
# Integração de Sistemas e XML

PROF. Sergio Serra

# Navegação e Transformação em XML

Aula 03


Núcleo de Computação Eletrônica

Universidade Federal do Rio de Janeiro

# Roteiro da Aula

1. Navegação – **XPath**

2. Transformação - **XSLT**

# XPath

- O significado de um elemento pode depender de seu contexto
- **Navegação dentro do documento XML para encontrar este contexto**
- **Encontrar os elemento através de caminhos**
- Para chegar a um elemento:
  - Como em URL:
    - Uso de **caminho absoluto** - Especificar toda a hierarquia de elementos de uma árvore XML
    - Uso de **caminho relativo** - Especificar, em qualquer ponto do caminho, elementos relativos ao elemento contexto
- Explorar a estrutura hierárquica do documento

# XPath

- Localizar com precisão um elemento de interesse
- Exemplo: **Obter o último nome do autor de um livro**

```
<livro>
    <autor>
      <nome>
        <primeiro>John</primeiro>
        <ultimo>Smith</ultimo>  --->> RESULTADO CORRETO
      </nome>
    </author>      ...
    <chapter>
      <autor>
       <nome>
         <primeiro>John</primeiro>
         <ultimo>Smith</ultimo>  --->> RESULTADO INCORRETO
       </nome>
      </author>
     </chapter>
 </livro>
```

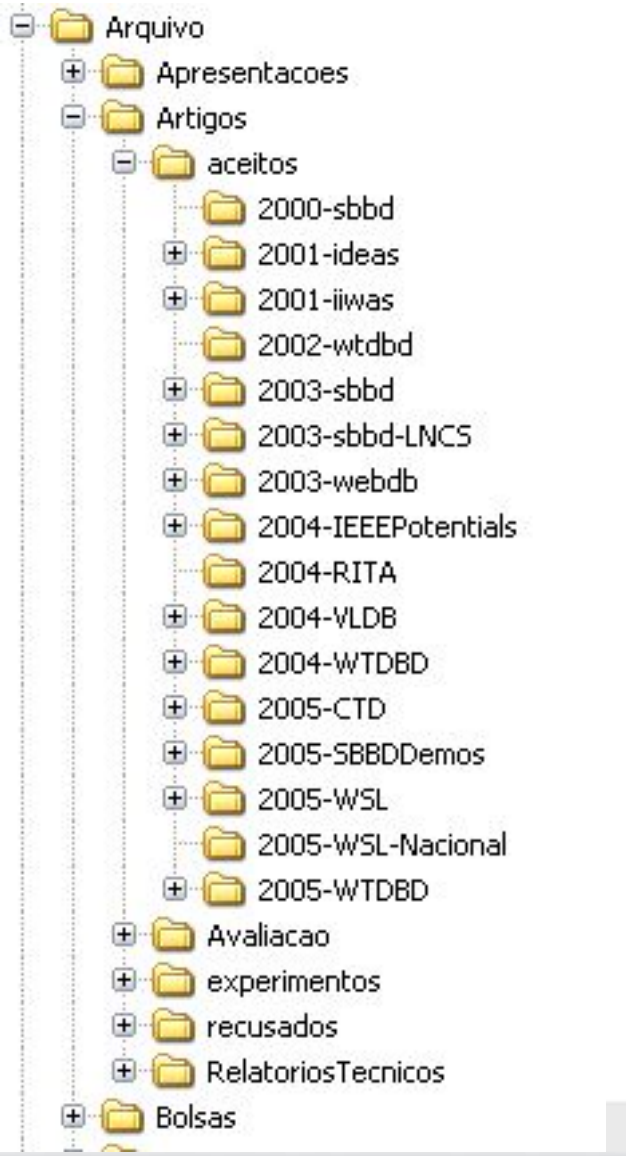
# XPath

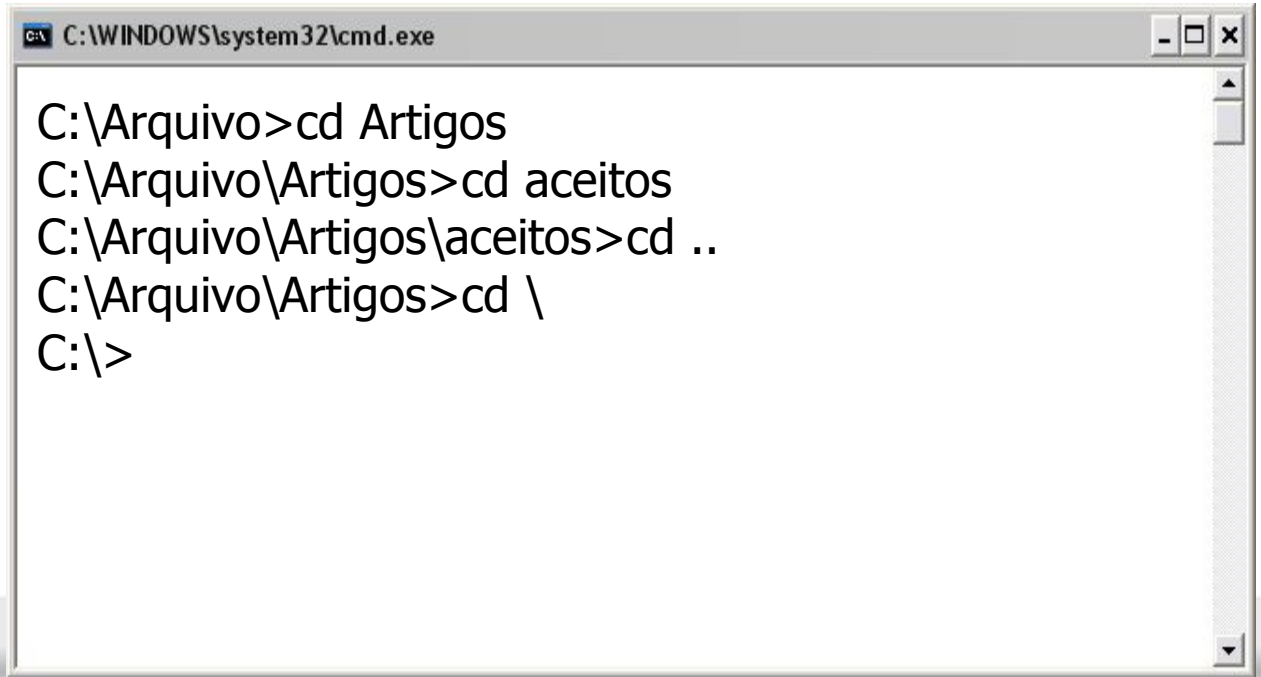
- XPath usa expressões
  - *Strings* com **símbolos significativos**;
  - Identificar o elemento titulo que é filho direto de um elemento livro:

livro/titulo

String com símbolo significativo "/"

