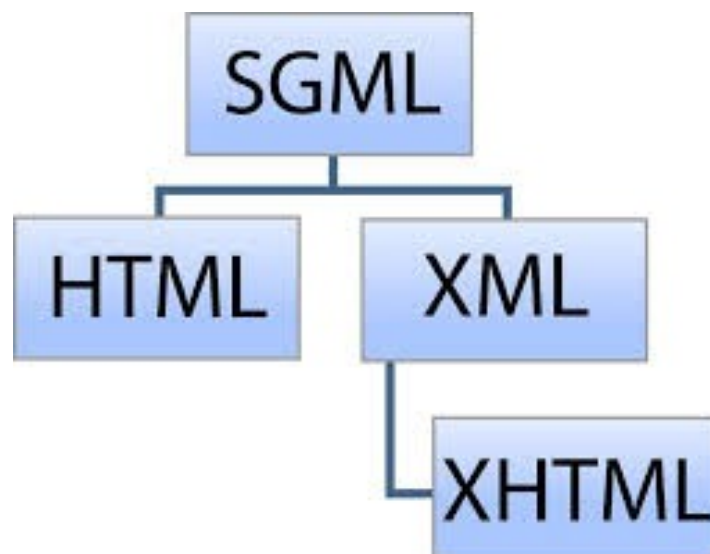


XML



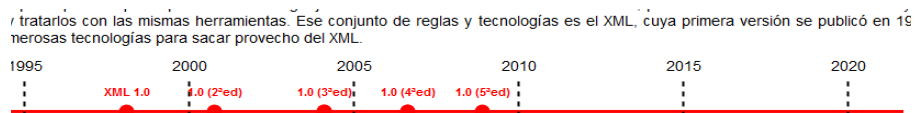
1.-Qué es el XML.....	4
Ejemplo de documento XML.....	5
1.1.-Conceptos y vocabulario.....	5
1.2.-Documentos bien formados.....	11
1.3.-Documentos válidos.....	11
1.4.-Otras recomendaciones XML.....	12
2.-DTD: Definición de Tipo de Documento.....	14
2.1.-Qué es una DTD.....	14
2.2.-Referencia a una DTD en un documento XML.....	14
2.3.-Declaraciones.....	15
Declaración de entidades.....	15
Declaración de notaciones.....	16
Declaración de elementos.....	16
Declaración de atributos.....	22
3.- Hojas de estilo CSS en XML.....	31
3.1.-CSS en XML.....	31
3.2.-Documentos XML sin hojas de estilo.....	31
3.3.-Documentos XML con hojas de estilo.....	32
3.4.-Ejemplos de hojas de estilo para XML.....	35
La propiedad display.....	35
Mostrar elementos como listas ordenadas.....	37
Los atributos class e id.....	39
4.-XSL: Lenguaje de Hojas de Estilo Extensible.....	41
4.1.-XPath: XML Path language.....	41
4.1.1.-Qué es XPath.....	41
4.1.2.-Árbol del documento.....	41
Tipos de nodos.....	44
4.1.3.-Sintaxis de la expresiones XPath.....	45
4.1.4.-Sintaxis abreviada.....	46
Eje (en inglés, axis).....	46
Predicado (en inglés, predicate).....	48
Selección de nodos (en inglés, node test).....	51
Pasos de búsqueda consecutivos.....	53
Expresiones anidadas.....	55
5.- XSLT: Transformaciones XSL.....	58
5.1.-El lenguaje de programación XSLT.....	58
5.2.-Hojas de estilo XSLT.....	58
5.3.-Enlazar documentos XML con hojas de estilo XSLT.....	60
5.4.-Abrir documentos XML con hojas de estilo XSLT en el navegador.....	60
5.5.-Ejemplos de plantillas XSLT.....	61
Plantillas vacías o no existentes.....	61
La instrucción <xsl:value-of>.....	62
Generar texto adicional.....	64
Aplicar reglas a subnodos: la instrucción <xsl:apply-templates>.....	66
Saltos de línea y espacios en blanco: las instrucciones <xsl:text> y <xsl:strip-space>.....	67

La instrucción <xsl:attribute>.....	69
6.-XML Copy Editor.....	72
6.1.-Conseguir XML Copy Editor.....	72
6.2.-Interface y configuración de XML Copy Editor.....	72
6.3.-Juego de caracteres.....	75
6.4.-Comprobar que un documento está bien formado.....	76
6.5.-Comprobar que un documento es válido.....	78
6.6.-Enlazar una hoja de estilo CSS.....	80
6.7.-Evaluar una expresión XPath.....	84
6.8.-Aplicar una transformación XSLT.....	88

1.-Qué es el XML

El XML (eXtensible Markup Language = Lenguaje de Marcas Extensible) no es un lenguaje de marcas, sino un metalenguaje, es decir, el XML define las reglas generales que debe cumplir un lenguaje de marcas y la manera de definir un lenguaje de marcas.

El XML fue creado por el [W3C](#) a finales de los 90. El W3C se creó en 1994 para tutelar el crecimiento y organización de la web. Su primer trabajo fue normalizar el HTML, el lenguaje de marcas con el que se escriben las páginas web. Al crecer el uso de la web, crecieron las presiones para ampliar el HTML, pero el W3C decidió que la solución no era ampliar el HTML, sino **crear unas reglas para que cualquiera pudiera crear lenguajes de marcas adecuados a sus necesidades**, pero manteniendo unas estructuras y sintaxis comunes que permitieran compatibilizarlos y tratarlos con las mismas herramientas. Ese conjunto de reglas y tecnologías es el XML, cuya primera versión se publicó en 1998. Posteriormente, el W3C ha ido desarrollando numerosas tecnologías para sacar provecho del XML.



Lógicamente, el HTML de los años 90 no cumplía las normas del XML ya que el HTML era anterior al XML. El creador del HTML, [Tim Berners-Lee](#), se había basado en el SGML, otro conjunto de reglas para la creación de lenguajes de marcas creado en los años 80 y más complejo que el XML. Una vez creado el XML, el W3C aprobó en el año 2000 el XHTML, una versión del HTML que sí que cumplía las reglas del XML. Durante los primeros años del siglo XXI, el W3C intentó sin éxito que se dejara de utilizar el HTML y en su lugar se utilizara el XHTML.

En 2007 el W3C reconoció su fracaso y retomó el desarrollo del HTML (aunque siguió incluyendo en él una versión XHTML, basada en XML). En octubre de 2014 se publicó la recomendación HTML 5, que incluye las dos variantes: HTML 5, no basada en XML, y XHTML 5, basada en XML (una tercera variante, llamada HTML políglota ha sido abandonada).

En general, se puede decir que el XML ha tenido un gran éxito y se ha convertido en una herramienta básica para el intercambio y almacenamiento de información en muchos sectores de la informática.

Pero, curiosamente, uno de los sectores en los que el XML no se ha impuesto es en el entorno web, probablemente porque el XML, con su exigencia de ausencia de errores en los documentos, es demasiado estricto para una web que en gran parte todavía se elabora manualmente.

Ejemplo de documento XML

Un documento XML es similar a una página web, salvo que los nombres de las etiquetas y atributos no son los del HTML.

```
<?xml version="1.0" encoding="UTF-8"?>
<países>
  <pais>
    <nombre>España</nombre>
    <capital>Madrid</capital>
  </pais>
  <pais>
    <nombre>Francia</nombre>
    <capital>Paris</capital>
  </pais>
</países>
```

Por supuesto, la misma información se puede guardar con otra estructura:

```
<?xml version="1.0" encoding="UTF-8"?>
<países>
  <pais nombre="España" capital="Madrid" />
  <pais nombre="Francia" capital="París" />
</países>
```

```
<HTML>
  <HEAD>
    <TITLE>Name</TITLE>
  </HEAD>
  <BODY>
    <P>John Doe</P>
  </BODY>
</HTML>
```

Podemos crear un archivo XML como este:

```
<name>
  <first>John</first>
  <last>Doe</last>
</name>
```

1.1.-Conceptos y vocabulario

Documento XML

Un documento XML es un documento de **texto plano** (sin formato).

Procesador XML (*XML processor*) y aplicación (*application*)

Cuando una aplicación necesita leer un documento XML, la aplicación recurre a un procesador XML. El procesador XML (o **analizador XML**, en inglés *XML parser*) es el que lee el documento, analiza el contenido y le pasa la información en un formato estructurado a la aplicación. La recomendación XML especifica lo que debe hacer el procesador, pero no entra en lo que hace después la aplicación con esa información.

Caracteres (*characters*)

Los documentos XML pueden estar codificados en distintos juegos de caracteres (**UTF-8**, **ISO-8859-1**, etc).

Marcas (*mark-up*) y contenido (*content*)

El texto que contiene un documento XML se divide en marcas y contenido. Las marcas pueden ser de dos tipos: **etiquetas o referencias a entidades**. Todo lo que no son marcas es contenido.

Etiquetas (*tags*)

Una etiqueta es un identificador que empieza por el carácter **"<"** y termina por **">"**. Existen tres tipos de etiquetas:

- las etiquetas de apertura (***start-tag***), que empiezan por el carácter < y terminan por >. Por ejemplo:

```
<ciudad>
```

- las etiquetas de cierre (***end-tag***), que empiezan por los caracteres </ y terminan por >. Por ejemplo:

```
</ciudad>
```

- las etiquetas vacías (***empty tag***), que empiezan con el carácter < y terminan por >. Por ejemplo: (Se autocierra sola, no tienen contenido)

```
<linea />
```

Elementos (*elements*)

Un elemento es un componente lógico de un documento que o bien comienza por una **etiqueta de apertura y termina por la etiqueta de cierre** correspondiente o que consiste en una única **etiqueta vacía**. El contenido de un elemento es todo lo que se encuentra entre las etiquetas de apertura y cierre, incluso si estos son también elementos en cuyo caso se llaman elementos hijos.

Atributos (*attributes*)

Un atributo es un componente de las etiquetas que consiste en una pareja nombre (*name*) / valor (*value*). Se puede encontrar en las etiquetas de apertura o en las etiquetas vacías, pero no en las de cierre. En una etiqueta no puede haber dos atributos con el mismo nombre. La sintaxis es siempre nombreAtributo="valorAtributo". Por ejemplo:

```
<profesor nombre="Juan" apellidos="Pérez Ramírez" />
```

Comentarios (*comments*)

Un comentario es una etiqueta que comienza por <!-- y acaba por -->. Los comentarios no pueden estar dentro de otras marcas y no pueden contener los caracteres "--". Dentro de un comentario las entidades de carácter no se reconocen, es decir, sólo se pueden utilizar los caracteres del juego de caracteres del documento. Por ejemplo:

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

Declaración XML (*XML declaration*)

La declaración XML es una etiqueta que comienza por <?xml y termina por ?> y que proporciona información sobre el propio documento XML. Aunque no es obligatoria es conveniente que aparezca, y debe aparecer siempre al principio del documento. No es una instrucción de procesamiento, pero tiene la misma sintaxis (empieza por <? y acaba por ?>).

La declaración xml indica el juego de caracteres del documento. El juego de caracteres que se utiliza en este curso es UTF-8 (normalmente) o ISO-8859-1:

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

En XML, el nombre del juego de caracteres se debe escribir en mayúsculas (UTF-8 en vez de utf-8).

La declaración xml también indica la versión XML utilizada. Aunque existe la versión XML 1.1, la versión más común sigue siendo la versión XML 1.0.

Se pueden utilizar otros juegos de caracteres, como ISO-8859-1 (Europeo occidental):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Es importante que el juego de caracteres que aparece en la declaración sea el juego de caracteres en que realmente está guardado el documento, porque si no el procesador XML puede tener problemas leyendo el documento.

*XML usa el estándar de representación **ISO 10646**. Se trata de un estándar que engloba a todos los anteriores (como, por ejemplo, UNICODE) y que gracias al uso de 31 bits para la codificación de cada carácter, permite representar cualquier lenguaje humano conocido. Este estándar está en*

continúa revisión y contiene más de setenta mil caracteres diferentes. Pero, el uso directo de este estándar conlleva consigo la incompatibilidad con las herramientas actuales y la producción de archivos de gran tamaño. Por ello, en vez de usar la codificación directa usando 4 bytes (denominado UCS-4) se han definido representaciones alternativas como UTF-8. Esta codificación representa los caracteres ASCII con un solo byte, lo que asegura compatibilidad y un tamaño reducido en los documentos que contengan sólo información tipo ASCII.

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

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

- `&#código_decimal;`
- `ódigo_hexadecimal;`

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

Definición de tipo de documento (DTD, *Document Type Definition*)

Una DTD es un documento que define la estructura de un documento XML: los elementos, atributos, entidades, notaciones, etc, que pueden aparecer, el orden y el número de veces que pueden aparecer, cuáles pueden ser hijos de cuáles, etc. El procesador XML la utiliza para verificar si un documento es válido, es decir, si el documento cumple las reglas del DTD.

Declaración de tipo de documento (DOCTYPE, *Document type declaration*)

Una declaración de tipo de documento es una etiqueta que comienza por `<!DOCTYPE` y acaba por `>` y que indica la(s) DTD(s) que debe utilizar el procesador XML para validar el documento. La DTD puede estar incluida en el propio documento o ser un documento externo. Por ejemplo, el siguiente ejemplo muestra la declaración de tipo de documento que se debe incluir en los documentos XHTML 1.0 de tipo strict (en este caso, la DTD es un documento externo):


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Instrucciones de procesamiento (PI, *processing instruction*)

Una instrucción de procesamiento en una etiqueta que empieza por "<?" y acaba por "?>" y que contiene instrucciones dirigidas a las aplicaciones que leen el documento. Pueden aparecer en cualquier lugar del documento. Por ejemplo:

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

Entidades de carácter

En XML se utilizan varias entidades de carácter de HTML, para poder escribir en cadenas de texto los caracteres que delimitan las marcas o las cadenas de texto :

Referencia a entidad	Carácter
<	<
>	>
&	&
'	'
"	"

Secciones CDATA (*CDATA section*)

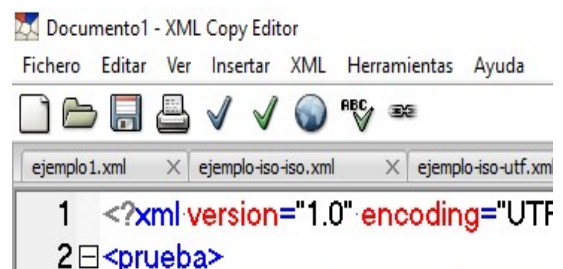
Una sección CDATA es una etiqueta que comienza por <![CDATA[y termina por]]> y cuyo contenido el procesador XML no interpreta como marcas sino como texto. Es decir, que si aparecen los caracteres especiales (< & " ') en una sección CDATA, el procesador XML no interpreta que empieza una marca, sino que lo considera un carácter más. Se suele utilizar en documentos en los que aparecen muchas veces esos caracteres especiales para no tener que estar utilizando las referencias a entidades (< y &) que dificultan la lectura del documento.

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

```
<prueba>
```

```
<texto>Los caracteres < y & no pueden escribirse  
si no es como comienzo de marcas</texto>
```

```
</prueba>
```



ejemplo1.xml - XML Copy Editor
Fichero Editar Ver Insertar XML Herramientas Ayuda



```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <texto>Los caracteres &lt; y &amp; no pueden escribirse
  si no es como comienzo de marcas</texto>
</prueba>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <texto><![CDATA["Los caracteres < y & no pueden escribirse
  si no es como comienzo de marcas"]]></texto>
</prueba>
```



Referencias a entidades

En XML se pueden definir entidades y utilizarlas como se utilizan las entidades de carácter en el HTML. Una entidad se define mediante una etiqueta que comienza por **<!ENTITY** y termina por **>** y **contiene el nombre y el valor de la entidad**. Por ejemplo:

Como las entidades de carácter del HTML, para hacer referencia a una entidad se escribe sin espacios intermedios el carácter "&", el nombre de la entidad y el carácter ";". Al abrir el documento XML el procesador sustituye la referencia a la entidad por su valor. Por ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE autor [
  <!ENTITY yo "John Doe">
  <!ELEMENT autor (#PCDATA)>
]>
<autor>&yo;</autor>
```

```
<autor>John Doe</autor>
```

1.2.-Documentos bien formados

Un documento XML debe estar bien formado, es decir debe cumplir las **reglas de sintaxis de la recomendación XML**. Para que un documento esté bien formado, al menos debe cumplir los siguientes puntos:

- El documento contiene únicamente caracteres **Unicode válidos**.
- Hay un elemento **raíz** que contiene al resto de elementos.
- Los nombres de los elementos y de sus atributos **NO contienen espacios**.
- El **primer carácter** de un nombre de elemento o de atributo puede ser una letra, dos puntos (:) o subrayado (_).
- El resto de caracteres pueden ser también números, guiones (-) o puntos (.).
- Los caracteres "<" y "&" sólo se utilizan como comienzo de marcas.
- Las etiquetas de apertura, de cierre y vacías están **correctamente anidadas** (no se solapan) y no falta ni sobra ninguna etiqueta de apertura o cierre.
- Las etiquetas de cierre coinciden con las de apertura (incluso en el **uso de mayúsculas y minúsculas**).
- Las **etiquetas de cierre no contienen atributos**.
- Ninguna etiqueta tiene dos **atributos con el mismo nombre**.
- Todos los atributos tienen algún valor.
- Los valores de los atributos están entre **comillas**.
- No existen referencias en los valores de los atributos.

Si un documento XML no está bien formado, no es un documento XML. Los procesadores XML deben rechazar cualquier documento que contenga errores.

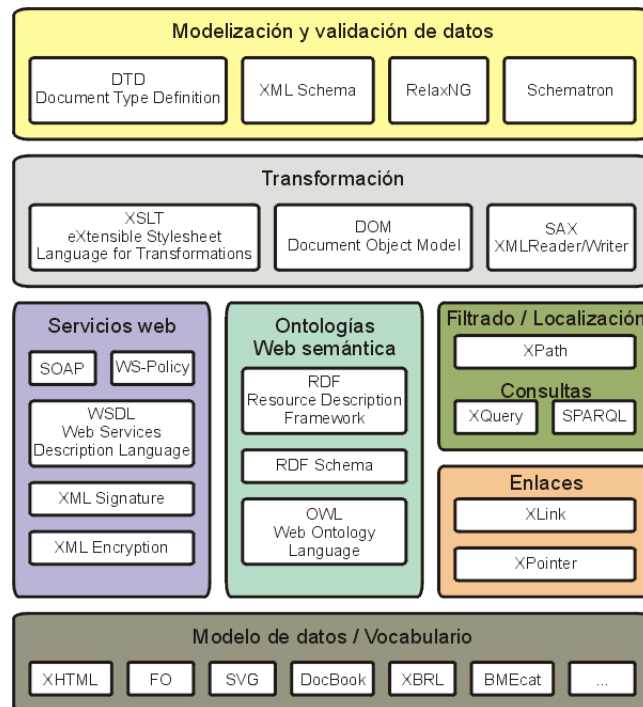
1.3.-Documentos válidos

Un documento XML bien formado puede ser válido. Para ser válido, un documento XML debe:

- incluir una referencia a una gramática,
- incluir únicamente elementos y atributos definidos en la gramática,
- cumplir las reglas gramaticales definidas en la gramática.

Existen varias formas de definir una gramática para documentos XML, las más empleadas son :

- DTD (Document Type Definition = Definición de Tipo de Documento). Es el modelo más antiguo, heredado del SGML.
- XML Schema. Es un modelo creado por el W3C como sucesor de las DTDs.
- Relax NG. Es un modelo creado por OASIS, más sencillo que XML Schema.



1.4.-Otras recomendaciones XML

El W3C y otras organizaciones de normalización han publicado numerosas recomendaciones relacionadas con XML. El cuadro siguiente cita algunas de ellas agrupándolas por temas

Las más empleadas son las siguientes:

XML Namespaces (Espacios de nombres XML)

Define los mecanismos para permitir que en un documentos se utilicen elementos y atributos de diferentes vocabularios, sin tener que preocuparse de que algunos nombres coincidan.

XML Base

Define el atributo `xml:base`, que puede utilizarse como base para resolver las referencias a URI relativas en un elemento XML

XML Infoset

Describe un modelo de datos abstracto para documentos XML a partir de elementos de información. Se utiliza en las especificaciones de lenguajes XML, para describir restricciones en el lenguaje.

`xml:id`

Define el atributo `id`

XPath

Define las expresiones XPath que sirven para identificar los componentes de un documento XML y facilitar su acceso a los programas que procesan documentos XML.

XSLT

Lenguaje de transformación de documentos XML a otros formatos (XML o no XML)

XSL Formatting Objects (XSL-FO)

Lenguaje de marcas para formatear documentos XML que se usa, por ejemplo, para generar PDFs.

XQuery

Lenguaje de consulta orientado a XML, que permite acceder, manipular y devolver fragmentos de documentos XML.

XML Signature

Define la sintaxis y las reglas de procesamiento para crear firmas digitales en documentos XML.

XML Encryption

Define la sintaxis y las reglas de procesamiento para encriptar documentos XML.

Otras recomendaciones del W3C relacionadas con el XML no han tenido mucho éxito, como XInclude, XLink y XPointer.

2.-DTD: Definición de Tipo de Documento

2.1.-Qué es una DTD

Una DTD es un documento que **define la estructura de un documento XML**: los elementos, atributos, entidades, notaciones, etc, que pueden aparecer, el orden y el número de veces que pueden aparecer, cuáles pueden ser hijos de cuáles, etc. El procesador XML utiliza la DTD para verificar si un **documento es válido**, es decir, si el documento cumple las reglas del DTD.

2.2.-Referencia a una DTD en un documento XML

La DTD que debe utilizar el procesador XML para validar el documento XML se indica mediante la etiqueta DOCTYPE. La DTD puede estar incluida en el propio documento, ser un documento externo o combinarse ambas.

- La DTD puede incluirse en el propio documento, con la siguiente sintaxis:

```
<!DOCTYPE nombre [  
... declaraciones ...  

```

- La DTD puede estar en un **documento externo** y, si sólo va a ser utilizada por una única aplicación, la sintaxis es la siguiente:

```
<!DOCTYPE nombre SYSTEM "uri">
```

Se puede combinar una **DTD externa con una DTD interna**, con la siguiente sintaxis:

```
<!DOCTYPE nombre SYSTEM "uri" [  
... declaraciones ...  

```

- La DTD puede estar en un documento externo y, si va a ser **utilizada por varias aplicaciones**, la sintaxis es la siguiente:

```
<!DOCTYPE nombre PUBLIC "fpi" "uri">
```

Se puede combinar una DTD externa con una DTD interna, con la siguiente sintaxis:

```
<!DOCTYPE nombre PUBLIC "fpi" "uri" [  
... declaraciones ...  

```

En todos estos casos:

- "**nombre**" es el nombre del tipo de documento XML, que **debe coincidir con el nombre del elemento raíz del documento XML.**
- "**uri**" es el camino (absoluto o relativo) hasta la DTD.
- "**fpi**" es un indentificador público formal (Formal Public Identifier).

2.3.-Declaraciones

Las DTDs describen la estructura de los documentos XML mediante declaraciones. Hay cuatro tipos de declaraciones:

- Declaraciones de **entidades**
- Declaraciones de **notaciones**
- Declaraciones de **elementos**, que indican los elementos permitidos en un documento y su contenido (que puede ser simplemente texto u otros elementos).
- Declaraciones de **atributos**, que indican los atributos permitidos en cada elemento y el tipo o valores permitidos de cada elemento.

Declaración de entidades

Una entidad consiste en un nombre y su valor (son similares a las constantes en los lenguajes de programación). Con algunas excepciones, el procesador XML sustituye las referencias a entidades por sus valores antes de procesar el documento. Una vez definida la entidad, se puede utilizar en el documento escribiendo una referencia a la entidad, que empieza con el carácter "&", sigue con el nombre de la entidad y termina con ";". (es decir, &nombreEntidad;)

Las entidades pueden ser **internas o externas** y tanto unas como otras pueden ser **generales o paramétricas**.

Las declaraciones de **entidades internas (generales)** siguen la siguiente sintaxis:

```
<!ENTITY nombreEntidad "valorEntidad">
```

En las declaraciones de **entidades externas (generales)** se distinguen dos casos:

- **La entidad hace referencia a un fichero de texto y en ese caso la entidad se sustituye por el contenido del archivo.**

La entidad puede ser una **entidad de sistema**, con la siguiente sintaxis:

```
<!ENTITY nombreEntidad SYSTEM "uri">
```

o puede ser una **entidad pública**, con la siguiente sintaxis:

```
<!ENTITY nombreEntidad PUBLIC "fpi" "uri">
```

- **La entidad hace referencia a un fichero que no es de texto (por ejemplo, una imagen) y en ese caso la entidad no se sustituye por el contenido del archivo.**

La entidad puede ser una entidad de sistema, con la siguiente sintaxis:

```
<!ENTITY nombreEntidad SYSTEM "uri" NDATA tipo>
```

o puede ser una entidad pública, con la siguiente sintaxis:

```
<!ENTITY nombreEntidad PUBLIC "fpi" "uri" NDATA tipo>
```

En todos estos casos:

- **"nombreEntidad"** es el nombre de la entidad.
- **"valorEntidad"** es el valor de la entidad.
- **"uri"** es el camino (absoluto o relativo) hasta un archivo.
- **"tipo"** es el tipo de archivo (gif, jpg, etc).
- **"fpi"** es un indentificador público formal (Formal Public Identifier).

Las declaraciones de entidades paramétricas siguen la mismas sintaxis que las generales, pero llevan el carácter "%" antes del nombre de la entidad. Por ejemplo:

```
<!ENTITY % nombreEntidad "valorEntidad">
```

```
<!ENTITY % nombreEntidad SYSTEM "uri">
```

```
<!ENTITY % nombreEntidad SYSTEM "uri" NDATA tipo>
```

La diferencia entre entidades generales y paramétricas es que las entidades paramétricas se sustituyen por su valor en todo el documento (incluso en la propia declaración de tipo de documento) mientras que las generales no se sustituyen en la declaración de tipo de documento.

Declaración de notaciones

Las notaciones se usan en XML para definir las entidades externas que no va a analizar el procesador XML (aunque sí lo hará la aplicación que trate un documento). Para hacer referencia estas entidades no se utiliza la notación habitual (&nombreEntidad;), sino que se utiliza el nombre de la entidad directamente.

Declaración de elementos

Las declaraciones de los elementos siguen la siguiente sintaxis:

```
<!ELEMENT nombreElemento (contenido)>
```






en la que "nombreElemento" es el nombre del elemento, y "(contenido)" una expresión que describe el contenido del elemento.

Para definir el contenido del elemento se pueden utilizar los términos **EMPTY, (#PCDATA) o ANY** o escribir expresiones más complejas:

- **EMPTY**: significa que el elemento es vacío, es decir, que no puede tener contenido. Los elementos vacíos pueden escribirse con etiquetas de apertura y cierre sin nada entre ellos, ni siquiera espacios, o con una etiqueta vacía. EMPTY debe escribirse sin paréntesis.

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  




```

<code><ejemplo></ejemplo></code>	
<code><ejemplo /></code>	
<code><ejemplo>Esto es un ejemplo</ejemplo></code> <code><!-- ERROR: contiene texto --></code>	
<code><ejemplo><a></ejemplo></code> <code><!-- ERROR: contiene un elemento <a> --></code>	

- **(#PCDATA)**: significa que el elemento puede contener texto. #PCDATA debe escribirse entre paréntesis.

