Material didáctico para exposición teórica

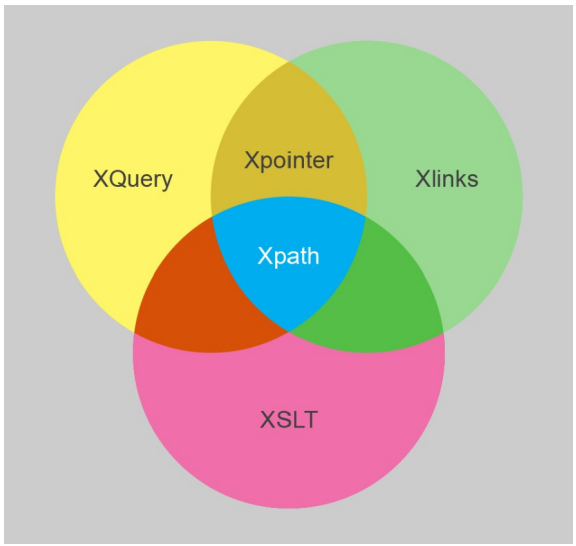
Familia profesional	IFC	Informática e comunicacións
Ciclo formativo	CSIFC03 CSIFC02 CSIFC01	Desenvolvemento de aplicacións web Desenvolvemento de aplicacións multiplataforma Administración de sistemas informáticos en rede
Grao		Superior
Módulo profesional	MP0373	Linguaxes de marcas e sistemas de xestión de información
Unidade didáctica	**UD02**	**XML**
Actividade	**A03**	**Busca de información con XPath**
Profesoras		María del Carmen Fernández Lameiro María Elena Goy López

Índice

1. XPath

1.1 Linguaxes para o procesamento de documentos XML

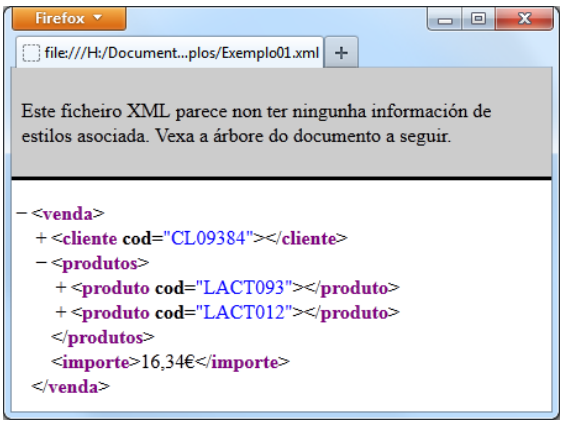
Ao redor da linguaxe XML existe un conxunto de tecnoloxías que se empregan para procesar e obter información dos documentos XML. Ademais das linguaxes de definición de gramáticas XML específicas, como *DTD* e *XML Schemas,* que se empregarán no proceso de validación dos documentos XML, existen outras tecnoloxías como as seguintes:



- *XPath*. É unha linguaxe que permite extraer información dun documento XML.
- *XLink*. É unha linguaxe para a creación de hipervínculos nun documento XML.
- *XPointer*. É unha linguaxe que permite que os hipervínculos enlacen con partes específicas dun documento XML.
- *XQuery*. É unha linguaxe que permite realizar consultas a documentos XML, do mesmo xeito que SQL o fai coas bases de datos relacionais.
- *XSLT*. É unha linguaxe que permite transformar documentos XML, obtendo novos documentos XML con diferente estrutura ou documentos noutros formatos.

1.2 Procesamento dun documento XML: estrutura en árbore

Un *parser* ou analizador é un programa capaz de procesar un documento XML; xeralmente o resultado obtido é a representación deste documento en forma de árbore. Moitas aplicacións integran un parser XML, por exemplo, os navegadores web que procesan o documento e amosan a árbore obtida como resultado:



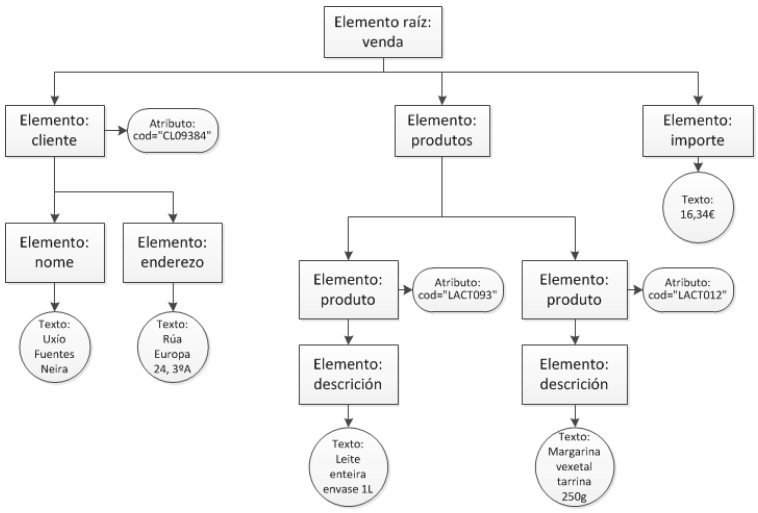
O modelo máis empregado para almacenar e procesar as árbores XML é DOM (*Document Objetc Model*, Modelo de Obxectos do Documento). O DOM é un estándar do W3C (*World Wide Web Consortium*) que tamén se emprega no procesamento dos documentos HTML das páxinas web. O estándar do DOM pode ser:

- DOM base. É un modelo estándar para calquera documento estruturado.
- DOM HTML. É o modelo específico para documentos HTML.
- DOM XML. É o modelo específico para documentos XML.

