Séance 4
Publier la TEI et Epidoc

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

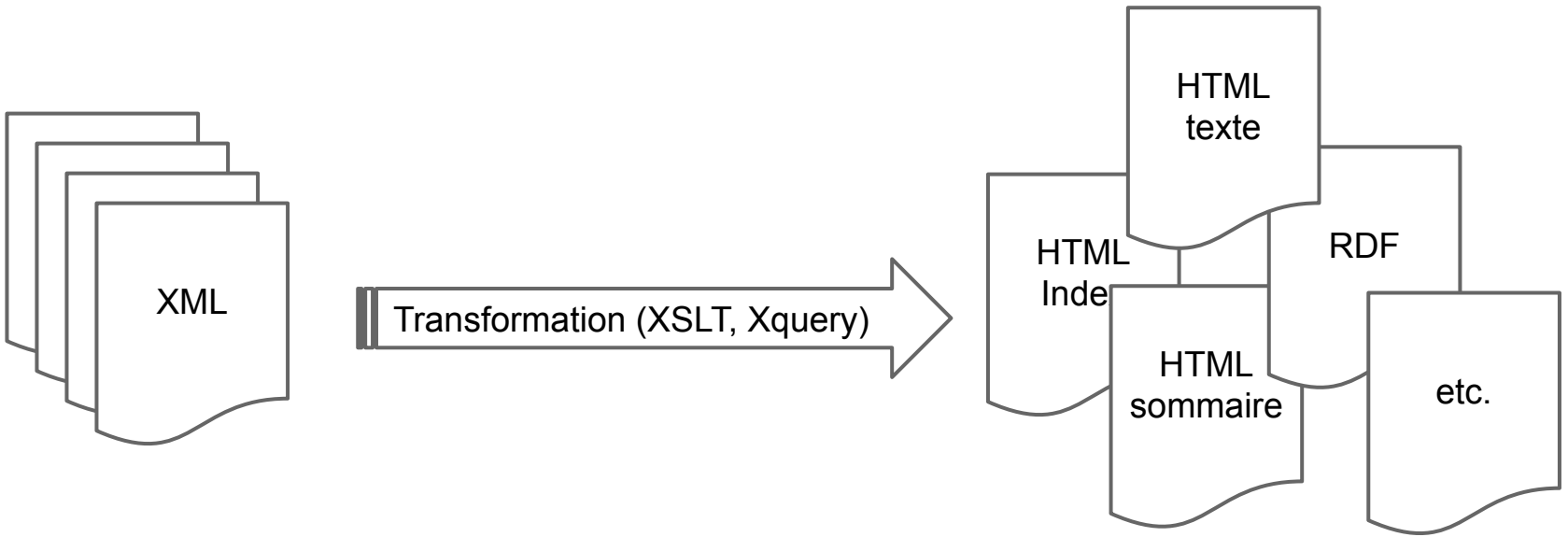
1. Visualisez le corrigé de l'exercice 4 dans un navigateur à l'aide du fichier CSS `affichage-tei.css`
2. Comparez avec la version en ligne https://inslib.kcl.ac.uk/irt2009/IRT308.html
3. Regardez le fichier CSS et observer les choix de présentation. Vous pouvez vous aider du document `ressources/memo-css3.pdf`

Comprendre la notion de “transformation” : introduction à XSLT

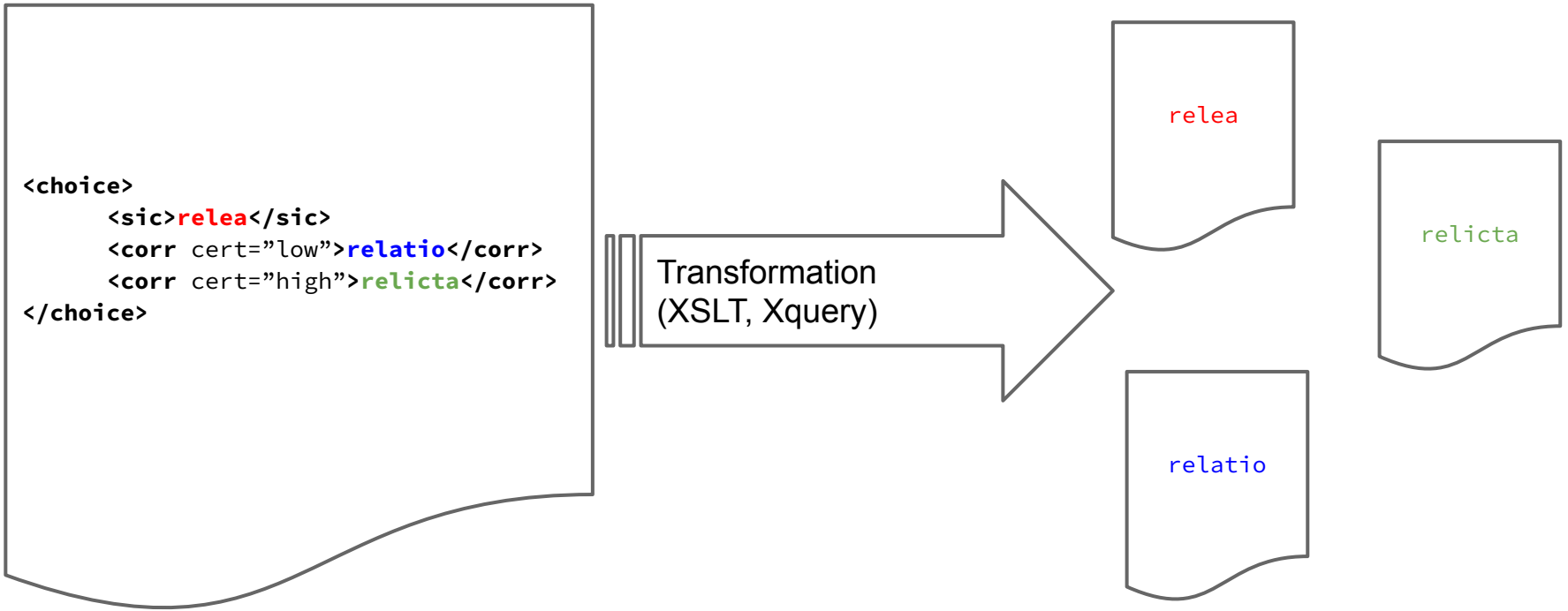
Pourquoi “transformer” l’arbre XML TEI ou EpiDoc ?