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (#PCDATA)>  




```

<ejemplo />	
<ejemplo>Esto es un ejemplo</ejemplo>	
<ejemplo><a></ejemplo> <!-- ERROR: contiene un elemento <a> -->	

- **ANY:** significa que el elemento puede contener cualquier cosa (texto y otros elementos). ANY debe escribirse sin paréntesis.

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo ANY>  
  <!ELEMENT a ANY>  


```





<ejemplo />	
<ejemplo>Esto es un ejemplo</ejemplo>	
<ul style="list-style-type: none"> • <ejemplo><a>un ejemplo</ejemplo> 	

Para indicar que un elemento puede o debe contener otros elementos se deben indicar los elementos, utilizando los conectores y modificadores siguientes:

- **, (coma):** significa que el elemento contiene los elementos en el **orden indicado**.

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (a, b)>  
  <!ELEMENT a EMPTY>  
  <!ELEMENT b EMPTY>  






```

<ejemplo><a /></ejemplo>	
<ejemplo><a /><c /></ejemplo> <!-- ERROR: contiene un elemento <c /> -->	
<ejemplo><a /></ejemplo> <!-- ERROR: falta el elemento -->	
<ejemplo><a /></ejemplo> <!-- ERROR: el orden no es correcto -->	

- **| (o lógico):** significa que el elemento contiene uno de los dos elementos.





```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (a | b)>  
  <!ELEMENT a EMPTY>  
  <!ELEMENT b EMPTY>  


```

<ejemplo><a /></ejemplo>	
<ejemplo></ejemplo>	
<ejemplo><a /></ejemplo> <!-- ERROR: están los dos elementos -->	
<ejemplo></ejemplo> <!-- ERROR: no hay ningún elemento -->	



- **?**: significa que el elemento **puede aparecer o no, pero sólo una vez.**



```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo (a, b?)>
  <!ELEMENT a EMPTY>
  <!ELEMENT b EMPTY>
]>
```

<ejemplo><a /></ejemplo>	
<ejemplo><a /></ejemplo>	
<ejemplo></ejemplo> <!-- ERROR: falta el elemento <a /> -->	
<ejemplo></ejemplo> <!-- ERROR: el elemento aparece dos veces →	

- *****: significa que el elemento puede no aparecer o aparecer una o más veces.




```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo (a*, b)>
  <!ELEMENT a EMPTY>
  <!ELEMENT b EMPTY>
]>
```

<ejemplo></ejemplo>	
<ejemplo><a /></ejemplo>	

<ejemplo><a /><a /></ejemplo>	
<ejemplo><a /></ejemplo> <!-- ERROR: el elemento <a /> aparece después de 	




- **+**: significa que el elemento tiene que aparecer una o más veces (no puede no aparecer).

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo (a+, b)>
  <!ELEMENT a EMPTY>
  <!ELEMENT b EMPTY>
]>
```




<ejemplo><a /></ejemplo>	
<ejemplo><a /><a /></ejemplo>	
<ejemplo></ejemplo> <!-- ERROR: falta el elemento <a /> →	

- **()**: permite agrupar expresiones.

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo (a, (a|b))>
  <!ELEMENT a EMPTY>
  <!ELEMENT b EMPTY>
]>
```

<ejemplo><a /></ejemplo>	
<ejemplo><a /><a /></ejemplo>	
<ejemplo><a /></ejemplo> <!-- ERROR: falta el elemento <a /> o -->	

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo ((a, b)|(b, a))>
  <!ELEMENT a EMPTY>
  <!ELEMENT b EMPTY>
]>
```

<ejemplo><a /></ejemplo>	
<ejemplo><a /></ejemplo>	
<ejemplo><a /><a /></ejemplo> <!-- ERROR: sólo admite <a /> o <a /> -->	

Declaración de atributos

Una declaración de atributos sigue la siguiente sintaxis:

```
<!ATTLIST nombreElemento nombreAtributo tipoAtributo valorInicialAtributo >
```

en la que:

- "**nombreElemento**" es el nombre del elemento para el que se define un atributo.

- "**nombreAtributo**" es el nombre del atributo.
- "**tipoAtributo**" es el tipo de datos .
- "**valorInicialAtributo**" es el valor predeterminado del atributo (aunque también puede indicar otras cosas).

Para definir varios atributos de un mismo elemento, se puede utilizar una o varias declaraciones de atributos. Los siguientes ejemplos son equivalentes:

```
<!ATTLIST nombreElemento nombreAtributo1 tipoAtributo1 valorInicialAtributo1>
<!ATTLIST nombreElemento nombreAtributo2 tipoAtributo2 valorInicialAtributo2>
```

```
<!ATTLIST nombreElemento
  nombreAtributo1 tipoAtributo1 valorInicialAtributo1
  nombreAtributo2 tipoAtributo2 valorInicialAtributo2
>
```

Los tipos de atributos son los siguientes:






- **CDATA:** el atributo contiene caracteres (sin restricciones).

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo EMPTY>
  <!ATTLIST ejemplo color CDATA #REQUIRED>
]>
```





<ejemplo color="" />	
<ejemplo color="amarillo" />	
<ejemplo color="azul marino #000080" />	
<ejemplo /> <!-- ERROR: falta el atributo "color", obligatorio debido al #REQUIRED -->	
<ejemplo sabor="dulce" /> <!-- ERROR: el atributo "sabor" no está definido -->	

- **NMTOKEN**: el atributo sólo contiene letras, dígitos, y los caracteres punto ".", guión "-", subrayado "_" y dos puntos ":".

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo EMPTY>
  <!ATTLIST ejemplo color NMTOKEN #REQUIRED>
]>
```




<ejemplo color="" />	
<ejemplo color="azul-marino" />	
<ejemplo color="1" />	
<ejemplo color="azul marino" /> <!-- ERROR: hay un espacio en blanco -->	
<ejemplo color="#F0F0F0" /> <!-- ERROR: contiene el carácter # -->	

- **NMTOKENS**: el atributo sólo contiene letras, dígitos, y los caracteres punto ".", guión "-", subrayado "_", dos puntos ":" (como el tipo NMTOKEN) y también espacios en blanco.
- <!DOCTYPE ejemplo [
 <!ELEMENT ejemplo EMPTY>
 <!ATTLIST ejemplo color NMTOKENS #REQUIRED>
]>

<ejemplo color="" />	
<ejemplo color="1" />	
<ejemplo color="azul marino" />	
<ejemplo color="2*2" /> <!-- ERROR: hay un asterisco -->	

- **valores:** el atributo sólo puede contener uno de los términos de una lista. La lista se escribe entre paréntesis, con los términos separados por una barra vertical "|".

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo EMPTY>
  <!ATTLIST ejemplo color (azul|blanco|rojo) #REQUIRED>
]>
```

<ejemplo color="" />	
<ejemplo color="azul" />	
<ejemplo color="verde" /> <!-- ERROR: "verde" no está en la lista de valores -->	

- **ID:** el valor del atributo (no el nombre) debe ser único y no se puede repetir en otros elementos o atributos.

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo (libro*)>  
  <!ELEMENT libro (#PCDATA) >  
  <!ATTLIST libro codigo ID #REQUIRED>  

```

```
<ejemplo>  
  <libro codigo="L1">Poema de Gilgamesh</libro>  
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>  
</ejemplo>
```



```
<ejemplo>  
  <libro codigo="1">Poema de Gilgamesh</libro>  
  <!-- ERROR: el valor de un atributo de tipo ID no puede empezar con un número -->  
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>  
</ejemplo>
```



```
<ejemplo>  
  <libro codigo="L1">Poema de Gilgamesh</libro>  
  <libro codigo="L1">Los preceptos de Ptah-Hotep</libro>  
  <!-- ERROR: no se puede repetir un atributo de tipo ID -->  
</ejemplo>
```



- **IDREF:** el valor del atributo debe coincidir con el valor del atributo ID de otro elemento.

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo ((libro|prestamo)*)>  
  <!ELEMENT libro (#PCDATA) >  
  <!ATTLIST libro codigo ID #REQUIRED>  
  <!ELEMENT prestamo (#PCDATA) >  
  <!ATTLIST prestamo libro IDREF #REQUIRED>  

```

```
<ejemplo>  
  <libro codigo="L1">Poema de Gilgamesh</libro>  
  <prestamo libro="L1">Numa Nigerio</prestamo>  
</ejemplo>
```



```
<ejemplo>
  libro codigo="L1">Poema de Gilgamesh</libro>
  <prestamo libro="L2">Numa Nigerio</prestamo>
  <!-- ERROR: el valor "L2" no es ID de ningún elemento -->
</ejemplo>
```



- **IDREFS:** el valor del atributo es una serie de valores **separados por espacios** que coinciden con el valor del atributo ID de otros elementos.

```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo ((libro|prestamo)*)>
  <!ELEMENT libro (#PCDATA) >
  <!ATTLIST libro codigo ID #REQUIRED>
  <!ELEMENT prestamo (#PCDATA) >
  <!ATTLIST prestamo libro IDREFS #REQUIRED>
]>
```

```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
  <prestamo libro="L1 L2">Numa Nigerio</prestamo>
</ejemplo>
```



```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
  <prestamo libro="L1">Numa Nigerio</prestamo>
</ejemplo>
```



```
<ejemplo>
  <libro codigo="L1">Poema de Gilgamesh</libro>
  <libro codigo="L2">Los preceptos de Ptah-Hotep</libro>
  <prestamo libro="L3">Numa Nigerio</prestamo>
  <!-- ERROR: el valor "L3" no es ID de ningún elemento -->
</ejemplo>
```







- **ENTITY:** el valor del atributo es alguna entidad definida en la DTD.
- **ENTITIES:** el valor del atributo es alguna de las entidades de una lista de entidades definida en la DTD.
- **NOTATION:** el valor del atributo es alguna notación definida en la DTD.

Los valores iniciales de los atributos son los siguientes:

- **#REQUIRED:** el atributo es obligatorio, aunque no se especifica ningún valor predeterminado.

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  
  <!ATTLIST ejemplo color CDATA #REQUIRED>  





```

<code><ejemplo color="" /></code>	
<code><ejemplo color="amarillo" /></code>	
<code><ejemplo color="azul marino #000080" /></code>	
<code><ejemplo /></code> <code><!-- ERROR: falta el atributo "color" --></code>	

- **#IMPLIED:** el atributo no es obligatorio y no se especifica ningún valor predeterminado.





```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  
  <!ATTLIST ejemplo color CDATA #IMPLIED>  

```

<ejemplo />	
<ejemplo color="" />	
<ejemplo color="amarillo" />	
<ejemplo color="azul marino #000080" />	

- **#FIXED valor:** el atributo tiene un valor fijo.





```
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo EMPTY>
  <!ATTLIST ejemplo color CDATA #FIXED "verde">
]>
```

<ejemplo />	
<ejemplo color="verde" />	
<ejemplo color="" /> <!-- ERROR: el atributo "color" no tiene el valor "verde" -->	
<ejemplo color="amarillo" /> <!-- ERROR: el atributo "color" no tiene el valor "verde" -->	

- **valor:** el atributo tiene un valor predeterminado.

```
<!DOCTYPE ejemplo [  
  <!ELEMENT ejemplo EMPTY>  
  <!ATTLIST ejemplo color CDATA "verde">  

```

<code><ejemplo /></code>	
<code><ejemplo color="" /></code>	
<code><ejemplo color="amarillo" /></code>	
<code><ejemplo color="verde" /></code>	

3.- Hojas de estilo CSS en XML

3.1.-CSS en XML

Las hojas de estilo CSS se desarrollaron para utilizarse junto con el lenguaje de marcas HTML, pero un documento XML también puede hacer referencia a una hoja de estilo CSS y visualizarse en el navegador.

El W3C aprobó en junio de 1999 la recomendación [Associating Style Sheets with XML documents 1.0](#), editada por James Clark y en octubre de 2010 aprobó la recomendación [Associating Style Sheets with XML documents 1.0 \(2ª edición\)](#), editada por James Clark, que definen cómo vincular documentos XML con hojas de estilo CSS.

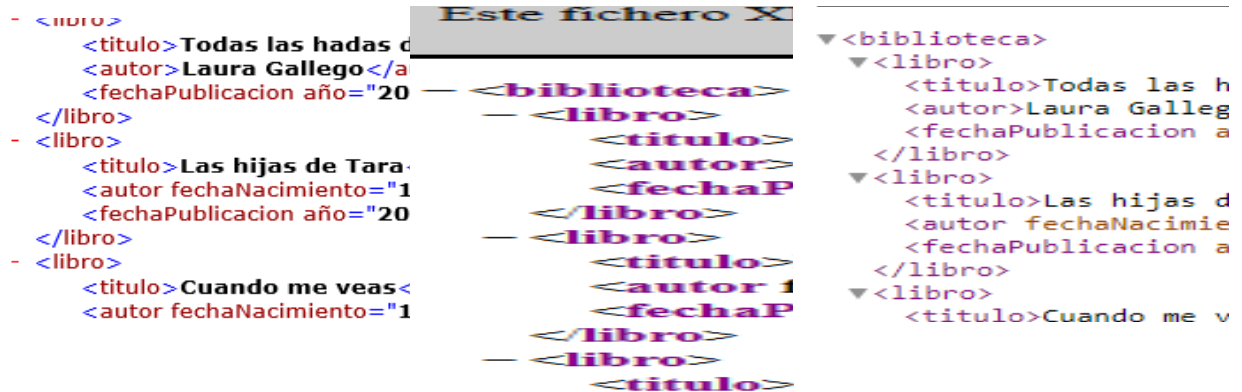
3.2.-Documentos XML sin hojas de estilo

Cuando un documento XML no enlaza a una hoja de estilo, los navegadores muestran el contenido completo del documento, etiquetas incluidas.

Por ejemplo, el documento siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>Todas las hadas del reino</titulo>
    <autor>Laura Gallego</autor>
    <fechaPublicacion año="2015"/>
  </libro>
  <libro>
    <titulo>Las hijas de Tara</titulo>
    <autor fechaNacimiento="11/10/1977">Laura Gallego</autor>
    <fechaPublicacion año="2018"/>
  </libro>
  <libro>
    <titulo>Cuando me veas</titulo>
    <autor fechaNacimiento="11/10/1977">Laura Gallego</autor>
    <fechaPublicacion año="2017"/>
  </libro>
</biblioteca>
```

... se vería así al abrirlo en los diferentes navegadores:



3.3.-Documentos XML con hojas de estilo

De acuerdo con la recomendación [Associating Style Sheets with XML documents 1.0 \(2º edición\)](#), un documento XML puede enlazar a una hoja de estilo mediante la instrucción de procesamiento **<?xml-stylesheet ?>**, de forma similar a como se hace en una página web XHTML con la etiqueta **<link />**. En ambos casos el atributo href incluye el camino absoluto o relativo a la hoja de estilo CSS.

La diferencia es que la etiqueta **<link />** forma parte del encabezamiento (etiqueta **<head>**), mientras que la instrucción de procesamiento **<?xml-stylesheet ... ?>** va al principio del documento, después de la declaración XML, como muestra el siguiente ejemplo:

Ejemplo de enlace a CSS en HTML5

Ejemplo de enlace a CSS en HTML5

```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <title>HTML 5</title>
  <link rel="stylesheet" type="text/css"
href="estilo.css" />
</head>
<body>
  <p>Esta página es HTML 5 válido.</p>
</body>
</html>

```

Ejemplo de enlace a CSS en XML

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo_2.css"?>
<libro>
  <titulo>La dentista </titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="2016">
</libro>

```

Si un documento XML enlaza a una hoja de estilo, los navegadores ya no muestran las etiquetas y aplican la hoja de estilo.

- En el ejemplo siguiente, se puede comprobar que el navegador ya no muestra las etiquetas, sino únicamente el texto contenido en ellas. Y como la hoja de estilo no contiene ninguna regla, todo el texto se ve de la misma manera.

Pero también se observa que el navegador muestra el texto de los dos elementos <titulo> y <autor> uno detrás de otro. Eso se debe a que el navegador aplica únicamente **la hoja de estilo enlazada y no aplica la hoja de estilo predeterminada del navegador**. Más adelante en esta misma unidad, el apartado la propiedad **display** comenta cómo definir la forma de presentación de los elementos (como bloques, listas o tablas, por ejemplo).

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo_2_1.css"?>
<libro>
  <titulo>Cuando me veas</titulo>
  <autor>Laura Gallego</autor>
  <fechaPublicacion año="2015"/>
</libro>
```

Resultado en el navegador:

Cuando me veas Laura Gallego

- En el ejemplo siguiente, la hoja de estilo sí que modifica el aspecto del elemento <titulo> (aunque el texto se siga mostrando todo junto, por el mismo motivo que en el ejemplo anterior)..

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo_2_2.css"?>
<libro>
  <titulo>La dentista </titulo>
  <autor>Laura Gallego</autor>
  <fechaPublicacion año="2016">
</libro>
```

```
titulo{
  color:red;
}
```

La dentista Milan Kundera

Es posible "obligar" al navegador a aplicar la hoja de estilo predeterminada del navegador:

- Si el documento XML utiliza **el espacio de nombres del html** y utiliza etiquetas html, el navegador aplica la hoja de estilo predeterminada del navegador a esos elementos, como muestra el siguiente ejemplo:

XML	Resultado
<pre><?xml version="1.0" encoding="UTF-8"?> <libro xmlns="http://www.w3.org/1999/xhtml"> <h1>La vida está en otra parte</h1> <autor>Milan Kundera</autor> <personaje>Jaromil</personaje> <fechaPublicacion año="1973"/> </libro></pre>	<p>La vida está en otra parte</p> <p>Milan Kundera Jaromil</p>

- Pero si no se utiliza el **espacio de nombres**, aunque se utilicen etiquetas **html** el **navegador no aplica la hoja de estilo predeterminada**, como muestra el siguiente ejemplo:

XML	Resultado
<pre><?xml version="1.0" encoding="UTF-8"?> <libro> <h1>La vida está en otra parte</h1> <autor>Milan Kundera</autor> <personaje>Jaromil</personaje> <fechaPublicacion año="1973"/> </libro></pre>	<p>Este fichero XML no parece tener ninguna información de estilo asociada debajo el árbol del documento.</p> <pre>- <libro> <h1>La vida está en otra parte</h1> <autor>Milan Kundera</autor></pre>

Como ocurre en el HTML, ...

- Un documento XML puede enlazar a varias hojas de estilo, que se aplicarán simultáneamente:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo_3_1.css"?>
<?xml-stylesheet href="ejemplo_3_2.css"?>
<libro>
  <titulo>Cuando me veas</titulo>
  <autor>Laura Gallego</autor>
  <fechaPublicacion año="2017"/>
</libro>
```

```
/* ejemplo_3_1.css */
titulo {
  color: red;
}
/* ejemplo_3_2.css */
autor {
  font-size: 200%;
}
```

- El enlace a la hoja de estilo puede contener el atributo **title**, cuyo contenido se muestra en Firefox en el menú **Ver > Estilo de página**.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo_4_1.css" title="EstiloLM"?>
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

```
titulo {
  color: red;
}
```

- Un documento XML puede enlazar a hojas de estilo alternativas, que en Firefox se pueden seleccionar mediante el menú **Ver > Estilo de página**.

Para ser tratadas como hojas de estilo alternativas, basta con que los **enlaces incluyan el atributo title**. La primera hoja de estilo enlazada es la hoja de estilo predeterminada y el resto, alternativas.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo_5_1.css" title="Estilo"?>
<?xml-stylesheet href="ejemplo_5_2.css" title="Otro estilo"?>
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

- Si se quiere indicar explícitamente qué hoja de estilo son las alternativas, se puede incluir el **atributo alternate con el valor yes**.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo_6_1.css" title="Estilo"?>
<?xml-stylesheet alternate="yes" href="ejemplo_6_2.css"
title="Otro estilo"?>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
```

```
/* ejemplo_6_1.css */
titulo {
  color: red;
}
/* ejemplo_6_2.css */
autor {
  font-size: 200%;
}
```

3.4.-Ejemplos de hojas de estilo para XML

La propiedad display

Al escribir una hoja de estilo para un documento XML hay que tener en cuenta que los navegadores no aplican generalmente la hoja de estilo predeterminada (salvo que se utilicen etiquetas html asociándolas al espacio de nombres del xhtml como en los ejemplos anteriores), así que **al escribir una hoja de estilo para un documento XML hay que definir propiedades que normalmente no se suelen definir**.

La más importante es la **propiedad display**, que establece el modo de visualización del elemento.

Los valores posibles de la propiedad display son none, block, compact, inline, inline-block, inline-table, list-item, marker⁽⁻⁾, run-in, table, table-caption, table-cell, table-column, table-column-group, table-footer-group, table-header-group, table-row y table-row-group.

Los modos de visualización más utilizados son:

- **none** (ninguno) permite ocultar elementos.
- **block** (bloque) es el modo de visualización de los elementos de tipo bloque, como párrafos (<p>), encabezados (<h1>, <h2>, ...), etc. El elemento ocupa toda la ventana en horizontal y ocupa el espacio vertical necesario para alojar el contenido del elemento.

