SCC0265 – Sistemas Interativos Web

A Linguagem XML
& estilos & apresentação

Renata Pontin M. Fortes

(renata@icmc.usp.br)

PAE: Willian Watanabe (watinha@gmail.com)

Instituto de Ciências Matemáticas e de Computação

ICMC-USP S.Carlos, 2010

Roteiro

- XML com CSS
- Noções de vínculos de dados
- Vínculos com tabelas
- XSL

Publicando Documentos XML

XML com CSS?

- As CSSs (*Cascating Style Sheets*) foram desenvolvidas para trabalhar com HTML, mas podem ser usadas com XML.
- Usando CSS pode-se especificar como os elementos (tags) XML serão formatados pelo browser.

- Considere o arquivo Livro1.xml a seguir e seu correspondente Livro1.css

XML com CSS

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/css" href="Livro1.css"?>
<DOCUMENTO>
  <TITULO>The Meditations</TITULO>
  <AUTOR>by Marcus Aurelius</AUTOR>
  <SECAO>Book One</SECAO>
  <P> Primeiro paragrafo do livro </P>
  <P> Segundo paragrafo do livro </P>
  <P> Terceiro paragrafo do livro </P>
</DOCUMENTO>
```

XML com CSS

```
TITULO {
    display: block;
    font-size: 24pt;
    font-weigth: bold;
    text-align: center;
    text-decoration:underline
}
AUTOR {
    display: block;
    font-size: 18pt;
    font-weigth: bold;
    text-align: center
}
```

XML com CSS

```
SECAO {
    display: block;
    font-size: 16pt;
    font-weigth: bold;
    text-align: center;
    font-style:italic
}
P {
    display: block;
    margin-top: 10
}
```

Vínculos entre XML e HTML

- O vínculo de dados serve para “ligar” um .xml a um .html
- Torna-se nítida agora a separação entre o conteúdo e a formatação da página.

Vínculos entre XML e HTML

- Considere os arquivos clientes.xml e clientes.html:

```
<?xml version="1.0"?>
<clientes>
      <cliente>
            <nome> Maria das Dores </nome>
            <RG> 1.256.789 </RG>
            <data_compra> 10/05/2003 </data_compra>
      </cliente>
      <cliente>
            <nome> José da Silva </nome>
            <RG> 9.547.123 </RG>
            <data_compra> 12/12/2003 </data_compra>
      </cliente>
</clientes>
```

Vínculos entre XML e HTML

```
ml>
<head>
   <title>Vínculos entre XML e HTML</title>
</head>
<body>
   <xml id="cli" src="clientes.xml"></xml>
   <p> <H2>Cliente: <span datasrc="#cli" datafld="nome"></span><
   <br/>
   <p><b>RG do cliente: <span datasrc="#cli" datafld="RG"></span></b><
  <p><b>Data da compra: <span datasrc="#cli" datafld="data_compra"></s
        </b></p>
</body>
tml>
```

Vínculos entre XML e HTML

- O vínculo é feito pela linha:

```
<xml id="cli" src="clientes.xml"></xml>
```

- Também usa-se a tag **span** com os atributos **datasrc** e **datafld**:

```
<span datasrc="#cli" datafld="nome">
```

- Para ver o resto das informações do XML, deve-se usar botões ou tabelas.

Vínculos com Botões

- *Para percorrer os outros registros do XML:*

```
.........
 </p>
      <br>
      <button onclick="cli.recordset.moveFirst()">
              &lt;&lt; </button>
      <button onclick="if (!cli.recordset.BOF)
              cli.recordset.movePrevious()">
              &lt;  </button>
     <button onclick="if (!cli.recordset.EOF)
              cli.recordset.moveNext()">
              &gt;  </button>
     <button onclick="cli.recordset.moveLast()">
              &gt;&gt;  </button>
    </body>
</html>
```

Vínculos com Tabelas

```
<html>
   <head>
      <title>Vínculos entre XML e HTML com Tabela</title>
   </head>
   <body>
      <xml id="cli" src="cliente.xml"></xml>
      <table datasrc="#cli" border=1>
   <thead>
   <th>Nome</th>
   <th>RG</th>
   <th>Data da Compra</th>
   </thead>
   <tr>
   <td><span datafld="nome"></span></td>
   <td><span datafld="RG"></span></td>
   <td><span datafld="data_compra"></span></td>
   </tr>
   </table>
    </body>
</html>
```

Vínculos com Tabelas

- A linha

```
<xml id="cli" src="cliente.xml"></xml>
```

permite conectar a página html ao documento cliente.xml.

- A linha

```
<table datasrc="#cli" border=1>
```

permite que uma tabela seja criada com o número de linhas necessário para todos os dados.

Vínculos com Tabelas

- Na linha

