



Master 2 RADI 2004 / 2005

Applications documentaires

Devoir XML / XSLT

Julien Van Den Bossche

contact@julienvdb.com

<http://www.julienvdb.com/universite/master/xml>

Sommaire

1. Introduction	Page 3
2. Extraction de structures de traits	Page 3
3. Transformation d'une structure de traits au format MathML	Page 5
4. Document XML + MathML	Page 8
4.1. Document généré manuellement	Page 8
4.2. Ecriture de la feuille XSLT	Page 10
5. Avantages XFS	Page 11
6. Conclusion	Page 12

1. Introduction

Les structures de traits sont des mécanismes très génériques en linguistique pour décrire des paires (propriétés, valeurs). Nous allons voir comment nous allons pouvoir représenter ses données sous un autre format que celui d'origine (XFS). XFS respecte le standard XML. Nous allons voir comment peut on extraire une partie des données du fichier d'entrée via des feuilles de style XSL. Nous allons donc manipuler les expressions XPATH pour pouvoir nous situer dans l'arbre de représentation des données. Nous verrons donc l'intérêt de ces standardisations qui nous permettent de générer de multiples documents sous différent format via un fichier d'entrée et des feuilles de styles XSLT. Le travail a été effectué avec Windows XP SP2 et avec le processeur XT XSLT.

2. Extraction d'une structure de trait

Une structure de traits est repérable dans le fichier *annotations.lss* par les balises *lss :sem*. Ces balises sont des fils de la racine du document XML (*lss :semantics*).

Dans une structure de traits on peut soit retrouver une nouvelle structure de trait ou bien un fragment textuel.

Une structure de traits possède deux attributs qui vont nous permettre d'accéder à cette structure via une feuille XSLT. Ces attributs sont *type* et *id*.

Dans notre feuille de style permettant cette première extraction il a fallu définir les espaces de noms. Le premier correspondant à celui d'une feuille XSLT et le deuxième est celui du fichier *annotations.lss*.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:lss="http://www.linguastream.org/LSS/2.0">
....
```

Comme on vient de le voir notre recherche va se faire sur les attributs *type* et *id* d'une structure de traits. Ces attributs vont donc être définis comme paramètres :

```
<xsl:param name="type" />
<xsl:param name="id" />
```

Notre fichier de sortie contiendra une structure de traits donc il faut récupérer la racine du fichier *annontations.lss* puis appliquer les autres templates que l'on définit plus bas :

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

Maintenant on va rechercher tous nœuds *lss:sem* et on va s'arrêter sur le nœud dont le type et la valeur correspondent à nos paramètres en entrée (*\$type* et *\$id*). Si on trouve le nœud souhaité alors on copie les fils *lss:text* et *lss:value*.
On utilise des expressions XPATH pour ce faire :

```
<xsl:template match="lss:sem">
<xsl:if test="@type=$type and @id=$id">
<lss:sem>
  <xsl:copy-of select="lss:text" />
  <xsl:copy-of select="lss:value" />
</lss:sem>
</xsl:if>
</xsl:template>
```

Après avoir appliqué la feuille de style on obtient le résultat suivant pour le type *intro* avec l'id à *I* :

```
<?xml version="1.0" encoding="utf-8"?>
<lss:semantics xmlns:lss="http://www.linguastream.org/LSS/2.0">
  <lss:sem>
    <lss:text>Depuis le milieu des années 1980</lss:text>
    <lss:value>
      <sem>
        <periode>
          <type>d</type>
          <debut>
            <annees>
              <type>milieu</type>
              <annee>1980</annee>
            </annees>
          </debut>
        </periode>
      </sem>
      <axis>time</axis>
    </lss:value>
  </lss:sem>
```

```

<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:lss="http://www.linguastream.org/LSS/2.0">
  <xsl:param name='type' />
  <xsl:param name='id' />

  <xsl:template match='/'>
    <lss:semantics>
      <xsl:apply-templates/>
    </lss:semantics>
  </xsl:template>

