

# Humanités numériques : structuration des données et des documents textuels

E. ROUQUETTE

Cours 1 – 12 février 2025

Markdown et Pandoc (fin)

# Utilisation de base de pandoc

- ▶ Créer un dossier dédié au cours, y importer le fichier markdown créé avec Dillinger sous le nom `exercice.md`
- ▶ Ouvrir un terminal et se placer dans le répertoire (dossier) que l'on vient de créer
- ▶ Taper : `pandoc exercice.md -o exercice.html`
- ▶ Ouvrir le fichier obtenu avec un navigateur
- ▶ Refaire la même opération avec les changements nécessaires pour obtenir un fichier odt
- ▶ Observez → Les métadonnées sont utilisées dans le fichier odt ainsi créé

# Convertir en pdf avec pandoc

- ▶ Le programme utilisé par pandoc est  $\text{\LaTeX}$ 
  - ▶ Très belle mise en forme
  - ▶ Facilement paramétrable
  - ▶ **Mais** : long à installer...
- ▶ D'autres programmes peuvent être utilisés ; plus longs à paramétrer, mais plus légers à installer. Exemple :
  - ▶ wkhtmltopdf (« Web Kit html to pdf »)

→ Installer wkhtmltopdf ici :

<https://wkhtmltopdf.org/downloads.html>

# Convertir avec pandoc et wkhtmltopdf

Pour convertir en pdf avec un autre moteur que  $\text{\LaTeX}$ , utiliser l'option `--pdf-engine` :

```
pandoc exercice.md --pdf-engine=wkhtmltopdf -o exercice.pdf
```

Pour mettre une table des matières, utiliser l'option `--toc` :

```
pandoc exercice.md --pdf-engine=wkhtmltopdf --toc -o exercice.pdf
```

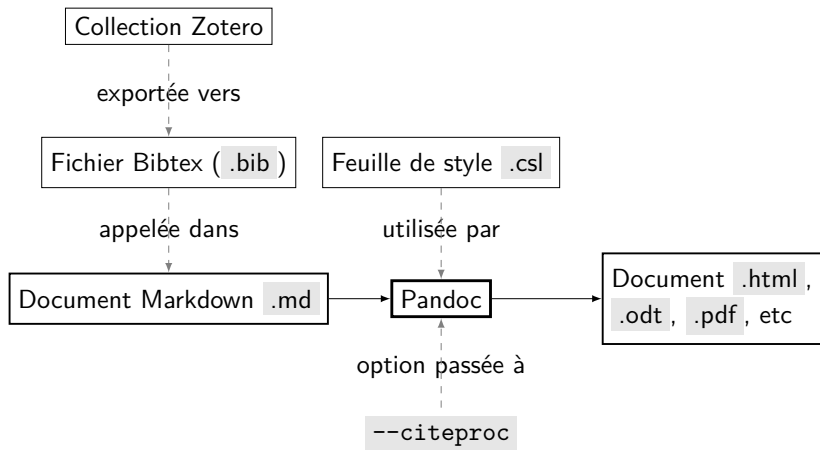
# Convertir avec pandoc et wkhtmltopf

Il est possible d'ajouter un template ou une feuille de style CSS (*Cascading Style Sheets*). Il faut pour cela compléter l'en-tête YAML (le bloc de métadonnées). Tester avec la feuille de style `exemple.css`

```
---  
title: titre  
author: auteur  
date: 12 février 2025  
css: exemple.css  
---
```

**NB** : Ce n'est pas nécessaire quand pandoc utilise  $\text{\LaTeX}$

# Une bibliographie avec Pandoc, Zotero et Bibtex : principe



# Exporter sa collection Zotero vers un fichier bibtex

## Export simple avec Zotéro

clic droit sur la collection – exporter la collection – Format bibtex

→ Exporter une collection et placer le fichier `.bib` obtenu dans le dossier d'exercice



