XPath

W3C fue consciente desde el principio de la necesidad de contar con un modelo de datos común, obviamente basado en la estructura de árbol, sobre el que se dispusiera de un lenguaje capaz de localizar los componentes específicos de un documento XML. Una vez definido se podrían resolver problemas aparentemente tan diversos como:

- Manejar bases de datos de documentos, ya que al interaccionar con la información que contiene cada documento se pueden efectuar las búsquedas correspondientes al manejo de una base documental.
- Dar un marco genérico para poder referenciar, obtener y destacar fragmentos de documentos para su posterior explotación.
- Transformar documentos que en particular servirán para poder aplicar la presentación concreta deseada sobre el resultado de esta transformación.

Se observa que todos estos objetivos tienen en común la necesidad de resolver la localización de componentes de un documento XML, ya que una vez conseguido se pueden construir otras tecnologías y aplicaciones, que permitirán alcanzar los requisitos exigidos a XML en aquellos puntos relacionados con la posibilidad de manipular documentos.

Objetivos

- Conocer la recomendación XPath.
- Entender la recomendación XPath como base de otras tecnologías.
- Aprender a construir expresiones válidas XPath para sacar información de un documento XML.

Contenidos

1. Introducción a XPath

La consecuencia de la necesidad de poder manejar los documentos XML con una cierta solvencia, fue que, solo un año después de sacar la especificación XML, W3C dio a conocer la primera recomendación de XPath (XML Path Language). Es un lenguaje declarativo que proporciona una sintaxis y un modelo de datos para poder localizar y dirigirse a una parte de un documento dado, incorporándole además algunas funcionalidades propias de un lenguaje de propósito general.

Como es conocido, una de las limitaciones de HTML consiste en la dificultad que existe a la hora de seleccionar aquellos fragmentos de un documento, que en su origen no hubieran sido especialmente destacados por el autor. En particular, ello significa que solo se pueden enlazar aquellas partes del documento previamente marcadas al efecto. Ello empezó a ser crítico a medida que la Web crecía, ya que al incrementarse el número de usuarios potenciales de un determinado documento, crece la importancia de poder escoger con libertad las partes a destacar del mismo. Un simple ejemplo: Obsérvese que al crear cursos on line, es básico poder señalar o destacar con libertad una porción o segmento de texto, al igual que se subrayan unos apuntes o un libro.

Para conseguir estas funcionalidades, XPath empezó abordando la posibilidad de localizar los componentes de un documento, siempre con el objetivo de poder destacar fragmentos, tanto de un documento XML como de las entidades externas.

XPath es, por tanto, un lenguaje destinado a encaminar hacia una parte del documento para operar sobre él.

A modo de conclusión debe retenerse que gracias a XPath se está en condiciones de poder atravesar el árbol de un documento XML, camino de sus estructuras internas, para un posterior procesado de las mismas. XPath es un lenguaje que permite seleccionar nodos de un documento XML y calcular valores a partir de su contenido. Existen tres versiones de XPath aprobadas por el W3C, aunque la versión más utilizada sigue siendo la versión 1:

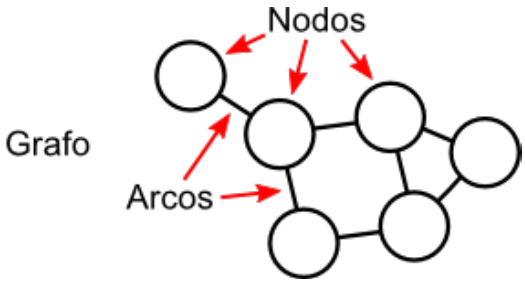
- noviembre de 1999: XML Path Language 1.0
- enero de 2007: XML Path Language 2.0
- diciembre de 2010: XML Path Language 2.0 (2º edición)
- abril de 2014: XML Path Language 3.0

2.Conceptos básicos de XPath

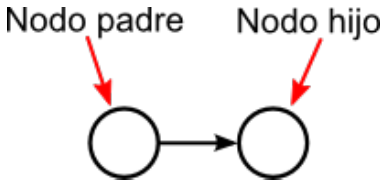
El árbol XPath.

XPath considera un documento XML como un árbol de nodos.

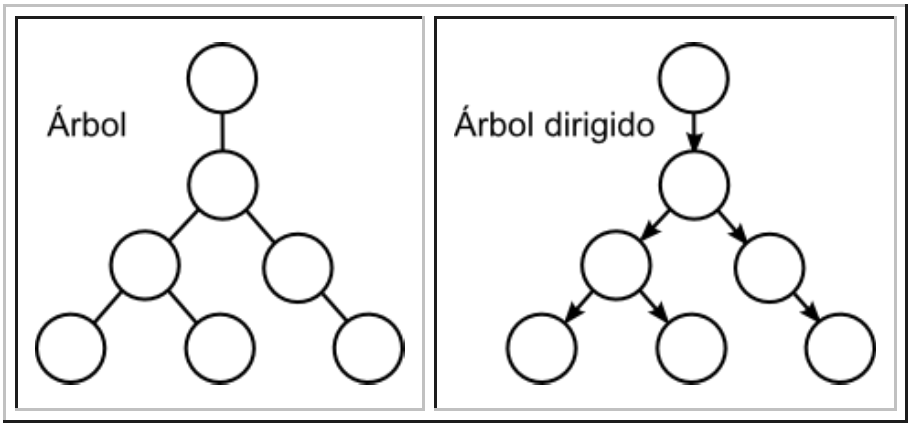
Un grafo es un conjunto de objetos llamados nodos o vértices unidos por enlaces llamados arcos o aristas. Un grafo dirigido es un grafo en el que los arcos tienen dirección.



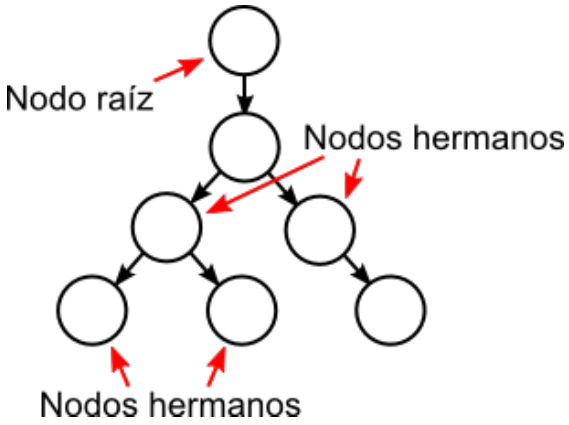
Cuando dos nodos están unidos por un arco con dirección, el nodo padre es el nodo del que parte el arco y el nodo hijo es el nodo al que llega el arco.



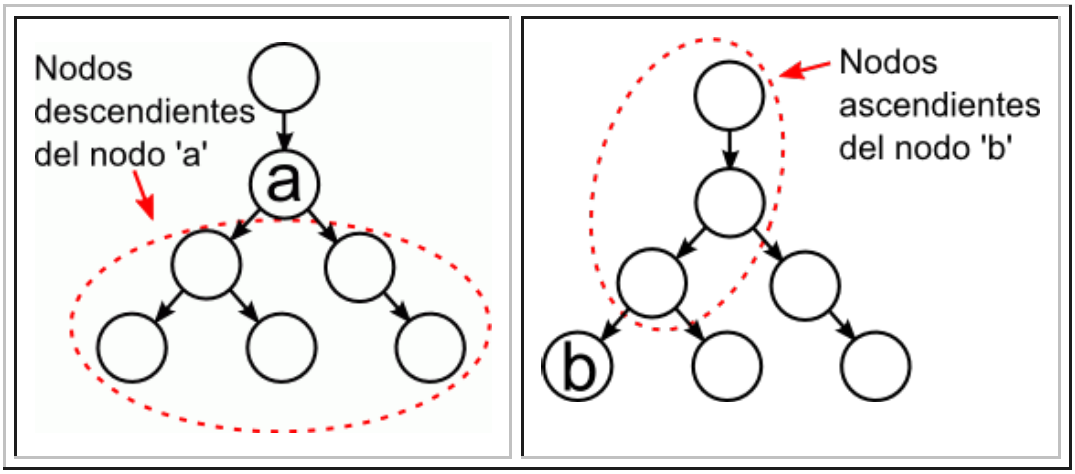
Un árbol es un grafo en el que cualquier pareja de vértices están conectada por un único camino (es decir, que no hay ciclos). Un árbol dirigido es un árbol en el que las aristas tienen dirección y todos los nodos menos uno tienen un único padre.



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.



Al ser el objetivo principal de XPath dirigirse a una parte de un documento XML, hay que facilitar la posibilidad de moverse a través de él, cosa que se hace utilizando una estructura jerárquica en árbol, y en consecuencia debe contar con un mecanismo que permita al lenguaje seleccionar información del documento objeto del procesamiento. XPath trata a todo documento como un árbol, de forma que siempre cuenta con la capacidad de poder saber si un nodo de este árbol se ajusta o no a la expresión XPath que se utilice en cada momento.

Al objeto de seleccionar partes de un documento, XPath construye internamente un árbol de nodos llamado árbol XPath (excepto las posibles declaraciones DOCTYPE que no se representan) en el que partir de la raíz, el propio documento, se diversifica a lo largo de los elementos hasta las hojas constituidas bien por valores indivisibles (llamados **átomos**) tales como números, cadenas, etc., o bien por **referencias** a nodos de otro documento. En coherencia con ello, XPath maneja cuatro tipos de datos:

- **cadenas** (de caracteres Unicode),
- **números** (en coma flotante),
- **valores booleanos** (true o false)
- **conjuntos de nodos** que en la práctica se tratan como una lista.

Los nodos de un árbol XPath pueden ser de siete tipos:

- **Raíz**
- **Elemento**
- **Atributo**
- **Texto**
- **Comentario**
- **Instrucción de Proceso**
- **Espacio de Nombres.**

En este árbol cada nodo intermedio contiene listas ordenadas de nodos hijos, siendo la relación entre un nodo padre y un nodo hijo que el primero contiene al segundo; por ello, además del raíz, solo pueden tener hijos los siguientes nodos: elemento, comentario, texto e instrucción de proceso, mientras que los nodos atributo y espacio de nombres se considera que solo describen a su nodo padre y no contienen nodo alguno.

El uso del término "**Path**"obedece a que la estructura que crea XPath se inspira en el tradicional "File Path" o ruta de acceso utilizada en un disco duro (carpetas y archivos) aunque realmente la semántica de XPath sea mucho más parecida a la del SQL en Bases de Datos.

Como veremos, al especificar su sintaxis, XPath utiliza como expresión patrón para identificar los nodos de un documento. La barra (/) al inicio de la expresión, seguida de una lista con los nombres de los elementos hijos que describen un recorrido a través del documento, de forma que tiene capacidad para seleccionar los sucesivos elementos que se ajustan al mismo.

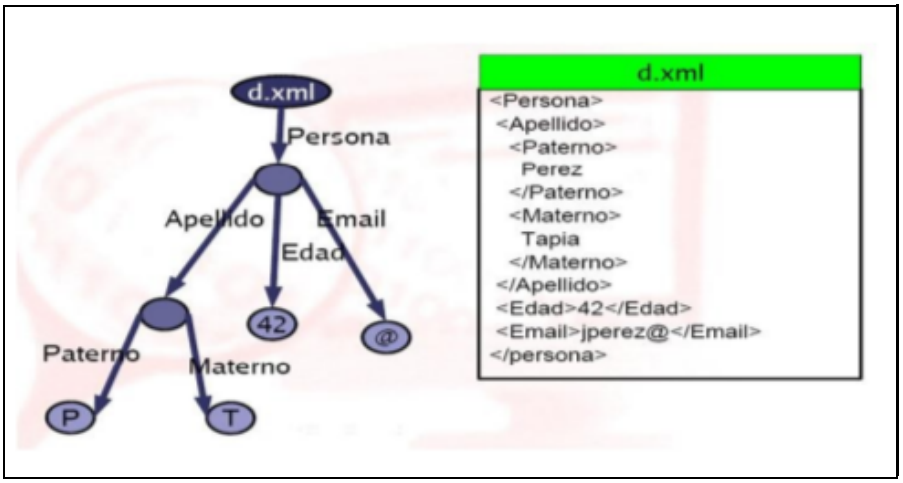
En el sentido semántico, es importante señalar que a pesar de que la sintaxis de XPath tenga un estructura similar a la de un path en Unix como "/usr/home/pedro/docs", la cual hace referencia a un único archivo docs que cuelga del conjunto de directorios /usr/home/pedro, cuando una expresión análoga aparece en XPath, presenta dos diferencias fundamentales respecto a la del Sistema Operativo:

- La primera consiste en que al considerar una expresión XPath como: /libro/capitulo/parrafo, se hace referencia a **TODOS** los elementos parrafo que cuelguen directamente de **CUALQUIER** elemento capitulo que cuelgue de **CUALQUIER** elemento libro, que a su vez cuelgan del nodo raíz.
- La segunda estriba en que XPath no devuelve los elementos que cumplen con el patrón que representa cada expresión, sino que solo devuelve una referencia, es decir, una lista de punteros a los elementos que encajan con el patrón de localización. Evidentemente, esta lista resultado puede estar vacía o contener uno o muchos nodos del árbol XPath correspondiente.