# É como uma árvore de diretórios...



# XPath - Expressões

- Expressão é uma string de texto que consiste de instruções para selecionar alguma estrutura de marcação
- Símbolos especiais:

  "/", "*", ".", ".."
- Expressão mais simples:

  //livro
- Expressões podem ser usadas para selecionar
  - atributos, comentários, instruções de processamento, ou seqüência de texto entre elementos

# Caminhos

- Usado quando quer selecionar algo, independente do contexto corrente
  - Para indicar a raiz do documento, usa-se "/"

    /book/title
  - Selecionar ocorrências de um elemento cujos antecessores não importam, usa-se "//"

    //para

# Caminhos

- Quando os nomes de elementos entre o elemento contexto e o descendente requerido não é conhecido use “*”
  - Ex: selecionar títulos de capítulos e da introdução

    /livro/introducao/titulo e /livro/capitulo/titulo

    **/livro/*/titulo**
- Múltiplos "*" podem ser usados para selecionar um maior nível de profundidade
  - Um “*” para cada nível

# Caminhos

- Uma forma abreviada para os múltiplos "*" é //
  - Ex: selecionar todos os elementos paragrafo que ocorrem em qualquer lugar de capitulo

```
/capitulo//paragrafo
```

- Para selecionar todos os paragrafo **diretamente** dentro do elemento corrente:

```
//paragrafo
```

- Seleção do elemento corrente:

```
.
```

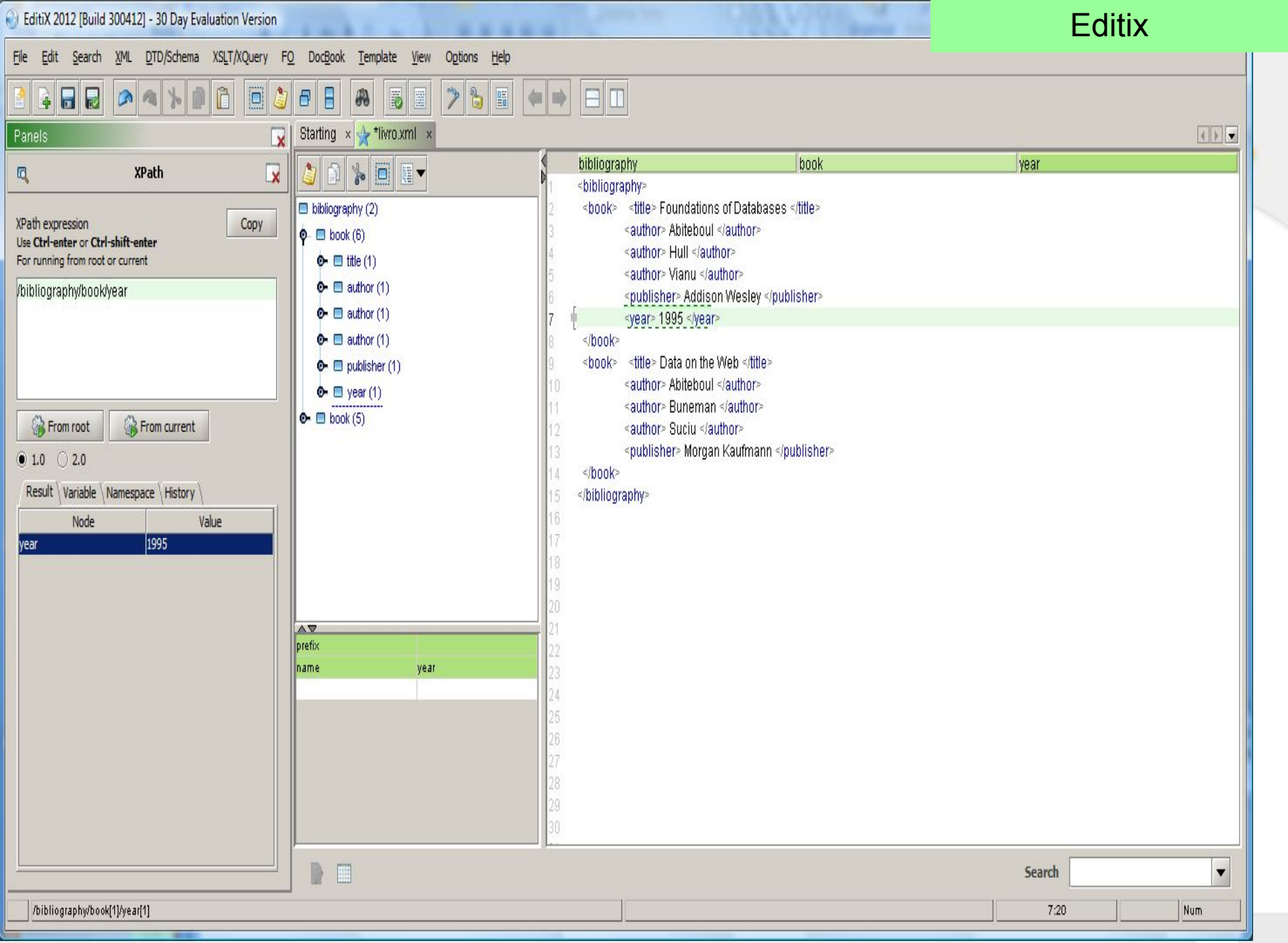
# Brincando com XPath...

