# Cours « Document Structuré »
# Master TAL (pluriTAL.org)
# Travaux XML / XPATH / XSLT / XQUERY

## Table des matières

# Les outils

## Edition de documents XML

Vous pourrez utiliser n'importe quel très bon éditeur de texte, par exemple Notepad++ (et surtout pas WORD), ou bien des éditeurs XML.

A noter que Notepad++ dispose de plugins XML qu'il est possible d'installer et de tester…

Vous pourrez par exemple utiliser les outils suivants (liste non exhaustive) :

**XMLCooktop** (sous windows), **XML Copy Editor**, **EditX** (très bon éditeur, disponible à cette adresse http://www.editix.com/, version gratuite ici : http://free.editix.com/download.html )

**XMLSpy** (sur machine SF)

**Oxygen** (avec licence Inalco ou Paris 3) : toutes les fonctionnalités utiles pour ces TDs sont disponibles avec ce logiciel

Ou des éditeurs « classiques » comme **Notepad++** ou **Atom** qui permettent de traiter des documents XML en installant les plugins adéquats dans ces éditeurs.

## Validation

Pour vérifier vos documents (correction syntaxique et validation)

**RXP, AltovaXML** (sous windows)

**XMLCooktop** (sous windows)

**XML Copy Editor**

Etc.

## Requêtes XPath

**XmlCooktop** (sous windows)
**XML Copy Editor**
**BaseX** *(cf infra)*

## Transformation XSLT

**Un processeur en "ligne" : http://www.tal.univ-paris3.fr/plurital/outils/ProcXSLT-javascript/**
(SF fera une démo..)
**XMLCOOKTOP** (sous windows)
**AltovaXSLT** (sous windows)
**XML Copy Editor**

**Et xsltproc dans la ligne de commandes Unix**

## Requêtes XQUERY

**BaseX**

- **BaseX** est un gestionnaire de bases de données XML. Il gère des données nativement représentées en XML. Les requêtes s'écrivent en XQuery : un langage permettant l'intégration de requêtes XPath dans des structures de contrôle plus complexes.
- XQuery tutoriel : https://en.wikibooks.org/wiki/XQuery
- gratuit/multiplateforme (sous ubuntu présent dans les dépôts)
- la doc : http://docs.basex.org/wiki/Getting_Started
- Téléchargement ici : http://basex.org/products/download/all-downloads/

SF fera une démo au cours de la présentation de XQuery

## Le travail à rendre

L'objectif final de ce travail est de construire :

un site web regroupant uniquement des fichiers au format XML

(accompagnés de feuilles de styles XSLT)

Les contenus des pages de ce site étant la résolution des exercices présentés ci-dessous…

Le site final devra contenir au moins 6 exercices

Les exercices n°3, 5, 14, 15, 19… sont obligatoires

☺ Rien ne vous empêche de les faire tous !!!!! ☺

## Exercice 1 : Structurer l'information (vers une grammaire de document)

XML permet de structurer une information. Il est donc nécessaire, avant d'envisager d'utiliser ce format, de se familiariser avec cette structuration. Le paragraphe suivant contient de l'information "en vrac".

Réorganisez-la de manière à mettre en évidence sa structure logique, sans forcément passer par une mise en forme XML.

Une bouteille d'eau Cristaline de 150 cl contient par litre 71 mg d'ions positifs calcium, et 5,5 mg d'ions positifs magnésium. On y trouve également des ions négatifs comme des chlorures à 20 mg par litre et des nitrates avec 1 mg par litre. Elle est recueillie à **St-Cyr la Source**, dans le département du Loiret. Son code barre est `3274080005003` et son pH est de 7,45. Comme la bouteille est sale, quelques autres matériaux comme du fer s'y trouvent en suspension.

Une seconde bouteille d'eau Cristaline a été, elle, recueillie à la source d'**Aurèle** dans les Alpes Maritimes. La concentration en ions calcium est de 98 mg/l, et en ions magnésium de 4 mg/l. Il y a 3,6 mg/l d'ions chlorure et 2 mg/l de nitrates, pour un pH de 7,4. Le code barre de cette bouteille de 50 cl est `3268840001008`.

Une bouteille de même contenance est de marque Volvic, et a été puisée à... **Volvic**, bien connu pour ses sources donnant un pH neutre de 7. Elle comprend 11,5 mg/l d'ions calcium, 8,0 mg/l d'ions magnésium, 13,5 mg/l d'ions chlorures et 6,3 mg/l d'ions nitrates. Elle contient également des particules de silice. Son code barre est `3057640117008`.

PS : Volvic est dans le Puy-de-Dôme...

# Exercice 2 : DTD

## Phase 1

Rédiger une `DTD` pour une bibliographie. Cette bibliographie :

- contient des livres et des articles ;

Les informations nécessaires pour un livre sont :

- son titre général ;
- les noms des auteurs ;
- ses tomes et pour chaque tome, leur nombre de pages ;
- des informations générales sur son édition comme par exemple le nom de l'éditeur, le lieu d'édition, le lieu d'impression, son numéro ISBN ;

Les informations nécessaires pour un article sont :

- son titre ;
- les noms des auteurs ;
- ses références de publication : nom du journal, numéro des pages, année de publication et numéro du journal
- on réservera aussi un champ optionnel pour un avis personnel.

Tester cette `DTD` avec un fichier `XML` que l'on écrira *ex-nihilo* et validera.

## Phase 2

Modifier la `DTD` précédente...

- en ajoutant un attribut optionnel `soustitre` à l'élément `titre` ;
- en faisant de l'élément `tome` un élément vide et en lui ajoutant un attribut requis `nb_pages` et un attribut optionnel `soustitre` ;
- en faisant de l'élément `nom_journal` un attribut de l'élément `journal` et en lui donnant comme valeur par défaut `Feuille de Chou` ;
- en faisant de l'élément `annee` un attribut de type énuméré, prenant comme valeurs possibles 2000, 2001, 2002, "avant_2000" et "inconnue" et proposant comme valeur par défaut `inconnue`.

