

# Tema 3: Lenguajes para el almacenamiento y transmisión de la información

## Contenido

3.1 Tipos de lenguajes de marcas .....	2
3.2 Definición de XML.....	2
3.2.1 Lenguajes basados en XML .....	3
3.2.2 Usos de XML .....	5
3.2.3 Tecnologías relacionadas con XML .....	5
3.2.4 Software para producir XML .....	6
3.3 Estructura y sintaxis de XML .....	7
3.3.1 Estructura de un documento XML .....	8
3.4 Documentos XML bien formados .....	12
3.4.1 Reglas generales .....	12
3.4.2 Reglas para los elementos .....	13
3.4.3 Reglas para los atributos.....	13
3.4.4 Documentos válidos.....	14
3.5 Espacios de nombres .....	14
3.5.1 Diferenciar elementos.....	15
3.5.2 Uso del atributo xmlns .....	16
3.5.3 Espacios de nombres por defecto.....	16
3.5.4 Uso de espacios de nombres en etiquetas interiores .....	17
3.5.5 Uso de espacios de nombres por defecto en etiquetas interiores.....	18
3.5.6 Cancelar espacios por defecto .....	18

## 3.1 Tipos de lenguajes de marcas

- ✚ **Orientados a la presentación.** En ellos al texto común se añaden palabras encerradas en símbolos especiales que contienen indicaciones de formato que permiten a los traductores de este tipo de documentos generar un documento final en el que el texto aparece con el formato indicado. Es el caso de **HTML** en el que se indica cómo debe presentarse el texto (y no por ejemplo lo que significa el mismo) también se considera así los archivos generados por los procesadores de texto tradicionales en los que al texto del documento se le acompaña de indicaciones de formato (como negrita, cursiva,...)
- ✚ **Orientados a la descripción.** En ellos las marcas especiales permiten dar significado al texto pero no indican cómo se debe presentar en pantalla el mismo. Sería el caso de **XML** (o de **SGML**) en el que la presentación nunca se indica en el documento; simplemente se indica una semántica de contenido que lo hace ideal para almacenar datos (por ejemplo si el texto es un nombre de persona o un número de identificación fiscal).
- ✚ **Orientados a procedimientos.** Se trata de documentos en los que hay texto marcado especialmente que en realidad se interpreta como órdenes a seguir y así el archivo en realidad contiene instrucciones a realizar con el texto. Es el caso de **LaTeX** o **PostScript** donde por ejemplo se puede indicar una fórmula matemática.

## 3.2 Definición de XML

XML es un lenguaje de marcas que se ha estandarizado y se ha convertido en uno de los formatos más populares para intercambiar información.

La realidad es que XML siempre ha estado muy ligado al éxito de HTML. Se planteó por los problemas crecientes que se fueron observando en las páginas web.

HTML tiene estos problemas como formato de intercambio de información:

- ✚ La mayoría de etiquetas HTML no son semánticas; es decir, no sirven para decir el tipo de contenido que tenemos sino para indicar el formato. Por ejemplo la etiqueta **h1** sí es semántica ya que indica que el texto que contiene es un encabezado de nivel principal. Mientras que la etiqueta HTML clásica **font** sirve para colorear o cambiar el tipo de letra, sin indicar qué tipo de texto tenemos, se aplica a cualquier texto.
- ✚ HTML es un lenguaje rígido, no es extensible. Es decir, no podemos añadir etiquetas ya que ningún navegador las reconocerá. Cada vez que se decide añadir una etiqueta hay que cambiar el estándar y los navegadores se deben adaptar a los cambios.

Por ello, al crear XML se plantearon estos objetivos:

- ✚ Debía ser similar a HTML (de hecho se basa en el lenguaje SGML base para el formato HTML)
- ✚ Debía ser extensible, es decir, que sea posible añadir nuevas etiquetas sin problemas. Esa es la base del lenguaje XML.
- ✚ Debía tener unas reglas concisas y fáciles, además de estrictas.
- ✚ Debía ser fácil de implantar en todo tipo de sistemas. XML nace con una vocación multiplataforma, como base de intercambio de información entre sistemas de toda índole.
- ✚ Debía ser fácil de leer por los humanos y fácil crear procesadores de software XML.

### 3.2.1 Lenguajes basados en XML

Algunos de los lenguajes estándares de marcado basados en XML son:

- RSS.** *Really Simple Syndication*, aunque hay otras interpretaciones de los acrónimos. Para producir contenidos sindicables, se utiliza fundamentalmente para producir noticias. Es una de las aplicaciones XML más utilizada.
- Atom.** Formato semejante al anterior, pensado también para distribuir información desde una web. Muchas veces complementa a RSS.
- ePUB.** Formato de libro digital que se ha convertido en un estándar por la gran implantación que está teniendo en todos los dispositivos de lectura digitales. En realidad es un archivo comprimido (ZIP) que contiene tres documentos XML que son los que especifican la estructura y contenido del documento.
- DITA.** *Darwin Information Typing Architecture*, se utiliza para producir documentos técnicos y está siendo cada vez más popular. Tiene capacidad para incluir numerosos metadatos y para poder adaptarse a las necesidades de las entidades que utilicen el lenguaje. Permite dividir en temas reutilizables los documentos.
- MathML.** Pensado para representar documentos con expresiones matemáticas.

