Comprendre et lire le XML

Morgane Pica

Comprendre et lire le XML

Morgane Pica

Ingénieure d'Études du projet ConDÉ
morgane.pica@unicaen.fr

18/11/2021

RÉGION NORMANDIE

CONDE

CRISCO

Comprendre et lire le XML

I.

Qu’est-ce que le XML ?

I. Qu'est-ce que le XML ? — 1. Définitions

eXtensible Markup Language

- **Langage** informatique.
- **Markup** : langage d'encodage = d'enrichissement et/ou structuration des contenus.
- **eXtensible** : possibilité de convoquer plusieurs standards dans un même document.

I. Qu'est-ce que le XML ? — 1. Définitions

eXtensible Markup Language

- **Langage** informatique.
- **Markup** : langage d'encodage = d'enrichissement et/ou structuration des contenus.
- **eXtensible** : possibilité de convoquer plusieurs standards dans un même document.

- Utilisé pour enrichir les textes, décrire des objets (concrets ou abstraits).
- Fait partie des **standards** du W3C :

> "The World Wide Web Consortium (W3C) is an international community that develops open standards to ensure the long-term growth of the Web."

https://www.w3.org/

I. Qu'est-ce que le XML ? — 1. Définitions

eXtensible Markup Language

XML ≠ format

I. Qu'est-ce que le XML ? — 1. Définitions

eXtensible Markup Language

XML ≠ format

- XML = uniquement **syntaxe**.
- Nécessite de définir un vocabulaire et une grammaire spécifiques en plus.
- Ces vocabulaires/grammaires sont les fameux « **espaces de noms** » dont nous parlerons cet après-midi.

I. Qu'est-ce que le XML ? — 1. Définitions

eXtensible Markup Language

XML ≠ format

- Ce matin je vous apprends donc à lire la syntaxe XML.
- Après ce cours, vous serez capables de déchiffrer n'importe quel document XML.
- Pour construire un document XML, il faut par contre comprendre les espaces de noms.

I. Qu'est-ce que le XML ? — 1. Définitions

Principes généraux du XML

- **Langage strictement hiérarchique** (chaque élément est contenu par d'autres éléments et peut en contenir d'autre, pas de chevauchement d'éléments), modélisable comme arborescence.
- **Langage à balises** (les débuts et fins d'éléments sont respectivement marqués par des balises ouvrantes et fermantes).
- Permet de stocker les différentes chaînes de caractères formant un fichier dans des **éléments qui définissent leur nature.**
- Permet donc de **récupérer** les chaînes de caractères d'une certaine nature.
- Permet un stockage des **métadonnées en interne** (dans des éléments dédiés uniquement aux m.d.).

I. Qu'est-ce que le XML ? — 2. Usages en SHS

Le XML pour nous

- En pratique, *peut* décrire n'importe quoi (même si cela ne signifie pas qu'il *doit* être utilisé pour tout).
- Très largement utilisé, donc facilement partageables dans la communauté scientifique.
- Particulièrement utile pour l'encodage de données textuelles.
- Donc très utilisé pour construire des corpus textuels.
- (Pour info : Le HTML utilise la syntaxe XML pour structurer les pages internet.)

Avantages et inconvénients

- En pratique, *peut* décrire n'importe quoi mais ce n'est pas toujours le langage le plus approprié aux données qu'on veut encoder.
- Sa structure est hiérarchique : selon les données, c'est un avantage ou un inconvénient. Pour structurer un texte c'est un gros avantage. (cf. II.)
- Permet de renseigner la *nature* des choses (éléments de textes, structures linguistiques…), donc :
- Comme on gère l'affichage à part, on peut affranchir le document des codes visuels. (cf. III.)

II.

Lire le XML

II. Lire le XML — 1. Hiérarchie ?

Un langage hiérarchique

= Tout appartient à quelque chose d'autre.

II. Lire le XML — 1. Hiérarchie ?

Un langage hiérarchique

= Tout appartient à quelque chose d'autre.

Pensez de manière hiérarchique :

- *Bâtiment > étage > pièce > étagère > livre > chapitre > phrase > mot > caractère.*
- *Ville > quartier > rue > bâtiment > étage > appartement…*
- *Personne > organe > cellule > noyau > ADN > gène.*
- *Université > département > sujet > professeur > cours > chapitre…*

