

UT5: XPath

Objetivos de UT5

- Describir la información transmitida en los documentos XML y sus reglas.
- Identificar las tecnologías relacionadas con la definición de documentos XML.
- **Analizar la estructura y sintaxis específica utilizada en la descripción.**
- Crear descripciones de documentos XML.
- **Utilizar descripciones en la elaboración y validación de documentos XML.**
- Asociar las descripciones con los documentos.
- Utilizar herramientas específicas.
- Documentar las descripciones.

Índice

Objetivos de UT5	1
Introducción a XPath.....	2
Expresiones XPath. Nodos y valores atómicos	2
Expresiones XPath. Sintaxis	4
Expresiones XPath. Tipos de datos	5
Expresiones XPath. Ejes (AXES).....	6
Expresiones XPath. Nodos de comprobación	6
Expresiones XPath. Predicados.....	7
Expresiones XPath. Operadores.....	7
Expresiones XPath. Funciones	7
Ejemplo.....	9

Introducción a XPath

XPath es una recomendación de W3C (3.0 de 2014) y el mayor de los 4 lenguajes que forman XSL (Lenguaje para hojas de eStilo eXtensibles). Los otros lenguajes son **XSLT**, XSL-FO (discontinuado) y **XQuery**.

La principal capacidad de XPath es navegar a por los elementos y los atributos del XML, utilizando una sintaxis de rutas (*path-like*) similar a la de los sistemas de archivos. Además, cuenta con más de 200 funciones nativas para manejar strings, números, booleanos, comparar fechas, manipulación y secuenciación de nodos, etc.

Las expresiones XPath pueden utilizarse en JavaScript, Java, **XSD**, **XSLT**, PHP, Python, C....

Herramienta para probar: <https://codebeautify.org/Xpath-Tester> o desde XML Copy Editor (XML→Evaluar XPath ó F9).

Expresiones XPath. Nodos y valores atómicos

Todas las expresiones XPath devuelven una secuencia que puede estar vacía o compuesta por conjuntos de ítems. Los ítems de XML que pueden ser **nodos** o **valores atómicos**.

- En XPath, hay 7 tipos de **nodos**: **raíz**, **elemento**, **atributo**, **texto**, **espacio de nombres**, **instrucción de procesamiento** y **comentario**. Los elementos y los atributos pueden tener un tipo base que debe validarse contra un esquema.
 - **Raíz**. Cada documento tiene exactamente un nodo raíz (*root*), que es la raíz del árbol.
Este nodo contiene un hijo del tipo nodo de comentario por cada comentario que se encuentre fuera del elemento raíz del documento.
Contiene un hijo del tipo nodo de instrucción de procesamiento para cada instrucción de procesamiento externa al elemento raíz.
Contiene un hijo del tipo nodo de elemento para el elemento raíz del documento.
El nodo raíz no tiene nodo padre.
El valor del nodo raíz es el valor del elemento raíz.
 - **Elemento**. Un nodo de elemento (*element*) tiene un nombre, un URI de espacio de nombre, un nodo padre y una lista de nodos hijos que puede incluir otros nodos de elemento, nodos de comentario, nodos de instrucción de procesamiento y nodos de texto.
Un nodo de elemento también tiene una colección de atributos y una colección de espacios de nombre. No se consideran hijos.
El valor de un nodo de elemento es el texto incluido entre las etiquetas de inicio y de cierre que permanezca después de haberse eliminado todas las etiquetas, comentarios e instrucciones de procesamiento.
 - **Atributo**. Un nodo de atributo (*attribute*) tiene un nombre, un URI de espacio de nombre, un valor y un elemento padre.
Los elementos son padres del atributo pero los atributos no son hijos de sus elementos padres.
Un nodo de atributo tiene un nodo padre pero no tiene hijos.
Los atributos xmlns no se representan como nodos de atributo.
El valor de un nodo de atributo es el valor del atributo.
 - **Texto**. Cada nodo de texto (*text*) representa el texto entre etiquetas, instrucciones de procesamiento y comentarios.
Un nodo de texto tiene un nodo padre pero no tiene hijos.
El valor de un nodo de texto es el texto del nodo.
 - **Espacio de nombre**. Un nodo de espacio de nombre (*namespace*) representa un espacio de nombre en el ámbito de un elemento.
Cada declaración de espacio de nombre mediante un atributo xmlns produce un nodo de espacio de nombre en dicho elemento y en todos sus descendientes.
Cada nodo de espacio de nombre tiene un elemento padre, pero no es hijo.
El nombre de un nodo de espacio de nombre es el prefijo.

