



UT1-Reconocimiento de las características de lenguajes de marcas.

2º curso DAM distancia 22-23

Contenidos

1.- Lenguajes de Marcas.

2.- Evolución de los lenguajes de marcas.

2.1.- GML (Generalized Markup Language).

2.2.- SGML (Standard Generalized Markup Language).

2.3.- HTML (HyperText Markup Language).

2.4.- XML (eXtensible Markup Language).

2.5.- XML vs HTML.

2.6.- Comparación de XML con SGML.

2.7.- Etiquetas.

2.8.- Herramientas de edición.

3.- XML, estructura y sintaxis.

3.1.- El prólogo.

3.2.- El ejemplar. Los elementos.

3.2.1.- Atributos.

3.2.2.- Ejercicio Detectar errores.

4.- Documentos XML bien formados.

5.- Utilización de espacios de nombres en XML.

5.1.- Ejemplo: Utilización de espacios de nombres.

1.- Lenguajes de Marcas. Introducción.

Un "lenguaje de marcas" es un modo de codificar un documento donde, **junto con el texto**, se incorporan **etiquetas, marcas o anotaciones con información adicional** relativa a la **estructura** del texto o su **formato de presentación**.

Permiten **hacer explícita la estructura de un documento**, su **contenido semántico** o cualquier otra información lingüística o extralingüística que se quiera hacer patente.

Todo lenguaje de marcas está **definido en un documento denominado DTD (Document Type Definition)**. En él se establecen las marcas, los elementos utilizados por dicho lenguaje y sus correspondientes etiquetas y atributos, su sintaxis y normas de uso.

Ejemplo:

```
1  <carta>
2      <fecha>22/11/2006</fecha>
3      <presentacion>Estimado cliente:</presentacion>
4      <contenido>bla bla bla bla ...</contenido>
5      <firma>Don José Gutiérrez González</firma>
6  </carta>
```

1.- Lenguajes de Marcas. Introducción.

En la práctica, en un mismo documento pueden combinarse varios tipos diferentes de lenguajes de marcas.

Los lenguajes de marcas se pueden clasificar como sigue:

- **De presentación:** Define el formato del texto.
- **De procedimientos:** Orientado también a la presentación pero, en este caso, el programa que representa el documento debe interpretar el código en el mismo orden en que aparece.
- **Descriptivo o semántico:** Describen las diferentes partes en las que se estructura el documento pero sin especificar cómo deben representarse.

Algunos ejemplos de lenguajes de marcado agrupados por su ámbito de utilización son:

Documentación electrónica:

- **RTF (Rich Text Format):** Formato de Texto Enriquecido, fue desarrollado por Microsoft en 1987. Permite el intercambio de documentos de texto entre distintos procesadores de texto.
- **TeX:** Su objetivo es la creación de ecuaciones matemáticas complejas.
- **Wikitexto:** Permite la creación de páginas wiki en servidores preparados para soportar este lenguaje.
- **DocBook:** Permite generar documentos separando la estructura lógica del documento de su formato. De este modo, dichos documentos, pueden publicarse en diferentes formatos sin necesidad de realizar modificaciones en el documento original.

1.- Lenguajes de Marcas. Introducción.

Tecnologías de Internet:

- **HTML, XHTML:** (Hypertext Markup Language, eXtensible Hypertext Markup Language): Su objetivo es la creación de páginas web.
- **RSS:** Permite la difusión de contenidos web

Otros lenguajes especializados:

- **MathML (Mathematical Markup Language):** Su objetivo es expresar el formalismo matemático de tal modo que pueda ser entendido por distintos sistemas y aplicaciones.
- **VoiceXML (Voice Extended Markup Language):** tiene como objetivo el intercambio de información entre un usuario y una aplicación con capacidad de reconocimiento de habla.
- **MusicXML (Music Extended Markup Language):** Permite el intercambio de partituras entre distintos editores de partituras.

2.- Evolución de los lenguajes de marcas. Introducción

En los años 70 surgieron unos lenguajes informáticos, distintos de los lenguajes de programación, orientados a la gestión de información.

