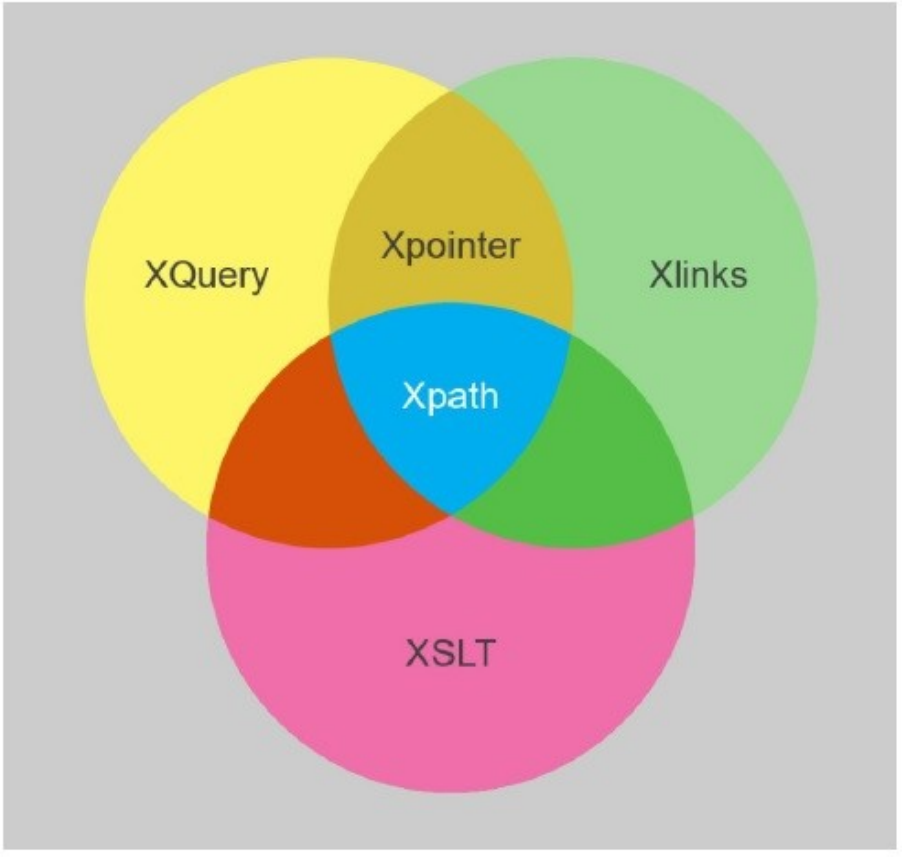
# XPath

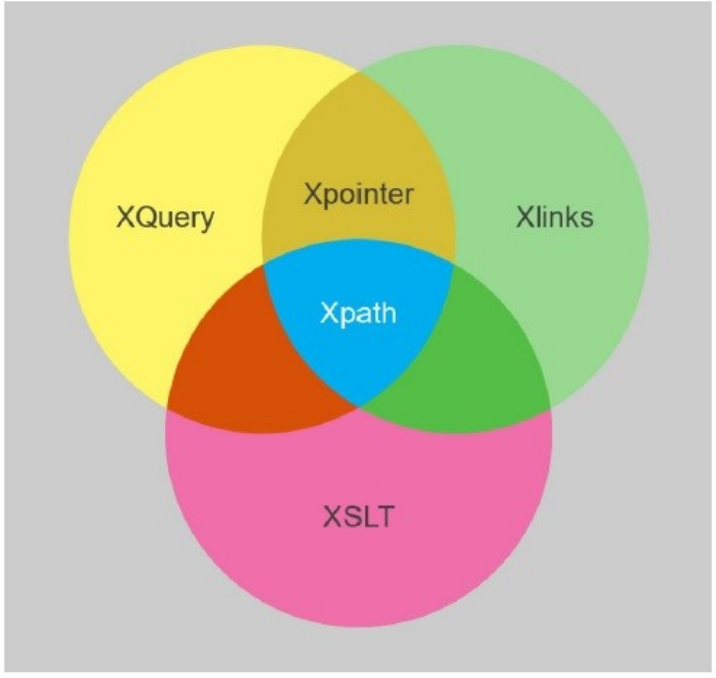
Además de los lenguajes de definición de gramáticas XML específicas, (DTDs y XML Schemas), que se emplearán para validar los documentos XML, existen otras tecnologías asociadas a XML:



**XPath**. Lenguaje para extraer información de un XML.

**XPointer**. Lenguaje que enlaza los hipervínculos con partes específicas de un XML.

**XQuery**. Lenguaje para consultas a un XML, (como SQL).




**XLink**. Lenguaje para la creación de hipervínculos de un XML.

**XSLT**. Lenguaje para hacer transformaciones a un XML obteniendo otras estructuras u otros formatos.

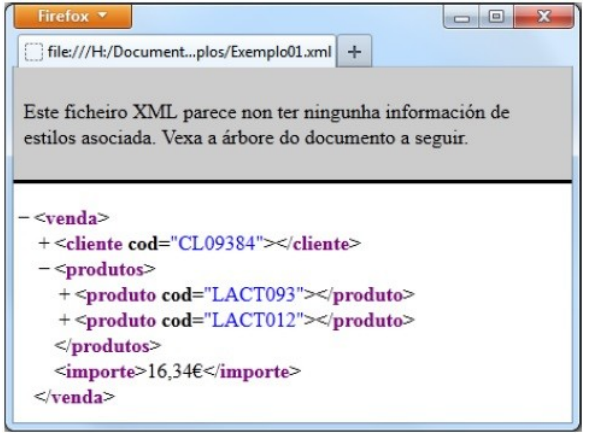
# Procesamiento de un documento XML: estructura en árbol.