- **inline** (elemento en línea) es el modo de visualización de etiquetas como ``, ``, ``, `<a>`, etc. El elemento sólo ocupa el espacio necesario para alojar el contenido del elemento. En HTML los elementos en-línea tienen que estar contenidos dentro de elementos de bloque.
- **list-item (lista)** es el modo de visualización de elementos de listas como ``, `<dd>`, `<dt>`.
- **table (tabla), table-row (fila) y table-cell (celda)** se combinan para mostrar elementos en forma de tabla.

Los ejemplos a continuación muestran el documento siguiente utilizando diferentes hojas de estilo

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

```
<biblioteca>
```

```
  <libro>
```

```
    <titulo>Todas las hadas del reino</titulo>
```

```
    <autor>Laura Gallego</autor>
```

```
    <fechaPublicacion>2015</fechaPublicacion>
```

```
  </libro>
```

```
  <libro>
```

```
    <titulo>Las hijas de Tara</titulo>
```

```
    <autor fechaNacimiento="11/10/1977">Laura Gallego</autor>
```

```
    <fechaPublicacion>2018</fechaPublicacion>
```

```
  </libro>
```

```
  <libro>
```

```
    <titulo>Cuando me veas</titulo>
```

```
    <autor fechaNacimiento="11/10/1977">Laura Gallego</autor>
```

```
    <fechaPublicacion>2017</fechaPublicacion>
```

```
  </libro>
```

```
</biblioteca>
```

El único elemento de bloque es `<libro>`:

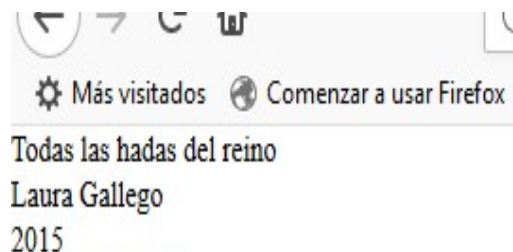
```
libro {  
  display: block;  
}
```

Resultado



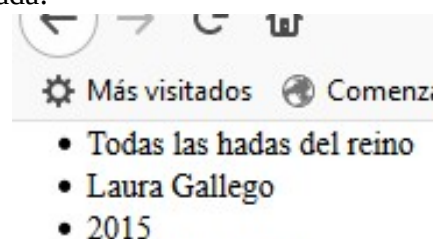
- `<titulo>`, `<autor>` y `<fechaPublicacion>` son bloques:

```
titulo, autor, fechaPublicacion {  
  display: block;  
}
```



- Mostrando el documento como una lista no ordenada:

```
titulo, autor, fechaPublicacion {  
  display: list-item;  
  list-style: disc inside;  
  margin-left: 20px;  
}
```



```
}
```

- Mostrando el documento como una lista ordenada (en el apartado siguiente se comenta con más detalle este caso):

```

biblioteca {
  counter-reset: contador;
}

libro:before {
  content: counter(contador, upper-alpha) ". ";
  counter-increment: contador;
}

titulo {
  display: inline;
  margin-left: 5px;
}

autor, fechaPublicacion {
  display: block;
  margin-left: 25px;
}

```

- Mostrando el documento como una tabla:

```

biblioteca {
  display: table;
  border-spacing: 10px;
  border: black 1px solid;
}

libro {
  display: table-row;
}

titulo, autor, fechaPublicacion {
  border: black 1px solid;
  display: table-cell;
}

```

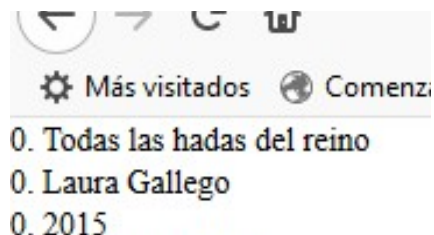
Mostrar elementos como listas ordenadas

Si queremos mostrar elementos en forma de lista ordenada, hay que tener en cuenta que *Firefox no muestra correctamente los contadores de las listas ordenadas*:

```

titulo, autor, fechaPublicacion {
  display: list-item;
  list-style: upper-alpha inside;
  margin-left: 0;
}

```



Nota: Firefox no aumenta el valor del contador.

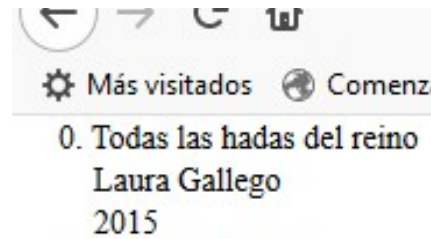
Además, hay que tener en cuenta que **los valores del contador aumentan en Chrome y Edge si los elementos con contadores se encuentran dentro del mismo elemento**. En el ejemplo siguiente, el contador sólo se encuentra en los elementos <titulo> y por tanto el contador no aumenta, ya que los elementos <titulo> pertenecen a diferentes elementos <libro>.

```

titulo {
  display: list-item;
  list-style: upper-alpha inside;
  margin-left: 20px;
}

autor, fechaPublicacion {
  display: block;
  margin-left: 35px;
}

```



Nota: Aunque los tres navegadores se comportan de la misma manera, Google Chrome y Edge no aumentan el contador porque no se debe hacer en este caso, mientras que Firefox no aumenta el valor del contador porque no lo hace nunca.

Si el contador se asocia al elemento <libro>, Chrome y Edge sí que aumentan el valor del contador, puesto que todos los elementos <libro> pertenecen al mismo elemento <biblioteca>.

```

libro {
  display: list-item;
  list-style: upper-alpha inside;
  margin-left: 20px;
}
titulo {
  display: inline;
}
autor, fechaPublicacion {
  display: block;
  margin-left: 20px;
}

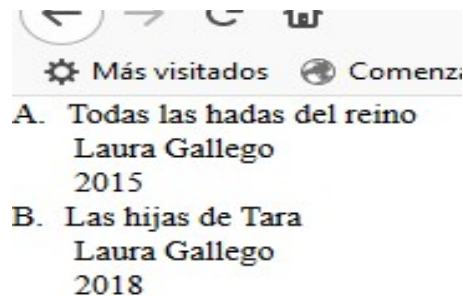
```

Para conseguir que se vean las listas ordenadas en todos los navegadores, hay que utilizar contadores CSS sobre **pseudo-clases y pseudo-elementos**.

```

biblioteca {
  counter-reset: contador;
}
libro:before {
  content: counter(contador, upper-alpha) ". ";
  counter-increment: contador;
}
titulo {
  display: inline;
  margin-left: 5px;
}
autor, fechaPublicacion {
  display: block;
  margin-left: 25px;
}

```



Al utilizar contadores, ya no necesario que los elementos con contador se encuentren contenidos en el mismo elemento:

```

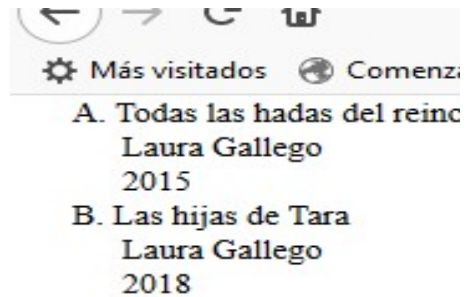
biblioteca {
  counter-reset: contador;
}

titulo:before {
  content: counter(contador, upper-alpha) ". ";
  counter-increment: contador;
}

titulo, autor, fechaPublicacion {
  display: block;
  margin-left: 25px;
}

autor, fechaPublicacion {
  margin-left: 45px;
}

```



Los atributos **class** e **id**

Cuando un documento XML incluye atributos **class** e **id** y la hoja de estilo hace referencia a ellos mediante los selectores almohadilla (#) y punto (.), **Firefox y Microsoft Edge aplican ambas reglas, pero Google Chrome aplica únicamente el selector almohadilla**, como muestra el ejemplo siguiente:

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="css/ejemplo_8_1.css"?>
<libro>
  <titulo class="color">La vida está en otra parte</titulo>
  <autor id="grande">Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>

```

```

.color {
  color: red;
}

#grande {
  font-size: 200%;
}

```

Firefox

La vida está en otra parte **Milan Kundera**

Google Chrome

La vida está en otra parte Milan Kundera

Microsoft Edge

La vida está en otra parte **Milan Kundera**

Otra solución es utilizar en la hoja de estilo el **selector de atributo** (por ejemplo, [atributo="valor"]), como muestra el siguiente ejemplo:

```

[class=color] {
  color: red;
}

[id=grande] {
  font-size: 200%;
}

```



Si se utilizan **selectores de atributos**, los nombres de los atributos no tienen por qué ser **class** e **id**, como muestra el siguiente ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo.css"?>
<libro>
  <titulo color="rojo">La vida está en otra parte</titulo>
  <autor tamaño="grande">Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

```
[color=rojo] {
  color: red;
}

[tamaño=grande] {
  font-size: 200%;
}
```



4.-XSL: Lenguaje de Hojas de Estilo Extensible

Aunque las hojas de estilo CSS se pueden aplicar a los documentos XML, las hojas de estilo tienen muchas limitaciones ya que se crearon para complementar al HTML, es decir, para ver páginas web en pantallas de ordenador. De la misma manera que el XML es una generalización del HTML, el W3C creó una generalización de las hojas de estilo CSS a la que se denominó XSL (eXtensible Stylesheet Language), es decir, Lenguaje de hojas de estilo extensible.

El W3C ha desarrollado tres lenguajes:

- **XPath**: un lenguaje para referirse a partes de un documento XML.
- **XSLT** (XSL Transformation, es decir, Transformaciones XSL): un lenguaje para transformar documentos XML.
- **XSL-FO** (XSL Formatting Objects, es decir, Objetos de formato XSL): un lenguaje para especificar el formato de un documento XML y posteriormente convertirlo a PDF o PostScript.

4.1.-XPath: XML Path language

4.1.1.-Qué es XPath

XPath es un lenguaje que permite seleccionar nodos de un documento XML y calcular valores a partir de su contenido. Existen varias versiones de XPath aprobadas por el W3C.

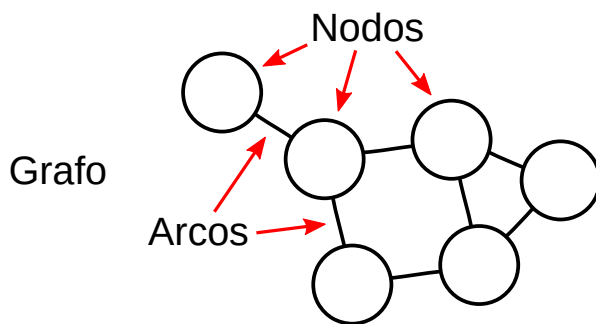
- noviembre de 1999: [XML Path Language \(XPath\) 1.0](#)
- enero de 2007: [XML Path Language \(XPath\) 2.0](#)
- diciembre de 2010: [XML Path Language \(XPath\) 2.0 \(2ª edición\)](#)
- abril de 2014: [XML Path Language \(XPath\) 3.0](#)
- marzo de 2017: [XML Path Language \(XPath\) 3.1](#)



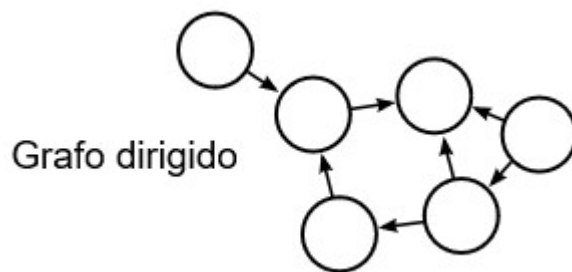
4.1.2.-Árbol del documento

XPath considera un documento XML como un árbol de nodos. En Informática, un árbol es una estructura de datos que equivale a un árbol matemático. En Matemáticas un árbol es un caso particular de grafo. Los siguientes términos definidos en teoría de grafos se utilizan también en Informática y en XPath:

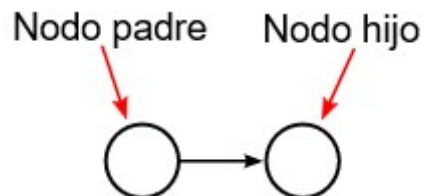
- Un **grafo** es un conjunto de objetos llamados nodos o vértices unidos por enlaces llamados arcos o aristas.



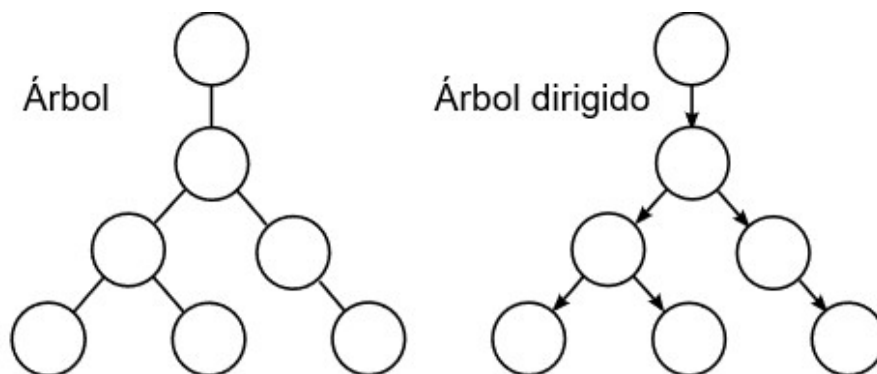
- Un **grafo dirigido** es un grafo en el que los arcos tienen dirección.



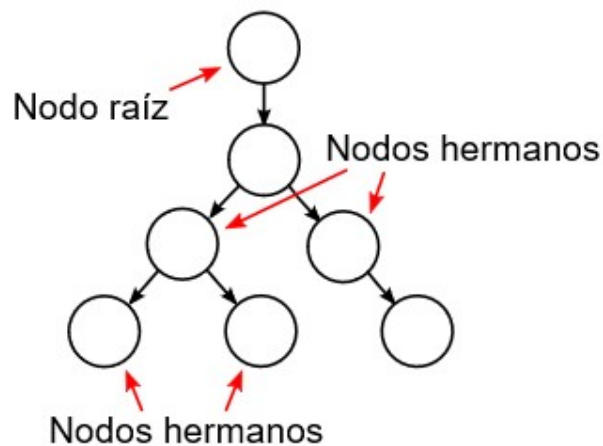
- Cuando dos nodos están unidos por un arco con dirección, el **nodo padre** es el nodo del que parte el arco y el **nodo hijo** es el nodo al que llega el arco.



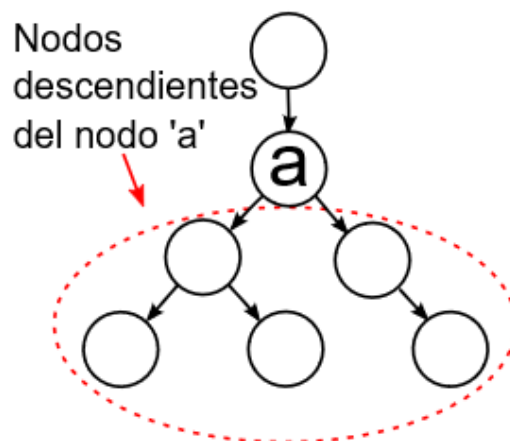
- Un **árbol** es un grafo en el que cualquier pareja de vértices están conectados por un único camino (es decir, que no hay ciclos). Un **árbol dirigido** es un árbol en el que las aristas tienen dirección y todos los nodos menos uno tienen un único padre.



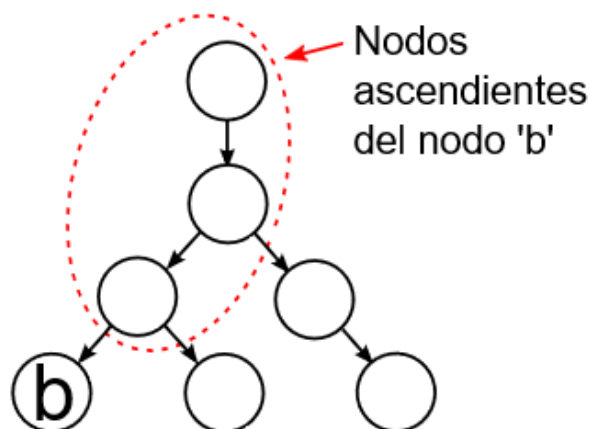
- El **nodo raíz** de un árbol dirigido es el único nodo sin padre. Los **nodos hermanos** son los nodos que tienen el mismo padre.



- Los **nodos descendientes** de un nodo son todos los nodos a los que se llega desde el nodo: los hijos, los hijos de los hijos, etc.



- Los **nodos ascendientes** de un nodo son todos los nodos de los que un nodo es descendiente: el padre, el padre del padre, etc.



Tipos de nodos

Un documento XML puede representarse como un árbol dirigido, considerando por ejemplo los elementos como nodos y que un elemento es padre de los elementos que contiene. Pero en XPath no sólo los elementos son nodos, en realidad hay siete tipos de nodos:

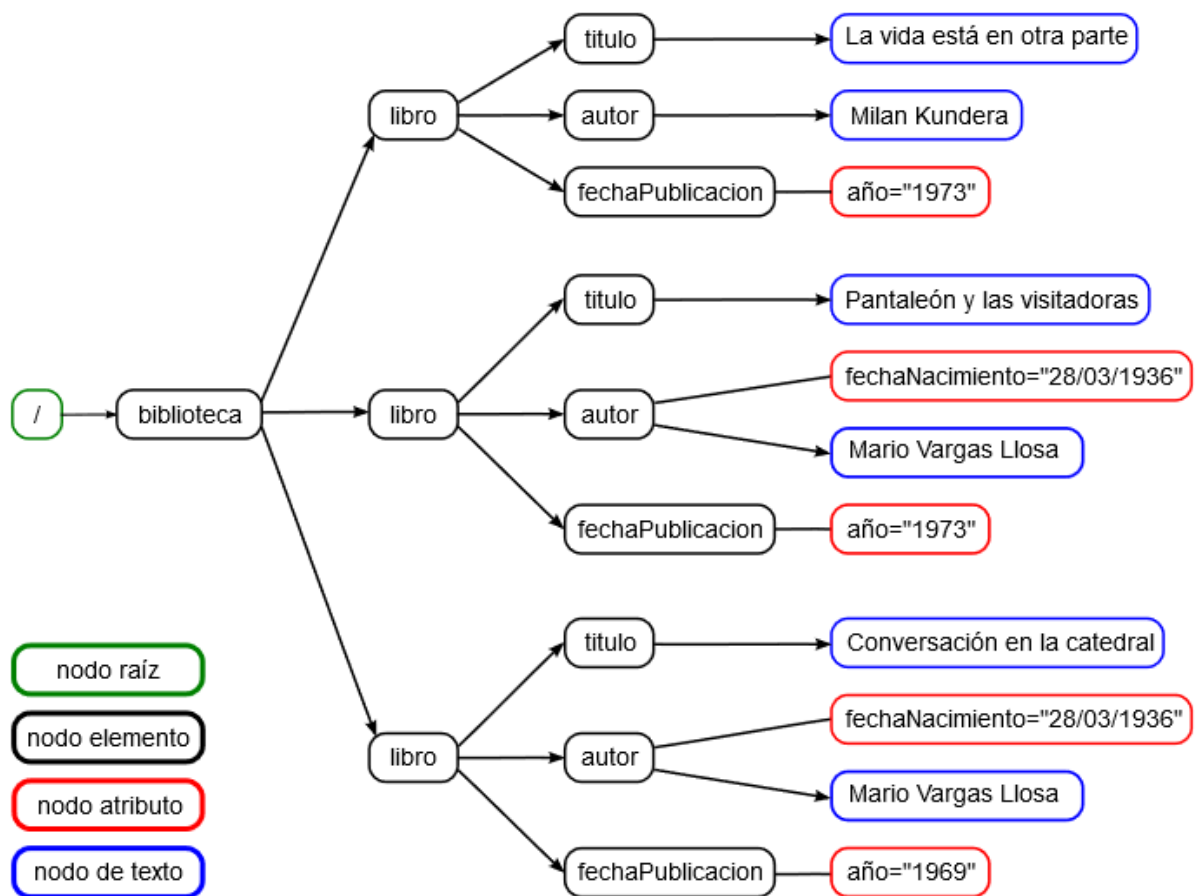
- Raíz
- Elemento
- Atributo
- Texto
- Comentario
- Instrucción de procesamiento
- Espacio de nombres

Nota: La declaración DOCTYPE no se considera como nodo.

Por ejemplo, el documento XML siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Conversación en la catedral</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1969"/>
  </libro>
</biblioteca>
```

se puede representar mediante el siguiente grafo:



Los nodos atributos y de texto no son como los nodos elemento. Por ejemplo, los nodos atributo y de texto no pueden tener descendientes. En realidad el nodo atributo ni siquiera se considera como hijo, sino como una etiqueta adosada al elemento. El texto contenido por una etiqueta sí que se considera hijo del elemento, aunque las expresiones XPath suelen trabajar con nodos elemento y para referirse a los atributos o al texto se utilizan notaciones especiales.

4.1.3.-Sintaxis de la expresiones XPath

Una expresión XPath es una cadena de texto que representa un recorrido en el árbol del documento. Las expresiones más simples se parecen a las rutas de los archivos en el explorador de Windows o en la shell de GNU/Linux.

Evaluar una expresión XPath es buscar si hay nodos en el documento que se ajustan al recorrido definido en la expresión. El resultado de la evaluación son todos los nodos que se ajustan a la expresión. Para poder evaluar una expresión XPath, el documento debe estar bien formado.

Las expresiones XPath se pueden escribir de dos formas distintas:

- **sintaxis abreviada:** más compacta y fácil de leer.

- **sintaxis completa:** más larga pero con más opciones disponibles

Las expresiones XPath se pueden dividir en pasos de búsqueda. Cada paso de búsqueda se puede a su vez dividir en tres partes:

- **eje:** selecciona nodos elemento o atributo basándose en sus nombres.
- **predicado:** restringe la selección del eje a que los nodos cumplan ciertas condiciones.
- **selección de nodos:** de los nodos seleccionados por el eje y predicado, selecciona los elementos, el texto que contienen o ambos.

4.1.4.-Sintaxis abreviada

Veamos unos ejemplos de expresiones XPath de sintaxis abreviada y el resultado de su evaluación en el documento de ejemplo anterior:

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Conversación en la catedral</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1969"/>
  </libro>
</biblioteca>
```

Eje (en inglés, axis)

El eje nos permite seleccionar un subconjunto de nodos del documento y corresponde a recorridos en el árbol del documento. Los nodos elemento se indican mediante el nombre del elemento. Los nodos atributo se indican mediante @ y el nombre del atributo.

/: (barra) si está al principio de la expresión, indica el nodo raíz, si no, indica "hijo". Debe ir seguida del nombre de un elemento.

/biblioteca/libro/autor	<autor>Milan Kundera</autor>
	<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
	<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>

/autor No devuelve nada porque <autor> no es hijo del nodo raíz.

/biblioteca/autor No devuelve nada porque <autor> no es hijo de <biblioteca>.

/biblioteca/libro/autor/@fechaNacimiento fechaNacimiento="28/03/1936"
fechaNacimiento="28/03/1936"

/biblioteca/libro/@fechaNacimiento No devuelve nada porque <libro> no tiene el atributo fechaNacimiento.

Nota: En XPath 1.0 no se puede seleccionar únicamente el valor del atributo, sino que se obtienen respuestas del tipo nombreDelAtributo=ValorDelAtributo

//: (doble barra) Indica "descendiente" (hijos, hijos de hijos, etc.).

/biblioteca//autor <autor>Milan Kundera</autor>
autor dentro de biblioteca <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
 <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>

//autor <autor>Milan Kundera</autor>
autor donde sea <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
 <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>

//autor//libro No devuelve nada porque <libro> no es descendiente de <autor>.

//@año todos los atributos año de cualquier elemento

año="1973"
año="1973"
año="1969"

/.: (barra punto punto) Indica el elemento padre.

Nota: En el resultado de los ejemplos siguientes se obtienen únicamente los nodos que tienen el atributo fechaNacimiento.

/biblioteca/libro/autor/@fechaNacimiento/.. <autor fechaNacimiento="28/03/1936">Mario
Vargas Llosa</autor>

 <autor fechaNacimiento="28/03/1936">Mario
Vargas Llosa</autor>

```
//@fechaNacimiento/... <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas
    Llosa</autor>
    <fechaPublicacion año="1973"/>
</libro>
<libro>
    <titulo>Conversación en la catedral</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas
    Llosa</autor>
    <fechaPublicacion año="1969"/>
</libro>
```

Libros con fecha de publicación

|: permite indicar varios recorridos. (o)

```
//autor//titulo    <titulo>La vida está en otra parte</titulo>
                   <autor>Milan Kundera</autor>
                   <titulo>Pantaleón y las visitadoras</titulo>
                   <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
                   <titulo>Conversación en la catedral</titulo>
                   <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

```
//autor//titulo//@año <titulo>La vida está en otra parte</titulo>
                      <autor>Milan Kundera</autor>
                      año="1973"
                      <titulo>Pantaleón y las visitadoras</titulo>
                      <autor fechaNacimiento="28/03/1936">Mario Vargas
                      Llosa</autor>
                      año="1973"
                      <titulo>Conversación en la catedral</titulo>
                      <autor fechaNacimiento="28/03/1936">Mario Vargas
                      Llosa</autor>
                      año="1969"
```

Predicado (en inglés, predicate)

El predicado se escribe entre corchetes, a continuación del eje. Si el eje ha seleccionado unos nodos, el predicado permite restringir esa selección a los que cumplan determinadas condiciones.

