Interroger et transformer des données en XML-TEI

Initiation à Xpath et XSLT

Séance 3 Édition électronique

Master mention Humanité numériques Rennes 2, 14/12/2021

Introduction

Encoder son corpus en XML-TEI permet non seulement de rendre ses données plus pérennes et interopérables, mais aussi plus facilement interrogeables, afin de faciliter l'analyse d'un gros corpus. Des données en XML peuvent également être transformées vers un autre format et publier selon l'encodage sémantique qui a été appliqué aux données.

Un langage de requête pour les données en XML : XPath

XPath

Xpath (*path expression*) est un langage de requête. Il permet d'exprimer un chemin dans l'arborescence d'un document XML.

Xpath a été conçu comme un langage intégré, et non pas un langage autonome. Il fonctionne donc comme un langage-module à incorporer dans d'autres langages (XSLT en l'occurrence).

Il permet de pointer vers des éléments à sélectionner, sans opérer de modification sur ces derniers. Il prend la forme d'une liste de nœuds depuis le point d'entrée où l'on se trouve dans le document. Il accepte aussi les opérations simples et les expressions régulières.

N.B. : Pour mieux visualiser l'arborescence de son encodage, le logiciel Oxygen XML Editor propose un éditeur d'arbre XML dans le menu « Outils ».

XPath

- **La syntaxe XPath**

Une expression de chemin correspond à une séquence d'étapes séparées par l'opérateur « / ». Sans indication particulière la relation se fait d'un élément parent vers un élément enfant.

Exemple : Pour cibler l'élément <hi> descendant de <p> → p/hi

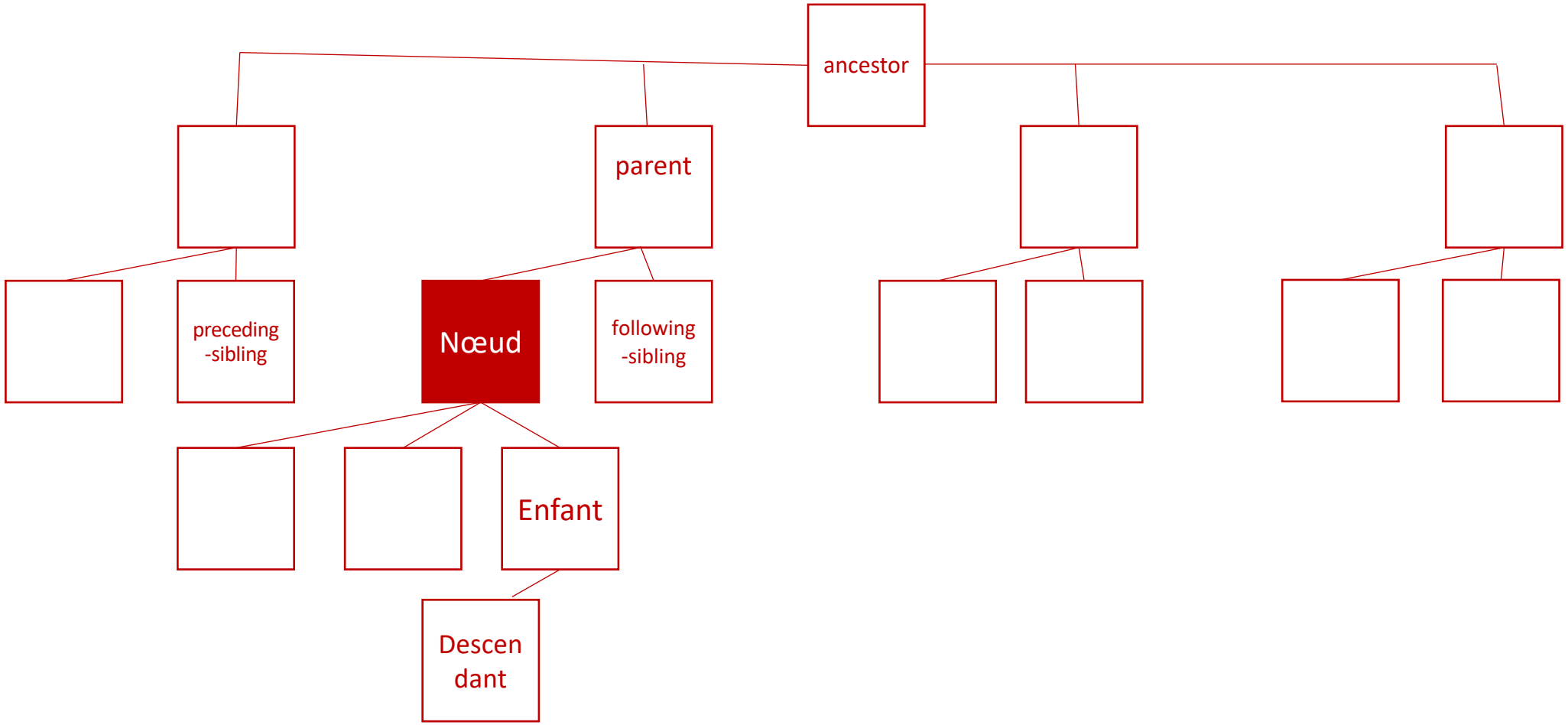
```
<p>
   <hi rend="lettrine">I</hi>nsint corur&e_tilde_n;t <pc type="orig"
      rend="colon_for_number" function="subdistinctio">.</pc>iii<choice>
      <orig>j</orig>
      <reg>i</reg>
   </choice><pc type="orig" rend="colon_for_number" function="subdistinctio">.</pc><pc
      type="reg"> </pc>iorz &z_et;<pc type="reg"> </pc><pc type="orig"
      rend="colon_for_number" function="subdistinctio">.</pc>iii<choice>
      <orig>j</orig>
      <reg>i</reg>
   </choice><pc type="orig" rend="colon_for_number" function="subdistinctio">.</pc><lb/>
```