# Exporter sa collection Zotero vers un fichier bibtex

## Garder à jour son fichier .bib avec betterbibtex

<https://retorque.re/zotero-better-bibtex/>

### Installation :

1. télécharger le XPI file et le sauvegarder. ⚠ sous firefox, faire clic droit et enregistrer la cible du lien (un simple clic sur le lien ne marchera pas)
2. Dans Zotero :  
extension - ajouter depuis un fichier - installer - redémarrer zotero

# Exporter sa collection Zotero vers un fichier bibtex

## Garder à jour son fichier .bib avec betterbibtex

### Configuration :

#### 1. Configurer Betterbibtex pour l'exportation :

édition - paramètres - betterbibtex :

- ▶ « exportation » : décocher « exporter les caractères unicode »
- ▶ choisir « ajouter les url à l'exportation dans le champs url »
- ▶ onglet « divers » : décocher « appliquer la capitalisation aux titres »
- ▶ onglet « clef de citation » : possibilité de choisir la façon dont les clefs seront générées

#### 2. Mettre en place la synchronisation entre un fichier BibTex et une collection dans Zotero : faire un clic-droit sur la collection à exporter :

exporter - choisir « betterbibtex » - cocher « garder à jour »

→ Exporter sa collection avec Betterbibtex (remplacer la collection existante)

# Comprendre un fichier bibtex

Un fichier `.bib` est composé d'un ensemble de références bibliographiques (« entrées »)

- ▶ à chaque entrée est associée une clef ("key") qui l'identifie
- ▶ chaque entrée comporte un certain nombre d'éléments de description bibliographique (nature de la référence, titre, auteur, date, etc), chacun indiqué dans le champ correspondant.

# Comprendre un fichier bibtex

exemple d'entrée bibliographique :

```
@book{isidoredeseville2020,  
title = {Etymologies. Livre I. La grammaire},  
author = {{Isidore de Séville}},  
editor = {Spevak, Olga},  
translator = {Spevak, Olga},  
year = {2020},  
series = {Auteurs latins du Moyen Âge},  
number = {31},  
publisher = {Les Belles Lettres},  
address = {Paris},  
keywords = {sources, Étymologies},  
}
```

# Compléter son en-tête YAML pour appeler la bibliographie

---

**title:** titre


**author:** auteur

**date:** 12 février 2025

**css:** exemple.css

**bibliography:** ma\_bibliographie.bib

---

 [Rappel : ne pas mettre d'espace ou d'accent dans les noms des dossiers ou des fichiers]

# Appeler une référence bibliographique en markdown

- Appel d'une référence bibliographique[@key, 67-73]
- Avec préfixe[Voir par exemple @key, 2]
- Plusieurs références[@key1, 3-5 ; @key2, 78]

**nb** Selon le style bibliographique choisi, la référence sera appelée dans le corps du texte ou en note de bas de page

Exercice :

- ▶ Dans le fichier d'exercices, ajouter une ou plusieurs références bibliographiques
- ▶ Tester la conversion vers un format pdf, odt ou docx, soit avec son éditeur markdown, soit en ligne de commande avec pandoc :

```
exercice.md --citeproc -o exercice.pdf
```

# Changer le style bibliographique

- ▶ télécharger un style `.csl` (par exemple sur Zotero Style Repository) et mettez-le dans votre dossier. Vous pouvez le faire aussi à partir d'un style installé dans Zotero :  
édition - préférences - éditeur de style - sélectionner un style - enregistrer sous
- ▶ compléter l'en-tête YAML :

```
---  
title: titre  
author: auteur  
date: 12 février 2025  
css: exemple.css  
bibliography: ma_bibliographie.bib  
csl: mon_style.csl  
---
```

Exercice : enregistrer dans votre dossier avec le style Chicago Manual of Style fullnote et tester

# Pour aller plus loin : quelques autres possibilités de l'en-tête YAML

```
---  
title: titre  
subtitle: un sous-titre  
author: auteur  
date: 12 février 2025  
css: exemple.css  
abstract: un résumé  
keywords:  
- mot 1  
- mot 2  
bibliography: ma_bibliographie.bib  
csl: mon_style.csl  
lang: fr-FR  
toc: true  
numbersections: true  
#ceci est un commentaire YAML
```



