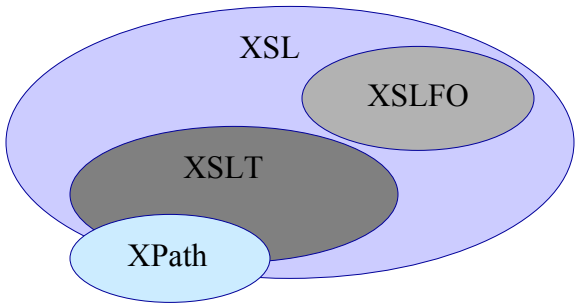
XSLT

Processamento Estruturado de Documentos 2003

By jcr

Sub-linguagens



Processo de Transformação

1. Construção da árvore documental abstracta (exemplificar)
2. Análise da stylesheet procurando instruções para aplicar à árvore
3. As instruções estão contidas em modelos (templates)

Dois modelos de processamento

- tree-driven (DOM)
  - manipulação directa da ADA previamente criada
- event-driven (SAX)
  - transformações executadas na abertura e fecho das anotações

Modelo (template)

- Cada modelo tem duas partes:
  - um seleccionador de nodos
  - as instruções que compõem a transformação que se pretende aplicar
- O processador procura primeiro por um modelo a ser aplicado à raiz **'/'**.
- Geralmente é este modelo que coordena toda a transformação

Contexto (conceito)

- Todas as acções em XSL têm em conta o contexto
  - o contexto é semelhante à directoria corrente num sistema de ficheiros
  - todas as acções são relativas ao contexto

Anatomia de uma stylesheet

- xsl:stylesheet
- Elementos de topo
- xsl:template
- Outros elementos

Exemplo: o poema

```
<?xml version="1.0" encoding="iso-8859-1"?>
<poema>
  <titulo>"Soneto Já Antigo</titulo>
  <autor>(Álvaro de Campos)</autor>
  <corpo>
    <quadra>
      <verso>Olha, <nome>Daisy</nome>: quando eu morrer tu hás-de</verso>
      <verso>dizer aos meus amigos aí de<lugar>Londres</lugar>,</verso>
      <verso>embora não o sintas, que tu escondes</verso>
      <verso>a grande dor da minha morte. Irás de</verso>
    </quadra>
    ...
    <terno>
      <verso>embora não o saibas, que morri...</verso>
      <verso>Mesmo ele, a quem eu tanto julguei amar,</verso>
      <verso>nada se importará... Depois vai dar </verso>
    </terno>
    ...
  </corpo>
  <data>(1922)</data>
</poema>
```

xsl:stylesheet

xsl:stylesheet

```
<xsl:stylesheet
  id = id
  version = number
  <!-- Conteúdo: (xsl:import*,elementos-top) -->
</xsl:stylesheet>

<xsl:transform
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
...
</xsl:stylesheet>
```


Elementos de topo (filhos de xsl:stylesheet)

- xsl:import
- xsl:include
- xsl:strip-space
- xsl:preserve-space
- xsl:output
- xsl:key
- xsl:decimal-format
- xsl:namespace-alias
- xsl:attribute-set
- xsl:variable
- xsl:param
- xsl:template

Elementos de topo: xsl:output

```
<xsl:output
    [method = "html|text|xml"]
    [encoding = "endoding-name"]
    [omit-xml-declaration = "yes|no"]
    [standalone = "yes|no"]
    [indent = "yes|no"] />
```

```
<xsl:output
    omit-xml-declaration="yes"
    method="html"
    encoding="iso-8859-1"
    indent="yes"/>
```


Elementos de topo: xsl:template

```
<xsl:template
    [match = "xpath-exp"]
    [priority = "número-inteiro"]
    [name = "identificador"]
    [mode = "identificador"] />
```

```
<xsl:template match="titulo">
  <h2><xsl:apply-templates/></h2>
</xsl:template>
```

xsl:apply-templates

```
<xsl:apply-templates
    [select = "node-set-exp"]
    [mode = "mode"] />
```

```
<xsl:template match="/poema">
  <H2><xsl:value-of select="titulo"/></H2>
  <HR/>
  <xsl:apply-templates select="corpo"/>
  <HR/>
</xsl:template>
```


xsl:value-of

```
<xsl:value-of
    [select = "xpath-exp"]
    [disable-output-escaping = "yes|no"] />
```

```
<xsl:template match="data">
   <h3><xsl:apply-templates/></h3>
</xsl:template>
```

```
<xsl:template match="data">
   <h3><xsl:value-of select="."/></h3>
</xsl:template>
```

templates por omissão

Para nodos elemento e nodo raiz

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

Para um determinado modo m

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

Para nodos texto e atributo

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


A stylesheet mais simples!