Utiliser cette `DTD` pour créer un fichier `XML` valide.

## Exercice 3 : TEI

⇨ **Voir le contenu de l'exercice TEI proposé par I. Galleron**

# Exercice 4 : Funeral Blues…

## Fichier de travail : Funeral Blues (tei)

**(disponible sur iCampus, dossier TP (ressources pour les tps, slides de présentation, corpus, outils...))**

Le texte **FuneralBlues.xml** contient un poème de W. H. Auden (1907-1973). Il est extrait en l'occurrence d'un recueil, *Tell Me the Truth about Love, ten poems by W. H. Auden*, publié en 1976 par Vintage Books, New York. Il est codé en XML en utilisant les conventions de la TEI

- Définir donc une feuille de style (appelée `TEI2TablePremiersVers.xsl`) qui, sur des textes respectant la proposition TEI, extrait le nom de l'auteur, du recueil, et le premier vers de chaque poème. Dans la feuille de style, dont le résultat est un document HTML, commenter en XML le fonctionnement global et les règles qui le nécessitent.

- Définir une feuille de style (appelée `NombreVers.xsl`) dont le résultat est un document HTML qui affiche le nombre de vers du poème.

# Exercice 5 : DUCHN (xml)

## Fichier de travail : duchn.xml (avec annotations morphosyntaxiques)

**(disponible sur iCampus, dossier TP (ressources pour les tps, slides de présentation, corpus, outils...))**

Le texte **duch.xml** contient le texte du Père Duchesne :

***Le Père Duchesne*** *est le titre de différents journaux qui ont paru sous plusieurs plumes durant la Révolution française. Le plus populaire était celui de Jacques-René Hébert, qui en a fait paraître 385 numéros de septembre 1790 jusqu'à onze jours avant sa mort à la guillotine, survenue le 4 germinal An II (24 mars 1794).*

Cf https://fr.wikipedia.org/wiki/Le_P%C3%A8re_Duchesne_(R%C3%A9volution_fran%C3%A7aise)

- Définir une feuille de style dont le résultat est un document HTML et donnant à voir uniquement le texte. On essaiera de formater au mieux l'affichage (séparer les mots… saut de ligne après paragraphe…)
- Définir une feuille de style dont le résultat est un document HTML et donnant à voir uniquement le texte en surlignant les mots dont le lemme est « aristocrate » ou « patriote » (stylage à définir par vous pour marquer le soulignement visé).
- Définir une feuille de style dont le résultat est un document TXT contenant la liste des mots contenant la séquence « citoy », puis la séquence « patriot » ; compter aussi la fréquence de chacun des mots
- Définir une feuille de style dont le résultat est un document TXT contenant la liste des catégories ; compter aussi la fréquence de chacune
- Définir une feuille de style permettant d'afficher le texte en surlignant les mots de catégories NOM ou ADJ (stylage à définir par vous pour marquer le soulignement visé).
- Définir une feuille de style dont le résultat est un document TXT contenant la liste des séquences NOM ADJ
- Idem avec NOM PREP NOM

# Exercice 6 : Transcription de l'ORAL (corpus CFPP2000)

Le **C**orpus de **F**rançais **P**arlé **P**arisien (**CFPP2000**) est composé d'un ensemble d'interviews sur les quartiers de Paris et de la proche banlieue.

Présentation détaillée : http://cfpp2000.univ-paris3.fr/CFPP2000.pdf

Les données à traiter sont disponibles ici :

http://cfpp2000.univ-paris3.fr/Corpus.html#transcriptionTEI

On trouvera derrière ce lien l'ensemble des transcriptions du corpus (initialement au format *Transcriber* et converties ici) au format TEI.

Ces fichiers ont été construits par ce programme :

http://modyco.inist.fr/transcriberjs/doku.php?id=public:formattei

La page « corpus nouvelles transcriptions CFPP2000 » est une page contenant des données en cours de mise à jour pour le corpus CFPP2000.

Le travail réalisé ici sera utilisé pour enrichir

ce corpus et ses outils.

**Références :**

- Site officiel TEI
- Proposition de format TEI/ISO pour l'oral
- Spécifications utilisées aujourd'hui par Ortolang et TranscriberJS
- Site IRCOM GT2 et les discussions des formats

## Partie 1

Pour chacun des documents :

- Vérifier/Compléter les transcriptions au regard des recommandations utilisées par Ortolang (cf supra pour le lien) :
- Par exemple : la balise `media` devra porter un attribut `type` spécifiant le type d'enregistrement et un attribut `mimeType` spécifiant le type mime du média (ici `mimeType="audio/wav"` pour toutes les transcriptions)

- Vérifier que le document est valide

## Partie 2 :

- Créer une feuille de styles XSLT permettant d'afficher les tours de parole de `spk1` (en affichant aussi le nom associé à `spk1`)
- Créer une feuille de styles XSLT permettant d'afficher uniquement tous les tours de parole (en affichant aussi le nom associé à chaque intervenant), chaque intervenant devra être affiché dans un couleur différente
- Créer une feuille de styles XSLT permettant d'afficher tous les tours de parole (en affichant aussi le nom associé à chaque intervenant), chaque intervenant devra être affiché dans un couleur différente. On ajoutera aussi les intervalles temporels de ces tours de parole

## Partie 3 :

En s'inspirant de la page « *Recherche dans les textes des transcriptions de CFPP2000* » accessible ici : http://cfpp2000.univ-paris3.fr/search-transcription/index2.html

**Objectif** : Construire une interface permettant de rechercher des occurrences de mot dans l'ensemble des transcriptions au format TEI.