Por exemplo, o seguinte 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>
```

Preséntase do seguinte xeito empregando o modelo DOM XML:



Nunha árbore DOM XML, represéntanse tódolos compoñentes do documento XML mediante nodos. Parar crear a árbore debe terse en conta a orde na que aparecen os elementos dentro do documento XML; por exemplo, o produto de código "`LACT093`" é anterior ao de código "`LACT012`". Non é relevante a orde dos atributos dentro dun elemento.

O estándar DOM XML define tamén unha interface de programación orientada a obxectos, cos obxectos correspondentes aos distintos nodos dun documento XML, e as propiedades e os métodos para acceder a eles (obter o seu contido, modificalos, eliminalos ou engadir novos nodos).

Existe outro modelo que tamén se emprega con documentos XML: o XDM (*XQuery and XPath Data Model*, Modelo de Datos de XPath e XQuery). XDM emprégase en XQuery e nas versións 2.0 e 3.0 de XPath e XSLT, e baséase nos mesmos principios que DOM.

1.2.1 Tipos de nodos no modelo DOM XML

Os nodos no modelo DOM XML poden ser de distintos tipos. Os principais son:

- Nodo *Document* (Documento). Representa ao documento XML enteiro. Ten un único fillo de tipo elemento (o nodo raíz).
- Nodo *Element* (Elemento). Representa un elemento dun documento XML. Poden ter identificadores únicos (pódese especificar no documento de validación) para acceder a eles de xeito directo.
- Nodo *Attr* (Atributo). Representa un atributo dun elemento. Aínda que se lles denomina nodos, na estrutura de árbore considérase aos atributos como unha información engadida aos nodos *Element* e non como fillos deles.
- Nodo *Text* (Texto). Representa ao texto dun elemento. Contén todos os carácteres que non están dentro dalgunha etiqueta.

Tamén existen outros tipos de nodos, como:

- Nodo *Comment* (Comentario).
- Nodo *CDATASection* (Sección CData).
- Nodo *ProcessingInstruction* (Instrución de Procesamento).
- Nodo *Entity* (Entidade).

As relacións entre os nodos dunha árbore DOM XML son as seguintes:

- Soamente existe un *nodo raíz*, de tipo "Elemento".
- Un nodo "Elemento" pode ter ou non *nodos fillos*. Un nodo "Elemento" sen fillos é un *nodo folla*. Os nodos dos outros tipos nunca teñen fillos.
- Tódolos nodos a excepción do raíz teñen un e soamente un *nodo pai*. Tampouco teñen pai os nodos de tipo "Atributo", que como xa dixemos cólganse na árbore do nodo "Elemento" que os contén, pero non se consideran fillos deste.
- Chámanse *nodos irmáns* a aqueles nodos "Elemento" que teñen o mesmo pai.

1.2.2 A interface de programación DOM XML

Como xa se indicou antes, DOM XML define tamén unha interface de programación para manexar os nodos da árbore correspondente a un documento XML. Esta interface de programación inclúe propiedades como:

- *nodeName*, para obter o nome dun nodo.
- *parentNode*, para obter o pai dun nodo.
- attributes, para obter os atributos dun nodo.

E tamén métodos como:

- *getElementsByTagName(nome)*, obtén os elementos cunha etiqueta determinada.
- *hasAttributes()*, indica se un elemento ten ou non atributos.
- *removeChild(nodo)*, elimina a un fillo dado dun nodo.

Así por exemplo, se quixéramos eliminar do documento anterior o nodo correspondente ao enderezo do cliente poderíamos facer:

```
// supoñemos o documento XML cargado en docXML
enderezo=docXML.getElementsByTagName("enderezo")[0];
cliente=enderezo.parentNode;
cliente.removeChild(enderezo);
```

As expresións que empregan DOM para percorrer e localizar nodos nunha árbore XML son en moitos casos complexas e moi largas. Como se verá a continuación, a linguaxe XPath ofrece unha forma moito máis sinxela e efectiva de facer esta tarefa.

1.3 Expresións XPath

XPath (*XML Path*) é unha linguaxe para acceder ás distintas partes dun documento XML. Empregando XPath podemos seleccionar e facer referencia a texto, elementos, atributos e calquera outra información contida dentro dun documento XML. Non é unha linguaxe XML; ten a súa propia sintaxe.

Por exemplo, no documento XML anterior poderíase empregar a seguinte expresión para obter o enderezo do cliente:

```
/venda/cliente/enderezo
```

Existen tres versións de XPath: 1.0, 2.0 e 3.0. As dúas primeiras son recomendacións (versións finais) desenvolvidas polo W3C, e a terceira é unha versión candidata. XPath 1.0 (novembro 1999) emprega DOM XML e aínda segue a ser con diferenza a versión máis empregada. XPath 2.0 (decembro 2010) emprega o modelo XDM. XPath 3.0 (xaneiro 2013) empregará XDM 3.0. Nesta actividade utilizarase XPath 1.0.

En XPath 1.0, as expresións poden empregar e devolver os seguintes tipos de datos:

- Números en coma flotante (*floating point*).
- Valores booleanos (verdadeiro ou falso).
- Cadeas de carácteres codificadas en Unicode.
- Conxuntos de nodos, que poden conter cero, un ou máis nodos.

1.3.1 Rutas de localización

O tipo máis común de expresión en XPath é a ruta de localización. Unha ruta de localización permite a selección dun conxunto de nodos, partindo dun nodo contexto no que se avaliará a expresión. O resultado de avaliar unha ruta de localización é sempre un conxunto de nodos que poden ser de distintos tipos, non soamente nodos "Elemento".

As rutas de localización poden ser absolutas ou relativas:

- Relativas con relación a un contexto. Estas rutas non comezan cunha barra "`/`". Por exemplo, ao empregar XPath como parte doutra linguaxe como XSLT, pódese utilizar a expresión relativa:

  ```
  produto/descrición
  ```

  No contexto formado polo seguinte conxunto de nodos:

  ```
  <produto cod="LACT012">
    <descrición>Margarina vexetal tarrina 250g</descrición>
  </produto>
  ```

  E obteríase o nodo "`descrición`" do produto.
- Absolutas. Estas rutas comezan cunha barra "/" para indicar que o contexto da expresión é o nodo "Documento", é dicir, o documento XML enteiro.

1.3.2 Pasos de localización

Os pasos de localización (*location steps*) son cada unha das partes dunha ruta de localización separadas por "`/`". Cada un dos pasos de localización vai refinando a procura de datos a través dos nodos da árbore. Por exemplo, a ruta de localización "`/venda/cliente/enderezo`" componse de tres pasos de localización. O primeiro fai referencia ao nodo raíz "`venda`", que colga do nodo documento "`/`"; o seguinte fai referencia ao nodo "`cliente`" que é fillo de "`venda`"; e o mesmo para o nodo "`enderezo`".

Co resultado obtido de avaliar un paso de localización, obtense o contexto do seguinte paso de localización. Constan dun eixo (*axis*), un test de nodo (*node test*) e opcionalmente dun predicado. A sintaxe é a seguinte:

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

1.3.2.1 Eixos

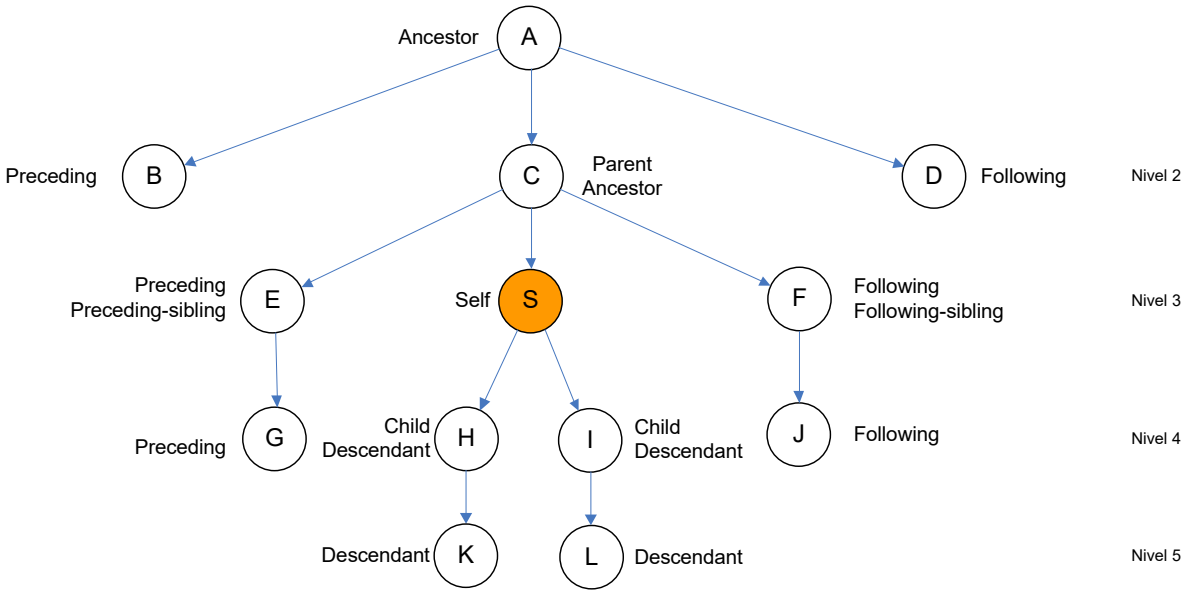
Os eixos indican, con respecto ao nodo contexto, o conxunto de nodos sobre os cales se avaliará o test de nodo e o predicado se existe. Basicamente trátase de realizar un primeiro filtro de nodos da árbore para obter o resultado cos datos buscados. Os eixos que podemos empregar en XPath 1.0 son:

- `self`. O propio nodo contexto.
- `child`. Os fillos do nodo contexto.
- `parent`. O pai do nodo contexto.
- `ancestor`. Os antepasados do nodo contexto.

- `ancestor-or-self`. O nodo contexto e os seus antepasados.
- `descendant`. Os descendentes do nodo contexto.
- `descendant-or-self`. O nodo contexto e os seus descendentes.
- `following`. Os nodos que se encontran despois do nodo contexto no documento que non son descendentes do mesmo nin *attribute* nin *namespace*.
- `following-sibling`. Os nodos que son irmáns do nodo contexto e que se encontran despois del no documento.
- `preceding`. Os nodos que se encontran antes do nodo contexto no documento que non son antepasados do mesmo nin *attribute* nin *namespace.*
- `preceding-sibling`. Os nodos que son irmáns do nodo contexto e que se encontran antes del no documento.
- `attribute`. Os nodos atributo do nodo contexto.
- `namespace`. Os nodos de espazo de nomes do nodo contexto.

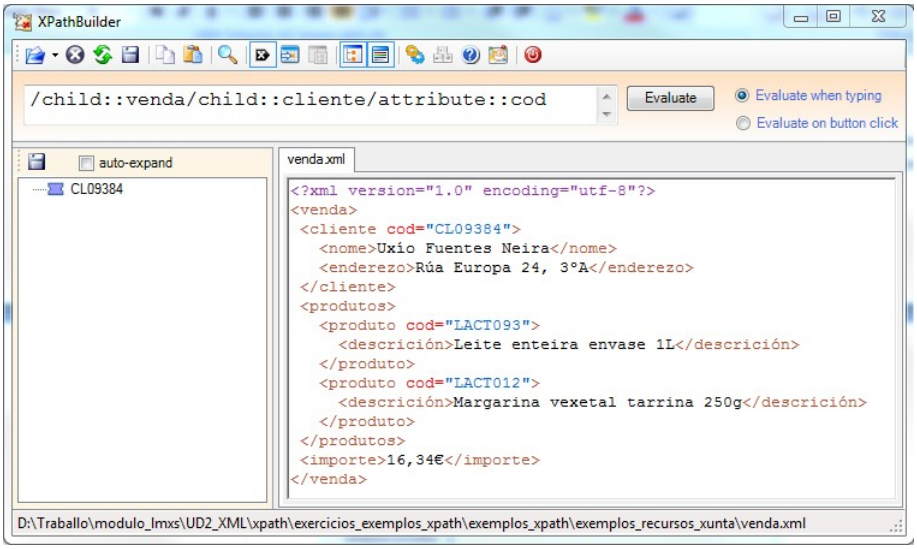
No seguinte exemplo poden verse esbozados os nodos de tipo *Element* dun documento XML exemplo e a representación gráfica en forma de árbore co nome dos eixos relativos ao nodo *self*.