- **[@atributo]**: selecciona los elementos que tienen el atributo.

```
//autor[@fechaNacimiento]    <autor fechaNacimiento="28/03/1936">Mario
                              Vargas Llosa</autor>
                              <autor fechaNacimiento="28/03/1936">Mario
                              Vargas Llosa</autor>
```

- **[número]**: si hay varios resultados selecciona uno de ellos por número de orden; **last()** selecciona el último de ellos

```
//libro[1]                  <libro>
                              <titulo>La vida está en otra parte</titulo>
                              <autor>Milan Kundera</autor>
```



```
<fechaPublicacion  
año="1973"/> </libro>
```

```
//libro[last()] <libro>  
  <titulo>Conversación en la catedral</titulo>  
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>  
  <fechaPublicacion año="1969"/>  
</libro>
```

```
//libro[last()-1] <libro>  
  <titulo>Pantaleón y las visitadoras</titulo>  
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>  
  <fechaPublicacion año="1973"/>  
</libro>
```

- **[condicion]:** selecciona los nodos que cumplen la condición.

Los predicados permiten definir condiciones sobre los valores de los atributos. En las condiciones se pueden utilizar los operadores siguientes:

- operadores lógicos: and, or, not()
- operadores aritméticos: +, -, *, div, mod
- operadores de comparación: =, !=, <, >, <=, >=

Las comparaciones se pueden hacer entre valores de nodos y atributos o con cadenas de texto o numéricas. En el caso de las cadenas de texto deben estar rodeadas por comillas simples o dobles. En el caso de las cadenas numéricas, las comillas son optativas.

- La condición puede utilizar el valor de un atributo (utilizando @) o el texto que contiene el elemento.

En los ejemplos siguientes se obtienen respectivamente los elementos

<fechaPublicacion> cuyo atributo año es posterior/mayor a 1970 y los elementos

<libro> cuyo subelemento <autor> tiene como contenido "Mario Vargas Llosa":

```
//fechaPublicacion[@año>1970] <fechaPublicacion año="1973"/>  
 <fechaPublicacion año="1973"/>
```

//libro[autor="Mario Vargas Llosa"]

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario
Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario
Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

- Para hacer referencia al propio valor del elemento seleccionado se utiliza el punto (.).

//@año[.>1970]

```
año="1973"
año="1973"
```

//autor[.="Mario Vargas Llosa"]

```
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

- Un predicado puede contener condiciones compuestas.
En los ejemplos siguientes se seleccionan, respectivamente , los libros escritos por Mario Vargas Llosa y publicados en 1973 (primer ejemplo) y los libros escritos por Mario Vargas Llosa o publicados en 1973 (segundo ejemplo):

//libro[autor="Mario Vargas Llosa" **and** fechaPublicacion/@año="1973"]

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

//libro[autor="Mario Vargas Llosa" **or** fechaPublicacion/@año="1973"]

```
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

- Se pueden escribir varios predicados seguidos, cada uno de los cuales restringe los resultados del anterior, como si estuvieran encadenados por la operación lógica and. En el ejemplo siguiente se seleccionan los libros escritos por Mario Vargas Llosa y publicados en 1973:

- `//libro[autor="Mario Vargas Llosa"][fechaPublicacion/@año="1973"]`

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
```

Selección de nodos (en inglés, **node test**)

La selección de nodos se escribe a continuación del eje y el predicado. Si el eje y el predicado han seleccionado unos nodos, la selección de nodos indica con qué parte de esos nodos nos quedamos.

- **/node()**: selecciona todos los hijos (elementos o texto) del nodo.
- **//node()**: selecciona todos los descendientes (elementos o texto) del nodo.

`//libro/node()`

```
<titulo>La vida está en otra parte</titulo>
<autor>Milan Kundera</autor>
<fechaPublicacion año="1973"/>
<titulo>Pantaleón y las visitadoras</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1973"/>
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1969"/>
```

`//autor/node()`

```
Milan Kundera
Mario Vargas Llosa
Mario Vargas Llosa
```

//libro//**node()**

```
<titulo>La vida está en otra parte</titulo>
La vida está en otra parte
<autor>Milan Kundera</autor>
Milan Kundera
<fechaPublicacion año="1973"/>
<titulo>Pantaleón y las visitadoras</titulo>
Pantaleón y las visitadoras
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
Mario Vargas Llosa
<fechaPublicacion año="1973"/>
<titulo>Conversación en la catedral</titulo>
Conversación en la catedral
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
Mario Vargas Llosa
<fechaPublicacion año="1969"/>
```

- **/text()**: selecciona únicamente el texto contenido en el nodo.
//text(): selecciona únicamente el texto contenido en el nodo y todos sus descendientes.

//autor/**text()**

```
Milan Kundera
Mario Vargas Llosa
Mario Vargas Llosa
```

//libro/**text()**

No devuelve nada porque <libro> no contiene texto.

//libro//**text()**

```
La vida está en otra parte
Milan Kundera
Pantaleón y las visitadoras
Mario Vargas Llosa
Conversación en la catedral
Mario Vargas Llosa
```

- **/***: selecciona todos los hijos (sólo elementos) del nodo.
//*: selecciona todos los descendientes (sólo elementos) del nodo.
/biblioteca/*

```
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
```

```
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1969"/>
</libro>
```

//autor/*

No devuelve nada porque <autor> sólo contiene texto.

/biblioteca/*

```
<libro>
  <titulo>La vida está en otra parte</titulo>
  <autor>Milan Kundera</autor>
  <fechaPublicacion año="1973"/>
</libro>
<titulo>La vida está en otra parte</titulo>
<autor>Milan Kundera</autor>
<fechaPublicacion año="1973"/>
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<titulo>Pantaleón y las visitadoras</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1973"/>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
<titulo>Conversación en la catedral</titulo>
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
<fechaPublicacion año="1969"/>
```

- /@*: selecciona todos los atributos del nodo.
//@*: selecciona todos los atributos de los descendientes del nodo.

//@*

```
año="1973"
fechaNacimiento="28/03/1936"
año="1973"
fechaNacimiento="28/03/1936"
año="1969"
```

//libro/@*

No devuelve nada porque <libro> no tiene atributos.

//autor/@*

```
fechaNacimiento="28/03/1936"
fechaNacimiento="28/03/1936"
```

Nota: En XPath 1.0 no se puede seleccionar únicamente el valor del atributo, sino que se obtienen respuestas del tipo nombreDelAtributo=ValorDelAtributo

Pasos de búsqueda consecutivos

Una expresión XPath puede contener varios pasos de búsqueda consecutivos. Cada uno incluirá su eje (y en su caso, su predicado) y el último paso de búsqueda incluirá en su caso una selección de nodos. Cada paso de búsqueda trabaja a partir de los nodos seleccionados por el paso de búsqueda anterior.

En el ejemplo siguiente se obtienen los títulos de los libros publicados después de 1970, mediante dos pasos de búsqueda:

- en el primer paso (`//fechaPublicacion[@año>1970]`) se seleccionan los elementos `<fechaPublicacion>` cuyo atributo año es superior a 1970.
- en el segundo paso (`/../titulo`), se seleccionan primero los elementos padre (`/..`) de los `<fechaPublicacion>` seleccionados en el primer paso de búsqueda (es decir, elementos `<libro>`) y a continuación sus subelementos `<titulo>`.

```
//fechaPublicacion[@año>1970]/../titulo
```

```
<titulo>La vida está en otra parte</titulo>  
<titulo>Pantaleón y las visitadoras</titulo>
```

Un determinado resultado se puede obtener mediante un sólo paso de búsqueda o mediante varios pasos.

- En los ejemplos siguientes se obtienen los libros escritos por Mario Vargas Llosa de dos formas distintas:
 - mediante un sólo paso de búsqueda. Se seleccionan los elementos `<libro>` cuyo subelemento `<autor>` tiene como contenido la cadena "Mario Vargas Llosa".

```
//libro[autor="Mario Vargas Llosa"]
```

```
<libro>  
  <titulo>Pantaleón y las visitadoras</titulo>  
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>  
  <fechaPublicacion año="1973"/>  
</libro>  
<libro>  
  <titulo>Conversación en la catedral</titulo>  
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>  
  <fechaPublicacion año="1969"/>  
</libro>
```

- mediante dos pasos de búsqueda. En el primer paso se seleccionan los elementos <autor> cuyo contenido es la cadena "Mario Vargas Llosa". En el segundo paso de búsqueda se seleccionan los elementos padre (es decir, los elementos <libro>).

```
//autor[.="Mario Vargas Llosa"]/..
```

```
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
```

- En los ejemplos siguientes se obtiene el autor que ha publicado libros en 1969 de varias formas distintas:

```
//@año[.=1969]/../autor
```

```
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

```
//libro[fechaPublicacion/@año=1969]/autor
```

```
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

```
//fechaPublicacion[@año=1969]/../autor
```

```
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

```
//autor[../fechaPublicacion/@año=1969]
```

```
<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
```

Expresiones anidadas

Las expresiones XPath pueden anidarse, lo que permite definir expresiones más complicadas. Por ejemplo, en el documento utilizado anteriormente:

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
</biblioteca>
```

```
</libro>
<libro>
  <titulo>Pantaleón y las visitadoras</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1973"/>
</libro>
<libro>
  <titulo>Conversación en la catedral</titulo>
  <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
  <fechaPublicacion año="1969"/>
</libro>
</biblioteca>
```

Un ejemplo de expresión anidada sería, por ejemplo, obtener los títulos de los libros publicados el mismo año que la novela "La vida está en otra parte". Esta información no está directamente almacenada en el documento, pero se puede obtener la respuesta en dos pasos:

- obtener primero el año en que se publicó la novela "La vida está en otra parte":

```
//libro[titulo="La vida está en otra parte"]/fechaPublicacion/@año
año="1973"
```

- y obtener después los títulos de los libros publicados en 1973:

```
//libro[fechaPublicacion/@año=1973]/titulo
<titulo>La vida está en otra parte</titulo>
<titulo>Pantaleón y las visitadoras</titulo>
```

Estas dos expresiones se pueden unir en una única expresión, sustituyendo en la segunda expresión el valor 1973 por la primera expresión:

```
//libro[fechaPublicacion/@año=//libro[titulo="La vida está en otra
parte"]/fechaPublicacion/@año]/titulo
<titulo>La vida está en otra parte</titulo>
<titulo>Pantaleón y las visitadoras</titulo>
```

Como cada una de las expresiones puede escribirse de varias maneras, en realidad hay muchas formas de encontrar la respuesta. Por ejemplo, en la solución siguiente los predicados se encuentran al final del eje en cada subexpresión:

```
//titulo[../fechaPublicacion/@año=//@año[../titulo="La vida está en otra parte"]]
<titulo>La vida está en otra parte</titulo>
<titulo>Pantaleón y las visitadoras</titulo>
```


Otro ejemplo de expresión anidada sería obtener los títulos de los libros del mismo autor que la novela "Pantaleón y las visitadoras". Como en el ejemplo anterior, la respuesta puede obtenerse en dos pasos:

- obtener primero el autor de la novela "Pantaleón y las visitadoras":
`//libro[titulo="Pantaleón y las visitadoras"]/autor/text()`
`Mario Vargas Llosa`
- y obtener después los títulos de los libros escritos por Mario Vargas Llosa:
`//libro[autor="Mario Vargas Llosa"]/titulo`
`<titulo>Pantaleón y las visitadoras</titulo>`
`<titulo>Conversación en la catedral</titulo>`

Estas dos expresiones se pueden unir en una única expresión, sustituyendo en la segunda expresión el valor "Mario Vargas Llosa" por la primera expresión:

```
//libro[autor=//libro[titulo="Pantaleón y las visitadoras"]/autor/text()]/titulo  
<titulo>Pantaleón y las visitadoras</titulo>  
<titulo>Conversación en la catedral</titulo>
```

Un detalle importante es que no hay que escribir la primera expresión entre comillas.

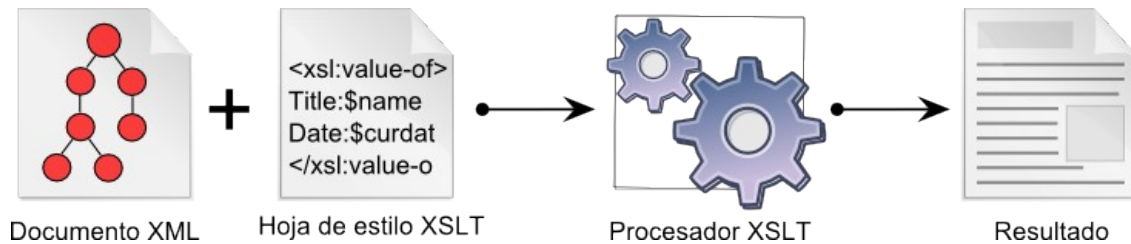
Incluso se puede omitir la selección de nodos `/text()` de la segunda expresión y escribir la expresión XPath así:

```
//libro[autor=//libro[titulo="Pantaleón y las visitadoras"]/autor]/titulo  
<titulo>Pantaleón y las visitadoras</titulo>  
<titulo>Conversación en la catedral</titulo>
```

5.- XSLT: Transformaciones XSL

5.1.-El lenguaje de programación XSLT

XSLT (Transformaciones XSL) es un lenguaje de programación declarativo que permite generar documentos a partir de documentos XML, como ilustra la imagen siguiente:



- El documento XML es el documento inicial a partir del cual se va a generar el resultado.
- **La hoja de estilo XSLT es el documento que contiene el código fuente del programa, es decir, las reglas de transformación que se van a aplicar al documento inicial.**
- El procesador XSLT es el programa de ordenador que aplica al documento inicial las reglas de transformación incluidas en la hoja de estilo XSLT y genera el documento final.
- El resultado de la ejecución del programa es un nuevo documento (que puede ser un documento XML o no).

XSLT se utiliza para obtener a partir de un documento XML otros documentos (XML o no). A un documento XML se le pueden aplicar distintas hojas de estilo XSLT para obtener distintos resultados y una misma hoja de estilo XSLT se puede aplicar a distintos documentos XML.

El lenguaje XSLT está normalizado por el W3C, que ha publicado tres versiones de este lenguaje:

- noviembre de 1999: [XSLT 1.0](#)
- enero de 2007: [XSLT 2.0](#)
- junio de 2017: [XSLT 3.0](#)

s entre estas versiones, lo que se cuenta en esta lección es válido para todas ellas.

Aunque hay incompatibilidades entre estas versiones, lo que se cuenta en este tema es válido para todas ellas.

5.2.-Hojas de estilo XSLT

XSLT es un lenguaje declarativo. Por ello, las hojas de estilo XSLT no se escriben como una secuencia de instrucciones, sino como una **colección de plantillas (template rules)**. **Cada plantilla**

establece cómo se transforma un determinado elemento (definido mediante expresiones XPath). La transformación del documento se realiza de la siguiente manera:

- El procesador analiza el documento y construye el árbol del documento.
- El procesador recorre el árbol del documento desde el nodo raíz.
- En cada nodo recorrido, el procesador aplica o no alguna plantilla:
 - **Si a un nodo no se le puede aplicar ninguna plantilla, su contenido se incluye en el documento final** (el texto del nodo, no el de los nodos descendientes). A continuación, el procesador recorre sus nodos hijos.
 - **Si a un nodo se le puede aplicar una plantilla, se aplica la plantilla. La plantilla puede generar texto que se incluye en el documento final. En principio, el procesador no recorre sus nodos hijos, salvo que la plantilla indique al procesador que sí que deben recorrerse los nodos hijos.**
- Cuando el procesador ha recorrido el árbol, se ha terminado la transformación.

Una hoja de estilo XSLT es un documento XML que contiene al menos las etiquetas siguientes:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
</xsl:stylesheet>
```

Estas etiquetas son:

- la declaración xml `<?xml>`, propia de cualquier documento XML.
- la instrucción `<xsl:stylesheet>` es la etiqueta raíz de la hoja de estilo, sus atributos indican la versión y el espacio de nombres correspondiente.

Dentro de la instrucción `<xsl:stylesheet>` se pueden encontrar los llamados *elementos de alto nivel y las plantillas*, como en el ejemplo siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

  <xsl:output method="xml" indent="yes"/>

  <xsl:template match="/">
  </xsl:template>

</xsl:stylesheet>
```

Estas etiquetas son

- el elemento de alto nivel `<xsl:output>` indica el tipo de salida producida.
- la instrucción `<xsl:template>` es una plantilla.
 - El atributo `match` indica los elementos afectados por la plantilla y contiene una expresión XPath.

- El contenido de la instrucción define la transformación a aplicar (si la instrucción no contiene nada, como en el ejemplo anterior, sustituirá el nodo por nada, es decir, eliminará el nodo, aunque conservará el texto contenido en el elemento).

Cuando se aplica una plantilla a un nodo, en principio no se recorren los nodos descendientes. Para indicar que sí queremos recorrer los nodos descendientes y aplicarles las plantillas que les correspondan, hay que utilizar la instrucción `<xsl:apply-templates />`, como en el ejemplo siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

  <xsl:template match="/">
    <xsl:apply-templates />
  </xsl:template>

  <xsl:template match="elemento">
  </xsl:template>

</xsl:stylesheet>
```

5.3.-Enlazar documentos XML con hojas de estilo XSLT

Se puede asociar de forma permanente una hoja de estilo XSLT a un documento XML mediante la instrucción de procesamiento `<?xml-stylesheet ?>`, la misma que permite asociar hojas de estilo CSS. La instrucción de procesamiento `<?xml-stylesheet ... ?>` va al principio del documento, después de la declaración XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ejemplo.xsl"?>
```

Cuando se visualiza en un navegador web un documento XML enlazado con una hoja de estilo XSLT, los navegadores muestran el resultado de la transformación, aunque si se muestra el código fuente de la página, los navegadores muestran el documento XML original.

Nota: Google Chrome no muestra los documentos XML que enlazan a hojas de estilo XSLT abiertos como archivos locales (file:///...).Firefox y Microsoft Edge sí lo hacen.

5.4.-Abrir documentos XML con hojas de estilo XSLT en el navegador

Al abrir en un navegador una página XML enlazada con una hoja de estilo XSLT, el navegador muestra el resultado de la transformación. Pero no muestra el código fuente obtenido como resultado, sino interpretando ese código fuente, como cuando se enlaza una hoja de estilo CSS vacía.

Realmente, tanto en los ejemplos como en los ejercicios no es necesario abrir los archivos en el navegador. Basta con comprobar que se obtiene el resultado deseado en XML Copy Editor (u otro editor que estemos utilizando).

5.5.-Ejemplos de plantillas XSLT

Vamos a ver ejemplos de plantillas trabajando sobre el documento siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Conversación en la catedral</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1969"/>
  </libro>
</biblioteca>
```

Si los ejemplos de esta página se abren en el navegador, el resultado no coincide en casi todos los casos con el que se muestra en esta página ya que los navegadores no respetan los saltos de línea ni los espacios en blanco, ni muestran las etiquetas. Los resultados que se muestran en esta página son los que se obtienen con XML Copy Editor.

Plantillas vacías o no existentes

- Si no hay plantillas, el procesador simplemente recorre todos los nodos y extrae el texto contenido por cada nodo.

En el ejemplo siguiente, el resultado incluye el contenido de los nodos <título> y <autor> puesto que no hay ninguna plantilla.

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

</xsl:stylesheet>
```

Resultado(ejemplo 1)

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

La vida está en otra parte
Milan Kundera

Pantaleón y las visitadoras
Mario Vargas Llosa

Conversación en la catedral
Mario Vargas Llosa
```

- Si hay una plantilla vacía, el procesador no genera ningún resultado en el documento final ni recorre los nodos hijos.

En el ejemplo siguiente, el resultado incluye el contenido de los nodos <titulo>, ya que no hay regla para ellos, pero los de <autor> se pierden porque la plantilla es vacía.

XSLT

Resultado(ejemplo2)

```
<?xml version="1.0" encoding="UTF-8"?> <?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Tr
ansform" version="1.0">
    <xsl:template match="autor">
    </xsl:template>
</xsl:stylesheet>
```

La vida está en otra parte
Pantaleón y las visitadoras
Conversación en la catedral

- En el caso más extremo, si la plantilla vacía se aplica al nodo raíz, el procesador no genera ningún resultado en el documento final ni recorre ningún nodo hijo.

XSLT

Resultado(ejemplo3)

```
<?xml version="1.0" encoding="UTF-8"?> <?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Tr
ansform" version="1.0">
    <xsl:template match="/">
    </xsl:template>
</xsl:stylesheet>
```

La instrucción <xsl:value-of>

La instrucción <xsl:value-of> extrae el contenido del nodo seleccionado.

- En el ejemplo siguiente, el documento final contiene los autores de los libros porque la plantilla los genera con la instrucción <xsl:value-of>. Como se ha aplicado una plantilla al nodo <libro>, sus hijos (<titulo>, <autor> y <fechaPublicacion>) no se recorren. Por eso, los títulos de los libros no aparecen en el documento final.

XSLT

Resultado (ejemplo4)

```
<?xml version="1.0" encoding="UTF-8"?> <?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Tr
ansform" version="1.0">
    <xsl:template match="libro">
        <xsl:value-of select="autor"/>
    </xsl:template>
</xsl:stylesheet>
```

Milan Kundera
Mario Vargas Llosa
Mario Vargas Llosa

- En el ejemplo siguiente, se obtienen el título y el autor de los libros, pero uno a continuación de otro. Los saltos de línea se crean tras cada aplicación de la regla (es decir, a cada libro), pero no en el interior de la regla.

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Tr
ansform" version="1.0">

  <xsl:template match="libro">
    <xsl:value-of select="titulo"/>
    <xsl:value-of select="autor"/>
  </xsl:template>

</xsl:stylesheet>
```

Resultado(ejemplo5)

```
<?xml version="1.0" encoding="UTF-8"?>
La vida está en otra parteMilan
Kundera
Pantaleón y las visitadorasMario
Vargas Llosa
Conversación en la catedralMario
Vargas Llosa
```

- En el ejemplo siguiente, los autores se obtienen gracias a la regla que extrae el contenido del nodo (el carácter punto "." hace referencia al propio elemento) y **los títulos se obtienen porque al no haber reglas para ese nodo se extrae el contenido**. El resultado es el mismo que el de un ejemplo anterior, pero por motivos distintos.

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Tr
ansform" version="1.0">

  <xsl:template match="autor">
    <xsl:value-of select="."/>
  </xsl:template>

</xsl:stylesheet>
```

Resultado(ejemplo6)

```
<?xml version="1.0" encoding="UTF-8"?>
La vida está en otra parte
Milan Kundera

Pantaleón y las visitadoras
Mario Vargas Llosa

Conversación en la catedral
Mario Vargas Llosa
```

También se pueden extraer los valores de los atributos, utilizando @.

- En el ejemplo siguiente, las fechas de publicación se obtienen gracias a la regla que extraen el valor del atributo y los títulos y autores se obtienen porque al no haber reglas para ese nodo se extrae el contenido.