- **Instrucción de procesamiento.** Un nodo de instrucción de procesamiento (*processing-instruction*) tiene un destino, datos, un nodo padre y ningún hijo.
El nombre del nodo de instrucción de procesamiento es su destino.
El valor del nodo de instrucción de procesamiento son los datos de la instrucción de procesamiento.
- **Comentario.** Un nodo de comentario (*comment*) representa un comentario.
Tiene un nodo padre y no tiene hijos.
El valor de un comentario es el contenido del mismo, sin incluir `<!--` y `-->`.
Los nodos atributos y texto no pueden tener descendientes, pero se diferencian en que los elementos texto son considerados como nodos hijos de la etiqueta.
- Los **valores atómicos** son los "datos" que aparecen asociados a los atributos o el contenido de los nodos texto. Son cadenas, número o booleanos.
- La declaración XML y la declaración de tipo de documento no se incluyen en la vista XPath de un documento XML.
- Todas las referencias de entidad, referencias de carácter y secciones CDATA se resuelven antes de que se haya creado el árbol XPath.

```

1. <universidad>
2.   <carrera>
3.     <asignatura nombre = Redes>
4.       <alumno>Pepi</alumno>
5.       <alumno>Fernando</alumno>
6.       <alumno>Carolina</alumno>
7.       <alumno>María</alumno>
8.     </asignatura>
9.   </carrera>
10.  <carrera>
11.    <asignatura nombre = SPI>
12.      <alumno>Álvaro</alumno>
13.    </asignatura>
14.  </carrera>
15.  <carrera>
16.    <asignatura nombre = PFC>
17.      <alumno>Nando</alumno>
18.    </asignatura>
19.  </carrera>
20. </universidad>

