Initiation à XSLT

July 2009

XSL: un ensemble de standards complementaires

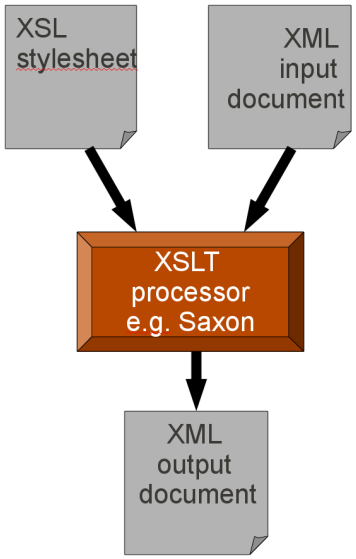
- XPath: un syntaxe normalise pour la definition des sous-parties d'une arborescence XML
- XSLT: un langage informatique pour la transformation des structures XML
- XSL FO: un langage XML pour la description d'affichage des pages

XSLT

Un feuille de style XSLT

- s'exprime en XML
- transforme un document XML dans un autre
- doit se servir des *noms d'espace* pour distinguer les éléments qui represente instructions des éléments traites et des éléments produits
- est exprime par un <xsl:stylesheet> rassemblant au moins un <xsl:template>

Comment se servir d' XSLT ?



Transformation?

Une operation fondamentalle. Par exemple, on veut

```
<item  n="1">
  <name>Burger</name>
</item>
<item  n="2">
  <name>Campbell</name>
</item>
<item  n="3">
  <name>Casagrande</name>
</item>
```

mais on a

```
<persName>
  <forename>Milo</forename>
  <surname>Casagrande</surname>
</persName>
<persName>
  <forename>Corey</forename>
  <surname>Burger</surname>
</persName>
<persName>
  <forename>Naaman</forename>
  <surname>Campbell</surname>
</persName>
```

Un exemple

A partir de ceci :

```
<div type="recette" n="34">
  <head/>
  <list>
    <item>pates</item>
    <item>fromage râpé</item>
  </list>
  <p>Faire bouiller les pates, et melanger avec le fromage.</p>
</div>
```

on veut produire :

```
<html>
  <h1>34: Pasta pour les novices</h1>
  <p>Ingrédients: pates fromage râpé</p>
  <p>Faire bouiller les pates, et melanger avec le fromage.</p>
</html>
```

Comment exprimer cela en XSL?

```
<xsl:stylesheet
   xpath-default-namespace="http://www.tei-c.org/ns/1.0"  version="2.0">
  <xsl:template  match="div">
    <html>
      <h1>
        <xsl:value-of  select="@n"/>:
      <xsl:value-of  select="head"/>
      </h1>
      <p>Ingrédients:
      <xsl:apply-templates  select="list/item"/>
      </p>
      <p>
        <xsl:value-of  select="p"/>
      </p>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Structure d'une feuille de style XSL

```
<xsl:stylesheet
   xpath-default-namespace="http://www.tei-c.org/ns/1.0"  version="2.0">
  <xsl:template  match="div">
<!-- ....  do something with div elements....-->
  </xsl:template>
  <xsl:template  match="p">
<!-- ....  do something with p elements....-->
  </xsl:template>
</xsl:stylesheet>
```

The `div` and `p` are *XPath expressions*, which specify which bit of the document is matched by the template.

Any element not starting with xsl: in a template body is put into the output.

Les règles d'or de XSLT

1. Si aucun template ne correspond a un element, traiter les elements qu'il contient
2. Si aucun element reste a traiter par regle 1, sortir le texte contenu par un element
3. Un element n'est traite que si un template lui correspond
4. `xsl:apply-templates select="XX"` traitez les templates disponibles pour l' element indique par le XPath "XX"; `xsl:value-of select="XX"` sortez le contenu de l'element indique par le XPath "XX".
5. (`xsl:template match="XX"` ne traite rien: il definit un correspondence entre un template et un element)
6. L'ordre des templates est sans significance
7. Tout partie du document est traitable part tout template, plusieurs fois.
8. Un stylesheet ne peut contenir que du XML bien-forme

Deux astuces

Vous etes conseille de fournir ces attributs sur <stylesheet>:

```
<xsl:stylesheet
   xpath-default-namespace="http://www.tei-
c.org/ns/1.0"  version="2.0">....
</xsl:stylesheet>
```

... ce qui signifie :

1. tout element sans prefix dans une expression XPath est considere d'appartenir a l'espace de noms TEI
2. on se sert de la version 2.0 de la specification XSLT.

A simple test file

```
<text>
  <front>
    <div>
      <p>Material up front</p>
    </div>
  </front>
  <body>
    <div>
      <head>Introduction</head>
      <p  rend="it">Some sane words</p>
      <p>Rather more surprising words</p>
    </div>
  </body>
  <back>
    <div>
      <p>Material in the back</p>
    </div>
  </back>
</text>
```

Feature: apply-templates