- 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



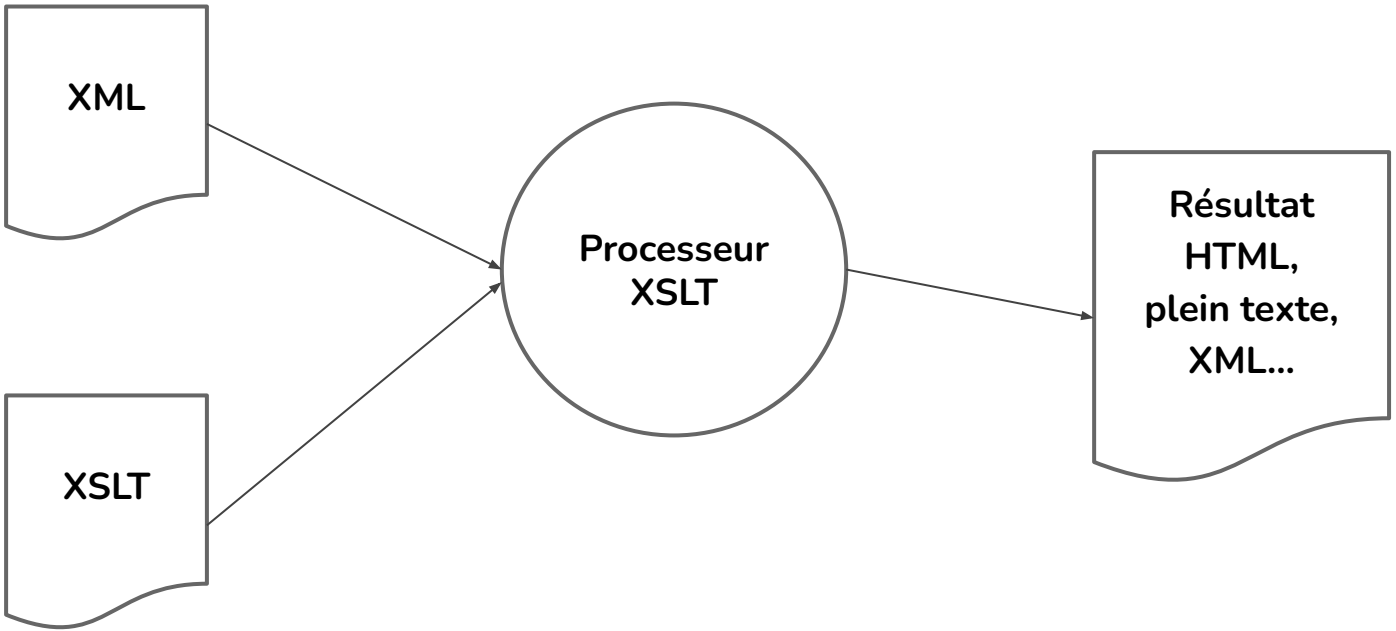
“Single source publishing”

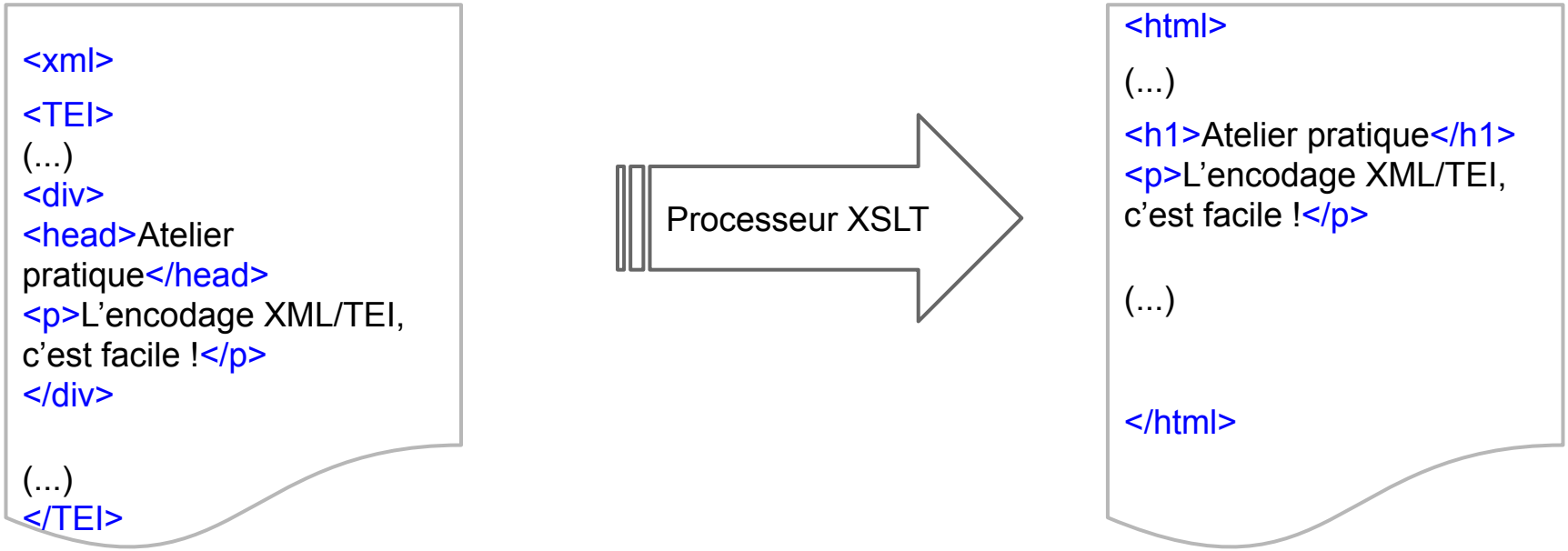


XSLT : un langage de transformation

- *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	XSLT	HTML
document en entrée (input)	Traitement	Résultat (output)