Con el desarrollo de los editores y procesadores de texto aparecieron los primeros lenguajes informáticos especializados en tareas de descripción y estructuración de información: los lenguajes de marcas.

Paralelamente, también, emergieron otros lenguajes informáticos, orientados a la representación, almacenamiento y consulta eficiente de grandes cantidades de datos: lenguajes y sistemas de bases de datos.

Los lenguajes de marcas surgieron, inicialmente, como lenguajes formados por el conjunto de códigos de formato que los procesadores de texto introducen en los documentos para dirigir el proceso de presentación (impresión) mediante una impresora.

Como en el caso de los lenguajes de programación, inicialmente estos códigos de formato estaban ligados a las características de una máquina, programa o procesador de textos concreto y, en ellos, inicialmente no había nada que permitiese al programador ("formateador" de documentos en este caso) abstraerse de las características del procesador de textos y expresar de forma independiente a éste la estructura y la lógica interna del documento.

2.- Evolución de los lenguajes de marcas. Introducción

Ejemplo:

```
1 <times 14><color verde><centrado> Este texto es un ejemplo para mostrar la utilización primitiva de las marcas</centrado></color></times 14>  
2 <color granate><times 10>< cursiva>Para realiza este ejemplo se utilizan etiquetas de nuestra invención. </ cursiva>  
3 Las partes importantes del texto pueden resaltarse usando la  
4 <negrita>negrita</negrita>, o el <subrayar>subrayado</subrayar></times 10></color>
```

Al imprimirlo se obtendría:

Este texto es un ejemplo para mostrar la utilización primitiva de las marcas

Para realiza este ejemplo se utilizan etiquetas de nuestra invención. Las partes importantes del texto pueden resaltarse usando la **negrita** , o el subrayado

Este es un ejemplo de marcado de un documento con el objetivo de presentar la información.

2.- Evolución de los lenguajes de marcas.

2.1.- GML (Generalized Markup Language).

Uno de los **problemas** que se conocen **desde hace décadas en la informática es la falta de estandarización en los formatos de información usados por los distintos programas.**

Para resolver este problema, en los años sesenta **IBM** encargó a Charles F. Goldfab la construcción de un sistema de edición, almacenamiento y búsqueda de documentos legales.

El formato de documentos que se creó como resultado de este trabajo fue **GML**, cuyo objetivo era describir los documentos de tal modo que el resultado fuese independiente de la plataforma y la aplicación utilizada.

2.- Evolución de los lenguajes de marcas.

2.2.- SGML (Standard Generalized Markup Language).

El formato GML evolucionó hasta que en 1986 dio lugar al estándar **ISO 8879** que se denominó **SGML**. Éste era un lenguaje muy complejo y requería de unas herramientas de software caras. Por ello su uso quedó relegado a grandes aplicaciones industriales. Ejemplo de SGML sencillo:

```
1  <email>
2      <remitente>
3          <persona>
4              <nombre> Pepito </nombre>
5              <apellido> Grillo </apellido>
6          </persona>
7      </remitente>
8      <destinatario>
9          <direccion> pinocho@hotmail.com </direccion>
10     </destinatario>
11     <asunto>¿quedamos?</asunto>
12     <mensaje> Hola, he visto que ponen esta noche la película que querías ver. ¿Te apetece ir?<
13 </email>
```

2.- Evolución de los lenguajes de marcas.

2.3.- HTML (HyperText Markup Language).

En 1989/90 Tim Berners-Lee creó el **World Wide Web** y se encontró con la necesidad de organizar, enlazar y compatibilizar gran cantidad de información procedente de diversos sistemas. Para resolverlo creó un lenguaje de descripción de documentos llamado **HTML**, que, en realidad, era una combinación de dos estándares ya existentes:

- **ASCII**: Es el formato que cualquier procesador de textos sencillo puede reconocer y almacenar. Por tanto es un formato que permite la transferencia de datos entre diferentes ordenadores.
- **SGML**: Lenguaje que permite dar estructura al texto, resaltando los títulos o aplicando diversos formatos al texto.

HTML es una versión simplificada de **SGML**, ya que sólo se utilizaban las instrucciones absolutamente imprescindibles. Era tan fácil de comprender que rápidamente tuvo gran aceptación, logrando lo que no pudo **SGML**.

