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

Jean-Damien Généro

2025, 1er dec. - 2026, 20 janv.

École nationale des chartes – M2 TNAH

XSLT. Règles basiques

XSLT/ `<xsl:template>`

- Un *`<xsl:template>`* contient une ou plusieurs règles ;
- L'*`@match`* contient le chemin XPath vers la balise qui sera le point de départ de la ou des règles, qui peut s'appliquer à :
  - La balise désignée dans le *`@match`* ;
  - Les enfants ou descendants de cette balise ;
  - N'importe quelle balise de l'arbre *via* des axes (*ancestor*, *preceding*, etc.).
  - La partie de l'arbre sur laquelle la règle s'applique est en général indiquée dans le *`@select`* de l'instruction XSL (*cf. infra*).

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

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

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

- Ouvrir Oxygen.
- Désactiver le paramètre par défaut :
  Preferences > *XML* > *XML Parser* > *RELAX NG*,
  décocher *Add default attributes*

- Ouvrir *`class_2_journal_lefevre.xml`*
- Ouvrir *`class_2_ex_instructions.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="//text/body/div[1]">
    <body>
        <div>
            <head type="chap">
                <xsl:value-of select="./head"/>
            </head>
            <xsl:copy-of select="./p"/>
        </div>
    </body>
</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="//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 directement dans une règle : remplacer le contenu d'une balise, écrire le préambule d'un doc LaTeX.

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

Règles basiques : notions essentielles

- Les balises XML et HTML peuvent être écrites directement 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 directement soit dans un `<xsl:text>`.

Les quatre principales instructions XSL

XSLT/ Les quatre principales instructions XSL

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

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

- Copie de la balise matchée par le `@match`, 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 instructions (2/4) : `<xsl:copy-of>`

- Copie à l'identique de la balise matchée par le `@select` et de ses nœuds enfants (balises, attributs, textes).
- Élément vide (pas de règles internes).
- **Impossible de modifier les éléments copiés !**
- 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>
```

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

- Renvoie uniquement la valeur textuelle de la balise matchée par le `@select`.
- Élément vide (pas de règles internes).
- Intérêt → copier du texte sans les balises (utile pour LaTeX).

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

- 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 instructions (4/4) : `<xsl:apply-templates/>`

- Ouvrir *att_apply_templates_class2.xsl*
- *<xsl:apply-templates/>* est une **instruction récursive** : le processeur va **examiner les nœuds enfants** de la balise matchée par le *@match* dans l'ordre et **appliquer les règles** qui leur sont associées.

XSLT/ L'élément `<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/>* (*cf.* fichier).

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

XSLT/ L'élément `<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/ L'élément `<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 ») *sans avoir à mettre ces règles dans un seul `<xsl:template/>`*.
- 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/ L'élément `<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 instruction : `<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, avec une expression XPath
  - *`@level="single|multiple|any"`* → niveaux de l'arbre pris en compte pour le comptage ;
  - *`@format="1|01|A|a|I|i"`* → format des numéros.

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

```
<xsl:template match="//body//p">
        <xsl:copy>
            <xsl:attribute name="n">
                <xsl:number
                    count="//div[@n]/p"
                    level="any"
                    format="1"/>
            </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 pas importance ;
- **Attention → restez lisible + commentez votre code !**

- Si plusieurs règles s'appliquent à un même nœud $\rightarrow$ **une seule règle est exécutée**.
- XSLT examine la **priorité** de chaque règle : plus un match est spécifique, plus il a de chances d'être choisi :
  - Priorité par défaut : *`node()`*, *`text()`* et `*` à -0.5 (générique); les balises à 0.5 (spécifique).
  - Priorité explicite : *`@priority`*.

→ Écrire une feuille de style XSL avec le fichier du *Journal de Jean le Fèvre* :

- Constituer un fichier XSLT valide ;
- Ajouter un `<editionStmt/>` (avec `<edition/>` et `<respStmt/>`) ;
- 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>`.