

```

The editor has several floating windows on the right side: 'Elemento actual' (Current element), 'Insertar Elemento' (Insert element), 'Insertar hermano' (Insert sibling), and 'Insertar entidad' (Insert entity). The status bar at the bottom shows 'Ln 5 Col 1'.

**Figura 2.2**  
Estructura básica de una página web en lenguaje HTML.

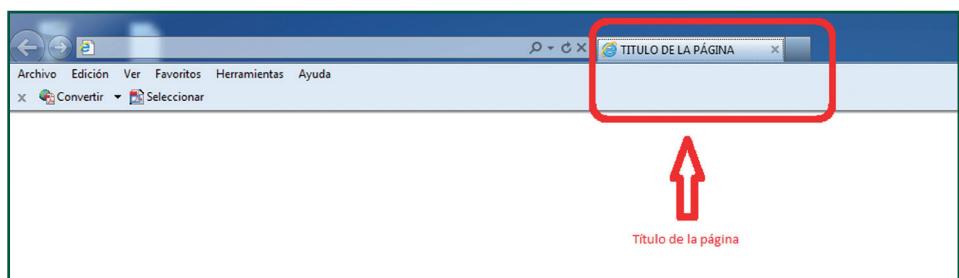
La etiqueta <body> es una instrucción que indica al explorador de Internet que ahí empieza el área de contenido de la página, es decir, el cuerpo de la página en donde se ubicará el contenido. El resultado es un documento con la siguiente estructura:

```
<html>
  <head>
    Etiquetas y contenidos del encabezado
    Datos que no aparecen en nuestra página, pero que son importantes para
    catalogarla: Título, palabras clave...
  </head>

  <body>
    Etiquetas y contenidos del cuerpo
    Parte del documento que será mostrada por el navegador: Texto e imágenes
  </body>
</html>
```

La cabecera del documento contiene información sobre éste: su título, información a utilizar por los programas buscadores y, en general, datos que no se consideran parte del contenido del documento, pero que aportan información sobre él (Figura 2.3). El componente obligatorio y más importante de la cabecera es el título del documento (elemento *Title*). Los navegadores no presentan como contenido lo incluido en *head*, aunque pueden hacer que esta información esté disponible mediante otros recursos. El elemento *head*, además de *Title*, puede contener, en cualquier orden, los siguientes elementos: *Meta*, *Link*, *Base*, *Isindex*, *Style* y *Script*.

**Figura 2.3**  
Ejemplo de cabecera de un documento.



El <*Title*> es el componente más importante de la cabecera y se introduce mediante el elemento *Title*. Todo documento HTML ha de incluirlo. El navegador no lo presentará junto al contenido del documento, pero hará que esté siempre disponible para el usuario. Los navegadores gráficos más comunes lo presentan en la barra de títulos de la ventana y lo utilizan como referencia al incluirlo en la lista de recursos favoritos del usuario (lo que constituye una razón más para que el título proporcione una descripción adecuada del documento). El elemento *Title* no pue-

de contener otros elementos ni tener ninguna indicación de formato. Sí, en cambio, pueden utilizarse entidades de caracteres para acentos, símbolos especiales, etc. No debe confundirse el elemento *Title*, que afecta a todo el documento, con el atributo *title*, que afecta a múltiples elementos. El cuerpo de un documento HTML constituye el contenido del mismo y será visualizado en la pantalla de su programa navegador (si se trata de un navegador no visual, será presentado como sonido, en braille, etc.). El cuerpo del documento puede contener una gran variedad de elementos que soportan una aún mayor variedad de atributos. La mayoría de los elementos que pueden aparecer en el cuerpo del documento pueden clasificarse en **elementos a nivel de bloque** y **elementos a nivel de texto o inline**.

## Para saber más

Puedes consultar todos los elementos de HTML, así como ejemplos de utilización, en la web de w3schools ([www.w3schools.com/html/](http://www.w3schools.com/html/)).

### 2.1.1 Elementos a nivel de bloque y elementos a nivel de línea

Generalmente, los elementos a nivel de bloque pueden contener otros elementos de bloque, además de elementos de línea. En cambio, los elementos a nivel de línea, generalmente, sólo pueden contener texto y otros elementos de línea. Un ejemplo de uso de un elemento de bloque sería div, que puede utilizarse para la distribución de los elementos en la forma que se presentarán en la pantalla.

```
<div style="float: right;">Texto</div>
```

Un ejemplo de uso de un elemento de línea sería strong, que da formato al texto de una línea al visulizarlo en negrita.

```
<strong>lo más importante</strong>
```

Los elementos de bloque organizan el texto en estructuras más grandes que los elementos de línea. Por defecto, los elementos a nivel de bloque se formatean de manera diferente que los elementos a nivel de línea; los primeros, generalmente, comienzan en una nueva línea y los segundos no. Por razones técnicas relacionadas con el algoritmo de texto bidireccional, unos y otros elementos difieren en cómo heredan la información acerca de la dirección del texto. Veamos a continuación cuáles son los elementos de bloque y los elementos de línea:

- **Elementos de bloque.** ADDRESS, BLOCKQUOTE, CENTER, DIR, DIV, DL, FIELDSET, FORM, HI, H2, H3, H4, H5, H6, HR, ISINDEX, MENU, NOFRAMES, NOSRIPT, OL, P, PRE, TABLE, UL.
- **Elementos de línea.** A, ACRONYM, APPLET, B, BASEFONT, BDO, BIG, BR, CITE, CODE, DFN, EM, FONT, I, IFRAME, IMG, KBG, MAP, OBJETCT, Q, S, SAMP, SCRIPT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TT, U, VAR.

## 2.2 Identificación de etiquetas y atributos de HTML

El lenguaje HTML utiliza etiquetas para comunicarse con el navegador web. Las etiquetas, también llamadas **tags**, se expresan entre los símbolos < y > e indican propiedades. Muchas de las etiquetas se cierran con el signo / (barra diagonal) para indicar al programa que las propiedades aplicadas con esa instrucción se terminan ahí. Si el signo de cierre (/) no se aplica convenientemente, el programa seguirá aplicando las propiedades de la instrucción anterior a todas las líneas inferiores hasta que se encuentre otra instrucción que modifique las propiedades actuales.

Es importante señalar que las etiquetas pueden escribirse indistintamente en mayúsculas o minúsculas; es decir, que <html>, <HTML> o <HtMl> son la misma etiqueta. Sin embargo, es aconsejable escribirlas en minúscula, ya que otras tecnologías pueden convivir con nuestro HTML (XML, por ejemplo) y no son tan permisivas, por lo que es recomendable tener en cuenta esa norma desde el principio para evitar fallos triviales en un futuro.

Por otro lado, toda etiqueta que se abre debe cerrarse; ya que, de lo contrario, el navegador podría dar resultados inesperados. Excepciones a esta regla son algunas etiquetas que no lo necesitan, como es el caso de <br> o <hr>.

Las etiquetas pueden contener **atributos**. Los atributos definen diferentes posibilidades de la instrucción HTML (Figura 2.4). Estos atributos se definirán en la etiqueta de inicio, y consistirán normalmente en el nombre del atributo y el valor que toma separados por un signo de igual.

El orden en que se incluyan los atributos es indiferente, pues no afecta al resultado. El valor de un atributo deberá estar indicado entre comillas, sobre todo cuando el valor que toma el atributo tiene más de una palabra. Algunos atributos pueden aceptar cualquier valor; pero otros, en cambio, son más limitados.

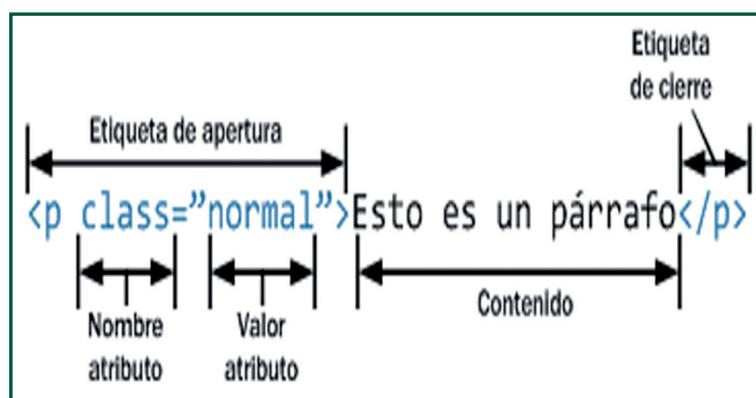


Figura 2.4

Esquema de la construcción de una etiqueta.

La etiqueta suele componerse de dos partes: una apertura de forma general, <etiqueta>, y un cierre de tipo </etiqueta>; y todo lo incluido en el interior de esa etiqueta sufrirá las modificaciones que caracterizan a dicha etiqueta (Figura 2.5).

```

1. <HTML>
2.   <HEAD>
3.     <title>¡Hola Mundo!</title>
4.   </HEAD>
5.
6.   <BODY>
7.     <h1>¡Hola Mundo!</h1>
8.     <p>Este es mi primer documento HTML</p>
9.   </BODY>
10. </HTML>
11.

```

**Figura 2.5**

Las etiquetas del documento HTML deben abrirse y cerrarse correctamente.

Dentro de las etiquetas <body></body> se representa el contenido del propio documento. Éste puede contener distintos tipos de elementos, de los cuales enumeraremos a continuación los más importantes:

- **<h1></h1>**. Define un encabezado de tipo 1. Pueden utilizarse desde <h1> hasta <h6> para representar un árbol de encabezados, siendo los de tipo 1 los de mayor nivel, si consideramos el <h1>, como título más importante, y utilizamos para su presentación en el navegador un tipo de letra de mayor tamaño y con formato de título.
- **<p></p>**. Sirve para representar un párrafo de texto.
- **<a href="URL">Enlace</a>**. Permite definir un hipervínculo a una URL. El contenido que aparece entre <a> y </a> será el que se represente en la página web.
- **<img />src="imagen.jpg" width="200" height="100" />**. Permite incorporar una imagen en el contenido que se va a representar. El atributo *src* sirve para indicar la URL en donde se encuentra la imagen, mientras que el atributo *alt* permite definir un texto alternativo a la imagen que se mostrará en navegado-

res sin soporte gráfico. Además, pueden utilizarse los atributos *width* y *height* para representar la imagen con un determinado tamaño: <imgsrc="Imagen.jpg" width=100 height=200 alt="Imagen">

- **<br>**. Inserta un salto de línea.
- **<b></b> y <i></i>**. Sirven para insertar texto en negrita y en cursiva, respectivamente.
- **<ul></ul>**. Define listas no numeradas en las que cada nuevo ítem se encuentra delimitado por <li></li>. Para definir listas ordenadas puede utilizarse la etiqueta <ol>.
- **<table></table>**. Sirve para escribir una tabla en el contenido que se desea representar. Durante un tiempo, las tablas se utilizaban para maquetar, lo que hoy en día no es recomendable. Las etiquetas <tr> </tr> sirven para empezar una nueva fila, mientras que las etiquetas <td> </td> sirven para delimitar una nueva columna.
- **<form></form>**. Permite crear formularios mediante las etiquetas <input>, <option>, <textarea>, <button>, etc. Los atributos *action* y *method* permiten definir la URL a la que se enviarán los datos del formulario y el método de envío (*get* o *post*).

Los elementos pueden llevar asociados propiedades llamadas **atributos**; los atributos pueden tener valores. Los pares atributo/valor siempre se colocan antes del final de la etiqueta de inicio del elemento; es decir, antes del signo '>', pero nunca en la etiqueta de fin. Puede utilizarse cualquier número de pares atributo/valor, separados por espacios. Los atributos pueden sucederse en cualquier orden.

El nombre del atributo va seguido por el signo "=" y el valor del atributo. El valor de los atributos se delimita por comillas dobles (ASCII decimal 34) o simples (ASCII decimal 39). Las comillas simples pueden formar parte del valor del atributo cuando éste se delimita por comillas dobles y viceversa. En algunos casos puede prescindirse de las comillas, aunque la especificación recomienda utilizarlas siempre. Los nombres de los atributos pueden escribirse indiferentemente con mayúsculas o minúsculas; los valores de los atributos generalmente también. El W3C hace una advertencia explícita y recomienda no confundir elementos y etiquetas. Dado que el uso del término etiquetas está tan extendido en la literatura sobre HTML, para evitar confusiones se adoptará la siguiente convención:

- Al utilizar el término *elemento*, nos referiremos tanto a sus etiquetas de inicio y fin (si ésta existe) como a su contenido (si éste existe).
- Al utilizar el término *etiqueta*, nos referiremos a las etiquetas de inicio y fin y, en general, a todo lo que encontramos entre cada par de símbolos '<' y '>'.

Tipos de atributos:

- **Atributos básicos.** Se pueden utilizar prácticamente en todas las etiquetas. Por ejemplo: style = "texto". Establece de forma directa los estilos CSS de un elemento.
- **Atributos para internacionalización.** Los utilizan las páginas que muestran sus contenidos en varios idiomas o aquellas que quieren indicar de forma explícita el idioma de sus contenidos. Por ejemplo: lang = "código de idioma". Indica el idioma del elemento mediante un código predefinido.
- **Atributos de eventos.** Sólo se utilizan en las páginas web dinámicas creadas con JavaScript. Por ejemplos: onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove. Permiten controlar los eventos producidos sobre cada elemento de la página.
- **Atributos para elementos que pueden obtener el foco.** Por ejemplo: tabindex = "numero". Establece la posición del elemento en el orden de tabulación de la página.

Debido a su importancia en HTML, debemos destacar los atributos **id** y **class**, que pueden ir asociados a casi todos los elementos:

- **Atributo id.** Asigna un nombre a un elemento en el documento. Este nombre no puede repetirse en un mismo documento, por lo que permite identificar cualquier instancia, cualquier aparición en el documento de un mismo elemento. Tiene varios usos en HTML:

- Como selector en una hoja de estilo.
- Como ancla de destino en enlaces de hipertexto.
- Como recurso para referenciar un elemento particular desde un script.
- Como nombre declarado de un elemento OBJECT.
- Como propósito general (por ejemplo, para identificar campos cuando se extraen datos desde una página HTML a una base de datos).

## Recuerda

También puedes encontrar información y ejemplos de utilización del lenguaje html en la página de W3Schools ([www.w3schools.com/html/](http://www.w3schools.com/html/)).

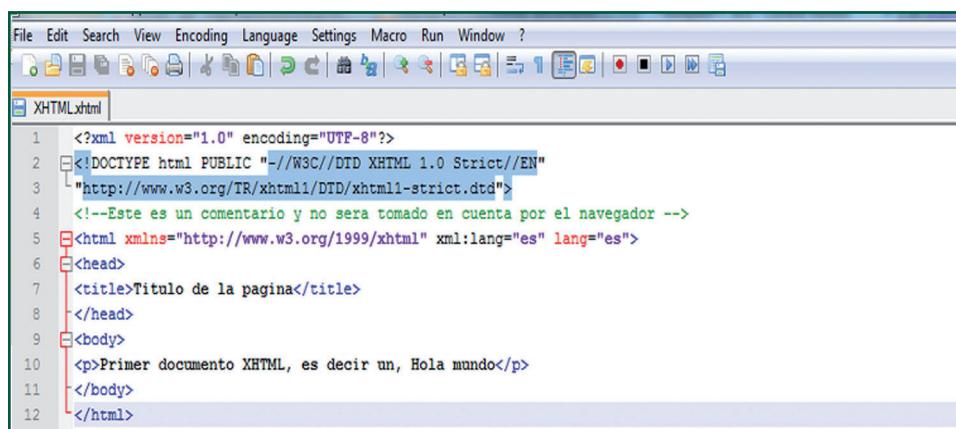
- **Atributo class.** Asigna uno o más nombres de clase a un elemento, por lo que puede decirse entonces que el elemento pertenece a esa clase o clases. Un mismo nombre de clase puede ser compartido por varias instancias (varias apariciones en un documento) de un mismo elemento y distintos elementos pueden pertenecer a la misma clase. Tiene varios usos en HTML:

- Como selector de estilo cuando el autor desea asignar información de estilo a un grupo de elementos.
- Como propósito general.

Además de los elementos y atributos mostrados a modo de ejemplo previamente, en HTML existen otros elementos y atributos. Para un conocimiento completo de HTML se recomienda consultar la web del consorcio W3C. El enlace es el siguiente: [www.w3c.es](http://www.w3c.es).

## 2.3 XHTML: Diferencias sintácticas y estructurales con HTML

**XHTML** (*eXtensible HyperText Markup Language*) es uno de los lenguajes derivados de XML utilizados en las páginas web. XHTML está basado en HTML, pero con la diferencia de que XHTML es más estricto con el estándar XML. De hecho, la primera versión del XHTML es una mera adaptación del HTML a la sintaxis del XML. XHTML sigue muchas de las convenciones de HTML, por esto los navegadores que soportan HTML podrán ver los documentos XHTML. XHTML, como su nombre indica, es más extenso (Figura 2.6).



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--Este es un comentario y no sera tomado en cuenta por el navegador --&gt;
&lt;html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es"&gt;
&lt;head&gt;
&lt;title&gt;Titulo de la pagina&lt;/title&gt;
&lt;/head&gt;
&lt;body&gt;
&lt;p&gt;Primer documento XHTML, es decir un, Hola mundo&lt;/p&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
```

**Figura 2.6**

El lenguaje XHTML (*Extensible Hypertext Markup Language*).

El hecho de que HTML sea un lenguaje estándar provoca que su evolución sea muy conservadora, lo que implica que sea difícil integrar funcionalidades adicionales que formen parte del estándar como tal. Ante esta problemática, muchos navegadores han sido diseñados para ejecutar tareas que se hallan fuera del estándar HTML, lo que permite diseños más versátiles. El uso de elementos HTML propietarios restringen que determinados diseños sean visualizados de la misma manera, o bien correctamente en distintos navegadores. A pesar de que los sitios más avanzados en Internet poseen mecanismos, mediante los *Headers* (Cabezas), para detectar el tipo de navegador que está solicitando información del documento y así retornar un diseño apropiado, esto requiere de un gran diseño y esfuerzo para poder mantener distintas versiones de acuerdo a los navegadores (Browsers) que soliciten información. Por esta razón, es recomendable, para sitios de Internet en general, utilizar el estándar HTML.

Veamos a continuación cuáles son las principales diferencias entre XHTML y HTML:

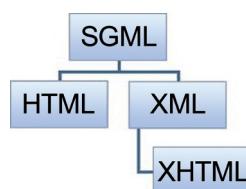
- Los nombres de las etiquetas de elementos y los nombres de los atributos deben ir en minúscula, con el fin de mantener el mismo criterio en todo el documento.
- Los valores de los atributos deben ir entre comillas dobles ("") o comillas simples ('').
- Todos los elementos tienen que estar cerrados, ya tengan contenido (<p>...</p>) o no (<br/>).
- Los elementos deben estar correctamente anidados.
- Los valores de pares atributo = valores iguales no pueden ser simplificados, por ejemplo <dlcompact ='compact'> no puede expresarse como <dlcompact>.
- Algunos elementos son obligatorios, como html, body, head, etc.
- Debe incluir una declaración de tipo de documento al principio del documento.
- Además, existen ciertas incompatibilidades en el anidamiento de elementos; por ejemplo, <a> no puede contener otros elementos <a>.
- No se puede escribir contenido en el body sin introducirlo en una etiqueta.
- Todos los atributos tienen valor, no se pueden minimizar los atributos; es decir, todos deben tener un valor asignado. Un ejemplo de ello es el atributo "selected" de un elemento OPTION que se indicaba para decir que debe aparecer como seleccionado por defecto. Debemos colocarlo siempre asignándole algún valor.
- No se deben insertar elementos *block* dentro de los elementos *inline*. Los elementos de tipo "bloque", como <p> o <div>, son más generales que los elementos de tipo "en línea", como <strong> o <em>. Por ello, no podemos colocar elementos tipo bloque dentro de otros menos generales como los en línea.
- Los *scripts* y estilos deben colocarse en bloque CDATA. Debido a las características del XML, caracteres como "<" o "&" pueden ser interpretados como parte del propio etiquetado del documento XHTML y, para evitar que esto ocurra, se encapsulan los bloques de *script* o estilos; de tal modo que, cuando se abre un bloque de *script* o de estilos CSS, para evitar casos de incompatibilidad, debemos colocar su contenido dentro de un bloque CDATA.

## Recuerda

**Todos los documentos XHTML válidos deben llevar un elemento llamado DOCTYPE, el cual no forma parte del documento en sí, sino que define el tipo de DTD (Document Type Definition) que debemos emplear en nuestros documentos de forma obligatoria.**

## 2.4 Ventajas de XHTML sobre HTML

**XHTML** (*eXtensible HyperText Markup Language*) es la versión modernizada del tradicional HTML. XHTML es un lenguaje de marcas semántico, lo que quiere decir que este lenguaje no define el aspecto de las cosas, sino lo que significan (Figura 2.7).



**Figura 2.7**

El lenguaje XHTML ofrece mayores ventajas que los demás.

Por ejemplo, si tenemos el título de nuestra página, en lugar de decir "Lo quiero grande en letras rojas", le indicamos al navegador que "éste es el título principal de la página. Haz algo para que destaque". Y ese "algo" lo dejamos a decisión del navegador. XHTML también es un lenguaje basado en etiquetas (*tags*). En realidad, las ventajas de XHTML son aún pocas sobre HTML, pues se trata todavía de la primera versión de este lenguaje, lo que podría considerarse como una versión de transición. Veamos a continuación algunas de estas ventajas:

- Estandarización de la gramática.
- Los navegadores, buscadores, lectores de pantalla, etc., no tienen que "deducir" lo que el autor del documento ha querido decir, ya que el mismo autor lo aclara mediante los datos adicionales a los que nos hemos referido al definir XHTML.
- Los programas que pueden leer XML también pueden leer XHTML y pueden utilizar páginas web como fuente de información.
- Se pueden incorporar elementos de distintos espacios de nombres XML (como MathML y Scalable Vector Graphics).
- Con XML se pueden utilizar fácilmente herramientas creadas para procesamiento de documentos XML genéricos (editores, XSLT, etc.).
- XHTML es XML y, por lo tanto, se puede integrar con otras tecnologías XML.
- Los documentos XHTML se pueden visualizar en navegadores HTML.
- XHTML puede utilizar DOM de HTML o el de XML.
- XHTML puede extenderse o reducirse hasta acomodarse a las necesidades de la aplicación.

Si tus documentos son XHTML puro (sin incluir otros lenguajes de etiquetado), entonces las diferencias no serán muy significativas actualmente. Sin embargo, a medida que proliferan las herramientas XML, como XSLT para la transformación de documentos, las ventajas de usar XHTML serán más evidentes. XForms, por ejemplo, permitirá editar documentos XHTML u otros tipos de documentos XML de forma sencilla.

Las aplicaciones de Web Semántica serán capaces de sacar provecho de los documentos XHTML. Si tus documentos son algo más que XHTML, por ejemplo si incluyen MathML, SMIL, SVG u otros lenguajes, las ventajas serán inmediatas; ese tipo de combinaciones no son posibles con HTML.

## 2.5 Versiones de HTML y de XHTML

En este apartado estudiaremos las diferentes versiones de HTML y de XHTML desde el punto de vista sintáctico y estructural.

## 2.5.1 HTML

Los estándares HTML son HTML 2.0, HTML 3.2, HTML 4.0 y HTML 4.01 (Figura 2.8).

Todos son comprendidos por los navegadores actuales, pero el más extendido y el que debe utilizarse es **HTML 4.01**. Aún puede usarse sin problemas el HTML 3.2, pero le falta alguna etiqueta, ya que se ha visto afectado por la introducción de la filosofía CSS (*Cascading Style Sheets*), hojas de estilos en cascada.



**Figura 2.8**

HTML 4.01 es el lenguaje más utilizado actualmente.

Además, cada versión tiene sus variantes, por lo que es preciso elegir una. Por ejemplo, el HTML 4.01 tiene tres:

- **HTML 4.01 Strict.** Es la versión más limpia y simple de HTML. Con este doctype sólo puedes usar las partes del HTML que se refieren a la estructura y, en general, no permite utilizar las partes que se relacionan con el estilo.

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

- **HTML 4.01 Transitional.** Se trata de una combinación de todos los HTML en la que se aceptan las etiquetas obsoletas (Figura 2.9). Se permite el uso de etiquetas de estilo, que realmente no tienen cabida en HTML, pero que se introdujeron antes de que el CSS existiera.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```



**Figura 2.9**

El W3C proporciona validación para todas y cada una de las diferentes versiones de HTML.

- **HTML 4.01 Frameset.** Esta versión de HTML permite utilizar un conjunto de marcos (frames) en lugar del body, por lo que pueden combinarse varias páginas en una sola.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
```

**HTML 5** es la quinta revisión importante del lenguaje; con esta versión, HTML no cambia en lo esencial, pues la mayoría de los elementos se conservan. El estándar se enriquece con las nuevas herramientas de gestión de contenidos (blogs, agregadores, páginas personales y de inicio, etc.), y facilita la inclusión de elementos multimedia. Uno de los principales cambios de HTML5 es la inclusión en el estándar del DOM (Modelo de Objeto de Documento), un componente fundamental de HTML, si bien siempre había sido tratado de forma separada. En HTML5 (Figura 2.10) no existe diferencia entre mayúsculas y minúsculas; es decir, podemos utilizar etiquetas en mayúsculas, en minúsculas e incluso mezclarlas ambas y la página seguirá siendo válida. Tampoco se exigen las etiquetas de cierre de los elementos, y las comillas en los atributos son opcionales. En HTML5 la declaración DOCTYPE es bastante más sencilla:

```
<!DOCTYPE html>
```

**Figura 2.10**

El lenguaje HTML5 presenta nuevas herramientas de gestión de contenidos con respecto a las versiones anteriores, si bien conserva características esenciales.



Podemos dividir el contenido de una página web en varias categorías que, en su mayoría, no son novedades de HTML5, pero que nos ayudan a entender cómo se organiza internamente la información. Estas categorías son útiles a la hora de agrupar elementos dentro de una página. Por ejemplo, la categoría de información de flujo describe todos los elementos que podemos utilizar en una página, pero podemos subdividir el contenido de flujo en categorías más pequeñas, como titulares o textos. Veamos cuáles son:

En la **categoría de los metadatos**, que es la información que establece el modo de restitución, o el comportamiento, del resto del contenido de la página web, destacaríamos la etiqueta `<meta>`, que suele contener una descripción de la información contenida dentro de la página, o palabras clave, y que los motores de búsqueda utilizan para clasificar las páginas. Otros elementos como `<style>` y `<script>` se consideran también metadatos puesto que intervienen en la presentación y actividad del contenido principal. Los metadatos se ubican en la sección `<head>` del documento:

- En la **categoría del flujo**, que son todas aquellas etiquetas utilizadas para definir contenidos, podríamos destacar las etiquetas `<p>`, `<h1>`, `<ol>`, `<table>`, etc. El contenido de flujo normalmente es texto o un archivo insertado, como una imagen o un vídeo. En HTML5 aparecen algunos elementos nuevos dentro de esta categoría, como `<article>`, `<aside>`, `<audio>`, `<canvas>`, `<hgroup>`, entre otros.
- En la **categoría de secciones**, que es una nueva categoría de HTML5, actualmente se incluyen cuatro elementos: `<article>`, `<aside>`, `<nav>` y `<section>`. El W3C define el contenido de secciones como aquellos elementos que “definen el alcance de cabeceras y pies de página”. El contenido de las secciones es un subconjunto del contenido de flujo. Veremos más adelante cómo se distribuye una página en secciones.
- En la **categoría de títulos** se contemplan todos los elementos de encabezamiento que vienen utilizándose en HTML 4.0: `<h1>`, `<h2>` y sucesivos. HTML5 incorpora, además, el elemento `<hgroup>`, pensado para agrupar dos o más títulos. El contenido de titulares forma parte del contenido de flujo (Figura 2.11).
- En la **categoría de textos** se engloba todo el texto del documento, incluidos los elementos delimitadores de texto dentro de los párrafos. El contenido de texto es un subconjunto del contenido de flujo. Podríamos destacar el uso de las etiquetas `<p>` que contienen en su interior las etiquetas `<strong>` y `<em>`.
- En la **categoría de contenidos embebidos**, que son importados de otros recursos, como imágenes, videos o archivos, podríamos destacar la aparición de etiquetas como `<audio>`.
- En la **categoría de contenidos interactivos**, que son aquellos elementos más básicos, como la etiqueta `<a>` utilizada en los vínculos, también podemos destacar los elementos `<textarea>` o `<button>` que se utilizan en los formularios.

```
1  <!DOCTYPE html>
2  <head>
3  <title>HTML5 Ejemplo 1</title>
4  </head>
5  <body>
6
7  <canvas id="miCanvas" height="200" width="400"
8  style="border: 1px solid #c3c3c3">
9  </canvas>
10
11 </body>
12 </html>
13
14 <!-- Otty Guitarras --->
```

**Figura 2.11**  
Ejemplo documento HTML5.

## Para saber más

**El organismo W3C (World Wide Web Consortium) elabora las normas que deben seguir los diseñadores de páginas web para crear las páginas HTML. Las normas oficiales están escritas en inglés y pueden consultarse de forma gratuita en la dirección de la Especificación oficial de HTML 5 (<http://www.w3.org/TR/html5/>).**

Sin embargo, uno de los elementos más innovadores es el nuevo elemento <header>, que representa un grupo de textos de presentación o ayudas a la navegación que debe contener la cabecera de sección (un elemento h1-h6 o un elemento hgroup) o que pueda utilizarse como contenedor de una tabla de secciones a modo de índice de contenidos, un formulario de búsqueda de interés; y puede ser utilizado en una misma página varias veces.

### 2.5.2 XHTML

La versión más actual de XHTML es la 1.1. La versión 1.0 de XHTML es la aplicación del lenguaje de marcas XML al HTML, por lo que básicamente tiene las mismas funcionalidades, aunque cumple las especificaciones más estrictas de XML. XHTML 1.0 es la versión más parecida a HTML 4 e incluye también tres variantes:

- **XHTML Transitional**, que permite el uso de las capacidades de presentación de HTML y está pensado para trabajar con navegadores con soporte de CSS limitado. Cuando tu página web se ajuste a las normas básicas XHTML, pero todavía utilices algunas etiquetas HTML para la presentación.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

- **XHTML Strict**, que no permite el uso de los elementos y atributos relacionados con aspectos de presentación y está concebido para ser usado con hojas de estilo CSS. Cuando tu página web se ajuste a las normas de XHTML y los usos de CSS para la separación total entre contenido y presentación. Las páginas son más semánticas y ordenadas.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- **XHTML Frameset**, que permite el uso de marcos (*frames*) para dividir la ventana del navegador. Es idéntico al doctype transitional, excepto en el uso de la etiqueta <frameset> en lugar de <body>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

La versión XHTML 1.1 es la nueva versión, que surge de XHTML 1.0 Strict. Esta versión no permite el uso de elementos y atributos relacionados con el estilo (pero sí, obviamente, hojas de estilo, como CSS); e introduce el concepto de **modularización**. XHTML Basic 1.0 y 1.1 son versiones simplificadas de XHTML 1.1 que están pensadas para dispositivos de capacidades más limitadas de procesado y presentación, como televisores, teléfonos, móviles, PDA, etc.

XHTML 2.0 se halla actualmente, y desde hace años, en proceso de estandarización (*Working Draft*) por parte del W3C. Retoma la línea iniciada con XHTML 1.1 de modularizar XHTML, y añade nuevos módulos como *XML Events* y *XForms*. Estos módulos nacen con el objetivo de minimizar el uso de *scripts* dentro de documentos XHTML mediante la inclusión, como parte del propio lenguaje, de los medios necesarios para implementar las funcionalidades más importantes que requerían el uso de dichos *scripts* (eventos y formularios). Es probable que esta especificación desaparezca en favor de HTML5.

El lenguaje XHTML avanza en el proyecto del World Wide Web Consortium, para lograr una web semántica, donde la información y la presentación estén separadas.

XHTML presenta las ventajas del XML, que son muchas. Si un programa o navegador que está analizando el XHTML detecta un XML mal formado (un documento erróneo o no bien formado), está obligado a detenerse e indicar que es incorrecto. Eso simplifica enormemente el código de un navegador web y permite que en Internet sólo pueda existir el código XHTML bien formado y compatible con todos los analizadores de XML; si bien eso no asegura que las etiquetas estén bien utilizadas (Figura 2.12).



**Figura 2.12**

Ejemplo de pantalla de validación de código HTML y XHTML.

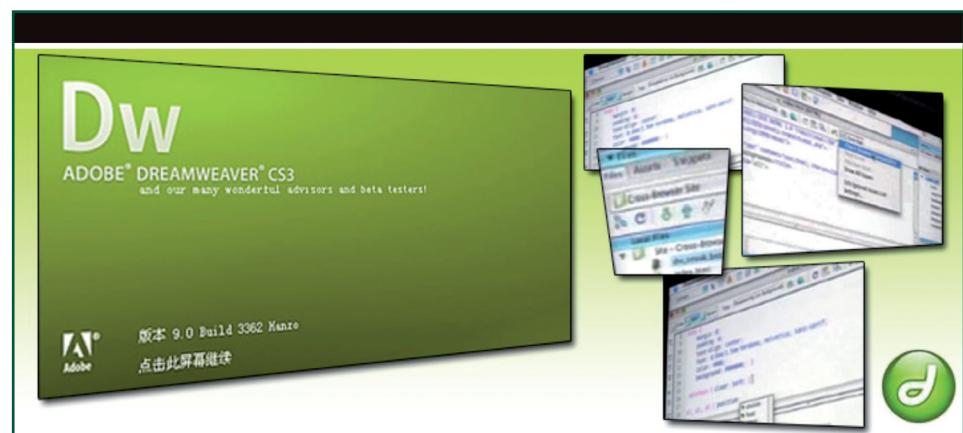
## 2.6 Herramientas de diseño web

Para elegir las herramientas que hay que utilizar en el diseño web, antes debemos identificar las fases del proceso que constituyen el ciclo de vida de un desarrollo web. El diseño consiste en crear esbozos de la web final mediante una herramienta gráfica, como puede ser Photoshop, GIMP o Inkscape. A continuación, se realiza la maquetación, generalmente con HTML/CSS, que consiste en convertir los esbozos creados en la fase anterior en plantillas HTML con su respectiva hoja de estilos y las imágenes utilizadas.

Existe en el mercado un gran número de editores de páginas web que abarcan desde pequeñas herramientas (tales como *Easy Web Editor*: <http://www.easywebeditor.com>), pasando por convertidores de texto a páginas web (los principales procesadores de texto, como Word, también poseen esta funcionalidad), hasta herramientas mucho más complejas que permiten un contenido dinámico y aplicaciones multimedia (tales como los distintos editores de Adobe para crear páginas en Flash que permiten animaciones, audio, sitios web con gráficos vectoriales y gráficos de mapas de bits, tecnologías 3D, interactividad, etc.).

Los primeros editores o herramientas que salieron al mercado y que facilitaban la conversión de un texto escrito en un documento con formato HTML para la Web se limitaban a traducir el texto al lenguaje de programación HTML, por lo que solían mantener una estructura lineal. Sin embargo, hoy en día, existen potentes editores y generadores de páginas web que facilitan la construcción de grandes y complejos sitios web sin tener ningún conocimiento de lenguaje HTML, y que elaboran de forma automática HTML, además de JavaScript, CSS (Hojas de Estilo) y mapas de imágenes, mapas del sitio web, etc., y que incluyen además, entre otras muchas prestaciones, facilitar la publicación del sitio web dentro de la World Wide Web de forma automática.

Los editores WYSIWYG (*What You See Is What You Get*, un juego de palabras que podemos traducir como “lo que ves es lo que obtienes”) son programas en los que ya no hay que trabajar con lenguaje HTML, sino directamente en lenguaje natural, y que nos muestran lo que realmente se va a ver en pantalla y no el código subyacente, por lo que podríamos denominarlos *editores visuales*. Dreamweaver de Adobe sería un ejemplo de ello (Figura 2.13). Además del navegador necesario para poder ver los resultados de nuestro trabajo, necesitaremos evidentemente otra herramienta capaz de crear la página en sí. Un archivo HTML (una página) no es más que un texto. Por ello, para programar en HTML necesitamos un editor de textos.



**Figura 2.13**

Adobe Dreamweaver,  
un editor WYSIWYG.

Es recomendable utilizar el bloc de notas que viene con Windows u otro editor de textos sencillo. Sin embargo, debemos tener cuidado con algunos editores más complejos, como WordPad o Microsoft Word, pues colocan su propio código especial al guardar las páginas y HTML es únicamente texto plano, por lo que podríamos tener problemas.

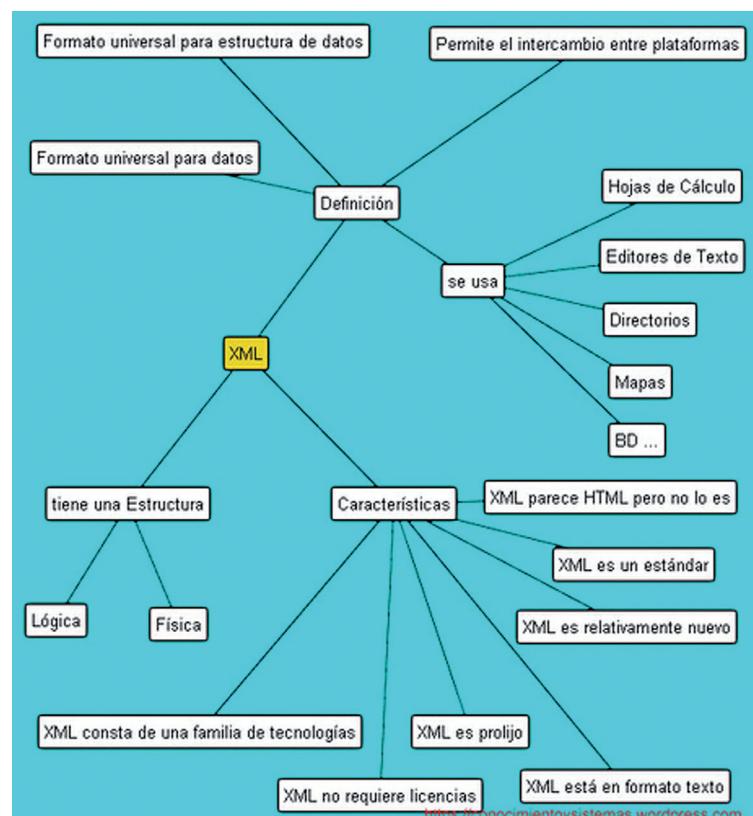
Existe otro tipo de editores específicos para la creación de páginas web que ofrece muchas facilidades y nos permite aumentar nuestra productividad. No obstante, es aconsejable, en un principio, utilizar una herramienta lo más sencilla posible para poder prestar la máxima atención a nuestro código y familiarizarnos lo antes posible con él. Siempre tendremos tiempo, más adelante, de utilizar editores más versátiles. Algunos recursos disponibles en Internet:

- **Directarios de plantillas gratuitos:** OSWD, OpenWebDesign, Open Designs, CSS Tinderbox.
- **Aplicaciones para diseño web:** Nvu, Cssed, Quanta Plus, Bluefish, GIMPShop.
- **Plantillas completas:** TemplateNavigator, TemplateBox, Template Monster, Art for the Web, Free CSS Templates, Effex-Media, DotcomWebDesign, Themebot Design Templates, Groovy Lizard, Interspire, Joyful Heart Designs, OpenSourceTemplates, Ricky's Web Templates, Free CSS Templates, FreekTemplates.com, Template Perfection, TemplateWorkz, Zymic, OpenSourceTemplates, Freelayouts, FreeCSSTemplates.org, Free Layouts, Free Templates Online, Free Website Templates, Six-Shooter Media, Free Flash Template, Layouts4Free.com, Solucija, MasterTemplates, TemplateWorld, Painted Pixels, CSSFill, ThemesBase, Arcsin Design Templates, Snakeye Web Templates, Steve's Templates, GordonMac.com, Web Design Helper, Webmaster Resources.
- **Layouts CSS:** Layout Gala, AndreasVilkund.com, CSS Template Directory, BenMedowcroft.com, Mollio, Real World Style, Little Boxes, The Layout Reservoir, Protagonist Web Resources, Glish, FU2K, Position is Everything, ThreeColumnLayouts, ssi-developer, MIS Web Design, Ruthsarian Layouts, A List Apart: CSS, Code-Sucks.com, Particle Tree, Project Seven, MyCelly, Ideas, CSS Library, IntensiveStation, IronMyers, The CSS Playground, WordPress Theme Viewer, Fluid 2-Column Template, Avinash 2.0.
- **Generadores y herramientas CSS:** CSS Creator, Inknoise, Strange Banana, Nidahas, Maketemplate, IBDjohn, CSS Creator, PsychHo, Firdamatic, WordPress Theme Generator.
- **Plantillas y temas para blogs:** SmashingMagazine, Kaushal Sheth, ErraticWisdom, Scribez, pinkdesign, Textgarden.org, Blogger Templates, Blogfashions, FinalSense, WP Themes, Drupal Theme Garden.

## 2.7 Transmisión de información mediante lenguajes de marcas

Los problemas relacionados con el intercambio de información entre aplicaciones y máquinas informáticas son tan antiguos como la propia informática. El problema surge cuando hemos realizado un determinado trabajo con un software en un ordenador concreto y después queremos pasar dicho trabajo a otro software en ese mismo u otro ordenador. En la informática actual, el problema se agrava ante nuestra necesidad de disponibilidad global del trabajo; es decir, ante la posibilidad de tener que ver dicho trabajo en dispositivos de todo tipo, como miniordenadores, PDA, tabletas y teléfonos móviles.

En realidad, el **lenguaje de marcas** es una tecnología muy sencilla que dispone a su alrededor de otras tecnologías que la complementan y la hacen mucho más grande, pues le otorgan unas posibilidades enormes y muy básicas para la sociedad de la información (Figura 2.14). Los lenguajes de marcas representan una manera distinta de hacer las cosas, un modo más avanzado, cuya principal novedad consiste en compartir los datos con los que se trabaja, a todos los niveles, en todas las aplicaciones y soportes. El lenguaje de marcas juega un papel importante en nuestro mundo actual, ya que permite compartir la información de una manera segura, fiable y fácil.

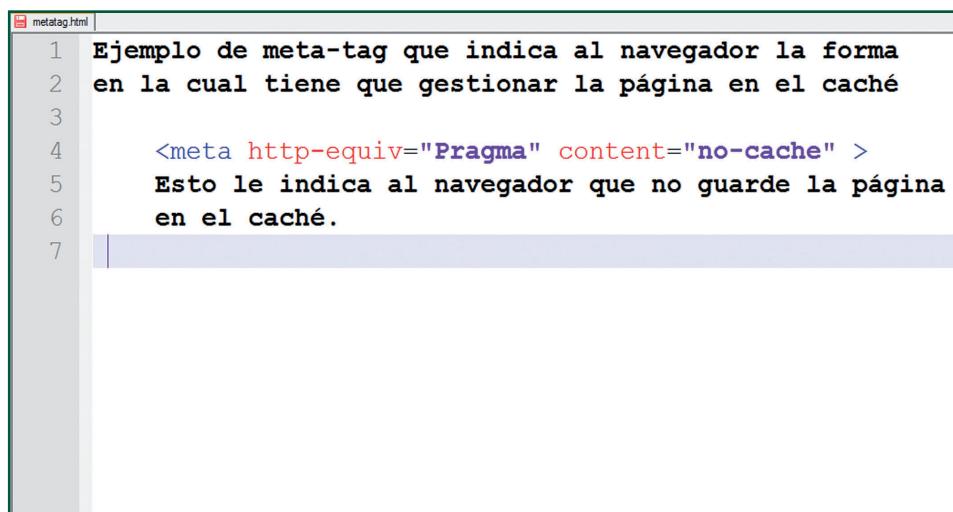


**Figura 2.14**

El lenguaje de marcas XML facilita el intercambio de información entre diferentes plataformas.

## 2.7.1 Los *Meta tags*

Dentro de los lenguajes de marcas podemos destacar los ***meta tags***, que son unas etiquetas pertenecientes al HTML que deben escribirse dentro del *tag* general <head> y que podemos definirlos como líneas de código que indican a los buscadores que lo indexan con qué términos debe buscarse la página para facilitar el acceso y el flujo de la información. Según la utilización, caracterización y objetividad de dichos *meta*, puede obtenerse una excelente posición en el listado resultante de una búsqueda (Figura 2.15).