- Arquivo: artigo.xml
  - Disponível no moodle

---

- Faça download/Use o **Editix**
  - Abrir o arquivo XML a ser consultado, digitar a expressão XPath e apertar a seta verde.

- XPath Visualizer (opcional)
  - Disponível em http://xpathvisualizer.codeplex.com/


Editix
EditiX 2012 [Build 300412] - 30 Day Evaluation Version
File Edit Search XML DTD/Schema XSLT/XQuery FO DocBook Template View Options Help
Panels
XPath
XPath expression
Use Ctrl-enter or Ctrl-shift-enter
For running from root or current
Copy
/bibliography/book/year
From root
From current
1.0
2.0
Result
Variable
Namespace
History
Node
Value
year
1995
Starting
*livro.xml
bibliography (2)
book (6)
title (1)
author (1)
author (1)
author (1)
publisher (1)
year (1)
book (5)
prefix
name
year
bibliography
book
year
<bibliography>
<book> <title> Foundations of Databases </title>
<author> Abiteboul </author>
<author> Hull </author>
<author> Vianu </author>
<publisher> Addison Wesley </publisher>
<year> 1995 </year>
</book>
<book> <title> Data on the Web </title>
<author> Abiteboul </author>
<author> Buneman </author>
<author> Suciu </author>
<publisher> Morgan Kaufmann </publisher>
</book>
</bibliography>
Search
/bibliography/book[1]/year[1]
7:20
Num

# Exercício 1

Usando o documento XML fornecido (artigo.xml), crie expressões XPath para as seguintes consultas:

1. Selecionar as instituições dos autores do artigo
2. Selecionar todos os parágrafos das seções do artigo
3. Selecionar nomes dos autores do artigo propriamente dito e das referencias bibliográficas
4. Selecionar pai do elemento endereco
5. Selecionar avô do elemento paragrafo
6. Selecionar todas as ocorrências de endereço

# Filtros

- Usados para remover itens indesejados de uma lista para criar uma nova lista.
  - Ex: Selecionar os parágrafos de todos os capítulos, no entanto somente o primeiro parágrafo em cada capítulo.

    /book/chapter/para[1]
- Usa "[", e "]", para manipular o predicado. Os resultados do teste são um valor *booleano*, e a seleção só ocorre quando o valor é *true*.

# Testes de elementos e atributos

- Elementos
  - O nome de um elemento pode aparecer representando um elemento que deve estar presente como um filho
    - Ex. Selecionar um elemento nota se ele contém diretamente um elemento tituto : /nota[titulo]
  - O valor de um elemento pode ser testado:
    - Selecionar a nota cujo titulo seja "Inicial": /nota[titulo="Inicial"]

# Comparações

- É possível também utilizar outros tipos de comparações

```
//capitulo[@numero>5]
```

# Testes de elementos e atributos

- Atributos
  - podem ser selecionados dependendo do valor ou da existência  ou não de um atributo
  - O símbolo @ é usado para representar um atributo e precede o nome do atributo
    - Selecionar o atributo autor do elemento livro: /livro/@autor
    - Seleciona todo parágrafo com o valor do atributo tipo igual a ‘secreto’:  /para[@tipo='secreto']

# Exercício 2

1. Selecionar o autor cujo nome é Maria Ana
2. Selecionar a obra da bibliografia cujo ano é 1999 e o local é University of Pennsylvania
3. Selecionar a seção cujo número é s2 e que contém um parágrafo cujo conteúdo é ...
4. Selecionar o atributo título das seções

# Funções

- Os filtros em XPath podem incluir funções para tratar strings, booleanos, etc.
- **Os próximos slides trazem um resumo destas funções**

- Exemplo: /livro[starts-with(@autor, “John”)]

  Retorna os livros que possuem um atributo autor cujo conteúdo começa com a string “John”

# Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na página da W3C


✔ Funções para *nodos (elementos)*

| ***Nome*** | ***Sintaxe*** | ***Descrição*** |
|---|---|---|
| `count()` | `count(node-set) = number` | Retorna o número de nodos de um node-set |
| `id()` | `id(value) = node-set` | Seleciona elementos pelo seu ID único |
| `last()` | `last() = number` | Retorna o número da posição do ultimo nodo em uma lista de nodos processados |
| `local-name()` | `local-name(node)= string` | Retorna a parte local de um nodo. Um nodo geralmente consiste de um prefixo, uma vírgula e seguida de um nome local |
| `name()` | `name(node) = string` | Retorna o nome de um nodo |
| `namespace-uri()` | `namespace-uri(node) = uri` | Retorna a URI da *namespace* de um nodo específico |
| `position()` | `position() = number` | 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()` | `string=concat(val1, val2, ..)`<br>**Exemplo:**<br>`concat('The',' ','XML')`<br>`Resultado: 'The XML'` | Retorna a concatenação de todos os seus argumentos |
| `contains()` | `bool=contains(val,substr)`<br>**Exemplo:**<br>`contains('XML','X')`<br>`Resultado: true` | Retorna true se a segunda string está contida na primeira |
| `normalize-space()` | `string=normalize-space(string)`<br>**Exemplo:**<br>`normalize-space(' The XML ')`<br>`Resultado: 'The XML'` | Normaliza os espaços em broncos para um só |
| `starts-with()` | `bool=starts-with(string,substr)`<br>**Exemplo:**<br>`starts-with('XML','X')`<br>`Resultado: true` | Retorna true se a primeira string inicia com a segunda |
| `string()` | `string(value)`<br>**Exemplo:**<br>`string(314)`<br>`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)`<br>**Exemplo:**<br>`string-length('Beatles')`<br>`Resultado: 7` | Retorna o número de caracteres em uma string |
| `substring()` | `string=substring(string,start,length)`<br>**Exemplo:**<br>`substring('Beatles',1,4)`<br>`Resultado: 'Beat'` | Retorna a parted a string indicada nos argumentos |
| `substring-after()` | `string=substring-after(string,substr)`<br>**Exemplo:**<br>`substring-after('12/10','/')`<br>`Resultado: '10'` | Retorna a parte da string que está depois do argumento substr |
| `substring-before()` | `string=substring-before(string,substr)`<br>**Exemplo:**<br>`substring-before('12/10','/')`<br>`Resultado: '12'` | Retorna a parted a string que está antes do argumento substr |
| `translate()` | `string=translate(value,string1,string2)`<br>**Exemplo:**<br>`translate('12:30','30','45')`<br>`Resultado: '12:45'`<br>`translate('12:30','03','54')`<br>`Resultado: '12:45'`<br>`translate('12:30','0123','abcd')`<br>`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`<br>**`Exemplo:`**<br>`ceiling(3.14)`<br>`Resultado: 4` | Retorna o menor inteiro que não pe menor do que o argumento |
| `floor()` | `floor(number) = number`<br>**`Exemplo:`**<br>`floor(3.14)`<br>`Resultado: 3` | Retorna o maior inteiro que não é maior do que o argumento |
| `number()` | `number(value) = number`<br>**`Exemplo:`**<br>`number('100')`<br>`Resultado: 100` | Converte o valor do argumento para um numérico |
| `round()` | `round(number)= integer`<br>**`Exemplo:`**<br>`round(3.14)`<br>`Resultado: 3` | Arredonda o argumento ao inteiro mais próximo |
| `sum()` | `sum(nodeset)=number`<br>**`Exemplo:`**<br>`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*** |
|---|---|---|
| `boolean()` | `bool=boolean(value)` | Converte o argumento e retorna true ou false |
| `false()` | `false()`<br><br>**`Exemplo:`**<br>`number(false())`<br>`Resultado: 0` | Retorna false |
| `lang()` | `bool=lang(language)` | Retorna true se a linguagem do argumento casa com a linguagem do elemento xsl:lang |
| `not()` | `bool=not(condition)`<br><br>**`Exemplo:`**<br>`not(false())` | Retorna true se a condição de argumento for falsa, e falsa se a condição for verdadeira |
| `true()` | `true()`<br><br>**`Exemplo:`**<br>`number(true())`<br>`Resultado: 1` | Retorna true |

