

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

## Description

Point sur les fonctions XML utilisables à partir des fichiers Calames

## Public

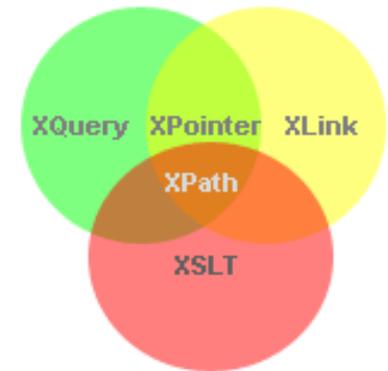
Personnels chargés de la gestion d'instruments de recherche en XML/EAD

## Intervenants

Michael Jeulin, Jean-Marie Feurtet

# Sommaire

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



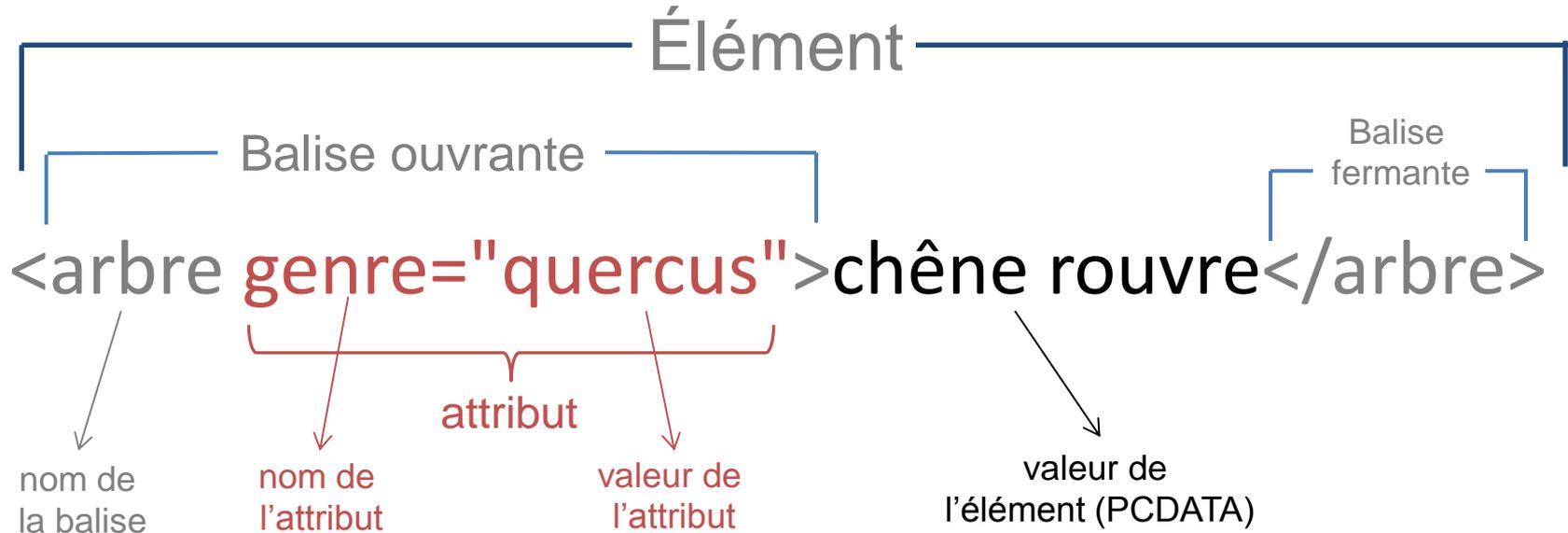
# Rappels sur le langage XML

# 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 :

- \* Éléments (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++)

