Apuntes de XML

{Abrirllave.com – Tutoriales de informática

Contenidos del tutorial de XML www.abrirllave.com/xml/

(Estos apuntes incluyen enlaces a los apartados y ejercicios resueltos del tutorial en la Web)

1. Qué es XML

XML (*eXtensible Markup Language*, Lenguaje de Marcado eXtensible) es un lenguaje desarrollado por **W3C** (*World Wide Web Consortium*) que está basado en **SGML** (*Standard Generalized Markup Language*, Lenguaje de Marcado Generalizado Estándar).

XML es un lenguaje utilizado para el almacenamiento e intercambio de datos estructurados entre distintas plataformas.

XML es un metalenguaje, es decir, puede ser empleado para definir otros lenguajes, llamados dialectos XML. Por ejemplo, algunos lenguajes basados en XML son:

- **GML** (*Geography Markup Language*, Lenguaje de Marcado Geográfico).
- **MathML** (*Mathematical Markup Language*, Lenguaje de Marcado Matemático).
- **RSS** (*Really Simple Syndication*, Sindicación Realmente Simple).
- **SVG** (*Scalable Vector Graphics*, Gráficos Vectoriales Escalables).
- **XHTML** (*eXtensible HyperText Markup Language*, Lenguaje de Marcado de Hipertexto eXtensible).
- etc.

2. Elementos

Los documentos XML están formados por texto plano (sin formato) y contienen marcas (etiquetas) definidas por el desarrollador. Dichas marcas, es recomendable que sean lo más descriptivas posible y, para escribirlas, se utilizan los caracteres *menor que* "**<**", *mayor que* "**>**" y *barra inclinada* "**/**".

EJEMPLO Si en un documento XML se quiere guardar el nombre **Elsa**, se puede escribir:

```
<nombre>Elsa</nombre>
```

La sintaxis utilizada en el ejemplo es la básica para escribir un elemento en XML:

```
<etiqueta>valor</etiqueta>
```

Obsérvese que, entre la etiqueta de inicio (**<nombre>**) y la etiqueta de fin (**</nombre>**) se ha escrito el dato (**valor**) que se quiere almacenar. En este caso **Elsa**.

Elementos vacíos

En un documento XML, **un elemento puede no contener ningún valor.** En tal caso hay que escribir:

```
<etiqueta></etiqueta>
```

Se puede expresar lo mismo escribiendo:

```
<etiqueta/>
```

EJEMPLO Para escribir el elemento "nombre" vacío, se puede escribir:

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

O también:

```
<nombre/>
```

Relaciones padre-hijo entre elementos

EJEMPLO Por otra parte, un elemento (*padre*) puede contener a otro u otros elementos (*hijos*):

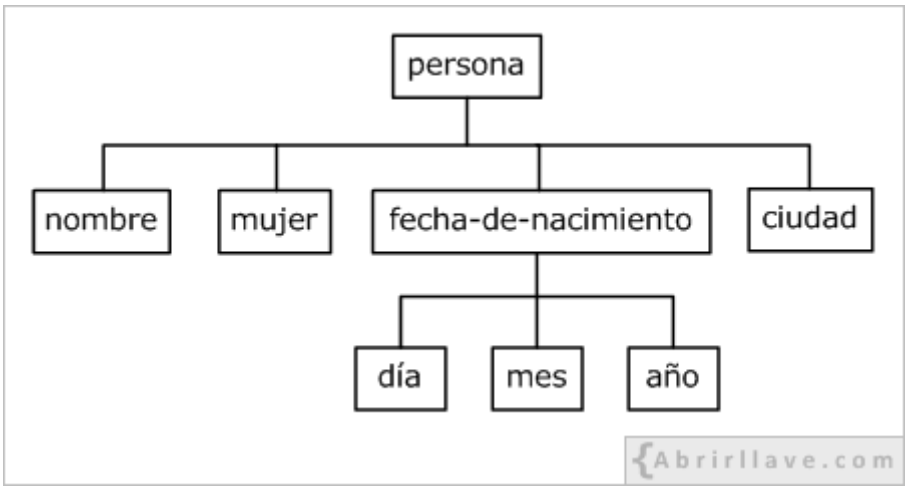
```
<persona>
   <nombre>Elsa</nombre>
   <mujer/>
   <fecha-de-nacimiento>
      <día>18</día>
      <mes>6</mes>
      <año>1996</año>
   </fecha-de-nacimiento>
   <ciudad>Pamplona</ciudad>
</persona>
```

En este ejemplo, el elemento "persona" contiene cuatro elementos (*hijos*): "nombre", "mujer", "fecha de nacimiento" y "ciudad". A su vez, el elemento "fecha de nacimiento" contiene otros tres elementos (*hijos*): "día", "mes" y "año".

Véase que, de todos los elementos que aparecen en este ejemplo, sólo el elemento "mujer" está vacío.

Elemento raíz de un documento XML

Todo documento XML tiene que tener un único elemento raíz (*padre*) del que desciendan todos los demás. En este caso, el elemento raíz es "persona". Gráficamente, la estructura de elementos de este documento se puede representar como se muestra a continuación:



De esta forma, **la estructura de cualquier documento XML se puede representar como un árbol invertido de elementos.** Se dice que los elementos son los que dan estructura semántica al documento.

Elementos con contenido mixto

EJEMPLO Un elemento puede contener contenido mixto, es decir, texto y otros elementos:

```
<persona>
   <nombre>Elsa</nombre> vive en <ciudad>Pamplona</ciudad>.
</persona>
```

En este ejemplo, el elemento "persona" contiene los elementos "nombre" y "ciudad", además de los textos **" vive en "** y **"."**.