```

```

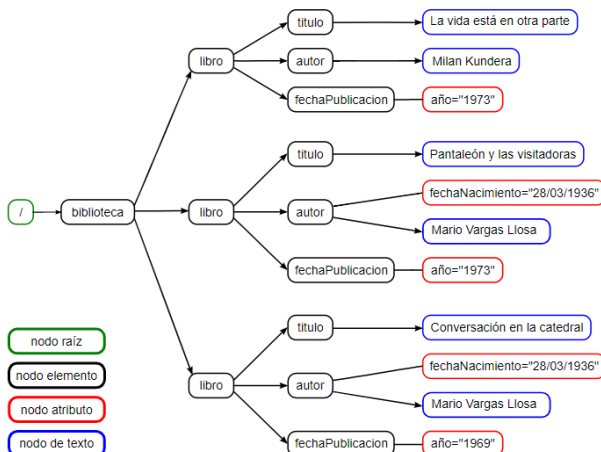
1. /
2. +--universidad
3.   |
4.   +--carrera
5.     |
6.     +--asignatura
7.       |
8.       +--alumno
9.         |
10.        +--(texto)Pepi
11.      +--alumno
12.        |
13.        +--(texto)Fernando
14.      +--alumno
15.        |
16.        +--(texto)Carolina
17.      +--alumno
18.        |
19.        +--(texto)María
20.    +--carrera
21.      |
22.      +--asignatura
23.        |
24.        +--alumno
25.          |
26.          +--(texto)Álvaro
27.    +--carrera
28.      |
29.      +--asignatura
30.        |
31.        +--alumno
32.          |
33.          +--(texto)Nando
34.
35.
36.
37.
38.

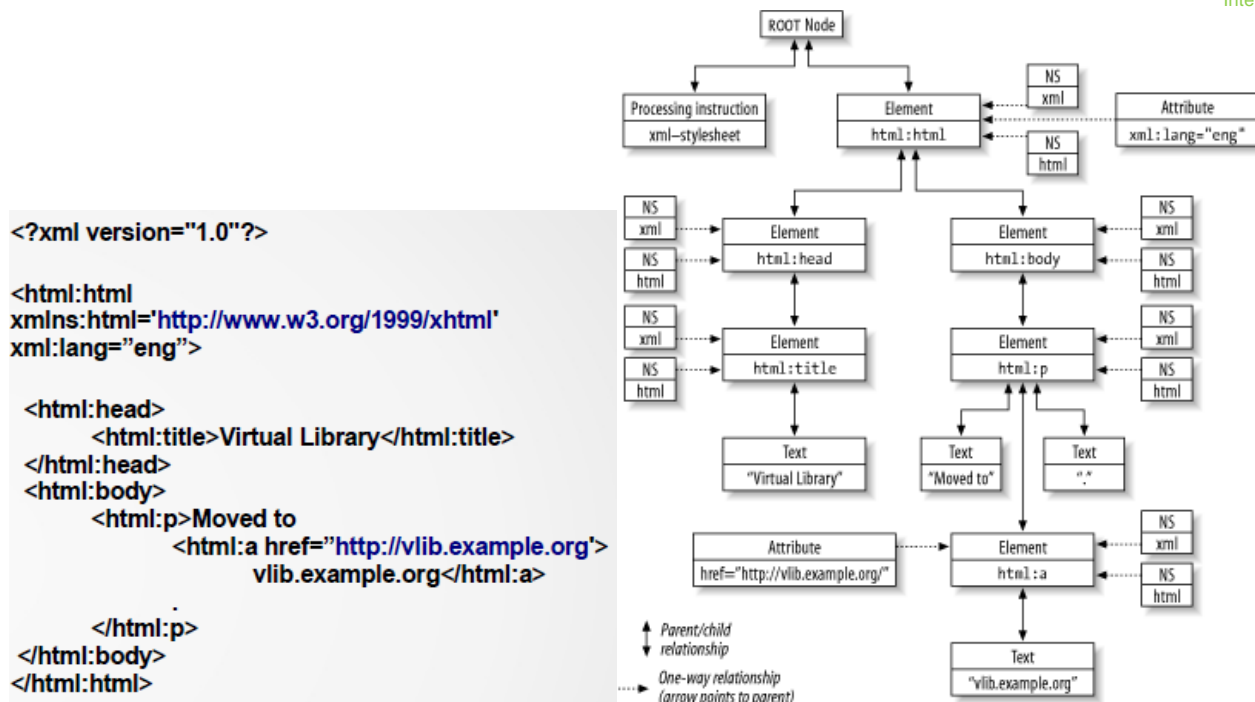
```

```

<?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>

```





Expresiones XPath. Sintaxis

Para seleccionar nodos, XPath utiliza **expresiones de ruta**, cadenas de texto que representan un recorrido en el árbol XML. Evaluar una expresión XPath es buscar elementos en el XML que se ajustan al recorrido definido en la expresión. El resultado son todos los nodos que se ajustan a la expresión.

Las expresiones XPath se construyen a través de **palabras claves, símbolos y operadores**. Distingue entre **mayúsculas y minúsculas**.

Existen los siguientes tipos de expresiones: **literales**, llamadas a **funciones**, **node-sets**, **booleanas**, **numéricas** y **cadenas**.

Las expresiones XPath son relativas al nodo actual que se esté recorriendo, así que una misma expresión puede generar distintos resultados dependiendo del nodo (contexto) donde se ejecute.

El contexto viene definido principalmente por: el nodo de contexto, el número de elementos hermanos del nodo de contexto y el número que indica la posición del nodo de contexto dentro de sus hermanos.

La expresión más útil de XPath es una **ruta de localización** que identifica un conjunto de nodos de un documento. El conjunto puede estar vacío, puede contener un solo nodo o puede contener varios nodos. Los nodos del conjunto pueden ser de cualquier tipo. Una ruta de localización se crea a partir de sucesivos pasos de localización (*location step*) separados entre sí por el carácter **/**. Formato de una ruta de localización:

[**/**] paso de localización / paso de localización / paso de localización ...

Cada paso de localización se evalúa como relativo a un nodo determinado en el documento denominado nodo de contexto.

Los pasos de localización contienen **ejes**, **nodos de comprobación** y **predicados**.