```
<?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>
```



Algunhas consideracións importantes sobre os eixos son que o eixo por defecto é "`child`", e que se pode empregar o símbolo "`@`" no lugar do eixo "`attribute`". Por exemplo, as seguintes expresións son equivalentes:

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



1.3.2.2 Tests de nodo

O test de nodo serve para que unha vez identificado un conxunto de nodos co eixo adecuado, se poidan especificar exactamente os nodos dese conxunto que se desexan. Os test de nodo poden ser:

- *nome_dun_nodo*. Selecciona todos os nodos co nome indicado.
- `*`. Selecciona todos os elementos e atributos.
- `node()`. Selecciona todos os nodos (de calquera tipo).
- `text()`. Selecciona os nodos de texto.
- `comment()`. Selecciona os nodos de comentario.
- `processing-instructions()`. Selecciona os nodos de procesamento de instrucións.

Por exemplo, se quixéramos obter o conxunto de tódolos atributos do documento, poderíamos facer:

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

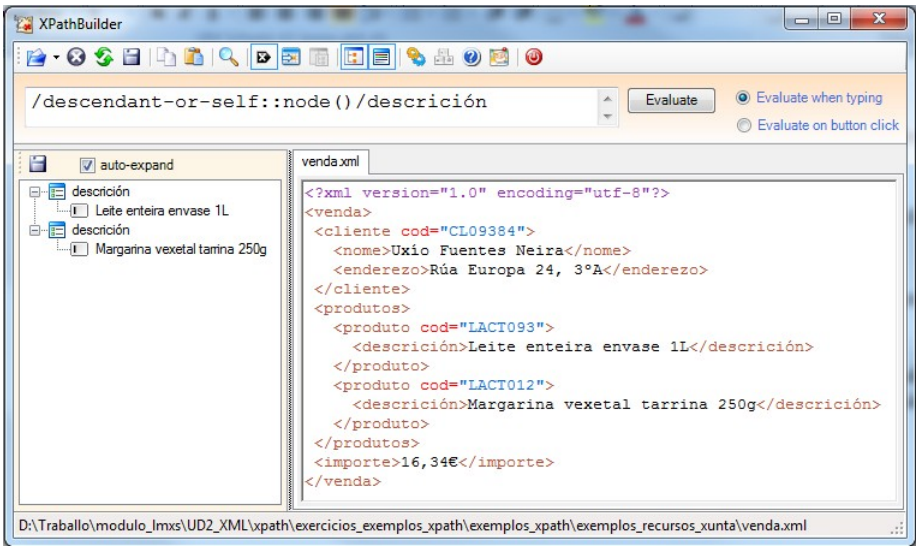
Isto é, seleccionamos tódolos nodos do documento, e despois quedámonos cos seus atributos. Se quixeramos soamente os atributo de nome "`cod`", poderiamos facer:

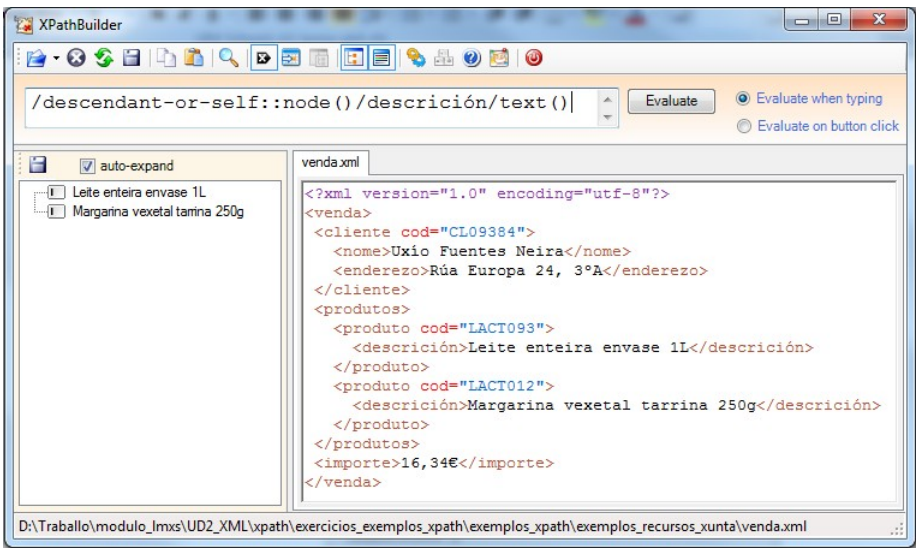
```
//descendant-or-self::node()/@cod
```

E calquera das seguintes formas é válida para obter os textos das descricións dos produtos:

```
/venda/produtos/produto/descrición/text()
/descendant-or-self::node()/descrición/text()
```

Obsérvese que non é o mesmo obter como resultado un conxunto de nodos que os nodos de tipo "Texto", tal e como se ve no seguinte exemplo, con `/descendant-or-self::node()/descrición` e `/descendant-or-self::node()/descrición/text()`.





Debido ao seu frecuente uso, existen abreviaturas para algunhas rutas de localización, como as seguintes:

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

- "`..`" equivale a "`parent::node()`"

Por exemplo, unha expresión equivalente á anterior é:

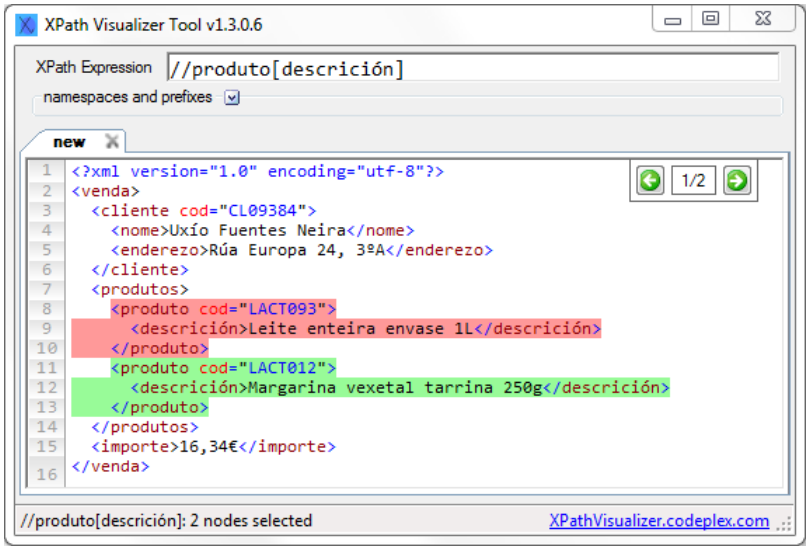
```
//descrición/text()
```

1.3.2.3 Predicados

Os predicados son condicións opcionais nun paso de localización, e introdúcense entre corchetes despois do test de nodo. Axudan a axustar a busca no conxunto de nodos que nos interesan. Cada predicado pode conter unha ruta de localización relativa ao nodo actual.

Unha forma habitual que se emprega nos predicados, fai uso dunha característica especial de XPath: calquera conxunto de nodos non baleiro é tratado de forma booleana como "verdadeiro", mentres que a un conxunto de nodos baleiro asígnaselle o valor booleano "falso". Por exemplo, nas seguintes expresión equivalentes, o predicado filtra os produtos obtendo soamente aqueles que teñen un nodo elemento fillo de nome "`descrición`":