Nouvelles installations dans VsCode

Installer les plugin VSCode :

- HTML Preview
- XSL Transform

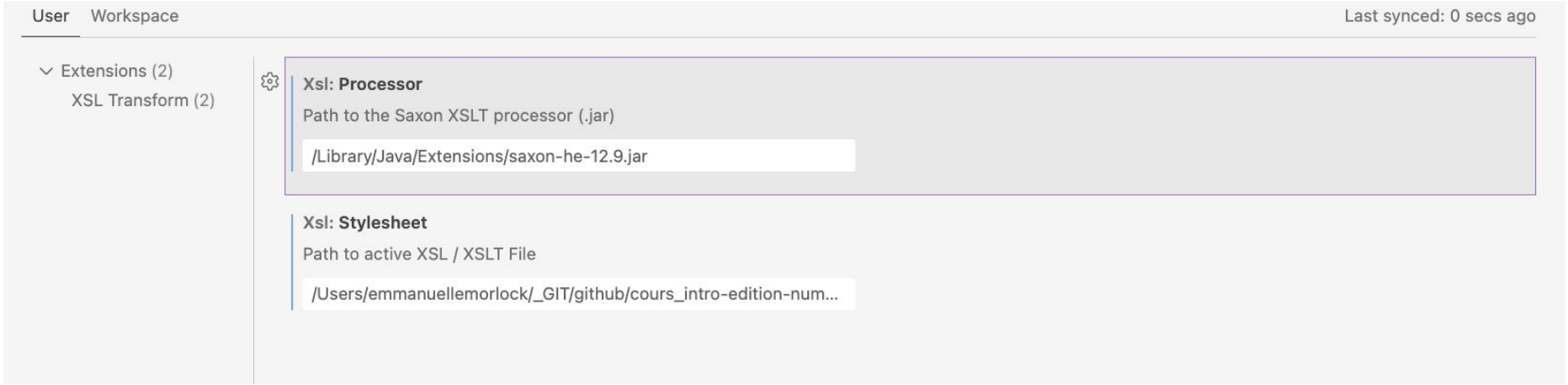
Installer également :

- 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://francescagiannetti.com/vscode-for-tei/


User
Workspace
Last synced: 0 secs ago
Extensions (2)
XSL Transform (2)
Xsl: Processor
Path to the Saxon XSLT processor (.jar)
/Library/Java/Extensions/saxon-he-12.9.jar
Xsl: Stylesheet
Path to active XSL / XSLT File
/Users/emmanuellemorlock/_GIT/github/cours_intro-edition-num...

Lancer la tranformation

- Ouvrez le fichier XML “ressources/MOR-01-01.xml”
- Ouvrez la palette de commandes avec MAJ + CMD + P (mac)
- 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

exercice 6 :