II. Lire le XML — 1. Hiérarchie ?

Un langage hiérarchique

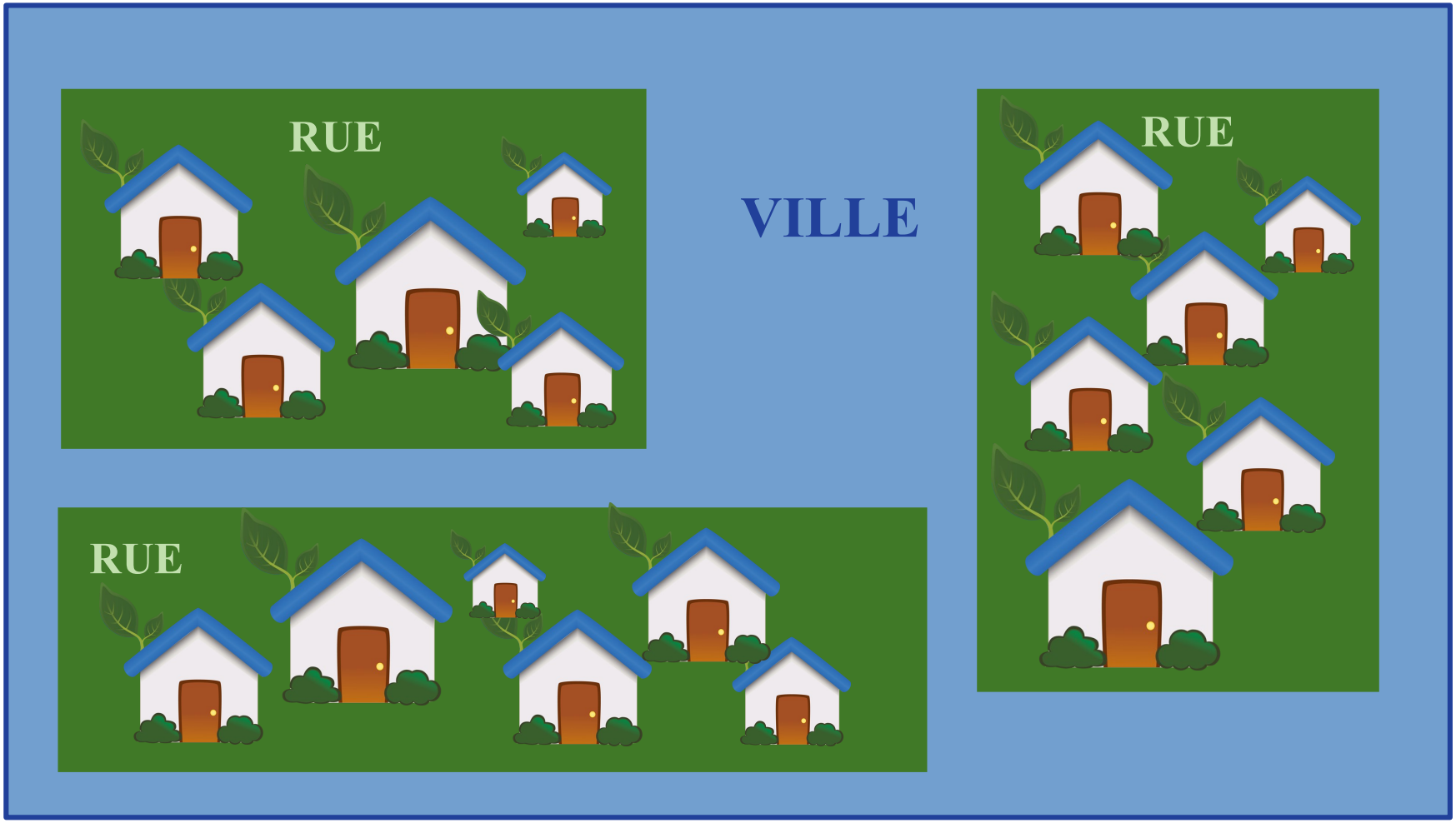
Chaque élément est contenu dans un autre.

Par exemple, dans ma structure de description de ville, l'élément "étage" serait contenu dans l'élément "bâtiment", qui serait dans l'élément "rue", etc.

- *Ville > quartier > rue > bâtiment > étage > appartement…*

II. Lire le XML — 1. Hiérarchie ?

Un langage hiérarchique



Éléments vs balises

Cette hiérarchie, il faut la décrire…
avec des caractères.

- L'ordinateur a besoin de signes pour repérer l'enrichissement.
  C'est à cela que servent les **balises**.

élément ≠ balise

Éléments vs balises

Pensez au panneaux de signalisation.

- Un panneau d'entrée de ville et une ville, ce n'est pas la même chose.

élément ≠ balise

Comprendre et lire le XML

II. Lire le XML —2. La syntaxe XML

II. Lire le XML —2. La syntaxe XML

Éléments vs balises

Le panneau ne constitue pas l'autoroute.
Il sert à vous indiquer que vous y entrez ou la quittez.

De même, une balise sert à indiquer à l'ordinateur
qu'on entre dans un élément ou qu'on le quitte.

II. Lire le XML —2. La syntaxe XML

Éléments vs balises

Contrairement à la signalisation qui utilise des symboles, les balises utilisent des caractères.

Éléments vs balises

On a ici un symbole identique et on distingue entrée et sortie par la barre rouge.

Éléments vs balises

On a ici un symbole identique et on distingue entrée et sortie par la barre rouge.

Les balises ouvrantes et fermantes d'un même élément partagent le même nom (sinon il serait difficile de dire qu'elles se réfèrent au même élément). On les distingue à un `/` supplémentaire dans la balise fermante.

Éléments vs balises

<balise>

balise ouvrante

</balise>

balise fermante

Éléments vs balises

<balise>

↓

balise ouvrante

</balise>

↓

balise fermante

Les balises ouvrantes et fermantes d'un même élément partagent le même nom. On les distingue à un / supplémentaire dans la balise fermante.

Éléments vs balises

<balise>

↓

balise ouvrante

</balise>

↓

balise fermante

Ici, on marque le début et la fin d'un élément nommé « balise ». Tout ce qui est entre ces deux balises, *ainsi que les balises elles-mêmes*, font partie de l'élément.

Éléments vs balises

<balise/>

balise autofermante

Lorsqu'un élément ne contient pas d'autre élément, ni de texte non plus, on dit qu'il est vide. On le note avec une balise autofermante. Notez que le slash est de l'autre côté de la balise par rapport à une balise fermante.

Balises

<balise>Cet élément “balise” contient une chaîne de caractères.</balise>

<phrase>Mais faisons semblant d’utiliser de vrais éléments.</phrase>

Balises

<phrase>
Je peux aussi l'écrire comme ça.
</phrase>

<phrase>
J'ai **indenté** mon texte parce qu'il est à l'intérieur de l'élément "phrase".
</phrase>

<phrase>
Ça ne change absolument rien pour l'ordinateur, mais pour l'œil humain, c'est une convention visuelle qui nous aide à repérer quel élément est dans lequel.
</phrase>

Balises

<phrase>

Les **couleurs** sont aussi une convention visuelle utile pour l'œil humain.