```
<td><span datafld=“nome"></span></td>
```

a tag span junto com o atributo datafld permite referenciar cada campo do .xml a uma célula da tabela.

- OBS: Se o .xml fosse composto de 1.000 filmes, o comportamento seria o mesmo.

Vínculos com Atributos

- Pode-se exibir o conteúdo dos atributos da mesma maneira.
- Considere o XML e o HTML a seguir:

```
<?xml version="1.0"?>
<filmes>
      <item tipo="Fita" codigo="D123">
            <tipof>Drama</tipof>
            <nome> ... E o Vento Levou </nome>
      </item>
      <item tipo="DVD" codigo="A23">
            <tipof>Aventura</tipof>
            <nome> As Minas do Rei Salomão </nome>
      </item>
</filmes>
```

Vínculos com Atributos

```
<html>
   <head>
      <title>Trabalho XML</title>
   </head>
   <body>
      <xml id="flm" src="filmes.xml"></xml>
      <table datasrc="#flm" border=1>
          <thead>
           <th>Tipo Item</th>
           <th>Tipo</th>
           <th>Codigo</th>
           <th>Filme</th>
          </thead>
```

Vínculos com Atributos

```
        <tr>
            <td><span datafld="tipo"></span></td>
            <td><span datafld="tipof"></span></td>
            <td><span datafld="codigo"></span></td>
            <td><span datafld="nome"></span></td>
         </tr>
        </table>
    </body>
</html>
```

XML Namespaces

XML Namespaces

- Provêem um método para evitar conflitos entre nomes de elementos.

- Definem um método para definir nomes de elementos e atributos XML, associando-os com referências URI.

Conflitos de nomes

Documento XML que carrega informações em uma tabela:

```
<table>
<tr>
 <td>Apples</td>
 <td>Bananas</td>
</tr>
</table>
```

Documento XML que carrega informações sobre uma mesa:

```
<table>
     <name>Table</name>
     <width>80</width>

<length>120</length>
</table>
```

Resolvendo esse problema...

- Prefixos
- Namespaces

Atributo de Namespace

- É colocado na start-tag de um elemento e possui a sintaxe:

  `xmlns:namespace-prefix="namespace"`

- A especificação de *namespaces* do W3C estabelece que o namespace deveria ser um URI (*Uniform Resource Identifier*).

Prefixos

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

```
<f:table>

<f:name>Table</f:name>

<f:width>80</f:width>

<f:length>120</f:length>
</f:table>
```

Namespaces

```
<h:table
xmlns:h="http://www.w3.com/TR/html4">

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

```
<f:table
xmlns:f="http://www.w3schools.com/fur">

<f:name>Table</f:name>
    <f:width>80</f:width>

<f:length>120</f:length>
</f:table>
```

XSL
(*eXtensible Stylesheet Language*)

Princípios Básicos

Introdução

- As XSLs são as folhas de estilo XML.

- XSL consiste de duas partes:
  - um método para **transformar** documentos XML;
  - um método para **formatar** documentos XML.

Introdução

- Com XSL, pode-se:
  - transformar XML em HTML,
  - filtrar e ordenar dados XML,
  - formatar dados XML baseados em seus valores (como mostrar números negativos em vermelho) e
  - fornecer dados XML para diferentes saídas (como tela, papel ou voz).

Introdução

Por quê duas linguagens de estilo?

	CSS	XSL
Pode ser usado com HTML?	sim	não
Pode ser usado com XML?	sim	sim
Linguagem de transformação?	não	sim
Sintaxe	CSS	XML

Introdução

- XSL consiste de duas partes:
  - **XSLT** *(XSL Transformation)*
    - Linguagem para transformar documentos XML em outros documentos XML.
  - *XSL Formatting Objects* (**XSL-FO**)
    - vocabulário para formatar documentos XML.
    - Define o resultado de uma transformação XSL em uma forma de saída apropriada.
- Usa **XPath** como tecnologia auxiliar.

XSLT e XPath

- **XSLT** é utilizada para definir as transformações XML.
- **XSLT** pode:
  - acrescentar ou remover elementos ao arquivo de saída,
  - re-arranjar os elementos,
  - tomar decisões sobre quais elementos apresentar ou não.

- **XPath** é utilizada para definir “padrões” compatíveis para as transformações.

XSLT e XPath

- XSLT utiliza XPath para definir as partes do documento fonte que “casam” com um ou mais templates pré-definidos.

- XSLT transforma a parte que “casou” do documento fonte em um documento resultante.