- On commencera par regarder le code *javascript* contenu dans la page « Recherche (regexp) ... »
- Ce code permet d'interroger un fichier XML regroupant toutes les transcriptions CFPP2000 au format *Transcriber*.
- Il s'agit donc d'adapter ce code pour permettre de faire le même genre de travail sur l'ensemble des transcriptions au format TEI
- Pour le module de recherche, on ne gardera dans un premier temps que la zone de recherche dite « **Recherche à la google »**

(SF vous aidera…)

# Exercice 7 : Exploration d'un fichier TEI

## Fichier de travail : le dormeur du val (tei)

**(disponible sur iCampus, dossier TP (ressources pour les tps, slides de présentation, corpus, outils...))**

Le fichier de travail a l'allure suivante :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="application/xml" href="AuteurRecueilTitrePoemeHTML.xsl"?>
<TEI.2>
<teiHeader>
<fileDesc>
<titleStmt>
<title>Poésies</title>
<author>Arthur Rimbaud</author>
</titleStmt>
</fileDesc>
</teiHeader>
<text>
<group>
<text>
<front>
<head>Le Dormeur du Val</head>
<dateline>7 octobre 1870</dateline>
</front>
<body>
...
```

Chacune des questions suivantes doit conduire à construire une (voire plusieurs) feuille de styles pour réaliser les traitements visés :

- Afficher le nom de l'auteur, le titre du poème, la date, et le recueil, dans cet ordre et au sein d'une page HTML bien formée.
- Afficher le titre et la date du poème, les vers les uns après les autres, sans balise (sortie texte brut)
- Afficher le 1er et le 3ème vers de chaque groupe de vers au sein d'une page HTML bien formée.
- Afficher seulement le 1er et le 3ème quatrain au sein d'une page HTML bien formée.
- Afficher chaque vers précédé de son numéro d'ordre au sein du groupe de vers (<lg>).
- Idem en ajoutant à chaque groupe de vers son numéro d'ordre.
- Engendrer une page HTML bien formée dans laquelle le poème (seul, sans titre) est au sein d'un tableau, avec une ligne par vers, et une ligne entre chaque strophe. Penser à aligner les lignes à gauche.
- Afficher les vers qui contiennent un point-virgule ou deux points (:) (format texte brut)
- Afficher les quatrains en gras et les tercets en italiques au sein d'une page HTML bien formée.

# Exercice 8 : « Regrouper des éléments »

## Fichier de travail : les infirmières du corpus Prématurés00

**(disponible sur iCampus, dossier TP (ressources pour les tps, slides de présentation, corpus, outils...))**

Le fichier à traiter provient du corpus **Prématurés00** : les infirmières (triées par ancienneté).

On veut désormais factoriser les informations sur les infirmières relevant d'une même classe d'ancienneté.

On utilise alors la possibilité avec XSLT d'examiner les frères aînés d'un nœud donné (de la même manière qu'on peut examiner les ascendants, les descendants).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="iso-8859-1"/>
<xsl:template match="/table">
  <html>
  <body>
  <h1 align="center">Infirmières</h1>
  <xsl:for-each select="ligne">
    <xsl:if test="not(preceding-
sibling::ligne/classe_anciennete=./classe_anciennete)">
      <h2><xsl:value-of select="./classe_anciennete"/></h2>
    </xsl:if>
  </xsl:for-each>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Le résultat est le suivant :

```
<html><body>
<h1 align="center">Infirmières</h1>
<h2>entre 1 et 5 ans</h2>
<h2>moins de 1 an</h2>
<h2>plus de 5 ans</h2>
</body></html>
```

Compléter la feuille de style ci-dessus pour que le résultat soit un tableau HTML avec un titre occupant toute la largeur du tableau pour chaque classe d'ancienneté, un titre pour chaque colonne et une ligne pour chaque infirmière.

Le résultat sera de la forme :

| *Numéro* | *Age* | *années d'étude* | *diplôme* | *ancienneté* | *service* |
|---|---|---|---|---|---|
| entre 1 et 5 ans | | | | | |
| 2 | 29 | 3 | BEPC | 2.50 | Jour |
| 9 | 27 | 3 | BEPC | 3.00 | Jour |
| 13 | 34 | 3 | BEPC | 3.00 | Jour |
| 18 | 27 | 4 | BEPC | 2.50 | Jour |
| 36 | 26 | 4 | BEPC | 2.00 | Jour |
| 39 | 33 | 4 | BEPC | 3.00 | Jour |
| 43 | 26 | 4 | BEPC | 2.00 | Jour |
| 61 | 26 | 4 | BEPC | 3.00 | Jour |
| 62 | 33 | 3 | BEPC | 1.00 | Jour |
| 44 | 29 | 3 | BEPC | 2.00 | Nuit |
| 47 | 30 | 3 | BEPC | 3.00 | Nuit |
| 73 | 29 | 3 | BAC | 1.50 | Jour |
| moins de 1 an | | | | | |
| 41 | 26 | 3 | BEPC | 0.00 | Non connu |
| 24 | 26 | 4 | BEPC | 0.00 | Jour |
| 32 | 26 | 3 | BEPC | 0.00 | Jour |
| 34 | 31 | 4 | BEPC | 0.00 | Jour |
| 46 | 21 | 3 | BEPC | 0.00 | Jour |
| 65 | 21 | 3 | BEPC | 0.50 | Jour |
| 67 | 25 | 4 | BEPC | 0.00 | Jour |
| 68 | 24 | 4 | BEPC | 0.00 | Jour |
| 70 | 29 | 3 | BEPC | 0.00 | Jour |
| 81 | 25 | 3 | BEPC | 0.00 | Jour |
| 97 | 25 | 3 | BEPC | 0.00 | Jour |
| 1 | 23 | 4 | BEPC | 0.00 | Nuit |
| 33 | 34 | 4 | BAC | 0.00 | Jour |

# Exercice 9 : Exploration d'une recommandation W3C taggée

## Fichier de travail : recommandation XSLT du W3C taggée

**(disponible sur iCampus, dossier TP (ressources pour les tps, slides de présentation, corpus, outils...))**

## Phase 1 : document et grammaire

- Modifier l'entête du document en y insérant vos références personnelles (nom, prénom, email, etc.)
- Vérifier la bonne formation du document
- Créer la DTD et établir le lien à la DTD
- Vérifier que le document est valide

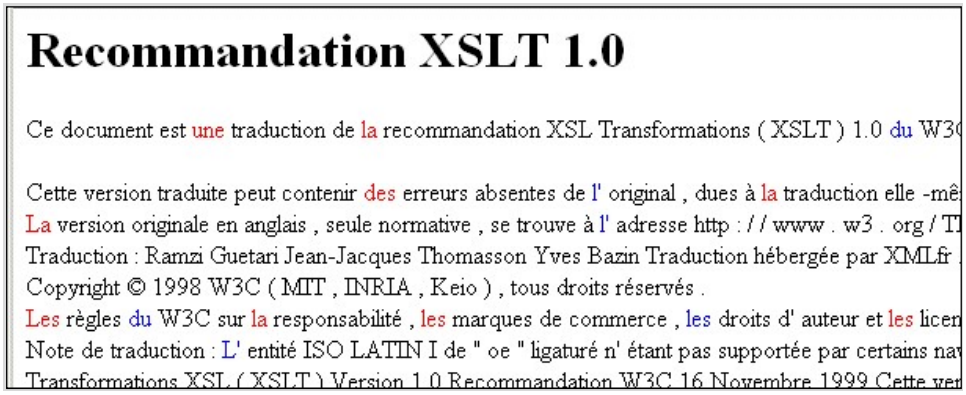
## Phase 2 : définir des requêtes xpath

- Rechercher tous les NOM
- Rechercher tous les VERBE
- Rechercher tous les NOM précédés d'un DET
- Rechercher toutes les séquences NOM ADJ ou ADJ NOM
- Rechercher tous les NOM précédés d'un DET

## Phase 3 : transformation xslt

- Définir une feuille de styles minimale pour formater le document XML fourni en HTML.
- Définir une feuille de styles de telle sorte que les déterminants au féminin apparaîtront en rouge et les déterminants masculins en bleu

**Sortie**




Définir une feuille de styles de telle sorte :

- nom des éléments = catégorie grammaticale
- valeur des éléments = forme des tokens

**Sortie**

