TEMA 3. LENGUAJES PARA ALMACENAMIENTO Y TRANSMISIÓN DE INFORMACIÓN. XML

Lenguajes de Marcas
CFGS DAW 1

Pascual Ligero
pascual.ligero@ceedcv.es
2016/2017

Versión:171108.1945

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

ÍNDICE DE CONTENIDO

UD02. XML

1. INTRODUCCIÓN. XML

XML (Extensible MarkupLanguage, lenguaje de marcas extensible) es la última palabra de moda en Internet. También es una tecnología que madura rápidamente con múltiples aplicaciones en el mundo real, particularmente para la gestión, visualización y organización de los datos. En conjunto con su lenguaje de visualización XSL y el modelo de objeto documento DOM, es una tecnología esencial para cualquier usuario de lenguajes de marcas en la red o en la intranet.

Se trata de un formato de archivos de texto con marcado que deriva del original SGML, pero que le ha superado añadiendo otro tipo de reglas y de forma de trabajar.

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 cualquiera)
- 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 hay que cambiar el estándar y los navegadores se deben de adaptar a los cambios.

Por ello al crear XML se plantearon estos objetivos:

(1) Debía de ser similar a HTML (de hecho se basa en el lenguaje SGML base para el formato HTML)
(2) Debía de ser extensible, es decir que sea posible añadir nuevas etiquetas sin problemas. Esa es la base del lenguaje XML.
(3) Debía de tener unas reglas concisas y fáciles, además de estrictas.
(4) Debía de 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.

(5) Debía ser fácil de leer por los humanos y fácil crear procesadores XML software, llamados parsers (un procesador o analizador sintáctico, con él no solamente podemos comprobar si nuestros documentos están bien formados o son válidos, sino que también podemos incorporarlos a nuestras aplicaciones, de manera que estas puedan manipular y trabajar con documentos XML)

Básicamente XML es SGML, es decir una persona que sepa SGML no tendrá ningún problema en aprender XML, las bases son las mismas pero XML elimina gran parte de su complejidad. De hecho se dice que XML es un subconjunto de SGML; es SGML pero restringiendo o eliminando ciertas normas.

Por otro lado XML sirve para lo mismo que SGML: para diseñar lenguajes de mediante marcas. Es decir, XML define tipos de documentos de forma que junto a la información del documento hay una serie de etiquetas que sirven para clarificar lo que significa la información.

De esa manera XML (como ya hizo SGML) es un lenguaje que permite especificar otros lenguajes. Entendiendo como lenguajes en este caso, al conjunto de etiquetas se pueden utilizar en un documento; no solo qué etiquetas sino en qué orden y de qué forma se pueden utilizar.

1.1 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 de facto 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.
- 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.
- 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.

1.2 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 es ventajosa ya que facilita su aprendizaje), una escalabilidad (que permite que jamás se quede obsoleto el lenguaje) y una serie de tecnologías relacionadas más poderosas.

La razón del fracaso de HTML como estándar se debe a que cada navegador impone etiquetas propias, es decir que no hay un estándar real. Con XML no es posible esta situación, ya que la propia naturaleza del lenguaje no la hace posible, 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.

