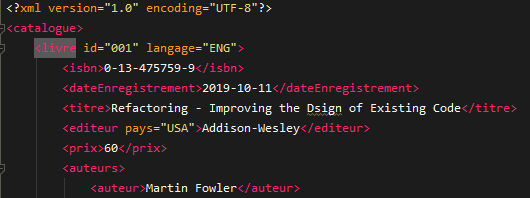
**Contexte** : nous allons aborder la manipulation de fichiers XML. XML est un format de fichiers fondamental, dont les dérivés comme HTML sont utilisés partout.

Si JSON concurrence fortement XML comme format d’échange de données, en particulier sur la partie Front end du Web (JavaScript, API Rest …), XML reste très utilisé voir incontournable pour les données orientées documentation. Cf par exemple description interface d’une application Android, données bibliothécaires …

**Exercice n°1** : nous partons d'un exemple réel qui concerne la classification internationale des livres et des auteurs pour les bibliothèques, dont XML fournit un standard.

L’objectif va être de lire et parser un fichier XML, sans utiliser de DTD, et de le convertir en objets Java, qui pourraient ensuite être utilisés dans une application, être persistés dans une base de données etc

Extrait du fichier XML à parser :



Ce fichier une fois parsé pourra être converti en objets Java, manipulé selon les besoins de l’application et sauvegardé éventuellement dans une base de données.

Le document joint XMLetParsing présente les principales notions autour de XML et du parsing utilisant la bibliothèque SAX.

Etape 1: créez un nouveau projet

Etape 2: créez une nouvelle classe Livre avec les attributs suivants :

* String : la langue, le titre, l’id, le code isbn, paysEdition et l’éditeur
* Date : la date d’enregistrement du livre
* int : le prix
* Collection de String : les auteurs

Etape 3 : ajoutez les accesseurs et un ou 2 constructeurs, avoir obligatoirement aussi un constructeur ***sans paramètre*** ;

Etape 4: créez une nouvelle classe MonParseurSax

Elle dérive de la classe **DefaultHandler**

Elle a pour attributs :

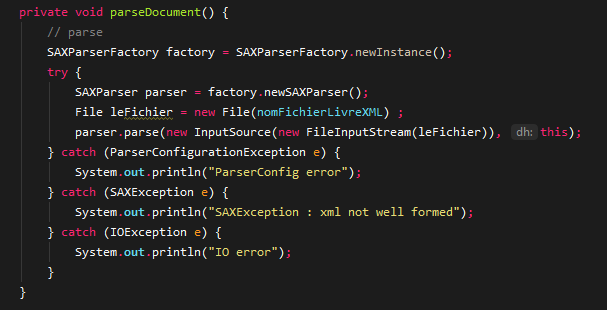
* Une liste de Livre
* Un String nommé nomFichierLivreXML
* Un attribut sdf défini comme ci-dessous
* Un Livre nommé unLivre
* Un String nommé valeurLue



Cette classe doit avoir une méthode **main** (commencez simplement à saisir main dans le code).

Etape 5 : ajout d’une méthode parseDocument()

Ajoutez une méthode parseDocument() comme ci-dessous :



Faites les imports de classe nécessaires. (on se met sur la classe en rouge, on tape Alt+Entrée et on choisit la classe à importer).

**Question 1: bloc try / catch**

A quoi sert le bloc catch / try ici ? Pourquoi a-t-on 3 blocs catch pour un seul bloc try ?

Etape 6 : copiez le fichier XML catalogue.xml dans le répertoire du projet, à la racine

Etape 7 : surcharge de la méthode **startElement**

* Utilisez IntelliJ pour overrider la méthode (code 🡪 Override Methods 🡪 startElement)
* Enlevez l’appel à la méthode mère (super)
* Implémentez les instructions suivantes

Si le qName est égal à la chaîne de caractère **livre** alors :

Instancier l’attribut unLivre

Initialisez l’attribut ***id*** de unLivre à partir de l’attribut **id** de l’élément **livre**

Initialisez l’attribut ***langue*** de unLivre à partir de l’attribut **langage** de l’élément **livre**

Remarques :

1. les attributs correspondant à l’élément sont fournies par le paramètre attributes (classe **Attributes**) de la méthode startElement.
2. Ci-dessous l’ extrait de documentation de la méthode getValue de la classe **Attributes**

**getValue**

public java.lang.String **getValue**(java.lang.String qName)

Look up an attribute's value by XML qualified (prefixed) name.