```
<doc>
   <taggersent>
      <DET>Ce</DET>
      <NOM>document</NOM>
      <VER>est</VER>
      <DET>une</DET>
      ...
      <NOM>novembre</NOM>
      <NUM>1999</NUM>
      <PUN>.</PUN>
   </taggersent>
   <taggersent>
      <DET>Cette</DET>
      <NOM>version</NOM>
      ...
      <ADJ>-même</ADJ>
      <PUN>.</PUN>
   </taggersent>
    ...
</doc>
```

**Un peu plus loin...**

```
<doc>
   <taggersent>
      <DET
gram="DEM:masc:sg">Ce</DET>
      <NOM
gram="masc:sg">document</NOM>
      <VER gram="sg">est</VER>
      <DET gram="femi:sg">une</DET>
      ...
      <NOM
gram="masc:sg">novembre</NOM>
      <NUM>1999</NUM>
      <PUN>.</PUN>
   </taggersent>
   <taggersent>
      <DET
gram="DEM:femi:sg">Cette</DET>
      <NOM
gram="femi:sg">version</NOM>
      ...
      <ADJ gram="femi:sg">-même</ADJ>
      <PUN>.</PUN>
   </taggersent>
   ...
</doc>
```

# Exercice 10 : Fichier clientèle Air France

## Fichier de travail : Air France

**(disponible sur iCampus, dossier TP (ressources pour les tps, slides de présentation, corpus, outils...))**

## Phase 1 : document et grammaire

- Modifier l'entête du document en y insérant vos références personnelles (nom, prénom, email, etc.)
- Vérifier la bonne formation du document
- Créer la DTD et établir le lien à la DTD

## Phase 2 : définir des requêtes xpath

- Rechercher les uttérances (associées aux élements <u>) de l'opérateur
- Rechercher les uttérances (associées aux élements <u>) du client
- Rechercher les uttérances (associées aux élements <u>) qui contiennent le mot "quand"
- Rechercher les uttérances (associées aux élements <u>) qui contiennent le le caractère '?'
- Rechercher les uttérances (associées aux élements <u>) qui précèdent celles qui contiennent le le caractère '?'

## Phase 3 : transformation xslt

**Etape 1 :**

- Définir une feuille de styles minimale pour formater le document XML fourni en HTML
- Définir une feuille de styles permettant d'afficher les uttérances (associées aux élements <u>) qui contiennent le mot "quand", avec :
  - uttérance précédente
  - uttérance suivante
  - contexte gauche (éventuellement tronqué)
  - contexte droit (éventuellement tronqué)

**Sortie XML**

