# Travaux Dirigés de JEE n°2

**Cours de Master 2 Informatique**

——

## En attendant le printemps. . .Mavenisation et Hibernation

L'ensemble du code, des noms de classes, des champs, des méthodes doivent être en anglais. Le code doit répondre aux normes SUN et être obligatoirement testé unitairement.

Le rendu du TD doit être fait par mail à l'adresse suivante : `loyaute@univ-mlv.fr`. Il s'agit de rendre un fichier zip constitué d'un rapport au format pdf et du code source dans un jar non exécutable

---

▸ **Exercice 1.** *Mavenisation d'un projet*

*Nous allons commencer par "Maveniser" le projet que nous allons développer durant les prochains TD. Pour ce faire, vous allez télécharger (utilisateur* `umlv-jee-course-read-only`*) le Superpom sur la repository suivante :*

*`http://umlv-jee-course.googlecode.com/svn/trunk/superpom`*

*Le Superpom permet de définir les versions qui doivent être utilisées pour l'ensemble des projets d'une entreprise. Tout les projets vont faire référence à ce parent et vont déclarer les dépendances dont ils ont besoin dans leur(s) fichier(s) pom.xml. Ce projet récupéré dans Eclipse, lancez dans Eclipse la commande :*

*mvn install*

*Cette commande va vous permettre d'installer au sein de votre repository maven ce projet et vous pourrez l'utiliser par la suite.*

*Nous allons maintenant créer un projet de type Maven dans Eclipse. Ce projet sera le projet principal et sera enrichi au fur et à mesure de projets modules (persistance, service, web, etc.).*

*Rajoutez dans votre fichier pom.xml les lignes suivantes :*

```
<parent>
        <groupId>fr.umlv.m2.jee</groupId>
        <artifactId>superpom</artifactId>
        <version>1.0.1</version>
    </parent>
```

*Ces lignes vous permettent d'utiliser l'ensemble des dépendances définies dans le Superpom sans avoir a définir les versions.*

*Dans un second temps vous allez rajouter un module qui s'appellera persistance via Eclipse. Observez les modifications sur votre pom.xml principal.*

*Nous allons maintenant lancer la compilation de votre module. Pour se faire cliquez sur votre module, puis faites "Maven...' tapez ensuite la règle "compile". Pour lancer les tests unitaires, tapez la règle "test". Pour l'installation (au sein de votre repository maven), packaging sous forme de jar, tapez "install".*

*Dans les dépendances, vous pouvez configurer le scope des dépendances. Par exemple, test pour dire que la dépendance n'est utile/utilisée que pour les tests unitaires. Provided pour spécifier que la librairie ne doit pas être ajoutée lors de la constitution du jar ou du war, etc.*

▸ **Exercice 2.** *Des auteurs...*

*Dans un premier temps, vous allez implanter la classe* `Author` *définie via les caractéristiques suivantes :*

1. *Un numéro de sécurité social (ssid) ;*
2. *Un nom de famille (lastname) ;*
3. *Un prénom (firstname) ;*
4. *Une date de naissance (birthday) ;*

*Pour la date de naissance vous utiliserez la librairie joda-time.*

▸ **Exercice 3.** *...et des livres*

*Vous allez maintenant implanter la classe* `Book` *définie par les caractéristiques suivantes :*

1. *Un numéro isbn (isbn) ;*
2. *Un titre (title) ;*
3. *Une date d'impression (printedDate) ;*
4. *Des auteurs (authors) ;*

*Pour la date d'impression vous utiliserez la librairie joda-time.*

▸ **Exercice 4.** *Annotation JPA pour enrichir nos beans métiers*

*Nous allons maintenant enrichir nos objets métiers* `Author` *et* `Book` *avec les annotations JPA en vue de les persister. Pour se faire, nous allons tout d'abord utiliser les deux annotations* `Entity` *et* `Table` *qui permettent de définir, respectivement, qu'une classe est une entité à persister et le nom de la table qui va contenir les données.*

*Dans un second temps, nous allons définir un identifiant technique (i.e. la clé primaire) à l'aide de l'annotation* `@Id`*. Nous laissons pour le moment Hibernate géré la séquence de génération de la clé primaire. Le nom de la colonne contenant cette clé primaire va être définie à l'aide de l'annotation* `@Column`*.*

*Nous allons maintenant définir l'ensemble des colonnes pour les caractéristiques d'un auteur et d'un livre. Concernant la date de naissance (classe* `Author`*) et la date d'impression (classe* `Book`*), il faut utiliser le package fournie avec la librairie joda-time pour la persistance et utiliser l'annotation* `@Type` *pour spécifier la "transformation" d'une* `DateTime` *de joda-time vers une "Date" base de donnée.*

▸ **Exercice 5.** *Ecriture des accès aux données*

*Téléchargez (utilisateur* `umlv-jee-course-read-only`*) maintenant la DAO générique fournie sur la repository suivante :*

*http://umlv-jee-course.googlecode.com/svn/trunk/umlv-dao*
*http://umlv-jee-course.googlecode.com/svn/trunk/umlv-hibernate*

*Attention à bien récupérer les deux projets !*

*L'ensemble des DAO que vous allez maintenant écrire (interface et implantation) vont étendre de l'interface de la DAO générique et de la classe d'implantation utilisant Hibernate.*

*Ecrivez les DAO pour la classe* `Author` *et* `Book`*.*

*Un certain nombre de méthodes sont déjà disponibles via la DAO générique. Nous enrichirons ces méthodes lors du TD suivant. Faites les tests...*