XML Bien Formado

Para crear un documento XML, debemos conocer las reglas sintácticas que establece la recomendación de XML (en este caso la 1.0).

Llamamos documento bien formado a un documento XML que cumple las reglas sintácticas de la recomendación de XML.

1. **La primera regla es la estructura de una etiqueta:**

   Están las etiquetas de inicio (<etiqueta atrib1="val1" ... atribn="valn">), de cierre (</etiqueta>) y las vacías (<etiqueta atrib1="val1" ... atribn="valn"/>) que incluyen la barra de cierre al final.

   Un elemento está formado por todo el código desde la etiqueta de inicio hasta la etiqueta de cierre. Por ejemplo, un elemento puede ser el siguiente:

```
<nombre> José </nombre>
```

   y también puede ser un elemento el siguiente:

```
<persona>
        <nombre> José </nombre>
        <apellido> Pérez </apellido>
</persona>
```

   La única diferencia es que en el primer elemento, solo hay datos entre la etiqueta de inicio y la de cierre, y el segundo elemento tiene a su vez otros elementos en su interior.

   El texto entre la etiqueta de inicio y de cierre de un elemento es llamado "contenido del elemento".

   A menudo, el contenido entre las etiquetas son solamente datos. Nos referimos a ellos como PCDATA (Parsed Character Data, o sea, Datos tipo Carácter Analizados Sintácticamente). Es un término extraño, heredado de SGML (si leiste la historia de XML, sabrás lo que es SGML).

   Por otra parte, hay que tener en cuenta que, detrás del nombre de una etiqueta se permite escribir un espacio en blanco o un salto de línea. Por ejemplo, sintácticamente es correcto escribir:

```
<ciudad  >Pamplona</ciudad
>
```

   Ahora bien, no puede haber un salto de línea o un espacio en blanco antes del nombre de una etiqueta, ni tampoco puede haber un espacio entre el símbolo < y / de la etiqueta de cierre:

```
<
ciudad>Pamplona</ ciudad>
```

2. Las etiquetas tienen que estar correctamente anidadas.

Esto quiere decir, abiertas y cerradas en el orden correcto unas dentro de otras. Por ejemplo, el siguiente anidamiento es incorrecto:

```
<p> Un texto <strong> resaltado <em> en cursiva </strong> </em> mal anidado </p>
```

Su correspondiente correcto sería:

```
<p> Un texto <strong> resaltado <em> en cursiva </em> </strong> bien anidado </p>
```

3. Los documentos XML solo tienen un elemento raíz.

Por ejemplo el siguiente documento XML es incorrecto:

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

En cambio el siguiente sí sería correcto:

```
<nombres>
        <nombre> Juan </nombre>
        <nombre> María </nombre>
</nombres>
```

4. Los elementos deben seguir las convenciones de nombres de XML.

Estas convenciones, son las siguientes:

- Los nombres pueden empezar con letras o guion bajo(_), pero no con números u otros símbolos de puntuación.
- Tras un primer carácter o guion bajo, se pueden añadir números, guiones altos o bajos y puntos.
- Los nombres no pueden tener espacios en su interior.
- Los nombres no pueden contener el símbolo ":", ya que está reservado para ciertas funciones.
- Los nombres no pueden empezar con la cadena "xml" en minúscula, mayúscula o mezcla de ellas.
- Es muy conveniente evitar espacios innecesarios al final de la etiqueta.
- Los nombres de los elementos pueden contener caracteres que no forman parte de la lengua inglesa (como á,ò, ü, ç), pero se desaconseja su uso por poder acarrear problemas en la codificación.

Estos son algunos ejemplos de etiquetas mal construidas, por no seguir las recomendaciones anteriores:

```
<xml-tag>
 <123>
<mola=xml>
<mi etiqueta>
```

EJEMPLO Los siguientes elementos no están escritos correctamente por incumplir alguna regla de sintaxis:

```
<Ciudad>Pamplona</ciudad>
<día>18</dia>
<mes>6<mes/>
<ciudad>Pamplona</finciudad>
<_rojo>
<2colores>Rojo y Naranja</2colores>
< Aficiones >Cine, Bailar, Nadar</ Aficiones >
<persona><nombre>Elsa</persona></nombre>
<color favorito>azul</color favorito>
```