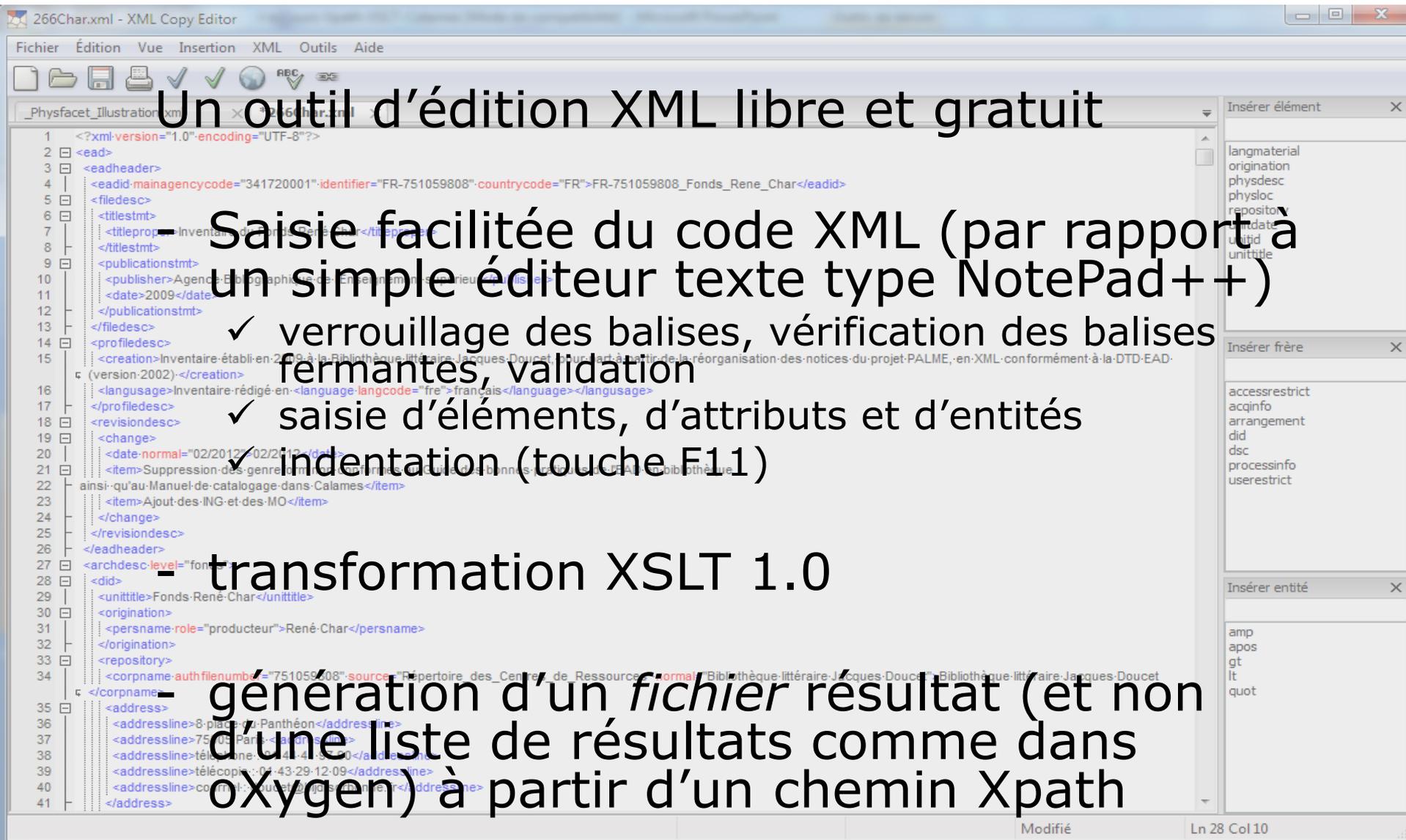
✓ verrouillage des balises, vérification des balises fermantes, validation