HTML se convirtió en un estándar general para la creación de páginas web. Además, desde su creación, tanto las herramientas de software como los navegadores, que permiten visualizar páginas **HTML**, no han parado de mejorar.

A pesar de todas estas ventajas **HTML** no es un lenguaje perfecto, sus principales desventajas son:

- No soporta tareas de impresión y diseño.
- El lenguaje no es flexible, ya que las etiquetas son limitadas.
- No permite mostrar contenido dinámico.
- La estructura y el diseño están mezclados en el documento.

2.- Evolución de los lenguajes de marcas.

2.3.- HTML (HyperText Markup Language).

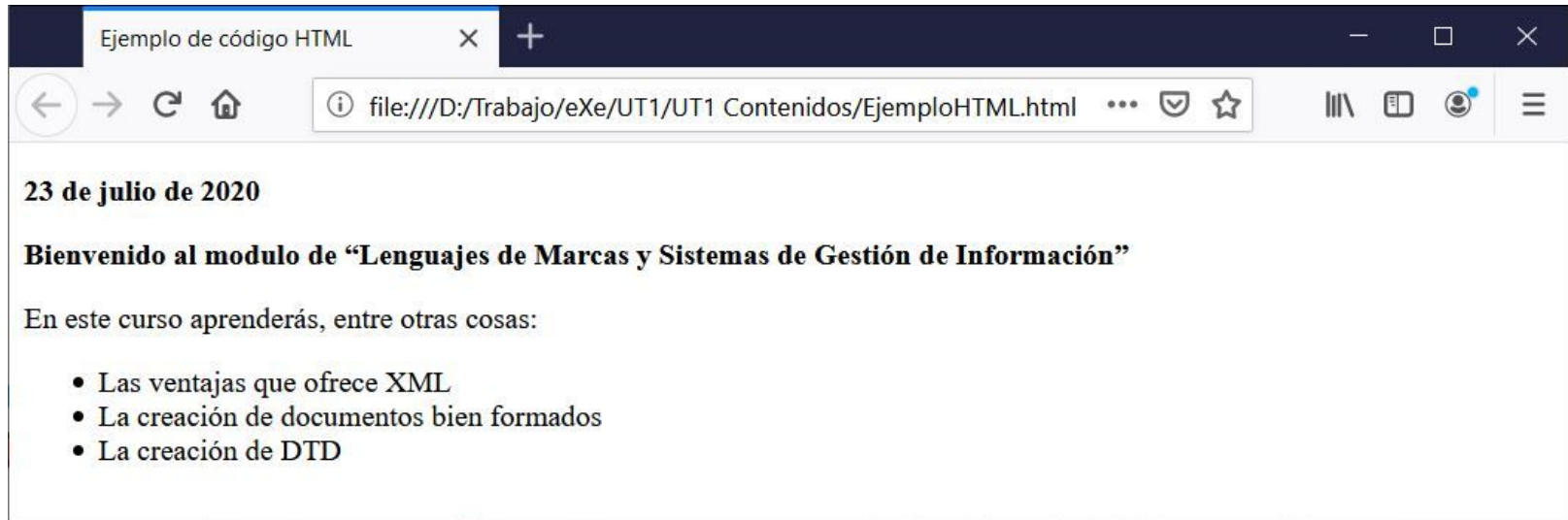
Ejemplo de documento HTML:

```
1  <html>
2    <head>
3      <title> Ejemplo de código HTML</title>
4    </head>
5    <body>
6      <p></p>
7      <p>
8        <b>23 de julio de 2020</b>
9      </p>
10     <p><b> Bienvenido al modulo de "Lenguajes de Marcas y Sistemas de Gestión de Informació
11     <p> En este curso aprender&aacute;s, entre otras cosas:<br/>
12     <ul>
13       <li>Las ventajas que ofrece XML </li>
14       <li>La creaci&oacute;n de documentos bien formados </li>
15       <li>La creaci&oacute;n de DTD</li>
16     </ul>
17   </p>
18 </body>
19 </html>
```

2.- Evolución de los lenguajes de marcas.

