# Engenharia de Ontologias (Ontology Engineering)

Universidade Federal de Uberlândia

Faculdade de Computação

Programa de Pós-Graduação em Ciência da Computação

Prof. Fabiano Azevedo Dorça

# XML Namespaces

- Conflitos de Nomes
  - ocorrerá quando for utilizado os **mesmos nomes** para descrever dois **tipos diferentes de elementos**.
- Namespaces XML são usados para fornecer **elementos e atributos com nome exclusivo** em um documento XML.
- São definidos em uma recomendação do W3C.

## XML Namespaces

- Uma instância XML pode conter nomes de elementos ou atributos de **mais de um vocabulário** XML.
- Se a cada vocabulário é dado um **namespace**, a **ambigüidade** entre elementos ou atributos pode ser resolvida.
- Exemplo:

# XML Namespaces

- Conflito de nomes em relação ao elemento table:

```
<root>
<table>
   <tr>
   <td>Maçãs</td>
   <td>Bananas</td>
   </tr>
</table>

<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
</root>
```

- Elemento <table> com diferentes conteúdo e definição.

## XML Namespaces

- Resolvendo o conflito de nomes usando um **prefixo**
- Os conflitos de nomes em XML podem ser facilmente evitados usando um ***prefixo de nome.***
- Este XML traz informações sobre uma tabela HTML e um móvel:

## XML Namespaces

```
<root>
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

## XML Namespaces

- No exemplo, **não haverá conflito** porque os dois elementos <table> têm nomes diferentes.
  - Namespaces XML - O atributo **xmlns**
- Ao usar **prefixos** em XML, um **namespace** para o prefixo **deve ser definido**.
- O **namespace** pode ser definido por um **atributo xmlns** na marca **inicial** de um elemento.
- A declaração de namespace tem a seguinte sintaxe.
  - xmlns:prefix = "URI".

# XML Namespaces

- Exemplo:

```
<root>

<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
     <h:td>Apples</h:td>
     <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="https://www.w3schools.com/furniture">
   <f:name>African Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>

</root>
```

## XML Namespaces

- No exemplo anterior:
  - O atributo **xmlns** no primeiro elemento <table> fornece um **namespace qualificado** ao prefixo h:
  - O atributo **xmlns** no segundo elemento <table> fornece um **namespace qualificado** ao prefixo f:
  - Quando um **namespace** é definido para um **elemento**, todos os elementos filho com o mesmo prefixo são associados com o **mesmo namespace**.
  - Namespaces também podem ser declarados no elemento raiz XML:

# XML Namespaces

```
<root  xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="https://www.w3schools.com/furniture">

<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

## XML Namespaces

- O **Uniform Resource Identifier *(URI)*** de namespace não é usado pelo analisador para procurar informações.
  - A finalidade de usar um URI é dar ao namespace **um nome único.**
- No entanto, as organizações costumam usar o namespace como um **ponteiro para uma página** da Web que contém **informações** sobre o namespace.
- Um **URI** é uma **seqüência de caracteres** que identifica um recurso da Internet.

## XML Namespaces

- O URI mais comum é o **Uniform Resource Locator** (URL) que identifica um endereço de domínio da Internet.
- Outro tipo, não tão comum de URI é o **Universal Resource Name** (URN).
- O uso de **namespaces** provê uma forma de referenciar elementos que possuem **nomes iguais**, porém, **semânticas diferentes**.
- Desta forma, pode-se utilizar dois elementos com nomes iguais, desempenhando **papeis diferentes** e pertencentes a **vocabulários diferentes**, em um **mesmo documento XML**.

## XML Namespaces

- A seguir um documento XML que utiliza **dois elementos** com o nome **title**,
  - um pertencente à linguagem XHTML
  - e o outro a uma linguagem definida pelo usuário.