- Dupliquez “transformation-simplissime”
- Comptez le nombre de références bibliographiques dans la 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
    - série de “templates” (règles) XSLT

```xml
<?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'abre 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**

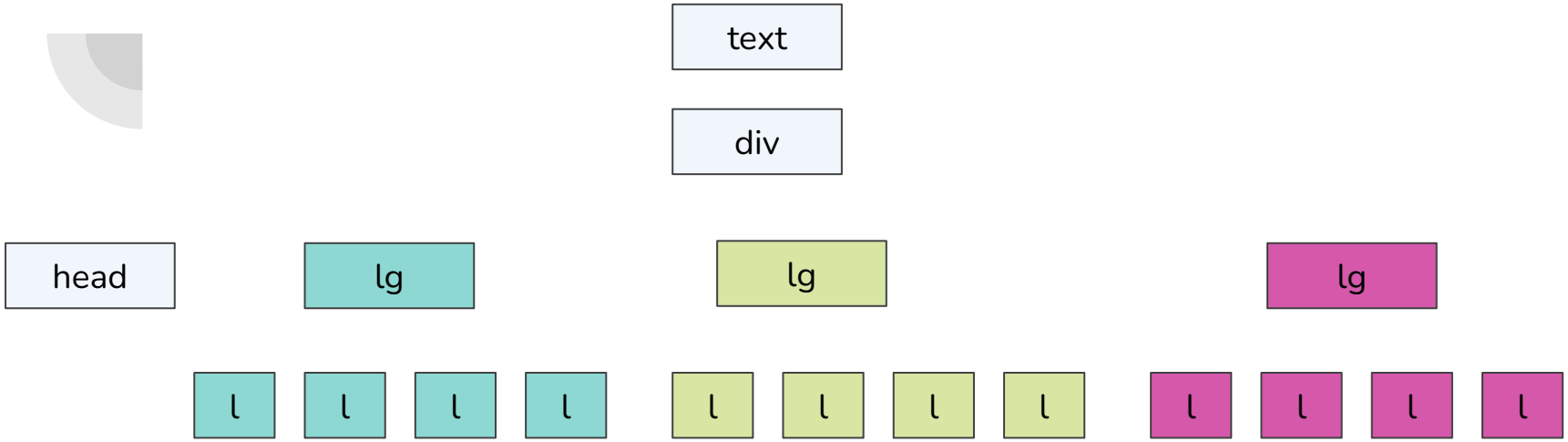
Différencier les éléments dans la feuille de style XSL

- 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

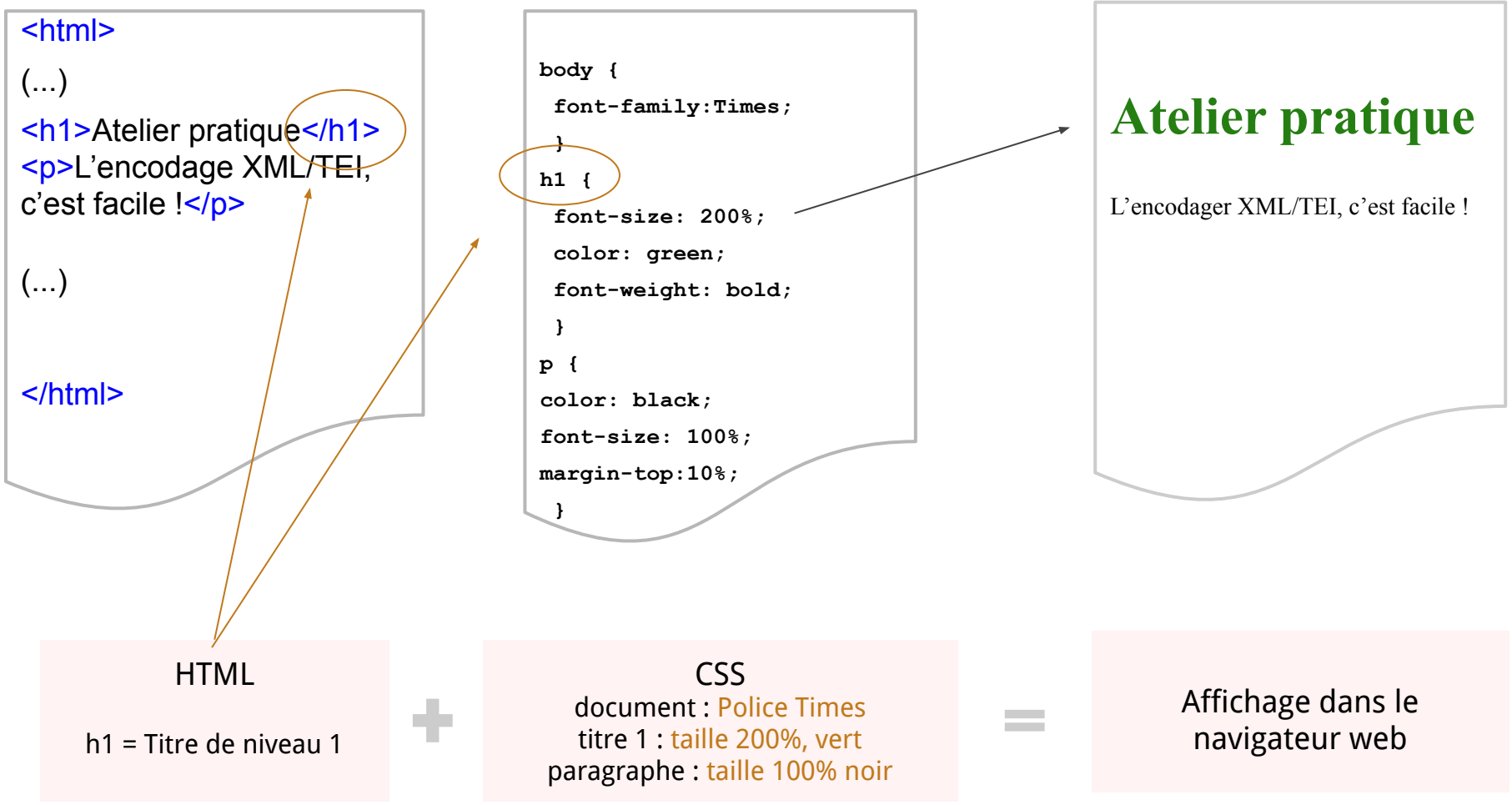
- Dossier ressources :
  - Transformer basic.xml avec basic.xsl et enregistrer dans basic.html


text
div
head
lg
lg
lg
l
l
l
l
l
l
l
l
l
l
l
l

```
<text>
   <body>
      <div>
         <head>Chanson d'automne</head>
         <lg>
            <l> Les sanglots longs</l>
            <l>Des violons</l>
            <l>De l’automne</l>
            <l>Blessent mon coeur</l>
            <l>D’une langueur</l>
            <l>Monotone.</l>
         </lg>
         <lg>
            <l>Tout suffocant</l>
            <l>Et blême, quand</l>
            <l>Sonne l’heure,</l>
            <l>Je me souviens</l>
            <l>Des jours anciens</l>
            <l>Et je pleure</l>
         </lg>

