

Séance 4

Publier la TEI et Epidoc avec XSLT

TD Antiquités numériques - 2ème partie

Université Lumière Lyon 2, site Berges du Rhône

Mardi 9/12/2025 - 12h00-14h00



Visualiser un fichier XML dans un navigateur






Cascading Style Sheets (CSS)

- CSS est le langage utilisé pour coder la présentation et le style des pages web au format HTML (HyperText Markup Language).
- CSS peut aussi permettre de visualiser un fichier XML directement dans un navigateur.
- Il suffit d'ajouter l'instruction suivante au début du document (juste avant l'élément `<TEI>` par exemple (pour lui indiquer où trouver ce fichier)
`<?xml-stylesheet type="text/css" href="affichage-tei.css"?>`
-




Exercice 5 : consigne

1. Visualisez le corrigé de l'exercice 4 dans un navigateur à l'aide du fichier CSS `affichage-tei.css`
2. Comparez avec la version en ligne
<https://inslib.kcl.ac.uk/irt2009/IRT308.html>
3. Regardez le fichier CSS et observer les choix de présentation. Vous pouvez vous aider du document
`ressources/memo-css3.pdf`



Comprendre la notion de “transformation” : introduction à XSLT

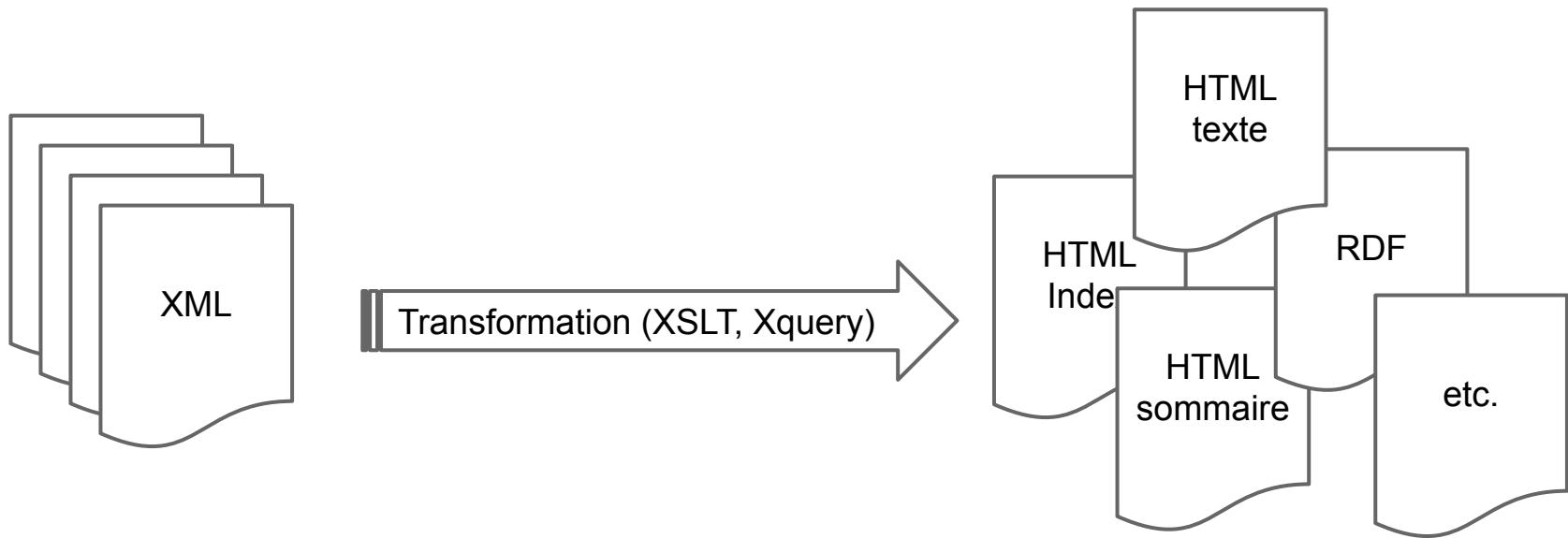




Pourquoi “transformer” l’arbre XML TEI ou EpiDoc ?

- XML ne “fait” rien, il se contente de décrire
- La publication de fichiers XML TEI ou EpiDoc requiert une étape de transformation permettant d’obtenir des formats de publication lisibles
 - HTML,
 - PDF
 - texte brut
 - ou bien d’autres arbres XML
- Les chaînes de publication basées sur XML établissent une séparation entre format d’encodage (données) et formats de publication (présentation des données) .