✓ saisie d'éléments, d'attributs et d'entités

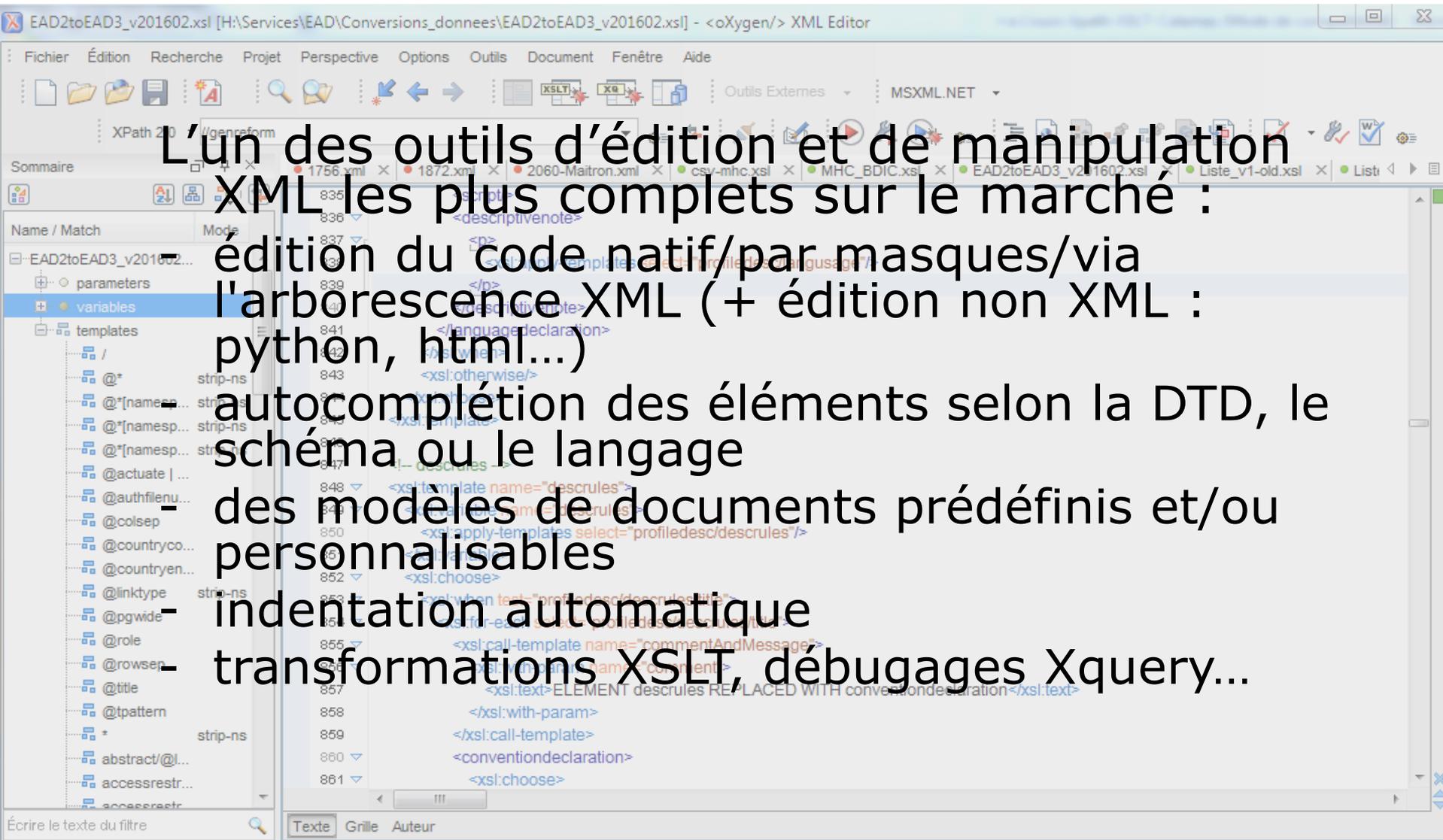
✓ indentation (touche F11)

- 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, debugages Xquery...

# Les chemins XML : XPath

# XPath :

## localiser des fragments XML

- XPath est un **langage de sélection**
  - ⇒ 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)



# 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-length()** 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 3<sup>e</sup> sous-`<c>` du 1<sup>er</sup> `<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

