Master 2 - GL

Fiche TP 1 : Introduction à XML

🎯 Objectifs pédagogiques

À l'issue de ce TP, vous devriez être capable de :

- Comprendre le rôle et l'importance du langage **XML** dans les applications modernes.
- Identifier les **différences entre XML, HTML et SGML**.
- Créer et **valider des documents XML** conformes à une structure définie.
- Utiliser les **parseurs DOM et SAX** en Java pour lire, parcourir et afficher le contenu d'un fichier XML.

1. Contexte et motivation

Le langage **XML (eXtensible Markup Language)** est un langage de balisage générique permettant de **structurer, stocker et échanger des données** de manière standardisée et indépendante de toute plateforme.
Il est omniprésent dans les **systèmes d'information modernes**, notamment pour :

- Les **fichiers de configuration** : Android, Maven, Spring, etc.
- Les **échanges de données** : SOAP, RSS, SVG, XHTML.
- Les **applications web et mobiles** : interopérabilité et portabilité des données.

2. Exemples de lecture d'un fichier XML

📄 Exemple 1 : Lecture avec DOM

Le parseur **DOM (Document Object Model)** charge **tout le document XML en mémoire** et permet un accès **hiérarchique** à ses éléments.

Fichier : ParserDOM.java

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.parse(new File("catalogue.xml"));

NodeList livres = doc.getElementsByTagName("livre");
for (int i = 0; i < livres.getLength(); i++) {
  Element livre = (Element) livres.item(i);
  String titre = livre.getElementsByTagName("titre").item(0).getTextContent();
  String auteur = livre.getElementsByTagName("auteur").item(0).getTextContent();
  String prix = livre.getElementsByTagName("prix").item(0).getTextContent();
```

```
    System.out.println(titre + " - " + auteur + " : " + prix);
}
```

📄 **Exemple 2 : Lecture avec SAX**

Le parseur **SAX (Simple API for XML)** lit le document de manière **séquentielle**, en déclenchant des **événements** à chaque balise rencontrée. Il est plus léger et mieux adapté aux **gros fichiers XML**.

Fichier : ParserSAX.java

```
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser parser = factory.newSAXParser();

parser.parse(new File("catalogue.xml"), new DefaultHandler() {
    @Override
    public void startElement(String uri, String localName, String qName, Attributes attributes) {
        // Traitement de début d'élément
    }

    @Override
    public void characters(char[] ch, int start, int length) {
        // Lecture du contenu textuel
    }

    @Override
    public void endElement(String uri, String localName, String qName) {
        // Traitement de fin d'élément
    }
});
```

Note: Le projet est individuel et devra être remis le mercredi 22 octobre, lors de la séance de travaux pratiques.

3. Travail à réaliser

Étapes :

1. **Créer un document XML** représentant un livre structuré selon la description suivante :
    - Le livre contient :
        - Une **liste d'auteurs** (chaque auteur a un *nom* et un *prénom*).
        - Une ou plusieurs **sections**.
        - Chaque section comporte **au moins deux chapitres**.
        - Chaque chapitre contient **au moins deux paragraphes**.
        - Le livre, les sections et les chapitres possèdent chacun un **titre**.
        - Chaque paragraphe contient du **texte libre**.

2. **Créer au moins deux exemples de livres** dans le fichier XML (livres.xml).
3. **Valider** le document XML à l'aide de : Choisissez l'outil qui vous convient et avec lequel vous êtes à l'aise.
    - **VS Code**, **XML Validator Online**, ou **Oxygen XML Editor**.
4. **Lire et afficher** le contenu du document XML à l'aide de :
    - Un parseur **DOM** (ParserDOM.java).
    - Un parseur **SAX** (ParserSAX.java).

4. Travail à rendre

Docs à rendre	
livres.xml	Fichier XML valide contenant les livres
ParserDOM.java	Lecture et affichage avec DOM
ParserSAX.java	Lecture et affichage avec SAX
Rapport PDF (1–2 pages)	Présentez : – La structure XML réalisée – Les différences DOM/SAX – Les **résultats obtenus** – **Les erreurs, difficultés ou bugs rencontrés**, et comment vous les avez résolus

Cette partie est facultative, mais elle peut vous permettre d'obtenir des points supplémentaires.

- Filtrage de données (ex. : afficher uniquement les livres d'un certain auteur ou au-dessus d'un certain prix).
- Export des résultats vers un **fichier texte** ou **CSV**.
- Ajout d'une **validation par DTD ou schéma XML (XSD)**.

6. Consignes pratiques

- Vérifiez la **bien-formation** du document XML : chaque balise doit être correctement fermée.
- Respectez la **hiérarchie logique** des éléments.
- Commentez votre code Java pour expliquer les étapes clés.
- Testez votre fichier avec plusieurs parseurs pour vérifier la compatibilité.
- Notez soigneusement **les erreurs ou exceptions** rencontrées (syntaxe XML, parsing, validation…) afin de pouvoir les expliquer dans le rapport.