</phrase>

<phrase>

Vous voyez que les balises étant en rouge, vous les repérez facilement.

</phrase>

<phrase>

Les éditeurs n'utilisent pas tous le même code couleur (il est souvent customisable) mais on s'habitue très vite.

</phrase>

II. Lire le XML —2. La syntaxe XML

Balises

```
<phrase>
        Complexifions un peu la structure de mes phrases :
</phrase>
<phrase>
        <mot>Je</mot>
        <mot>peux</mot>
        <mot>faire</mot>
        <mot>un</mot>
        <mot>élément</mot>
        <mot>par</mot>
        <mot>mot</mot>
        <ponctuation>.</ponctuation>
```

Balises

<phrase>
J'ai une structure XML correcte car hiérarchique :
</phrase>

<phrase>
<mot>Je</mot>
<mot>peux</mot>
<mot>faire</mot>
<mot>un</mot>
<mot>élément</mot>
<mot>par</mot>
<mot>mot</mot>
<ponctuation>.</ponctuation>

Attributs

<phrase>

Lorsque la balise elle-même ne suffit pas à décrire l'élément qu'elle marque, on peut utiliser un attribut qui donne des précisions sur la nature de l'élément.

</phrase>

```
<phrase type="averbale">
        <mot role="determinant">Un</mot>
        <mot role="nom">attribut</mot>
        <mot role="conjonction">ou</mot>
        <mot role="pronom">plusieurs</mot>
        <ponctuation type="point">.</ponctuation>
</phrase>
```

Attributs

<phrase type="localisation">
Un attribut est contenu **dans la balise ouvrante**.
</phrase>

<phrase type="separation">
Il est séparé du nom de la balise ou de l'attribut précédent **par une et une seule espace blanche**.
</phrase>

<phrase type="nom">
On voit le **nom** de ces attribut ('type') suivi d'un signe « = » , puis de la **valeur** qu'il contient entourée de **guillemets** (n'oubliez pas les guillemets).
</phrase>

Attributs

<phrase type="description" n="1">
Ceci est un élément « phrase » mais ce n'est pas n'importe lequel : il possède un attribut @type dont la valeur est « description ».
</phrase>

<phrase type="description" n="2">
C'est ici la seconde phrase descriptive, je leur ai donc ajouté un deuxième attribut portant leurs numéros respectifs ; les deux attributs sont séparés d'une espace blanche, tout comme le premier est séparé du nom de l'élément par une espace blanche.
</phrase>

Attributs

<phrase type="regle" sujet="valeurs">
Certains attributs ont des valeurs possibles prédéfinies, d'autres ont un *format* de valeur prédéfini, d'autres encore prennent n'importe quelle nombre et format de valeur.
</phrase>

<phrase type="regle" sujet="attributs">
Il est interdit de mettre deux fois le même attribut dans un même élément.
</phrase>

<phrase type="regle" sujet="attributs">
Contrairement à un élément, un attribut ne peut pas être vide.
</phrase>

Attributs

```
<phrase
        type="regle"
        obligatoire="non"
        sujet="indentation">
            Si j'ai beaucoup d'attributs et que la ligne est trop
            longue pour mon confort visuel, je peux mettre des
            sauts de ligne à la place des espaces, ça ne change
            toujours rien pour l'ordinateur.
</phrase>
```

Récapitulatif

- <balise> → balise ouvrante
- </balise> → balise fermante
- <balise/> → balise autofermante
- attribut="valeur" → attribut suivi de sa valeur

Attention : il faut toujours refermer un élément. Contrairement à HTML qui accepte parfois des erreurs, la plupart du temps XML ne supporte pas un élément non fermé. Il y aura une erreur, votre document sera *mal formé*.

II. Lire le XML —2. La syntaxe XML

Court exemple

```
<livre>
    <partie numero=”1”>
        <chapitre numero=”1”>
            <texte>
                Comme texte il y aurait des phrases.
            </texte>
        </chapitre>
        <chapitre numero=”2”>
            <texte>
                Mais cela prendrait trop de place.
            </texte>
        </chapitre>
        <chapitre numero=”3”>
            <texte>
                Alors je n’en ai pas inventé.
            </texte>
        </chapitre>
    </partie>
    <partie number=”2”/>
</livre>
```

II. Lire le XML —2. La syntaxe XML

Court exemple

```
<livre>
    <partie numero=”1” >
        <chapitre numero=”1” >
            <texte>
                Comme texte il y aurait des phrases.
            </texte>
        </chapitre>
        <chapitre numero=”2” >
            <texte>
                Mais cela prendrait trop de place.
            </texte>
        </chapitre>
        <chapitre numero=”3” >
            <texte>
                Alors je n’en ai pas inventé.
            </texte>
        </chapitre>
    </partie>
    <partie number=”2”/>
</livre>
```

Les éditeurs utilisent des pointillés pour relier les balises du même élément pour les rendre plus facilement repérables visuellement… dans les documents bien formés.

II. Lire le XML —2. La syntaxe XML

Court exemple

```
<livre>
    <partie numero=”1” >
        <chapitre numero=”1” >
            <texte>
                Comme texte il y aurait des phrases.
            </texte>
        </chapitre>
        <chapitre numero=”2” >
            <texte>
                Mais cela prendrait trop de place.
            </texte>
        </chapitre>
        <chapitre numero=”3” >
            <texte>
                Alors je n’en ai pas inventé.
            </texte>
        </chapitre>
    </partie>
    <partie number=”2”/>
</livre>
```

L’élément qui contient tous les autres est appelé la **racine**.