Una página web que contiene elementos MathML es un documento compuesto que contiene tanto elementos XHTML como MathML. El tipo del documento tiene que ser por tanto XHTML 1.1 + MathML 2.0 y se debe servir al navegador con el tipo MIME *application/xhtml+xml* (MIME: especificaciones dirigidas al intercambio a través de Internet de todo tipo de archivos: texto, audio, vídeo, etc. de forma transparente para el usuario)

Normalmente los servidores sirven los documentos que tienen la extensión .html con el tipo MIME *text/html* y los documentos que tienen la extensión .xhtml con el tipo MIME *application/xhtml+xml*, por lo que conviene guardar los documentos que incluyan elementos MathML con la extensión .xhtml.

En los navegadores que no son capaces de mostrar elementos MathML, las fórmulas matemáticas deben mostrarse como imágenes.

La siguiente página muestra un ejemplo sencillo de las posibilidades de MathML (se abre una página web, utiliza la opción *botón derecho / ver código fuente* para ver el código MathML):

[La ecuación de segundo grado](#)

- ODF u OD.** *Open Document for Office Applications*. Es un formato abierto que pretende ser un estándar como formato de intercambio entre documentos de oficina (hojas de cálculo, procesadores de texto, presentaciones, diagramas,...). Su popularidad aumentó notablemente cuando fue adoptado por el software **Open Office**, actualmente perteneciente a la empresa **Oracle**.
- OOXML.** *Office Open XML*. Formato propiedad de Microsoft y utilizado como formato XML para el software **Microsoft Office**. Pretende también ser un estándar.
- RDF.** *Resource Description Format*. Sirve para desarrollar documentos que describan recursos. Se trata de un proyecto ya antiguo para definir modelos de metadatos. Se basa en los modelos conceptuales como el modelo **entidad/relación** o los **diagramas de clases**, aunque actualmente se utiliza fundamentalmente para describir recursos web.
- SMIL.** *Synchronized Multimedia Integration Language*. Lenguaje sincronizado de integración multimedia. Utilizado para producir presentaciones de TV en la web, fundamentalmente.
- SOAP.** *Simple Object Access Protocol*. Protocolo estándar de comunicación entre objetos utilizado para comunicar con Servicios Web.
- SVG.** *Scalable Vector Graphics*. Gráficos de vectores escalables. Permite definir imágenes vectoriales pensadas

para ser publicadas en una página web.

SVG se está imponiendo poco a poco como formato estándar de gráficos vectoriales frente a otros formatos propietarios, y numerosos programas de edición de gráficos son capaces de importar y exportar en formato SVG.

Firefox (a partir de la versión Firefox 1.5), Internet Explorer 9 y Chrome son capaces de mostrar gráficos SVG sin necesidad de ningún plug-in.

Los ficheros gráficos SVG pertenecen al tipo MIME *image/svg+xml*, y se guardan en disco con la extensión *.svg*. Su código interno sigue las especificaciones XML, la estructura general de un fichero SVG es como la del siguiente ejemplo (copia y pega en un documento de texto de nombre *grafico.svg* el código que aparece a continuación y ábrelo desde el navegador):

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="540" height="200" viewBox="0 0 270 100" style="margin:-2em 0em -3em 0em">
  <defs>
    <radialGradient id="radial" cx="50%" cy="50%" fx="25%" fy="25%">
      <stop offset="0%" stop-color="#60bafc" />
      <stop offset="50%" stop-color="#409adc" />
      <stop offset="100%" stop-color="#005a9c" />
    </radialGradient>
    <path id="curve" d="M18,60 C100,10 180,110 264,60" fill="none"/>
  </defs>