```

metatag.html
1 Ejemplo de meta-tag que indica al navegador la forma
2 en la cual tiene que gestionar la página en el caché
3
4 <meta http-equiv="Pragma" content="no-cache" >
5 Esto le indica al navegador que no guarde la página
6 en el caché.
7

```

### Para saber más

Los ***meta tags*** son unas líneas de código HTML que se incorporan a la página principal de una web e indican a los buscadores que indexan páginas web con qué términos debe buscarse dicha página. Por lo tanto, están pensados para los buscadores y son muy importantes para el posicionamiento de una página web.

**Figura 2.15**

Los *meta tags* transmiten información a los buscadores web.

Según varios estudios, cuando una persona realiza una búsqueda en Internet no suele ir más allá del resultado 20, pues entre estos primeros puestos suele encontrar aquello que busca. Para permanecer en esos puestos, es conveniente tener unos *meta* optimizados.

Si una página web no tiene *meta tags*, de poco servirá que se haya dado de alta en buscadores. Si se incluye *meta tags* con palabras clave, es preciso agregar una descripción clara y específica para dichas palabras clave: el *meta-tag-description*, la descripción de las etiquetas que se visualizará en la pantalla de los resultados del buscador y, seguramente, será decisivo para que el internauta entre en el sitio. Esta información podría ser utilizada por los robots de búsqueda para incluirla en las bases de datos de sus buscadores y mostrarla en el resumen de búsquedas, o tenerla en cuenta durante éstas, y no resultará visible para un visitante normal. Estas etiquetas se utilizan también para especificar cierta información técnica, de la que pueden valerse los navegadores para mostrar la página, como el grupo de caracteres utilizado, el tiempo de expiración del contenido, la posibilidad de dejar la página en *caché* o calificar el contenido del sitio (“para adultos”, “violento”, etc.).

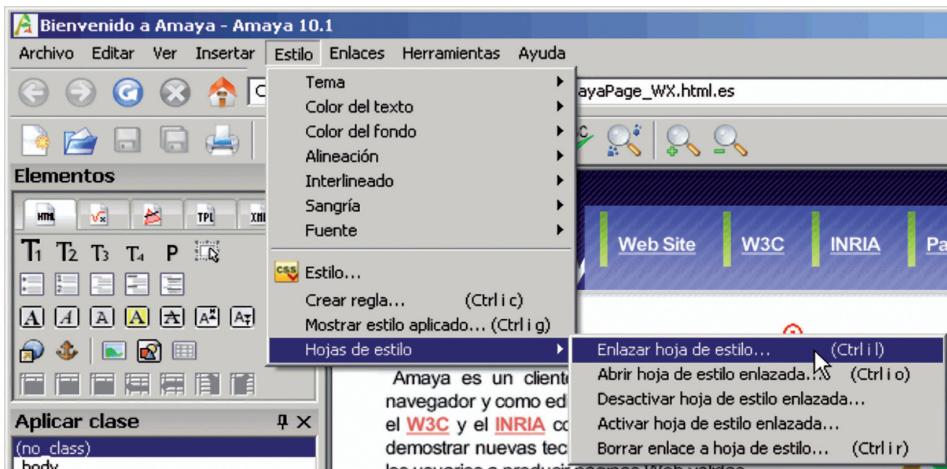
## 2.7.2 Web Service

Otra aplicación de los lenguajes de marcas con respecto a la transmisión de información es la **creación de Web Services**. Una *Web Service* es un componente de software que se comunica con otras aplicaciones, codificando los mensajes en XML y enviándolos a través de protocolos estándares de Internet, tales como el HTTP (*Hypertext Transfer Protocol*). Una Web Service es similar a un sitio web que no cuenta con una interfaz de usuario y que ofrece un servicio a las aplicaciones en lugar de a las personas. En vez de obtener solicitudes desde el navegador y retornar páginas web como respuesta, la Web Service recibe solicitudes a través de un mensaje formateado en XML desde una aplicación, realiza una tarea y devuelve un mensaje de respuesta también formateado en XML. Microsoft y otras empresas líderes están promocionando SOAP como estándar de los mensajes para las Web Services. Un mensaje SOAP presenta un aspecto similar al de una carta: se trata de un sobre que contiene una cabecera con la dirección del receptor del mensaje, un conjunto de opciones de entrega (tal como la información de encriptación) y un cuerpo o *body* con la información o datos del mensaje. Un ejemplo de Web Service que transmite información mediante el protocolo SOAP sobre http es:

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

## 2.8 Hojas de estilo

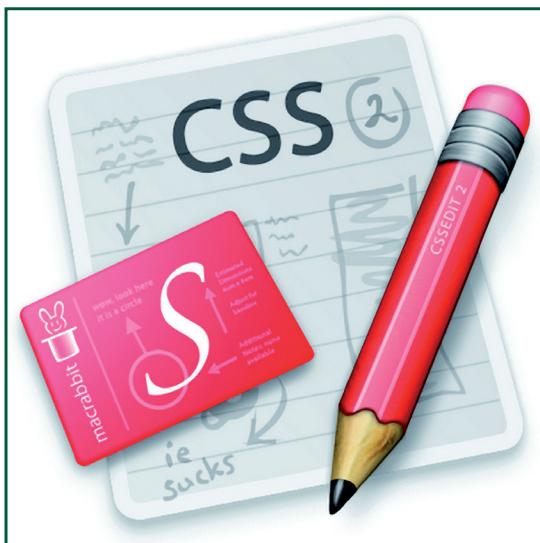
Las **hojas de estilo** (*style sheets*) son conjuntos de instrucciones, a veces en forma de archivo anexo, que se asocian a los archivos de texto y se ocupan de los aspectos de formato y de presentación de los contenidos, como tipo de fuente, tamaño de letras, justificación del texto, colores y fondos, etc. Las hojas de estilo permiten liberar la composición del texto de los aspectos visuales y favorecen que se estructure y anote mediante códigos que permiten un tratamiento más eficaz de los contenidos. El uso adecuado de las hojas de estilo es uno de los aspectos clave de la edición digital (Figura 2.16).

**Figura 2.16**

Las hojas de estilo son fundamentales para la edición de contenidos digitales.

## 2.8.1 Hojas de estilo en cascada

**CSS** (*Cascading Style Sheets*, hojas de estilo en cascada) es un lenguaje sencillo para la aplicación de estilos a un elemento XML o HTML. CSS es un lenguaje que se utiliza para definir la presentación de un documento estructurado escrito en HTML o XML (y, por extensión, en XHTML y otros lenguajes de marcas) (Figura 2.17).

**Figura 2.17**

CSS, hoja de estilo en cascada.

CSS tiene tres versiones: CSS1, CSS2 y CSS3. La CSS1 fue introducida para el lenguaje HTML, pero puede utilizarse, y con menos restricciones, en el lenguaje XML. La CSS2 sirve tanto para HTML como para XML, pero es en el XML donde alcanza la potencia que tiene. Debemos señalar que muchas características de los estándares no funcionan o funcionan en un navegador y no en otro, si bien ése no es un problema del estándar, sino de que no todos los navegadores están preparados.

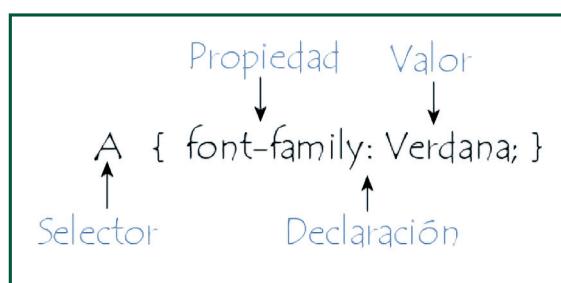
La versión CSS3 está dividida en varios documentos separados, llamados módulos. Cada módulo añade nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores con el fin de mantener la compatibilidad. Los trabajos en CSS3, comenzaron a la vez que se publicó la recomendación oficial de CSS2, y los primeros borradores de CSS3 fueron liberados en junio de 1999.

Debido a la modularización del CSS3, los diferentes módulos, hasta que se convierten en recomendaciones oficiales de la W3C, pueden encontrarse en diferentes fases de su desarrollo. Un mismo documento XML o HTML puede tener varias hojas de estilo CSS y estar formateado de diferentes formas según nuestros propósitos o del posible uso que se le vaya a dar al documento. Las CSS se expresan mediante una serie de reglas que guardaremos en un fichero de texto. Cada regla contiene el nombre del elemento al que se aplica y el estilo definido.

Existen algunas diferencias en la aplicación de CSS para HTML y la aplicación de CSS para XML. En la aplicación para el lenguaje XML, los elementos a los que se les puede asignar una regla de estilo no están limitados, los navegadores HTML no entienden el procesamiento de instrucciones, simplemente la hoja de estilos se incluye como una etiqueta *style* y los navegadores HTML tienen un formateo restringido, mientras que XML no.

Para poder visualizar un documento XML, es necesario especificar qué formato se debe utilizar. Para este propósito se utiliza la instrucción *xmlstylesheet*, que se indicará en la segunda línea del código del documento XML añadiendo una etiqueta en la que se indicará la ubicación del documento con el formato que debe aplicársele.

CSS tiene una sintaxis sencilla que utiliza unas cuantas palabras clave tomadas del inglés para especificar los nombres de sus selectores, propiedades y atributos. Consiste en una serie de reglas en las que cada una tiene uno o más selectores, y un bloque de estilos con los estilos, a aplicar a los elementos del documento que cumplan con el selector que le precede. Cada bloque de estilos se define entre llaves y está formado por una o varias declaraciones de estilo con el formato “**propiedad: valor;**” (Figura 2.18).

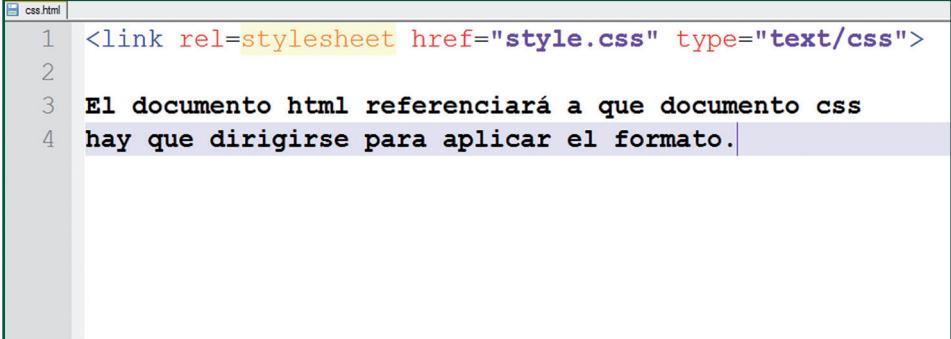


**Figura 2.18**

Ejemplo de la estructura que ha de tener una etiqueta en CSS.

Para dar formato a un documento HTML o XML, puede emplearse CSS de tres formas distintas:

- Mediante CSS introducido por el autor del documento HTML o XML:
  - Un **estilo en línea** (*inline*) es un método para insertar el lenguaje de estilo del documento directamente dentro de una etiqueta HTML o XML, indicando seguido del nombre de la etiqueta el formato que debe aplicarse al documento.
  - Una **hoja de estilo interna**, aplicación de una hoja de estilo que está incrustada dentro de un documento HTML o XML, dentro del elemento `<head>`, que se marca con la etiqueta `<style>` y, dentro de ella, se especifica el formato que debe aplicarse al documento.
  - Una **hoja de estilo externa**, aquella que está almacenada en un archivo diferente al archivo HTML o XML, en el que se guarda la información sobre el formato a aplicar al documento (Figura 2.19).



```
css.html
1 <link rel="stylesheet" href="style.css" type="text/css">
2
3 El documento html referenciará a que documento css
4 hay que dirigirse para aplicar el formato.
```

**Figura 2.19**  
Ejemplo de hoja de estilo externa.

- Estilos CSS introducidos por el usuario que puede ver el documento, mediante un archivo CSS especificado por las configuraciones del navegador, y que sobrescribe los estilos definidos por el autor en una o varias páginas web.
- Los estilos marcados por defecto por los *user agent* (agentes de usuario), que se utilizan para fijar un formato específico en diferentes elementos de un documento HTML; como, por ejemplo, los enlaces, que, por defecto, aparecerán subrayados y de color azul.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-7"?>
<?xml-stylesheet href="ejemplo0.css" type="text/css"?>
```

## Para saber más

También puedes encontrar más información ampliada en la página de W3Schools (<http://www.w3schools.com/css/>).

En este ejemplo el fichero de estilos se incluye en el documento XML. Debemos señalar que la instrucción posee dos atributos: **Href**, que indica la dirección absoluta o relativa en la que se encuentra el fichero de estilos; y **type**, que representa el MIME type, es decir, transmite la información al navegador sobre cómo tratar la extensión del archivo que indica el Href. En este caso, será text/css para un fichero CSS.

La sintaxis básica de una hoja de estilos CSS consta de dos partes: la primera funciona como selector para indicar qué elementos se van a ver afectados por el estilo; la segunda parte es el estilo propiamente dicho, que puede escribirse en una línea o varias para facilitar la lectura:

```
selector { propiedad: valor; propiedad2: valor; }
```

A continuación, veamos un ejemplo en el que se define el color y el tamaño de fuente para los párrafos, es decir, para las etiquetas <p>:

```
p {  
color: #f00;  
font-size: 10px;  
} .
```

Otro ejemplo de aplicación de reglas de estilo aplicados a una línea en HTML y su correspondiente en código CSS:

- **Ejemplo de código HTML:** <h2><font color="red">Texto de prueba</font></h2>
- **Ejemplo de código CSS:** H2 {color: red;}

Uno de los componentes básicos de las CSS son los **selectores**, que sirven para determinar qué reglas se han de aplicar a cada elemento y cuáles no. Para facilitar el trabajo de seleccionar las etiquetas, CSS define diferentes tipos de selectores. Veamos a continuación cuáles son:

- **Selector universal.** Se presenta con '\*' y selecciona todos los elementos de un documento.
- **Selector de etiquetas.** Selecciona un elemento por el nombre de su etiqueta.
- **Selector de hijos.** Se utiliza el símbolo ">" para definir antes del símbolo al padre; y después del símbolo, al hijo.

- **Selector de descendientes.** Se enumeran los descendientes separados con " ".
- **Selector de hermanos adyacentes.** Selecciona elementos que comparten el mismo parente y están situados inmediatamente después de él.
- **Selector por ID.** Selecciona los elementos concretos cuyo atributo *id* corresponde con el especificado por el selector.
- **Selector por class.** Selecciona los elementos cuyo atributo *class* corresponde con el especificado por el selector.
- **Selector de atributos.** Selecciona las etiquetas en función de sus atributos.
- **Selector de pseudoclases.** Selecciona los elementos a partir de condiciones que no son su nombre. Separan el nombre del elemento al que se aplicarán con el símbolo ":" (Figura 2.20).

## Css3 Selector demo

### Alternate row styling

**Even**

- A sample text here 1
  - A sample text here 2
- A sample text here 3
  - A sample text here 4

**Odd**

- A sample text here 1
- A sample text here 2
  - A sample text here 3
- A sample text here 4

**Every third row**

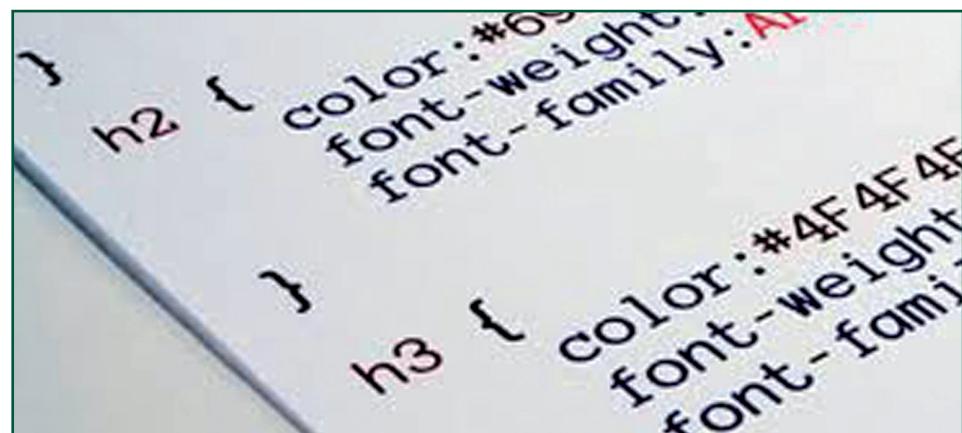
- A sample text here 1
- A sample text here 2
  - A sample text here 3
- A sample text here 4
- A sample text here 5
  - A sample text here 6

**Figura 2.20**

Los selectores son uno de los componentes principales de las CSS.

Otro de los componentes de la reglas CSS son las llamadas **declaraciones**. Se trata de la parte que se encarga de dar formato a las etiquetas seleccionadas. Se define entre llaves “{ }” y puede estar formada por una o más propiedades. Cada propiedad se delimita al final con el símbolo “;”. Las **propiedades** cumplen la condición de **herencia**; lo que implica que, si se aplica una condición de formato a un elemento, todos sus elementos hijos lo heredarán y adquirirán el formato especificado. A continuación, veamos cuáles son las principales propiedades y valores de las CSS (Figura 2.21):

- **Background**, o alguna de las variables *background-attachment*, *background-color*, *background-image*, *background-repeat* y *background-inherit* que son utilizadas para cambiar el color y la imagen del fondo de los elementos.
- **Border**, o cualquiera de sus variables *border-color*, *border-width*, *border-style*, *border-top*, que se utilizan para determinar los aspectos de un borde en todos los lados de un elemento.
- **Display**, que se utiliza para determinar cómo debe mostrarse un elemento, si como *inline* o como *block*; el valor por defecto de la propiedad *display* es *inline*.
- **Flota**, que sirve para determinar en qué lado de un elemento pueden flotar otros elementos. Se especifica con: *left*, *right*, *none* o *inherit*.
- **Font**, o cualquier alternativa *Font-family*, *Font-size*, *Font-variant*, *Font-weight*, que sirve para especificar el tipo de fuente para el texto.
- **Margin**, o cualquier variable *margin-top*, *margin-right*, *margin-bottom* o *margin-left*, utilizados para indicar la cantidad de espacio entre el lado del *element* y su *element* padre. O las propiedades de posicionamiento como *left*, *right*, *height*, *width*, que indican la distancia entre el elemento y el elemento padre.



**Figura 2.21**

Los selectores de CSS se colocan uno tras otro con sus correspondientes propiedades y valores.

## Resumen

El lenguaje HTML (*HyperText Markup Language* o lenguaje de marcado de hipertexto) es aquel lenguaje de marcas en el que se van definiendo los elementos del documento a través de etiquetas. Así pues, un documento HTML está constituido por texto y un conjunto de etiquetas que servirán para definir la forma con la que deberá presentarse el texto y otros elementos en la página. HTML es el lenguaje utilizado en las páginas web tanto en la estructura de la cabecera (*head*) como en el cuerpo (*body*). La etiqueta `<head>` indica el encabezado de la página, es decir, el área de la barra de título definida con la etiqueta obligatoria `<TITLE>`. Además, la cabecera incluirá información sobre la página para los motores de búsqueda, con los que estableceremos la ubicación de la página, añadiremos hojas de estilo y escribiremos todos aquellos scripts que consideremos necesarios.

Las etiquetas, también llamadas *tags*, se expresan con los símbolos `</>`, e indican propiedades. Las etiquetas deben cerrarse con el símbolo / (`<etiqueta/>`), aunque es recomendable la estructura `<etiqueta>...</etiqueta>`. Las etiquetas pueden contener atributos que califican la etiqueta. Estos atributos se definirán en la etiqueta de inicio y consisten normalmente en el nombre del atributo y el valor que toma, y van separados por un signo de igual. XHTML (*eXtensible HyperText Markup Language*) es uno de los lenguajes derivados de XML utilizado en las páginas web. XHTML está basado en HTML, pero es más estricto y afín al estándar XML.

Para escribir páginas con estos lenguajes de marcas, existe un gran número de programas que no sólo nos facilitan la tarea de edición, sino que permiten la creación de páginas de contenido dinámico, y algunos de ellos están orientados a la edición WYSIWYG (*What You See Is What You Get*), es decir, en el momento de escribir el código puedes ver cómo quedará efectivamente la página web. Los lenguajes de marcas también se aplican a la transmisión de información, normalmente con XML. También se utilizan en la transmisión de datos entre aplicaciones (Web Services), en donde un programa solicita datos a otro programa mediante un mensaje XML, y éste le devuelve la respuesta en ese formato. En esta transmisión, el mensaje puede estar estandarizado con el protocolo SOAP (*Simple Object Access Protocol*).

Las hojas de estilo (*style sheets*) son conjuntos de instrucciones, a veces en forma de archivo anexo, que se asocian a los documentos HTML para definir elementos de formato y presentación de dichos documentos. Las hojas de estilo en cascada o CSS (*Cascading Style Sheets*) es un lenguaje sencillo que se utiliza para la aplicación de estilos a un elemento XML o HTML. CSS tiene una sintaxis sencilla, que utiliza unas cuantas palabras claves tomadas del inglés para especificar los nombres de sus selectores, propiedades y atributos.

## Ejercicios de autocomprobación

**Indica si las siguientes afirmaciones son verdaderas (V) o falsas (F):**

1. El HTML (*HyperText Markup Language* o lenguaje de marcado de hipertexto) es el lenguaje con el que se escriben las páginas Web y nos permite mostrar textos, sonidos e imágenes y combinarlos a nuestro gusto.

2. Dentro de las etiquetas <body></body> se representa el contenido del propio documento. Éste no puede contener distintos tipos de elementos como por ejemplo:

```
<h1></h1>
<p></p>,
<a href="URL">Enlace</a>
<img />src="imagen.jpg" width="200" height="100"
<br>, <b></b> y <i></i>
<ul></ul>
<table></table>
<form></form>
```

3. En la categoría de contenidos embebidos son importados elementos de otros recursos, como imágenes, vídeos o archivos, en los que se puede destacar la aparición de etiquetas como <audio>.

4. Photoshop, GIMP o Inkscape son herramientas gráficas que se pueden elegir para crear esbozos de la web final.

5. Los *meta tags* son unas etiquetas pertenecientes al XML que deben escribirse dentro del *tag* general <head> y que podemos definir como líneas de código que indican a los buscadores que lo indexan con qué términos debe buscarse la página para facilitar el acceso y el flujo de la información.

6. Las hojas de estilo (*style sheets*) son conjuntos de instrucciones, a veces en forma de archivo anexo, que se asocian a los documentos XML para definir elementos de formato y presentación de dichos documentos.

7. *Background*, o alguna de las variables *background-attachment*, *background-color*, *background-image*, *background-repeat* y *background-inherit*, que son utilizadas para cambiar el color y la imagen del fondo de los elementos, es una propiedad de las CSS (*Cascading Style Sheets*).

**Completa las siguientes afirmaciones:**

8. El lenguaje \_\_\_\_\_ utiliza etiquetas para comunicarse con el navegador web. Las etiquetas, también llamadas \_\_\_\_\_, se expresan entre los símbolos \_\_\_\_\_ e indican \_\_\_\_\_. Es importante señalar que las etiquetas pueden escribirse indistintamente en mayúsculas o minúsculas. Las etiquetas pueden contener atributos que se definirán en la \_\_\_\_\_ de inicio, y consistirán normalmente en el nombre del atributo y el valor que toma separados por un signo de igual.
9. Es recomendable utilizar el bloc de notas que viene con \_\_\_\_\_ u otro editor de textos sencillo. Sin embargo, debemos tener cuidado con algunos editores más complejos, como WordPad o \_\_\_\_\_, pues colocan su propio código especial al guardar las páginas y \_\_\_\_\_ es únicamente texto \_\_\_\_\_, por lo que podríamos tener problemas.
10. Una *Web Service* es un componente de \_\_\_\_\_ que se comunica con otras aplicaciones, codificando los mensajes en \_\_\_\_\_ y enviándolos a través de protocolos estándares de Internet, tales como el \_\_\_\_\_. Una Web Service es \_\_\_\_\_ a un sitio web que no cuenta con una interfaz de usuario y que ofrece un servicio a las aplicaciones en lugar de a las \_\_\_\_\_.

Las soluciones a los ejercicios de autocomprobación se encuentran al final de esta Unidad Formativa. En caso de que no los hayas contestado correctamente, repasa la parte de la lección correspondiente.

## Soluciones a los ejercicios de autocomprobación

### Unidad 1

1. V
2. V
3. V
4. F. El lenguaje XML se utiliza para transportar y almacenar datos. El XML se centra en lo que son los datos (es un lenguaje descriptivo), mientras que el HTML se encarga de cómo tienen que visualizarse esos datos. Por tanto, XML no es un sustituto del HTML.
5. V
6. V
7. F. Las etiquetas son elementos que identifican y estructuran las diferentes partes de un documento XML. A la hora de escribir, los nombres de las etiquetas no pueden contener espacios.
8. sistema, anotaciones, etiquetas, texto, HTML.
9. XML, significado, datos, etiqueta, vacíos.
10. XML, elementos, URI (*Uniform Resource Identifier*).

### Unidad 2

1. V
2. F. Dentro de las etiquetas <body></body> se representa el contenido del propio documento. Éste puede contener distintos tipos de elementos como por ejemplo:  
`<h1></h1>  
<p></p>,  
<a href="URL">Enlace</a>  
<img />src="imagen.jpg" width="200" height="100"  
<br>, <b></b> y <i></i>  
<ul></ul>  
<table></table>  
<form></form>`
3. V
4. V
5. F. Los meta tags son unas etiquetas pertenecientes al HTML que deben escribirse dentro del tag general <head> y que podemos definir como líneas de código que indican a los buscadores que lo indexan con qué términos debe buscarse la página para facilitar el acceso y el flujo de la información.

6. F. Las hojas de estilo (*style sheets*) son conjuntos de instrucciones, a veces en forma de archivo anexo, que se asocian a los archivos de texto para definir elementos de formato y presentación de los contenidos.
7. V
8. HTML, tags, < y >, propiedades, etiqueta.
9. Windows, Microsoft Word, HTML, plano.
10. software, XML, HTTP (*Hypertext Transfer Protocol*), similar, personas.

## Índice

### MÓDULO: LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN UNIDAD FORMATIVA 1

<b>1. Características de los lenguajes de marcas.....</b>	6
1.1 Características comunes.....	6
1.2 Identificación de ámbitos de aplicación .....	8
1.3 Clasificación.....	10
1.4 XML: Estructura y sintaxis.....	13
1.4.1 Estructura .....	13
1.4.2 Atributos .....	16
1.4.3 Sintaxis .....	17
1.5 Etiquetas.....	17
1.5.1 Consejos de nomenclatura.....	18
1.6 Herramientas de edición.....	19
1.7 Elaboración de documentos XML bien formados .....	21
1.8 Utilización de espacios de nombres en XML .....	22
1.8.1 Espacio de nombres por defecto (atributo) .....	23
1.8.2 Espacios de nombres mediante prefijos.....	24
Resumen.....	25
Ejercicios de autocomprobación.....	26
 <b>2. Utilización de lenguajes de marcas en entorno web.....</b>	28
2.1 HTML: Estructura de una página web.....	28
2.1.1 Elementos a nivel de bloque y elementos a nivel de línea.....	31
2.2 Identificación de etiquetas y atributos de HTML.....	32

---

2.3 XHTML: Diferencias sintácticas y estructurales con HTML .....	36
2.4 Ventajas de XHTML sobre HTML.....	37
2.5 Versiones de HTML y de XHTML.....	38
2.5.1 HTML.....	39
2.5.2 XHTML.....	42
2.6 Herramientas de diseño web .....	43
2.7 Transmisión de información mediante lenguajes de marcas .....	46
2.7.1 Los <i>Meta tags</i> .....	47
2.7.2 Web Service .....	48
2.8 Hojas de estilo .....	48
2.8.1 Hojas de estilo en cascada.....	49
Resumen.....	55
Ejercicios de autocomprobación .....	56
Soluciones a los ejercicios de autocomprobación.....	58



# UNIDAD FORMATIVA 2

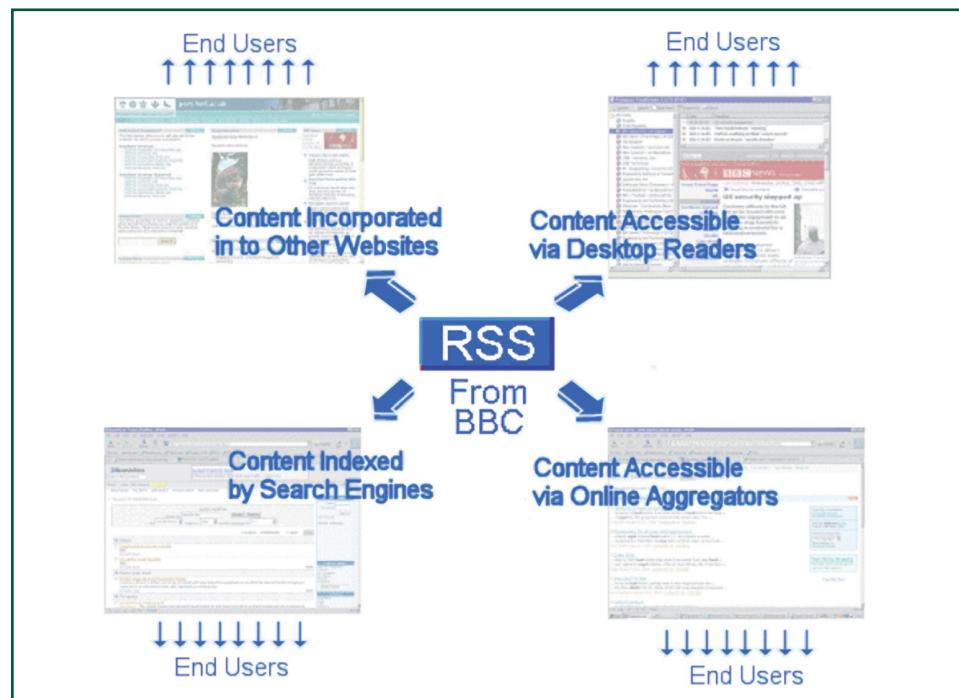
- Aplicación de los lenguajes de marcas a la sindicación de contenidos
- Definición de esquemas y vocabularios en XML

### 3. APLICACIÓN DE LOS LENGUAJES DE MARCAS A LA SINDICACIÓN DE CONTENIDOS

Cada día es más frecuente que todos nosotros hagamos uso de Internet para buscar información y poder tomar tanto decisiones profesionales como personales. Cuando buscamos información en la red podemos hacerlo de dos formas distintas: la primera es navegar por todas y cada una de las páginas en las que estemos interesados. Esto puede resultar a veces una tarea muy tediosa, incluso podemos llegar a caer en lo que se conoce como el “despiste internauta” o, lo que es lo mismo, perder mucho tiempo recabando poca información que nos resulte realmente de interés. La segunda forma, no tan conocida y que recibe un nombre un tanto complicado: **sindicación de contenidos**, es, sin embargo, mucho más sencilla (Figura 3.1).

La sindicación de contenidos, también conocida como *redifusión web*, está basada en el lenguaje de marcas XML y no es más que el reenvío de información contenida en un determinado sitio web a otro sitio de destino. Si utilizamos esta técnica, somos nosotros quienes decidimos suscribirnos a determinados sitios web y escogemos qué información queremos recibir de forma resumida y actualizada.

En esta unidad estudiaremos la forma de obtener la información mediante la sindicación de contenidos y la utilización del lenguaje de marcas.



### 3.1 Ventajas

La **sindicación de contenidos** o RSS, conocida también como suscripción a un canal o *feed* RSS, no es más que la acción de indicar a un determinado programa o página web que revise cada cierto tiempo un canal de información en busca de contenido adicional o información actualizada. Los archivos RSS se llaman comúnmente *feeds* RSS o canales RSS y contienen un resumen de lo publicado en el sitio web de origen (Figura 3.2).

Se estructura en uno o más ítems, es decir, en uno o más elementos que lo componen. Cada ítem consta de un título, un resumen de texto y un enlace a la fuente original en la web donde se encuentra el texto completo. Puede incluir, además, información adicional, como el nombre del autor o la fecha y la hora de publicación del contenido. Por tanto, cualquier fuente de información susceptible de poder ser seccionada en ítems, como, por ejemplo, los mensajes de un foro, pueden distribuirse utilizando RSS.

#### Para saber más

**Los Podcasting son una combinación de palabras iPod o Pod (reproductor de MP3) y broadcasting (retransmisión). Se parece a una suscripción de una revista hablada en la que recibimos los programas a través de Internet.**

The screenshot shows the 'FEED Validator' interface. At the top, it says 'FOR ATOM AND RSS AND KML'. Below that is a text input field containing 'http://feeds2.feedburner.com/WindowsTecnico' and a blue 'Validate' button. Underneath, the word 'Sorry' is displayed in bold. A message follows: 'This feed does not validate.' Below this, a bulleted list shows an error: '• line 2, column 647: dc:language must be an ISO-639 language code: [help] ... re Windows </description><dc:language /><generator>CommunityServer 2008. ...'. The entire screenshot is enclosed in a red border.

**Figura 3.2**

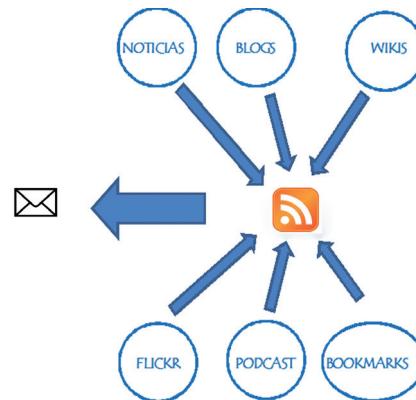
Los *feeds* o canales RSS son archivos RSS.

Si leemos el archivo RSS de un sitio web, es posible saber si se ha actualizado, y con qué noticias o textos, sin necesidad de acceder a sus páginas web. El archivo RSS contiene, además, un enlace específico para cada ítem contenido en el canal que dirige a la página web con el texto completo de la noticia.

Para leer los *feeds* o canales RSS es necesario utilizar un programa llamado *agregador*. Este tipo de programas se conoce también como lectores de *feeds* o canales, o agregadores de noticias, entre otras variaciones.

La sindicación de contenidos ofrece múltiples ventajas tanto para el autor del contenido, que consigue con el mismo esfuerzo una mayor difusión del contenido publicado en su sitio web, como para los lectores del mismo, ya que éstos consiguen tener su propio "resumen de prensa" actualizado sin tener que ser ellos los

que persigan la información a través de la red. De este modo, nosotros no trabajamos para la red, sino que es la red la que trabaja para nosotros (Figura 3.3).



**Figura 3.3**

La sindicación de contenidos facilita la búsqueda de información en Internet.

Las principales ventajas de la sindicación de contenidos para el autor del sitio web son las siguientes:

- Integración a estándares. Todos los agregadores de contenidos están basados en lenguaje de marcas XML y, por tanto, han de cumplir el estándar definido por éste, facilitando de esta forma su edición.
- Facilitar la suscripción de lectores y aumentar, de ese modo, el número de seguidores del sitio web publicado.
- Facilitar la redistribución en otros medios. El contenido del sitio web es publicado, a su vez, en otros sitios web.
- Se trata de una forma barata y eficiente para entregar el contenido a los suscriptores, clientes y socios.
- Aumenta el tráfico del sitio web. Permite un mayor alcance de la web en el mercado.

Las principales ventajas de la sindicación de contenidos para el lector o usuario del sitio web son:

- No tener que invertir demasiado tiempo buscando información actualizada de su interés, ya que es posible acceder rápidamente a todos los contenidos nuevos publicados en varios sitios sin tener que visitarlos uno por uno.
- Suscribirse y esperar la actualización, lo que permite disponer siempre de la última versión de la información actualizada sin tener que perseguirla.

- Integrar contenidos externos en medios propios que aumenten el valor informativo del sitio web.
- Facilita la forma de organizar la información que se encuentra en la web.
- Pueden obtenerse las últimas noticias de temas de interés en cuanto la información sea actualizada.
- Pueden recopilarse titulares de distintos sitios desde un mismo lugar.
- A diferencia de las notificaciones a través del correo electrónico, no existen direcciones electrónicas involucradas, de ese modo se evita publicidad, *spam*, virus, etc.
- Puede cancelarse la suscripción a una fuente web sin necesidad de aviso.

### 3.2 Ámbitos de aplicación

Los **ámbitos de aplicación** de la sindicación de contenidos abarcan muchos aspectos. Aunque se trata de una tecnología que ya lleva tiempo en el mercado, su uso todavía no se ha popularizado en exceso. Numerosos sitios web utilizan actualmente la sindicación de contenidos para agregar valor a sus servicios ya ofrecidos, como una forma de personalizarlos, diferenciarlos y atraer a más usuarios.

Veamos a continuación los principales ámbitos de actuación en la sindicación de contenidos:

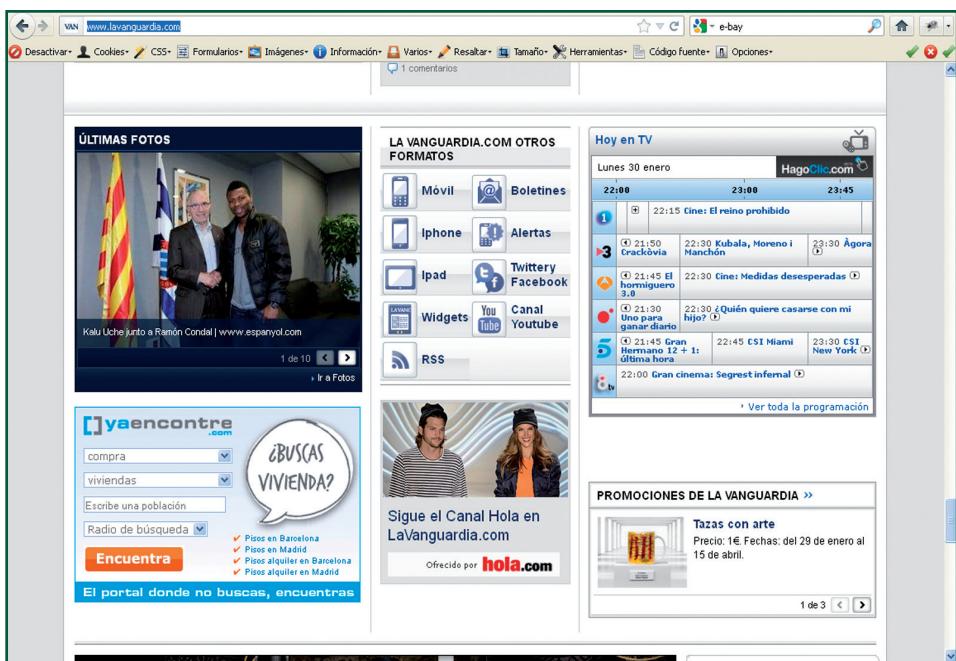
- **Aplicación en medios de comunicación.** Varios de los grandes portales de información han incorporado en sus páginas la sindicación de contenidos, de manera que algunos periódicos ofrecen a sus lectores noticias casi en tiempo real. Según algunos analistas, la sindicación de contenidos puede conducir a que los lectores, es decir, los clientes, cambien su manera de informarse: diarios e informativos de radio o televisión podrían dejar de ser un producto que el cliente compra para mantenerse informado. Se reducirían los intermediarios entre la fuente de información y el lector final. Un lector podría sindicar noticias y opiniones sobre un determinado tema a fin de que otros lectores accediesen a éstas. De esta manera, aumentaría la interconectividad de Internet, a la vez que se alteraría la función de las fuentes tradicionales de información.

Uno de los primeros pasos hacia la sindicación de contenidos lo constituyó la creación del *Publishing Requirements for Industry Standard Metadata* (PRISM). Para este proyecto, se desarrolló un módulo de tablas de contenido para revistas electró-

#### Para saber más

**Para saber si un sitio web ofrece sindicación a su contenido, basta con encontrar en él un icono de una de las versiones de RSS que incluya un enlace a un archivo de cualquier versión de RSS o de Atom.**

nicas. Actualmente, éste constituye uno de los servicios de valor agregado más relevantes de las revistas y, aunque muchas ofrecen aún la suscripción al servicio de tablas de contenido por correo electrónico, cada vez es mayor el número de editores e intermediarios que proporcionan canales RSS que contienen tablas de contenido, a menudo con enlaces a resúmenes o al texto completo (Figura 3.4).



**Figura 3.4**

Ejemplo de suscripción RSS en el periódico lavanguardia.com.

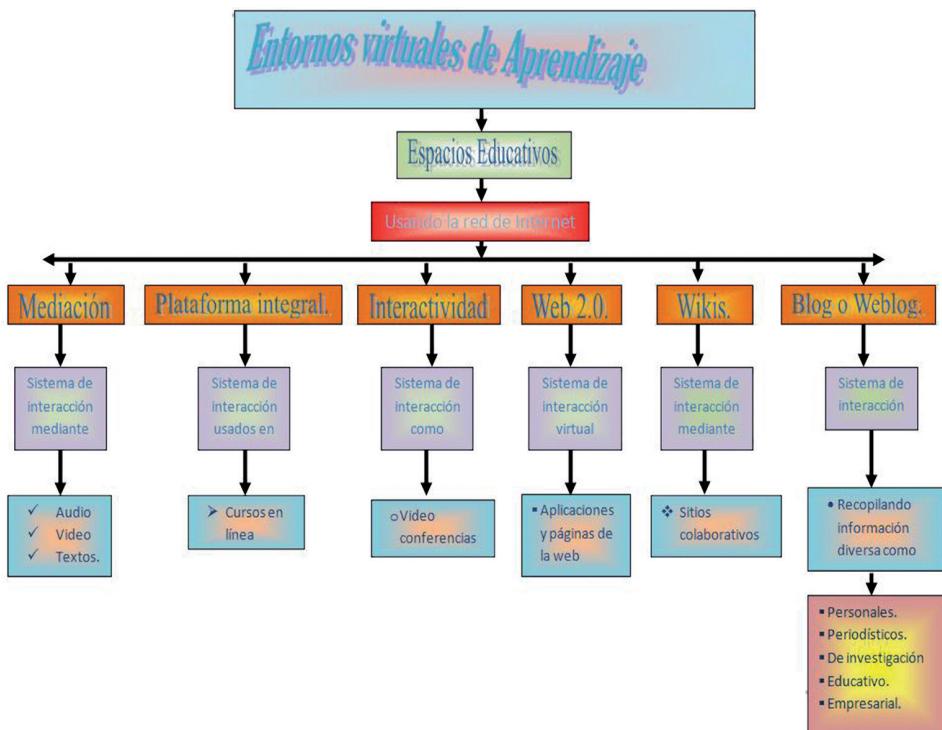
Si los editores y vendedores de revistas electrónicas sindican sus contenidos, los usuarios pueden escoger si desean recibir notificaciones para los artículos relevantes recientemente publicados por título o materia.