Vocabulaire des relations familiales pour décrire les relations entre éléments :

- **ancêtre**
- **parent**
- **frère**
- **enfant**
- **descendant**

II. Lire le XML —2. La syntaxe XML

Court exemple

```
<livre>
    <partie numero=”1” >
        <chapitre numero=”1” >
            <texte>
                Comme texte il y aurait des phrases.
            </texte>
        </chapitre>
        <chapitre numero=”2” >
            <texte>
                Mais cela prendrait trop de place.
            </texte>
        </chapitre>
        <chapitre numero=”3” >
            <texte>
                Alors je n’en ai pas inventé.
            </texte>
        </chapitre>
    </partie>
    <partie number=”2”/>
</livre>
```

- **Racine** : <livre>
- <texte> est toujours **enfant** de <chapitre>.
- <chapitre> peut avoir des **frères** <chapitre>
- <livre> est un **ancêtre** de tous les autres élément.

II. Lire le XML —3. La conformité

Un document "bien formé"

Un document bien formé suit la syntaxe du XML, c'est-à-dire :

1) HIÉRARCHIE STRICTE

= Pas de chevauchement d'éléments : Toujours fermer un élément avant d'ouvrir un autre élément du même niveau.

MAL FORMÉ :

```
<paragraphe>
        <titre>Voici un <phrase>titre.</titre>Puis une autre.</phrase>
</paragraphe>
```

BIEN FORMÉ :

```
<paragraphe>
        <titre>Voici un titre.</titre><phrase>Puis une autre.</phrase>
</paragraphe>
```

II. Lire le XML —3. La conformité

Un document "bien formé"

Un document bien formé suit la syntaxe du XML, c'est-à-dire :

2) PAS D'ÉLÉMENT NON REFERMÉ

= Fermez toujours les éléments que vous ouvrez.
Si l'élément est vide, utilisez une balise autofermante.

MAL FORMÉ :

```
<paragraphe>
    <titre>Voici un titre.<phrase>Puis une phrase.</phrase>
</paragraphe>
```

BIEN FORMÉ :

```
<paragraphe>
    <titre>Voici un titre.</titre><phrase>Puis une phrase.</phrase>
</paragraphe>
```

III.

Les bonnes pratiques

Interopérabilité ?

- Compatibilité avec d'autres données.
- Données compréhensibles par d'autres personnes et d'autres systèmes informatiques.
- Interrogeable de concert avec d'autres données du même format.
- Données plus pérennes puisque largement comprises et partagées.
- (Attention, la pérennité absolue n'existe pas, on ne peut qu'y tendre.)

III. Les bonnes pratiques — 1. Interopérabilité et standards

Interopérabilité ?

Ce qui implique :

- de s'assurer qu'il n'existe pas un standard correspondant déjà à nos données,
- de penser aux autres utilisateurs potentiels lorsqu'on met en forme des données.

Interopérabilité ?

Ce qui implique :

- de choisir des formats standards, bien installés dans la communauté scientifique, et les plus largement compris pour garantir le partage des données,
- de choisir des formats non propriétaires, pour éviter que les logiciels/systèmes associés soient abandonnés et qu'on ne sache plus les lire,
- d'être très attentifs à la documentation officielle d'un format choisi,
- de solliciter la communauté via des lieux d'échange en cas de doute (forums ou listes de diffusions).

Standards utiles en SHS

La communauté scientifique s'est déjà réunie pour produire un format XML permettant d'encoder des textes. C'est le standard XML-TEI (*Text Encoding Initiative*).

Il existe également le standard XML-EAD (*Encoded Archival Description*) qui traduit en syntaxe XML la norme internationale ISAD(G) pour le catalogage archivistique.

Choisir un format particulier ne signifie pas que les données ne sont pas convertibles ensuite dans d'autres formats ensuite si besoin.

Fond vs forme

Texte

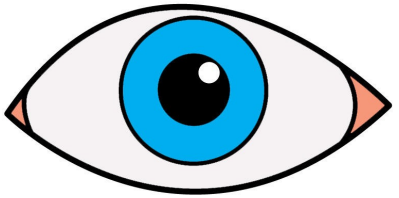
Un texte est une série orale ou écrite de mots perçus comme constituant un ensemble cohérent, porteur de sens et utilisant les structures propres à une langue (conjugaisons, construction et association des phrases...)1. Un texte n'a pas de longueur déterminée sauf dans le cas de poèmes à forme fixe comme le sonnet ou le haïku.

L'étude formelle des textes s'appuie sur la linguistique, qui est l'approche scientifique du langage.

III. Les bonnes pratiques — 2. Repenser son rapport aux textes

Fond vs forme

Texte



- Police plus grande
- Placé au-dessus
- Séparé du corps

Un **texte** *est une série orale ou écrite de mots perçus comme constituant un ensemble cohérent, porteur de sens et utilisant les structures propres à une langue (conjugaisons, construction et association des phrases...). Un texte n'a pas de longueur déterminée sauf dans le cas de poèmes à forme fixe comme le sonnet ou le haïku.*

L'étude formelle des textes s'appuie sur la linguistique, qui est l'approche scientifique du langage.

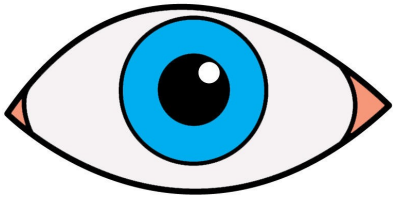
III. Les bonnes pratiques — 2. Repenser son rapport aux textes

Fond vs forme

Texte

Un **texte** *est une série orale ou écrite de mots perçus comme constituant un ensemble cohérent, porteur de sens et utilisant les structures propres à une langue (conjugaisons, construction et association des phrases...). Un texte n'a pas de longueur déterminée sauf dans le cas de poèmes à forme fixe comme le sonnet ou le haïku.*

L'étude formelle des textes s'appuie sur la linguistique, qui est l'approche scientifique du langage.