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="fechaPublicacion">
    <xsl:value-of select="@año"/>
  </xsl:template>
</xsl:stylesheet>
```

Resultado(ejemplo7)

```
<?xml version="1.0" encoding="UTF-8"?>
La vida está en otra parte
Milan Kundera
1973

Pantaleón y las visitadoras
Mario Vargas Llosa
1973

Conversación en la catedral
Mario Vargas Llosa
1969
```

Generar texto adicional

Se puede generar texto escribiendolo en la regla, por ejemplo, código html.

- En el ejemplo siguiente se obtienen los nombres de los autores porque la regla selecciona el nodo `<libro>`, pero además generamos las etiquetas `<p>`. El resultado sigue sin verse bien en el navegador, porque aunque hay etiquetas `<p>`, falta la etiqueta global `<html>`.

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="libro">
    <p><xsl:value-of select="autor"/></p>
  </xsl:template>
</xsl:stylesheet>
```

Resultado(ejemplo8)

```
<?xml version="1.0" encoding="UTF-8"?>
<p>Milan Kundera</p>
<p>Mario Vargas Llosa</p>
<p>Mario Vargas Llosa</p>
```

Dentro de la regla podemos hacer referencia a varios subnodos.

- En el ejemplo siguiente se obtienen los nombres de los autores y los títulos de los libros porque la regla selecciona el nodo `<libro>`, pero además generamos las etiquetas `<p>`. El resultado sigue sin verse bien en el navegador, porque aunque hay etiquetas `<p>`, falta la etiqueta global `<html>`.

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Trans
form" version="1.0">

  <xsl:template match="libro">
    <p><xsl:value-of select="autor"/></p>
    <p><xsl:value-of select="titulo"/></p>
  </xsl:template>

</xsl:stylesheet>
```

Resultado(ejemplo9)

```
<?xml version="1.0" encoding="UTF-8"?>
<p>Milan Kundera</p>
<p>La vida está en otra parte</p>
<p>Mario Vargas Llosa</p>
<p>Pantaleón y las visitadoras</p>
<p>Mario Vargas Llosa</p>
<p>Conversación en la catedral</p>
```

- Los siguientes ejemplos intentan conseguir el mismo resultado que el ejemplo anterior, pero utilizando dos reglas, y no lo consiguen:

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Trans
form" version="1.0">

  <xsl:template match="libro">
    <p><xsl:value-of select="autor"/></p>
  </xsl:template>

  <xsl:template match="libro">
    <p><xsl:value-of select="titulo"/></p>
  </xsl:template>

</xsl:stylesheet>
```

Resultado(ejemplo10)

```
<?xml version="1.0" encoding="UTF-8"?>
<p>La vida está en otra parte</p>
<p>Pantaleón y las visitadoras</p>
<p>Conversación en la catedral</p>
```

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

  <xsl:template match="libro">
    <p><xsl:value-of select="titulo"/></p>
  </xsl:template>

  <xsl:template match="libro">
    <p><xsl:value-of select="autor"/></p>
  </xsl:template>
</xsl:stylesheet>
```

Resultado(ejemplo11)

```
<?xml version="1.0" encoding="UTF-8"?>
<p>Milan Kundera</p>
<p>Mario Vargas Llosa</p>
<p>Mario Vargas Llosa</p>
```

¿Por qué en un caso se obtienen sólo los títulos y en el otro sólo los autores? Porque el procesador XSLT sólo aplica una regla a cada nodo. Si tenemos dos reglas para el mismo nodo, el procesador sólo aplica una de ellas (la última, en este caso).

Además de generar etiquetas, se puede generar texto.

- En el ejemplo siguiente se generan frases a partir del contenido de los nodos.

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

  <xsl:template match="libro">
    <p><xsl:value-of select="autor"/> escribió
    <xsl:value-of select="titulo"/></p>
  </xsl:template>
</xsl:stylesheet>
```

Resultado(ejemplo12)

```
<?xml version="1.0" encoding="UTF-8"?>
<p>Milan Kundera escribió "La vida
está en otra parte"</p>
<p>Mario Vargas Llosa escribió
"Pantaleón y las visitadoras"</p>
<p>Mario Vargas Llosa escribió
"Conversación en la catedral"</p>
```

Aplicar reglas a subnodos: la instrucción `<xsl:apply-templates>`

La instrucción `<xsl:apply-templates>` hace que se apliquen a los subelementos las reglas que les sean aplicables.

- En el ejemplo siguiente, se genera la etiqueta `<html>` además de unos párrafos con los nombres de los autores. Este ejemplo sí que se puede ver en el navegador ya que se interpreta como html.

XSLT

```
<?xml version="1.0" encoding="UTF-8"?><xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
```

```
<xsl:template match="/">
  <html>
    <h1>Autores</h1>
    <xsl:apply-templates />
  </html>
</xsl:template>
<xsl:template match="libro">
  <p><xsl:value-of select="autor"/></p>
</xsl:template>
</xsl:stylesheet>
```

Resultado(ejemplo13)

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

<html><h1>Autores</h1>
  <p>Milan Kundera</p>
  <p>Mario Vargas Llosa</p>
  <p>Mario Vargas Llosa</p>
</html>
```

La primera regla sustituye el elemento raíz (y todos sus subelementos) por las etiquetas `<html>` y `<h1>`, pero además aplica a los subelementos las reglas que les son aplicables. En este caso, sólo hay una regla para los elementos `<libro>` que generan los párrafos.

Salto de línea y espacios en blanco: las instrucciones `<xsl:text>` y `<xsl:strip-space>`

Al transformar un documento, los procesadores XSLT incorporan saltos de línea y espacios en blanco en el resultado, pero no lo hacen de forma uniforme. Por ejemplo, XML Copy Editor y Notepad++ (con el plug-in XML Tols) producen diferentes resultados.

No parece haber una solución sencilla que funcione en todos los procesadores, pero sí soluciones que funcionen en cada uno de ellos.

La instrucción `<xsl:strip-space>`

En el caso de XML Copy Editor, la forma más sencilla de mejorar el formato de presentación de los resultados, eliminando líneas en blanco innecesarias y sangrando los elementos anidados, es utilizar la instrucción `<xsl:strip-space>`. Pero debe tenerse en cuenta que esta instrucción no produce el mismo resultado en otros procesadores XSLT (como en Notepad++ con XML Tools).

La instrucción `<xsl:strip-space>` permite indicar si los elementos que contienen únicamente espacios en blanco se incluyen en la transformación o no.

- En el ejemplo anterior (del apartado sobre la instrucción `<xsl:apply-templates>`) la etiqueta `<h1>` se generaba en la misma línea que la etiqueta `<html>`, pero en el ejemplo siguiente se generan en líneas distintas (y las etiquetas se muestran sangradas) al utilizar la instrucción `<xsl:strip-space>`:

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:strip-space elements="*" />

  <xsl:template match="/">
    <html>
      <h1>Autores</h1>
      <xsl:apply-templates />
    </html>
  </xsl:template>

  <xsl:template match="libro">
    <p><xsl:value-of select="autor"/></p>
  </xsl:template>
</xsl:stylesheet>
```

Resultado (ejemplo14)

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
  <h1>Autores</h1>
  <p>Milan Kundera</p>
  <p>Mario Vargas Llosa</p>
  <p>Mario Vargas Llosa</p>
</html>
```

La instrucción <xsl:text>

En el caso de Notepad++ con XML Tools, se puede mejorar el formato de presentación de los resultados, insertando líneas en blanco innecesarias y sangrando los elementos anidados, utilizando la instrucción <xsl:text>. Pero debe tenerse en cuenta que esta instrucción no permite eliminar líneas en blanco que se producen en otros procesadores (como en XML Copy Editor).

La instrucción <xsl:text> permite generar texto que no se puede generar simplemente añadiéndolo (saltos de líneas y espacios en blanco, por ejemplo).

- En el ejemplo anterior (del apartado sobre la instrucción <xsl:apply-templates>) la etiqueta <h1> se generaba en la misma línea que la etiqueta <html>, pero en el ejemplo siguiente se generan en líneas distintas al añadir un salto de línea con la entidad de carácter
 (y un par de espacios para alinear la etiqueta <h1> con las etiquetas <p>):

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

  <xsl:template match="/">
    <html>
```

```
<xsl:text>&#10; </xsl:text>
<h1>Autores</h1>
<xsl:apply-templates />
</html>
</xsl:template>

<xsl:template match="libro">
  <p><xsl:value-of select="autor"/></p>
</xsl:template>

</xsl:stylesheet>
```

Resultado(ejemplo15)

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
  <h1>Autores</h1>
  <p>Milan Kundera</p>
  <p>Mario Vargas Llosa</p>
  <p>Mario Vargas Llosa</p>
</html>
```

La instrucción <xsl:attribute>

La instrucción <xsl:attribute> permite generar un atributo y su valor. Se utiliza cuando el valor del atributo se obtiene a su vez de algún nodo.

Por ejemplo, a partir del siguiente documento XML, se quiere generar la etiqueta . en la que el valor del atributo src sea el contenido de la etiqueta <imagen>.

```
<?xml version="1.0" encoding="UTF-8"?>
<licencias>
  <licencia>
    <nombre>Creative Commons By - Share Alike</nombre>
    <imagen>cc-bysa-88x31.png</imagen>
  </licencia>
</licencias>
```

- No se puede utilizar la instrucción <xsl:value-of> como en el ejemplo incorrecto siguiente:



XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
```

```
<xsl:template match="licencia">
  <p>" /></p>
</xsl:template>

</xsl:stylesheet>
```

Resultado

Error at line 5, column 19: not well-formed (invalid token)

En este caso el problema no es debido a la utilización de comillas dobles (también daría error si se hubieran utilizado comillas simples o entidades), sino que es necesario utilizar la instrucción `<xsl:attribute>`.

- Para generar un atributo en una etiqueta, es necesario utilizar la instrucción `<xsl:attribute>`, como en el ejemplo siguiente:

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">

  <xsl:template match="licencia">
    <p><img>
      <xsl:attribute name="src">
        <xsl:value-of select="imagen" />
      </xsl:attribute>
    </img>
  </p>
</xsl:template>

</xsl:stylesheet>
```

Resultado

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

En la hoja de estilo XSLT, la etiqueta `` se escribe con apertura y cierre, aunque en el resultado aparece como etiqueta monoatómica.

De todas formas, el navegador no mostraría todavía la imagen, puesto que no interpreta la etiqueta `` como la etiqueta de imagen del html.

- Como en ejemplos anteriores, para que la imagen se muestre en el navegador sería necesario generar también la etiqueta `<html>`:

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

  <xsl:template match="/">
    <html>
      <xsl:apply-templates />
    </html>
  </xsl:template>

  <xsl:template match="licencia">
    <p><img>
      <xsl:attribute name="src">
        <xsl:value-of select="imagen" />
      </xsl:attribute>
    </img>
    </p>
  </xsl:template>

</xsl:stylesheet>
```

Resultado

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

6.-XML Copy Editor

6.1.-Conseguir XML Copy Editor

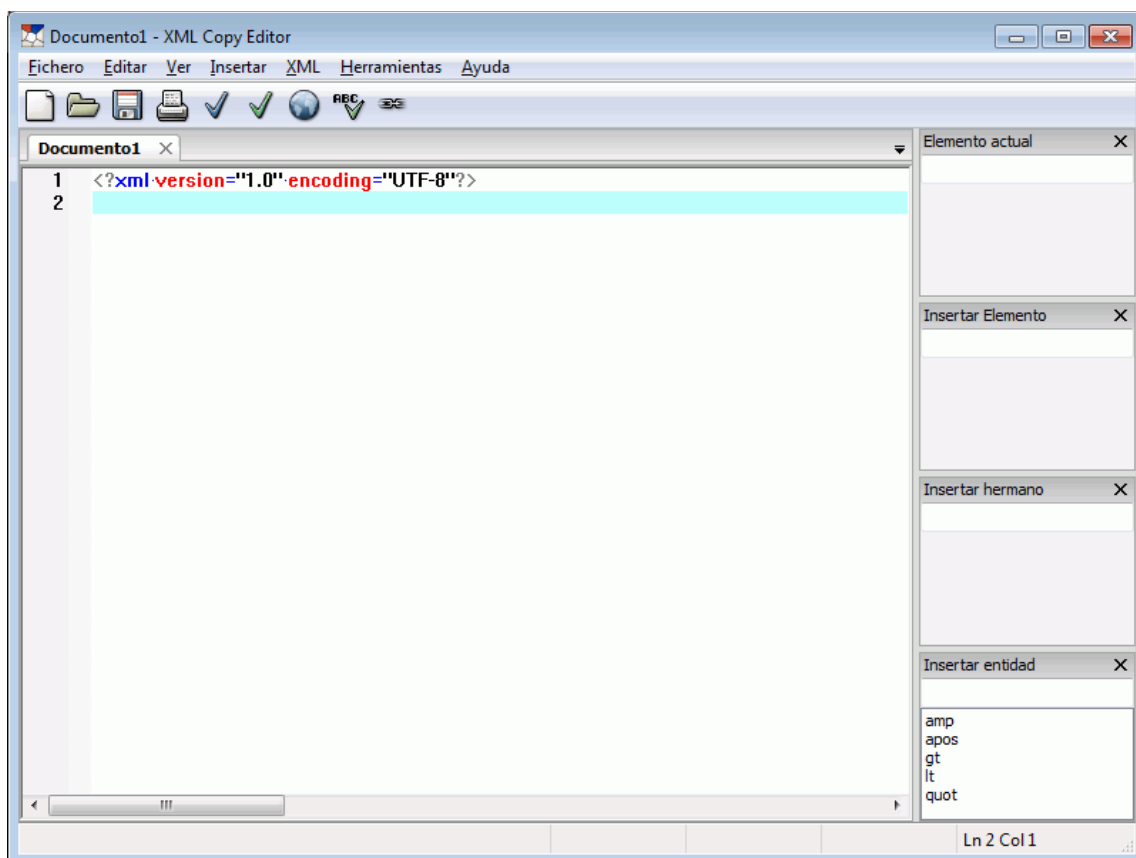
[XML Copy Editor](http://xml-copy-editor.sourceforge.net/) es un editor de documentos XML libre (GPL 2.0) y multiplataforma cuya página web es <http://xml-copy-editor.sourceforge.net/>.

La última versión disponible actualmente (abril de 2017) es la versión XML Copy Editor 1.2.1.3 (del 6 de septiembre de 2014). Enlace de descarga para Windows (32 bits): [XML Copy Editor 1.2.1.3](#) (9,2 MB).

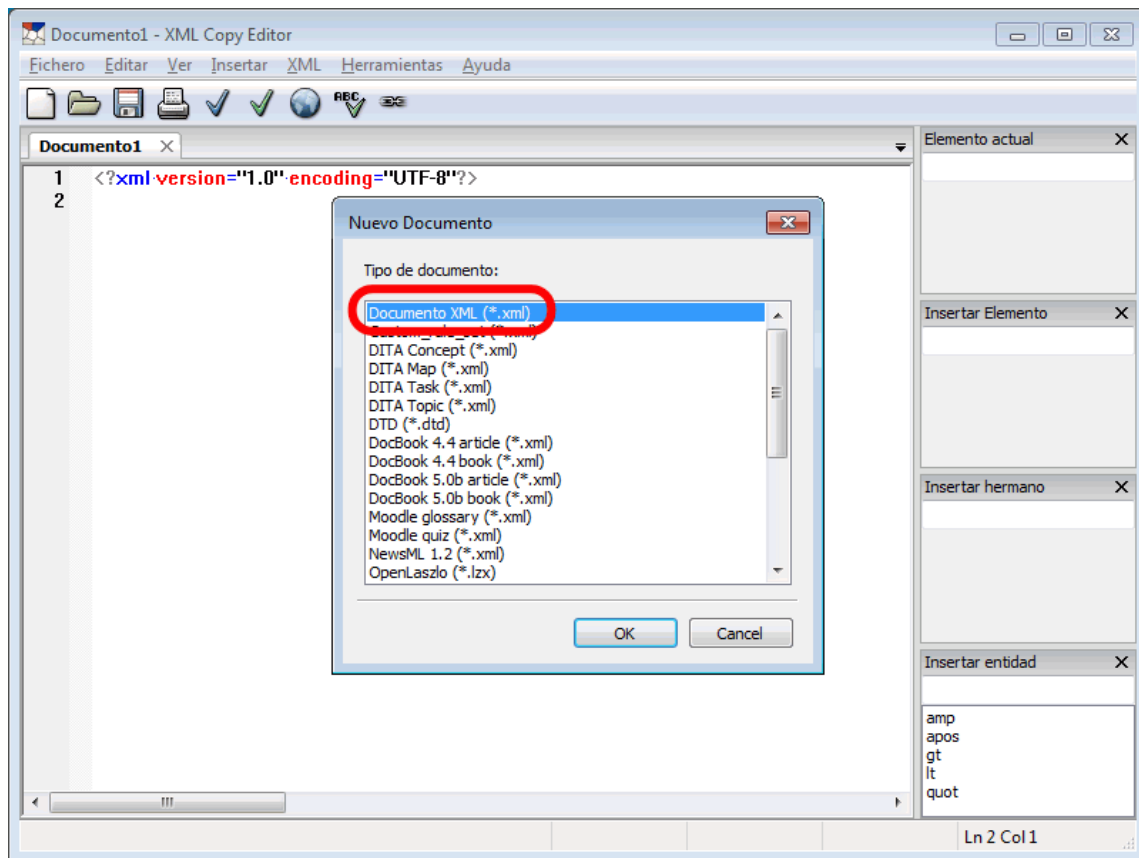
Nota: En caso de que esta versión dé problemas, se puede utilizar la versión XML Copy Editor 1.2.0.7 (del 11 de diciembre de 2009). Enlace de descarga para Windows: [XML Copy Editor 1.2.0.7](#) (6,28 MB).

6.2.-Interface y configuración de XML Copy Editor

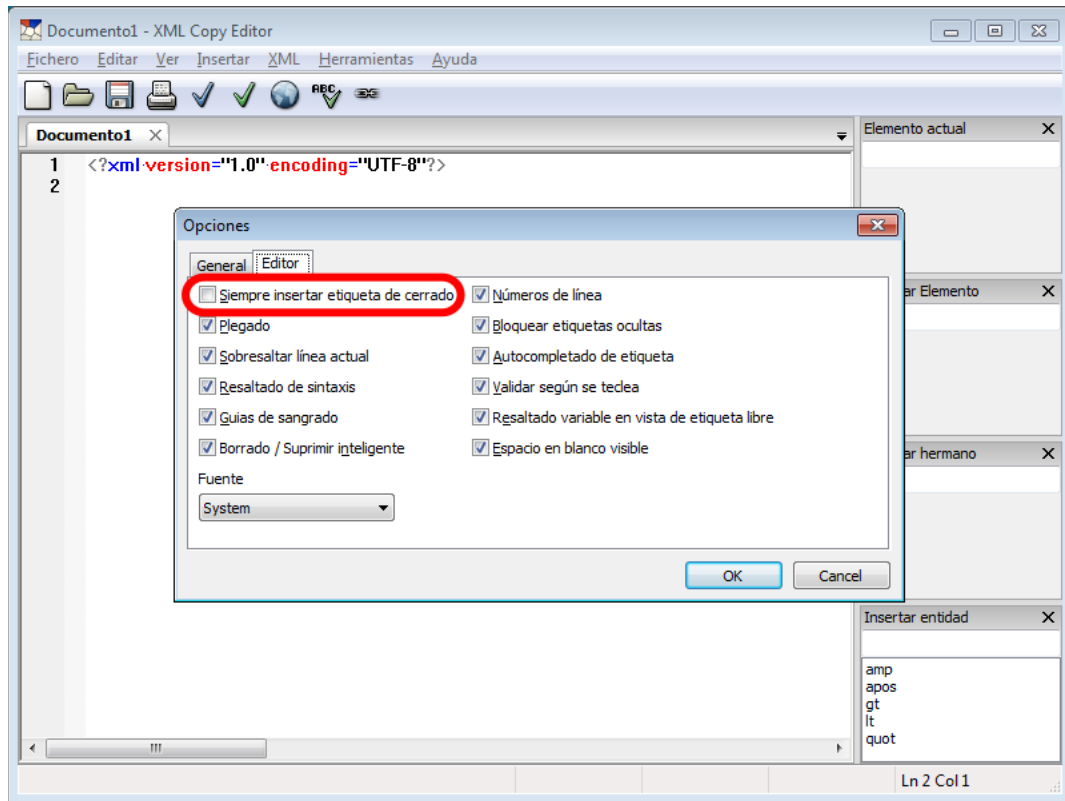
El interface de XML Copy Editor es similar al de cualquier editor de texto sin formato.



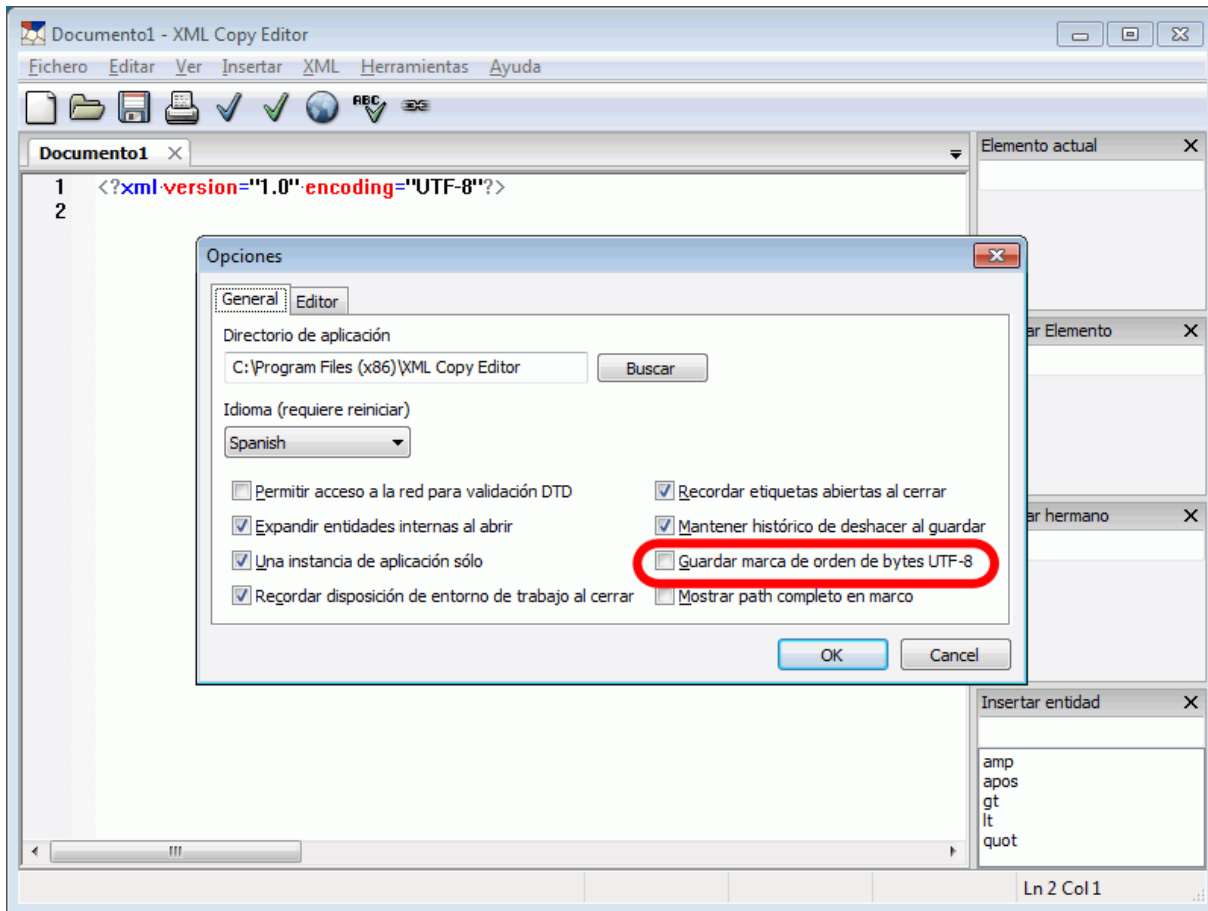
Al abrir XML Copy Editor el editor muestra la plantilla de un documento XML con la declaración XML. Si se quiere crear un documento XML desde cero, se puede elegir el menú Fichero > Nuevo... y elegir la opción Documento XML (*.xml), como muestra la imagen siguiente:



El menú Herramientas > Opciones > Editor permite configurar el comportamiento de XML Copy Editor al editar archivos XML. Personalmente, yo prefiero desmarcar la casilla "Siempre insertar etiqueta de cerrado".



