



XPath



Vanessa Braganholo
braganholo@dcc.ufrj.br

XPath

- ▶ Especificação: <http://www.w3.org/TR/xpath>
- ▶ Uma expressão XPath seleciona um conjunto de nodos
- ▶ Operadores principais:
 - ▶ / para dar um “passo” na árvore XML (percorrer uma relação pai-filho)
 - ▶ // para dar vários “passos” de uma vez (percorrer uma relação ascendente-descendente)

Exemplo

▶ /empregados/empregado

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Contexto

- ▶ Cada / muda o contexto atual da consulta:
/empregados

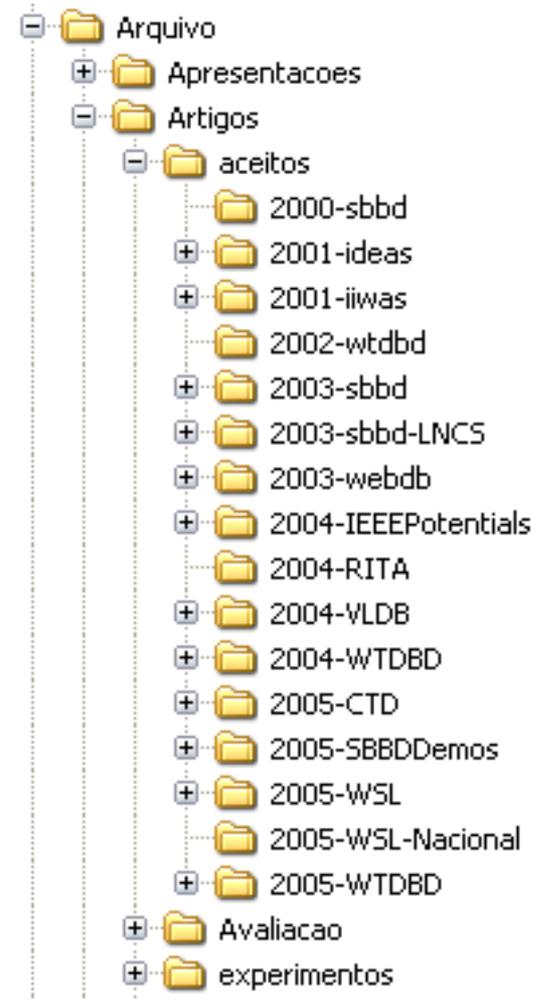
```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Contexto

- ▶ Cada / muda o contexto atual da consulta:
/empregados/empregado

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

É como uma árvore de diretórios...



```
C:\WINDOWS\system32\cmd.exe  
  
C:\Arquivo>cd Artigos  
C:\Arquivo\Artigos>cd aceitos  
C:\Arquivo\Artigos\aceitos>cd ..  
C:\Arquivo\Artigos>cd \  
C:\>
```

Retorno da Expressão

- ▶ Conjunto de nodos retornados é sempre o especificado pelo último passo do caminho

/empregados/empregado/nome

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Operador //

- ▶ Retorna os descendentes a partir do contexto atual
/empregados//nome //sobrenome

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Atributos

- ▶ Usa-se “@” na frente do nome do atributo
/empregados/empregado/@cod

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Operador .

- ▶ Selecciona o elemento corrente

/empregados/.

/empregados

- ▶ Útil para uso dentro de funções

Equivalentes!

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Operador ..

- ▶ Selecciona o pai do contexto atual
/empregados/empregado/..

```
<? xml version="1.0" ?>  
<empregados>  
  <empregado cod="E01" dept="D01">  
    <nome>João</nome>  
    <inicial-meio>S.</inicial-meio>  
    <sobrenome>Santos</sobrenome>  
  </empregado>  
  <empregado cod="E02" dept="D01">  
    <nome>Ana</nome>  
    <sobrenome>Ferraz</sobrenome>  
  </empregado>  
</empregados>
```

Operador *

- ▶ Substitui um passo do caminho

//empregado/*

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Xpath na prática...

- ▶ **Arquivo: artigo.xml**

- ▶ Disponível no Moodle
-

- ▶ **XML Exchanger Lite**

- ▶ Abrir o arquivo XML a ser consultado, digitar a expressão XPath e apertar a seta verde.

- ▶ **XPath Visualizer**

- ▶ Disponível em <http://www.topxml.com/xpathvisualizer/>
-





XPath:

Schema Outliner Editor Viewer

- Browsing/Navigation
- XPath
- XSLT Fragments
- cXML DTD
- VoiceXML RelaxNG
- XBRL XML Schema
- International
- XInclude
- XML Refactoring
- Grid Example
- Schema Viewer
- UBL XSLFO
- XSLT Debugger Sample
- Import Text
- Import Excel
- Import SQL/XML
- XML Diff and Merge
- Amazon SOAP
- Stock Quote WSDL
- Inclusive Canonicalization
- Exclusive Canonicalization
- Envelope XML Signature
- Detached XML Signature
- XQuery
- SVG