# Java e XPath

- 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

# Transformação de Documentos XML
# XSLT

# Importância de XSLT

- XSLT é um padrão para **transformação de documentos XML para qualquer representação textual**
  - *Templates* de transformação são aplicados a objetos XML
  - **Entrada**: documento XML
  - **Saída**: qualquer documento em formato texto (HTML, XML, TXT, RTF, etc)

# XSLT

- Um processador XSLT
  1. recebe como entrada um documento XML
  2. gera na saída um outro documento em formato texto
- Se o documento de saída for um documento XML
  - ele pode estar estruturado de acordo com uma DTD diferente da DTD do documento de entrada
- A transformação é especificada em um *style sheet*
- **Um *style sheet* segue a sintaxe do padrão XML**

# Princípio de funcionamento da transformação

Documento de entrada

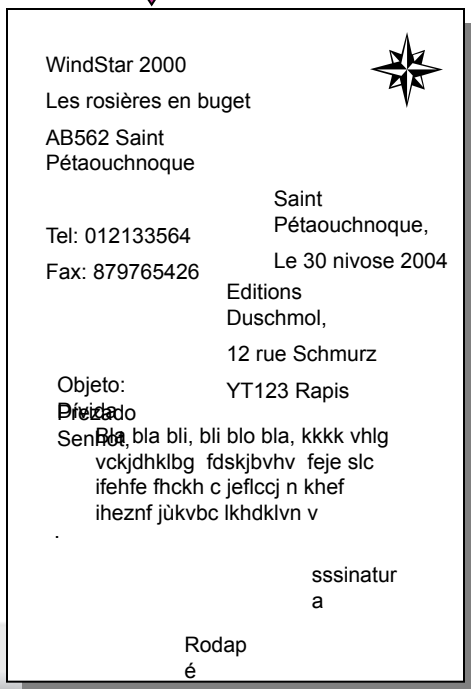
```
<carta>
  <cabecalho>
    . . .
  </cabecalho>

  <corpo>
    . . .
  </corpo>
</carta>
```

```
...
<template match="cabecalho">
   <apply-templates/>
</template>
...
```

Regras de transformação – stylesheets

Processador XSLT

Documento Transformado



# *Style Sheets*

- Um *style sheet* é formado por um conjunto de regras (*template)*
  - transformações são executadas de acordo com tais regras
- Cada regra "casa" com um tipo de elemento no documento de entrada utilizando expressões **XPath**
- As *tags* originais são substituídas por novas *tags* de saída

# Estrutura Geral

- O elemento raiz é denominado **stylesheet**
- O namespace de XSLT deve ser declarado xmlns:xsl=http://www.w3.org/1999/XSL/Transform

```
<?xml version="1.0"?>
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">
....
</xsl:stylesheet>
```

# Estrutura Geral

- Templates (ou regras) são criadas através do elemento <xsl:template>

```
<xsl:template match="title">
  ...
</xsl:template>
```

- Recursividade de templates é criada através do elemento <xsl:apply-templates>

```
<xsl:template match="book">
    ...
    <xsl:apply-templates/>
</xsl:template>
```

# Funcionamento

- O processo de transformação
  - XSLT usa XPath
    - Define partes do documento fonte que corresponde a um ou mais *templates* definidos
  - Quando uma correspondência é encontrada
    - XSLT transforma a parte correspondente do documento fonte em um documento resultado
  - As partes do documento fonte que não correspondem ao *template* não aparecem no resultado final

# Processamento

- Processador percorre documento XML
  - Encontra um elemento
  - Acessa a *style-sheet* para encontrar um template corresponde ao elemento do documento XML
  - Executa o *template*

# Exemplo Documento XML

```
<booklist>
    <book id="BOX00">
     <author>Box, D. and Skonnard, A. and Lam, J.</author>
     <editor>Series</editor>
     <title>Essential XML - Beyond Markup</title>
     <publisher>Addison-Wesley</publisher>
     <year>2000</year>
     <key/>
     <volume/>
     <number/>
     <series/>
     <address/>
     <edition/>
     <month>July</month>
     <note/>
     <annote/>
     <url>http://www.develop.com/books/essentialxml</url>
    </book>
    .....
```