Single source publishing





“Single source publishing”

```
<choice>  
  <sic>relea</sic>  
  <corr cert="low">relatio</corr>  
  <corr cert="high">relicta</corr>  
</choice>
```

Transformation
(XSLT, Xquery)

relea

relicta

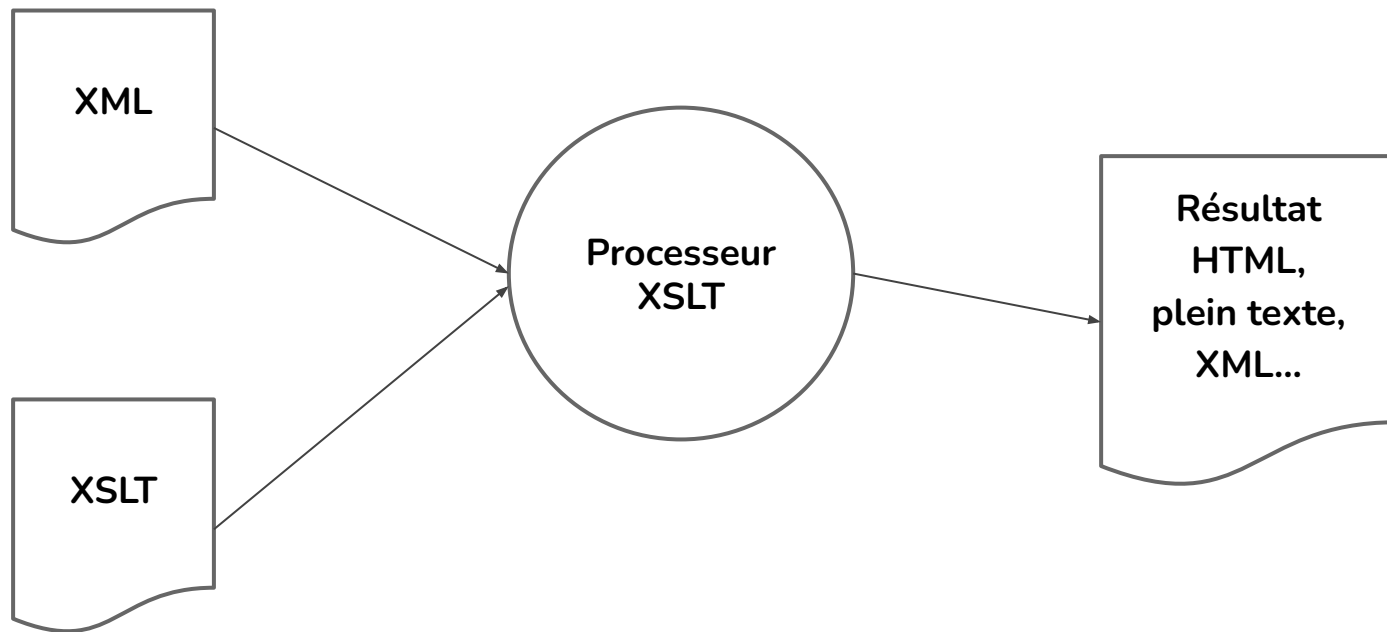
relatio



XSLT : un langage de transformation

- *eXtensible Stylesheet Language Transformations*
- Langage de transformation pour XML utilisé pour transformer la structure et le contenu d'un document XML en un autre format.
- XSLT est basé sur des règles de transformation (*templates*) qui définissent comment chaque élément XML doit être traité.

Principe de fonctionnement

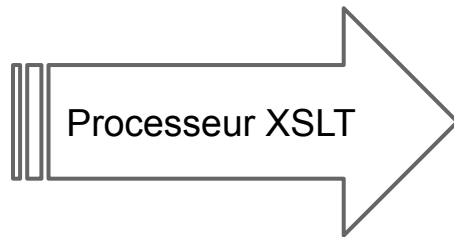


```
<xml>
<TEI>
(...)
<div>
<head>Atelier
pratique</head>
<p>L'encodage XML/TEI,
c'est facile !</p>
</div>

(...)
</TEI>
```

XML

document en entrée
(input)



```
<html>
(...)
<h1>Atelier pratique</h1>
<p>L'encodage XML/TEI,
c'est facile !</p>
(...)

</html>
```

XSLT

Traitement

HTML

Résultat (output)



Nouvelles installations dans VsCode

Installer les plugin VSCode :

- [HTML Preview](#)
- [XSL Transform](#)

