Qué es el lenguaje XPath

Introducción a XPath

XPath es un lenguaje XML que permite el acceso a información concreta de un documento XML, haciendo un recorrido a través de los elementos que lo compoe. Es decir, evaluar una expresión XPath consiste en buscar si hay elementos que satisfagan el recorrido indicado.

Este lenguaje se suele emplear en otras tecnologías o lenguajes y no en solitario. Por ejemplo, se utiliza en consultas XQuery o en transformaciones XSLT o XSL-FO.

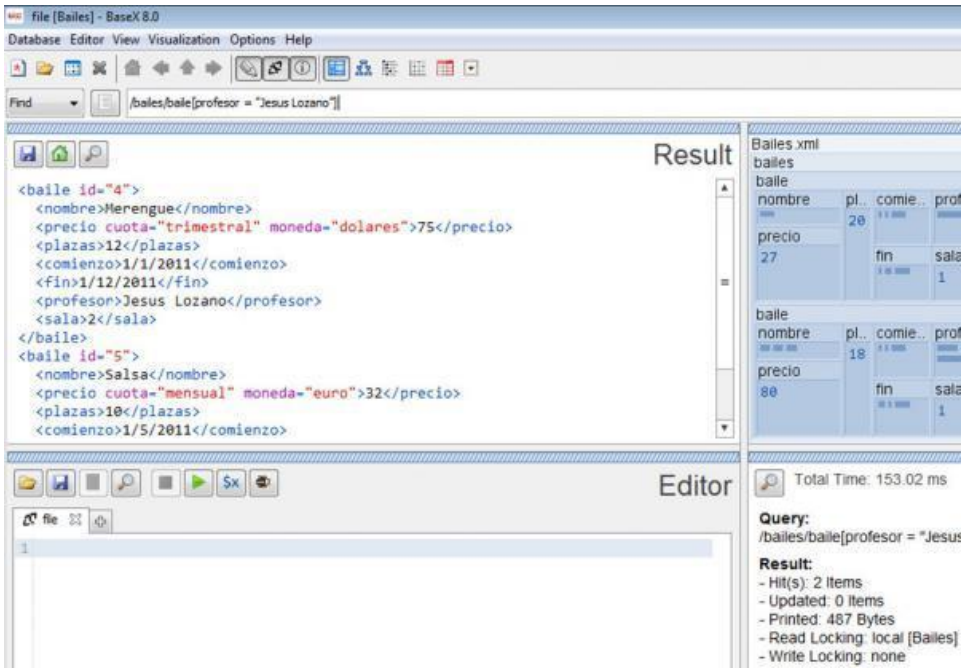
Un documento XML está compuesto por nodos, también llamados elementos, los cuales se componentes de etiquetas y contenido, siendo este contenido texto y otros nodos. Los nodos se organizan en forma de árbol, existiendo un único nodo raíz. A partir de él, todos los nodos pueden o no tener nodos hijos

Para todos los **ejemplos** siguientes vamos a tomar como referencia el siguiente documento XML:

```
<bib>
  <book id="1">
    <title>TCP/IP Illustrated</title>
    <author>Stevens</author>
    <publisher>Addison-Wesley</publisher>
    <year>2002</year>
  </book>
  <book id="2">
    <title>Advanced Programming in the Unix Environment</title>
    <author>Stevens</author>
    <publisher>Addison-Wesley</publisher>
    <year>2004</year>
  </book>
  <book id="3">
    <title>Data on the Web</title>
    <author>Abiteboul</author>
    <author>Buneman</author>
    <author>Suciu</author>
    <year>2006</year>
  </book>
</bib>
```

Software y herramientas online

Para probar XPath podemos utilizar el **software** libre BaseX o cualquier otro que comentamos aquí.



Y de manera online podemos utilizar:

- http://www.xpathtester.com/xpath

Direccionamiento o localización

El **direccionamiento** o **localización** es una ruta o camino de nodos de un documento XML que nos permitirán seleccionar un conjunto de ellos.

El direccionamiento puede ser:

- **Abosoluto**: si siempre se incluye el nodo raíz.
- **Relativo**: si se empieza a hacer referencia a los nodos desde un nodo diferente al nodo raíz.

Las **expresiones** que se pueden utilizar en un direccionamiento XPath son:

nodo	Elemento de nombre *nodo*
/nodo	El *nodo* se encuentra en la raíz del documento
nodo1/nodo2	El *nodo2* es hijo directo de *nodo1*
nodo1//nodo2	El *nodo2* es hijo del *nodo1* pero puede haber nodos intermedios
//nodo	El *nodo* está ubicado en cualquier nivel debajo del nodo raíz
@atributo	Atributo de nombre *atributo*
*	Cualquier elemento
@*	Cualquier atributo
.	Nodo actual
..	Nodo padre

Pongamos algunos ejemplos de su uso.

Para listar los títulos de los libros de la biblioteca:

```
/bib/book/author
```

Para listar los autores sin indicar de qué nodo son hijos:

```
//author
```

Para listar los identificadores de los libros:

```
/bib/book/@id
```

Para listar todos los atributos de cualquier nodo:

```
//@*
```

El acceso a **otras partes del nodo** se puede realizar con las siguientes funciones:

- node(): Devuelve el nodo completo. Es el comportamiento por defecto.
- text(): Devuelve el texto del nodo.