  <xsl:template match='lss:sem'>
    <xsl:if test="@type=$type and @id=$id">
      <lss:sem>
        <xsl:copy-of select='lss:text' />
        <xsl:copy-of select='lss:value' />
      </lss:sem>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>

```

3. Transformation d'une structure de traits au format MathML

Le format de la matrice est :

[X : Y] où X est le nom du nœud et Y est la valeur contenu dans ce nœud X. Y peut donc être du texte ou bien une autre structure de traits.

On va procéder avec le même principe que dans la première question en se plaçant dans l'arbre au nœud souhaité et ainsi on va pouvoir récupérer le contenu.

On se place déjà dans le nœud *lss :semantics* puis *lss :sem*.

La matrice est représentée par des formules MathML donc il ne faudra pas oublier de déclarer son espace de nom :

```

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:lss="http://www.linguastream.org/LSS/2.0"
xmlns="http://www.w3.org/1998/Math/MathML">

```

Ici on s'intéresse à la matrice et non à son texte d'annotations donc il faudra aller au nœud *lss :value*.

```

<xsl:template match="lss:semantics/lss:sem">
  <xsl:call-template name="matrice" />
</xsl:template>

```

Ensuite il faut regarder les fils du nœud racine que l'on vient de définir pour pouvoir appliquer les bonnes balises MathML.

```
<xsl:template name="matrice">
  .....
  .....
  .....
<xsl:param name="root" select="./lss:value" />
```

Pour chaque nœud fils, qui va devenir le nœud courant, on regarde son nombre de fils.

```
<xsl:for-each select="$root/*">
  <xsl:param name="current_node" select="." />
  <xsl:param name="compteur" select="count(./*)" />
```

- A. Si ce nombre est zéro alors cela signifie que ce nœud courant est une chaîne de caractères
- B. sinon s'il possède des fils alors cela signifie que ce nœud est une structure de trait.

Dans le cas A, on va récupérer le nom du nœud par l'instruction *local-name()* et la valeur du nœud par *xsl:value-of*

Dans le cas B on va afficher le nom du nœud puis on relance notre template *matrice* pour pouvoir afficher la structure de traits fille. Cette template va être lancée au fils de *root*.

```
<xsl:call-template name="matrice">
  <xsl:with-param name="racine" select="$root/*" />
</xsl:call-template>
```

Cette feuille de style nous donne le résultat suivant sur la structure de trait extraite par la feuille1 :

$$\left[\begin{array}{l} \text{sem} : \left[\begin{array}{l} \text{periode} : \left[\begin{array}{l} \text{type} : \text{d} \\ \text{debut} : \left[\begin{array}{l} \text{annees} : \left[\begin{array}{l} \text{type} : \text{milieu} \\ \text{annee} : 1980 \end{array} \right] \end{array} \right] \end{array} \right] \end{array} \right] \\ \text{axis} : \text{time} \end{array} \right]$$

Fichier feuille2.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:lss="http://www.linguastream.org/LSS/2.0"
xmlns="http://www.w3.org/1998/Math/MathML">

  <xsl:template match='lss:semantics/lss:sem'>
    <xsl:call-template name="matrice"/>
  </xsl:template>

  <xsl:template name="matrice">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <mfenced open="[" close="]">
        <htable columnalign="left">
          <xsl:param name="root" select="./lss:value"/>
          <xsl:for-each select="$root/*">
            <xsl:param name="current_node" select="."/>
            <xsl:param name="compteur"
select="count(./*)" />
            <xsl:if test="$compteur=0">
              <mtr>
                <mttd>
                  <mi><xsl:value-of select="local-
name($current_node)"/></mi>
                  : <xsl:value-of select="$current_node"/>
                </mttd>
              </mtr>
            </xsl:if>
            <xsl:if test="$compteur>0">
              <mtr>
                <mttd>
                  <mi><xsl:value-of select="local-
name($current_node)"/></mi>
                  : <xsl:call-template name="matrice">
                    <xsl:with-param name="root"
select="$root/*"/>
                  </xsl:call-template>
                </mttd>
              </mtr>
            </xsl:if>
          </xsl:for-each>
        </htable>
      </mfenced>
    </math>
  </xsl:template>
</xsl:stylesheet>
```

4. Document XHTML + MathML

4.1. Document généré manuellement

Dans ce fichier on va mélanger du MathML et du XHTML.

