XPath, XSLT : des langages XML appliqués à l'EAD

Description

Point sur les fonctions XML utilisables à partir des fichiers Calames

Public

Personnels chargés de la gestion d'instruments de recherche en XML/EAD

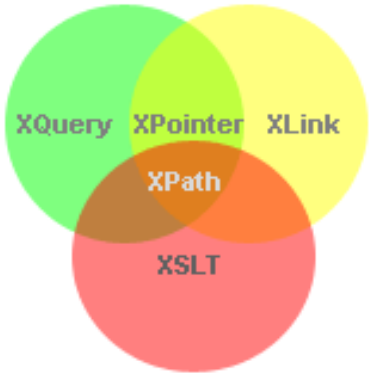
Intervenants

Michael Jeulin, Jean-Marie Feurtet

La formation débutera à 11h, merci de votre patience…
Attention : La session sera enregistrée afin d'être diffusée sur notre plateforme d'autoformation http://moodle.abes.fr.
En rejoignant cette session, vous consentez à ces enregistrements.

Sommaire

- Rappels sur le langage XML
- Les chemins XML : Xpath
- Le filtre d'export dans Calames
- Transformer l'XML-EAD : le langage XSLT



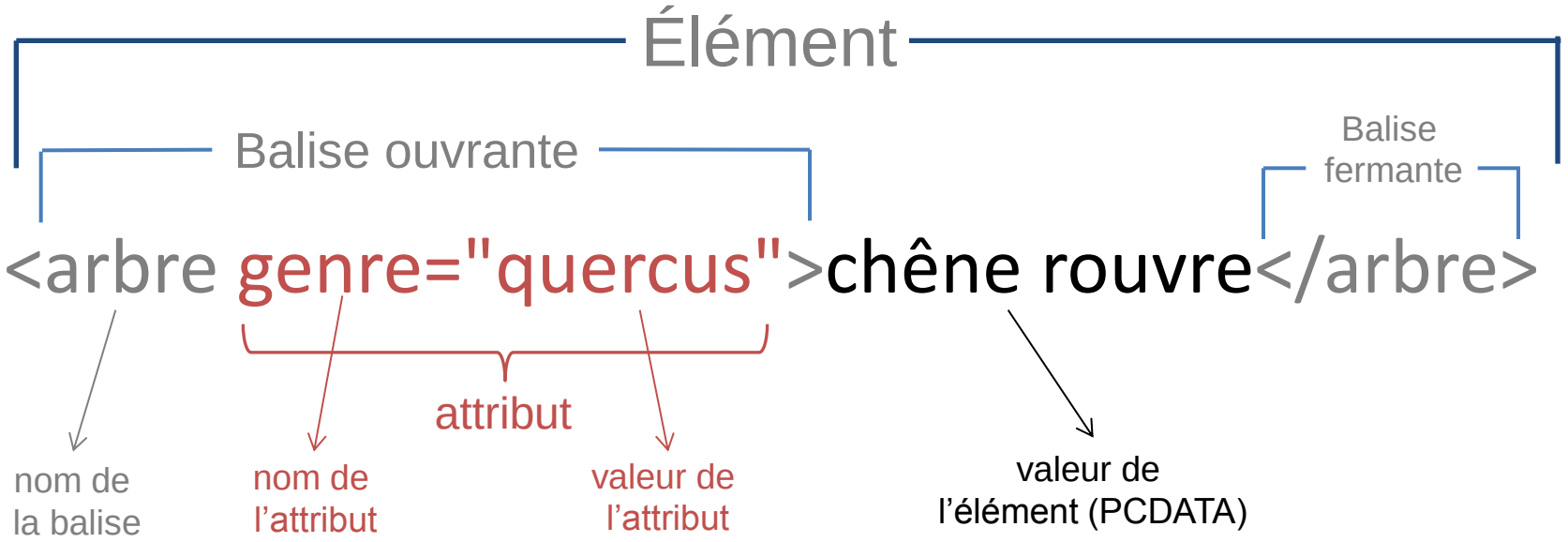
Rappels sur le langage XML

Quelques rappels sur XML

eXtensible Markup Language

- Langage de balisage étendu pour la représentation structurée de données (comme HTML, SGML…)
- Développé par le W3C dès 1996 (à p. de SGML)
- Distinction forte entre présentation (structure) et contenu (données)
- Avantages : universalité (principes simples), pérennité (indépendance vis-à-vis des techniques propriétaires), portabilité (via tous protocoles)

XML (rappels)



Notions-clés :

* Eléments (EMPTY, PCDATA, ANY), balises, attributs
* Document ***bien formé*** = respecte les règles syntaxiques de XML
* Document ***valide*** = respecte en plus une DTD (Définition de Type de Document).
* Emploi croissant des **schémas XML** (les DTD n'étant pas au format XML)

Des grammaires XML : DTD vs schémas

- Définissent la structure valide d'un document
- Définitions de Types de Documents (fichiers .dtd)
  - Fichier à balises SGML (≠ XML)
- XML Schema Document (fichiers .xsd)
  - full xml
  - Définition d'espaces de noms
  - Validation plus précise (typage des données)
- EAD = format XML exprimant la norme ISAD(G)

  ⇒ v1 DTD SGML 1998

  ⇒ v2 DTD XML 2002

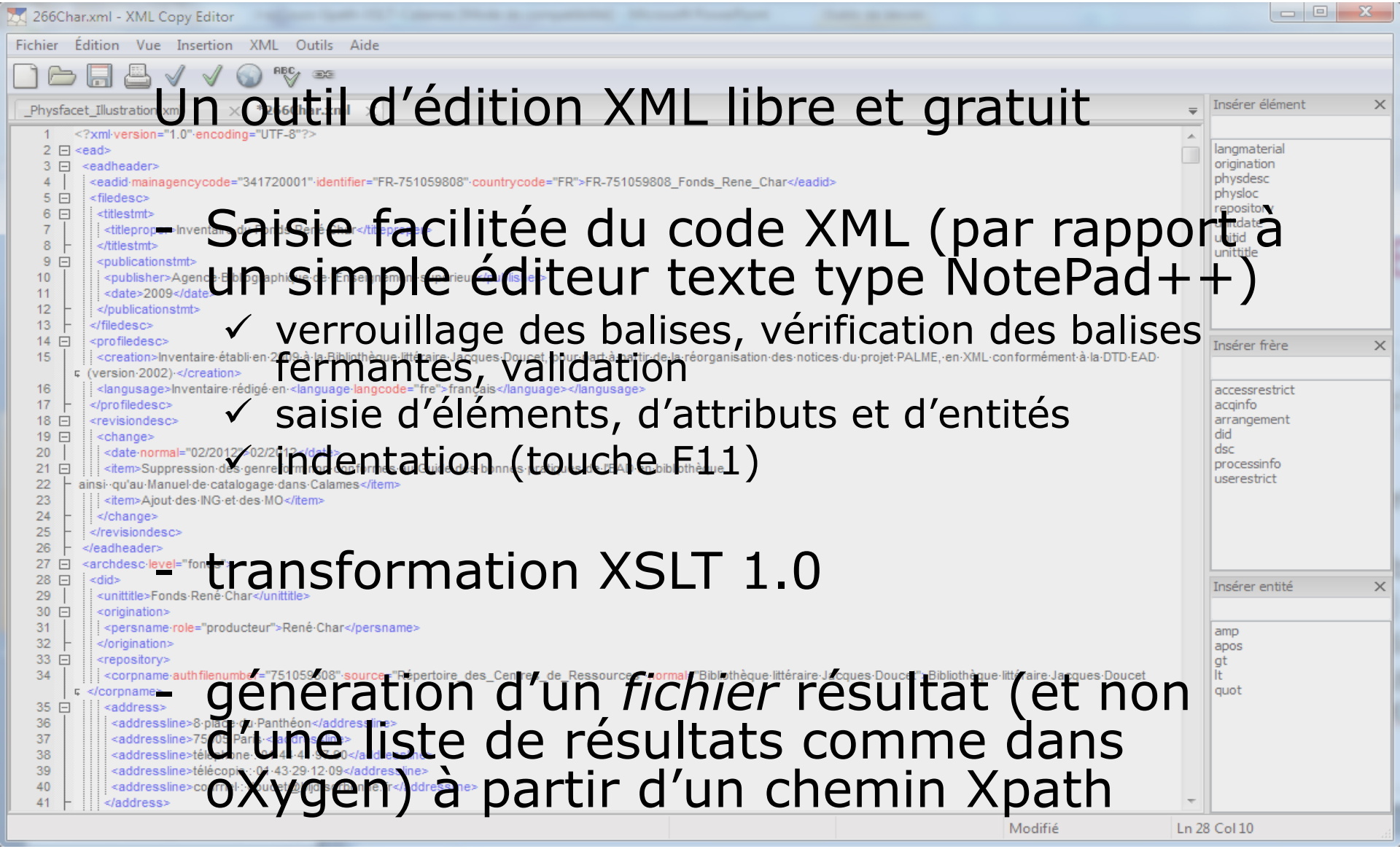
  ⇒ v3 XSD 2015

Editeurs XML

Trois outils d'édition XML seront abordés dans le cadre de ce j-e.cours :