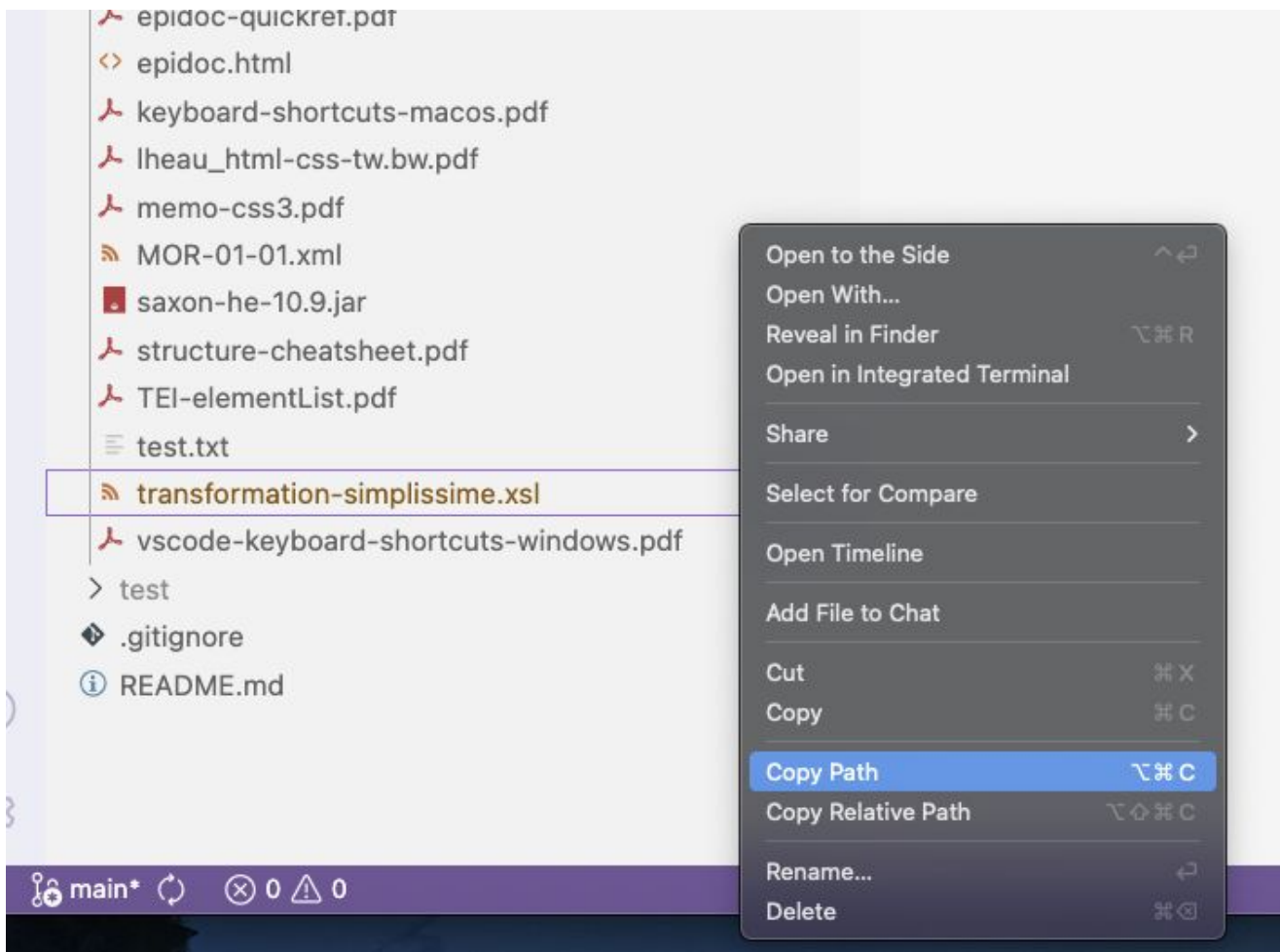
Installer également :

- [Java](#)
- Saxon (voir dossier ressources) : copiez le fichier `saxon-he-10.9.jar` dans un sous-répertoire “extension” de votre installation de java
 - Mac : /Library/Java/Extensions/saxon-he-10.9.jar

Configurer le plugin XSL transform :

- settings > XSL:Processor en indiquant le chemin de Java
- settings > XSL:stylesheet : indiquer le chemin de ressources/transformation-simplissime.xsl

Source : <https://francescaqiannetti.com/vscode-for-tei/>



Last synced: 0 secs ago

Xsl: Processor

Path to the Saxon XSLT processor (.jar)

/Library/Java/Extensions/saxon-he-10.9.jar

3

Xsl: Stylesheet

Path to active XSL / XSLT File

i-numerique-de-sources/ressources/transformation-simplissime.xsl



Lancer la tranformation

- Ouvrez le fichier XML “ressources/MOR-01-01.xml”
- Ouvrez la palette de commandes avec MAJ + CMD + P (mac) / MAJ + CTRL + P (pc)
- Tapez “set” et sélectionnez “Set XSL transform” puis ouvrez le fichier “transformation-simplissime.xsl” pour indiquer quel fichier de règles XSLT vous souhaitez utiliser (même si on l’a indiqué avant dans la configuration des “settings”)
- Tapez “transform” jusqu’à ce qu’apparaisse ‘Transform document xsl.transform’ dans la liste puis sélectionner avec “entrée”
- Enregistrez le résultat au format texte (et non XML) : par ex. “test.txt

MOR-01-01.xml X

ressources > MOR-01-01.xml

```
1 <?xml-model href="http://epidoc.stoa.org/schema/latest/tei-epidoc.rng" type="application/xml" schematypens="http://www.tei-c.org/ns/1.0" xml:id="R296">
2 <TEI xmlns="http://www.tei-c.org/ns/1.0" xml:id="R296">
3   <teiHeader>
4     <fileDesc>
5       <titleStmt>
6         <title>Stèle votive de Plumergat</title>
7         <title type="corpus">RIIG</title>
8       </titleStmt>
9       <editionStmt>
10         <edition>
11           <respSt
12             <re
13             <pe
14           </respSt
15           <respSt
16           <re
17           <pe
18         </respSt
19         <respSt
20         <re
21         <pe
22       </respSt
23       <respSt
24         <re
25         <re
26         <pe
27       </respSt
28     </editionSt
29     <publicatio
30       <author
31       <idno t
32       <idno t
33     <date when="2022">2022</date>
```

>set

Set XSL Stylesheet

recently used

Add Selection to Next Find Match

⌘ D other commands

Add Selection to Previous Find Match

Chat: Add Selection to Chat workbench.action.chat.attachSelection

Chat: Add Selection to Chat github.copilot.chat.attachSelection

Chat: Chat Settings

Chat: Configure Tool Sets...