En resumen, en el modelo XPath (que como veremos es compartido por otros lenguajes de la familia XML como XSLT y otros) los datos XML se representan en forma de nodos y valores, que sirven de operandos y de resultados de los operadores. Las sentencias XPath se representan en forma de secuencias, entendiendo como tal una colección de uno o más ítems, donde un ítem es un nodo o un valor atómico. Por tanto este modelo de datos se basa en 3 bloques:

- Valores atómicos de varios tipos, unos generales (cadenas, enteros, etc.) y otros más especializados como nombres cualificados o URI's.
- Árboles cuyos nodos representan el contenido de un documento XML.
- Secuencias (o listas) cuyos ítems son valores atómicos o referencias a nodos en los árboles correspondientes.

De acuerdo con este modelo, cuando se da una expresión XPath aplicada a un documento, su resultado no puede ser otro que una selección de nodos, o un valor atómico (o una sucesión de ellos).

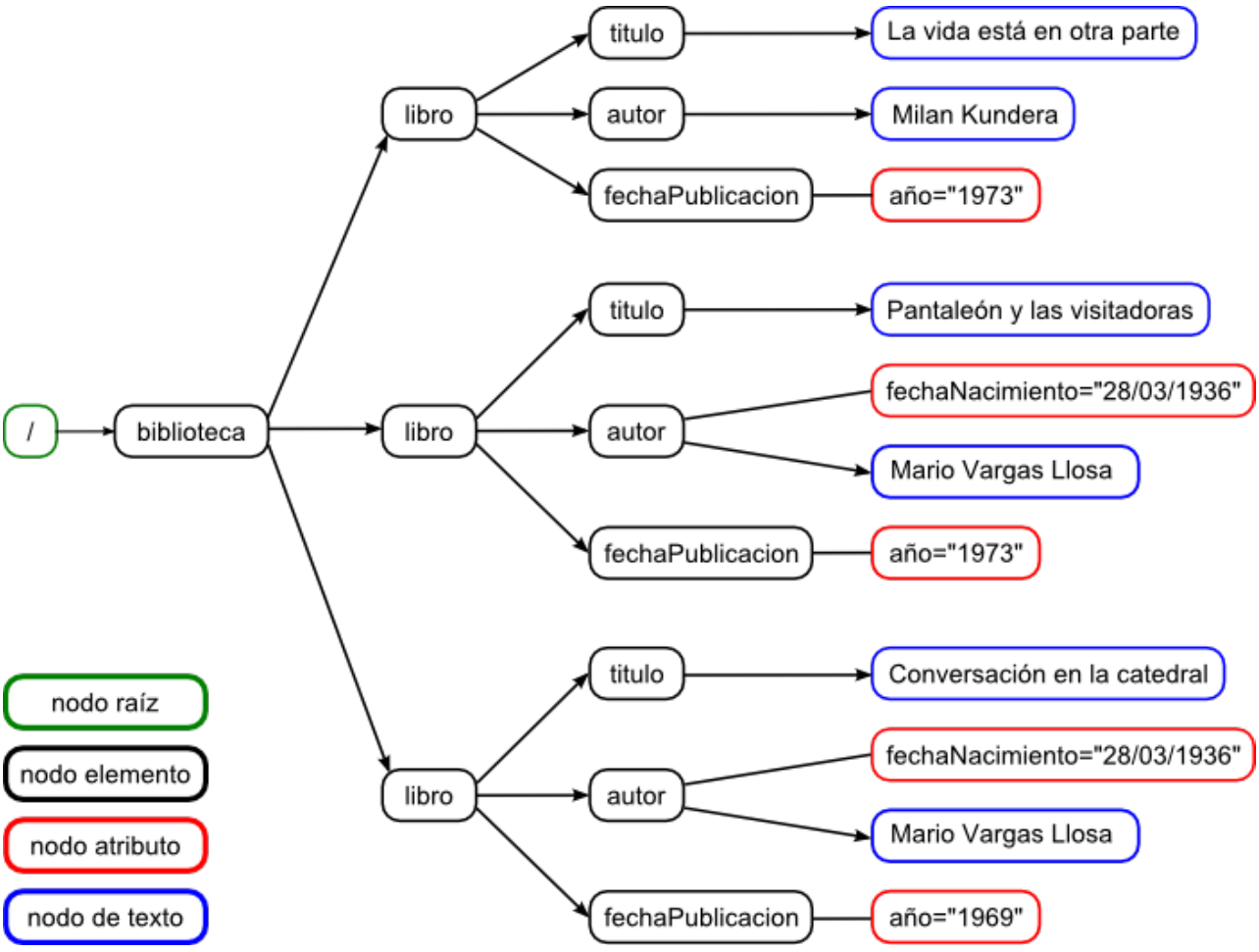


En la imagen anterior podemos ver un ejemplo de árbol XPath construido a partir del documento "d.xml".