2.3.- HTML (HyperText Markup Language).

En el navegador se visualizaría así:



2.- Evolución de los lenguajes de marcas.

2.4.- XML (eXtensible Markup Language).

Para resolver estos problemas de **HTML** el [W3C](#) establece, en 1998, el estándar internacional **XML**, un lenguaje de marcas puramente estructural que **no incluye ninguna información relativa al diseño**. Se convirtió con rapidez en estándar para el intercambio de datos en la Web. A diferencia de **HTML**, en XML las etiquetas indican el significado de los datos en lugar del formato con el que se van a visualizar los datos.

XML es un metalenguaje caracterizado por:

- Permitir definir etiquetas propias.
- Permitir asignar atributos a las etiquetas.
- Utilizar un esquema para definir de forma exacta las etiquetas y los atributos.
- La estructura y el diseño son independientes.

En realidad **XML** es un conjunto de estándares relacionados entre sí y que son:

- **XSL**, eXtensible Style Language. Permite definir hojas de estilo para los documentos XML e incluye capacidad para la transformación de documentos.
- **XML Linking Language**, incluye Xpath, Xlink y Xpointer. Determinan aspectos sobre los enlaces entre documentos XML.
- **XML Namespaces**. Proveen un contexto al que se aplican las marcas de un documento de XML y que sirve para diferenciarlas de otras con idéntico nombre válidas en otros contextos.
- **XML Schemas**. Permiten definir restricciones que se aplicarán a un documento XML. Actualmente los más usados son las DTD.

2.- Evolución de los lenguajes de marcas.

2.4.- XML (eXtensible Markup Language).

Ejemplo de documento XML:

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <!DOCTYPE biblioteca">
3 <biblioteca>
4     <ejemplar tipo_ejem="libro" titulo="XML practico" editorial="Ediciones Eni">
5         <tipo> <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"></libro> </tipo>
6         <autor nombre="Sebastien Lecomte"></autor>
7         <autor nombre="Thierry Boulanger"></autor>
8         <autor nombre="Ángel Belinchon Calleja" funcion="traductor"></autor>
9         <prestado lector="Pepito Grillo">
10             <fecha_pres dia="13" mes="mar" año="2009"></fecha_pres>
11             <fecha_devol dia="21" mes="jun" año="2009"></fecha_devol>
12         </prestado>
13     </ejemplar>
14     <ejemplar tipo_ejem="revista" titulo="Todo Linux 101. Virtualización en GNU/Linux" editorial="
15         <tipo>
16             <revista>
17                 <fecha_publicacion mes="abr" año="2009"></fecha_publicacion>
18             </revista>
19         </tipo>
20         <autor nombre="Varios"></autor>
21         <prestado lector="Pedro Picapiedra">
22             <fecha_pres dia="12" mes="ene" año="2010"></fecha_pres>
23         </prestado>
24     </ejemplar>
25 </biblioteca>
```

2.- Evolución de los lenguajes de marcas.

2.5.- XML vs HTML.

Ejemplos de ficheros XML y HTML con contenido similar.

Ejemplo fichero con código XML

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <!DOCTYPE libro>
3 <libro>
4   <titulo>XML practico </titulo>
5   <autor>SebastienLecomte</autor>
6   <autor>Thierry Boulanger</autor>
7   <editorial>Ediciones Eni</editorial>
8   <isbn>978-2-7460-4958-1</isbn>
9   <edicion>1</edicion>
10  <paginas>347</paginas>
11 </libro>
```

Ejemplo de fichero con código HTML

```
1 <html>
2   <head>
3     <title>Libro</title>
4   </head>
5   <body>
6     <h3>XML practico</h3><br>
7     <p>autores: Sebastien Lecomte,
8       Thierry Boulanger</p>
9     <ul>
10      <li>editorial: Ediciones Eni</li>
11      <li>isbn:978-2-7460-4958-1</li>
12      <li>edicion: 1 </li>
13      <li>paginas: 347</li>
14    </ul>
15  </body>
16 </html>
```

2.- Evolución de los lenguajes de marcas.

2.5.- XML vs HTML.