- Exemplo:

```
<?xml version="1.0"?>

<xsl:stylesheet
xmlns="http://www.w3.org/1999/xhtml"
xmlns:book= "http://www.book.com"
xmlns:xsl= "http://www.w3.org/1999/XSL/Transform"
version="1.0">

<xsl:template match="/">

<html>
<title>Lista de Livros</title>
<body>
  <book:title>
     <xsl:value-of select ="text[2]/text()"/><br/>
     <xsl:value-of select ="text[1]/text()"/><br/>
   </book:title>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Elemento Root

Namespace default

Expressão Xpath

## XML Namespaces

- A declaração
  - xmlns=”http://www.w3.org/1999/xhtml”
  - define um “namespace” **padrão** para o documento **XHTML**.
- Desta forma, todo elemento no **corpo do documento XML** que **não possuir um prefixo** seguido de “:” será pertencente ao “namespace” xmlns=” http://www.w3.org/1999/xhtml”.

## XML Namespaces

- Já as declarações
  - xmlns:book="http://www.book.com"
  - xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
- indicam que...
  - todo elemento precedido do prefixo book pertence ao *namespace* "http://www.book.com"
  - e todo elemento precedido do prefixo xsl pertence ao *namespace* xmlns:xsl="http://www.w3.org/1999/XSL/Transform".

## XML Namespaces

- No exemplo o **parser XML** terá condição de determinar qual elemento pertence a qual vocabulário e **interpretar corretamente.**
- O parser terá condições de utilizar o **vocabulário** exato em cada trecho do código, verificando assim, a **corretude** de cada elemento, dado a linguagem em que ele foi definido.
- O uso de “namespace”, também permitirá que um **processador XSLT** (XML Stylesheet Language Transformation) saiba distinguir quais elementos devem ser considerados como pertencentes à folha de estilo.

## XML Namespaces

- Sintaxe para Namespaces
    - Para associar um elemento a um determinado “namespace” é necessário declarar um “namespace” anteriormente.
- Uma declaração de um “namespace” possui um nome associado a uma URI (Uniforme Resource Identifier), que o identifica como sendo único.
- Por exemplo, o “namespace” xsl no exemplo anterior esta associado a URI “http://www.w3.org/1999/XSL/Transform”.

## XML Namespaces

- O **prefixo associado ao “namespace”, mais o nome do elemento**, formam juntos, um **identificador** único global conhecido como **nome qualificado**.

- Isto significa que o **elemento pertence a um determinado vocabulário** associado ao “namespace” declarado e deve ser considerado como tal.

# XSLT

- XSL Transformations, ou XSLT (eXtensible Stylesheet Language for Transformation

  - é uma linguagem de marcação XML
  - usada para criar documentos XSL
  - definem a apresentação dos documentos XML nos browsers e outros aplicativos que a suportem.

- XSL = Folhas de Estilo para XML

- Portanto, o XSL descreve como os elementos XML devem ser exibidos.

# XSLT

- XSLT é usado para transformar um documento XML em outro documento XML,
  - ou outro tipo de documento que é reconhecido por um navegador, como HTML e XHTML.
  - Normalmente, o XSLT faz isso transformando cada elemento XML em um elemento (X) HTML.
- Com XSLT é possível **adicionar/remover elementos e atributos** para o arquivo de saída.
- É possível **reorganizar e classificar elementos**, realizar testes e tomar decisões sobre quais elementos ocultar e exibir.

# XSLT

- Uma maneira comum de descrever o processo de transformação é dizer que
  - o XSLT transforma uma **árvore-fonte XML**
  - em uma árvore de **resultados XML**.

# XSLT

- O **elemento raiz** que declara o documento como uma folha de estilo XSL é <xsl:stylesheet> ou <xsl:transform>.

- <xsl:stylesheet> e <xsl:transform> são completamente sinônimos e podem ser usados indistintamente.

- A maneira correta de declarar uma folha de estilo XSL de acordo com a Recomendação W3C XSLT é:

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

ou:

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

# XSLT

- Exemplo: Transformar o seguinte documento XML ("cdcatalog.xml") em XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
      <title>Empire Burlesque</title>
      <artist>Bob Dylan</artist>
      <country>USA</country>
      <company>Columbia</company>
      <price>10.90</price>
      <year>1985</year>
  </cd>
...
</catalog>
```

## XSL Style Sheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns="http://www.w3.org/1999/xhtml">
<xsl:template match="/">
 <html>
 <body>
 <h2>My CD Collection</h2>
 <table border="1">
   <tr bgcolor="#9acd32">
     <th>Title</th>
     <th>Artist</th>
   </tr>
   <xsl:for-each select="catalog/cd">
   <tr>
     <td><xsl:value-of select="title"/></td>
     <td><xsl:value-of select="artist"/></td>
   </tr>
   </xsl:for-each>
 </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

O valor do atributo **select** é uma expressão **XPath**. Uma expressão XPath funciona como navegar em um sistema de arquivos; Onde uma barra invertida (/) seleciona subdiretórios.