Por ejemplo, el documento XML siguiente:

```xml
<?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 elementos y para referirse a los atributos o al texto se utilizan notaciones especiales.

2. Sintaxis y notación.

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.

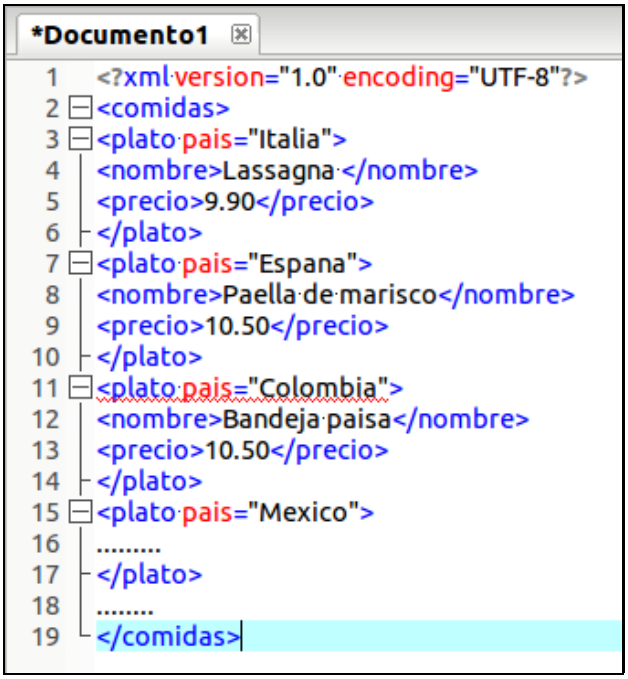
Las expresiones XPath se pueden escribir de dos formas distintas:

- sintaxis abreviada: más compacta y fácil de leer
- sintaxis completa: más larga pero con más opciones disponibles

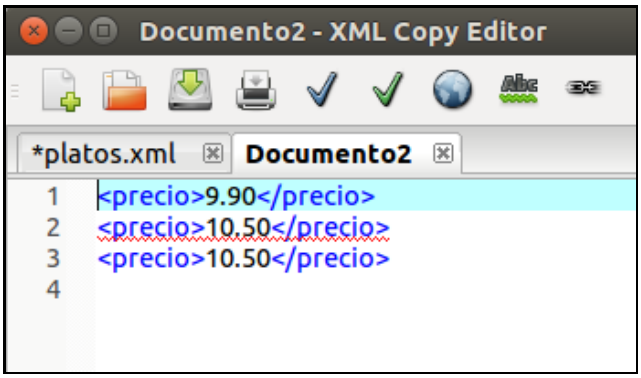
Las expresiones XPath se pueden dividir en pasos de búsqueda. Cada paso de búsqueda se puede a su vez dividir en tres partes:

- **eje**: indica el nodo o los nodos en los que se realiza la búsqueda
- **nodo de comprobación**: especifica el nodo o los nodos seleccionados dentro del eje
- **predicado**: permite restringir los nodos de comprobación

XPath utiliza la barra (/) seguida de una lista con los nombres de los elementos hijo, que en su conjunto describen un recorrido, a través del documento, de forma que selecciona los sucesivos elementos que se ajustan al recorrido expresado por la secuencia como expresión patrón para identificar los nodos de un documento, Así, dado el siguiente documento con la lista de platos internacionales de un restaurante:



Cuando se escriba la secuencia: "**/comidas/plato/precio**", en el marco adecuado, XPath acabará seleccionando **todos** los precios de **todos** los platos que figuran en él. El resultado de la expresión anterior sería la siguiente:



La colección de todos los nodos precio hijos de los nodos plato que a su vez son hijos del nodo comidas.

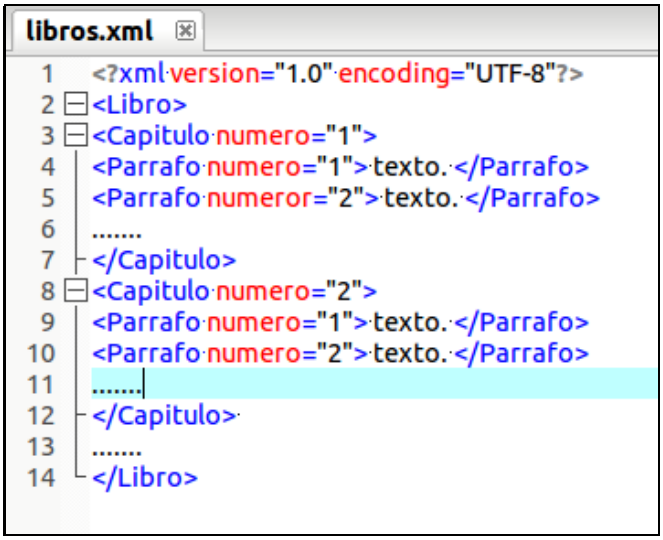
Trayecto de búsqueda.

En XPath se llama trayecto de búsqueda (location path) al conjunto de nodos sobre los que se evalúa una expresión XPath, que obtendrá como resultado otro conjunto de nodos, formado por aquellos que cumplen los criterios especificados. Para dar los nodos del trayecto, se empieza designando un punto concreto, llamado **nodo contexto**, que es el nodo del árbol XPath del documento desde el que se inicia el trayecto de búsqueda.

Tanto el concepto de nodo contexto como trayecto de búsqueda son análogos a los utilizados en la gestión de archivos. De esta forma, en XPath, a menos que se indique un trayecto explícito se entiende que el trayecto de búsqueda parte del nodo que en cada momento se esté procesando. Por ello, los trayectos de búsqueda pueden ser absolutos o relativos; los primeros empiezan con una barra **(/)** que determina el nodo contexto, mientras que uno relativo carece de ella al inicio, y depende del lugar del documento en el que se esté trabajando al invocar el trayecto de búsqueda.

Sea absoluto o relativo, un trayecto consiste en una serie de escalones separados por una barra, de forma que un trayecto de búsqueda absoluto es de la forma: “/escalon/escalon/...” y uno relativo adopta la formula: “escalon/escalon/... “.

La expresión de un trayecto se evalúa de izquierda a derecha. Así, dado el documento siguiente con la estructura habitual de un libro:



La expresión: "**/libro/capitulo/parrafo**", al empezar con /, selecciona el nodo raíz, independientemente del nodo contexto de cada momento; a continuación al leer libro, se seleccionan TODOS los elementos que cuelgan del nodo contexto (que en este caso es único por ser raíz), al seguir y leer capitulo se seleccionan TODOS los elementos que cuelgan del nodo contexto. Al llegar a este punto hay que hacer notar que si se estuviera gestionando un disco sería imposible que hubiera dos directorios con el mismo nombre colgando de un mismo directorio padre, cosa que sin

embargo, en un documento XML es perfectamente normal, como en el ejemplo donde son varios los hijos capitulo que cuelgan del elemento libro. Al continuar leyendo se llega al elemento <parrafo>, con lo que se indica seleccionar TODOS los elementos <parrafo> que cuelgan del nodo contexto (ahora capitulo) aunque haya varios nodos contexto.

Ante este tipo de situaciones, el evaluador de expresiones XPath los va a recorrer uno por uno, siguiendo la regla de "primero en profundidad" (empezar por la primera rama hasta llegar a un nodo hoja y volver para seguir por la rama siguiente para proceder de igual forma hasta agotar todas las posibilidades) mientras se evalúa un determinado nodo, éste sea el nodo contexto momentáneo.

Como se adelantó, el resultado al evaluar una expresión XPath es un conjunto de punteros a nodos que encajan con el patrón buscado. Aunque no es el objetivo profundizar en las particularidades de la sintaxis de XPath, ya hemos podido ver que la sintaxis habitual para tratar un árbol incorpora una serie de abreviaciones que son muy evidentes y que vale la pena citar:

- En un trayecto de búsqueda, cuando el nodo raíz es el nodo contexto, basta con escribir "/" .
- Una expresión como c**hild::Autor** se abrevia como Autor (como veremos el llamado eje child es el eje por defecto).
- El operador ("**//**") indica que se seleccionen todos los elementos del documento que cumplan los criterios con independencia del nivel que ocupen en el árbol. Por ejemplo: "//plato" selecciona todos los elementos plato de la lista de comidas.
- El operador ("**|**") selecciona varios recorridos; por ejemplo: "/comidas/plato/nombre | /comidas/plato/precio" proporciona todos los elementos nombre y elemento precio de los platos del documento.

Las abreviaturas anteriores, se pueden combinar y así:

- //nombre | //precio, Selecciona todos los elementos nombre y precio del documento.
- /comidas/plato/nombre | //precio, Selecciona todos los elementos nombre del elemento plato del elemento comidas, así como todos los elementos precio del documento.

3. Elementos de un trayecto de búsqueda

En todo trayecto de búsqueda de XPath se pueden distinguir tres elementos básicos llamados Ejes, Nodos y Predicados. Siendo esta sintaxis de localización en XPath:

- **nombreeje::nodocomprobacion[predicado]**

Como por ejemplo:

- **child::precio[precio=9.90]**

Ejes.

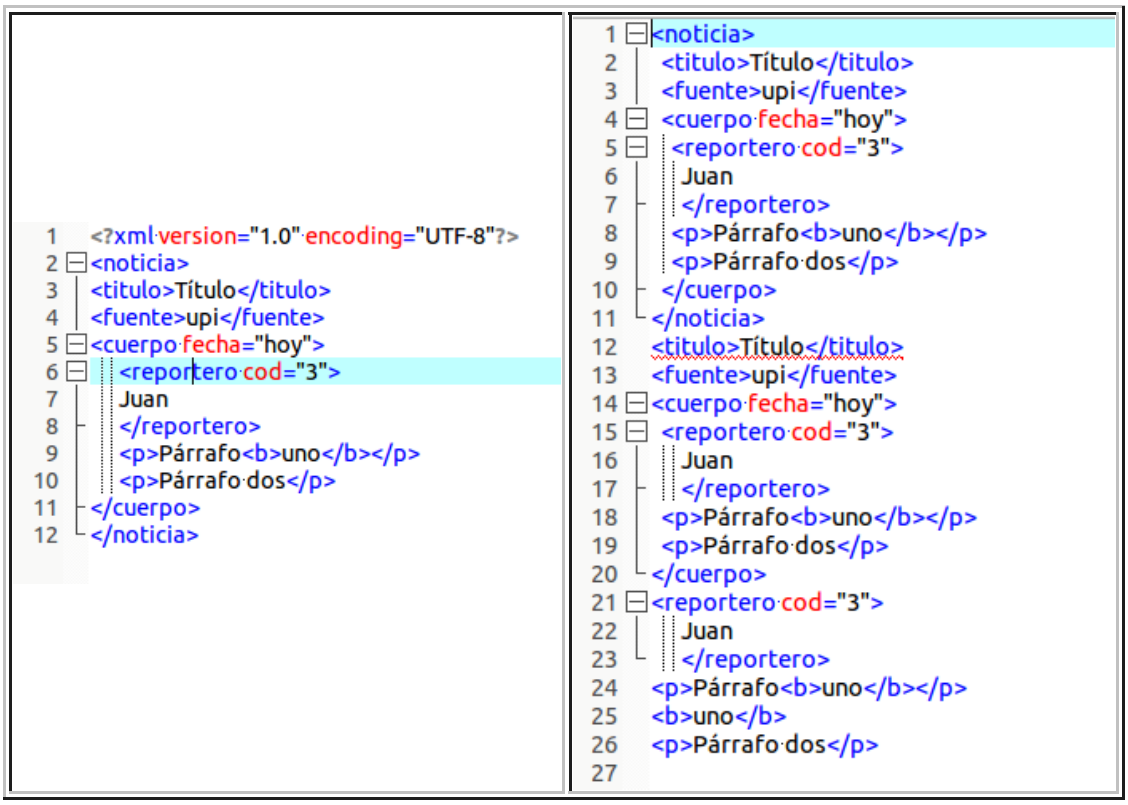
Los Ejes (axes) son elementos encargados de especificar en un trayecto de búsqueda la relación existente dentro del árbol XPath, entre los nodos que se quieren seleccionar en el trayecto, y el nodo contexto de donde se parte. El término inglés axes significa también, además del plural de eje, "cercenar" o "podar" y ambos conceptos serían adecuados, ya que con el eje incluido en un trayecto se realiza una selección de nodos de acuerdo con el patrón, dentro del árbol o mejor dicho, dentro del subárbol que cuelga del nodo contexto. Se observa, en los ejemplo vistos, que el uso de la barra / (salvo cuando ha servido para denominar el nodo raíz) determina un eje.

En XPath, existen 13 tipos de ejes siendo los más habituales los siguientes:

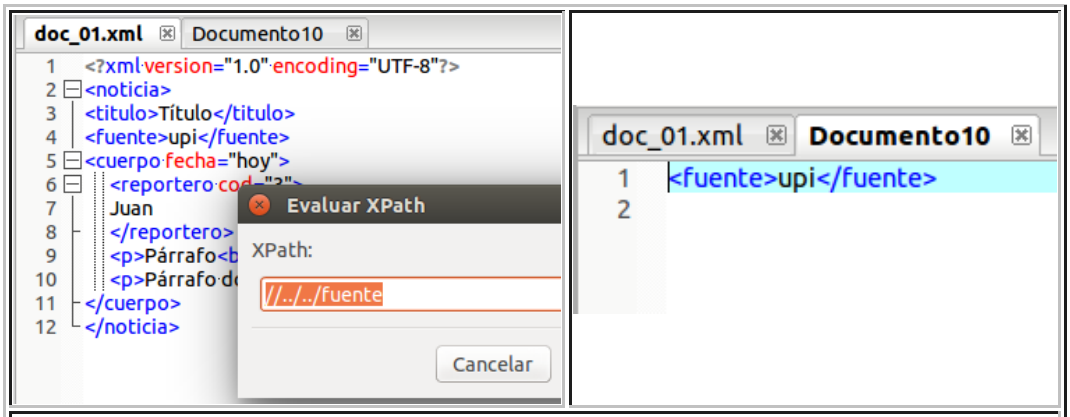
- **Self**: Identifica el nodo contexto y se especifica mediante un punto (.)
- **Child**: Describe los hijos del nodo contexto, y se expresa de la forma : "/child::", aunque como se ha indicado "child" puede omitirse de un trayecto de búsqueda, al ser el eje por defecto y así para seleccionar los títulos de libro basta con usar: /libro/titulo.
- **Descendant**: Selecciona cualquier nodo que sea descendiente del conjunto de nodos contexto, expresándose de la forma: "//descendant::", palabra que puede evitarse pues siempre se asume por defecto tras //. Para seleccionar todos los párrafos de libro, se escribe: /libro//parrafo.
- **Attribute**: Contiene los atributos de un determinado nodo y se abrevia usando @ como prefijo del elemento buscado; por ejemplo: "//@pais" selecciona todos los atributos con el nombre de país, aplicado al documento de platos.
- **Parent**: Indica el nodo padre del nodo contexto y se identifica con dos puntos seguidos(..). Así, para seleccionar los nodos que tienen algún hijo de parrafo escribiríamos: "//parrafo/.." mientras que para seleccionar los nodos capitulo que tienen algún hijo de tipo párrafo se escribe: "//capitulo/parrafo/.."

Algunos ejemplos.

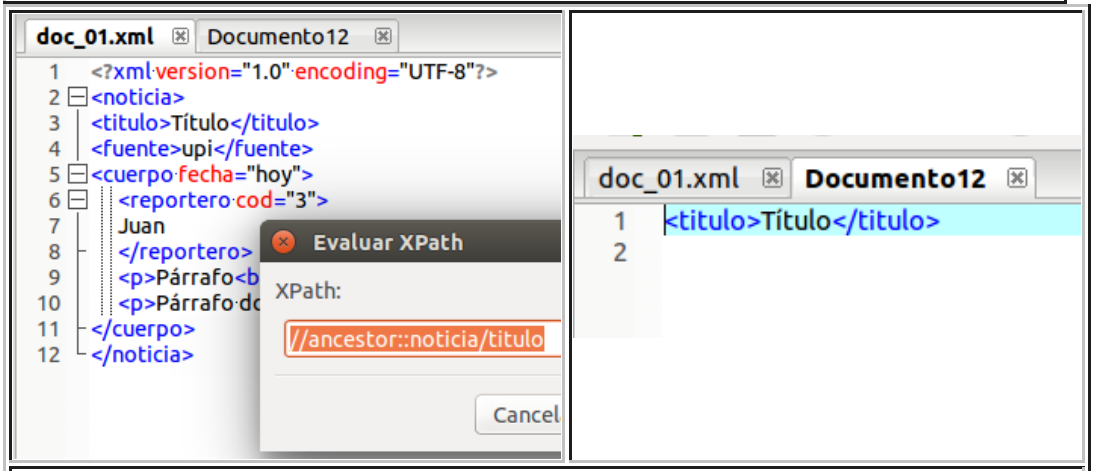
En las imágenes siguientes podemos ver algunos ejemplos de expresiones XPath que utilizan lo visto hasta ahora. Cada imagen se compone de un fichero XML sobre el que se aplican una serie de expresiones XPath y vemos el resultado obtenido.



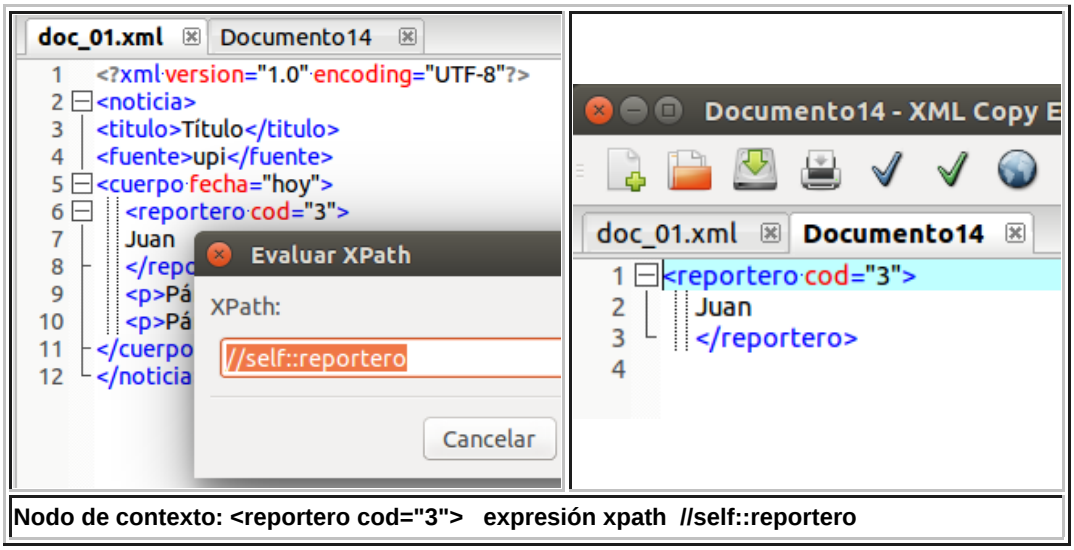
Nodo de contexto: <reportero cod="3"> expresión xpath //parent::*



Nodo de contexto: <reportero cod="3"> expresión xpath ../../fuente



Nodo de contexto: <reportero cod="3"> expresión xpath //ancestor::noticia/titulo



Nodo de contexto: <reportero cod="3"> expresión xpath //self::reportero

Otros ejemplos:

Para el siguiente documento XML,

```
<?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 está al principio de la expresión, indica el nodo raíz, si no, indica "hijo".

/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".

//: indica "descendiente" (hijos, hijos de hijos, etc.).

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

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

@atributo: selecciona el atributo.

/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

..: selecciona el elemento padre.

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

Nota: En este ejemplo se seleccionan únicamente los nodos "autor" que tienen el atributo fechaNacimiento.

|: permite elegir varios recorridos.

//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"

Nodos de comprobación o búsqueda

Los nodos de comprobación o búsqueda (node test) son los encargados de identificar un nodo o nodos concretos dentro de un eje, siendo su función dentro del trayecto de búsqueda que cada eje pueda determinar un tipo de nodo

(llamado principal) a la hora de efectuar la selección. Este nodo de comprobación puede indicarse bien por nombre o bien por tipo.

Son los nodos más importantes y usados:

- **node( )**: Devuelve todos los nodos de cualquier tipo. Así para seleccionar todos los nodos descendientes del tipo párrafo, se escribe: “/parrafo/node( )”

  Selecciona los nodos principales de cada trayecto (a excepción de los tipo: texto, comentario e instrucciones de proceso) indicando su nivel en el árbol. Así:

  - **“/comidas/plato/*”** selecciona todos los elementos hijos de todos los plato.
  - **“/catalogo/*/precio”** obtiene los precios de los elementos que son nietos del elemento catálogo.
  - **“/*/*/precio”** obtiene el precio de los elementos que tienen dos antecesores.
  - **“//*”** selecciona todos los elementos del documento.

- **text ( )**: Devuelve cualquier nodo de tipo texto. Así para seleccionar el texto de todos los nodos párrafo se escribe: “//parrafo/text( )” y para seleccionar TODO el texto que cuelga de todos los nodos tipo párrafo: “//parrafo//text( )”

Unos ejemplos basados el XML anterior:

node(): selecciona todos los nodos (elementos y texto).

//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 el contenido del elemento (texto).

//autor/text()	Milan Kundera Mario Vargas Llosa Mario Vargas Llosa
//libro/text()	No devuelve nada porque "libro" no contiene texto.

