Technique et chaîne de publication électronique avec XSLT (2/7)

Jean-Damien Généro

2023, 9 janv. - 13 fev.

École nationale des chartes – M2 TNAH

XSLT. Règles basiques

XSLT/ `<xsl:template>` (rappels)

- Une règle s'écrit dans un *`<xsl:template>`*;
- L'attribut *`@match`* contient le chemin XPath vers la balise concernée par la règle.

XSLT/ Contenu d'un `<xsl:template>`

- *`<xsl:template>`* peut contenir :
  - Balises XML ou HTML (avec leurs @tt) ou commandes LaTeX ;
  - Du texte ;
  - Des règles (*templates*) ou variables/param XSL : *`<xsl:.../>`* ;

Contenu d'un `<xsl:template>` : fichiers d'exemple

- Ouvrir Oxygen.
- Ouvrir *`/class-2/class2_teiAll_oxygen.xml`*.
- Ouvrir *`/class-2/class2_teiAll_xsl.xsl`*.

XSLT/ Contenu d'un `<xsl:template>` : XML, HTML, LaTeX (1/2)

- Des balises peuvent être écrites directement dans un *`<xsl:template>`* :

```
<xsl:template match="/TEI/text/body">
    <div>
        <head type="chap">
            <xsl:value-of select="./p[1]"/>
        </head>
        <xsl:copy-of select="./p[2]"/>
    </div>
</xsl:template>
```

XSLT/ Contenu d'un `<xsl:template>` : XML, HTML, LaTeX (2/2)

- Les balises XML ou HTML peuvent être écrites dans des *<xsl:element name="tag">* et les attributs dans des *<xsl:attribute name="@tt">* :

```
<xsl:template match="/TEI/text/body">
    <xsl:element name="div">
        <xsl:element name="head">
            <xsl:attribute name="type">chap</xsl:at
            <xsl:value-of select="./p[1]"/>
        </xsl:element>
        <xsl:copy-of select="./p[2]"/>
    </xsl:element>
</xsl:template>
```

XSLT/ Contenu d'un `<xsl:template>` : du texte (1/2)

- Du texte peut être mis dans une règle : pour remplacer le contenu d'une balise, pour écrire le préambule d'un doc LaTeX.
- (Dans le fichier d'exemple : remplacer tout.)

```
<xsl:template match="/">
    \documentclass[a4paper]{book}
    \usepackage[utf8]{inputenc}
    \usepackage[french]{babel}
    \usepackage{fontspec}
    \begin{document}
        <xsl:apply-templates/>
    \end{document}
</xsl:template>
```

XSLT/ Contenu d'un `<xsl:template>` : du texte (2/2)

- Du texte peut aussi être mis dans un *`<xsl:text>text</xsl:text>`* :

```
<xsl:template match="/TEI/text/body">
    <xsl:element name="body">
        <xsl:element name="head">
            <xsl:attribute name="type">chap
                </xsl:attribute>
            <xsl:value-of select="./p[1]"/>
        </xsl:element>
        <xsl:element name="p">
            <xsl:text>nouveau texte</xsl:text>
        </xsl:element>
    </xsl:element>
</xsl:template>
```

Règles basiques : notions essentielles

- Les balises XML et HTML peuvent être écrites en clair dans un *<xsl:template>*;
- Les balises XML et HTML peuvent aussi être écrites dans un *<xsl:element name="tag">* et un attribut dans un *<xsl:attribute name="@att">*.
- Le texte (commandes LaTeX) peut être mis soit en clair soit dans un *<xsl:text>*.

Les quatre principales règles XSL

XSLT/ Les quatre principales règles XSL

- *<xsl:copy/>*
- *<xsl:copy-of/>*
- *<xsl:value-of/>*
- *<xsl:apply-templates/>*

Principales règles (1/4) : `<xsl:copy>`

- Copie de la balise du nœud courant, sans les *namespaces*, attributs, texte, etc.
- *<xsl:copy/>* peut contenir d'autres règles, du texte, etc.
- Intérêt → copier un élément pour appliquer des règles à ses enfants.

```
<xsl:template match="/TEI/text/body">
    <xsl:copy>...</xsl:copy>
</xsl:template>
```