En su lugar, sería correcto escribir:

```
<Ciudad>Pamplona</Ciudad>
<día>18</día>
<mes>6</mes>
<ciudad>Pamplona</ciudad>
<_rojo/>
<colores2>Rojo y Naranja</colores2>
<Aficiones >Cine, Bailar, Nadar</Aficiones >
<persona><nombre>Elsa</nombre></persona>
<color.favorito>azul</color.favorito>
<color-favorito>azul</color-favorito>
<color_favorito>azul</color_favorito>
```

5. XML es sensible a las mayúsculas.

Esto quiere decir que <nombre>, <Nombre> y <NOMBRE> son etiquetas diferentes. ¿Qué sentido tendría utilizar las mayúsculas en XML? En ocasiones el nombre de una etiqueta está formada por más de una palabra. Hay diferentes convenciones para facilitar la lectura, como separar por "**.**", "**-**", por "**_**" e incluso utilizar una mayúscula al principio de cada palabra.

Por ejemplo: <tipoSuscripcion>. Lo importante es seguir siempre el mismo mecanismo a lo largo de todo el código.

6. Atributos

Los atributos en XML siempre van a ir con un elemento. Por ejemplo en la etiqueta <nombre apodo="ale"> está el atributo apodo, que depende del elemento nombre.

Y siempre tienen esa estructura:

```
nombre-atributo="valor-atributo"
```

Los nombres de los atributos deben cumplir las mismas normas de sintaxis que los nombres de los elementos. Además, **todos los atributos de un elemento tienen que ser únicos**.

Se pueden utilizar las comillas dobles (") o las comillas simples(‘).

EJEMPLO Si el valor de un atributo se escribe entre comillas dobles ("), dicho valor no podrá contener dicho carácter. Por ejemplo, **no es correcto** escribir:

```
<dato caracter="comilla doble(")"/>
```

Para ello, hay que utilizar la referencia a entidad **"** como se muestra a continuación:

```
<dato caracter="comilla doble(&quot;)"/>
```

De igual modo ocurre con la comilla simple ('), siendo **incorrecto** escribir, por ejemplo:

```
<dato caracter='comilla simple(')'/>
```

Por lo que, en este caso, habría que usar **'** como se muestra seguidamente:

```
<dato caracter='comilla simple(&apos;)'/>
```

Por otro lado, los valores de atributos escritos entre comillas dobles (") sí pueden contener al carácter comilla simple (') y a la inversa. Por ejemplo, es correcto escribir:

```
<dato caracter="comilla simple(')"/>
<dato caracter='comilla doble(")'/>
```

En estos casos, no es obligatorio usar las referencias a entidades, pero sí recomendable.

Los espacios en blanco repetidos y los retornos de carro son eliminados. De modo que en la siguiente etiqueta:

```
<test miatrib="ciertos datos          van aquí          ">
```

el analizador sintáctico asignará al atributo "miatrib" el valor

"ciertos datos van aquí ".

7. Comentarios

Los comentarios tienen la siguiente estructura:

```
<!-- Comentario aquí -->
```

En un documento XML, no se pueden escribir comentarios dentro de las etiquetas. Por ejemplo, no es correcto escribir:

```
<mujer <!-- elemento vacío --> />
```

Por otro lado, hay que tener en cuenta que en los comentarios de un documento XML no está permitido usar dos guiones seguidos:

```
<!-- Dos guiones seguidos -- en un comentario da error -->
```

De forma que, no es posible anidar comentarios en un documento XML.

8. Declaraciones xml

Un archivo XML proporciona una etiqueta para crear declaraciones que identifiquen las características del fichero, y así evitar problemas relacionados con codificación de caracteres, versión de xml, etc. Esta sección se denomina prólogo.

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

- **version** puede ser 1.0 o 1.1. Actualmente sigue predominando 1.0.
- **encoding** incluye la codificación de caracteres. Habitualmente, **ISO-8859-1**, **UTF-8**
- o UTF-16.
- **standalone** puede ser "yes" (especifica que el documento no depende de otros ficheros) o "no" (indicando que el documento depende de otros ficheros, como una DTD, que más adelante veremos).

En un documento XML **no es obligatorio que aparezca la declaración XML**. Ahora bien, si se incluye, tiene que aparecer en la primera línea del documento, y el carácter "<" debe ser el primero de dicha línea, es decir, antes no pueden aparecer espacios en blanco.