3. Normas de sintaxis básicas

En un documento XML, **todos los nombres de los elementos son *case sensitive***, es decir, sensibles a letras minúsculas y mayúsculas, teniendo que cumplir las siguientes normas:

- Pueden contener letras minúsculas, letras mayúsculas, números, *puntos* "**.**", *guiones medios* "**-**" y *guiones bajos* "**_**".
- Asimismo, pueden contener el carácter *dos puntos* "**:**". No obstante, su uso se reserva para cuando se definan espacios de nombres.
- El primer carácter tiene que ser una letra o un *guion bajo* "**_**".

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:**

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

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>
```

Las letras no inglesas (á, Á, ñ, Ñ...) están permitidas. Sin embargo, es recomendable no utilizarlas para reducir posibles incompatibilidades con programas que puedan no reconocerlas.

En cuanto al carácter *guion medio* "**-**" y al *punto* "**.**", aunque también están permitidos para nombrar etiquetas, igualmente se aconseja evitar su uso; el *guion medio* porque podría confundirse con el *signo menos*, y el *punto* porque, por ejemplo al escribir **color.favorito**, podría interpretarse que **favorito** es una propiedad del objeto **color**.

4. Atributos

Los elementos de un documento XML pueden tener atributos definidos en la etiqueta de inicio. **Un atributo sirve para proporcionar información extra sobre el elemento que lo contiene.**

EJEMPLO Dados los siguientes datos de un producto:

- Código: **G45**
- Nombre: **Gorro de lana**
- Color: **negro**
- Precio: **12.56**

Su representación en un documento XML podría ser, por ejemplo:

```
<producto codigo="G45">
  <nombre color="negro" precio="12.56">Gorro de lana</nombre>
</producto>
```

En este ejemplo se han escrito tres atributos: **codigo**, **color** y **precio**. Obsérvese que, sus valores (**"G45"**, **"negro"** y **"12.56"**) se han escrito entre *comillas dobles* (**"**). No obstante, también pueden ir entre *comillas simples* (**'**).

Si, por ejemplo, el atributo **codigo** se quisiera representar como un elemento, se podría escribir:

```
<producto>
  <codigo>G45</codigo>
  <nombre color="negro" precio="12.56">Gorro de lana</nombre>
</producto>
```

Como se puede apreciar, ahora el valor del código no se ha escrito entre comillas dobles.

Normas de sintaxis

EJEMPLO 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.** Por ejemplo, es incorrecto escribir:

```
<datos x="3" x="4" y="5"/>
```

Sin embargo, sí es correcto escribir:

```
<datos x="3" X="4" y="5"/>
```

Los atributos contenidos en un elemento, como en este caso **`x`**, **`X`** e **`y`**, deben separarse con espacios en blanco, no siendo significativo su orden.

5. Declaración XML

La declaración XML que se puede escribir al principio de un documento XML, empieza con los caracteres "**<?**" y termina con "**?>**" al igual que las instrucciones de procesamiento. Sin embargo, **la declaración XML no es una instrucción de procesamiento (o proceso).**

Atributos **`version`** y **`encoding`**

EJEMPLO Un documento XML podría contener la siguiente declaración XML:

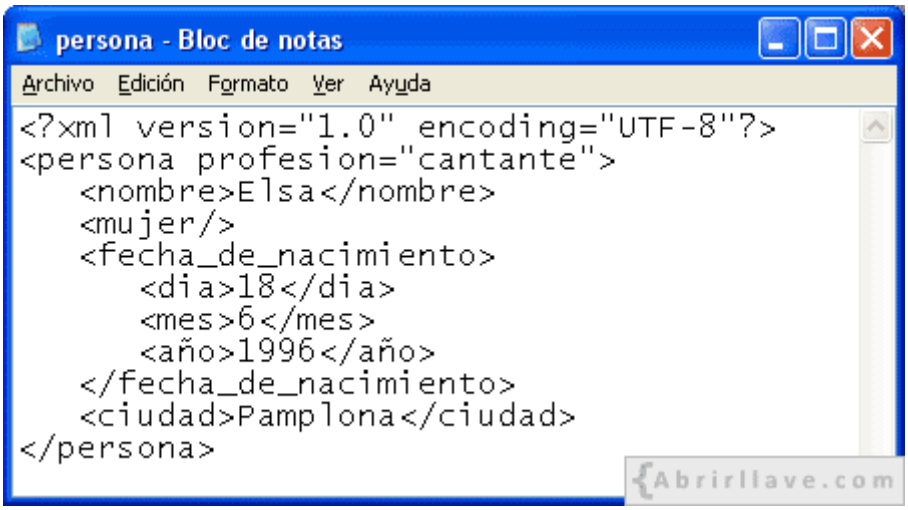
```
<?xml version="1.0" encoding="UTF-8"?>
```

En esta declaración XML, se está indicando que **1.0** es la versión de XML utilizada en el documento y **UTF-8** (*8-bit Unicode Transformation Format*, Formato de Transformación Unicode de 8 bits) es la codificación de caracteres empleada.

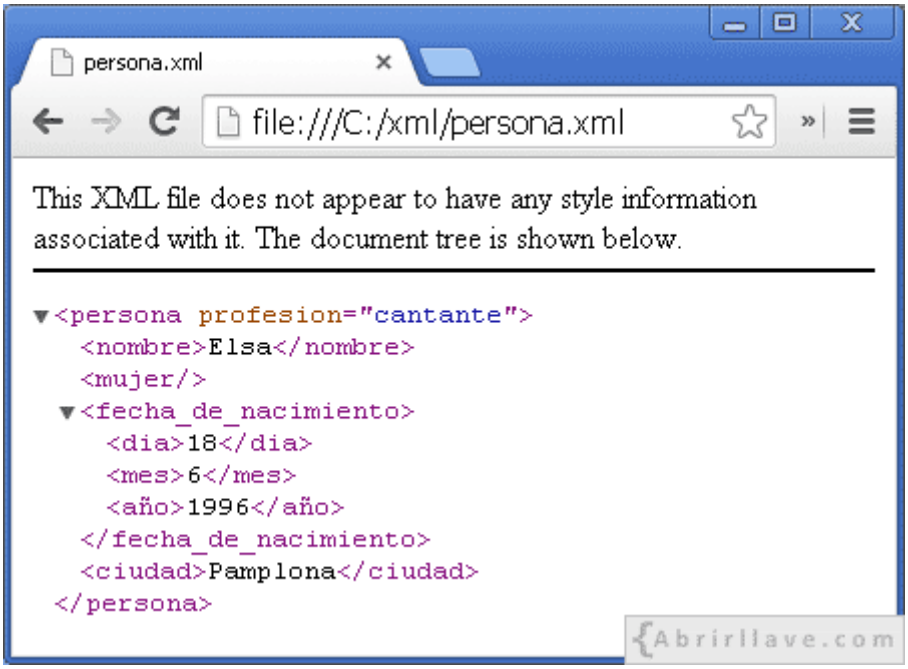
En un documento XML no es obligatorio que aparezca la declaración XML. Ahora bien, si se incluye, tiene que aparecer en la primera línea del documento, y el carácter "**<**" debe ser el primero de dicha línea, es decir, antes no pueden aparecer espacios en blanco.

Cómo crear un documento XML

EJEMPLO Si, por ejemplo, en el *Bloc de notas* de *Microsoft Windows* escribimos y guardamos (codificado en UTF-8) un archivo llamado ***"persona.xml"*** como se muestra en la siguiente imagen:



Al visualizar dicho archivo en un navegador web, como por ejemplo *Google Chrome*, se podrá ver algo parecido a:



Como se puede ver, a la izquierda de los elementos que tienen hijos, en este caso **`persona`** y **`fecha_de_nacimiento`**, aparece un pequeño triángulo. Por otra parte, el elemento `persona` es el único que tiene un atributo.

Atributo **`standalone`**

EJEMPLO En una declaración XML, además de los atributos **`version`** y **`encoding`**, también se puede escribir el atributo **`standalone`**, que puede tomar dos valores (**`"yes"`** o **`"no"`**):

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

Al escribir **`standalone="yes"`** se está indicando que el documento es independiente de otros, como por ejemplo de una **DTD** (*Document Type Definition*, Definición de Tipo de Documento) externa. En caso contrario, significará que el documento no es independiente.

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.

Ejercicios resueltos
• Ciudades de países • Hechos históricos

6. Instrucciones de procesamiento

En un documento XML, **una instrucción de procesamiento (*processing instruction*) sirve para indicar cierta información al programa que procese dicho documento.** Las instrucciones de proceso se escriben empezando con la pareja de caracteres "**<?**" y finalizando con "**?>**".

Cómo asociar un archivo CSS a un documento XML

EJEMPLO En un documento XML podría escribirse, por ejemplo, la siguiente instrucción de procesamiento:

```
<?xml-stylesheet type="text/css" href="estilo-animales.css"?>
```

Esta instrucción sirve para asociar el archivo **CSS** (*Cascading Style Sheets*, Hojas de Estilo en Cascada) ***"estilo-animales.css"*** al documento XML. Dicho archivo podría contener, por ejemplo, el siguiente código:

```
nombre{color:blue;font-size:40px}
patas{color:red;font-size:22px}
```

De forma que, dado por ejemplo el archivo ***"animales.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="estilo-animales.css"?>
<animales>
   <animal>
      <nombre>perro</nombre>
      <patas>4</patas>
   </animal>
   <animal>
      <nombre>pato</nombre>
      <patas>2</patas>
   </animal>
   <animal>
      <nombre>ballena</nombre>
      <patas>0</patas>
   </animal>
</animales>
```

En un navegador web se verá algo parecido a:

En un documento XML, no es obligatorio que aparezcan instrucciones de procesamiento.

Ejercicios resueltos
• Artículos • Fechas de un año • Mezclas de colores

7. Referencias a entidades

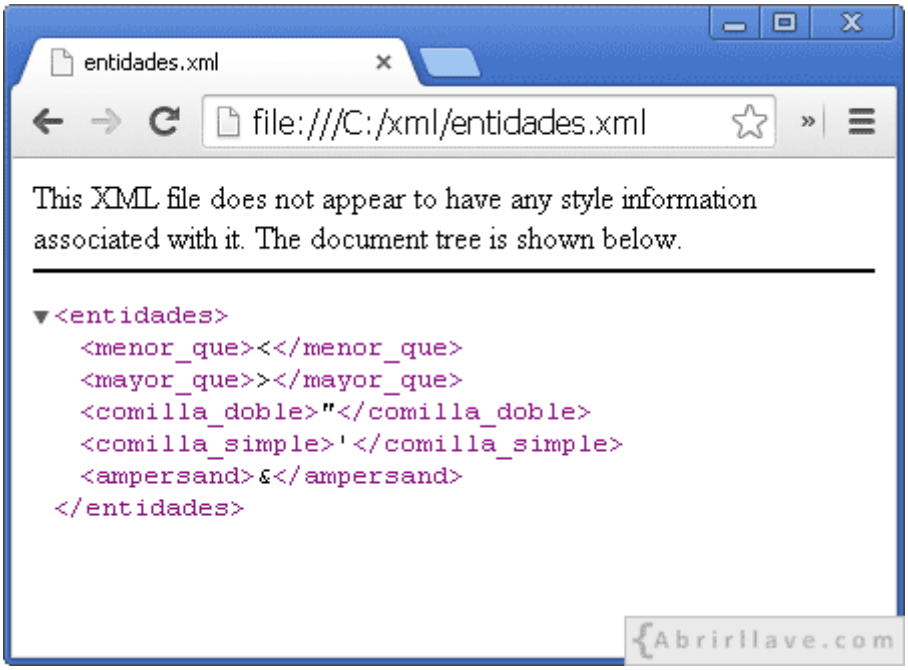
En XML existen algunos caracteres que son especiales por su significado y, para escribirlos en un documento XML, se pueden utilizar las referencias a entidades mostradas en la siguiente tabla:

Referencias a entidades en XML		
Carácter	***Entidad***	***Referencia a entidad***
< (*menor que*)	**lt** (*less than*)	**<**
> (*mayor que*)	**gt** (*greater than*)	**>**
" (*comilla doble*)	**quot** (*quotation mark*)	**"**
' (*comilla simple*)	**apos** (*apostrophe*)	**'**
& (*ampersand*)	**amp** (*ampersand*)	**&**

EJEMPLO Dado el archivo ***"entidades.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<entidades>
   <menor_que>&lt;</menor_que>
   <mayor_que>&gt;</mayor_que>
   <comilla_doble>&quot;</comilla_doble>
   <comilla_simple>&apos;</comilla_simple>
   <ampersand>&amp;</ampersand>
</entidades>
```

Al abrirlo en *Google Chrome* se podrá visualizar:



En el navegador web, se puede ver que donde se han escrito las referencias a entidades en el documento XML (por ejemplo **<**), se muestran los caracteres correspondientes (por ejemplo **<**).

Caracteres problemáticos en XML: menor que (<) y ampersand (&)

EJEMPLO En un documento XML, **el carácter "<" es problemático porque indica el comienzo de una etiqueta.** Por tanto, en vez de escribir, por ejemplo:

```
<condicion>a<b</condicion>
```

Habría que utilizar la referencia a entidad **<** escribiendo:

```
<condicion>a&lt;b</condicion>
```

EJEMPLO El carácter "**>**" sí puede utilizarse en el texto contenido en un elemento, no siendo incorrecto escribir, por ejemplo:

```
<condicion>a>b</condicion>
```

Ahora bien, se recomienda hacer uso de su referencia a entidad (**>**).

EJEMPLO En un documento XML, **el carácter ampersand "&" también es problemático, ya que se utiliza para indicar el comienzo de una referencia a entidad.** Por ejemplo, no es correcto escribir:

```
<condicion>a=1 && b=2</condicion>
```

En su lugar se debe escribir lo siguiente:

```
<condicion>a=1 &amp;&amp; b=2</condicion>
```

Uso de la comilla doble (") y de la comilla simple (') en atributos

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.

8. Referencias de caracteres

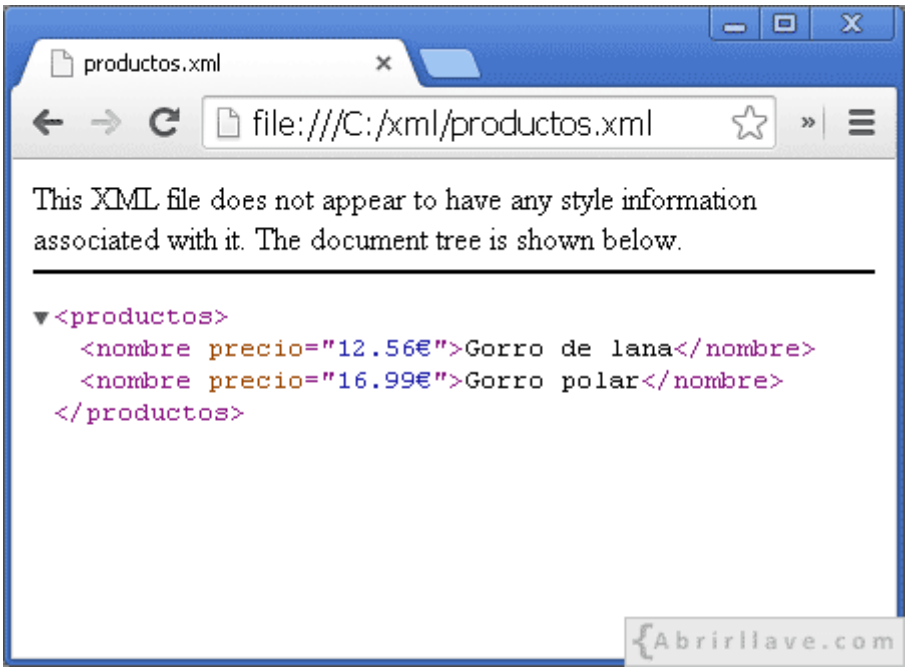
En un documento XML se pueden escribir referencias de caracteres Unicode con los símbolos **&#**, seguidos del valor decimal o hexadecimal del carácter Unicode que se quiera representar y, finalmente, añadiendo el carácter *punto y coma* "**;**".

Representación del carácter Euro (€) en XML

EJEMPLO Dado el documento XML ***"productos.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<productos>
   <nombre precio="12.56&#8364;">Gorro de lana</nombre>
   <nombre precio="16.99&#x20AC;">Gorro polar</nombre>
</productos>
```

Al visualizarlo en un navegador web, se podrá ver lo siguiente:



Obsérvese que, en este caso, para representar al símbolo del Euro (**€**), la primera vez se ha utilizado su valor decimal (**€**) en Unicode y, la segunda vez, su valor hexadecimal (**€**).