***: selecciona todos los elementos**

/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

//@*	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

Predicados

Los predicados (predicates) son elementos que en un trayecto de búsqueda permiten restringir el conjunto de nodos seleccionados por un eje, a aquellos que cumplan una cierta condición, de forma que un predicado especifica con mayor detalle la información que se requiere, permitiendo filtrar un conjunto dado. Para la obtención de un predicado se recurre a identificaciones o a las funciones propias de XPath que veremos a continuación escritas entre corchetes ([....]) .

Ejemplos de predicados:

- “/comidas/plato[1]” selecciona el primer plato de comidas.
- “comidas/plato[last()]” lo hace con el último (nótese que no existe la función first() ya que para ello se usa un predicado como plato[1]).
- “/comidas/plato[precio]” obtiene los elementos que tengan elemento precio.
- “comidas/plato[precio=10.50]/precio”, obtiene los precios de los platos con un precio cuyo valor sea 10.50.
- “//*[@num]” selecciona todos los elementos que tengan el atributo num .
- “//capitulo[parrafo/*[@href]]” selecciona todos los capítulos que tengan un párrafo que a su vez tenga algún elemento con atributo href.
- /libro/capitulo[@num="1"]/parrafo, que escoge aquellos elementos parrafo de todos los elementos capitulo que tengan un atributo, llamado num cuyo valor sea 1.

Es importante señalar que los predicados pueden sucederse uno a otro, cosa que tiene el efecto de la operación lógica “Y”. Así para seleccionar todos los capítulos que tengan un párrafo que tenga algún elemento con atributo href y a su vez capitulo tenga el atributo public con valor si se puede escribir de la forma:

- “//capitulo[parrafo/*[@href]][@public='si']”

Una conclusión relevante de lo visto es que el uso tanto de nodos de comprobación como de los predicados pone de manifiesto que XPath es más que un mero mecanismo de selección de varios
nodos a la vez, ya que como se ha visto, este lenguaje tiene la capacidad de individualizar un nodo con características definidas, especialmente mediante predicados incluidos dentro del trayecto.

Ejemplos con respecto del fichero biblioteca.xml

Los predicados se escriben entre corchetes

[@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 rodeadeas 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>

El predicado puede estar situado en cualquier parte de la expresión XPath, no sólo al final del eje.

En el ejemplo siguiente se obtienen los títulos de los libros publicados después de 1970. Para ello, los elementos se seleccionan los elementos <fechaPublicacion> cuyo atributo año es posterior/mayor a 1970, a continuación se seleccionan los elementos padre (/.., es decir los elementos <libro>) y a continuación los subelementos <titulo>

//fechaPublicacion[@año>1970]/../titulo	<titulo>La vida está en otra parte</titulo> <titulo>Pantaleón y las visitadoras</titulo>

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>

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>

Las condiciones pueden hacer referencia al propio elemento, a descendientes o ascendientes, lo que permite obtener el mismo resultado de varias maneras distintas.

En los ejemplos siguientes se obtienen los libros escritos por Mario Vargas Llosa de dos formas distintas:

- en el primer ejemplo se seleccionan los elementos <libro> cuyo subelemento <autor> tiene como contenido la cadena "Mario Vargas Llosa".
- en el segundo ejemplo se seleccionan los elementos <autor> cuyo contenido es la cadena "Mario Vargas Llosa" (para referirse al contenido del propio elemento seleccionado se utiliza el punto ".") y para obtener los elementos <libro> correspondientes se sube de nivel con los dos puntos "/..".

//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>
//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 haya 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>

Sintaxis abreviada (RESUMEN)

Ejes

- ***/***: si está al principio de la expresión, indica el nodo raíz, si no, indica "hijo".

- **//**: indica "descendiente" (hijos, hijos de hijos, etc.).
- **@atributo**: selecciona el atributo.
- **..**: selecciona el elemento padre.
- **|**: permite elegir varios recorridos.

Nodos de comprobación

- **node()**: selecciona todos los nodos (elementos y texto).
- **text()**: selecciona el contenido del elemento (texto).
- *****: selecciona todos los elementos
- **@***: selecciona todos los atributos

Predicados

Los predicados se escriben entre corchetes

- **[@atributo]**: selecciona los elementos que tienen el atributo.
- **[número]**: si hay varios resultados selecciona uno de ellos por número de orden; **last()** selecciona el último de ellos
- **[condicion]**: selecciona los nodos que cumplen la condición. La condición puede utilizar el valor de un atributo (utilizando @) o el texto que contiene el elemento.

  En las condiciones se pueden utilizar los operadores siguientes:

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

  Se pueden escribir varios predicados seguidos, teniendo en cuenta que cada uno restringe los resultados del anterior, como si estuvieran encadenados por la operación lógica and.

4.- Funciones Xpath

Para que XPath resulte más poderoso y útil existe una gran cantidad de funciones que se pueden usar en las expresiones XPath. Algunas de estas funciones se pueden utilizar para retornar nodos y conjuntos de nodos que no pueden ser encontrados por medio de las relaciones normales de hijo/padre y elemento/atributo. También existen funciones para utilizar con cadenas y números, que se pueden usar para recuperar información desde el documento original y para formatearla para la salida.

Las funciones se pueden reconocer fácilmente porque siempre terminan con "()". Algunas funciones necesitan información para funcionar, esta información se pasa en forma de parámetros.

Por ejemplo veremos más adelante una función para cadenas "string-length()", que devuelve la cantidad de caracteres de una cadena. Puede utilizarse de la siguiente manera:

- String-length('Esta es una cadena')

El parámetro es la cadena "Esta es una cadena". La función usará esta información para calcular el resultado que en este caso será 18.

Funciones de nodo

- **name**(). La función name() se usa para obtener el nombre de un nodo. Por ejemplo name(.) devuelve el nombre del nodo de contexto. Además '.' Es el parámetro por defecto por lo cual name()=name(.).
- **node**(). Retorna el propio nodo. No se suele utilizar.
- **procesing-instruction**(). Para retornar instrucciones de procesamiento. Esta función utiliza un parámetro opcional para especificar el nombre.
- **comment**(). La función comment() se usa para devolver comentarios. Hay que tener en cuenta que existen parsers que no pasan los comentarios y por lo tanto esta función no devolverá nunca ningún valor.
- **text**(). Esta función devuelve el contenido PCDATA de un nodo, sin el PCDATA de ningún hijo, si es que hubiera alguno. Por ejemplo: "/comidas/plato[1]/nombre/text()" devuelve: 'Lassagna'.

Funciones posicionales

- position(). Se utiliza para obtener la posición del nodo en un conjunto de nodos. Por ejemplo si tenemos el siguiente ejemplo:

  <nodes>
  <node>a</node>
  <node>b</node>
  <node>c</node>
  </nodes>

  Si queremos el Segundo nodo debemos especificarlo de la siguiente manera:

  "/nodes/node[position()=2]" esto nos devolverá "<node>b</node>".

  Como esta función se utiliza muy a menudo se puede abreviar "/nodes/node[2]" nos devuelve el mismo resultado.

- **last**(). Esta función nos devuelve la posición del último nodo en un conjunto de nodos. En el ejemplo anterior "/nodes/node[position()=last()] nos devolverá "<node>c</node>". No existe una función first() ya que es equivalente a "position()=1".
- **count**(). Retorna la cantidad de nodos en un conjunto. Por ejemplo, para obtener la cantidad de elementos node del XML anterior: "count(/nodes/node)".

Funciones numéricas

- **number**(). La función convierte texto PCDATA en un valor numérico. Por ejemplo si suponemos que tenemos

el siguiente elemento:" <element>256</element>". Para XPath el valor del elemento es una cadena que contiene los caracteres '2', '5', '6'. Para poder tratar el valor como un número debemos utilizar la función: "number(element)".

- **sum**(). Se utiliza para unir todos los valores numéricos en un conjunto de nodos. Por ejemplo si cambianos nuestro ejemplo XML anterior por:

  <nodes>
  <node>1</node>
  <node>2</node>
  <node>3</node>
  </nodes>

  Podemos sumar el valor total de nodes con la siguiente expresión: "sum(/nodes/node)". Si no hubiésemos cambiado el documento XML y aplicamos la expresión al original, la función nos devolvería 'NaN' (not a number). Ya que XPath no es capaz de convertir 'a', 'b', y 'c' a números.

Funciones Booleanas

- **boolean**(). Simplemente evalúa una expresión para verificar si es verdadero o falso utilizando las siguientes reglas:

  - Si el valor en numérico se considera falso si es 0 o el valor especial NaN. Si el número tiene cualquier otro valor (positivo o negativo) se considera verdadero.
  - Si el valor es una cadena se considera verdadero si su longitud es mayor que 0.
  - Si el valor es un conjunto de nodos, se considera verdadero si la colección no está vacía.
  - Cualquier otro tipo de objeto se convierte a booleano dependiendo del tipo de objeto.

  Por ejemplo la expresión "boolean(name)" se evaluará verdadera si existe un nodo name hijo del nodo contexto.

- **not**(). Para poder hacer algo cuando se cumple lo contrario de nuestra expresión. Por ejemplo si en la expresión anterior queremos hacer algo cuando no existe no un nodo hijo name, la expresión a utilizar sería: "not(boolean(name))".
- **true**() y **false**(). XPath nos proporciona estas dos funciones que no admiten parámetros y que siempre devuelven verdadero o falso respectivamente.

Funciones de cadena

Antes de comenzar a ver las distintas funciones resaltar que las funciones de cadena en XPath consideran de manera distinta las mayúsculas y las minúsculas.

- **string**(). La función convierte cualquier valor a cadena. Normalmente la función no es necesaria al leer los datos desde el árbol origen, ya que los datos están todos en formato de texto. Puede ser útil por ejemplo para convertir resultados de cálculos en cadenas de texto.
- **string-length**(). La función devuelve la cantidad de caracteres de la cadena pasada como parámetro incluidos los espacios.
- **concat**(). La función toma como parámetros varias cadenas y devuelve el resultado de concatenarlas. Por ejemplo "concat('esto', ' es ', ' una ', ' cadena')" devolvería la cadena 'esto es una cadena'. Puede ser útil por ejemplo para concatenar distintos CDATA.
- **contains**(). La función indica si una cadena contiene dentro de si misma otra cadena. Por ejemplo "contains('Esto es una cadena', 'es una')" devuelve verdadero.
- **stars-with**(). La función devuelve verdadero si la primera cadena comienza con la segunda cadena pasada como parámetro. "stars-with('Esto es una cadena', 'Esto')" devolvería verdadero.
- **substring**(). La función utiliza tres parámetros. La cadena de la que se van a extraer los caracteres, la posición donde se empieza a tomar los caracteres y por último la cantidad de caracteres que se toman. EL último parámetro es opcional y si se omite se toman todos hasta el final de la cadena. Solo hay que tener en cuenta que a diferencia de en muchos lenguajes de programación en XPath el primer elemento de una cadena es el 1. Por ejemplo la expresión "substring('Esto es la cadena principal', 12)" devuelve 'cadena principal'. Al omitir el

tercer parámetro a tomado desde el carácter 12 (inclusive) hasta el final de la cadena.

- **substring-after**(). La función retorna todos los caracteres de la cadena que están después del carácter pasado como segundo parámetro. Por ejemplo "substring-after('Esto es', 't')" devuelve la cadena 'o es'.
- **substring-before**(). La función devuelve al contrario de la anterior el texto anterior al carácter pasado como parámetro. Por ejemplo "substring-before('Esto es', 't')" devuelve la cadena 'Es'.
- **translate**(). La función resulta un poco más complicada de lo que puede parecer y lo mejor es ver algunos ejemplos:

  translate('12:30', '30', '45') --- > devuelve '12:45'
  translate('12:30', '03', '54') ---- > devuelve '12:45'

Si nos fijamos las dos devuelven el mismo resultado. Lo que realmente hace la función es tomar de los dos últimos parámetros los caracteres por separado y cambia el primero del segundo parámetro por el primero del tercero, el segundo del segundo parámetro por el segundo del tercero etc. De esta manera en el primer ejemplo le estamos diciendo cambia el 3 por un 4 y el 0 por un 5, y en el segundo le decimos cambia el 0 por un 5 y el 3 por un 4 que es lo mismo que en el primer ejemplo pero en orden inverso. Con el siguiente ejemplo puede quedar un poco más claro:

translate('12:30','0123', 'abcd') --- > devuelve 'bc:da'

Podemos utilizar la función para convertir de mayúsculas a minúsculas o viceversa:

translate('convertir', 'abcdefghijklmnopqrstuvwxyz', 'ABCDEFGHIJKLMNOPQRSTUVWXYZ')
translate('CONVERTIR', 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz')

Solo hay que tener en cuenta dos cosas más. Si un carácter no está en el segundo parámetro quedará como en el original: "translate('aBc', 'abc','012')" devuelve '0B2' ya que B no tiene correspondencia en el segundo parámetro.

Por otro lado si el segundo parámetro es más largo que el tercero todos los caracteres del segundo parámetro que no tienen correspondencia en el tercero desaparecen, por ejemplo "translate('aBc','abcB', '012')" devuelve '02'. El carácter 'B' desaparece ya que está en el segundo parámetro pero no tiene correspondencia en el tercero.