Chat: Reset Tool Confirmations

File: Reset Active Editor Read-only in Session

File: Set Active Editor Read-only in Session

File: Set Active Editor Writeable in Session

Help: Settings Sync

Merge Editor (Dev): Open Selection In Temporary Merge Editor

Merge Editor: Reset Choice for 'Close with Conflicts'

Move Last Selection to Next Find Match

⌘ K ⌘ D

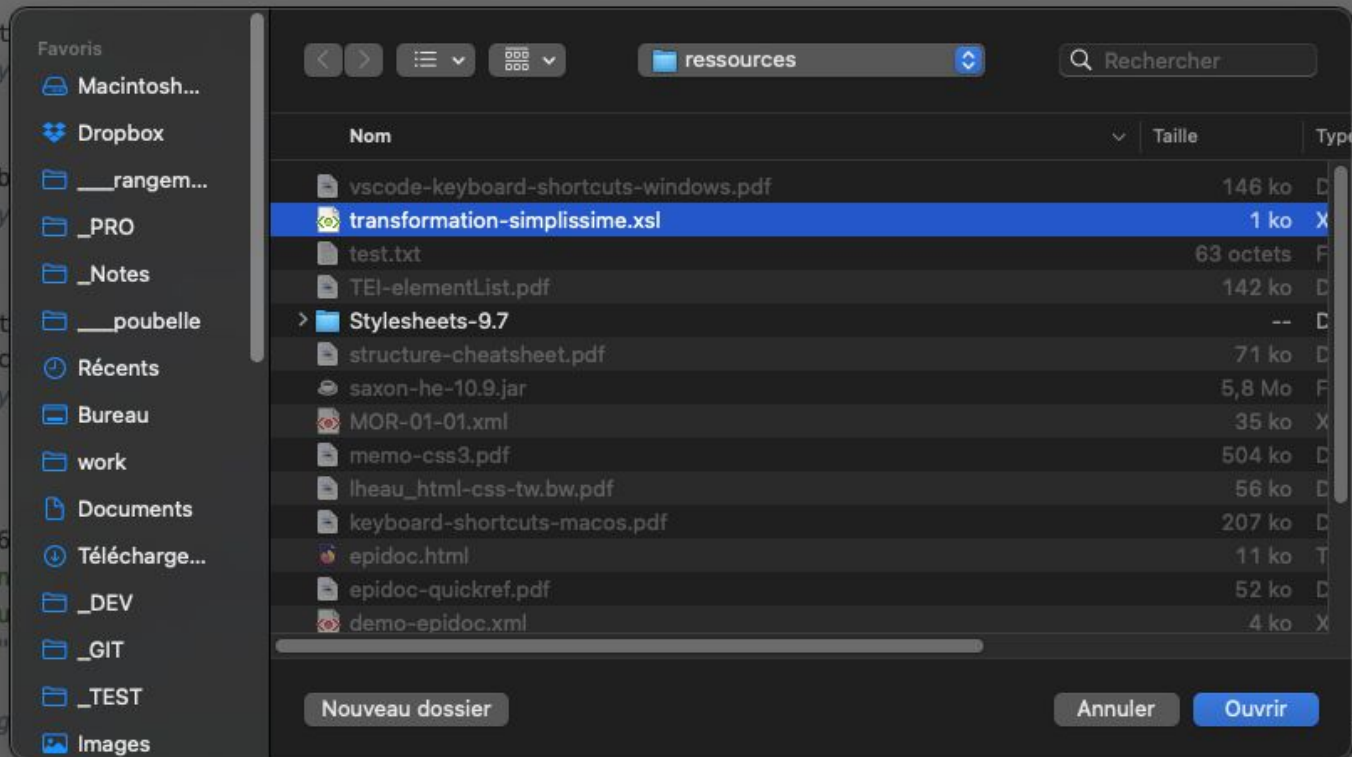
Move Last Selection to Previous Find Match