- **Aplicación en sitios comerciales.** La aplicación en el comercio es el núcleo de los nuevos modelos de negocio en el entorno virtual; por lo tanto, no resulta extraño que numerosas empresas y e-business ofrezcan canales RSS. Entre ellos, se encuentran motores de búsqueda, sitios de comercio electrónico y sitios de subastas, que permiten a los usuarios suscritos seguir, por ejemplo, la fluctuación de los precios. Y si bien una de las ventajas que se atribuye a los canales RSS es poder esquivar la publicidad que inunda a numerosos sitios web comerciales, se están creando páginas de anuncios intersticiales donde, cuando se accede al resumen, a menudo se encuentra un anuncio del patrocinador antes de redirigir al usuario a la página de noticias del periódico.
- **Aplicación en entornos virtuales de aprendizaje.** La sindicación de contenidos puede suponer una revolución en los entornos virtuales de aprendizaje, porque

implica recuperar, reutilizar y actualizar fácilmente los contenidos sindicados en repositorios diseñados para tal efecto. Los centros educativos pueden elaborar materiales de mayor calidad y más específicos, y adecuarlos a los diversos entornos de aprendizaje (Figura 3.5). Los canales RSS pueden combinarse y luego filtrarse por palabras clave mediante las aplicaciones llamadas RSS Mixers. Esto facilitaría, por ejemplo, el control del ritmo de aprendizaje de un grupo. Si en una clase cada estudiante tiene su propio blog y se combinan todos los canales de blogs y luego se filtran por palabras clave, el profesor podría hacerse una idea sobre si el contenido ha sido asimilado por los estudiantes, si existen lagunas en este conocimiento, y observar hasta qué punto resultó interesante el tema para el grupo.

## Recuerda

**Un archivo feed o canal RSS está construido con un subconjunto de instrucciones del lenguaje XML; por lo tanto, la información que contienen dichos archivos está estructurada con marcas o etiquetas.**



**Figura 3.5**

Gracias a la sindicación de contenidos, el sector educativo puede elaborar entornos virtuales de aprendizaje de gran calidad.

- **Aplicación a instituciones de información.** El número de aplicaciones de RSS asociadas a bibliotecas y otras instituciones de información crece exponencialmente. Las bibliotecas no sólo utilizan los canales RSS existentes como fuentes de información, sino que generan sus propios canales RSS como servicios de valor añadido para sus propios productos y servicios.
- **Aplicación para promoción de servicios y nuevas adquisiciones.** Uno de los usos principales de los canales RSS puede ser alertar a los suscriptores de un sitio web sobre las actividades actuales. Tal vez, la utilización más práctica de los canales RSS es la generación de listas de nuevas adquisiciones o productos.

## Recuerda

Un archivo **feed** o **canal RSS** está constituido por un subconjunto de instrucciones del lenguaje XML. Por lo tanto la información que contienen dichos archivos está estructurada mediante marcas o etiquetas.

### 3.3 Estructura de los canales de contenidos

Los **canales de contenidos** son archivos que se editan en RSS. RSS es un sublenguaje que surge de la aplicación del lenguaje de marcas XML. Por lo tanto, un archivo RSS es un documento de texto compuesto por etiquetas, acotadas entre los símbolos mayor y menor, que son similares a las utilizadas también en el XHTML y HTML (Figura 3.6).

```
<rss version="0.92">
<channel>
    <title>OnSoftware</title>
    <link>http://es.onsoftware.com</link>
    <description>El blog de software de Softonic</description>
    <lastBuildDate>Thu, 02 Apr 2009 10:55:05 +0000</lastBuildDate>
    <docs>http://backend.userland.com/rss092</docs>
    <language>en</language>

    <item>
        <title>Activar el superusuario en MAC OS X</title>
        <description>Quienes usan un Mac con cuenta de administrador pueden creer que el superusuario es algo del pasado. Al instalar MAC OS X Tiger o Leopard se pide una contraseña para el superusuario automáticamente desactivada. Cuando se necesita llevar a cabo ...</description>
        <link>http://es.onsoftware.com/p/activar-el-superusuario</link>
    </item>
    <item>
        <title>Lo mejor del mes: Marzo 2009</title>
        <description>
```

Figura 3.6

Ejemplo de lenguaje RSS.

Un archivo RSS no es más que un archivo de texto editado en lenguaje de marcas, normalmente editado en XML, que contiene información sobre el contenido de una determinada página web. Este archivo está estructurado con etiquetas, al igual que sucede con las páginas web. El documento RSS que se crea no tiene por qué utilizar la extensión ".rss", aunque es frecuente encontrarlo con esta nomenclatura. La extensión ".xml" también es común, ya que se trata del lenguaje que se utiliza en la sindicación de noticias. En otras ocasiones el archivo podrá llamarse de cualquier forma y será el navegador el encargado de interpretar que se trata de un canal RSS. Por lo que respecta al intercambio de datos entre empresas, aplicaciones, comunidades o personas, independientemente de la plataforma y software utilizados que utilicemos, podemos destacar **RDF** (*Resource Description Framework*), que es un lenguaje de marcas utilizado para describir los recursos web, como el título, el autor, la fecha de modificación, el contenido y la información de *copyright* de una página web. También podemos destacar el **OWL** (*Web Ontology Language*), que es un lenguaje de ontologías web, construido en la parte superior de RDF,

que sirve para procesar la información en Internet. El SPARQL es un lenguaje de consulta para RDF o lenguaje Query para RDF. SPARQL ofrece a los desarrolladores una forma de escribir las consultas que incorpora funciones para la recuperación de sentencias RDF a través de la amplia gama de RDF de información en Internet.

Por lo tanto, un canal de contenido está compuesto por un documento RSS, cuyas versiones pueden ser las 0.9x, 1.0 o 2.0, lo que viene a ser lo mismo que un archivo RDF y un archivo Atom, que, a la vez, son lo mismo que un canal RSS. Las extensiones más utilizadas para nombrar archivos correspondientes a canales o *feeds* son ".xml", ".rdf" o ".rss", aunque también pueden encontrarse extensiones tipo ".php" o ".js" menos comunes en este tipo de archivos.

El proceso de creación de canales de contenido se estructura de la siguiente manera: En primer lugar, se ha de crear el *feed* desde un editor de texto o "Bloc bloc de Notas" escrito en lenguaje de marcas. El documento XML debe incluir una declaración que lo determine como tal. Por esta razón, la primera línea del código RSS será la que define el lenguaje de marcas y el tipo de caracteres que utiliza. A continuación debemos elegir una de las tres especificaciones de RSS que existen y marcarla dentro de las etiquetas correspondientes <rss>. Las demás etiquetas que se utilizan para crear un *feed* han de situarse entre las dos etiquetas de <rss>, porque son las que indicarán que se está creando un canal RSS en el que se introducirán los contenidos queremos mostrar a los demás usuarios. Bastará con escribir la etiqueta <channel>. Cualquier *feed* o canal RSS se compone esencialmente de dos partes:

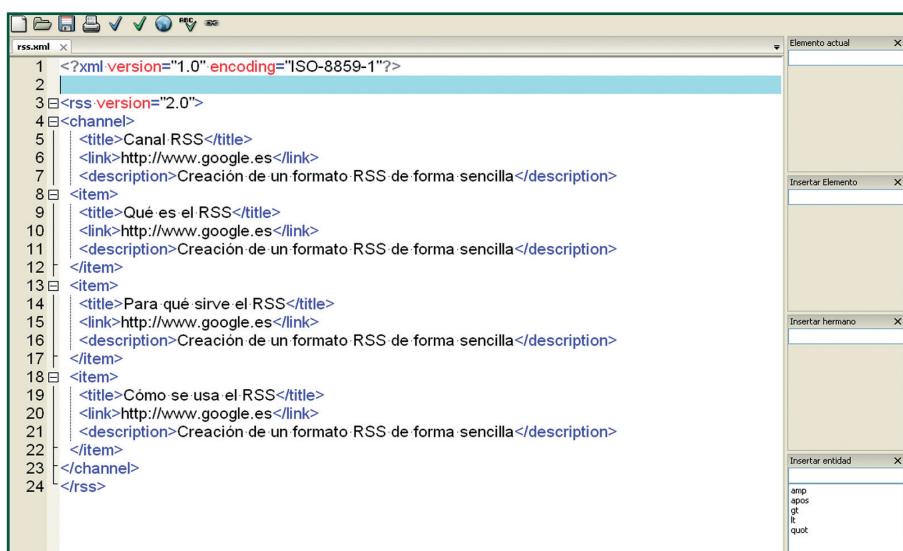
- **Elementos no variables.** Hay tres elementos no variables obligatorios en todos los canales RSS: el título, <title>, que hace referencia al nombre de nuestro *feed*; el enlace, <link>, que es la URL de nuestro sitio web; y la descripción, <description>, que informará al usuario del tipo de contenidos que se va a incluir en el canal. Estas tres líneas de código se escriben entre las dos etiquetas <channel>.
- **Elementos variables.** Los elementos variables de un canal RSS se denominan *ítem* y pueden incluirse varios *ítems* en un mismo canal. Cada ítem se crea con una etiqueta de principio y otra de final y se sitúa dentro de las etiquetas de <channel> justo después de los elementos no variables. Obligatoriamente debemos incluir tres elementos variables, aunque, como en el caso anterior, existen más. Los elementos obligatorios son: título, <title>; enlace, <link>; y descripción, <description>. Estos elementos describen cada uno de los artículos o informaciones que se van a ofrecer, y cuyo contenido se irá actualizando cada cierto tiempo.

Por último, una vez creado el canal y los artículos del *feed*, se debe guardar el código con el nombre adecuado y con las extensiones .rss, .rdf o .xml, ok, ya que, aunque sea un documento RSS, está escrito en lenguaje XML. De las tres extensiones, la que

más se utiliza es .rss, aunque puede ocurrir que no se lea porque no dispongamos de un navegador algo antiguo o debido a que algún *host* no lo reconozca. (Figura 3.7). Finalmente, para hacer accesible un canal RSS se debe crear un enlace desde un sitio web. Bastará con que, como la que se muestra a continuación, se inserte una línea en el código HTML de la página en la que se quiera enlazar:

```
<a type="application/rss+xml" href="nombre del canal RSS.xml">

</a>
```



**Figura 3.7**  
Ejemplo de edición de un archivo RSS.

### 3.4 Tecnologías de creación de canales de contenidos

Para la creación de canales de contenido puede utilizarse una de las tres tecnologías siguientes:

1. La primera tecnología es el lenguaje de programación php combinado con lenguajes de programación de entorno servidor que nos permitan generar los feeds rss. La estructura del documento php contiene dos partes: <channel> e <item> en donde <channel> contiene a <item>. El primer bloque antes de <item> se completa con los datos del blog o web, luego la etiqueta mencionada contendrá los datos del post o noticia a mostrar.
2. La segunda tecnología es la herramienta Rss Builder, que nos permite generar los feeds sin tener conocimientos de programación. Rss Builder es un programa interesante para quien tiene una página web y quiere publicar sus propios contenidos por RSS y no tiene la tecnología o los conocimientos necesarios para

generar los archivos RSS mediante un lenguaje de programación de servidor. En ese caso, RSS Builder resulta de gran utilidad, ya que tiene una interfaz sencilla donde puedes meter tus *posts* con descripciones y links o contenidos diversos. Una vez creados puedes publicar el archivo RSS creado en tu web.

3. La tercera tecnología son las herramientas online, como por ejemplo:

<http://www.feedyes.com>  
<http://feed43.com>  
<http://www.wotzwot.com/rssx1.php>  
<http://feedity.com>

## Para saber más

La página web oficial del programa Rss Builder es: [http://home.hetnet.nl/mr\\_2/43/bsoft/rssbuilder//](http://home.hetnet.nl/mr_2/43/bsoft/rssbuilder//).

### 3.5 Validación

En la creación de documentos RSS, además de seguir unos pasos en el desarrollo del archivo, como hemos visto en el apartado anterior, se ha de tener en cuenta algunos caracteres que deberemos evitar. No olvidemos que un archivo RSS es un archivo XML y que algunos caracteres no son admitidos en XML; por lo tanto, hay que evitarlos. Deberemos insertar un texto alternativo (*text string*) en su lugar. Una vez elaborado el documento RSS siguiendo los criterios del lenguaje de marcas, procederemos a la **validación**.

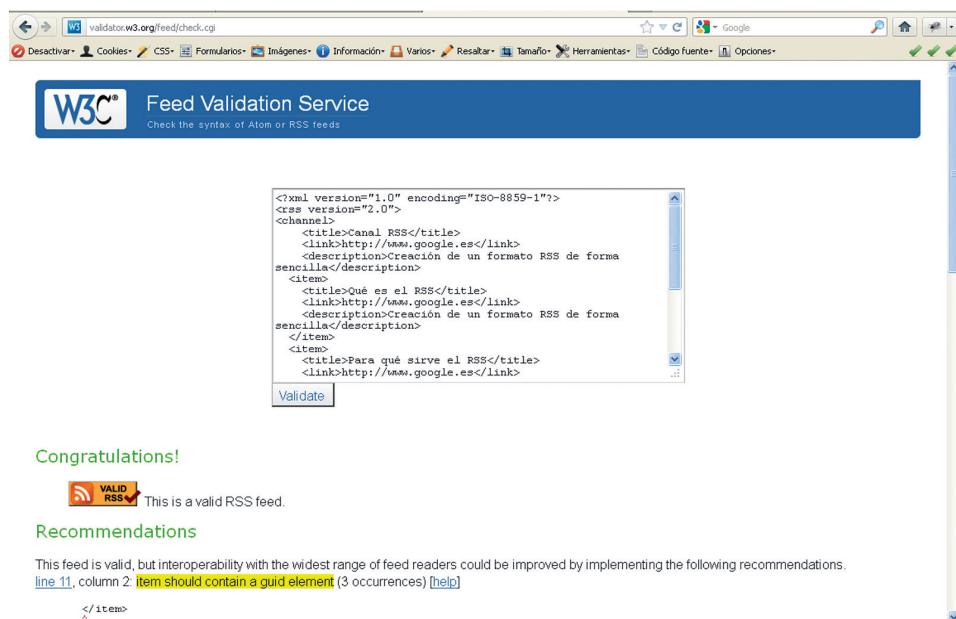
Para validar un archivo RSS, deberemos recurrir a un “validator RSS” en la red e introducir la URL del archivo en el formulario. En caso de que hubiera cualquier error, el validador lo detectará. Si, por el contrario, todo está bien codificado validará el archivo y se podrá poner un *gif* de la certificación de las páginas para confirmar la certificación.

Existen distintas versiones del lenguaje RSS: las versiones 0.90 y 0.91 son las primeras que fueron desarrolladas por Netscape. La versión 1.0 fue desarrollada por un grupo independiente que se inspiró en el formato RDF; la versión 2.0 fue el resultado de la adopción de la tecnología de Netscape por parte de la empresa UserLand Software. Todas ellas están basadas en el lenguaje originalmente definido por Netscape, pero no todas son compatibles entre sí. Veamos a continuación cuáles son las características principales de las distintas versiones del lenguaje RSS:

- **RDF.** La versión 1.0 del lenguaje RSS también es conocida como RDF. Por este motivo algunos *feeds* o canales RSS están etiquetados como “RSS 1.0” o “RDF” y guardados en archivos con extensión “.rdf”.
- **RSS2.** A la versión 2.0 del lenguaje RSS también se le llama RSS2. Por lo tanto, algunos *feeds* o canales RSS están etiquetados como “RSS2” o “RSS 2.0”.

- **Atom.** Atom también es un sublenguaje XML. No se corresponde ni se basa en ninguna versión de RSS, pero es un formato muy similar a éste y, sobre todo, tiene el mismo objetivo: permitir la distribución de contenidos y noticias de sitios web. Las mejoras que supone respecto a RSS (en cualquiera de sus versiones) hacen que su uso se extienda rápidamente a pesar de ofrecer mayor complejidad. Un documento Atom puede contener más información y ser más complejo, pero es más consistente que un documento RSS (Figura 3.8).

Como RSS y Atom están basados en el metalenguaje XML, en numerosas ocasiones los feeds RSS están etiquetados como XML y se guardan en archivos con extensión .xml. Sin embargo, también pueden tener extensión .rdf o .rss. En realidad, la extensión utilizada es lo de menos, pues lo importante es el contenido del documento.



**Figura 3.8**

Ejemplo de validación de documento RSS en W3C.

### 3.6 Utilización de herramientas

Un **lector o agregador de canales** es una aplicación local, o una aplicación basada en la web, que interpreta los archivos RSS y visualiza su contenido. Si incluimos varios canales RSS en el agregador, será posible leer una versión resumida, o previa, de los contenidos de múltiples sitios web sin necesidad de visitarlos individualmente, excepto cuando se desee leer la versión ampliada de las noticias seleccionadas.

Las noticias llegan al usuario cuando éste inicia su programa lector de canales RSS, en lugar de que el usuario, para leer las noticias, deba acudir a las diferentes fuentes o sitios web donde se publican.

Cada vez que el usuario añade un canal o *feed* a su programa agregador o lector de *feeds*, se dice que **se suscribe** a ese canal. Los archivos RSS, al igual que las páginas web, disponen de una URL o dirección de Internet; por lo tanto, para tener acceso o para suscribirse a un canal, hay que conocer, en primer lugar, su URL o dirección en Internet e indicar, en segundo lugar, cuál es dicha dirección en el programa agregador o lector de *feeds* de modo que pueda encontrarlo y mostrar su contenido.

Normalmente, los sitios web que disponen de canales RSS indican cuál es la URL de su *feed* de un modo similar a los descritos en nomenclaturas de los *feeds* o canales RSS. Existen también directorios de *feeds* RSS, listas de favoritos o buscadores de contenidos y *feeds* RSS. Existen, además, diferentes formas de acceder o suscribirse a los canales RSS. Hay muchos programas diseñados con este fin, con pocas diferencias entre unos y otros.

Algunos ejemplos podrían ser:

- **Lectores.** Son aquellas aplicaciones que se instalan en el ordenador del usuario; por ejemplo: *FeedReader*, *AmphetaDesk* y *FeedDemon*.
- **Agregadores.** Son lectores desde plataformas web, como por ejemplo: *Bloglines* y *Oddpost*.
- **Plugins para navegadores (browsers) o clientes de correo electrónico.** *NewsGator*, que adiciona canales a Microsoft Outlook y el panel lector de RSS del navegador Mozilla Firefox.
- **Navegadores con lectores de canales incorporados.** Opera, Internet Explorer y Chrome con su extensión para RSS y Atom.
- **Servicios web que suministran canales RSS suscritos a direcciones de correo electrónico.**
- **Sistemas de gestión de contenidos con módulos lectores de RSS incorporados.** PHP-Nuke o Mambo, gestores de aprendizaje como Moodle.

Entre los elementos más comunes y personalizables que contienen muchos de estos programas hay que destacar: Observatorios (*Watches*), canales especiales que buscan noticias nuevas por palabras clave; *News Bins*, áreas de almacenamiento de elementos a los que se desea retornar; Autodescubrimiento, aparece un ícono en la barra de estado cuando una página web tiene uno o más canales RSS para facilitar la suscripción.

## Recuerda

**El consorcio World Wide Web, W3C, desarrolló REDF (Resource Description Framework), basado en XML para poder definir datos.**

**Figura 3.9**

Imágenes utilizadas para identificar un enlace o canal RSS.

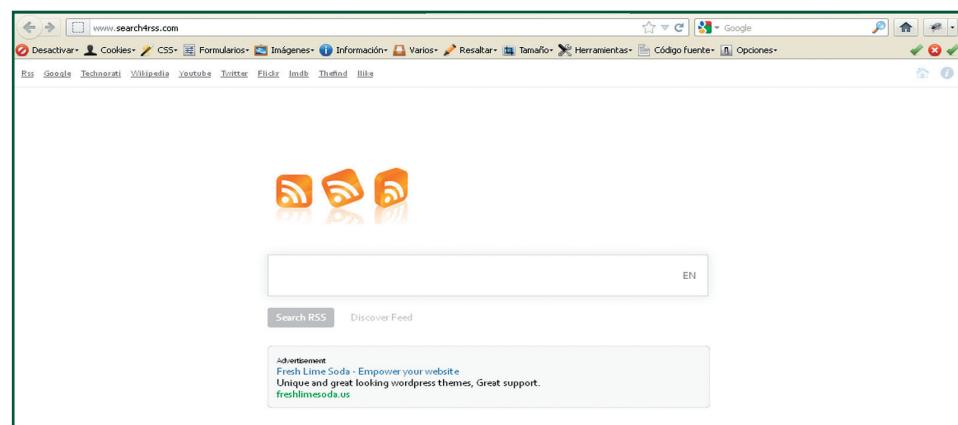
Otras formas de identificar un enlace a un canal o feed RSS que se utiliza habitualmente son las siguientes: *Syndicate this site* (Sindica este sitio), *Site Syndication* (Sindicar sitio), Canal o *Feed XML/RSS/RDF* (Canal o Archivo de suscripción), etc. También se utiliza una variada serie de iconos con algunas de las siglas referidas sobre fondo azul o naranja principalmente (Figura 3.9).



Otras aplicaciones pueden considerarse directorios o motores de búsqueda de RSS. El motor de búsqueda indica los ficheros RSS; los contenidos suelen tomarse de *weblogs* o de sitios de noticias; los suscriptores indican el tema que les interesa y reciben notificación tan pronto como su solicitud coincide con los registros que se añaden al motor de búsqueda.

## 3.7 Directorios de canales de contenidos

Los **directorios de canales de contenido** se conocen también como *motores de búsqueda de RSS*. El motor de búsqueda indica los ficheros RSS, en los que el suscriptor ha indicado el tema, o los temas, que desea seguir y recibir información tan pronto como su solicitud coincide con los requisitos añadidos al motor de búsqueda. La información que se publica por medio de RSS son notificaciones que informan cada vez que se actualiza un contenido, lo que hace que los nuevos contenidos estén disponibles de inmediato para los lectores o agregadores de canales y los motores de búsqueda de RSS (Figura 3.10).



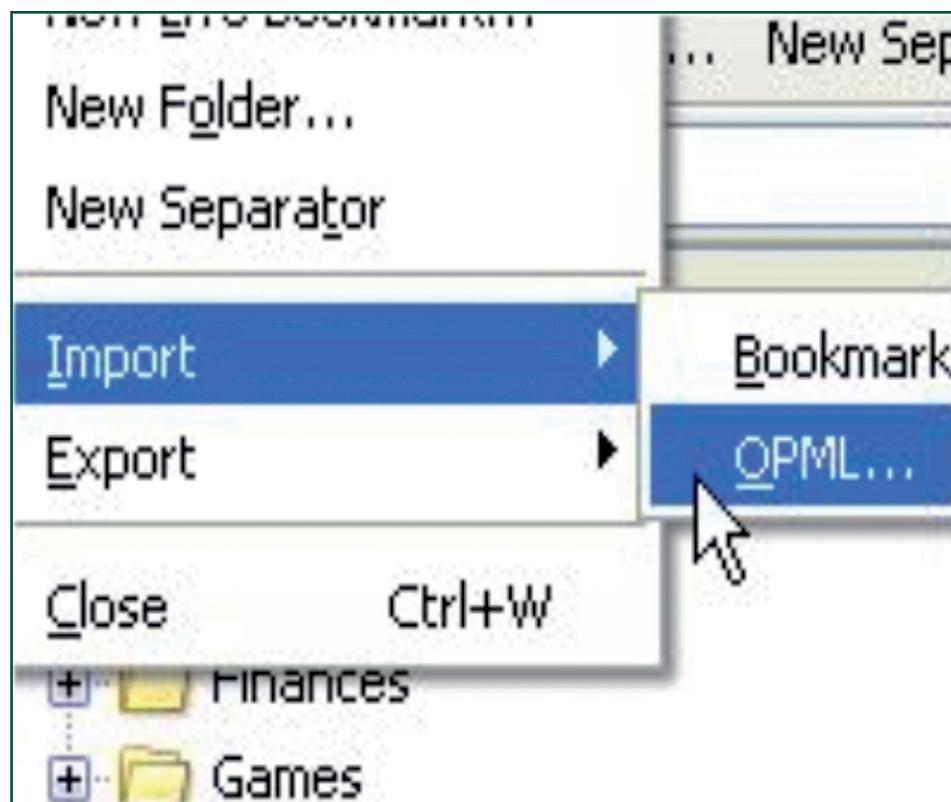
**Figura 3.10**

Ejemplo de buscador de RSS, Search 4 RSS.

A diferencia de las páginas web comunes, que son esencialmente pasivas y generalmente no están accesibles hasta que el motor de búsqueda las encuentra, y aun así, la mayoría de estas páginas tienen pocas oportunidades de que los usuarios naveguen en ellas, los motores de búsqueda de RSS pueden utilizarse para encontrar información casi en tiempo real.

Actualmente, todo tipo de contenido en la web es susceptible de sindicarse y, por lo tanto, de compartirse y distribuirse. Los canales se comparten mediante la aplicación XML denominada **OPML** (*Outline Processor Markup Language*) (Figura 3.11). Esta aplicación permite intercambiar listas de suscripción entre programas que leen ficheros RSS, como lectores y agregadores. Cuando un individuo o institución dispone en línea sus ficheros OPML, posibilita a otros que puedan suscribirse a todos los canales simplemente importando el OPML.

De esta manera, un portal puede incluir una lista de ficheros OPML, cada uno de los cuales incluye canales sobre temas específicos. OPML es muy utilizada por los suscriptores para sincronizar sus listas de canales entre sus ordenadores; por ejemplo, entre la casa y el trabajo, pero también puede utilizarse para sincronizar listas de canales entre los ordenadores de una biblioteca u otra organización, o para compartir documentos que pueden elaborarse en colaboración entre dos o más personas.



**Figura 3.11**

La aplicación OPML permite intercambiar listas de suscripción entre programas que leen archivos RSS.

### 3.8 Agregación

La **agregación de canales RSS** consiste en obtener los archivos RSS de fuentes de contenidos externas para presentarlos en nuestro sitio web. Eso es posible gracias a pequeñas aplicaciones llamadas **agregadores** o **lectores de feeds**, que se encargan de leer e interpretar esos archivos (Figura 3.12).



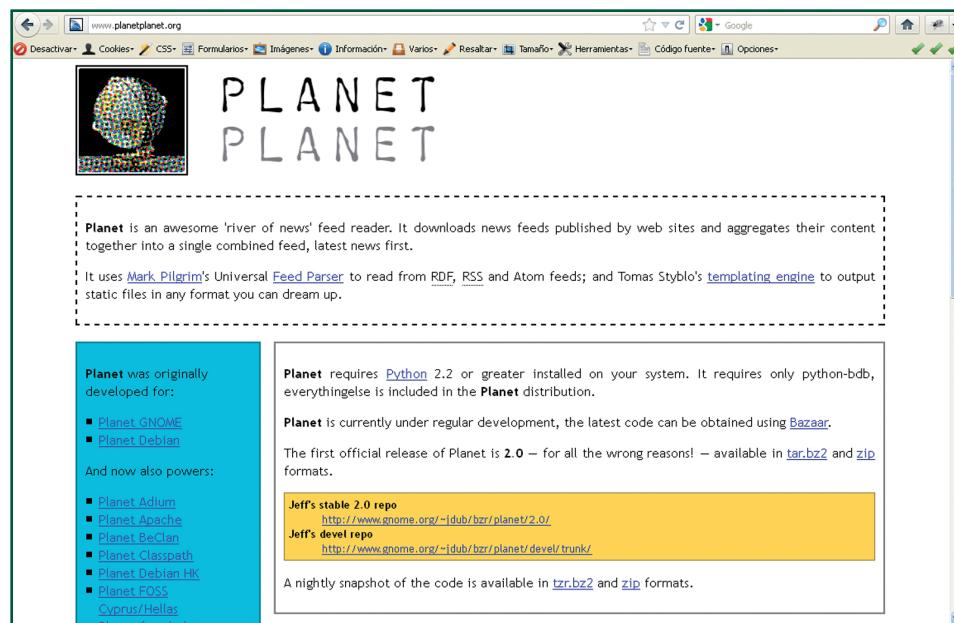
**Figura 3.12**  
Lector de feeds de Google.

La agregación nos permite acceder a una mayor información en menos tiempo, razón por la que hay que ser selectivos, para evitar intoxicarse con un exceso de información que no sea de nuestro interés.

A los agregadores, herramientas para la agregación de contenidos, se les llama también lectores de *feeds*, lectores RSS, *readers*, etc. Son programas a través de los cuales podemos acceder a todos los contenidos a los que estamos suscritos, siendo el punto de acceso más rápido y eficaz al menú informativo diario. Mediante los agregadores podemos suscribirnos a noticias, secciones, blogs, búsquedas predefinidas, fotos, vídeos, audio, etc., es decir, a cualquier contenido que publique un canal RSS.

Pese a que hemos hablado ya con anterioridad de las ventajas de los agregadores, éstos constituyen el único punto de acceso para conocer las últimas novedades que sean de nuestro interés. No tenemos que buscar la información, sino que la información viene a nosotros, lo que constituye un ahorro de tiempo y dinero. Sin embargo, debemos señalar un inconveniente al respecto, y es que se descontextualiza el contenido de la fuente original, es decir, no podemos acceder a las herramientas de enriquecimiento de contenido que suelen ofrecer las fuentes originales.

La combinación de varios canales RSS en uno solo también es posible. Para ello se han desarrollado los **planets**, que son agregadores de canales RSS con los que puede crearse una página web que contenga las últimas actualizaciones de los autores de los blogs propietarios de dichos canales (Figura 3.13). Para ello, basta con suscribir los canales deseados al *planet* para que las anotaciones aparezcan según se escriben. Toda la información afín, ya sea porque trate el mismo tema o porque pertenezca a un mismo grupo de usuarios, se concentra en un mismo sitio para poder consultarse dinámicamente. Un lector o agregador de *feeds* es una aplicación local, o basada en web, que interpreta los archivos RSS y visualiza su contenido.



**Figura 3.13**  
Ejemplo de herramienta para crear *planets*.

## Resumen

La sindicación de contenido o RSS (acrónimo que puede representar varias entidades: *Rich Site Summary*, *RDF Site Summary* o *Really Simple Syndication*) se conoce también como suscripción a un canal o *feed RSS*, y es la encargada de indicar a un determinado programa o página web que revise, cada cierto tiempo, un canal de información en busca de contenido adicional o información actualizada. Los archivos RSS se llaman comúnmente *feeds RSS* o canales RSS, y contienen un resumen de lo publicado en el sitio web de origen.

Los ámbitos de aplicación de la sindicación de contenidos abarcan muchos aspectos; por ejemplo, los medios de comunicación los emplean para distribuir sus noticias, y los sitios comerciales encuentran nuevos modelos de negocio que permiten la suscripción mediante RSS para informar sobre sus productos.

Los canales de contenidos son archivos que se editan en RSS. RSS es un sublenguaje surgido de la aplicación del lenguaje de marcas XML. Por tanto, un archivo RSS es un documento de texto compuesto por etiquetas acotadas entre los símbolos mayor y menor, similares a las utilizadas también en los lenguajes XHTML y HTML. Como lenguaje de marcas, el documento RSS puede describirse y validarse.

Los directorios de canales de contenido son también conocidos como los motores de búsqueda de RSS. La agregación de canales RSS consiste en obtener, de fuentes de contenidos externas, archivos RSS para presentarlos en nuestro sitio web, lo que es posible gracias a pequeñas aplicaciones llamadas agregadores, o lectores de *feeds*, que leen e interpretan esos archivos.

Existen diferentes formas de acceder o subscribirse a los canales RSS. Para ello, se pueden utilizar diferentes tipos de herramientas, como lectores (aplicaciones en el ordenador del usuario que interpretan el RSS y lo muestran de manera correcta), agregadores (lectores desde plataformas web), *plugins* para navegadores o navegadores con lectores incorporados, y servicios web que suministran canales RSS.

## Ejercicios de autocomprobación

**Indica si las siguientes afirmaciones son verdaderas (V) o falsas (F):**

1. La aplicación en medios de comunicación es un ámbito de actuación en la sindicación de contenidos.
2. Cualquier feed o canal RSS se compone de tres elementos no variables obligatorios: el título, <title>, que hace referencia al nombre de nuestro feed; el enlace, <link>, que es la URL de nuestro sitio web; y la descripción, <description>, que informará al usuario del tipo de contenidos que se va a incluir en el canal.
3. Para la creación de canales de contenido no puede utilizarse el lenguaje de programación php combinado con lenguajes de programación de entorno servidor, pues no permite generar los feeds rss.
4. Para validar un archivo RSS, deberemos recurrir a un “validator RSS” en la red e introducir la URL del archivo en el formulario. En caso de que hubiera cualquier error, el validador lo detectará. Si, por el contrario, todo está bien codificado, validará el archivo y se podrá poner un gif de la certificación de las páginas para confirmar la certificación.
5. Actualmente, todo tipo de contenido en la web es susceptible de sindicarse y, por lo tanto, de compartirse y distribuirse. Los canales se comparten mediante la aplicación XML denominada OPML (Outline Processor Markup Language), que permite intercambiar listas de suscripción entre programas que leen ficheros RSS, como lectores y agregadores.
6. La agregación de canales RSS consiste en obtener los archivos RSS de fuentes de contenidos externas para presentarlos en nuestro sitio web. Eso es posible gracias a pequeñas aplicaciones llamadas agregadores o lectores de feeds, que se encargan de leer e interpretar esos archivos.
7. Un lector o agregador de feeds es una aplicación local, o basada en web, que interpreta los archivos HTML y visualiza su contenido.

**Completa las siguientes afirmaciones:**

8. La sindicación de contenidos o RSS, también conocida como \_\_\_\_\_, está basada en el lenguaje de marcas \_\_\_\_\_ y no es más que el reenvío de información contenida en un determinado sitio web a otro sitio de destino. Los \_\_\_\_\_ RSS se llaman comúnmente \_\_\_\_\_ RSS o \_\_\_\_\_ RSS y contienen un resumen de lo publicado en el sitio web de origen.
9. A los \_\_\_\_\_, herramientas para la agregación de contenidos, se les llama también lectores de feeds, lectores RSS, readers, etc. Son \_\_\_\_\_ a través de los cuales podemos acceder a todos los contenidos a los que estamos \_\_\_\_\_, siendo el punto de acceso más rápido y eficaz al menú informativo diario. Se puede suscribir a noticias, secciones, blogs, búsquedas predeterminadas, fotos, vídeos, \_\_\_\_\_, etc., es decir, a cualquier contenido que publique un canal RSS.
10. La \_\_\_\_\_ de varios canales RSS en uno solo también es posible. Para ello se han desarrollado los \_\_\_\_\_, que son agregadores de canales RSS con los que puede crearse una \_\_\_\_\_ que contenga las últimas actualizaciones de los autores de los blogs propietarios de dichos canales. Para ello, basta con suscribir los canales deseados para que las anotaciones aparezcan según se escriben. Toda la \_\_\_\_\_ afín, ya sea porque trate el mismo tema o porque pertenezca a un mismo grupo de usuarios, se concentra en un mismo sitio para poder consultarse dinámicamente.

Las soluciones a los ejercicios de autocomprobación se encuentran al final de esta Unidad Formativa. En caso de que no los hayas contestado correctamente, repasa la parte de la lección correspondiente.

## 4. DEFINICIÓN DE ESQUEMAS Y VOCABULARIOS EN XML

El **lenguaje de marcas XML** tiene una ventaja que puede convertirse en un inconveniente: cualquiera puede crear sus propias etiquetas. Esto no es un problema si trabajamos solos; pero, ¿y si trabajamos en equipo?, ¿y si queremos exportar nuestros documentos?, ¿qué estándar seguiremos? ¿Cómo conseguiremos que nuestros documentos sean entendibles por otros usuarios? Por lo tanto, es importante definir el estándar que vayamos a utilizar en la edición de nuestros documentos XML para poder después procesarlos y compartirlos de manera adecuada, tanto con otros usuarios como con otros procesadores.

En esta unidad estudiaremos el lenguaje de marcas XML desde su estructura y sintaxis, conoceremos los métodos de definición de documentos XML y aprenderemos a editarlos y validarlos para cumplir un estándar y para que puedan ser correctamente interpretados.

De esta manera, facilitaremos que cualquier medio que tenga que interpretar nuestro archivo XML lo haga de la misma manera, y así, por su forma y estructura, poder incluirlo dentro de un grupo de archivos y documentos similares con el fin de facilitar su clasificación e interpretación, y de ese modo conseguir su estandarización.

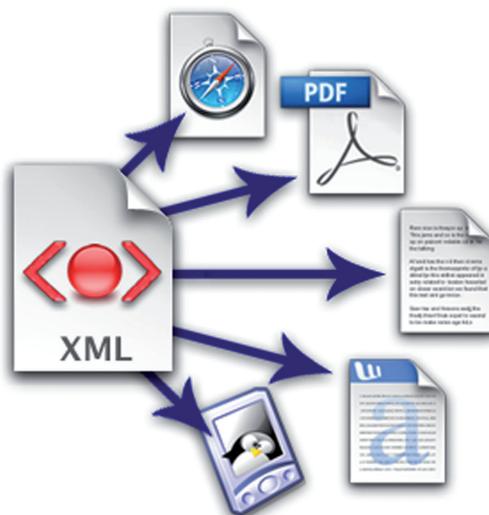
### 4.1 Definición de la estructura y sintaxis de documentos XML

Los **documentos XML** tienen una **estructura lógica** y una **estructura física**. La estructura lógica consta de un conjunto de declaraciones, elementos, comentarios, etc., que están incluidos en el documento mediante unas determinadas marcas. La estructura física consta de una serie de unidades, llamadas **entidades**, que indican los datos que contendrá el documento. Ambas estructuras deberán encajarse correctamente.

Como hemos dicho, las estructuras lógicas están indicadas mediante sus correspondientes marcas. Todos los elementos contenidos en ellas deben comenzar con una marca de inicio y terminar con una marca de finalización. En cuanto a las estructuras físicas, una entidad puede referirse a otras entidades con la finalidad de provocar su inclusión en el documento, lo que significa que la estructura de un documento XML permite anidar entidades y elementos dentro de otros mediante una estructura de árbol que se divide en ramas. Un documento comienza en una **entidad raíz**, o entidad de documento (*document entity*), que contiene a los demás elementos anidados; en un documento XML hay un solo elemento raíz.

La tecnología XML pretende dar solución al problema de expresar información estructurada de la manera más reutilizable posible. Una información estructurada significa que se compone de partes bien definidas que, a su vez, pueden contener otras partes. Dichas partes se llaman **elementos** y se les marcan mediante la utilización de etiquetas. Una etiqueta es una marca hecha en el documento, que señala una porción de éste como un elemento, es decir, un trozo de información que tiene un sentido claro y definido. Las etiquetas presentan la forma <nombre\_etiqueta> para nombrar el elemento que se está señalando.

Un documento XML hace uso de marcas para describir la estructura o las partes que lo componen. El significado de estas marcas está especificado en otro documento anexo, la DTD (*Document Type Definition*), que puede estar incluido en el documento XML principal o ser externo y utilizar un archivo de texto separado (Figura 4.1).



**Figura 4.1**

Un documento XML puede compartirse e interpretarse a través de diferentes aplicaciones.

## 4.2. Utilización de métodos de definición de documentos XML

Una **definición de tipo de documento** es una descripción de la estructura y la sintaxis de un documento XML. Su función es la de indicar la descripción de la estructura de datos del tal forma que nos permita el uso de una estructura común que mantenga la consistencia entre todos los documentos que utilicen la misma definición de tipo de documento. De esta forma, dichos documentos pueden ser validados dentro de un grupo de trabajo que utilice el mismo tipo de información. Un documento XML es válido cuando tiene una declaración de tipos asociada y el propio documento cumple con las reglas de dicha declaración. La declaración de tipos debe aparecer antes que el primer elemento del documento. La declaración de tipos contiene o hace referencia a aquellas declaraciones de marcas que proporcionan una gramática para esa clase de documentos.

Los documentos XML deben comenzar con una declaración XML que especifique cuál es la versión utilizada; por otra parte, el programa empleado para la interpretación del documento XML debería indicar con una señal de error la apertura de un documento cuya versión no pueda procesar. Algunas de las tecnologías más extendidas para la definición de tipos de documentos son las DTD (*Document Type Definition*), los esquemas XML (*XML Schema*), RELAX y Schematron.

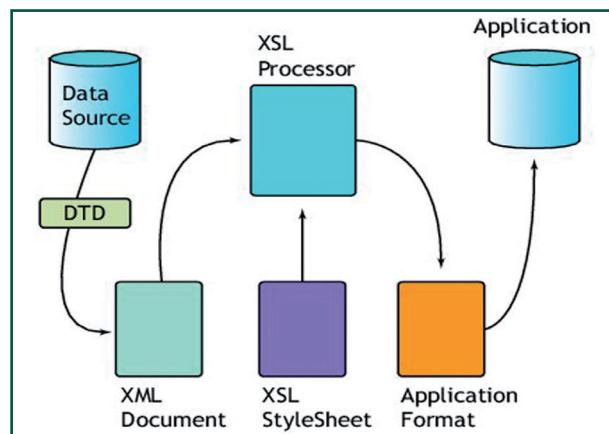
#### 4.2.1 DTD (*Document Type Definition*)

#### Recuerda

**Los documentos XML son utilizados principalmente para almacenamiento de información por su fácil interpretación y flexibilidad a la hora de crearlos.**

La **definición de tipo de documento** (DTD) es una descripción de estructura y sintaxis de un documento XML. Una DTD contiene un elemento raíz, así como una descripción de todos los elementos que intervienen en dicho elemento raíz. Incluye, además, los elementos permitidos y sus atributos, así como las reglas que afectan a la anidación de los primeros y a los valores de los segundos. Si cotejamos un documento con su DTD podremos comprobar si éste tiene una estructura y sintaxis válidas o no (Figura 4.2).

Las DTD se emplean generalmente para determinar la estructura de un documento mediante etiquetas. Una DTD describe los elementos que indican qué etiquetas son permitidas y el contenido de dichas etiquetas, la estructura que muestra el orden en el que aparecen las etiquetas en el documento, y el anidamiento que indica qué etiquetas van dentro de otras.



**Figura 4.2**

Ejemplo de cómo afecta el uso de DTD a un documento XML.

Existen tres formas de vincular los documentos XML con las DTD:

- **DTD interna.** Aquella que podemos encontrar dentro del documento XML.
- **DTD externa.** Aquella que podemos encontrar fuera del documento XML.
- **DTD.** Aquella que podemos encontrar tanto dentro como fuera del documento XML.

La forma más habitual de encontrar la vinculación de la DTD con el documento XML es mediante una DTD externa, puesto que nos permite beneficiarnos de la reutilización de DTD, es decir, una misma DTD puede ser usada por varios documentos XML. Sólo es recomendable el uso de una DTD interna en aquellos casos en los que la creación de un documento DTD externo supusiera un gran coste.

A continuación, veamos un ejemplo de uso de una DTD externa “libro.dtd” para un elemento, llamado libro, que será nuestro elemento raíz:

```
<!ELEMENT libro (autor,titulo,capitulo+)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT capitulo (titulocap,texto)>
<!ELEMENT titulocap (#PCDATA)>
<!ELEMENT texto (#PCDATA)>
```

Como podemos observar en este ejemplo, la definición comienza con `<!ELEMENT libro (autor,titulo,capitulo+)>`. Los nombres entre paréntesis y separados por comas indican que un libro consta de un autor, un título y uno o más capítulos (este es el significado del signo +) de forma obligatoria. Si deseáramos indicar que no todos los campos son obligatorios sino que alguno puede ser opcional, sustituiríamos el carácter separador, coma, por () y añadiríamos el signo (?) en el campo, para indicar que este campo no es ya obligatorio sino opcional.

En el resto de líneas, podemos ver la definición y el orden de los tipos de datos que vamos a tener en nuestro elemento raíz. El elemento final de la definición (#PCDATA) nos indica que el tipo de datos que podemos guardar en ese campo es de tipo carácter (*Parser Character Data*).

