

## 2. Documentos XML

Como ya hemos visto en el capítulo anterior un documentos XML no es más que un fichero de texto donde la información viene etiquetada. Este fichero está compuesto por una serie de elementos definidos en la recomendación XML de W3C. Estos documentos pueden ser leídos por personas (en un bloc de notas o en un navegador) o por sistemas que sacan información del documento XML. Normalmente para que lo lea un ordenador el fichero deberá cumplir una serie de reglas para que la aplicación que lo lea sea capaz de interpretarlo, los documentos XML que cumplen estas reglas se dice que son documentos *bien formados*.

*Objetivos.*

Conocer los elementos que conforman un documentos XML. Aprender a crear documentos XML para guardar información. Crear documentos XML *bien formados* para que estos puedan ser leídos e interpretados por otros sistemas.

*Contenidos.*

- 2.1. Estructura del documento XML
- 2.2. Codificación de los documentos
- 2.3. Elementos
- 2.4. Atributos
- 2.5. Identificación de los elementos
- 2.6. Uso de elementos o de atributos
- 2.7. Caracteres especiales
- 2.8. Comentarios e instrucciones de proceso
- 2.9. Documento bien formado
- 2.10. Ejercicios

### 2.1. Estructura del documento XML

Un documento XML está formado por una mezcla de datos e información de etiquetado sobre los mismos. La información de etiquetado se corresponde con el texto que aparece encerrado entre los caracteres ‘<’ y ‘>’ o entre ‘&’ y ‘;’. El juego de caracteres usado en un documento XML se corresponde con el estándar ISO 10646, denominado *Universal Character Set*. Más adelante, en el apartado dedicado a la codificación del documento, se estudian con más detalle los aspectos relacionados con este tema. Es importante resaltar que XML *distingue* entre mayúsculas y minúsculas, tanto en los datos como en el etiquetado. Por ejemplo, las etiquetas <name> y <NAME> son completamente diferentes. También es conveniente aclarar que en XML a diferencia de HTML los espacios en blanco usados en el documento son significativos, de hecho, en el estándar se especifica que el *parser* debe entregarlos a la aplicación. Por lo que se refiere a la estructura lógica, el documento puede incluir un prólogo con la siguiente información.

- La declaración XML, que es de carácter opcional. Pero muy conveniente.
- Una declaración de tipo de documento (DTD). Es opcional. Más adelante veremos en detalle los DTD. Solo anticipar que sirven para validar documentos XML.

En la declaración XML a su vez debe aparecer:

- La versión XML, usada en el documento. Normalmente la 1.0. este atributo es obligatorio.
- El tipo de codificación utilizada (encoding). Si no aparece esta información se supone la codificación por defecto. (Ver punto 2.2).
- Carácter autónomo (standalone) del documento. Indica si hay declaraciones de etiquetado externas al documento (standalone = "no"). O si el documento es autocontenido (standalone = "yes"). No es obligatorio.

Un ejemplo de declaración XML completa:

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

## 2.2. Codificación de los documentos

Como se acaba de comentar, XML usa el estándar de representación ISO 10646. Se trata de un estándar que engloba a todos los anteriores (como, por ejemplo, UNICODE) y que gracias al uso de 31 bits para la codificación de cada carácter, permite representar cualquier lenguaje humano conocido. Este estándar está en continua revisión y contiene más de setenta mil caracteres diferentes. Pero, el uso directo de este estándar conlleva consigo la incompatibilidad con las herramientas actuales y la producción de archivos de gran tamaño. Por ello, en vez de usar la codificación directa usando 4 bytes (denominado UCS-4) se han definido representaciones alternativas como UTF-8. Esta codificación representa los caracteres ASCII con un solo byte, lo que asegura compatibilidad y un tamaño reducido en los documentos que contengan sólo información tipo ASCII.

Hay que resaltar que en UTF-8 los caracteres del castellano no incluidos en ASCII (vocales acentuadas, ñe, etc.) ocupan 2 bytes. Debido a ello, en muchos entornos de edición en castellano se usa por defecto la codificación ISO-8859-1 o Latín 1. Se trata de una extensión de ASCII englobada también en la ISO 10646, que codifica los caracteres especiales de las lenguas europeas occidentales en un solo byte. Sin embargo, el estándar XML no requiere que un procesador sepa interpretar esta representación, siendo necesario, además indicarlo el principio del documento en la declaración XML.