9. Comentarios

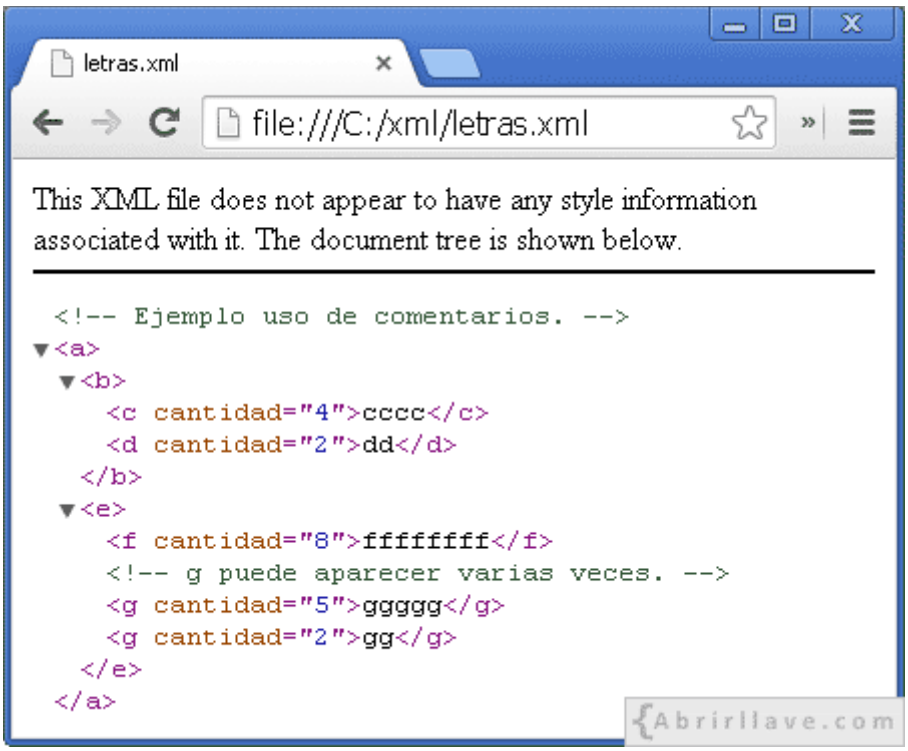
Para escribir comentarios en un documento XML, estos deben escribirse entre los caracteres "**<!--**" y "**-->**" . Por ejemplo:

```
<!-- Esto es un comentario escrito en un documento XML -->
```

EJEMPLO Dado el archivo XML ***"letras.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Ejemplo uso de comentarios.-->
<a>
   <b>
      <c cantidad="4">cccc</c>
      <d cantidad="2">dd</d>
   </b>
   <e>
      <f cantidad="8">ffffffff</f>
      <!--g puede aparecer varias veces.-->
      <g cantidad="5">ggggg</g>
      <g cantidad="2">gg</g>
   </e>
</a>
```

En un navegador web se verá:



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

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

EJEMPLO 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.**

10. Secciones CDATA

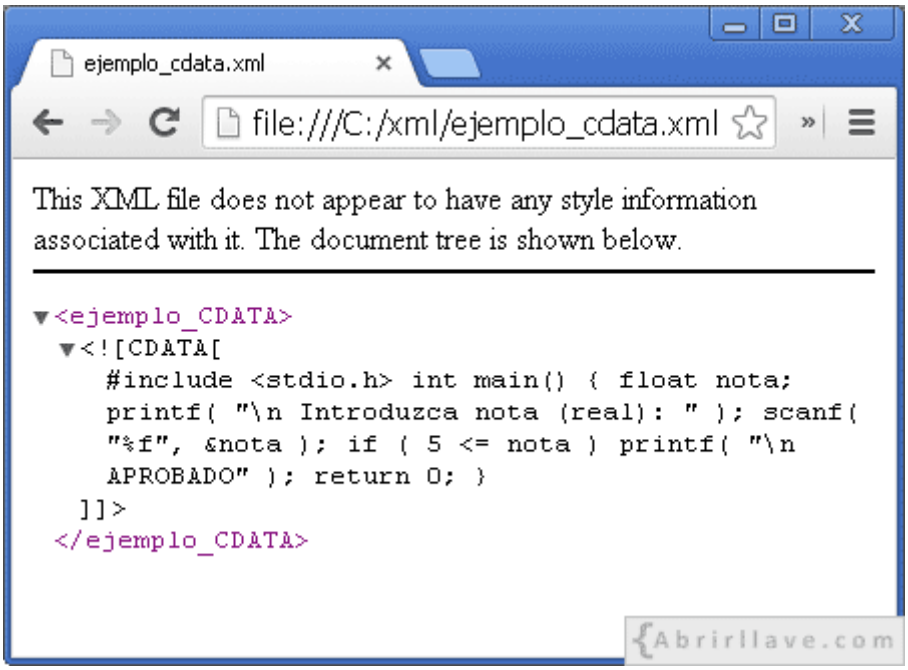
Un documento XML puede contener secciones **CDATA** (*Character DATA*) para escribir texto que no se desea que sea analizado. Por ejemplo, esto puede ser útil cuando se quiere escribir texto que contenga alguno de los caracteres problemáticos: *menor que* "**<**" o *ampersand* "**&**".

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.**

Por otra parte, no está permitido escribir espacios en blanco o saltos de línea en las cadenas de inicio "**<![CDATA[**" o fin "**]]>**" de una sección CDATA.

11. Espacios de nombres

Varios documentos XML se pueden combinar entre sí, pudiendo en estos casos coincidir el nombre de algunos elementos.

EJEMPLO Dos documentos XML podrían contener un elemento llamado "carta", pero con significados distintos.

