Fondamentaux XML

Séance S01 · Module 1 : XML

Objectifs

1. À quoi sert XML aujourd'hui
2. Écrire un document **bien formé**
3. Choisir entre **élément** et **attribut**
4. Entités, CDATA, commentaires
5. **Espaces de noms**

XML, c'est quoi ?

- *eXtensible Markup Language*, W3C, 1998
- Les balises **ne sont pas imposées** : vous décrivez **vos** données
- Il décrit la **structure et le sens**, pas la présentation

Où le trouve-t-on ? `.docx`/`.xlsx`, SVG, RSS, sitemaps, factures électroniques (UBL), ISO 20022, `pom.xml`, `AndroidManifest.xml`...

Anatomie d'un document

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Catalogue de la médiathèque -->
<catalogue>
  <livre isbn="9782070368228" genre="roman">
    <titre>L'Étranger</titre>
    <annee>1942</annee>
    <exemplaires>3</exemplaires>
    <disponible/>
  </livre>
</catalogue>
```

Prologue · commentaire · **racine unique** · attributs · élément vide

Un document est un arbre

- **Une seule** racine
- Chaque élément a **un seul** parent
- L'ordre des éléments est significatif

`catalogue` contient `livre (@isbn, @genre)`, qui contient `titre`, `annee`, `exemplaires` et `disponible`

Les règles d'un document bien formé (1/2)

❌	✅
`<livre/><livre/>`	une racine qui les englobe
`<titre>Dune`	`<titre>Dune</titre>`
`<a><b></a></b>`	`<a><b></b></a>`
`<Titre></titre>`	même casse

Les règles d'un document bien formé (2/2)

❌	✅
`annee=1965`	`annee="1965"`
deux fois `genre=`	un seul attribut de ce nom
`< 10 & plus`	`< 10 & plus`
`<1er>`, `<mon titre>`	`<premier>`, `<mon_titre>`

> Un parseur XML **s'arrête à la première erreur**, contrairement au HTML.

Vérifier avec xmllint

```
xmllint --noout catalogue.xml
```

```
catalogue.xml:5: parser error : Opening and ending tag mismatch:
titre line 5 and livre
```

> L'erreur réelle est souvent **un peu avant** la ligne indiquée.

Élément ou attribut ?

	Élément	Attribut
Sous-structure	✅	❌
Plusieurs valeurs	✅	❌
Texte long	✅	❌
Identifiant, code, unité	possible	✅

Attribut pour les métadonnées · **Élément** pour les données

Élément ou attribut : exemple

```
<!-- ❌ -->
<livre titre="Dune" auteur="Frank Herbert" resume="Sur Arrakis..."/>

<!-- ✅ -->
<livre isbn="9782266320481" langue="fr">
  <titre>Dune</titre>
  <auteur ref="A01"/>
  <resume>Sur la planète Arrakis...</resume>
</livre>
```

Entités prédéfinies

Entité	Caractère
`<`	`<` (**obligatoire**)
`&`	`&` (**obligatoire**)
`>` `&quot;` `&apos;`	`>` `"` `'`

`&#233;` → é · `&#x20AC;` → €

> `&eacute;` et ` ` sont des entités **HTML** : elles n'existent pas en XML.

CDATA, commentaires et instructions de traitement

```
<exemple><![CDATA[ if (a < b && b > 0) {} ]]></exemple>

<!-- commentaire : pas de "--" à l'intérieur -->

<?xml-stylesheet type="text/xsl" href="catalogue.xsl"?>
```

Espaces de noms : le problème

Le même nom peut avoir deux sens :

- `<titre>` d'un **livre**
- `<titre>` de civilité d'un **adhérent** (Mme, M.)

On lève l'ambiguïté en associant chaque nom à un **URI**.

Espaces de noms : la syntaxe

```
<catalogue xmlns="urn:bibliotech:catalogue"
           xmlns:m="urn:bibliotech:membres">
  <livre isbn="9782070368228">
    <titre>L'Étranger</titre>
  </livre>
  <m:adherent m:id="M01">
    <m:titre>Mme</m:titre>
  </m:adherent>
</catalogue>
```

Espaces de noms : à retenir

- L'URI est un **identifiant**, il n'est jamais téléchargé
- `xmlns="…"` s'applique aux **éléments** sans préfixe, **pas aux attributs**
- Le préfixe n'est **qu'un alias** : seul l'URI compte
- La portée couvre l'élément et **tous ses descendants**

XML, JSON et HTML

	XML	JSON	HTML
Balises	libres	—	fixées
Syntaxe	stricte	stricte	tolérante
Validation	DTD/XSD	JSON Schema	—
Requêtes	XPath/XSLT	jq	DOM
Contenu mixte	✅	❌	✅

JSON domine les **API**, XML les **documents** et les **échanges normalisés**.

Pièges fréquents

- Un espace **avant** `<?xml ?>`
- ` ` ou `<br>` copiés depuis du HTML
- `&` non échappé dans une URL
- La casse
- Un encodage réel différent de l'encodage déclaré

Au TP

1. Corriger 8 documents mal formés avec `xmllint`
2. Rédiger le `catalogue.xml` de **BiblioTech**
3. Bonus : convertir un livre en JSON et comparer

Questions ?

Prochaine séance : **S02 · Validation avec une DTD**