```
//produto[child::descrición]
//produto[descrición]
```



Pódense poñer condicións dentro dos parénteses. Por exemplo, a seguinte expresión obtería os nodos *produto* con código "`LACT012`":

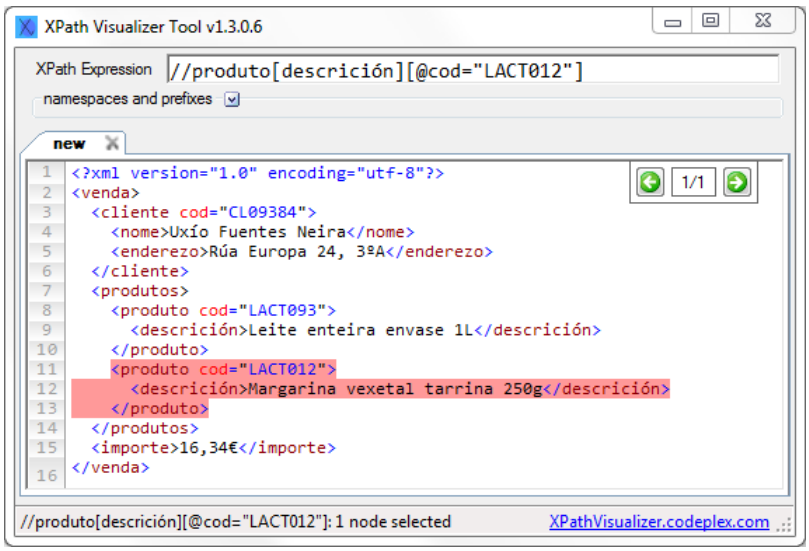
```
//produto[@cod="LACT012"]
```

Ou para obter os nodos *cliente* de nome "Uxío Fuentes Neira":

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

Un mesmo paso de localización pode conter cero, un ou varios predicados; neste último caso, poranse un a continuación do outro, cadanseu cos seus propios corchetes. Por exemplo, para obter os nodos *produto* que teñan descrición e código "`LACT012`":

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



1.3.3 Operadores

Nos predicados pódense utilizar outros operadores, ademais do operador "=", e funcións XPath para filtrar o conxunto de nodos en función do valor dalgunha estrutura do documento.

XPath 1.0 define os seguintes operadores que se poden empregar nas expresións.

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

1.3.4 Funcións

XPath 1.0 define as seguintes funcións que se poden empregar nas expresións.

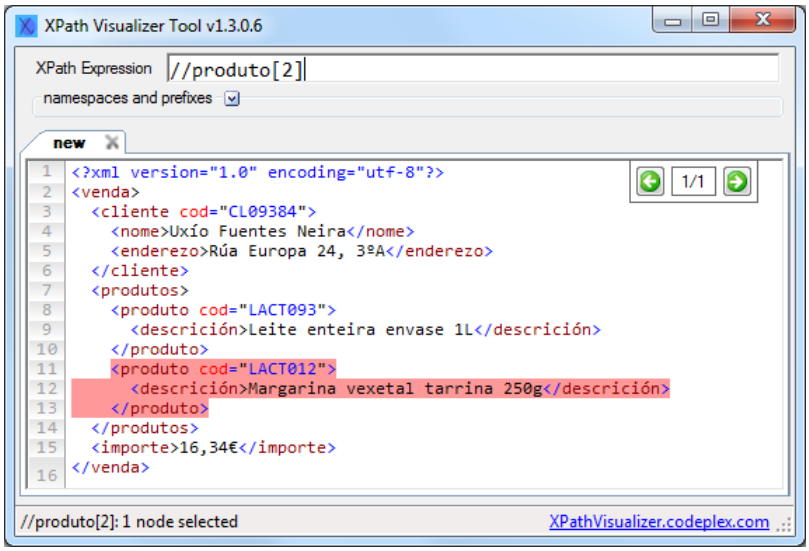
Función	Parámetros	Valor devolto	Descrición	Exemplo
boolean()	Conxunto de nodos, booleano, numérico ou cadea de carácteres	Booleano	Converte o parámetro a un valor booleano.	boolean(//produto)
false()		Booleano	Devolve falso.	false()