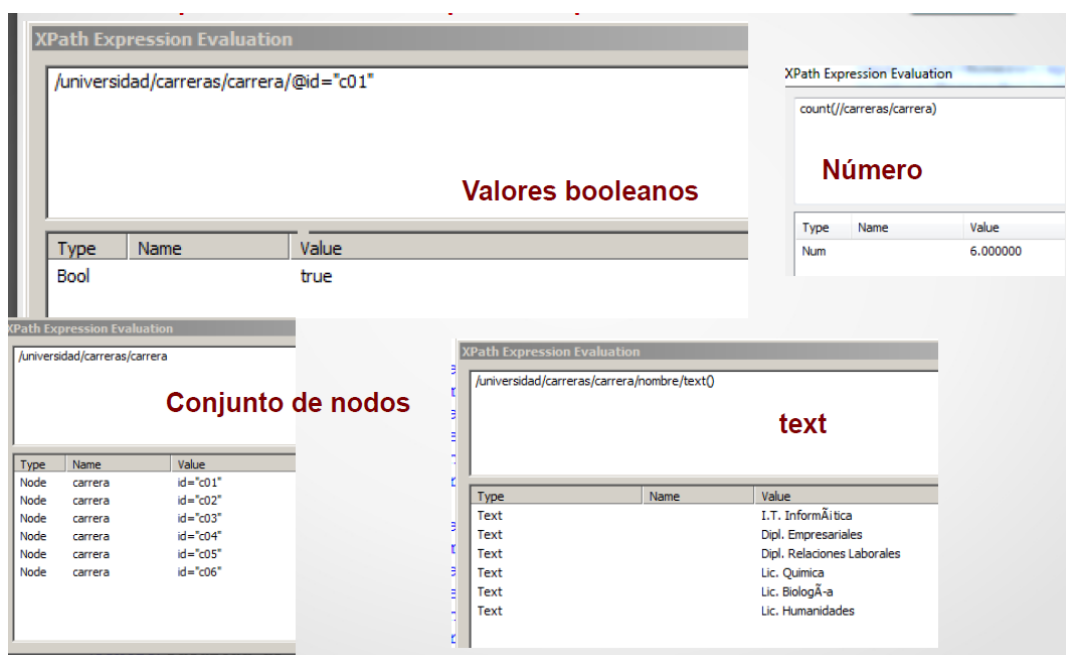
Las expresiones XPath se pueden dividir, como hemos dicho, en varios **pasos** de búsqueda que a su vez se puede dividir en tres partes con esta sintaxis: **eje::nodo de comprobación[predicado(s)]**.

- Un **eje** que indica la ruta donde se hacen las búsquedas.
- **Nodo (node test)** de comprobación que identifica los nodos seleccionados en el eje.
- El **predicado** permite restringir entre los nodos para afinar la búsqueda en función de unas condiciones.

Expresiones XPath. Tipos de datos

Las expresiones son evaluadas y pueden devolver 4 tipos distintos de resultados: **booleanos**, **numéricos**, **cadenas** o un **conjunto de nodos**.

- **Booleano.** El valor booleano (*boolean*) es un valor binario que puede ser verdadero o falso. Para obtenerlos se utilizan los operadores relacionales (**=**, **!=**, **<**, **>**, **<=**, **>=**). Se pueden combinar varias condiciones usando los operadores **and** y **or**. No hay literales booleanos. Para sustituirlos tenemos las funciones **true()** y **false()**. Se utilizan normalmente en los predicados de las rutas de localización.
- **Número.** Todos los números se ajustan a punto flotante de 64 bits. Pueden ser positivos o negativos. Los números incluyen los valores especiales **Inf** (infinito), **-Inf** (infinito negativo) y **NaN** (no es un número). Operadores: **+** (suma), **-** (resta), ***** (multiplicación), **div** (división), **mod** (resto).
- **Cadena:** Secuencia de cero o más caracteres. Los literales de la cadena se entrecomillan con comillas simples o dobles. Para concatenar se utiliza la función **concat()**.
- **Conjunto de nodos.** Un conjunto de nodos es una colección de cero o más nodos de un documento XML. Las rutas de localización producen la mayoría de los conjuntos de nodos. Un solo conjunto de nodos puede contener múltiples tipos de nodos.