```
<list>
   <item>
      <id com="COMMUNICATION I-4" n="38" who="O"/>
      <prev>oui</prev>
      <next>oui</next>
      <left>toutefois si vous voulez une une : somme exacte pour votre
remboursement je vous conseillerai</left>
      <right>même</right>
   </item>
   <item>
      <id com="COMMUNICATION I-11" n="7" who="O"/>
      <prev>début février</prev>
      <next>alors là justement c'est la deuxième question</next>
      <left>oui et votre retour aura lieu</left>
      <right/>
   </item>
    ...
</list>
```

**Sortie HTML**

COMMUNICATION I-13, n = 9, locuteur = C

*Précédent : attendez excusez - moi vous êtes intéressée par quel par quel vol*

donc ben je prends 17h quand même

*Suivant : vol visite*

COMMUNICATION I-13, n = 11, locuteur = O

*Précédent : c'est en classe b alors*

en classe b c'est bien ça et le retour aurait lieu quand

*Suivant : alors le retour a lieu le 2 février sur Air France 681*

COMMUNICATION I-22, n = 20, locuteur = C

*Précédent : voilà la correspondance que je vous ai proposée qui bien sûr prend pas mal de temps à Moscou*

oui quand on sait la : la température qu'il y fait cela s'annonce bien effectiven

*Suivant : remarquez je pense qu'ils chauffent l'aéroport*

COMMUNICATION I-27, n = 40, locuteur = O

*Précédent : ah bon d'accord*

ça fait quand même beaucoup moins cher

*Suivant : d'accord*

COMMUNICATION I-31, n = 4, locuteur = O

*Précédent : oui*

oui vous désirez partir quand

*Suivant : ben jeudi par exemple*

COMMUNICATION I-32, n = 39, locuteur = C

*Précédent : voilà*

quand quelque chose est en ordre [dit très vite] c'est parfaitmerci beauc

Remarque : on peut gérer (ou pas) la présence éventuelle de plusieurs "quand" dans une même uttérance

## Etape 2 :

Définir une feuille de style permettant d'obtenir une copie à l'identique du fichier de départ avec marquage des occurrences de "ah", "ah bon", "ah oui" :

soit : `<interj form='ah'/>, <interj form='ah bon'/> , ...`

soit : `<interj>ah</interj> <interj>ah bon</interj>`

**Sortie**