## XSLT

- Vincular a folha de estilos XSL ao documento XML

- Adicione a referência de folha de estilo XSL ao documento XML ("cdcatalog.xml"):

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
....
</catalog>
```

# XSLT

- Observações:
  - Uma vez que uma folha de estilo XSL é um documento XML, ela sempre começa com a declaração XML:

    <?xml version = "1.0" *encoding = "UTF-8"*?>

  - O elemento seguinte, <xsl:stylesheet>, define que este documento é um **documento de folha de estilo XSLT** (juntamente com o número de versão e atributos de namespaces XSLT).

# XSLT

- O elemento <xsl:template> define um modelo.
- O atributo match = "/" associa o modelo à raiz do documento fonte XML (é uma expressão XPath).
- O conteúdo dentro do elemento <xsl:template> define algum HTML para escrever na saída.

# XPath

- Linguagem de **Consulta** para XML
  - Cada vez **mais dados** disponíveis neste formato
  - Precisam ser consultados
  - Imita a **sintaxe** de navegação da **URI**
  - Delimitado pelas expressões de **caminhos**
  - Resultado **preserva ordem**, hierarquia e identidade dos objetos

# XPath

XPath é um elemento importante em XSLT.

- XPath pode ser usado para navegar através de elementos e atributos em um documento XML.

– Expressões de caminho XPath

- O XPath usa **expressões de caminho** para selecionar nós ou conjuntos de nós em um documento XML.

- Essas expressões de caminho se parecem muito com as expressões de caminho de sistemas de arquivos.

# XPath

- Funções padrão XPath
  - XPath inclui mais de 200 funções internas.
  - Existem funções para manipulação
    - seqüência de caracteres,
    - valores numéricos,
    - booleanos,
    - comparação de data e hora,
    - manipulação de nó, etc.

# XPath

- Xpath funciona basicamente como uma linguagem para **navegação** em **diretórios**, em que os diretórios são os **nós do documento XML**.

# XPath

- Uma expressão XPath descreve a **localização de um elemento ou atributo** em um documento XML.
- Começando pelo **elemento raiz**, podemos selecionar **qualquer elemento** no documento criando uma cadeia de elementos filhos.
- Cada elemento é separado por uma barra "/".

# XPath

- Considere o seguinte documento XML

```
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
```

# XPath

- XPath usa expressões de caminho para selecionar nós em um documento XML. O nó é selecionado seguindo um caminho ou etapas.
  - *Nodename* - Seleciona todos os nós com o nome "*nodename*"
  - / Seleciona a partir do nó raiz
  - // Seleciona os nós no documento a partir do nó atual, não importa onde eles estejam
  - . Seleciona o nó atual
  - .. Seleciona o pai do nó atual
  - @ Seleciona atributos

## XPath

- Exemplos:
  - bookstore - Seleciona o nó com o nome "bookstore"
  - /bookstore - Seleciona o elemento raiz
    - Obs. / representa caminho absoluto para um elemento
  - bookstore/book - Seleciona todos os elementos livros que são filhos do elemento livraria
  - //book - Seleciona todos os elementos livro, **independentemente do local** no documento
  - bookstore//book - Seleciona todos os elementos livro que são descendentes do elemento livraria, não importando onde estejam abaixo do elemento da livraria

# XPath

- Predicados
  - Os predicados são usados para localizar um nó específico ou um nó que contém um valor específico.
  - Os predicados são sempre especificados entre colchetes.
- Exemplos:
  - /bookstore/book[1]  - Seleciona o primeiro elemento livro em livraria.
  - /bookstore/book[last()]  - Seleciona o último livro que é filho do elemento livraria

## XPath

- /bookstore/book[last()-1]  - Seleciona o penúltimo livro
- /bookstore/book[position()<3]  - Seleciona os dois primeiros livros
- //title[@lang]  - Seleciona todos os elementos de título que possuem um atributo chamado *lang*
- //title[@lang='en']  - Seleciona todos os elementos de título que possuem um atributo "lang" com valor "en"
- /bookstore/book[price>35.00]  -  Seleciona todos os livros que têm um elemento preço com um valor maior que 35.00

# XPath

- /bookstore/book[price>35.00]/title - Seleciona todos os títulos dos livros que têm um elemento “preço” com um valor maior que 35,00

- Seleção de vários caminhos

  - Usando o Operador “|” em uma expressão XPath você pode selecionar vários caminhos.

- Exemplos:

  - //book/title | //book/price - Seleciona todos os elementos título E elementos preço de todos os elementos do livro

  - //title | //price - Seleciona todos os elementos preço E título no documento

# XPath

- Package javax.xml.xpath
  - Este pacote fornece uma **API** para a avaliação de expressões **XPath** em aplicações Java.
  - O exemplo a seguir demonstra o uso da API XPath para selecionar um ou mais nós de um documento XML:

```
XPath xpath = XPathFactory.newInstance().newXPath();
String expression = "/widgets/widget";
InputSource inputSource = new InputSource("widgets.xml");
NodeList nodes = (NodeList) xpath.evaluate(expression,
inputSource, XPathConstants.NODESET);
```

# XPath

- Fim.
- Prática 02.