Así se visualizará cada uno en un navegador:

```
<libro>  
  <titulo>XML practico </titulo>  
  <autor>SebastienLecomte</autor>  
  <autor>Thierry Boulanger</autor>  
  <editorial>Ediciones Eni</editorial>  
  <isbn>978-2-7460-4958-1</isbn>  
  <edicion>1</edicion>  
  <paginas>347</paginas>  
</libro>
```



XML practico

autores: Sebastien Lecomte, Thierry Boulanger

- editorial: Ediciones Eni
- isbn:978-2-7460-4958-1
- edicion: 1
- paginas: 347

2.- Evolución de los lenguajes de marcas.

2.6.- Comparación de XML con SGML.

XML	SGML
Uso sencillo	Uso complejo
Trabaja con  documentos bien formados. No exige que estén validados	Solo trabaja con  documentos válidos
Facilita el desarrollo de aplicaciones de bajo coste	Su complejidad hace que las aplicaciones informáticas para procesar SGML sean muy costosas
Es muy utilizado en informática y en más áreas de aplicación	Solo se utiliza en sectores muy específicos
Compatibilidad e integración con <u>HTML</u>	No hay una compatibilidad con HTML definida
Formato y estilos fáciles de aplicar	Formateo y estilos relativamente complejos
No usa etiquetas opcionales	

2.- Evolución de los lenguajes de marcas.

2.7.- Etiquetas.

Los lenguajes de marcas utilizan una serie de etiquetas intercaladas en un documento de texto sin formato. Dichas etiquetas serán posteriormente interpretadas por los intérpretes del lenguaje y ayudan al procesado del documento.

Las etiquetas **se escriben encerradas entre ángulos**, es decir < y >. Normalmente, se utilizan dos etiquetas: una de inicio y otra de fin para indicar que ha terminado el efecto que queríamos presentar. La única diferencia entre ambas es que **la de cierre lleva una barra inclinada "/"** antes del código.

<etiqueta>texto que sufrirá las consecuencias de la etiqueta</etiqueta>

2.- Evolución de los lenguajes de marcas.

2.7.- Etiquetas.



Ejemplo: Etiqueta HTML de subrayado (Underline)

```
1 | <u>Esto está subrayado</u>
```

En el navegador el texto se verá:

Esto está subrayado

Las últimas especificaciones emitidas por el [W3C](#) indican la necesidad de que vayan escritas **siempre en minúsculas** para considerar que el documento está correctamente creado.

2.- Evolución de los lenguajes de marcas.

2.8.- Herramientas de edición.

Para trabajar en XML es necesario editar los documentos y luego procesarlos, por tanto tenemos dos tipos de herramientas:

- **Editores XML:** Una característica de los lenguajes de marcas es que se basan en la utilización de ficheros de texto plano por lo que basta utilizar un procesador de texto normal y corriente para construir un documento XML.

Para crear documentos XML complejos e ir añadiendo datos es conveniente usar algún editor XML. Estos nos ayudan a crear estructuras y etiquetas de los elementos usados en los documentos, además algunos incluyen ayuda para la creación de otros elementos como DTD, hojas de estilo CSS o XSL, etc ...

2.- Evolución de los lenguajes de marcas.

2.8.- Herramientas de edición.

- **Procesadores XML:** permiten leer los documentos XML y acceder a su contenido y estructura. Un procesador es un conjunto de módulos de software, entre los que se encuentra un parser o analizador de XML, que comprueba que el documento cumple las normas establecidas para que pueda abrirse.

Los procesadores XML pueden obligar a trabajar sólo con documentos de tipo válido (entonces se denominan "validadores") o pueden sólo exigir que el documento esté bien formado ("no validadores"), o ambas cosas

Para publicar un documento XML en Internet se pueden utilizar los procesadores XSLT, que permiten generar archivos HTML a partir de documentos XML.

XML también se puede utilizar para el intercambio de datos entre aplicaciones. En este caso, hay que recurrir a motores independientes, que se ejecutan sin que nos demos cuenta. Por ejemplo [JAXP](#) de Oracle o [JAXB](#) dentro del JDK de Java (librerías que cuelgan de javax).