- Police plus grande
- Placé au-dessus
- Séparé du corps

= titre

Fond vs forme

Text
e

Chaîne de caractères,

- Police : « Liberation Sans »
- Taille : 22 px.
- Couleur : noire.
- Alignement : gauche.

Fond vs forme

Chaîne de caractères

- “Texte”

Enrichissement structurel

- = titre

Enrichissement formel

- police
- couleur
- position...

Fond vs forme

Enrichissement structurel

- = titre

XML

IV.

Les outils du XML

XPath

Langage permettant de décrire la position d'éléments cibles à l'intérieur d'une arborescence XML.

- Permet d'**analyser** plus rapidement une structure XML complexe pour comprendre l'organisation d'un (ensemble de) fichier(s).
- Permet de **cibler** des éléments pour les utiliser ou les modifier.
- On se réfère aux éléments selon les éléments parents/enfants et leur position dans l'ordre des éléments du même parent.
- Comprend des **fonctions** pour transformer les chaînes de caractères éventuellement obtenues.

XPath

Exemple XPath :

/TEI/text/body/div[3 and @type='chapitre']/head

- Commencer à l'élément racine '**TEI**'
- élément enfant '**text**'
- élément enfant '**body**'
- troisième élément enfant '**div**' possédant l'attribut 'type' à valeur 'chapitre'
- élément enfant '**head**'

→ On cherche à récupérer le titre du troisième chapitre dans un fichier en XML-TEI.

XQuery

Langage permettant de requêter et transformer un·des document·s XML.

- Syntaxe propre, concise et facile à apprendre.
- Utilise XPath pour cibler les éléments dans les XML.
- Permet de stocker les informations obtenues dans des variables pour pouvoir les utiliser/transformer plusieurs fois.
- Souvent utilisé pour assurer l'affichage d'un XML dans des pages HTML.
- Nécessite de connaître son document pour savoir quels éléments viser et quoi en faire.

XQuery

Exemple XQuery :

```
for $chapitre in /livre//chapitre
where $chapitre/@type != 'introduction'
let $chapTitre := $chapitre/titre
let $debut := $chapitre/texte/paragraphe[1]
return
        <div>
            <h>{$chapTitre}</h>
            <p>{$debut}</p>
        </div>
```

XSL

eXtensible Stylesheet Language.
Permet de transformer des documents XML.

- Syntaxe ne nécessitant pas d'apprentissage supplémentaire puisqu'en XML.
- Utilise XPath pour cibler les éléments dans les XML.
- Organisation en "templates" permettant de décrire le traitement de chaque élément séparément de la structure du document à produire. Un template est appelé là où il doit être appliqué.
- Souvent utilisé pour transformer le résultat d'une requête XQuery brute (réorganisation, rendu plus fin...). Plus précis pour certaines choses, moins pour d'autres. Les

IV. Les outils du XML — 1. Langages

XSL

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

    <xsl:output method="xml" version="1.1" encoding="UTF-8" omit-xml-declaration="no" indent="no" undeclare-prefixes="
    yes"/>

    <xsl:template match=".">
        <xsl:apply-templates/>
    </xsl:template>

    <!-- Copy all unchanging nodes. -->
    <xsl:template match="node()|@*">
        <xsl:copy copy-namespaces="no">
            <xsl:apply-templates select="node()|@*"/>
        </xsl:copy>
    </xsl:template>

    <!-- Keep only the contents of the second child of <choice> elements, which will necessarily be either <expan>, <
    reg> or <corr>. -->
    <xsl:template match="//choice"><xsl:value-of select="child::*[2]"/></xsl:template>

    <!-- Abandon facsimile image information, as well as now useless line beginnings. -->
    <xsl:template match="//lb"/>
    <xsl:template match="//facsimile/descendant-or-self::*"/>

