XSL TRANSFORMATIONS (XSLT) VERSION 1.0

Engenharia de Dados e Conhecimento

O que é?

- *Extensible Stylesheet Language Transformations* (XSLT) é uma linguagem de programação funcional usada para especificar como um documento XML de entrada é convertido num outro documento, de texto ou não.

Processo de transformação

- Um processador XSLT lê um documento XML e um documento XSLT (que é também um documento XML), ambos de entrada, e produz uma árvore de resultado como saída.
- Os documentos podem ser transformados usando uma aplicação *standalone* ou podem ser transformados numa aplicação maior que comunica com o processador XSLT através de uma API.

Namespace XSLT

- Todos os elementos XSLT padrão estão no *namespace* http://www.w3.org/1999/XSL/Transform.

```
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <!-- XSLT top-level elements go here -->
</xsl:stylesheet>
```

O que é XSLT?

- XSLT é uma linguagem para transformar documentos XML em outros documentos.
  - Através de XSLT é possível adicionar/remover elementos/atributos de e para um ficheiro.
  - É ainda possível reorganizar os elementos, efetuar testes e ainda decidir que elementos mostrar ou esconder.
- XSLT utiliza XPath.
  - XPath é utilizado para encontrar e selecionar elementos e/ou atributos num documento XML.

XSLT Elements

- XSLT define 37 elementos, que se dividem em três categorias que se sobrepõem:
  - 2 elementos raiz
  - 12 elementos de nível superior (que aparecem imediatamente a seguir ao elemento raiz)
  - 23 elementos de instrução.

XSLT – Elementos raíz

- <xsl:stylesheet>
- <xsl:transform>

- Tem de ser usado um destes elementos como elemento raíz do document XSLT.

```
<xsl:stylesheet
  id = id
  extension-element-prefixes = tokens
  exclude-result-prefixes = tokens
  version = number>
  <!-- Content: (xsl:import*, top-level-elements) -->
</xsl:stylesheet>
```

```
<xsl:transform
  id = id
  extension-element-prefixes = tokens
  exclude-result-prefixes = tokens
  version = number>
  <!-- Content: (xsl:import*, top-level-elements) -->
</xsl:transform>
```

XSLT - elementos de nível superior

1. xsl:attribute-set
2. xsl:decimal-format
3. xsl:import
4. xsl:include
5. xsl:key
6. xsl:namespace-alias
7. xsl:output
8. xsl:param
9. xsl:preserve-space
10. xsl:strip-space
11. xsl:template
12. xsl:variable

Focar-nos-emos mais nos assinalados a vermelho!

XSLT <xsl:import> Element

- Um documento XSLT pode importar outro documento XSLT usando o elemento <xsl:import>.
- Importar um documento é o mesmo que incluí-lo no documento atual;
  - As definições e regras do modelo no documento têm precedência sobre as regras do modelo e definições no documento importado.
- O elemento <xsl:import> tem um atributo “href” cujo valor é uma referência URI para identificar o documento a importar.

XSLT <xsl:import> Element

- Exemplo:

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="article.xsl"/>
  <xsl:import href="bigfont.xsl"/>
  <xsl:attribute-set name="note-style">
    <xsl:attribute name="font-style">italic</xsl:attribute>
  </xsl:attribute-set>
</xsl:stylesheet>
```

- Regras de precedência. Exemplo:
  - A *stylesheet* A importa as *stylesheets* B e C nessa ordem;
  - A *stylesheet* B, por sua vez, importa a *stylesheet* D;
  - A *stylesheet* C importa a *stylesheet* E.
  - **A precedência será, da maior para a menor: A, C, E, B, D.**

XSLT <xsl:output> Element

- O processador XSLT gera como resultado um conjunto de carateres, que se organizam em diferentes formatos, conforme o especificado pelo elemento <xsl: output>.