- XML Copy Editor (libre et gratuit)
- l'un des meilleurs éditeurs XML sur poste : oXygen (propriétaire et payant)
- XMAX, plugin de l'éditeur XMetal inséré dans l'environnement d'édition en ligne de Calames

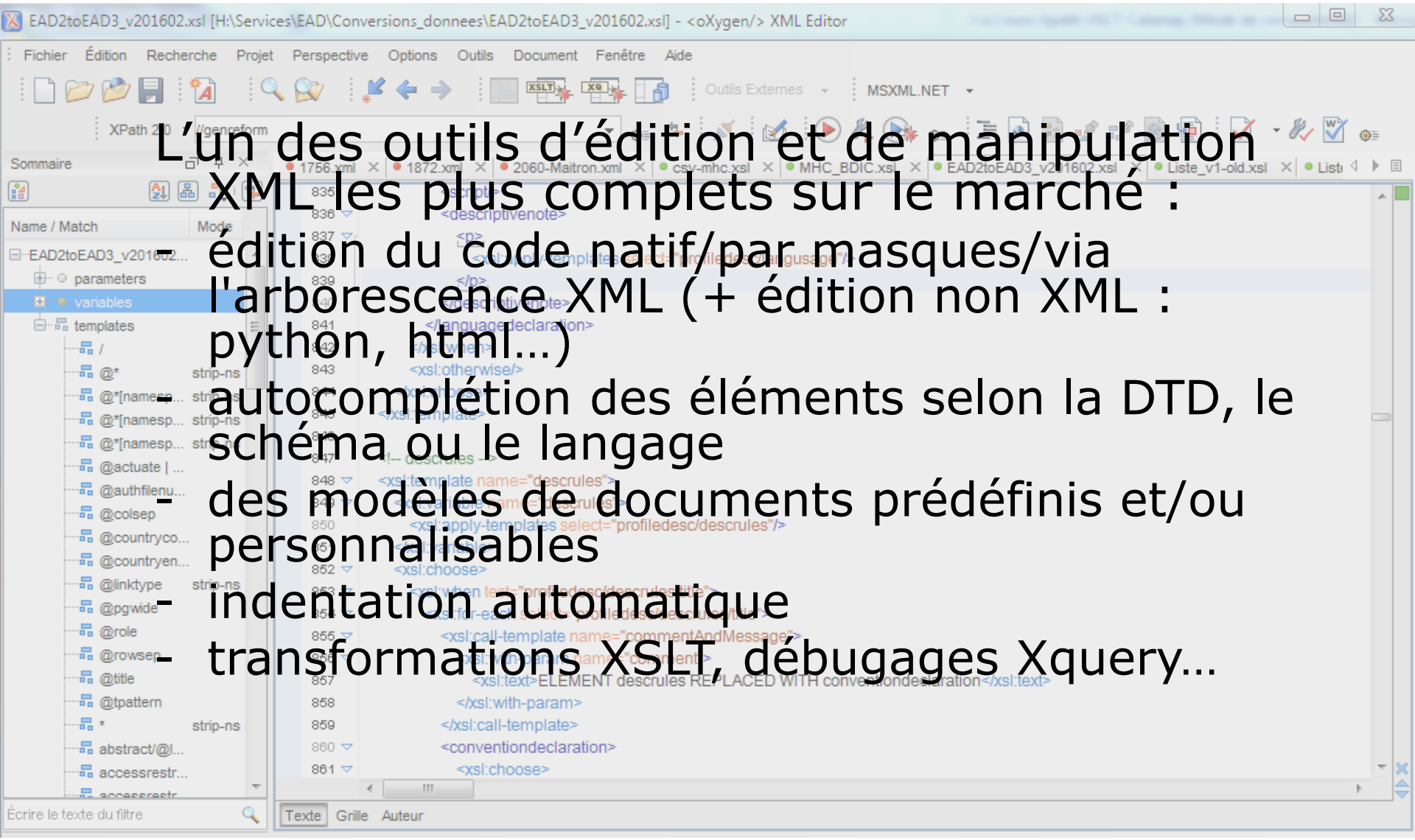
XML Copy Editor



Un outil d'édition XML libre et gratuit

- Saisie facilitée du code XML (par rapport à un simple éditeur texte type NotePad++)
  - ✓ verrouillage des balises, vérification des balises fermantes, validation
  - ✓ saisie d'éléments, d'attributs et d'entités
  - ✓ indentation (touche F11)
- transformation XSLT 1.0
- génération d'un *fichier* résultat (et non d'une liste de résultats comme dans oXygen) à partir d'un chemin Xpath

oXygen XML Editor



L'un des outils d'édition et de manipulation XML les plus complets sur le marché :

- édition du code natif/par masques/via l'arborescence XML (+ édition non XML : python, html…)
- autocomplétion des éléments selon la DTD, le schéma ou le langage
- des modèles de documents prédéfinis et/ou personnalisables
- indentation automatique
- transformations XSLT, débugages Xquery…

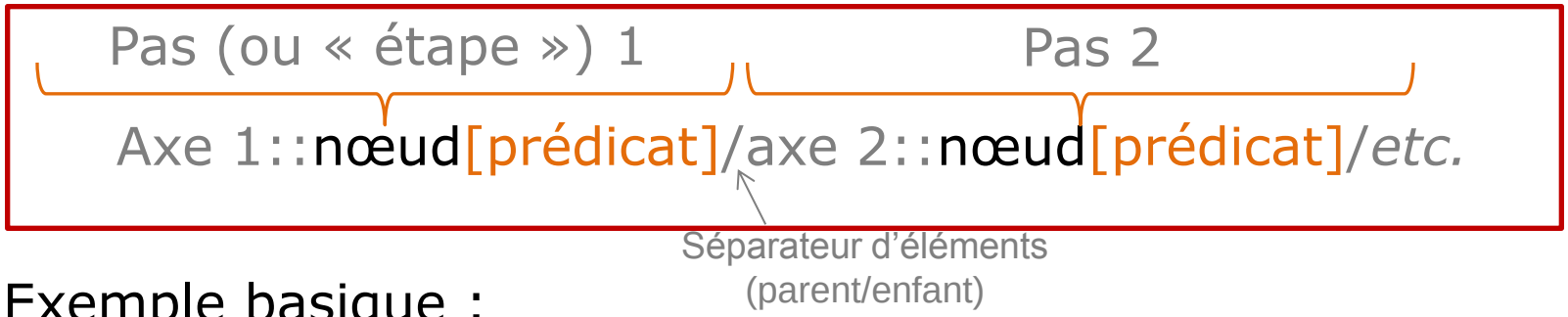
Les chemins XML : XPath

XPath : localiser des fragments XML

- XPath est un langage de sélection

  $\Rightarrow$pas de requête (≠ SQL pour les bases de données)
- Un chemin permet de parcourir pas à pas une arborescence xml
- Il peut être *relatif* ou *absolu*
- Les éléments sélectionnés à chaque pas peuvent être multiples (≠ chemin d'un fichier ou répertoire)
- Résultats possibles d'une expression XPath :
  - une chaîne de caractères (i.e. un nœud ou un nodeset)
  - un booléen ou un nombre (en cas d'usage de fonctions)