**Parameters:**

qName - The XML qualified name.

**Returns:**

The attribute value as a string, or null if the attribute is not in the list or if qualified names are not available.

Etape 7 : surcharge de la méthode *characters*

* Procédez comme à l’étape 7.
* Enlevez l’appel à la méthode mère (super)
* Implémentez l’instruction suivante

Initialisez l’attribut **valeurLue** (de type String) à partir des paramètres de la méthode characters.

Rappel :

1. la méthode *characters* du parser est appelée lorsque le parser rencontre des caractères à l'intérieur d'un élément.

Exemple :



1. Exemple de constructeur de la classe String

#### String

public String​(char[] value,

int offset,

int count)

Allocates a new String that contains characters from a subarray of the character array argument. The offset argument is the index of the first character of the subarray and the count argument specifies the length of the subarray. The contents of the subarray are copied; subsequent modification of the character array does not affect the newly created string.

**Parameters:**

value - Array that is the source of characters

offset - The initial offset

count - The length

**Throws:**

[IndexOutOfBoundsException](https://docs.oracle.com/javase/9/docs/api/java/lang/IndexOutOfBoundsException.html) - If offset is negative, count is negative, or offset is greater than value.length - count

Etape 8 : surcharge de la méthode **endElement**

* Procédez comme à l’étape 7.
* Enlevez l’appel à la méthode mère (super)
* Implémentez les instructions suivantes :

Si le paramètre qName vaut livre

alors on ajoute unLivre à la collection listeLivres

Si le paramètre qName vaut isbn

alors initialiser l’attribut isbn de unLivre avec la valeurLue (cf étape précédente).

Etape 9 : ajout d’une méthode d’affichage

Dans la même classe, ajoutez une méthode d’affichage qui parcourt la collection de livres listLivres et affiche chaque Livre correspondant.

On effectuera la modification nécessaire à la classe Livre pour que cet affichage soit pertinent

Etape 10 : Ajout d’un constructeur

Ajouter un constructeur qui prend un seul paramètre de type String.

Initialisez l’attribut nomFichierLivreXML à partir de ce paramètre.

Initialisez l’attribut listeLivres.

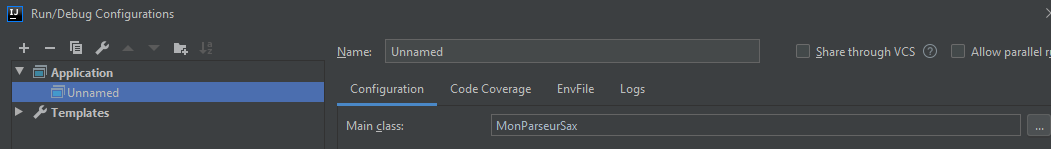
Appelez la méthode parseDocument

Appelez la méthode d’affichage créé à l’étape précédente.

Etape 11 : finalisation du main et test

Dans la méthode main, créez une instance de la classe courante avec en paramètre le catalogue de livres copié à l’étape 6.

Testez (définir une Configuration de type Application Java, exemple ci-dessous).



**Question 2 : analyse**

Pourquoi certaines balises sont lues dans la méthode startElement et d’autres dans la méthode endElement ?

Rappel : à quoi sert la méthode characters ?

**Question 3 : compléter le parseur**

Finalisez la classe en ajoutant :

Lecture de l’éditeur et du pays associé, valorisation de l’attribut correspondant au pays de l’éditeur.

Lecture et valorisation du titre, de chaque auteur, du prix et de la date d’enregistrement.

On se rappellera qu’il peut y avoir plusieurs auteurs.

Pour la date d’enregistrement, on utilisera notre sdf, instance de SimpleDateFormat. Cf par exemple <http://tutorials.jenkov.com/java-internationalization/simpledateformat.html>

**Exercice n°2 : analyse de la cohérence d'un fichier XML à partir du DTD**

Contexte :

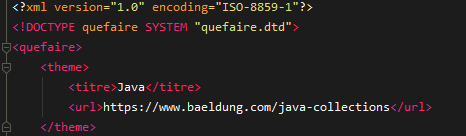
Dans ce 2ème exercice, nous allons examiner la cohérence d'un fichier XML à partir de son DTD associé.

Nous disposons de 3 fichiers, 2 fichiers XML quefaire\_dtd.xml et activites\_dtd.xml, et un fichier descripteur du XML au format DTD: quefaire.dtd.