         <lg>
            <l>Et je m’en vais</l>
            <l>Au vent mauvais</l>
            <l>Qui m’emporte</l>
            <l>Deçà, delà,</l>
            <l>Pareil à la</l>
            <l>Feuille morte.</l>
         </lg>
      </div>
   </body>
</text>
```

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


<html>
(...)
<h1>Atelier pratique</h1>
L'encodage XML/TEI, c'est facile !
(...)
</html>
body {
font-family:Times;
}
h1 {
font-size: 200%;
color: green;
font-weight: bold;
}
p {
color: black;
font-size: 100%;
margin-top:10%;
}
Atelier pratique
L'encodager XML/TEI, c'est facile !
HTML
h1 = Titre de niveau 1
+
CSS
document : Police Times
titre 1 : taille 200%, vert
paragraphe : taille 100% noir
=
Affichage dans le navigateur web

Pour aller plus loin

Cours 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 `**$leiden-style**` : différentes conventions d'affichage du texte (légères variations dans l'application des conventions de Leiden) :
    - ex 'panciera' (defaut), ila, london, petrae, seg, etc.
- Paramètre `**$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

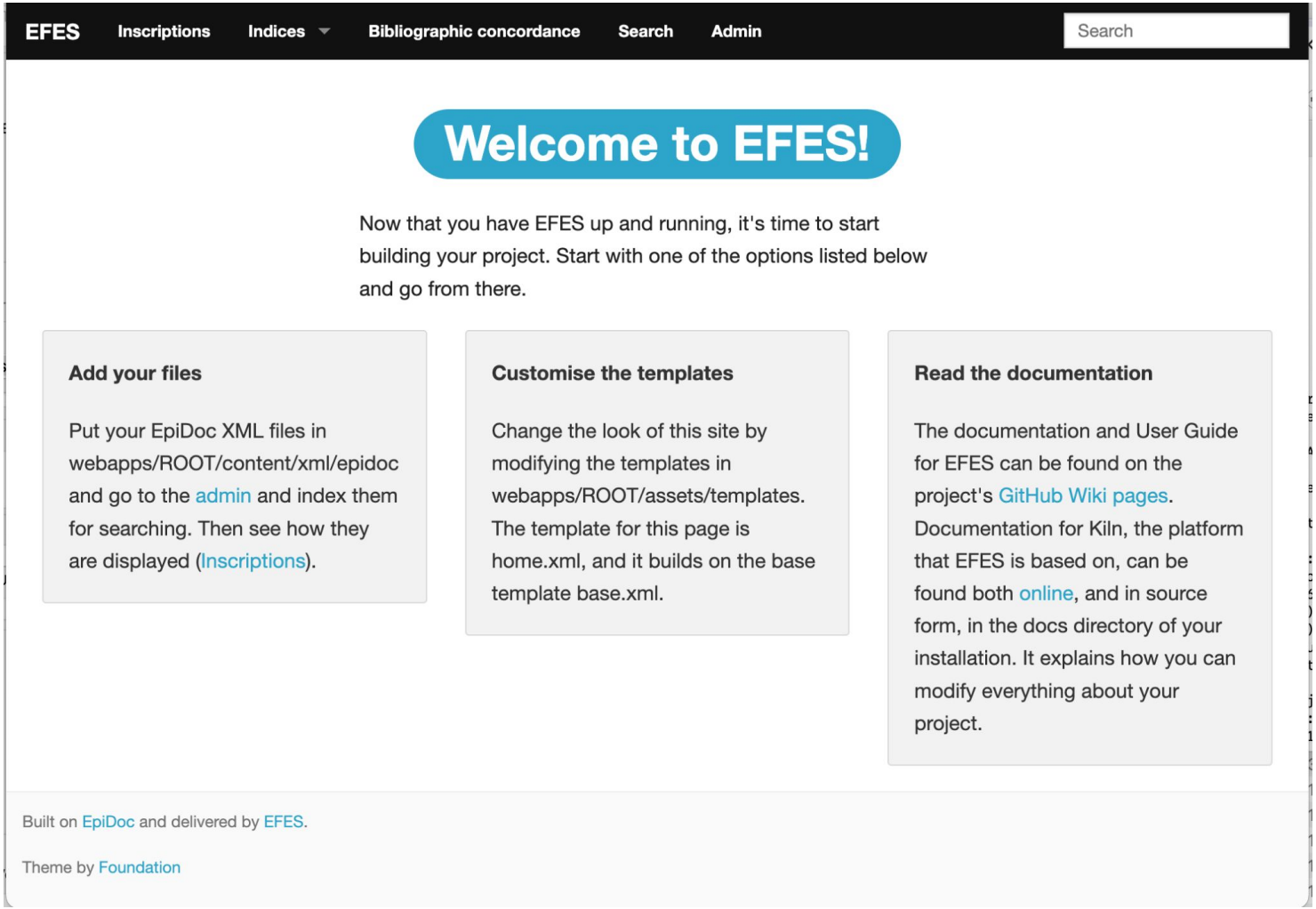
Systèmes de publication

Systèmes de publication : statique vs dynamique

- Statique :
  - Génération des pages HTML en amont
  - Contenu fixe, chaque page est un fichier HTML
  - Simple à déployer : ex. Github pages
  - Générateurs de sites statiques : Jekyll, Hugo, Eleventy…
  - Moins gourmant en ressources, moins vulnérable aux attaques, rapide

- Dynamique :
  - Génération des pages à la volée, en fonction de choix de l'utilisateur
  - Les données sont stockées dans une base de données et insérées dans des gabarits de pages HTML à partir de requêtes
  - Génération des pages envoyées au navigateur côté serveur ou côté client
  - Le contenu peut donc varier selon l'interaction qui produit des requêtes
  - PHP, Python avec Flask/Django, XQuery avec eXist-db ou BaseX…
  - Systèmes de publication intégrés : EFES, TEI Publisher

EFES


EFES
Inscriptions
Indices
Bibliographic concordance
Search
Admin
Search
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.
Built on EpiDoc and delivered by EFES.
Theme by Foundation

EFES

- EpiDoc Front-End Services
- Système de publication intégré pour EpiDoc
- Intègre les feuilles de styles XSLT que l'on vient de voir
- Intègre un des index et un tripplestore
- Accessible gratuitement sur github : https://github.com/EpiDoc/EFES