```
<carta>
   <palo>Corazones</palo>
   <numero>7</numero>
</carta>
```

```
<carta>
   <carnes>
      <filete_de_ternera precio="12.95"/>
      <solomillo_a_la_pimienta precio="13.60"/>
   </carnes>
   <pescados>
      <lenguado_al_horno precio="16.20"/>
      <merluza_en_salsa_verde precio="15.85"/>
   </pescados>
</carta>
```

Uso de espacios de nombres

De forma que, si se incluyen ambos elementos **<carta>** en un documento XML, se origina un conflicto de nombres. Para resolverlo, se pueden utilizar espacios de nombres (*XML Namespaces*). Por ejemplo, escribiendo:

```
<?xml version="1.0" encoding="UTF-8"?>
<e1:ejemplo xmlns:e1="http://www.abrirllave.com/ejemplo1"
   xmlns:e2="http://www.abrirllave.com/ejemplo2">

   <e1:carta>
      <e1:palo>Corazones</e1:palo>
      <e1:numero>7</e1:numero>
   </e1:carta>

   <e2:carta>
      <e2:carnes>
         <e2:filete_de_ternera precio="12.95"/>
         <e2:solomillo_a_la_pimienta precio="13.60"/>
      </e2:carnes>
      <e2:pescados>
         <e2:lenguado_al_horno precio="16.20"/>
         <e2:merluza_en_salsa_verde precio="15.85"/>
       </e2:pescados>
   </e2:carta>

</e1:ejemplo>
```

Sintaxis para definir un espacio de nombres

Para definir un espacio de nombres se utiliza la siguiente sintaxis:

```
xmlns:prefijo="URI"
```

En el ejemplo, obsérvese que, **xmlns** es un atributo que se ha utilizado en la etiqueta de inicio del elemento **<ejemplo>** y, en este caso, se han definido dos espacios de nombres que hacen referencia a los siguientes **URI** (*Uniform Resource Identifier*, Identificador Uniforme de Recurso):

- **http://www.abrirllave.com/ejemplo1**
- **http://www.abrirllave.com/ejemplo2**

Los prefijos definidos son **e1** y **e2**, respectivamente. Véase que, se han añadido dichos prefijos a las etiquetas que aparecen en el documento: **<e1:carta>**, **<e2:carta>**, **<e1:palo>**, etc.

Los URI especificados en un documento XML no tienen porqué contener nada, su función es ser únicos. No obstante, en un URI se puede mostrar información si se considera oportuno. Véase, por ejemplo:

- http://www.w3.org/1999/xhtml/
- http://www.w3.org/1999/XSL/Transform
- http://www.w3.org/2000/svg

Definición de espacios de nombres en elementos distintos al raíz

EJEMPLO En un documento XML, los espacios de nombres pueden definirse en el elemento raíz –como acabamos de ver– o, directamente, en los elementos que los vayan a utilizar. Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<e1:ejemplo xmlns:e1="http://www.abrirllave.com/ejemplo1">

   <e1:carta>
      <e1:palo>Corazones</e1:palo>
      <e1:numero>7</e1:numero>
   </e1:carta>

   <e2:carta xmlns:e2="http://www.abrirllave.com/ejemplo2">
      <e2:carnes>
         <e2:filete_de_ternera precio="12.95"/>
         <e2:solomillo_a_la_pimienta precio="13.60"/>
      </e2:carnes>
      <e2:pescados>
         <e2:lenguado_al_horno precio="16.20"/>
         <e2:merluza_en_salsa_verde precio="15.85"/>
      </e2:pescados>
   </e2:carta>

</e1:ejemplo>
```

En un documento XML es posible definir todos los espacios de nombres que se necesiten, pudiéndose mezclar –si fuese necesario– los elementos de dichos espacios de nombres.

Definición de un espacio de nombres por defecto

EJEMPLO Por otra parte, se puede definir un espacio de nombres por defecto mediante la siguiente sintaxis:

```
xmlns="URI"
```

De esta forma, tanto el elemento donde se ha definido el espacio de nombres, como todos sus sucesores (hijos, hijos de hijos, etc.), pertenecerán a dicho espacio de nombres. Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.abrirllave.com/ejemplo1">

  <carta>
    <palo>Corazones</palo>
    <numero>7</numero>
  </carta>

</ejemplo>
```

EJEMPLO En el siguiente ejemplo, inicialmente se define un espacio de nombres por defecto para el elemento **<ejemplo>** y los contenidos en él. Ahora bien, posteriormente, se define un segundo espacio de nombres, que por defecto afecta al segundo elemento **<carta>** que aparece en el documento y a sus sucesores:

- **<carnes>**
- **<pescados>**
- **<filete_de_ternera>**
- …

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.abrirllave.com/ejemplo1">

   <carta>
      <palo>Corazones</palo>
      <numero>7</numero>
   </carta>

   <carta xmlns="http://www.abrirllave.com/ejemplo2">
      <carnes>
         <filete_de_ternera precio="12.95"/>
         <solomillo_a_la_pimienta precio="13.60"/>
      </carnes>
      <pescados>
         <lenguado_al_horno precio="16.20"/>
         <merluza_en_salsa_verde precio="15.85"/>
      </pescados>
   </carta>

</ejemplo>
```

Cómo indicar que un elemento no pertenece a ningún espacio de nombres

EJEMPLO En un documento XML, para indicar que determinados elementos –o todos– no pertenecen a ningún espacio de nombres, se escribe el atributo **xmlns** vacío, es decir, **xmlns=""**.