**Module de sélection XPath**

**Zone d'édition et onglets des fichiers en édition**

**Arborescence (fichiers XML)**

**Résultats XPath**

```
1 <ead><eadheader><eadid mainagencycode="341720001" countrycode="FR"
1 identifier="FR-751139801_CNR_Saillant">FR-751139801-J Maitron</eadid><filedesc><titlestmt><titleproper>Invent
1 aire du Fonds Jean Maitron</titleproper><author>Centre d'Histoire
1 Sociale</author><sponsor>RETRO-2015-FINANCEMENT ABRES</sponsor></titlestmt><publicationstmt><publisher>Agence
1 bibliographique de l'Ense
1 supérieur</publisher><date
1 manuellement pour le Cent
1 EAD 2002</creation><langu
1 langcode="fre">Français</
1 otherlevel=""><did><reposit
1 authfilenumber="751139801
1 sociale du XXe siècle</compnum><address><addressline>75, rue Maitron</addressline><addressline>75004
1 Paris</addressline><addressline>Tél. : 01 44 78 33 87</addressline></address><extref
1 href="http://histoire-sociale.univ-paris1.fr">Site web du CHS</extref></repository><unitid
1 type="cote">JM</unitid><slntitle>Fonds Jean Maitron</slntitle><unitdate era="ce" calendar="gregorian"
1 normal="1836-1987">1836-1987</unitdate><physdesc><extent>40 boîtes</extent><dimensions>4 mètres
1 linéaires</dimensions></physdesc></did><scopecontent><p>Ce fonds rassemble les documents relatifs à
1 l'activité de Jean Maitron, des années 1940 à sa mort en 1987, en tant que chercheur, fondateur d'Instituts
1 de recherche, directeur du Dictionnaire biographique du mouvement ouvrier français et international, militant
1 du PSU. Ses archives ont été regroupées en grandes parties thématiques et chronologiques.</p><p>Les boîtes 1
1 à 5 contiennent les documents relatifs aux travaux et aux recherches de Jean Maitron sur l'anarchisme et les
1 mouvements révolutionnaires : collection de presse anarchiste, recueil d'archives d'anarchistes,
1 photographies, rapports d'activités de diverses organisations anarchistes du XX<emph render="super">è</emph>
1 siècle, articles et thèses.</p><p>Les boîtes 5 à 8 concernent la création et la gestion de l'Institut
1 Français d'Histoire Sociale, ainsi que de sa revue : <emph render="italic">Le Bulletin de l'IFHS</emph> qui
```

Info	Description - 1 élément
-	/ead[1]/archdesc[1]/dsc[1]/c[1] - xmlns:xml="http://www.w3.org/XML/1998/namespace" id="Calames-201617171405291"

XPath - 2060-Maitron.xml

C:\Users\feurtet\Desktop\2060-Maitron.xml    XPath -- réussite    U+004C    5:1

# Usage d'XPath dans oXygen

XPath pour trouver les composants dont l'ID contient la chaîne de caractères « Calames »

`//*[c/@id[contains(.,'Calames')]]`

Sélection du 1<sup>er</sup> résultat dans le fichier XML

Résultats (chemins absolus + contenus) : 117 éléments répondent à l'expression

« Endroit » : localisation relative dans le document (ligne / n° du caractère du début du résultat : dépend de l'indentation)