La grammaire XPath



- Exemple basique :

  /ead/archdesc/dsc/c[3]

  Prédicat = condition (position du <c>)

  ⇒ Sélectionne le 3e <c> de niveau 1 (= «enfant» de <dsc>)

- *Un peu plus* complexe :

  //*[@level="item"]/child::**node()**[position()=last()]

  axe 1 — test de nœud 1 (joker d'éléments) — prédicat 1 (avec un axe attribut abrégé @) — axe 2 — test de nœud 2 — prédicat 2 (faisant appel à deux fonctions)

  ⇒ Donne tous les *nœuds* derniers de leur fratrie, dont l'élément *parent*, quel qu'il soit et où qu'il soit, a un attribut LEVEL de valeur « item »

Nœuds et axes

- **nœud** = 1 point de l'arborescence
  - **nœud racine**, le plus élevé (introduit par « **/** »)
  - ***** est un joker représentant n'importe quel élément
  - **@*** est un joker pour n'importe quel attribut d'un élément
  - **text()** sélectionne le contenu d'un nœud textuel
    ⇒ attention en EAD : contenus mixtes !
  - autres types de nœuds : comment(), processing-instruction()
  - **node()** représente tout type de nœud

- **axes** exprimant les relations « généalogiques » dans l'arborescence, par rapport au *nœud courant :*
  - self:: le nœud courant lui-même(« vous êtes ici »)
  - child:: est l'axe par défaut (enfants), peut être omis
  - parent:: inverse de child::
    Ex. parent::**controlaccess** pour atteindre des points d'accès hors de <did>
  - ancestor:: tous les ancêtres / descendant:: toute la descendance
    - Variantes : descendant-or-self:: / ancestor-or-self::
  - preceding-sibling:: / following-sibling:: pour sélectionner une partie de fratrie
    ≠preceding:: / following:: pour sélectionner une partie de fichier xml selon l'ordre de lecture

Les fonctions XPath

- Introduites par le préfixe fn: (facultatif)
- Suivies de parenthèses (vides ou contenant des paramètres)

- Quelques fonctions usuelles :
  - ✓ count() : compter un nombre de nœuds
  - ✓ position() pour trouver la position d'un nœud dans une fratrie, last() pour trouver le dernier de ces nœuds
  - ✓ Fonctions de chaînes de caractères (strings) :
    - string() retourne la chaîne en fonction du type de nœud
    - string-lenth() pour connaître le nombre de caractères
    - contains() et starts-with() pour trouver des nœuds contenant ou commençant par telle chaîne
    - substring(), substring-before() et substring-after() pour extraire des sous-chaînes
    - normalize-space() pour supprimer des espaces surnuméraires
  - ✓ Fonctions s'appliquant aux nombres, d'intervention sur des séquences [sum()…], booléennes [not(), and, or]…
  - ✓ fn:document() pour accéder à un document extérieur au fichier XML de départ

Exemples XPath : chemins absolus et relatifs

- Chemin absolu / relatif
  - absolu = commence par un / (slash simple) : part de la racine du fichier XML/EAD, positions précisées en prédicats
    /**ead**/**archdesc**[1]/**dsc**[1]/**c**[1]/**c**[3]
    ⇒ *positions spécifiées en prédicats*
    ⇒ *renvoie le 3e sous-<c> du 1er <c> de premier niveau*
  - relatif = susceptible de sélectionner plusieurs nœuds résultats
    /**ead**/**archdesc**/**did**/**repository**
    ⇒ *chemin relatif, mais unique dans le contexte des bonnes pratiques EAD : renvoie l'organisme responsable mentionné en haut niveau*

- Chemin commençant par // (double slash) : le premier « pas » peut être situé à tout niveau de l'arborescence
  //**c**/**did**/**unitid**[not(@**type**="cote")]
  ⇒ *tous les <unitid> qui ne se sont pas du type « cote »*
  //**c**[**did**/**unitid**[@**type**="cote"]]
  ⇒ *tous les <c> avec un unitid de type « cote »*
  //**c**[//**unitid**[@**type**="cote"]]
  ⇒ *tous les <c> dont la descendance (y compris des sous-<c>) comporte un <unitid> de type « cote »*

Exemples de chemins XPath : autres fonctions courantes

- Usages du point :
  - simple «**.**» nœud contextuel
  - double «**..**» remonter au nœud parent

  //**unittitle**[(**..**/**..**/**c**)/@level="file"]

  ⇒ *mène aux <unittitle> qui descendent (directement et à l'exclusion des <c> ancêtres) d'un <c> de niveau « dossier »*

- Fonctions courantes : contains(), count()…

  //**physfacet**[@type="conditionnement"]/**text()**[contains(.,'Entoilage')][not(contains(.,'Encadrement'))]

  ⇒ *mène aux éléments décrivant le conditionnement et incluant la chaîne de caractère «Entoilage» mais pas «Encadrement»*

- Avec une union d'expressions (|=« pipe ») :

  count(//**dao** | //**daogrp**)

  ⇒ *dénombre les éléments signalant des reproductions numériques dans un fichier ou un fragment EAD (dao + daogrp)*

Exemples de chemins XPath : problématiques particulières à l'EAD

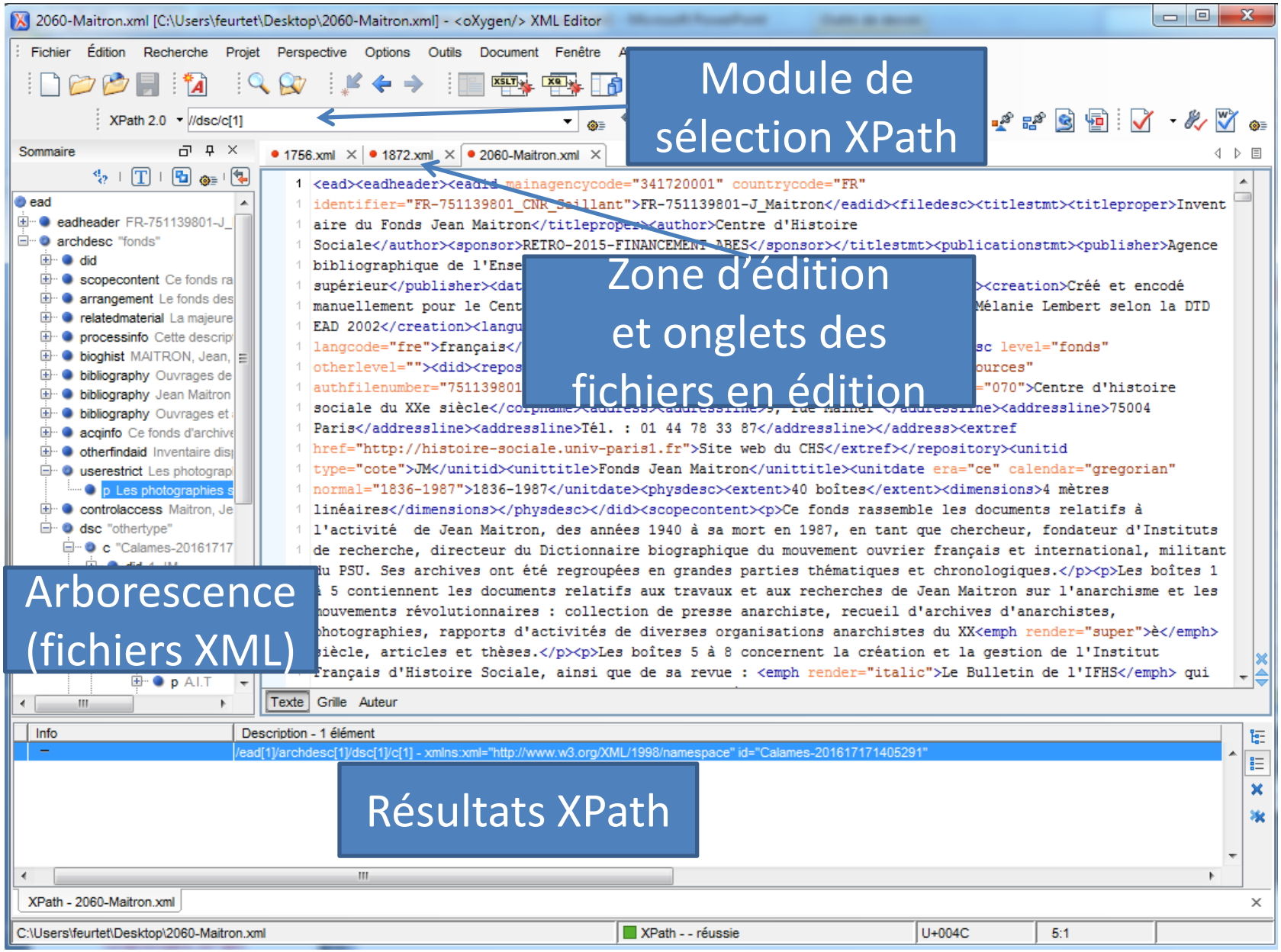
- Problématiques particulières en EAD
  - Un niveau descriptif correspond *stricto sensu* aux sous-éléments d'un <c>, *à l'exception* des sous-<c> (ou de <dsc> pour le haut niveau <archdesc>)
  - XPath est un moyen d'appliquer les notions d'héritage entre niveaux descriptifs EAD (notions qui ne sont pas explicites dans la structure XML)