Un parser o analizador es un programa capaz de procesar un documento XML; como resultado, generalmente se obtendrá la representación del documento en forma de árbol.

Ejm: Navegador: procesa el documento y muestra el árbol obtenido o un error.

```
<?xml version="1.0" encoding="utf-8"?>
<venda>
	<cliente cod="CL09384">
		<nome>Uxío Fuentes Neira</nome>
		<enderezo>Rúa Europa 24, 3°A</enderezo>
	</cliente>
	<produtos>
		<produto cod="LACT093">
			<descrición>Leite enteira envase 1L</descrición>
		</produto>
		<produto cod="LACT012">
			<descrición>Margarina vexetal tarrina 250g</descrición>
		</produto>
	</produtos>
	<importe>16,34€</importe>
</venda>
```

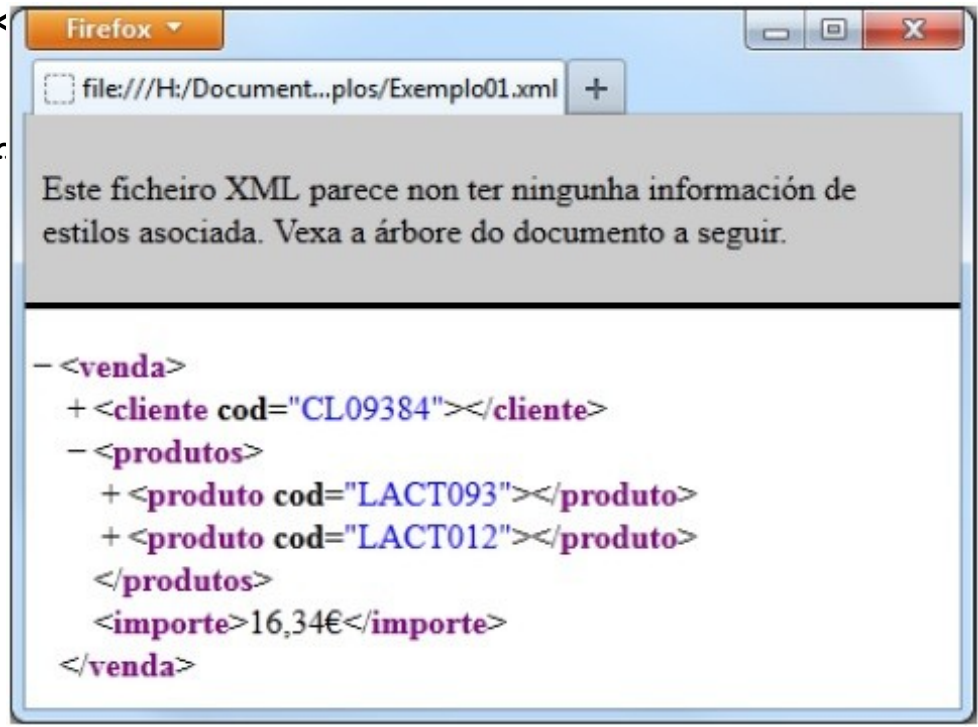
ANALIZADOR



# Procesamiento de un documento XML: estructura en árbol.



```
<?xml version="1.0" encoding="utf-8"?>
<venda>
    <cliente cod="CL09384">
        <nome>Uxío Fuentes Neira</nome>
        <enderezo>Rúa Europa 24, 3°A</enderezo>
    </cliente>
    <produtos>
        <produto cod="LACT093">
            <descrición>Leite enteira envase 1L<
        </produto>
        <produto cod="LACT012">
            <descrición>Margarina vexetal tarrir
        </produto>
    </produtos>
    <importe>16,34€</importe>
</venda>
```



# Procesamiento de un documento XML: estructura en árbol.

El modelo más empleado para almacenar y procesar los árboles XML es DOM, (Document Object Model, Modelo de Objetos del Documento).

DOM es un estándar de W3C, (World Wide Web Consortium), que también se usa en el procesamiento de los documentos HTML de las páginas web.

El estándar DOM puede ser:

- ♣ DOM base. Es un modelo estándar para cualquier documento estructurado.
- ♣ DOM HTML. Es el modelo específico para documentos HTML.
- ♣ DOM XML. Es el modelo específico para documentos XML.