```

1  <?xml version='1.0'?>
2  <artigo data="" ultima_revisao="" versao="1">
3    <autores>
4      <autor>
5        <nome>Arnaud Sahuguet</nome>
6        <instituicao>University of Pennsylvania</instituicao>
7        <endereco></endereco>
8      </autor>
9      <autor>
10       <nome>Maria Ana</nome>
11       <instituicao>UP</instituicao>
12       <endereco></endereco>
13     </autor>
14   </autores>
15   <!-- oi -->
16   <titulo_artigo>Everything you ever wanted to know about DTD's, but were afraid to ask</titulo_artigo>
17   <resumo></resumo>
18   <secao titulo = "Introduction" numero="s1">

```

artigo.xml

Errors XPath Results Bookmarks Find in Files

XPath results for "/artigo/@versao".

```

@ versao [/artigo/@versao]

```

Document Properties

Type:

Encoding: UTF-8

Validation: Location defined in D...

Schema: No location defined.

Completion: Inferred from Document.

XPath expression:

0 of 1/1 matches

```

- <artigo data="" ultima_revisao="" versao="1" >
- <autores >
  - <autor >
    <nome >Arnaud Sahuguet </nome >
    <instituicao >University of Pennsylvania </instituicao >
    <endereco / >
  </autor >
  - <autor >
    <nome >Maria Ana </nome >
    <instituicao >UP </instituicao >
    <endereco / >
  </autor >
</autores >
<!-- oi -->
<titulo_artigo >Everything you ever wanted to know about DTD's, but were afraid to ask </titulo_artigo >
<resumo / >
- <secao titulo="Introduction" numero="s1" >
  <paragrafo >For the last TWO YEARS ... </paragrafo >
  <paragrafo >For the last two years ... </paragrafo >
  - <paragrafo >
    From a database perspective
    <citacao ref="b1" / >
    for instance, DTDs can be useful to
  </paragrafo >
</secao >
  
```

XPath Visualizer – só funciona no Internet Explorer

Exercício 1

- ▶ Usando o documento XML fornecido, crie expressões XPath para as seguintes consultas:
 - ▶ Selecionar as instituições dos autores do artigo
 - ▶ Selecionar todos os parágrafos das seções do artigo
 - ▶ Selecionar nomes dos autores do artigo propriamente dito e das referencias bibliográficas
 - ▶ Selecionar pai do elemento endereço
 - ▶ Selecionar avô do elemento paragrafo
 - ▶ Selecionar todas as ocorrências de endereço



Filtros

- ▶ Restringem o conjunto de nodos seleccionados
- ▶ Podem ser colocados em qualquer passo do caminho

Filtros

- ▶ Sintaxe: Expressão booleana entre colchetes

//empregado[@cod="E01"]

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Contexto do Filtro

- ▶ Sempre o último passo percorrido antes do filtro
- ▶ Retorno nunca é o que está no filtro, mas o último passo do caminho da expressão

//empregado[nome="Ana"]

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Filtro de Posição

//empregado[1]

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

AND, OR e NOT são aceitos

```
//empregado[@dept='D01' AND nome='João']/sobrenome
```

Atenção: o XML Exchanger Lite só aceita and e or minúsculos

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Testes de elementos

- ▶ O nome de um elemento pode aparecer representando um elemento que deve estar presente como um filho
- ▶ Selecionar um elemento empregado se ele contém um sub-elemento inicial-meio: `//empregado[inicial-meio]`

```
<? xml version="1.0" ?>
<empregados>
  <empregado cod="E01" dept="D01">
    <nome>João</nome>
    <inicial-meio>S.</inicial-meio>
    <sobrenome>Santos</sobrenome>
  </empregado>
  <empregado cod="E02" dept="D01">
    <nome>Ana</nome>
    <sobrenome>Ferraz</sobrenome>
  </empregado>
</empregados>
```

Exercício 2

- ▶ Seleccionar o autor cujo nome é Maria Ana
- ▶ Seleccionar a obra da bibliografia cujo ano é 1999 e o local é University of Pennsylvania
- ▶ Seleccionar a seção cujo número é s2 e que contém um parágrafo cujo conteúdo é ...
- ▶ Seleccionar o atributo título das seções



Funções

- ▶ XPath possui muitas funções
- ▶ A maioria delas é utilizada dentro dos filtros
- ▶ Uma lista completa está disponível aqui:
<http://www.w3.org/TR/xpath#corelib>

Testes de posição

- ▶ Função `position()` retorna a localização sequencial do elemento testado
 - ▶ Selecionar somente o primeiro parágrafo dentre os já selecionados pelo padrão