XPath

- **Axes de relation en XPath :**
  - ancestor : ancestor::node()
  - ancestor-or-self : ancestor-or-self::node()
  - parent : parent::node()
  - descendant : descendant::node()
  - descendant-or-self : descendant-or-self::node()
  - child : child::node() ou node()
  - following : following::element
  - following-sibling : following-sibling::node()
  - preceding :preceding::node()
  - preceding-sibling : preceding-sibling::node()
  - self : self::node() ou .
  - attribute : attribute::node()

XPath

- **Axes de relation en XPath :**



XPath

- **Affiner les requêtes : les prédicats**

Le prédicat est noté entre crochets après l'élément auquel il se rapporte. On peut ainsi spécifier la position d'un élément, le nom ou la valeur de l'un de ses attributs.

Exemples :

- pc**[attribute::type="reg"]**
- p**[position()=1]** ou p**[1]**
- p**[attribute::type= "traduction"][1]**

XPath

- **Abréviations**

  - descendant : **//**
    Exemple : //p

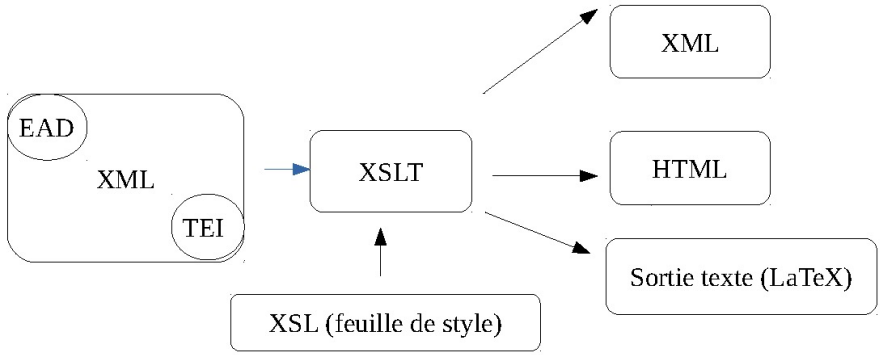
  - self : **.**
    Exemple : ./persName (tous les <persName> depuis là où je suis)

  - attribute : **@nomAttribut**
    Exemple : p[@type='traduction]

Un langage de transformation de données en XML : XSLT

Un langage de transformation de données en XML : XSLT

XSLT (*Extensible Stylesheet Language Transformation*) est un langage permettant de transformer des documents XML en d'autres documents XML, texte ou HTML. C'est un langage déclaratif basé sur des instructions non-ordonnées.



N.B. : Nous verrons dans le cadre de ce cours **XSLT version 2 et XPath 2.0** car les versions 3 ne sont pas encore implémentées partout. Les versions 2 sont des recommandations du W3C depuis 2007.