Algunos ejemplos:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE libros>
<libros>
   <libro>
      <titulo>XXX</titulo>
      <año>1890</año>
   </libro>
   <libro>
      <titulo>YYY</titulo>
      <año>1850</año>
   </libro>
   <libro>
     <año>1830</año>
  </libro>
</libros>
```

//libro[titulo="XXX"]/año	1890
//libro[not(titulo="XXX")]/año	1850

	1830
concat(//libro[1]/titulo,//libro[2]/año)	XXX1850
//libro[starts-with(titulo,'X')]/año	1890
//libro[contains(año,9)]/año	1890
//libro[position()=last()]/año	1830
count(//libro)	3
count(//libro/titulo)	2

5.- Utilidades de Xpath

Como conclusión vamos a tratar de explicar cuál es la utilidad de XPath dentro del mundo XML. Bien podemos hacer un símil entre la sentencia SELECT de SQL y XPath. Normalmente en la programación estándar de hoy en día se trabaja contra una base de datos relacional. Independientemente del lenguaje de programación utilizamos sentencias SELECT para recuperar
información de la base de datos según el programa la va necesitando. Con XPath y XML pasa lo mismo. XPath no es el fin sino el medio.

Cuando trabajemos con ficheros XML utilizaremos diferentes lenguajes XML (XSLT, XQUERY, etc.) o de programación (JAVA) que utilizarán XPath para poder seleccionar información del fichero XML para poder trabajar sobre ella.

Pero a menudo también utilizamos SQL contra una base de datos fuera de un lenguaje de programación. Bien sea para obtener información inmediata, bien para probar sentencias SQL que luego incorporaremos a nuestra base de datos. Con XPath igual, podemos necesitar en algún momento extraer cierta información de un documento XML o probar diferentes expresiones para luego incorporarlas a alguno de nuestros programas.

También estaremos de acuerdo en que para crear o modificar aplicaciones de gestión (que conecten con bases de datos relacionales) es indispensable conocer SQL. Cuanto mejor conozcamos SQL más fiables y rápidos serán nuestros programas. Por ejemplo, una persona que conozca la sentencia SELECT como 'SELECT * FROM' y no conozca el WHERE o el ORDER BY, estaremos de acuerdo en que por muy bien que pueda programar los programas serán pobres y difíciles de mantener. Ya que, siempre tendrá que filtrar y ordenar en el lado cliente.

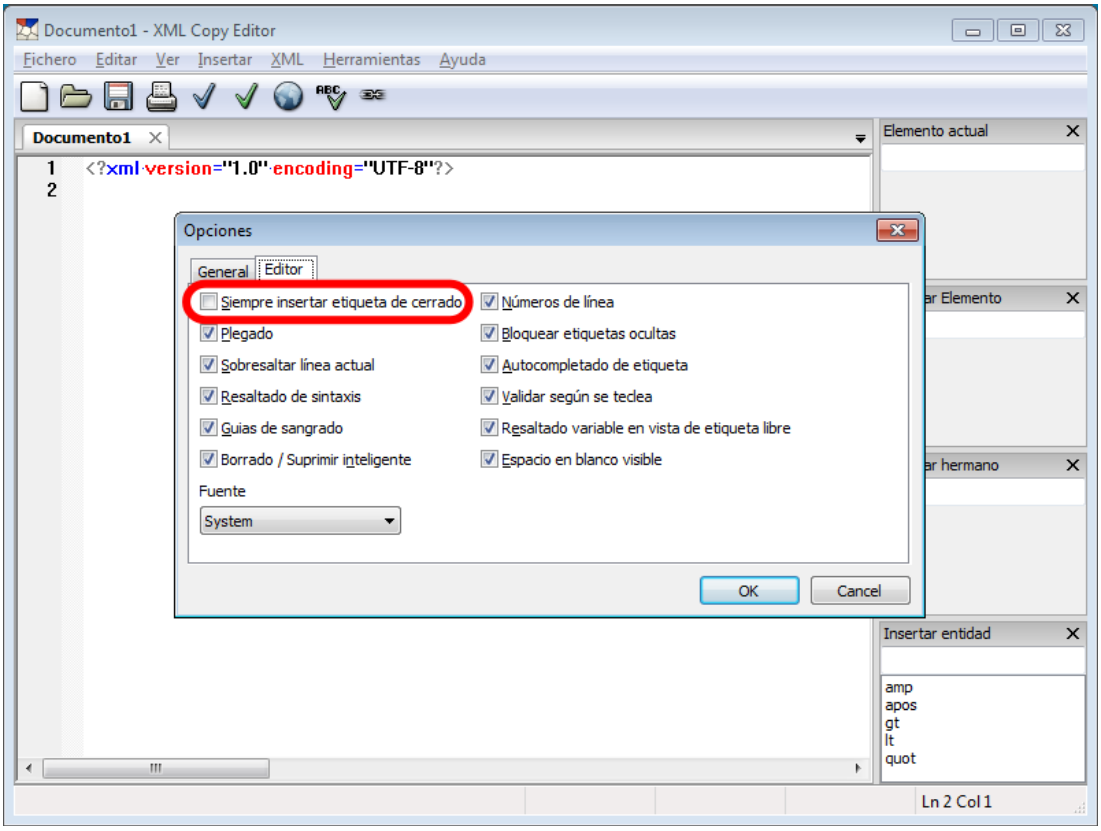
Con XPath igual, cuanto mejor lo conozcamos más eficientes serán nuestras aplicaciones y más sencillas de mantener. Si podemos evitar recorrer un árbol XML entero extrayendo solo la información que necesitamos en cada momento menos trabajo de programación, menos líneas de código, menos probabilidad de error en el programa.

6.- Probar expresiones Xpath

Para probar expresiones XPath vamos a utilizar la aplicación XML Copy Editor. Pero si utilizas un equipo Mac puedes instalar XMLSpear**.**

Veamos antes unos ajustes con XML Copy Editor:

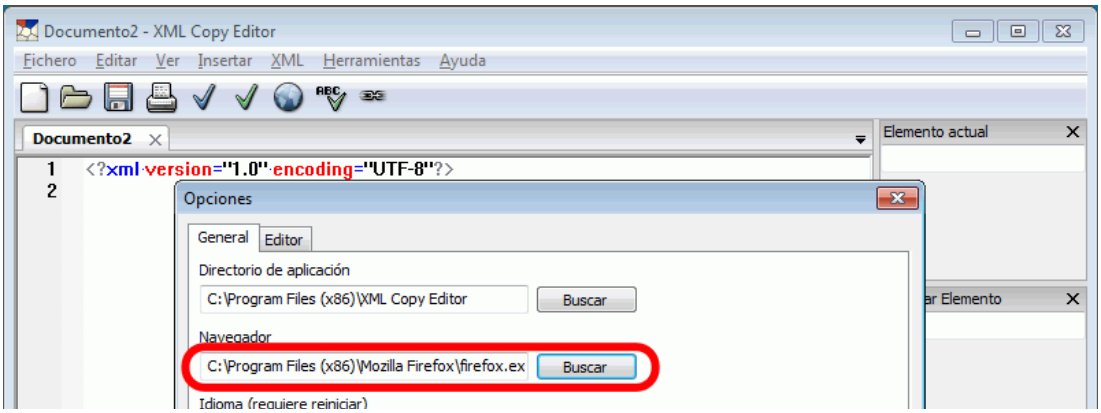
El menú **Herramientas > Opciones > Editor** permite configurar el comportamiento de XML Copy Editor al editar archivos XML. Personalmente, yo prefiero desmarcar la casilla "Siempre insertar etiqueta de cerrado", pero es opción que dependerá de tu forma de trabajo, en esta pantralla puedes marcar y desmarcar alguna que otra opción:

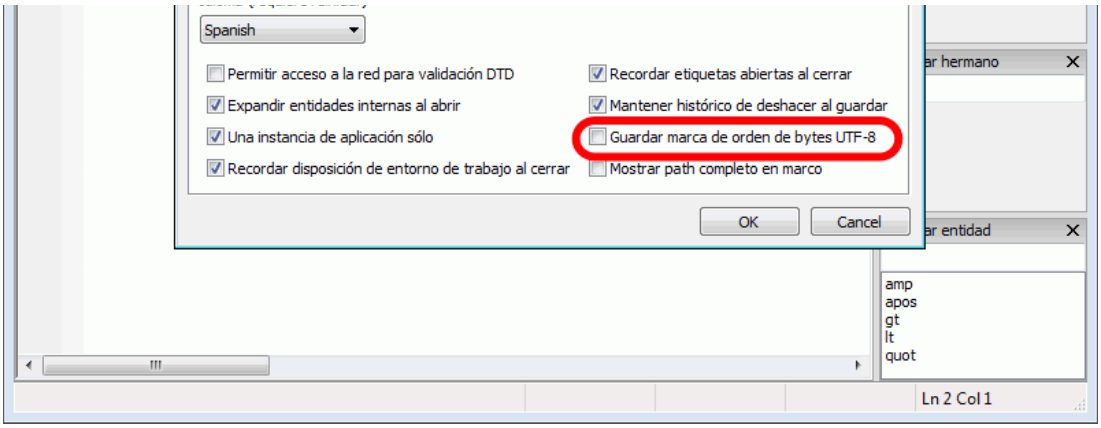


El menú **Herramientas > Opciones > General** de las versiones 1.2.0.7 es distinto a los posteriores:

La versión XML Copy Editor 1.2.0.7 permitía 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 (en la captura siguiente se ha cambiado a Firefox).

Se aconseja 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.





Juego de caracteres