- Application :

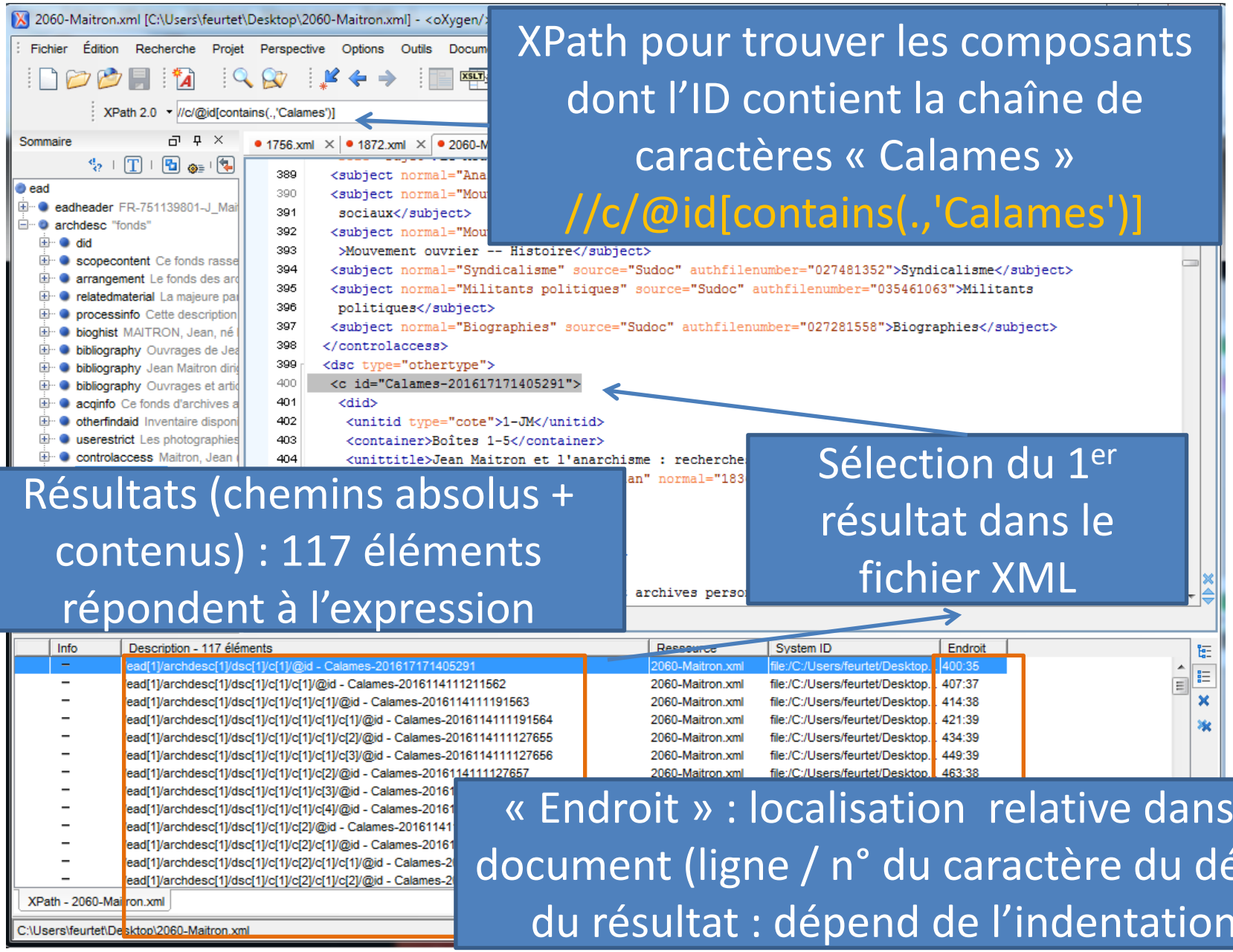
```
//c[ancestor::c/*[not(self::c)]//genreform[@type="type de document"]]
```

⇒ *…mène à tous les niveaux <c>* ***héritant*** *d'une indexation «type de document».*