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.abrirllave.com/ejemplo1">

   <carta>
      <palo>Corazones</palo>
      <numero>7</numero>
   </carta>

   <carta xmlns="http://www.abrirllave.com/ejemplo2">
      <carnes>
         <filete_de_ternera precio="12.95"/>
         <solomillo_a_la_pimienta precio="13.60"/>
      </carnes>
      <pescados xmlns="">
         <lenguado_al_horno precio="16.20"/>
         <merluza_en_salsa_verde precio="15.85"/>
      </pescados>
   </carta>

</ejemplo>
```

En este caso, el elemento **<pescados>** y sus hijos, no pertenecen a ningún espacio de nombres.

12. Espacios en blanco

En un documento XML, los espacios en blanco, las tabulaciones y los retornos de carro pueden ser tratados de un modo especial.

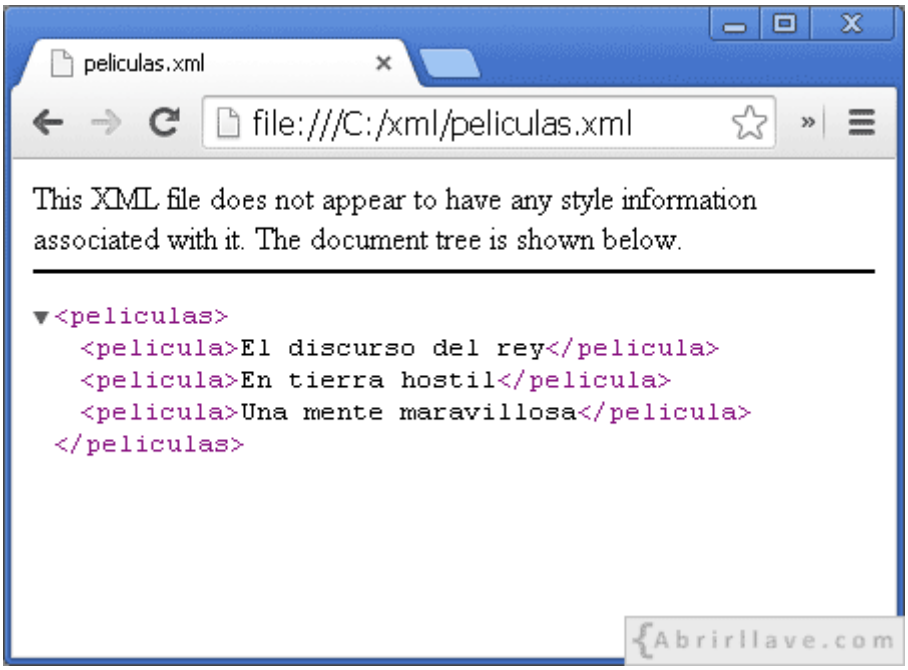
Espacios en blanco en el contenido (texto) de un elemento

EJEMPLO Dado el archivo XML ***"peliculas.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<peliculas>
   <pelicula>El discurso del rey</pelicula>
   <pelicula>En   tierra          hostil</pelicula>
   <pelicula>Una
      mente

maravillosa</pelicula>
</peliculas>
```

Al visualizar dicho archivo en *Google Chrome*, se verá algo parecido a:



Esto es debido a que, las tabulaciones, los retornos de carro y varios espacios en blanco contenidos en el texto de los elementos del documento, han sido representados como un único espacio en blanco.

Espacios en blanco en atributos

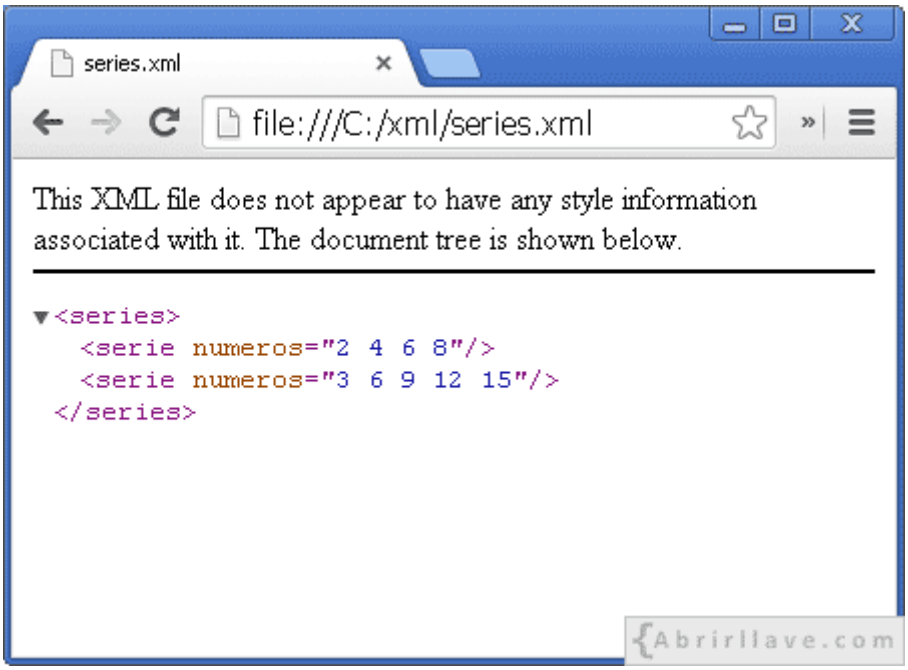
EJEMPLO De igual modo ocurre con los valores de los atributos. Por ejemplo, dado el archivo ***"series.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<series>
   <serie numeros="2 4 6 8"/>
   <serie numeros="3
6