La declaración xml indica el juego de caracteres del documento. Dos de los juegos de caracteres más habituales son iso-8859-1 o utf-8, que se indicarían con las siguientes declaraciones xml:

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

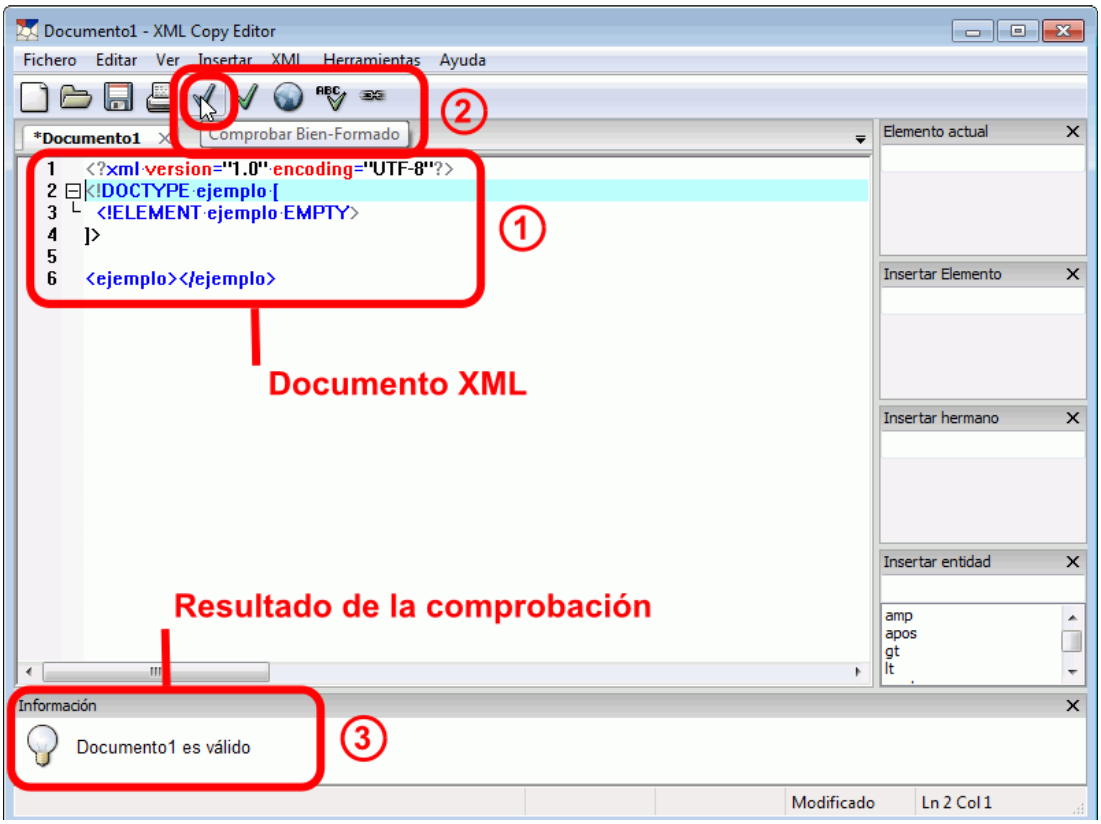
<?xml version="1.0" encoding="utf-8"?>

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.

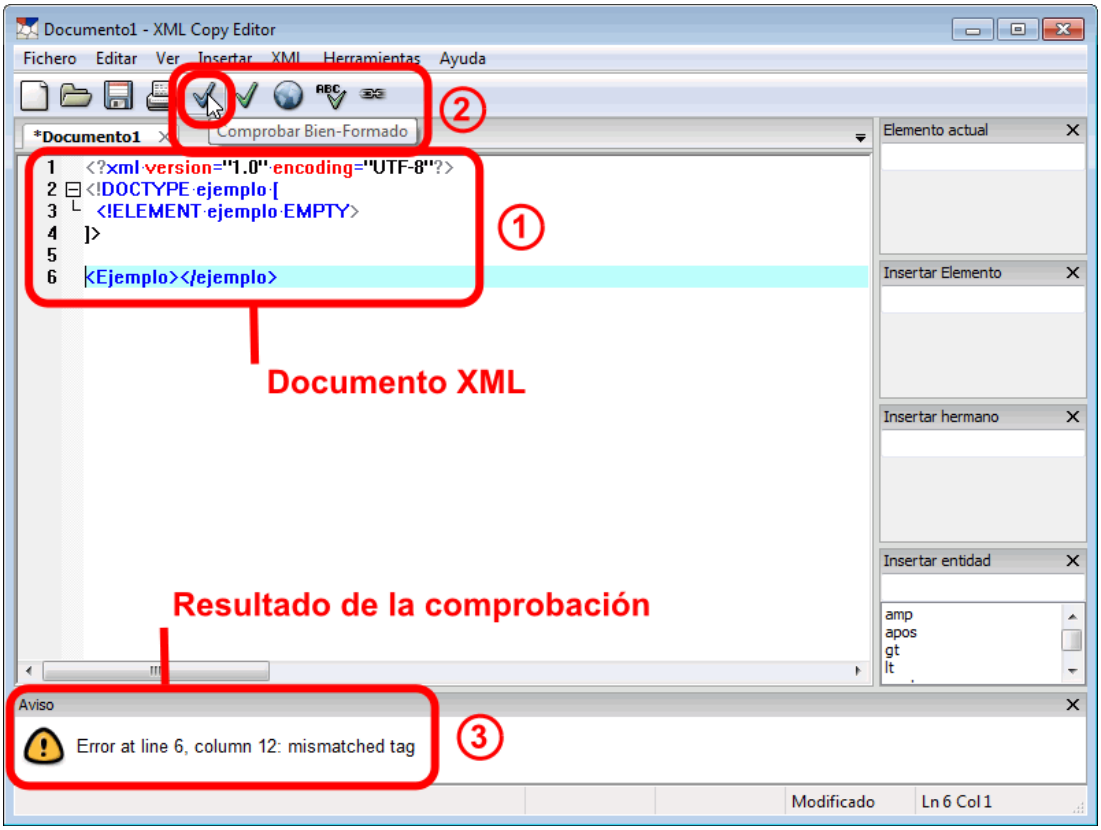
XML Copy Editor tiene en cuenta el juego de caracteres indicado en la declaración. Si se modifica la declaración, al guardar el documento se guarda en el juego correspondiente. Pero hay que tener en cuenta que otros editores, como el bloc de notas de Windows, no lo hace.

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:

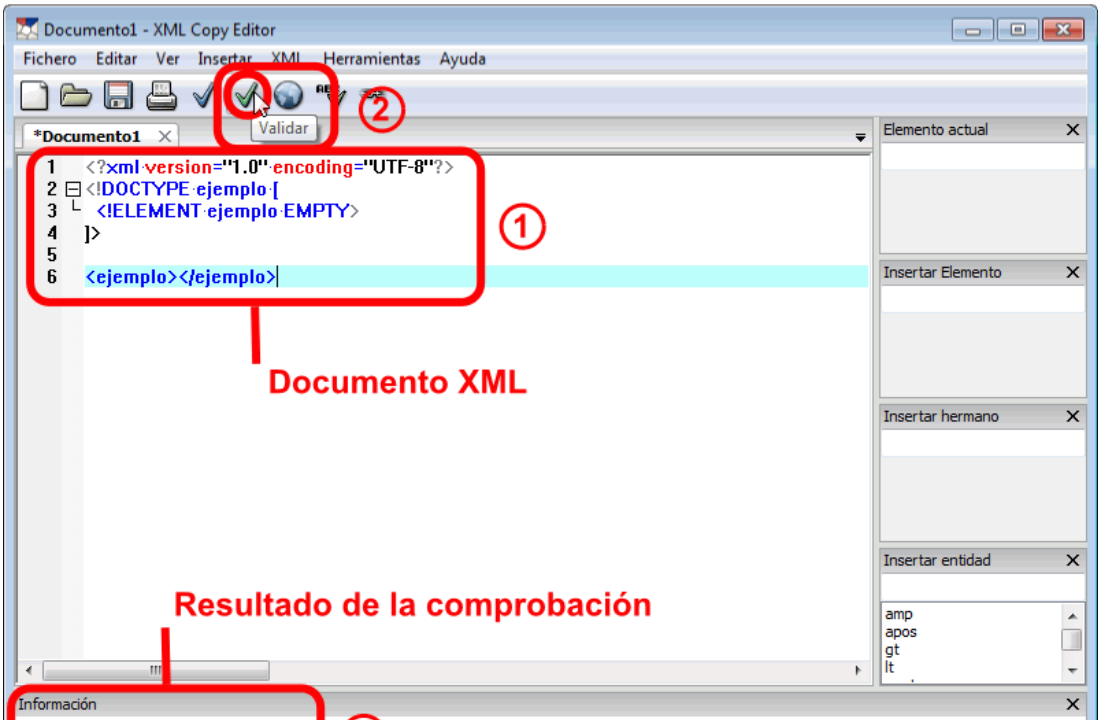


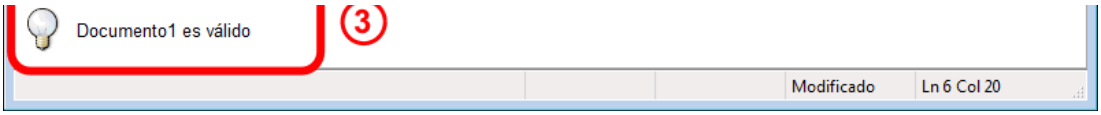
En caso de que el documento no esté bien formado, el programa lo indica en la parte inferior de la pantalla, como muestra la imagen siguiente. El mensaje inferior explica el tipo de error detectado y la línea en el que se ha detectado (en el ejemplo, que las etiquetas no coinciden). Si el documento contiene 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.



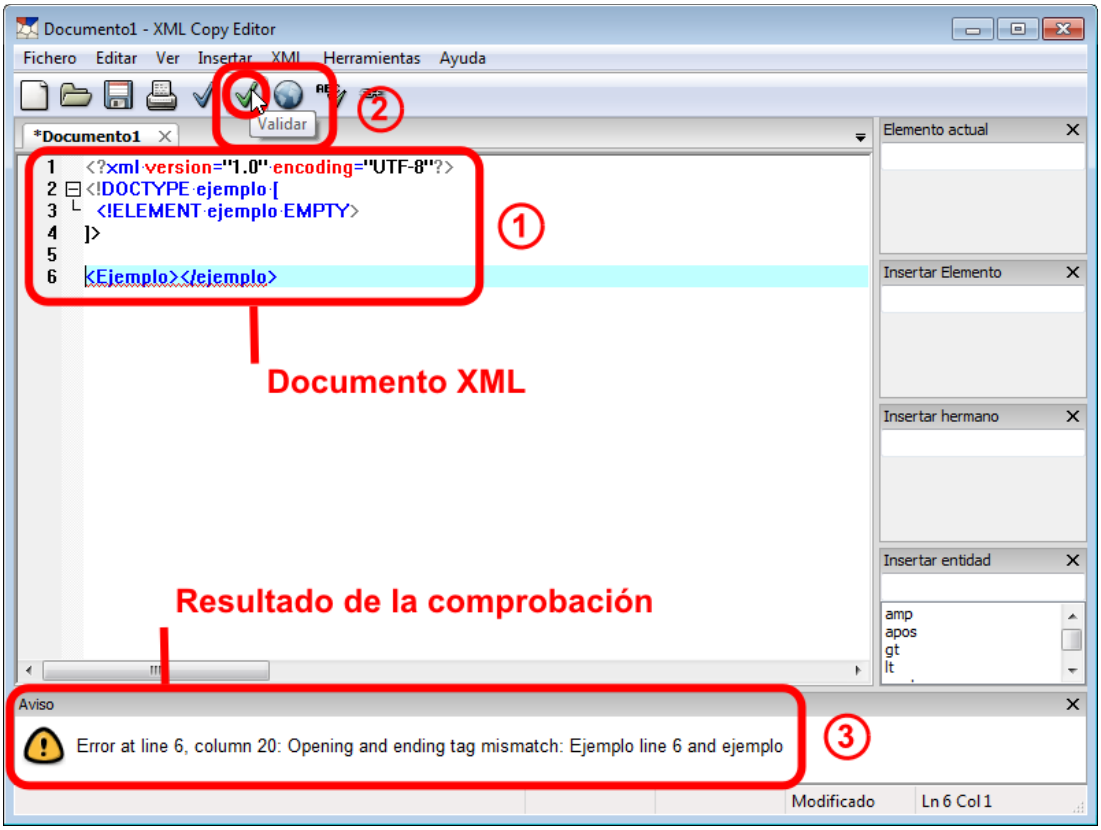
Comprobar que un documento es válido

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:



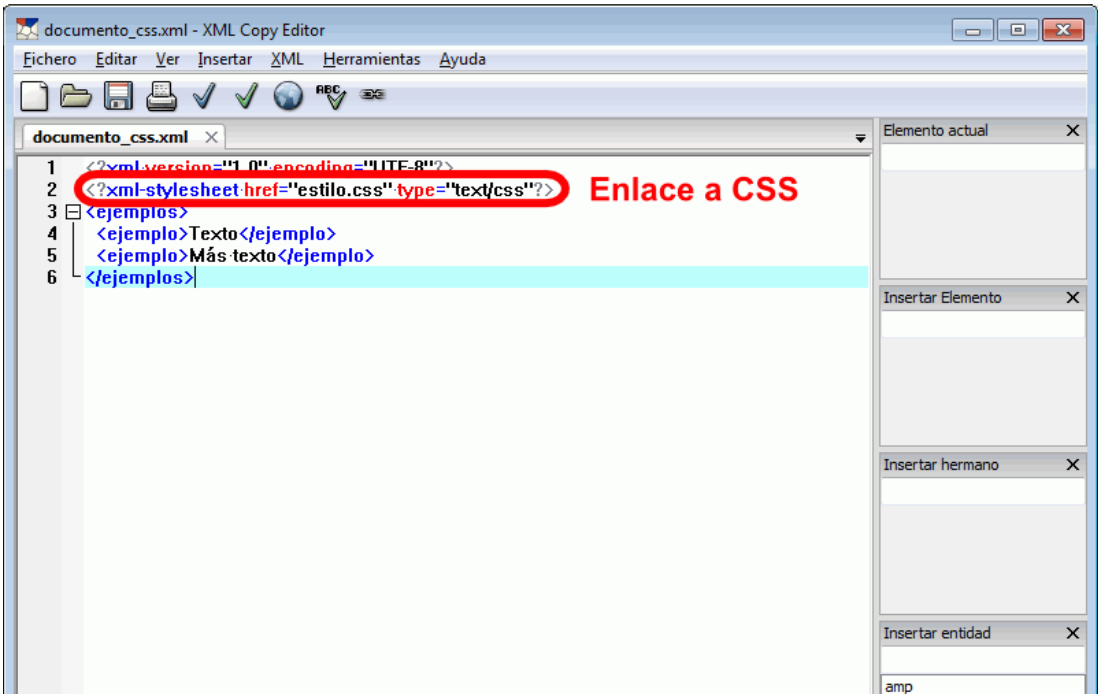


En caso de que el documento no sea válido, el programa lo indica en la parte inferior de la pantalla, como muestra la imagen siguiente. El mensaje inferior explica el tipo de error detectado y la línea en el que se ha detectado (en el ejemplo, que la etiqueta contiene texto). Si el documento contiene 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.



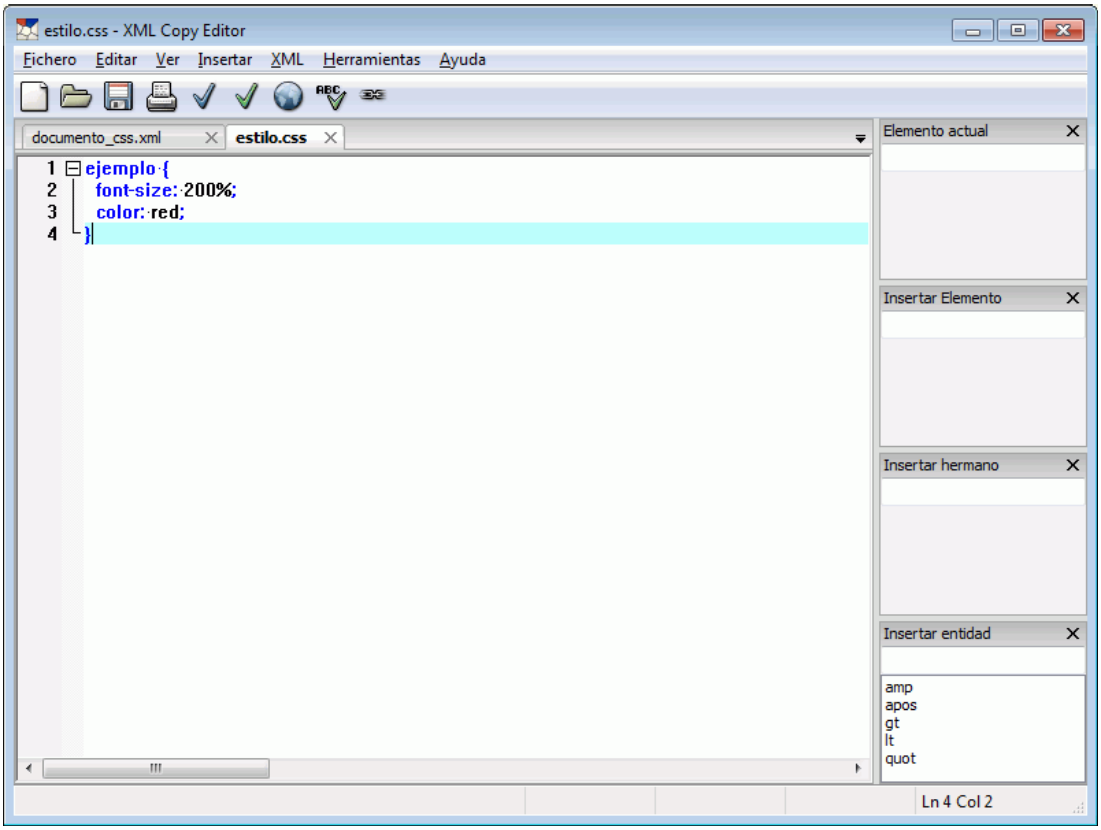
Enlazar una hoja de estilo CSS

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.

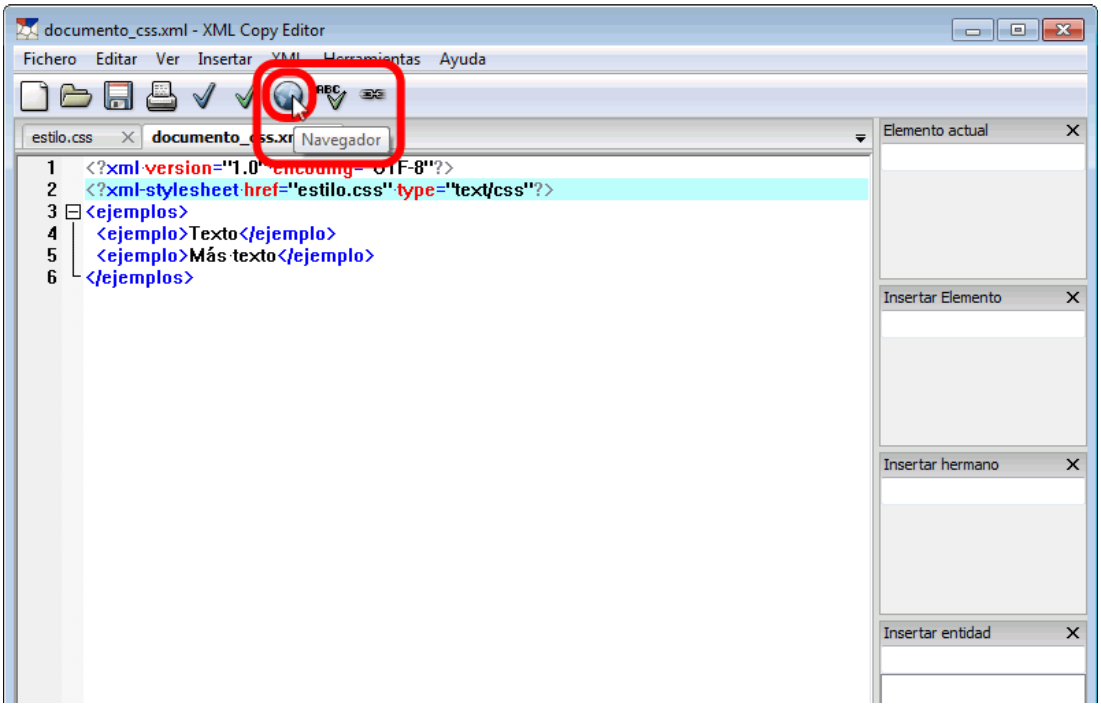


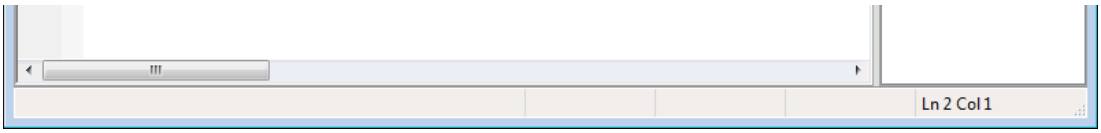
La hoja de estilo puede crearse con cualquier editor de texto sin formato o con el propio XML Copy Editor.

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).

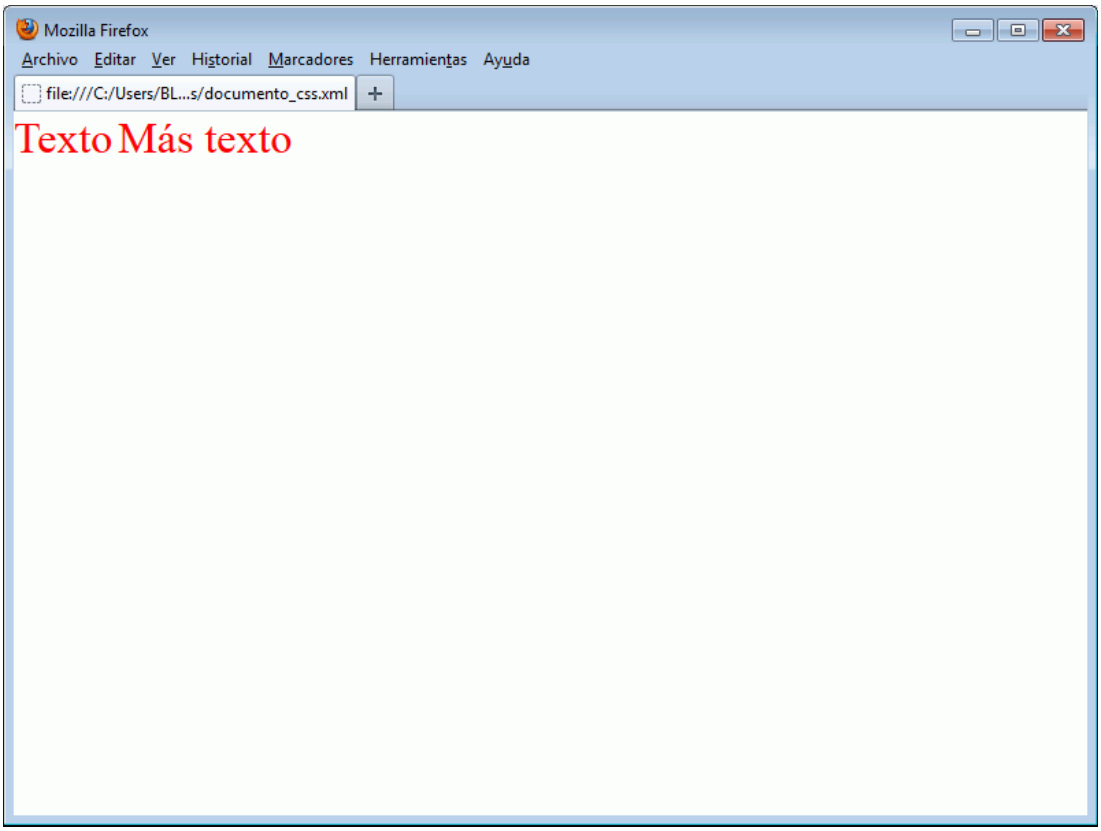


Lo que sí se puede hacer es pedir a XML Copy Editor que abra un documento en el navegador predeterminado:



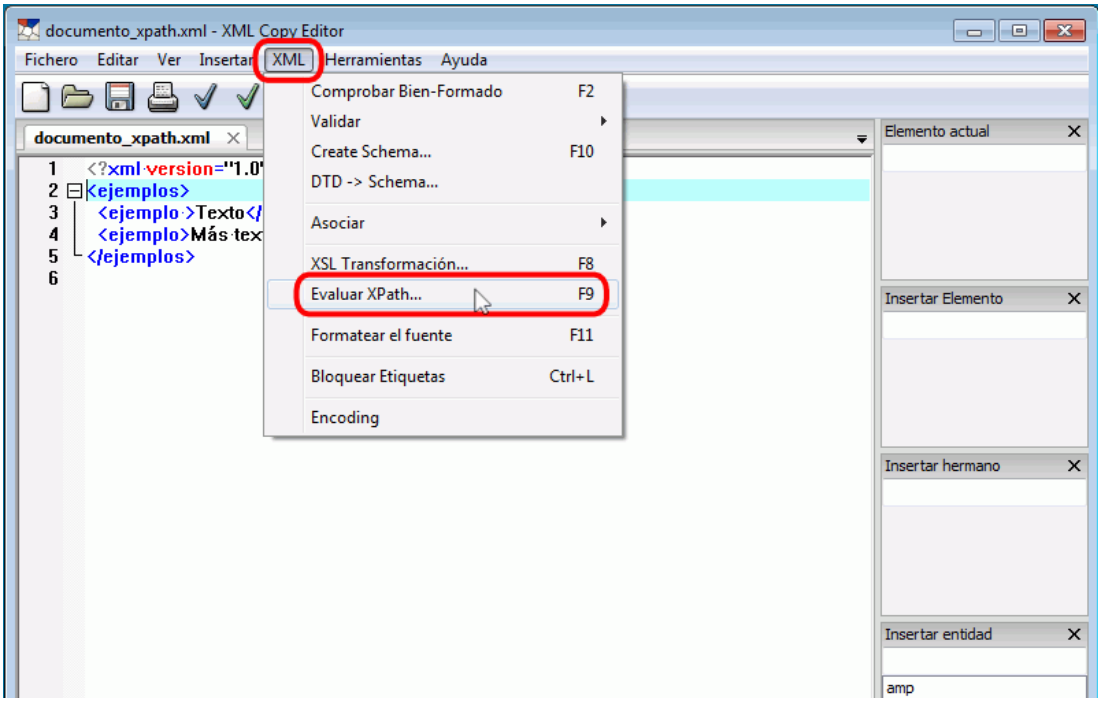


... y comprobar que se aplica la hoja de estilo:

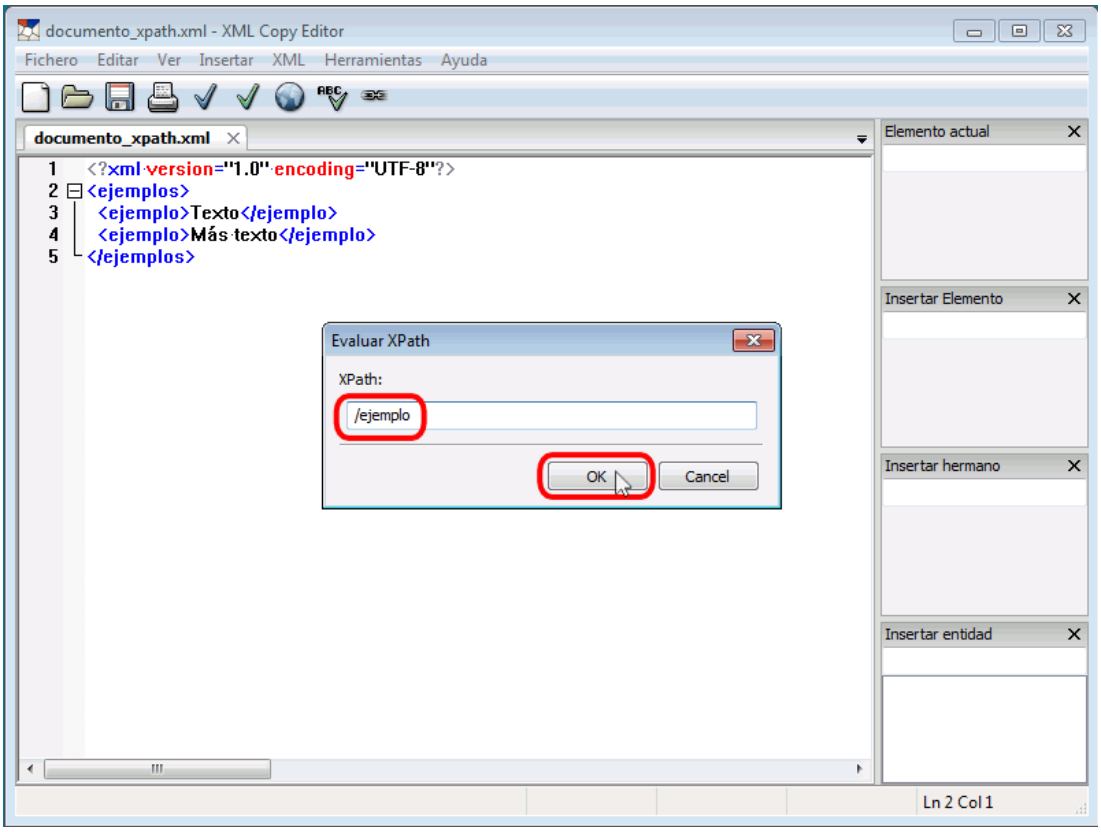


Evaluar una expresión XPath

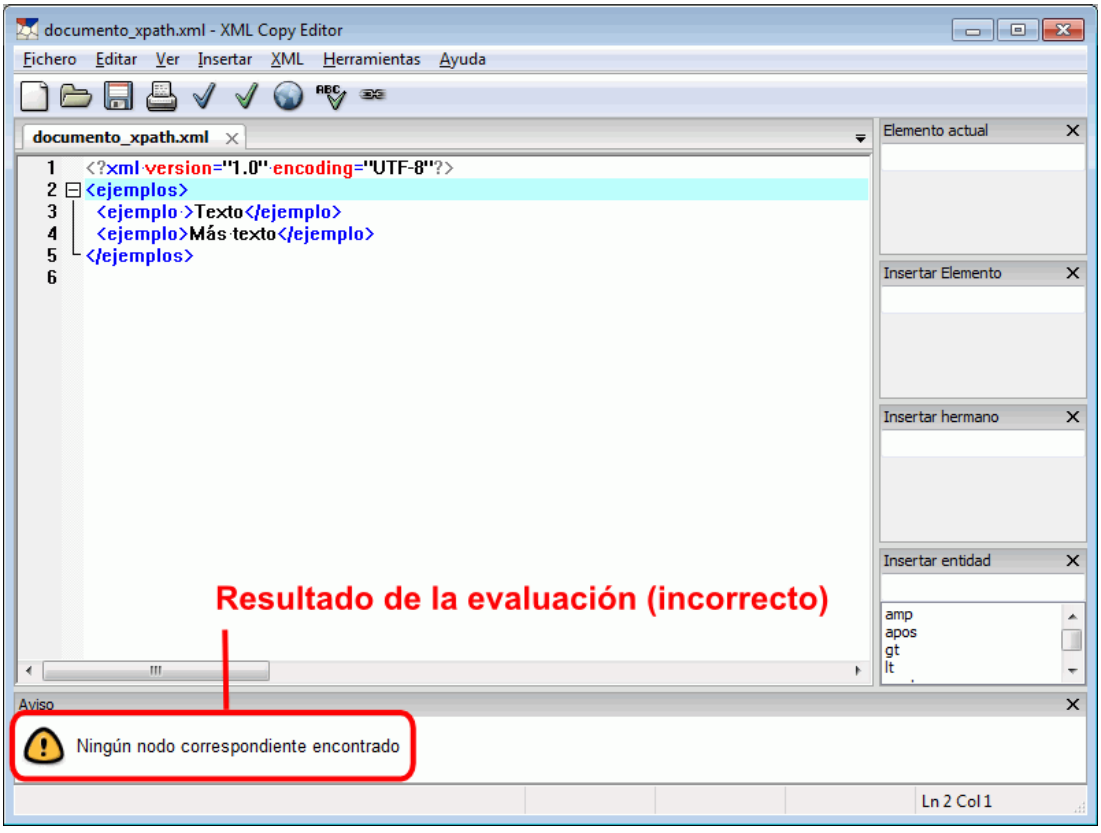
Para evaluar una expresión XPath en un documento XML se puede elegir el menú XML > **Evaluar XPath**... o pulsar la tecla **F9** .



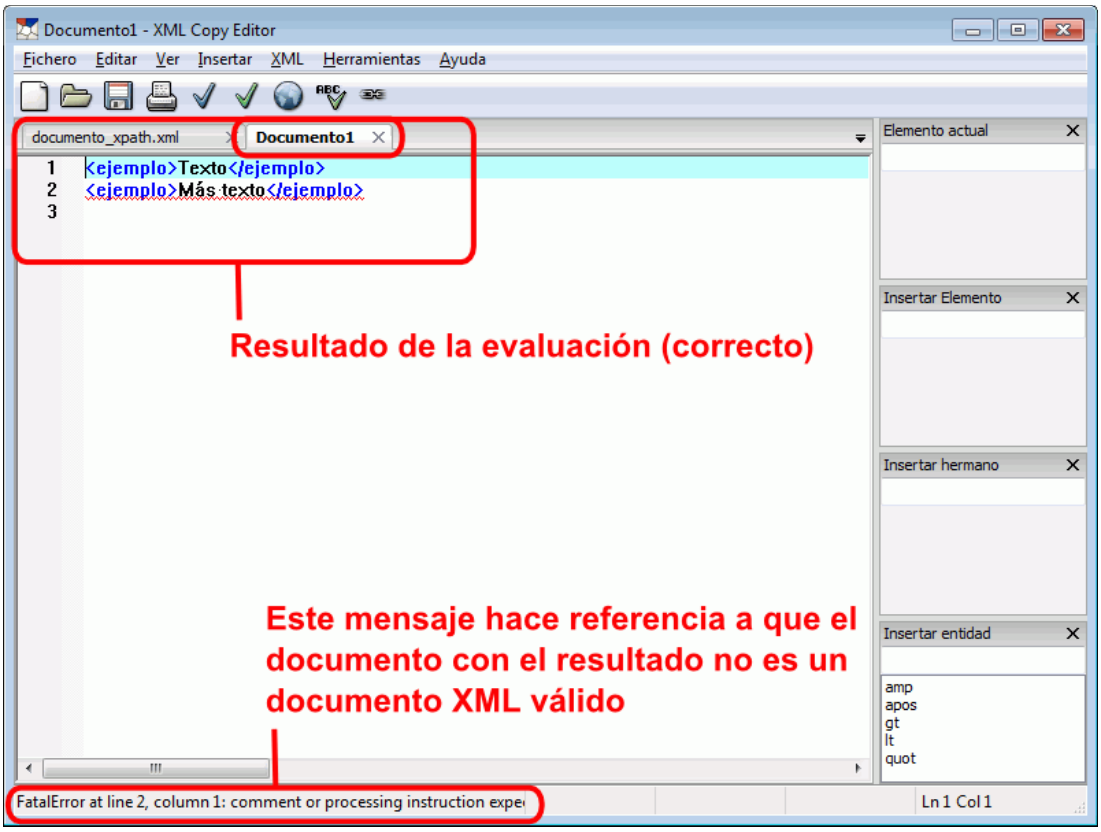
A continuación se introduce la expresión XPath a evaluar y se hace clic en OK:



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.



7.- Ejercicios

XPath - Ejercicio 1 - Expresiones simples

Dado el siguiente documento XML, escriba las expresiones XPath que devuelvan la respuesta deseada (mostrada en los cuadros).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ies>
   <nombre>IES Camp de Morvedre</nombre>
   <web>http://www.iescamp.es</web>
   <ciclos>
      <ciclo id="ASIR">
         <nombre>Administración de Sistemas Informáticos en Red</nombre>
         <grado>Superior</grado>
         <decretoTitulo año="2009"/>
      </ciclo>
      <ciclo id="DAW">
         <nombre>Desarrollo de Aplicaciones Web</nombre>
         <grado>Superior</grado>
         <decretoTitulo año="2010"/>
      </ciclo>
      <ciclo id="SMR">
         <nombre>Sistemas Microinformáticos y Redes</nombre>
         <grado>Medio</grado>
         <decretoTitulo año="2008"/>
      </ciclo>
   </ciclos>
</ies>
```