```
<xsl:stylesheet version="2.0"
   xpath-default-namespace="http://www.tei-c.org/ns/1.0">
  <xsl:template match="/">
    <html>
      <xsl:apply-templates/>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

```
<xsl:template match="TEI">
  <xsl:apply-templates select="text"/>
</xsl:template>
```

```
<xsl:template match="text">
  <h1>FRONT MATTER</h1>
  <xsl:apply-templates select="front"/>
  <h1>BODY MATTER</h1>
  <xsl:apply-templates select="body"/>
</xsl:template>
```

Feature: `value-of`

Templates for paragraphs and headings:

```
<xsl:template match="p">
  <p>
    <xsl:apply-templates/>
  </p>
</xsl:template>
<xsl:template match="div">
  <h2>
    <xsl:value-of select="head"/>
  </h2>
  <xsl:apply-templates/>
</xsl:template>
<xsl:template match="div/head"/>
```

La puissance d' XPath

L'attribut match sur <xsl:template> peut pointer partout dans un document.

`/`	la racine du document
`*`	tout element
`text()`	tout morceau de te
`name`	tout element nomme 'name'
`@name`	tout attribut nomme 'name'

exemple d'un Xpath comme object de <value-of>:

```
<xsl:value-of
   select="/TEI/teiHeader/fileDesc/titleStmt/title"/>
```

Plusieures templates peut etre specifies pour des contextes divers

Compare

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

with

```
<xsl:template match="div/head"> ...
</xsl:template>
<xsl:template match="figure/head"> ....
</xsl:template>
```

En cas de conflit...

Exemple de conflit:

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

Quel template correspondra a une occurrence donnee de <name> dans le document?

Prioritisation

En general, le template le plus specifique gagne.

```
<xsl:template match="*">
<!-- ... -->
</xsl:template>
<xsl:template match="tei:*">
<!-- ... -->
</xsl:template>
<xsl:template match="p">
<!-- ... -->
</xsl:template>
<xsl:template match="div/p">
<!-- ... -->
</xsl:template>
<xsl:template match="div/p/@n">
<!-- ... -->
</xsl:template>
```

Et en cas ou, un attribut priority est disponible sur <template> pour varier cette principe

Une astuce pour les valeurs d'attribut

On a :

```
<ref target="http://www.gallica.bnf.fr">site Gallica</ref>
```

On veut :

```
<a href="http://www.gallica.bnf.fr"/>
```

Ceci ne sera *pas* efficace :

```
<xsl:template match="ref">
  <a href="@target">
    <xsl:apply-templates/>
  </a>
</xsl:template>
```

parce qu'il donnera a l'attribut @*href* la valeur '@target'!

Un astuce syntaxique...

On utilise {} pour indiquer qu'une expressions doit etre evaluée:

```
<xsl:template match="ref">
  <a href="{@target}">
    <xsl:apply-templates/>
  </a>
</xsl:template>
```

ceci donnera a l'attribut @*href* la valeur de l' attribut @*target* quelle que soit la valeur de ce dernier

Bouclage: `for-each`

De temps en temps, il est utile de iterer sur un ensemble d' elements. Par exemple :

```
<xsl:template match="listPerson">
  <ul>
    <xsl:for-each select="person">
      <li>
        <xsl:value-of select="persName"/>
      </li>
    </xsl:for-each>
  </ul>
</xsl:template>
```

cf.

```
<xsl:template match="listPerson">
  <ul>
    <xsl:apply-templates select="person"/>
  </ul>
</xsl:template>
<xsl:template match="person">
  <li>
    <xsl:value-of select="persName"/>
  </li>
</xsl:template>
```

Conditions: `if`

On peut proposer des actions conditionelles :

```
<xsl:template match="person">
  <xsl:if test="@sex='1'">
    <li>
      <xsl:value-of select="persName"/>
    </li>
  </xsl:if>
</xsl:template>
```

cf.

```
<xsl:template match="person[@sex='1']">
  <li>
    <xsl:value-of select="persName"/>
  </li>
</xsl:template>
<xsl:template match="person"/>
```

Choix: choose

On peut specifier un choix multiple, selon ce qu'on trouve dans le texte:

```
<xsl:template match="person">
  <xsl:apply-templates/>
  <xsl:choose>
    <xsl:when test="@sex='1'">(male)
    </xsl:when>
    <xsl:when test="@sex='2'">(female)
    </xsl:when>
    <xsl:when test="not(@sex)">(no sex specified)
    </xsl:when>
    <xsl:otherwise>(unknown sex)
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Numerotation: number

On peut generer une numerotation derivée de la position des elements dans l'arborescence XML text.

1. Position à l'interieur de l'element parent:

```
<xsl:template match="p">
  <xsl:number/>
</xsl:template>
```

2. Position a l'interieur du document entier :

```
<xsl:template match="p">
  <xsl:number level="any"/>
</xsl:template>
```

3. Position a l'interieur d'un element ancetre specifie:

```
<xsl:template match="l">
  <xsl:number level="any" from="lg"/>
</xsl:template>
```

Sommaire

Maintenant vous comprenez comment

1. creer des templates
2. selectionner des morceaux de texte
3. definir des actions conditionnelles
4. numeroter les objets de sortie

On va experimenter cela....