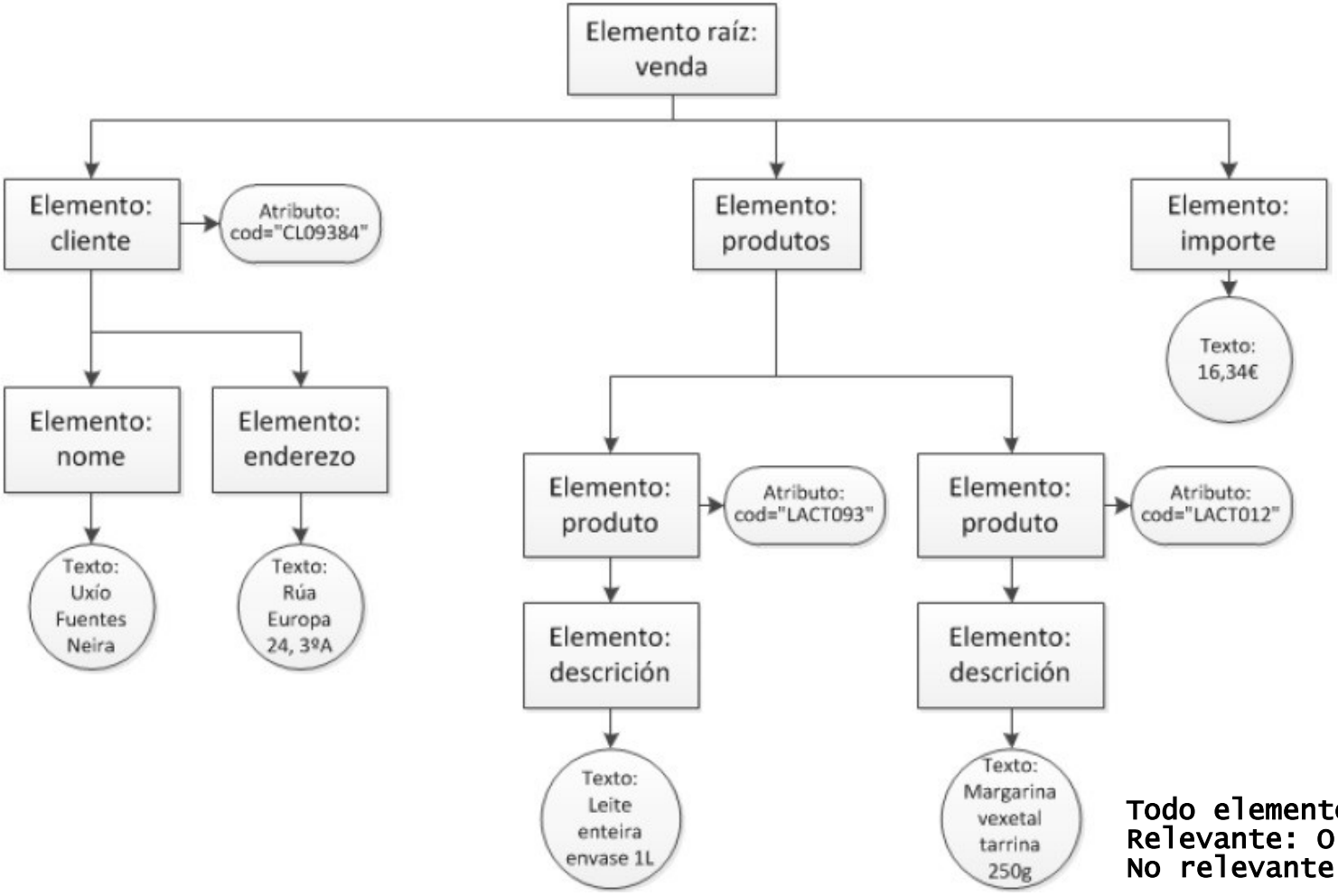
# Procesamiento de un documento XML: estructura en árbol.

Por ejemplo, el siguiente documento XML:

```
<?xml version="1.0" encoding="utf-8"?>
<venda>
    <cliente cod="CL09384">
        <nome>Uxío Fuentes Neira</nome>
        <enderezo>Rúa Europa 24, 3ºA</enderezo>
    </cliente>
    <produtos>
        <produto cod="LACT093">
            <descrición>Leite enteira envase 1L</descrición>
        </produto>
        <produto cod="LACT012">
            <descrición>Margarina vexetal tarrina 250g</descrición>
        </produto>
    </produtos>
    <importe>16,34€</importe>
</venda>
```

# Procesamiento de un documento XML: estructura en árbol.

## Empleando el modelo DOM XML quedaría:




Todo elemento = Nodo.
Relevante: Orden de elementos.
No relevante: Orden atributos.

## Tipos de nodos en el modelo DOM XML

Los nodos en el modelo DOM XML pueden ser de distintos tipos. Los principales son:

♣ Nodo Document (Documento). Representa al documento XML entero: '/'. Tiene un único hijo de tipo elemento (o nodo raíz).

♣ Nodo Element (Elemento). Representa un elemento de un documento XML. Pueden tener identificadores únicos (se puede especificar en el documento de validación) para acceder a ellos de forma directa.

♣ Nodo Attr (Atributo). Representa un atributo de un elemento. Aunque que se les denomina nodos, en la estructura de árbol se considera a los atributos como una información añadida a los nodos Element y no como hijos de estos.

♣ Nodo Text (Texto). Representa al texto de un elemento. Contiene todos los caracteres que no están dentro de alguna etiqueta.