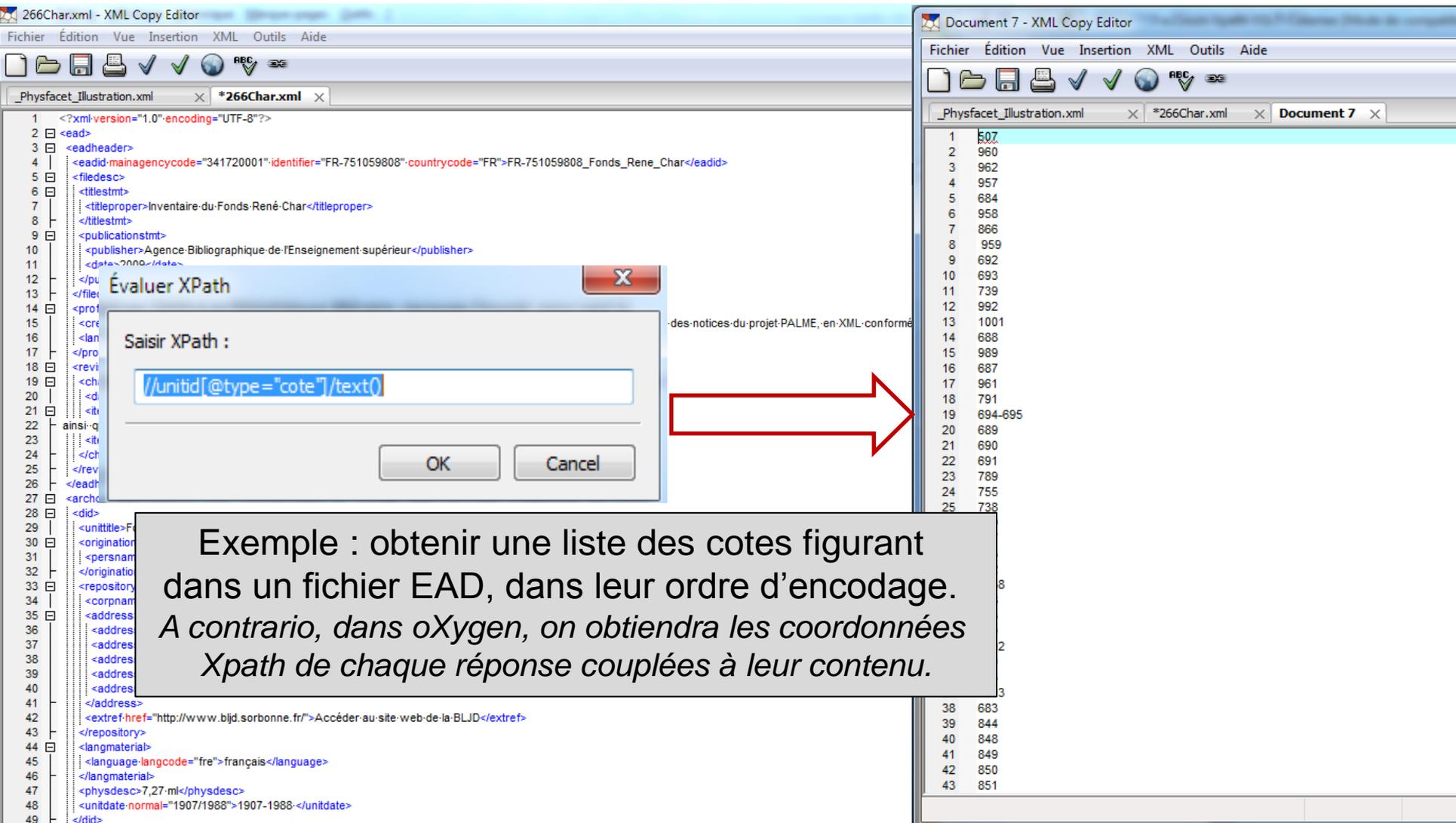
The screenshot shows the oXygen XML editor interface. The main window displays an XML document with the following content:

```
389 <subject normal="Ana
390 <subject normal="Mou
391 sociaux</subject>
392 <subject normal="Mou
393 >Mouvement ouvrier -- Histoire</subject>
394 <subject normal="Syndicalisme" source="Sudoc" authfilenumber="027481352">Syndicalisme</subject>
395 <subject normal="Militants politiques" source="Sudoc" authfilenumber="035461063">Militants
396 politiques</subject>
397 <subject normal="Biographies" source="Sudoc" authfilenumber="027281558">Biographies</subject>
398 </controlaccess>
399 <dsc type="othertype">
400 <c id="Calames-201617171405291">
401 <did>
402 <unitid type="cote">1-JM</unitid>
403 <container>Boites 1-5</container>
404 <unittitle>Jean Maitron et l'anarchisme : recherche
```

The XPath 2.0 expression `//*[c/@id[contains(.,'Calames')]]` is entered in the search bar. The search results are displayed in a table below the editor:

Info	Description - 117 éléments	Resource	System ID	Endroit
-	ead[1]/archdesc[1]/dsc[1]/c[1]/@id - Calames-201617171405291	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	400:35
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[1]/@id - Calames-2016114111211562	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	407:37
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[1]/c[1]/@id - Calames-2016114111191563	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	414:38
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[1]/c[1]/c[1]/@id - Calames-2016114111191564	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	421:39
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[1]/c[1]/c[2]/@id - Calames-2016114111127655	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	434:39
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[1]/c[1]/c[3]/@id - Calames-2016114111127656	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	449:39
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[1]/c[2]/@id - Calames-2016114111127657	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	463:38
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[1]/c[3]/@id - Calames-2016114111127658	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	470:38
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[1]/c[4]/@id - Calames-2016114111127659	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	477:38
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[2]/@id - Calames-2016114111127660	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	484:38
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[2]/c[1]/@id - Calames-2016114111127661	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	491:38
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[2]/c[1]/c[1]/@id - Calames-2016114111127662	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	498:38
-	ead[1]/archdesc[1]/dsc[1]/c[1]/c[2]/c[1]/c[2]/@id - Calames-2016114111127663	2060-Maitron.xml	file:/C:/Users/feuret/Desktop	505:38

# Usage d'XPath dans XML Copy Editor



The image shows two instances of the XML Copy Editor. The left instance displays an XML document with a dialog box titled "Évaluer XPath" (Evaluate XPath) open. The dialog box has a text input field containing the XPath expression `//unitid[@type="cote"]/text()` and buttons for "OK" and "Cancel". A red arrow points from the dialog box to the right instance. The right instance shows a list of document numbers (607, 960, 962, 957, 684, 958, 866, 959, 692, 693, 739, 992, 1001, 688, 989, 687, 961, 791, 694-695, 689, 690, 691, 789, 755, 738, 683, 844, 848, 849, 850, 851) in a list view.