Un langage de transformation de données en XML : XSLT

- **Fonctionnement**

« Un processeur XSLT lit un arbre XML en entrée et une feuille de style XSL et produit un arbre résultat en sortie. », Elliotte Rusty Harold, W. Scott Means, Philippe Ensarguet [et al.], *XML en concentré*, Paris, O'Reilly, 2005, p. 519.

« Par défaut, un processeur XSLT lit le document XML d'entrée de haut en bas, commençant à l'élément racine et descendant dans l'arborescence en suivant l'ordre d'apparition des éléments (dans l'arbre XML source). Les règles modèles sont activées dans l'ordre de rencontre des éléments. Ceci signifie qu'une règle modèle pour un élément sera activée avant les règles modèles correspondant à ses sous-éléments. », Elliotte Rusty Harold, W. Scott Means, Philippe Ensarguet [et al.], *XML en concentré*, Paris, O'Reilly, 2005, p. 164.

Un langage de transformation de données en XML : XSLT

- **Comment manipuler XSLT ?**
  - Mise en place d'une règle : Utilisation de **templates** et **@match** pour sélectionner un élément de l'arbre XML

On peut ajouter du texte directement à l'intérieur de la règle, ou bien des balises et motifs pour récupérer certains éléments, les transformer, les trier, selon des **fonctions XSL ou XPath**.

```
<xsl:template match="mon_element_xml">
```

Un langage de transformation de données en XML : XSLT

- **Comment manipuler XSLT ?**
  - Ajouter des **éléments** via une feuille xsl :
    - mettre directement l'élément
    - passer par **<xsl:element>** avec l'**@name**

```
<xsl:template match="mon_element_xml">
 <p>Ici, il y avait mon élément</p>
</xsl:template>
```

```
<xsl:template match="mon_element_xml">
<xsl:element name="p">Ici, il y avait
  mon élément</xsl:element>
</xsl:template>
```

```
<xsl:template match="mon_element_xml">
   <xsl:element name="p">
        <xsl:attribute name="type">
             <xsl:text>valeur_attribut</xsl:text>
        </xsl:attribute>
        <xsl:text>Ici, il y avait mon élément</xsl:text>
    </xsl:element>
</xsl:template>
```

```
<xsl:template match="mon_element_xml">
        <p type="{./chemin_Xpath}">Ici,
          il y avait mon élément</p>
</xsl:template>
```

```
<xsl:template match="mon_element_xml">
  <p type="valeur_attribut">Ici, il y avait
    mon élément</p>
</xsl:template>
```

Un langage de transformation de données en XML : XSLT

- **Comment manipuler XSLT ?**
  - Ajouter des **attributs** via une feuille xsl :
    - mettre directement l'attribut et sa valeur
    - mettre directement l'attribut et récupérer sa valeur via une requête XPath
    - passer par **<xsl:attribute>**, l'**@name** et **<xsl:text>** pour la valeur d'attribut

Un langage de transformation de données en XML : XSLT

- **Comment manipuler XSLT ?**
  - La règle motif **apply-templates** : elle permet d'indiquer que les règles définies dans la feuille xsl pourront être appliquées pour les éléments enfants de l'élément appelé dans <xsl:template>.

```
<xsl:template match="mon_element_xml">
     <xsl:element name="p">
          <xsl:apply-templates/>
     </xsl:element>
</xsl:template>
```

Un langage de transformation de données en XML : XSLT

- **Comment manipuler XSLT ?**
  - Les règles motifs **copy et copy-of** :
    - « L'élément XSL copy copie le nœud courant du document source vers le document de sortie. Il ne copie que le nœud lui-même. Cependant il ne copie pas ses enfants et ses attributs. », Elliotte Rusty Harold, W. Scott Means, Philippe Ensarguet[et al.], *XML en concentré*, Paris, O'Reilly, 2005, p. 525.
    - « L'instruction xsl:copy-of insère le fragment d'arbre résultat identifié par l'attribut select dans le document de sortie. Cette instruction copie les nœuds spécifiques sélectionnés par l'expression et tous leurs enfants, attributs, espaces de noms et descendants. C'est en cela qu'il diffère de xsl:copy.», Elliotte Rusty Harold, W. Scott Means, Philippe Ensarguet[et al.], *XML en concentré*, Paris, O'Reilly, 2005, p. 525.

```
<xsl:template match="mon_element_xml">
        <xsl:copy-of select="chemin_Xpath"/>
</xsl:template>
```

```
<xsl:template match="mon_element_xml">
        <xsl:copy/>
</xsl:template>
```