En un documento XML, escribir la declaración XML es opcional. Pero, si se escribe, el atributo **version** es **obligatorio** indicarlo. Sin embargo, los atributos **encoding** y **standalone** son opcionales y, por defecto, sus valores son "UTF-8" y "no", respectivamente.

Por otra parte, cuando se escriba el atributo **encoding**, siempre deberá aparecer después de **version**. Y, respecto al atributo **standalone**, siempre que se escriba, deberá ser en último lugar.

9. Caracteres PCDATA ilegales

Hay caracteres reservados que no se pueden incluir en la PCDATA debido a que son usados en la sintaxis XML: los caracteres <, y &.

En vez de ellos, se deben usar los símbolos especiales siguientes:

```
&lt; equivale a <
&amp; equivale a &
```

Se pueden escribir referencias de caracteres Unicode con los símbolos &#, seguidos del valor decimal o hexadecimal del carácter Unicode que se quiera representar y, finalmente, añadiendo el carácter punto y coma “;”.

Ejemplo: el símbolo del Euro (€) puede escribirse € ó €

10. Secciones CDATA

Cuando hay fragmentos de texto llenos de símbolos "<", "&", etc, puede llegar a ser ilegible con símbolos especiales. Por ejemplo, la cadena "**6 es < 7 & 7 es > 6**", se escribiría así:

```
6 es &lt; 7 &amp; 7 &gt; 6
```

Para ellos podemos utilizar secciones **CDATA** (término heredado de SGML) del siguiente modo:

```
<![CDATA[6 es < 7 & 7 > 6]]>
```

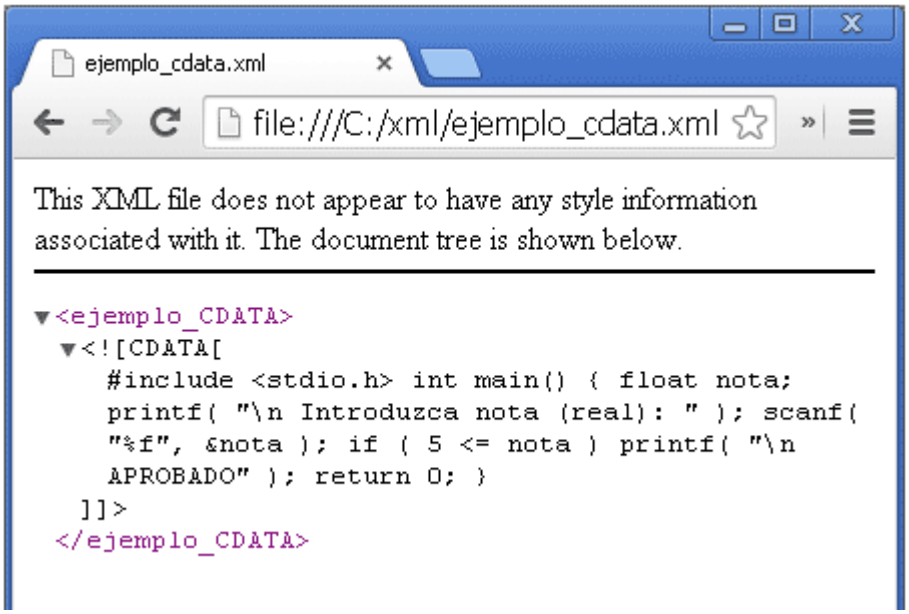
Un documento XML puede contener secciones CDATA (Character DATA) para escribir texto que no se desea que sea analizado. Por ejemplo, esto puede ser útil cuando se quiere escribir texto que contenga alguno de los **caracteres problemáticos: menor que "<" o ampersand "&"**.

En un documento XML, para incluir una sección CDATA, esta se escribe comenzando con la cadena de caracteres "**<![CDATA[**" y terminando con los caracteres "**]]>**".

EJEMPLO Una sección CDATA puede contener, por ejemplo, el código fuente de un programa escrito en lenguaje C:

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo_CDATA>
<![CDATA[
#include <stdio.h>
int main()
{
   float nota;
   printf( "\n   Introduzca nota (real): " );
   scanf( "%f", &nota );
   if ( 5 <= nota )
      printf( "\n   APROBADO" );
   return 0;
}
]]>
</ejemplo_CDATA>
```

En un navegador web se visualizará algo parecido a:



Dentro de una sección CDATA no se puede escribir la cadena "**]]>**". En consecuencia, no se pueden anidar secciones CDATA.