```
<u who="C" n="5">si <interj>ah bon</interj> ben il n'y a que vous
alors il n'y a que vous alors</u>
...
<u who="O" n="5">tarif vacances .... <interj>ah</interj></u>
```

# Exercice 11 : Collections du musée des Augustins – ville de Toulouse

La quasi-totalité des collections conservées au musée des Augustins est accessible par moteur de recherche à cette adresse :

http://www.augustins.org/les-collections/documentation/base-de-donnees

L'URL suivante :

https://data.toulouse-metropole.fr/explore/dataset/collections-du-musee-des-augustins-ville-de-toulouse/

donne accès à l'inventaire des œuvres acquises par le musée des Augustins depuis sa création en 1793. On trouvera sur ce site différents formats pour cet inventaire. On récupèrera notamment le fichier au format **JSON** (https://data.toulouse-metropole.fr/explore/dataset/collections-du-musee-des-augustins-ville-de-toulouse/download/?format=json&timezone=Europe/Berlin)

**Vous trouverez sur iCampus une version de cet inventaire au format XML (augustins.xml).**

1. Choisir quelques entrées communes à l'inventaire au format XML et au même inventaire au format **JSON**.
    - Faire une brève présentation de **JSON** (*cf* wikipédia par exemple)
    - Comparez les différentes entrées en parallèle.
2. A partir de l'inventaire fourni au format XML, construire sa DTD.
3. A partir de l'inventaire fourni au format XML, construire une feuille de styles XSLT minimale pour afficher « proprement » le contenu de l'inventaire au format HTML en classant les œuvres par auteur.
4. Construire une feuille de styles permettant de n'afficher que les références des œuvres (n° d'inventaire par exemple) et l'auteur
5. A partir de la feuille de styles précédente, ajoutez une fonctionnalité permettant d'afficher le contenu complet d'une seule entrée (en cliquant sur l'entrée par exemple)

    ➔ démo en cours par SF

    - On pourra par exemple s'inspirer de la solution mise en œuvre sur la page Mémoires du site plurital.org (via une fonctionnalité *javascript*)
6. Construire un programme (perl, python…) permettant de passer du format JSON disponible en ligne ci-dessus vers un fichier au format XML pour produire un nouvel état de l'inventaire au format XML

# Exercice 12 : Petit projet « un conte… »

## Fichier de travail : un conte à votre façon

**(disponible sur iCampus, dossier TP (ressources pour les tps, slides de présentation, corpus, outils...))**

Le conte de Raymond Queneau « un conte à votre façon » (*cf* lien supra) est à l'origine des romans dont vous êtes le héros. Il offre de multiples parcours.

Proposer une structure de document XML pour représenter ce conte et le passage d'un endroit du conte à un autre.

Essayer de construite une feuille de style permettant de réaliser, à partir du document XML obéissant à la structure choisi, un document HTML permettant la navigation effective d'un endroit du conte à l'autre.

# Exercice 13 : Corpus d'alignement textuel

## Partie 1

### Fichiers de travail

- lepromose-lapromesse.xml
- lepromise.html
- lapromesse.html
- la DTD et le schéma transread

**Ces fichiers sont disponibles dans une archive sous iCampus.**

### Lectures

Ces ressources ont été récupérées sur le site du projet TRANSREAD : https://transread.limsi.fr/

- **Question 1** : décrire brièvement les données à traiter et le rôle de chacun des fichiers
- **Question 2** : à partir du fichier lepromose-lapromesse.xml, écrire une feuille de style XSLT permettant d'afficher dans un tableau HTML l'alignement des 2 volets (une ligne du tableau devant contenir 2 segments alignés)
- **Question 3** : à partir du fichier lepromose-lapromesse.xml, écrire une feuille de style XSLT permettant d'afficher uniquement le volet français (idem pour le volet anglais)

## Partie 2

### Fichiers de travail

- L'archive `SentenceAlignmentConfidenceAnnotation.tgz` disponible sur iCampus
- Cf A collection of confidence annotations of sentence alignment links - see the README file.

- **Question 1** : décrire brièvement les données à traiter et le rôle de chacun des fichiers
- **Question 2** : à partir des 3 fichiers 3000-3599.txt, 3600-4199.txt, 4200-4799.txt, proposez une structure XML (une grammaire donc) pour représenter les informations contenues dans ces fichiers. On pourra automatiser la construction du fichier XML final via un script Perl ou Python.
- **Question 3** : à partir du ou des fichiers XML produits à la question 2, construire une feuille de styles XSLT pour afficher en parallèle les 2 volets de l'alignement (EN vs FR)

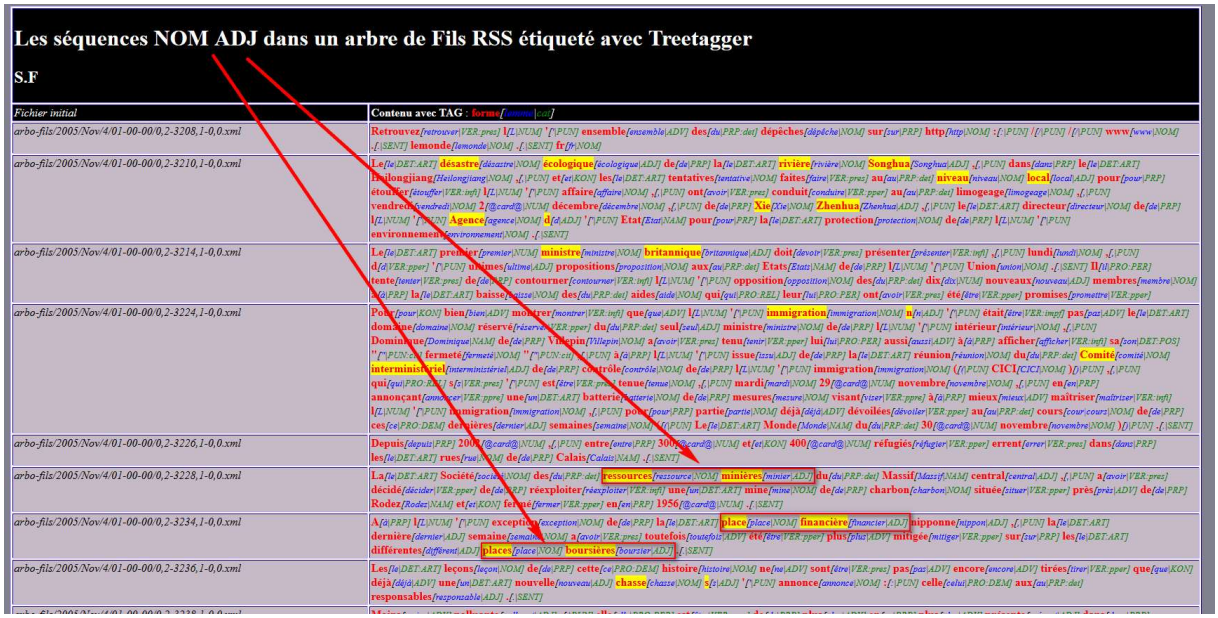
# Exercice 14 : Extraction dans les Fils RSS du Monde

## 1. Fichiers de travail : rubrique « A la Une » 2017 étiquetée avec treetagger

**Ce fichier est disponible dans une archive sous iCampus : 3208-2017-v2.zip**

**XSLT**

1. Construire une feuille de styles pour afficher les contenus étiquetés avec *TreeTagger*
2. Construire des feuilles de style pour extraire (au format TXT) au moins 3 patrons morpho-syntaxiques (de longueurs différentes) sur les mêmes fichiers
3. Construire une feuille de styles pour afficher les contenus étiquetés avec *TreeTagger* en mettant au jour un patron morphosyntaxique. Ci-dessous le patron NOM ADJ :




**XQuery** (via BASEX)

1. Construire une requête pour extraire les contenus textuels des titres
2. Construire une requête pour extraire les contenus textuels des descriptions
3. Construire une requête pour extraire les contenus textuels des titres ET des descriptions
4. Construire une requête pour extraire les patrons morpho-syntaxiques NOM ADJ
5. Construire une requête pour extraire les patrons morpho-syntaxiques NOM PRP NOM

## 2. Fichiers de travail : fichiers étiquetés avec udpipe dans le cours Projet Encadré

# On utilisera ici les fils RSS de l'année 2021 étiquetés avec udpipe dans le cours *Projet Encadré*.

A priori, vous aurez construit un fichier par rubrique et ce fichier sera étiqueté via udpipe, ce fichier sera ensuite reformaté en XML (cf ressources fournies par SF)