Enunciado	Resultado
1.1.- Nombre del Instituto:	<nombre>IES Camp de Morvedre</nombre>
1.2.- Página web del Instituto:	http://www.iescamp.es
1.3.- Nombre de los Ciclos Formativos:	Administración de Sistemas Informáticos en Red Desarrollo de Aplicaciones Web Sistemas Microinformáticos y Redes
1.4.- Siglas por las que se conocen los Ciclos Formativos:	id="ASIR" id="DAW" id="SMR"
1.5.- Años en los que se publicaron los decretos de título de los Ciclos Formativos:	año="2009" año="2010" año="2008"
1.6.- Ciclos Formativos de Grado Medio (se trata de obtener el elemento <ciclo>	<ciclo id="SMR"> <nombre>Sistemas Microinformáticos y Redes</nombre> <grado>Medio</grado>

completo):	<decretoTitulo año="2008"/> </ciclo>
1.7.-. Nombre de los Ciclos Formativos de Grado Superior:	<nombre>Administración de Sistemas Informáticos en Red</nombre> <nombre>Desarrollo de Aplicaciones Web</nombre>
1.8.- Nombre de los Ciclos Formativos anteriores a 2010:	Administración de Sistemas Informáticos en Red Sistemas Microinformáticos y Redes
1.9.- Nombre de los Ciclos Formativos de 2008 o 2010:	Desarrollo de Aplicaciones Web Sistemas Microinformáticos y Redes