## Quelques options de pandoc

Syntaxe :

```
pandoc fichier.md --option1 --option2... -o output.pdf
```

`--citeproc` : faire une bibliographie

`--pdf-engine=xxx` : le logiciel pour produire le pdf (par exemple xelatex)

`--template=xxx` : spécifier un template à pandoc

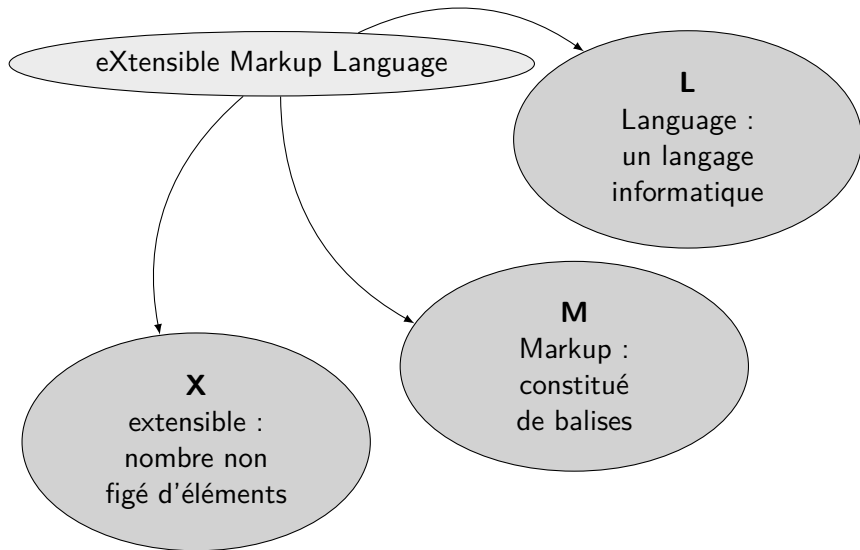
## XML : introduction

# Les langages à balises : rappel

- ▶ Markdown
- ▶  $\text{\LaTeX}$
- ▶ html
- ▶ XML

→ **Mise en forme** (typographique) vs **mise en sens** (sémantique)

# XML : présentation



Contexte :

- ▶ convergence numérique
- ▶ échange de l'information
- ▶ Multiplication des supports

→ **faciliter l'échange** de contenus complexes (arbres, textes enrichis, etc.) entre différents systèmes informatiques (**interopérabilité**)

# Historique

## Quelques dates

**1970 SGML**, Standard Generalized Markup Language, créé par IBM pour échanger les données.

**1989-1992 HTML**, HyperText Markup Language, par Tim Berners-Lee, spécialisé dans l'affichage des données.

**1996** création, par un groupe de travail, de XML pour décrire les données

**1998 XML** devient une **norme du W3C**

**1999** reformulation d'HTML 4 conformément au principe d'XML → **XHTML 1.0**

# Concepts-clefs (rappel)

## Pérennité

Besoin de formats qui durent dans le temps pour ne pas avoir sans cesse à convertir les fichiers au fil du temps.

## Interopérabilité

Besoin de formats interopérables, c'est-à-dire qui puissent être échangés et réutilisés, également pour ne pas avoir à convertir sans cesse ses données

## W3C (World Wide Web Consortium )

Le W3C est un consortium qui vise la standardisation des technologies du Web. Il promeut des standards pour favoriser l'interopérabilité et la pérennité des données.

# Qu'est-ce qu'XML ?

- ▶ XML ne « fait » rien !
- ▶ XML est un langage de structuration des données :
  - ▶ Les informations (données) sont contenues dans des balises
  - ▶ XML décrit les données
  - ▶ XML stocke les données
- ▶ Tout type de données :
  - ▶ Textes
  - ▶ Textes structurés
  - ▶ Image
  - ▶ ...

Mais surtout du texte !