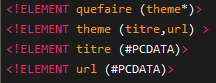
Objectif :

Le but est de créer une classe, AnalyseConformiteXMLDTDSax.java, pour vérifier si la structure d’un fichier XML est conforme à sa DTD. Le moyen sera la librairie SAX de parsing et validation.

Extrait de fichier XML en entrée



La DTD correspondant au fichier et servant à la validation



Etape 1: créer un projet Java, appelez-le comme vous voulez, et créez une classe AnalyseConformiteXMLDTDSax avec une méthode main dedans.

Etape n°2 : copiez le code suivant : methodeParsingXMLavecDTD.txt

Ce code fournit une méthode nommée **validateWithDTDUsingSAX** qui permet de valider la conformité d’un fichier XML par rapport à son DTD.

Etape n°3 :copie des fichiers XML et du DTD

Copiez les fichiers quefaire\_dtd.xml, quefaire.dtd et activites\_dtd.xml à la racine du projet.

Etape n°4 :dans la méthode main

Modifier la signature de la méthode afin de préciser qu’elle est susceptible de lever une Exception.

Ajoutez le code nécessaire dans la méthode main pour afficher dans la console le résultat de la méthode **validateWithDTDUsingSAX** avec pour paramètre quefaire\_dtd.xml (on remarquera que c’est une méthode statique).

**Question 4 : analyse du résultat**

Définissez une configuration et exécutez le programme.

Que donne le résultat ?

Remplacez quefaire\_dtd.xml par activites\_dtd.xml . Que donne le résultat ? Pourquoi ?

Consultez un des liens suivants : <https://en.wikipedia.org/wiki/Document_type_definition> (en anglais) ou OpenClassRomoms (<https://tutoriel-xml.rolandl.fr/partie-2-creez-des-definitions-pour-vos-documents-xml/introduction-aux-definitions-et-aux-dtd.html> ).

Servez-vous des documents précédents pour modifier le DTD pour qu’il marche avec les 2 fichiers XML (quefaire\_dtd.xml et activites\_dtd.xml).

**Question 5 : correction d’un DTD**

Copiez le fichier etudiant1.xml à la racine du projet

Créez sa DTD correspondante et validez.

Copiez le fichier etudiant2.xml à la racine du projet

Créez sa DTD correspondante et validez.

**Exercice n°3 : création de fichiers XML à partir de programme Java**

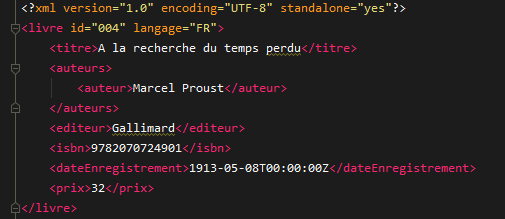
Contexte: On a vu précédemment comment lire un fichier XML en Java, avec ou sans DTD, en utilisant un parser. En l’occurrence nous avons utilisé la librairie java SAX.

Il est souvent nécessaire de faire l’opération inverse, c’est-à-dire de générer un document XML à partir de données stockées dans une BDD que l’on va manipuler avec des objets Java.

Dans cet exercice, nous allons étudier une ***nouvelle librairie java, JAXB,*** très populaire et utilisée et voir comment générer un document XML grâce à cette librairie. Dans l’exercice suivant, nous étudierons les tests unitaires dédiés à la production de documents XML, de manière plus étendue et précise que ce que nous avons fait auparavant.

Objectif:

Dans une nouvelle classe Java du 1er projet, nous allons créer une instance de Livre (A la recherche du temps perdu de Marcel Proust …), créer une méthode utilisant Jaxb pour créer le fichier suivant :



Etape 1 : reprise du 1er projet de l’exercice 1

Reprenez le projet créé pour l’exercice 1.

On va se baser sur la documentation suivante pour créer des annotations Jaxb dans notre classe existante Livre

<https://howtodoinjava.com/jaxb/jaxb-annotations/>

Les annotations pourront être directement importées (va Alt+Entrée) puisque la librairie Jaxb est incluse dans le JDK.

Etape 2 : création des annotations

Nous avons besoin de créer les annotations suivantes :

@XmlRootElement : devant le nom de la classe

@XmlAttribute : devant le getter lié à la langue et le getter lié à l’ID

Attention : il faudra modifier le nom de l’attibut qui doit s’appeler langage dans le fichier XML

@XmlElement : devant les getters du titre, isbn, editeur, prix et enregistrement.