The image displays four screenshots of the 'XPath Expression Evaluation' tool, each showing a different type of result:

- Valores booleanos:** The expression `/universidad/carreras/carrera/@id="c01"` is evaluated, resulting in a Boolean value of `true`.
- Número:** The expression `count(/carreras/carrera)` is evaluated, resulting in a numeric value of `6.000000`.
- Conjunto de nodos:** The expression `/universidad/carreras/carrera` is evaluated, resulting in a set of six nodes, each representing a career with an ID from 'c01' to 'c06'.
- text:** The expression `/universidad/carreras/carrera/nombre/text()` is evaluated, resulting in a text value containing a list of career names: 'I.T. Informática', 'Dipl. Empresariales', 'Dipl. Relaciones Laborales', 'Lic. Química', 'Lic. Biología', and 'Lic. Humanidades'.

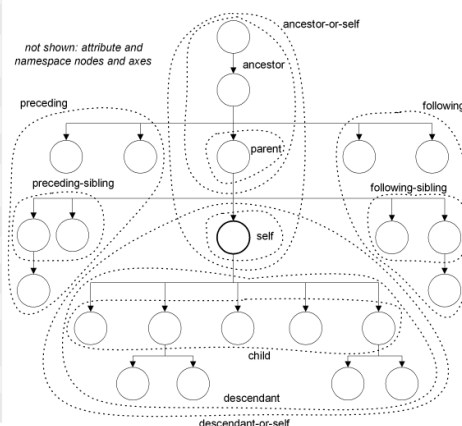
Expresiones XPath. Ejes (AXES)

Los ejes se utilizan para encontrar elementos en alguna parte del documento, para lo que proporcionamos la ruta donde queremos buscar, como la búsqueda en un sistema de archivos. Se establece el nodo actual y se navega por el XML utilizando rutas que pueden ser absolutas (comienzan con /) o relativas y pueden tener varios pasos separados por "/".

- / si está al principio de la expresión, indica nodo raíz, sino, indica "hijo".
- // indica cualquier descendiente (hijos, hijos de hijos, etc.)-
- @atributo selecciona el atributo.
- .. selecciona el elemento padre.
- . selecciona el nodo actual.
- | permite varios recorridos.

Estas relaciones entre nodos se pueden expresar también en formato no compacto:

AxisName	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
preceding-sibling	Selects all siblings before the current node
self	Selects the current node



Expresiones XPath. Nodos de comprobación

Seleccionan los elementos en la ruta o eje actual.

- **name** Selecciona los elementos con ese nombre.
- ***** Selecciona todos los elementos en el nodo actual.
- **@*** Selecciona todos los atributos en el nodo actual.
- **node()** Selecciona todos los nodos (textos y elementos).
- **text()** Selecciona el contenido de un elemento (texto).
- **path1 | path2** Selecciona los elementos de la ruta 1 y los de la ruta 2.
- **comment()** Selecciona el contenido de un elemento comentario.
- **processing-instruction()** Selecciona el contenido de las instrucciones de procesamiento.

Ejemplos de comprobaciones **ejes::nodos de comprobación** (sintaxis extendida):

Example	Result
child::book	Selects all book nodes that are children of the current node
attribute::lang	Selects the lang attribute of the current node
child::*	Selects all element children of the current node
attribute::*	Selects all attributes of the current node
child::text()	Selects all text node children of the current node
child::node()	Selects all children of the current node
descendant::book	Selects all book descendants of the current node
ancestor::book	Selects all book ancestors of the current node
ancestor-or-self::book	Selects all book ancestors of the current node - and the current as well if it is a book node
child::* / child::price	Selects all price grandchildren of the current node

Expresiones XPath. Predicados

Para encontrar un nodo concreto se utilizan predicados, encerrados entre corchetes []. Se utilizan para introducir criterios que permitan seleccionar los nodos que devolvería la expresión. Los más utilizados son:

- [n] Si hay varios nodos en el resultado se selecciona el enésimo.
- [last()] Si hay varios nodos en el resultado se selecciona el último.
- [last()-1] Si hay varios nodos en el resultado se selecciona el penúltimo.
- [position()<3] Si hay varios nodos en el resultado se seleccionan los dos primeros.
- [@nombre] Los elementos que tengan un atributo con el nombre.
- [@nombre="valor"] Los que tenga un atributo nombre y ese valor.
- [nombre>valor] Tiene dentro un elemento nombre cuyo valor es > valor.
- [e1>valor]/e2 Selecciona los elementos llamados e2 hijos de los elementos e1 con valores por encima del valor.

