

Séance 4 : Conditions et boucles

xsl:if

xsl:if contient un modèle, instancié si et seulement si l'expression Xpath contenue dans son attribut obligatoire *@test* est vraie.

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

xsl:choose, xsl:when, xsl:otherwise

L'élément xsl:choose sélectionne une possibilité dans une liste de choix. Cet élément contient au moins un xsl:when avec un attribut *@test*, quand la condition de *@test* est vérifiée, les motifs contenus par when sont exécutés. ». L'élément xsl:choose peut avoir un sous-élément optionnel xsl:otherwise dont le contenu est retourné si aucun élément xsl:when ne correspond.

```
<xsl:template match="mon_element">
  <xsl:choose>
    <xsl:when test="chemin_xpath_a_verifier"></xsl:when>
    <xsl:otherwise/>
  </xsl:choose>
</xsl:template>
```

Exercice

À partir du fichier Lucain.xml, reproduire l'arbre TEI – Ajouter sur les rdg l'attribut *@type* de valeur "main" quand @wit comprend #Z et "sub" dans tous les autres cas.

III-xsl:for-each, xsl:sort et xsl:for-each-group

for-each

L'instruction xsl:for-each itère sur les nœuds sélectionnés par son attribut *@select* et applique le modèle de son contenu à chacun

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

Exercice

À partir du fichier Lucain.xml, reproduire l'arbre TEI, puis modifier la structure de l'apparat.

Pour chaque apparat, créer deux rdgGrp :

- Le premier possède l'attribut @type="main" et doit contenir le lemme et toutes les leçons de Z (Z1, Z2 etc.) que vous récupèrez à l'aide de for-each.
- Le deuxième possède l'attribut @type="sub" et doit contenir toutes les leçons qui ne contiennent pas Z que vous récupèrez à l'aide de for-each.

sort

L'instruction `xsl:sort` apparaît comme un enfant de `xsl:apply-templates` ou `xsl:for-each`. Il change l'ordre dans lequel les modèles sont appliqués à la liste de nœuds contextuelle de l'ordre du document en un autre ordre, comme l'ordre alphabétique.

Liste des attributs :

- `@select` : clé du tri;
- `@data-type` : par défaut le type est alphabétique, mais avec la valeur "number", on peut passer sur un tri numérique
- `@order` : par défaut "ascending"
- `@case-order` : "upper-first" ou "lower-fisrt".

```
<xsl:for-each select="element">  
  <xsl:sort select="cle_tri" order='ascending|descending' />  
  <xsl:copy-of select="."/>  
</xsl:for-each>
```

Exercice

Reprendre l'exercice précédent et trier l'ordre d'apparition des leçons en fonction de l'attribut wit.

for-each-group

L'instruction *xsl:for-each-group* itère sur les groupes de nœuds sélectionnés par son attribut *select* applique le modèle de son contenu à chacun d'entre eux, tandis que l'attribut *@group-by* sert de clé. Dans le motif, la fonction xpath *current-grouping-key()* permet de retourner la valeur de la clé de regroupement de la boucle en cours.

```
<xsl:for-each-group select="element"  
                  group-by="cle_regroupement">  
    [motifs à appliquer]  
</xsl:for-each-group>
```

Exercice

À partir du fichier triApp.xml, reproduire l'arbre TEI, puis modifier la structure de l'apparat :

Pour chaque appareil, créer des `<rdgGrp>` en fonction de la valeur de *@type* des `<rdg>` contenue dans `<app>` , ajouter sur cet élément la valeur de la clé courante dans un attribut *@type*. Chaque `rdgGrp` doit contenir toutes les leçons d'un même type.