TP N° 05

Objectif

L'objectif de cette série de TP est d'introduire une API de Mapping Objet/XML. Ces API permettent de passer d'une manière (presque) transaprente d'une modélisation Objet vers Xml et vice-versa. Nous prennons comme exemple l'API JAX-B de la plateforme Jakarta EE (anciennement Java EE).

Principe de fonctionnement

JAX-B utilise les annotations sur les méthodes d'un JavaBeans pour spécifier les différents éléments du document XML (racine, éléments, attributs).

Pour exploiter la classe annotée, il faut faire appel à deux méthodes :

- "Marshaller" : cette classe permet de créer un document XML à partir d'un objet Java (JavaBean),
- "Unmarshaller" : cette classe permet de faire la conversion dans le sens inverse, c'est à dire, elle permet de créer un objet Java (JavaBean) à partir d'un fichier XML.

Les annotations à utiliser sont :

- XmlRootElement : pour préciser la racine du fichier XML,
- XmlAttribute : si l'attribut Java doit apparaître comme un attribut dans le fichier XML,
- XmlElement : si l'attribut Java doit apparaître comme un élément XML,
- XmlElementWrapper : permet de regrouper une collection d'éléments (généralement de même type),
- XmlTransient : si l'attribut Java ne doit pas apparaître dans le fichier XML.

Exemple

Soit la classe "Etudiant" suivante :

```
public class Etudiant {
    int id;
    String nom;
    String prenom;

    public Etudiant() {
    }

    public Etudiant(int id, String nom, String prenom) {
        this.id = id;
        this.nom = nom;
        this.prenom = prenom;
    }

}
```

Pour définir les différents éléments XML, nous ajoutons les annotations comme suit :

```
public class Etudiant implements Serializable {

    @XmlAttribute
    int id;

    @XmlElement
    String nom;

    @XmlElement
    String prenom;

    public Etudiant() {
    }

    public Etudiant(int id, String nom, String prenom) {
        this.id = id;
        this.nom = nom;
        this.prenom = prenom;
    }

}
```

Pour stocker un Etudiant dans un fichier, il faut créer un Marshaller :

```
JAXBContext contexte = JAXBContext.newInstance(Etudiant.class);
Marshaller marshaller = contexte.createMarshaller();
```

La dernière étape consiste à appeler la méthode “marshall” en lui passant, comme paramètres, l'objet à sauvegarder (une instance de classe Etudiant dans notre cas) et le fichier à créer (une instance de la classe File) :

```
marshaller.marshal(etudiant, fichier);
```

Le résultat est le fichier XML suivant :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<etudiant id="1">
    <nom>Benomar</nom>
    <prenom>Omar</prenom>
</etudiant>
```

Enoncé

Nous avons la classe “Cient” avec les attributs suivants :

- id_client,
- nom,
- prenom,
- téléphone,
- adresse

Chaque client possède plusieurs véhicules qui seront stockés dans une List (ArrayList par exemple). Un véhicule se caractérise par :

- id_vehicule
- marque
- modèle
- couleur

Travail demandé

Il est demandé d'écrire un programme qui crée un client (un exemple d'un client qui possède plusieurs véhicules), et l'enregistre, par la suite, dans un fichier XML en utilisant JAX-B,