```
<!-- Category: top-level-element -->
<xsl:output
  method = "xml" | "html" | "text" | qname-but-not-ncname
  version = nmtoken
  encoding = string
  omit-xml-declaration = "yes" | "no"
  standalone = "yes" | "no"
  doctype-public = string
  doctype-system = string
  cdata-section-elements = qnames
  indent = "yes" | "no"
  media-type = string />
```

XSLT <xsl:output> Element

- Encoding
  - O atributo encoding especifica a codificação preferida para usar nos resultados.
  - Os processadores XSLT são obrigados a respeitar os valores de UTF-8 e UTF-16. Para outros valores, se o processador XSLT não oferecer suporte à codificação especificada, pode assinalar um erro ou usar UTF-8 ou UTF-16 em seu lugar.
- Indent
  - Se o atributo de indentação tomar o valor “yes”, então o documento XML de saída, será preenchido com espaços em branco na esquerda dos elementos da árvore de resultados;
  - Se o atributo de indentação tiver o valor “no”, não serão adicionados quaisquer espaços em branco ou mudanças de linha no resultado.
    - O valor padrão é “no” porque, normalmente, estes dados são utilizados entre máquinas e a representação visual (com os espaços) não acrescentam semântica e fazem crescer o tamanho da resposta

XSLT <xsl:template> Element

- O elemento <xsl:template> é utilizado na construção de modelos de representação e o <xsl:apply-templates> na sua aplicação.

```
<!-- Category: instruction -->
<xsl:apply-templates
     select = node-set-expression
     mode = qname>
  <!-- Content: (xsl:sort | xsl:with-param)* -->
</xsl:apply-templates>
```

- Na ausência do atributo **select**, o template aplica-se a todos os filhos do nó atual, incluindo os nós de texto.
- No uso do atributo **select**, serão filtrados os nós a processar.
  - O resultado da expressão deve ser avaliado como um conjunto de nós.
  - O conjunto de nós selecionados serão processados na ordem do documento, a menos que uma especificação de ordenação (sort) esteja presente.

XSLT <xsl:template> Element

- Múltiplos elementos <xsl:apply-templates> podem ser usados num modelo único para fazer reordenação simples.
  - O exemplo seguinte cria duas tabelas HTML. A primeira tabela é preenchida com vendas no mercado interno, enquanto a segunda tabela é preenchida com as vendas externas.

```
<xsl:template match="product">
  <table>
    <xsl:apply-templates select="sales/domestic"/>
  </table>
  <table>
    <xsl:apply-templates select="sales/foreign"/>
  </table>
</xsl:template>
```

Variáveis e Parâmetros

- Uma variável é um nome que pode ser vinculado a um valor. O valor para o qual essa variável está vinculada pode ser de qualquer dos tipos retornados por expressões (string, date, …).
- Há dois elementos que podem ser usados para variáveis: **xsl:variable** e **xsl:param**.
  - Um elemento xsl:variable suporta apenas um valor padrão
  - Quando um *stylesheet* possui um elemento do tipo xsl:param, podem ser passados parâmetros que são usados em lugar do valor padrão.
- Tanto o xsl:variable como xsl:param possuem um atributo obrigatório, **name**, que especifica o nome da entidade.

<xsl:variable>

- Sintaxe:

```
<!-- Category: top-level-element -->
<!-- Category: instruction -->
<xsl:variable
  name = qname
  select = expression>
  <!-- Content: template -->
</xsl:variable>
```

- Exemplos:

```
<xsl:variable name="n">2</xsl:variable>
...
<xsl:value-of select="item[position()=$n]"/>
```

```
<xsl:variable name="n" select="2"/>
...
<xsl:value-of select="item[$n]"/>
```

Estes exemplos devolvem o terceiro elemento de uma lista

<xsl:param>

- Sintaxe:

```
<!-- Category: top-level-element -->
<xsl:param
 name = qname
 select = expression>
 <!-- Content: template -->
</xsl:param>
```

- Exemplo:

```
<xsl:param name="lang" select="en"/>
<xsl:variable name="messages" select="document(concat('resources/', $lang, '.xml'))/messages"/>

<xsl:template name="localized-message">
 <xsl:param name="name"/>
 <xsl:message>
  <xsl:value-of select="$messages/message[@name=$name]"/>
 </xsl:message>
</xsl:template>

<xsl:template name="problem">
 <xsl:call-template name="localized-message"/>
  <xsl:with-param name="name">problem</xsl:with-param>
 </xsl:call-template>
</xsl:template>
```

Elementos de instrução

1. xsl:apply-imports
2. xsl:apply-templates
3. xsl:attribute
4. xsl:call-template
5. xsl:choose
6. xsl:comment
7. xsl:copy
8. xsl:copy-of
9. xsl:element
10. xsl:fallback
11. xsl:for-each
12. xsl:if
13. xsl:message
14. xsl:number
15. xsl:otherwise
16. xsl:param
17. xsl:processing-instruction
18. xsl:sort
19. xsl:text
20. xsl:value-of
21. xsl:variable
22. xsl:with-param
23. xsl:when

XSLT <xsl:value-of> Element

- O elemento <xsl:value-of> é usado para recolher o valor de um elemento ou atributo em xml.
- O seu atributo “select” é obrigatório e representa uma expressão que é avaliada.
- O resultado é convertido para uma string.
  - Se o valor recolhido for vazio, nenhum resultado será criado.
  - O resultado criado será colocado junto ao texto já existente.
- Sintaxe:

```
<!-- Category: instruction -->
<xsl:value-of
  select = string-expression
  disable-output-escaping = "yes" | "no" />
```

XSLT <xsl:value-of> Element

- Exemplos:

```
<xsl:template match="person">
  <p>
   <xsl:value-of select="@given-name"/>
   <xsl:text> </xsl:text>
   <xsl:value-of select="@family-name"/>
  </p>
</xsl:template>
```

Este exemplo cria um parágrafo HTML (<p>) a partir de um elemento XML "person" com os **atributos** "given-name" e "family-name".
O parágrafo conterá o valor do atributo "given-name" seguido por um espaço e pelo valor do atributo "family-name".

```
<xsl:template match="person">
  <p>
   <xsl:value-of select="given-name"/>
   <xsl:text> </xsl:text>
   <xsl:value-of select="family-name"/>
  </p>
</xsl:template>
```

Este exemplo cria um parágrafo HTML (<p>) a partir de um elemento XML "person" com os **elementos** "given-name" e "family-name".
O parágrafo conterá o valor do elemento "given-name" seguido por um espaço e pelo valor do elemento "family-name".

XSLT <xsl:for-each> Element

- A instrução <xsl:for-each> contém um modelo, que é instanciado para cada nó selecionado pela expressão especificada pelo atributo de select – que é obrigatório.
- O resultado da expressão deve ser avaliado como um conjunto de nós.
- Sintaxe:

```
<!-- Category: instruction -->
<xsl:for-each
 select = node-set-expression>
 <!-- Content: (xsl:sort*, template) -->
</xsl:for-each>
```

XSLT <xsl:for-each>

- Exemplo:

```
<customers>
  <customer>
    <name>…</name>
    <order>…</order>
    <order>…</order>
  </customer>
  <customer>
    <name>…</name>
    <order>…</order>
    <order>…</order>
  </customer>
</customers>
```