**Exemple : une archive quasi similaire « base-talismane-pour-cours-xml.zip » (disponible sur iCampus)**

L'archive précédente contient 3 fichiers construits en 2018 avec un étiquetage via Talismane (même genre de sortie que pour udpipe)

- L'étiquetage produit via Talismane sur la rubrique « A la Une » de l'année 2017
- Un script perl transformant le fichier précédent au format XML
- Le fichier XML produit par le script précédent

C'est donc à partir du 3ème fichier que l'on pourrait travailler. Il conviendra de générer vos propres fichiers XML pour chacune des rubriques à traiter.

**XSLT**

1. Construire une feuille de styles pour afficher les contenus étiquetés des « titres » uniquement
2. Peut-on envisager de construire une feuille de styles pour afficher chaque « titre » en regard de sa « description » : par exemple un tableau avec 2 colonnes, à gauche les titres et à droite les descriptions (un alignement en somme) ? si cela n'est pas possible directement, quel est l'alignement minimal que l'on peut mettre en œuvre ? Construire cet alignement minimal…
3. Construire des feuilles de style pour extraire (au format TXT) au moins 3 patrons morpho-syntaxiques (de longueurs différentes) sur les « titres » uniquement puis sur les « descriptions » uniquement
4. Construire une feuille de styles pour afficher les contenus étiquetés en mettant au jour un patron morphosyntaxique sur les « titres » uniquement puis sur les « descriptions » uniquement
5. Construire une feuille de styles pour extraire (au format TXT) les items en relation de dépendance syntaxique de type OBJ (les classer et les compter). Essayer de tenir compte de la différence titre/description
6. Construire une feuille de styles pour extraire la liste des POS en relation de dépendance syntaxique de type OBJ (les classer et les compter)

**XQuery** (via BASEX)

1. Construire une requête pour compter le nombre de titres, le nombre de descriptions.
2. Construire une requête pour extraire les contenus textuels des « titres »
3. Construire une requête pour extraire les contenus textuels des « descriptions »
4. Construire une requête pour extraire les contenus textuels des titres ET des descriptions

5. Construire une requête pour extraire au moins 3 patrons morpho-syntaxiques (de longueurs différentes)
6. Construire une requête pour extraire les items portant une relation de dépendance syntaxique de type OBJ
7. Construire une requête pour extraire les items en relation de dépendance syntaxique de type SUB (on affichera les formes ou les lemmes connectés par cette relation)

# Exercice 15 : Patrons morphosyntaxiques / Relations de dépendance sur le Base TrameurFromRhapsodie

## Fichier de travail : la base TrameurFromRhapsodie

**(disponible sur iCampus, dossier TP (ressources pour les tps, slides de présentation, corpus, outils...))**

**Présentation de la base (*cf* le site du Trameur)**

**Base "Rhapsodie2Trameur"** construite à partir des ressources développées dans le cadre du projet **Rhapsodie**.

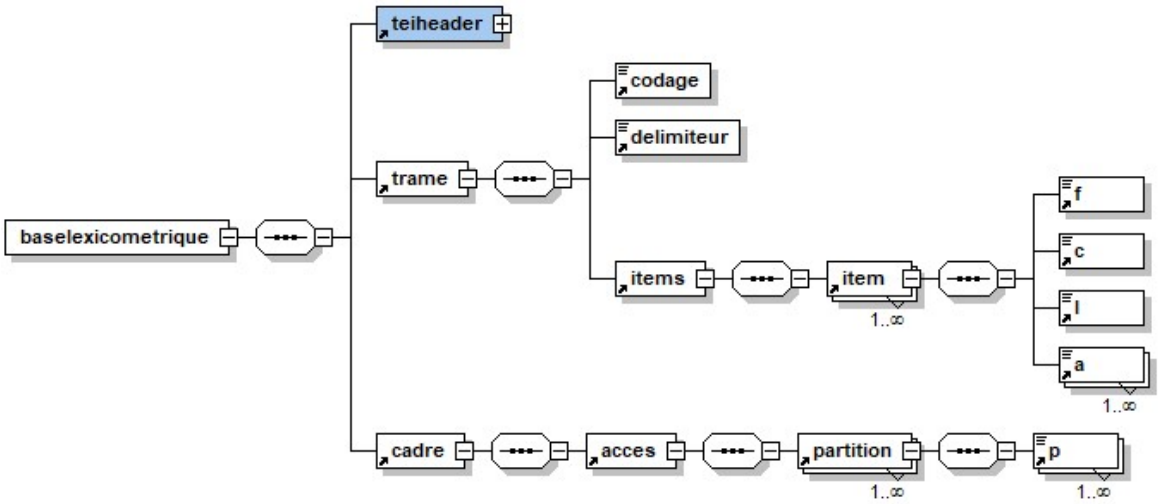
**SOURCES** : projet Rhapsodie, http://projet-rhapsodie.fr/

**Descriptif et sources des annotations** : **Annotations Rhapsodie pour le Trameur (v8)**[1] (pdf)
Ce document contient la présentation du processus de transcodage des annotations Rhapsodie pour construire une base textométrique et des différents processus de traitements des annotations de dépendance avec Le Trameur.

L'archive disponible sur icampus contenant la base contient aussi un petit fichier README.

La base est au format XML et son schéma d'organisation des données et le suivant :

[1] http://www.tal.univ-paris3.fr/trameur/bases/rhapsodie2trameur-v8.pdf




SF fera une présentation en cours.

**Travail à réaliser :**

**XSLT**

1. Construire une feuille de styles pour afficher uniquement le texte (sortie TXT)
2. Construire une feuille de styles pour afficher (au format HTML) le texte en insérant sur chaque item sa POS
3. Construire une feuille de styles pour extraire (en format TXT) au moins 2 patrons morpho-syntaxiques de longueurs différentes
4. Construire une feuille de styles pour extraire (au format TXT) les items en relation de dépendance syntaxique de type SUJET (les classer et les compter)
5. Construire une feuille de styles pour extraire la liste des POS en relation de dépendance syntaxique de type SUJET (les classer et les compter)

**Questions subsidiaires :**

Construire une feuille de styles pour afficher (au format HTML) tout le texte de la base en mettant au jour une relation de dépendance syntaxique choisie (par exemple OBJ) : l'affichage doit mettre au jour le dépendant et le gouverneur

**XQuery**

(via BASEX)

1. Construire une requête pour extraire les items portant une relation de dépendance syntaxique de type OBJ

2. Construire une requête pour extraire les items en relation de dépendance syntaxique de type OBJ (on affichera les formes ou les lemmes connectés par cette relation)

# Exercice 16 : le corpus ECRISCOL

Source : http://syled.univ-paris3.fr/ecriscol/CORPUS-TEST/index.html

Le projet de recherche ECRISCOL est centré sur l'analyse des écrits produits en situation scolaire. Il s'agit de faire le rapport entre des traits caractéristiques de ces écrits et des situations d'apprentissage et d'enseignement suscitant leur production, de manière à faire apparaître des dispositifs didactiques favorisant certains types d'écrits et certaines stratégies ou procédures d'écriture. Les données (les copies d'élèves) sont accessibles par niveau d'étude. Sous chaque niveau, on trouvera des liens donnant à voir l'ensemble des devoirs d'une classe donnée (avec pour certains devoirs, différentes versions du même devoir). Chacune des copies est visible avec les différentes opérations de réécriture transcrites (et les corrections réalisées) et en regard du manuscrit original (avec la possibilité de zoomer sur ce manuscrit original).