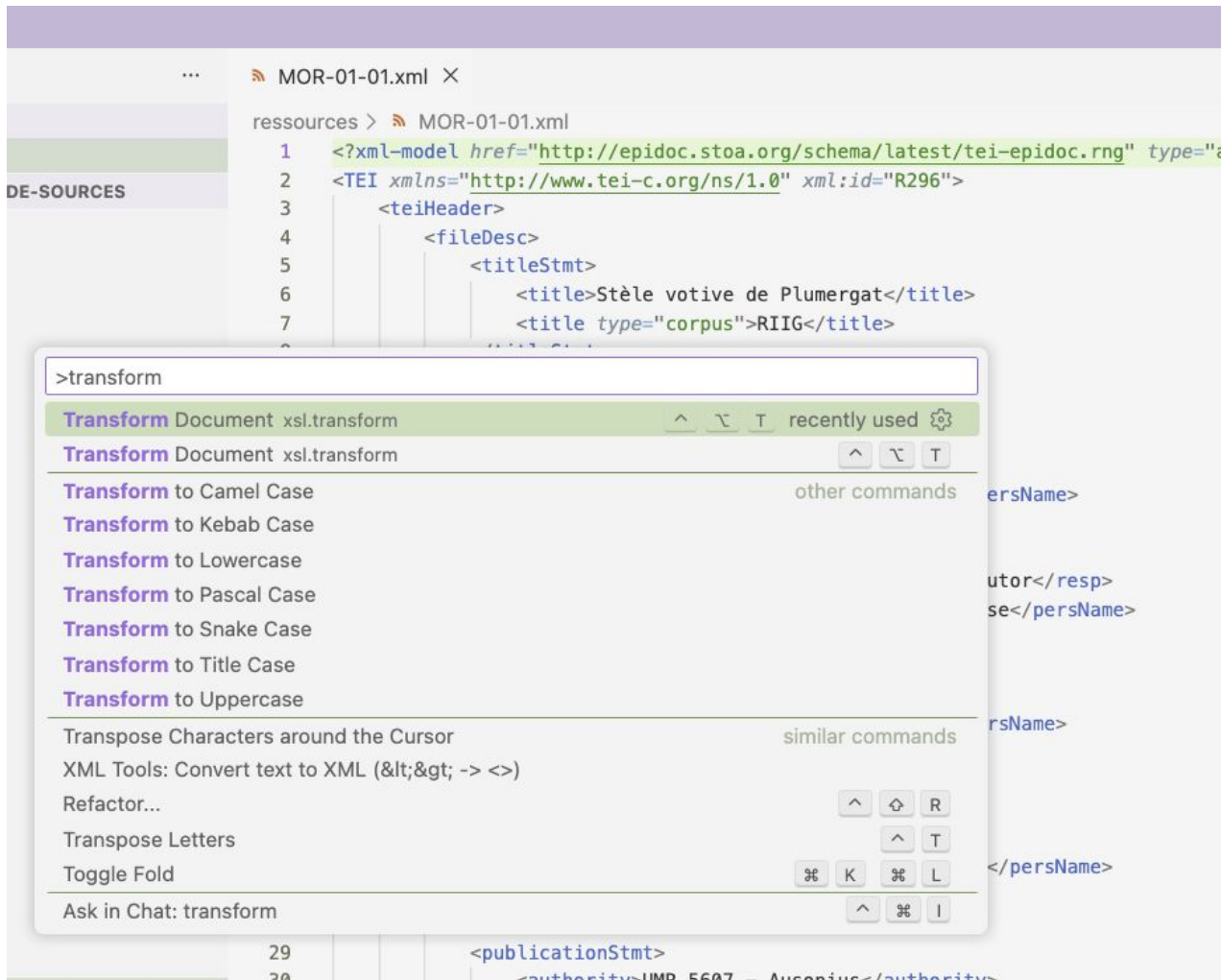
Move Selected Text Left

Move Selected Text Right

Preferences: Configure Language Specific Settings


```
>RIIG</edition>
nt>
sp>First editor</resp>
sName key="MLE">Michel Lejeune</persName>
mt>
nt>
sp>Project
sName key
mt>
nt>
sp>Contrib
sName key
mt>
nt>
sp>Metadat
sp>TEI enc
sName key
mt>
nt>
sStmt>
ty>UMR 56
pe="filen
pe="corpu
en="2022"
ility>
ence targ
bility>
onStmt>
```