3.- XML, estructura y sintaxis.

Introducción.

El **XML**, o Lenguaje de Etiquetas Extendido, es lenguaje de etiquetas, creadas por el programador, que estructuran y guardan de forma ordenada la información. No indica cómo deben presentarse los datos por sí mismo, solamente organiza la estructura.

El **XML** ahorra tiempos de desarrollo y proporciona ventajas, dotando a webs y a aplicaciones de una forma realmente potente de guardar, procesar y transmitir la información.

Al igual que en **HTML** un documento **XML** es un documento de texto, en este caso con extensión ".xml", compuesto de parejas de etiquetas, estructuradas en árbol, que describen una función en la organización del documento, que puede editarse con cualquier editor de texto y que es interpretado por los navegadores Web.

3.- XML, estructura y sintaxis.

Introducción.

El marcado en **XML** son etiquetas que se añaden a un texto para estructurar el contenido del documento. Esta información extra (o meta-información) permite a los ordenadores "interpretar" la información. El marcado es todo lo que se sitúa entre los **caracteres** "<" y ">" o "&" y ";"

Los datos carácter son los que forman la verdadera información del documento **XML**.

El marcado puede ser tan rico como se quiera. Puede ser interesante detectar necesidades futuras y crear documentos con una estructura fácilmente actualizables.

Los documentos XML pueden tener **comentarios**, que no son interpretados por el interprete XML. Estos se incluyen entre las cadenas "<!--" y "-->", pueden estar en cualquier posición en el documento salvo:

- Antes del prólogo.
- Dentro de una etiqueta.

Los documentos XML pueden estar formados por una parte opcional llamada prólogo y otra parte obligatoria llamada ejemplar (lo vemos a continuación).

3.- XML, estructura y sintaxis.

3.1.- El prólogo

Si se incluye, el prólogo **debe preceder al ejemplar del documento**. Su inclusión facilita el procesado de la información del ejemplar. El prólogo está dividido en dos partes:

- **La declaración XML:** En el caso de incluirse ha de ser la primera línea del documento, de no ser así se genera un error que impide que el documento sea procesado.

El prólogo puede tener tres funciones:

- *Declaración la versión de XML usada para elaborar el documento. Para ello se utiliza la etiqueta:*

```
<?xml versión= "1.0" ?>
```

- *Declaración de la codificación empleada para representar los caracteres. Determina el conjunto de caracteres que se utiliza en el documento. Para ello se escribe:*

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


3.- XML, estructura y sintaxis.

3.1.- El prólogo

- *Declaración de la autonomía del documento.* Informa de si el documento necesita de otro para su interpretación. Para declararlo hay que definir el prólogo completo:

```
<?xml versión= "1.0" encoding="iso-8859-1" standalone="yes" ?>
```

En este caso, el documento es independiente, de no ser así el atributo standalone hubiese tomado el valor "no".

- **La declaración del tipo de documento**, define qué tipo de documento estamos creando para ser procesado correctamente. Toda declaración de tipo de documento comienza por la cadena:

```
<!DOCTYPE Nombre_tipo ...>
```

3.- XML, estructura y sintaxis.

3.2.- El ejemplar. Los elementos.

Es la parte más importante de un documento **XML**, ya que contiene los datos reales del documento. Está formado por elementos anidados.

Los elementos son los distintos bloques de información que permiten definir la estructura de un documento XML.

Están delimitados por una etiqueta de apertura y una etiqueta de cierre.

A su vez, los elementos pueden estar formados por otros elementos y/o por atributos.

En realidad, **el ejemplar es el elemento raíz (root) de un documento XML**. Todos los datos de un documento XML han de pertenecer a un elemento del mismo.

Los nombres de las etiquetas han de ser autodescriptivos, lo que facilita el trabajo que se hace con ellas.

3.- XML, estructura y sintaxis.

3.2.- El ejemplar. Los elementos.



Ejemplo

Dado el siguiente código XML...