XPath

- Sintaxe

Transformação XML para HTML

- XML exemplo:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="cd_catalogo.xsl"?>
  <CATALOGO>
            <CD codigo="R456">
                  <TITULO>Empire Burlesque</TITULO>
                  <ARTISTA>Bob Dylan</ARTISTA>
             </CD>
             <CD codigo="R123">
                  <TITULO>The right thing to do</TITULO>
                  <ARTISTA>Carly Simon</ARTISTA>
             </CD>
</CATALOGO>
```

Transformação XML para HTML

```
<?xml version='1.0'?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
   <html> <body>
     <table border="2" bgcolor="yellow">
        <tr>
          <th>Titulo</th>
          <th>Artista</th>
       </tr>
      <xsl:for-each select="CATALOGO/CD">
        <tr>
           <td><xsl:value-of select="TITULO"/></td>
           <td><xsl:value-of select="ARTISTA"/></td>
       </tr>
     </xsl:for-each>
   </table> </body> </html>
</xsl:template>
</xsl:stylesheet>
```

Transformação XML para HTML

- Ao abrir o arquivo **cd_catalogo.xml** no Browser, qual será o resultado ??

Transformação XML para HTML

- A *tag* **`xsl:stylesheet`** define o início do XSL.
- A *tag* **`xsl:template`** define o início da descrição do formato (template).
  - O atributo **`match="/"`** associa o formato à raiz (/) do documento XML fonte.
- O resto do documento descreve o formato, exceto as últimas duas linhas que definem o fim do formato e o fim do XSL.

Transformação XML para HTML

- O elemento **<xsl:value-of select>** é usado para selecionar elementos XML para a saída da transformação XSL.

- O elemento XSL **<xsl:for-each select>** é usado para selecionar todos os elementos para a saída da transformação XSL.
  - O elemento **xsl:for-each** localiza elementos no documento XML e repete uma parte do formato para cada um.

Transformação XML para HTML

- O atributo **order-by**:
  - Pode-se usar o atributo order-by junto ao elemento for-each para ordenar a saída.
  - (+) ou (-) definem se a ordenação será ascendente ou descendente.
  - Exemplo:

```
<xsl:for-each select="CATALOGO/CD" order-by="+ ARTISTA">
```

Transformação XML para HTML

- Filtro com XSL:
  - Para filtrar o arquivo XML, é preciso adicionar um filtro no elemento **for-each** no arquivo XSL.

  - Exemplo:

```
<xsl:for-each
  select="CATALOGO/CD[ARTISTA='Bob Dylan']">
```

  - Operadores válidos para o filtro são:
    - **=** (igual)
    - **!=** (diferente)
    - **<** (menor que)
    - **>** (maior que)

Transformação XML para HTML

- Condição IF em XSL
  - basta adicionar um elemento **xsl:if** no arquivo XSL.
  - Exemplo:

```
<xsl:for-each select="CATALOGO/CD">
   <xsl:if "ARTISTA='Bob Dylan'">
      <tr>
         <td><xsl:value-of select="TITULO"/></td>
         <td><xsl:value-of select="ARTISTA"/></td>
      </tr>
   </xsl:if>
</xsl:for-each>
```

Transformação XML para HTML

Condição CHOOSE em XSL

- Serve para testar o conteúdo do XML.

Adiciona-se **`xsl:choose`**, **`xsl:when`** e **`xsl:otherwise`** no arquivo XSL.

- Formato da condição choose:

```
<xsl:choose>
<xsl:when test=".[ARTISTA='Bob Dylan']">
...
</xsl:when>
<xsl:otherwise>
...
</xsl:otherwise>
</xsl:choose>
```

Transformação XML para HTML

- Trecho do arquivo XSL (tabela) com condição choose:

```
<table border="2" bgcolor="yellow">
<tr> <th>Titulo</th> <th>Artista</th> </tr>
<xsl:for-each select="CATALOGO/CD">
   <tr>
     <td><xsl:value-of select="TITULO"/></td>
     <xsl:choose>
        <xsl:when test=".[ARTISTA='Bob Dylan']">
            <td bgcolor="red"><xsl:value-of select="ARTISTA"/></td>
        </xsl:when>
       <xsl:otherwise>
            <td><xsl:value-of select="ARTISTA"/></td>
       </xsl:otherwise>
    </xsl:choose>
  </tr>
</xsl:for-each>
</table>
```

Transformação XML para HTML

- O resultado do exemplo anterior no browser ??

Templates

O elemento `<xsl:apply-templates>`

Exemplo 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>
.....
```

Apresentação em XML - XPath

- Uma expressão de caminho pode ser:
  - *absoluta* (/): possuindo interpretação independente do contexto
    - Exemplo:

```
/booklist
```

  - *relativa*: que é avaliada em relação a um determinado elemento do documento (o contexto)
    - Exemplo (contexto:booklist):

```
book/publisher
```

Apresentação em XML - XPath

- Coringas: “*”

```
/*/author
```

- Operadores: “//”, “..” e “.”

