

# Technique et chaîne de publication électronique avec XSLT

---

Jean-Damien Généro

2023, 9 janv. - 13 fev.

École nationale des chartes – M2 TNAH

## Contact

- [jean-damien.genero@cnrs.fr](mailto:jean-damien.genero@cnrs.fr)

# Objectifs

- XPath

- Naviguer dans un arbre XML
- Manipuler les principales fonctions XPath

- XSLT

- Manipuler les règles (*templates*) basiques
- Manipuler les conditions et les boucles

- Édition numérique

- Transformer un document XML en un autre document XML
- Transformer un document XML en un document HTML
- Transformer un document XML en un document  $\text{\LaTeX}$

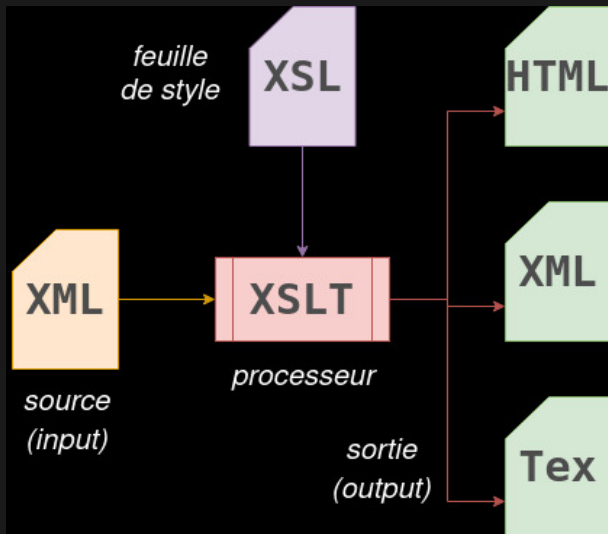
# Écosystème XML

---

# Écosystème XML/ Principes généraux

- **XML** → *a markup language that encodes a description of a doc's storage layout and logical structure;*
  - **XPath** → *use a path notation for navigating through the hierarchical structure of an XML doc;*
- **XSL** → *a language for expressing stylesheets;*
  - *stylesheet* → *express intentions about how a structured content [i. e. XML doc] should be presented;*
- **XSLT** → *a language designed for transforming XML documents into other XML documents.*

# Écosystème XML/ Schéma



# Écosystème XML/ Navigation XPath et de règles XSL

- **Navigation XPath :**
  - Doc XML = structuré, on parle d'« **arbre** » (*XML tree*);
  - **Itinéraire** vers une balise (« **parent** » → « **enfant** »);
  - Rédaction dans une **syntaxe propre**.
- **Règle XSL (*template*) :**
  - **Agir** sur une ou plusieurs balises (*transform*);
  - Exemples : **copier** balise + contenu, copier uniquement le contenu et l'**insérer** dans une nouvelle balise, **ajouter** ou **supprimer** un attribut, etc.;
  - Rédaction dans une **syntaxe XML** (langage à balises).
- **Utilisation avec d'autres langages**  
(*Python : lxml*).

# Écosystème XML/ Exemple d'un chemin XPath

- Comment accéder aux balises `<p>` avec XPath ?

```
<TEI>
  <teiHeader/>
  <text>
    <body>
      <p>title</p>
      <p>text</p>
    </body>
  </text>
</TEI>
```

- `<TEI>` → `<text>` → `<body>` → `<p>`
- Traduction XPath : `/TEI/text/body/p`
  - Simplification : `//body/p` (`/TEI/text/body/p`)



# Écosystème XML/ Exemple d'une règle XSL

- Comment appliquer une règle XSL aux balises `<p>`?
  - Une règle XSL est toujours contenue dans une balise `<xsl:template/>` possédant un `@match`.
- Indiquer le chemin XPath vers les `<p>` dans l'`@match`.