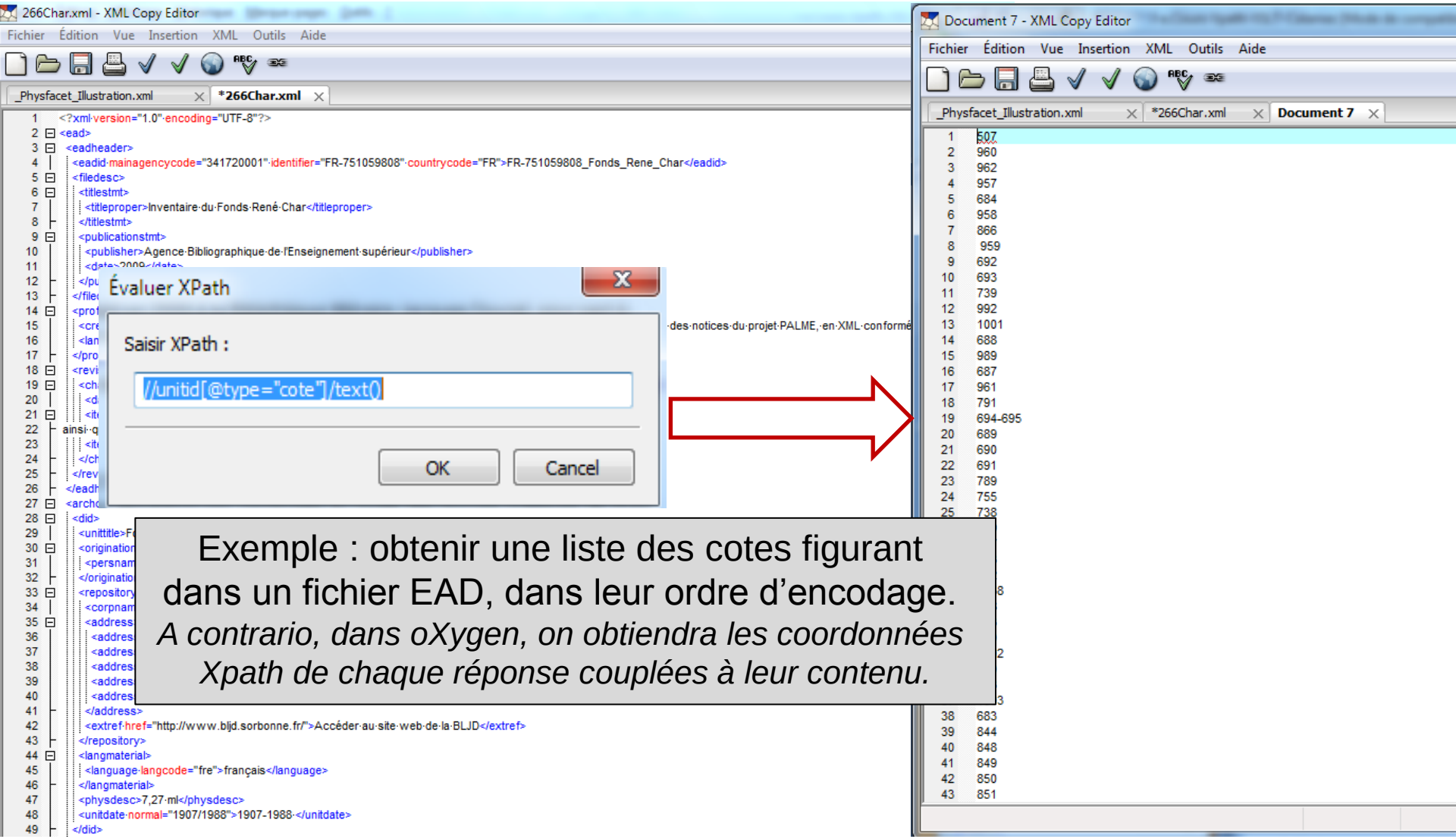
Usage d'XPath dans oXygen



Usage d'XPath dans oXygen



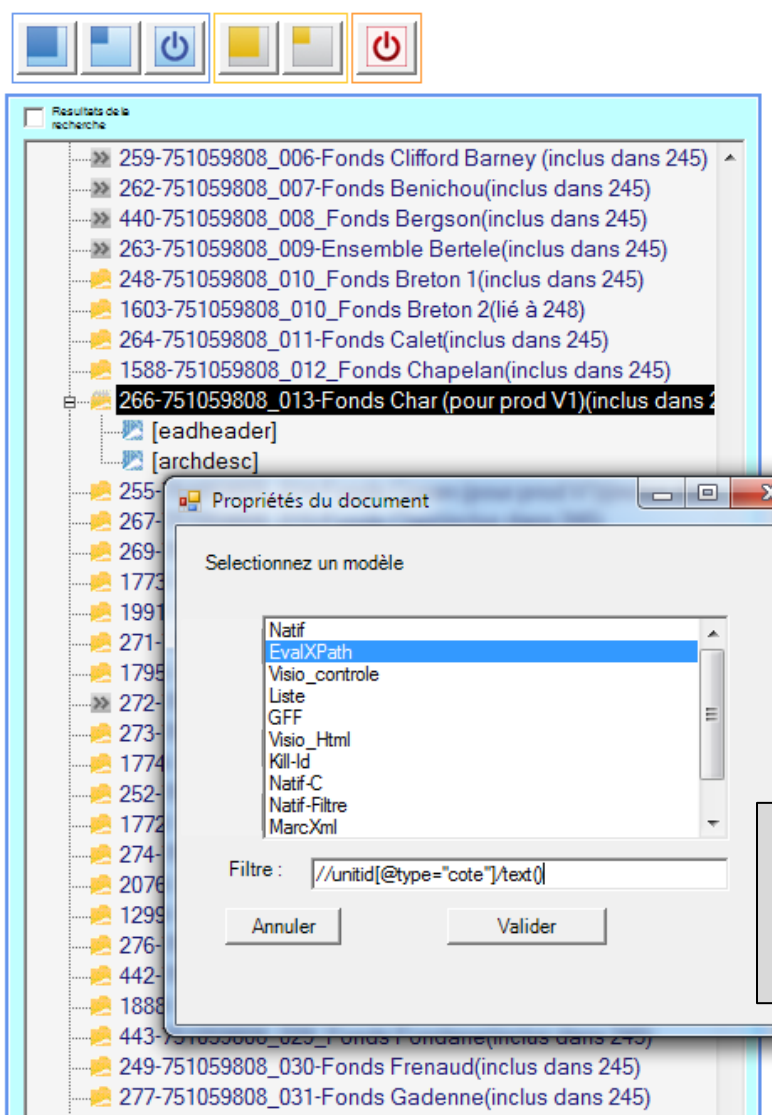
Usage d'XPath dans XML Copy Editor

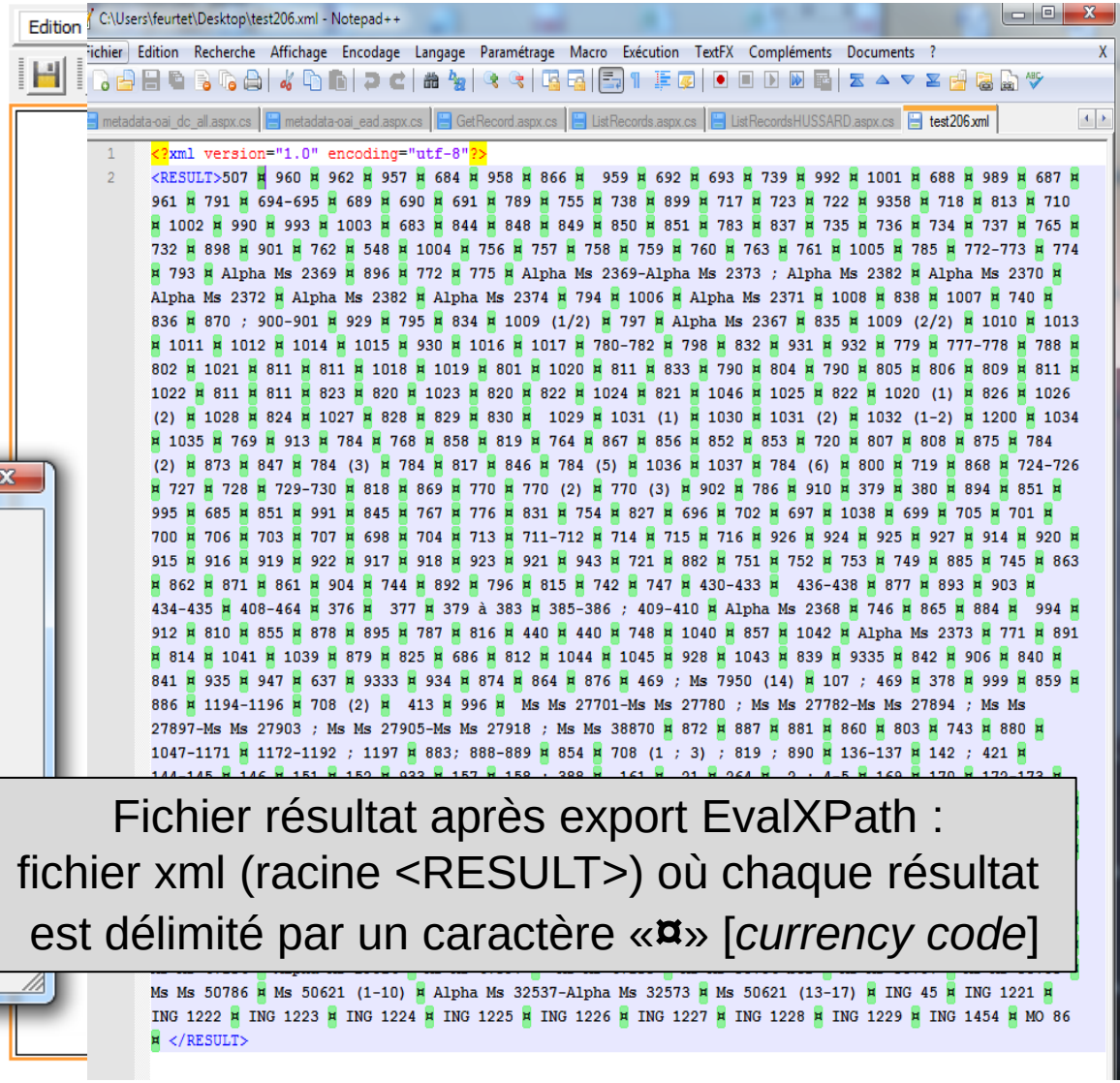


Exemple : obtenir une liste des cotes figurant dans un fichier EAD, dans leur ordre d'encodage.

A contrario, dans oXygen, on obtiendra les coordonnées Xpath de chaque réponse couplées à leur contenu.