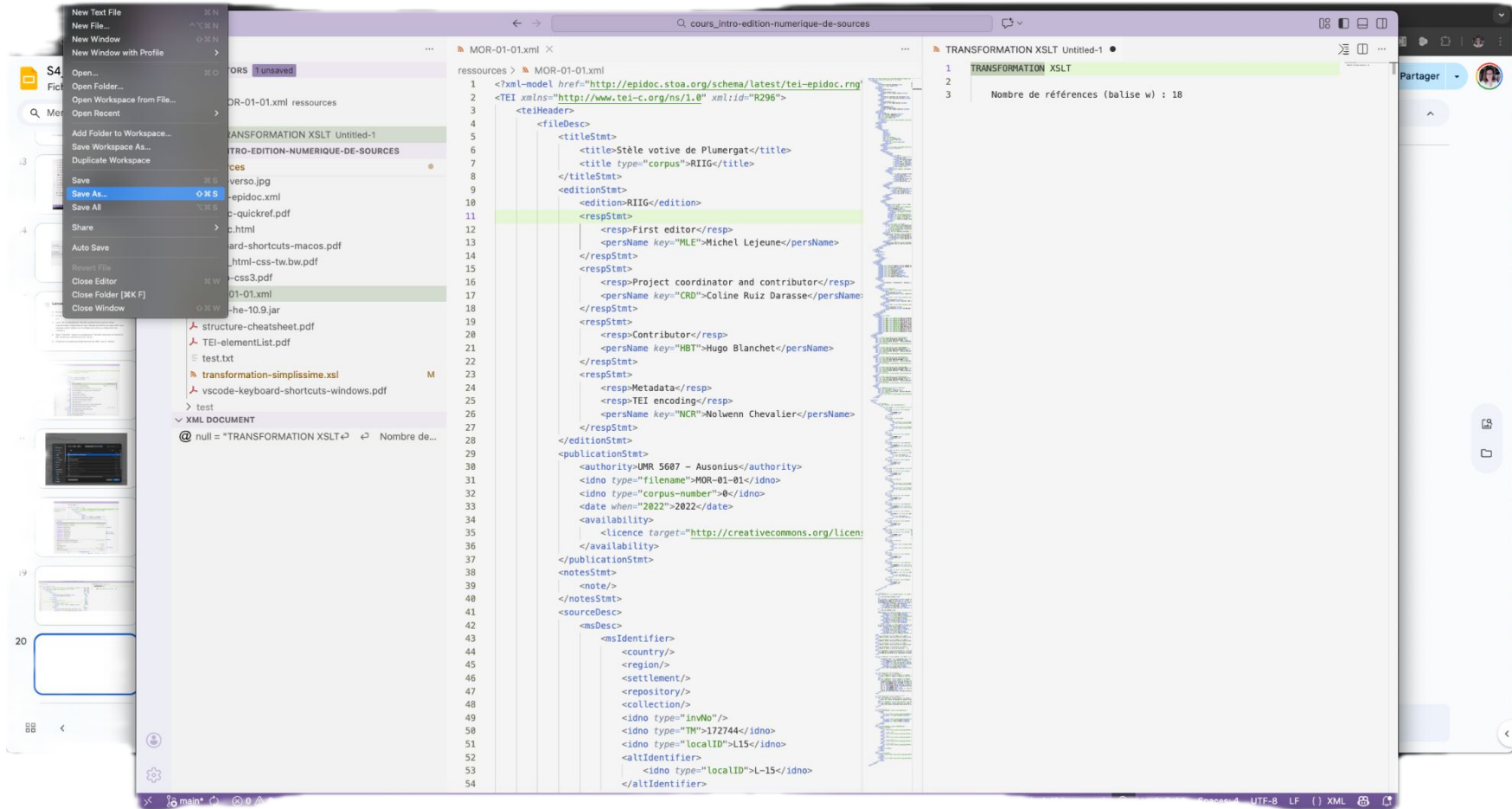
MOR-01-01.xml ✕

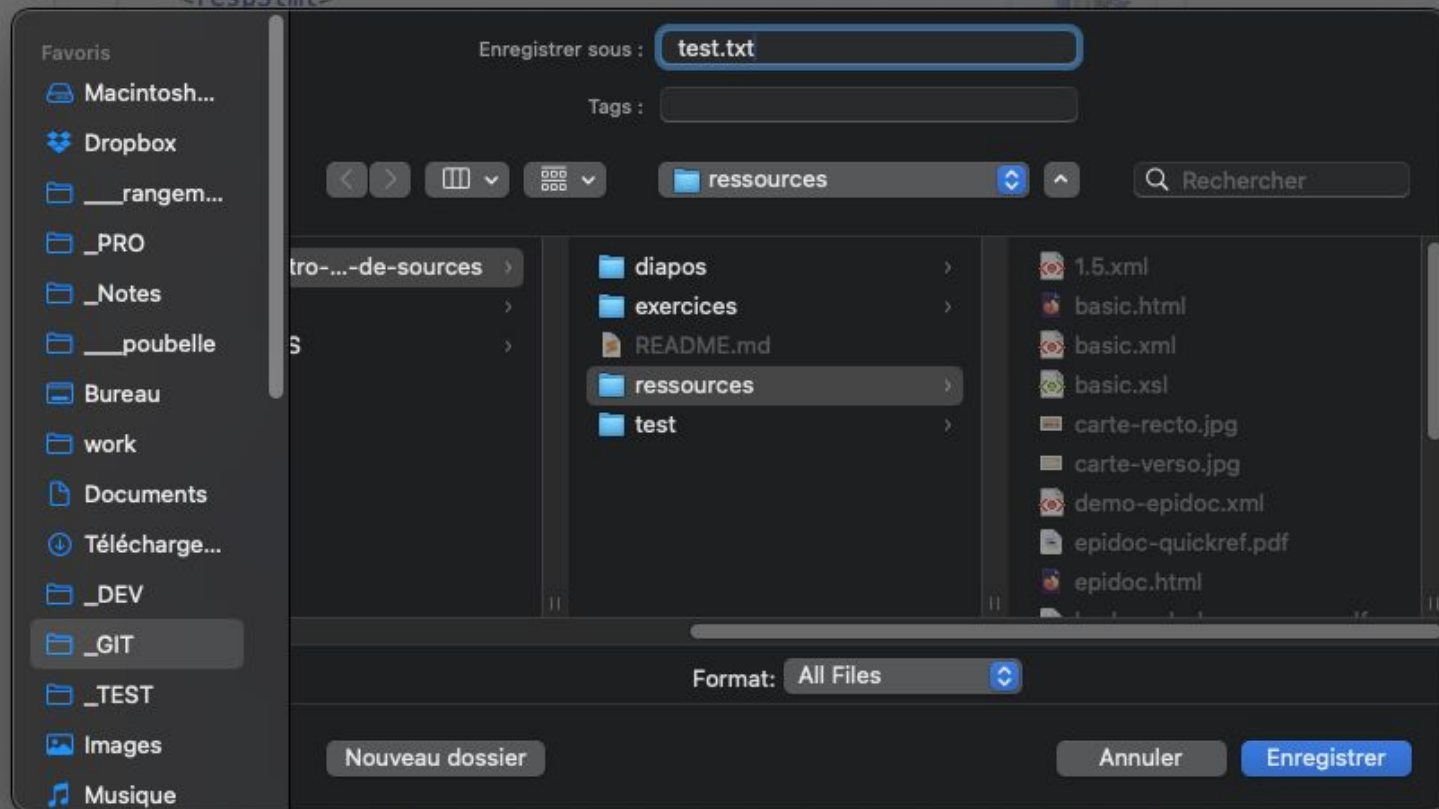
ressources > MOR-01-01.xml

```
1 <?xml-model href="http://epidoc.stoa.org/schema/latest/tei-epidoc.rng" type="application/xml" schematypens="http://www.w3.org/2001/XMLSchema-instance"/>
2 <TEI xmlns="http://www.tei-c.org/ns/1.0" xml:id="R296">
3   <teiHeader>
4     <fileDesc>
5       <titleStmt>
6         <title>Stèle votive de Plumergat</title>
7         <title type="corpus">RIIG</title>
8       </titleStmt>
9       <editionStmt>
10        <edition>RIIG</edition>
11        <respStmt>
12          <resp>First editor</resp>
13          <persName key="MLE">Michel Lejeune</persName>
14        </respStmt>
15        <respStmt>
16          <resp>Project coordinator and contributor</resp>
17          <persName key="CRD">Coline Ruiz Darasse</persName>
```