Expresiones XPath. Operadores

Operator	Description	Example
	Computes two node-sets	//book //cd
+	Addition	6 + 4
-	Subtraction	6 - 4
*	Multiplication	6 * 4
div	Division	8 div 4
=	Equal	price=9.80
!=	Not equal	price!=9.80
<	Less than	price<9.80
<=	Less than or equal to	price<=9.80
>	Greater than	price>9.80
>=	Greater than or equal to	price>=9.80
or	or	price=9.80 or price=9.70
and	and	price>9.00 and price<9.90
mod	Modulus (division remainder)	5 mod 2

Expresiones XPath. Funciones

XPath incluye funciones para el manejo de nodos, cadenas y números. Las más utilizadas son:

- sum("conjunto de nodos"): devuelve la suma de los valores de los nodos.
- count("conjunto de nodos"): devuelve el número de nodos.
- position(): devuelve la posición del elemento actual.
- last(): devuelve la posición del último elemento (número de nodos).
- not("expresion"): devuelve lo contrario que retorne la expresión.
- name("conjunto de nodos"): devuelve el nombre del primer elemento.
- number("arg"): convierte arg en un número. Si falla NaN (*Not a Number*), 1 true y 0 false.

boolean(arg)	Convierte el argumento a boolean, siguiendo estas reglas <ul style="list-style-type: none"> Si el argumento es un boolean devuelve su valor. Un node-set devuelve false si está vacío, en caso contrario devuelve true. Un string devuelve true si su longitud es distinta de cero. Un numérico devolverá true si el argumento es positivo ó negativo, false si es cero ó NaN. Si no es ninguno de estos tipos dependerá del tipo en particular.
not(arg)	Devuelve true si el argumento es false, false si el argumento es true.
true()	Devuelve true.
false()	Devuelve false.
lang()	Devuelve true si en el nodo de contexto está especificado el atributo xml:lang (<para xml:lang="en"/>) y el valor de este coincide con el valor del string que se le da como parámetro, false en caso contrario. Si el nodo de contexto es el especificado como ejemplo, lang("en") devuelve true.

number count(node-set)	Devuelve el número de elementos del node-set.
id(arg)	El objeto que se pasa como parámetro se convierte a string, y se devuelven todos los elementos con in parámetro ID igual que el argumento
number last()	Devuelve un número que contiene el tamaño del contexto.
string local-name(node-set)	Devuelve la parte local del nombre expandido del primer elemento del node-set.
string name(node-set)	Devuelve el QName del primer elemento del node-set.
string namespace-uri(node-set)	Devuelve el URI del nombre expandido, si no existe, si es nulo o el node-set está vacío retorna una cadena vacía
number position()	Devuelve la posición del elemento de contexto dentro del contexto.

number(arg)	Convierte el argumento a numérico <ul style="list-style-type: none"> Si es una cadena con un número válido devuelve el valor, en caso contrario devuelve NaN Si es un boolean devuelve 1 si es true, 0 si es false Si es un node set este es convertido a string y después a numérico igual que si el parámetro fuese una cadena Si no es ninguno de estos tipos dependerá del tipo en particular.
sum(node-set)	Devuelve el valor de la suma de todos los elementos del node-set convertido a numérico.
floor(number)	Devuelve el entero menor más cercano al parámetro
ceiling(number)	Devuelve el entero mayor más cercano al parámetro
round(number)	Devuelve el entero más cercano al parámetro