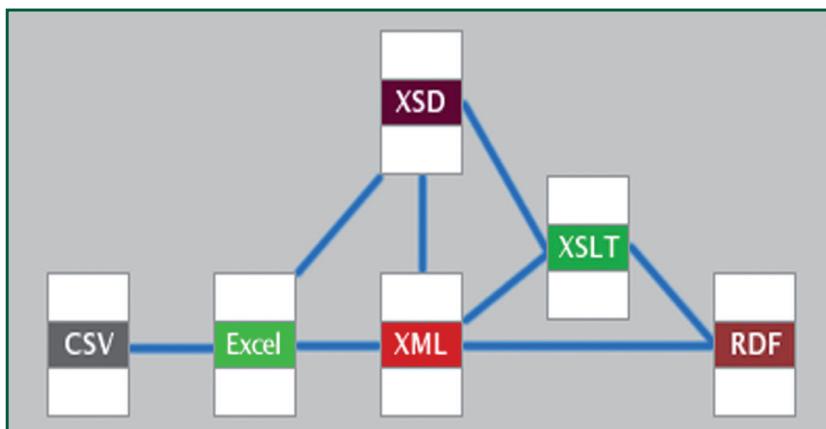
Una vez hayamos creado nuestra DTD, el siguiente paso será **referenciarla** dentro del código XML. Para conseguir nuestro objetivo, añadiremos una referencia a la DTD externa, en la línea de identificación de tipo de documento, tal como se muestra a continuación:

```
<!DOCTYPE libro SYSTEM "libro.dtd">
```

#### 4.2.2 Esquemas XML

Los **esquemas XML** son una tecnología que permite la validación de la definición de tipos de documentos XML. A diferencia de las DTD, en las que sólo se permite establecer que el contenido de un elemento sea una cadena de texto, en los es-

quemas XML puede validarse, además, que un elemento cumpla una característica determinada (Figura 4.3). Los documentos esquema suelen utilizar la extensión XSD (XML Schema Definition).



**Figura 4.3**

Ejemplo de cómo el uso de XSD afecta a un documento XML.

Otra ventaja que presentan los esquemas XML con respecto a las DTD es que éstas utilizan la misma sintaxis que los documentos XML, lo que sólo nos obliga a aprender unas pocas etiquetas adicionales que nos permitirán codificar este tipo de elementos.

El modelo de contenido de los esquemas XML es abierto, lo que significa que es posible añadir elementos y atributos secundarios que no hayan sido definidos en el esquema del documento. Por el contrario, las DTD tienen modelos de contenido cerrados, lo que obliga a declarar todos los elementos y atributos que se utilicen.

Los esquemas XML permiten la herencia de elementos, que consiste en asociar elementos derivados entre sí, reteniendo las relaciones existentes entre ellos. Del mismo modo, permiten los tipos de datos restringidos, con los que puede controlarse el intervalo o el formato de los datos de un determinado elemento o atributo. Los esquemas XML posibilitan la creación de vocabularios XML específicos, que definen claramente las relaciones entre los elementos del documento XML. De esta manera pueden definirse vocabularios para el ámbito del dominio que se necesiten, desde el marcado de información para dispositivos WAP (vocabulario WML) hasta el marcado necesario para la representación de gráficos (vocabulario SVG).

Los esquemas XML tienen la capacidad de establecer restricciones en la estructura de los documentos XML mediante dos metodologías:

- Determinando el modelo de contenido de los documentos, es decir, definiendo el orden y la anidación de los elementos, así como los tipos de datos que pueden contener los elementos.

## Para saber más

**La herramienta**  
**Definición de esquemas XML (Xsd.exe) genera clases de esquemas XML o de Common Language Runtime a partir de archivos XDR, XML y XSD, o a partir de clases de un ensamblado de motor en tiempo de ejecución.**

- Mediante la utilización de los espacios de nombres en los esquemas XML, que permite que dos elementos distintos tengan el mismo nombre, cada uno correspondiente a su espacio de nombres. Para poder utilizar de forma correcta los espacios de nombres nos basaremos en las referencias URI (*Uniform Resource Identifiers*), que nos garantizarán que el nombre de los recursos a los que hacemos referencia sean únicos. Veamos a continuación un ejemplo de esquema XML:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="libro">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="autor" type="xs:string"/>
        <xs:element name="titulo" type="xs:string"/>
        <xs:element name="titulocap" type="xs:string"/>
        <xs:element name="texto" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

### 4.2.3 RELAX NG

La tecnología RELAX NG sirve para la definición y representación de tipos de documentos XML. RELAX NG utiliza la sintaxis XML. Está basada en la gramática y, con respecto a la utilización de los esquemas XML, es muy intuitiva y más fácil de entender, de aprender y de utilizar; lo que, hoy en día, hace comprensible su popularidad.

RELAX NG tiene, además, un elevado poder expresivo, lo que le permite, por ejemplo, validar elementos intercalados que pueden aparecer en cualquier orden.

Las aplicaciones de definición de documentos y validación para RELAX NG son más sencillas que en los esquemas XML, ya que son más fáciles de utilizar e implementar; tienen, además, la capacidad de usar *plug-ins* de definiciones de tipos de datos utilizados en los esquemas XML, lo que le permite a las aplicaciones de definición de documentos disponer de la combinación de las ventajas de ambos lenguajes. RELAX NG soporta, además, espacios de nombres y tiene capacidad para añadir librerías de datos. RELAX NG se ha convertido recientemente en un estándar ISO igual que la segunda parte de DSDL (*Document Schema Definition Languages*).

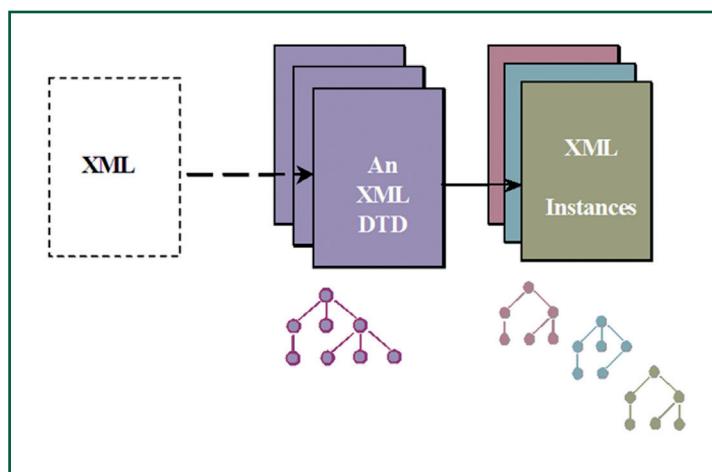
#### 4.2.4. Schematron

La tecnología **Schematron**, a diferencia de las anteriormente descritas, se inspira en afirmaciones en lugar de en la gramática. Al basarse en una serie de reglas, utiliza expresiones de acceso, en vez de expresiones gramaticales, para definir lo que se permite en un documento XML, y lo que no. Si el documento cumple estas reglas, podremos considerarlo un documento XML válido.

Este método de validación de los documentos XML aporta una gran flexibilidad en la descripción de estructuras relacionales. El único inconveniente es que se trata de una tecnología muy limitada a la hora de determinar la estructura básica del documento, si bien dicho inconveniente puede solucionarse combinando Schematron con las otras tecnologías de definición de tipos de documentos XML. Al igual que con RELAX NG, Schematron se está estandarizando como una parte del DSDL del modelo ISO.

### 4.3 Creación de descripciones

Crear una **descripción** es como crear nuestro propio lenguaje de marcas para una aplicación específica. Por ejemplo, podemos crear una descripción que defina una tarjeta de visita. A partir de ella, tendremos una serie de elementos XML que nos permitirán definir tarjetas de visita. Puesto que XML es un sistema para definir lenguajes, no puede tener una sola descripción (al contrario que HTML); de modo que quien necesite utilizar XML para intercambio de datos deberá definir su propia descripción. Básicamente, una descripción establece las reglas de un documento e indica qué etiquetas pueden usarse, qué atributos, las relaciones entre etiquetas, etc. (Figura 4.4).



**Figura 4.4**

La definición de tipo de documento debe crearse para cada documento XML.

## Recuerda

**Las entidades son tipos de objetos que permiten representar caracteres que no pueden incluirse como texto, caracteres especiales, etc. Las entidades permiten, al ser referenciadas, que el procesador reemplace el identificador por el contenido asignado como valor, que puede incluir marcado.**

Las descripciones admiten cuatro tipos de declaraciones, tal como puede apreciarse a continuación:

- **Declaraciones de tipo de elemento (*Element Type Declarations*)**. Establecen qué elementos pueden formar parte del documento y cuáles pueden formar parte de su interior (los elementos se anidan unos dentro de otros).
- **Declaraciones de listas de atributos (*Attribute List Declarations*)**. Los atributos se utilizan para añadir información adicional a los elementos de un documento.
- **Declaraciones de Entidades (*Entity Declarations*)**. XML hace referencia a objetos que no deben ser analizados sintácticamente según las reglas XML, sino mediante el uso de entidades. Las entidades pueden ser internas o externas, analizadas o no analizadas, y generales o parametrizadas.
- **Declaraciones de notación (*Notation Declarations*)**. Se utilizan para indicar el tipo de ficheros binarios referenciados mediante las indicaciones a entidades externas.

La declaración del tipo de documento comienza en la primera línea y termina con "].>". Las **declaraciones de tipo elemento** son las líneas que empiezan con "<!ELEMENT". También pueden declararse atributos, entidades y anotaciones para una descripción. La descripción puede definirse parcial o completamente en otro lugar.

Los documentos XML que se ajustan a su descripción se denominan **válidos**. El concepto de validez no tiene nada que ver con el de estar bien formado. Un documento bien formado es aquel que, simplemente, respeta la estructura y sintaxis definidas por la especificación de XML. Un documento bien formado puede, además, ser válido si cumple las reglas de una descripción determinada. También existen documentos XML sin una descripción asociada; en ese caso no son válidos, pero sí, en cambio, pueden estar bien formados.

A continuación, mostraremos dos maneras de especificar formatos de comunicación en XML, entre los que destacan las DTD y los XML Schema.

### 4.3.1 DTD (*Document Type Definition*)

La DTD es la descripción de cómo se construye un documento XML, desde el punto de vista estructural y sintáctico, para que se ajuste a las necesidades previamente analizadas, es decir, la DTD establece qué elementos son aceptados y qué posiciones deben ocupar en el interior de un documento XML.

Cuando se define una DTD, y se referencia dentro de un documento XML, se establece una relación entre el léxico y las reglas sintácticas que debe cumplir el lenguaje.

El significado semántico se proporciona en el uso de la DTD y del documento XML al generar las aplicaciones que lo utilizan.

Por lo tanto, antes de crear un documento XML, deberemos analizar qué elementos y etiquetas vamos a necesitar para la aplicación que queramos crear. Veamos a continuación un ejemplo de cómo utilizar una DTD. Si quisieramos almacenar los mensajes de móviles que se envían a un servidor, los datos que se guardarían serían los siguientes:

Número de teléfono del usuario.

Fecha de envío.

Hora de envío.

Contenido del mensaje.

Un ejemplo de documento XML que guardara estos mensajes SMS podría ser el siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE BDsms SYSTEM "bdSMS.dtd">
<bdSMS>
<SMS>
<telefono>123456</telefono>
<fecha>20/03/2012 </fecha>
<hora>10:35</hora>
<mensaje>Hola, ¿cómo estas?</mensaje>
</SMS>
<SMS>
<telefono>654321</telefono>
<fecha>22/03/2012 </fecha>
<hora> 15:05</hora>
<mensaje>estoy muy bien</mensaje>
</SMS>
<SMS>
<telefono>111111</telefono>
<fecha> 25/03/20 12 </fecha>
<hora>09:10</hora>
<mensaje>Hoy hace mucho frio</mensaje>
</SMS>
</bdSMS>
```

La DTD que permite establecer el formato de intercambio de estos mensajes SMS en el documento XML será la siguiente (fichero "bdSMS.dtd"):

```
<!ELEMENT bdSMS (SMS*)>
<!ELEMENT SMS (telefono, fecha, hora, mensaje)>
<!ELEMENT telefono (#PCDATA)>
<!ELEMENT fecha (#PCDATA)>
<!ELEMENT hora (#PCDATA)>
<!ELEMENT mensaje (#PCDATA)>
```

Debemos señalar que el fichero bdSMS.dtd está referenciado en el documento XML. Si se deseara incluir todo en el mismo documento XML, entonces el código sería el siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE bdSMS
[
<!ELEMENT bdSMS (SMS*)>
<!ELEMENT SMS (telefono, fecha, hora, mensaje)>
<!ELEMENT telefono (#PCDATA)>
<!ELEMENT fecha (#PCDATA)>
<!ELEMENT hora (#PCDATA)>
<!ELEMENT mensaje (#PCDATA)>
]>
<bdSMS>
<SMS>
<telefono>1234569</telefono>
<fecha>20/03/2012 </fecha>
<hora>10:35</hora>
<mensaje>Hola, ¿cómo estas?</mensaje>
</SMS>
<SMS>
<telefono>654321</telefono>
<fecha>22/03/2012 </fecha>
<hora> 15:05</hora>
<mensaje>estoy muy bien</mensaje>
</SMS>
<SMS>
<telefono>111111</telefono>
<fecha> 25/03/20 12 </fecha>
<hora>09:10</hora>
<mensaje>Hoy hace mucho frio</mensaje>
```

```
</SMS>
</bdSMS>
```

Al ver el ejemplo de la base de datos de SMS, podemos observar una serie de elementos definidos en una DTD:

- **Elemento.** Es el bloque principal con el que se construyen los documentos XML. Los elementos se declaran de dos maneras: <!ELEMENT nombre\_del\_elemento categoría> o <!ELEMENT nombre\_del\_elemento (nodos\_hijos); en este segundo caso, pueden definir el nombre\_del\_elemento como el nodo padre del que cuelga un conjunto de nodos\_hijos (separados por comas).

Si hubiera algún elemento que debiera quedarse vacío, utilizaríamos la palabra EMPTY para indicarlo.

- **Atributo.** Es el parámetro que sirve para añadir más información a un elemento. Los atributos se declaran del siguiente modo: <!ATTLIST nombre\_elemento nombre\_atributo tipo\_atributo valor-pordefecto>

**-nombre\_elemento.** Es el elemento al que se le quiere añadir el atributo.

**-nombre\_atributo.** Es el nombre del atributo que se quiere añadir.

**-tipo\_atributo.** Existen muchos tipos de atributos, entre los que cabe destacar:

- **CDATA.** Es un texto. Puede contener cualquier carácter.
- **ID.** Es un parámetro que permite identificar al elemento de manera única en todo un documento XML.
- **IDREF.** Es un identificador de otro elemento del propio documento XML.
- **IDREFS.** Es una lista de identificadores de otros elementos generalmente definida por el usuario.
- **NMTOKEN.** Es un texto que sólo podrá tener letras, dígitos, guión “-”, subrayado “\_”, punto “.”, y dos puntos “..”. Es lo que se conoce como nombres válidos XML.
- **NMTOKENS.** Es una lista de nombres XML válidos. Es como un NMTOKEN pero, a diferencia de éste, se incluyen los espacios en blanco “ ”, tabuladores o retornos de carro.
- **ENTITY.** Es una entidad que ofrece una entrada abreviada que puede hallarse en el documento XML.
- **ENTITIES.** Es una lista de entidades.
- **NOTATION.** Se utiliza para especificar datos no-XML.

## Recuerda

Puesto que XML es un sistema para definir lenguajes, no puede tener una sola DTD (al contrario que HTML). En lugar de eso, quien necesite usar XML para el intercambio de datos debe definir su propia DTD.

Los atributos podrán ser declarados **obligatorios** #REQUIRED u **optativos** #IMPLIED; o fijos #FIXED o por defecto #DEFAULT.

- **Entidad.** En XML existen algunos caracteres que tienen un significado especial. La aparición de dichos caracteres como datos almacenables hace que puedan confundirse con entidades propias de un documento XML. Se consideran caracteres especiales las expresiones incluidas en la tabla de la Figura 4.5.

**Figura 4.5**  
Ejemplo de interpretación de caracteres especiales en un documento XML.

CARACTERES ESPECIALES	INTERPRETACIÓN
&nbsps	Espacio en blanco
&gt	>
&lt	<
&quot	&

Los caracteres especiales pueden usarse dentro del propio documento XML y, tras parsear el documento, tomarán el valor indicado en la columna *interpretación* de la tabla anterior.

- **PCData (Parsed Character Data).** Indica que, entre la etiqueta de apertura y cierre del elemento al que se aplica, se almacenarán caracteres como texto y se analizarán por un *parser*.
- **CDATA (Character Data).** Este elemento es prácticamente igual que PCData, aunque el contenido, entre sus etiquetas de apertura y cierre, no será analizado por el *parser* de análisis de entidades y elementos XML.

### 4.3.2. Esquema XML

Hasta ahora hemos hablado de cómo generar un lenguaje XML con la definición de las reglas que lo componen (DTD). Sin embargo, otra manera de formalizar esas reglas es mediante el lenguaje **XML Schema** (*Schema Definition*), también llamado XSD.

Un esquema XML define qué elementos y atributos pueden aparecer en un documento XML. Define, además, qué elementos son compuestos; es decir, indica qué elementos hijos deben aparecer y en qué orden, qué elementos pueden ser vacíos y cuáles pueden incluir texto asociado. Define también la obligatoriedad y la optatividad.

En el siguiente ejemplo definiremos el esquema del documento XML utilizando de nuevo la base de datos SMS que hemos empleado como ejemplo en la definición de las DTD:

```
<?xml version="1.0" encoding="ISO-8859-1 "?>
<bdSMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation='http://www.gmrv.es bdSMSSchema.xsd">
<SMS>
<telefono>123456</telefono>
<fecha>20/03/2012 </fecha>
<hora>10:35</hora>
<mensaje>Hola, ¿cómo estas?</mensaje>
</SMS>
<SMS>
<telefono>654321</telefono>
<fecha>22/03/2012 </fecha>
<hora> 15:05</hora>
<mensaje>estoy muy bien</mensaje>
</SMS>
<SMS>
<telefono>111111</telefono>
<fecha> 25/03/20 12 </fecha>
<hora>09:10</hora>
<mensaje>Hoy hace mucho frio</mensaje>
</SMS>
</bdSMS>
```

Si el esquema está en un fichero local, en lugar de utilizar xsi:schemaLocation e indicar una ubicación en la red, podemos emplear esta otra forma:

```
<?xml version="1.0" encoding="ISO-8859-1 "?>
<bdSMS xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:noNamespaceSchemaLocation="bdSMSSchema.xsd">
```

En el primer ejemplo, teníamos una DTD que permitía validar el documento. Ahora definiremos el esquema en los mismos términos que el anterior (fichero bdSMSSchema.xsd):

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema' '
  version="0.1" xml:lang="es">
```

```
<xs:element name=>>bdSMS<>>
<xs:complexType>
<xs:sequence>
<xs:element name=>>SMS<> maxOccurs=>>unbounded<>>
<xs:complexType>
<xs:sequence>
<xs:element name=>>telefono<> type=>>xs:string<>/>
<xs:element name=>>fecha<> type=>>xs:string<>/>
<xs:element name=>>hora<> type=>>xs:string<>/>
<xs:element name=>>mensaje<> type=>>xs:string<>/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Veamos a continuación los principales elementos de los que se compone un esquema XML:

- **Elemento raíz.** El elemento principal de un esquema, que puede componerse de muchos elementos enlazados, se conoce como elemento raíz y se trata de un elemento obligatorio:

```
<?xml version="1.0" ?>
<xs:schema>
...
</xs:schema>
```

El elemento raíz puede contener atributos. Entre los atributos más importantes destaca el xmlns, que indica cuál es el espacio de nombres en el que se basa para saber qué elementos y tipos de datos son los soportados en el esquema:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
...
</xs:schema>
```

- **Elementos simples.** Son aquellos que el usuario puede utilizar para definir su propio lenguaje XML. Pueden contener información y no tienen otros elementos hijos asociados a ellos ni atributos en su interior.

Para definir un nuevo elemento simple, utilizaremos la siguiente sintaxis:

```
<xs:element name="nombre_elemento" type="tipo_elemento">
```

Vemos, pues, que el nombre puede ser aquel que desee el usuario, siempre y cuando no esté repetido. Sin embargo, el tipo deberá ser forzosamente uno de los siguientes:

<b>-xs:string:</b>	<xs:element name="nombre" type="xs:string">
<b>-xs:date:</b>	<xs:element name="fechaNacimiento" type="xs:date">
<b>-xs:time:</b>	<xs:element name="hora" type="xs:time">
<b>-xs:dateTime:</b>	<xs:element name="fecha" type="xs:date">
<b>-xs:decimal:</b>	<xs:element name="precio" type="xs:decimal">
<b>-xs:integer:</b>	<xs:element name="vueltas" type="xs:integer">
<b>-xs:boolean:</b>	<xs:element name="pagado" type="xs:boolean">
<b>-xs:hexBinary:</b>	<xs:element name="imagen" type="xs:hexBinary">

- **Atributos.** Son complementos de información que pueden asignarse a un elemento previamente declarado. Veamos a continuación cuál es la sintaxis de un atributo:

```
<xs:attribute name="nombre_atributo" type="tipo_atributo">
```

- **Restricciones.** En algunos momentos es importante establecer el rango de valores en los que un elemento puede moverse. Las restricciones no son sólo útiles para los elementos sino que pueden aplicarse también a los atributos. Existe, además, la posibilidad de establecer restricciones sobre la longitud del valor contenido por un elemento.

Las principales restricciones que cabe destacar son:

- xs:length:** Establece una longitud fija.
- xs:minLength:** Establece un mínimo en la longitud.
- xs:maxLength:** Establece un máximo en la longitud.
- xs:maxExclusive:** Es el máximo valor. Serán válidos todos los valores que sean menores que el indicado.
- xs:minExclusive:** Es el mínimo valor. Serán válidos todos los valores que sean mayores que el indicado.
- xs:totalDigits:** Indica el número exacto de cifras permitidas en el elemento o atributo. Debe ser mayor que cero.
- xs:fractionDigits:** Indica el número máximo de decimales que se permite en el valor del elemento. Debe ser mayor o igual a cero.

## Recuerda

**Si no se utiliza DTD, el documento debe comenzar con un Declaración de Documento único (SDD, Standalone Document Declaration) que indique: <?XML version="1.0" standalone="yes"?>**

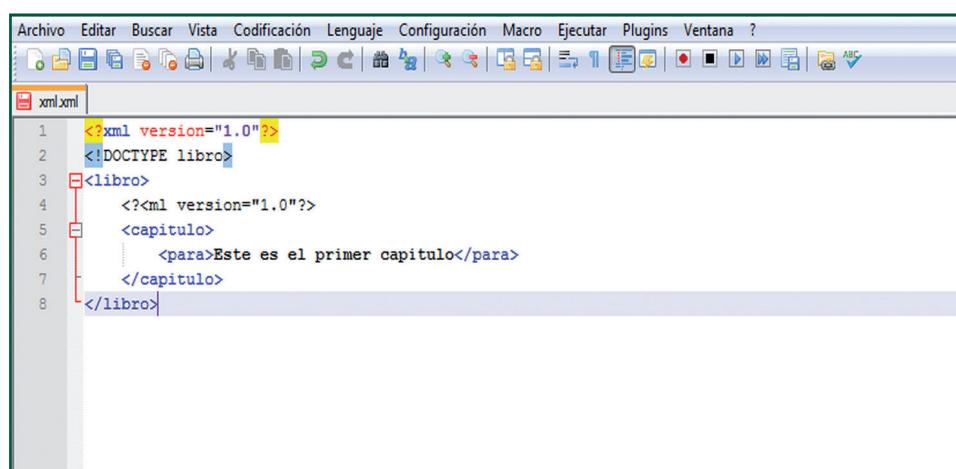
- **Elementos complejos.** Son aquellos que se componen de otros elementos y que pueden contener atributos propios. Son elementos complejos los que contienen otros elementos en su interior, los que contienen atributos en su interior, y los que contienen otros elementos y atributos en su interior.
- **Secuencia de elementos.** Permiten marcar cuántas veces deben aparecer determinados elementos. Un indicador permite determinar el orden en el que se establecen los elementos hijos, la frecuencia con la que éstos aparecen y el grupo al que pertenecen.

Veamos a continuación algunos ejemplos de secuencias de elementos:

- <xs:all>**. Especifica que todos los elementos hijos pueden aparecer en cualquier orden determinado, siempre y cuando sólo aparezcan una única vez.
- <xs:Choice>**. Especifica que, de entre los elementos hijos, sólo pueden aparecer uno u otro.
- <xs:sequence>**. Permite indicar el orden específico en el que deben aparecer los elementos hijos.
- <xs:maxOccurs>**. Indica el máximo número de veces que un elemento hijo puede aparecer.
- <xs:minOccurs>**. Indica el mínimo número de veces que un elemento hijo puede aparecer.

## 4.4 Asociación con documentos XML

Una vez creada una descripción de tipo de documento, el siguiente paso será **asociarla** al documento XML al que se aplicará (Figura 4.6).



```
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Macro Ejecutar Plugins Ventana ?
xml.xml
1  <?xml version="1.0"?>
2  <!DOCTYPE libro>
3  <libro>
4      <?xml version="1.0"?>
5      <capítulo>
6          |   <para>Este es el primer capítulo</para>
7      </capítulo>
8  </libro>
```

**Figura 4.6**

Ejemplo del principio de un documento XML en el que se especifica el DOCTYPE.

Documentar es una manera de explicar lo que se ha hecho, cómo se ha hecho y cómo funciona, pensando en la posibilidad de que le corresponda a otra persona la tarea de continuar o modificar nuestro trabajo. Alguien que lea nuestra declaración de descripción entenderá cómo hemos trabajado la estructura de nuestro documento. Un procesador que lea la DTD asociada a nuestro documento nos indicará si éste es válido o no, es decir, si el documento se adapta a la descripción o si, por el contrario, existen errores.

## Recuerda

**Los documentos XML Schema usualmente se guardan con extensión .xsd, por eso son conocidos como XSD.**

Básicamente, un documento XML está formado por elementos y sus atributos. Una descripción debe ser capaz de definir todos los elementos de un documento, los atributos de cada uno de los elementos, y la relación entre ellos. El modo de asociar una descripción a un documento será mediante la **Declaración de Tipo de Documento (DOCTYPE)**. Dicha declaración deberá escribirse después de la declaración de XML y antes de la aparición de cualquier elemento del documento.

La Declaración de Tipo de Documento (*DOCTYPE*) debe ir seguida del nombre del elemento documento (el elemento raíz del documento) y, a continuación las definiciones. El nombre escrito en esta declaración debe ser el mismo que el del elemento documento, de otro modo el procesador dará un error.

Al igual que la descripción, el *DOCTYPE* puede estar definido dentro del mismo documento XML o en un archivo externo. La sintaxis para una declaración de documento incluida dentro del archivo sería la siguiente:

```
<!DOCTYPE nombre_del_elemento_documento [ ...definiciones... ]>
```

La etiqueta DOCTYPE, si no hay declaración XML, irá siempre en la primera línea, o bien, justo después.

Por lo que respecta a la sintaxis para indicar una declaración de tipo de documento en un documento externo, sería la siguiente:

```
<!DOCTYPE nombre_del_elemento_documento SYSTEM "identificador_de_sistema">
<!DOCTYPE nombre_del_elemento_documento PUBLIC "identificador_publico" "identificador_de_sistema">
```

Existe también la posibilidad de que se den ambos casos, es decir, que una parte de las declaraciones se encuentren en un archivo externo y otras definidas dentro del cuerpo.

## 4.5 Validación

Un archivo de datos se considera un documento XML si está “bien formado” de acuerdo con la especificación XML; si cumple, además, otra serie de condiciones, como pueden ser, por ejemplo, las impuestas por su descripción, se considerará un documento XML válido.

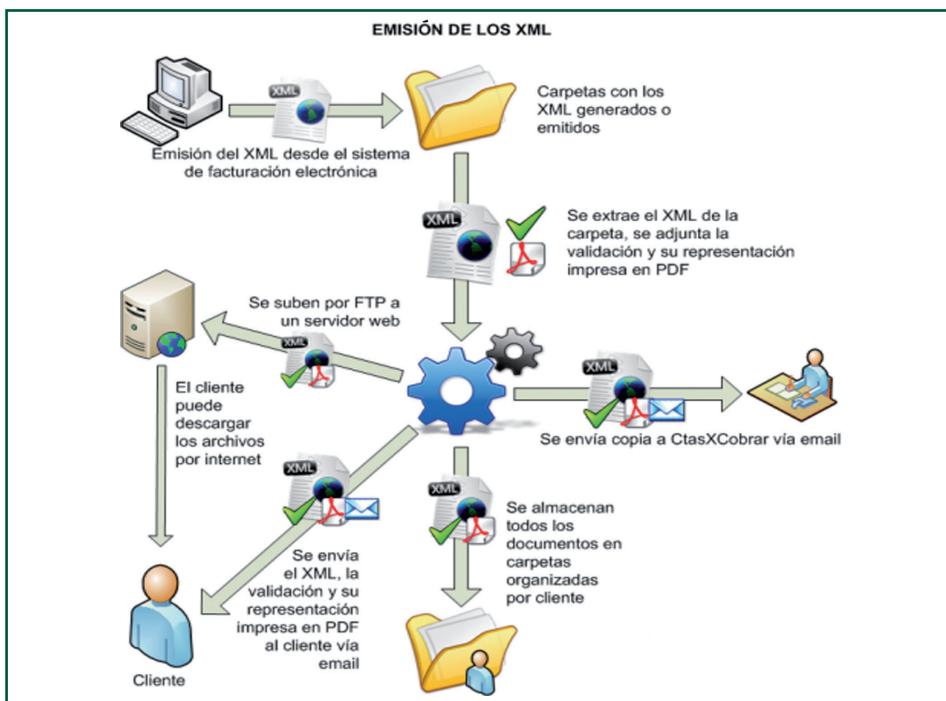
Un documento XML se considera bien formado si al tomarlo como un todo cumple la especificación de documento; es decir, si se compone de un prólogo, de un elemento principal (que puede incluir otros elementos) y, si se da el caso, terminar con comentarios, instrucciones de procesamiento o espacios en blanco.

La **validación XML** es la comprobación de que un documento editado en lenguaje XML está bien formado y se ajusta a una estructura definida. Un documento bien formado sigue las reglas básicas de XML establecidas para el diseño de documentos. Un documento válido respeta, además, las normas dictadas por su descripción o por el esquema utilizado.

Si un documento XML tiene asociada una descripción, el procesador deberá leer y validar ese documento con respecto a dicha descripción, es decir, chequear su sintaxis, su vocabulario, que los valores que aparezcan en él sean valores aceptados por su descripción, etc. Un procesador valida un documento a medida que lo lee, y si encuentra un error, detiene el proceso y avisa que ha detectado un fallo de validez. Un documento XML, conforme a las especificaciones sintácticas que describe el estándar XML, se dice que es un **documento bien formado**. Independientemente, un documento XML conforme con su documento de descripción asociado, se dice que es un **documento válido**.

Los documentos XML deben basarse en la sintaxis definida en la especificación XML para ser correctos (documentos bien formados). Esta sintaxis impone sus propias reglas, como es el caso de la coincidencia de mayúsculas y minúsculas en los nombres de etiqueta, comillas obligatorias para los valores de atributo, etc. Sin embargo, para tener un control más preciso sobre el contenido de los documentos, es necesario un proceso de análisis más exhaustivo, y que se lleva a cabo mediante la descripción.

La validación es la parte más importante de este análisis, ya que determina si un documento creado se ajusta a las restricciones descritas en el esquema utilizado para su construcción. Controlar el diseño de documentos a través de esquemas aumenta su grado de fiabilidad, consistencia y precisión, y facilita su intercambio entre aplicaciones y usuarios. Cuando creamos documentos XML válidos aumentamos su funcionalidad y utilidad (Figura 4.7).



## Recuerda

El World Wide Web Consortium (W3C) ofrece los estándares de DTD o XML Schema, así como validadores para ambas opciones.

**Figura 4.7**

Ejemplo de validación de flujo de trabajo de un documento XML.

La validación se encarga de verificar la corrección de los datos. Si bien validar un esquema o descripción no garantiza al cien por cien que los datos sean correctos, sí, en cambio, nos permite detectar formatos nulos o valores fuera de rango y, por lo tanto, incorrectos. Con la validación también se asegura la integridad de los datos, pues al validar se comprueba que toda la información obligatoria se halle en el documento. Y por último, al validar un documento XML obtenemos el entendimiento compartido de los datos, pues, a través de la validación se comprueba que el emisor y receptor perciban el documento de la misma manera, que lo interpreten igual.

## 4.6 Herramientas de creación y validación

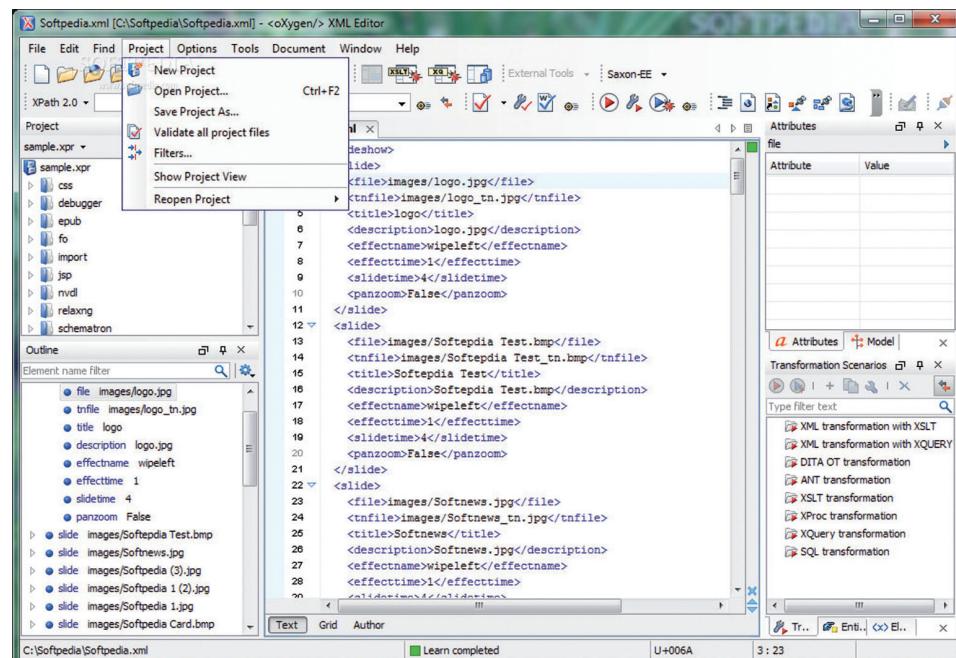
La creación manual de documentos XML, e incluso su manipulación, puede producir todo tipo de errores, desde tipográficos y sintácticos hasta de contenido. Existen, sin embargo, editores de XML que facilitan la tarea de crear documentos válidos y bien formados, puesto que son capaces de advertir los errores básicos cometidos e incluso de escribir automáticamente la sintaxis, más sencilla, necesaria.

Para poder editar un documento válido, existen editores XML capaces de leer la descripción del documento e interpretarla para configurar luego una lista desplegable con los elementos disponibles enumerados en la descripción, con el fin de utilizar correctamente los nombres de los elementos y evitar así la inclusión de al-

gún elemento no definido en el esquema. También pueden advertir del olvido de una etiqueta obligatoria e, incluso, impedir que se produzca ese tipo de descuidos, no dando por finalizado el documento si existen errores de este tipo.

Veamos a continuación cuáles son los editores más utilizados (Figura 4.8):

- XML Pro de Vervet Logic (*open source*).
- XMLSpy de Altova.
- Liquid XML Studio de Liquid Technologies.
- <oXygen/> XML Editor.
- Turbo XML de TIBCO (Plataforma de desarrollo integrado de XML).
- XML Notepad de Microsoft.
- XMLwriter de Wattle Software.



**Figura 4.8**

Ejemplo del editor de XML oXygen.

Los documentos XML se procesan a través de **analizadores**: aplicaciones que leen el documento, lo interpretan y generan una salida basada en sus contenidos y en la marca utilizada para su descripción. El resultado se muestra en un periférico de visualización, como una ventana de navegación o una impresora. Los procesadores hacen posible la presentación y distribución de documentos XML.

Estos analizadores pueden tener o no la capacidad también de validar los documentos, ya que comprueban que los documentos XML estén bien formados, aunque sólo los **analizadores validantes** pueden efectuar la validación. Los analiza-

dores no validantes, sin embargo, procesan a mayor velocidad porque no tienen que comprobar tantos detalles como los validantes.

Prácticamente todos los procesadores de XML son capaces de validar con un esquema basado en la descripción. Si el documento viola alguna de las reglas de su descripción, el analizador advertirá del error y el proceso del documento se detendrá. Un buen procesador validante no sólo informará de la presencia de un error, sino que elaborará un mensaje y especificará cuál ha sido su causa. El procesador Xerces de Apache, además de ser el *parser* más completo de XML, incluye soporte de XML Schema.

Por lo general, se suele utilizar un analizador validante para comprobar la corrección de los documentos, así como uno no validante para su presentación. Una vez sabemos que un documento es válido, no hay ningún motivo para comprobar su validez cada vez que se procese y, de ese modo, optimizaremos el tiempo del proceso.

Veamos a continuación algunas de las herramientas para validar archivos online existentes:

- Validador de documentos XML usando XML Schema:  
<http://stone.com> o [j.com/validador\\_xml.php](http://j.com/validador_xml.php)
- Validación XML
- Markup Validation Service de W3C
- Validador de XML Schema REC (20010502)
- XML DOM Validate

## Para saber más

**Si bien lo recomendable es utilizar un validador integrado en el editor XML, resulta interesante, en determinadas circunstancias, disponer de un programa validador específico.**  
**Por ejemplo, en el caso de que tengamos que validar XML directamente desde Internet, podemos acceder a sitios concretos que realizarán dicha validación en línea; es el caso de la web <http://validator.w3.org>.**

## 4.7 Documentación de especificaciones

Como hemos visto, el lenguaje XML nace como un formato estandarizado para el intercambio de información entre sistemas. Se desarrolla a partir del lenguaje SGML y, si inicialmente se definía la estructura de los documentos mediante descripción, actualmente se opta por definir la estructura mediante un esquema XML.

El lenguaje XML abarca un amplio conjunto de especificaciones. Veamos cuáles son:

- **XSL (eXtensible Stylesheet Language).** Es un lenguaje que se utiliza para convertir documentos XML a otros documentos XML; puede convertir un documento XML que obedezca a una descripción en otro que obedezca a otra diferente; un documento XML bien formado a otro que siga una descripción; o, lo más

habitual, convertirlo a “formatos finales”, tales como XHTML o WML (usado en los móviles WAP).

El lenguaje XSL abarca los siguientes sublenguajes (Figura 4.9):

-**XSLT (eXtensible Stylesheet Language Transformations)**. Transforma un documento XML en otro XHTML o XML.

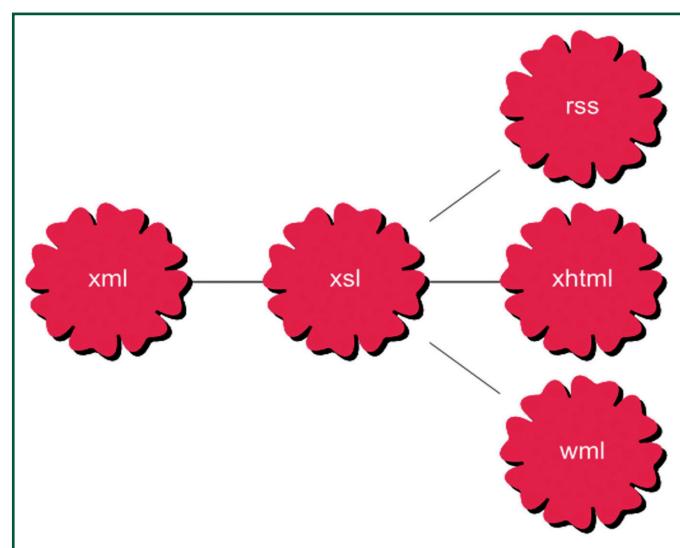
-**XSL-FO (eXtensible Stylesheet Language Formatting Objects)**. Formatea un documento XML.

-**Xpath (XML Path language)**. Navega entre partes de un documento XML.

- **XLink (XML Linking Language)**. Es una recomendación del W3C. Este lenguaje permite crear enlaces e hipervínculos entre documentos XML. Con XLink cualquier elemento XML puede comportarse como un enlace. Los enlaces pueden definirse en un documento aparte o dentro del mismo; cabe distinguir entre enlaces simples y enlaces complejos o multidireccionales.

- **XQuery (XML Query)**. Es una recomendación del W3C. Es el lenguaje definido para realizar consultas sobre documentos XML, el equivalente al SQL de las bases de datos relacionales. Se utiliza para extraer información con el fin de utilizarla en un *web service*, para generar resúmenes, transformar XML data a XHTML, buscar documentos con información determinada, etc.

Debido al uso cada vez más extenso del lenguaje XML, surgen nuevas necesidades tales como la firma digital de documentos XML; para, ello se han desarrollado dos especificaciones: XMLDSig y XMLAdES.



**Figura 4.9**

Ejemplo de múltiples formatos de salida con la aplicación XSL.

## Resumen

Los documentos XML tienen una estructura lógica y una estructura física. La estructura lógica está compuesta de un conjunto de declaraciones, elementos, comentarios, etc., que se incluyen en el documento mediante unas marcas determinadas. La estructura física consta de una serie de unidades, llamadas entidades, que indican los datos que contendrá el documento.

Una definición de tipo de documento define la estructura y la sintaxis de un documento XML. Su función es indicar la estructura de datos del documento XML, del tal forma que nos permita su posterior validación en base a su definición.

Algunas de las tecnologías más extendidas para la definición de tipos de documentos son los DTD (*Document Type Definition*), los esquemas XML (*XML Schema*), RELAX y Schematron.

La definición de tipo de documento (DTD) es una descripción de estructura y sintaxis de un documento XML. Una DTD contiene un elemento raíz y una descripción de todos los elementos que intervienen en dicho elemento raíz. Los esquemas XML se utilizan para lo mismo que los DTD, pero a diferencia de éstos también permiten validar que un elemento cumpla una determinada característica.

El documento de definición, independientemente de la tecnología, nos permite crear nuestro propio lenguaje de marcas y debe contener qué elementos pueden formar parte del documento (*Element Type Declarations*), qué atributos pueden utilizarse (*Attribute List Declarations*), qué objetos pueden ser referenciados (*Entity Declarations*) y los tipos de ficheros binarios que pueden enlazarse (*Notation Declaration*).

Para asociar una descripción a un documento utilizaremos la Declaración de Tipo de Documento DOCTYPE. Al igual que la descripción, el DOCTYPE puede estar definido dentro del mismo documento XML o en un archivo externo.

Un documento XML se considera “bien formado” si, al tomarlo como un todo, el documento XML cumple la especificación de documento. La validación XML es la comprobación de que un documento editado en lenguaje XML está bien formado y se ajusta a una estructura definida.

Existen editores de XML que facilitan la tarea de crear documentos válidos y bien formados, ya que pueden advertir de los errores básicos cometidos e incluso escribir automáticamente la sintaxis más sencilla necesaria.

## Ejercicios de autocomprobación

**Indica si las siguientes afirmaciones son verdaderas (V) o falsas (F):**

1. El lenguaje de marcas XML puede crear sus propias etiquetas y esto es una ventaja que puede convertirse en un inconveniente en trabajo en equipo.
2. Los documentos XML tienen una estructura lógica y una estructura física. La estructura lógica consta de un conjunto de declaraciones, elementos, comentarios, etc., que están excluidos del documento mediante unas determinadas marcas. La estructura física consta de una serie de unidades, llamadas entidades, que indican los datos que contendrá el documento. Ambas estructuras deberán encajarse correctamente.
3. La tecnología RELAX NG sirve para la definición y representación de tipos de documentos XML. RELAX NG utiliza la sintaxis XML. Está basada en la gramática y, con respecto a la utilización de los esquemas XML, es muy intuitiva y más fácil de entender, de aprender y de utilizar, lo que hoy en día hace comprensible su popularidad.
4. XML es un sistema para definir lenguajes y puede tener una sola descripción, al contrario que HTML.
5. Los documentos XML que se ajustan a su descripción se denominan válidos. El concepto de validez tiene que ver con el de estar bien formado. Un documento bien formado es aquel que, simplemente, respeta la estructura y sintaxis definidas por la especificación de XML.
6. La validación XML es la comprobación de que un documento editado en lenguaje XML está bien formado y se ajusta a una estructura definida. Un documento bien formado sigue las reglas básicas de XML establecidas para el diseño de documentos. Un documento válido respeta, además, las normas dictadas por su descripción o por el esquema utilizado.
7. Para asociar una descripción a un documento utilizaremos la Declaración de Tipo de Documento DOCTYPE. Al igual que la descripción, el DOCTYPE no puede estar definido dentro del mismo documento XML pero en un archivo externo.

**Completa las siguientes afirmaciones:**

8. Una definición de tipo de documento es una descripción de la estructura y la \_\_\_\_\_ de un documento \_\_\_\_\_. Su función es la de indicar la descripción de la \_\_\_\_\_ de datos del tal forma que nos permita el uso de una estructura común que mantenga la consistencia entre todos los documentos que utilicen la misma definición de tipo de documento.
9. Un esquema XML define qué elementos y \_\_\_\_\_ pueden aparecer en un documento XML. Define, además, qué elementos son \_\_\_\_\_; es decir, indica qué elementos hijos deben aparecer y en qué \_\_\_\_\_, qué elementos pueden ser \_\_\_\_\_ y cuáles pueden incluir texto asociado. Define también la \_\_\_\_\_ y la optionalidad.
10. Existen \_\_\_\_\_ de XML que facilitan la tarea de crear documentos válidos y bien formados, ya que pueden advertir de los \_\_\_\_\_ básicos cometidos e incluso escribir automáticamente la \_\_\_\_\_ más sencilla necesaria.

Las soluciones a los ejercicios de autocomprobación se encuentran al final de esta Unidad Formativa. En caso de que no los hayas contestado correctamente, repasa la parte de la lección correspondiente.

## Soluciones a los ejercicios de autocomprobación

### Unidad 3

1. V
2. V
3. F. Para la creación de canales de contenido puede utilizarse el lenguaje de programación tipo “php” combinado con lenguajes de programación de entorno servidor que permita generar los feedsrss.
4. V
5. V
6. V
7. F. Un lector o agregador de feeds es una aplicación local, o basada en web, que interpreta los archivos RSS y visualiza su contenido.
8. redifusión Web, XML, archivos, feeds, canales
9. agregadores, programas, suscritos, audio.
10. combinación, planets, página web, información.

### Unidad 4

1. V
2. F. Los documentos XML tienen una estructura lógica y una estructura física. La estructura lógica consta de un conjunto de declaraciones, elementos, comentarios, etc., que están incluidos del documento mediante unas determinadas marcas. La estructura física consta de una serie de unidades, llamadas entidades, que indican los datos que contendrá el documento. Ambas estructuras deberán encajarse correctamente.
3. V
4. F. XML es un sistema para definir lenguajes y no puede tener una sola descripción, al contrario que HTML.
5. F. El concepto de validez no tiene nada que ver con el de estar bien formado.
6. V
7. F. Al igual que la descripción, el DOCTYPE puede estar definido dentro del mismo documento XML o en un archivo externo.
8. sintaxis, XML, estructura.
9. atributos, compuestos, orden, vacíos, obligatoriedad.
10. editores, errores, sintaxis.

# Índice

## MÓDULO: LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

### UNIDAD FORMATIVA 2

<b>3. Aplicación de los lenguajes de marcas a la sindicación de contenidos.....</b>	64
3.1 Ventajas .....	65
3.2 Ámbitos de aplicación .....	67
3.3 Estructura de los canales de contenidos.....	70
3.4 Tecnologías de creación de canales de contenidos .....	72
3.5 Validación.....	73
3.6 Utilización de herramientas.....	74
3.7 Directorios de canales de contenidos.....	76
3.8 Agregación.....	78
Resumen.....	80
Ejercicios de autocomprobación.....	81
<b>4. Definición de esquemas y vocabularios en XML.....</b>	83
4.1. Definición de la estructura y sintaxis de documentos XML.....	83
4.2. Utilización de métodos de definición de documentos XML.....	84
4.2.1 DTD ( <i>Document Type Definition</i> ) .....	85
4.2.2 Esquemas XML.....	86
4.2.3 Relax NG.....	88
4.2.4 Schematron.....	89
4.3 Creación de descripciones.....	89
4.3.1 DTD ( <i>Document Type Definition</i> ) .....	90
4.3.2 Esquema XML.....	94

4.4 Asociación con documentos XML .....	98
4.5 Validación.....	100
4.6 Herramientas de creación y validación .....	101
4.7 Documentación de especificaciones .....	103
Resumen.....	105
Ejercicios de autocomprobación.....	106
Solucioens a los ejercicios de autocomprobación.....	108

# UNIDAD FORMATIVA 3

- Conversión y adaptación de documentos XML
- Almacenamiento de información
- Sistemas de gestión empresarial

## 5. CONVERSIÓN Y ADAPTACIÓN DE DOCUMENTOS XML

El lenguaje **XML** permite separar, de manera efectiva, los datos que se almacenan: por un lado, la estructura o semántica en la que se organizan los datos, y, por otro lado, la presentación de los mismos. Aunque, con un simple editor de texto, podemos visualizar un fichero XML, éste no es el mejor modo para presentar los datos que están almacenados en su interior. Para ello, deberemos utilizar una herramienta que nos permita convertir y transformar los datos en el formato que deseemos.

En esta unidad aprenderemos a convertir y adaptar los documentos XML, así como a aplicarlos y presentarlos con herramientas diseñadas para este propósito. Estudiaremos las **hojas de estilo XSL**, que, como las hojas de estilo en cascada CSS de los documentos HTML, ejercen un absoluto control sobre los datos y permiten establecer criterios como: qué datos queremos ver y en qué orden deseamos visualizarlos, así como también establecer filtros sobre ellos y definir formatos de salida para su representación. Se trata, por lo tanto, de una herramienta de procesado muy potente que conoceremos en profundidad.

Como hemos dicho, un documento XML no está compuesto de datos, lo que, como tales, pueden utilizarse en múltiples aplicaciones informáticas y con objetivos bien diferentes. Por lo tanto, es posible que necesitemos convertirlo a un formato más comprensible para los usuarios, como puede ser: HTML, RTF, PDF, etc., mediante los lenguajes de hojas de estilo.

### 5.1 Técnicas de transformación de documentos XML

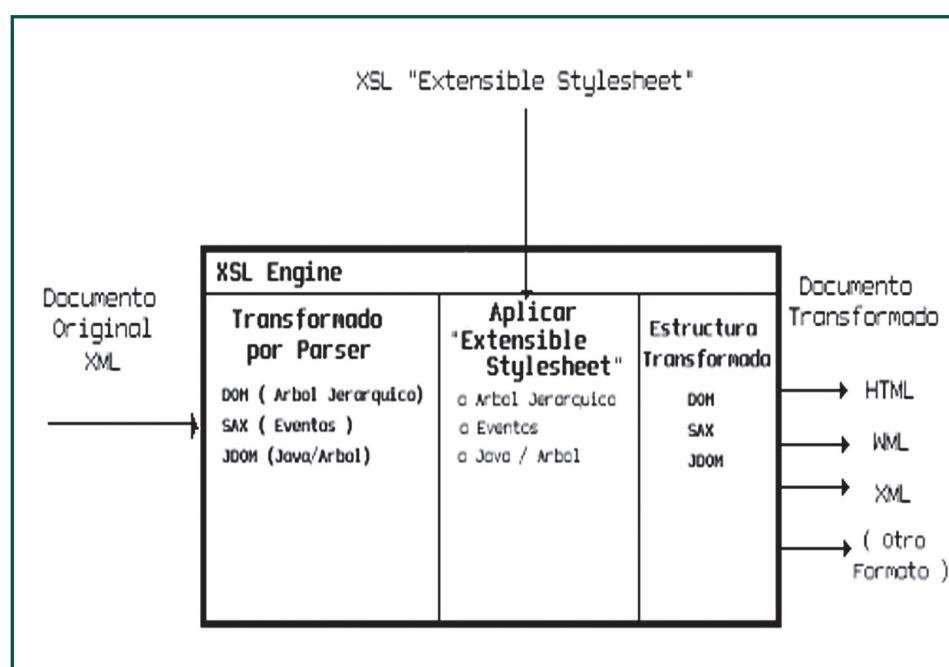
El **lenguaje de marcas XML** ha sido creado principalmente con la intención de poder compartir la información almacenada en los documentos XML. XML, por su gran flexibilidad, sencillez y amplia capacidad de adaptación, se ha convertido en el lenguaje por excelencia para intercambiar información. En realidad, cualquier documento XML es válido para compartir información, siempre y cuando la anidación de los marcadores sea correcta.

Para poder intercambiar, de manera sencilla, la información almacenada en un documento XML con múltiples aplicaciones, deberemos utilizar un lenguaje de transformación que nos permita cambiar el formato de salida del documento XML por el formato adecuado para la presentación de los datos, con el fin de conseguir que las aplicaciones interpreten de forma correcta el documento XML. El lenguaje elegido para especificar las transformaciones suele ser XSL (*eXtensible Stylesheet*

*Language*), el cual busca patrones dentro de un documento e indica cómo reestructurarlos para conseguir su correcta interpretación.

Veamos un ejemplo: una empresa emplea el siguiente formato para almacenar los nombres de los autores de libros: <autor><nombre>Arturo</nombre><apellido>Perez\_Reverte</apellido></autor>, mientras otra utiliza un formato más simple, en el que el nombre es almacenado como una sola palabra: <autor>Arturo Perez Reverte</autor>. Para poder mostrar toda la información unificada, la regla de transformación que permita pasar del primer formato al segundo será concatenar el nombre y apellido del autor, para generar su nombre con una sola palabra. XSL buscará la etiqueta <autor>, y después indicará que las palabras que aparecen dentro de <nombre> y <apellido> para este autor deberán estar concatenadas.

Como hemos dicho, para convertir un documento XML utilizaremos XSL porque es un lenguaje que nos permite definir un conjunto de reglas que, aplicadas sobre un documento XML, lo transforman en un resultado formateado más adecuado a nuestros intereses. Para ello, deberemos almacenar esas reglas dentro de un fichero, llamado **hoja de estilo XSL** (Figura 5.1).



**Figura 5.1**  
Esquema del proceso de transformación de un documento XML aplicándole XSL.

El lenguaje XSL se compone de tres sublenguajes: XSL, XSLT, y XPath; que, pese a ser tres especificaciones diferentes, se complementan y pueden utilizarse conjuntamente.

- **XSLT (*eXtensible Stylesheet Language Transformations*)**. Es el lenguaje para transformar documentos XML que permite convertir documentos XML de una sintaxis a otra. Una XSLT no es más que un fichero de texto, por lo que es necesario un motor XSL que, combinando el XML y la XSL correspondiente, nos proporcione el formato de salida. XSLT se utiliza para transformar un documento XML, de manera que éste pueda ser, por ejemplo, reconocido por un navegador como si fuera un archivo HTML.
- **XSL-FO (*eXtensible Stylesheet Language Formatting Objects*)**. Lenguaje de hojas extensibles de formateo de objetos que permite especificar el formato visual con el cual se quiere presentar un documento XML, y que se utiliza principalmente para generar documentos PDF.
- **XPath (*XML Path Language*)**. Lenguaje que no está basado en XML, y que sirve para acceder o buscar porciones de un documento XML y para navegar en documentos XML; podemos utilizarlo para hacer referencias a las diferentes partes que componen el documento.

No obstante, los lenguajes de hojas de estilo no son la única forma de convertir un documento XML a otro formato. Mediante los lenguajes Java, Perl, VisualBasic, Phyton o cualquier otro que incluyan *parsers* de XML, podremos crear aplicaciones que lean ficheros XML y los conviertan al formato deseado.

Las ventajas del lenguaje de hojas de estilos XSL con respecto a los lenguajes con *parsers* de XML radican en que éstos son ficheros de texto y, por lo tanto, para cambiar el aspecto de nuestro documento XML bastará con crear una nueva hoja de estilos, sin por ello tener que modificar la aplicación de los estilos que se procesan.

Algunos navegadores ya son capaces de interpretar estos lenguajes de hojas de estilo, por lo que son capaces de representar XML. Desde una única fuente de datos también podemos presentar el XML en múltiples formatos, así como personalizar fácilmente la presentación de la información con sólo cambiar la hoja de estilos. De este modo no sólo podremos personalizar su presentación, sino, además, filtrar el contenido, es decir, presentar sólo parte de éste o bien su totalidad, según, por ejemplo, el perfil de usuario que consulte la información.

El fichero que almacena la hoja de estilo XSL como fichero XML deberá estar bien formado. La estructura de los documentos XSL se compone de elementos y operadores.

Otra de las técnicas que pueden utilizarse para transformar un documento XML es DSSSL (*Document Style Semantics and Specification Language*), median-

## Recuerda

Para representar correctamente un documento XML, deberemos tener bien definidos el documento XML, crear una hoja de estilo XSL bien formada y vincular al documento XML la hoja de estilo XSL.

te la cual podemos transformar y formatear documentos SGML o XML. Con la aplicación DSSSL podemos transformar un documento SGML o XML en otro SGML o XML, o bien convertir un documento SGML o XML a un formato de presentación.

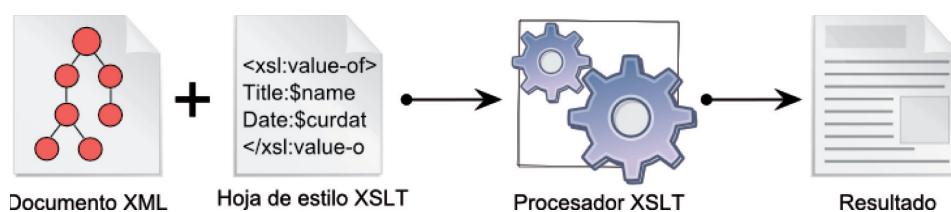
En el lenguaje HTML, las DSSSL son las hojas de estilo para los documentos HTML, con las que podríamos eliminar del código HTML gran parte de las etiquetas que le dan formato, con el fin de obtener ficheros HTML más pequeños y cambiar el aspecto de varias páginas HTML con sólo modificar las CSS que llevan asociadas. En la actualidad se ha extendido su uso a documentos XML, por lo que es posible representar un documento XML asociándole una CSS. De esta manera, un navegador ya no sólo es capaz de representar HTML sino también XML junto CSS.

## 5.2 Formatos de salida

Los **documentos XML** pueden modificarse con el fin de conseguir su presentación en el formato de salida necesario para cada situación y adaptarlos a su correcta utilización.

El lenguaje XSL no sólo permite hacer transformaciones en documentos XML, sino generar también cualquier tipo de documento, de documento XML (HTML, texto plano, programas en lenguajes de programación como Java, C++, etc.). Suelen utilizarse los *browsers*, navegadores como Firefox y Explorer, para poder desplegar documentos XML. Si tenemos en cuenta que XML es el mejor formato para almacenar información, es conveniente que nuestros documentos en la web estén en ese formato. Si un documento XML tiene que ser desplegado por un *browser*, se utilizará un conjunto de reglas XSLT para generar un documento HTML desde la fuente XML, el cual será utilizado por el *browser* al desplegar la información.

Las hojas de estilo XSLT realizan la transformación del documento mediante una o varias reglas de plantilla adjuntas al documento fuente que debe ser transformado. Estas reglas de plantilla alimentan a un procesador de XSLT, que realiza las transformaciones deseadas colocando el resultado en un archivo de salida o, como en el caso de una página web, directamente en el monitor del usuario (Figura 5.2).



**Figura 5.2**  
Proceso de transformación de un documento XML para modificar su presentación.

El tipo de programación con las hojas de estilo XSLT es completamente diferente a otros lenguajes a los que estamos acostumbrados, como C++ o PERL, pues se parece más a lenguajes como AWK u otros como ML o Schema.

Las características principales de la programación con las hojas de estilo XSLT son las siguientes:

- **No hay efectos secundarios.** Una instrucción debe hacer lo mismo, cualquiera que sea el camino de ejecución que lleve hasta ella. Esto significa que no vamos a poder trabajar con variables globales, ni con bucles en los que podamos incrementar el valor de una variable. Para solucionar estas carencias del sistema, nos apoyaremos en sistemas recursivos.
- **La programación está basada en reglas.** Esto significa que, cuando se cumple una regla en la entrada, se procesa un output en la salida. En este aspecto se parece al lenguaje AWK, o a cierto estilo de programación en PERL, pero no en el resto de funcionalidades.

Existen varias alternativas a la hora de realizar la transformación de un documento XML:

- **Procesador XALAN (procesador de XSLT de Apache).** Una herramienta escrita en Java que lo único que necesita para funcionar es una máquina virtual Java, tal como la de Microsoft, Sun o IBM. Xalan precisa un *parser* de XML para funcionar, como Xerces, aunque el fichero correspondiente viene incluido en la distribución.
- **Procesador MSXML.** Ejecutable que se limita a llamar a la biblioteca de transformación de Internet Explorer.

Se puede acudir a la biblioteca de transformación desde un programa.

- **Enlace entre el fichero XML y la hoja XSLT.** El fichero puede verse directamente en Internet Explorer o en otro navegador que soporte XSLT; el único inconveniente es que el fichero queda “ligado” a esta vista, por lo menos si se abre directamente.

### 5.3. Ámbitos de aplicación

El ámbito de aplicación de la transformación de documentos editados en lenguaje de marcas XML es muy amplio. Entre los diferentes ámbitos de aplicación en tecnologías, podemos destacar el XHTML, que es utilizado principalmente en la

creación de las páginas web. Algunas otras aplicaciones de la transformación del XML pueden ofrecer mecanismos más versátiles para mostrar datos. También se pueden conseguir buscadores inteligentes, debido a que la información en los documentos XML está etiquetada por su significado de forma precisa y, por lo tanto, podemos localizarla de forma mucho más clara que en documentos editados en HTML.

## Recuerda

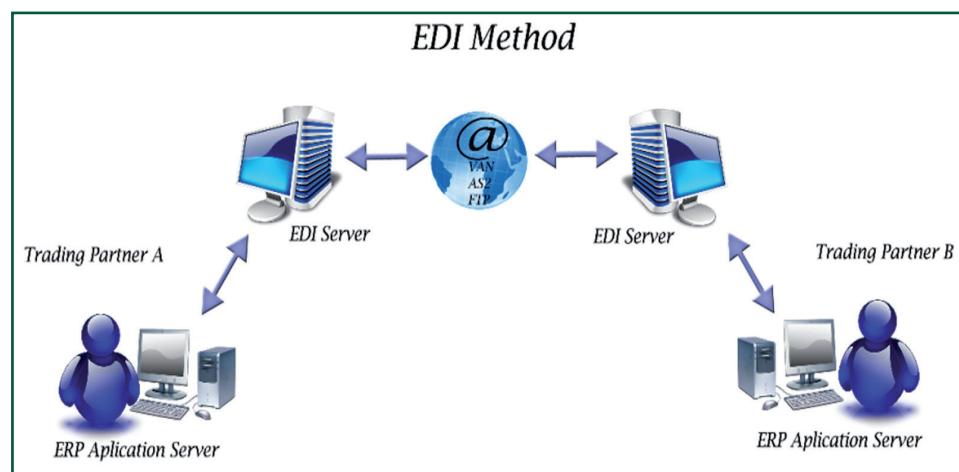
RSS también es un sublenguaje utilizado en la sindicación de contenidos.

Otro ámbito de aplicación de la transformación de los documentos XML es el intercambio de información entre sistemas heterogéneos; como, por ejemplo, el intercambio realizado entre diferentes sistemas de directorio para montar un metadirectorio.

Debido a las ventajas presentadas por el XML, y dado que los autores y proveedores pueden diseñar sus propios tipos de documentos, su uso está cada vez más extendido. A continuación, veamos los siguientes sublenguajes aplicados a diferentes tecnologías:

- **CDF (Channel Definition Format).** Describe un canal, una porción de un sitio web que hemos bajado a nuestro disco duro y se actualiza periódicamente conforme cambia la información. Un archivo CDF concreto contiene datos que especifican una página web inicial y la frecuencia con que se actualiza. Son los canales o *feeds* creados por Microsoft en el explorador con tecnología *push*.
- **RDF (Resource Description Framework).** Esquema de descripción de recursos. Se trata de una de las aplicaciones más importantes que permitirá describir los datos de cada documento, así como definir las relaciones que existen entre los datos XML.
- **OSD (Open Software Description Format).** Es un formato abierto de descripción de software que permite el desarrollo de software en múltiples plataformas. Describe el reparto de software a través de la red. Las etiquetas XML con las que está descrito definen los componentes, la versión, la plataforma en la que ha sido creado, la relación con otros componentes, etc., lo que permite que se simplifique el proceso de instalación para el usuario y permita también una fácil utilización de las actualizaciones.
- **CML (Chemical Markup Language).** Lenguaje de marcas en el ámbito químico, utilizado principalmente para transmitir información molecular.
- **MathML (Mathematical Markup Language).** Lenguaje de marcas en el ámbito de las matemáticas, muy adecuado para codificar signos matemáticos, símbolos científicos, etc.

- **DocBook.** Lenguaje de marcado basado en XML que se utiliza en el campo de la documentación técnica. Especificación desarrollada por OASIS. Define la estructura y el significado de algunos contenidos.
- **EDI (Electronic Document Interchange).** Lenguaje utilizado principalmente en el intercambio electrónico de datos (Figura 5.3).



**Figura 5.3**  
Ejemplo de intercambio electrónico de datos con el sublenguaje EDI.

- **OFX (Open Financial Exchange).** Lenguaje aplicado al intercambio financiero abierto.
- **TEI (Text Encoding Initiative).** Lenguaje que trata de establecer etiquetas que propicien la descripción de textos científicos y literarios.

## 5.4 Descripción de la estructura y sintaxis

La **hoja de estilos XSLT** es también un documento XML y consta de una serie de reglas que conforman su estructura y sintaxis. Contiene básicamente tres tipos de elementos que componen la hoja de estilo:

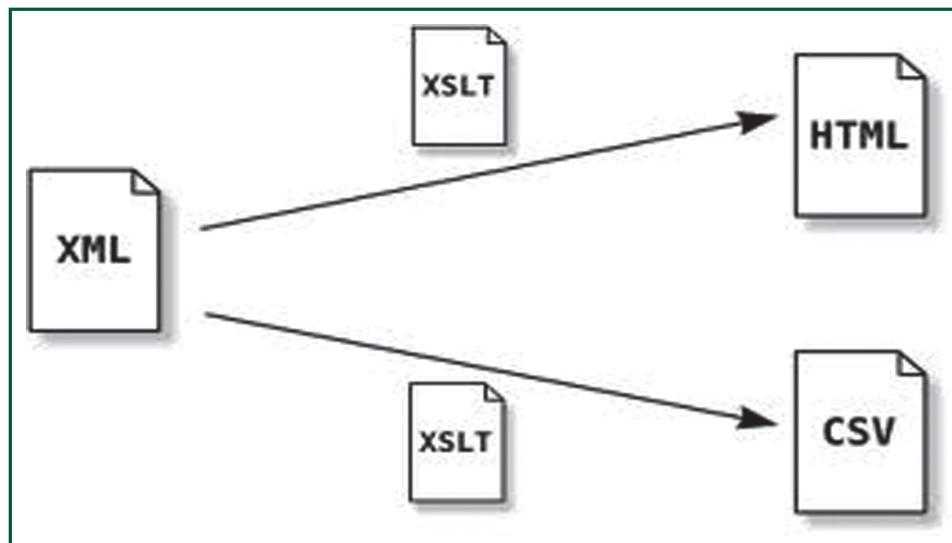
- **Elementos de XSLT.** Pertenecen al grupo de nombres utilizados por XSL y, por lo tanto, sus etiquetas llevan el prefijo "xsl:". Equivalen a las palabras clave del lenguaje de programación y son definidos por el estándar e interpretados por cualquier procesador de XSLT.
- **Elementos LRE (Literal Result Elements).** Estos elementos no pertenecen a XSLT, sino que se repiten en la salida, es decir, muestran una información y no deben formar parte del XSL, como, por ejemplo, un elemento "<fecha>".

- **Elementos de extensión.** Elementos no estándar, como los LRE, que son utilizados por implementaciones concretas del procesador utilizado.

La composición de una hoja de estilo XSLT se inicia con el elemento raíz “xsl:stylesheet”, aunque puede utilizarse también “xsl:transform”, que es prácticamente igual. Como atributos principales relacionados con “xsl:stylesheet”, cabe destacar la versión que se puede indicar, que suele ser la versión 1.0 o la 2.0 y también puede indicarse “xmlns:xsl”, que asigna el namespace xsl. Debemos recordar que las etiquetas de XSL comienzan por el prefijo “xsl:” y, en este caso, irá seguido del valor para XSLT (Figura 5.4).

### Para saber más

W3Schools ha creado una página web dinámica en la que se puede obtener información referente a XML y relacionada con cualquier aspecto de este lenguaje de marca: [www.w3schools.com/xsl/](http://www.w3schools.com/xsl/).



**Figura 5.4**

Aplicación de hojas de estilos XSLT para transformar los documentos XML.

“xsl:stylesheet” también puede presentar otros atributos, como “extension-element-prefixes:”, que sirve para declarar los prefijos de elementos que deben ser elementos de extensión y no LRE (elementos de resultado literal); y “exclude-result-prefixes:”, que sirve para hacer que los elementos de ciertos namespaces (prefijos) no se reproduzcan en la salida:

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

Veamos a continuación en el documento XSLT la definición de la hoja de estilos y el namespace asociados al documento:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Los documentos XSLT están formados por las plantillas, las cuales nos indican a partir de qué elemento del documento XML vamos a empezar a trabajar.

## Para saber más

**Los ficheros CSV son un tipo de documento de formato abierto que permite representar datos en forma de tabla. Las columnas se representan separándolas por comas; y las filas, separándolas por líneas.**

Veamos a continuación las principales instrucciones utilizadas en las hojas de estilo XSLT con sus principales atributos:

- **xsl:value-of select="".** Permite evaluar una expresión XPath (lenguaje con el que pueden construirse expresiones que recorren y procesan un documento XML) . El contenido del nodo actual viene dado por la expresión “.”.
- **xsl:for-each select="".** Aplica la instrucción a los nodos del conjunto especificados en el atributo “select”. Utiliza el atributo “select”.
- **xsl:sort.** Permite ordenar los resultados de una iteración “for-each”. El atributo “select” permite indicar la etiqueta de la ordenación.
- **xsl:if.** Permite ejecutar una instrucción en el caso de que se cumpla una condición. El atributo “test” indica la condición. Suele asociarse con una instrucción “for-each”.
- **xsl:choose.** Contiene elementos xsl:when. Utiliza el atributo “test” (similar al de xsl:if). Son los diferentes “casos” de una sentencia CASE.
- **if/else o switch.** No tiene atributos, pero va unido a dos etiquetas: <xsl:when>, que incluye el atributo “test” que indica la condición; <xsl:otherwise>, sin atributos, que engloba las instrucciones que se deben ejecutar.
- **xsl:output.** Indica si la salida es XML, HTML o texto normal.

La instrucción <xsl:text> permite generar texto que no puede generarse simplemente añadiéndolo. La instrucción <xsl:attribute> permite generar un atributo y su valor; se utiliza cuando el valor del atributo se obtiene de algún nodo. Por ejemplo, puede generarse la etiqueta <img>, en la que el valor del atributo href es el contenido de la etiqueta <imagen>.

## 5.5 Utilización de plantillas

El lenguaje XSL está basado en plantillas. Las plantillas son una definición que permite la selección de un subconjunto de los registros existentes en un documento XML. La plantilla deberá incluir dos tipos de código: instrucciones XSL, para seleccionar los elementos a mostrar, y etiquetas que, una vez efectuada la selección de registros, se aplicarán sobre éstos y se creará una salida formateada m4c7\_008bis. La utilización de plantillas de hojas de estilo XSL es una opción muy práctica para reducir tiempo en la elaboración de los documentos (Figura 5.5).

**Figura 5.5**

Ejemplo de plantilla XSLT con sus componentes.

Si no estamos utilizando plantillas, el procesador simplemente extraerá el texto contenido en los nodos y lo mostrará todo seguido, puesto que no existen etiquetas que lo delimiten. En el caso de que haya una plantilla vacía, el procesador no sustituirá el nodo y no extraerá el texto contenido.

Cuando un documento XML es procesado por un analizador o *parser*, construye con la información que contiene un árbol de nodos. Este árbol comienza con un elemento raíz, que se diversifica a lo largo de los elementos que cuelgan de él y acaba en nodos hoja, que contienen sólo texto, comentarios instrucciones de proceso, o incluso están vacíos y sólo tienen atributos. El árbol de nodos comienza en el nodo raíz y acaba en los nodos hoja. Por ejemplo, Xpath selecciona partes del documento XML basándose en la estructura en árbol.

El nodo raíz se identifica por "/". No debe confundirse el nodo raíz con el elemento raíz del documento.

Cualquier elemento de un documento XML se convierte en un nodo elemento dentro del árbol. Cada elemento tiene su nodo padre. El nodo padre de cualquier elemento es, a su vez, un elemento, excepto el elemento raíz, cuyo padre es el nodo raíz.

Los nodos elemento tienen, a su vez, hijos, que pueden ser nodos de cualquier tipo, excepto nodos raíz. Algunos tipos de nodos son:

- **Nodo texto.** Referencia todos los caracteres del documento que no están marcados con una etiqueta.

## Para saber más

En la web de W3Schools se definen todos los componentes del XSL, así como ejemplos de utilización (<http://w3schools.com/xsl>).

- **Nodo atributo.** Es como una etiqueta añadida a los hijos del nodo elemento. Cada nodo atributo consta de un nombre, un valor que es siempre una cadena.

- **Nodos comentario y nodos de instrucciones de proceso.** Al contenido de estos nodos puede accederse con la propiedad string-value.

En el lenguaje XSL, el elemento “<xsl:template>” se utiliza para crear plantillas y el atributo “match” que lo acompaña se utiliza para asociar la plantilla con un elemento XML. El valor del atributo “match” es una expresión XPath; por ejemplo, la barra “/” de “match =’/’” define el documento completo y, por lo tanto, la plantilla se aplicará sobre todo el documento XML.

El valor del atributo “match” del elemento “<xsl:template>” es un patrón de búsqueda que indica qué nodos debe transformar la plantilla XSL. Una plantilla XSL “<xsl:template>” es un bloque delimitado de contenido y reglas XSL.

El elemento <xsl:template>, además del atributo “match”, tiene los siguientes parámetros opcionales:

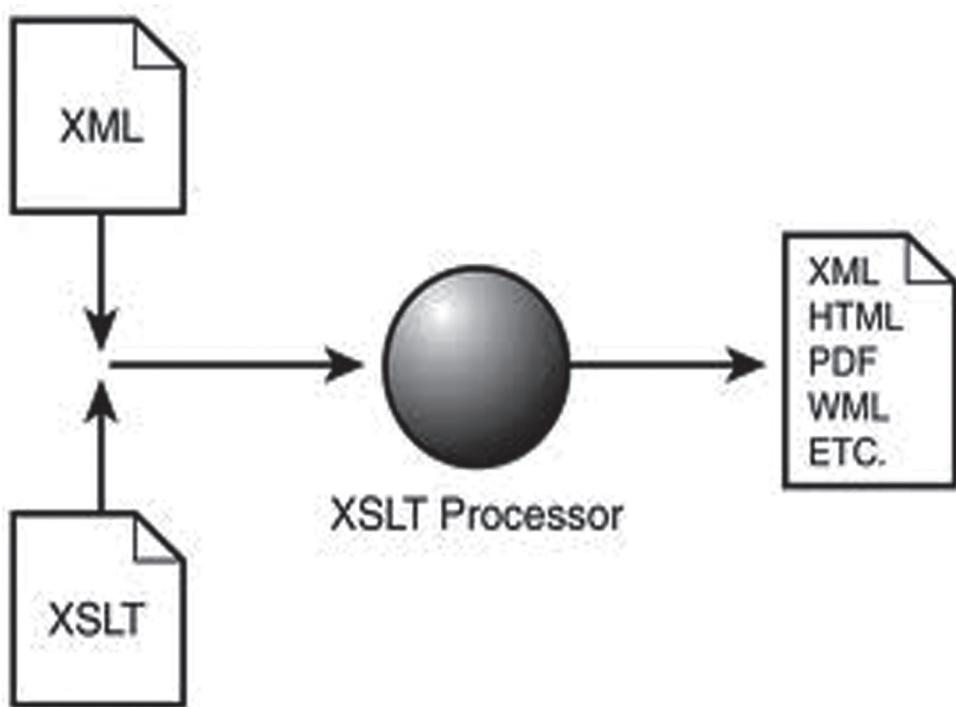
- **Name.** Indica el nombre de la selección; es un atributo opcional si ya se ha indicado el atributo “match”.
- **Priority.** Indica la prioridad de la selección, cuanto más alto es el valor, más alta la prioridad.

## 5.6 Utilización de herramientas de procesamiento

Los documentos XML pueden editarse de dos formas: como si de tratase de cualquier otro fichero ASCII, utilizando un editor estructurado, como XEmacs, NotePad++, o mediante un editor específico para XML, que entienda las particularidades del lenguaje y nos ayude a cerrar las etiquetas.

Para la edición de documentos XML hay muchas opciones, tanto en Windows como en Linux, aunque la mayoría son de pago. Por ejemplo, la herramienta de edición XMLSpy tiene un buen entorno y funciona sólo en Windows, aunque es relativamente inestable. eXcelon Stylus permite aplicar transformaciones en un entorno de tres paneles, es bastante económico para uso personal o académico y tiene una versión de prueba de treinta días. Además, está basado en Java, funciona tanto en Windows como en Linux, completa las etiquetas y es aceptablemente rápido. Está basado también en algunas herramientas libres, como Batik y FOP de Apache. Otra opción, bastante simple, es XMLShell, que permite hacer transformaciones XSLT simples.

Los mismos entornos incluyen facilidades para poder validar el código XML resultante, aunque la validación también puede llevarse a cabo mediante los analizadores XML, de los que hay muchos, de buena calidad y la mayoría de ellos gratuitos. Uno de los más conocidos y utilizados es Xerces (Figura 5.6).

**Figura 5.6**

La información se muestra mediante un procesador XSLT.

La mayoría de los validadores pueden trabajar de forma independiente o como librerías desde el lenguaje de programación elegido.

Hoy en día, la mayor parte de los navegadores actuales son capaces de entender el lenguaje XML. Por ejemplo, Internet Explorer lee los ficheros XML y los trata de una forma especial, presentando la jerarquía de las etiquetas en diferentes niveles.

Otros navegadores, como Mozilla o Netscape, también entienden XML, aunque no permiten editar lo forma adecuada ni presentarlo de forma jerárquica como Internet Explorer.

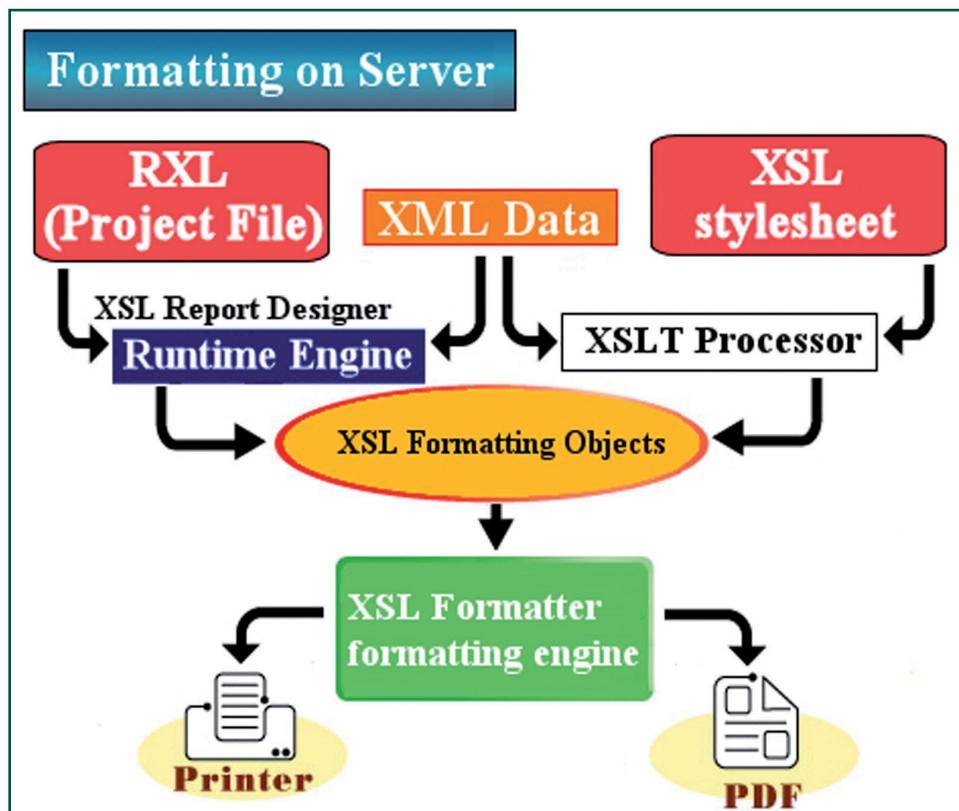
En algunos casos, los navegadores son capaces también de aplicar transformaciones como XSLT o CSS. En realidad, para editar XML y XSL no hace falta ningún editor especial, aunque se recomienda un editor de XML, o incluso un entorno integrado que nos ayude a identificar bien el código, a cerrar las etiquetas en el orden correcto, o que nos sugiera la etiqueta y atributos admisibles en cada momento según la DTD indicada.

## Recuerda

**Las hojas de estilo XSL pueden insertarse en la cabecera del documento XML y sólo serán reutilizables en dicho documento. Si se genera una hoja de estilo externa, podrá aplicarse a distintos documentos XML.**

## 5.7 Verificación del resultado

**Verificar el resultado** de un documento realizado en XSL es un proceso que consiste en comprobar que el resultado de la salida presentada se corresponda con las especificaciones del documento (Figura 5.7).



Las acciones que deben realizarse para la verificación del documento son las siguientes:

- Comprobar el contenido de un documento XSL.
- Determinar que el documento XSL es una instancia válida del vocabulario, gramática o reglas correctas.
- Definir los elementos que pueden aparecer dentro de un documento XSL y los atributos que pueden asociarse a un elemento.
- Definir si un elemento está vacío o puede incluir texto.
- Definir el valor por defecto de un atributo.
- Definir los elementos que pueden contener elementos hijo.
- Definir la secuencia de los elementos hijo que aparecen en un elemento.
- Definir el número de elementos hijo.

## 5.8 Depuración

Depurar el lenguaje XSL consiste en ver las variables que lo componen, definir los puntos de interrupción que lo forman y examinar el código. El **depurador de XSL** puede utilizarse para limpiar una hoja de estilos XSLT.

Debemos tener en cuenta que XSL posee todas las características de un lenguaje de programación, por lo que si no tenemos una estrategia de depuración adecuada resultará muy complicado poder detectar los posibles errores que hayamos cometido en el código, ya sea desde el punto de vista de la sintaxis como de la lógica de la aplicación que estemos desarrollando (Figura 5.8).

The screenshot shows a Mozilla Firefox browser window displaying the W3C XML Technology page. The URL in the address bar is [www.w3.org/standards/xml/](http://www.w3.org/standards/xml/). The page title is "XML TECHNOLOGY". On the left, there is a sidebar titled "STANDARDS" with links to "Web Design and Applications", "Web Architecture", "Semantic Web", "XML Technology", "Web of Services", "Web of Devices", "Browsers and Authoring Tools", "All Standards and Drafts", and "About W3C Standards". The main content area contains several sections: "XML Essentials", "Schema", "Security", "Transformation", "Query", "Components", "Processing", "Internationalization", and "Publishing". Each section provides a brief description of its purpose and how it relates to XML.

**Figura 5.8**

El W3C ofrece todo tipo de información para validar y depurar cualquier documento XML.

Para generar plantillas XSL se precisa de un entorno que permita fácilmente comprobar la corrección de la salida obtenida. Existen varios entornos para este propósito. Uno de ellos es <oXygen/> ([www.oxygenxml.com](http://www.oxygenxml.com)), que facilita la depuración de plantillas en un entorno fácil de manejar.

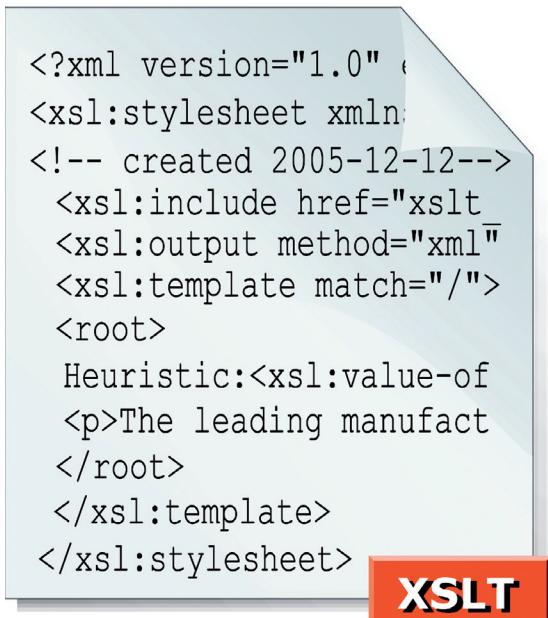
El software *oXygen XML Editor* está basado en Java. Con él pueden crearse y editarse documentos XML, XSL, TXT y DTD, con soporte especial para todas las particularidades del lenguaje XML. Resulta bastante flexible a la hora de editar código con soporte Unicode, código de colores para los comandos de sintaxis, soporte

para CSS autocompletado de etiquetas XML y analizador de sintaxis, entre otras muchas opciones. Incorpora una interfaz de diseño nítido que facilita el trabajo con soporte para *drag-and-drop* y permite, además, previsualizar nuestro trabajo tanto en formato XML como XHTML.

Otra aplicación para tratar documentos XSL es Altova XMLSpy Standard Edition, en su versión 2005. Se trata de un editor XML de nivel de entrada para que el usuario pueda visualizar, validar y editar documentos en el lenguaje XML. Es la herramienta perfecta para aquellos usuarios que necesiten visualizar sus archivos en el lenguaje XML, DTD, esquemas XML, XSLT y también archivos XQuery, y, además, podrá llevar a cabo tareas de edición ligeras.

## 5.9 Elaboración de documentación

La **elaboración de documentos XSL** se lleva a cabo con la ayuda de un software específico, como por ejemplo Altova StyleVision, Stylus Studio, Antena Casa Formatter V5, Ecrion XF de Productos, etc. En el principio del documento se deberá tener asociada (con la correspondiente instrucción de procesamiento) la hoja de estilo XSLT que se está creando, de manera que en todo momento, al introducir cualquier cambio en la hoja XSLT, pueda comprobarse el resultado de la transformación cargando el documento XML en el navegador (los navegadores actuales soportan XML y XSLT). El proceso continuo de ensayo y error servirá para comprobar la viabilidad del documento (Figura 5.9).



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Format" version="1.0">
    <!-- created 2005-12-12-->
    <xsl:include href="xslt.xsl" />
    <xsl:output method="xml" encoding="UTF-8" />
    <xsl:template match="/">
        <root>
            Heuristic:<xsl:value-of select="." />
            <p>The leading manufacturer of high-quality
            </p>
        </root>
    </xsl:template>
</xsl:stylesheet>
```

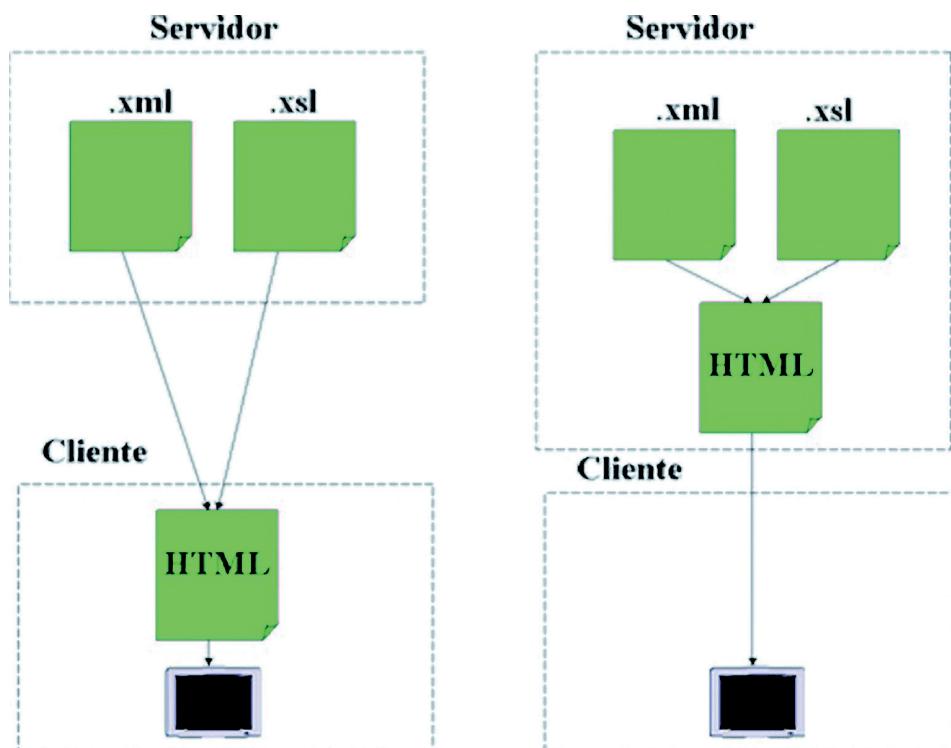
**XSLT**

**Figura 5.9**

Las hojas de estilo XSLT realizan la transformación del documento mediante unas reglas de plantilla.