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

Results de la recherche

- 259-751059808\_006-Fonds Clifford Barney (inclus dans 245)
- 262-751059808\_007-Fonds Benichou(inclus dans 245)
- 440-751059808\_008\_Fonds Bergson(inclus dans 245)
- 263-751059808\_009-Ensemble Bertele(inclus dans 245)
- 248-751059808\_010\_Fonds Breton 1(inclus dans 245)
- 1603-751059808\_010\_Fonds Breton 2(lié à 248)
- 264-751059808\_011-Fonds Calet(inclus dans 245)
- 1588-751059808\_012\_Fonds Chapelan(inclus dans 245)
- 266-751059808\_013-Fonds Char (pour prod V1)(inclus dans 245)**

[eadheader]

[archdesc]

Propriétés du document

Selectionnez un modèle

- Natif
- EvalXPath**
- Visio\_controle
- Liste
- GFF
- Visio\_Html
- Kill-Id
- Natif-C
- Natif-Filtre
- MarcXml

Filtre :

Annuler Valider

Edition C:\Users\feurtet\Desktop\test206.xml - Notepad++

1 <?xml version="1.0" encoding="utf-8" ?>

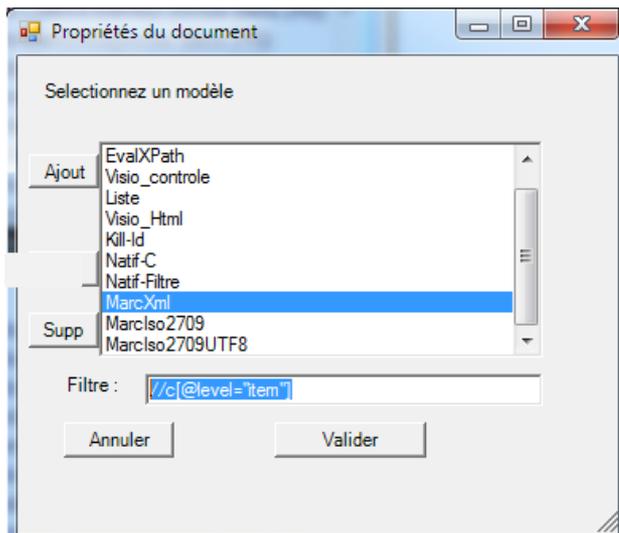
2 <RESULT>507 960 962 957 684 958 866 959 692 693 739 992 1001 688 989 687 961 791 694-695 689 690 691 789 755 738 899 717 723 722 9358 718 813 710 1002 990 993 1003 683 844 848 849 850 851 760 783 837 735 736 734 737 765 732 898 901 762 548 1004 756 757 758 759 760 763 761 1005 785 772-773 774 793 Alpha Ms 2369 896 772 775 Alpha Ms 2369-Alpha Ms 2373 ; Alpha Ms 2382 Alpha Ms 2370 Alpha Ms 2372 Alpha Ms 2382 Alpha Ms 2374 794 1006 Alpha Ms 2371 1008 838 1007 740 836 870 ; 900-901 929 795 834 1009 (1/2) 797 Alpha Ms 2367 835 1009 (2/2) 1010 1013 1011 1012 1014 1015 930 1016 1017 780-782 798 832 931 932 779 777-778 788 802 1021 811 811 1018 1019 801 1020 811 833 790 804 790 805 806 809 811 1022 811 811 823 820 1023 820 822 1024 821 1046 1025 822 1020 (1) 826 1026 (2) 1028 824 1027 828 829 830 1029 1031 (1) 1030 1031 (2) 1032 (1-2) 1200 1034 1035 769 913 784 768 858 819 764 867 856 852 853 720 807 808 875 784 (2) 873 847 784 (3) 784 817 846 784 (5) 1036 1037 784 (6) 800 719 868 724-726 727 728 729-730 818 869 770 770 (2) 770 (3) 902 786 910 379 380 894 851 995 685 851 991 845 767 776 831 754 827 696 702 697 1038 699 705 701 700 706 703 707 698 704 713 711-712 714 715 716 926 924 925 927 914 920 915 916 919 922 917 918 923 921 943 721 882 751 752 753 749 885 745 863 862 871 861 904 744 892 796 815 742 747 430-433 436-438 877 893 903 434-435 408-464 376 377 379 à 383 385-386 ; 409-410 Alpha Ms 2368 746 865 884 994 912 810 855 878 895 787 816 440 440 748 1040 857 1042 Alpha Ms 2373 771 891 814 1041 1039 879 825 686 812 1044 1045 928 1043 839 9335 842 906 840 841 935 947 637 9333 934 874 864 876 469 ; Ms Ms 27701-Ms Ms 27780 ; Ms Ms 27782-Ms Ms 27894 ; Ms Ms 886 1194-1196 708 (2) 413 996 Ms Ms 27701-Ms Ms 27780 ; Ms Ms 27782-Ms Ms 27894 ; Ms Ms 27897-Ms Ms 27903 ; Ms Ms 27905-Ms Ms 27918 ; Ms Ms 38870 872 887 881 860 803 743 880 1047-1171 1172-1192 ; 1197 883; 888-889 854 708 (1 ; 3) ; 819 ; 890 136-137 142 ; 421 144 145 146 151 152 923 157 158 388 161 91 264 2 4 5 160 170 172 173 Ms Ms 50786 Ms Ms 50621 (1-10) Alpha Ms 32537-Alpha Ms 32573 Ms Ms 50621 (13-17) ING 45 ING 1221 ING 1222 ING 1223 ING 1224 ING 1225 ING 1226 ING 1227 ING 1228 ING 1229 ING 1454 MO 86 </RESULT>