  <circle cx="138" cy="50" r="38" fill="url(#radial)" stroke="#005a9c" />

  <text font-family="Verdana" font-size="20" fill="#ff9900"><textPath xlink:href="#curve"
method="stretch" style="visibility: visible;">Scalable <tspan fill="white">Vector</tspan>
Graphics</textPath></text>
</svg>
```

**VoiceXML.** Se utiliza para representar diálogos vocales.

**WSDL.** *Web Services Description Language.* Lenguaje para definir Servicios Web.

**XHTML.** Versión del lenguaje de creación de páginas web, **HTML**, que es compatible con las normas XML.

### 3.2.2 Usos de XML

#### Contenido web

XML podría pasar a reemplazar a HTML como el formato en el que se escriben las páginas web. Es de hecho una de sus vocaciones ya que ofrece una sintaxis más rígida que resulta ventajosa porque facilita su aprendizaje, una escalabilidad que permite que jamás se quede obsoleto el lenguaje, y una serie de tecnologías relacionadas más poderosas. Todo XML tendrá un documento de validación conocido por los navegadores que especifica exactamente qué elementos y de qué forma se pueden utilizar en el documento.

#### Intercambio de información entre aplicaciones

El hecho de que XML almacene información mediante documentos de texto plano, facilita que se utilice como estándar, ya que no se requiere software especial para leer su contenido: es texto y es entendible por cualquier software.

#### Computación distribuida

Se trata de la posibilidad de utilizar XML para intercambiar información entre diferentes computadoras a través de las redes. Las ventajas de XML están relacionadas con el hecho de que con él se crean documentos inocuos (no pueden contener código maligno como virus o espías), con lo que la seguridad de esos sistemas es total. Al ser archivos de texto, los routers no tienen ningún problema en encaminarlos, entienden que no es una amenaza.

#### Información empresarial

XML es un formato que tiene cada vez más importancia para generar documentos empresariales por la facilidad de estructurar los datos de la forma más apropiada para la empresa. Un documento XML se parece mucho a una pequeña base de datos, con la ventaja de que es muy fácil darle formato de salida por pantalla o impresora.

### 3.2.3 Tecnologías relacionadas con XML

XML posee un gran número de tecnologías para dar funcionalidad, presentación o integración con otros lenguajes. Las más importantes son:

- ✚ **DTD. *Document Type Definition***, definición de tipo de documento. Es un lenguaje que permite especificar documentos cuyas reglas han de cumplir los documentos XML a los que se asocien. Es decir, permite crear **documentos de validación** para archivos XML.
- ✚ **XML Schema**. La función que cumple esta tecnología es la misma que la anterior, la diferencia está en que los documentos XML Schema poseen una sintaxis 100% XML, por lo que es un formato orientado a suplir al anterior.
- ✚ **Relax NG**. Otro formato de definición de validaciones para documentos XML. Es una alternativa a las dos anteriores y la que tiene un formato más sencillo.
- ✚ **Namespacing, espacios de nombres**. Permite conseguir nombres de elementos que carecen de ambigüedad: es decir nombres únicos dentro de los documentos XML.
- ✚ **XPath**. Lenguaje de consulta que permite seleccionar o acceder a partes de un documento XML.
- ✚ **CSS. *Cascade StyleSheet***. Permiten dar formato a los documentos XML o HTML.
- ✚ **XSLT**. Sirve para lo mismo que CSS, dar formato a un documento, en este caso XML, pero tiene más posibilidades.

- ✚ **XQuery**. Permite consultar datos de los documentos XML, manejándolos como si fuera una base de datos.
- ✚ **DOM. Document Object Model**. Permite acceder a la estructura jerárquica del documento normalmente para utilizarla dentro de un lenguaje de programación.
- ✚ **SAX. Simple API for XML**. Permite el uso de herramientas para acceder a la estructura jerárquica del documento XML a través de otro lenguaje; se usa mucho en Java.
- ✚ **XForms**. Formato de formularios de introducción de datos.
- ✚ **XLink**. Permite crear hipervínculos en un documento XML.
- ✚ **XPointer**. Semejante al anterior, especifica enlaces a elementos externos al documento XML.

### 3.2.4 Software para producir XML

El software necesario para crear documentos XML es el siguiente:

- ✚ Un **editor de texto plano** para escribir el código XML. Bastaría un editor como el **bloc de notas** de Windows o el clásico **vi** de Linux; o las opciones de editores capaces de colorear el código como **emacs**, **Notepad++** o **SublimeText**.
- ✚ Un **analizador sintáctico** o **parser**. Programa capaz de leer y comprobar que el documento escrito en lenguaje XML está bien formado o incluso que es válido. Podemos dividir los parsers XML en dos grupos principales:
  - **Sin validación**: el parser no valida el documento utilizando un DTD, sino que sólo chequea que el documento esté bien formado de acuerdo a las reglas de sintaxis de XML (sólo hay una etiqueta raíz, las etiquetas están cerradas, etc).
  - **Con validación**: además de comprobar que el documento está bien formado según las reglas anteriores, comprueba el documento utilizando un DTD (ya sea interno o externo).

Actualmente, **cualquier navegador** incorpora un XML-parser, aunque **Apache Xerces** es quizá el más popular analizador de documentos XML ya que analiza sintácticamente y valida.

- ✚ Un **procesador XML** que sea capaz de producir un resultado visual sobre el documento XML. También un simple navegador puede hacer esta función, pero cuando se aplican formatos visuales sobre el documento XML (como los creados mediante **XSL**) entonces hace falta un software especial que convierta los datos a la forma final visible por el usuario. **Apache Xalan** y **Saxon** son los dos procesadores más conocidos.

Nosotros utilizaremos **XML Copy Editor** para analizar sintácticamente, validar y procesar transformaciones XSL

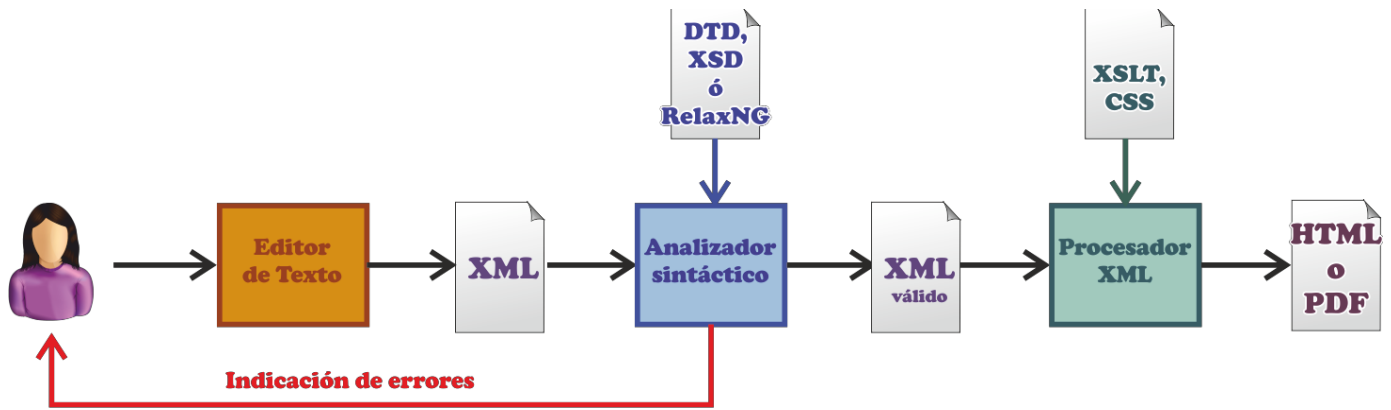


Ilustración 1, Proceso completo de funcionamiento productivo de un XML

Hay entornos de trabajo que incluyen todas esas prestaciones dentro del mismo paquete software.

### 3.3 Estructura y sintaxis de XML

Los documentos XML, en definitiva, son documentos de lenguajes de marcas donde hay contenido textual y etiquetas (marcas) que permiten clasificar dicho texto indicando su significado.

Las etiquetas en XML se deciden a voluntad, no hay una serie de etiquetas que se pueden utilizar, la función de XML es definir tipos de documentos etiquetados.

Ejemplo de XML:

```

<persona>
  <nombre>Jorge</nombre>
  <apellido>Sánchez</apellido>
</persona>
  
```

XML llama **elemento** al conjunto formado por las etiquetas de inicio y de fin y lo que contienen.

Repasemos las normas a seguir para escribir código XML:

- ✚ Las etiquetas sirven para indicar elementos. El nombre de la etiqueta se indica entre los símbolos `< y >`
- ✚ Las etiquetas se cierran indicando `</` seguido del nombre de la etiqueta que se está cerrando.
- ✚ XML distingue entre mayúsculas y minúsculas siendo buena práctica escribir las etiquetas en minúsculas.
- ✚ Se pueden espaciar y tabular las etiquetas a voluntad. Es buena práctica que un elemento interno a otro aparezca en el código con una sangría mayor (por eso en el ejemplo anterior *nombre*, que está dentro de *persona*, aparece sangrado)
- ✚ Los comentarios en el código se inician con los símbolos `<!--` y terminan con `-->`
- ✚ Según la **W3C** (organismo de estandarización de XML), el texto en un documento XML debe de estar codificado en **Unicode** (normalmente **UTF-8**)

### 3.3.1 Estructura de un documento XML

Los documentos XML están formados por:

- ✚ **Prólogo.** Se trata de la primera zona del documento y sirve para describir qué tipo de documento es.
- ✚ **Elemento raíz.** Todo el contenido del documento debe de estar incluido en el llamado elemento raíz, se trata de un elemento obligatorio que se abre tras el prólogo y se debe cerrar justo al final. De este modo cualquier elemento está dentro del elemento raíz. Contiene:
  - **Más elementos**
  - **Atributos**
  - **Contenido textual**
  - **Entidades**
  - **Comentarios**

#### 3.3.1.1 Reglas para los nombres

En XML, los elementos y sus atributos tienen un nombre o **identificador** que debe cumplir estas reglas:

- ✚ XML distingue mayúsculas y minúsculas por lo que hay que tener cuidado al utilizar el nombre desde otro punto del documento.
- ✚ Los **identificadores deben comenzar** por una letra, y después le seguirán más letras, números o el signo de subrayado o guión bajo.
- ✚ No pueden empezar con la palabra XML ni en mayúsculas ni en minúsculas ni en ninguna combinación de mayúsculas ni minúsculas.
- ✚ Los espacios en blanco no están permitidos en los nombres de los elementos. Por ejemplo :  
`<nombre elemento> </nombre elemento>` es erróneo

#### 3.3.1.2 Elementos del prólogo

##### Declaración XML

Se trata de la primera línea de un documento XML e indica el tipo de documento XML que es para poder validarlo. En realidad es opcional, pero es muy recomendable.

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

Indica la versión XML del documento y la codificación (utf-8 es la habitual).

##### Instrucciones de procesamiento

Un documento XML puede incluir instrucciones de este tipo para indicar un documento para validar el código XML, darle formato u otras funciones. Por ejemplo:

```
<?xml-stylesheet type="text/xsl" href="stylesheet.xsl"?>
```

Esta instrucción asocia un documento **xsl** al documento XML para poder darle un formato de salida (para especificar la forma en la que los datos se muestran por pantalla por ejemplo).



### 3.3.1.3 Comentarios

Como se ha indicado antes comienzan con el símbolo `<!--` y terminan con `-->`. Dentro puede haber cualquier texto que se utiliza con fines explicativos o de documentación del código.

Los comentarios no pueden meterse dentro de la etiqueta de un elemento, ni tampoco puede contener etiquetas de apertura o cierre.

### 3.3.1.4 Elementos

Son la base del documento XML. Sirven para dar significado al texto o a otros elementos o para definir relaciones entre distintos elementos y datos.

Ej: `<nombre>Jorge</nombre>`

`<nombre>` Es un etiqueta de apertura

`</nombre>` Es una etiqueta de cierre

`<nombre>Jorge</nombre>` Es un elemento (el elemento *nombre*)

**Jorge** es el contenido del elemento

El contenido de un elemento puede contener simplemente texto:

`<descripción>`

Producto con precio rebajado debido a su escasa demanda

`</descripción>`

O puede contener otros elementos (o ambas cosas). En este caso, el elemento *persona* consta de un elemento *nombre* y otro *apellido*.

`<persona>`

`<nombre>Jorge</nombre>`

`<apellido>Sánchez</apellido>`

`</persona>`

Los elementos se deben abrir y cerrar con la etiqueta que sirve para definir el elemento y siempre deben estar anidados. El siguiente ejemplo sería erróneo puesto que se cierra *persona* antes que *apellido*:

`<persona>`

`<nombre>Jorge</nombre>`

`<apellido>Sánchez</persona>`

`</apellido>`

Puede haber incluso elementos vacíos:

`<casado></casado>`

En este caso se pueden cerrar en la propia etiqueta de apertura:

`<casado />`

### 3.3.1.5 Atributos

Se definen **dentro de las etiquetas de apertura** de los elementos. Se indica su nombre seguido del signo = y del valor entre comillas dobles que se le da al atributo. Ejemplo:

```
<persona puesto="programador">
  <nombre>Jorge</nombre>
  <apellido>Sánchez</apellido>
</persona>
```

Un elemento puede contener varios atributos:

```
<persona puesto="programador" tipo="web">
  <nombre>Jorge</nombre>
  <apellido>Sánchez</apellido>
</persona>
```

### 3.3.1.6 Texto

El texto está siempre entre una etiqueta de apertura y una de cierre. Eso significa que todo texto es parte de un elemento XML.

Se puede escribir cualquier carácter Unicode en el texto, pero no es válido utilizar caracteres que podrían dar lugar a confusión como los signos separadores mayor (>), menor (<) y ampersand (&) por ejemplo.

### 3.3.1.7 CDATA

Existe la posibilidad de marcar texto para que no sea procesado como parte de XML, eso se consigue colocándolo dentro de un elemento CDATA. El formato es el siguiente:

```
<![CDATA [ texto no procesable... ]]>
```

Esto permite utilizar los caracteres < y > por ejemplo y no serán considerados como separadores de etiquetas.

Ejemplo:

```
<?xml version="1.0"?>
<documento>
  <título>Prueba</título>
  <ejemplo>
    <![CDATA[En HTML la negrita se escribe: <strong>]]>
  </ejemplo>
</documento>
```

En el ejemplo, los símbolos < y > no se toman como una etiqueta XML, sino como texto normal.

Otro uso de CDATA es colocar dentro de este elemento código de lenguajes de scripts como **Javascript** para que no sean interpretados como parte de XML.

### 3.3.1.8 Entidades

Las entidades representan caracteres individuales. Se utilizan para poder representar caracteres especiales o bien caracteres inexistentes en el teclado habitual. Se trata de códigos que empiezan con el signo **&** al que sigue el nombre de la entidad o el número Unicode del carácter que deseamos representar.

En XML hay definidas cinco entidades:

<b>&amp;gt;</b>	Símbolo >
<b>&amp;lt;</b>	Símbolo <
<b>&amp;amp;</b>	Símbolo &
<b>&amp;quot;</b>	Símbolo “
<b>&amp;apos;</b>	Símbolo ’

También podemos representar caracteres mediante entidades con número. De modo que el **&#241;** representa a la letra ñ (suponiendo que codificamos en Unicode, que es lo habitual). El número puede ser hexadecimal, por ejemplo, también para la eñe sería **&#xF1;**

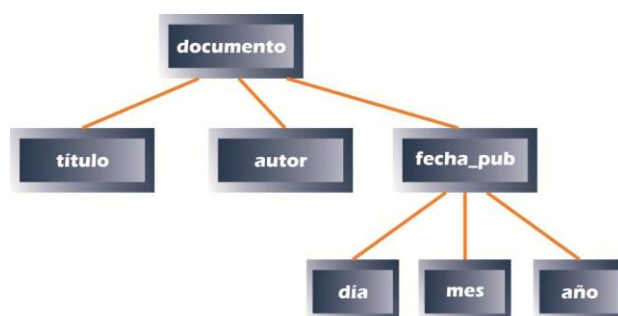
### 3.3.1.9 Jerarquía XML

Los elementos de un documento XML establecen una jerarquía que estructura el contenido del mismo. Esa jerarquía se puede representar en forma de árbol.

Así por ejemplo el archivo XML:

```
<?xml version="1.0"?>
<documento>
  <título>Apuntes de XML</título>
  <autor>Jorge Sánchez</autor>
  <fecha_pub>
    <día>18</día>
    <mes>Enero</mes>
    <año>2009</año>
  </fecha_pub>
</documento>
```

Gráficamente de forma jerárquica se podría expresar así:



**Ilustración 2, Estructura jerárquica de un documento XML**

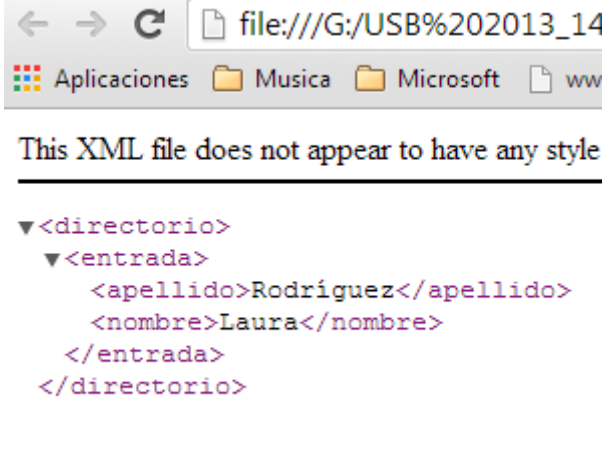
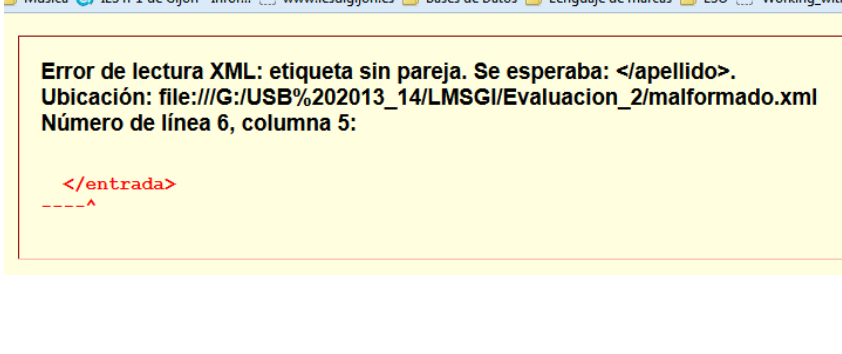
### 3.4 Documentos XML bien formados

Se habla de documento XML **bien formado** (**well formed**) cuando cumple las reglas basadas en la gramática especificada por el W3C .

Debemos tomar las reglas para producir XML bien formado como reglas obligadas, de hecho los analizadores (**parsers**) de XML indicarían error en caso de no cumplirlas. Es decir, un documento XML bien formado es un **documento analizable** (pero no significa que sea **válido** que es un concepto que se explicará en la siguiente unidad).

En la actualidad, los navegadores web pueden distinguir los documentos bien formados de los documentos mal formados. Si el documento está bien formado el navegador mostrará el árbol XML, si por el contrario, el documento contiene errores, el navegador los señalará y se situará en el primer elemento que contenga un error.

Por ejemplo:

<pre>&lt;?xml version="1.0" encoding="UTF-8" ?&gt; &lt;directorio&gt;   &lt;entrada&gt;     &lt;apellido&gt;Rodríguez&lt;/apellido&gt;     &lt;nombre&gt;Laura&lt;/nombre&gt;   &lt;/entrada&gt; &lt;/directorio&gt;</pre>	
<pre>&lt;?xml version="1.0" encoding="UTF-8" ?&gt; &lt;directorio&gt;   &lt;entrada&gt;     &lt;apellido&gt;Rodríguez     &lt;nombre&gt;Laura&lt;/nombre&gt;   &lt;/entrada&gt; &lt;/directorio&gt;</pre>	

#### 3.4.1 Reglas generales

**Dentro de los datos textuales** no se pueden utilizar los símbolos de mayor (>), menor (<), ampersand (&) ni las comillas simples o dobles. Se deben de utilizar entidades o deben estar incluidos en una sección CDATA.

En principio, **en los datos textuales**, los símbolos de separación de caracteres como espacios en blanco, tabuladores y saltos de línea, no se tienen en cuenta, se ignoran. Pero sí es posible que sean significativos en algunos elementos si así se indica en su documento de validación. No tener en cuenta los espacios, significa que en el texto contenido en un elemento, aunque dejemos 25 espacios, se entenderá que hemos dejado uno solo.

### 3.4.2 Reglas para los elementos

- ✚ Se deben cerrar primero las etiquetas de los últimos elementos abiertos.
- ✚ Todos los elementos poseen etiquetas de apertura y de cierre; es decir toda etiqueta que se abra se debe de cerrar. Si la etiqueta no tiene contenido, se debe cerrar en la propia etiqueta al estilo `<br />`
- ✚ Todos los documentos XML deben de tener un único elemento raíz.
- ✚ Los nombres de los elementos comienzan con letras y pueden ir seguidos de letras, números, guiones o de puntos (los guiones y los puntos no son muy recomendables)
- ✚ Los nombres de los elementos no pueden comenzar con el texto **xml** tanto en minúsculas como en mayúsculas o combinando ambas.

### 3.4.3 Reglas para los atributos

- ✚ Los nombres de atributos siguen las mismas normas que los nombres de los elementos.
- ✚ Los atributos sólo se pueden colocar en etiquetas de apertura y nunca en las de cierre.
- ✚ Los valores de los atributos deben ir entrecomillados (sin importar si se usan comillas simples o dobles)

Ejemplos:

```
<nombre sexo="Hombre">Antonio</nombre>
<nombre sexo='Mujer'>Sara</nombre>
<nombre sexo="Mujer">Eva</nombre> <!-- error -->
```

*La última línea es incorrecta porque no se pueden combinar ambas comillas.*

- ✚ Todos los atributos deben tener valor asociado. Es decir, la etiqueta `<hr noshade>` válida en HTML, no sería válida en XML:

```
<hr noshade>
```

Debería ser, por ejemplo, asignando al atributo *noshade* su valor correspondiente:

```
<hr noshade="noshade">
```

### 3.4.4 Documentos válidos

Un documento XML bien formado puede ser válido. Para ser válido, un documento XML debe:

- ✚ Incluir una referencia a una gramática.
- ✚ Incluir únicamente elementos y atributos definidos en la gramática.
- ✚ Cumplir las reglas gramaticales definidas en la gramática.

Existen varias formas de definir una gramática para documentos XML, las más empleadas son:

- ✚ DTD (Document Type Definition = Definición de Tipo de Documento). Es el modelo más antiguo, heredado del SGML.
- ✚ XML Schema. Es un modelo creado por el W3C como sucesor de las DTDs.
- ✚ Relax NG. Es un modelo creado por OASIS, más sencillo que XML Schema.

## 3.5 Espacios de nombres

Los espacios de nombres (*namespacing* en inglés) permiten definir la pertenencia de los elementos y los atributos de un documento XML a una familia de vocabulario XML. El nombre de un elemento puede pertenecer a varias familias sin que ello implique que poseen el mismo significado. Así, es posible integrar las etiquetas de diferentes lenguajes dentro de un mismo documento XML (HTML, XHTML, MathML,...).

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <title>Documento de prueba</title>
  <content>
    <html>
      <head>
        <title>Titulo HTML</title>
      </head>
      <body>
        Texto del documento
      </body>
    </html>
  </content>
  <author>Jorge</author>
</document>
```

En el ejemplo anterior se usan etiquetas en inglés para el documento (algo muy habitual en el mundo empresarial) y eso hace que la etiqueta *title* se repita en contextos distintos, el primero es para poner un título genérico al documento (y es una etiqueta de la empresa en cuestión) y la segunda se corresponde a la etiqueta *title* del lenguaje **XHTML**.