En el menú Herramientas > Opciones > General se puede desactivar la opción "Guardar marca de orden de bytes UTF-8", ya que la marca de orden de bytes ([BOM](#)) no tiene mucho sentido en UTF-8, puesto que en UTF-8 no existe el problema del orden de los bytes en los caracteres que se codifican con varios caracteres.



Nota: En la versión XML Copy Editor 1.2.0.7 y anteriores se podía elegir el navegador predeterminado de XML Copy Editor, independientemente del navegador predeterminado del sistema. Haciendo clic en *Buscar* se debía elegir el ejecutable del navegador.

6.3.-Juego de caracteres

La declaración xml indica el juego de caracteres del documento. El juego de caracteres que se utiliza en este curso es UTF-8:

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

Se pueden utilizar otros juegos de caracteres, como ISO-8859-1 (Europeo occidental):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

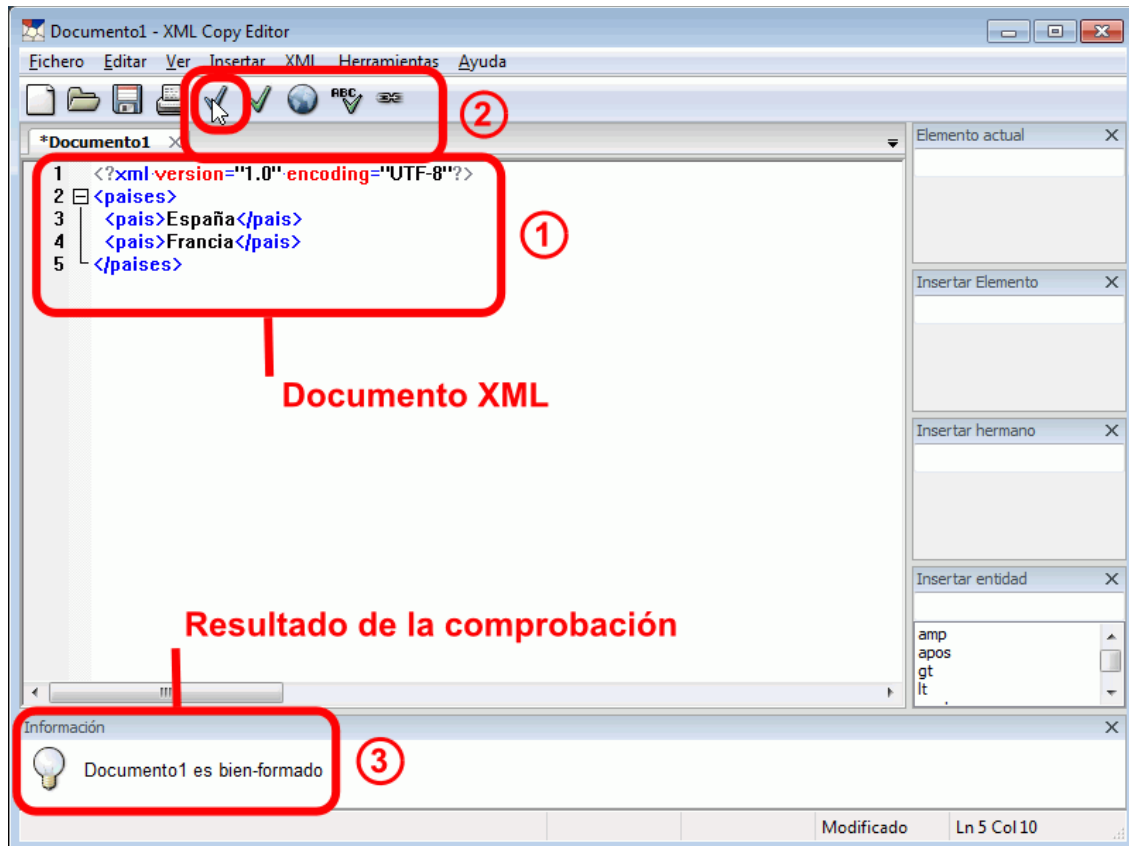
Es importante que el juego de caracteres que aparece en la declaración sea el juego de caracteres en que realmente está guardado el documento, porque si no el procesador XML puede tener problemas leyendo el documento.

XML Copy Editor tiene en cuenta el juego de caracteres indicado en la declaración. Si se modifica la declaración, al guardar el documento se guarda en el juego correspondiente. Pero hay que tener en cuenta que otros editores, como el bloc de notas de Windows, no lo hace.

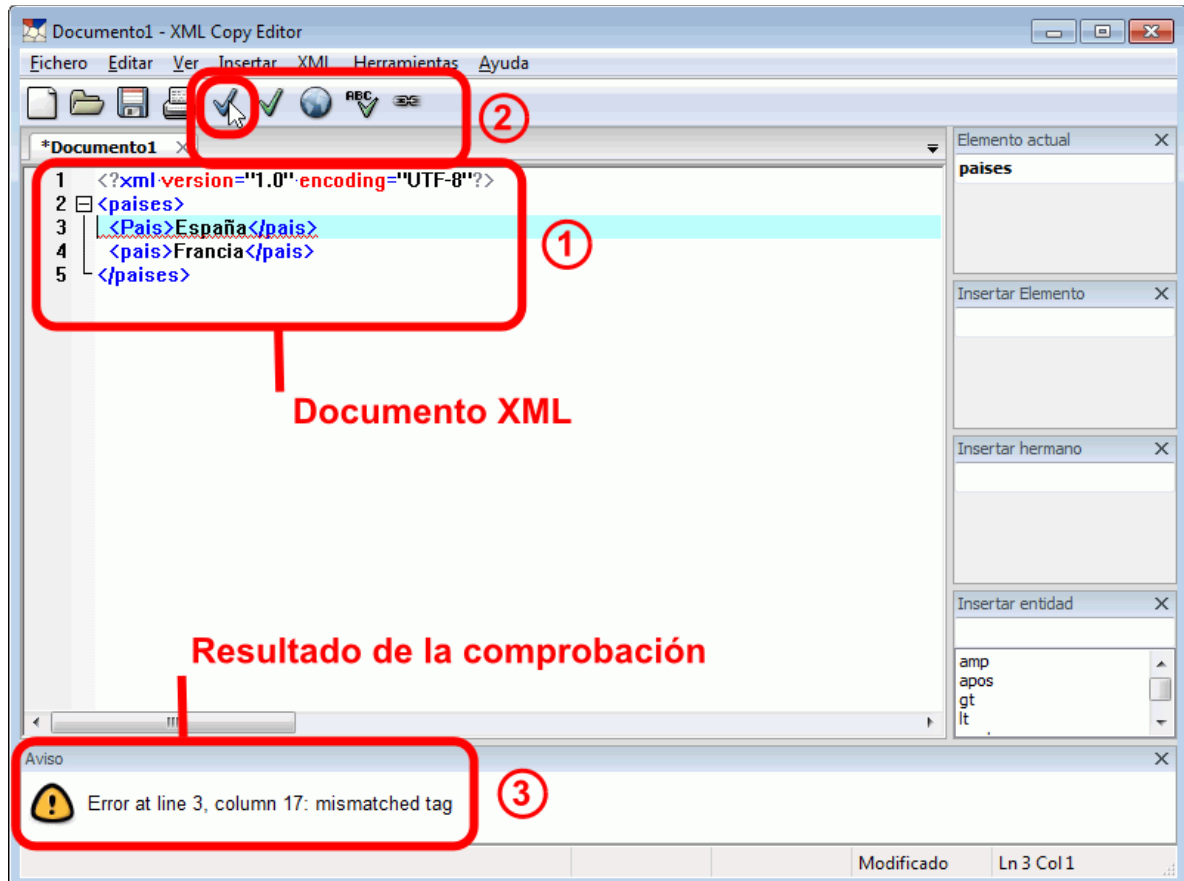
6.4.-Comprobar que un documento está bien formado

Para comprobar si un documento está bien formado, se puede elegir el menú XML > Comprobar Bien-Formado, hacer clic en el botón correspondiente, o pulsar la tecla F2.

- En caso de que el documento esté bien formado, el programa lo indica en la parte inferior de la pantalla, como muestra la imagen siguiente:



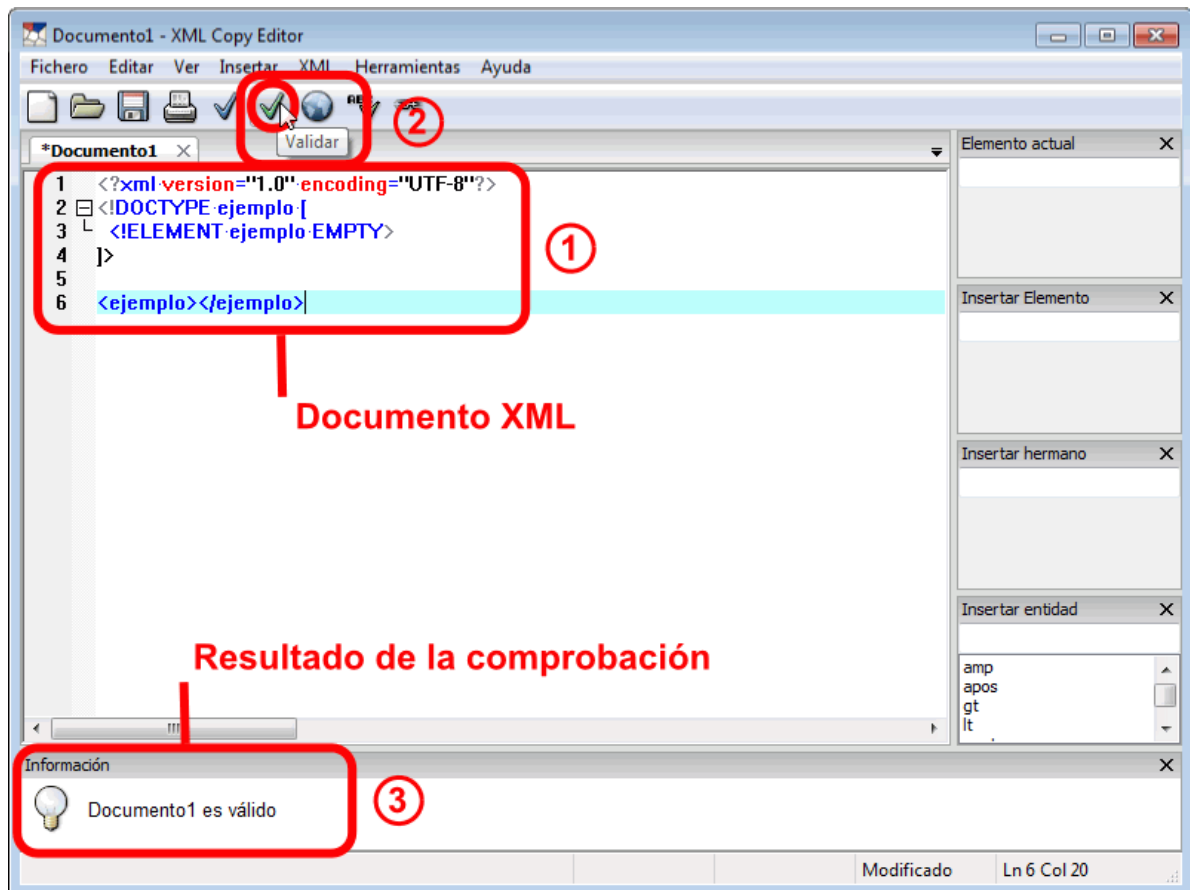
- En caso de que el documento no esté bien formado, el programa lo indica en la parte inferior de la pantalla, como muestra la imagen siguiente. El mensaje inferior explica el tipo de error detectado y la línea en el que se ha detectado (en el ejemplo, las etiquetas de la línea 3 no coinciden). Si el documento contiene varios errores, sólo se muestra uno de ellos, por lo que una vez corregido un error es necesario repetir la comprobación hasta que el documento esté bien formado.



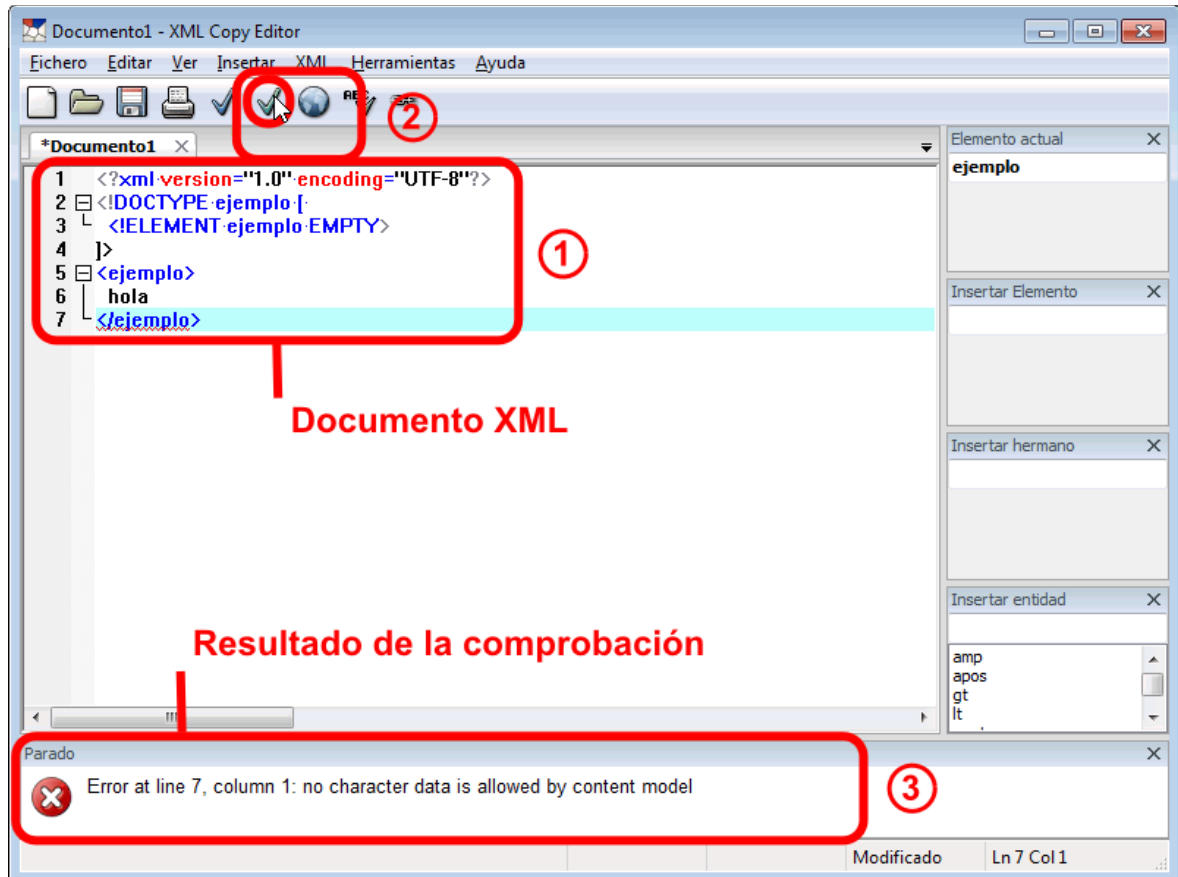
6.5.-Comprobar que un documento es válido

Para comprobar si un documento es válido, se puede elegir el menú XML > Validar > DTD/XML Schema, hacer clic en el botón correspondiente, o pulsar la tecla F5.

- En caso de que el documento sea válido, el programa lo indica en la parte inferior de la pantalla, como muestra la imagen siguiente:

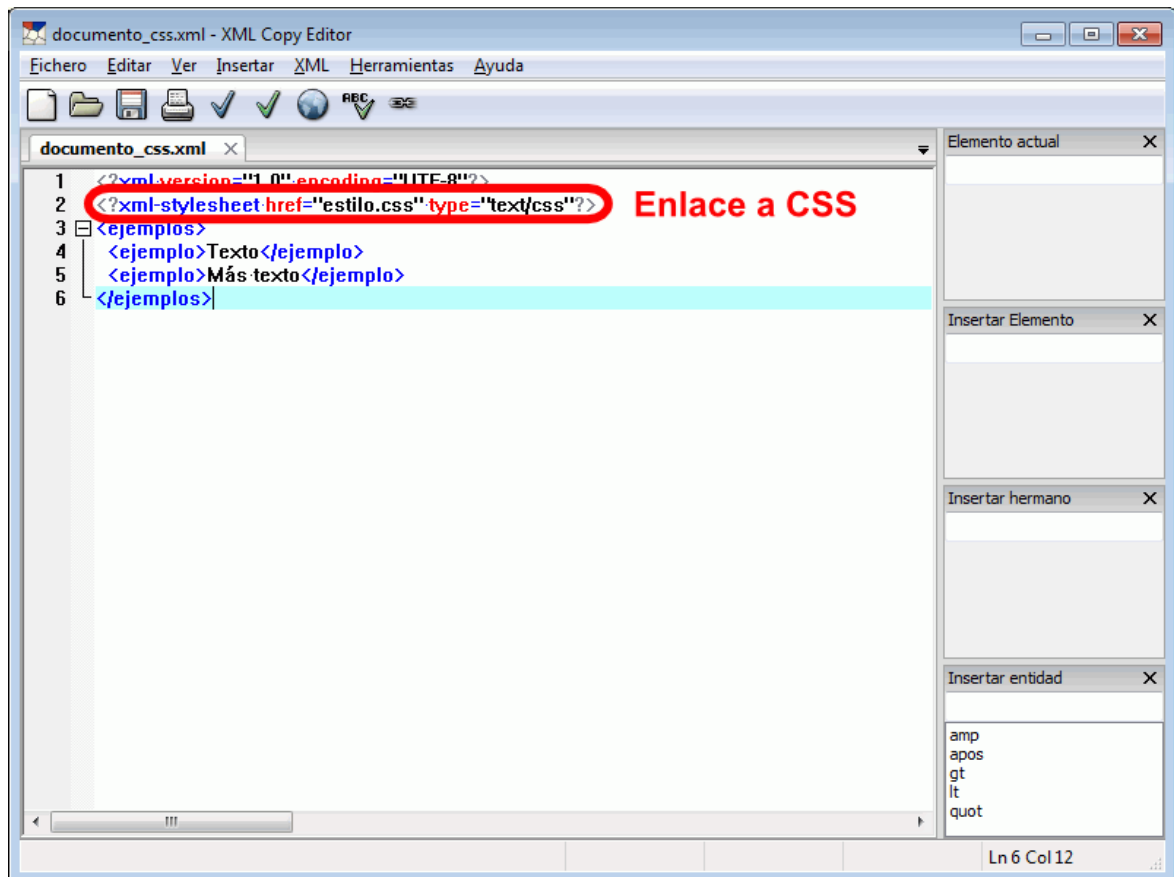


- En caso de que el documento no sea válido, el programa lo indica en la parte inferior de la pantalla, como muestra la imagen siguiente. El mensaje inferior explica el tipo de error detectado y la línea en el que se ha detectado (en el ejemplo, que la etiqueta contiene texto cuando la DTD lo define vacío). Si el documento contiene varios errores, sólo se muestra uno de ellos, por lo que una vez corregido un error es necesario repetir la validación hasta que el documento sea válido.



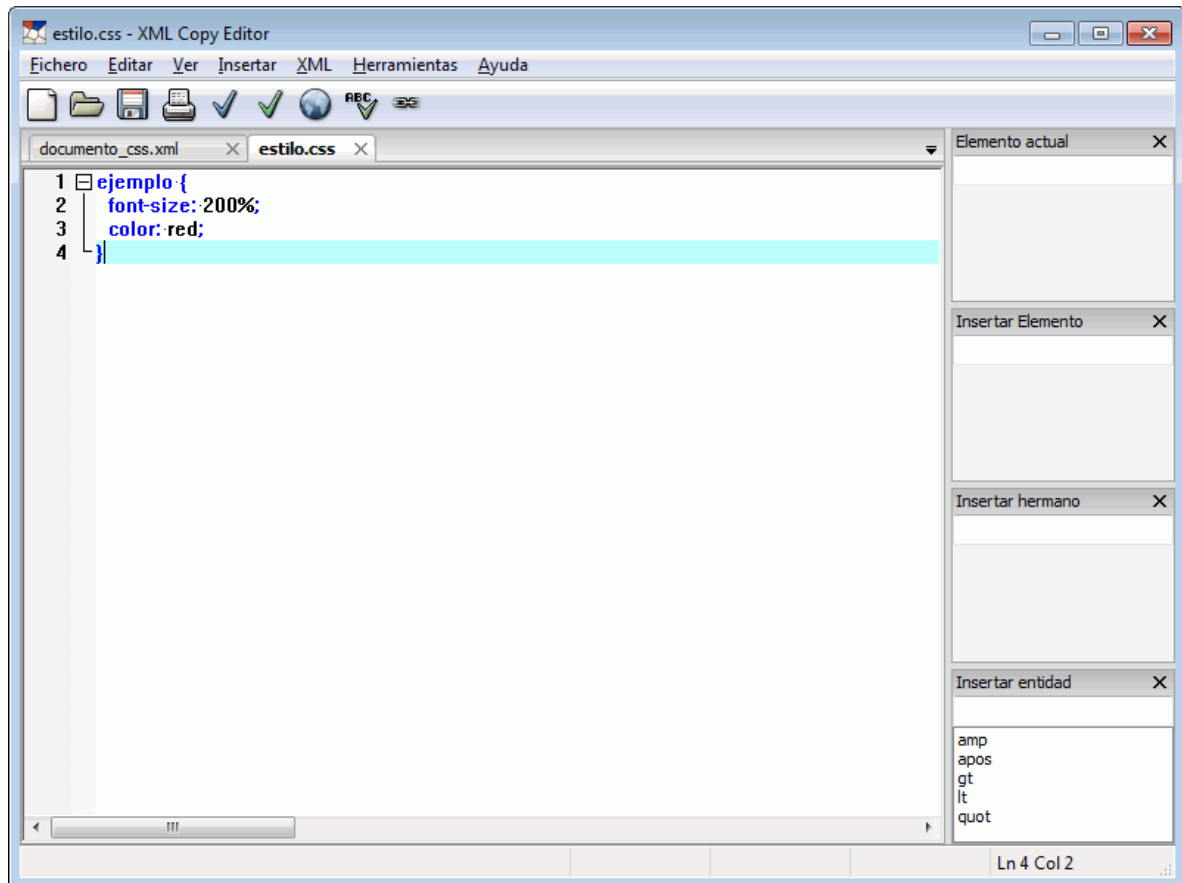
6.6.-Enlazar una hoja de estilo CSS

- XML Copy Editor no incluye opciones de menú relacionadas con el enlace a hoja de estilo CSS, por lo que es necesario escribir manualmente la instrucción de procesamiento.