```
//empregado[position()=1]
```

ou

```
//empregado[1]
```



Testes de posição

- ▶ Função `last()` localiza o último elemento (retorna o número da posição do último elemento)
 - ▶ Selecionar o último empregado
`//empregado[last()]`
- ▶ Função `count()` retorna o número de ocorrências de um elemento
 - ▶ Selecionar seções que contenham apenas dois parágrafos
`//secao[count(paragrafo)=2]`



Função NOT

- ▶ Função `not()` para reverter o resultado do teste
 - ▶ Selecionar todas as notas, exceto a terceira
`//nota[not(position)=3]`
 - ▶ Selecionar uma nota que não contém um elemento título
`//nota[not(titulo)]`
 - ▶ Selecionar todos os capítulos, exceto aquele que tenha o atributo número com valor 10
`//capitulo[not(@numero='10')]`



Comparações

- ▶ Selecionar todos paragrafos, mas não o último

```
//paragrafo[position() != last()]
```

- ▶ Outras comparações:

```
//paragrafo[position() > 2]
```

```
//paragrafo[position() >= 3]
```

```
//paragrafo[position() > 2 and position() < last()]
```

```
//paragrafo[position() = 2 or position() = 4]
```



Tratamento de Strings

- ▶ Função `contains(par1, par2)` retorna `true` se `par1` contém o texto em `par2`

`par1` pode ser `text()` ou `.`



Função contains()

- ▶ Usando "text()", testa somente o conteúdo textual do elemento
- ▶ Selecionar título que contenha a palavra "relacional"

```
//título[contains(text(), "relacional")]
```

```
<título>Modelo relacional</título>
```



Tratamento de Strings

- ▶ Usando ".", testa o elemento secao e seus subelementos
 - ▶ Selecionar secao que contenha a palavra "relacional"

```
//secao[contains(., "relacional")]
```

```
<secao>Esta secao apresenta...
```

```
  <paragrafo>O modelo relacional ...</paragrafo>
```

```
  <paragrafo>Como já mencionado, ...</paragrafo>
```

```
</secao>
```



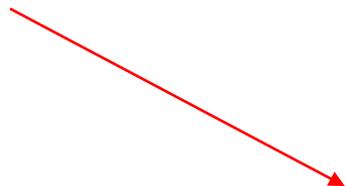
Tratamento de Strings

- ▶ Função `starts-with()` testa o texto no começo da string. Não pode haver espaço em branco.
 - ▶ Selecionar título que inicie com a palavra "Introdução"

```
//título[starts-with(., "Introdução")]
```

```
<título>Introdução a JSP</título>
```

```
<título>_Introdução a JSP</título>
```



Não funciona!



Tratamento de Strings

- ▶ Função `string()` converte o valor do argumento para string
 - ▶ Exemplo:

`string(//capitulo[1]/@numero)`, retorna o valor do atributo numero do primeiro capitulo, em formato string

- ▶ Função `normalize-space()`
 - ▶ No meio da string, reduz vários espaços em branco para um único caractere espaço
 - ▶ Remove completamente os espaços do início e fim da string

```
//titulo[contains(normalize-space(.), "Introdução a JSP")]
```

```
<titulo> Introdução a JSP </titulo>
```



Tratamento de Strings

- ▶ Função `concat()` concatena strings. Pode ter um ou mais parâmetros:
`concat(texto1, texto2,..., texton)`

- ▶ Retornar a seção que fale da autora do livro

```
//secao[contains(.,concat (./../autor/nome/text()," ",  
../autor/sobrenome/text()))]
```

```
<livro>
```

```
  <autor>
```

```
    <nome>Ana</nome>
```

```
    <sobrenome>Silva</sobrenome>
```

```
  </autor>
```

```
  <secao> A autora Ana Silva ...</secao>
```



```
  <secao> ... </secao>
```

```
</livro>
```



Tratamento de Strings

- ▶ Função `translate()` converte caracteres de acordo com um esquema de mapeamento.
 - ▶ Uso: comparações case-insensitive
 - ▶ Parâmetros: string para converter, caracteres para modificar no texto fonte, e valores a serem colocados