   9
  12 15"/>
</series>
```

En un navegador web se podrá visualizar:



Espacios en blanco entre elementos

EJEMPLO Obsérvese que, los siguientes documentos XML contienen la misma información, pero, escrita de distinta forma:

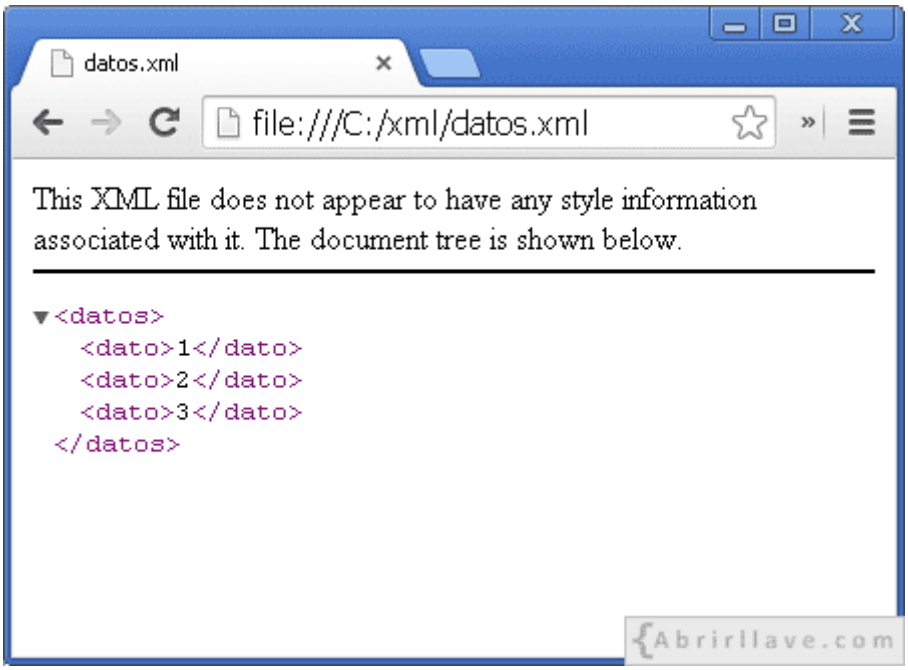
```
<?xml version="1.0" encoding="UTF-8"?>
<datos>
   <dato>1</dato>
   <dato>2</dato>
   <dato>3</dato>
</datos>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<datos><dato>1</dato><dato>2</dato><dato>3</dato></datos>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<datos><dato>1</dato>  <dato>2</dato>


<dato>3</dato></datos>
```

En todos los casos, en *Google Chrome* veremos:



Las aplicaciones que hacen uso de documentos XML suelen hacer este tratamiento de las tabulaciones, retornos de carro y espacios en blanco.

Uso del atributo **`xml:space`**

EJEMPLO Por otra parte, los elementos de un documento XML pueden contener el atributo predefinido **`xml:space`** con el valor **`"preserve"`** para indicar que los espacios que aparecen en el contenido (texto) de dicho elemento, y sus sucesores, deben ser preservados:

```
<clasificacion xml:space="preserve">
1       Fernando Alonso       1:55.341
2       Lewis Hamilton        1:55.729
3       Sebastian Vettel      1:56.122
</clasificacion>
```

Los únicos valores que admite el atributo **`xml:space`** son **`"preserve"`** y **`"default"`**, siendo este último su valor por defecto cuando no se escribe dicho atributo.

El valor **`"default"`** indica que la aplicación que haga uso del documento XML es la encargada de decidir cómo tratar los espacios en blanco. Ahora bien, aún indicando el valor **`"preserve"`**, hay que tener en cuenta que no todos los programas que hacen uso de documentos XML reconocen este atributo.

13. Documentos XML bien formados y válidos

Se dice que **un documento XML está bien formado (*well-formed document*) cuando no tiene errores de sintaxis**. Esto incluye los siguientes aspectos:

- Los nombres de los elementos y sus atributos deben estar escritos correctamente.
- Los valores de los atributos deben estar escritos entre comillas dobles o simples.
- Los atributos de un elemento deben separarse con espacios en blanco.
- Se tienen que utilizar referencias a entidades donde sea necesario.
- Tiene que existir un único elemento raíz.
- Todo elemento debe tener un elemento padre, excepto el elemento raíz.
- Todos los elementos deben tener una etiqueta de apertura y otra de cierre.
- Las etiquetas deben estar correctamente anidadas.
- Las instrucciones de proceso deben estar escritas de forma correcta.
- La declaración XML debe estar en la primera línea escrita correctamente.
- Las secciones CDATA y los comentarios deben estar correctamente escritos.

Por otro lado, se dice que **un documento XML es válido (*valid*) cuando, además de no tener errores de sintaxis, no incumple ninguna de las normas establecidas en su estructura**. Dicha estructura se puede definir utilizando distintos métodos, tales como:

- **DTD** (*Document Type Definition*, Definición de Tipo de Documento).
- **XML Schema**.
- **RELAX NG** (REgular LAnguage for XML Next Generation).

Ejercicios resueltos
• Lista de marcadores de páginas web • Equipos de fútbol

14. Recursos de XML

- ***Tutorial de XML***
  http://www.abrirllave.com/xml/
- ***Chuleta de XML***
  http://www.abrirllave.com/xml/chuleta-de-xml.php
- ***Ejercicios resueltos de XML***
  http://www.abrirllave.com/xml/ejercicios-resueltos.php
- ***Test de autoevaluación de XML***
  http://www.abrirllave.com/xml/test-de-autoevaluacion.php