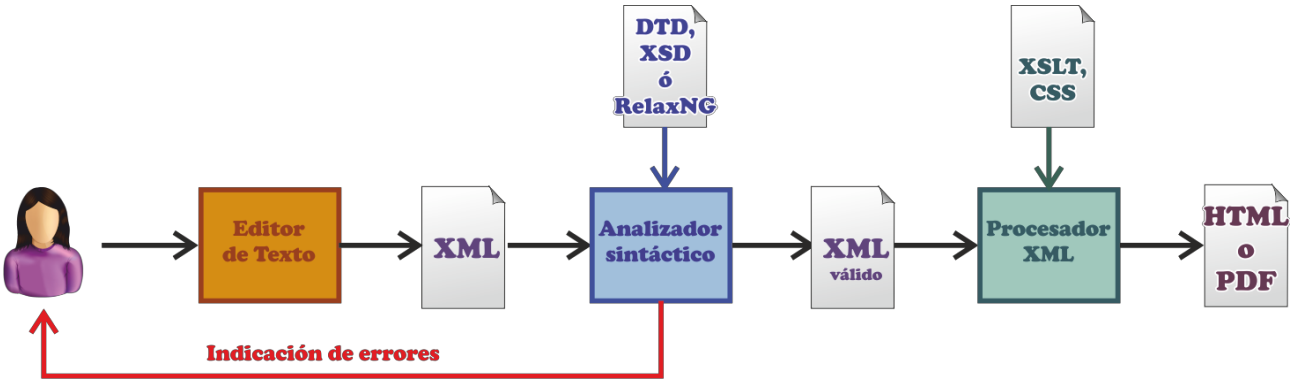
1.3 Tecnologías relacionadas con XML

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. Hojas de estilo en cascada. Permiten dar formato a los documentos XML o HTML.
- **XSLT**. Sirve para lo mismo que CSS, dar formato a un documento XML. Tiene muchas más posibilidades que CSS.
- **XQuery**. Permite consultar datos de los documentos XML, manejándolo 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



1.4 Software para producir XML

En principio XML se puede escribir desde cualquier editor de texto plano (como el bloc de notas de Windows o el editor vi de Linux). Pero es más interesante 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:

(1) 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++, SublimeText o Brackets.

(2) Un analizador sintáctico o parser, programa capaz de entender y validar el lenguaje XML. Apache Xerces es quizá el más popular validador de documentos XML.

(3) Un procesador XML que sea capaz de producir un resultado visual sobre el documento XML. 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

Hay entornos que tienen estas prestaciones en un único paquete.

2. ESTRUCTURA XML

2.1 Cuestiones básicas

Los documentos XML en definitiva son documentos de lenguajes de marcas, donde hay texto normal 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; de hecho la función de XML es definir tipos de documentos etiquetados.

Ejemplo de XML:

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

El código es similar a HTML, sólo que las etiquetas se deciden según nos interese. Pero el funcionamiento el mismo:

- Las etiquetas tienen cierre que se indica con el signo / antes del nombre de la etiqueta
- Las etiquetas afectan al texto (y otras etiquetas) que están entre la apertura y el cierre

Por otro lado XML llama **elemento** a las etiquetas; mejor dicho, elemento sería tanto la etiqueta como lo que contiene.

En cualquier caso las normas son:

- 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 y >
- 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 <!-- 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)

2.2 Estructura de un documento XML

Los documentos XML se dividen en:

- **Prólogo**. Se trata de la primera zona del documento y sirve para describir qué tipo de documento es. Es similar al apartado **head** de **HTML**. Puede contener
  - **Declaración del documento**, que permite indicar el tipo de documento XML que es.
  - **Instrucciones** para el procesado del documento
  - **Comentarios**
  - **Indicación del documento DTD, XSD o Relax NG** para comprobar si el mismo es válido según las reglas impuestas por dicho documento.
  - Indicación de otros documentos que afectan al actual, como por ejemplo los documentos **XSLT** que permiten especificar la forma en la que el documento se debe mostrar en pantalla.
- **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**
  - **Texto normal**
  - **Entidades**
  - **Comentarios**

2.3 Reglas para los nombres

En XML los elementos, atributos, ... tienen un nombre (más correctamente llamado **identificador**), 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. 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.

2.4 Elementos del prólogo

Declaración XML

Se trata de la primera línea de un documento XML e indica el tipo de documento XML que es (y así poder validar el mismo). En realidad es opcional, pero es muy recomendable. Es:

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

Indica la versión XML del documento y la codificación (utf-8 es la habitual). Existen mas versiones de xml, como la 1.1 que permite usar carácteres no ASCII estándar en las definiciones de las etiquetas. Nosotros nos quedamos con la 1.0 que vale para el 99 % de los casos.

