Université de Montréal

XSLT Style Sheets

Jamel Eddine Jridi

Introduction

XSL un langage de transformation pour transferer un document arboresence de XML a un doc graphique de HTML (pas complete) de tout le docuemnts XML vers un autre document XML

- Une famille de spécifications pour la transformation de documents XML
  - **XSLT**: est un langage de transformation des documents XML.
  - **XPath**: est un langage (non XML) pour localiser une portion d'un document XML
  - **XSL-FO**: est le vocabulaire qui décrit les mises en forme de documents XML quel que soit le support : écran, papier, audio, etc.
- XSLT décrit comment transformer des documents XML en d'autres documents XML tels que XHTML.
- Un processeur XSLT prend un document XML en entrée et produit une sortie sur la base des spécifications dans un document XSLT

Principe de XSLT

- Un document XSLT est composé des templates
- XPath est utilisé pour spécifier les éléments auxquels les templates devraient appliquer
- Le contenu d'un template spécifie comment l'élément identifié sera traité
- Le processeur XSLT cherchera les parties du document d'entrée correspondant à un template et appliquer le contenu de ce template lorsqu'une correspondance est trouvée.
- Une des applications les plus courantes de XSLT est de transformer un document XML en un document XHTML
- Une feuille de style XSLT peut être associé à un document XML en ajoutant l'instruction suivante :

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

Exemple de déclaration

data.xml :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="render.xsl"?>
<message>Howdy!</message>
```

render.xsl :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
          xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <!-- one rule, to transform the input root (/) -->
   <xsl:template match="/">
       <html>
          <body>
                <h1><xsl:value-of select="message"/></h1>
          </body>
       </html>
   </xsl:template>
</xsl:stylesheet>
```

en examen: <xsl: ... >

/message/ --> acces a description

pour acces a description: message/description

```
<message>
        <description> ABC </description>
</message>
```

XML:

```
<A>
   <B></B>
  <C></C>
</A>

template match "B"
template match "C"

template match ="/"
    hmtl
        body
              xsl:apply-templates
```

eleve avec note plus grande que 80

//eleve[note spt: 80]

Template XSLT

- Il doit y avoir au moins un `template` dans une feuille de style

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

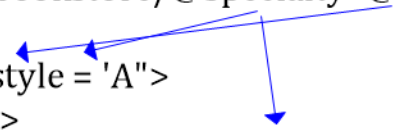
au moins un templete qui va appliquer sur la racine

- La valeur de l'attribut `match` est une expression XPath qui spécifie à quels noeuds le template s'applique
- `'/'` correspond au noeud racine de la structure du document entier
- Le premier `template` est appliqué automatiquement (celui de la racine)
- Tous les autres templates sont appliqués uniquement en utilisant l'instruction `xsl:apply-templates`

Principe de XPath

- Une expression XPath commençant par un `"/"` spécifie des noeuds dans une position absolue par rapport au noeud racine du document
- Dans le cas contraire, l'expression précise des nœuds par rapport au noeud courant, qui est le noeud en cours de traitement.
- L'expression `"."` fait référence au noeud courant
- La balise `apply-templates` utilise l'attribut `select` pour choisir les noeuds qui correspondent aux templates
- Exemples XPath (**important**)

book[/bookstore/@specialty=@style]



<book style = 'A">
</book>
<bookstore speciality="A">
</bookstore>

bookstore//title
booksotre
    title   cela
    book
        title   et cela

livre[@name="..."]

price[@exchange > 15] / total /@value   il existe aussi   total//*@value c-a-d tt les elements dans le total qui ont l'attribut value

author[name][3] = 3e element author qui a element name

author[name[3]] ou author[name[3]="..."]   author/last()   le dernier element dans author

si on travaille avec la racine il faut ajoute un back slash avant ca

* remplace n'importe quel element et on ne peut pas mettre un element juste apres un autre donc on   soit *[ ] soit */

authors/author[position()=2] equivalent a authors/author [2]

methodes importants SUM et COUNT

xsl:apply-templates

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
```

```
<  >
   <cd>
      <title>..</..>
      <artist> ...</..>
      <name>...</name>
   </cd>
   <cd>
      <title>..</..>
      <artist> ...</..>
   </cd>
   <cd>
      <title>..</..>
      <artist> ...</..>
   </cd>
</..>
```

xsl:call-template

- `<xsl:call-template>` utilisé pour transformer XSLT de manière modulaire.
- Vous pouvez nommer un template, puis l'appeler, semblable à la façon dont vous souhaitez appeler une méthode en Java
- La syntaxe est :

```
<xsl:template name="TemplateName">
   …
</xsl:template>
```

- Appeler le template :

```
<xsl:call-template name="TemplateName"/>
```

xsl:for-each

- `xsl:for-each` permet d'exécuter plusieurs fois la même série d'instructions.
- La syntaxe est :

```
<xsl:for-each   select="XPath expression">
   …
</xsl:for-each>
```

- Example: pour sélectionner tous les livres (`//livre`) et faire une liste non ordonnée (`<ul>`) de leurs titres (`titre`)

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

xsl:if

- `xsl:if` permet d'inclure le contenu si une condition donnée (dans l'attribut `test`) est vraie.
- La syntaxe est :

```
<xsl:if   test="condition">
   …
</xsl:if>
```

- Example:

```
<xsl:for-each  select="//livre">
    <xsl:if  test="author='Victor Hugo'">
      <li>
         <xsl:value-of  select="titre"/>
      </li>
    </xsl:if>
 </xsl:for-each>
```

xsl:choose

- `xsl:choose … xsl:when … xsl:otherwise`
- La syntaxe est :

```
<xsl:choose>
      <xsl:when test="condition">
        ... some code ...
      </xsl:when>
      <xsl:otherwise>
        ... some code ...
      </xsl:otherwise>
 </xsl:choose>
```

- Si aucun `<xsl:when>` est vrai, le contenu de `<xsl:otherwise>` est traitée.
- Si aucun `<xsl:when>` est vrai, et pas d'élément `<xsl:otherwise>`, rien ne se passe .

xsl:sort

- On peut placer `xsl:sort` dans un `xsl:for-each`
- Exemple:

```
<ul>
  <xsl:for-each select="//livre">
      <xsl:sort select="auteur"/>
      <li>
           <xsl:value-of select="titre"/>
           <xsl:value-of select="auteur"/>
      </li>
  </xsl:for-each>
</ul>
```

- Cet exemple crée une liste de titres et auteurs, trié par auteur.

xsl:attribute

- Ajouter des attributs à des éléments.
- Supposons que XML :

```
<name>Université de Montréal</name>
<url>http://www.umontreal.ca</url>
```

- Résultat en XHTML :

```
<a href="http://www.umontreal.ca">
    Université de Montréal
</a>
```

- Example:

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

n! = n * (n-1) * (n-2) * ... (n-7)!

fact (n)
    n * fact (n-1)

```
<xsl: for-each select="fact">
<li>
    fact(<xsl: value-of select="."/> = <xsl:call-template name="fact"><xsl:with-param name = "n" select="."/>
</li>
</xsl ...>

<xsl:template name="fact">
        <xsl:param name="n"/>
        <xsl: if test="$n = 1">
        </xsl:if>
       <xsl:if test="$n !gt 1">
            <xsl:varaible name = "r">
                <xsl:call-template name="fact">
                    <xsl: with-param name n select="$n - 1/>
                </xsl:call-template>
            <xsl: value-of select="$n * $r"/>
      </xsl:if>
</xsl:template>
</xsl:stylesheet>
```