Chaque copie est associée à un fichier au format XML (TEI).

Les copies du corpus ECRISCOL au format XML/TEI sont disponibles dans cette archive zippée :

http://syled.univ-paris3.fr/ecriscol/BASE-X/ARCHIVE-BASEX.zip

(Présentation en cours par SF)

- Dézippez cette archive dans un dossier
- Importer toutes les copies de ce dossier dans une base que vous nommerez ECRISCOL (copies présentes dans chaque dossier TEI de l'archive)

**Travail à réaliser :**

## Requête XPath simple

1. Afficher toutes les opérations de suppression par niveaux d'étude (balise `mod` avec attribut `type` ayant la valeur « `del` »)
2. Afficher toutes les opérations d'insertion par niveaux d'étude (balise `mod` avec attribut `type` ayant la valeur « `add` »)
3. Afficher toutes les opérations de remplacement par niveaux d'étude (balise `mod` avec attribut `type` ayant la valeur « `subst` »)
4. Afficher tous les commentaires des enseignants par niveaux d'étude (balise metamark)
5. Afficher toutes les corrections orthographiques par niveaux d'étude (balise `mod` avec attribut `type` ayant la valeur « `corr` »)
6. Afficher tous les soulignements par niveaux d'étude (balise `hi`)
7. Afficher les commentaires des enseignants précédés ou suivis par un soulignement

## Requête XQuery

1. Compter le nombre de copies par niveaux d'étude
    - On pourra commencer par compter le nombre de copies globalement
2. Compter le nombre d'opérations de réécriture par type et par niveaux d'étude
    - On pourra commencer par compter le nombre d'opérations de réécriture par copie
3. Compter le nombre de corrections orthographiques par niveaux d'étude
    - Compter aussi les différents types de corrections (voir l'attribut cat associé à une balise mod ayant un attribut type ayant pour valeur « corr »)
4. Compter le nombre de commentaires de l'enseignant par niveaux d'étude

# Exercice 17 : un dictionnaire

Source : http://infolingu.univ-mlv.fr/DonneesLinguistiques/Dictionnaires/telechargement.html

## Fichier de travail : le fichier dela-fr-public-u8.dic.xml

**(disponible sur iCampus, dossier TP (ressources pour les tps, slides de présentation, corpus, outils...))**

## Requête XPath simple

On travaillera de préférence avec BaseX

- Compter le nombre d'entrée du dictionnaire
- Compter les différents types de catégories disponibles dans le fichier

## Feuille de styles XSLT

1. Construire une feuille de styles XSLT pour extraire les 200 éléments entry à partir du 2000ième (les éléments 2001 à 2200). En sortie : un fichier XML avec le contenu initial complet des entry visées.
2. Construire une feuille de style permettant d'extraire une liste de toutes les formes de tous les verbes du dictionnaire. En sortie un fichier TXT.

## Requête XQuery

1. Compter le nombre de verbes, de noms etc.

## Exercice 18 : Un peu de contrepèteries !

Les contrepèteries sont un jeu de langage où une séquence est produite à partir d'une autre en intervertissant une ou plusieurs sous-séquences.
Exemples :

1. Le vantard s'épile / le vampire s'étale
2. La vie des mots / L'ami des veaux
3. Pauline est coquette / Paulette est coquine
4. un mot de vous / un mou de veau
5. L'opéra / L'apéro
6. Un chauvin / un vin chaud
7. des balades dans l'arène / des baleines dans la rade
8. de beaux hôtels / de belles autos
9. Laurent est charmeur / Roland est marcheur
10. Jacques est en Iran / Jean est en Irak

La contrepèterie ne se soucie pas de l'orthographe (les sous-séquences échangées peuvent avoir des orthographes différentes).

1 - Vous choisirez une représentation XML (dans un fichier nommé contrepeteries.xml) qui maintienne le lien entre les deux versants de la contrepèterie.

Vous fournirez les 10 exemples ci-dessus selon la représentation que vous aurez choisie.

2 - Vous définirez deux feuilles de style dont le résultat est un document HTML :

1. la première (ContrepeteriesVersant1a.xsl) extrait le versant gauche de la contrepètrie ;
2. la seconde (ContrepeteriesVersant1b.xsl) la partie droite.

Les 2 versants sont numérotés (de manière à ce que les numéros correspondent d'un versant à l'autre).

3 - A partir de ces feuilles de style, faites une seule feuille de style (ContrepeteriesVersant2.xsl) avec un lien, pour chaque contrepèterie du versant gauche vers le versant droit correspondant. On se facilitera la vie en ajoutant un identifiant unique à chaque contrepèterie et en s'en servant.

4 - Quelle représentation XML imaginer pour avoir un lien plus fin entre les deux versants de la contrepèterie, en figurant clairement le lien entre les séquences échangées.

# Exercice 19 : Ulysse pour finir

On commencera par récupérer le corpus du travail : **ULYSSE** à cette adresse :

http://www.tal.univ-paris3.fr/mkAlign/corpus/corpus-ulysse/ulysse.xml

**ATTENTION** : il faut récupérer le fichier XML original, si pb, vous me dites et je vous l'envoie.

On lira aussi le descriptif de ce corpus...

## Partie 1 : XSLT

- Ecrire une feuille de styles XSLT pour extraire uniquement le volet anglais de cet alignement (au format txt)
- Ecrire une feuille de styles XSLT pour extraire uniquement le volet français de cet alignement (au format txt)
- A partir des commandes précédentes, écrire une « commande » pour compter le nombre de mots dans chacun des 2 volets de cet alignement
- Après avoir choisi un "mot" anglais ou français, écrire une feuille de styles XSLT pour extraire tous les bi-contextes contenant ce mot (le contexte contenant le mot choisi et sa traduction)
    - Faire une sortie HTML dans laquelle le mot choisi initialement est mis en valeur « graphiquement »

## Partie 2 : XQuery

- Après avoir choisi un « mot » anglais ou français, écrire une requête pour extraire tous les bi-contextes contenant ce mot (le contexte contenant le mot choisi et sa traduction)