Las extensiones deberán ir acompañadas de su correspondiente esquema XSD y de una guía de uso que detalle la descripción de las diferentes etiquetas, así como sus atributos y reglas de uso (Figura 5.10).

Por lo que respecta a las **características de estilo**, éstas sirven para mejorar la presentación visual de los documentos mediante atributos *style* o un elemento <style> de cabecera incorporados en la propia hoja XSLT.



**Figura 5.10**  
Estructura de una hoja de estilo XSL.

## Resumen

Para convertir un documento XML de modo que pueda utilizarse por una aplicación, por ejemplo cuando es presentado por un navegador o en el intercambio entre sistemas heterogéneos, se usan lenguajes de transformación, de los que el más utilizado para especificar las transformaciones es XSL (*Extensible Stylesheet Language*). Se trata de un lenguaje que busca patrones dentro de un documento e indica cómo reestructurarlos para su correcta interpretación.

El lenguaje de transformación XSL está dividido en tres subfamilias: XSLT (Extensible Stylesheet Language Transformations), que son hojas de estilo que permiten convertir documentos XML de una sintaxis a otra; XSL-FO, que es un lenguaje de hojas extensibles de formateo de objetos que permite especificar el formato visual con el cual se quiere presentar un documento XML; y XPath o *XML Path Language*, que es una sintaxis, no basada en XML, que permite acceder o buscar porciones de un documento XML.

La hoja de estilos XSLT es también un documento XML. Contiene básicamente tres tipos de elementos que componen el archivo: los elementos de XSLT que pertenecen al espacio de nombres utilizados por XSL, y son el equivalente a las palabras clave de una lenguaje de programación; los elementos LRE (*Literal Result Elements*) que muestran una información y no tienen que ser parte del XSL; y los elementos de extensión que son elementos no-estándar, al igual que los LRE, pero para implementaciones concretas.

Depurar el lenguaje de transformación XSL consiste en ver las variables que lo componen, definir los puntos de interrupción que lo forman y examinar el código. El depurador de XSL se puede utilizar para depurar una hoja de estilos XSLT.

La elaboración de documentos XSL se hará con la ayuda de un software específico de XML. Normalmente este tipo de software no sólo se utiliza para la edición textual del documento, sino que son verdaderos entornos con funcionalidades añadidas que aumentan la productividad y la seguridad en la generación de este tipo de documentos.

## Ejercicios de autocomprobación

**Indica si las siguientes afirmaciones son verdaderas (V) o falsas (F):**

1. Un documento XML está compuesto de datos, que, como tales, pueden utilizarse en múltiples aplicaciones informáticas y con objetivos bien diferentes.
2. El lenguaje XSL se compone de tres sublenguajes: XSL, XSLT, y XPath, que, pese a ser tres especificaciones diferentes, se complementan y pueden utilizarse conjuntamente.
3. Los lenguajes de hojas de estilo son la única forma de convertir un documento XML a otro formato.
4. Los ficheros CSV son un tipo de documento de formato abierto que permite representar datos en forma de tabla.
5. El lenguaje XSL está basado en plantillas. Las plantillas son una definición que permite la selección de un subconjunto de los registros existentes en un documento XML.
6. Definir si un elemento está vacío o puede incluir texto y definir el valor por defecto de un atributo son acciones que deben realizarse para la verificación del documento.
7. El lenguaje de transformación XSL está dividido en tres subfamilias: XSLT (*Extensible Stylesheet Language Transformations*); XSL-FO; y XPath o *XML Path Language*.

**Completa las siguientes afirmaciones:**

8. Hoy en día, la mayor parte de los \_\_\_\_\_ actuales son capaces de entender el lenguaje XML. Por ejemplo, Internet Explorer lee los ficheros XML y los trata de una forma especial, presentando la \_\_\_\_\_ de las \_\_\_\_\_ en diferentes niveles. Mozilla o \_\_\_\_\_, también entienden \_\_\_\_\_, aunque no permiten editarlo de forma adecuada ni presentarlo de forma jerárquica como Internet Explorer.

9. Depurar el lenguaje \_\_\_\_\_ consiste en ver las \_\_\_\_\_ que lo componen, \_\_\_\_\_ los puntos de interrupción que lo forman y examinar el código. El \_\_\_\_\_ de XSL puede utilizarse para limpiar una hoja de \_\_\_\_\_ XSLT.
10. La elaboración de documentos \_\_\_\_\_ se hará con la ayuda de un software específico de \_\_\_\_\_. Normalmente este tipo de \_\_\_\_\_ no sólo se utiliza para la edición textual del documento, sino que son verdaderos entornos con funcionalidades añadidas que aumentan la \_\_\_\_\_ y la \_\_\_\_\_ en la generación de este tipo de documentos.

Las soluciones a los ejercicios de autocomprobación se encuentran al final de esta Unidad Formativa. En caso de que no los hayas contestado correctamente, repasa la parte de la lección correspondiente.

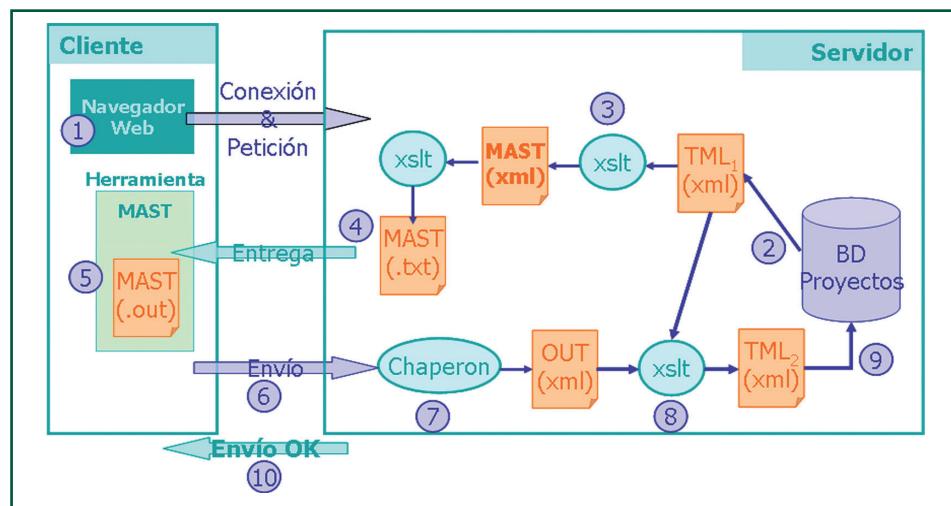
## 6. ALMACENAMIENTO DE INFORMACIÓN

Una de las principales ventajas de la informática es la posibilidad de almacenar información. Los ordenadores pueden guardar información de forma permanente, incluso después de apagados. El **almacenamiento de la información** hace posible guardar datos y también programas para tratarlos y reproducirlos. Para ello, precisamente, se utilizan las bases de datos, que son una parte fundamental de todas las organizaciones, porque en ellas se almacena la información.

Actualmente, a diario se genera gran cantidad de información que debemos almacenar y procesar. Y no sólo eso, pues también hay que tratar y realizar operaciones con dicha información. Por ello es preciso utilizar un sistema que nos ayude a seleccionar, catalogar y ordenar la información que consideramos relevante, así como también a desechar aquella que esté duplicada o sea superflua. Este sistema se conoce con el nombre de **Sistema de Gestión de Bases de Datos** (SGBD).

En este apartado conoceremos los sistemas de almacenamiento de datos vinculados al lenguaje XML, así como las técnicas vinculadas al tratamiento y almacenamiento de datos para XML.

Hemos dicho que para almacenar la información se utilizan las bases de datos. En Internet podemos encontrar varios tipos de bases de datos, que son diferentes a las bases de datos relacionales. Para poder integrar estas bases de datos se precisa un navegador y una interfaz conocida como *Common Gateway Interface* (CGI), que nos permitirá conectarnos a la base de datos. Hoy en día, las bases de datos más utilizadas son las XML (Figura 6.1).



**Figura 6.1**  
Ejemplo de desarrollo de una base de datos nativa XML.

## 6.1 Sistemas de almacenamiento de información

El **sistema de almacenamiento de información de XML** puede dividirse en dos categorías: la información centrada en los datos y la información centrada en el contenido. Si el documento XML tiene una estructura bien definida y contiene datos actualizables utilizados de maneras diversas, hablaremos de un **documento centrado en los datos**. El sistema de almacenamiento centrado en los datos suele incluir documentos menos estructurados y es muy apropiado para ítems. Por el contrario, los datos centrados en el documento suelen ser de tamaño y contenido más diversos que los centrados en los datos, que contienen datos de tamaño limitado y reglas más rígidas para campos opcionales y contenido.

Los sistemas de almacenamiento XML son útiles para ambos tipos de información, ya que XML está siendo ampliamente utilizado en sistemas que administran ambos tipos de datos. La mayoría de estos sistemas de almacenamiento están concebidos para servir a uno de estos formatos de almacenamiento de datos mejor que al otro. Las bases de datos relacionales tradicionales son más adecuadas cuando se trata de requerimientos centrados en los datos, mientras que los sistemas de administración de contenidos son más adecuados para almacenar datos centrados en el documento. Los sistemas de almacenamiento XML suelen contener ambas categorías de datos dentro de la misma aplicación (Figura 6.2).

**Figura 6.2**

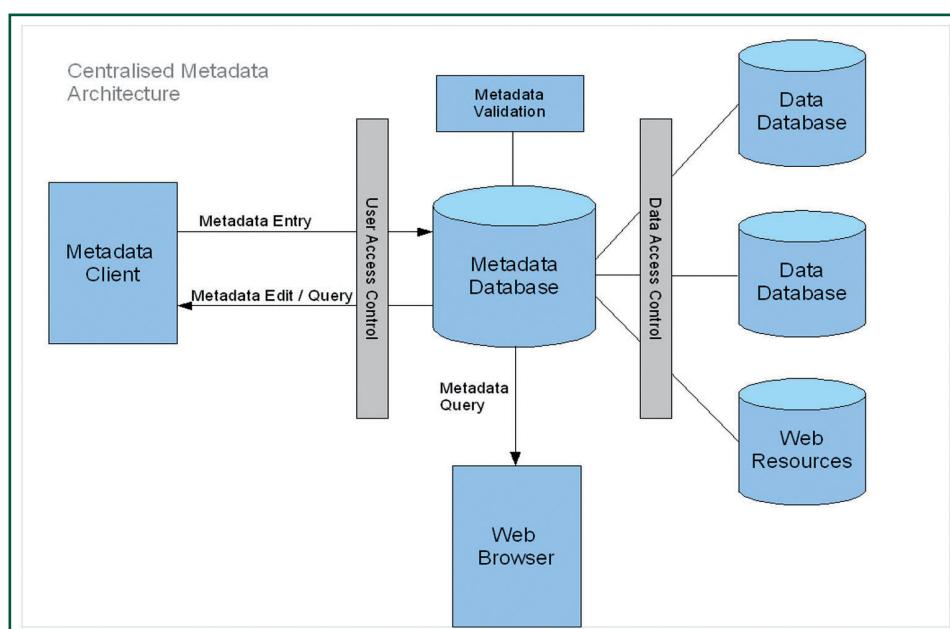
El sistema de almacenamiento de la información es muy importante para poder acceder a ella posteriormente.



Los sistemas de bases de datos deben ser capaces de exponer los datos relacionales como XML y almacenar el XML recibido como datos relativos con el fin de transferir, obtener y almacenar de manera transparente los datos requeridos por la aplicación.

La diferencia entre la información centrada en los datos y la centrada en el documento no resulta siempre fácil de identificar. Por lo general, los requisitos que manejan los datos se inclinan hacia una dirección u otra. Es preciso, por lo tanto, tomarse un tiempo para saber si la información está centrada en los datos o en los documentos, puesto que puede resultar una ventaja a la hora de elegir el modo de almacenamiento. El lenguaje XML se está extendiendo rápidamente como un lenguaje estándar de representación e intercambio de datos en Internet. Por lo tanto, las bases de datos XML nativas se están convirtiendo en una importante alternativa, ya que siguen almacenando toda la información de manera relacional o bien almacenan el documento entero en formato *Binary Large Object* (BLOB).

El lenguaje XML está provocando la aparición de nuevas tecnologías, entre ellas una nueva generación de bases de datos que, si bien se halla todavía en fase de investigación y desarrollo, puede resultar en un futuro una buena alternativa a las ya conocidas bases de datos relacionales. Estas bases de datos se fundamentan en el lenguaje XML o en la base de datos nativa en XML (*native XML database*). Hoy en día, este tipo de base de datos es muy adecuado para aquellas empresas u organizaciones que guardan su información en diferentes formatos, ya que les permite pasar rápidamente la información a un formato XML, con la aplicación de su preferencia, y de ese modo almacenarla en una de estas bases de datos y evitarse el proceso de conversión de un formato a otro; si tenemos en cuenta, además, que todavía no existe un lenguaje estándar para el procesamiento de los datos, dichas bases de datos son muy adecuadas para realizar búsquedas sobre los documentos almacenados (Figura 6.3).



**Figura 6.3**  
Ejemplo de gestión de datos con XML.

## 6.2 Inserción y extracción de información en XML

Con el objetivo de manipular los datos entre un documento XML y una base de datos y viceversa, los datos en la base de datos deben ser mapeados a la estructura del documento XML. Sencillamente, el documento XML completo es mapeado a una columna única en la base de datos. Estrategias más complejas incluyen el mapeo de cada elemento a una columna correspondiente en la base de datos, o el mapeo de la estructura del documento XML a la base de datos (Figura 6.4).

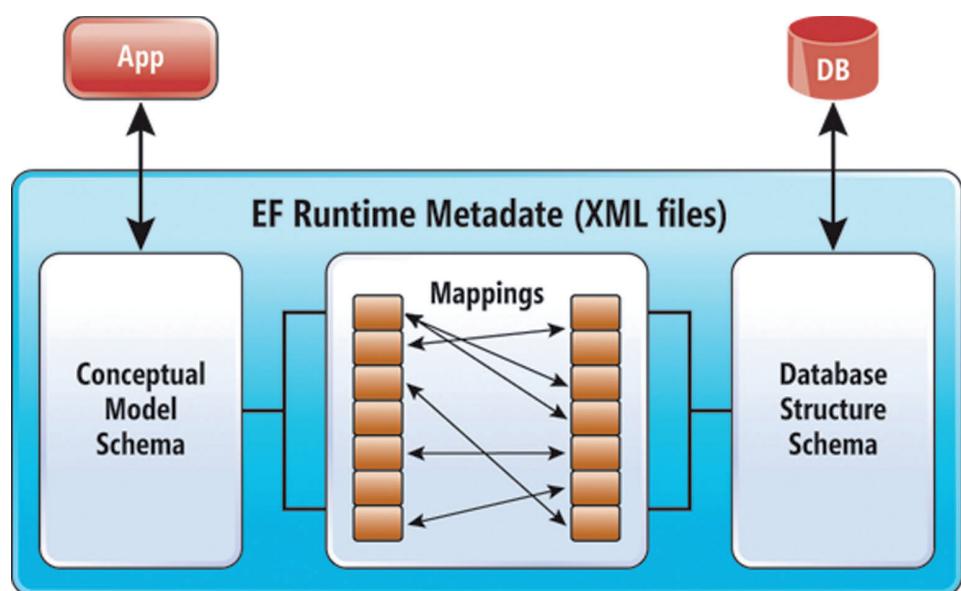


Figura 6.4

Los documentos XML almacenan información y gestionan su inserción y extracción.

### 6.2.1 Estrategias de almacenamiento

Las estrategias de inserción y extracción en el almacenamiento de información en XML se basan en tres procedimientos básicos:

- **Almacenar el documento completo en formato de texto como un gran objeto binario (BLOB) en una base de datos relacional.** Es una buena estrategia si el documento XML contiene un contenido estático que sólo puede ser modificado cuando el documento completo es reemplazado. Se trata de una estrategia común para aquellos datos centrados en el documento a medida que son obtenidos y almacenados como una unidad. Almacenar el documento completo en formato de texto es fácil de implementar, ya que éste no precisa mapeo o traducción, aunque puede limitar la búsqueda, el indexamiento y la granularidad de la obtención de documentos XML. El documento XML, además, se conserva igual que antes de ser almacenado, lo que minimiza el trabajo de rearmarlo después de su obtención.

- Almacenar una forma modificada del documento completo en el sistema de archivos.** Este método es muy adecuado cuando el número de documentos es pequeño y los documentos XML no se actualizan ni se transfieren con frecuencia entre sistemas de archivos. Almacenar una forma modificada del documento completo en el sistema de archivos suele ser una limitación, ya que el sistema de archivos no contiene una base de datos muy buena. Se trata de un método adecuado para un número menor de documentos XML; además, sólo puede ser incluido en el diseño, ya que el documento XML se desplaza por el sistema de archivos; eventualmente, los contenidos de un documento XML podrían terminar en una base de datos.
- Mapear la estructura de los datos en el documento en la base de datos.** Si la estructura del documento XML no es compatible con la estructura de la base de datos, el documento deberá transformarse para ajustarlo a la estructura de la base de datos antes de almacenarlo. Mapear la estructura de los datos a la base de datos es una opción muy común y constituye el foco de muchas de las características habilitadoras de XML en los principales productos de bases de datos (Figura 6.5). Un ejemplo de ello sería mapear un documento XML que contiene una orden de venta con tablas, como *Órdenes*, *Ítems*, *Partes* y *Clientes*; u objetos como *Orden*, *Ítem*, *Parte* y *Cliente*.

## Para saber más

El **mapeo objeto-relacional** es una técnica de programación para convertir datos del sistema de tipos utilizado en un lenguaje de programación orientado a objetos al utilizado en una base de datos relacional empleando para ello un motor de persistencia. El resultado es una base de datos orientada a objetos virtual sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo).

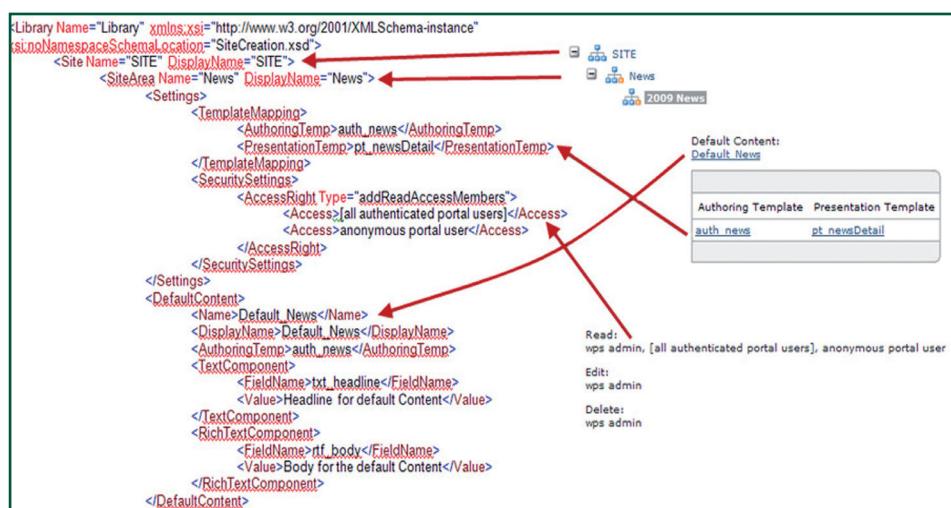


Figura 6.5

Ejemplo de mapeo entre los objetos y las secciones de XML y Lotus Web Content.

## 6.3. Técnicas de búsqueda de información en documentos XML

El lenguaje XML utiliza técnicas de búsqueda de información en aquella información que se halla contenida en sus documentos. Podríamos destacar los lenguajes **Xpath** y **Xquery**.

### 6.3.1 El lenguaje XPath

El lenguaje XPath utiliza la estructura anidada de los documentos XML y permite presentar los documentos en una estructura de árbol, donde cada elemento del árbol se conoce como **nodo**. Xpath proporciona el mecanismo para acceder a cada nodo, utiliza la posición en el árbol en la que se encuentra y obtiene la ruta para encontrarlo. XPath no es un lenguaje de programación, ni un lenguaje de estilos, sino un lenguaje que proporciona la sintaxis necesaria para acceder a elementos y atribuirlos dentro de un documento XML. El acceso, mediante Xpath, es rápido y directo, pues no es preciso tener que recorrer todo el árbol para encontrar los elementos y los atributos buscados. El lenguaje Xpath está compuesto por símbolos, con los que se definen las rutas y los distintos nodos. Los nodos son las etiquetas y atributos que constituyen el documento XML. Existen distintos tipos de nodos:

- **Nodo padre.** Cada elemento y atributo tiene un parente.
- **Nodo hijo.** Los nodos elemento pueden tener 0, 1 o más hijos.
- **Nodo ancestro.** Un nodo parente, el parente de ese nodo, etc.
- **Nodo descendiente.** Un nodo hijo, los nodos hijo de ese nodo, etc.

### 6.3.2 El lenguaje XQuery

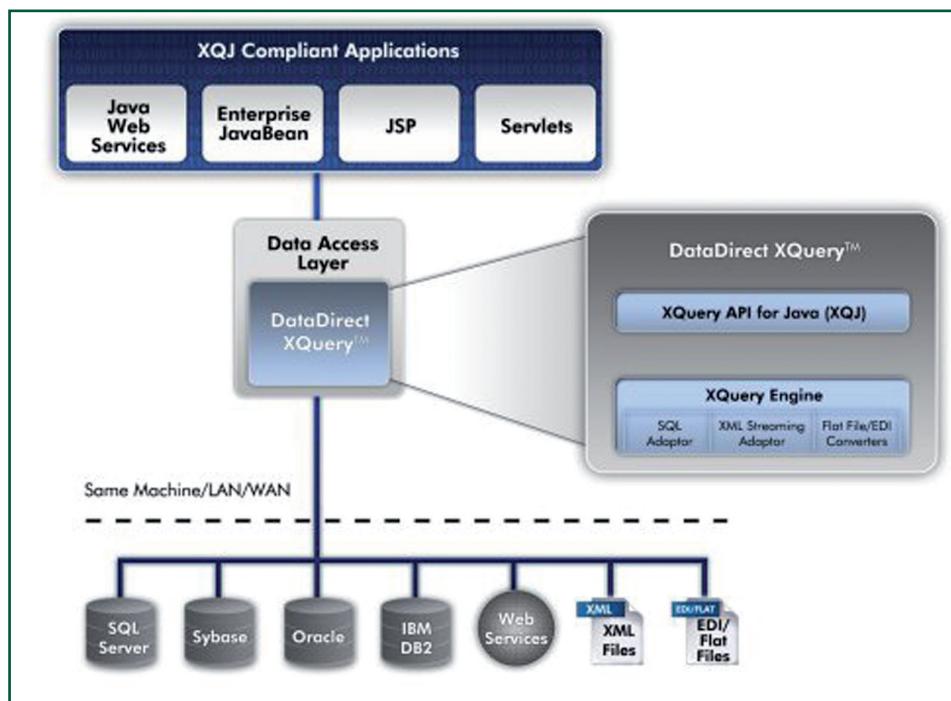
**XQuery** es un lenguaje de consulta diseñado para el almacenamiento de datos XML. Desde el punto de vista semántico es similar a SQL, aunque incluye algunas capacidades de programación. XQuery proporciona los medios para poder extraer y manipular la información de documentos XML, o de cualquier fuente de datos que pueda ser representada mediante XML, como, por ejemplo, bases de datos relacionales o documentos ofimáticos. XQuery utiliza expresiones XPath para acceder a determinadas partes del documento XML. También puede construir nuevos documentos XML a partir de los resultados de la consulta. Del mismo modo, puede utilizarse una sintaxis similar a XML si se conoce la estructura (elementos y atributos) con antelación o, en caso contrario, utilizar expresiones de construcción dinámica de nodos. Estos constructores se definen como expresiones dentro del lenguaje, y pueden anidarse arbitrariamente. El lenguaje **XQuery** se basa en el mo-

delo de árbol de la información contenida en el documento XML, que consiste en siete tipos distintos de nodo: elementos, atributos, nodos de texto, comentarios, instrucciones de procesamiento, espacios de nombres y nodos de documentos.

- El sistema de tipos utilizado por el lenguaje **XQuery** considera todos los valores como secuencias, asumiéndose un valor simple como una secuencia de un solo elemento. Los elementos de una secuencia pueden ser valores atómicos o nodos. La lista completa de los tipos disponibles está basada en tipos originales definidos en XML Schema.
- XQuery es un lenguaje declarativo porque permite un acceso a los datos similar al de las bases de datos y la obtención de información directa de un documento XML. Puede ser utilizado para obtener datos, y especifica el resultado pero no las acciones que deben emprenderse para obtenerlo. Es similar a SQL en funcionalidad (Figura 6.6).

## Recuerda

**Los nodos son elementos de un documento XML. Cada elemento tiene su nodo padre. El nodo padre de cualquier elemento es, a su vez, otro elemento, excepto el elemento raíz, cuyo padre es el nodo raíz.**



**Figura 6.6**  
Ejemplo de la arquitectura del lenguaje XQuery.

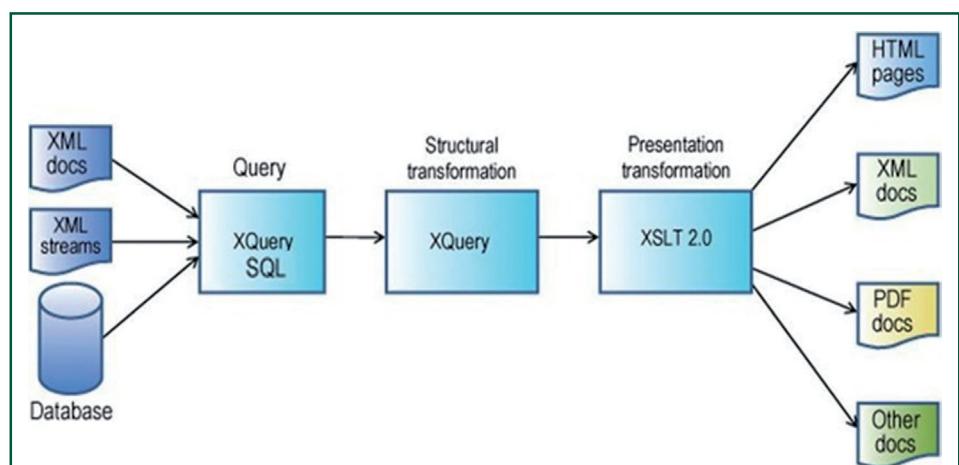
## 6.4 Lenguajes de consulta y manipulación

Actualmente, XML se ha convertido en una herramienta de uso cotidiano en los entornos de tratamiento de información y programación. Sin embargo, a medida que se emplea en proyectos de mayor complejidad y la cantidad de datos almacenados en XML aumenta, se puede comprobar que las herramientas más habi-

tuales para manipular desde un programa un árbol con un conjunto de datos en XML, los *parsers* SAX y DOM, no son prácticas para manejar grandes y complejas colecciones de datos en XML.

El principal problema a la hora de procesar colecciones de datos en XML a bajo nivel mediante *parsers* DOM y SAX es la necesidad de escribir una gran cantidad de código para poder realizar el procesamiento.

En este sentido, XQuery es el lenguaje diseñado para escribir consultas sobre colecciones de datos expresados en XML. Abarca desde archivos XML hasta bases de datos relacionales con funciones de conversión de registros a XML. Su principal función es extraer información de un conjunto de datos organizados como un árbol de etiquetas XML. En este sentido, XQuery es independiente del origen de los datos (Figura 6.7).



**Figura 6.7**

XQuery es el motor de búsqueda de datos en XML.

Una consulta en lenguaje XQuery lee una secuencia de datos en XML y devuelve como resultado otra secuencia de datos en XML (Figura 6.8).

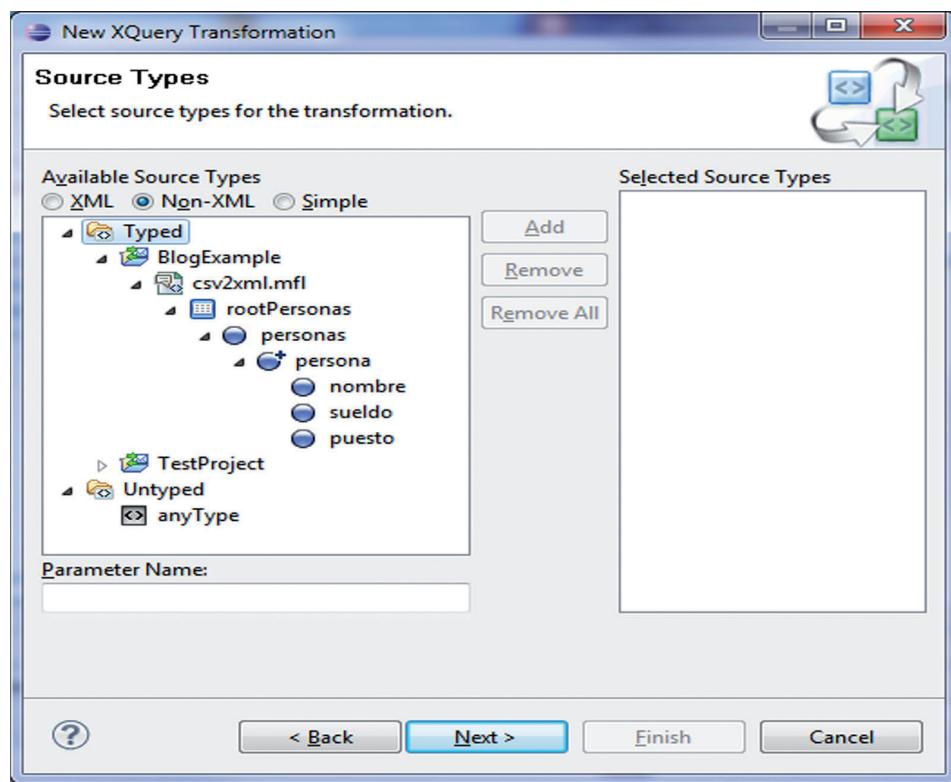
En XQuery las consultas están compuestas por cinco cláusulas: FOR, LET, WHERE, ORDER BY y RETURN. Las consultas siguen la norma FLWOR (las siglas en inglés de FOR, LET, WHERE, ORDER BY y RETURN). Veamos a continuación en qué consisten:

- **FOR.** Vincula una o más variables a expresiones escritas en Xpath, creando un flujo de secuencias en el que cada secuencia está vinculada a una de las variables.
- **LET.** Vincula una variable al resultado completo de una expresión, añadiendo esos vínculos a las secuencias generadas por una cláusula FOR. Si no existe ninguna cláusula FOR, hay que crear una secuencia única que contenga esos vínculos.

- **WHERE.** Filtra las secuencias y elimina todos los valores que no cumplen con las condiciones establecidas.
- **ORDER BY.** Ordena las secuencias según el criterio establecido.
- **RETURN.** Construye el resultado de la consulta para una determinada secuencia, después de haber sido filtrada por la cláusula *where* y ordenada por la cláusula *order by*.

A continuación, veamos las reglas que debe cumplir cualquier consulta escrita en XQuery:

- FOR y LET sirven para crear las tuplas (cada uno de los valores que toma una variable) con las que trabajará el resto de las cláusulas de la consulta, y éstas pueden utilizarse tantas veces como se desee en una consulta, incluso dentro de otras cláusulas.
- Sólo puede declararse una única cláusula WHERE.
- Sólo puede declararse una única cláusula ORDER BY.
- Sólo puede declararse una única cláusula RETURN.
- Ninguna de las cláusulas FLWOR es obligatoria en una consulta XQuery.



**Figura 6.8**  
Ejemplo de uso de XQuery con SourceXQuery.

Existen una serie de **funciones XQuery** aptas para hacer consultas. Veamos a continuación algunos ejemplos de estas funciones:

- **doc("nombreDocumento.xml")**. Permite extraer datos del documento indicado.
- **doc("nombreDocumento.xml")//nodo\_raíz/nodo\_hijo**. Permite acceder a un conjunto de nodos bien identificados; se ha de añadir el camino en el árbol de los nodos del documento para acceder a los datos que necesitamos.
- **doc("nombreDocumento.xml")//nodo\_raíz/nodo\_hijo[condición\_busqueda]**. Indica una condición que ha de cumplir el resultado de la búsqueda.

Veamos un ejemplo: un documento XML llamado DB\_BailesDeSalon.xml contiene información sobre una academia de baile; a continuación, veamos cómo aplicar búsquedas de información mediante XQuery:

```
<?xml version="1.0" encoding="ISO-8859-1 "?>
<Bailes>
  <baile id="1">
    <nombre>Tango</nombre>
    <precio cuota="mensual" moneda="euro">27</precio>
    <plazas> 20</plazas>
    <comienzo> 1/1/2012</comienzo>
    <fin> 1122/2012 </fin>
    <profesor> Pedro Pérez</profesor>
    <sala>1 </sala>
  </baile>
  <baile id="2">
    <nombre>Cha-cha-cha</nombre>
    <precio cuota="trimestral" moneda=" euro">80</precio>
    <plazas> 18</plazas>
    <comienzo> 1/2/2012</comienzo>
    <fin>31/7/2012</fin>
    <profesor>Alberto Ruiz</profesor>
    <sala>1 </sala>
  </baile>
  <baile id="3">
    <nombre>Rock</nombre>
    <precio cuota=>mensual<> moneda=>euro<>>30</precio>
    <plazas> 15</plazas>
    <comienzo>1/1/2012</comienzo>
```

```
<fin>1/12/2012</fin>
<profesor>Maruja Fernández</profesor>
<sala>1</sala>
</baile>
<baile id=>4<>
<nombre>Merengue</nombre>
<precio cuota=>trimestral<> moneda=>dolares<>>75</precio>
<plazas> 12</plazas>
<comienzo>1/1/2012</comienzo>
<fin>1/12/2012</fin>
<profesor>Marisa Saez</profesor>
<sala>2</sala>
</baile>
<baile id=>5<>
<nombre>Salsa</nombre>
<precio cuota=>mensual<> moneda=>euro<>>32</precio>
<plazas>10</plazas>
<comienzo>1/1/2012</comienzo>
<fin>1/12/2012</fin>
<profesor>Jaime Feliciano</profesor>
<sala>2</sala>
</baile>
```

La primera búsqueda es saber qué bailes se realizan en la sala número 1:

```
for $baile in doc(<<DB_BailesDeSalon.xml>>) //Bailes/baile
where $baile/sala=1  return $baile/nombre
```

Esta función nos mostraría un listado de bailes que se realizan en la sala 1 con formato de documento XML; es decir, mantendría el formato de presentación con etiquetas. Para evitarlo, sería necesario añadir el código que se muestra a continuación a la cláusula “return” para que extrajera los datos de ese elemento XML:

```
for $baile in doc("DB_BailesDeSalon.xml") //Bailes/baile
let $n:=$baile/nombre
where $baile/sala=1
return data($n)
```

De esta forma, la búsqueda de datos puede complicarse tanto como sea necesario si utilizamos las funciones FLWOR.

## Para saber más

Para una mayor información y actualizaciones sobre XPath y XQuery, véase el siguiente enlace: <http://www.w3schools.com>

Por ejemplo, para buscar el nombre de los profesores que dan clases con cuotas mensuales, se aplicaría la siguiente función:

```
for $baile in doc("DB_BailesDeSalon.xml")//Bailes/baile  
let $profesor:=$baile/profesor  
where $baile/precio[@cuota="mensual"]  
return $profesor
```

O también podríamos obtener los nodos de aquellos bailes que se imparten en la sala número 2 y cuyo precio sea menor de 35 euros:

```
for $baile in doc("DB_BailesDeSalon.xml")//Bailes/baile  
let $n:=$baile/nombre  
where $baile/sala = 2 and $baile/precio < 35  
and $baile/precio[@moneda=>>euro>>]  
return $n
```

El lenguaje XQuery ofrece también la posibilidad de insertar datos en la base de datos, reemplazarlos o eliminarlos. Para ello, utilizará INSERT, REPLACE, VALUE, DELETE o RENAME.

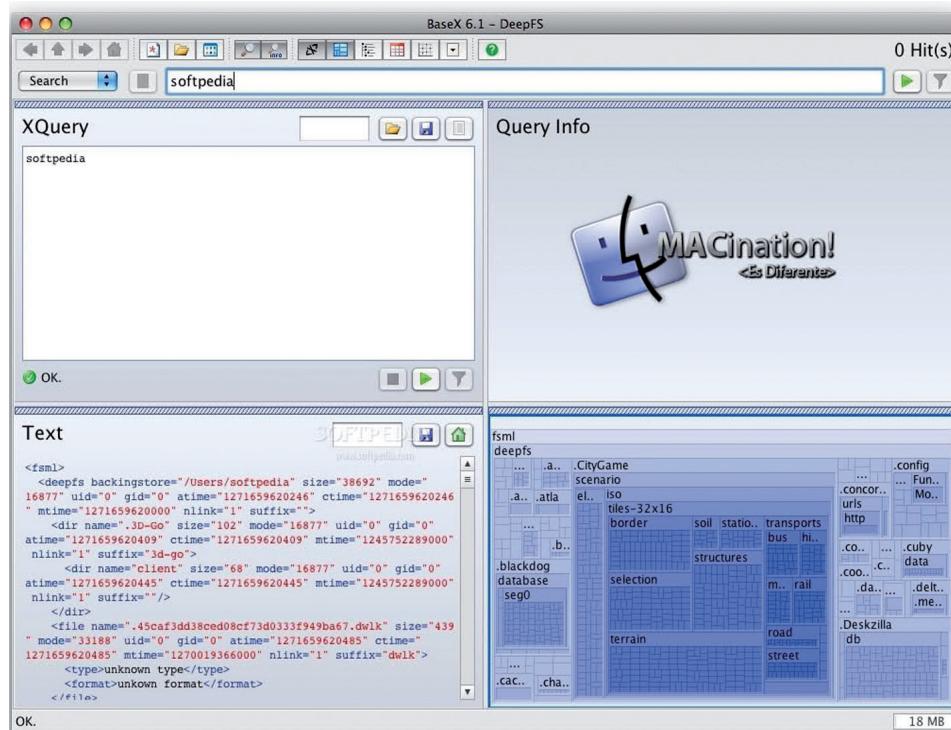
## 6.5 Almacenamiento XML nativo

Las **bases de datos nativas en XML** definen un modelo lógico para el documento XML además, almacenan y recuperan documentos igual que los XML. Este modelo de almacenamiento y recuperación debe incluir, por lo menos, atributos como PC-DATA y documentos en orden. Ejemplos de estos modelos son XPath, XML Infoset y aquellos que implican DOM y SAX 1.0.

Todas las bases de datos relacionales están centradas en los datos, pues lo éstas almacenan en sus campos son datos simples. Una base de datos nativa en XML no posee campos ni almacena datos simples, sino documentos XML, por lo que se las denomina *bases de datos centradas en documentos* (Figura 6.9).

El procesamiento de información en este tipo de bases de datos puede parecer muy ventajoso; sin embargo, no es realmente así debido al formato jerárquico en el que está almacenada la información. Muchas bases de datos necesitan que el usuario recupere todo el documento XML, que lo actualice con el XML API de su preferencia y que, posteriormente, vuelva a almacenar el documento en el repositorio. Eso es debido a que aún no existe un lenguaje estándar que permita la actualización, inserción o eliminación de elementos de un documento XML. Existe

el lenguaje Xupdate, que permite realizar actualizaciones en un documento XML, pero no es aún un estándar, por lo que numerosos gestores de este tipo de bases de datos no lo utilizan.



**Figura 6.9**

Las bases de datos nativas en XML almacenan la información en formato XML.

Hemos dicho que una base de datos nativa en XML almacena la información en formato XML, pero esto es solamente una deducción lógica, pues este tipo de bases de datos tienen repositorios (paquete de archivos) con un formato "tipo XML," como DOM o Infoset. En estos mismos repositorios se guardan los índices que se generan por cada documento XML almacenado.

Este tipo de bases de datos nativas no utiliza SQL como lenguaje de consulta, sino el lenguaje XPath. Algunas bases de datos permiten seleccionar los elementos que deberán tener índices, mientras otras indexan todo el contenido del documento. El problema que tienen las búsquedas en las bases de datos nativas es que no permiten realizar búsquedas demasiado complejas, como, por ejemplo, ordenar los resultados de la búsqueda bajo un determinado criterio, ya que XPath no fue creado para realizar búsquedas en bases de datos, sino simplemente para realizarlas en un solo documento.