true()		Booleano	Devolve verdadeiro.	true()
not()	Conxunto de nodos, 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)
sum()	Conxunto de nodos	Numérico	Devolve a suma dos valores dos nodos que se pasan como parámetros.	sum(//importe) 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") //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 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) //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()," ")
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()]
				
				//produto[last()-1]

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]

A función `position()` emprégase habitualmente no predicado para seleccionar o nodo correspondente a unha posición determinada dentro do conxunto de nodos do contexto. Isto pódese facer tamén de forma abreviada indicando a posición directamente no predicado, de tal xeito que ámbalas dúas expresións seguintes son equivalentes:

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

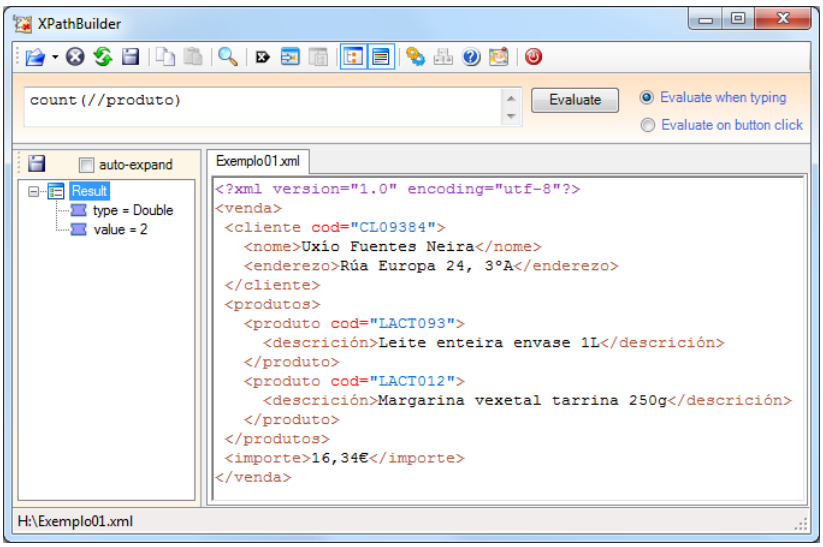


1.3.5 Outras expresións

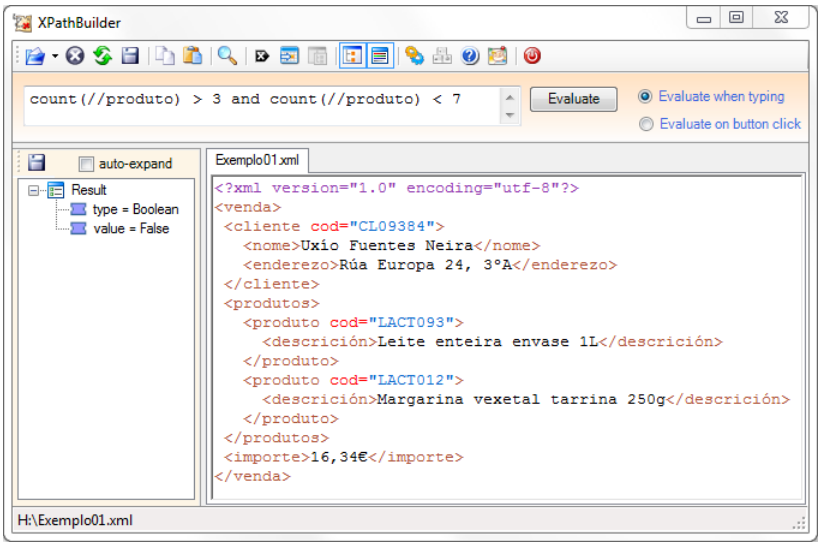
Aínda que a ruta de localización é o tipo máis común de expresión en XPath, podemos empregar os operadores e funcións anteriores para crear diversos tipos de expresións. Algunhas delas pode que devolvan números ou valores booleanos ou cadeas de texto ou un conxunto de nodos, e que non se poderá ver o seu resultado en *XPath Visualizer* pero si en *XPathBuilder,* como se mostra nos exemplos seguintes.