# Exemplo Documento XML

```
.....
<book id="MAR99">
 <author>Maruyama, H. and Tamura, K. and Uramoto, N.</author>
 <title>XML and Java: Developing of Web Applications</title>
 <publisher>Addison-Wesley</publisher>
 <year>1999</year>
 <address>MA</address>
 <month>August</month>
</book>
<book id="BRA00">
 <author>Bradley, N.</author>
 <title>The XML Companion</title>
 <publisher>Addison-Wesley</publisher>
 <year>2000</year>
 <address>Great Britain</address>
 <edition>2</edition>
 <month>August</month>
</book>
</booklist>
```

# Processamento recursivo

```
<?xml version="1.0"?>
<books>
  <book>
    <title>ABC</title>
    <author>John</author>
  </book>
  <book>
    <title>DEF</title>

<author>Joseph</author>
  </book>
</books>
```

```
<xsl:stylesheet version="1.0"
 xmlns:xsl:....>

  <xsl:template
match="books">
      ...
      <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="book">
      ...
      <xsl:apply-templates/>
  </xsl:template>

</xsl:stylesheet>
```

# Processamento recursivo

```
<?xml version="1.0"?>
<books>
  <book>
    <title>ABC</title>
    <author>John</author>
  </book>
  <book>
    <title>DEF</title>

<author>Joseph</author>
  </book>
</books>
```

```
<xsl:stylesheet version="1.0"
 xmlns:xsl:....>

   <xsl:template
match="books">
      ...
      <xsl:apply-templates/>
   </xsl:template>

   <xsl:template match="book">
      ...
      <xsl:apply-templates/>
   </xsl:template>

</xsl:stylesheet>
```

# Processamento recursivo

```
<?xml version="1.0"?>
<books>
  <book>
    <title>ABC</title>
    <author>John</author>
  </book>
  <book>
    <title>DEF</title>

<author>Joseph</author>
  </book>
</books>
```

```
<xsl:stylesheet version="1.0"
 xmlns:xsl:....>

   <xsl:template
match="books">
      ...
      <xsl:apply-templates/>
   </xsl:template>

   <xsl:template match="book">
      ...
      <xsl:apply-templates/>
   </xsl:template>

</xsl:stylesheet>
```

# Processamento recursivo

```
<?xml version="1.0"?>
<books>
  <book>
    <title>ABC</title>
    <author>John</author>
  </book>
  <book>
    <title>DEF</title>

<author>Joseph</author>
  </book>
</books>
```

```
<xsl:stylesheet version="1.0"
 xmlns:xsl:....>

   <xsl:template
match="books">
       ...
       <xsl:apply-templates/>
   </xsl:template>

   <xsl:template match="book">
       ...
       <xsl:apply-templates/>
   </xsl:template>

</xsl:stylesheet>
```

# Processamento recursivo

```
<?xml version="1.0"?>
<books>
  <book>
    <title>ABC</title>
    <author>John</author>
  </book>
  <book>
    <title>DEF</title>

<author>Joseph</author>
  </book>
</books>
```

```
<xsl:stylesheet version="1.0"
 xmlns:xsl:....>

   <xsl:template
match="books">
      ...
      <xsl:apply-templates/>
   </xsl:template>

   <xsl:template match="book">
      ...
      <xsl:apply-templates/>
   </xsl:template>

</xsl:stylesheet>
```

# Regras *default*

- XSLT possui algumas regras *default*
- Ex: Uma *stylesheet* vazia aplica as regras *default* ao documento XML que está sendo processado
  - Processa o documento todo
  - Coloca todo o conteúdo dos elementos texto na saída

# Exemplo (book.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output  method="text" indent="yes"/>



</xsl:stylesheet>
```

# Resultado

Box, D. and Skonnard, A. and Lam, J.SeriesEssential XML - Beyond MarkupAddison-Wesley2000Julyhttp://www.develop.com/books/essentialxmlMaruyama, H. and Tamura, K. and Uramoto, N.XML and Java: Developing of Web ApplicationsAddison-Wesley1999MAAugustBradley, N.The XML CompanionAddison-Wesley2000Great Britain2August

# Regras *default*

- Regra 1:
  - Processar todo o documento XML

```
<xsl:template match="/|*">
   <xsl:apply-templates/>
</xsl:template>
```

- xsl:apply-templates faz com que os filhos do nodo atual sejam processados recursivamente

# Regras *default*

- Regra 2:
  - Copiar o conteúdo texto dos elementos para a saída

```
<xsl:template match="text()">
   <xsl:value-of select="."/>
</xsl:template>
```

- xsl:value-of select=“.” faz com que o conteúdo do nodo atual seja copiado para a saída

# Exemplo (book_1.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
  <xsl:output  method="text" indent="yes"/>

    <xsl:template match="booklist">
    ================Lista de Livros================
      <xsl:apply-templates/>
    ================Fim da lista==================
    </xsl:template>


</xsl:stylesheet>
```

# Resultado

==========Lista de livros==========

Box, D. and Skonnard, A. and Lam, J.SeriesEssential XML - Beyond MarkupAddison-Wesley2000Julyhttp://www.develop.com/books/essentialxmlMaruyama, H. and Tamura, K. and Uramoto, N.XML and Java: Developing of Web ApplicationsAddison-Wesley1999MAAugustBradley, N.The XML CompanionAddison-Wesley2000Great Britain2August

==========Fim da lista==========

# Função name()

- XPath possui uma função name() que pode ser usada para imprimir o nome do elemento que casou com uma determinada regra

```
<xsl:template match="book">
  <xsl:value-of select="name() ‘/>
</xsl:template>
```

# Exemplo (book_2.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
  <xsl:output  method="text" indent="yes"/>

  <xsl:template match="*">
      <xsl:value-of select="name()"/>
      <xsl:apply-templates/>
   </xsl:template>