- Outros caminhos: *parent::, child::, preceding-sibling:, following-sibling:, ancestor::, descendant::, self::, attribute::, following::*

```
ancestor::book/title
```
(contexto author)

Apresentação em XML - XPath

- Testes de posição

```
/booklist/book[position()=1]/title
```

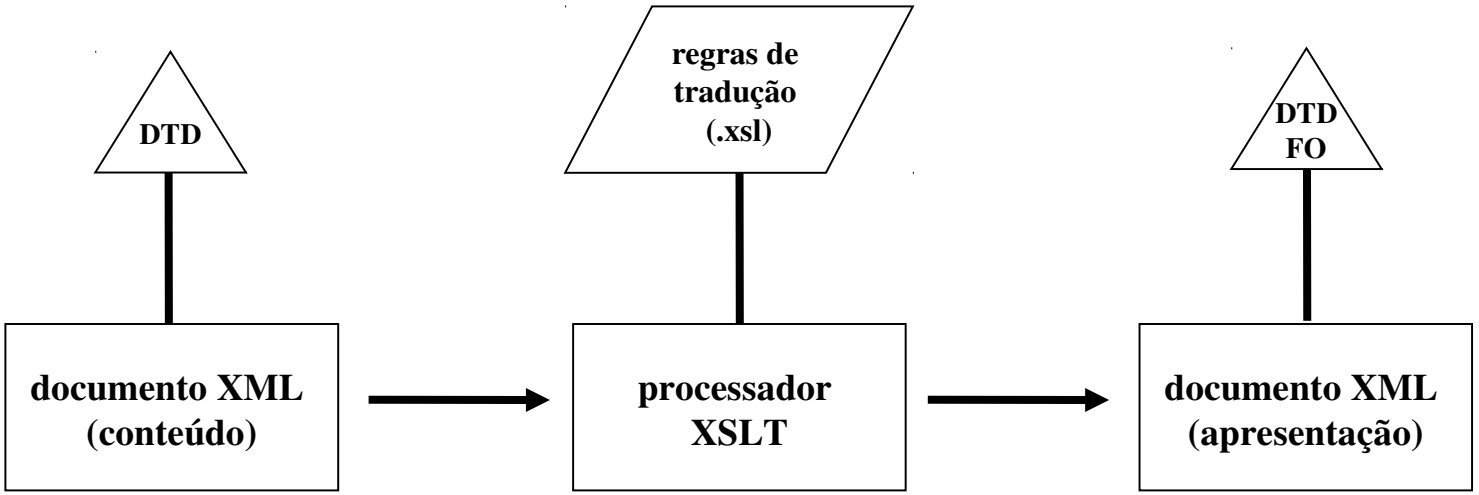
- Acesso a atributos

```
/booklist/book/@id
```

```
/booklist/book[@id='MAR99']/author
```

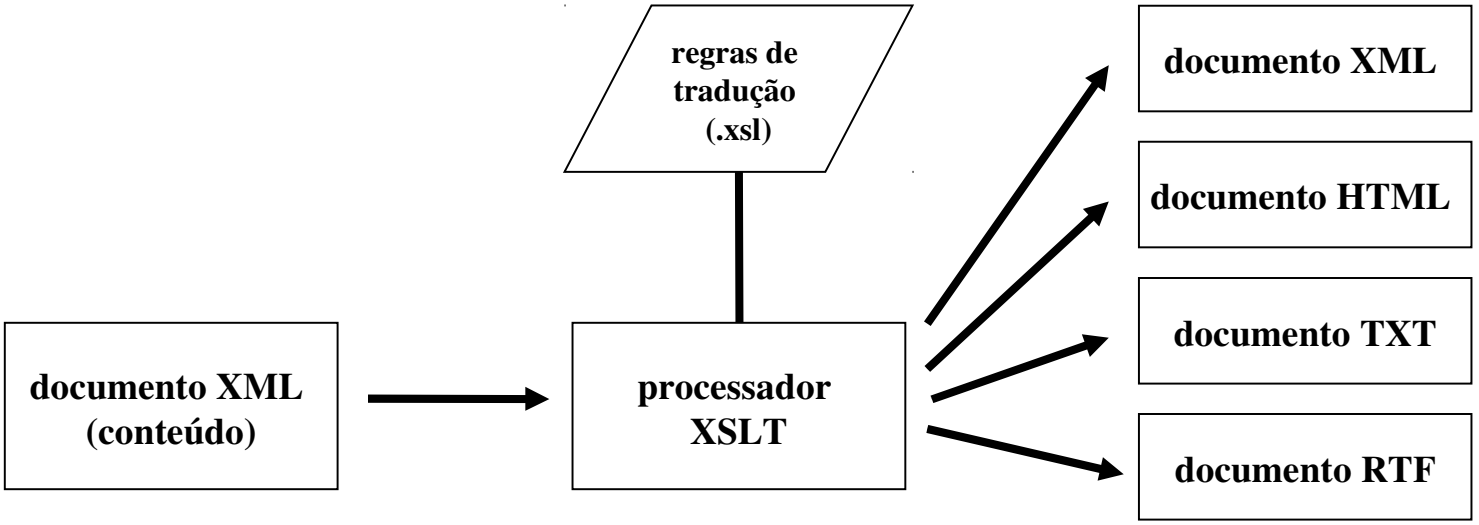
Apresentação em XML - XSLT

- XSL Transformations (XSLT)
  - O objetivo é gerar um documento na linguagem XSL:FO