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output
    omit-xml-declaration="yes"
    method="html"
    encoding="iso-8859-1"
    indent="yes"/>

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

</xsl:stylesheet>
```

O resultado da transformação

```
 "Soneto Já Antigo  (Álvaro de Campos)
Olha, Daisy: quando eu morrer tu hás-de
dizer aos meus amigos aí de Londres,
embora não o sintas, que tu escondes      a
grande dor da minha morte. Irás de
Londres p'ra Iorque, onde nasceste (dizes
que eu nada que tu digas acredito),
contar áquele pobre rapazito      que me
deu horas tão felizes, ...
```


Transformação selectiva

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output
    omit-xml-declaration="yes"
    method="html"
    encoding="iso-8859-1"
    indent="yes"/>

<xsl:template match="/">
  <xsl:apply-templates select="poema/corpo/quadra"/>
</xsl:template>

</xsl:stylesheet>
```

xsl:copy e xsl:copy-of

```
<xsl:copy>
    ...
</xsl:copy>
```

```
<xsl:copy-of
    select = "xpath-exp" />
```


Transformação identidade (v1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output
    omit-xml-declaration="yes"
    method="text"
    encoding="iso-8859-1"
    indent="yes"/>

<xsl:template match="@*|node()">
  <xsl:copy>
    <xsl:apply-templates select="@*|node()"/>
  </xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

Transformação identidade (v2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output
    omit-xml-declaration="no"
    method="xml"
    encoding="iso-8859-1"
    indent="yes"/>

<xsl:template match="/">
     <xsl:copy-of select="."/>
</xsl:template>

</xsl:stylesheet>
```


xsl:for-each - iterando

```
<xsl:for-each
    ...
</xsl:for-each>
```

```
<xsl:for-each select="aula">
   <h3><xsl:value-of select="data"/></h3>
   ...
</xsl:foreach>
```

xsl:sort - alterando a ordem

```
<xsl:sort
    select = "xpath-exp"
    [data-type = "text|number"]
    [order = "ascending|descending"]
    [case-order = "upper-first|lower-first"] />
```

```
<xsl:for-each select="aula">
  <xsl:sort
    select="data" order="ascending"/>
  <h3><xsl:value-of select="data"/></h3>
   ...
</xsl:foreach>
```


Transformando os sumários

O modelo da raiz:

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

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

Transformando os sumários (2)

O modelo das aulas:

```
<xsl:template match="aula[@tipo='T']">
  <HR>Aula Teórica:
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="aula[not(@tipo='T')]">
  <HR>Aula Prática:
  <xsl:apply-templates/>
</xsl:template>
```


Transformando os sumários (3)

O modelo do professor:

```
<xsl:template match="url">
  <xsl:variable name="conteudo" select="."/>
  <A HREF="{$conteudo}">
    <xsl:value-of select="."/>
  </A><BR/>
</xsl:template>

<xsl:template match="nome">
  <xsl:apply-templates/>
  <BR/>
</xsl:template>
```

Transformando os sumários (4)

O modelo da data:

```
<xsl:template match="data">
  [<B>
    <xsl:apply-templates/>
  </B>]
  <BR/>
</xsl:template>
```


Transformando os sumários (5)

Sumários e parágrafos:

```
<xsl:template match="sumario">
  <BR/>
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="p">
  <P/>
  <xsl:apply-templates/>
</xsl:template>
```

Transformando os sumários (6)

Os restantes modelos:

```
<xsl:template match="disciplina">
  <H1><xsl:apply-templates/></H1>
</xsl:template>

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

<xsl:template match="item">
  <LI><xsl:apply-templates/></LI>
</xsl:template>
```


Demonstração

- Fazer demonstração da stylesheet criada e do output que é gerado

Geração dum índice

- É necessário fazer mais de uma travessia da árvore documental (não há variáveis globais).
- Vamos usar o atributo “mode”.

Gerar o índice com XSLT

```
<xsl:template match="/">
    <H3>Índice Cronológico das aulas</H3>
    <xsl:apply-templates mode="indice"/>
    <HR/>
    <xsl:apply-templates/>
</xsl:template>
```

Gerar o índice com XSLT (2)

```
<xsl:template match="data">
  <xsl:variable name="conteudo" select="."/>
  [<B>
    <A NAME="{$conteudo}">
    <xsl:value-of select="."/>
    </A>
  </B>]
  <BR/>
</xsl:template>
```

```
<xsl:temp
    <xsl:v
    [<A H
      <xsl
    </A>]
</xsl:template>
```

Novo modelo para a data

Gerar o índice com XSLT (3)

```
<xsl:template match="text()|@*" mode="indice" priority="-1">
<!-- filtra todos os nodos textuais e atributo -->
</xsl:template>
```

Demonstração final

- Realizar a demonstração final com e sem a filtragem de texto