TEI Publisher


News
Features
Quick Start
Vision
Services
Documentation
FAQ
Contact
Source
tei
Publisher
The Instant Publishing Toolbox
Try it
Featured Demos

Qu’est-ce que TEI Publisher ?

Un environnement de conception et de mise au point d’applications web pour des éditions de textes.

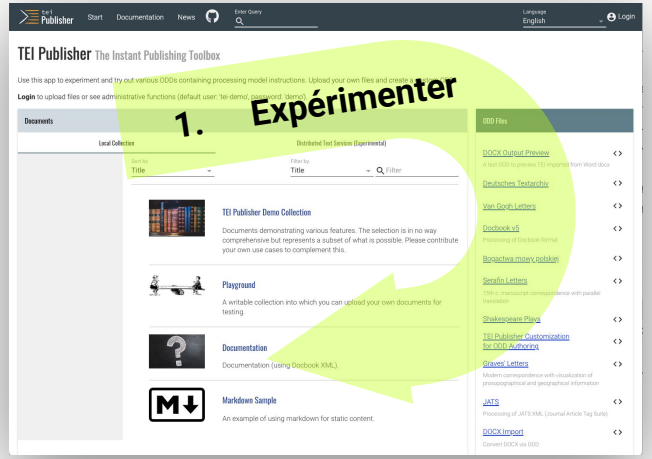
- **Basé sur des normes et des standards.**
  TEI, XQuery, Web Components, OpenAPI…
- **Open Source** (eXist Solutions).
- Sur **eXist-db** (V5 minimum) pour fonctionner.
- Conçu pour la **TEI** / utilisable avec **Docbook**, **JATS** et **Markdown** / extensible à tout vocabulaire **XML**
- Une **communauté d’utilisateurs** en progression constante.

3 Briques principales

- **Environnement de conception et de développement**
  TEI Publisher App
- **Moteur de transformations XQuery**
  TEI Publisher Lib
- **Collection de composants d’interface web**
  PB Components

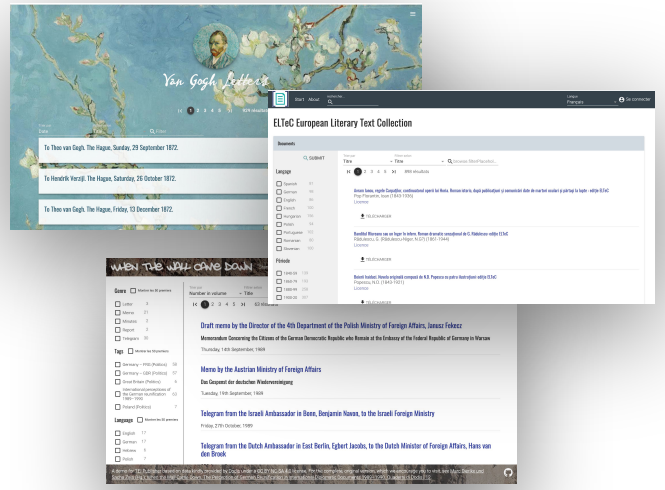
Créer des transformations en TEI et générer des applications autonomes

TEI Publisher App



2. Générer

Applications autonomes



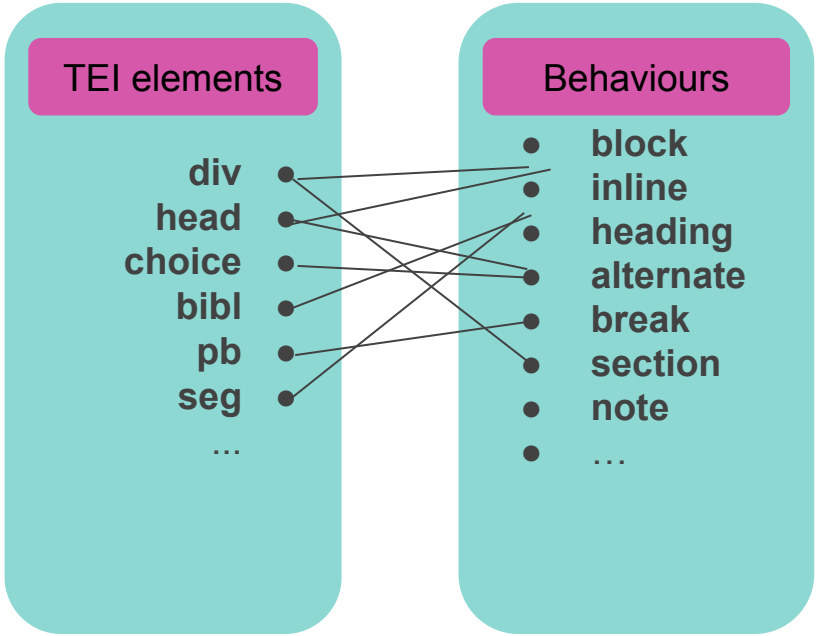
Démo 1

https://teipublisher.com/

Les Processings Models de la TEI

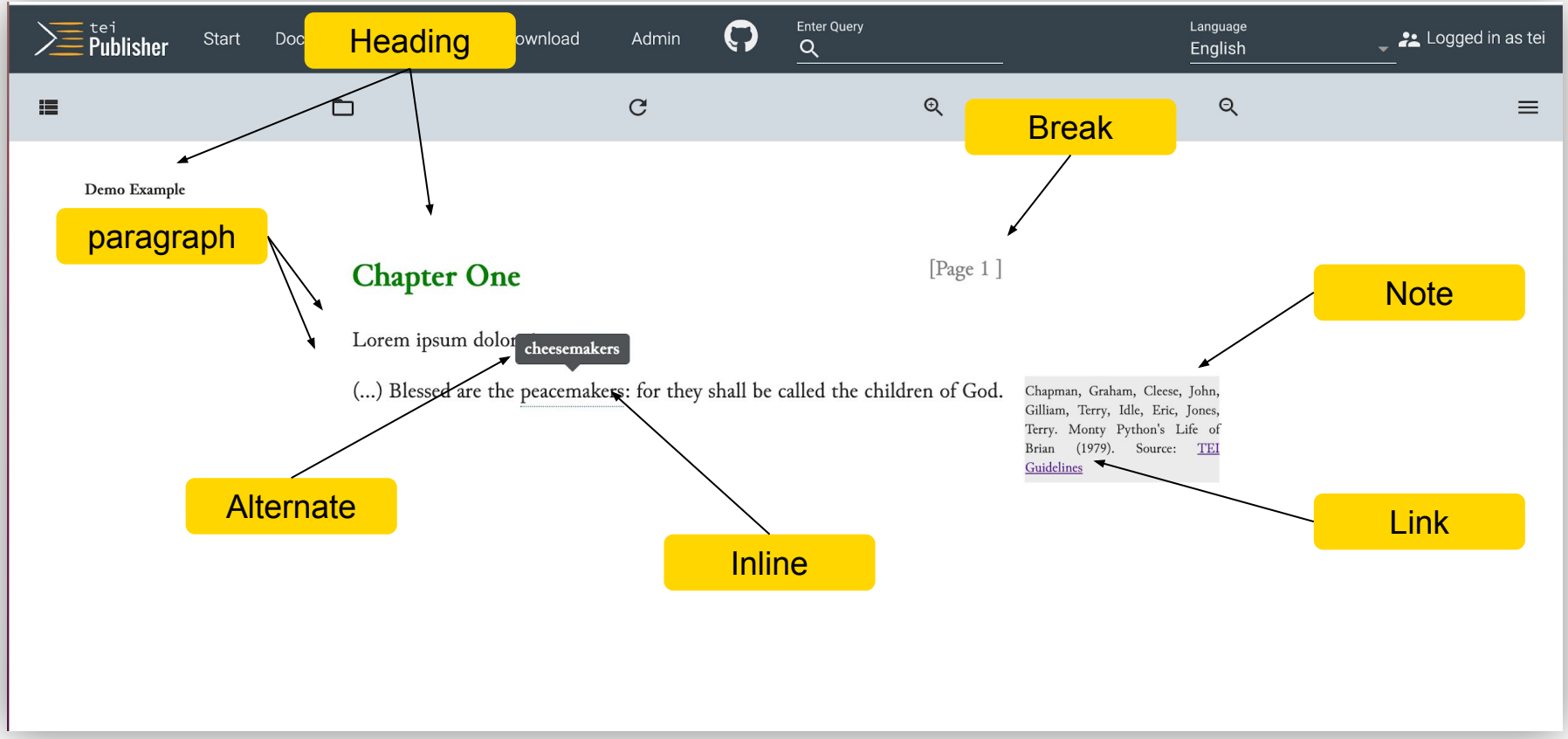
Expression formelle en TEI, indépendante de tout langage de développement, de suggestions de transformation en référence à des “comportements abstraits” (Behaviours) et pour un format de sortie donné.

Des descriptions de mappings entre balises et comportement qui sont explicitées dans le fichier de spécification ODD d’un projet (en complément du schéma et de la documentation).



Comportements suggérés par les Guidelines TEI

1. alternate
2. anchor
3. block
4. body
5. break
6. cell
7. cit
8. document
9. figure
10. glyph
11. graphic
12. heading
13. index
14. inline
15. link
16. list
17. listItem
18. metadata
19. note
20. omit
21. paragraph
22. row
23. section
24. table
25. text
26. title


tei Publisher
Start
Doc
Heading
ownload
Admin
Enter Query
Language
English
Logged in as tei
Break
Demo Example
paragraph
Chapter One
[Page 1]
Note
Lorem ipsum dolor
cheesemakers
(...) Blessed are the peacemakers: for they shall be called the children of God.
Chapman, Graham, Cleese, John, Gilliam, Terry, Idle, Eric, Jones, Terry. Monty Python's Life of Brian (1979). Source: TEI Guidelines
Alternate
Inline
Link

Alignement balises / comportements

TEI Source