Attention : il faudra modifier le nom de l’élément correspondant à enregistrement qui doit s’appeler dateEnregistrement dans le fichier XML

@XmlTransient : devant le getter de paysEdition

Pour les auteurs : il faudra mettre les instructions suivantes devant le getter des auteurs :

@XmlElementWrapper(name="auteurs")

@XmlElement(name="auteur")

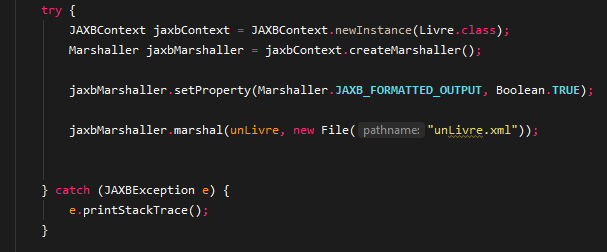
Etape 3 : création d’une classe GenereLivre

Créer une classe GenereLivre

Ajoutez une méthode main

Ajoutez une méthode jaxbVersXml qui prend en paramètre un Livre et ne renvoie rien

Dans cette méthode, ajoutez le code suivant :



Dans la méthode main, créez une instance de Livre qui corresponde à ce qu’on souhaite produire (cf objectif).

Pour la date, vous pourrez passer par un SimpleDateFormat. Pour la liste des auteurs, pensez à la méthode statique Arrays.asList pour spécifier facilement la liste (ici un seul) des auteurs.

Toujours dans la méthode main, appelez alors la méthode jaxbVersXml avec le livre défini en paramètre.

Etape 4:ajoutez une nouvelle configuration

ajoutez une nouvelle configuration (Application, GenereLivre comme classe à éxécuter).

Exécutez le programme et vérifiez que le document XML a bien été crée et est conforme aux attentes !

AVANCE :

en option, créez une classe Catalogue, avec comme attribut une liste de livres.

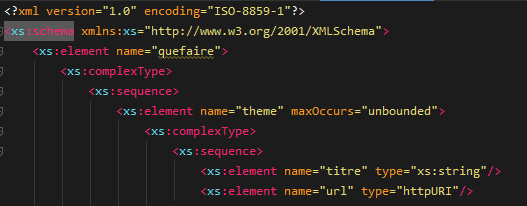
Générez un fichier catalogue.xml avec une liste de livres (1 seul suffira).

Modifiez ce qu’il faut pour pouvoir également afficher l’attribut pays de l’élément editeur

Modifiez le parseur SAX pour lire correctement une date complète.

**Exercice n°4 : validation d’un fichier XML au moyen d’un schéma XSD**

**Objectif** : après quelques questions préliminaires, nous allons reprendre le même fichier qu’à l’exercice 2 mais avec un descripteur de fichier au format XSD (XML Schema) plutôt qu’une DTD. Ci-dessous extrait du schéma :



Nous allons ensuite utiliser la librairie SAX pour valider notre document quefaire\_xsd.xml par rapport à son schéma.

Ensuite nous chercherons à créer 2 schémas XSD correspondant aux fichiers XML d’étudiants (avec les 2 variantes).

**Question 6: pertinence des XSD**

6.1. Quelles sont les limites des documents DTD ? Leurs forces et faiblesses ?

6.2. Quels sont les avantages et inconvénients des documents XSD ? Par rapport aux DTD ?

Etape 1: reprendre le projet de l’exercice n°2

Etape 2: créer une nouvelle classe AnalyseConformiteXSD.

Insérez le code fourni.

Importez les classes nécessaires.

Etape 3: ajoutez une méthode main

Dans cette méthode main, on imprimera dans la console le résultat de la méthode validateWithExtXSDUsingSAX après avoir remarqué que :

* Cette méthode est statique
* Elle prend 2 paramètres, le fichier XML à valider et le schéma XSD servant à décrire (et valider) le document XML

Etape 4 : ajout configuration d’exécution

Ajoutez une configuration de type Application avec pour classe la classe créée et exécutez le programme.

**Question 7: création de schéma de document XML**

Reportez vous par exemple à :

<http://blog.paumard.org/cours/xml/chap03-description-xml-schema.html>

7.1 Créez un schéma de document pour etudiant1

Copiez le fichier XML etudiant1\_xsd.xml dans le projet.

Créez un fichier XSD correspondant.

Validez.

7.2 Créez un schéma de document pour etudiant2

Copiez le fichier XML etudiant2\_xsd.xml dans le projet.

Créez un fichier XSD correspondant.

Validez.