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

ou

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

# XPath

---

- **Nœud** (*node*) = un composant de l'arbre XML. Sept types, dont : **balise**, **attribut**, *text*, *namespace*.
- Écrire un chemin (*expression*) : succession de nœuds séparés par l'**opérateur** / ;
  - */TEI/text/body/p*
- Axe (direction) basique : **parent** → **enfant**

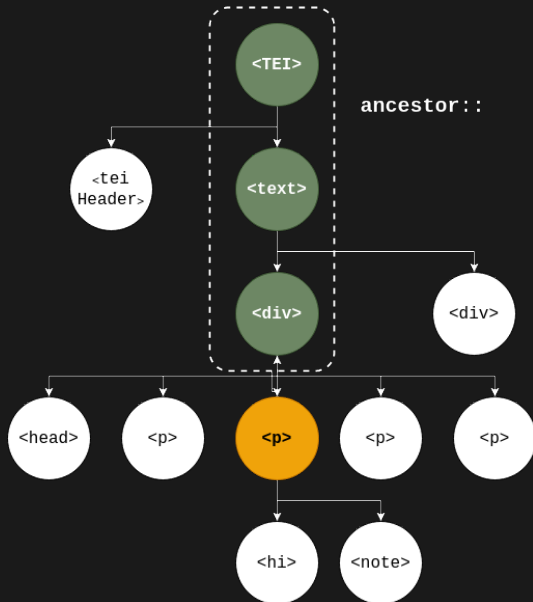
## XPath/ Exercice n°1. *Saint Julien l'hospitalier*, II.

- Donner le chemin de la racine vers le premier paragraphe.
- Donner le chemin de la racine vers *<author>* dans le *<sourceDesc>*.

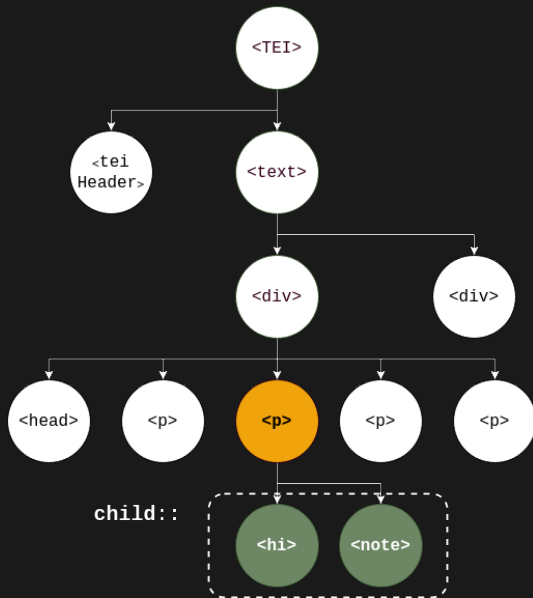
## XPath/ Axes de relation

- Un axe permet de qualifier la **relation** entre le **nœud de contexte** et un ou des **autre(s) nœud(s)**.
- 13 axes XPath, dont :
  - *parent::* → nœud immédiatement au-dessus;
  - *child::* → nœud immédiatement en-dessous;
  - *following-sibling::* → nœud(s) après celui de contexte et qui ne fait/ont pas partie de ses descendants;
  - *ancestor, descendant, preceding*, etc.

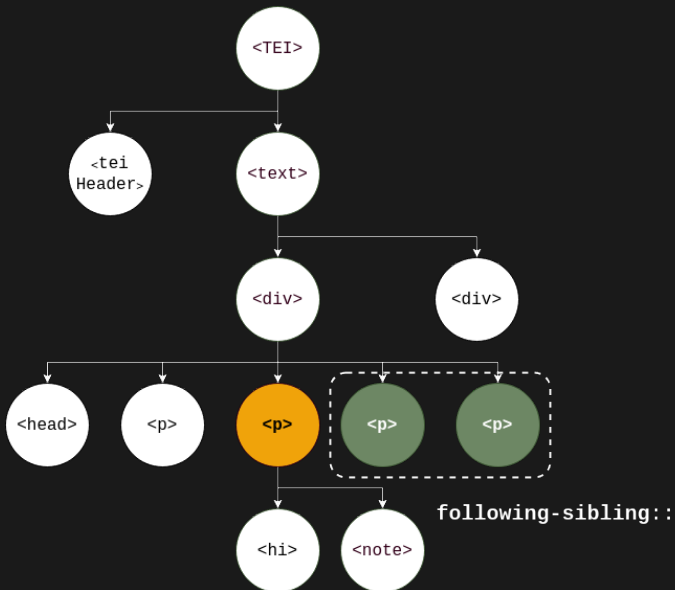
## XPath/ Axe ancestor::



## XPath/ Axe child::



## XPath/ Axe following-sibling::





## XPath/ Exercice n°2. *Saint Julien l'hospitalier*, II.

- Donner le chemin le plus court de la racine au titre du chapitre.