TRANSFORMATION XSLT Untitled-1 ●

```
1 TRANSFORMATION XSLT
2
3 Nombre de références (balise w) : 18
```







exercice 6 :

- Dupliquez “transformation-simplissime”
- Modifiez la règle XSLT pour compter cette fois le nombre de références bibliographiques (élément <bibl/>) se trouvant dans les blocs bibliographiques du fichier XML (les `div[@type='bibliography']`)...



Comment ça marche ?

- Le fichier XML en entrée est un arbre
- Le fichier XSLT contient une série de règles de transformation qui sont traitées “de bas en haut” en suivant la structure de l’arbre XML
- Un fichier XSLT est donc une série de “templates” (règles) XSLT non ordonnées (contrairement à des instructions de traitement que le programme devrait exécuter les unes après les autres). Ici l’ordre est donné par l’arbre XML. Le processeur XSLT descend dans l’arborescence et recherche les règles s’appliquant à chaque élément.
- Lorsqu’il n’y a pas de règle pour un élément parcouru, le processeur traite le noeud conformément aux règles par défaut (built-in). Typiquement : afficher le contenu textuel de l’élément.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.w3.org/1999/xhtml"
  xpath-default-namespace="http://www.tei-c.org/ns/1.0"
  exclude-result-prefixes="#all">

  <xsl:output method="xml"/>

  <xsl:template match="/">
    <!--(...)-->
  </xsl:template>

  <xsl:template match="head">
    <!-- (...)-->
  </xsl:template>

  <xsl:template match="title">
    <!--(...)-->
  </xsl:template>

  <xsl:template match="note">
    <!-- (...)-->
  </xsl:template>

</xsl:stylesheet>
```



```
<xsl:template match="/">
  <!-- Ecrit un titre dans le fichier suivi d'un saut de ligne -->
  <xsl:text>TRANSFORMATION XSLT

  </xsl:text>
  <!-- Dénombrer les mots (balises w) en utilisant XPath -->

  <xsl:text>Nombre de mots (balise w) : </xsl:text>
  <xsl:value-of select="count(//w)"/>
</xsl:template>
```



XSL:templates

- un template = 1 règle de transformation.
- Chaque template traite un morceau de l'arbre XML.
- Le template est exécuté lorsque le processeur XSLT atteint le noeud concerné dans son parcours de l'arbre du fichier XML en entrée.
- Le template traite ce qui est indiqué dans l'attribut **@match** et qui est exprimé en **XPATH**.



Différencier les éléments dans la feuille de style XSL

- On a affaire à **3 documents différents** :
 - le fichier XML (.xml)
 - le fichier XSLT contenant les templates / règles de transformation (.xsl)
 - le fichier résultat (par ex .txt ou .html)
- Dans le fichier XSL :
 - les noms d'éléments présents dans les attributs @match ou @select : correspondent aux éléments du fichier XML d'entrée
 - les éléments précédés de l'espace de nom "xsl:" : les éléments de transformation appartenant au "programme XSLT"
 - les éléments sans préfixe : correspondent aux éléments du format de sortie (si html)

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xsl:stylesheet version="2.0"
```

```
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
  xmlns="http://www.w3.org/1999/xhtml"
```