## Tipos de nodos en el modelo DOM XML

También existen otros tipos de nodos, como:

- Nodo Comment (Comentario).
- Nodo CDATASection (Sección Cdata).
- Nodo ProcessingInstruction (Instrucción de Procesamiento).
- Nodo Entity (Entidad).

## Tipos de nodos en el modelo DOM XML

Las relaciones entre los nodos de un árbol DOM XML son las siguientes:

♣ Solamente existe un nodo raíz, de tipo "Elemento".

♣ Un nodo "Elemento" puede tener o no nodos hijos. Un nodo "Elemento" sin hijos es un nodo hoja. Los nodos de los otros tipos nunca tienen hijos.

♣ Todos los nodos a excepción del raíz tienen uno y solamente un nodo padre. Tampoco tienen padre los nodos de tipo "Atributo", que como ya dijimos se cuelgan en el árbol del nodo "Elemento" que los contiene, pero no se consideran hijos de este.

♣ Se llaman nodos hermanos a aquellos nodos "Elemento" que tienen el mismo padre.

## Expresiones XPath

XPath (XML Path) es un lenguaje para acceder a las distintas partes de un documento XML.

Empleando XPath podemos seleccionar y hacer referencia a texto, elementos, atributos y cualquier otra información contenida dentro de un documento XML. No es un lenguaje XML; tiene su propia sintaxis.

Por ejemplo, en el documento XML anterior se podría emplear la siguiente expresión para obtener la dirección del cliente:

/venda/cliente/enderezo

# Expresiones XPath

Existen tres versiones de XPath: 1.0, 2.0 e 3.0. Las dos primeras son recomendaciones (versiones finales) desarrolladas por W3C, y la tercera es una versión candidata.

XPath 1.0 (noviembre 1999) emplea DOM XML y aún sigue siendo con diferencia la versión más usada.

XPath 2.0 (diciembre 2010) usa el modelo XDM.

XPath 3.0 (enero 2013) emplea XDM 3.0.

En esta actividad se utilizará XPath 1.0.

# Expresiones XPath

En XPath 1.0, las expresiones pueden usar y devolver los siguientes tipos de datos:

- Números en coma flotante (floating point).
- Valores booleanos (verdadero o falso).
- Cadenas de caracteres codificadas en Unicode.
- Conjuntos de nodos, que pueden contener cero, uno o más nodos.

# Rutas de localización

El tipo más común de expresión en XPath es la ruta de localización.

Una ruta de localización permite la selección de un conjunto de nodos, partiendo de un nodo contexto en el que se evaluará la expresión.

El resultado de evaluar una ruta de localización es siempre un conjunto de nodos que pueden ser de distintos tipos, no solamente nodos "Elemento".

## Rutas de localización

Las rutas de localización pueden ser absolutas o relativas:

♣ Relativas con relación a un contexto. Estas rutas no comienzan con una barra "/". Por ejemplo, al emplear XPath como parte de otra lenguaje como XSLT, se puede utilizar la expresión relativa:

producto/descripción

…en el contexto formado por el siguiente conjunto de nodos:

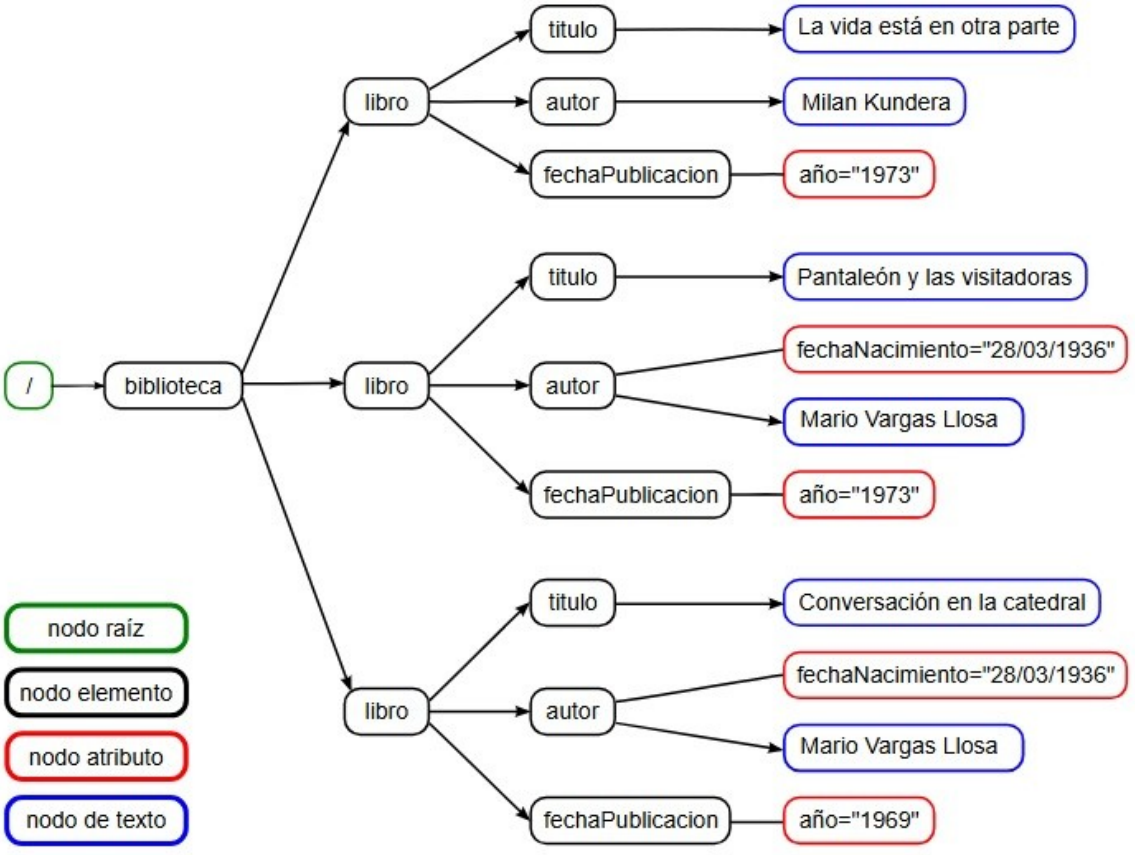
```
<producto cod="LACT012">
      <descripción>Margarina tarrina 250g</descripción>
</producto>
```