- Contar o número de produtos dunha venda, que devolve un número:

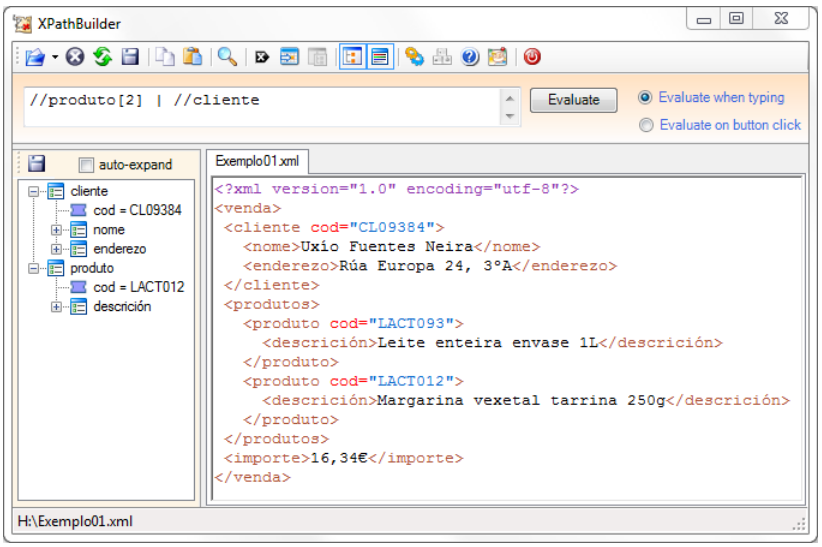
```
count(//produto)
```



- Comprobar se o número de produtos dunha venda cumpre ou non certas condicións, que devolve un valor booleano:
  `count(//produto) > 3 and count(//produto) < 7`

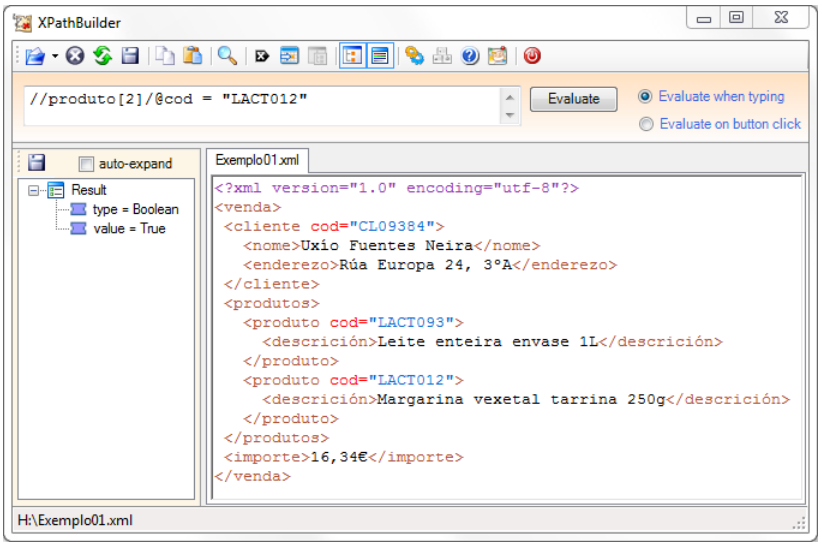


- Obter os datos dun produto e os do cliente, que devolve un conxunto de nodos:
  `//produto[2] | //cliente`



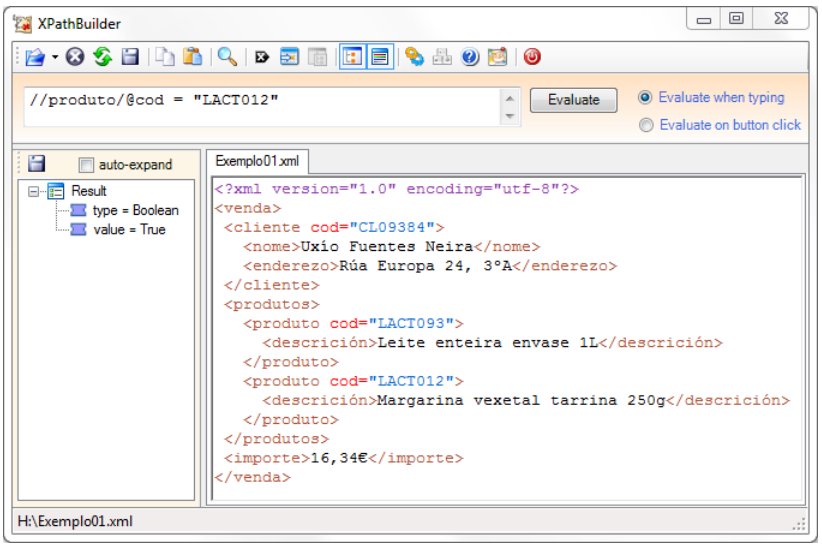
- Comprobar o valor do código dun produto determinado, que devolve un valor booleano:

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



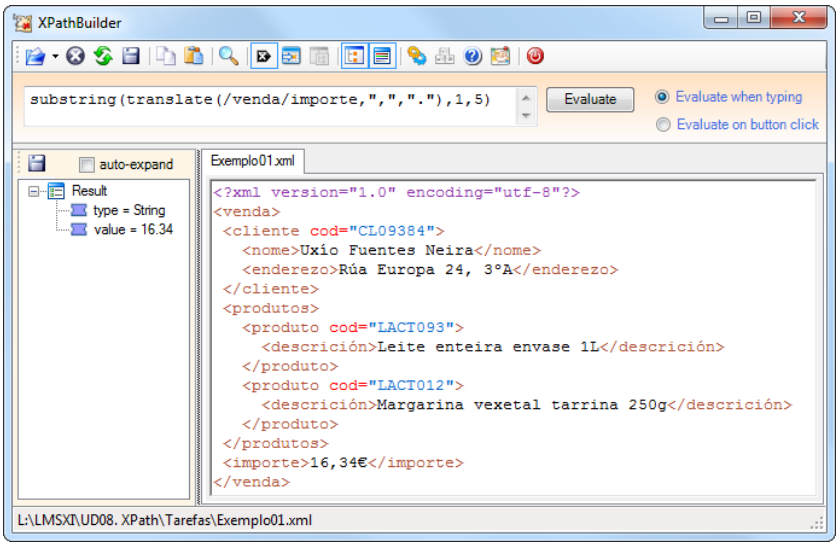
- Comprobar se existe algún produto cun código determinado, que devolve un valor booleano:

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



- Obter o importe da venda, cambiando a coma por un punto, que devolve unha cadea de texto:

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



1.4 XPath e espazos de nomes

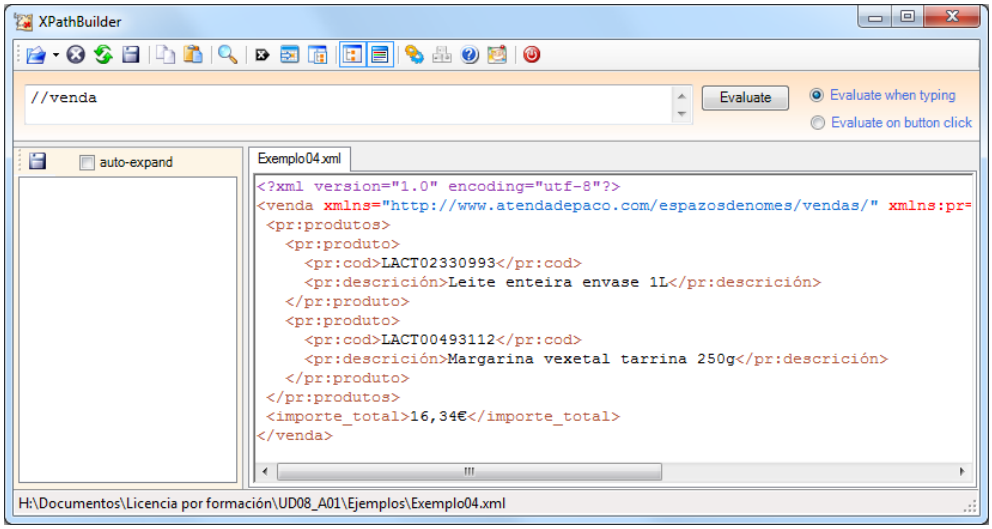
As expresións XPath sobre un documento XML con espazos de nomes, deben conter os prefixos respectivos onde corresponda. Por exemplo, a expresión correcta para obter o conxunto de nodos correspondentes a todos os produtos no seguinte documento XML sería `//pr:produto`.