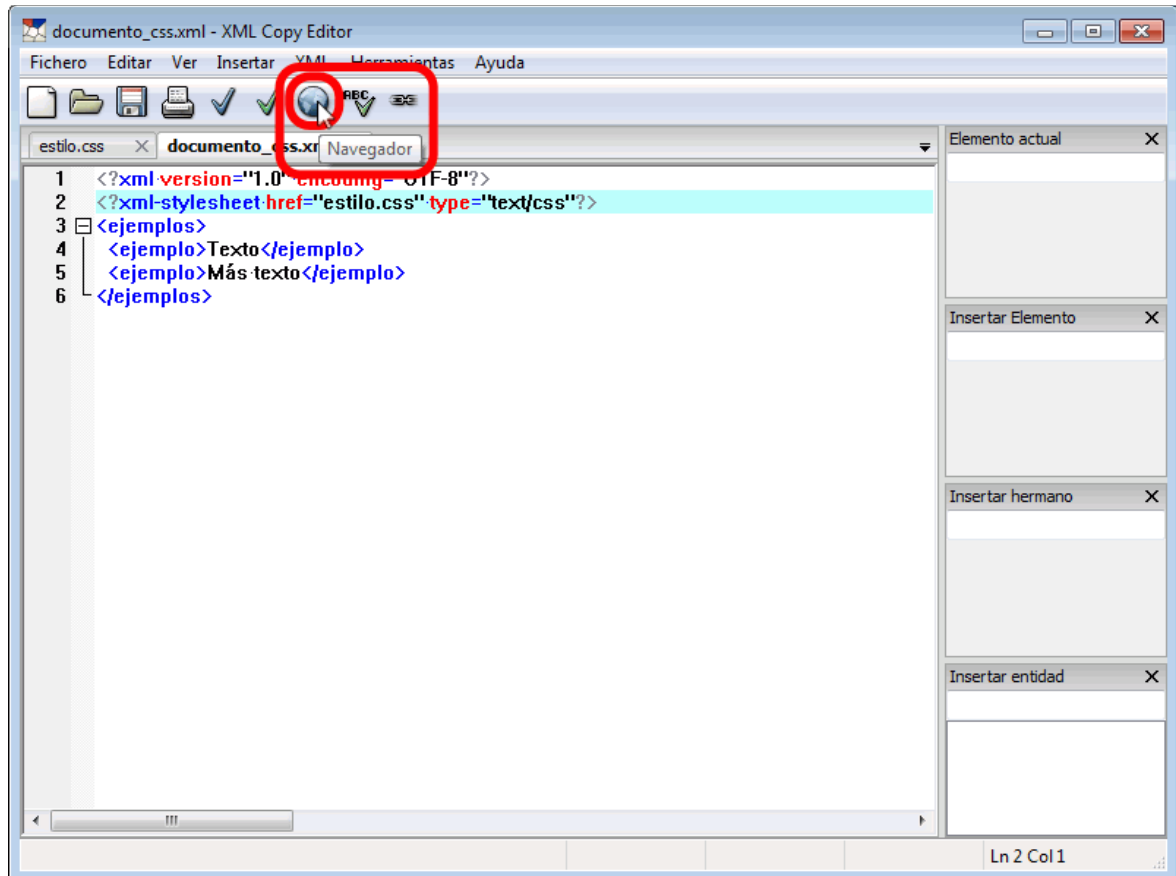


- La hoja de estilo puede crearse con cualquier editor de texto sin formato o con el propio XML Copy Editor.

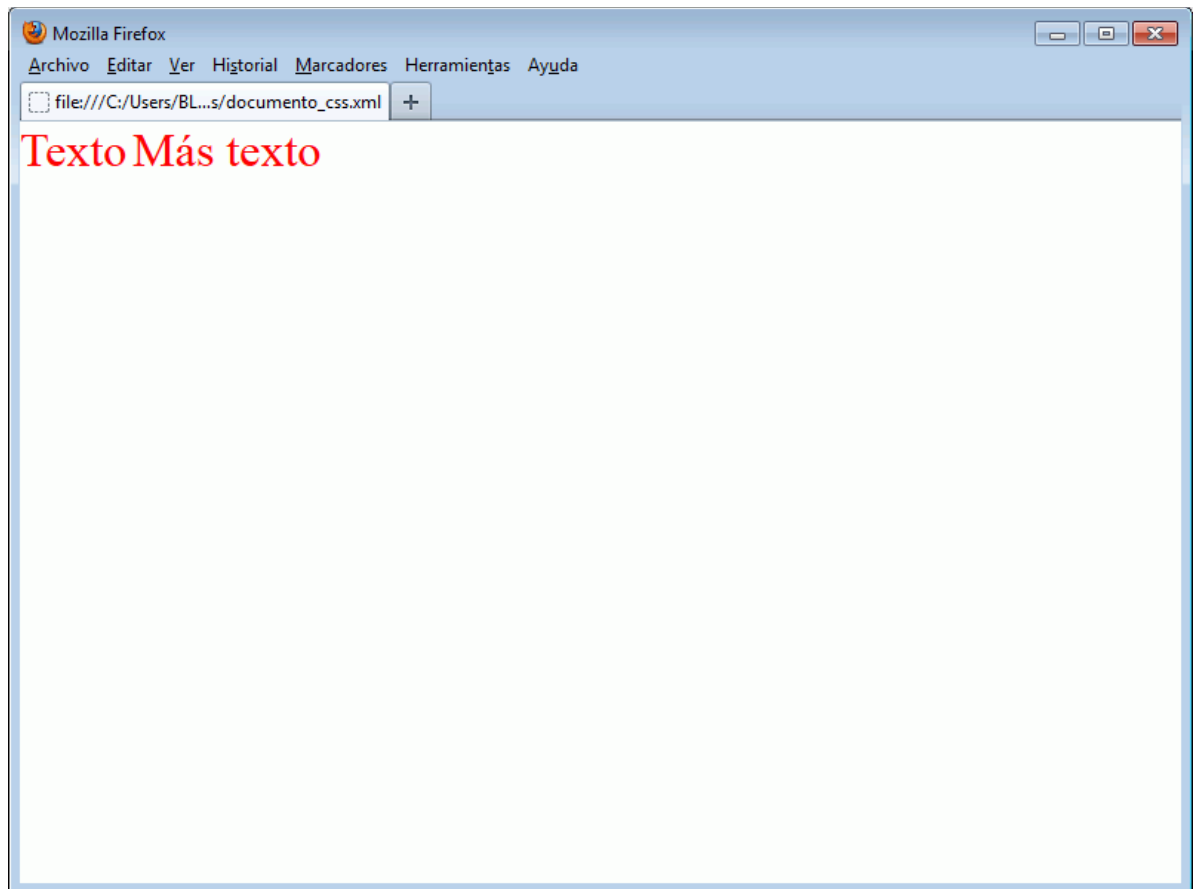
Al crear un nuevo documento, XML Copy Editor no ofrece la posibilidad de crear una hoja de estilo css, pero se puede crear un nuevo documento XML, guardarlo con el nombre y extensión deseados (en el ejemplo, estilo.css), borrar la declaración XML y escribir la hoja de estilo. Para que se coloree el código, puede ser necesario recargar el documento (mediante el menú Fichero > Recargar).



- Lo que sí se puede hacer es pedir a XML Copy Editor que abra un documento en el navegador predeterminado:

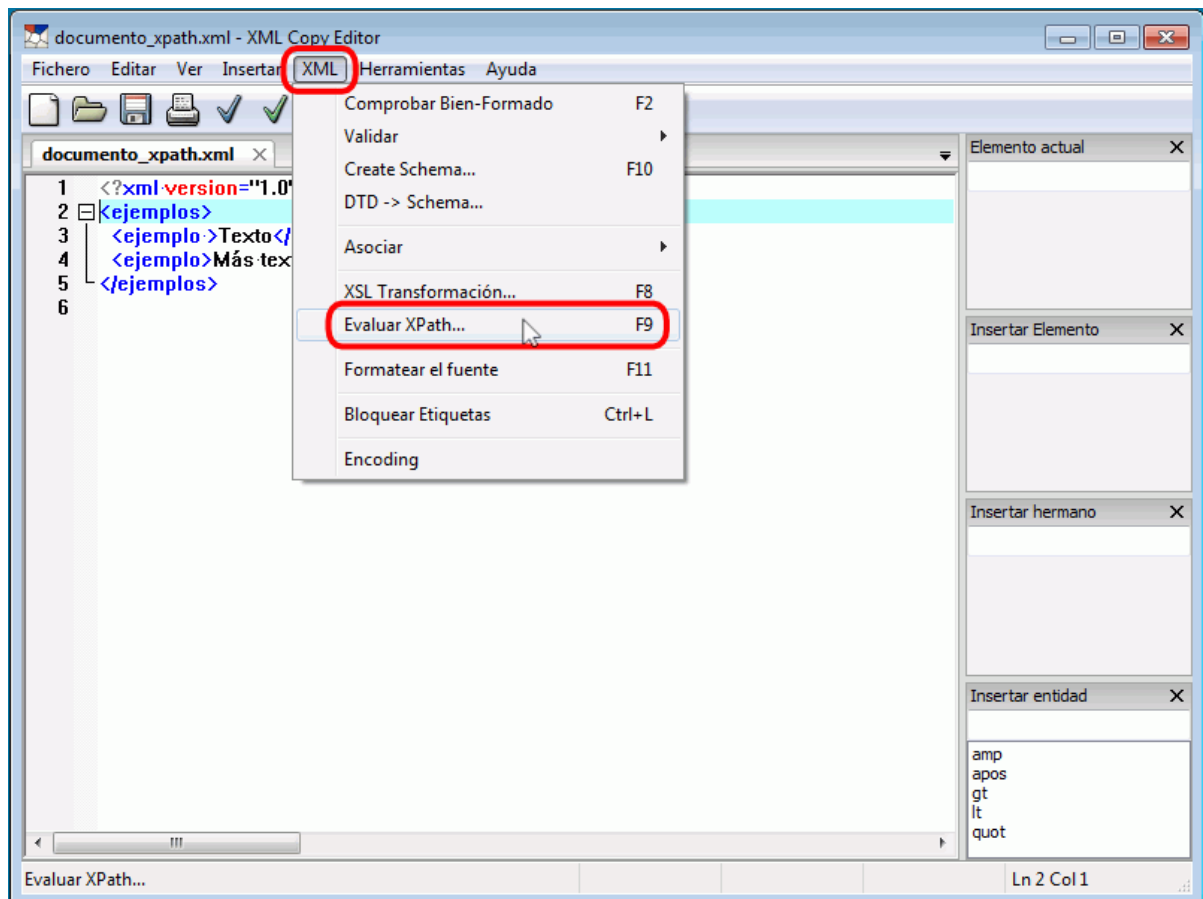


- ... y comprobar que se aplica la hoja de estilo:

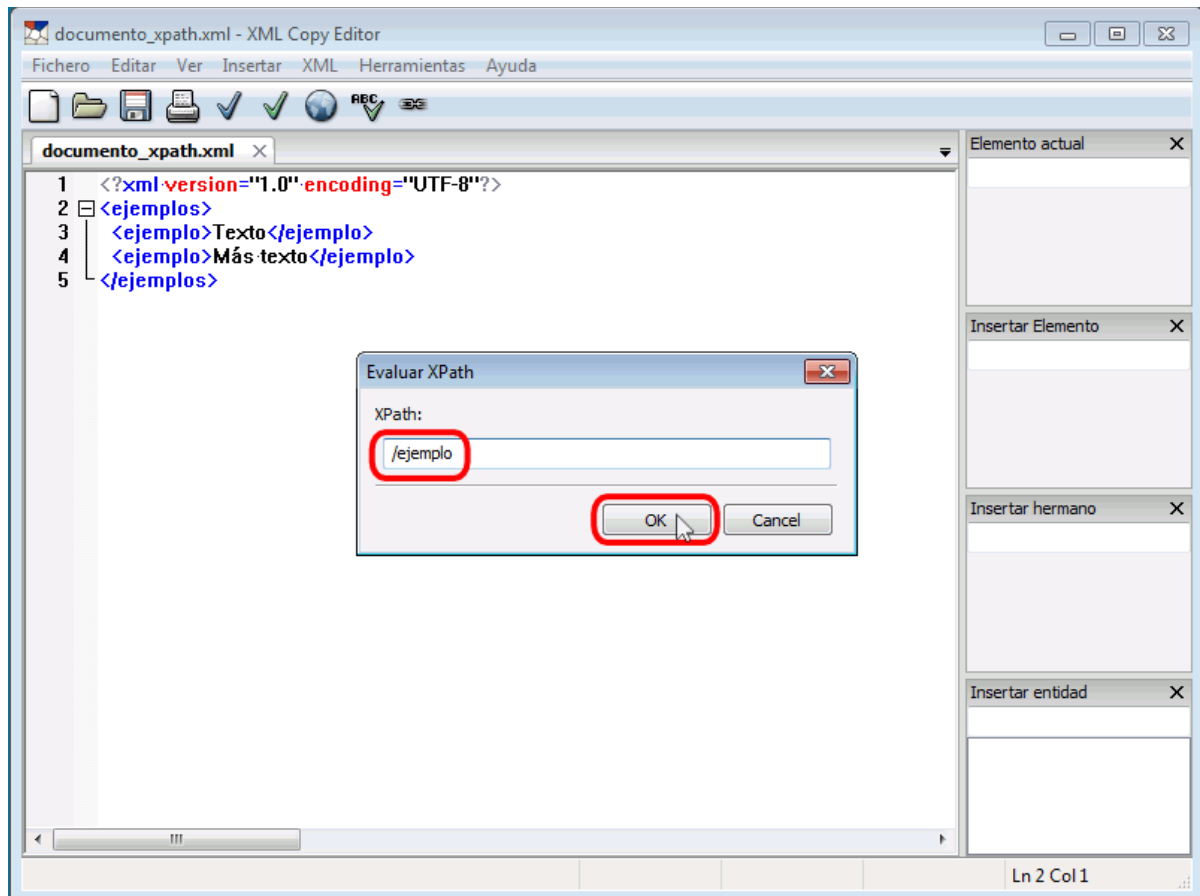


6.7.-Evaluar una expresión XPath

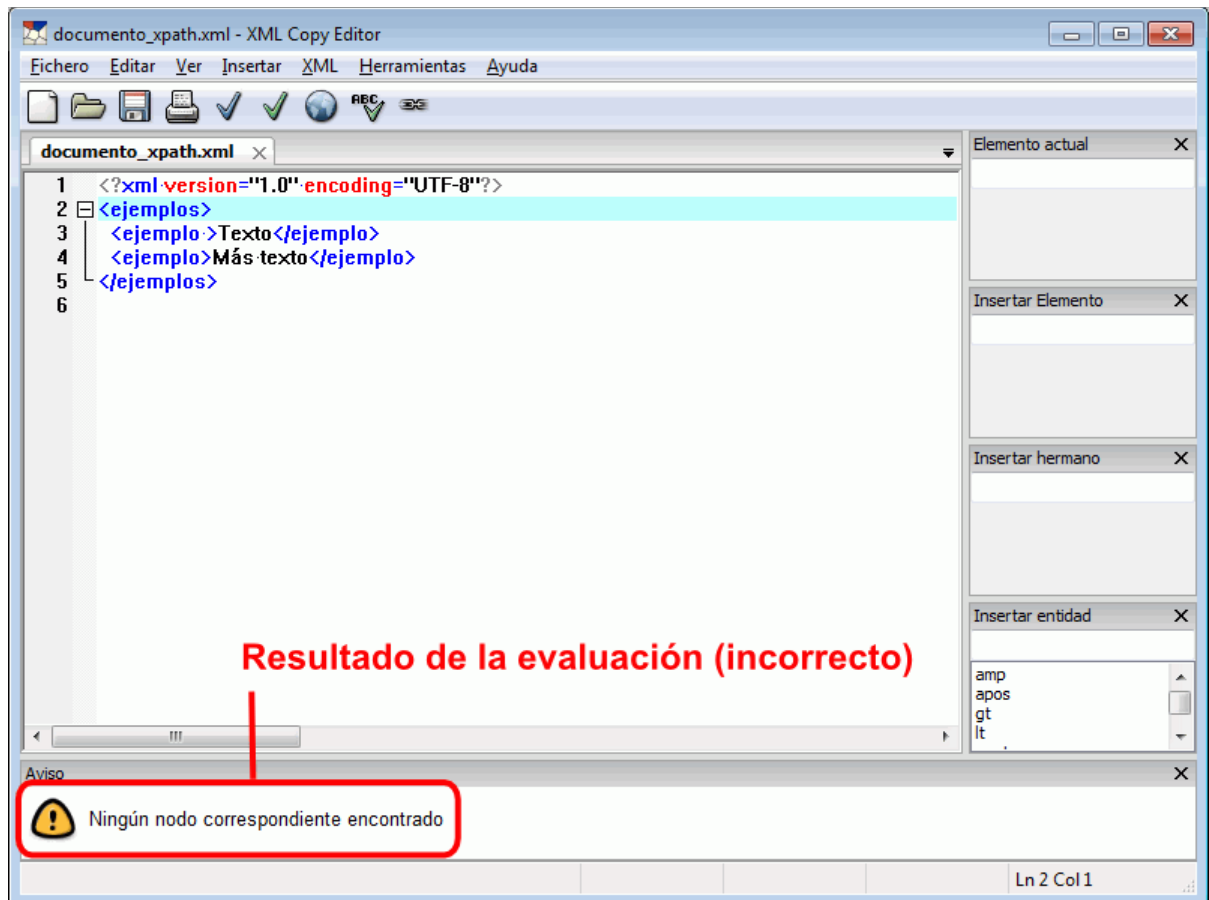
Para evaluar una expresión XPath en un documento XML se puede elegir el menú XML > Evaluar XPath... o pulsar la tecla F9 .



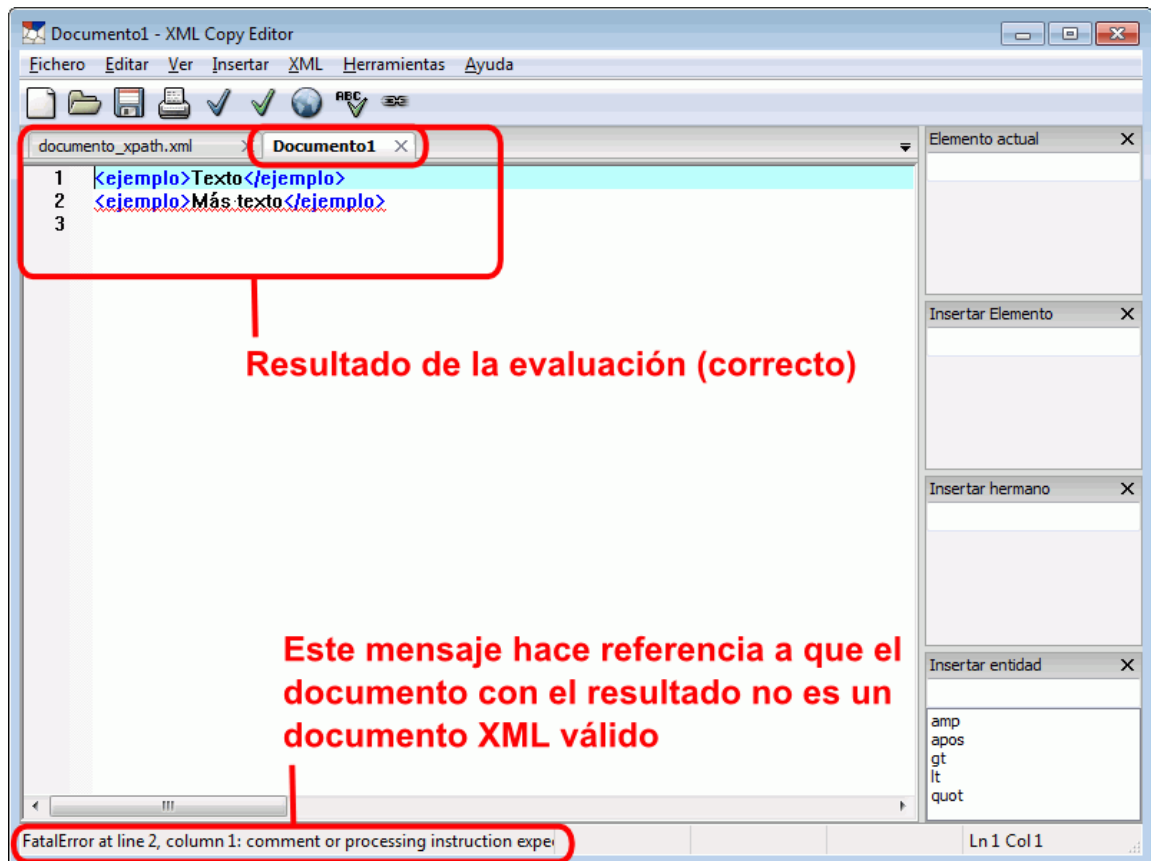
- A continuación se introduce la expresión XPath a evaluar y se hace clic en OK:



- Si la expresión no produce ningún resultado, XML Copy Editor lo indica en la parte inferior:



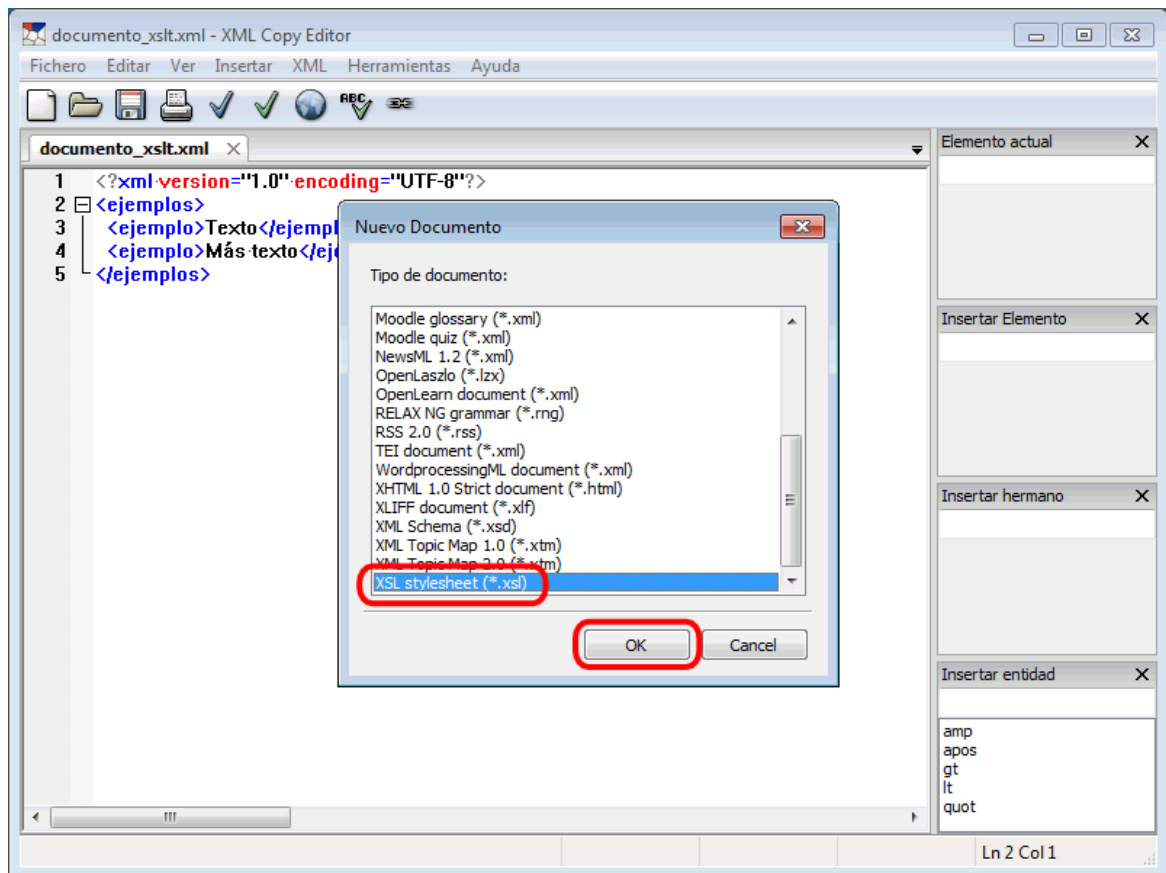
- Si la expresión produce algún resultado (por ejemplo si la expresión hubiera sido `//ejemplo`), el resultado se muestra en una nueva pestaña. El mensaje de error que aparece en la parte inferior hace referencia a que el documento que contiene la respuesta no es un documento XML válido (no tiene por ejemplo, una etiqueta que englobe todos los elementos), pero eso no quiere decir que la evaluación de XPath sea incorrecta.