    <!-- Change part of the <edition> description. -->
    <xsl:template match="/TEI//editionStmt/edition">
        <xsl:copy copy-namespaces="no"><xsl:value-of select="replace(child::text(), 'complète', 'simplifiée')"/></xsl:
        copy>
    </xsl:template>
```

Extrait d'une transformation écrite pour le projet ConDÉ.

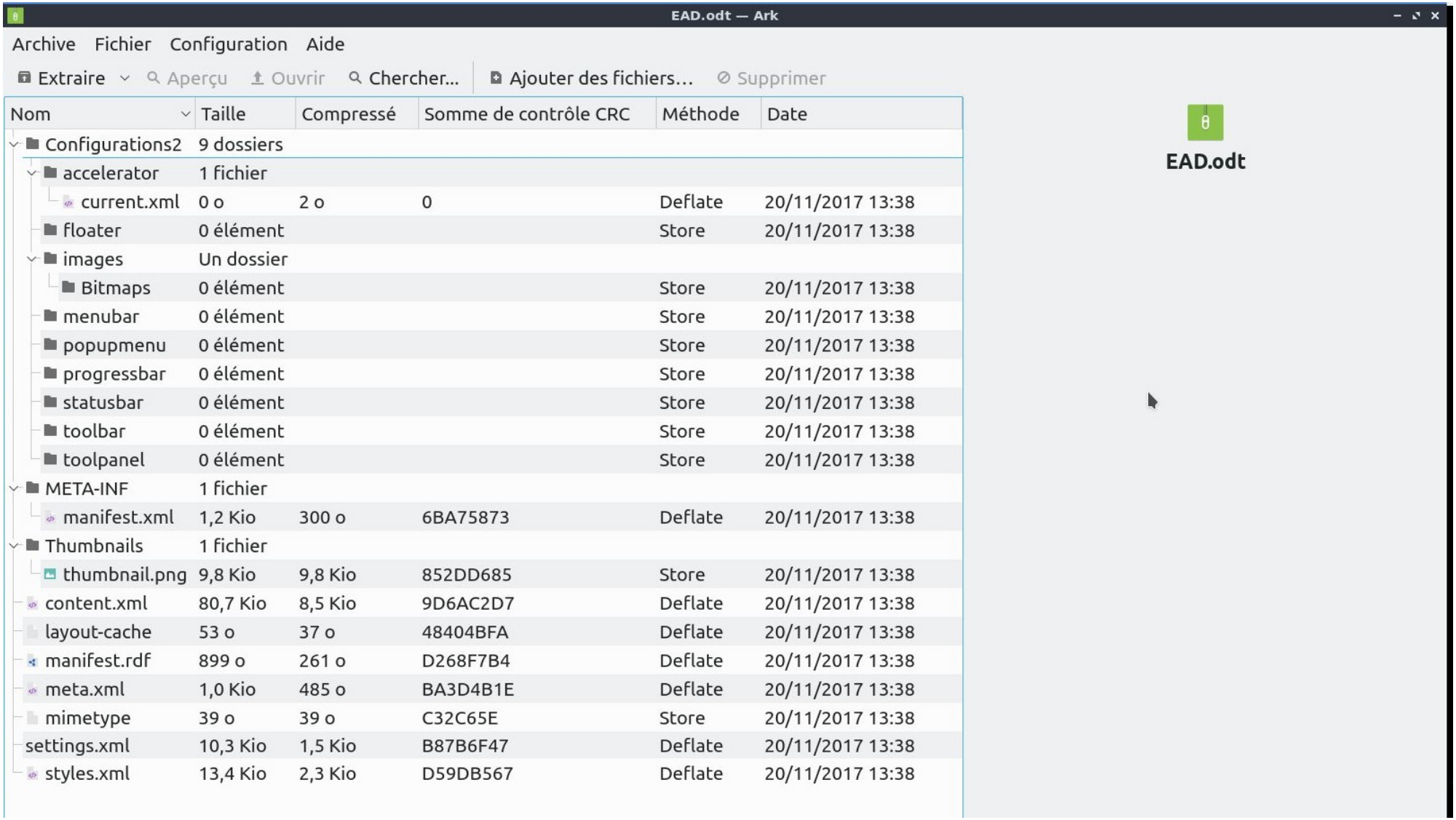
Pourquoi pas Word/Writer ?

Un traitement de texte est un logiciel de mise en page.

- Il n'est pas fait pour du code ! Toute la mise en page (italiques, centrage, alinéas, marges sur la page, largeur de la page...) est encodée en XML derrière.
- → Les fichiers sont lourds… et ils ne sont même pas des fichiers mais des dossiers zipés.
- Il y a plein de transformations automatiques (guillemets, espaces insécables, détection des dates…), en particulier sur des caractères signifiants en XML.

IV. Les outils du XML — 2. Éditeurs de texte

Pourquoi pas Word/Writer ?



Nom	Taille	Compressé	Somme de contrôle CRC	Méthode	Date
Configurations2	9 dossiers				
accelerator	1 fichier				
current.xml	0 o	2 o	0	Deflate	20/11/2017 13:38
floater	0 élément			Store	20/11/2017 13:38
images	Un dossier				
Bitmaps	0 élément			Store	20/11/2017 13:38
menubar	0 élément			Store	20/11/2017 13:38
popupmenu	0 élément			Store	20/11/2017 13:38
progressbar	0 élément			Store	20/11/2017 13:38
statusbar	0 élément			Store	20/11/2017 13:38
toolbar	0 élément			Store	20/11/2017 13:38
toolpanel	0 élément			Store	20/11/2017 13:38
META-INF	1 fichier				
manifest.xml	1,2 Kio	300 o	6BA75873	Deflate	20/11/2017 13:38
Thumbnails	1 fichier				
thumbnail.png	9,8 Kio	9,8 Kio	852DD685	Store	20/11/2017 13:38
content.xml	80,7 Kio	8,5 Kio	9D6AC2D7	Deflate	20/11/2017 13:38
layout-cache	53 o	37 o	48404BFA	Deflate	20/11/2017 13:38
manifest.rdf	899 o	261 o	D268F7B4	Deflate	20/11/2017 13:38
meta.xml	1,0 Kio	485 o	BA3D4B1E	Deflate	20/11/2017 13:38
mimetype	39 o	39 o	C32C65E	Store	20/11/2017 13:38
settings.xml	10,3 Kio	1,5 Kio	B87B6F47	Deflate	20/11/2017 13:38
styles.xml	13,4 Kio	2,3 Kio	D59DB567	Deflate	20/11/2017 13:38

IV. Les outils du XML — 2. Éditeurs de texte

Pourquoi pas Word/Writer ?

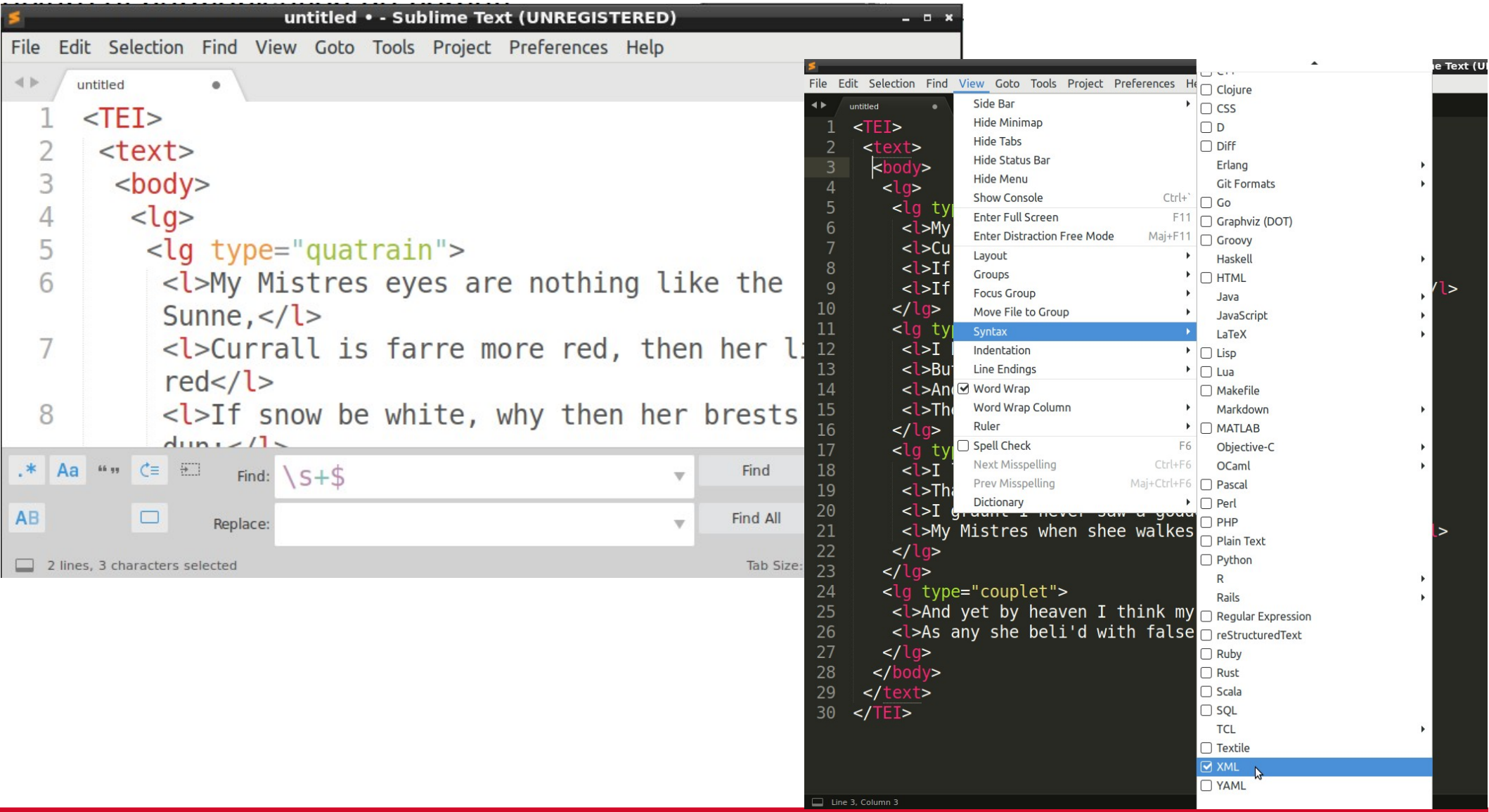