Y se obtendría el nodo "descripción" del producto.

# Rutas de localización

♣ Absolutas. Estas rutas comienzan con una barra "/" para indicar que el contexto de la expresión es el nodo "Documento", es decir, el documento XML entero.

```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>
```



## Pasos de localización

Los pasos de localización (location steps) son cada una de las partes de una ruta de localización separadas por "/".

Cada paso de localización va refinando la búsqueda de datos a través de los nodos del árbol.

Por ejemplo, la ruta de localización:

"/venta/cliente/direccion"

Se compone de tres pasos de localización:

El primero hace referencia al nodo elemento raíz "venta", que cuelga del nodo documento "/".

El siguiente hace referencia al nodo "cliente" que es hijo de "venta"; y el mismo para el nodo "direccion".

## Pasos de localización

Como resultado obtenido de evaluar un paso de localización, se obtiene el contexto del siguiente paso de localización.

Constan de un eje (axis), un test de nodo (node test) y opcionalmente de un predicado. La sintaxis es la siguiente:

```
eje::test-nodo[predicado]
```

## Ejes

Los ejes indican, con respecto al nodo contexto, el conjunto de nodos sobre los cuales se evaluará el test de nodo y el predicado si existe.

Básicamente se trata de realizar un primer filtro de nodos del árbol para obtener el resultado con los datos buscados.

