Document Composer:
uma aplicação XML para extracção de informação de repositórios XML

José Carlos Ramalho

Departamento de Informática
Universidade do Minho
Portugal

Motivação

- Ensinar a manipular o modelo de dados que existe nos principais motores de XML
  - Utilização do XPath
  - Sem conhecimento do XSLT

Ferramentas semelhantes

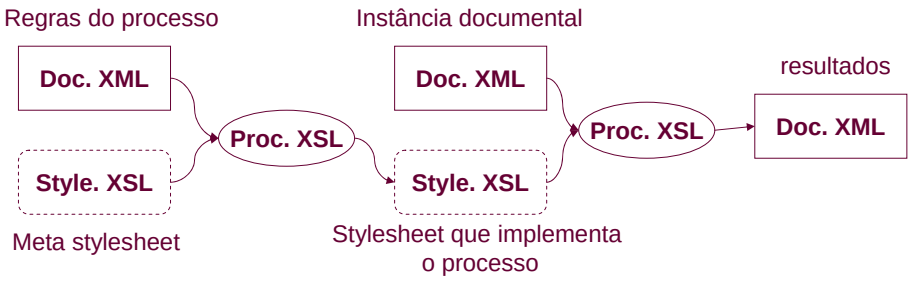
- XML Spy XPath tool
  - Pode ser usada apenas com conhecimentos de XPath
  - Não permite extracção de resultados
- Implementações do XPath ou XQuery
  - Exigem o conhecimento da linguagem em que estão implementados

Arquitectura do Document Composer

- Linguagem de Consulta
  - XPath embebido em XML
- Cálculo de resultados
  - Stylesheet XSLT
- Visualização de resultados
  - Stylesheet XSLT

Como criar a camada de abstracção?

- Utilizando stylesheets de “2ª geração”



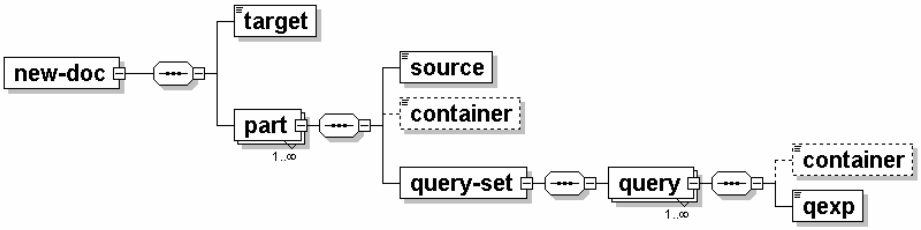
Exemplos de processos

- XCSL, Schematron: validação semântica em documentos XML
- GRAPHOTRON: definição de vistas sobre grafos
- XTCHE: já apresentado
- MTRANS: conversão de modelos MOF

No nosso caso

- Regras do processo: queries escritas em XPath
- Processo: extracção dos nodos que forem seleccionados
- Meta stylesheet: definida nos próximos slides
- Stylesheet que implementa o processo: copia para a saída os nodos seleccionados
- Resultados: documento XML composto pelos nodos seleccionados

XPQL: XPath Query Language



- Algumas preocupações adicionais:
  - Poder colocar queries a mais do que um documento
  - Poder colocar os resultados de uma querie numa subárvore específica
  - Especificar a entrada e a saída conjuntamente

Exemplo1: Arquivo Sonoro de EVO

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
  <arq>
    <doc>
      <prov>Alentejo</prov>
      <local>Santa Vitória, Beja</local>
      <tit>Disse a laranja ao limão</tit>
      <musico>Jorge Montes Caranova
              (viola campaniça)</musico>
      <file t="MP3">d1/evo001.mp3</file>
      <duracao>1:02</duracao>
      …
  </ar
```

- Seleccionar o título das músicas que contêm a palavra “Jesus”
- Seleccionar o nome de todos os músicos referidos
- Seleccionar o título de todas as músicas de “Castelo Branco”

Exemplo1: XPQL

```
<?xml version="1.0" encoding="UTF-8"?>
<new-doc>
   <target>doc-result</target>
   <part>
     <source>arq-son-EVO.xml</source>
     <container>result-set</container>
     <query-set>
       <query>
         <container>result</container>
         <qexp>//tit[contains(.,'Jesus')]</qexp>
       </query>
       <query>
          <container>result</container>
          <qexp>//musico</qexp>
       </query>
       <query>
         <container>result</container>
         <qexp>//doc/tit[contains(../local,'Castelo Branco')]</qexp>
       </query>
     </query-set>
   </part>
</new-doc>
```

Exemplo2: Lista de publicações e autores

- Existe um repositório XML com publicações
- Um autor só é preenchido no primeiro registo, nos seguintes é utilisada uma referência
- Pretende-se uma lista de publicações do ano 2003
- Devido às referências, a resposta tem duas partes: publicações e autores

Exemplo2: XPQL