Instrucciones de procesamiento

Un documento XML puede incluir instrucciones de este tipo que indica un documento para validar el 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 (especificar la forma en la que los datos se muestran por pantalla por ejemplo).

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 tanto de apertura como de cierre.

Elementos

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

Hay una confusión entre lo que es un elemento y lo que es una etiqueta. En este caso por ejemplo:

```
<nombre>Juan</nombre>
```

- <nombre> Es un etiqueta de apertura
- </nombre> Es una etiqueta de cierre
- <nombre>Juan</nombre> Es un elemento (el elemento nombre)
- Juan 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 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; siempre se debe cerrar el último elemento que se abrió. **Esto es un error**:

```
<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 />
```

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. Ejemplo:

```
<persona complejidad=”alta”>
```

```
        <nombre>Juan</nombre>
        <apellido>López</apellido>
    </persona>
```

Un elemento puede contener varios atributos:

```
<persona privacidad="alta" tipo="autor">
        <nombre>María</nombre>
        <apellido>García</apellido>
</persona>
```

Un atributo es una información complementaria asociada a un elemento. 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>
```

Al diseñar un documento XML puede surgir la duda de cuándo usar elementos y cuándo atributos.

El ejemplo anterior, en otro contexto, quizás se hubiese pensado formarlo solo por parte de elementos, es decir, con una estructura similar a:

```
<ingrediente>
        <tipo> tomates</tipo>
        <peso> 5,73 </peso>
        <unidad> gramos </gramos>
        <precision> 0,01 </precision>
</ingrediente>
```

En este caso está claro que es preferible la primera opción, ya que parece lógico que cada elemento peso incluya como atributos la unidad de medida y la precisión. En otros casos es más dudoso.

Aunque no se puede dar un criterio válido para todos los casos sí 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.

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

Texto

El texto como se comentó antes 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 < y > por ejemplo

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. Formato:

<! [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.

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:

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

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

2.5 Jerarquías en XML

En este punto vamos a analizar cómo se estructuran los datos en un documento XML. Cuando debemos tratar con grandes cantidades de información, o incluso cantidades no tan grandes.

Usualmente es mejor agruparlas en subtemas relacionados en lugar de tener toda la información en un único gran grupo. Por ejemplo, este capítulo está dividido en subtemas que a su vez se subdividen en párrafos. Un formulario de impuestos se divide en subsecciones, a lo largo de páginas múltiples. Esto hace que la información resulte más comprensible y también más accesible.

Los desarrolladores de software han utilizado este paradigma durante muchos años, usando una estructura denominada modelo de objeto. En este modelo de objeto toda la información que está siendo modelada se subdivide en varios objetos y estos, por su parte se agrupan en una jerarquía.

XML también agrupa la información en jerarquías. Los elementos de nuestros documentos se ordenan por medio de relación padre/hijo o hermano/hermano. Estos componentes se denominan elementos.

Consideremos nuestro ejemplo <name>, mostrado jerárquicamente:

- <name>
  - <first>
    - “John”

- <midle>
  - “Fitzjerald Johansen”
- <last>
  - “Doe”

<name> es padre de <first>, que, por su parte, es hijo de <name>. <first>, <midle> y <last> son todos hermanos (es decir, son todos hijos de <name>). Observar también que el texto es también hijo de un elemento. Por ejemplo “John” es hijo de <first>.

Esta estructura también se denomina **árbol**. Todas las partes del árbol que contiene **hijos** se denominan ramas, mientras que las partes que no tienen hijos se denominan **hojas**.

Dependiendo del contenido de los elementos se pueden clasificar en:

- Elementos con **contenido de elemento**. Aquellos que poseen hijos que son a su vez otros elementos. En nuestro ejemplo <name>
- Elementos de **contenido simple**. Aquellos que solo contienen texto. En nuestro ejemplo <first>, <midle> y <last>.
- Elementos de **contenido combinado**. Cuando poseen tanto texto como otros elementos.
  - En nuestro ejemplo no hay ningún caso. Pero, por ejemplo:

```
‘<doc>
        <parent>
                This is some
                <em>text</em>
                in my element
        </parent>
</doc>’.
```

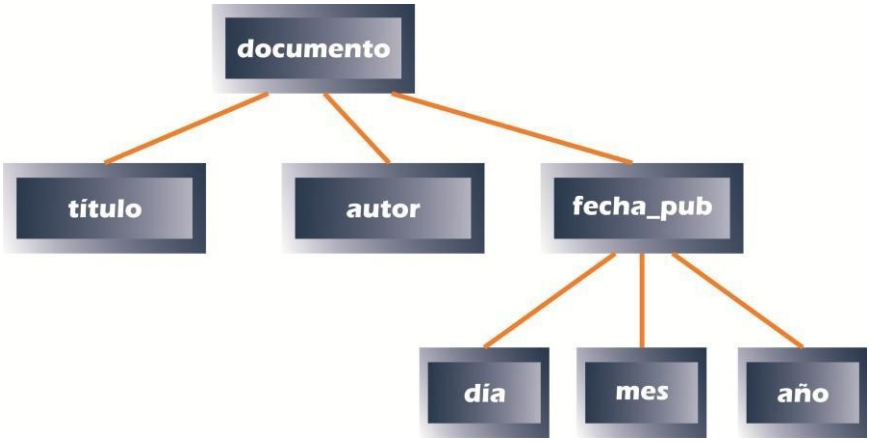
En este ejemplo <parent> tiene tres hijos:

- Un hijo de texto “this is some”
- Un hijo
- Otro hijo de texto “in my element”

Las relaciones también se pueden definir haciendo la analogía del árbol un poco más compleja. <doc> es el ancestro de ; es un descendiente de <doc>.

Después que se comprendan las relaciones jerárquicas entre los elementos (y el texto que contiene), se tendrá una idea más completa de la naturaleza de XML. También estará mejor preparado para trabajar con algunas de las otras tecnologías relacionadas con XML, que hacen amplio uso de este paradigma.

Gráficamente se representa de la siguiente manera:



3. DOCUMENTOS BIEN FORMADOS

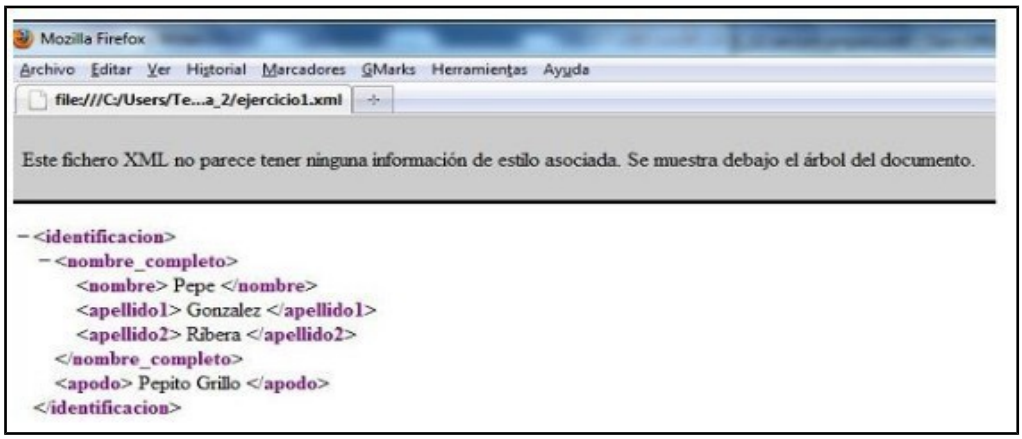
Sin la presencia de un DTD, no se puede comprobar la validez de un documento, sólo se puede comprobar si está bien formado. A continuación se enumeran algunas reglas que deben cumplir un documento para que se le considere **bien formado**:

- Dentro del texto común 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 el texto normal, 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.
- En cualquier caso todos los caracteres escritos en el documento XML forman parte del mismo, será una cuestión posterior si se tienen en cuenta o no para presentar los datos del documento XML en pantalla o impresión.
- Los elementos deben esta anidados adecuadamente., se cierran primero los últimos elementos abiertos.
- 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.
- 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.

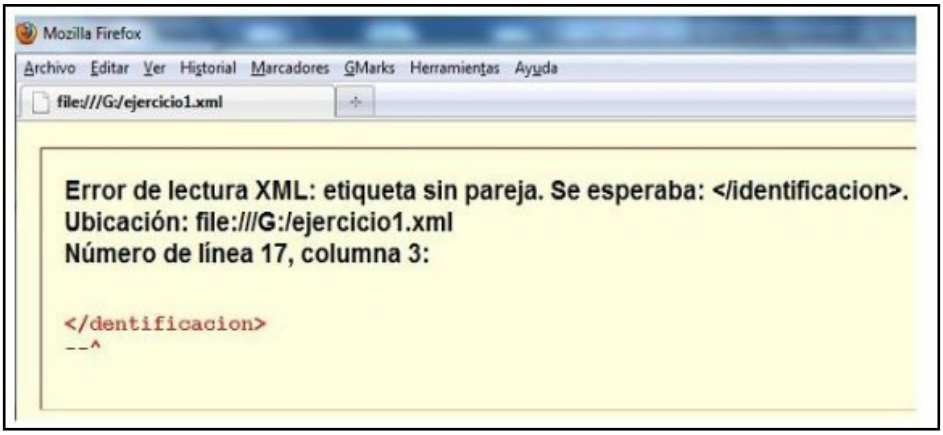
La comprobación de que un documento está bien formado la podemos hacer simplemente con el navegador. Si editamos el ejemplo 1 con un editor de texto plano

(bloc de notas, notepad, gedit, etc.,), sin la línea de DOCTYPE que es para llamar a la validación, como veremos en el tema siguiente, y luego lo abrimos con un navegador (Firefox) visualizamos el siguiente resultado:

```
<identificacion>
    <nombre_completo>
        <nombre> Pepe </nombre>
        <apellido1> González </apellido1>
        <apellido2> Ribera </apellido2>
    </nombre_completo>
    <apodo> Pepito Grillo </apodo>
</identificacion>
```

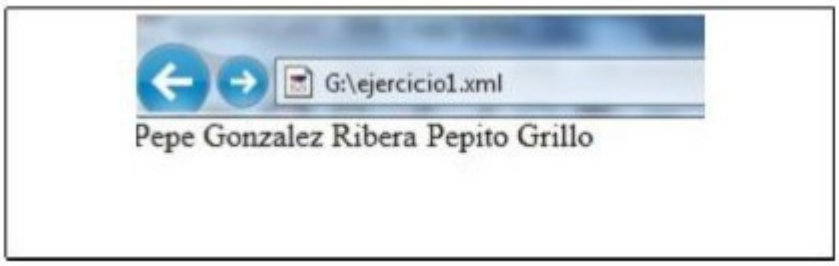


Sin embargo, si hubiésemos cometido cualquier error, como cambiar el nombre de cierre de una etiqueta obtenemos:



Este tratamiento de visualización será distinto en cada navegador, incluso muchas veces cambia de una versión a otra del mismo navegador. Por ejemplo, en el Explorer 9 el ejemplo correcto sería validado mostrando también la estructura del fichero:

Pero al visualizar el ejemplo con errores, simplemente nos visualiza el contenido de las etiquetas, sin darnos más explicación del error:



Por lo tanto y para unificar criterios a partir de ahora nos referiremos siempre a los resultados utilizando Mozilla Firefox o Chrome.

4. BIBLIOGRAFÍA

http://jorgesanchez.net/#lmsgi