```
<?xml version="1.0" encoding="utf-8"?>
<venda xmlns:pr="http://www.atendadepaco.com/espazosdenomes/produtos/">
  <pr:produtos>
    <pr:produto>
      <pr:cod>LACT02330993</pr:cod>
      <pr:descrición>Leite enteira envase 1L</pr:descrición>
    </pr:produto>
    <pr:produto>
      <pr:cod>LACT00493112</pr:cod>
      <pr:descrición>Margarina vexetal tarrina 250g</pr:descrición>
    </pr:produto>
```

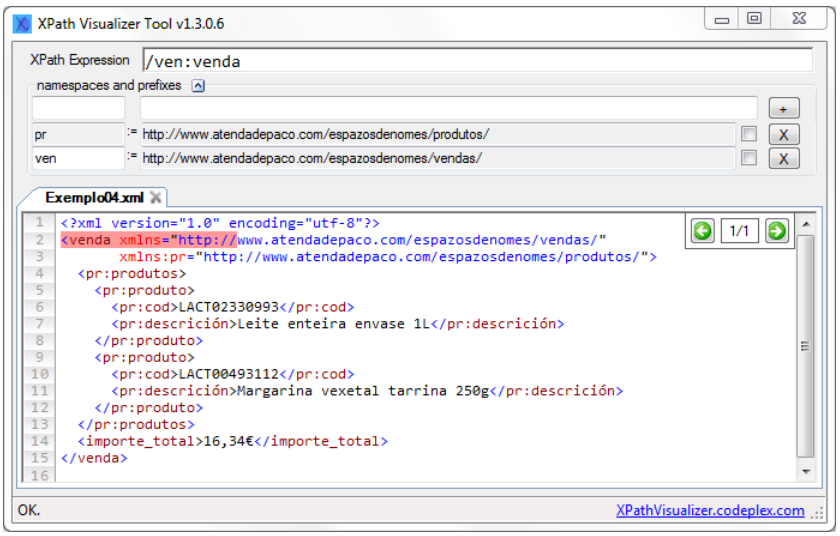
```
  </pr:produtos>
  <importe_total>16,34€</importe_total>
</venda>
```

Cando o documento ten definido un espazo de nomes por defecto, non ten un prefixo asociado ao espazo de nomes e entón poderíase pensar que a expresión `//venda` obtería os nodos descendentes no seguinte documento XML, pero non obtería ningún resultado xa que se buscarían os elementos *venda* que non estean asociados a ningún espazo de nome.

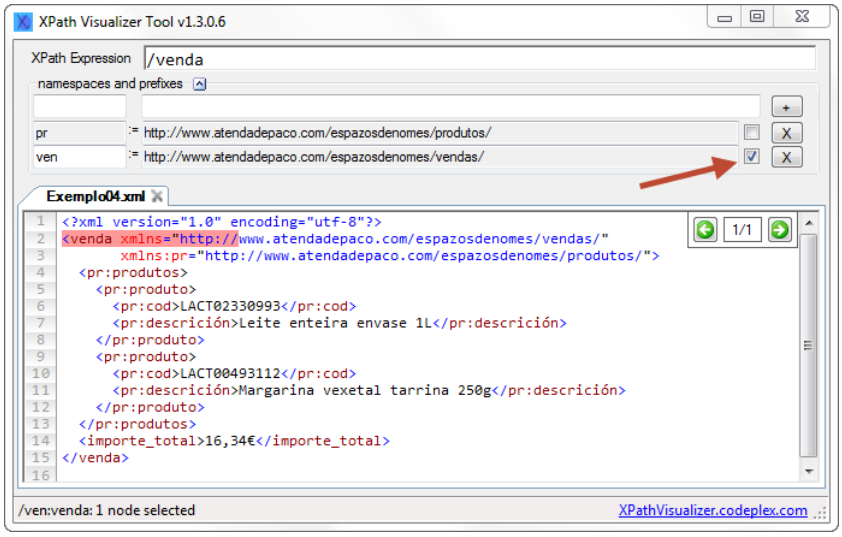
```
<?xml version="1.0" encoding="utf-8"?>
<venda
  xmlns="http://www.atendadepaco.com/espazosdenomes/vendas/"
  xmlns:pr="http://www.atendadepaco.com/espazosdenomes/produtos/">
  <pr:produtos>
    <pr:produto>
      <pr:cod>LACT02330993</pr:cod>
      <pr:descrición>Leite enteira envase 1L</pr:descrición>
    </pr:produto>
    <pr:produto>
      <pr:cod>LACT00493112</pr:cod>
      <pr:descrición>Margarina vexetal tarrina 250g</pr:descrición>
    </pr:produto>
  </pr:produtos>
  <importe_total>16,34€</importe_total>
</venda>
```



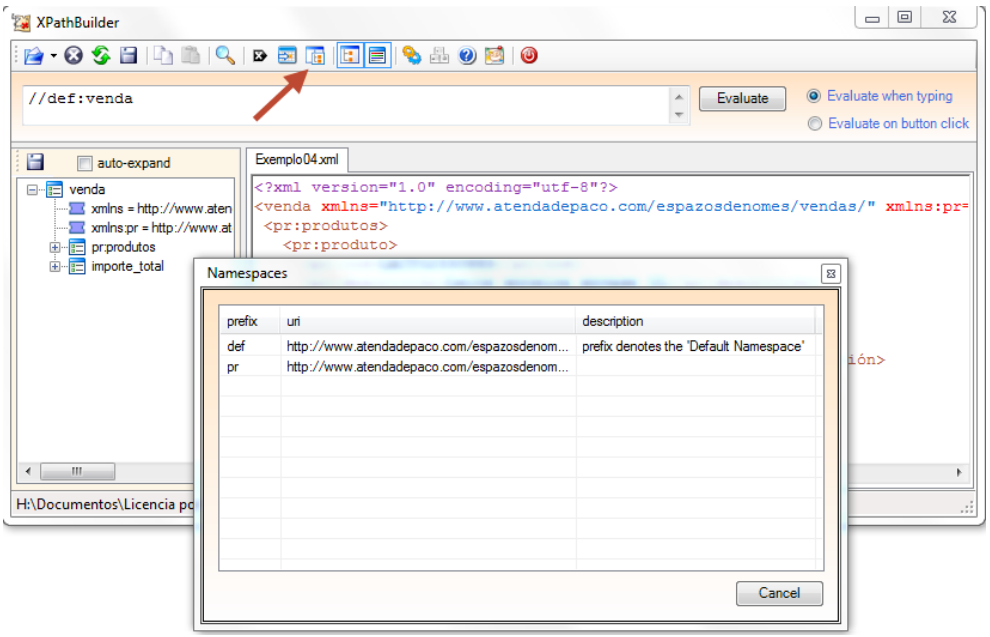
Para evitar este problema, hai que especificar dalgún xeito o prefixo que imos a empregar na expresión XPath para facer referencia ao espazo de nomes por defecto. Todos os procesadores XPath permiten especificalo dalgún xeito, e tamén a maioría de aplicacións para avaliar expresións. Por exemplo, en *XPath Visualizer* podemos despregar a sección "*namespaces and prefixes*" e modificar o prefixo a empregar para o espazo de nomes por defecto.



Ademais tamén se pode marcar un espazo de nomes "por defecto" dentro dunha expresión, de xeito que o empregue dentro da expresión se non empregamos ningún.



En *XPath Builder*, podemos ver pero non modificar o prefixo que asigna automaticamente a aplicación ao espazo de nomes por defecto e teremos que empregar ese prefixo nas nosas expresións.



2. Documentos de apoio ou referencia

- PIN RODRIGUEZ, Margarita, LOURIDO ESTÉVEZ, Víctor M. *Unidade didáctica 5(Transformación de documentos XML), actividade 1(Expresións XPath) do módulo Linguaxes de marcas e sistemas de xestión da información*. Xunta de Galicia, Consellería de Cultura, Educación e Ordenación Universitaria. 2013.