- Un éditeur de texte sert à lire les formats texte sans s'encombrer de mise-en-page. Vous pouvez ouvrir n'importe quel fichier de code avec un éditeur de texte, il vous sera montré en tant que chaîne de caractères.

Lorsque vous faites du code ou de l'encodage, surtout n'utilisez *jamais* un logiciel de traitement de texte qui rajoutera automatiquement du code sur du code, ce qui rendra votre code totalement illisible, ajoutera des corrections automatiques…

- Certains éditeurs de texte ont des **fonctionnalités dédiées** au code et aux fichiers texte, dont vous aurez besoin : réglage de l'encodage des caractères, expressions régulières, recherche dans tous les fichiers d'un dossier / tous les fichiers ouverts, aide visuelle à la lecture des éléments d'encodage / de code

IV. Les outils du XML — 2. Éditeurs de texte

Exemple : SublimeText



Quelques éditeurs de texte

Quelques éditeurs de texte :

- SublimeText → *www.sublimetext.com*
- jEdit → *www.jedit.org*
- Atom → *atom.io*
- TextWrangler → *www.barebones.com/products/textwrangler/*
- Notepad++ → *https://notepad-plus-plus.org/*
- Vim → *www.vim.org*
- Emax → *www.gnu.org/software/emacs/*

Éditeurs spécialisés dans le XML :

XML Mind → *www.xmlmind.com/xmleditor/*

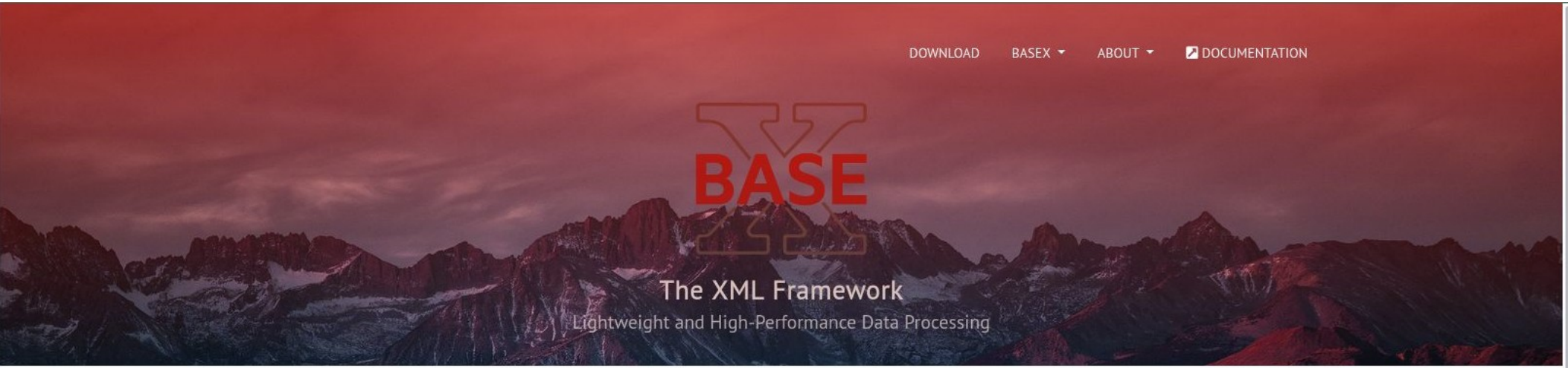
Attention, Oxygen (extrêmement complet) propose des versions payantes mais également une version d'essai d'un mois. Pour un travail ponctuel, pensez-y.

Oxygen → *www.oxygenxml.com*

Bibliographie succinte :

- *BaseX*. Outil collaboratif, depuis 2005. URL : https://basex.org/.
- *MaX*. Pôle Document Numérique et Centre de Ressources Technologiques pour les TIC, Université de Caen. Url : http://www.unicaen.fr/recherche/mrsh/document_numerique/outils/max.
- *TEI Guidelines*. Text Encoding Initiative Consortium (éd.) 1994-2020. *Text Encoding Initiative Guidelines*. https://tei-c.org.
- *TEI Repository. Text Encoding Initiative Consortium (éd.) 1994-2020. Text Encoding Initiative Repository. https://github.com/TEIC/TEI.*
- *XML*.World Wide Web Consortium (éd.) 2013-2020. *eXtensible Markup Language, Fifth edition*.
- *XSL*. World Wide Web Consortium (éd.) 2009-2021. *eXtensible Stylesheet Language*, version 3.0. URL : https://www.w3.org/Style/XSL/.