Apresentação em XML - XSLT

- XSL Transformations (XSLT)
  - Pode-se gerar documentos em qualquer formato texto



Apresentação em XML - XSLT

- Através da XSLT é possível
  - adicionar prefixos e/ou sufixos a um conteúdo de texto
  - eliminar, criar, reestruturar e ordenar elementos
  - reusar elementos em qualquer parte do documento
  - transformar dados em formato XML em qualquer outro formato baseado em texto
  - especificar os objetos de formatação XSL que são aplicados a cada classe de elementos

Apresentação em XML - XSLT

- Processador XSLT
  - Entrada: documento XML
  - Saída: documento em formato texto

- Caso o formato de saída seja XML, o documento pode se basear em uma DTD diferente da DTD do documento de entrada

- A transformação é especificada em uma folha de estilo utilizando a sintaxe do padrão XML

Apresentação em XML - XSLT

- *Folhas de estilo*
  - formadas por um conjunto de regras (*template*) de transformação
  - cada regra “casa” com um tipo de elemento no documento de entrada referenciado por expressões *XPath*

Apresentação em XML - XSLT

- *Estrutura geral de uma folha de estilo*
  - *Deve se ter um elemento raiz 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>
```

Apresentação em XML - XSLT

- *Referência no documento de entrada*

```
<?xml-stylesheet type=\"text/xsl\"
   href=\“stylesheet.xsl\"?>
```

- onde “stylesheet.xsl” referencia o arquivo com as regras de transformação XSLT

Apresentação em XML - XSLT

- *Processamento recursivo*
  - O processador varre o documento XML, encontra um elemento e busca na *style sheet* um *template* que “casa” com o elemento em questão
  - Executa a regra de transformação
  - Ao encontrar a regra “xsl:apply-templates”, processa os filhos deste elemento de forma análoga

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- *Processamento recursivo*

```
<?xml version="1.0"?>
<?xml-stylesheet type=\"text/xsl\"
href=\"stylesheet.xsl\"?>
<books>
        <book>
                <title>ABC</title>
                <author>John</author>
        </book>
        <book>
                <title>DEF</title>

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

documento.xml

```
<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:template match="title">
                ...
                <xsl:apply-templates/>
        </xsl:template>
        <xsl:template match="author">
                ...
                <xsl:apply-templates/>
        </xsl:template>
</xsl:stylesheet>
```

stylesheet.xsl

Apresentação em XML - XSLT

- Uma *folha de estilos* vazia aplica as *regras default* ao documento XML de entrada
  - Processa todo o documento
  - Coloca na saída todo o conteúdo dos elementos texto

Apresentação em XML - XSLT

- Exemplo 1: a *style sheet* a seguir

```
<?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>
```

- gera o seguinte resultado

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

Apresentação em XML - XSLT

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

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

Apresentação em XML - XSLT

- Exemplo 1: a *style sheet* a seguir

```
<?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>
```

- gera o seguinte resultado

==========Lista de livros==========
Box, D. and Skonnard, A. and Lam, J.SeriesEssential XML – Beyond MarkupAddison-Wesley2000Julyhttp://www.develop.com/books/essentialxml
Maruyama, H. and Tamura, K. and Uramoto, N.XML and Java: Developing of Web ApplicationsAddison-Wesley1999MAAugustBradley, N.The XML

Apresentação em XML - XSLT

- Exemplo 1: a *style sheet* a seguir

```
<?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>
```

- gera o seguinte 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.
```

Apresentação em XML - XSLT

- Atributo *select*
- *Operador "//"*

Apresentação em XML - XSLT

- Folha de estilos

```
<?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>
```

Apresentação em XML - XSLT

- Resultado

Livro

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

Apresentação em XML - XSLT

- Folha de estilos

```
<?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">
      Lista de livros dos autores:==============================
      <xsl:apply-templates select="//author"/>
      ==========================================
      <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:template match="title">
      TITULO: <xsl:value-of select="."/>
   </xsl:template>
```

Apresentação em XML - XSLT

- Resultado

Lista de livros dos autores:==========
AUTOR: Box, D. and Skonnard, A.
AUTOR: Maruyama, H. and Tamura,
AUTOR: Bradley, N.
==========
_____ Livro _____
AUTOR: Box, D. and Skonnard, A.
TITULO: Essential XML - Beyond
_____ Livro _____
AUTOR: Maruyama, H. and Tamura,
TITULO: XML and Java: Developing
_____ Livro _____
AUTOR: Bradley, N.
TITULO: The XML Companion

Apresentação em XML - XSLT

- Geração de espaços em branco no resultado