Voici le fichier :

Fichier composite.xhtml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Document XHTML + MathML</title>
</head>
<body>
<p><span style="color:blue; font-weight:bold">Texte :</span> <b>Depuis le
milieu des années 1980</b></p>
<p style="color:blue; font-weight:bold">Matrice :</p>

<math xmlns:lss="http://www.linguastream.org/LSS/2.0"
xmlns="http://www.w3.org/1998/Math/MathML">
  <mfenced open="[" close="]">
    <table columnalign="left">
      <mtr>
        <mtd>
          <mi>sem</mi> :
          <math>
            <mfenced open="[" close="]">
              <table columnalign="left">
                <mtr>
                  <mtd>
                    <mi>periode</mi> :
                    <math>
                      <mfenced open="[" close="]">
                        <table columnalign="left">
                          <mtr>
                            <mtd>
                              <mi>type</mi>: d
                            </mtd>
                          </mtr>
                        </table>
                    </math>
                  <mi>debut</mi> :
                  <math>
                    <mfenced open="[" close="]">
                      <table columnalign="left">
                        <mtr>
                          <mtd>
                            <mi>annees</mi> :
                            <math>
                              <mfenced open="[" close="]">
                                <table columnalign="left">
                                  <mtr>
                                    <mtd>
                                      <mi>type</mi> : milieu
                                    </mtd>
                                  </mtr>
                                </table>
                              </math>
                            </mtd>
                          </mtr>
                        </table>
                    </math>
                  </mtd>
                </mtr>
              </table>
            </mfenced>
          </math>
        </mtd>
      </mtr>
    </table>
  </mfenced>
</math>
```


4.2. Ecriture de la feuille XSLT

On va donc ressortir toutes les structures de traits avec leurs annotations provenant de *annotations.lss*

Pour ce faire nous allons rajouter l'espace de nom de XHTML lors de la création de la template qui crée le document de sortie :

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

On définit une template *texte* qui va nous permettre de récupérer le texte d'annotations d'un nœud. On procède de la même manière que pour la template de la matrice sauf que l'on n'a pas de problème de récursion (matrice dans les matrices).

Ensuite nous allons regarder chaque nœud *lss :sem* puis nous allons rechercher la valeur de la balise *lss :text* et nous allons appliquer la template *matrice* que l'on a déjà défini.

La feuille XSL est la suivante :

feuille3.xsl

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:lss="http://www.linguastream.org/LSS/2.0"
xmlns="http://www.w3.org/1998/Math/MathML">
<xsl:import href='feuille2.xsl' />
  <xsl:template match='/'>
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <title>Document composite</title>
      </head>
      <body>
        <xsl:for-each select="lss:semantics/lss:sem">
          <br/>
          <p><span style="color:blue; font-weight:bold">Texte
: </span>
          <b><xsl:value-of select="lss:text"/></b></p>
          <br/>
          <p><span style="color:blue; font-weight:bold">Matrice
: </span>
          <xsl:call-template name="matrice"/></p>
          <hr/>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Résultat :

Texte : une dizaine d'années

Matrice :
$$\left[\begin{array}{c} \text{periode : } \left[\begin{array}{c} \text{type : df} \\ \text{duree : 1} \\ \text{grain : 10} \end{array} \right] \end{array} \right]$$

Texte : années

Matrice :
$$\left[\begin{array}{c} \text{tag : nom} \\ \text{stag : com} \\ \text{lemma : années} \end{array} \right]$$

Le résultat comportant de nombreuses pages, je n'ai pas tout imprimé, mais ce document est disponible sur <http://www.julienvdb.com/universite/master/xml/sortie3.xhtml>

5. Pertinence du format XFS

Le format XFS me semble correct car on retrouve une structure d'arbre. L'accès à un nœud de l'arbre est assez facile. N'importe quel nœud est identifiable de part le système de balises ouvrantes et fermantes.

Concernant les autres langages qui permettent la représentation des structures de traits on peut retrouver le langage TagML.

TAGML est un standard de description de ressources nécessaires à un analyseur LTAG en XML. Il se traduit par une DTD/XML.

6. Conclusion

Ce devoir m'a permis de bien voir l'intérêt et la puissance d'un langage respectant le standard XML. J'ai pu ainsi manipuler les expressions XPATH et générer des feuilles de style XSLT. Un apprentissage très enrichissant... et qui à mon avis me ressortira aisément plus tard.