En principio, la adaptación del modelo de una base de datos relacional a XML es relativamente sencilla. Una vez obtenido el modelo relacional podríamos realizar

una transformación de tablas a un documento XML con sólo crear una DTD y un documento XML bien formado (Figura 6.10). Veamos a continuación cuáles son los pasos que debemos realizar:

**Figura 6.10**  
Ejemplo de agregación de una  
DTD a un archivo XML.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE persona SYSTEM "persona.dtd">
<persona>
    <nombre nombre_pila="Edwin" apellido_paterno="Plachu"></nombre>
    <profesion>Ingeniero en Sistemas de computo</profesion>
</persona>
```

persona.dtd (archivo extra colocado en el mismo directorio)

```
<!ELEMENT persona (nombre, profesion)>
<!ELEMENT profesion (#PCDATA)>
<!ELEMENT nombre (#PCDATA)>
<!ATTLIST nombre nombre_pila CDATA #REQUIRED
                apellido_paterno CDATA #REQUIRED >
```

- Cada tabla del modelo relacional será un elemento dentro de la DTD:

```
<!DOCTYPE Libros[ <!ELEMENT Libros (libro)*>
]>
```

En este caso, tenemos una tabla, llamada “Libros” que almacena tuplas de tipo “libro” (cero o más libros). Cada tupla de la tabla se llama “libro”. A continuación es necesario indicar qué campos componen la tupla:

```
<!ELEMENT libro (Cod_Libro, Titulo, Autores, Editorial,
Edicion, ISBN, NumPaginas )>
```

- Cada columna de la tabla deberá establecerse como un tipo de dato almacenable (*char*, *integer*, etc.):

```
<!ELEMENT Cod_Libro (#PCDATA)>
<!ELEMENT Titulo (#PCDATA)>
<!ELEMENT Editorial (#PCDATA)>
<!ELEMENT Edicion (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT NumPaginas (#PCDATA)>
```

- Si existe una columna compleja (como puede ser la de "Autores"), deberá crearse un nuevo elemento. En este caso, un libro puede tener uno o más autores ('+'). Definiremos también los tipos de datos que debemos almacenar:

```
<!ELEMENT Autores (autor)+>
<!ELEMENT autor (Cod_Autor, Nombre, Apellidos, FechaNa-
cimiento)>
<!ELEMENT Cod_Autor (#PCDATA)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Apellidos (#PCDATA)>
<!ELEMENT FechaNacimiento (#PCDATA)>
```

- Si dos tablas están relacionadas entre sí a través de una tercera tabla de relación, se establecerá un nuevo elemento y se procederá con los pasos siguientes. La DTD será la siguiente:

```
<!DOCTYPE Libros [
  <!ELEMENT Libros (libro)*>
    <!ELEMENT libro (Cod_Libro, Titulo, Autores,
Editorial, Edicion, ISBN,
          NumPaginas)>
  <!ELEMENT Cod_Libro(#PCDATA)>
  <!ELEMENT Titulo (#PCDATA)>
  <!ELEMENT Editorial (#PCDATA)>
  <!ELEMENT Edicion (#PCDATA)>
  <!ELEMENT ISBN (#PCDATA)>
  <!ELEMENT NumPaginas (#PCDATA)>

  <!ELEMENT Autores (autor)+>
  <!ELEMENT autor (Cod_Autor, Nombre, Apellidos,
FechaNacimiento)>
  <!ELEMENT Cod_Autor (#PCDATA)>
  <!ELEMENT Nombre (#PCDATA)>
  <!ELEMENT Apellidos (#PCDATA)>
  <!ELEMENT FechaNacimiento (#PCDATA)>
]>
```

Veamos a continuación un ejemplo de documento XML ajustado a la anterior DTD:

```
<?xml version=>1.0>
<Libros>
<libro>
```

```
<Cod_Libro> 1 </Cod_Libro>
<Titulo>Don Quijote de la Mancha</Titulo>
<Editorial>Juan de la Cuesta</Editorial>
<Edicion>3</Edicion>
<ISBN>97884667 45840</ISBN>
<NumPaginas> 176</NumPaginas>
<Autores>
<autor>
<Cod_Autor>1</Cod_Autor>
<Nombre>Miguel</Nombre>
<Apellidos>de Cervantes Saavedra</Apellidos>
<FechaNacimiento>29/09/1547 </FechaNacimiento>
</autor>
</Autores>
</libro>
```

## Recuerda

Un documento XML  
siempre debe ir asociado  
a una DTD o a un XSD.

## 6.6 Herramientas de tratamiento y almacenamiento de información en formato XML

Los administradores disponen de un servidor de bases de datos orientado a objetos o postrelacional, lo que les permite realizar las tareas de almacenamiento y recuperación de objetos XML de forma más sencilla. Sin embargo, es preciso crear procesos intermedios de ensamblaje y desensamblaje.

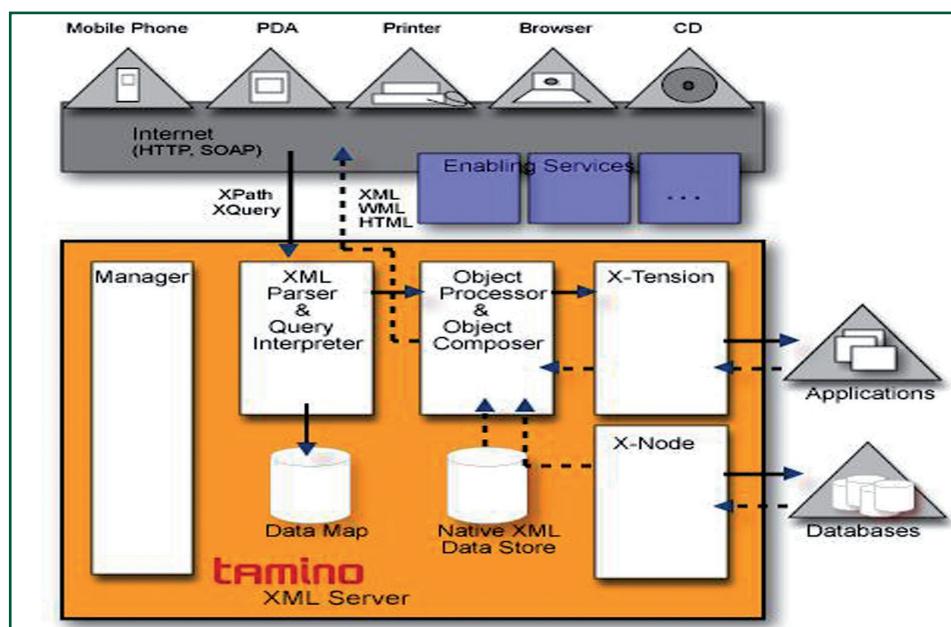
### 6.6.1 Tamino XML Server

Una de las plataformas para la administración de datos basada en XML y otros estándares de tecnologías de Internet es la **Tamino XML Server**. Su característica principal es el almacenamiento de documentos XML en forma nativa. El núcleo de Tamino XML Server es un servidor base que provee un mecanismo de elevada funcionalidad.

**Tamino XML Server** habilita servicios que proveen un amplio rango de herramientas y componentes necesarios para el desarrollo productivo de soluciones basadas en XML enfocadas a la web, como la publicación e intercambio de documentos electrónicos o en Internet. Tamino ha sido creada para el lenguaje XML, y es capaz de convertir en una tarea trivial el almacenamiento y recuperación de estructuras XML de cualquier tipo.

Veamos a continuación los principales componentes de **Tamino XML Server** que permiten llevar a cabo la realización de tareas gráficas:

- **Editor de Esquemas Tamino.** Archivo afín a los estándares de esquemas XML; es posible importar DTD y esquemas SQL y convertirlos en esquemas Tamino.
- **XML Loaders.** Tamino provee herramientas para cargar datos XML dentro de la base de datos desde una fuente externa.
- **X-Plorer.** Herramienta para consulta, exploración y manipulación de los contenidos de Tamino XML Server.
- **XQuery Tool.** Herramienta para consulta y manipulación (actualización) de los contenidos de Tamino XML Server (Figura 6.11).
- **Tamino Interactive Interface.** Interfaz basada en el explorador que permite definir y borrar colecciones y esquemas; cargar y borrar instancias XML de un esquema, y consultar una base de datos.
- **Almacenamiento.** Para realizar el almacenamiento de datos XML en Tamino, es preciso definir un esquema de acuerdo a los estándares que conforman un esquema XML. A continuación, se cargarán los datos, según este esquema, a través de la interfaz interactiva o mediante el cargador de datos Tamino.
- **Lenguajes de consulta.** Tamino soporta dos lenguajes de consulta: el Tamino X-Query (basado en el estándar XPath) y el lenguaje de consulta estándar Xquery, recomendado por el W3C Tamino.



**Figura 6.11**  
Ejemplo de arquitectura con el empleo de Tamino XML Server.

### 6.6.2 eXist

Otra plataforma para la administración de datos es **eXist**, un SGBD (Sistema de Gestión de Bases de Datos) XML nativo *Open Source* (de distribución gratuita) que posee las funcionalidades básicas de cualquier gestor nativo, además de integrar algunas técnicas avanzadas, como búsquedas de términos, búsquedas por proximidad y basadas en expresiones regulares (Figura 6.12).

**Figura 6.12**  
eXist, gestor de bases de datos  
nativas XML *open source*.



```
<SPEECH>
<SPEAKER>HAMLET</SPEAKER>
<LINE>Rest, rest, perturbed spirit!</LINE>
<STAGEDIR>They </STAGEDIR>
<LINE>So, gentle </LINE>
<LINE>Will my do companion you?</LINE>
<LINE>And so a man His + is</LINE>
<LINE>To his he in dire to you,</LINE>
<LINE>God will him not let us is together;</LINE>
<LINE>Call your fay you I p</LINE>
<LINE>The time is out of joint: O cursed spite,</LINE>
<LINE>That ever I was born to set it right!</LINE>
<LINE>Nay, come, let's go together.</LINE>
</SPEECH>
```

### 6.6.3 Oracle 11g XML DB

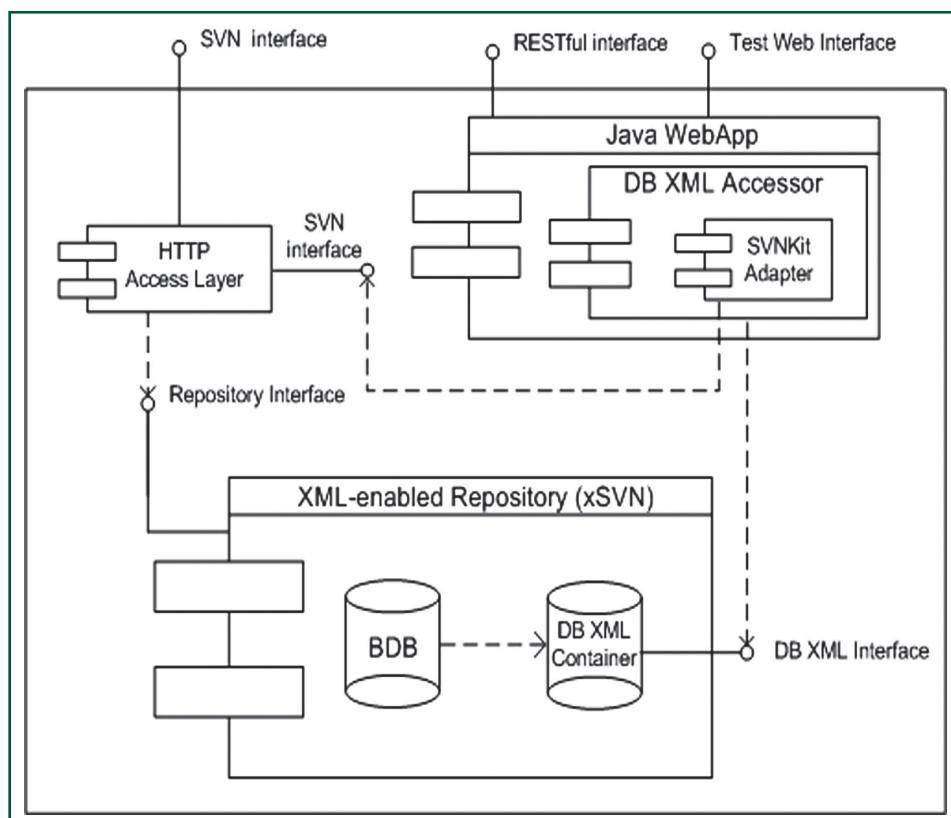
La plataforma **Oracle 11g XML DB** extiende la base de datos y comprende tanto las funcionalidades que soporta una base de datos nativa en XML (*Native XML Database*), como las funcionalidades de una base de datos relacional.

**Oracle 11g XML DB** provee métodos de acceso estándar para la navegación y consulta de documentos XML y absorbe los métodos de acceso estándar dentro de la BD Oracle 11g. La **Oracle 11g XML DB** permite almacenar, consultar, actualizar, transformar o procesar documentos XML, y, al mismo tiempo, da acceso SQL a los mismos datos XML. Permite, además, realizar operaciones XML sobre datos SQL.

### 6.6.4 dbXML

**dbXML** es una base de datos XML nativa, desarrollada con el lenguaje Java 2 Standard Edition versión 1.4, que puede operar en todas las plataformas en las que Java 2 Standard Edition 1.4 ha sido portado.

**dbXML** incluye un repositorio XML que permite al contenido XML ser organizado y administrado mediante la estructura Archivo/Carpeta/URL. Provee una infraestructura de almacenamiento, contenido y lenguaje de programación independiente para almacenar y administrar datos XML, además de métodos estándares para acceder y actualizar documentos XML, como FTP, HTTP y WebDAV. Permite el uso de API estándar para acceder y manipular el contenido XML mediante Java, C y PL/SQL (Figura 6.13).



**Figura 6.13**

Ejemplo de dbXML trabajando con XML.

## 6.6.5 Oracle XML DB

**Oracle XML DB** no es un servidor separado, su nombre indica un grupo de tecnologías relacionadas con el almacenamiento y recuperación de documentos XML disponibles en la base de datos Oracle (Figura 6.14).

El núcleo de Oracle XML DB es **XMLType**, un tipo de datos que soporta un conjunto de métodos construidos para el procesamiento de los documentos XML a través de su ciclo de vida. XMLType puede representar un documento XML como una instancia (de XMLType) en SQL. Las instancias de XMLType pueden ser guardadas mediante un almacenamiento estructurado o no estructurado.

## Para saber más

Para obtener mayor información acerca de las bases de datos, puedes dirigirte a la página <http://www.maestrosdelweb.com/principiantes/¿que-son-las-bases-de-datos/>.

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree shows a connection to 'XE\_LOCAL' containing various database objects like COUNTRIES, DEPARTMENTS, EMPLOYEES, etc. The main window displays an SQL statement in the 'Enter SQL Statement:' field:

```
SELECT XMLElement("EMPLEADO",
XMLAttributes(employee_id as "ID"),
XMLForest(department_id as "DEP",first_name as "FIRST_NAME"))
from employees;
```

The results pane shows the output of the query, which is a list of XML documents representing employee data. Each document contains an 'EMPLEADO' element with attributes 'ID' and 'DEP', and a child 'FIRST\_NAME' element.