```
<?xml version="1.0" encoding="UTF-8"?>
<new-doc>
 <target>bienio0304</target>
 <part>
   <source>jcrpubs.xml</source>
   <container>pubs2003</container>
   <query-set>
     <query>
       <qexp>//*[year='2003']</qexp>
     </query>
     <query>//autor</query>
   </query-set>
 </part>
</new-doc>
```

Cálculo das queries

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

 <xsl:namespace-alias stylesheet-prefix="axsl" result-prefix="xsl"/>

<xsl:template match="/">
<axsl:stylesheet version="1.0">
    <axsl:output method="xml" omit-xml-declaration="no"
          encoding="iso-8859-1" standalone="yes" indent="yes"/>
    <axsl:template match="/">
      <xsl:element name="{new-doc/container}">
        <xsl:apply-templates select="new-doc/part"  mode="do-all-parts"/>
      </xsl:element>
    </axsl:template>
    <xsl:apply-templates/>
</axsl:stylesheet>
 </xsl:template>
```

- Coexistência de dois níveis de comandos XSL

Cálculo das queries (2)

```
<xsl:template match="query" mode="do-all-parts">
   <axsl:apply-templates mode="P{count(../../preceding-sibling::*)}Q{count(preceding-
sibling::*)}" select="document('{../../source}')"/>
   </xsl:template>

 <xsl:template match="part" mode="do-all-parts">
   <xsl:choose>
     <xsl:when test="container">
        <axsl:element name="{container}">
           <xsl:apply-templates mode="do-all-parts" select="query-set"/>
        </axsl:element>
     </xsl:when>
     <xsl:otherwise>
         <xsl:apply-templates mode="do-all-parts" select="query-set"/>
     </xsl:otherwise>
   </xsl:choose>
</xsl:template>
```

- Garantir a disjunção na selecção: separar as queries em travessias distintas

Cálculo das queries (3)

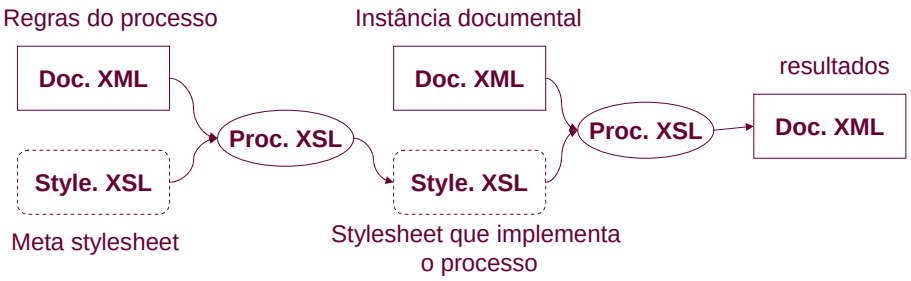
```
<xsl:template match="query">
  <axsl:template mode="P{count(../../preceding-sibling::*)}
                       Q{count(preceding-sibling::*)}" match="{qexp}">
      <xsl:choose>
        <xsl:when test="container">
          <xsl:element name="{container}">
            <axsl:copy-of select="."/>
          </xsl:element>
        </xsl:when>
        <xsl:otherwise>
           <axsl:copy-of select="."/>
        </xsl:otherwise>
      </xsl:choose>
  </axsl:template>
</xsl:template>
```

Produção de Resultados

```
<?xml version="1.0" encoding="UTF-8"?>
<new-doc>
  <target>doc-result</target>
  <part>
    <source>arq-son-EVO.xml</source>
    <container>result-set</container>
    <query-set>
      <query>
        <container>result</container>
        <qexp>...</qexp>
      </query>
      ...
    </query-set>
  </part>
  ...
</new-doc>
```

Pipeline de execução

- Utilizando “makefiles”



Conclusão

- Testada nalguns casos práticos de pequena dimensão
- No contexto de aprendisagem teve bons resultados
- Em desenvolvimento:
  - Interface gráfica
  - Suporte de NameSpaces
  - Novos atributos associados ao elemento container
  - Criar um NameSpace próprio e uma lista de casos de uso.
  - Suportar a pipeline de execução através de XPDL.

Q&A

XML: Aplicações e Tecnologias Associadas

http://vecpar.fe.up.pt/xata2005

10 e 11 de Fevereiro
Braga