```
//paragrafo[contains(translate(normalize-space(.), "abcdefghijklmnopqrstuvwxyz", "ABCDEFGHIJKLMNOPQRSTUVWXYZ"), "ELEMENTO")]
```

<paragrafo>EM XML, UM ELEMENTO É ...</paragrafo>

<paragrafo>Um documento XML deve possuir um elemento raiz ...</paragrafo>

- ▶ os dois elementos paragrafo são recuperados



Tratamento de Números

- ▶ Operadores + e - podem ser usados:
 - ▶ `//nota[3]` é equivalente a `//nota[1 + 2]`
- ▶ Função `mod()`
 - ▶ Fornece o resto de uma divisão truncada
 - ▶ Selecionar parágrafos pares:
`//paragrafo[position() mod 2 = 0]`



Filtros Múltiplos

- ▶ Usados para combinar um teste de posição e um outro tipo de teste
 - ▶ Selecionar nomes de companhias, e então extrair o terceiro nome da lista
`//nome[companhia][3]`

...

```
<nomes>
```

```
<nome>< Pessoa>...</ Pessoa></nome>
```

```
<nome>< companhia>...</ companhia></nome>
```

```
<nome>< Pessoa>...</ Pessoa></nome>
```

```
</nomes>
```

```
<nomes>
```

```
<nome>< Pessoa>...</ Pessoa></nome>
```

```
<nome>< companhia>...</ companhia></nome>
```

```
<nome>< Pessoa>... </ Pessoa></nome>
```

```
</nomes>
```

...



Filtros Múltiplos

- ▶ Usados para combinar um teste de posição e um outro tipo de teste
 - ▶ Selecionar o terceiro nome, fornecendo o nome da companhia (só seleciona se for uma companhia!)

```
//nome[3][companhia]
```

...