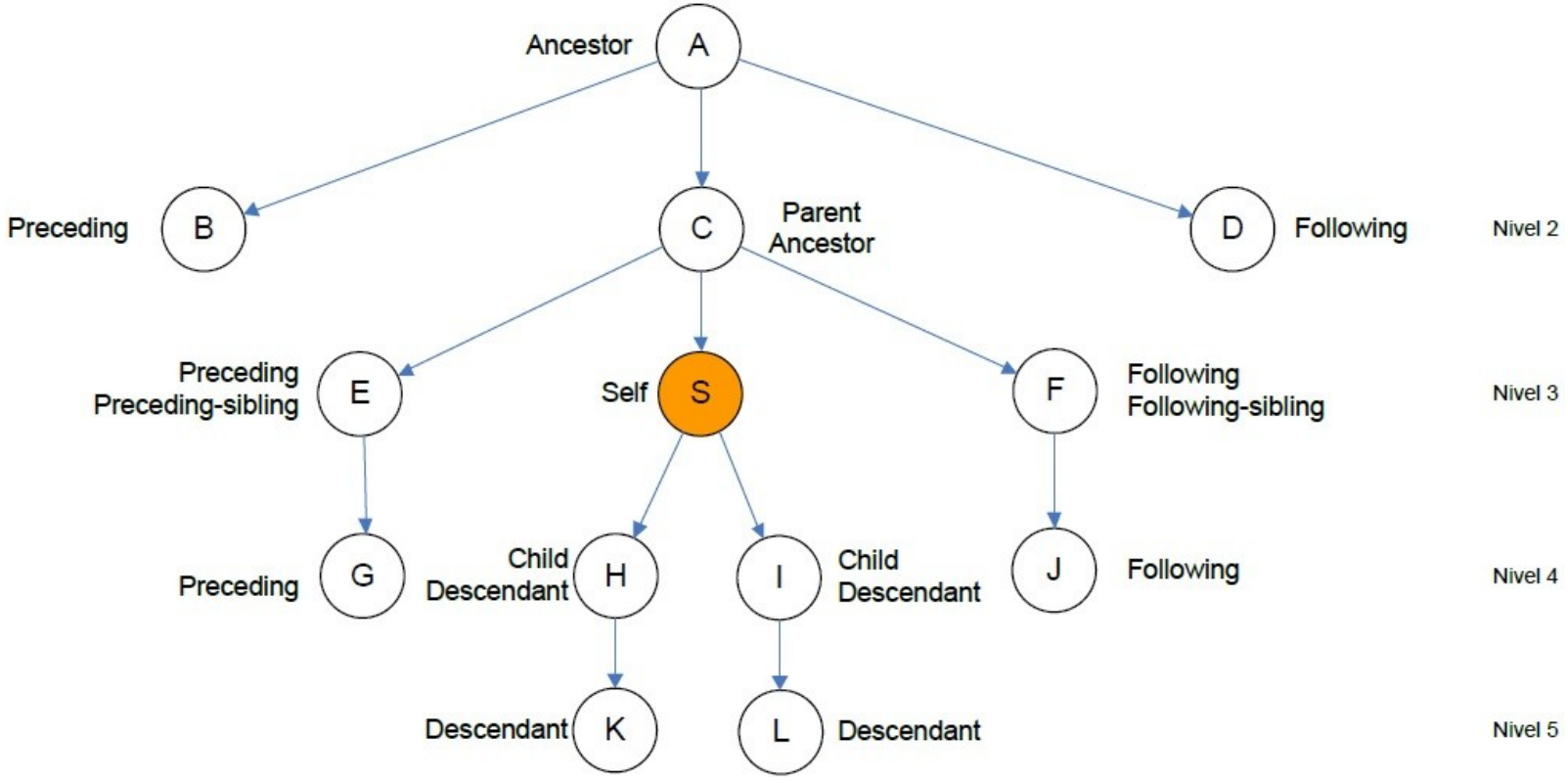
# Ejes

Los ejes que podemos emplear en XPath 1.0 son:

- self. El propio nodo contexto.
- child. Los hijos del nodo contexto.
- parent. El padre del nodo contexto.
- ancestor. Los antepasados del nodo contexto.
- ancestor-or-self. El nodo contexto y sus antepasados.
- descendant. Los descendientes del nodo contexto.
- descendant-or-self. El nodo contexto y sus descendientes.
- following. Los nodos que se encuentran después del nodo contexto en el documento que no son descendientes del mismo ni attribute ni namespace.
- following-sibling. Los nodos que son hermanos del nodo contexto y que se encuentran despues de él en el documento.
- preceding. Los nodos que se encuentran antes del nodo contexto en el documento que no son antepasados del mismo ni attribute ni namespace.
- preceding-sibling. Los nodos que son hermanos del nodo contexto y que se encuentran antes de él en el documento.
- attribute. Los nodos atributo del nodo contexto.
- namespace. Los nodos de espacio de nombres del nodo contexto.

# Ejes

```
<?xml version="1.0" encoding="UTF-8"?>
<A>
    <B>
       ...
    </B>
    <C>
        <E>
            <G>
               ...
            </G>
        </E>
        <S>
            <H>
                <K>
                   ...
                </K>
            </H>
            <I>
                <L>
                   ...
                </L>
            </I>
        </S>
        <F>
            <J>
               ...
            </J>
        </F>
    </C>
    <D>
       ...
    </D>
</A>
```



## Ejes

Algunas consideraciones importantes sobre los ejes son:

El eje por defecto es "child", y que se puede emplear el símbolo "@" en lugar del eje "attribute".

Por ejemplo, las siguientes expresiones son equivalentes:

```
/venda/cliente/@cod

/child::venda/child::cliente/attribute::cod
```

## Tests de nodo

El test de nodo sirve para que una vez identificado un conjunto de nodos con en eje adecuado, se puedan especificar exactamente los nodos de ese conjunto que se desean. Los test de nodo pueden ser:

- nombre_de_nodo. Selecciona todos los nodos con el nombre indicado.
- *. Selecciona todos los elementos y atributos.
- node(). Selecciona todos los nodos (de cualquier tipo).
- text(). Selecciona los nodos de texto.
- comment(). Selecciona los nodos de comentario.
- processing-instructions(). Selecciona los nodos de procesamiento de instruciones.

## Tests de nodo

Por ejemplo, si quisiéramos obtener el conjunto de todos los atributos del documento, podríamos hacer:

```
//descendant-or-self::node()/@*
```

Esto es, seleccionamos todos los nodos del documento, y después nos quedamos con sus atributos. Si quisiéramos solamente los atributos de nombre "cod", podríamos hacer:

```
//descendant-or-self::node()/@cod
```

Y cualquiera de las siguientes formas es válida para obtener los textos de las descripciones de los productos:

```
/venta/productos/producto/descripción/text()

/descendant-or-self::node()/descripción/text()
```

Obsérvese que no es lo mismo obtener como resultado un conjunto de nodos que los nodos de tipo "Texto", tal y como se ve en el siguiente ejemplo, con

```
/descendant-or-self::node()/descripción y

/descendant-or-self::node()/descripción/text().
```

## Tests de nodo

Debido a su frecuente uso, existen abreviaturas para algunas rutas de localización, como las siguientes:

- "//" equivale a "descendant-or-self::node()"
- "." equivale a "self::node()"
- ".." equivale a "parent::node()"

Por ejemplo, una expresión equivalente a

```
/descendant-or-self::node()/descripcion/text()
```

se escribiría:

```
//descripcion/text()
```

## Predicados

Los predicados son condiciones opcionales en un paso de localización, y se introducen entre corchetes después del test de nodo. Ayudan a ajustar la búsqueda en el conjunto de nodos que nos interesan.