# Avantages de XML

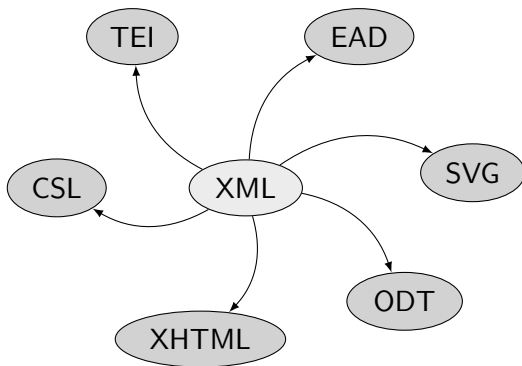
## Avantages internes

- ▶ séparation entre fond et forme, contenu et présentation
- ▶ langage simple : du texte
- ▶ lisible par l'humain et la machine
- ▶ langage structuré
- ▶ souple car extensible
- ▶ composable (plusieurs langages au sein d'un même document)

## Avantages externes

- ▶ indépendant des plateformes
- ▶ standard du W3C : longévité et interopérabilité
- ▶ adaptable à tous les types de description qu'on veut mener (on peut créer des langages)

**XML est un métalangage.**



# Omniprésence de XML

- ▶ standards XML utilisés dans de nombreuses bases de données (formats BnF : METS, MODS...)
- ▶ langage XML/EAD utilisé dans la description des fonds d'archives et manuscrits
- ▶ langage XML/TEI largement utilisé dans les projets de recherche en SHS (éditions numériques).

Liste d'éditions numériques sur le site [www.digitale-edition.de](http://www.digitale-edition.de)  
Exemple d'édition numérique en TEI : <http://stendhal.demarre-shs.fr/index.php>

# Outils liés à XML

XPath **sélection** des données

XPointer **lien** entre les données

XQuery **interrogation** des données

XSL **transformation** des données

schémas **standardisation** de la validation des documents

# Ce que fait ressortir XML

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <titre>L'Éducation sentimentale</titre>
  <auteur>Gustave Flaubert</auteur>
  <texte>
    <partie><titrePartie>Première partie</titrePartie>
      <chapitre><numeroChapitre>I</numeroChapitre>
        <paragraphe>Le <date>15 septembre 1840</date>,
          ↪ vers six heures du matin, la
          ↪ <nomBateau>Ville-de-Montereau</nomBateau>,
          ↪ près de partir, fumait à gros tourbillons
          ↪ devant le quai <nomLieu>
          ↪ Saint-Bernard</nomLieu>.
        </paragraphe>
      </chapitre>
    </partie>
  </texte>
</document>
```

# Ce que fait ressortir XML

Gustave Flaubert

Première partie

I

Le 15 septembre 1840, vers six heures du matin, la Ville-de-Montereau, près de partir, fumait à gros tourbillons devant le quai Saint-Bernard.

# Ce que fait ressortir XML

Gustave Flaubert

Première partie

I

Le 15 septembre 1840, vers six heures du matin, la Ville-de-Montereau, près de partir, fumait à gros tourbillons devant le quai Saint-Bernard.

Des métadonnées

...

`<titre>L'Éducation sentimentale</titre>`

`<auteur>Gustave Flaubert</auteur>`

...

# Ce que fait ressortir XML

Gustave Flaubert

Première partie

I

Le 15 septembre 1840, vers six heures du matin, la Ville-de-Montereau, près de partir, fumait à gros tourbillons devant le quai Saint-Bernard.

Une structure

<partie>, <chapitre>, <paragraphe>



# Ce que fait ressortir XML

Gustave Flaubert

Première partie

I

Le 15 septembre 1840, vers six heures du matin, la Ville-de-Montereau, près de partir, fumait à gros tourbillons devant le quai Saint-Bernard.

Des informations sémantiques sur le texte

<date>, <nomBateau>, <nomLieu>

# Les balises structurent l'information

- ▶ elles peuvent encadrer une information de type texte, avec une balise ouvrante et une balise fermante