Por ejemplo, para listar sólo los títulos de los libros de la biblioteca sin las etiquetas:

```
/bib/book/author/text()
```

Filtrar el acceso a elementos

El direccionamiento de XPath también permite filtrar el conjunto de nodos o información a la que se accede mediante la consulta utilizando condiciones en nodos. El filtro se especifica mediante **corchetes [ ]** seguidos del nodo al que se le aplica dicho filtro.

Podemos utilizar los siguientes **operadores**:

and	
or	
not	
=	

!=	
<	
>	
<=	
>=	
+	Suma
-	Resta
*	Multiplicación
div	División
mod	Resto de la división
|	Unión de resultados

.

Funciones numéricas

round()	Redondeo	round(3.14) = 3
abs()	Valor absoluto	abs(-7) = 7
floor()	Redondeo inferior	floor(7.3) = 7
ceiling	Redondeo superior	ceiling(7.3) = 8

Funciones de cadena

substring()	Subcadena	substring('TicArte', 1, 4) = TicA
starts-with()	Cadena comienza por	starts-with('XML', 'X') = true
ends-with()	Cadena finaliza por	ends-with('XML', 'X') = false
contains()	Cadena contiene	contains('XML', 'ML') = true
normalize-space()	Espacios normalizados	normalize-space(' Doc XML ') = 'Doc XML'
translate()	Cambia caracteres en una cadena	translate('Doc XML', 'Doc', 'File') = 'File XML'
string-length()	Longitud de una cadena	string-length('TicArte') = 7
upper-case()	Cadena a mayúsculas	upper-case('xml') = 'XML'
lower-case()	Cadena a minúsculas	lower-case('XML') = 'xml')

Funciones de posición de elementos

position() = n	Nodo que se encuentra en la posición 'n'
elemento[n]	Nodo en la posición 'n' de los que se llaman *nodo*
last()	El último nodo de un conjunto
last() - i	El último menos i nodos

Funciones que devuelven nodos

name()	Nombre del nodo actual
root()	Elemento raíz
node()	Nodos descendientes del actual
comment()	Comentarios del nodo
processing-instruction()	Instrucciones de procesamiento
exist()	Si existe el nodo o no
empty()	Si el nodo está vacío o no

Funciones de agregado

count()	Contar los nodos
avg()	Media del contenido de los nodos
max()	Valor máximo del contenido de los nodos
min()	Valor mínimo del contenido nodos
sum()	Suma del contenido de los nodos

A continuación exponemos una serie de ejemplos que ayuden a clarificar el uso de filtros.

Mostrar todos los autores y publicadores. Utilizaremos el operador *unión* para unir los dos conjuntos de nodos:

```
/bib/book/author|/bib/book/publisher
```

Suponiendo que el documento XML tiene libros y diccionarios, podemos ver todos los autores:

```
/bib/(book|dictionary)/author
```

Mostrar el libro número 2:

```
/bib/book[2]
/bib/book[position()=2]
```

Mostrar del libro número 2 al número 5:

```
/bib/book[2 to 5]
```

Mostrar el último libro:

```
/bib/book[last()]
```

Mostrar los libros anteriores al año 2003. Los números no necesitan comillas.

```
/bib/book[year<2003]
```

Mostrar los libros cuyo autor sea Stevens. Las cadenas deben ir siempre entre comillas. Se muestran diferentes formas de conseguir lo mismo.

```
/bib/book[author="Stevens"]
/bib[book/author="Stevens"]/book
//author[.="Stevens"]/..
```

Mostrar el título de los libros cuyo autor sea Stevens. Hay que fijarse que la condición va en el nodo *book* pero luego mostramos el nodo *title* que contiene.

```
/bib/book[author="Stevens"]/title
```

Mostrar los títulos de los libros cuyo identificador sea el 2:

```
/bib/book[@id="2"]/title
```

Mostrar los títulos de los libros de Stevens publicados en el 2002. Podemos utilizar el operador *and*.

```
/bib/book[author="Stevens" and year=2002]/title
```

Mostrar los títulos de los libros de Stevens publicados en el 2002. Podemos concatenar diferentes condiciones, cada una en sus corchetes, que es el mismo significado que el operador and.

```
/bib/book[author="Stevens"][year=2002]/title
```

Mostrar los títulos de los libros que comienzan por 'T':

```
/bib/book[starts-with(author,"T")]
```

Mostrar los títulos de los libros que tengan una longitud de 20 caracteres:

```
/bib/book[string-length(title,20)]/title
```

Consultas XPath anidadas

Las **consultas XPath anidadas** consisten en incluir una consulta XPath que devuelta un cierto valor dentro de la condición de otra consulta XPath.

Por ejemplo, mostrar todos los títulos de los libros del autor que escribió "TCP/IP Illustrated". Si nos fijamos, en ningún caso nos están diciendo el nombre del autor, por lo que tendremos que hacer una consulta previa para localizarlo. Los pasos serían los siguientes:

1) Encontrar primero al autor del libro "TCP/IP Illustrated", pero mostrando sólo el texto sin las etiquetas. Esta consulta nos devolverá *Stevens*.

```
/bib/book[title="TCP/IP Illustrated"]/author/text()
```

2) A continuación debemos encontrar todos los libros que escribió dicho autor, en nuestro caso *Stevens*.

```
/bib/book[author="Stevens"]/title
```

3) Pero claro, la consulta anterior no es la correcta, porque no podemos usar el nombre de *Stevens*. Tendremos que sustuir la consulta 1 dentro de la consulta 2, pero eliminando las comillas dobles:

```
/bib/book[author=/bib/book[title="TCP/IP
Illustrated"]/author/text()]/title
```