Principales règles (2/4) : `<xsl:copy-of>`

- Copie à l'identique de l'ensemble du nœud courant et de ses nœuds enfants (balises, attributs, textes).
- Le nœud courant est toujours contenu dans le *`@select`*.
- Intérêt → reproduire rapidement une partie de l'arbre que l'on ne veut pas modifier.

```
<xsl:template match="/TEI/text/body">
    <xsl:copy-of select="."/>
</xsl:template>
```

 Principales règles (3/4) : `<xsl:value-of>`

- **Renvoie uniquement la valeur textuelle du nœud courant.**
- Le nœud courant est toujours contenu dans le *@select*.
- Intérêt → copier du texte sans les balises (utile pour LaTeX).

```
<xsl:template match="/TEI/text/body">
    <xsl:value-of select="."/>
</xsl:template>
```

 Exercice. Combinaison des règles

- Quelle différence entre ces templates ?

```
<!-- 1 -->
<xsl:template match="//body/p">
    <xsl:copy-of select="."/>
</xsl:template>
<!-- 2 -->
<xsl:template match="//body/p">
    <xsl:copy>
        <xsl:value-of select="."/>
    </xsl:copy>
</xsl:template>
```

XSLT/ Principales règles (4/4) : `<xsl:apply-templates/>`

- *`<xsl:apply-templates/>`* est une **instruction récursive** : le processeur va **examiner les nœuds enfants** dans l'ordre et **appliquer les règles** qui leur sont associées.

XSLT/ La règle `<xsl:apply-templates/>` (1/4)

- Sans elle, le processeur s'arrête à l'emplacement désigné par le *@match* du *<xsl:template/>* et ne traite pas les éléments enfants.
- Exemple : la balise *<TEI/>*.

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

XSLT/ La règle `<xsl:apply-templates/>` : `@select` (2/4)

- *`@select`* permet d'appliquer les règles uniquement au nœud sélectionné.

- Ex. → inverser *`<teiHeader/>`* et *`<text/>`* (*cf.* fichier) :

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

XSLT/ La règle `<xsl:apply-templates/>` : `@mode` (3/4)

- *`@mode`* permet d'appliquer des règles différentes **à un même élément XML** en fonction de son emplacement ou de son contenu (= son « mode »).
- Le *`@mode`* doit être présent et dans le *`<xsl:apply-templates/>`* et dans le *`<xsl:template/>`* avec la même valeur (= le nom du mode).

XSLT/ La règle `<xsl:apply-templates/>` : `@mode` (4/4)

- Exemple d'utilisation de *`@mode`* : créer une table des matières (*cf.* fichier).

```
<xsl:template match="//body">
    <xsl:copy>
        <xsl:apply-templates />
        <div>
            <head>Table des matières</head>
            <xsl:apply-templates mode="toc"/>
        </div>
    </xsl:copy>
</xsl:template>
```

XSLT/ Une autre règle : `<xsl:number/>` (1/2)

- *`<xsl:number/>`* « compte des éléments de façon séquentielle ».
- Attributs :
  - *`@count`* → définit les éléments de l'input qui seront numérotés dans l'output;
  - *`@level="single|multiple|any"`* → niveau de l'arbre pris en compte pour le comptage;
  - *`@format="1|01|A|a|I|i"`* → format des numéros.

XSLT/ Une autre règle : <xsl:number/> (2/2)

```
<xsl:template match="//body/p">
    <xsl:copy>
        <xsl:attribute name="n">
            <xsl:number
                count="//body/p"
                level="multiple"
                format="a"/>
        </xsl:attribute>
        <xsl:value-of select="."/>
    </xsl:copy>
</xsl:template>
```

- XSLT commence par chercher la règle à appliquer au nœud racine ;
- Cette règle fait appel à d'autres avec *`apply-templates`* (avec ou sans @tt), elles sont appliquées dans l'ordre ;
- S'il n'y a pas de règle, il passe au suivant.

- Donc : l'ordre d'écriture des règles n'a aucune importance ;
- **Attention → restez lisible + commentez votre code !**

→ Écrire une feuille de style XSL avec le fichier de *St Julien l'Hospitalier* :

- Constituer un fichier XML-TEI valide ;
- Numéroter dans des formats différents les `<head>` et les `<p>` ;
- Transformer les `<hi rend="b"/>` en des `<persName>` et les `<hi rend="i"/>` en des `<placename>` ;
- Bonus pour les plus aguérris : ajouter le début du §3 (lien dans le `<teiHeader/>`).