```
<xsl:template match="/">
  <html>
    <head>
      <title>Customers</title>
    </head>
    <body>
      <table>
        <tbody>
          <xsl:for-each select="customers/customer">
            <tr>
              <th>
                <xsl:value-of select="name"/>
              </th>
              <xsl:for-each select="order">
                <td>
                  <xsl:value-of select=“.”/>
                </td>
              </xsl:for-each>
            </tr>
          </xsl:for-each>
        </tbody>
      </table>
    </body>
  </html>
</xsl:template>
```

Transformação de um documento XML num documento html -> Percorre todos os clientes e cria uma linha nova para cada um. Cria a primeira coluna para o nome e colunas subsequentes para cada encomenda.

XSLT <xsl:sort> Element

- A ordenação é especificada pela adição de um ou vários elementos <xsl:sort> como filho(s) de um elemento <xsl:apply-templates> ou <xsl:for-each>.
  - O primeiro <xsl:sort> especifica a chave principal de ordenação; o segundo <xsl:sort> especifica a chave de classificação secundária e assim por diante.
  - Quando um <xsl:apply-templates> ou <xsl:for-each> têm um ou mais <xsl:sort>, em vez de processar os nós selecionados na ordem do documento, ele ordena primeiramente os nós de acordo com as chaves de classificação especificadas e, só depois, os processa de acordo com o resultado da ordenação.
  - Quando usado dentro de um elemento <xsl:for-each>, o elemento <xsl:sort> deve ser o primeiro elemento dentro da estrutura.

XSLT <xsl:sort> Element

- O elemento <xsl:sort> tem um atributo de “select” cujo valor é uma expressão.
  - O atributo order indica se a ordenação é crescente ou decrescente
  - O atributo lang especifica em que língua a ordenação deverá ser feita. Se não for especificada, toma o valor do ambiente em que está a ser executada a ordenação.
  - O data-type especifica o tipo de dados existente nas strings. O valor padrão é “text”.
- Sintaxe:

```
<xsl:sort
  select = string-expression
  lang = { nmtoken }
  data-type = { "text" | "number" | qname-but-not-ncname }
  order = { "ascending" | "descending" }
  case-order = { "upper-first" | "lower-first" } />
```

XSLT <xsl:sort> Element

- Exemplos:

```
<employees>
  <employee>
    <name>
      <given>James</given>
      <family>Clark</family>
    </name>
    ...
  </employee>
</employees>
```

```
<xsl:template match="employees">
  <ul>
    <xsl:apply-templates select="employee">
      <xsl:sort select="name/family"/>
      <xsl:sort select="name/given"/>
    </xsl:apply-templates>
  </ul>
</xsl:template>

<xsl:template match="employee">
  <li>
    <xsl:value-of select="name/given"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="name/family"/>
  </li>
</xsl:template>
```

Exemplo de uma transformação de um documento XML num documento html ...
Percorre os clientes todos, ordena-os por sobrenome e depois por nome e cria uma lista
<nome>espaço<sobrenome>

XSLT <xsl:if> Element

- O elemento <xsl:if> é usado para colocar uma condição única.
- Tem um atributo “test” que especifica uma expressão. A expressão é avaliada e o objeto resultante é convertido para um valor booleano.
  - Se o resultado for verdadeiro, então o modelo do conteúdo é instanciado, caso contrário, não se cria nada.
- Sintaxe:

```
<!-- Category: instruction -->
<xsl:if
 test = boolean-expression>
 <!-- Content: template -->
</xsl:if>
```

XSLT <xsl:if> Element

- Exemplos:

```
<xsl:template match="namelist/name">
  <xsl:apply-templates/>
  <xsl:if test="not(position()=last())">, </xsl:if>
</xsl:template>
```

Este exemplo, pega numa lista de nomes (namelist/name) e coloca-a numa linha com os nomes separados por uma vírgula – à exceção do último.