```
<balise>texte</balise>
```

- ▶ elles peuvent être auto-fermantes

```
<balise/>
```

- ▶ elles peuvent contenir d'autres balises

```
<balise1>
```

```
  <balise2>contenu</balise2>
```

```
    <balise3/>
```

```
</balise1>
```

# Les balises permettent de stocker et d'interroger l'information

- ▶ Obtenir à partir d'un même fichier plusieurs sorties différentes
- ▶ Interroger le contenu d'un texte à partir de son sémantisme (recherche de noms de personne, de noms de lieu, de dates, etc)

# Forme des balises

Balise ouvrante :

Chevron gauche - nom de la balise - chevron droit

<balise>

# Forme des balises

Balise ouvrante :

Chevron gauche - nom de la balise - chevron droit

Balise fermante :

Chevron gauche - slash - nom de la balise -chevron droit

<balise>Contenu de la balise</balise>

# Forme des balises

Balise ouvrante :

Chevron gauche - nom de la balise - chevron droit

Balise fermante :

Chevron gauche - slash - nom de la balise -chevron droit

Balise autofermante (balise « borne », *milestones*)

Chevron gauche - nom de la balise - slash - chevron droit

<balise>Contenu de la balise</balise>

<balise/>

# Forme des balises

Balise ouvrante :

Chevron gauche - nom de la balise - chevron droit

Balise fermante :

Chevron gauche - slash - nom de la balise -chevron droit

Balise autofermante (balise « borne », *milestones*)

Chevron gauche - nom de la balise - slash - chevron droit

`<balise>`Contenu de la balise`</balise>`

`<balise/>`

**Exemple :**

`<paragraphe>` début d'un paragraphe au milieu duquel

↪ il y a un changement de page `<pagebreak/>` suite

↪ du paragraphe `</paragraphe>`

# Syntaxe



# Le prologue

```
<?xml version="1.0" encoding="UTF-8"?>
```

Un document XML commence obligatoirement par une **déclaration XML (ou prologue)**. Il indique la version de la recommandation XML qui sert de base à l'écriture du fichier (il n'y a eu à cette date qu'une seule version, la version 1.0), ainsi que l'encodage de caractères utilisé.

Cette déclaration permet à la machine de comprendre que le document en question est un document XML : c'est une **instruction pour le processeur**

# Les éléments

Un élément est le contenu d'une balise :

`<element>contenu</element>`

Élément vide (*milestone*) :

`<element/>`

# Les éléments

Règles (principe de conformité) :

**Nom** Un élément doit être entouré de deux balises de même nom (sensible à la casse)

# Les éléments

Règles (principe de conformité) :

**Nom** Un élément doit être entouré de deux balises de même nom (sensible à la casse)

**Clôture** Un élément ouvert doit être fermé obligatoirement

# Les éléments

Règles (principe de conformité) :

**Nom** Un élément doit être entouré de deux balises de même nom (sensible à la casse)

**Clôture** Un élément ouvert doit être fermé obligatoirement

**Imbrication** Un élément ne peut pas en chevaucher un autre

# Les éléments

Règles (principe de conformité) :

**Nom** Un élément doit être entouré de deux balises de même nom (sensible à la casse)

**Clôture** Un élément ouvert doit être fermé obligatoirement

**Imbrication** Un élément ne peut pas en chevaucher un autre

⚠ [On ne peut pas écrire :]

`<element/>contenu</Element>`

ni :

`<element/>contenu`

ni :

`<element/><element2>contenu</element></element2>`

# Les éléments

XML est un métalangage ; donc :

- ▶ il n'est **pas doté de balises spécifiques** : on peut inventer ses propres balises.

`<p>contenu</p>` ✓

`<par>contenu</par>` ✓

`<paragraph>contenu</paragraph>` ✓

- ▶ il sert d'**architecture** pour les langages XML/TEI, XML/EAD qui, eux sont dotés d'une grammaire précise (listes de balises et de valeurs, manière de les agencer)

`<p>contenu</p>` ✓

`<par>contenu</par>` ✗

# Les attributs et leur valeur

Les attributs se placent dans la balise ouvrante ou la balise autofermante et sont suivis de leur valeur entre guillemets :