En cualquier caso, siempre se puede incluir cualquier carácter del juego definido por ISO 10646, aunque sea indirectamente, mediante el mecanismo denominado *referencias a carácter*. Una *referencia a carácter* especifica el valor numérico del carácter ISO 10646 deseado. A continuación, se muestra el formato de esta *referencia a carácter*, para el caso que el número sea decimal o hexadecimal respectivamente:

- &#código\_decimal;
- &#xcódigo\_hexadecimal;

La preferencia siempre será utilizar un editor convencional para crear los documentos XML en formato ISO-8859-1 (también conocido como ISO-latin-1). Aunque en este caso será obligatorio incluir el atributo *encoding* en el encabezamiento del documento. Con este formato podremos utilizar los caracteres especiales como las vocales acentuadas o las ñes. También existe el ISO-8859-15 (ISO-latin-15) que es igual que el ISO-8859-1 pero incluye además con el símbolo €.

## 2.3. Elementos

Un documento XML está formado por una jerarquía de elementos a partir de un elemento raíz único. Se podría definir el elemento como el componente semántico básico de un documento. Un elemento consta de una etiqueta de inicio, un contenido y una etiqueta de fin. Es importante resaltar que cada dato de un documento tiene que formar parte del contenido de algún elemento. La sintaxis de la etiqueta de inicio es `<NombreElemento>` y la de fin es `</NombreElemento>`.

El nombre de un elemento debe empezar por una letra o el carácter subrayado, seguido por letras, dígitos, el carácter punto, el carácter guión o el carácter subrayado. Además, el nombre no deberá comenzar por la cadena XML, ya sea en mayúsculas o en minúsculas. Estas son las características de un nombre XML válido que también se aplican a atributos y entidades como veremos más adelante. Como ya vimos en el primer tema los elementos se pueden clasificar según el contenido en:

- Elementos vacíos. Como `<BR/>` o `<HR/>` de HTML.
- Elementos que solo contienen texto.
- Elementos que contienen otros elementos.
- Elementos que contiene texto y otros elementos.

## 2.4. Atributos

Un atributo representa una información complementaria asociada a un elemento. Aparece en la etiqueta de inicio o en la de elemento vacío. Cada elemento puede tener una lista de atributos asociada, en la que el orden es intrascendente pero no pueden aparecer atributos repetidos. Un atributo consiste en una pareja de nombre y valor, donde el valor debe aparecer encerrado entre comillas simples o dobles., siguiendo la siguiente sintaxis:

```
<elemento atributo1="valor1" atributo2="valor2">
```

Por ejemplo:

```
<peso unidad="gramos" precision="0,01">5,73</peso>
```

El nombre del atributo al igual que el del elemento debe tratarse de un nombre XML válido.

## 2.5. Identificación de los elementos

Cuando se confecciona un documento, hay que empezar identificando cuáles son los elementos que aparecerán en el mismo. Esta descomposición en elementos no es algo que se pueda hacer de forma automática, sin que tenga una única solución válida. Una determinada solución puede ser más o menos adecuada dependiendo del uso que se vaya a hacer del documento. Para aclarar estas consideraciones, se plantea a continuación un ejemplo sencillo pero que permite observar la dimensión del problema.

Supóngase que se pretende incluir el nombre de una persona en un documento. Se plantean varias alternativas:

```
<nombre>Juan Martín Fernández Moreno de la Vega </nombre>  
<nombre>  
  <nombre_pila>Juan Martín</nombre_pila>  
  <apellidos>Fernández Moreno de la Vega</apellidos>  
</nombre>  
<nombre>  
  <nombre_pila>Juan Martín</nombre_pila>  
  <primer_apellido>Fernández</primer_apellido>  
  <segundo_apellido>Moreno de la Vega</segundo_apellido>  
</nombre>
```

Revisando las alternativas se observa que se ha ido ganando en contenido semántico pero también en complejidad, dado que hay que incluir más etiquetas.