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...



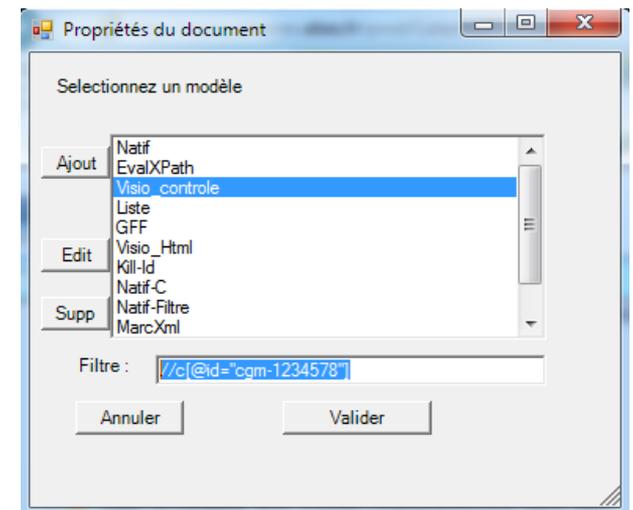
**Attention** : transformer les guillemets droits en deux simples quotes

" → ' + '



# 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

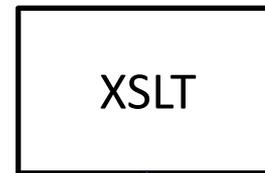
Pour quoi faire?

- **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  
...



.xsl



Oxygen, Saxon, Oracle, etc.  
(fonctions SQL/XML dans la  
base de données Calames)



XML,  
HTML...

# 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...

[dsc]

131-212319801-01a Dijon

- [eadheader]
- [archdesc]
  - [did]
  - [accessrestrict]
  - [userrestrict]
  - [prefercite]
  - [dsc]
    - [c] MS 1 MS 365 « La Coustume du duché
    - [c] MS 2 MS 178003 4091 « Notes sur l'O
    - [c] MS 3-4 MS 178001 4109 « Institutes c
    - [c] MS 5-6 MS 178004 7489 « Recueil d'a
    - [c] MS 7 MS 178002 61347 « Notes sur le

[c] [did] [unitid] MS 1 [unitid]

[unitid] MS 365 [unitid]

[unittitle] « La Coustume du duché de Bourgongne, corrigée par M [emph] e [emph] Estienne Cousin, advocat en Parlement et esleu du pays. 1621 » [unittitle]

[unitdate] XVII [emph] e [emph] siècle [unitdate]

[langmaterial] [language] Français [language] [langmaterial]

[physdesc]

[physfacet] Papier [physfacet]