```
<element attribut="valeur de l'attribut">contenu</element>
```

```
<elementVide attribut="valeur de l'attribut"/>
```



# Les attributs et leur valeur

Les attributs se placent dans la balise ouvrante ou la balise autofermante et sont suivis de leur valeur entre guillemets :

```
<element attribut="valeur de l'attribut">contenu</element>
```

```
<elementVide attribut="valeur de l'attribut"/>
```

Les attributs permettent de spécifier les éléments. Les spécifications qu'ils permettent sont de plusieurs ordres, par exemple :

# Les attributs et leur valeur

Les attributs se placent dans la balise ouvrante ou la balise autofermante et sont suivis de leur valeur entre guillemets :

```
<element attribut="valeur de l'attribut">contenu</element>
```

```
<elementVide attribut="valeur de l'attribut"/>
```

Les attributs permettent de spécifier les éléments. Les spécifications qu'ils permettent sont de plusieurs ordres, par exemple :

- ▶ une numérotation

```
<paragraphe n="1">contenu du paragraphe 1</p>
```

```
<paragraphe n="2">contenu du paragraphe 2</p>
```

# Les attributs et leur valeur

Les attributs se placent dans la balise ouvrante ou la balise autofermante et sont suivis de leur valeur entre guillemets :

```
<element attribut="valeur de l'attribut">contenu</element>
```

```
<elementVide attribut="valeur de l'attribut"/>
```

Les attributs permettent de spécifier les éléments. Les spécifications qu'ils permettent sont de plusieurs ordres, par exemple :

- ▶ une numérotation

```
<paragraphe n="1">contenu du paragraphe 1</p>
```

```
<paragraphe n="2">contenu du paragraphe 2</p>
```

- ▶ des types liés à la structure du document

```
<titre type="principal">L'Éducation sentimentale</titre>
```

```
<titre type="partie">Première partie</titre>
```

# Les attributs et leur valeur

Les attributs se placent dans la balise ouvrante ou la balise autofermante et sont suivis de leur valeur entre guillemets :

```
<element attribut="valeur de l'attribut">contenu</element>
```

```
<elementVide attribut="valeur de l'attribut"/>
```

Les attributs permettent de spécifier les éléments. Les spécifications qu'ils permettent sont de plusieurs ordres, par exemple :

- ▶ une numérotation

```
<paragraphe n="1">contenu du paragraphe 1</p>
```

```
<paragraphe n="2">contenu du paragraphe 2</p>
```

- ▶ des types liés à la structure du document

```
<titre type="principal">L'Éducation sentimentale</titre>
```

```
<titre type="partie">Première partie</titre>
```

- ▶ des types liés au sémantisme des éléments

```
... devant le quai <nom type="lieu">Saint-Bernard</nom>
```

# Les attributs et leur valeur

Utilisation :

**Spécification** Utilisé pour spécifier un élément

# Les attributs et leur valeur

Utilisation :

**Spécification** Utilisé pour spécifier un élément

**Situation** Placés après le nom de l'élément dans la balise ouvrante

# Les attributs et leur valeur

Utilisation :

**Spécification** Utilisé pour spécifier un élément

**Situation** Placés après le nom de l'élément dans la balise ouvrante

**Répétabilité** Un même attribut ne peut pas être réutilisé dans la même balise

# Les commentaires

Un autre type d'élément peut apparaître dans les documents XML, les commentaires :

```
<!-- Un commentaire -->
```

Utilisation :



# Les commentaires

Un autre type d'élément peut apparaître dans les documents XML, les commentaires :

```
<!-- Un commentaire -->
```

Utilisation :

**Syntaxe** Les commentaires sont placés entre les marques <!-- (ouvrante) et --> (fermante)

# Les commentaires

Un autre type d'élément peut apparaître dans les documents XML, les commentaires :