¿Cuál es la mejor solución? Esta pregunta no tiene una respuesta directa. Depende de lo que luego tengamos que hacer con el documento. Si posteriormente en el tratamiento del documento siempre vamos a trabajar con el nombre como un todo (sin diferenciar partes) obviamente la primera solución es la óptima. Pero si necesitamos procesar apellidos y nombre por separado la primera solución es la peor de las tres. Por lo tanto para determinar los elementos que van a formar parte del documento XML exige tener una idea preestablecida de los tipos de procesamiento que se que prevén se van a realizar sobre el documento.

Ahora hay que tener presente una cosa. Si vamos a procesar el nombre entero efectivamente la primera solución es la mejor de las tres, pero, ¿y si en un futuro nos piden un nuevo procesamiento donde es necesario separar nombre y apellidos? Seguramente nos vamos a encontrar con problemas ya que hay muchos nombres como el del ejemplo donde es difícil determinar dónde comienza y dónde termina cada elemento. Pero si aunque en un principio no era necesario que hubiéramos elegido la tercera opción, ahora separar en el procesamiento el nombre y los apellidos es muy sencillo (es más ya lo tenemos).

Por lo tanto a la hora de escoger los elementos no solo hay que ver las necesidades presentes de procesamiento sino intentar anticiparse a posibles necesidades futuras dotando de la mayor semántica posible a nuestro documento. Un consejo es, que siempre que dudemos, debemos asignar un elemento diferente a la información en cuestión.

## 2.6. Uso de elementos o de atributos

Otra pregunta que surge cuando se diseña un documento XML es cuándo usar elementos y cuándo atributos. Supóngase por ejemplo, un documento que contiene las transparencias de una presentación. Parece lógico que cada transparencia incluya un título, ¿Se debe crear como elemento o como atributo?

```
<transparencia><título>Uso de elementos o atributos</título>...</transparencia>
```

O

```
<transparencia titulo=”Uso de elementos o atributos”>...</transparencia>
```

Aunque no se puede dar un criterio válido para todos los casos si podemos dar algunas pautas que ayuden a tomar una decisión:

- Si la información tiene una estructura interna debe ser un elemento
- Si contiene una gran cantidad de información, parece más adecuado, un elemento
- Podemos hacer el símil que un elemento es un sustantivo y un atributo un adjetivo
- Los mecanismos de procesamiento y presentación de documentos, permiten tener mejor control sobre los elementos. Por tanto aquella información que tenga un procesamiento o presentación complejos debe ser un elemento.

Aplicando estas pautas podemos concluir que el título de la transparencia debe ser un elemento, tiene carácter sustantivo y debe ser presentado al usuario. Como contraejemplo imaginemos que queremos incluir para cada transparencia información sobre el tiempo que le debemos dedicar (poco, medio, mucho). Esta información no tiene carácter sustantivo y no tiene porqué ser presentada, por tanto, puede ser perfectamente un atributo.

Como consejo final, en caso de duda, se debe utilizar un elemento.

## 2.7. Caracteres especiales

Si observamos el siguiente ejemplo:

```
<comparacion>6 es < 7 & 7 > 6</comparacion>
```

Si visualizamos el texto anterior con un navegador nos dará un error ya que al encontrarse con el carácter ‘<’ espera una etiqueta y no un carácter en blanco.

Existen varios caracteres que tienen un significado especial (<, &, >, ‘ y “). ¿Qué ocurre si se requiere incluir uno de estos caracteres dentro del documento XML? En el caso que pueda existir algún tipo de ambigüedad, será necesario establecer un mecanismo que permita distinguir si el carácter debe interpretarse como especial o no.

Veamos una primera manera de indicar que no se interprete como especiales ninguno de los caracteres de un texto: las secciones *CDATA*.

```
<lista>
  <nombre>John Doe</nombre>
  <correo><![CDATA[<jdoe@server.com>]]</correo>
</lista>
```

En el ejemplo, la sección *CDATA*, que se inicia con la cadena `<![CDATA[` y termina con `]]`, hace que no se interprete como etiquetas el texto que contiene.

En general nos podemos encontrar con los siguientes casos:

- Los caracteres < y & siempre se interpretan como especiales, excepto en secciones especiales *CDATA* y comentarios. Si se quiere que tengan un significado especial se deben utilizar referencias a carácter.
- No se puede incluir la secuencia `]]>` excepto como indicación de final de una sección

*CDATA*. Si se quiere utilizar dicha secuencia habrá que utilizar la referencia a carácter correspondiente a >.