### 3.5.1 Diferenciar elementos

La solución es anteponer al nombre de la etiqueta un nombre que indique el propietario de la misma:

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <jorge.title>Documento de prueba</jorge.title>
  <content>
    <html>
      <head>
        <html.title>Titulo HTML</html.title>
      </head>
      <body>
        Texto del documento
      </body>
    </html>
  </content>
  <author>Jorge</author>
</document>
```

Ese prefijo diferenciador es el espacio de nombres al que pertenece la etiqueta, pero usado así tendríamos el problema de que con un sufijo tan corto se podría repetir. Por ello, una solución es indicar la URL de la entidad responsable de la etiqueta:

```
<?xml version="1.0" encoding="UTF-8"?>
<www.jorgesanchez.net.document>
  <www.jorgesanchez.net.title>
    Documento de prueba
  </www.jorgesanchez.net.title>
  <www.jorgesanchez.net.content>
    <www.w3c.org.html>
      <www.w3c.org.head>
        <www.w3c.org.title>
          Titulo HTML
        </www.w3c.org.title>
      </www.w3c.org.head>
      <www.w3c.org.body>
        Texto del documento
      </www.w3c.org.body>
    </www.w3c.org.html>
  </www.jorgesanchez.net.content>
  <www.jorgesanchez.net.author>
    Jorge
  </www.jorgesanchez.net.author>
</document>
```

Pero el documento quedaría muy poco legible. Por ello se aplican espacios de nombres, de modo que se asigna una URI a un prefijo de etiqueta y así cada vez que en el documento se utiliza el prefijo, se sabe que se refiere a la URI indicada. Las URIs son únicas, la raíz de la **URI** es el dominio web de la empresa y se añade la ruta al recurso.

### 3.5.2 Uso del atributo xmlns

Los espacios de nombre se deben cargar con antelación al inicio del documento XML, antes de poder utilizar los prefijos. Todas las etiquetas en XML pueden hacer uso del atributo **xmlns** (*xml namespaces*) que permite asignar un espacio de nombres a un prefijo dentro del elemento en el que se usa el espacio de nombres. Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns:jorge = "http://www.jorgesanchez.net/document"
          xmlns:html = "http://www.w3c.org/html" >
  <jorge:title>
    Documento de prueba
  </jorge:title>
  <jorge:content>
    <html:html>
      <html:head>
        <html:title>Titulo HTML</html:title>
      </html:head>
      <html:body>
        Texto del documento
      </html:body>
    </html:html>
  </jorge:content>
  <jorge:author>
    Jorge
  </jorge:author>
</document>
```

### 3.5.3 Espacios de nombres por defecto

En el caso en el que la mayoría de las etiquetas en un documento pertenezcan a un mismo espacio de nombres, lo lógico es indicar el espacio de nombres por defecto. Eso se hace sin indicar prefijo en el atributo **xmlns**. Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns = "http://www.jorgesanchez.net/document"
          xmlns:html = "http://www.w3c.org/html" >
  < title>
    Documento de prueba
  </ title>
  < content>
    <html:html>
      <html:head>
        <html:title>Titulo HTML</html:title>
      </html:head>
      <html:body>
        Texto del documento
      </html:body>
    </html:html>
  </ content>
  < author>
    Jorge
  </ author>
</document>
```



Las etiquetas sin prefijo se entiende que pertenecen al espacio de nombres [www.jorgesanchez.net/document](http://www.jorgesanchez.net/document), para las del otro espacio se usa el prefijo.

### 3.5.4 Uso de espacios de nombres en etiquetas interiores

El atributo **xmlns** no tiene por qué utilizarse en el elemento raíz, se puede posponer su declaración al primer elemento que pertenezca al espacio de nombres deseado. Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns = "http://www.jorgesanchez.net/document">
  <title>
    Documento de prueba
  </title>
  <content>
    <html:html xmlns:html = "http://www.w3c.org/html" >
      <html:head>
        <html:title>Titulo HTML</html:title>
      </html:head>
      <html:body>
        Texto del documento
      </html:body>
    </html:html>
  </content>
  <author>
    Jorge
  </author>
</document>
```

Puede ocurrir que cuando se manejan diferentes documentos XML se utilicen las mismas etiquetas. Aunque el contexto sería distinto, tendríamos un problema si manejamos ambos documentos con el mismo software, ya que el analizador, no sabría cómo manejar ambas etiquetas iguales

### 3.5.5 Uso de espacios de nombres por defecto en etiquetas interiores

Un documento puede declarar espacios por defecto en etiquetas interiores lo que permite aún más versatilidad en los documentos. Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns="http://www.jorgesanchez.net/document">
  <!-- comienza el espacio de nombres de jorgesanchez.net -->
  <title>
    Documento de prueba
  </title>
  <content>
    <html xmlns="http://www.w3c.org/html">
      <!-- desde aquí el espacio ahora es el de html -->
      <head>
        <title>
          Titulo HTML
        </title>
      </head>
      <body>
        Texto del documento
      </body>
    </html>
    <!-- fin del espacio html, regresa el espacio jorgesanchez.net -->
  </content>
  <author>
    Jorge
  </author>
</document>
```

### 3.5.6 Cancelar espacios por defecto

Si se usa el atributo `xmlns=""` en el interior de una etiqueta, indica que ese elemento y sus hijos no usan ningún espacio de nombres.