string string(object)	Convierte el objeto pasado como parámetro a cadena de caracteres. <ul style="list-style-type: none"> Un node-set devuelve el valor de convertir a string el primer elemento. Un numérico <ul style="list-style-type: none"> NaN devuelve NaN Valor cero devuelve 0 El valor infinito es convertido a (-)Infinity Si es un valor entero devuelve (-)valor si número decimales Cualquier otro caso devuelve (-)valor con separador decimal y al menos un decimal Booleano devuelve 'true' ó 'false'. Si no es ninguno de estos tipos dependerá del tipo en particular.
string concat(string, string, string*)	Devuelve la concatenación de todos sus argumentos
boolean starts-with(string, string)	Devuelve true si la primera cadena comienza con la segunda cadena, false en caso contrario.
boolean contains(string, string)	Devuelve true si la primera cadena contiene la segunda cadena, false en caso contrario.
string substring-before(string, string)	Devuelve una subcadena formada por los caracteres de la cadena de parámetro hasta que se encuentra la primera ocurrencia de la segunda cadena.
string substring-after(string, string)	Devuelve una subcadena formada por los caracteres de la cadena de parámetro desde que se encuentra la primera ocurrencia de la segunda cadena hasta el final.

string substring(string, number, number?)	Devuelve una cadena compuesta por los caracteres del primer argumento, comenzando en la posición que indique el Segundo argumento hasta el final, si hay tercer argumento indica la longitud de la cadena a devolver
number string-length(string?)	Devuelve la longitud en caracteres del argumento. Si no hay parámetro se convierte a cadena el nodo de contexto y se aplica la función.
string normalize-space(string?)	Devuelve la cadena del argumento normalizada, eliminando espacios en blanco iniciales y finales y reduciendo secuencias de espacios a uno solo. Si no hay parámetro se convierte a cadena el nodo de contexto y se aplica la función.
string translate(string, string, string)	Devuelve una cadena que es el resultado de sustituir caracteres en el primer parámetro. Esta sustitución viene indicada por los parámetros 2 y 3, de modo que si en la cadena 1 encontramos un carácter de la cadena 2 se sustituye por el mismo carácter de la cadena 3 situado en la misma posición, esto se ve mejor con un ejemplo, translate("bar","abc","ABC") devuelve "BAR"

Ejemplo

Como hemos mencionado XPath es un lenguaje que permite seleccionar nodos de un documento XML y calcular valores a partir de su contenido.

Estos ejemplos se hacen sobre el fichero book.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<bookstore>
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <author>Giada De Laurentiis</author>
  <year>2006</year>
  <price>29.99</price>
</book>
</bookstore>
```

Todas las búsquedas se hacen desde el nodo raíz.

1. Ruta de localización:

- `/bookstore/book`: Selecciona todos los nodos "book" que están en la ruta.
- `//year`: Selecciona todos los nodos "year" a partir del nodo raíz.
- `/bookstore/book/title/text()`: Selecciona todos los nodos texto (información) de los elementos seleccionados con la ruta.
- `/bookstore/book/title/@lang`: Selecciona todos los atributos llamados "lang" de los elementos seleccionados con la ruta.
- `//book[1]/title/text()`: Selecciona la información del nodo title del **primer nodo** book.
- `.`: Selecciona el nodo actual.
- `..`: Selecciona al nodo padre.
- `*`: Selecciona todos los nodos

2. Filtrado de nodos:

- `/bookstore/book[title="Everyday Italian"]`: Predicado que filtra de todos los nodos seleccionados con la ruta, aquellos que tienen un nodo hijo cuya información coincide con la indicada.
- `/bookstore/book[year>2005]`: En este caso se hace una comparación numérica.
- `/bookstore/book/title[@lang="en"]`: Ahora la selección se hace con un atributo.
- `/bookstore/book/title[@lang="en"]/text()`: Ejemplo igual que el anterior, pero en este caso se selecciona el nodo texto (información).
- `//book[@category="CHILDREN" and price="29.99"]`: Se pueden utilizar expresiones lógicas: **and** y **or**.

3. Algunas funciones xpath

- `count(//book/title)`: Devuelve el número de nodos seleccionados.
- `sum(//book/price)`: Devuelve la suma de los valores de los nodos seleccionados.
- `//book/author[contains(text(),'De')]`: Devuelve los autores cuya información contiene la subcadena "De".

Más detalle y ejercicios en: <https://www.mclibre.org/consultar/xml/lecciones/xml-xpath.html>.