- Los caracteres ‘ y “ no pueden aparecer en valores y atributos que usen como delimitador el mismo tipo de comillas. Para resolverlo utilizaremos como delimitador el carácter contrario al que necesitamos incluir en el texto. Si necesitamos los dos tipos de comillas tendremos que recurrir a las referencias a carácter.

Aunque en caso de conflicto se puede utilizar la referencia a carácter (&#código-decimal;), es recomendable usar las entidades predefinidas que representan a estos caracteres:

Entidad	Carácter	Entidad	Carácter
&lt;	<	&gt;	>
&amp;	&	&apos;	‘
&quot;	“	&aacute;	á

En [siguiente enlace](http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references) de [wikipedia:](http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references) podemos encontrar las referencias a carácter más habituales en nuestro idioma.

## 2.8. Comentarios e instrucciones de proceso

Los comentarios en XML siguen la misma sintaxis que en HTML:

```
<!-- esto es un comentario -->
```

Por los que se refiere a instrucciones de proceso, estas permiten especificar en un documento el nombre de una aplicación, así como los datos que se le pretenden pasar como argumentos a la aplicación.

```
<?aplicación datos?>
```

El parser proporciona esta información al programa que procesa el documento que la tratará como considere oportuno, por ejemplo arrancando dicha aplicación y pasándole los parámetros indicados.

El uso de *PI* (Process Instructions) es muy poco frecuente y, en general, en la comunidad XML se suele evitar su uso. Pero si existe una buena razón se puede utilizar. Por ejemplo, podemos utilizar las PI para colocar la clase de información que colocamos en los comentarios de HTML, como código de javascript. De esta forma nos aseguramos que esta información se la pasa el parser a la aplicación.

La declaración XML (<?xml ....?>) aunque por su sintaxis pueda parecerlo no es una PI. Pero hay que tener en cuenta que si incluimos esta declaración en otro lugar que no sea al principio del fichero XML está puede ser tomada como una PI.

## 2.9. Documento bien formado

Sin la presencia de un DTD, no se puede comprobar la validez de un documento, sólo se puede comprobar si está bien formado. A continuación se enumeran algunas reglas que deben cumplir un documento para que se le considere bien formado:

- Los elementos deben estar anidados adecuadamente.
- Los valores de los atributos deben encerrarse entre comillas simples o dobles.
- Todo elemento debe tener una etiqueta de fin o utilizar la etiqueta de elemento vacío.
- El elemento debe tener un único elemento raíz.
- Todo texto debe estar incluido en un elemento.

## 2.10. Ejercicios

El primer ejercicio viene resuelto para que sirva de guía para resolver los siguientes.

### 2.10.1.Coches

Imaginemos un concesionario de coches que periódicamente debe enviar información sobre los vehículos que tiene en oferta a un portal publicitario de compra-venta de coches. La información que debe enviar es:

- La marca
- El modelo
- Motor
- Matrícula
- Kilómetros
- Precio original
- Precio oferta
- Extras
- fotos

Un documento XML bien formado que contenga esta información puede ser el siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<oferta>
  <vehiculo>
    <marca>ford</marca>
    <modelo color="gris">focus</modelo>
    <motor combustible="gasolina">duratorec 1.4</motor>
    <matricula>1234AAA</matricula>
    <kilometros>12500</kilometros>
    <precio_inicial>12000</precio_inicial>
    <precio_oferta>10000</precio_oferta>
    <extra valor="250">pintura metalizada</extra>
    <extra valor="300">llantas</extra>
    <foto>11325.jpg</foto>
    <foto>11326.jpg</foto>
  </vehiculo>
  <vehiculo>
    <marca>ford</marca>
    <modelo color="gris">focus</modelo>
    <motor combustible="diesel">duratorec 2.0</motor>
    <matricula>1235AAA</matricula>
    <kilometros>125000</kilometros>
    <precio_inicial>10000</precio_inicial>
    <precio_oferta>9000</precio_oferta>
    <extra valor="250">pintura metalizada</extra>
    <extra valor="200">spoiler trasero</extra>
    <extra valor="500">climatizador</extra>
    <foto>11327.jpg</foto>
    <foto>11328.jpg</foto>
  </vehiculo>
</oferta>
```



Este documento podemos abrirlo con un navegador por ejemplo *firefox* y veríamos la siguiente imagen. Los navegadores actuales traen implementados *parsers* de XML por lo tanto podemos utilizarlos para ver que los XML están bien formados. Si por ejemplo en el documento anterior hubiéramos cometido algún error como puede ser no cerrar una etiqueta o escribir erróneamente el nombre de una etiqueta el navegador nos hubiera informado con un error.

