Formatarea documentelor XML utilizând XSLT

Mihai Gabroveanu

Separarea conținutului de stil

- Avantajele separării conţinutului de stilul de afişare:
  - reutilizarea fragmentelor de date: acelaşi conţinut poate fi afişat diferit în contexte diferite
  - formate multiple de ieşire: PDF, HTML, RTF, etc.
  - stilurile se pot construi în funcţie de preferinţele cititorului: culori, dimensiune, formă, etc.
  - stiluri standardizate: stylesheet-uri unitare se pot aplica asupra conţinutului

Ce este XSL?

- XSL (Extensible Stylesheet Language) este o familie de limbaje utilizată pentru transformarea şi randarea documentelor XML
- XSL constă din următoarele trei limbaje:
  - XPath = limbaj utilizat pentru a selecta părți ale unui document XML în vederea trasformarii cu XSLT
  - XSLT (XSL Transformations) = limbaj utilizat pentru transformarea documentelor XML în alte tipuri le documente (cel mai adesea în XHTML, deci poate fi afisat)
  - XSL-FO (XSL Formatting Objects) un limbaj pentru formatare (inlocuitor pentru CSS) utilizat pentru transformarea documentelor XML în documente cu format binar (PDF, RTF, etc.)

XPath

- Considerăm următorul document XML:

```
<?xml version="1.0"?>
<biblioteca>
  <carte id="1000">
    <titlu>XML</titlu>
    <autor>Gregory Brill</autor>
     <an>2001</an>
  </carte>
  <carte id="1012">
    <titlu>Java and XML</titlu>
    <autor>Brett McLaughlin</autor>
    <an>2006</an>
  </carte>
</biblioteca>
```

- Expresiile XPath sunt similare căilor ce identifică fişierele şi directoare într-un sistem de fişiere
  - / reprezintă documentul în sine (dar nu elemente specifice)
  - /biblioteca identifică elementul rădacină
  - /biblioteca/carte selectează toti copii cu numele carte
  - /biblioteca/carte[0] selectează primul element carte
  - /biblioteca/carte[0]/@id selectează atributul id din primul element carte
  - //autor selectează toate elementele de tip autor, indiferent de locul unde se gasesc în document

XSLT

- XSLT = Extensible Stylesheet Language Transformations
- Standard W3C, 1999: http://www.w3.org/TR/xslt
- XSLT utilizat pentru trasformarea documentelor XML în alt tip de document
- XSLT utilizează două fişiere de intrare:
  - Documentul XML ce conţine datele
  - Documentul XSL ce conţine:
    - "framework" în care se vor insera datele,
    - Comenzile XSLT ce se vor executa

Sintaxa fişierelor .xsl

- Un document XSLT are extensia .xsl
- Un document XSLT începe cu:

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

- Conţine unu sau mai multe template-uri de forma:

```
<xsl:template match="/"> ... </xsl:template>
```

- Se termină cu:

```
</xsl:stylesheet>
respectiv
</xsl:transform>
```

Exemplu

- Fisierul email.xml:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="email.xsl"?>
<EMAIL id="121">
   <FROM>mihaiug@central.ucv.ro</FROM>
   <TO>giurca@tu-cottbus.de</TO>
   <SUBJECT>Hello World!</SUBJECT>
</EMAIL>
```

- Fiserul email.xsl:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
  <xsl:template match="/">
    <html>
      <body>
        <h1>Mesaj</h1>
        <b>From:</b><xsl:value-of select="EMAIL/FROM"/><br/>
        <b>To:</b><xsl:value-of select="EMAIL/TO"/><br/>
        <b>Subiect:</b><xsl:value-of select="EMAIL/SUBJECT"/><br/>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

XSLT Processor

- Fisierul email.html:

```
<html>
  <body>
    <h1>Mesaj</h1>
     <b>From:</b> mihaiug@central.ucv.ro <br/>
    <b>To:</b> giurca@tu-cottbus.de <br/>
    <b>Subiect:</b>Hello World!<br/>
  </body>
</html>
```

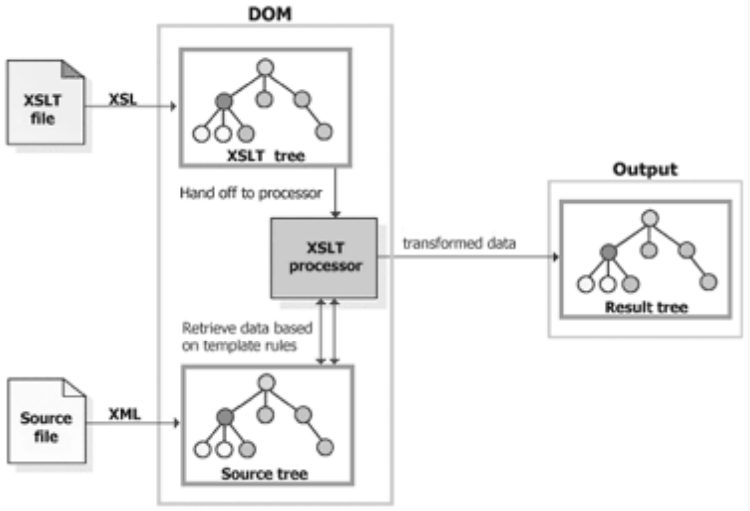
Transformarea documentului XML

- Template-ul <xsl:template match="/"> ne spune că selectăm întregul document XML (radacina)
- Elementele <html><body><h1>Mesaj</h1>From: sunt copiate identic în arborele de ieşire
- Comanda XSLT <xsl:value-of select="EMAIL/FROM"/> selectează valoarea nodului FROM care este copil al nodului EMAIL
- Elementele
To: sunt copiate identic în arborele de ieşire
- Comanda XSLT <xsl:value-of select="EMAIL/TO"/> selectează valoarea nodului TO care este copil al nodului EMAIL
- Etc.

Cum funcționează?

- Documentul sursa XML este parsat şi transformat într-un arbore sursă XML
- Cu ajutorul XPath se definesc template-uri care identifică anumite parţi din arborele sursă cu care se *potrivesc* (*match*)
- Prin intermediul XSLT sunt *transformate* părţile care s-au potrivit cu template-urile, informațiile astfel obțiunte sunt inserate în arborele rezultat
- Arborele rezultat este trimis la ieșire ca document rezultat
- Părțile din documentul sursă care nu se potrivesc cu nici un template sunt copiate în arborele rezultat nemodificate.

Procesorul XSLT



Utilizări ale XSLT

- Prin intermediul unor procesoare XSLT, precum JAXP, Saxon, Xalan, etc. putem aplica transformări XSLT asupra documentelor XML generând astfel noi documente
- Un server poate utiliza XSLT pentru a transforma o mulţime de fişiere XML în fişiere HTML, înainte de a le trimite clientului (transformare server side)
- Browserele moderne sunt capabile să aplice transformări XSLT asupra documentelor XML pentru a le transforma în documente HTML (transformare client side)
- Etc.

Sintaxa comenzilor XSLT

- Limbajul XSLT pune la dispoziţie elemente pentru:
  - Definirea regulilor de transformare
  - Definirea de variabile și parametri
  - Procesarea repetitiva, condiţionată, sortarea elementelor
  - Includerea de fişiere cu alte transformări

<xsl:value-of>

- <xsl:value-of select="*expresie XPath*"/> selectează conţinutul unui element/atribut și îl adaugă în stream-ul de ieșire
- Exemplu :

```
<h1>Mesaj:<i><xsl:value-of select="EMAIL/@id"/></i></h1>
<b>From:</b><xsl:value-of select="EMAIL/FROM"/><br/>
<xsl:value-of select="1+7"/>
```

<xsl:for-each>

- <xsl:for-each> permite aplicarea repetată a unui set de actiuni asupra unei mulţimi de noduri
- Sintaxa:

```
<xsl:for-each   select="Expresie XPath">
    <!-- Text de inserat si/sau reguli de aplicat -->
</xsl:for-each>
```

- Exemplu: Să presupunem că dorim sa afişăm pentru fiecare carte selectată (//carte) o lista neordonată (<ul>) cu titlurile acestora (titlu):

```
<ul>
  <xsl:for-each select="//carte">
    <li><xsl:value-of select="titlu"/></li>
  </xsl:for-each>
</ul>
```

Filtrarea

- Putem restricţiona mulţimea de elemente prin adăugarea de criterii de selecţie în expresia XPath:

```
<ul>
   <xsl:for-each select="//carte[./an &gt; 2002]">
     <li><xsl:value-of select="titlu"/></li>
   </xsl:for-each>
 </ul>
```

- Se vor afişa toate titlurile de cărti care au apărut după anul 2002

<xsl:sort>

- In interiorul unei bucle xsl:for-each putem introduce criterii de sortare utilizând xsl:sort

```
<xsl:sort  select="element" order="ascending|descending">
```

- Exemplu:

```
<ul>
   <xsl:for-each select="//carte">
     <xsl:sort select="an"/>
     <li>   <xsl:value-of select="an"/>
            <xsl:value-of select="titlu">
     </li>
   </xsl:for-each>
 </ul>
```

<xsl:if>

- <xsl:if> ne permite includerea/aplicarea unui set de reguli dacă condiţia specificata în atributul test este adevărată:

```
<xsl:if  test="conditie">
      <!-- Text de inserat si/sau reguli de aplicat -->
</xsl:if>
```

- Exemplu:

```
<xsl:for-each  select="//carte">
  <xsl:if test="autor='Brett McLaughlin'">
    <li>
      <xsl:value-of  select="titlu"/>
    </li>
  </xsl:if>
</xsl:for-each>
```

- Afişează toate carţile scrise de Brett McLaughlin

<xsl:choose>

- Comanda xsl:choose … xsl:when … xsl:otherwise este o instrucţiune de selecţie multiplă
- Sintaxa:

```
xsl:choose>
   <xsl:when test="conditie1">
        ... Instructiuni 1...
   </xsl:when>
   <xsl:when test="conditieN">
        ... Instructiuni N...
   </xsl:when>