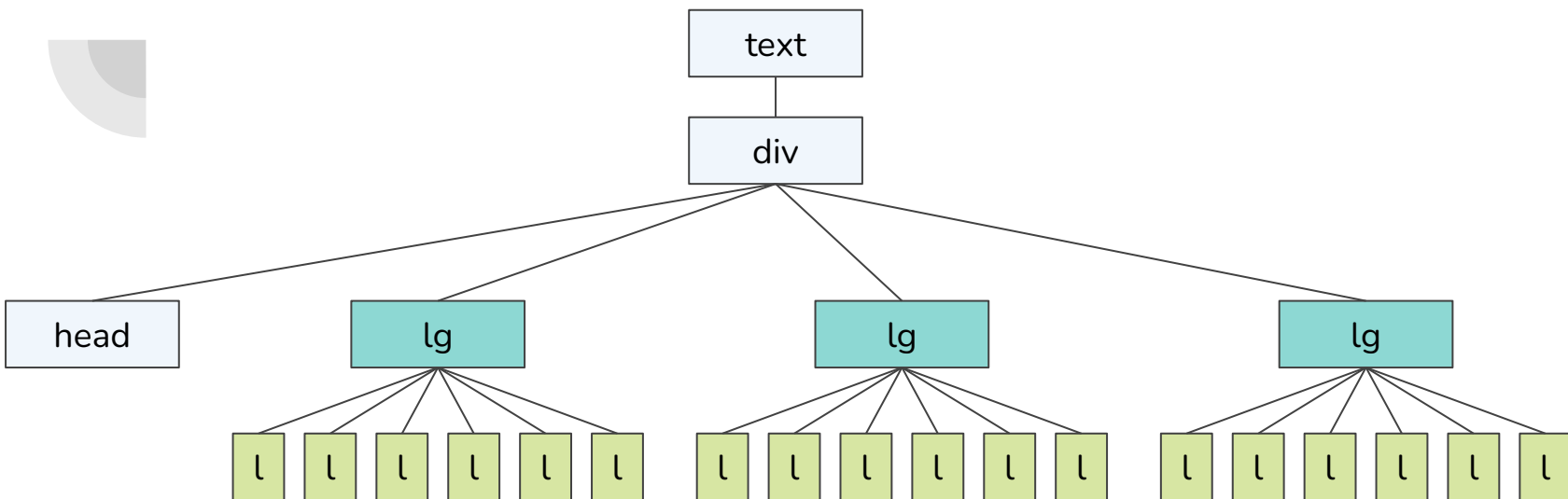
```
  xpath-default-namespace="http://www.tei-c.org/ns/1.0"
```

```
  exclude-result-prefixes="#all" >
```



Une transformation pas à pas

- Dossier ressources :
 - Transformer basic.xml avec basic.xsl et enregistrer dans basic.html




```
<xsl:template match="head">  
  <h2><xsl:apply-templates/></h2>  
</xsl:template>
```




Pour aller plus loin

- Cours vidéo XSLT de Syd Bauman et Martin Holmes sur la plateforme “Dariah Teach”
 - <https://teach.dariah.eu/course/view.php?id=32>



Utilisation des feuilles de style EpiDoc





Les feuilles de style EpiDoc

- Mises à disposition gratuitement sur <https://github.com/EpiDoc/Stylesheets>
- Maintenues par la communauté (une version par an environ)
- Paramétrables
 - via un fichier de paramètre ou des commandes directement passées au processeur XSLT
 - par exemple : intervalle de numérotation, type d'apparat critique, édition diplomatique ou interprétative...
- Paramètre ``$leiden-style`` : différentes conventions d'affichage du texte (légères variations dans l'application des conventions de Leiden) :
 - ex 'panciera' (default), ila, london, petrae, seg, etc.
- Paramètre ``$edn-structure`` : choix de mise en page et d'utilisation sélective du `teiHeader`



Installation

1. Téléchargez le jeu de fichiers XSLT depuis github (<https://github.com/EpiDoc/Stylesheets>) et placez le fichier .zip de la version 9.7 dans le dossier “ressources”
2. Dé-compressez le fichier .zip
 - Le point d’entrée pour une transformation vers html est “**start-edition.xsl**”
 - Le fichier de paramètres est “**global-parameters.xsl**”




Exercice : transformer un fichier EpiDoc à l'aide du jeu de feuilles de styles

- Ouvrez un fichier XML (par ex. le corrigé de l'exercice 4)
- Ouvrez la palette de VSCode, cherchez la commande “Set XSL Stylesheet”
- Indiquez “Stylesheets9.7/start-edition.xsl”
- Utilisez la commande “Transform document xsl.transform”
- Utilisez la commande “Format document” pour indenter le balisage html
- Enregistrez le résultat au format html “**exo-4.html**” par exemple
- Utilisez la commande “HTML: Open Preview” pour voir le résultat de la transformation dans un navigateur



EFES : un système de publication intégré pour EpiDoc



Welcome to EFES!

Now that you have EFES up and running, it's time to start building your project. Start with one of the options listed below and go from there.

Add your files

Put your EpiDoc XML files in `webapps/ROOT/content/xml/epidoc` and go to the [admin](#) and index them for searching. Then see how they are displayed ([Inscriptions](#)).

Customise the templates

Change the look of this site by modifying the templates in `webapps/ROOT/assets/templates`. The template for this page is `home.xml`, and it builds on the base template `base.xml`.

Read the documentation

The documentation and User Guide for EFES can be found on the project's [GitHub Wiki pages](#). Documentation for Kiln, the platform that EFES is based on, can be found both [online](#), and in source form, in the docs directory of your installation. It explains how you can modify everything about your project.



EFES

- *EpiDoc Front-End Services (EFES°*
- Système de publication intégré pour EpiDoc
- Intègre les feuilles de styles XSLT que l'on vient de voir
- Intègre des index et un tripplestore
- Accessible gratuitement sur github :
<https://github.com/EpiDoc/EFES>
- Installe un serveur web en local sur votre machine
 - Lancement en ligne de commande (cf. suivre la documentation)