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <!DOCTYPE libro>
3  <libro>
4      <titulo>XML practico </titulo>
5      <autor>Sebastien Lecomte</autor>
6      <autor>Thierry Boulanger</autor>
7      <editorial>Ediciones Eni</editorial>
8      <isbn>978-2-7460-4958-1</isbn>
9      <edicion>1</edicion>
10     <paginas>347</paginas>
11 </libro>
```

El ejemplar es el elemento `<libro>`, que a su vez está compuesto de los elementos `<autor>`, `<editorial>`, `<isbn>`, `<edicion>` y `<paginas>`

3.- XML, estructura y sintaxis.

3.2.- El ejemplar. Los elementos.

La formación de elementos ha de cumplir ciertas normas para que queden perfectamente definidos y que el documento XML al que pertenecen pueda ser interpretado por los procesadores XML sin generar ningún error fatal. Dichas reglas son:

- En todo documento **XML** debe existir **un elemento raíz, y sólo uno.**
- Todos los elementos tienen una **etiqueta de inicio y otra de cierre**. En el caso de que en el documento existan **elementos vacíos**, se pueden sustituir las etiquetas de inicio y cierre por una de elemento vacío. Ésta se construye como la etiqueta de inicio, pero sustituyendo el carácter ">" por "/>". Es decir, **<elemento></elemento>** puede sustituirse por: <elemento/>
- Al anidar elementos hay que tener en cuenta que no puede cerrarse un elemento que contenga algún otro elemento que aún no se haya cerrado.
- Los **nombres de las etiquetas de inicio y de cierre de un mismo elemento han de ser idénticos**, respetando las mayúsculas y minúsculas. Pueden ser cualquier cadena alfanumérica que no contenga espacios y no comience ni por el carácter dos puntos, ":", ni por la cadena "xml" ni ninguna de sus versiones en que se cambien mayúsculas y minúsculas ("XML", "XmL", "xML",...).

3.- XML, estructura y sintaxis.

3.2.- El ejemplar. Los elementos.

- El contenido de los elementos no puede contener la cadena "]]>" por compatibilidad con SGML. Además no se pueden utilizar directamente los caracteres mayor que, >, menor que, <, ampersand, &, dobles comillas, ", y apostrofe, '. En el caso de tener que utilizar estos caracteres se sustituyen por las siguientes cadenas:

Carácter	Cadena
>	>
<	<
&	&
"	"
'	'

- Para utilizar caracteres especiales, como £, ©, ®,... hay que usar las expresiones &#D; o &#H; donde D y H se corresponden respectivamente con el número decimal o hexadecimal correspondiente al carácter que se quiere representar en el código UNICODE. Por ejemplo, para incluir el carácter de Euro, €, se usarían las cadenas € o €

3.- XML, estructura y sintaxis.

3.2.- El ejemplar. Atributos.

Los atributos permiten **añadir propiedades a los elementos** de un documento.

Los atributos **no pueden organizarse en ninguna jerarquía**, no pueden contener ningún otro elemento o atributo y no reflejan ninguna estructura lógica.

No se debe utilizar un atributo para contener información susceptible de ser dividida.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <!DOCTYPE biblioteca >
3 <biblioteca>
4   <ejemplar tipo_ejem="libro" titulo="XML práctico" editorial="Ediciones Eni">
5     <tipo> <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"></libro> </tipo>
6     <autor nombre="Sebastien Lecomte"></autor>
7     <autor nombre="Thierry Boulanger"></autor>
8     <autor nombre="Angel Belinchon Calleja" funcion="traductor"></autor>
9     <prestado lector="Pepito Grillo">
10       <fecha_pres dia="13" mes="mar" año="2009"></fecha_pres>
11       <fecha_devol dia="21" mes="jun" año="2009"></fecha_devol>
12     </prestado>
13   </ejemplar>
14 </biblioteca>
```

Como se observa en el ejemplo, los atributos se definen y dan valor **dentro de una etiqueta de inicio o de elemento vacío**, a continuación del nombre del elemento o de la definición de otro atributo, siempre separado de ellos por un espacio. **Los valores del atributo van precedidos de un igual** que sigue al nombre del mismo y tienen que definirse **entre comillas simples o dobles**.

Los **nombres de los atributos** han de cumplir las mismas reglas que los de los elementos, y **no pueden contener el carácter menor que, <**.

