

# Technique et chaîne de publication électronique avec XSLT

---

Jean-Damien Généro

2025, 1<sup>er</sup> dec. - 2026, 20 janv.

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  $\text{\LaTeX}$
- Coder une chaîne de publication électronique utilisant XSLT avec Python

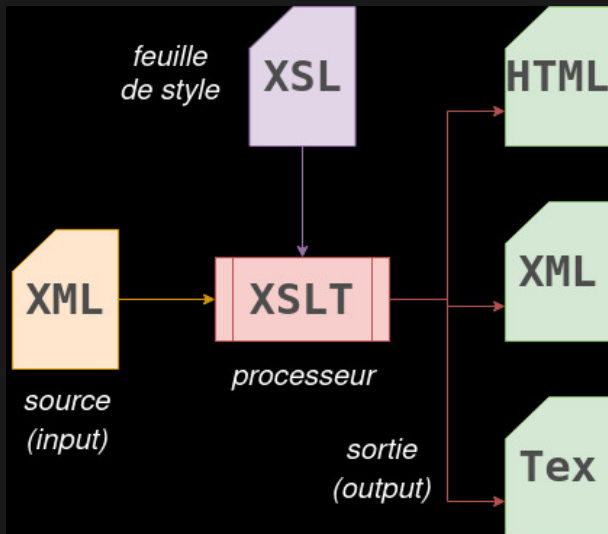
# Écosystème XML

---

# Écosystème XML/ Principes généraux

- **XML** → *langage de balisage qui encode une description de la mise en page et de la structure logique d'un document*
  - **XPath** → *langage de requête pour naviguer dans la structure hiérarchique d'un document XML à l'aide de chemins;*
- **XSL** → *spécifications pour écrire des feuilles de style (stylesheet);*
- **XSLT** → *un langage conçu pour transformer des documents XML en d'autres documents selon les spécifications XSL.*