Equivalent Calames : l'export EvalXPath



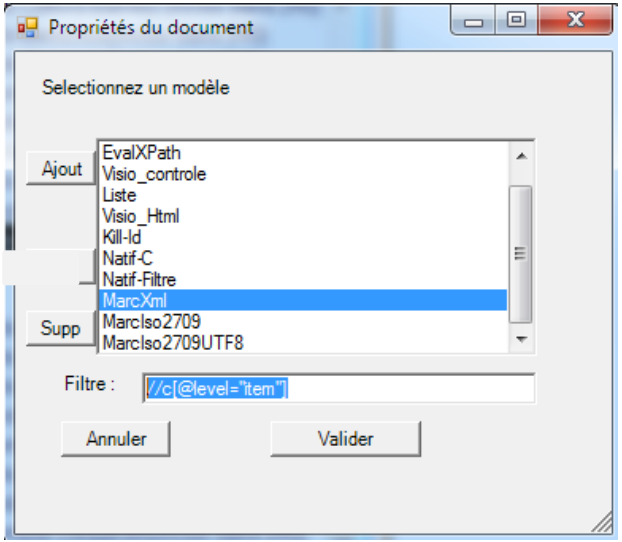


Fichier résultat après export EvalXPath : fichier xml (racine <RESULT>) où chaque résultat est délimité par un caractère «¤» [*currency code*]

Le filtre d’export dans Calames

Exports Calames utilisant le filtre XPath

- Nécessaire pour les exports :
  - EvalXPath
  - Natif-C / Natif-Filtre
  - Marc (XML ou ISO 2709)
- Spécifie un périmètre à l'ensemble des transformations qu'applique une feuille de style XSLT
- Adapter les chemins XPath en fonction de l'export
  ⇒ par ex. «Natif-C» ne peut exporter que des composants <c> : inutile de sélectionner d'autres éléments…



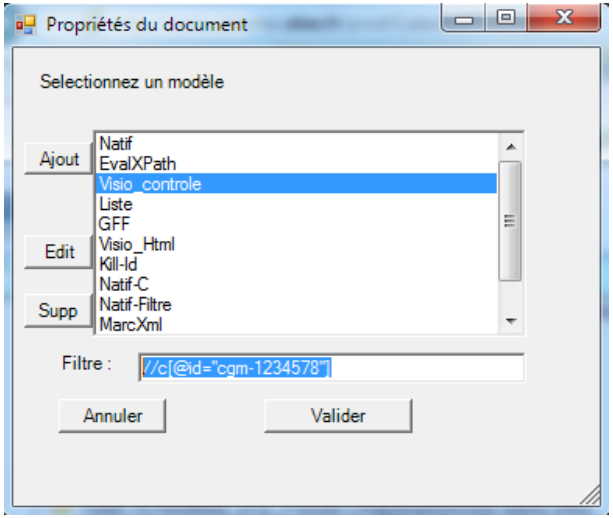
Attention *: transformer les guillemets droits en deux simples quotes*

" → ' + '

Exports Calames sans recours au filtre XPath

- Le filtre est inopérant pour les exports :
  - Natif / Kill-ID
  - Visio_Controle / Visio_Html
  - Liste

- Ces exports appliquent toujours leurs transformations à la racine du fichier XML/EAD
  ⇒Ne pas tenir compte de la valeur par défaut :
  //c[@id=''cgm-1234578'']



Transformer l’XML-EAD : le langage XSLT

XSLT : eXtensible Stylesheet Language Transformation

Pour quoi faire?

- Mettre en forme un fichier XML pour publication statique ou dynamique / en ligne (html, etc.)

  ⇒ Utilisé notamment par l'interface publique Calames :

  affichage détaillé des niveaux descriptifs en pages HTML / RDFa

- Conversion vers d'autres DTD / schémas XML, ou vers d'autres formats :

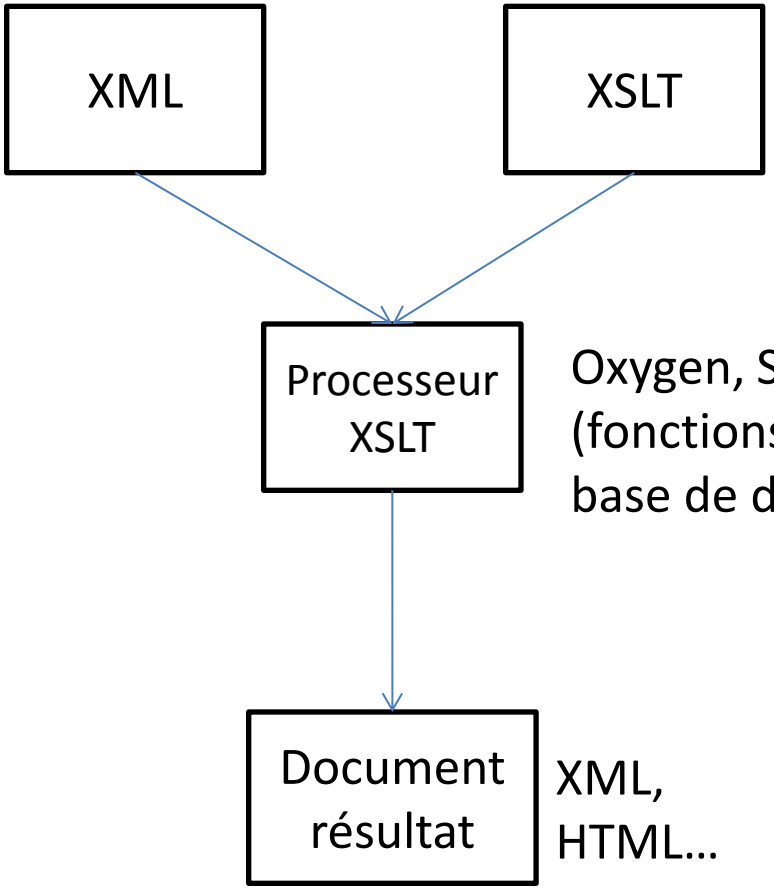
  ⇒ Ex.: EAD2→EAD3, EAD→XML-Dublin Core, EAD→UnimarcXML, EAD→CSV…

- Modifications de masse

  ⇒ Ex.: ajout de liens <dao @href> à partir d'un tableau de correspondances

XSLT : Comment ça marche ?

EAD, TEI, MarcXML …



XSLT : Comment ça marche ?

- « Feuille de style » exprimée en XML (verbosité), et non langage de programmation
  - Décrit un « arbre résultat » construit à partir du document source :
    - Les balises introduites par l'espace de nom <xsl:…> appellent des actions
    - Toute autre balise sera copiée dans le résultat
  - Notion de nœud contextuel
  - Ordre de traitement : du plus spécifique au plus général
  - Pas de possibilité de modifier les « variables » XSL
- Basée sur des *« templates »*, associant des règles à une sélection de nœuds par un chemin Xpath
- Versions 1.0, 2.0 ou 3.0 (attention aux problèmes de compatibilité !)

XSLT : Comment ça marche ?

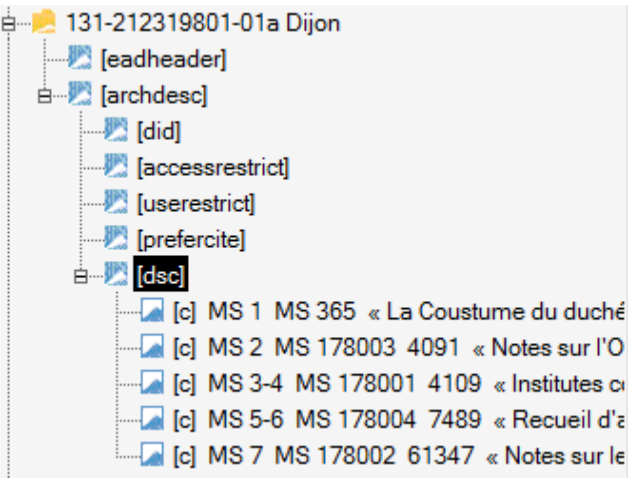
Quelques fonctions courantes et essentielles en XSLT