```
- <oferta>
  - <vehiculo>
    <marca>ford</marca>
    <modelo color="gris">focus</modelo>
    <motor combustible="gasolina">duratorc 1.4</motor>
    <matricula>1234AAA</matricula>
    <kilometros>12500</kilometros>
    <precio_inicial>12000</precio_inicial>
    <precio_oferta>10000</precio_oferta>
    <extra valor="250">pintura metalizada</extra>
    <extra valor="300">llantas</extra>
    <foto>11325.jpg</foto>
    <foto>11326.jpg</foto>
  </vehiculo>
  - <vehiculo>
    <marca>ford</marca>
    <modelo color="gris">focus</modelo>
    <motor combustible="diesel">duratorc 2.0</motor>
    <matricula>1235AAA</matricula>
    <kilometros>125000</kilometros>
    <precio_inicial>10000</precio_inicial>
    <precio_oferta>9000</precio_oferta>
    <extra valor="250">pintura metalizada</extra>
    <extra valor="200">spoiler trasero</extra>
    <extra valor="500">climatizador</extra>
    <foto>11327.jpg</foto>
    <foto>11328.jpg</foto>
  </vehiculo>
</oferta>
```

### 2.10.2. Libros


Tenemos que mostrar información sobre libros en nuestra página web. Nos ofrecen un servicio donde a partir de un tema nos devolverán un fichero XML con información de los libros encontrados en diferentes librerías electrónicas relacionados con esa temática. Nos piden que escribamos un documentoXML bien formado como ejemplo. Para lo cual se pide escribir un documentoXML con información de libros encontrados en varias librerías electrónicas (amazon, y la casa del libro por ejemplo) sobre la temática 'XML'. Como mínimo sobre cada libro necesitermos:

- Su código ISBN
- Su título
- El nivel de profundidad en el tratamiento de la materia (básico, intermedio avanzado)
- Los autores
- La editorial
- La fecha de publicación
- La página web del libro en caso de que tenga.
- El precio

En el documento XML pondremos información de al menos dos librerías, y para cada librería al menos información de 3 libros.

### 2.10.3. CD música

Escribir un documento XML bien formado para almacenar la información del CD contenida en la siguiente imagen. Podemos observar que tenemos dos bloques de información. En la parte superior, información sobre el disco en general y en la parte inferior las canciones que lo componen. El documento XML debe contener toda la información que tenemos en la imagen.



ARTISTA: **ABC (2)**  
TÍTULO: The Collection  
SELLO: **KARUSSELL INTERNATIONAL**  
REFERENCIA: 551 831-2  
AÑO: 1996  
PAIS: Germany  
FORMATO: CD - Compilation  
GENERO: Funk / Soul, Electronic  
ESTILO: Funk Disco, Synth Pop  
OBSERVACIONES: Licensed from Mercury Records Ltd (London)

**CD 1**

- 01 Poison Arrow - [03:23]
- 02 The Look Of Love (Part One) - [03:29]
- 03 The Night You Murdered Love - [04:53]
- 04 Tears Are Not Enough - [03:29]
- 05 Be Near Me - [03:37]
- 06 I'm In Love With You - [04:35]
- 07 The Real Thing - [04:54]
- 08 Love's A Dangerous Language - [03:40]
- 09 The Greatest Love Of All - [05:34]
- 10 When Smokey Sings - [04:18]
- 11 One Better World - [05:38]
- 12 Vanity Kills - [03:29]
- 13 Bite The Hand - [03:07]
- 14 Chicago (Abridged) - [03:40]
- 15 Minneapolis - [02:57]
- 16 Think Again - [03:39]
- 17 Never More Than Now - [06:03]
- 18 All Of My Heart - [04:49]