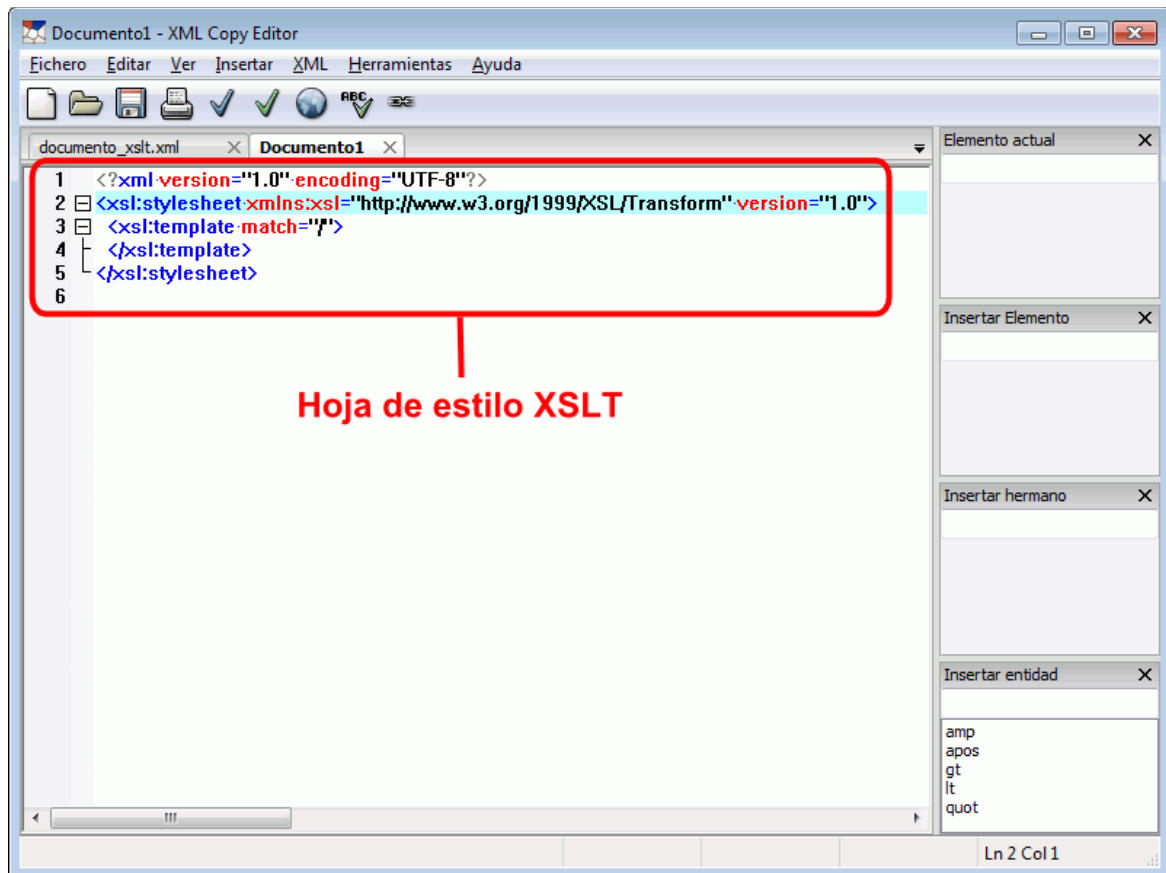
6.8.-Aplicar una transformación XSLT

Nota: Si se aplica una transformación XSLT que tiene errores, a menudo XML Copy Editor sigue mostrando el mismo mensaje de error al volver a aplicar la transformación aunque se haya corregido el error. En estos casos no queda más remedio que reiniciar XML Copy Editor.

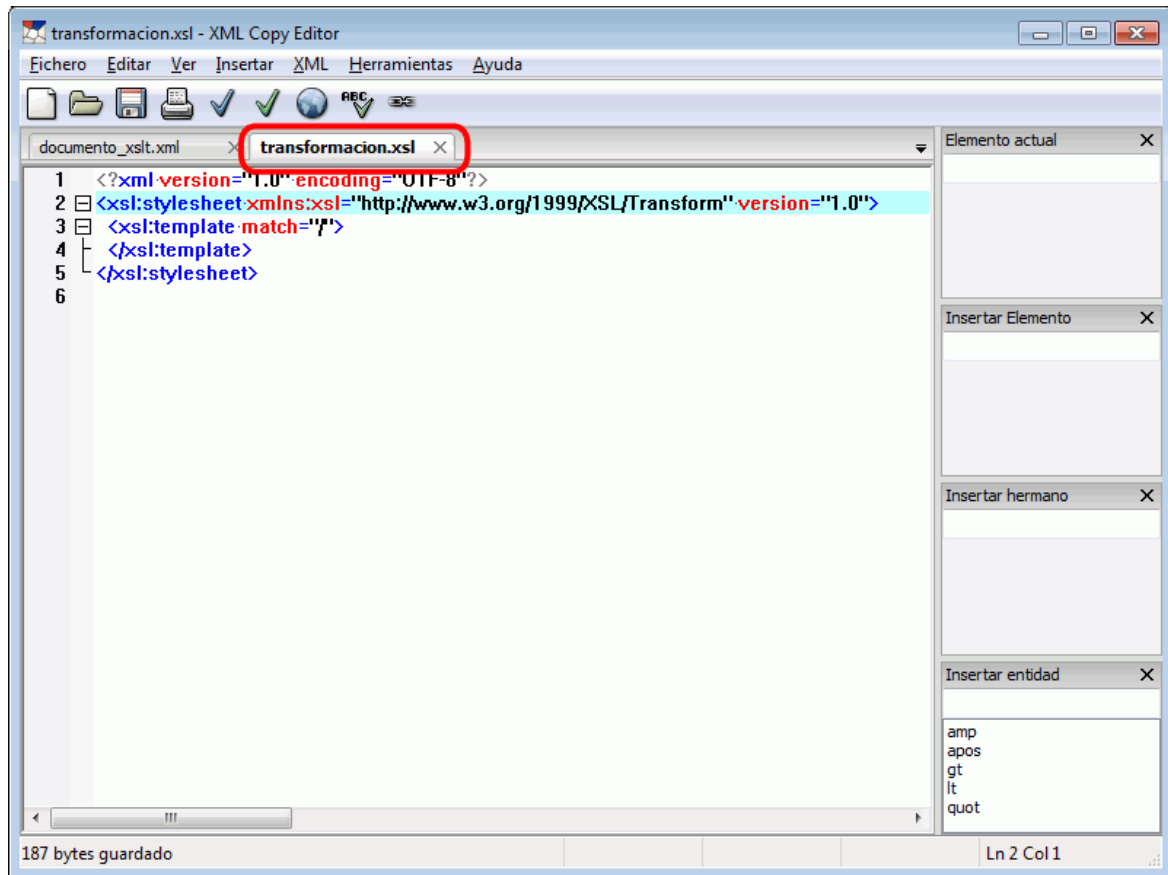
- XML Copy Editor permite crear hojas de estilo XSLT mediante el menú Fichero > Nuevo... > XSL stylesheet:



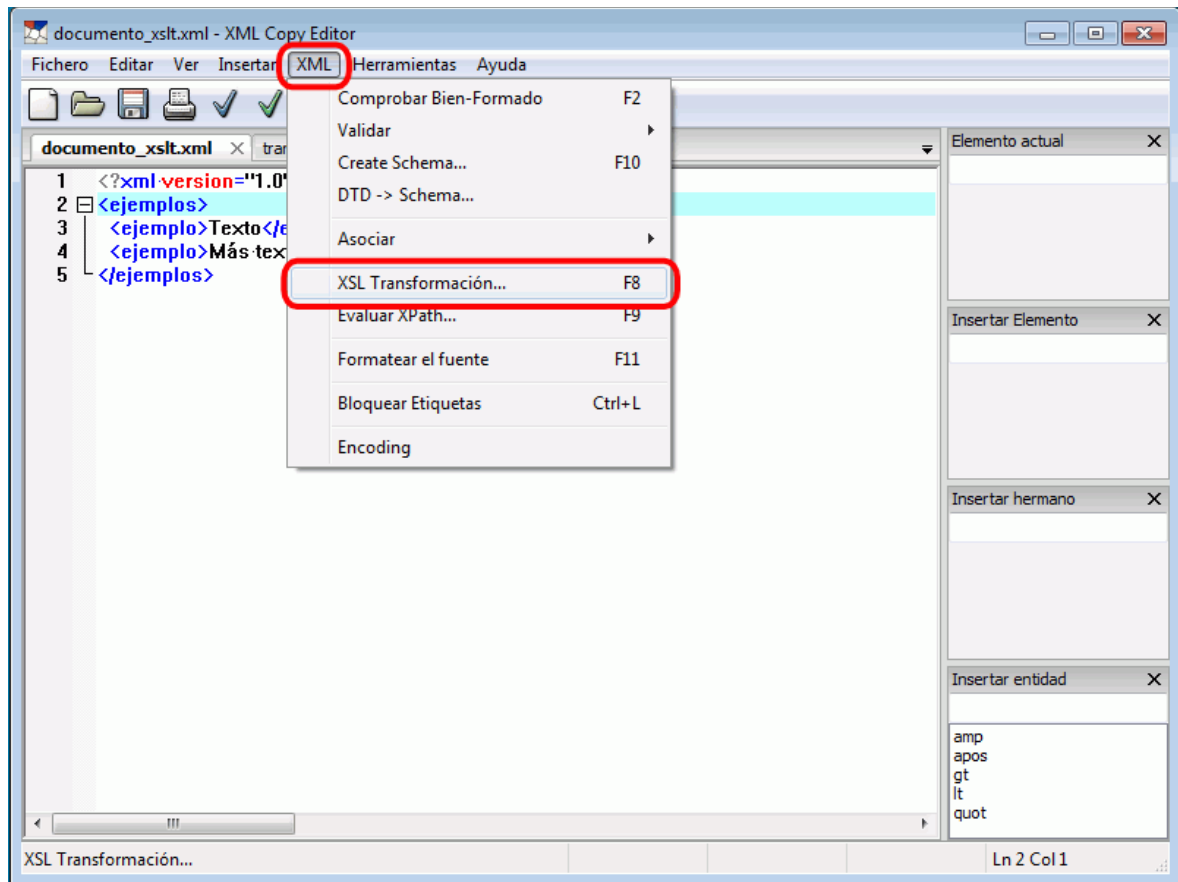
- La hoja de estilo se crea en una nueva pestaña con una plantilla:



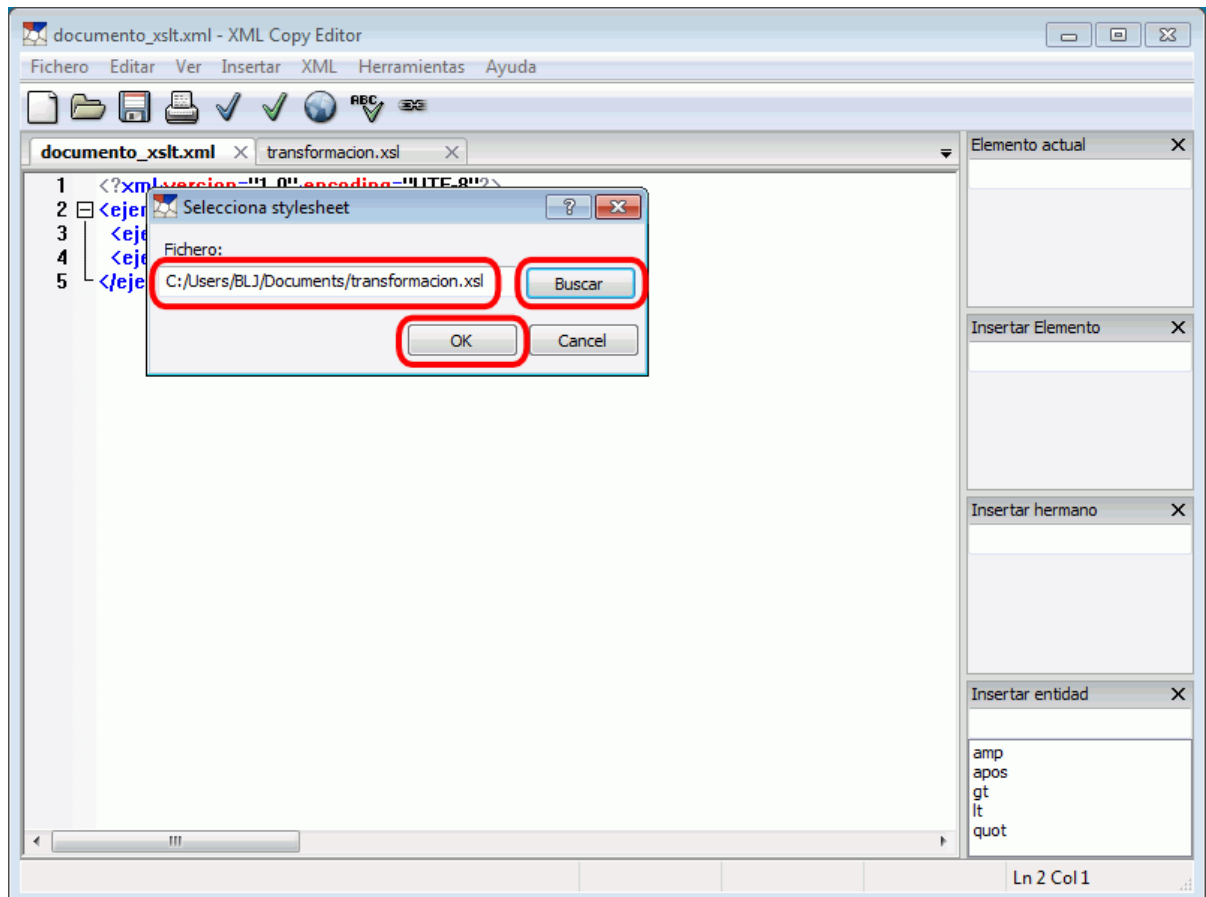
- Para poder aplicar posteriormente la transformación XSLT es necesario guardar la hoja de estilo (con la extensión .xsl):



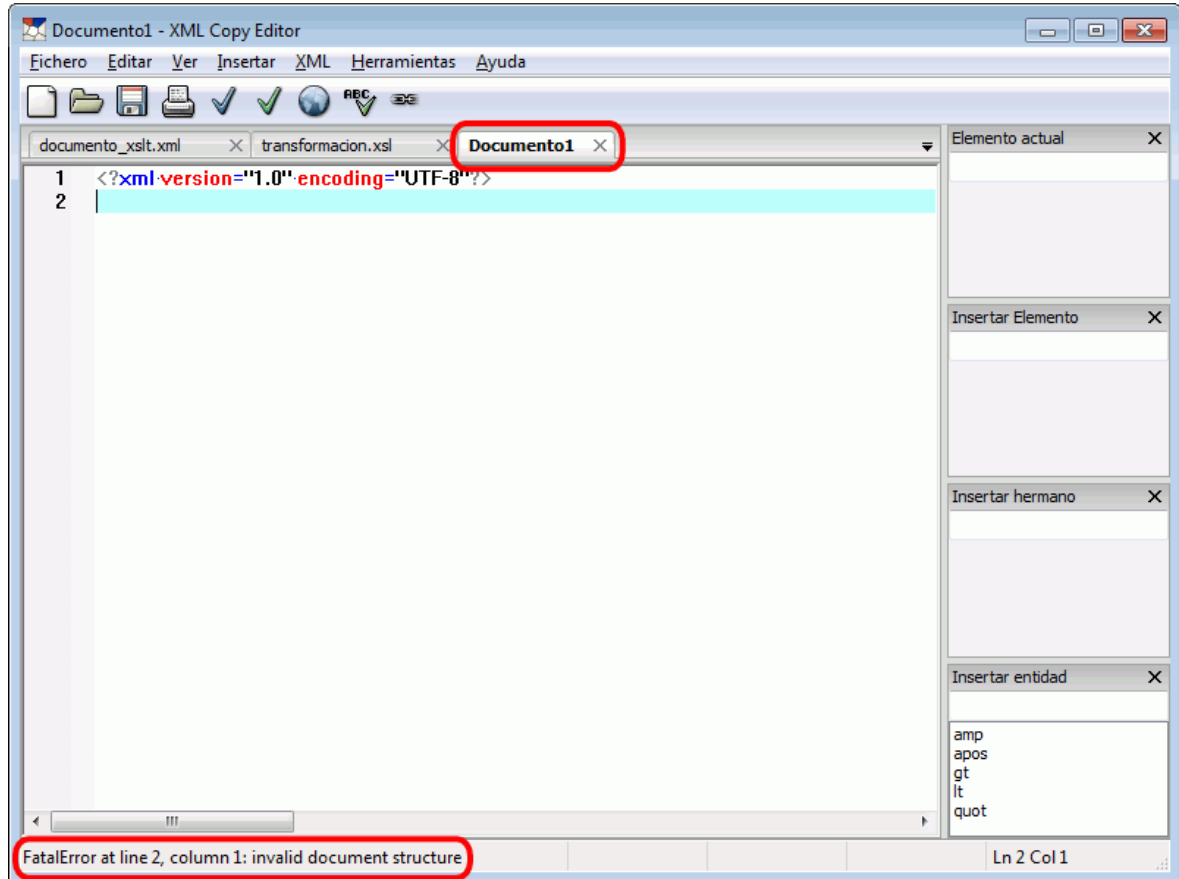
- Para aplicar una transformación XSLT a un documento XML, se debe elegir primero la pestaña en la que se encuentra el documento y elegir el menú XML > XSL Transformación... o pulsar la tecla F8.



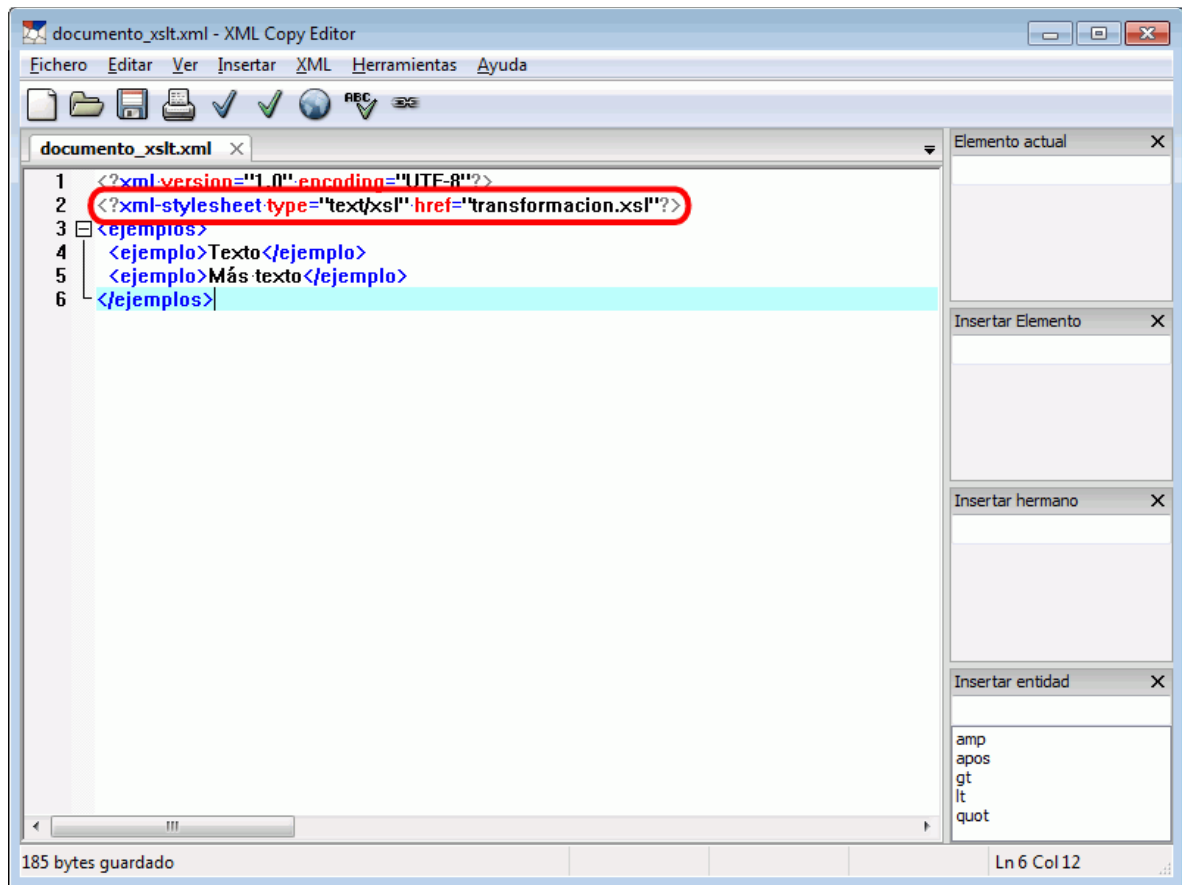
- Si el documento no enlaza ninguna hoja de estilo XSLT, XML Copy Editor solicita al usuario el nombre del archivo, que puede seleccionarse haciendo clic en el botón Buscar. Una vez seleccionado, haga clic en el botón OK.



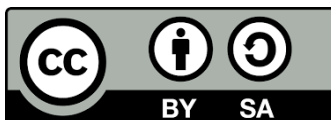
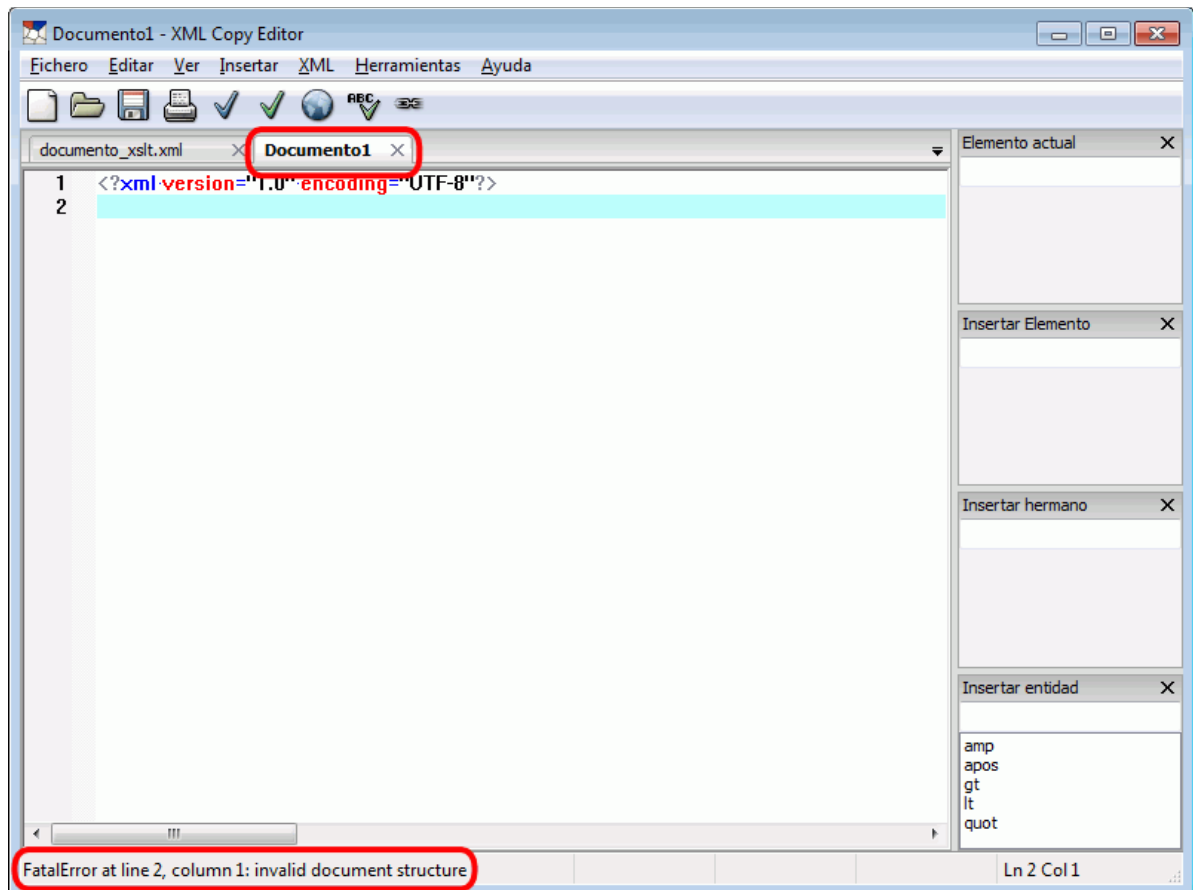
- El resultado (en este caso un documento que tan sólo incluye la declaración xml) se mostrará en una nueva pestaña. El aviso de la barra de estado no significa que la transformación no haya funcionado correctamente, sino que el resultado de la transformación no es un documento válido (puesto que sólo incluye la declaración xml):



- Si el documento XML incluye un enlace a la hoja de estilo XSLT ...



- ... al elegir el menú XML > XSL Transformación... o pulsar la tecla F8, XML muestra directamente el resultado de la transformación en una nueva pestaña (en este caso un documento que tan sólo incluye la declaración xml). El aviso de la barra de estado no significa que la transformación no haya funcionado correctamente, sino que el resultado de la transformación no es un documento válido (puesto que sólo incluye la declaración xml):



Estos materiales están basados en el curso XML: Lenguaje de marcas extensible por Bartolomé Sintés Marco que se distribuye bajo una Licencia Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional (CC BY-SA 4.0).