- *XPath*. World Wide Web Consortium (éd.) 1999-2020. *XML Path Language*. URL : https://www.w3.org/TR/xpath/.

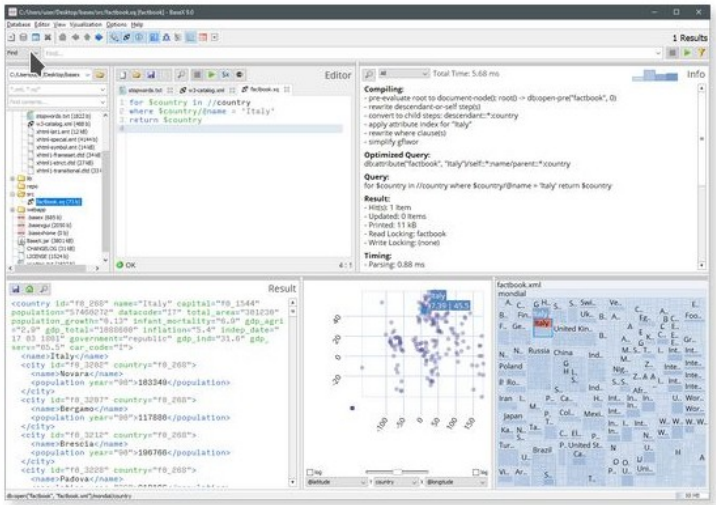
IV. Les outils du XML — 3. BaseX



BaseX is a robust, high-performance XML database engine and a highly compliant XQuery 3.1 processor with full support of the W3C Update and Full Text extensions. It serves as excellent framework for building complex data-intensive web applications.
It comes with interactive user interfaces (desktop, web-based) that give you great insight into your data.

BaseX is completely Open Source. Join our mailing lists to get regular updates. BaseX GmbH offers professional support, develops individual software solutions, and provides training on XML technologies. If you like BaseX and want to support its open source development, consider making a donation. BaseX is lightweight, easy to install and runs out of the box...

Download BaseX 9.6.3



- Excellent outil qui peut être utilisé à la fois seul pour explorer des données et comme base pour construire une interface (comme MaX, le moteur d'affichage XML de la MRSH de Caen).

- Il gère entre autres :
  - La visualisation des données XML,
  - La gestion des documents en collections,
  - L'écriture et exécution de requêtes XQuery...

- Il est OpenSource !

Merci pour votre attention !

Des questions ?

Bibliographie succinte :

- *BaseX*. Outil collaboratif, depuis 2005. Url : https://basex.org/.
- *MaX*. Pôle Document Numérique et Centre de Ressources Technologiques pour les TIC, Université de Caen. Url : http://www.unicaen.fr/recherche/mrsh/document_numerique/outils/max.
- *TEI Guidelines*. Text Encoding Initiative Consortium (éd.) 1994-2020. *Text Encoding Initiative Guidelines*. Url : https://tei-c.org.
- *TEI Repository*. Text Encoding Initiative Consortium (éd.) 1994-2020. *Text Encoding Initiative Repository*. Url : https://github.com/TEIC/TEI.
- *XML*.World Wide Web Consortium (éd.) 2013-2020. *eXtensible Markup Language, Fifth edition*.
- *XSL*. World Wide Web Consortium (éd.) 2009-2021. *eXtensible Stylesheet Language*, version 3.0. Url : https://www.w3.org/Style/XSL/.
- *XPath*. World Wide Web Consortium (éd.) 1999-2020. *XML Path Language*. Url : https://www.w3.org/TR/xpath/.