## XPath/ Prédicats : définition

- Prédicat (filtre) = **condition** qui doit être satisfaite par le nœud courant (existence d'un attribut, valeur d'un attribut, position d'une balise, etc.).
- Écrit entre crochets [ ].
- `//div[@n='2']/head` = le `<head>` de la `<div>` avec un `@n` de valeur 2.
- `//body/div[@n='2']/p[2] = ?`

## XPath/ Prédicats : exemples

- `//tag[position()=1]` ou `//tag[2]` → condition de position (`<tag>` n°2);
- `//tag[last()]` → idem (dernier `<tag>`);
- `//tag[@foo='bar']` → expression logique (`<tag>` avec un `@foo` dont la valeur est `bar`);
- `//tag[@foo='bar' and @baz='qux']`  
→ `<tag>` avec un `@foo` dont la valeur est `bar` et un `@baz` dont la valeur est `qux`.

## XPath/ Exercice n°3. *Saint Julien l'hospitalier*, II.

- Utiliser un prédicat pour sélectionner les `<title>` ayant pour texte *Trois Comtes*.
- Utiliser un prédicat pour sélectionner le **deuxième** ou le **cinquième paragraphe** du **premier chapitre** (deux solutions).

- Arbre XML, racine, nœuds (7);
- Chemin XPath, opérateur /;
- Axes de relation (13);
- Prédicats.

## XSLT : première approche

---

## XSLT/ Définition

- XSLT → un **langage de programmation déclaratif**;
- Permet de **transformer un doc XML** en un autre doc (*.xml*, *.html*, *.tex*, etc.);
- Transformation opérée par un **processeur XSLT**
  - **Construit** l'arbre output, **transforme** l'arbre input et **séréalise** le document output;
  - Le plus connu → **Saxon** (en ligne de commande);
  - Généralement **intégrés** à des logiciels (**Oxygen**) ou des navigateurs.

## XSLT/ La feuille de style

- Feuille de style XSL == un doc XML `.xsl`;
- Contient des instructions ou règles (*templates*);
- Élément racine : `<xsl:stylesheet>`
- Dans Oxygen → ouvrir un nouveau doc « XSLT Stylesheet ».
- Observer les attributs de l'élément racine.



## XSLT/ Élément racine : `<xsl:stylesheet>` (1/2)

Attributs présents de base dans Oxygen :

- `@xmlns:xsl` : espace de nom (*namespace*) XSL;
- `@xmlns:xs` : espace de nom XML;
- `@xmlns:tei` : espace de nom TEI;
- `!@exclude-result-prefixes` : liste des préfixes qui ne seront pas copiés dans l'output (*xs*, à ajouter : *tei*);
- `@version` : version de XSL (1, 2 ou 3).

## XSLT/ Élément racine : `<xsl:stylesheet>` (2/2)

Attributs à ajouter dans Oxygen :

- *@xmlns* : espace de nom de l'élément racine de l'output;
  - évite l'ajout de cet espace de nom sur chaque élément "matché" par une règle (contenu dans le *@match* de `<xsl:template>`).
- *@xpath-default-namespace* : espace de nom des chemins XPath.
  - évite de devoir écrire le préfix *tei:* devant chaque nœud d'une expression XPath.

## XSLT/ Élément `<xsl:output>`

- Instruction de premier niveau ;
- « contrôle les caractéristiques du document de sortie ».

`<xsl:output`

`method="xml | html | text"`

`indent="yes | no"`

`omit-xml-declaration="yes | no"`

`encoding="UTF-8"`

`/>`

## XSLT/ Écriture d'une règle : `<xsl:template>` (1/3)

- `<xsl:template>` → définir une règle;
- Possède un *`@match`* avec pour valeur un chemin XPath qui donne l'emplacement du nœud sur lequel sera appliquée la règle.
- Appliquer une règle vide en sélectionnant la racine :

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

## XSLT/ Écriture d'une règle : `<xsl:template>` (2/3)

- Appliquer la règle *copy-of* sur la racine :

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

- Comment copier le `<body>` dans l'output, mais sans le `<teiHeader>`?

## XSLT/ Écriture d'une règle : <xsl:template> (3/3)

- Comment obtenir cet arbre en output?

```
<div>
  <head>Trois comtes</head>
  <!-- copie de la div n°1 de l'input -->
  <div n="1">
    <head>I.</head>
    ...
  </div>
  <!-- copie de la div n°2 de l'input -->
  <div n="2">
    <head>II.</head>
    ...
  </div>
</div>
```