Un langage de transformation de données en XML : XSLT

- Comment manipuler XSLT ?
  - La règle motif **value-of** : « L'élément xsl:value-of calcule la valeur textuelle d'une expression Xpath et l'insère dans l'arbre résultat. », Elliotte Rusty Harold, W. Scott Means, Philippe Ensarguet[et al.], *XML en concentré*, Paris, O'Reilly, 2005, p. 543.

```
<xsl:template match="mon_element_xml">
        <xsl:value-of select="chemin_Xpath"/>
</xsl:template>
```

Un langage de transformation de données en XML : XSLT

- **Pour aller plus loin**
  - La règle motif **xsl:number** pour automatiser une numérotation formatée. On y trouve les attributs :

- L'**@count** est un motif qui spécifie les nœuds à comptabiliser dans les nœuds spécifiés

- L'**@level** défini le nombre de niveaux "any/multiple/single »

- L'**@format** permet de paramétrer le format de numérotation

```
<xsl:template match="mon_element_xml">
    <xsl:number count="element_a_compter" level="" format=""/>
</xsl:template>
```

Un langage de transformation de données en XML : XSLT

- **Pour aller plus loin**
  - **L'attribut mode** de <xsl:apply-templates> : « 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.

```
 <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>
```

Un langage de transformation de données en XML : XSLT

- **Pour aller plus loin**
  - La création de variables avec **xsl:variable** :
    « L'élément *xsl:variable* attache une valeur de n'importe quel type (chaîne de caractères, nombre, ensemble de nœuds, etc.) à un nom. Cette variable peut être déréférencée par la suite en utilisant la forme $nom dans une expression Xpath. » Elliotte Rusty Harold, W. Scott Means, Philippe Ensarguet[et al.], *XML en concentré*, Paris, O'Reilly, 2005, p. 544.

```
<xsl:value-of select="$variable"/>
```

```
<xsl:variable name="nom_variable" select="Xpath"/>
ou
<xsl:variable name="variable"> Règles internes à la variable
</xsl:variable>
```

Un langage de transformation de données en XML : XSLT

- **Pour aller plus loin**
  - **Fonction XPath** et traitement de chaînes de caractères :
    - **concat()** pour souder des chaînes de caractères
    - **replace()** pour modifier une chaîne de caractère
    - **upper-case()** pour passer en majuscule une chaîne de caractères
    - **lower-case()** pour passer en minuscule une chaîne de caractères

Pour voir davantage de fonctions XPath : https://www.w3schools.com/xml/xsl_functions.asp.

```
<xsl:value-of select="concat('element1', 'element2')"
```

```
<xsl:value-of select="replace(element_a_modifier,
                              'caracteres_a_remplacer',
                              'caractere_de_remplacement')"/>
```

Un langage de transformation de données en XML : XSLT

- **Pour aller plus loin**
  - **Conditions et boucles** :
    - **xsl:if** pour appliquer une règle que si l'expression dans l'**@test** est rencontrée
    - **xsl:choose/xsl:when/xsl:otherwise** pour sélection des possibilités dans une liste de choix, avec au moins la condition xsl:when et l'**@test**, tandis que xsl:otherwise traite tous les cas qui ne sont pas concernés par la condition exposée dans xsl:when

```
<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>
```

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

Un langage de transformation de données en XML : XSLT

- **Pour aller plus loin**
  - ➢ **Conditions et boucles** :
    - **xsl:for-each** pour itérer sur les nœuds sélectionnés par l'**@select** et appliquer le modèle de contenu à chacun de ces nœuds
    - **xsl:sort** pour attribuer un ordre particulier aux nœuds sélectionnés (instruction enfant de xsl:apply-templates ou xsl:for-each). On trouve comme 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:template match="mon_element">
    <xsl:for-each select="sous_elements_a_traiter">
      [motifs à appliquer]
    </xsl:for-each>
</xsl:template>
```

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

Un langage de transformation de données en XML : XSLT

- **Pour aller plus loin**
  - ➢ **Conditions et boucles** :
    - **xsl:for-each-group** pour itérer sur un groupe de nœuds sélectionnés par l'**@select**, tandis que l'**@group-by** permet de créer des sous-groupes parmi les nœuds sélectionnés. La fonction XPath **current-grouping-key()** permet de retourner la valeur de la clé de regroupement de la boucle en cours.

```
<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>
```