```
<xsl:text> </xsl:text>
```

Apresentação em XML - XSLT

- Folha de estilos – Geração de documento XML de resultado

```
<?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>
```

Apresentação em XML - XSLT

- 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>
```

Apresentação em XML - XSLT

- Folha de estilos – Geração de documento BibTex

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="text"/>
    <xsl:template match="booklist">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="book">
        @Book{<xsl:value-of select="@id"/>,<xsl:if test="string-
    length(author)>0">
        author = {<xsl:value-of select="author"/>}</xsl:if><xsl:if
    test="string-length(title)>0">,
        title = {<xsl:value-of select="title"/>}</xsl:if><xsl:if test="string-
    length(address)>0">,
        addres = {<xsl:value-of select="address"/>}</xsl:if><xsl:if
    test="string- length(editor)>0">,
        editor = {<xsl:value-of select="editor"/>}</xsl:if><xsl:if
    test="string- length(edition)>0">,
        edition = {<xsl:value-of select="edition"/>}</xsl:if><xsl:if
    test="string- length(month)>0">,
        month = {<xsl:value-of select="month"/>}</xsl:if><xsl:if test="string-
    length(year)>0">,
        year = {<xsl:value-of select="year"/>}</xsl:if>
        }
    </xsl:template>
</xsl:stylesheet>
```

Apresentação em XML - XSLT

- Resultado

```
@Book{BOX00,
      author = {Box, D. and Skonnard, A. and Lam, J.},
      title = {Essential XML - Beyond Markup},
      editor = {Series},
      month = {July},
      year = {2000}
}
@Book{MAR99,
      author = {Maruyama, H. and Tamura, K. and Uramoto, N.},
      title = {XML and Java: Developing of Web Applications},
      addres = {MA},
      month = {August},
      year = {1999}
}
@Book{BRA00,
      author = {Bradley, N.},
      title = {The XML Companion},
      addres = {Great Britain},
      edition = {2},
      month = {August},
      year = {2000}
}
```

Apresentação em XML - XSLT

- Folha de estilos – elemento sort

```
<?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:sort select="year"/></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>
```

Apresentação em XML - XSLT

- Resultado

_____ Livro _____
AUTOR: Maruyama, H. and Tamura, K. and Uramoto, N.
_____ Livro _____
AUTOR: Box, D. and Skonnard, A. and Lam, J.
_____ Livro _____
AUTOR: Bradley, N.

Apresentação em XML - XSLT

- Variável

```
<xsl:variable name="cor">verde</xsl:variable>
```

- *Template*

```
<h1> A cor é
<xsl:value-of select="$cor"/>.<h1>
```

- Resultado

A cor é **verde**.

Apresentação em XML - XSLT

- Utilizando valores de atributos na saída (@)

Apresentação em XML - XSLT

- Documento de entrada

```
<nome-completo primeiro="João" segundo="Silva"/>
```

- *Template*

```
<xsl:template match=”nome-completo">
   <pessoa nome="{@primeiro} {@segundo}"/>
</xsl:template>
```

- Resultado

<pessoa nome="João Silva"/>

Apresentação em XML - XSLT

- Documento de entrada

```
<nome-completo primeiro="João" segundo="Silva"/>
```

- *Template*

```
<xsl:template match=”nome-completo">
   <pessoa>
       <xsl:value-of select="@primeiro"/>
       <xsl:value-of select="@segundo"/> -
       <xsl:apply-templates/>
   </pessoa>
</xsl:template>
```

- Resultado

```
<pessoa>João Silva - ... </pessoa>
```

Apresentação em XML - XSLT

- Obtendo maior controle sobre a execução
  - *for-each*
  - *if*
  - *choose*

Apresentação em XML - XSLT

- *Template*

```
<xsl:template match="text()“/>
<xsl:template match="booklist">
   <xsl:for-each select="book">
      LIVRO--------------------
      AUTOR: <xsl:value-of select="author"/>
   </xsl:for-each>
</xsl:template>
```

- 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.

Apresentação em XML - XSLT

- *Template*

```
<xsl:template match="text()“/>
<xsl:template match="book">
    LIVRO ----------------------
    <xsl:choose>
        <xsl:when test="author='Bradley, N.'">
            AUTOR 1
        </xsl:when>
        <xsl:otherwise>
            OUTRO AUTOR....
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>
```

- Resultado

```
LIVRO ----------------------
OUTRO AUTOR....
LIVRO ----------------------
OUTRO AUTOR....
LIVRO ----------------------
AUTOR 1
```

Apresentação em XML - XSLT

- *Template*