```

<EMPLEADO>
<ID>100</ID>
<DEP>90</DEP>
<FIRST_NAME>Steven</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>101</ID>
<DEP>90</DEP>
<FIRST_NAME>Neena</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>102</ID>
<DEP>90</DEP>
<FIRST_NAME>Lex</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>103</ID>
<DEP>60</DEP>
<FIRST_NAME>Alexander</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>104</ID>
<DEP>60</DEP>
<FIRST_NAME>Bruce</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>105</ID>
<DEP>60</DEP>
<FIRST_NAME>David</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>106</ID>
<DEP>60</DEP>
<FIRST_NAME>Valli</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>107</ID>
<DEP>60</DEP>
<FIRST_NAME>Diana</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>108</ID>
<DEP>100</DEP>
<FIRST_NAME>Nancy</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>109</ID>
<DEP>100</DEP>
<FIRST_NAME>Daniel</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>110</ID>
<DEP>100</DEP>
<FIRST_NAME>John</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>111</ID>
<DEP>100</DEP>
<FIRST_NAME>Ismael</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>112</ID>
<DEP>100</DEP>
<FIRST_NAME>Jose Manuel</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>113</ID>
<DEP>100</DEP>
<FIRST_NAME>Luis</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>114</ID>
<DEP>30</DEP>
<FIRST_NAME>Den</FIRST_NAME>
</EMPLEADO>
<EMPLEADO>
<ID>115</ID>
<DEP>30</DEP>
<FIRST_NAME>Alexander</FIRST_NAME>
</EMPLEADO>
```

**Figura 6.14**

Ejemplo de servidor Oracle trabajando con datos XML.

Los datos en las tablas XMLType, y las tablas con columnas XMLType, pueden ser almacenados como un CLOB o nativamente mediante una estructura XML.

Los datos en las vistas de los XMLType pueden ser almacenados en tablas locales o remotas, a las que podemos acceder a través de DBlinks; tanto las tablas como las vistas XMLType pueden ser indexadas.

Existen varias opciones para acceder a los datos en el repositorio XML: HTTP, WebDAV y FTP, y SQL a través de Oracle Net Services, incluido JDBC. El lenguaje XPath reescrito es la clave para mejorar el desempeño de las sentencias SQL, que contienen expresiones Xpath, y convierten las funciones en sentencias SQL relacionales. Cuando los datos son almacenados en XMLType mediante un esquema XML y se realizan consultas con XPath, dichas consultas pueden ser potencialmente reescritas directamente en la subcapa de las columnas objeto-relacionales.

150

## Resumen

Para almacenar la información de documentos XML se pueden utilizar bases de datos relacionales. Una alternativa es la utilización de una base de datos nativa en XML (*native XML database*) que mejora la realización de búsquedas sobre los documentos almacenados.

El proceso de almacenamiento y lectura requiere mover datos entre un documento XML y una base de datos, y viceversa; los datos en la base de datos deben ser mapeados en la estructura del documento XML. Una estrategia es mapear el documento XML completo en una columna única en la base de datos, sin embargo otra posibilidad más compleja es mapear cada elemento en una columna correspondiente en la base de datos o la estructura del documento XML en la base de datos.

Se utilizan los lenguajes Xpath y Xquery para consultar, y en el caso de Xquery también actualizar la información almacenada como XML en las bases de datos. Xpath no es un lenguaje de programación, ni un lenguaje de estilos, sino un lenguaje que proporciona la sintaxis necesaria para acceder a elementos y atribuirlos dentro de un documento XML. Xpath está compuesto por símbolos que permiten definir las rutas y los distintos nodos (etiquetas y sus atributos).

Desde el punto de vista semántico, el lenguaje XQuery es semánticamente similar a SQL, aunque incluye algunas capacidades de programación. XQuery proporciona los medios para extraer y manipular información de documentos XML, o de cualquier fuente de datos que pueda ser representada mediante XML, como, por ejemplo, bases de datos relacionales o documentos ofimáticos.

XQuery utiliza expresiones XPath para acceder a determinadas partes del documento XML. También incluye la posibilidad de construir nuevos documentos XML a partir de los resultados de la consulta. Una consulta en XQuery es una expresión que lee una secuencia de datos en XML y devuelve, como resultado, otra secuencia de datos en XML. En XQuery las consultas pueden estar compuestas por cláusulas de hasta cinco tipos distintos. Las consultas siguen la norma FLWOR (leído como *flower*), siendo FLWOR las siglas de FOR, LET, WHERE, ORDER BY RETURN.

## Ejercicios de autocomprobación

**Indica si las siguientes afirmaciones son verdaderas (V) o falsas (F):**

1. El Sistema de Gestión de Bases de Datos (SGBD) es un sistema que nos ayuda a seleccionar, catalogar y ordenar la información que consideramos relevante, así como también a desechar aquella que esté duplicada o sea superflua.
2. Con el objetivo de manipular los datos entre un documento XML y una base de datos, y viceversa, los datos en la base de datos no deben ser mapeados a la estructura del documento XML.
3. La estrategia de inserción y extracción en el almacenamiento de información en XML se basa exclusivamente en el procedimiento básico de almacenar el documento completo en formato de texto como un gran objeto binario (BLOB) en una base de datos relacional.
4. El lenguaje XML utiliza técnicas de búsqueda de información en aquella información que se halla contenida en sus documentos, donde se destaca los lenguajes Xpath y Xquery.
5. El lenguaje XPath utiliza la estructura anidada de los documentos XML y permite presentar los documentos en una estructura de árbol, donde cada elemento del árbol se conoce como nodo.
6. Una consulta en XQuery es una expresión que lee una secuencia de datos en XML y devuelve, como resultado, otra secuencia de datos en XML.
7. FLWOR es la sigla de FOR, LET, WHERE, ORDER BY RETURN.

**Completa las siguientes afirmaciones:**

8. Los sistemas de \_\_\_\_\_ XML son útiles para ambos tipos de \_\_\_\_\_, ya que \_\_\_\_\_ está siendo ampliamente utilizado en sistemas que administran ambos tipos de \_\_\_\_\_. La mayoría de estos sistemas están concebidos para servir a uno de estos \_\_\_\_\_ de almacenamiento de datos mejor que al otro.

9. Los administradores disponen de un \_\_\_\_\_ de bases de datos orientado a objetos o postrelacional, lo que les permite realizar las tareas de \_\_\_\_\_ y \_\_\_\_\_ de objetos \_\_\_\_\_ de forma más sencilla. Sin embargo, es preciso crear \_\_\_\_\_ intermedios de ensamblaje y desensamblaje.

10. El lenguaje XQuery es semánticamente similar a \_\_\_\_\_, aunque incluye algunas capacidades de \_\_\_\_\_, que proporciona los medios para extraer y manipular \_\_\_\_\_ de documentos XML, o de cualquier fuente de \_\_\_\_\_ que pueda ser representada mediante XML, como, por ejemplo, bases de datos \_\_\_\_\_.

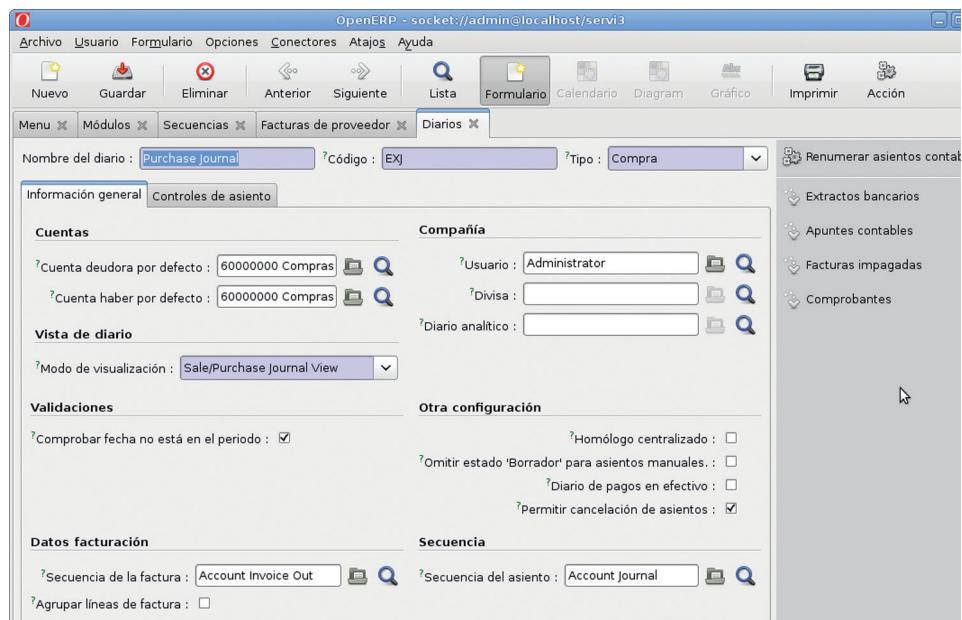
Las soluciones a los ejercicios de autocomprobación se encuentran al final de esta Unidad Formativa. En caso de que no los hayas contestado correctamente, repasa la parte de la lección correspondiente.

## 7. SISTEMA DE GESTIÓN EMPRESARIAL

Los **sistemas de gestión empresarial** (ERP, *Enterprise Resource Planning*) son sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocio asociadas a los aspectos operativos o productivos de una empresa y que eliminan complejas conexiones entre sistemas de distintos proveedores.

Este tipo de sistemas suele estar basado en una arquitectura modular, en la que cada módulo gestiona las funciones de un área empresarial diferente, como puede ser: nóminas, finanzas, gestión de proyectos, sistema de gestión geográfica, contabilidad, logística, stock, pedidos, etc. Dichas áreas realizan distintas funciones pero se interrelacionan entre sí al compartir información.

Es importante señalar que los sistemas ERP son integrales, es decir, que constituyen una agrupación de todos los módulos que los componen, que, a su vez, aúnan todos los procesos de gestión de la empresa (Figura 7.1).



**Figura 7.1**  
Ejemplo de software ERP.

Por ello, es importante disponer de un sistema integrado que dé soporte informático al tratamiento de la información, con la finalidad de evitar duplicidades y errores.

En esta unidad conoceremos la importancia estratégica del sistema informático elegido en función del elevado coste de la adquisición y adaptación del mismo.

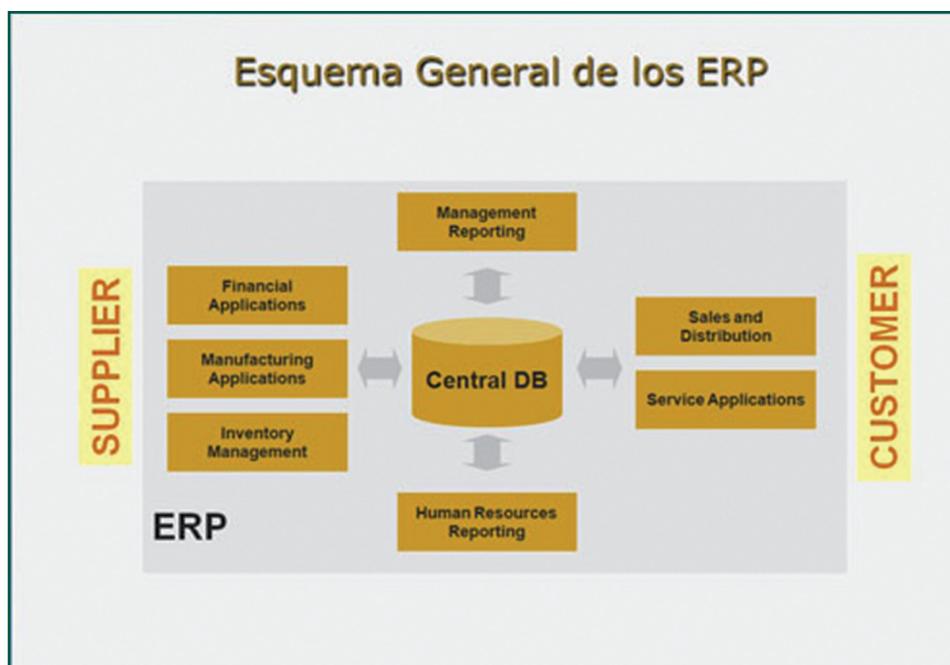
## 7.1 Instalación del sistema ERP

Dentro del sistema de gestión empresarial deberemos elegir adecuadamente un sistema informático que garantice la correcta gestión de la empresa. Para ello, deberemos analizar en profundidad los diversos ámbitos de la empresa, desde la producción, comercialización, planificación estratégica, hasta la gestión económica, la gestión de recursos humanos, los modelos de gestión de calidad, etc. Y, a continuación, proceder a la elección del sistema de gestión ERP más adecuado.

El **sistema ERP** (*Enterprise Resource Planning*) constituye la plataforma ideal para un software de gestión empresarial. Se trata de un software de gran envergadura, con una adaptación del cien por cien al funcionamiento de la empresa, aunque de elevado coste debido a la amplia implantación que proporciona (Figura 7.2).

La empresa puede disponer también del software CRM (*Customer Relationship Management*) para llevar a cabo la relación con los clientes, por tratarse de un sistema informático de apoyo a la gestión de las relaciones con los clientes, la venta y el marketing.

Otros productos profesionales que resultan muy adecuados para grandes empresas son: SAP Business Suite, Microsoft Business Solutions o Microsoft Dynamics, Oracle y, como software libre, OpenERP u Openbravo, algunos de ellos basados en lenguaje de marcas XML.



**Figura 7.2**  
Esquema general de un ERP.

### 7.1.1 Principales objetivos del sistema ERP

Veamos a continuación cuáles son los principales objetivos del sistema ERP:

- Optimización de los procesos empresariales, lo que comportará una reducción de costes y ahorro de tiempo.
- Acceso a toda la información de forma precisa (integridad de datos). Disponibilidad, con un solo clic, de toda la información digitalizada.
- Posibilidad de compartir información entre todos los componentes de la organización; al tratar los datos con la misma estructura facilita que ésta sea compartida.
- Eliminación de datos y operaciones innecesarias.

La instalación y configuración de una solución ERP requiere un conocimiento exhaustivo de la propia empresa (Figura 7.3).



**Figura 7.3**

Ejemplo de un sistema ERP con módulos.

Sin embargo, el coste temporal y económico que éste suponga se verá, sin duda, ampliamente recompensado porque optar por un ERP nos permitirá realizar un seguimiento íntegro y completo de nuestra empresa. Veamos a continuación cuáles son sus principales ventajas:

- Detección de puntos débiles en la gestión empresarial.
- Optimización de los procesos que se realicen en la empresa.
- Información centralizada, actualizada y coherente con los procesos internos que se ejecuten en cada momento.
- Acceso a toda la información de la empresa de manera modular y basada en roles.

- Compartición de la información relevante para cada uno de los procesos que lo requieran.
- Reducción de los costes asociados y reducción de tiempos.
- Análisis del estado de la empresa desde un punto de vista global.

El objetivo fundamental de un sistema ERP es proporcionar apoyo a los clientes del negocio y ofrecer tiempos rápidos de respuesta a sus problemas; así como facilitar el manejo de una información fidedigna que comporte la toma de decisiones acertadas y la disminución de los costos totales de operación. Las metodologías de implantación de los ERP no son a veces todo lo simples que se desearía, ya que intervienen múltiples facetas.

Es sabido que no existen recetas mágicas ni guiones explícitos para implantaciones exitosas; solamente trabajo bien realizado, una correcta metodología y tener en cuenta una serie de aspectos antes y durante el proceso de implantación del sistema ERP, inclusive cuando éste entre en funcionamiento. Por ello, antes, durante y después de la implantación de un ERP, es conveniente efectuar el siguiente análisis para determinar las necesidades de las distintas áreas de trabajo (Figura 7.4):



**Figura 7.4**

Las diferentes áreas de trabajo de una empresa deben estar relacionadas entre sí.

## Para saber más

Para mayor información sobre el software ERP, consulta <http://www.informatica-hoy.com.ar/software-erp/sistemas-ERP-de-software-libre.php>.

- Definición de resultados que podemos obtener con la implantación de un ERP.
- Definición del modelo de negocio.
- Definición del modelo de gestión.
- Definición de la estrategia de implantación.
- Evaluación de oportunidades para software complementario al producto ERP.
- Alineamiento de la estructura y plataformas tecnológicas.
- Análisis del cambio organizativo.
- Una visión completa de la solución a implantar.
- Implantación del sistema.
- Controles de calidad.
- Auditoría del entorno técnico y del entorno de desarrollo.
- *Benchmarking* de la implantación.

## 7.2 Identificación de flujos de información

En cualquier proceso en el que se producen intercambios o **flujos de información**, el impacto de las nuevas tecnologías es realmente muy importante porque de ellas dependerá que pueda redefinirse el proceso por completo. En otras palabras, gracias a las nuevas tecnologías toda la información puede digitalizarse y, por lo tanto, gestionarse automáticamente mediante los sistemas de información, y comunicarse entre ellos a coste cero a través de las redes (Intranet, Extranet e Internet) (Figura 7.5).



**Figura 7.5**

La información digitalizada fluye entre los ordenadores de la empresa que tienen acceso a la información mediante Internet, Intranet o Extranet.

En el ámbito empresarial, debemos destacar dos áreas en las que se generan importantes intercambios de información:

- Gestión de relaciones con los clientes (área clientes).
- Gestión de relaciones con los proveedores (área proveedores).

En la **gestión de relaciones con los clientes** existen numerosas oportunidades de mejora de procesos: consultas y presentación de información a clientes, automatización de fuerza de ventas, seguimiento de acciones de comunicación, personalización de mensajes, elaboración de ofertas personalizadas y muchísimas más. El software CRM (Gestor de Relaciones con los Clientes) es un ERP especializado en la relación con los clientes (Figura 7.6).

The screenshot shows the 'Datos del Cliente – GESPYMES' interface. At the top, it displays the URL: http://192.168.70.242/gremio\_carpinteros/pymes/infocliente2.php?tipo=1&cod=6. The header includes the GESPYMES logo and navigation links for 'DATOS PRINCIPAL', 'CLIENTES', 'PRESUPUESTOS', 'ORDEN DE TRABAJO', 'PARTES DE TRABAJO', 'PROVEEDORES', 'PRODUCTOS', 'ADMINISTRACIÓN', 'CONTABILIDAD', 'INFORMES FINANCIEROS', 'ABRIR NUEVA VENTANA', 'PORTADA', 'COPIAS DE SEGURIDAD', 'ACERCA DE', 'ACTUALIZAR GESPYMES', and 'CERRAR SESIÓN'. The main content area is titled 'Astillas Rotas, S.L.' and shows a table of 'Personas de contacto' (Contact Persons) with two entries: 'Manuel' (Cargo: Dpto. Administración, Teléfono: 6788 234 12) and 'Rafa' (Cargo: Economista). Below this, a section titled 'Facturación' (Billing) displays a table of total sales figures for June 2009, the second quarter, and the year-to-date, along with a link to view all invoices. At the bottom, a table titled 'Facturas pendientes de cobro' (Pending Invoices) lists several invoices with their details and amounts.

**Figura 7.6**

Ejemplo de gestión de clientes desde un CRM.

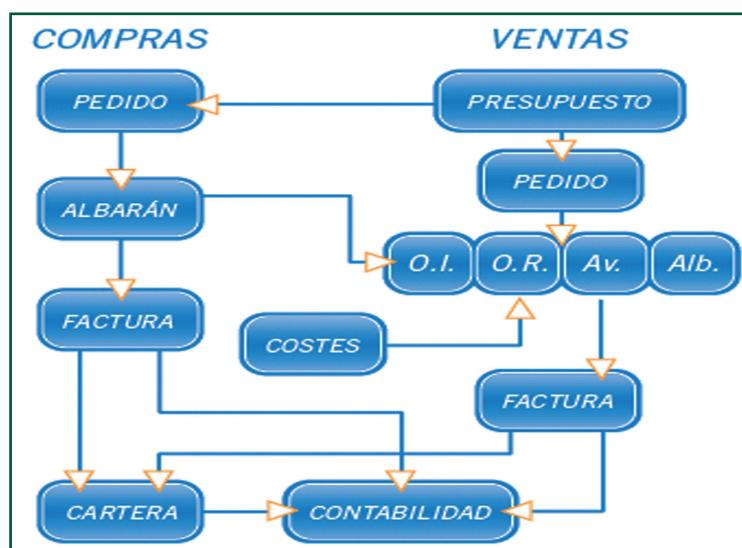
Son habituales las ofertas a partir de ciertos parámetros definidos por el cliente. Con una aplicación a través de Internet (un formulario, por ejemplo) que solicite los datos básicos para la elaboración del presupuesto y que sea accesible desde el sitio web de la empresa, el cliente con solo introducir su código de identificación podrá elaborar sus propios presupuestos. Además, al identificarse, el sistema reconocerá las tarifas y descuentos que se le aplican a dicho cliente y basándose en ellos se le ofrecerá su precio definitivo. Con ello se consigue una reducción de costes, ya que este proceso puede realizarse a coste cero con respecto a lo que supondría que dicho presupuesto solicitado por el cliente tuviera que ser elaborado por un empleado de la empresa. También se obtiene la posibilidad de que el cliente adquiera una mayor información sobre el presupuesto y le permita realizar un análisis de los distintos parámetros con el fin de poder, por ejemplo, comparar el precio de las distintas combinaciones.

En la **gestión de relaciones con los proveedores** (área proveedores), una de las opciones más interesantes por lo que respecta al empleo de las nuevas tecnologías de la información es la posibilidad de enviar información rápidamente a través de la **cadena de valor**. Dicho de otro modo, estas tecnologías permiten conocer en tiempo real la planificación de producción de los proveedores, pudiendo consultar directamente su sistema de información, gestionar los *stocks* junto con la producción y *stocks* de proveedores, conocer el estado de un pedido, mejorar la gestión logística conjunta, etc.

Con esto se conseguirá un control exhaustivo de entrada y registro de recepciones. La incorporación de nuevas tecnologías en el ámbito empresarial facilitará también la recogida de información e introducción al ERP para que éste pueda validar que existe realmente una orden de compra que responde a las especificaciones definidas. Asimismo permitirá a la empresa llevar a cabo un registro de sus facturas: introducción de éstas automáticamente a partir del ERP del proveedor a través de Internet.

### 7.2.1 Diagrama de flujo de información de funciones cruzadas

No obstante, la gestión de la relación con los clientes y la gestión de la relación con los proveedores no son los únicos flujos de información que se producen en una empresa; también dentro del proceso empresarial podríamos destacar el **diagrama de flujo de información de funciones cruzadas** para mostrar la relación existente entre un proceso empresarial y las unidades funcionales (como pueden ser los departamentos) responsables del proceso en cuestión (Figura 7.7).



**Figura 7.7**

La relación entre los departamentos de una misma empresa también genera sistemas de flujos de información.

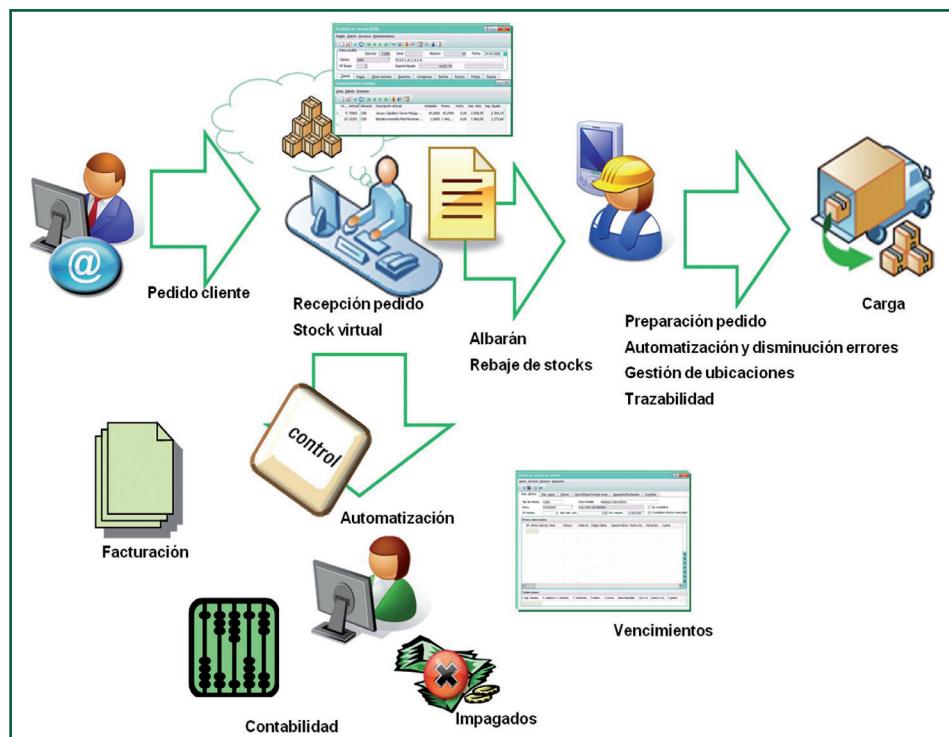
Para elaborar un diagrama de flujo correcto, deberemos utilizar un sistema lógico y práctico de cuáles son los objetivos del diagrama de flujo y, a continuación, estructurar la solución del problema independientemente del procesador que vayamos a utilizar.

## 7.3 Adaptación y configuración

Una peculiaridad que distingue a un software ERP de cualquier otro software empresarial es que se trata de un sistema integral, modular y adaptable. Analicemos a continuación estas características del software ERP.

### 7.3.1 Integrales

Los sistemas ERP permiten controlar los diferentes procesos de una empresa. Si tenemos en cuenta que todos los departamentos se relacionan entre sí, el resultado de un proceso será el punto de partida del siguiente (Figura 7.8).



**Figura 7.8**

Mediante un ERP pueden controlarse los distintos procesos de una empresa.

Cuando, por ejemplo, en una empresa un cliente hace un pedido, se crea una orden de venta que desencadena el proceso de producción, de control de inventarios, de planificación de distribución del producto, de cobros y, por

## Para saber más

Para obtener más información sobre los sistemas ERP *Open Source*, consulta <http://www.openerpsite.com/erp-openerp-modulos>.

supuesto, sus respectivos movimientos contables. Si la empresa no utiliza un ERP, necesitará disponer de varios programas que controlen todos los procesos mencionados; con el inconveniente de que, al no estar integrados, la información se duplicará, aumentará el margen de contaminación (sobre todo por errores de captura) y se creará un escenario favorable para las malversaciones. Sin embargo, con un ERP, el operador simplemente capturará el pedido y el sistema se encargará de todo lo demás, por lo que la información no se manipulará y estará protegida.

### 7.3.2 Modulares

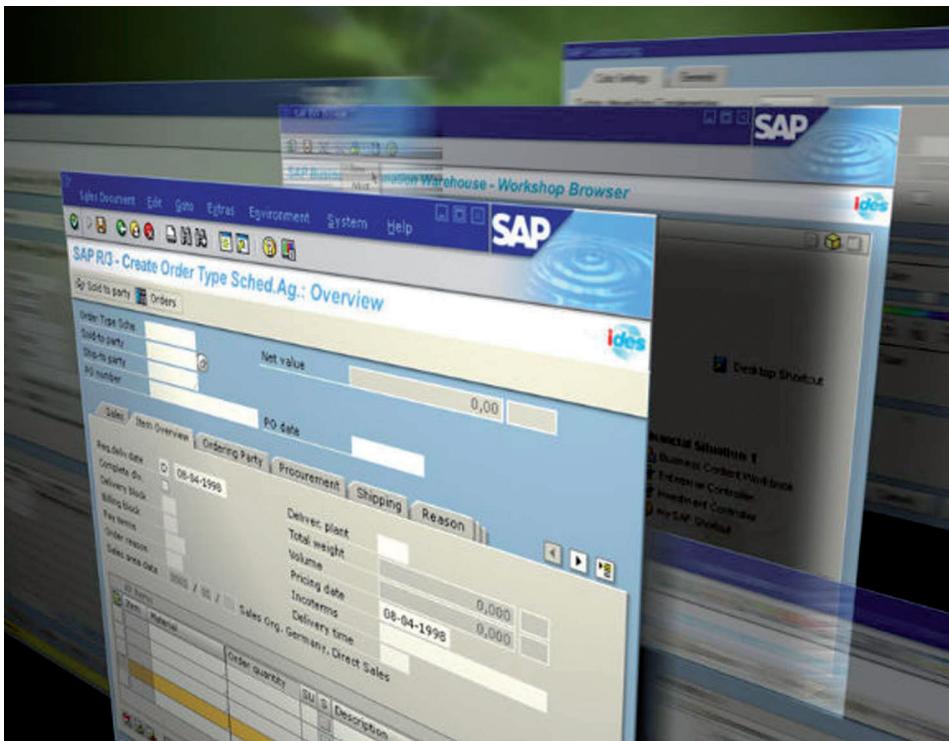
Los ERP entienden que una empresa es un conjunto de departamentos que se encuentran interrelacionados por la información que comparten entre sí y que se genera a partir de sus procesos. Una ventaja de los ERP, tanto económica como técnica, es que la funcionalidad está dividida en módulos (ventas, materiales, finanzas, control de almacén, recursos humanos) que pueden instalarse según los requerimientos del cliente.

### 7.3.3 Adaptables

Los ERP están creados para adaptarse a la idiosincrasia de cada empresa, lo que consiguen por medio de la configuración o **parametrización** (valor añadido fundamental con el que debe contar cualquier ERP para poder adaptarse a las necesidades concretas de cada empresa) de los procesos, de acuerdo con las salidas que se precisen de cada uno. Los ERP más avanzados suelen incorporar herramientas de programación de cuarta generación para el desarrollo rápido de nuevos procesos.

A la hora de elegir un sistema de gestión empresarial ERP, deberemos analizar las diferentes soluciones que hay en el mercado. En los últimos años, el tejido empresarial dedicado a los ERP ha crecido de manera sustancial y ha podido adaptar los módulos a modelos empresariales tan distintos como la administración pública, sistemas de telecomunicaciones, centros sanitarios, construcción, servicios energéticos, etc.

Hoy en día, el mayor proveedor de herramientas ERP es SAP (<http://www.sap.com>), con una cuota de mercado de un 30%; otros proveedores comerciales son Oracle E-Business Suite, Sage Logic Class/ERP X3 y Microsoft Dynamics. También afloran, cada vez más, los ERP *Open Source*, como OpenBravo, AbanQ, OpenERP y Fisterra, un ERP especializado para talleres mecánicos (Figura 7.9).

**Figura 7.9**

SAP, actualmente el mayor proveedor ERP del mercado.

## 7.4 Integración de módulos

El ERP puede tener menús modulares configurables para adaptarlos a las necesidades de cada usuario; eso significa que un ERP es un único programa con multiplicidad de bibliotecas, pero con acceso a una base de datos centralizada.

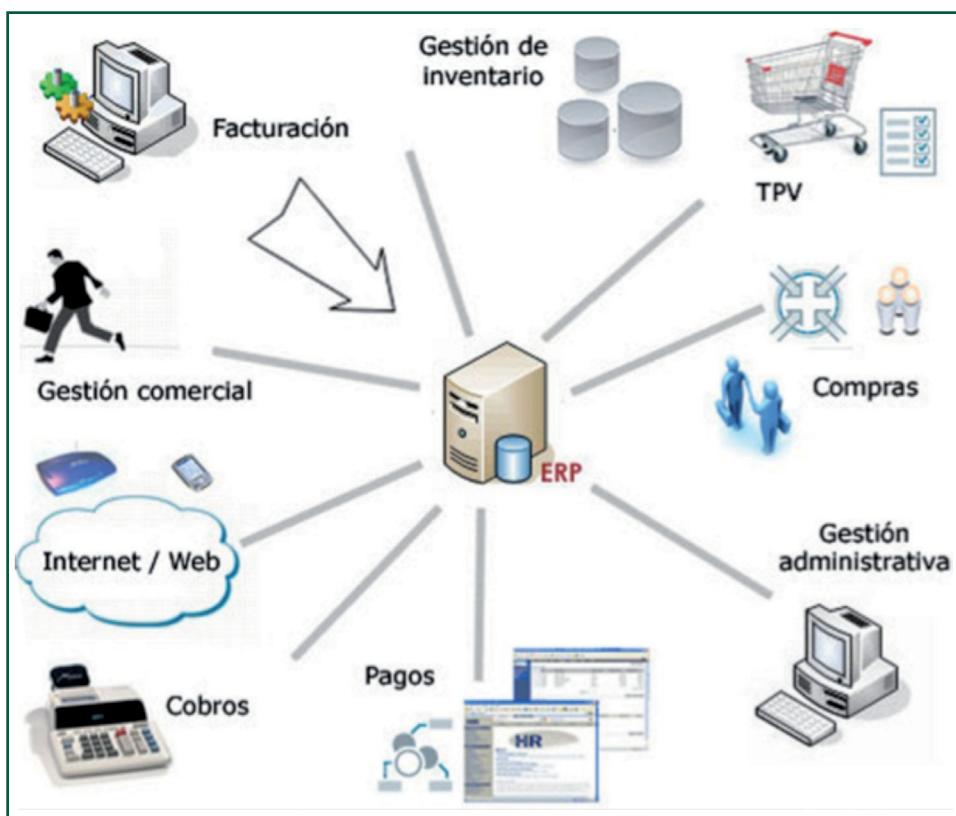
La tendencia actual es la de ofrecer aplicaciones especializadas para cada empresa, lo que se conoce como **versiones** o **aplicaciones sectoriales** especialmente indicadas para cumplir procesos de negocio de un determinado sector.

En ocasiones, las soluciones ERP son complejas y difíciles de implantar debido a que requieren un desarrollo personalizado para cada empresa, pero parten de la parametrización inicial de la aplicación que es común. Las personalizaciones requieren un gran esfuerzo en tiempo y, por consiguiente, también en dinero, para modelar todos los procesos de negocio de la vida real en la aplicación.

Un ERP consta de numerosos módulos que pueden interconectarse entre sí. De este modo, una única herramienta ERP puede dar servicio a empresas muy distintas y cambiar el conjunto de módulos activos y las relaciones entre ellos.

Los módulos, por lo tanto, son la parte central de una herramienta ERP porque permitirán almacenar, buscar, mostrar y representar cada proceso interno de la

empresa. Una herramienta ERP se compone de los siguientes módulos activos (Figura 7.10): **Finanzas** (base del ERP. Almacenamiento de cada transacción y su impacto administrativo; facilita también las auditorías), **Producción** (núcleo que se encarga de los movimientos físicos de los artículos), **Inventario y Logística** (módulo que se encarga del stock, almacenes, flujos de entrada y de salida), **Ventas y Marketing** (interfaz de interacción con los clientes) y **Recursos Humanos** (gestión de personal).



**Figura 7.10**

Un sistema de gestión empresarial puede estar formado por diferentes módulos, como las finanzas, los recursos humanos, las ventas, el stock, etc.

## 7.5 Elaboración de informes

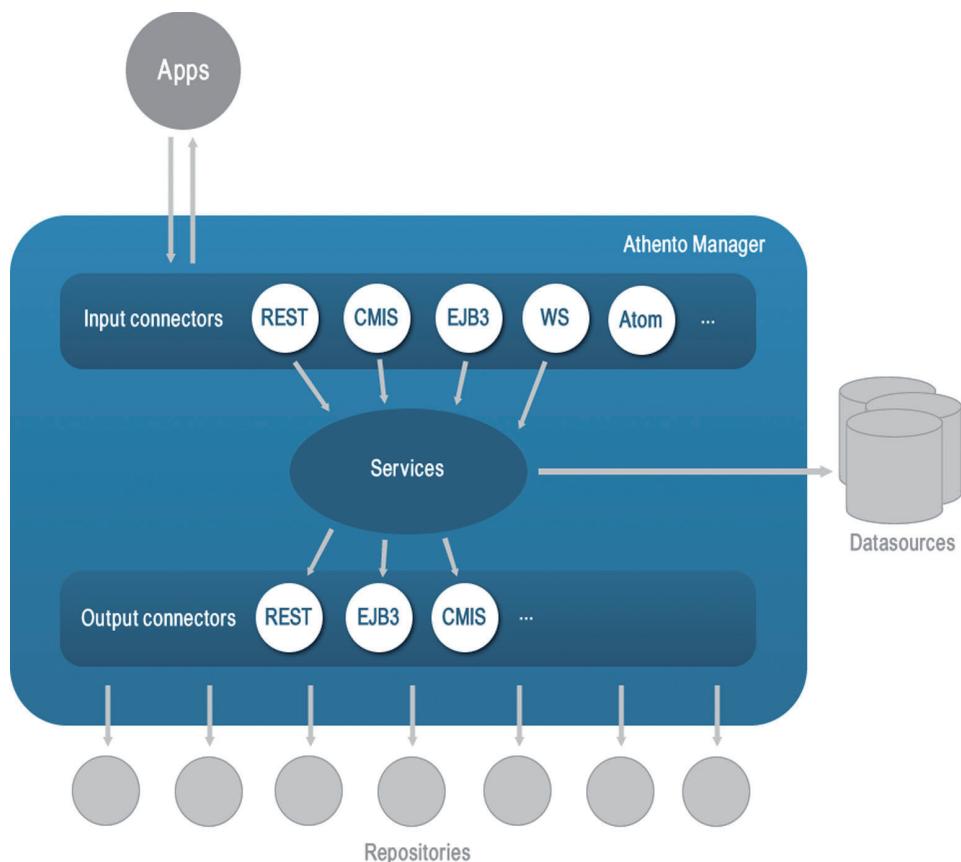
El almacenamiento de la información empresarial en un ERP permite centralizar toda la logística de negocio. Una vez aplicadas la instalación y las políticas de seguridad, los usuarios deberán utilizar este sistema para poder garantizar la coherencia de los datos y la continuidad del negocio. En según qué casos, por ejemplo, deberemos realizar informes sobre nóminas o facturación, por lo que el propio ERP deberá suministrar estos datos de manera fácil y sencilla mediante formularios predefinidos.

No todos los sistemas ERP tienen el mismo modo de generar formularios. Por ello, vamos a ver a continuación las principales técnicas de extracción de información

que pueden resultar independientes del sistema ERP implantado. Los distintos métodos de extracción de información son los siguientes: conectores, exportación a documentos ofimáticos; importación y exportación en ficheros XML; importación y exportación en ficheros CSV.

### 7.5.1 Conectores (importación y exportación)

Los **conectores** son pequeños módulos de software que permiten acceder a las bases de datos de los sistemas ERP (Figura 7.11). Según el ERP que se utilice, la programación de estos conectores se realizará en un lenguaje de programación como C/C++, Python, Shell, Java, etc. La creación de un conector requiere tener ciertos conocimientos de dicho lenguaje de programación y, sobre todo, de la arquitectura y estructura del propio ERP. Como es obvio, un conector que se desarrolle para un ERP no tiene por qué funcionar en otro ERP distinto; sin embargo, la programación adecuada del conector que se precise nos permitirá ser más ágiles a la hora de generar documentación no contemplada en la implantación inicial. Permite, además, adaptar el sistema ERP a nuevas necesidades.



**Figura 7.11**  
Ejemplo de conectores de sistemas ERP.

Cabe destacar que, si un pequeño conector programado por un usuario del ERP se publica como Open Source, todo usuario de cualquier otra empresa que use el mismo ERP podría utilizarlo. Esto genera una corriente muy positiva porque las necesidades de unos pueden ser cubiertas con las necesidades de otros y su coste puede ser cero.

### 7.5.2 Exportación a documentos ofimáticos

Con respecto a la **exportación a documentos ofimáticos**, todo dependerá del ERP implantado y si éste posee la característica de exportación de información. Muchos de ellos permiten generar un documento tipo procesador de textos que contenga el informe solicitado. Si el informe que se desea obtener no es completo, siempre se puede partir de uno parcial y completarlo manualmente con los datos que no exporta en un inicio. También pueden solicitarse datos en bruto tal y como se haría con una base de datos normal; es decir, generando un documento tipo hoja de cálculo con el que trabajar posteriormente para extraer la información buscada.

### 7.5.3 Importación y exportación en ficheros XML

En los últimos tiempos, los sistemas ERP suelen utilizar importadores de otros sistemas ERP para realizar una transición más sencilla. Dichos importadores hacen una conversión previa a XML con el fin de reconocer entidades y atributos que le son conocidos; al fin y al cabo, la estructura de la información a almacenar en una nómina o en una factura suele ser muy similar en un sistema ERP o en otro. Si el sistema ERP puede importar y exportar a XML, nada impide que utilicemos esos datos exportados para generar una hoja XSL y aplicar las transformaciones necesarias con el fin de obtener la información requerida (Figura 7.12)



**Figura 7.12**

La exportación de la información contenida en un ERP puede realizarse mediante XML.

### 7.5.4 Importación y exportación en ficheros CSV

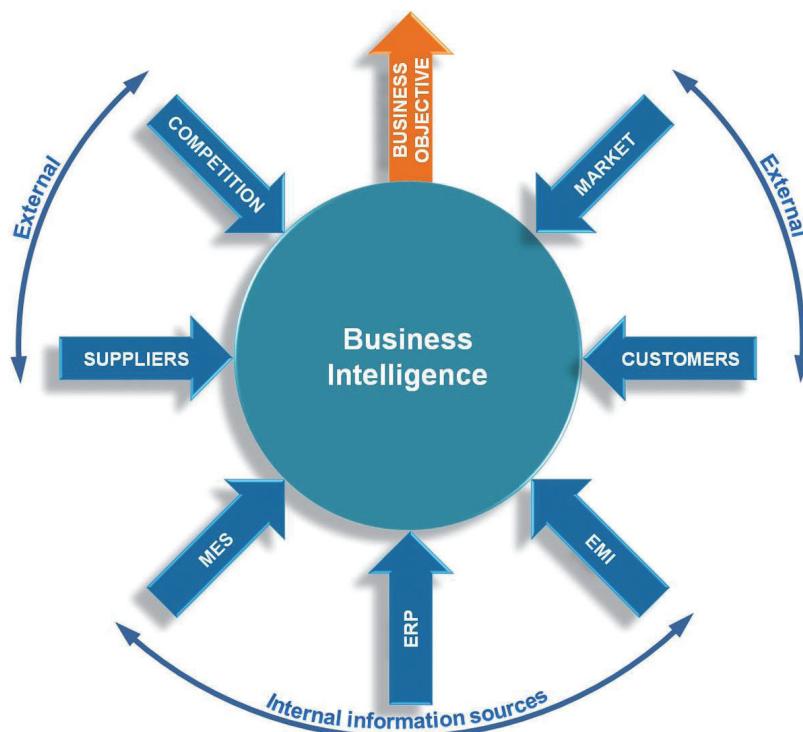
Por lo que respecta a la **exportación en ficheros CSV** (*Comma-Separated Values*), se trata de un método notablemente utilizado en los últimos años. Cualquier fichero CSV podría leerse con una hoja de cálculo. Por lo tanto, el simple conocimiento de creación de fórmulas o *scripts* podría facilitar la extracción de la información que inicialmente no nos permite extraer un formulario predeterminado del ERP.

### 7.5.5 Business Intelligence

El término **Inteligencia Empresarial** (*Business Intelligence*) hace referencia a la utilización de datos en una empresa cuya finalidad es la de facilitar la toma de decisiones. Abarca la comprensión del funcionamiento actual de la empresa y se anticipa a los acontecimientos futuros con el objetivo de ofrecer un conocimiento exhaustivo para respaldar las decisiones empresariales (Figura 7.13).

#### Recuerda

La extracción de la información en documentos XML puede utilizarse para un tratamiento informático de la información mediante las bases de datos XML nativas, XSL, etc.



**Figura 7.13**  
Ejemplo de relaciones en *Business Intelligence*.

Las herramientas de inteligencia se basan en la utilización de un sistema de información inteligente, que se configura con los datos extraídos de aquellos de producción, con información relacionada con la empresa y sus ámbitos, y con datos económicos.

Las herramientas y técnicas ELT (*Extract, Transform, Load*), encargadas de extraer, transformar y cargar, extraen, depuran y preparan los datos de las distintas fuentes (homogeneización) para luego cargarlos en un almacén de datos.

La vida o el período de éxito de un software de inteligencia de negocios dependerá del nivel de éxito de lo que haga en beneficio de la empresa que lo utilice: si la empresa es capaz de incrementar su nivel financiero y administrativo, la inteligencia de negocios utilizada prometerá larga vida; por el contrario, el software será sustituido por otro que aporte mejores resultados.

Por último, las herramientas de inteligencia analítica posibilitan el modelado de las representaciones con base en consultas para poder crear un cuadro de mando integral que sirva de modelo para la presentación de informes. Estas herramientas pueden ser módulos independientes del ERP instalado en la empresa o productos de otros proveedores de software que se integren con el ERP de la empresa.

## 7.6 Planificación, implementación y verificación de la seguridad

Cualquier entorno empresarial que haya informatizado sus procesos, debe contar con un sistema que permita conservar los datos de manera íntegra durante todo su período útil. Por tanto, deberá evitarse que se produzca una serie de riesgos que comprometan la estabilidad del sistema y la propia continuidad del negocio empresarial.

Para aumentar la seguridad de los sistemas, es preciso analizar los riesgos a los que puede estar sometido nuestro entorno informatizado (sea ERP o no). Por ello, los riesgos pueden clasificarse en riesgos físicos y riesgos lógicos.

### 7.6.1 Riesgos físicos

Los **riesgos físicos** son aquellos que pueden suceder cuando fallan algunos componentes electrónicos de nuestro sistema informático. Fallos como la rotura de los discos duros y la rotura de memorias son los más habituales; y las causas pueden ser múltiples, desde el propio desgaste del aparato hasta agresiones externas, como altas temperaturas, robo de componentes, etc.

Para evitar los riesgos físicos, es preciso ubicar los sistemas informáticos en habitaciones acondicionadas para su función. Deben estar climatizadas para impedir el sobrecalentamiento de los sistemas y, a ser posible, estar dotadas de sistemas de detección de incendios, con medidas preventivas y correctivas. El control de ac-

ceso a la sala debe estar protegido para evitar robos y hurtos de material; y, sobre todo, es preciso aplicar una política de copias de seguridad que permita salvaguardar los datos importantes en el caso de imprevistos (Figura 7.14).



**Figura 7.14**

Es importante que la información tratada por los sistemas de gestión ERP sea centralizada y el administrador se encargue de realizar un *backup* adecuado con tal de garantizar la información.

### 7.6.2 Riesgos lógicos

Los **riesgos lógicos** son aquellos que pueden suceder cuando existe una política inadecuada de autenticación en los sistemas informáticos: accesos no autorizados; *bugs* y errores en el sistema operativo, o en el software utilizado; intrusiones externas a través de la red; o una instalación de software incompatible o con una funcionalidad oculta (troyano) que limite el correcto funcionamiento del sistema informático.

Una gestión de los riesgos adecuada de un sistema informático garantizará una continuidad de la línea de negocio ante cualquier contingencia que pueda suceder. Por ello, es importante realizar la identificación de los riesgos durante todo el período vigente del sistema informático, es decir, antes, durante y después de su instalación.

En el ámbito de los ERP, los datos que almacenan las bases de datos deberían ser la principal fuente de preocupación. Una buena política de copias de seguridad

## Recuerda

**Es muy importante realizar un plan de mantenimiento que incluya la periodicidad y el tipo de copias de seguridad que queramos realizar en el sistema.**

(Figura 7.15) que garantice el almacenamiento en otras ubicaciones distintas fuera del edificio de la empresa permitirá, si el sistema sufre cualquier daño, restaurarlo en cualquier otro sitio con los datos salvados hasta el momento; de todas formas, en caso de que se dañe cualquiera de los funcionamientos, deberemos aplicar el **plan de contingencia**, que marcará los procedimientos alternativos a seguir para recuperar el normal funcionamiento de la empresa.

**Figura 7.15**

Muestra de cómo se realizan las copias de seguridad de forma centralizada.



### 7.6.3 Plan de contingencia

El plan de contingencia es aquel que se aplica en caso de que, por accidente, tanto interno como externo, se vea dañado cualquiera de los funcionamientos de la empresa. Dicho plan ha derivado en lo que se conoce hoy en día como **Plan de Continuidad del Negocio**, que es aquél que establece que una empresa debe preparar sus planes acerca de cómo actuar de cara a futuros acontecimientos, lo que supone un avance a la hora de superar cualquier eventualidad que pueda acarrear pérdidas. La principal función de un Plan de Continuidad del Negocio es la de garantizar la continuidad de las operaciones de la empresa. Para poderlo elaborar, deberemos tener en cuenta sus cuatro etapas:

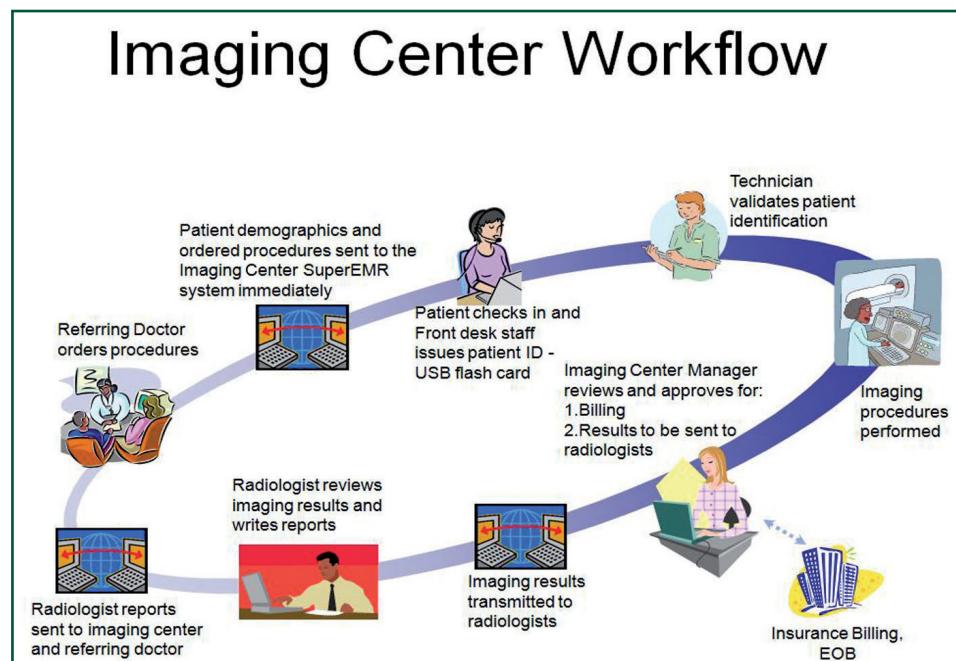
1. Evaluación
2. Planificación
3. Pruebas de viabilidad
4. Ejecución

Las tres primeras etapas hacen referencia al componente preventivo, mientras que la última etapa se refiere a la ejecución del plan, después de haber ocurrido el siniestro. En caso de siniestro, una buena planificación aumenta la capacidad de organización y constituirá el punto de partida para poder dar una respuesta.

## 7.7 Integración con aplicaciones ofimáticas

Los ECM (*Enterprise Content Management*) son soluciones integrales para la gestión de los contenidos empresariales. Dichas soluciones se componen de diferentes módulos que, juntos, pueden ofrecer una solución global, aunque también pueden utilizarse de manera independiente. Los módulos principales de un ECM son: DM (*Document Management*); RM (*Record Management*); WCM (*Web Content Management*); BPM (*Business Process Management*); Workflow, y KM (*Knowledge Management*):

- **DM (Document Management).** Herramientas que permiten la gestión y el almacenamiento de gran cantidad de documentos. Disponen de potentes herramientas de búsqueda y recuperación de información que posibilitan la localización de documentos de forma ágil y rápida. Estas soluciones de gestión documental realizan gestión de versiones, relación de documentos entre sí e integración con las aplicaciones ofimáticas, así como garantizan el acceso seguro a los documentos, permiten workflows, incluyen firma electrónica, etc.
- **RM (Record Management).** Herramientas aptas para la gestión de aquellos documentos que la empresa desea conservar (normas jurídicas, documentos legales, administrativos, corporativos, etc). Gestionan tanto la documentación electrónica como los documentos que se conservan en papel.
- **WCM (Web Content Management).** Herramientas para crear y administrar los contenidos de las páginas web. Permiten gestionar, de forma independiente, el contenido y el diseño, así como controlar los niveles de acceso a los contenidos y su publicación en la web.
- **BPM (Business Process Management).** Herramientas cuyo objetivo es gestionar y administrar los procesos de negocio y sus flujos de información. Se encargan de representar, controlar, monitorizar y optimizar los procesos de negocio de la empresa identificando sus flujos de trabajo y la documentación asociada a dichos procesos.
- **Workflow.** Herramientas para la gestión de los flujos de documentación. Permiten la generación de flujos de trabajo con diversas tareas de manera secuencial o en paralelo, donde los documentos son parte esencial de los mismos (Figura 7.16).



**Figura 7.16**  
Muestra de la herramienta Workflow para gestionar los flujos de información.

- **KM (Knowledge Management).** Herramientas que permiten buscar, rastrear, analizar, administrar, proteger, compartir e integrar aquella información que resulta ser de interés para la empresa. Se encargan de indexar y localizar la documentación buscada en diversas fuentes (documentos, correos electrónicos, páginas web, bases de datos, etc.).

Como hemos dicho ya, una de las formas de gestionar y almacenar documentos suele ser la de generar informes en PDF para una perfecta impresión. También pueden generarse archivos en Word o Excel que después pueden modificarse antes de ser enviados a un cliente por carta, mail o fax de forma automática.

Un sistema ERP puede integrarse con software comercial:

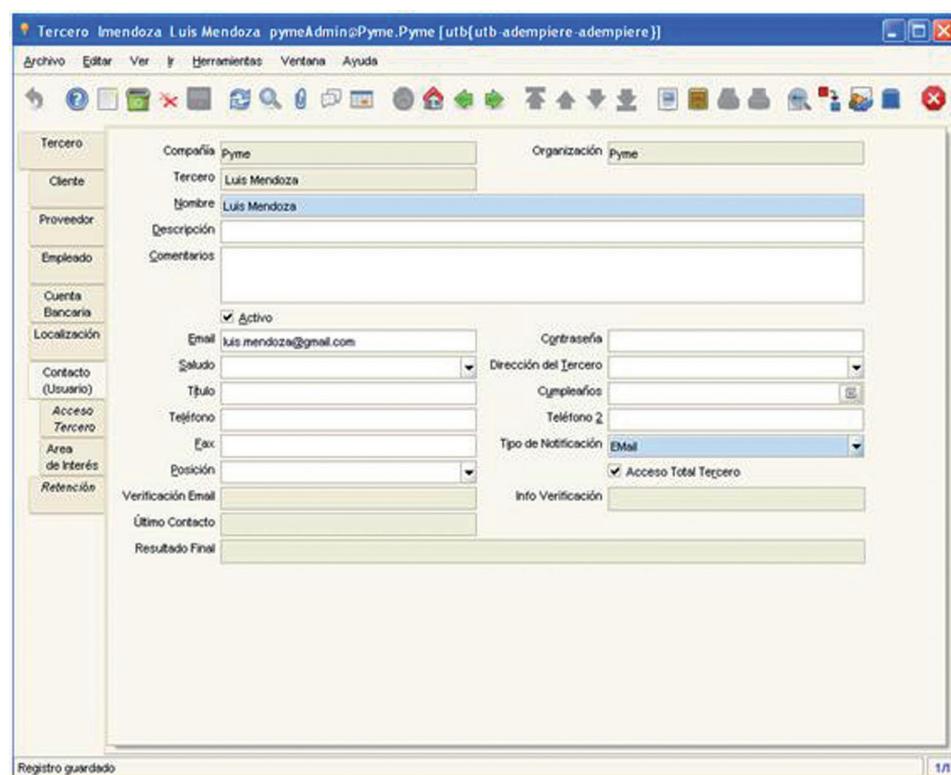
- Visualización con Adobe Reader (PDF).
- Importación y exportación de Microsoft Office.
- Exportación a Excel (o CSV).

Existen también conectores con software libre, como:

- OpenOffice: la suite ofimática de código abierto desarrollada por Sun Microsystems/Oracle.
- Mozilla Thunderbird: cliente de correo electrónico de los creadores de Mozilla Firefox.
- Google maps: servidor de aplicaciones de mapas en la web (gratuito).

- Jasper Reports (iReport): herramienta de creación de informes Java libre.
- Magento: aplicación de comercio electrónico online.
- oCommerce: aplicación de comercio electrónico online.
- Joomla: gestor de contenidos (integración parcial a través de xml-rpc).
- Dia UML: software de diagramas para implementación de módulos.

A veces, un usuario desea integrar parte del sistema ERP en algún módulo con el fin de poder utilizarlo conjuntamente con el correo electrónico. Una integración con el gestor de correo permitirá, por ejemplo, definir plantillas de envío para cada uno de los clientes con el objetivo de poder realizar tareas de envío y recepción de correo y adjuntar documentación personalizada para cada envío (Figura 7.17). Para que esto sea posible, es necesario instalar dos módulos: el primero, instalado en el propio ERP, como un conector; y el segundo, en el gestor de correo electrónico que utilice el usuario que exige esta funcionalidad.



**Figura 7.17**

Un sistema ERP puede integrarse con un gestor de correo para realizar tareas masivas de mailing a sus clientes.

### 7.7.1 Tablas dinámicas de Microsoft Excel

Además de los diferentes sistemas de software de gestión ERP o CRM, recientemente se suelen utilizar diversos tipos de herramientas internas que permiten consultar y explotar la información dentro de la empresa. Una de estas herramientas



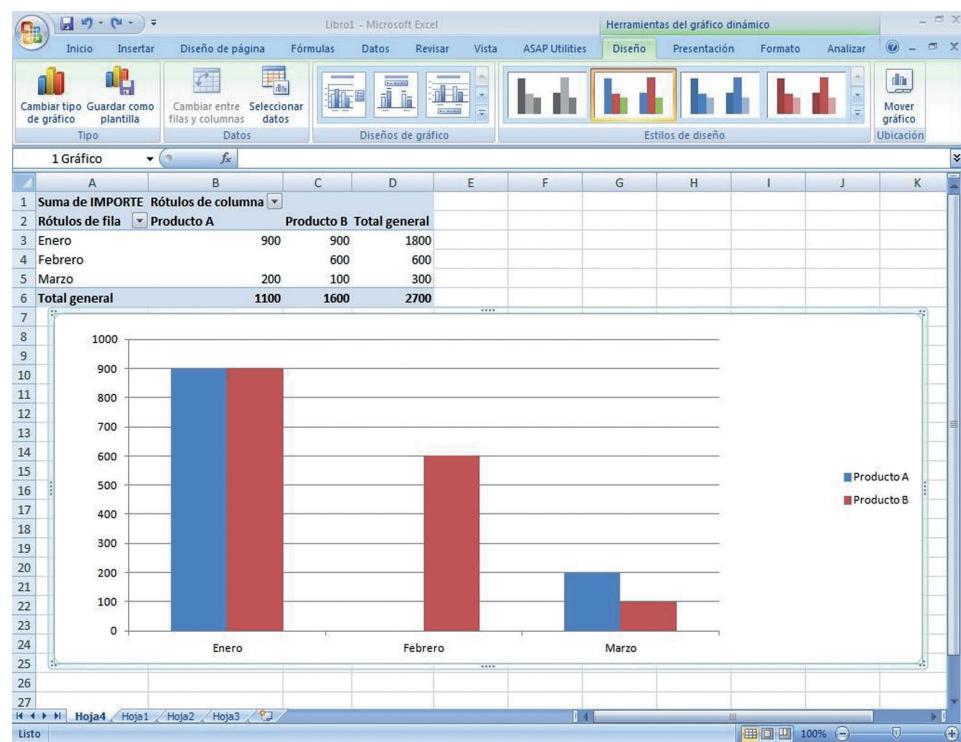
**Para ampliar este tema, puedes ver el videotutorial *Tablas dinámicas de Microsoft Excel*, que encontrarás en el Campus online.**

## Para saber más

**Microsoft nos ofrece todo el software necesario, tanto en términos de servidor como en términos de cliente, para implementar un sistema de *Business Intelligence* si unimos los productos Microsoft SQL Server y Microsoft Office.**

tas son las **tablas dinámicas de Microsoft Excel**; que, en su versión 2007, ya eran muy potentes, pero que en la versión 2013 resultan fundamentales para poder manejar cantidades enormes de datos con seguridad y rapidez (Figura 7.18). Se trata de una herramienta que permite trabajar con los datos de origen, siempre en formato tabla, para analizarlos de una forma flexible y dinámica; en otras palabras, podemos tener una visión de los datos desde diferentes perspectivas: temporal, de calidad y de cantidad. Constituye, por lo tanto, un complemento perfecto a la hora de explotar los datos desde un ERP o un CRM, porque posee una serie de características que hacen que sea una herramienta de *Business Intelligence*. Veamos a continuación cuáles son algunas de las características principales de las tablas dinámicas de Microsoft Excel:

- Permite realizar cuadros de análisis tipo ABC, en porcentajes y en valores absolutos.
- Permite poner indicadores en los datos integrados desde la hoja de cálculo al de las tablas dinámicas y viceversa.
- Ayuda a crear todo tipo de gráficos con enormes posibilidades de filtros y presentaciones.
- Presenta los datos con minigráficos de tendencias y opciones para crear cuadros de mando de segmentación de datos.
- Facilita con la opción “otras fuentes” la posibilidad de unir datos de tablas de diferentes orígenes desde el propio Excel y sin restricciones de cantidad de datos.



**Figura 7.18**

Muestra de una tabla dinámica de Microsoft Excel.

## 7.8 Exportación de información

Un sistema ERP puede exportar información a documentos ofimáticos siempre y cuando posea la característica que se conoce con el nombre de **exportación de información** (Figura 7.19).



**Figura 7.19**

Hoy en día, puede accederse fácilmente a la información desde cualquier ordenador conectado a Internet.

Muchos ERP permiten, a partir de un formulario interno del ERP, generar un documento tipo procesador de textos que contiene el informe solicitado. Si el informe no es completo, podremos tomar uno parcial y completarlo manualmente con los datos que no exporta al inicio.

Otra opción sería solicitar los datos en bruto, tal y como se haría con una base de datos normal, generando una hoja de cálculo con la que poder trabajar para extraer la información requerida.

Por lo que respecta a la **exportación CSV** (*Comma-Separated Values*), ésta es ampliamente utilizada. CSV son documentos en formato abierto y sencillo para representar datos en forma de tabla, en la que las columnas se separan por comas (o punto y coma si la coma es el separador decimal) y las filas por saltos de línea. Los campos que contengan una coma, un salto de línea o una comilla doble deben ser encerrados entre comillas dobles.

A continuación, veamos un ejemplo de datos CSV:

Año,Marca,Modelo  
1997,Ford,Fiesta  
2000,Peugeot,207

Cualquier fichero CSV puede ser leído con una hoja de cálculo; el simple conocimiento de creación de fórmulas o *scripts* puede facilitar la extracción de información que, inicialmente, no nos permite extraer un formulario predeterminado del ERP.

## Resumen

Los sistemas de gestión empresarial (ERP, *Enterprise Resource Planning*) son métodos de gestión de información que proporcionan soluciones informáticas integradas a todas las áreas de la empresa.

Durante el proyecto de implementación de un sistema de gestión empresarial, deben analizarse los procesos de la empresa, pues seguramente dicho sistema va a impactar en dichos procesos, modificándolos e incluso redefiniéndolos por completo. Para su aplicación se utilizan los diagramas de flujo de manera formal, es decir, sin ser explícitos en la tecnología.

Los ERP deben ser sistemas integrales, modulares y adaptables. Integrales porque permiten controlar los diferentes procesos de manera integrada, en los que un dato sólo se introduce solo una vez; modulares porque los ERP entienden que una empresa es un conjunto de departamentos interrelacionados; y adaptables porque los ERP están creados para adaptarse a la idiosincrasia de cada empresa mediante la parametrización o configuración de los componentes.

Los módulos permiten almacenar, buscar, mostrar y representar cada proceso interno de la empresa. Una herramienta ERP debería tener, como mínimo, el módulo de Finanzas, el de Producción, el de Logística, el de Marketing y Ventas, y el de Recursos Humanos.

Un ERP debe disponer de herramientas para la visualización de la información que permitan llevar a cabo las operaciones y la toma de decisiones. Existen estrategias que consisten en la exportación de datos a otras aplicaciones, como hojas de cálculo y PDF. En el caso concreto de ayuda a la toma de decisiones, existen aplicaciones que permiten estructurar los datos de los diferentes módulos y mostrarlos, de manera resumida, para este cometido. Estas herramientas se denominan Inteligencia de Negocio o Inteligencia Empresarial (*Bussiness Intelligence*).

Otro aspecto que debemos analizar de los sistemas de gestión empresarial es la seguridad. Debemos contemplar los riesgos físicos, es decir, los fallos en los componentes físicos del sistema informático, y los lógicos, o sea aquellos que provienen de autenticaciones falsas en el sistema.

Un complemento necesario a los ERP son las soluciones integrales para la gestión de los contenidos empresariales (*ECM Enterprise Content Management*). Estas soluciones incluyen diferentes módulos que pueden unirse para dar una solución global, o bien utilizarse de forma independiente. Los principales módulos de un ECM son: la Gestión de Documentos (DM), Gestión de Contenidos Web (WCM), Gestión de Archivos (RM), Gestión de Procesos de Negocio (BPM), Workflow, etc.

## Ejercicios de autocomprobación

**Indica si las siguientes afirmaciones son verdaderas (V) o falsas (F):**

1. Dentro del sistema de gestión empresarial se debe elegir un sistema informático que garantice la correcta gestión de la empresa. Para ello, se debe analizar en profundidad los diversos ámbitos de la empresa, desde la producción, comercialización, planificación estratégica, hasta la gestión económica, la gestión de recursos humanos, los modelos de gestión de calidad, etc.
2. Uno de los principales objetivos del sistema ERP es la posibilidad de compartir información entre todos los componentes de la organización.
3. En cualquier proceso en el que se producen intercambios o flujos de información, el impacto de las nuevas tecnologías es muy importante porque de ellas dependerá que pueda redefinirse el proceso por completo.
4. En la gestión de relaciones con los proveedores (área proveedores), una de las opciones más interesantes por lo que respecta al empleo de las nuevas tecnologías de la información es la imposibilidad de enviar información rápidamente a través de la cadena de valor.
5. Una peculiaridad que distingue a un software ERP de cualquier otro software empresarial es que se trata de un sistema integral, modular y adaptable.
6. Los riesgos lógicos son aquellos que pueden suceder cuando existe una política adecuada de autenticación en los sistemas informáticos: accesos no autorizados, bugs y errores en el sistema operativo, o en el software utilizado.
7. Para analizar la seguridad de los sistemas de gestión empresarial se debe contemplar los riesgos físicos, es decir, los fallos en los componentes físicos del sistema informático, y los lógicos, o sea aquellos que provienen de autenticaciones falsas en el sistema.

**Completa las siguientes afirmaciones:**

8. Los \_\_\_\_\_ entienden que una empresa es un conjunto de departamentos que se encuentran interrelacionados por la \_\_\_\_\_ que comparten entre sí y que se genera a partir de sus procesos. Una ventaja de los ERP, tanto \_\_\_\_\_

como técnica, es que la funcionalidad está dividida en \_\_\_\_\_ (ventas, materiales, finanzas, control de almacén, recursos humanos) que pueden instalarse según los requerimientos del \_\_\_\_\_.

9. El \_\_\_\_\_ de la información empresarial en un \_\_\_\_\_ permite centralizar toda la logística de \_\_\_\_\_. Una vez aplicadas la instalación y las políticas de \_\_\_\_\_, los usuarios deberán utilizar este sistema para poder garantizar la coherencia de los datos y la \_\_\_\_\_ del negocio.

10. Los riesgos \_\_\_\_\_ son aquellos que pueden suceder cuando fallan algunos componentes \_\_\_\_\_ de nuestro sistema \_\_\_\_\_. Fallos como la rotura de los discos duros y la rotura de memorias son los más habituales; y, sobre todo, es preciso aplicar una política de copias de \_\_\_\_\_ que permita salvaguardar los \_\_\_\_\_ importantes en el caso de imprevistos.

Las soluciones a los ejercicios de autocomprobación se encuentran al final de esta Unidad Formativa. En caso de que no los hayas contestado correctamente, repasa la parte de la lección correspondiente.

## Soluciones a los ejercicios de autocomprobación

### Unidad 5

1. F. Un documento XML no está compuesto de datos, que, como tales, pueden utilizarse en múltiples aplicaciones informáticas y con objetivos bien diferentes.
2. V
3. F. Los lenguajes de hojas de estilo no son la única forma de convertir un documento XML a otro formato.
4. V
5. V
6. V
7. V
8. navegadores, jerarquía, etiquetas, Netscape, XML.
9. XSL, variables, definir, depurador, estilos.
10. XSL, XML, software, productividad, seguridad.

### Unidad 6

1. V
2. F. Con el objetivo de manipular los datos entre un documento XML y una base de datos, y viceversa, los datos en la base de datos deben ser mapeados a la estructura del documento XML.
3. F. Las estrategias de inserción y extracción en el almacenamiento de información en XML se basan en tres procedimientos básicos: almacenar el documento completo en formato de texto como un gran objeto binario (BLOB) en una base de datos relacional; almacenar una forma modificada del documento completo en el sistema de archivos; mapear la estructura de los datos en el documento en la base de datos.
4. V
5. V
6. V
7. V
8. almacenamiento, información, XML, datos, formatos.
9. servidor, almacenamiento, recuperación, XML, procesos.
10. SQL, programación, información, datos, relacionales.

### Unidad 7

1. V
2. V

3. V

4. F. En la gestión de relaciones con los proveedores (área proveedores), una de las opciones más interesantes por lo que respecta al empleo de las nuevas tecnologías de la información es la posibilidad de enviar información rápidamente a través de la cadena de valor.

5. V

6. F. Los riesgos lógicos son aquellos que pueden suceder cuando existe una política inadecuada de autenticación en los sistemas informáticos: accesos no autorizados, bugs y errores en el sistema operativo, o en el software utilizado.

7. V

8. ERP, información, económica, módulos, cliente.

9. almacenamiento, ERP, negocio, seguridad, continuidad.

10. físicos, electrónicos, informático, seguridad, datos.

## Índice

### MÓDULO: LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

#### UNIDAD FORMATIVA 2

<b>5. Conversión y adaptación de documentos XML.....</b>	112
5.1 Técnicas de transformación de documentos XML.....	112
5.2 Formatos de salida .....	115
5.3 Ámbitos de aplicación .....	116
5.4 Descripción de la estructura y sintaxis.....	118
5.5 Utilización de plantillas.....	120
5.6 Utilización de herramientas de procesamiento.....	122
5.7 Verificación del resultado.....	124
5.8 Depuración .....	125
5.9 Elaboración de documentación .....	126
Resumen.....	128
Ejercicios de autocomprobación .....	129
<b>6. Almacenamiento de información.....</b>	131
6.1 Sistemas de almacenamiento de información.....	132
6.2 Inserción y extracción de información en XML.....	134
6.2.1 Estrategias de almacenamiento.....	134
6.3. Técnicas de búsqueda de información en documentos XML .....	136
6.3.1 El lenguaje XPath.....	136
6.3.2 El lenguaje XQuery .....	136
6.4 Lenguajes de consulta y manipulación .....	137
6.5 Almacenamiento XML nativo.....	142

6.6 Herramientas de tratamiento y almacenamiento de información en formato XML.....	146
6.6.1 Tamino XML Server.....	146
6.6.2 <i>eXist</i> .....	148
6.6.3 Oracle 11g XML DB.....	148
6.6.4 dbXML.....	148
6.6.5 Oracle XML DB .....	149
Resumen.....	151
Ejercicios de autocomprobación .....	152
 <b>7. Sistemas de gestión empresarial .....</b>	 154
7.1 Instalación del sistema ERP .....	155
7.1.1 Principales objetivos del sistema ERP .....	156
7.2 Identificación de flujos de información .....	158
7.2.1 Diagrama de flujo de información de funciones cruzadas .....	160
7.3 Adaptación y configuración.....	161
7.3.1 Integrales.....	161
7.3.2 Modulares .....	162
7.3.3 Adaptables.....	162
7.4 Integración de módulos .....	163
7.5 Elaboración de informes.....	164
7.5.1 Conectores (importación y exportación) .....	165
7.5.2 Exportación a documentos ofimáticos .....	166
7.5.3 Importación y exportación en ficheros XML.....	166
7.5.4 Importación y exportación en ficheros CSV .....	167
7.5.5 <i>Business Intelligence</i> .....	167
7.6 Planificación, implementación y verificación de la seguridad.....	168
7.6.1 Riesgos físicos .....	168

7.6.2 Riesgos lógicos.....	169
7.6.3 Plan de contingencia .....	170
7.7 Integración con aplicaciones ofimáticas.....	171
7.7.1 Tablas dinámicas de Microsoft Excel .....	173
7.8 Exportación de información.....	175
Resumen .....	177
Ejercicios de autocomprobación .....	178
Soluciones a los ejercicios de autocomprobación.....	180