```
<nomes>
```

```
<nome>< Pessoa>...</ Pessoa></ nome>
```

```
<nome>< companhia>...</ companhia></ nome>
```

```
<nome>< Pessoa>...</ Pessoa></ nome>
```

```
</ nomes>
```

```
<nomes>
```

```
<nome>< Pessoa>...</ Pessoa></ nome>
```

```
<nome>< companhia>...</ companhia></ nome>
```

```
<nome>< Pessoa>...</ Pessoa></ nome>
```

```
</ nomes>
```

...



Outras funções

- ▶ **Comentários:**

- ▶ Selecione comentários que estejam dentro de livro

`//livro/comment()`

- ▶ **Instruções de processamento**

- ▶ Encontre instruções de processamento que estejam dentro do elemento livro

`//livro/processing-instruction()`



Exercício 3

- ▶ Usando o mesmo documento XML, crie expressões XPath (pode ser necessário usar o resumo das funções XPath que está nos próximos slides)
 - ▶ Retorne todas as seções do artigo que possuem pelo menos um subelemento figura (use a função boolean)
 - ▶ Selecione as seções ímpares
 - ▶ Selecione as seções ímpares que possuem pelo menos um subelemento figura
 - ▶ Selecione os parágrafos que contêm uma instrução de processamento
 - ▶ Encontre autores que possuam "Ana" no nome. Não devem estar nas referencias bibliográficas.
 - ▶ A versão do artigo
 - ▶ Selecione o parágrafo que tenha ambos os atributos "numero" e "tipo"
 - ▶ Nome do(s) autor(res) da bibliografia cujo titulo da obra é "Union Types for Semistructured Data ".
 - ▶ Selecione o ano da publicação de bibliografias, cujo nome do autor contenha a palavra "Abiteboul"



Exercício 3

- ▶ Um parágrafo cujo tamanho da string que ele contém é 27
- ▶ Selecione o terceiro parágrafo de uma seção
- ▶ Selecione um parágrafo que contenha o nome do segundo autor do artigo e sua instituição. Entre estes dois dados existe a string "from"
- ▶ Selecione todo e qualquer parágrafo que tenha a string "two years". Trate o uso de maiúsculas e minúsculas!!
- ▶ Selecionar seções que contenham somente dois parágrafos
- ▶ Selecione os elementos "ano" descendentes de bibliografia
- ▶ Encontre a instrução de processamento de algum elemento paragrafo.



Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na

página da W3C

✓ Funções para *nodos* (elementos)

Nome	Sintaxe	Descrição
<code>count()</code>	<code>count(node-set) = number</code>	Retorna o número de nodos de um node-set
<code>id()</code>	<code>id(value) = node-set</code>	Seleciona elementos pelo seu ID único
<code>last()</code>	<code>last() = number</code>	Retorna o número da posição do ultimo nodo em uma lista de nodos processados
<code>local-name()</code>	<code>local-name(node) = string</code>	Retorna a parte local de um nodo. Um nodo geralmente consiste de um prefixo, uma vírgula e seguida de um nome local
<code>name()</code>	<code>name(node) = string</code>	Retorna o nome de um nodo
<code>namespace-uri()</code>	<code>namespace-uri(node) = uri</code>	Retorna a URI da <i>namespace</i> de um nodo específico
<code>position()</code>	<code>position() = number</code>	Retorna a posição em uma lista de nodos do nodo que está sendo processado



Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na

página da W3C

✓ Funções para *string*

Nome	Sintaxe e Exemplo	Descrição
concat()	<code>string=concat(val1, val2, ..)</code> Exemplo: <code>concat('The', ' ', 'XML')</code> Resultado: 'The XML'	Retorna a concatenação de todos os seus argumentos
contains()	<code>bool=contains(val, substr)</code> Exemplo: <code>contains('XML', 'X')</code> Resultado: true	Retorna true se a segunda string está contida na primeira
normalize-space()	<code>string=normalize-space(string)</code> Exemplo: <code>normalize-space(' The XML')</code> Resultado: 'The XML'	Normaliza os espaços em brancos para um só
starts-with()	<code>bool=starts-with(string, substr)</code> Exemplo: <code>starts-with('XML', 'X')</code> Resultado: true	Retorna true se a primeira string inicia com a segunda
string()	<code>string(value)</code> Exemplo: <code>string(314)</code> Resultado: '314'	Converte o valor do argumento para string

Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na

página da W3C

✓ Funções para *string*

Nome	Sintaxe e Exemplo	Descrição
string-length()	number=string-length(string) Exemplo: string-length('Beatles') Resultado: 7	Retorna o número de caracteres em uma string
substring()	string=substring(string,start,length) Exemplo: substring('Beatles',1,4) Resultado: 'Beat'	Retorna a parteda string indicada nos argumentos
substring-after()	string=substring-after(string,substr) Exemplo: substring-after('12/10','/') Resultado: '10'	Retorna a parte da string que está depois do argumento substr
substring-before()	string=substring-before(string,substr) Exemplo: substring-before('12/10','/') Resultado: '12'	Retorna a parteda string que está antes do argumento substr
translate()	string=translate(value,string1,string2) Exemplo: translate('12:30','30','45') Resultado: '12:45' translate('12:30','03','54') Resultado: '12:45' translate('12:30','0123','abcd') Resultado: 'bc:da'	Executa reposição character a character.

Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na página da W3C

✓ Funções para *numéricos*

Nome	Sintaxe e Exemplo	Descrição
ceiling()	ceiling(number) = number Exemplo: ceiling(3.14) Resultado: 4	Retorna o menor inteiro que não pe menor do que o argumento
floor()	floor(number) = number Exemplo: floor(3.14) Resultado: 3	Retorna o maior inteiro que não é maior do que o argumento
number()	number(value) = number Exemplo: number('100') Resultado: 100	Converte o valor do argumento para um numérico
round()	round(number)= integer Exemplo: round(3.14) Resultado: 3	Arredonda o argumento ao inteiro mais próximo
sum()	sum(nodeset) =number Exemplo: sum(/cd/price)	Retorna o valor total de um conjunto numérico de valores em um node-set



Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na página da W3C

✓ Funções *booleanas*

Nome	Sintaxe e Exemplo	Descrição
<code>boolean()</code>	<code>bool=boolean(value)</code>	Converte o argumento e retorna true ou false
<code>false()</code>	<code>false()</code> Exemplo: <code>number(false())</code> Resultado: 0	Retorna false
<code>lang()</code>	<code>bool=lang(language)</code>	Retorna true se a linguagem do argumento casa com a linguagem do elemento <code>xsl:lang</code>
<code>not()</code>	<code>bool=not(condition)</code> Exemplo: <code>not(false())</code>	Retorna true se a condição de argumento for falsa, e falsa se a condição for verdadeira
<code>true()</code>	<code>true()</code> Exemplo: <code>number(true())</code> Resultado: 1	Retorna true



Xpath e Java

- ▶ Java tem um pacote para lidar com expressões XPath
 - ▶ **javax.xml.xpath**
 - ▶ Documentação:
<http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/xpath/package-summary.html>

