CIHAM (UMR 5648), CNRS

Approfondissement de XSLT

Biblissima+, scripts et manuscrits

Ariane Pinche
ariane.pinche@cnrs.fr

Biblissima+, scripts et manuscrits, 5-7 mai 2025

Table of Contents

apply-templates et l'attribut select

```
<xsl:template match="mon_element_TEI">
  <xsl:apply-templates select="sous_elements_xml"/>
</xsl:template>
```

Démonstration dans Oxygen

apply-templates et l'attribut mode

> *Parfois le même contenu en entrée doit apparaître plusieurs fois dans le document de sortie, formaté selon un modèle différent à chaque fois. [...] Les éléments* `xsl:apply-templates` *et* `xsl:template` *peuvent avoir un attribut* `mode` *optionnel qui associe différentes règles à différents usages. L'attribut* `mode` *d'un élément* `xsl:template` *identifie dans quel mode cette règle-modèle doit être activée. Un élément* `xsl:apply-templates` *avec un attribut* `mode` *n'active que la règle modèle avec l'attribut* `mode` *correspondant.*

Elliotte Rusty Harold, W. Scott Means, Philippe Ensarguet *et al.*, *XML en concentré*, Paris, O'Reilly, 2005, p.171.

Exemple avec mode

```
<xsl:template match="element">
  <xsl:apply-templates mode="nom_mode"/>
</xsl:template>

<xsl:template match="sous_element" mode="nom_mode">
  Règles correspondant au mode
</xsl:template>
```

Démonstration dans Oxygen

Structurer un document en HTML (1)

- À partir du fichier `Structure_HTML.html` fourni et des exemples étudiés en cours, commencez à structurer en HTML le texte proposé pour l'exercice.
- Créez une nouvelle feuille XSLT pour générer cette structure.
- Configurez correctement le préambule :
  - Déclarez l'espace de noms TEI.
  - Configurez son utilisation par défaut dans XPath à l'aide de `xpath-default-namespace`.
- Reproduisez en HTML une structure composée de deux sections :
  1. Une section contenant la liste des personnages.
  2. Une section contenant le texte de la scène.
- Pour la liste des personnages, utilisez les attributs `mode` et `select` afin de générer les éléments HTML (<ul>;<li>)

Structurer un document en HTML (2)

- Dans la seconde section, commencez à structurer le texte de la pièce en répliques, en appliquant les notions abordées en session 1.
- Détails de l'implémentation :
  - Chaque prise de parole (`<sp>`) devient une `<div class="speech">` avec un attribut `data-speaker` correspondant au locuteur.
  - À l'intérieur de la réplique, ajouter le nom en gras () du locuteur suivi de <:>, puis le texte de la réplique.
  - Les noms de lieux et de personnages sont balisés avec des `<span>` dotés respectivement des classes `placeName` et `persName`.
  - Pour les éléments `app`
    - le texte des lemmes est intégré dans les éléments <p> en gras;
    - Les variantes textuelles (rdg) sont intégrées sous forme de `<span class="variant">` avec un attribut `title` indiquant les témoins (@wit).
  - Les didascalies (`<stage>`) sont rendues dans des `<p class="stage-direction">`.

Table of Contents

xsl:if

- xsl:if contient un motif, instancié si et seulement si l'expression XPath contenue dans son attribut obligatoire @test est vraie.

Exemple

```
<xsl:template match="mon_element">
   <xsl:if test="chemin_Xpath_a_verifier">
      <xsl:copy-of select="."/>
   </xsl:if>
</xsl:template>
```

Démonstration dans Oxygen

xsl:choose, xsl:when, xsl:otherwise

- L'élément `xsl:choose` sélectionne une possibilité dans une liste de choix.
- Il contient au moins un `xsl:when` avec un attribut `@test` : lorsque la condition est vérifiée, les motifs sont exécutés.
- Il peut aussi contenir un sous-élément optionnel `xsl:otherwise`, exécuté quand aucun `xsl:when` n'est validé.

xsl:choose, xsl:when, xsl:otherwise

Exemple

```
<xsl:template match="mon_element">
   <xsl:choose>
      <xsl:when test="chemin_xpath_a_verifier">
         [motifs à appliquer]
      </xsl:when>
      <xsl:otherwise/>
   </xsl:choose>
</xsl:template>
```

Démonstration dans Oxygen

xsl:for-each

- L'instruction `xsl:for-each` itère sur les nœuds sélectionnés par l'attribut `@select` et applique le modèle de son contenu à chacun des éléments du nœud.

Exemple

```
<xsl:template match="mon_element">
   <xsl:for-each select="sous_elements_a_traiter">
      [motifs à appliquer]
   </xsl:for-each>
</xsl:template>
```

Démonstration dans Oxygen

xsl:sort

- L'instruction xsl:sort est enfant de xsl:apply-templates ou xsl:for-each.
- Elle modifie l'ordre des nœuds contextuels selon une clé de tri.

Attributs :

- @select : clé de tri ;
- @data-type : "text" (par défaut) ou "number" ;
- @order : "ascending" (par défaut) ou "descending" ;
- @case-order : "upper-first" ou "lower-first".

Exemple

```
<xsl:template match="mon_element">
   <xsl:for-each select="sous-elements">
      <xsl:sort select="cle_tri"
                order="ascending"/>
      <xsl:copy-of select="."/>
   </xsl:for-each>
</xsl:template>
```

xsl:for-each-group

- L'instruction xsl:for-each-group permet d'itérer sur des groupes de nœuds.
- L'attribut @select désigne les éléments à traiter.
- L'attribut @group-by détermine la clé de regroupement.
- La fonction current-grouping-key() retourne la valeur de regroupement en cours.

Exemple

```
<xsl:template match="mon_element">
   <xsl:for-each-group select="sous-elements"
                       group-by="cle_regroupement">
      [motifs à appliquer]
   </xsl:for-each-group>
</xsl:template>
```

Démonstration dans Oxygen