[extent] 84 pages, table et 59 pages [extent]

[dimensions] 350 x 270 mm [dimensions]

[physfacet] Reliure parchemin [physfacet]

[physdesc] [did]

[controlaccess] [persname] Cousin, Etienne [persname] [title] La coutume du duché de Bourgogne [title]

[controlaccess]

[controlaccess] [geogname] Bourgogne [geogname] [subject] Coutume [subject] [controlaccess]

[controlaccess] [subject] Droit coutumier [subject] [geogname] Bourgogne [geogname] [controlaccess] [c]

[c] [did] [unitid] MS 2 [unitid]

[unitid] MS 178003 [unitid]

[unitid] 4091 [unitid]

[unittitle] « 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 M [emph] e [emph] [Jean] Melenet, avocat au Parlement de Dijon » [unittitle]

# 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 Bourgogne, 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 suite 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

The image displays a screenshot of an XSLT transformation process. On the left, a tree view shows the resulting HTML structure. The root is 'HTML', containing 'HEAD' and 'BODY'. The 'BODY' contains an 'h2' element 'Liste des composants du fichier', followed by a 'p' element 'Légende :'. The 'p' element contains a 'b' element 'Cote' and an 'i' element 'Intitulé'. Below this is a 'ul' element containing three 'li' elements, each with a 'b' element and a 'span' element. The 'span' elements have a 'color:chocolate' style. The 'ul' element is followed by another 'ul' element containing three 'li' elements, each with a 'b' element and a 'span' element. The 'span' elements have a 'color:chocolate; font-style:italic' style. The 'ul' element is followed by a final 'ul' element containing one 'li' element with a 'b' element and a 'span' element. The 'span' element has a 'color:chocolate; font-style:italic' style.

In the center, a file explorer shows the 'liste-essai.xml' file with a 'templates' folder containing a 'c' file. A red arrow points from the 'c' file to the 'span' elements in the HTML tree view.

On the right, the XSLT code is shown. The code is in French and defines an XSLT transformation. The root template is named 'f' and matches the root element. It outputs an HTML document with a title 'Liste des composants du fichier' and a body with a background color of 'Linen'. The body contains an 'h2' element 'Liste des composants du fichier', a 'p' element 'Légende : <b>Cote</b>. - <i>Intitulé</i>.', and an 'ul' element. The 'ul' element is generated by applying a template named 'c' to the root element. The 'c' template matches the root element and outputs an 'ul' element containing three 'li' elements, each with a 'b' element and a 'span' element. The 'span' elements have a 'color:chocolate' style. The 'ul' element is followed by another 'ul' element containing three 'li' elements, each with a 'b' element and a 'span' element. The 'span' elements have a 'color:chocolate; font-style:italic' style. The 'ul' element is followed by a final 'ul' element containing one 'li' element with a 'b' element and a 'span' element. The 'span' element has a 'color:chocolate; font-style:italic' style.

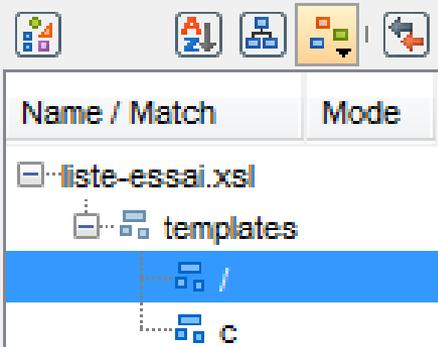
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl:output method="html" encoding="UTF-8" />
4 <xsl:template match="/">
5 <HTML>
6 <HEAD>
7 <TITLE>Liste des composants du fichier
8 <xsl:value-of select="ead/archdesc/did/unittitle" />
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/@identifiant" />
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.xml ». Dans le 1<sup>er</sup> template général (ciblant la racine), est reportée cette structure HTML

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

(Fichier XML/EAD source)

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

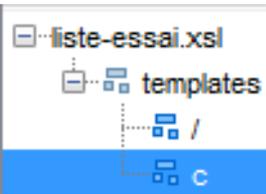


```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" ?>
3   <xsl:output method="html" encoding="UTF-8" ?>
4   <xsl:template match="/" ?>
5     <HTML ?>
6       <HEAD ?>
7         <TITLE>Liste des composants du fichier
8         <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" />
```

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 1<sup>er</sup> 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

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

2<sup>e</sup> 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.xml

# 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