Cada predicado puede contener una ruta de localización relativa al nodo actual.
Una forma habitual que se emplea en los predicados, hace uso de una característica especial de XPath: cualquier conjunto de nodos no vacío, es tratado de forma booleana como "verdadero", mientras que a un conjunto de nodos vacío se le asigna el valor booleano "falso".

Por ejemplo, en las siguientes expresiones equivalentes, el predicado filtra los productos obteniendo solamente aquellos que tienen un nodo elemento hijo de nombre "descripción":

```
//producto[child::descripción]

//producto[descripción]
```

## Predicados

Se pueden poner condiciones dentro de los paréntesis Por ejemplo, la siguiente expresión obtendría los nodos producto con código "LACT012":

```
//producto[@cod="LACT012"]
```

O para obtener los nodos cliente de nombre "Uxío Fuentes Neira":

```
//cliente[nombre/text()="Uxío Fuentes Neira"]
```

Un mismo paso de localización puede contener cero, uno o varios predicados; en este último caso, se pondrán uno a continuación del otro, cada uno con sus propios corchetes. Por ejemplo, para obtener los nodos producto que tengan descripción y código "LACT012":

```
//producto[descripción][@cod="LACT012"]
```

## Operadores

En los predicados se pueden utilizar otros operadores, además del operador "=", y funciones XPath para filtrar el conjunto de nodos en función del valor de alguna estructura del documento.

# Operadores

XPath

| Operador | Tipo | Descrición | Exemplo |
|---|---|---|---|
| = | Booleano | Devolve verdadeiro se o valor dos dous operandos coincide, falso en caso contrario. | count(//produto) = 3 |
| != | Booleano | Devolve verdadeiro se o valor dos dous operandos non coincide, falso en caso contrario. | count(//produto) != 3 |
| < | Booleano | Devolve verdadeiro se o valor do primeiro operando é menor que o valor do segundo, falso en caso contrario. | count(//produto) < 3 |
| <= | Booleano | Devolve verdadeiro se o valor do primeiro operando é menor ou igual que o valor do segundo, falso en caso contrario. | count(//produto) <= 3 |
| > | Booleano | Devolve verdadeiro se o valor do primeiro operando é maior que o valor do segundo, falso en caso contrario. | count(//produto) > 3 |
| >= | Booleano | Devolve verdadeiro se o valor do primeiro operando é maior ou igual que o valor do segundo, falso en caso contrario. | count(//produto) >= 3 |
| and | Booleano | Devolve verdadeiro se o valor de ambos operandos é verdadeiro, falso en caso contrario. | count(//produto) > 3 and count(//produto) < 7 |
| or | Booleano | Devolve falso se o valor de ambos operandos é falso, verdadeiro en caso contrario. | count(//produto) < 3 or count(//produto) > 7 |
| - | Numérico | Devolve a resta dos operandos. | count(//produto) - 1 |
| + | Numérico | Devolve a suma dos operandos. | count(//produto) + 1 |
| * | Numérico | Devolve o produto dos operandos. | count(//produto) * 2 |
| div | Numérico | Devolve a división dos operandos. | count(//produto) div 2 |
| mod | Numérico | Devolve o resto da división enteira dos operandos. | count(//produto) mod 2 |
| \| | Conxunto de nodos | Une os operandos, que deben ser conxuntos de nodos, nun novo conxunto de nodos. | //produto \| //cliente |

# Funciones

| Función | Parámetros | Valor devolto | Descrición | Exemplo |
|---|---|---|---|---|
| boolean() | Conxunto de nodos, booleano, numérico ou cadea de carácteres | Booleano | Converte o parámetro a un valor booleano. | boolean(//produto) |
| false() | | Booleano | Devolve falso. | false() |
| true() | | Booleano | Devolve verdadeiro. | true() |
| not() | Conxunto de nodos, booleano, numérico ou cadea de carácteres | Booleano | Devolve verdadeiro se o valor do operando é falso, verdadeiro en caso contrario. | not (count(//produto) < 3) |
| lang() | Cadea de carácteres | Booleano | Devolve verdadeiro se a linguaxe definida con xml:lang coincide coa especificada no parámetro, falso en caso contrario. | lang("es") |
| ceiling() | Numérico | Numérico | Devolve o primeiro enteiro maior que o valor do parámetro. | ceiling(8 div 3) |
| floor() | Numérico | Numérico | Devolve o primeiro enteiro menor que o valor do parámetro. | floor(8 div 3) |
| round() | Numérico | Numérico | Devolve o enteiro máis próximo ao valor do parámetro. | round(8 div 3) |