```
<xsl:template match="text()“/>
<xsl:template match="book">
   LIVRO ------------------------
   <xsl:if test="author='Bradley, N.'">
      AUTOR 1
   </xsl:if>
</xsl:template>
```

- Resultado

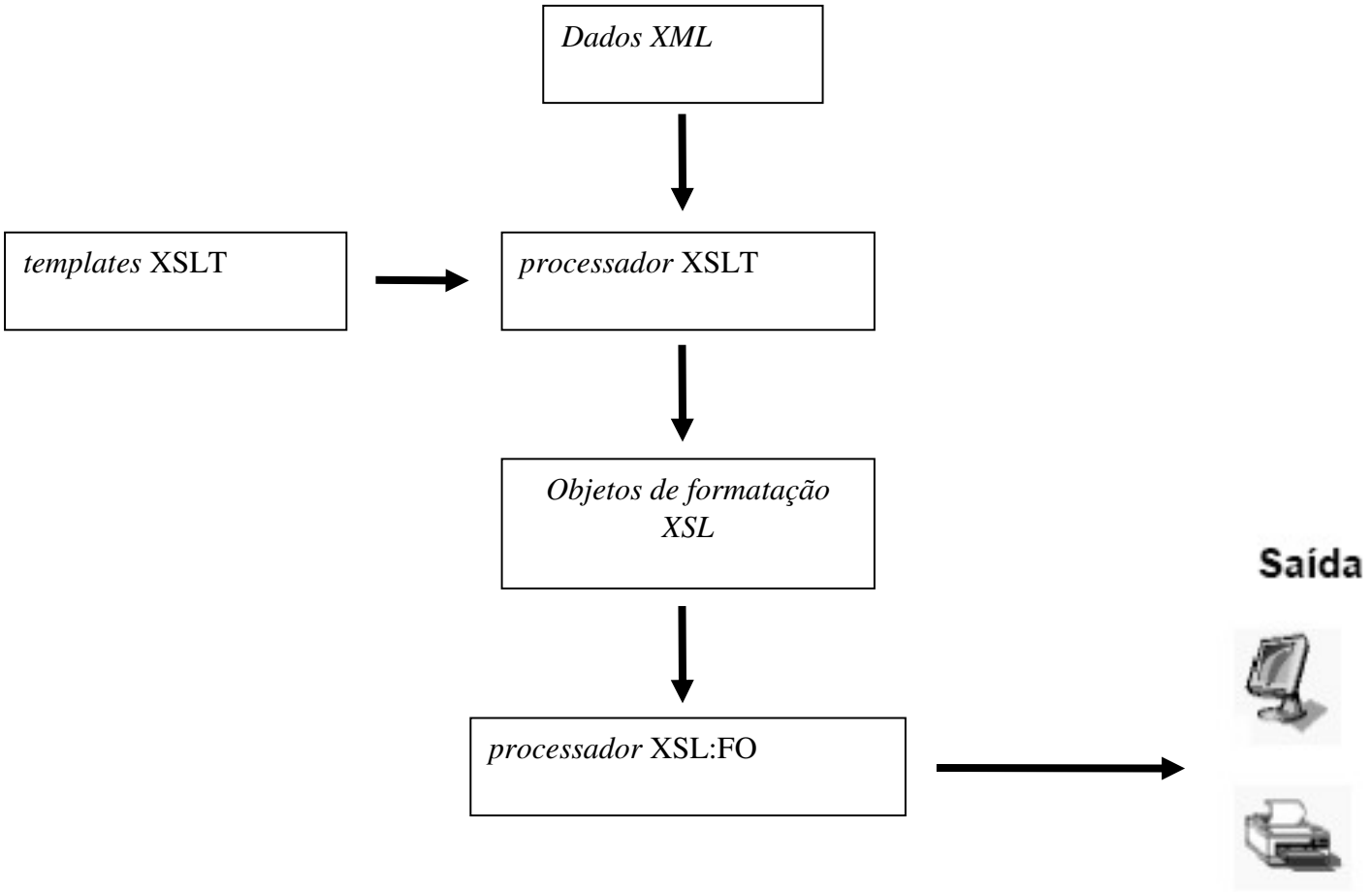
LIVRO ----------
LIVRO ----------
LIVRO ----------
AUTOR 1

Apresentação em XML – XSL:FO

- **XSL Formatting Objects:** é a parte da XSL que especifica como uma informação estruturada pode ser apresentada ou visualizada. Ela pode ser vista como uma linguagem XML para especificar: layout, paginação, ou outras informações de estilo, tais como cor, fontes, espaçamento, entre outras

- Mais abrangente do que a CSS, pois permite a apresentação em dispositivos além do navegador Web

Apresentação em XML – XSL:FO



Apresentação em XML – XSL:FO

- Exemplo de criação de tabela em XSL:FO

