JAXB 1.0

Exemple du fichier XML généré par JAXB :

```
<bibliotheque>
  <livre>
    <titre>titre 1</titre>
    <auteur>auteur 1</auteur>
    <editeur>editeur 1</editeur>
  </livre>
  <livre>
    <titre>titre 2</titre>
    <auteur>auteur 2</auteur>
    <editeur>editeur 2</editeur>
  </livre>
  <livre>
    <titre>titre 3</titre>
    <auteur>auteur 3</auteur>
    <editeur>editeur 3</editeur>
  </livre>
</bibliotheque>
```

Erreur possible :

Si le fichier répertoire du fichier XML n'existe pas une exception sera levée :

```
C:\java\jaxb>xjc -d sources test.xsd
parsing a schema...
compiling a schema...
java.lang.reflect.InvocationTargetException
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.
java:39)
```

Utilisation de JAXB :

Nécessité d'un object de type JAXBContext, qui sera le point d'entrée pour utiliser l'API JAXB.

Pour créer un tel objet, il suffit d'utiliser la méthode newInstance() qui attendra en paramètre le nom du package contenant les interfaces générées.

Exemple :

```
JAXBContext jc = JAXBContext.newInstance("com.jmdoudoux.test.jaxb");
```

Ensuite, pour créer en mémoire les objets qui représenteront notre fichier XML, il faut ç partir de l'instance du type JAXBContext, appeler la méthode createUnmarshaller() qui renvoie un objet de type Unmarshaller :

```
Unmarshaller unmarshaller = jc.createUnmarshaller();
```

Ensuite, pour parcourir le fichier il suffira de le charger en utilisant la commande suivante :

```
Bibliotheque bibliotheque = (Bibliotheque) unmarshaller.unmarshal(new
File("test.xml"));
```

Exemple d'utilisation :

```
Bibliotheque bibliotheque = (Bibliotheque) unmarshaller.unmarshal(new
File("test.xml"));

      List livres = bibliotheque.getLivre();
      for (int i = 0; i < livres.size(); i++) {
        LivreType livre = (LivreType) livres.get(i);
        System.out.println("Livre ");
        System.out.println("Titre   : " + livre.getTitre());
        System.out.println("Auteur  : " + livre.getAuteur());
        System.out.println("Editeur : " + livre.getEditeur());
        System.out.println();
      }
```

Cet exemple renverra alors le résultat suivant :

```
Livre
Titre   : titre 1
Auteur  : auteur 1
Editeur : editeur 1
Livre
Titre   : titre 2
Auteur  : auteur 2
Editeur : editeur 2
Livre
Titre   : titre 3
Auteur  : auteur 3
Editeur : editeur 3
```

Création d'un fichier XML :

Pour générer un fichier XML, on peut utiliser la classe ObjectFactory qui permettra ainsi de créer une instance des autres classes générées :

```
ObjectFactory objFactory = new ObjectFactory();

Bibliotheque bibliotheque = (Bibliotheque)
objFactory.createBibliotheque();
```

Ensuite il permet simplement d'utiliser les méthodes d'objectFactory pour remplir l'instance des objets que nous voulons générer sous le format XML :

```
List livres = bibliotheque.getLivre();
      for (int i = 1; i < 4; i++) {
        LivreType livreType = objFactory.createLivreType();
        //    LivreType livre = objFactory.createLivreType();
        livreType.setAuteur("Auteur" + i);
        livreType.setEditeur("Editeur" + i);
        livreType.setTitre("Titre" + i);
        livres.add(livreType);
      }
```

Pour la création, il faudra, comme pour parcourir le fichier, obtenir un objet du type JAXBContext grâce à la méthode newInstance().

Ensuite, nous utiliserons la méthode createMarshaller() qui permettra d'obtenir un objet de type Mashaller qui va formater le document XML.

Pour ce formatage il est possible de préciser des propriétés en utilisant la méthode setProperty() suivant de constantes telles que :

JAXB_ENCODING : Permet de préciser l'encodage du document XML

JAXB_FORMATTED_OUTPUT : Indique sous forme d'un booléen si le document XML doit être formaté.

Exemple d'utilisation :

```
public static void main(String[] args) {
    try {

      ObjectFactory objFactory = new ObjectFactory();

      Bibliotheque bibliotheque = (Bibliotheque)
objFactory.createBibliotheque();
      List livres = bibliotheque.getLivre();
      for (int i = 1; i < 4; i++) {
        LivreType livreType = objFactory.createLivreType();
        //    LivreType livre = objFactory.createLivreType();
        livreType.setAuteur("Auteur" + i);
        livreType.setEditeur("Editeur" + i);
        livreType.setTitre("Titre" + i);
        livres.add(livreType);
      }
      JAXBContext jaxbContext =
JAXBContext.newInstance("com.jmdoudoux.test.jaxb");
      Marshaller marshaller = jaxbContext.createMarshaller();
      marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, new
Boolean(true));
      Validator validator = jaxbContext.createValidator();

      marshaller.marshal(bibliotheque, System.out);
    } catch (Exception e) {
    }
  }
}
```

Autres versions :

JAXB 2.0 et annotations : https://www.jmdoudoux.fr/java/dej/chap-jaxb.htm#:~:text=47.1.,JAXP%20(SAX%20ou%20DOM)

JAXWS / WSDL :

https://www.codeflow.site/fr/article/jax-ws

https://thierry-leriche-dessirier.developpez.com/tutoriels/java/creer-tester-webservice-jaxws-soapui-5-min/

JAX-WS est une API normalisée permettant de créer et de consommer des services WEB SOAP (Simple Object Access Protocol).

SOAP est une spécification WML permettant d'envoyer des messages sur un réseau.

JAX-WS est un framework qui simplifie l'utilisation de SOAP, faisant partie de Java Standard. Car SOAP est lourd en XML et il est donc préérable de l'utiliser avec des outils / frameworks.