# Funciones

| Función | Parámetros | Valor devolto | Descrición | Exemplo |
|---|---|---|---|---|
| sum() | Conxunto de nodos | Numérico | Devolve a suma dos valores dos nodos que se pasan como parámetros. | sum(//importe)<br>ceiling(sum(//importe)) |
| count() | Conxunto de nodos | Numérico | Devolve o número de nodos do conxunto de nodos. | count(//produto) |
| avg() | Conxunto de nodos numéricos | Numérico | Devolve a media dos argumentos | avg(//notas) |
| concat() | Varias cadeas de carácteres | Cadea de carácteres | Concatena nunha cadea tódalas que se lle pasan como parámetros. | concat("Don ", //nome/text()) |
| contains() | Dúas cadeas de carácteres | Booleano | Devolve verdadeiro se a primeira cadea contén á segunda, falso en caso contrario. | contains(//nome/text(),"Uxío") |
| normalize-space() | Cadea de carácteres | Cadea de carácteres | Devolve unha cadea como a que se lle pasa como parámetro, quitando os espazos ao comezo, ao final, e os duplicados. | normalize-space(//nome/text()) |
| starts-with() | Dúas cadeas de carácteres | Booleano | Devolve verdadeiro se a primeira cadea comeza coa segunda, falso en caso contrario. | starts-with(//nome/text(),"Uxío")<br>//produto[starts-with(@cod,"LA")] |
| string-length() | Cadea de carácteres | Numérico | Devolve o número de carácteres da cadea. | string-length(//nome/text()) |
| substring() | 1º: Cadea de carácteres<br>2º e 3º: Numérico | Cadea de carácteres | Da cadea que recibe como primeiro parámetro, devolve tantos carácteres como indique o terceiro parámetro, contando a partir da posición que indique o segundo parámetro. | substring(//nome/text(), 6, 7)<br><br>//produto[substring(@cod,string-length(@cod),1)=3] |
| substring-after() | Dúas cadeas de carácteres | Cadea de carácteres | Devolve a cadea do primeiro parámetro a partir da primeira ocorrencia do segundo parámetro. | substring-after(//nome/text()," ") |

# Funciones

| Función | Parámetros | Valor devolto | Descrición | Exemplo |
|---|---|---|---|---|
| substring-before() | Dúas cadeas de carácteres | Cadea de carácteres | Devolve a cadea do primeiro parámetro anterior á primeira ocorrencia do segundo parámetro. | substring-before(//nome/text()," ") |
| translate() | Tres cadeas de carácteres | Cadea de carácteres | Devolve a cadea do primeiro parámetro, substituíndo tódalas ocorrencias dos carácteres do segundo parámetro polos carácteres do terceiro parámetro. | translate(//enderezo/text(),",","-") |
| string() | Conxunto de nodos, booleano, numérico ou cadea de carácteres | Cadea de carácteres | Devolve o parámetro convertido a unha cadea de carácteres. | string(//nome) |
| id() | Cadea de carácteres | Conxunto de nodos | Devolve o nodo do elemento co ID especificado como parámetro. | id("G0097763") |
| last() | | Conxunto de nodos | Devolve o número de nodos no contexto actual. Pódese empregar para acceder ao último nodo do contexto. | //produto[last()]<br><br>//produto[last()-1] |

# Funciones



| Función | Parámetros | Valor devolto | Descrición | Exemplo |
|---|---|---|---|---|
| local-name() | Conxunto de nodos | Cadea de carácteres | Devolve o nome local (non o nome cualificado) do primeiro nodo no conxunto de nodos que se lle pasa como parámetro. | local-name(//nome) |
| name() | Conxunto de nodos | Cadea de carácteres | Devolve o nome cualificado do primeiro nodo no conxunto de nodos que se lle pasa como parámetro. | name(//nome) |
| namespace-uri() | Conxunto de nodos | Cadea de carácteres | Devolve o URI do espazo de nomes do primeiro nodo no conxunto de nodos que se lle pasa como parámetro. | namespace-uri(//produto) |
| position() | | Numérico | Devolve a posición (comezando con 1) do nodo contexto no conxunto de nodos do contexto actual. | //produto[position()=2] |

La función position() se usa habitualmente en el predicado para seleccionar el nodo correspondiente a una posición determinada dentro del conjunto de nodos del contexto. Esto se puede hacer también de forma abreviada indicando la posición directamente en el predicado, de tal forma que las siguientes expresiones son equivalentes:

```
//produto[position()=2]
//produto[2]
```

## Otras expresiones

Aunque la ruta de localización es el tipo más común de expresión en XPath, podemos usar los operadores y funciones anteriores para crear diversos tipos de expresiones.

Algunas de ellas puede que devuelvan números o valores booleanos o cadenas de texto o un conjunto de nodos, y que no se pudiera ver su resultado en XPath Visualizer pero si en XPathBuilder, como se muestra en los ejemplos siguientes.

♣ Contar el número de produtos de una venta, (devuelve un número):

```
count(//produto)
```

♣ Comprobar si el número de produtos de una venta cumple o no ciertas condiciones, (devuelve un valor booleano):

```
count(//produto) > 3 and count(//produto) < 7
```

♣ Obtener los datos de un producto y los del cliente, (devuelve un conjunto de nodos):

```
//produto[2] | //cliente
```

## Otras expresiones

♣ Comprobar el valor del código de un producto determinado, (devuelve un valor booleano):

```
//produto[2]/@cod = "LACT012"
```

♣ Comprobar si existe algún producto con un código determinado, (devuelve un valor booleano):

```
//produto/@cod = "LACT012"
```

♣ Obtener el importe de la venta, cambiando la coma por un punto, (devuelve una cadena de texto):

```
substring(translate(/venda/importe,",","."),1,5)
```

# XPath Y espacios de nombres



♣♣♣