```
<!-- Un commentaire -->
```

Utilisation :

**Syntaxe** Les commentaires sont placés entre les marques <!-- (ouvrante) et --> (fermante)

**Rôle** Ils servent à à faire des remarques sur les usages de l'encodage, sur les choses à faire, etc. Ils sont destinés à un lecteur humain

## Bien formé ou mal formé ?

`<par>du texte</par>`

`<par><article>du</article><nom>texte</nom></par>`

`<par><article>du <nom></article>texte</nom></par>`

`<par><article>du <nom>texte</nom></article></par>`

`<par type="texte"> du texte</par>`

`<par type=texte> du texte</par>`

`<par type="texte"> du texte<par/>`

`<par type="texte"> du texte<nom>nom</par>`

`<segment type="texte" type="nombre"> du texte</par>`

# L'arborescence

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <titre>L'Éducation sentimentale</titre>
  <auteur>Gustave Flaubert</auteur>
  <texte>
    <partie><titrePartie>Première partie</titrePartie>
      <chapitre><numeroChapitre>I</numeroChapitre>
        <paragraphe>Le <date>15 septembre 1840</date>,
          ↪ vers six heures du matin, la
          ↪ <nomBateau>Ville-de-Montereau</nomBateau>,
          ↪ près de partir, fumait à gros tourbillons
          ↪ devant le quai
          ↪ <nomLieu>Saint-Bernard</nomLieu>.
        </paragraphe>
      </chapitre>
    </partie>
  </texte>
</document>
```

# L'arborescence

Un élément racine

`<document>`

...

`</document>`

L'élément racine est obligatoire ; il contient tous les autres éléments

# L'arborescence

Les éléments enfants de l'élément racine

```
<document>  
  <titre>...</titre>  
  <auteur>...</auteur>  
  <texte>...</texte>  
</document>
```

# L'arborescence

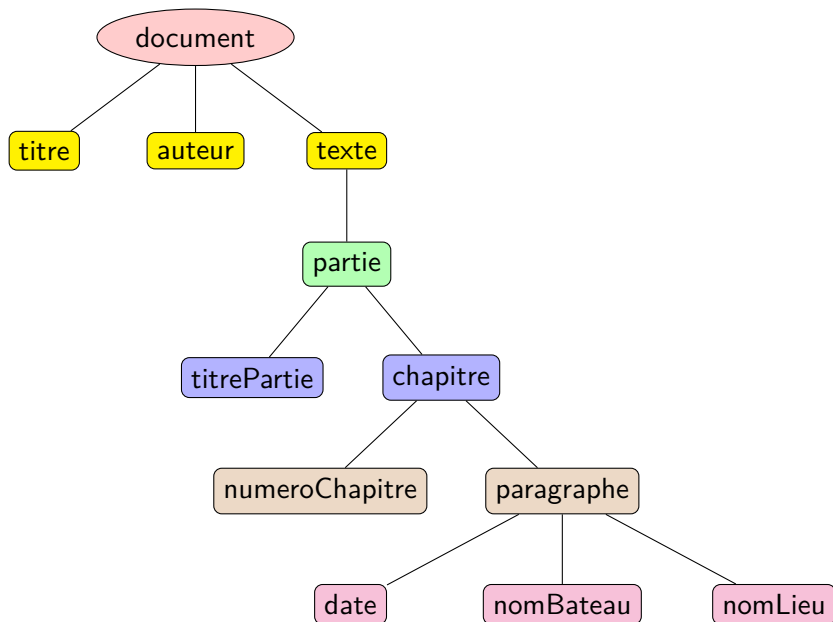
Les éléments enfants de l'élément racine

```
<document>  
  <titre>...</titre>  
  <auteur>...</auteur>  
  <texte>...</texte>  
</document>
```

...Qui eux-mêmes peuvent avoir des éléments enfants

```
<texte>  
  <partie>  
    <chapitre>  
      <paragraphe>...</paragraphe>  
    </chapitre>  
  </partie>  
</texte>
```

# L'arborescence





# Récapitulatif

Le langage XML **est** :

- ▶ un langage à **balises**
- ▶ un langage à **structuration arborescente**, qui fonctionne avec des nœuds parents et enfants
- ▶ un **métalangage**

# Récapitulatif

Le langage XML **est** :