XPath - Ejercicio 2 - Expresiones simples

Dado el siguiente documento XML, escriba las expresiones XPath que devuelvan la respuesta deseada (mostrada en los cuadros).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ies>
  <modulos>
    <modulo id="0228">
      <nombre>Aplicaciones web</nombre>
      <curso>2</curso>
      <horasSemanales>4</horasSemanales>
      <ciclo>SMR</ciclo>
    </modulo>
    <modulo id="0372">
      <nombre>Gestión de bases de datos</nombre>
      <curso>1</curso>
      <horasSemanales>5</horasSemanales>
      <ciclo>ASIR</ciclo>
    </modulo>
    <modulo id="0373">
      <nombre>Lenguajes de marcas y sistemas de gestión de información</nombre>
      <curso>1</curso>
      <horasSemanales>3</horasSemanales>
      <ciclo>ASIR</ciclo>
      <ciclo>DAW</ciclo>
    </modulo>
    <modulo id="0376">
      <nombre>Implantación de aplicaciones web</nombre>
      <curso>2</curso>
      <horasSemanales>5</horasSemanales>
      <ciclo>ASIR</ciclo>
    </modulo>
  </modulos>
</ies>
```

2.1.- Nombre de los módulos que se imparten en el Instituto:	Aplicaciones web Gestión de bases de datos Lenguajes de marcas y sistemas de gestión de información Implantación de aplicaciones web

2.2.- Nombre de los módulos del ciclo ASIR:	Gestión de bases de datos Lenguajes de marcas y sistemas de gestión de información Implantación de aplicaciones web
2.3.- Nombre de los módulos que se imparten en el segundo curso de cualquier ciclo:	Aplicaciones web Implantación de aplicaciones web
2.4.- Nombre de los módulos de menos de 5 horas semanales:	Aplicaciones web Lenguajes de marcas y sistemas de gestión de información
2.5.- Nombre de los módulos que se imparten en el primer curso de ASIR:	Gestión de bases de datos Lenguajes de marcas y sistemas de gestión de información
2.6.- Horas semanales de los módulos de más de 3 horas semanales:	4 5 5

XPath - Ejercicio 2 - Expresiones anidadas

Dado el siguiente documento XML, escriba las expresiones XPath que devuelvan la respuesta deseada (mostrada en los cuadros).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ies>
  <nombre>IES Camp de Morvedre</nombre>
  <web>http://www.iescamp.es</web>
  <ciclos>
    <ciclo id="ASIR">
      <nombre>Administración de Sistemas Informáticos en Red</nombre>
      <grado>Superior</grado>
      <decretoTitulo año="2009" />
    </ciclo>
    <ciclo id="DAW">
      <nombre>Desarrollo de Aplicaciones Web</nombre>
      <grado>Superior</grado>
      <decretoTitulo año="2010" />
    </ciclo>
    <ciclo id="SMR">
      <nombre>Sistemas Microinformáticos y Redes</nombre>
      <grado>Medio</grado>
      <decretoTitulo año="2008" />
    </ciclo>
  </ciclos>
  <modulos>
    <modulo id="0228">
      <nombre>Aplicaciones web</nombre>
      <curso>2</curso>
      <horasSemanales>4</horasSemanales>
      <ciclo>SMR</ciclo>
    </modulo>
    <modulo id="0372">
      <nombre>Gestión de bases de datos</nombre>
      <curso>1</curso>
      <horasSemanales>5</horasSemanales>
      <ciclo>ASIR</ciclo>
    </modulo>
    <modulo id="0373">
      <nombre>Lenguajes de marcas y sistemas de gestión de información</nombre>
      <curso>1</curso>
      <horasSemanales>3</horasSemanales>
      <ciclo>ASIR</ciclo>
      <ciclo>DAW</ciclo>
    </modulo>
    <modulo id="0376">
      <nombre>Implantación de aplicaciones web</nombre>
      <curso>2</curso>
      <horasSemanales>5</horasSemanales>
      <ciclo>ASIR</ciclo>
    </modulo>
  </modulos>
</ies>
```

3.1.- Nombre de los módulos del ciclo "Sistemas Microinformáticos y Redes". Nota: en la expresión final no deben aparecer las siglas SMR:	Aplicaciones web
3.2.- Nombre de los ciclos que incluyen el módulo "Lenguajes de marcas y sistemas de gestión de información":	Administración de Sistemas Informáticos en Red Desarrollo de Aplicaciones Web

3.3.- Nombre de los módulos de ciclos de Grado Superior:	Gestión de bases de datos Lenguajes de marcas y sistemas de gestión de información Implantación de aplicaciones web
3.4.- Nombre de los módulos de ciclos cuyo título se aprobó en 2008:	Aplicaciones web Grado de los ciclos con módulos de primer curso: Superior Superior