</xsl:stylesheet>
```

# Resultado

booklist: book: author: Box, D. and Skonnard, A. and Lam, J.editor: Seriestitle: Essential XML - Beyond Markuppublisher: Addison-Wesleyyear: 2000key: volume: number: series: address: edition: month: Julynote: annote: url: http://www.develop.com/books/essentialxmlbook: author: Maruyama, H. and Tamura, K. and Uramoto, N.title: XML and Java: Developing of Web Applicationspublisher: Addison-Wesleyyear: 1999address: MAmonth: Augustbook: author: Bradley, N.title: The XML Companionpublisher: Addison-Wesleyyear: 2000address: Great Britainedition: 2month: August

# Exemplo (book_3.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
  <xsl:output  method="text" indent="yes"/>

    <xsl:template match="*">
    </xsl:template>

    <xsl:template match="booklist">
    ================Lista de livros================
      <xsl:apply-templates/>
    ================Fim da lista==================
    </xsl:template>


</xsl:stylesheet>
```

# Resultado

==========Lista de livros==========

==========Fim da lista==========

# Exemplo (book4.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
  <xsl:output  method="text" indent="yes"/>


  <xsl:template match="author">
    ==================Livro do autor==================
     <xsl:apply-templates />
    ==================Detalhes do livro===============
  </xsl:template>

</xsl:stylesheet>
```

# Resultado

==========Livro do autor==========
Box, D. and Skonnard, A. and Lam, J.
==========Detalhes do livro==========
SeriesEssential XML - Beyond MarkupAddison-Wesley2000Julyhttp://www.develop.com/books/essentialxml
==========Livro do autor==========
Maruyama, H. and Tamura, K. and Uramoto, N.
==========Detalhes do livro==========
XML and Java: Developing of Web ApplicationsAddison-Wesley1999MAAugust
==========Livro do autor==========
Bradley, N.
==========Detalhes do livro==========
The XML CompanionAddison-Wesley2000Great Britain2August

# Exemplo (book_5.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
  <xsl:output  method="text" indent="yes"/>

    <xsl:template match="booklist">
    ================Lista de livros================
      <xsl:apply-templates/>
    ================Fim da lista==================
    </xsl:template>

    <xsl:template match="author">
    ================Livro do autor================
      <xsl:apply-templates/>
    ================Detalhes do livro=============
    </xsl:template>
</xsl:stylesheet>
```

# Resultado

==========Lista de livros==========

==========Livro do autor==========
Box, D. and Skonnard, A. and Lam, J.
==========Detalhes do livro==========
SeriesEssential XML - Beyond MarkupAddison-Wesley2000Julyhttp://www.develop.com/books/essentialxml
==========Livro do autor==========
Maruyama, H. and Tamura, K. and Uramoto, N.
==========Detalhes do livro==========
XML and Java: Developing of Web ApplicationsAddison-Wesley1999MAAugust
==========Livro do autor==========
Bradley, N.
==========Detalhes do livro==========
The XML CompanionAddison-Wesley2000Great Britain2August
==========Fim da lista==========

# Exemplo (book_06.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output  method="text" indent="yes"/>

    <xsl:template match="text()"/>

    <xsl:template match="booklist">
      <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="book">
    _____ Livro _____
      <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="author">
    AUTOR: <xsl:value-of select="."/>
    </xsl:template>
</xsl:stylesheet>
```

# Resultado

_____ Livro_____

AUTOR: Box, D. and Skonnard, A. and Lam, J.
_____Livro_____

AUTOR: Maruyama, H. and Tamura, K. and Uramoto, N.
_____Livro_____

AUTOR: Bradley, N.

# Exercício 1

- Altere os arquivos de exemplo, para que a saída seja:

==========Livro==========

(imprimir o título do livro)

======Detalhes=====

author: (imprimir o nome do autor)

title:  (imprimir o título do livro)

# Processamento seletivo

- O atributo select do elemento apply-templates é utilizado para selecionar determinados filhos para serem processados e ignorar o restante

```
<xsl:template match="booklist">
    <xsl:apply-templates
    select="book[@id='MAR99']"/>
</xsl:template>
```

# Exemplo (book_7.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output  method="text" indent="yes"/>

    <xsl:template match="text()"/>

    <xsl:template match="booklist">
      <xsl:apply-templates select="book[@id='MAR99']"/>
    </xsl:template>

    <xsl:template match="book">
    _____ Livro _____
      <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="author">
    AUTOR: <xsl:value-of select="."/>
    </xsl:template>
</xsl:stylesheet>
```

# Resultado

_____Livro_____

AUTOR: Maruyama, H. and Tamura, K. and Uramoto, N.

# Gerando um novo documento XML

- É possível gerar novas *tags* XML na saída, produzindo um novo documento XML de saída a partir da entrada
- Tudo o que não possui o *namespace* de XSLT é copiado para a saída

```
<xsl:template match="book">
    <livro>
        <xsl:apply-templates/>
    </livro>
</xsl:template>
```

# Exemplo (09-sample.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output indent="yes"/>

    <xsl:template match="text()“/>

    <xsl:template match="booklist">
      <ListaLivros><xsl:apply-templates/></ListaLivros>
    </xsl:template>

    <xsl:template match="book">
      <Livro><xsl:apply-templates/></Livro>
    </xsl:template>

    <xsl:template match="author">
      <Autor><xsl:value-of select="."/></Autor>
    </xsl:template>

    <xsl:template match="title">
    <Titulo><xsl:value-of select="."/></Titulo>
    </xsl:template>
</xsl:stylesheet>
```

# Resultado