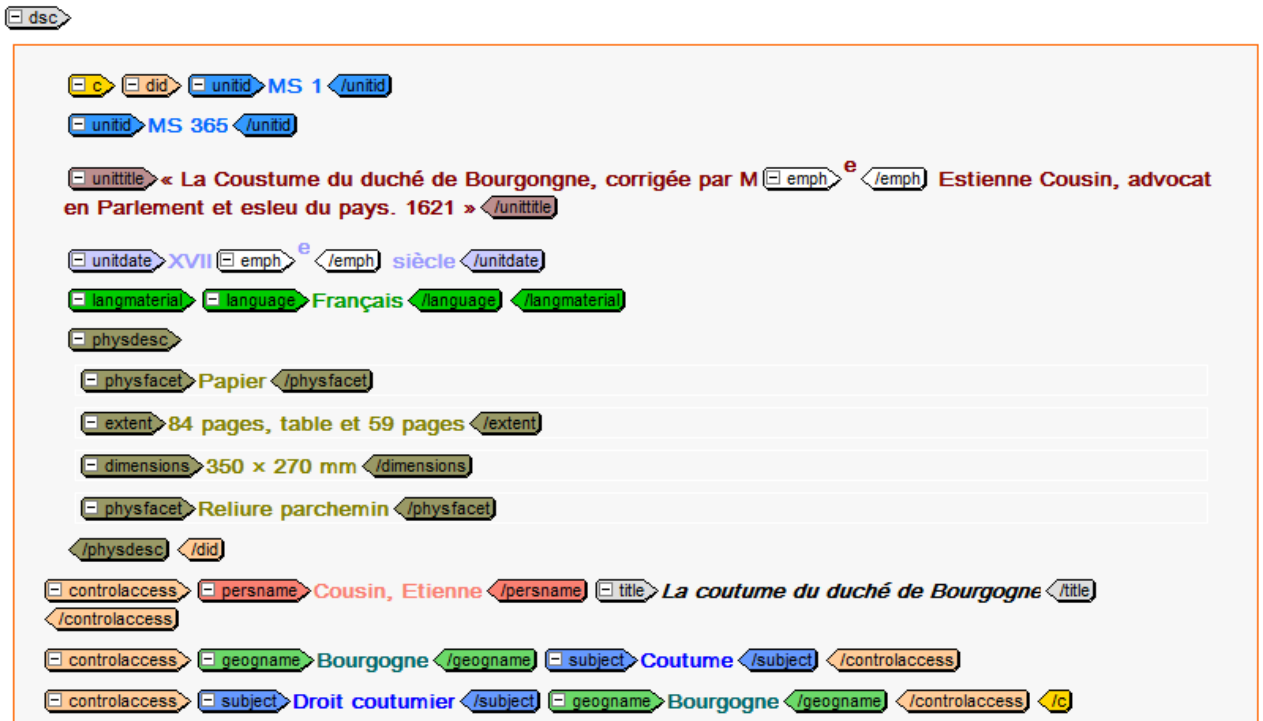
- racine <xsl:stylesheet> : indique la version employée d'XSLT et l'adresse canonique des ns. xsl
- <xsl:output> : indique le format de sortie
- <xsl:template> : englobe les règles à appliquer à un nœud donné
- … appelé par <xsl:apply-templates> ou <xsl:call-template>
- <xsl:value-of> : utilisé pour *sélectionner et extraire* (attribut @select) les valeurs d'éléments
- <xsl:for-each> : en employant un XPath relatif (menant à un *nodeset*) dans l'attribut @select, crée une boucle d'application de règles à chacun des nœuds ciblés
- <xsl:if> : pour poser une condition d'application de règles via l'attribut @test

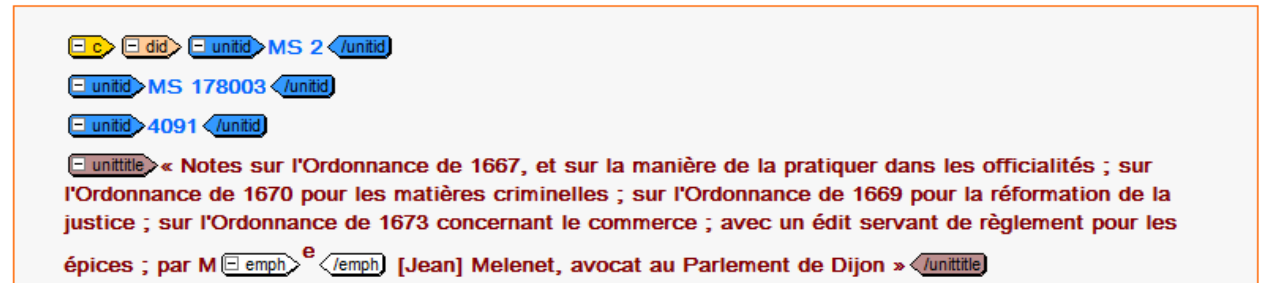
Pour composer un XSLT : partir d'un exemple de fichier résultat, et inférer les règles à appliquer au fichier source

Un exemple de transformation XML→HTML (export «Liste»)

Exemple : à partir d'un inventaire de manuscrits en XML/EAD…







Un exemple de transformation XML→HTML (export «Liste»)

… générer un fichier HTML affichant une liste à puces de Titres et de Cotes …

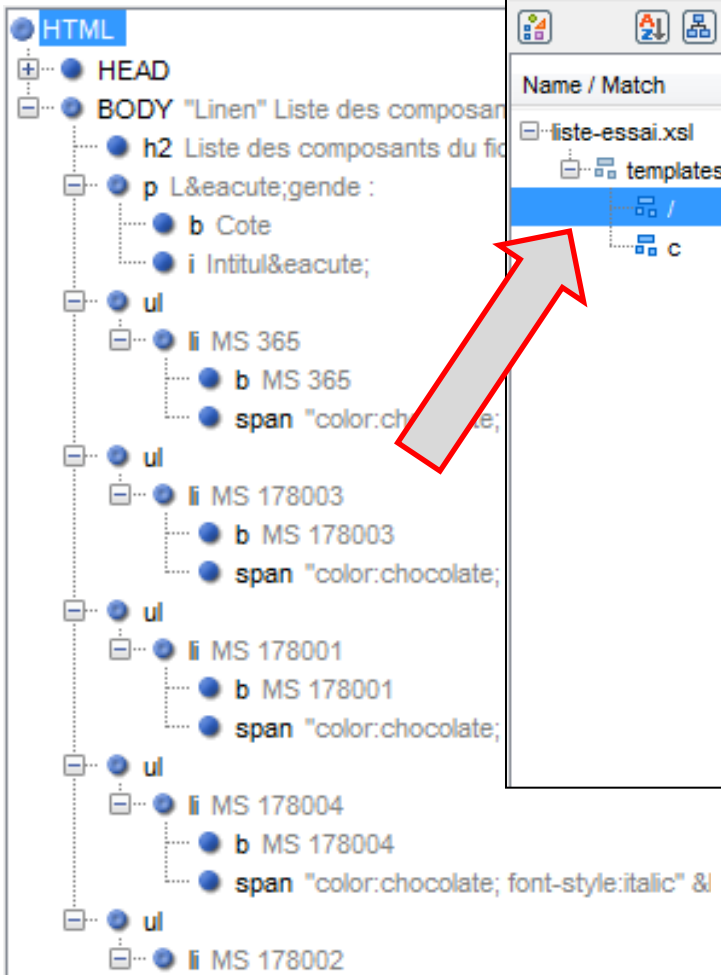
Liste des composants du fichier : Manuscrits de l'Université de Bourgogne (Dijon). Service commun de la documentation. Section Droit-Lettres (FRCGMBPF-212319801-01a)

Légende : **Cote**. - *Intitulé*.