```
<xsl:template match="item">
  <tr>
    <xsl:if test="position() mod 2 = 0">
      <xsl:attribute name="bgcolor">yellow</xsl:attribute>
    </xsl:if>
    <xsl:apply-templates/>
  </tr>
</xsl:template>
```

Neste exemplo, todas as linhas pares da tabela são formatadas com o fundo a amarelo

XSLT <xsl:choose> Element

- O elemento <xsl:choose> é utilizado em conjunto com os elementos <xsl:when> e <xsl:otherwise> para definir múltiplas condições de teste.
- Cada elemento <xsl:when> tem um único atributo, “test”, que especifica uma expressão.

- Quando um elemento <xsl:choose> é processado, cada um dos elementos <xsl:when> é testado por sua vez, avaliando a expressão e convertendo o objeto resultante num valor booleano.
  - Só o primeiro <xsl:when> cujo valor “test” for verdadeiro é instanciado.
  - Se nenhum <xsl:when> tomar o valor verdadeiro, o conteúdo do elemento <xsl: otherwise> é instanciado.
  - Se nenhum elemento <xsl:when> tomar o valor verdadeiro, e não existir um elemento <xsl:otherwise> nada será criado.

XSLT <xsl:choose> Element

- Sintaxe:

```
<!-- Category: instruction -->
<xsl:choose>
  <!-- Content: (xsl:when+, xsl:otherwise?) -->
</xsl:choose>

<xsl:when
  test = boolean-expression>
  <!-- Content: template -->
</xsl:when>

<xsl:otherwise>
  <!-- Content: template -->
</xsl:otherwise>
```

XSLT <xsl:choose> Element

- Exemplo:

1. Xxx
   a) Yyy
   b) Yyy
      i. Zzz
      ii. zzz
2. Xxxx

Este exemplo enumera os itens de uma lista ordenada com algarismos arábicos, letras ou algarismos romanos, dependendo da profundidade das listas.

```
<xsl:template match="orderedlist/listitem">
 <fo:list-item indent-start='2pi'>
  <fo:list-item-label>
   <xsl:variable name="level" select="count(ancestor::orderedlist) mod 3"/>
   <xsl:choose>
    <xsl:when test='$level=1'>
     <xsl:number format="a"/>
    </xsl:when>
    <xsl:when test='$level=2'>
     <xsl:number format="i"/>
    </xsl:when>
    <xsl:otherwise>
     <xsl:number format="1"/>
    </xsl:otherwise>
   </xsl:choose>
   <xsl:text>. </xsl:text>
  </fo:list-item-label>
  <fo:list-item-body>
   <xsl:apply-templates/>
  </fo:list-item-body>
 </fo:list-item>
</xsl:template>
```

XSLT <xsl:apply-templates> Element

- O elemento <xsl:apply-templates> aplica um template ao elemento corrente ou aos *child nodes* do elemento corrente.
- Se o elemento <xsl:apply-templates> tiver um atributo de *select*, serão processados os *child element* cujo valor coincida com o do atributo
  - O atributo select pode também ser utilizado para seleccionar a ordem em que os child nodes serão processados.

XSLT <xsl:apply-templates> Element

- Exemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?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>
```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose

cdcatalog.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
    <tr bgcolor="#9acd32">
      <th align="left">Title</th>
      <th align="left">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>
```

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose

Exemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>
<xsl:template match="cd">
        <p>
        <xsl:apply-templates select="title"/>
        <xsl:apply-templates select="artist"/>
        </p>
</xsl:template>
<xsl:template match="title">
Title: <span style="color:#ff0000"><xsl:value-of select="."/></span><br />
</xsl:template>
<xsl:template match="artist">
Artist: <span style="color:#00ff00"><xsl:value-of select="."/></span><br />
</xsl:template>
</xsl:stylesheet>
```

My CD Collection

Title: Empire Burlesque
Artist: Bob Dylan

Title: Hide your heart
Artist: Bonnie Tyler

Title: Greatest Hits
Artist: Dolly Parton

Title: Still got the blues
Artist: Gary Moore

Title: Eros
Artist: Eros Ramazzotti

Title: One night only
Artist: Bee Gees

Title: Sylvias Mother
Artist: Dr.Hook

Title: Maggie May
Artist: Rod Stewart

Title: Romanza
Artist: Andrea Bocelli