- ▶ un langage à **balises**
- ▶ un langage à **structuration arborescente**, qui fonctionne avec des nœuds parents et enfants
- ▶ un **métalangage**

Le langage XML **doit** :

- ▶ avoir une **balise racine**
- ▶ correspondre au principe de **conformité** (pas de chevauchement)

Créer ses premiers documents XML avec  
Oxygen

## Quelques règles sur le nom des éléments

- ▶ **pas de caractère diacritique** : pas `<pièce>` mais `<piece>`

## Quelques règles sur le nom des éléments

- ▶ **pas de caractère diacritique** : pas `<pièce>` mais `<piece>`
- ▶ **pas d'espace dans les noms**, puisque les espaces servent à délimiter les noms des éléments des attributs

## Quelques règles sur le nom des éléments

- ▶ **pas de caractère diacritique** : pas `<pièce>` mais `<piece>`
- ▶ **pas d'espace dans les noms**, puisque les espaces servent à délimiter les noms des éléments des attributs
- ▶ les **majuscules** sont utilisées pour **distinguer deux mots** dans un élément : `<metadonneesDoc>`

## Quelques règles sur le nom des éléments

- ▶ **pas de caractère diacritique** : pas `<pièce>` mais `<piece>`
- ▶ **pas d'espace dans les noms**, puisque les espaces servent à délimiter les noms des éléments des attributs
- ▶ les **majuscules** sont utilisées pour **distinguer deux mots** dans un élément : `<metadonneesDoc>`
- ▶ les majuscules **ne doivent pas commencer** un élément : `<MetadonneesDoc>`


## Quelques règles sur le nom des éléments

- ▶ **pas de caractère diacritique** : pas `<pièce>` mais `<piece>`
- ▶ **pas d'espace dans les noms**, puisque les espaces servent à délimiter les noms des éléments des attributs
- ▶ les **majuscules** sont utilisées pour **distinguer deux mots** dans un élément : `<metadonneesDoc>`
- ▶ les majuscules **ne doivent pas commencer** un élément : `<MetadonneesDoc>`
- ▶ Les noms d'éléments ne doivent pas non plus commencer par un chiffre.



Dans Oxygen : Nouveau document - document XML

Quelque raccourcis pour commencer :

- ▶ double clic : sélectionne un texte
- ▶ **Ctrl + E** : mettre entre balises le texte sélectionné, ou simplement insérer des balises
- ▶ **Ctrl-barre oblique** : Ajouter la même balise que précédemment. Raccourci modifiable :  
option - raccourcis clavier - modifier « encadrer par »
- ▶ Indenter/clarifier la mise en forme :  
**Document-source-formater et indenter** - **Ctrl-Maj-P** - Icône  
bleue 

## Exercice : encoder un texte court

1. Encodex le poème *L'Adieu* d'Apollinaire (fichier `adieu.txt`)
2. Affichez l'arborescence de votre document XML : Outil - éditeur d'arbre XML

## Exercice : encoder un texte plus long

1. Encodiez la première page des *Souvenirs d'histoire locale* sur Charles-Louis-Joseph Destable, par Charles Cerf, que vous trouverez sur Gallica :  
<https://gallica.bnf.fr/ark:/12148/bpt6k96742375>.  
Le texte en format txt est dans le fichier `destable.txt`
2. Observez comment oXygen valide ou invalide un document
3. Affichez l'arborescence de votre document XML

## Exercice : encoder un texte plus long

- ▶ Ne pas oublier le prologue XML
- ▶ Encoder soigneusement les métadonnées : celles du document encodé, et celles du document XML lui-même
- ▶ Encoder la structure du texte
- ▶ Encoder les particularités du document et tout élément qui paraîtra pertinent
- ▶ Indenter le fichier pour qu'il soit plus lisible

Pour cela :

- ▶ Réfléchir en amont à vos choix d'encodage
- ▶ Partir de la structure globale pour aller ensuite vers une structuration plus fine
- ▶ Trouver des noms d'éléments parlant
- ▶ Utiliser des attributs et des valeurs d'attributs