```
<?xml version="1.0" encoding="UTF-16"?>
<ListaLivros>
    <Livro>
     <Autor>Box, D. and Skonnard, A. and Lam, J.</Autor>
     <Titulo>Essential XML - Beyond Markup</Titulo>
    </Livro>
    <Livro>
     <Autor>Maruyama, H. and Tamura, K. and Uramoto, N.</Autor>
     <Titulo>XML and Java: Developing of Web Applications</Titulo>
    </Livro>
    <Livro>
     <Autor>Bradley, N.</Autor>
     <Titulo>The XML Companion</Titulo>
    </Livro>
</ListaLivros>
```

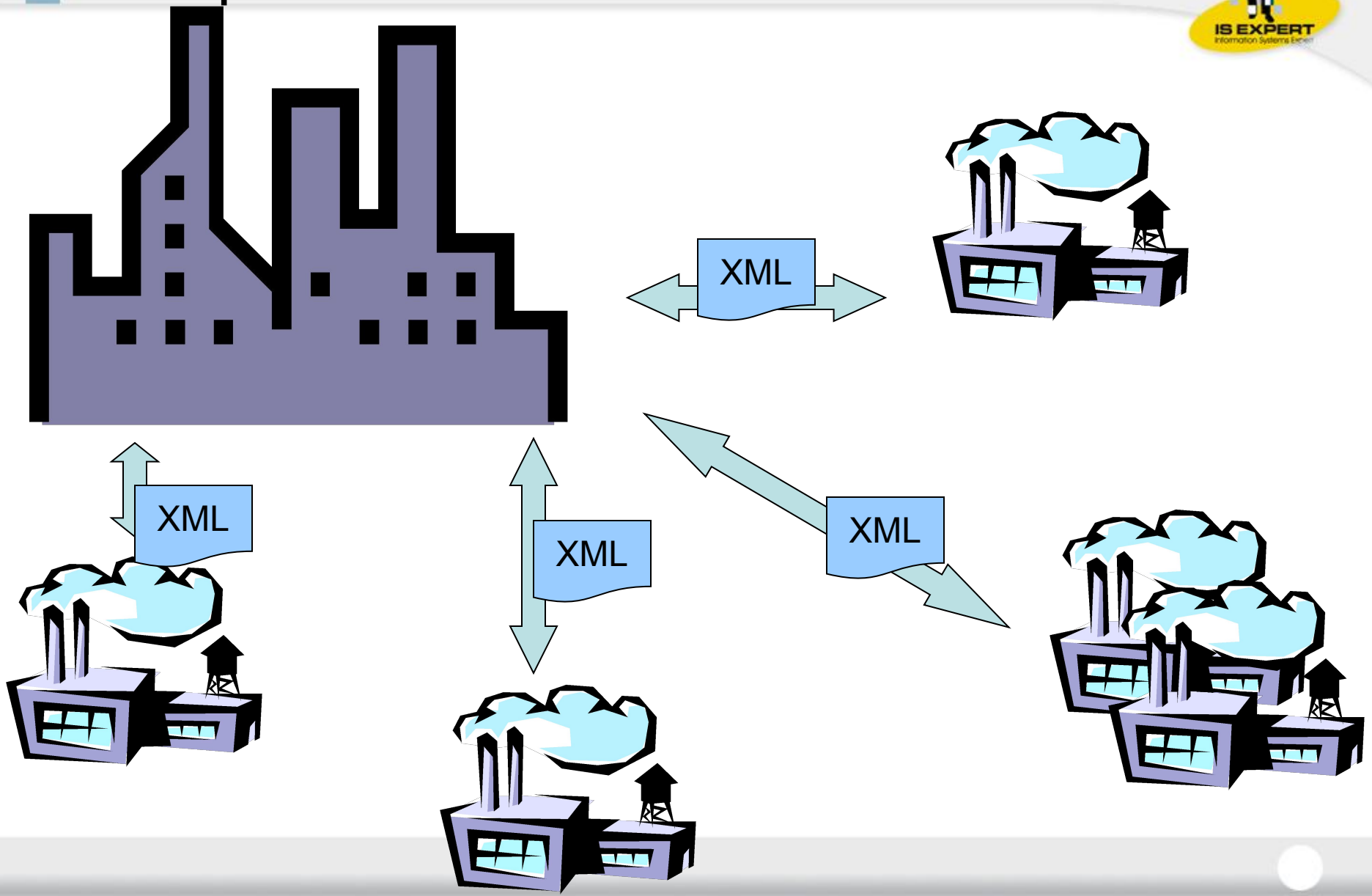
# Vejam mais em

- http://www.w3.org/TR/xslt#section-Applying-Template-Rules

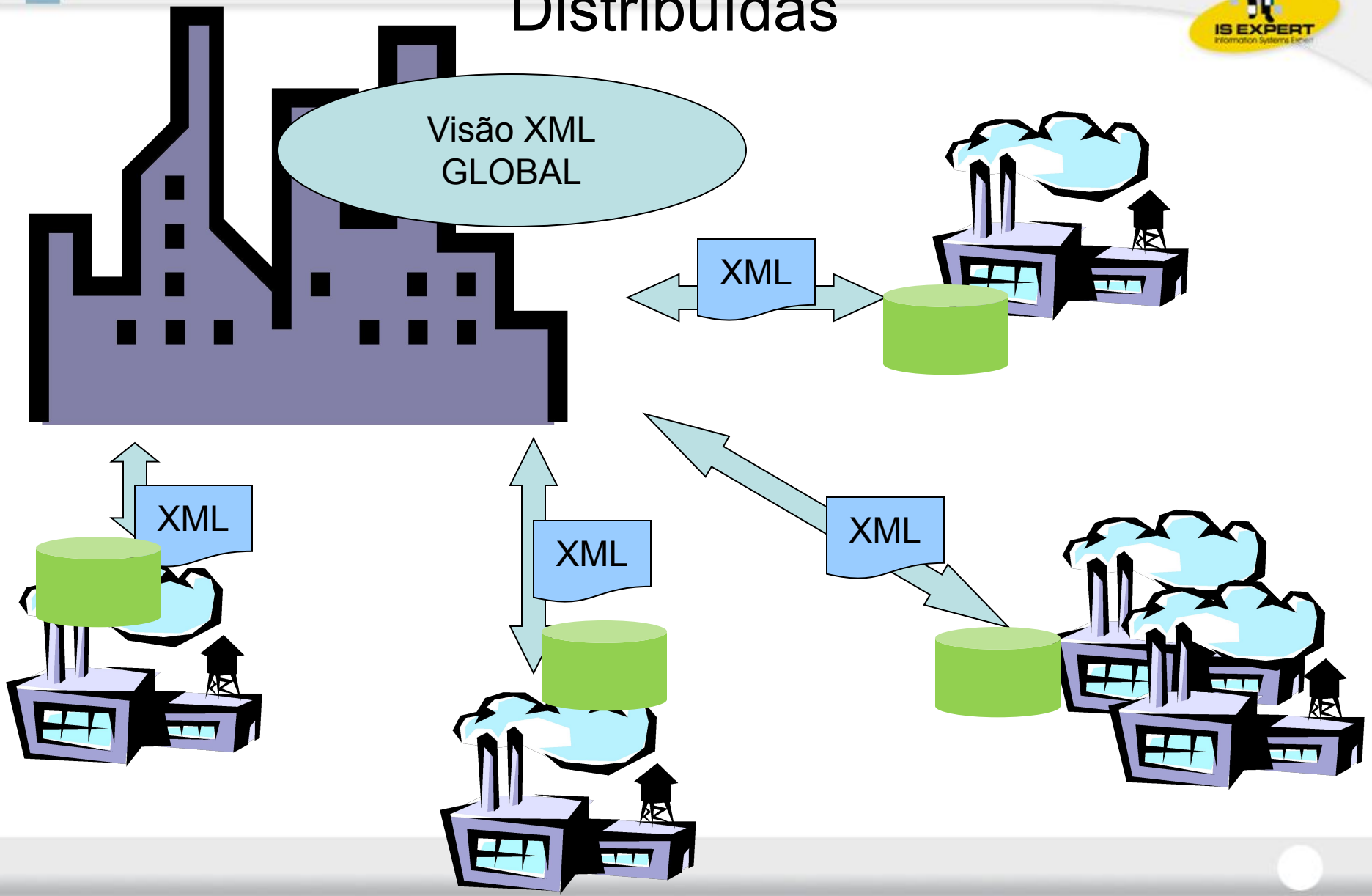
# Para que serve mesmo o Xpath?

E onde entra a integração de sistemas e interoperabilidade?

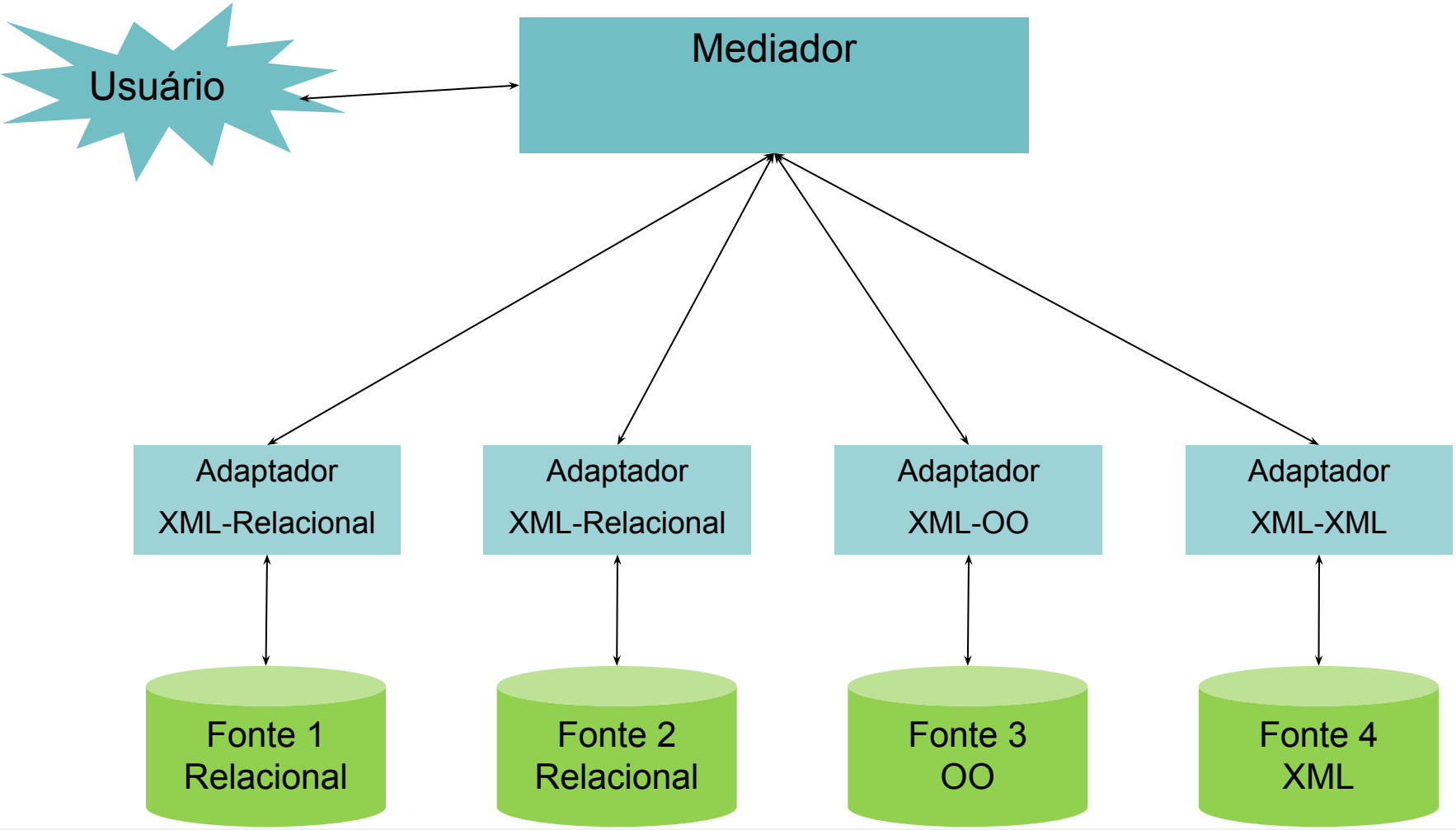
# Interoperabilidade – Ex. Comércio eletrônico



# Integração de Sistemas – Ex. Visões XML Distribuídas



# Arquitetura Clássica de Integração



# Web Services