# 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* : librairie *lxml*.

# Écosystème XML/ Exemple d'un chemin XPath dans un XML tree

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

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

- `<TEI>` → `<text>` → `<body>` → `<div>` → `<p>`
- Traduction XPath : `/TEI/text/body/div/p`
- Simplification : `//body/div/p` (`/TEI/text/body/div/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/div/p">  
    <xsl:copy-of select="."/>  
</xsl:template>
```

ou

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

# XPath

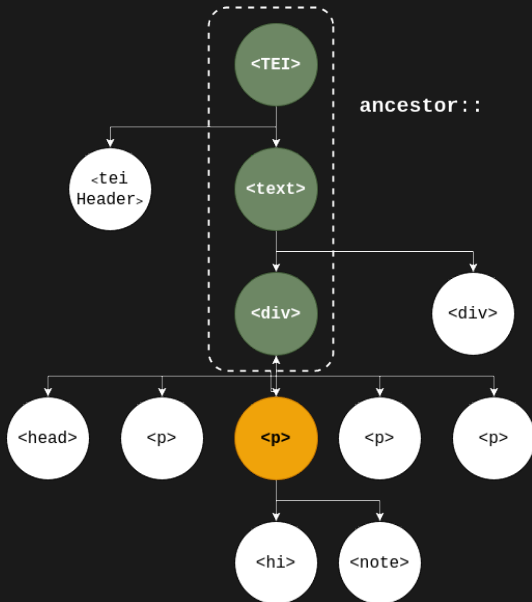
---

- **Nœud** (*node*) (7) = composant de l'arbre : *root*, *element*, *attribute*, *text*, *comment*, *namespace* et *processing instruction*.
- Deux rôles : *current node* (fixe, le premier du chemin) ou *context node* (variable, celui que XPath évalue à l'instant *t*);
- Écrire un chemin (*expression*) : succession de nœuds séparés par l'**opérateur** / ;
  - Tester dans Oxygen :  
*/TEI/text/body/div/p*

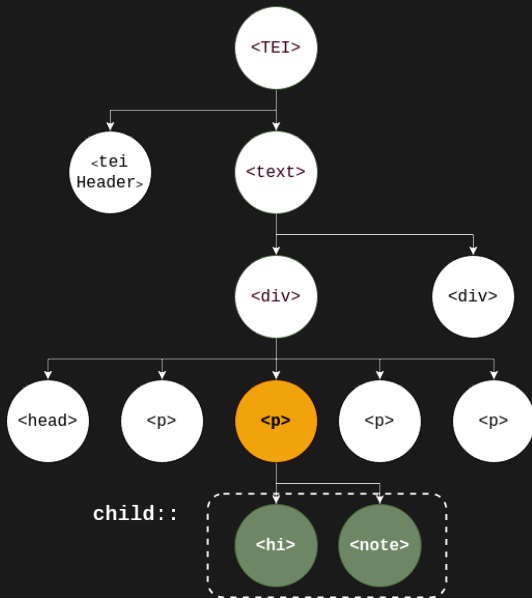
## XPath/ Axes de relation

- Un axe permet de qualifier la **relation** entre le **nœud de contexte** et un ou des **autre(s) nœud(s)**.
- Direction basique : **parent** → **enfant**.
- 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.

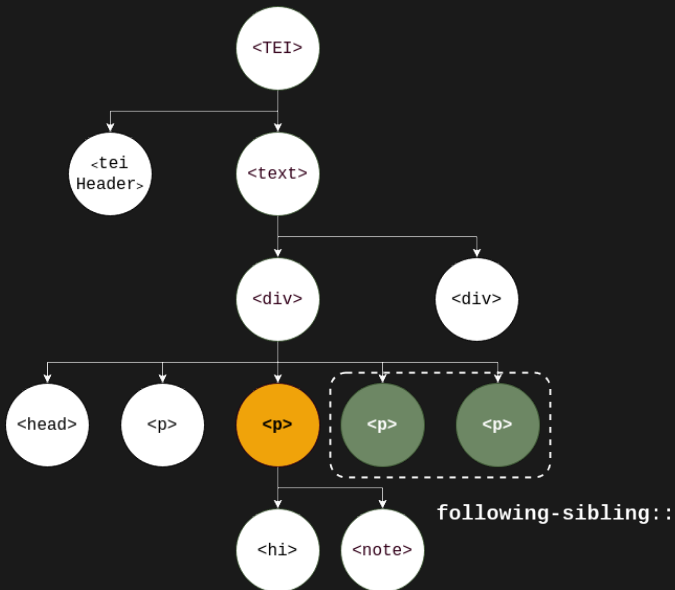
## XPath/ Axe ancestor::



## XPath/ Axe child::



## XPath/ Axe following-sibling::



## XPath/ Abréviations des axes

- 1<sup>er</sup> élément de l'arbre (*root node*) → */*  
(attention : différent de la racine <TEI>)
- *descendant-or-self::* → *//*
- *self::* → *.*
- *attribute::* → *@*
- Donc : vous pouvez écrire *//div/@n* au lieu de *//div/attribute::n*.



## XPath/ Prédicats : définition

- Prédicat (filtre) = **condition** qui doit être satisfaite par le nœud de contexte (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()=2]` ou `//tag[2]` → condition de position (<tag> n°2);
- `//tag[last()]` → dernier <tag>;
- `//tag[@foo='bar']` → expression logique (<tag> avec un *@foo* dont la valeur est *bar*);

## XPath/ Notions essentielles

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

- Donner le chemin de la racine vers `<msName>` dans le `<sourceDesc>`.
- Utiliser un prédicat pour sélectionner le `<idno>` avec le `@source="gallica"`.

## XSLT : élément racine et instructions de premier niveau

---

## XSLT/ Définition

- XSLT → un **langage de programmation**;
- 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 **intégré** à des logiciels (**Oxygen**) ou des librairies (**python : lxml**).

## 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>` (1/4)

Espaces de nom (*namespace*) et attributs présents de base dans Oxygen :

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



## XSLT/ Élément racine : `<xsl:stylesheet>` (2/4)

Attributs à remplacer dans Oxygen *pour une transformation vers XML-TEI* :

- remplacer `@xmlns:xs` par `@xmlns:tei` → espace de nom de l'élément racine de l'output.
  - évite l'ajout de `tei:` à chaque élément "matché" par une règle (`@match`).
- remplacer `xs` par `tei` dans `@exclude-result-prefixes`.

## XSLT/ Élément racine : `<xsl:stylesheet>` (3/4)

Attributs à ajouter dans Oxygen *pour une transformation vers XML-TEI* :

- *`@xpath-default-namespace`* : espace de nom des chemins XPath de la feuille de style (<http://www.tei-c.org/ns/1.0>).
  - évite de devoir écrire le préfixe *tei:* devant chaque nœud d'une expression XPath.
- *`@xmlns`* avec l'adresse de l'espace de nom TEI.
  - détermine l'espace de nom de l'ensemble du document de sortie.

## XSLT/ Élément racine : `<xsl:stylesheet>` (4/4)

- **attention** : TEI est un standard XML parmi d'autres ! Pour EAD, changer l'espace de nom TEI par celui d'EAD et le préfixe *tei:* par *ead:*.
- Atributs pour la transformation vers HTML ou LaTeX :
  - *@xmlns:xsl* (adresse de l'espace de nom XSLT)
  - *@xpath-default-namespace* (adresse de l'espace de nom TEI)
  - *@version* (version de XSL utilisée)

## XSLT/ Instruction de 1<sup>er</sup> niveau : <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>` (1/4)

- une règle peut être écrite de plusieurs manières;
- `<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.
- Un doc peut être composé d'une seule règle ou d'une succession de règle. Important → **cohérence des XPath.**

## XSLT/ Écriture d'une règle : `<xsl:template>` (2/4)

- 1. Appliquer une règle vide sur la racine :

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

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

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

## XSLT/ Écriture d'une règle : `<xsl:template>` (3/4)

- Effet d'une règle vide → la partie de l'arbre d'entrée sélectionnée n'est pas copiée dans l'arbre de sortie.
- Effet de `<xsl:copy-of/>` → copie à l'identique la balise matchée par le `@select` et ses nœuds enfants;
- Comment copier le `<text>` dans l'output, mais sans le `<teiHeader>`?

## XSLT/ Écriture d'une règle : <xsl:template> (4/4)

- Comment obtenir cet arbre en output?

```
<div>
  <head>Extrait du Journal de
    Jean Le Fèvre</head>
  <div n="1">
    <!-- copie de la div n°1 de l'input -->
  </div>
  <div n="2">
    <!-- copie de la div n°2 de l'input -->
  </div>
</div>
```



## XSLT : première approche. Notions essentielles

- Transformation et feuille de style XSL;
- Manipulation de XSL avec Oxygen;
- Élément racine : `<xsl:stylesheet>` et ses attributs (en-tête XSL);
- Instructions de premier niveau :
  - `<xsl:output>`;
  - `<xsl:template>` et son *@match*.