Funciones XPath

Para XPath existen gran cantidad de funciones que se pueden usar en sus expresiones. 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.

Ejemplo:
Función para cadenas "string-length()" que devuelve la cantidad de caracteres de una cadena.
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

node(). Retorna el propio nodo.

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'.

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(.).

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.

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>b</node>
    <node>b</node>
    <node>b</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 tenemos:

```
<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 ejemplo anterior, la función nos devolvería 'NaN' (not a number). Ya que XPath no es capaz de convertir 'a', 'b', y 'c' a números.

ceiling(). Evalúa el número decimal y devuelve el número entero más pequeño que resulta mayor que el paràmetro.
Ejemplo: ceiling(4.2) dará como resultado 4.

round(). Redondea el número decimal pasado como parámetro. Devolverá el número entero más cercano al decimal.

Funciones Booleanas

boolean().Evalúa una expresión para verificar si es verdadero o falso segun las siguientes reglas:

- Si el valor es 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.

not(). Para poder hacer algo cuando se cumple lo contrario de nuestra expresión.

true() y **false()**. XPath nos proporciona estas dos funciones que no admiten parámetros y que siempre devuelven verdadero o falso respectivamente.

Funciones de cadena

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.

starts-with(). La función devuelve verdadero si la primera cadena comienza con la segunda cadena pasada como parámetro.
"starts-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 primero ejemplo le estamos diciendo cambia el 3 por un 4 y el 0 por un 5, y en el segundo le decimos cambial 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')
```

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.

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.