```
<TEI>
(...)
<text>
<body>
        <div type=”chapter”>

                <head>un titre</head>

                <p> lorem ipsum
                (...)
                </p>
```

ODD

```
<elementSpec ident="div">
        <model predicate="@type='chapter'" behaviour="section"/>
        <model behaviour="block"/>
</elementSpec>
```

```
<elementSpec ident="head">
        <model behaviour="heading"/>
</elementSpec>
```

```
<elementSpec ident="p">
        <model behaviour="paragraph"/>
</elementSpec>
```

Exemple de fonctions pour le comportement “anchor”

Pour chaque format de sortie, on a une fonction XQuery très simple pour
le (HTML, LaTeX, etc.) correspondant au comportement

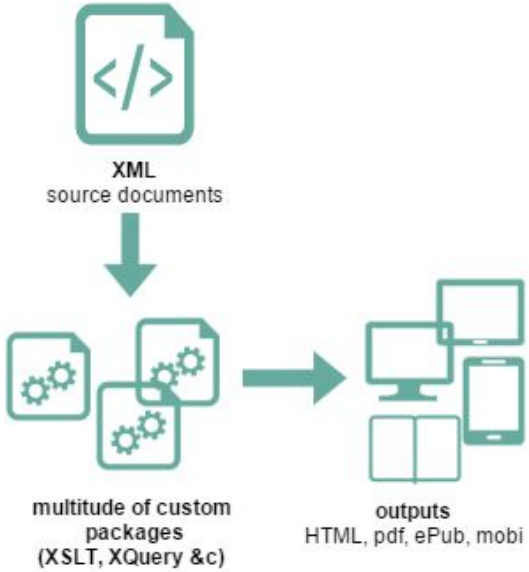
html-functions.xql

```
declare function pmf:anchor($config as map(*), $node as node(), $class as xs:string+, $content, $id as item()*) {
    <span id="{$id}"/>
};
```

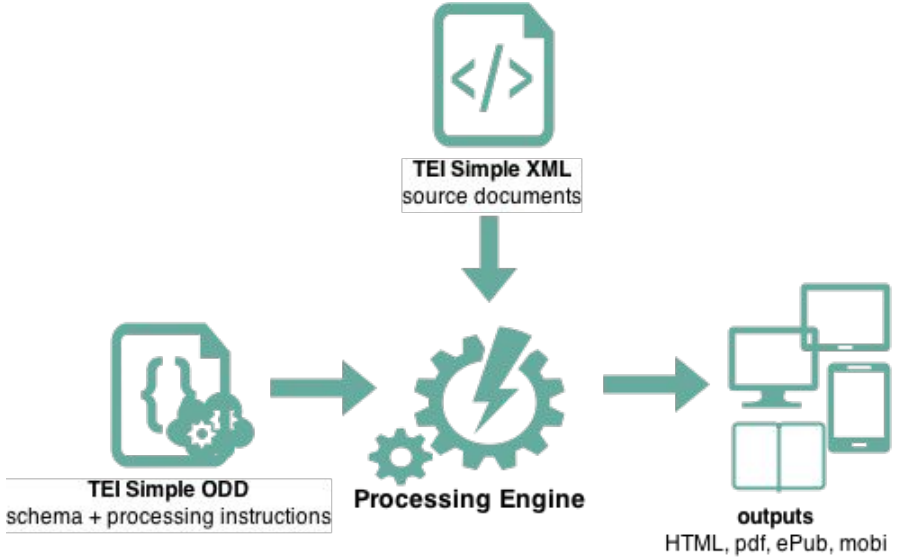
latex-functions.xql