```
<fo:table border=" 1pt solid black" background-color="# EEEEEE">
    <fo:table-column column-width=" 20mm"/>
    <fo:table-column column-width=" 30mm"/>
    <fo:table-body>
        <fo:table-row>
            <fo:table-cell border=" 2pt solid black" padding=" 3pt">
                <fo:block> HTML </fo:block>
            </fo:table-cell>
            <fo:table-cell border=" 2pt solid black" padding=" 3pt">
                <fo:block> XML- FO </fo:block>
            </fo:table-cell>
        </fo:table-row>
        <fo:table-row>
            <fo:table-cell border=" 1pt solid black" padding=" 3pt">
                <fo:block> abaixo HTML </fo:block>
            </fo:table-cell>
            <fo:table-cell border=" 1pt solid black" padding=" 3pt">
                <fo:block> abaixo XML- FO </fo:block>
            </fo:table-cell>
        </fo:table-row>
...
    </fo:table-body>
```

Apresentação em XML – CSS vs XSL

- Acredita-se que, apesar de ambas tecnologias (CSS e XSL) permitirem a apresentação de documentos XML, a principal diferença existente entre elas é que a CSS possui escopo de apresentação limitado aos navegadores Web, enquanto a XSL permite a apresentação de documentos em navegadores ou qualquer outro dispositivo que aceite como entrada documentos em formato de texto

Elementos XSLT

Elemento	Descrição
`xsl:apply-imports`	Aplica um template para uma folha de estilo importada.
`xsl:apply-templates`	Aplica um template ao elemento atual e aos seus filhos.
`xsl:attribute`	Adiciona um atributo ao elemento mais próximo.
`xsl:attribute-set`	Define um conjunto de atributos.
`xsl:call-template`	Maneira de chamar um template já definido.
`xsl:choose`	Maneira de escolher entre um número de alternativas baseadas em condições.
`xsl:comment`	Cria um comentário XML.
`xsl:copy`	Copia para a saída o elemento atual sem seus sub-elementos e atributos.
`xsl:copy-of`	Copia para a saída o elemento atual com seus sub-elementos e atributos.

Elementos XSLT (Continuação)

Elemento	Descrição
`xsl:decimal-format`	Define conversão decimal-string
`xsl:element`	Adiciona um novo elemento à saída.
`xsl:fallback`	Alternativa para instruções não implementadas.
`xsl:for-each`	Cria um loop no documento de saída.
`xsl:if`	Maneira de criar estrutura condicional.
`xsl:import`	Importa uma folha de estilo.
`xsl:include`	Inclui uma folha de estilo.
`xsl:key`	Fornece maneira de definir uma chave.
`xsl:message`	Escreve uma mensagem na saída.
`xsl:namespace-alias`	Mapeia um namespace para outro namespace.
`xsl:number`	Escreve um número formatado na saída.

Elementos XSLT (Continuação)

Elemento	Descrição
xsl:otherwise	Indica o que deve ocorrer se nenhuma das condições de <xsl:when> for satisfeita na condição <xsl:choose>.
xsl:output	Maneira de controlar a saída transformada.
xsl:param	Maneira de definir parâmetros.
xsl:preserve-space	Definir e manipular espaços em branco.
xsl:processing-instruction	Escreve uma instrução de processamento na saída.
xsl:sort	Maneira de definir ordenação.
xsl:stylesheet	Define o elemento raiz da folha de estilo.
xsl:template	Define um template para a saída.
xsl:text	Escreve texto na saída.
xsl:value-of	Insere um valor na saída.

Elementos XSLT (Continuação)

Elemento	Descrição
`xsl:variable`	Maneira de declarar variáveis.
`xsl:when`	Define uma condição a ser testada e executa uma ação se a condição for verdadeira. É um elemento filho de <xsl:choose>.
`xsl:with-param`	Maneira de passar parâmetros para templates.

Ferramentas

- Para processar um documento XML utilizando XSL, é preciso um *parser* XML com uma XSL *engine*.

- Processadores XSLT
  - XT (James Clark em Java)
  - Oracle's XML Parser for Java v2
  - Apache Xalan-Java

- Browsers com suporte a XSLT
  - Netscape 6 (Recomendação W3C)
  - IE 5.0 e 5.5 (versão Working Draft)
  - IE 6.0 (Recomendação W3C)
  - Firefox

- Apache Cocoon → processador que fica no lado servidor

Tarefa

- Desenvolver 2 DIFERENTES formas de apresentação de sua grade horária;
  - Que inverta a disposição dos dias da semana e dos horários, como uma nova grade;
  - Que apresente a lista de disciplinas cursadas em ordem alfabética
- Desenvolver as folhas de estilos para gerar essas 2 visões diferentes dos dados de sua grade horária.
-
- Entregar via agora.tidia-ae.usp.br, como Atividades, até a próxima aula.

Referências Bibliográficas

- Página oficial de XSL no site do W3C:
  http://www.w3c.org/Style/XSL/
- XPath:

  http://www.w3.org/TR/xpath

  www.w3schools.com/xpath/

Resumo

Visto:

XSL

Xpath

namespace