    <xsl:otherwise>
        ... instructiuni ...
    </xsl:otherwise>
</xsl:choose>
```

<xsl:text>

- <xsl:text>...</xsl:text> permite generarea de noduri text.
- Sintaxa:

  <xsl:text>*continut nod text*</xsl:text>
- Exemplu:

  <xsl:text>*Limbajul XML*</xsl:text>

Crearea de elemente XML data

- Presupunem ca avem următorul fragment XML

```
<nume>Lect. Dr. Mihai Gabroveanu</name>
<url>http://inf.ucv.ro/~mihaiug</url>
```

- Și dorim să-l transformăm în:

```
<a href="http://inf.ucv.ro/~mihaiug">
Lect. Dr. Mihai Gabroveanu</a>
```

Crearea de elemente – Soluția 1

- Sintaxa instrucţiunilor de creare a elementelor și atributelor

```
<xsl:element name="nume-element">
   <xsl:attribute name="nume-atribut1">
   <!-- Valoare atribut1-->
   </xsl:attribute>
   …
   <xsl:attribute name="nume-atributN">
   <!-- Valoare atributN-->
   </xsl:attribute>
   …
   <!-- Continutul elementului-->
</xsl:element>
```

Crearea de elemente – Exemplu (I)

- Exemplu:

```
<xsl:element name="a">
   <xsl:attribute name="href">
      <xsl:value-of select="url"/>
   </xsl:attribute>
   <xsl:value-of select="nume"/>
</xsl:element >
```

- Rezultat:
  Lect. Dr. Mihai Gabroveanu

Crearea de elemente – Exemplu (II)

- Exemplu:

```
<a>
   <xsl:attribute name="href">
      <xsl:value-of select="url"/>
   </xsl:attribute>
   <xsl:value-of select="nume"/>
</a>
```

- Rezultat:
  Lect. Dr. Mihai Gabroveanu

Crearea de elemente - Solutia 2

- Un Attribute Value Template (AVT) constă din acolade { } utilizate în valoarea unui atribut
- Conținutul referit între acolade este înlocuit cu valoarea lui
- Exemplu:

```
<a href="{url}">
   <xsl:value-of select="nume"/>
</a>
```

- Rezultat:
  Lect. Dr. Mihai Gabroveanu

Variabile

- Definirea unei variabile

  <xsl:variable name="*numeVariabila*" select="*expresieXPath*" />
- Referirea la o variabila: $numeVariabila
- Exemplu:

<xsl:variable name="medie" select="(nota[1]+nota[2]) div 2" />
<xsl:text>Media notelor este:<xsl:text><xsl:value-of select="$medie"/>

Modularizarea

- Modularizarea – programele de dimensiuni mari/complexe sunt împărțite
  - In limbajele de programare avansate acest lucru se realizeaza prin itermediul functiilor şi procedurilor
  - In XSL putem realiza ceva similar prin intermediul xsl:apply-templates
- De exemplu, presupunem că avem un XML pentru reprezentarea unei cărți – book - cu următoarele componente titlePage, tableOfContents, chapter, and index
  - Putem crea cate un template pentru a procesa fiecare din aceste tipuri

Exemplu

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

- <xsl:template match="tableOfContents">
  <h1>Table of Contents</h1>
  <xsl:apply-templates select="chapterNumber"/>
  <xsl:apply-templates select="chapterName"/>
  <xsl:apply-templates select="pageNumber"/>
  </xsl:template>
- Etc.

xsl:apply-templates

- Elementul <xsl:apply-templates> aplică o regulă template asupra nodului curent sau copiilor elementului curent
- Daca adaugam atributul select, atunci regula template se aplica numai asupra copiilor care fac *match* cu expresia specificata
- Daca avem mai elemente <xsl:apply-templates> cu atribute select, atunci nodurile copil sunt procesate în ordinea în care apar elementele <xsl:apply-templates>

Aplicarea template-urilor asupra copiilor

- <carte>
  <titlu>XML</titlu>
  <autor>Gregory Brill</autor>
  </carte>

- <xsl:template match="/">
  <html> <head></head> <body>
  <xsl:value-of select="/carte/titlu"/>
  <xsl:apply-templates select="/carte/autor"/>
  </body> </html>
  </xsl:template>

  <xsl:template match="/carte/autor">
  by <xsl:value-of select="."/>
  </xsl:template>

Apelul unui template

- Unui template ii putem asigna un nume :

  <xsl:template name="*numeTemplate*">
  ...body of template...
  </xsl:template>
- Apelul:

  <xsl:call-template name="*numeTemplate*"/>
- sau:

  <xsl:call-template name="*numeTemplate*">
  ...parametri...
  </xsl:call-template>

Template-uri cu parametri

- Definirea parametrilor:
  <xsl:template name="doOneType">
  <xsl:param name="header"/>
  <xsl:param name="nodes"/>
  ...body of template...
  </xsl:template>
- Apelul
- <xsl:call-template name="doOneType">
  <xsl:with-param name="header" select="'Lectures'"/>
  <xsl:with-param name="nodes" select="//lecture"/>
  </xsl:call-template>