```
declare function pmf:anchor($config as map(*), $node as node(), $class as xs:string+, $content, $id as item()*) {
    "\label{" || $id || "}"
};
```

Transformations directes



Transformation via l'ODD



Credits : Magdalena Turska
http://tei.it.ox.ac.uk/Talks/2015-08-maynooth/img/xmlproc.png

Structure d'un Processing Model

1. La balise TEI à traiter…

2. Le comportement à appliquer

```
<elementSpec ident="head" mode="change">
    <model predicate="parent::div" behaviour="heading">
        <param name="level" value="count(ancestor::div)"/>
        <outputRendition>color:green;</outputRendition>
    </model>
</elementSpec>
```

3. ciblage XPath

4. en passant un paramètre

5. et avec une instruction de stylage optionnelle

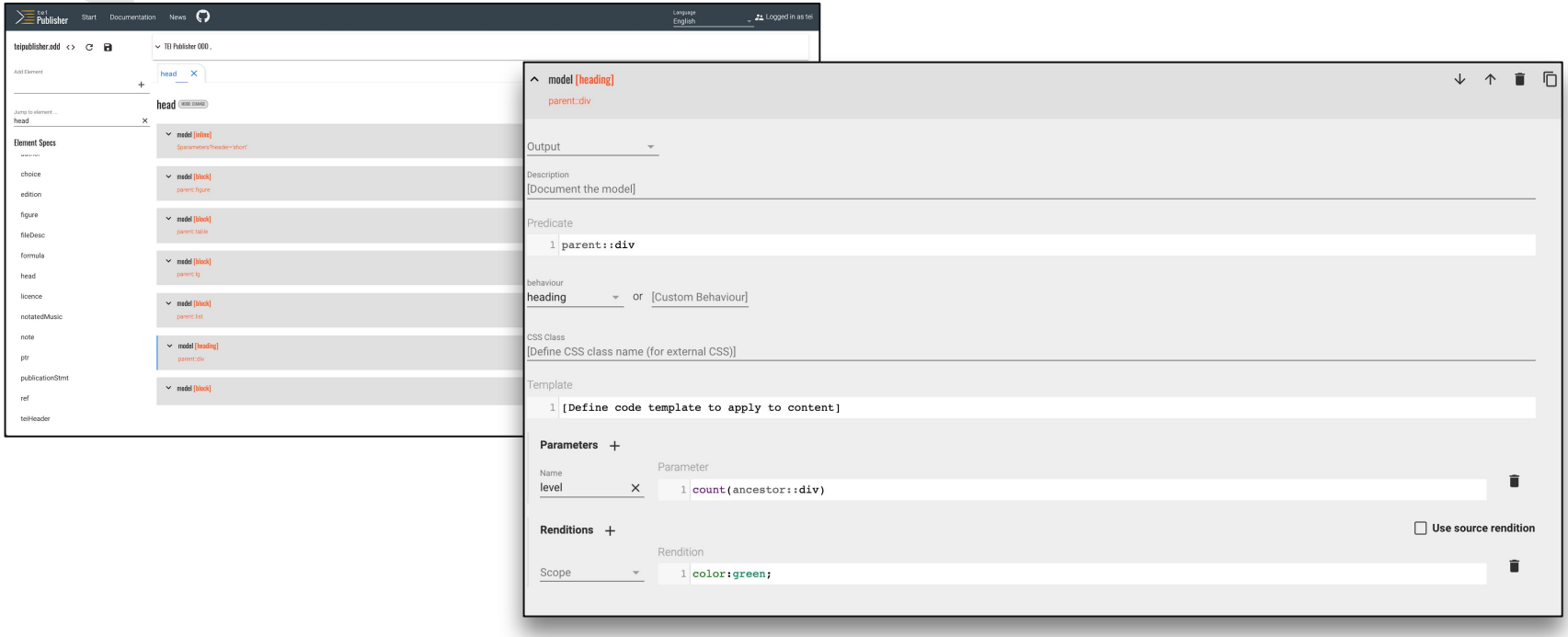
Affichages définis par TEI simple Print : exemple de <head>

Cas de figure prévus

- titre de figure
- titre de tableau
- titre de groupe de vers
- titre de liste
- titre de division

```
<elementSpec ident="head" mode="change">
  <model predicate="parent::figure" behaviour="block">
    <outputRendition>font-style: italic;</outputRendition>
  </model>
  <model predicate="parent::table" behaviour="block">
    <outputRendition>font-style: italic;</outputRendition>
  </model>
  <model predicate="parent::lg" behaviour="block">
    <outputRendition>font-style: italic;</outputRendition>
  </model>
  <model predicate="parent::list" behaviour="block">
    <outputRendition>font-weight: bold;</outputRendition>
  </model>
  <model predicate="parent::div" behaviour="heading">
    <param name="level" value="count(ancestor::div)"/>
  </model>
  <model behaviour="block"/>
</elementSpec>
```

Interface d'édition



Avantages

- Code plus compact
  - Les fonctions de transformation ne sont codées qu'une seule fois / réutilisées autant que nécessaire
  - Maintenance facilitée
- Choix documentés
  - Etend le contrôle scientifique des éditeurs sur les choix d'interfaces
  - Standardisés en TEI : garantie d'intelligibilité à long terme
- Définition possible de nouveaux comportements
  - soit "en dur" dans des modules XQuery personnalisés
  - soit dans l'ODD via l'élément <pb:behaviour>

Démo 2

https://teipublisher.com/

Exercice : utilisation de l’ODD epidoc-test via VSCode

- Dans VSCode, ouvrez un fichier EpiDoc
- Cliquez sur l’icône teipublisher
- Choisissez l’ODD “epidoc-test”
- Comparez avec les transformations XSLT du même fichier
- Modifiez en ligne l’ODD dans le bac à sable TEI Publisher
    - https://teipublisher.com/exist/apps/tei-publisher/index.html

Exercices

Exercices (options)

- Changer de style leiden
- Créer un fichier css pour les corrigés de l'exercice 4 et l'utiliser pour la transformation
- Installer EFES (https://github.com/EpiDoc/EFES) et l'explorer
- Publier un document dans le bac à sable de TEI Publisher et ajuster les ODD