4.- Documentos XML bien formados.

Todos los documentos **XML** deben verificar las reglas sintácticas que definen la recomendación del **W3C** para el estándar **XML**. Esas normas básicas son:

- El documento ha de tener definido un **prólogo** con la declaración xml completa.
- Existe **un único elemento raíz para cada documento**: es un solo elemento en el que todos los demás elementos y contenidos se encuentran anidados.
- Hay que cumplir las reglas sintácticas del lenguaje XML para definir los distintos elementos y atributos del documento.

¿Está bien formado el siguiente documento XML?

```
1  <?xml version="1.0"?>
2  <mensaje>
3      <destinatario>Tomas</ destinatario>
4      <remitente>Juan</ remitente>
5      <asunto>
6          <contenido> No olvides ir a recogerme al aeropuerto mañana por la mañana!</contenido>
7  </mensaje>
```

5.- Utilización de espacios de nombres en XML.

- Permiten definir la pertenencia de los elementos y los atributos de un documento XML al contexto de un vocabulario XML.
- De este modo se resuelven las ambigüedades que se pueden producir al juntar dos documentos distintos, de dos autores diferentes, que han utilizado el mismo nombre de etiqueta para representar cosas distintas.
- Los espacios de nombres, también conocidos como "name spaces", permiten dar un nombre único a cada elemento, indexándolos según el nombre del vocabulario adecuado. Además están asociados a un URI que los identifica de forma única.
- En el documento, las etiquetas ambiguas se sustituyen por otras en las que el nombre del elemento está precedido de un prefijo, que determina el contexto al que pertenece la etiqueta, seguido de dos puntos, ":". Esto es:

<prefijo:nombre_etiqueta></prefijo:nombre_etiqueta>

Esta etiqueta se denomina "nombre cualificado". Al definir el prefijo hay que tener en cuenta que no se pueden utilizar espacios ni caracteres especiales y que no puede comenzar por un dígito.

5.- Utilización de espacios de nombres en XML.

Antes de poder utilizar un prefijo de un espacio de nombres, para resolver la ambigüedad de dos o más etiquetas, es necesario declarar el espacio de nombres, es decir, asociar un índice con el URI asignado al espacio de nombres, mediante un **atributo especial xmlns**. Esto se hace entre el prólogo y el ejemplar de un documento XML y su sintaxis es la siguiente:

<conexion>://<direccionservidor>/<apartado1>/<apartado2>/...

Ejemplo:

XML de alumnos:

```
1 <?xml version="1.0" encoding="iso-8859-1" standalone="yes"
2 <!DOCTYPE alumnos>
3 <alumnos>
4   <nombre>Fernando Fernández González</nombre>
5   <nombre>Isabel González Fernández</nombre>
6   <nombre>Ricardo Martínez López</nombre>
7 </alumnos>
```

XML de profesores

```
1 <?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
2 <!DOCTYPE profesores>
3 <profesores>
4   <nombre>Pilar Ruiz Pérez</nombre>
5   <nombre>Tomás Rodríguez Hernández</nombre>
6 </profesores>
```

5.- Utilización de espacios de nombres en XML.

En el ejemplo anterior, si uniéramos los dos documentos en uno único, sin usar espacios de nombres, no se distinguiría los profesores de los alumnos ya que en los dos casos la etiqueta <nombre> se llama igual.

Para resolverlo necesitamos definir un espacio de nombres para cada contexto.

```
1  <?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
2  <!DOCTYPE miembros>
3  <alumnos xmlns:alumnos="http://DAM/alumnos">
4  <profesores xmlns:profesores="http://DAM/profesores">
5  <asistentes>
6      <alumnos:nombre>Fernando Fernández González</alumnos:nombre>
7      <alumnos:nombre>Isabel González Fernández</alumnos:nombre>
8      <alumnos:nombre>Ricardo Martínez López</alumnos:nombre>
9      <profesores:nombre>Pilar Ruiz Pérez</profesores:nombre>
10     <profesores:nombre>Tomás Rodríguez Hernández</profesores:nombre>
11 </asistentes>
```