- **MS 365**. - *« La Coustume du duché de Bourgongne, corrigée par Me Estienne Cousin, advocat en Parlement et esleu du pays. 1621 »*
- **MS 178003**. - *« Notes sur l'Ordonnance de 1667, et sur la manière de la pratiquer dans les officialités ; sur l'Ordonnance de 1670 pour les matières criminelles ; sur l'Ordonnance de 1669 pour la réformation de la justice ; sur l'Ordonnance de 1673 concernant le commerce ; avec un édit servant de règlement pour les épices ; par Me [Jean] Melenet, avocat au Parlement de Dijon »*
- **MS 178001**. - *« Institutes coutumières, ou Manuel de plusieurs proverbes, sentences et règles du droit commun et plus ordinaire de la France, par Me Antoine Loizel, avec les notes, observations et commentaire de M. Davot, professeur de l'Université. A Dijon, 1765 »*
- **MS 178004**. - *« Recueil d'arrêts notables du Parlement de Dijon, par M. de B. [J.-B. Fleutelot de Beneuvre]. 1780 »*
- **MS 178002**. - *« Notes sur les Ordonnances des mois d'avril 1667, aoust 1669, aoust 1670, mars 1673, et sur l'édit des épices, du mois de mars 1673. A la suitte sont des Remarques sur la manierre de pratiquer l'Ordonnance de 1667 dans les officialités ; par Me [Jean] Melenet, avocat au Parlement de Bourgogne. Dijon, 1750 »*

Un exemple de transformation XML→HTML (export «Liste»)

Structure du fichier HTML attendu en résultat



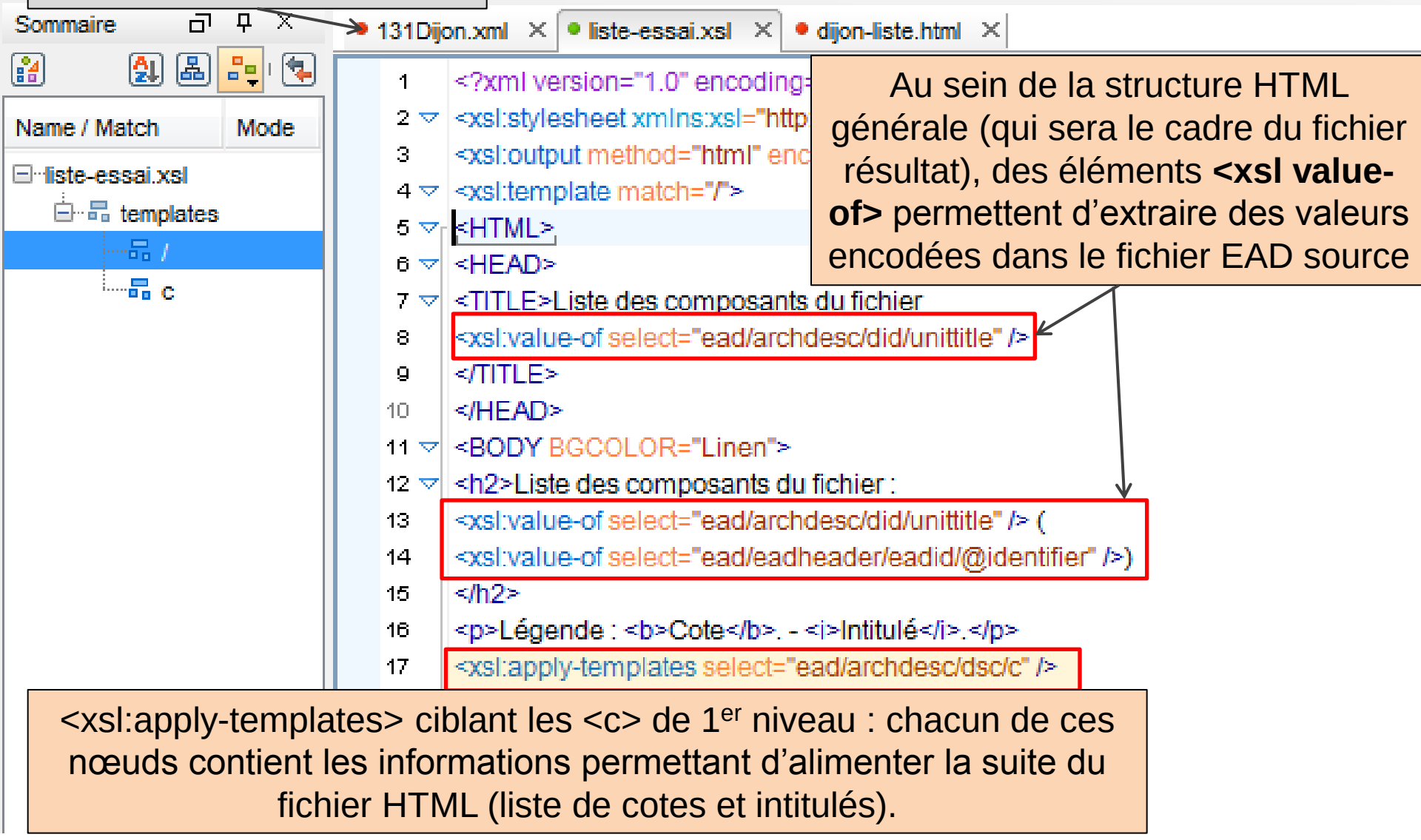
131Dijon.xml | liste-essai.xsl | dijon-liste.html

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3  <xsl:output method="html" encoding="UTF-8" />
4  <xsl:template match="/">
5  <HTML>
6  <HEAD>
7  <TITLE>Liste des composants du fichier
8  <xsl:value-of select="ead/archdesc/did/unittitle" />
9  </TITLE>
10 </HEAD>
11 <BODY BGCOLOR="Linen">
12 <h2>Liste des composants du fichier :
13 <xsl:value-of select="ead/archdesc/did/unittitle" /> (
14 <xsl:value-of select="ead/eadheader/eadid/@identifier" />)
15 </h2>
16 <p>Légende : <b>Cote</b>. - <i>Intitulé</i>.</p>
17 <xsl:apply-templates select="ead/archdesc/dsc/c" />
18 </BODY>
19 </HTML>
20 </xsl:template>
21 <xsl:template match="c">
```

Soit le fichier XSLT « liste-essai.xsl ».
Dans le 1er template général (ciblant la racine), est reportée cette structure HTML

Un exemple de transformation XML→HTML (export «Liste»)

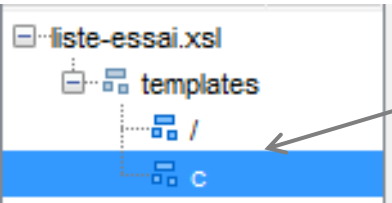
(Fichier XML/EAD source)



Au sein de la structure HTML générale (qui sera le cadre du fichier résultat), des éléments **<xsl value-of>** permettent d'extraire des valeurs encodées dans le fichier EAD source

<xsl:apply-templates> ciblant les <c> de 1er niveau : chacun de ces nœuds contient les informations permettant d'alimenter la suite du fichier HTML (liste de cotes et intitulés).

Un exemple de transformation XML→HTML (export «Liste»)




```
18  </BODY>
19  </HTML>
20  </xsl:template>
21  <xsl:template match="c">
22  <ul>
23  <li>
24  <xsl:if test="normalize-space(did/unitid) != ''">
25  <b>
26  <xsl:value-of select="normalize-space(did/unitid)" />
27  </b>
28  </xsl:if>
29  <xsl:if test="normalize-space(did/unittitle) !=''">
30  <xsl:if test="normalize-space(did/unitid) !=''">
31  <xsl:text>. -
32  </xsl:text>
33  </xsl:if>
34  <span style="color:chocolate; font-style:italic">
35  <xsl:value-of select="normalize-space(did/unittitle)" />
36  </span>
37  </xsl:if>
38  <xsl:apply-templates select="c" />
39  </li>
40  </ul>
41  </xsl:template>
42  </xsl:stylesheet>
```

Second template : règles s'appliquant à tout élément <c> du fichier EAD source

1er test : si présence d'un élément <unitid>, extraction de sa valeur (avec normalisation des espaces) en /<b>

2e test : si présence d'un élément <unittitle> : -ponctuation si une valeur cote a déjà été extraite -extraction de sa valeur (avec mise en forme)

Apply-templates : application récursive sur tous les <c>

← Fin de la feuille de style liste-essai.xsl

Conclusion

- Pour en savoir plus :
  - XPath : cours de Ph. Poulard (INRIA)
  - XSLT : cours de F. Torre (Univ. Lille)
- A noter : le langage de requête et d'extraction/transformation XQuery, s'appuyant sur XPath, est utilisé par les procédures d'indexation des données Calames