Hibernate

Première Partie (mise en place)

En premier lieu, il s'agit d'enrichir notre application de Phone Contacts. Reprenez la page **addContact.jsp**. Celle-ci doit maintenant proposer un formulaire qui contient, en plus des champs que vous avez déjà implémentés en TP Servlet/JSP, les informations suivantes : L'adresse du contact (voir classe **Address** dans le diagramme de classe ci-dessous), ces différents numéros de tél (voir classe **PhoneNumber**, on supposera qu'il en a trois max i.e. mobile, maison, bureau et qui sont optionnels) et finalement les groupes auxquels il peut appartenir (Amis, collègues, famille, etc. voir la classe **ContactGoup**, on peut imaginer que certains groupes seront proposés par défaut au moment du rajout du contact et que si ce choix est insuffisant, donner la possibilité à l'utilisateur de spécifier un nouveau groupe), sachant qu'un contact peut faire partie de plusieurs groupes en même temps. Le formulaire doit également avoir un bouton « submit » et un bouton « reset ».

A cet effet, vous devez rajouter dans votre package « **domain** » toutes les classes du diagramme donné ci-dessous (sauf Entreprise). Pour chaque classe, générez les getters et setters de ses attributs et définissez un constructeur sans paramètres. Pour les classes DAO, on verra par la suite. Dans tous les cas c'est à vous de voir si vous aurez besoin d'en définir ou pas. Pas la peine de le demander à votre prof.

Ensuite configurer votre environnement en suivant les étapes décrites dans la section Requirements.

Une fois l'environnement prêt, votre premier contact avec Hibernate sera de :

1- Créer un fichier de mapping (**.hbm.xml**) par classe métier présente dans le diagramme UML. **IMPORTANT : le fichier de mapping doit se trouver dans le même répertoire que la classe métier**. Vous devez prendre en compte non seulement les propriétés de la classe mais également ses associations ainsi que leurs navigabilités et leurs multiplicités. C'est ici pour vous l'occasion d'apprendre à écrire un fichier de mapping et surtout de maîtriser les one-to-one, one-to-many, etc.

ATTENTION :

- **Faire étape par étape, i.e. d'abord créer le fichier de mapping XML pour une seule classe sans association puis le tester. Ensuite rajouter les associations une à une et tester à chaque fois sinon ça devient très dur d'identifier les sources des conflits en cas d'erreurs.**

- **Ne pas utiliser d'utilitaires pour cet exercice. Le jour de l'examen, on ressentira la différence entre ceux qui le font manuellement et ceux qui utilisent un outil!!**

2- Créer le fichier de configuration Hibernate (**.cfg.xml**). **IMPORTANT : ce fichier doit se trouver la racine de votre répertoire source du projet (src).** C'est dans ce fichier que vous allez donner les informations sur la data source (BD), les classes à mapper, le mode d'utilisation de la base (création, mise à jour, etc.).

3- Une fois les étapes 1 et 2 réalisées, faites un test pour voir si les tables sont automatiquement générées dans la base de données. Ne passez pas à la deuxième partie tant que cette étape n'est pas réalisée. **Pour tester la création des tables par hibernate il faut :**

- Que la base existe au niveau de votre SGBD

- Qu'elle n'est aucune table. C'est hibernate qui va les créer pour vous
- Rajouter ces lignes de code dans un des DAO (après avoir commenter tout le code JDBC) :

```
Session session = null;

try{

SessionFactory sessionFactory =
new Configuration().configure().buildSessionFactory();


session = sessionFactory.openSession();

} catch(Exception e){
 System.out.println(e.getMessage());

}
```

A ce niveau, vous aurez appris à écrire des fichiers de configuration, des fichiers de mapping classes=> tables et Java Beans.

Deuxième Partie (application)

Dans cette partie, l'idée est que vous commenciez à utiliser concrètement Hibernate dans votre code.

1- Commencez par enlever ou commenter votre code JDBC (qui se trouve normalement dans les classes DAOxxx) et à le remplacer par du code Hibernate **(voir exemple du cours)**. Vous devez le faire pour toutes les parties de votre application (ajouter contact, supprimer ou rechercher un contact à travers son identifiant, etc.) Ici, si vous avez besoin de définir de nouveaux DAO faites-le ! Encore une fois, on compte sur vous pour enrichir votre application de la manière la plus originale qu'il soit (fonctionnalités avancée, nouvelles fonctionnalités, etc.).

2- Dans cette étape, il vous ait demandé de pratiquer l'héritage. Pour cela rajouter la classe **Entreprise** au package « **domain** » et tout ce que cela implique (fichier de mapping, génération des getters, setters, constructeurs, etc.). Ensuite optez pour une des stratégies d'héritage fournies par Hibernate et implémentez-là dans le contexte de votre projet. Vérifiez le résultat au niveau de la base de données.

3- Modifiez maintenant votre formulaire de création d'un nouveau contact pour y inclure l'information si c'est un contact ou bien une entreprise. Pour les entreprise il y'aura une information de plus qui est le numéro de SIRET.

4- Au niveau de votre code de traitement arrangez-vous pour avoir des instances de contact et des instances d'entreprise dans la base de données.

Troisième Partie (HQL)

Dans cette dernière partie, il s'agit de s'exercer à utiliser le langage HQL pour interroger la base de données. Pour cela, vous devez avoir quelque part dans votre code des requêtes HQL (pas de requêtes trop simples SVP).

- Des requêtes de type from…..jointure, etc.
- Des requêtes avec Critères
- Des requêtes avec l'Exemple.

Requirements Environnement Servlet/JSP + Hibernate :

Pour Servlet/JSP

Télécharger la dernière release d'Eclipse version JEE développeur à l'adresse suivante : eclipse.org -> downloads->Eclipse for JEE dev. Cette version doit contenir WTP (Web Tool Platform)

Télécharger la dernière version du JDK
Sur : http://developers.sun.com/downloads/

Après installation (de préférence directement sur C: ou bien C:\MonRep\JDK1.X), assurez vous d'avoir une variable d'environnement JAVA_HOME qui pointe vers le répertoire d'installation du JDK.

Rajoutez à la variable PATH de votre environnement l'entrée suivante : %JAVA_HOME%\bin

Télécharger la dernière version de Tomcat et déziper là quelque part sur votre disque. Avoir un c:/tomcat6.0/ serait pas mal.

Tester Tomcat en lancant le serveur à partir de son répertoire bin puis sur firefox, tapez http://localhost:8080/
La fenêtre Tomcat est sensée s'afficher. Pour l'administration allez sur Tomcat Manager

Télécharger la dernière version de MySQL database server Sur : http://dev.mysql.com/downloads/
Pour pouvoir y accéder partout depuis une fenêtre Dos, rajouter une entrée dans votre variable d'environnement PATH : c:\MonRepInstall\...\MySQL Server 6.0\bin;

Lors de l'installation faire attention au login/mot de pass à définir pour la base. Choisir par exemple root (login) et root (password) ou bien root(login) et (rien du tout) comme password

Télécharger la dernière version du driver MySQL pour Java sur :
http://dev.mysql.com/downloads/connector/j/

Placez le .jar du connector JDBC dans le repertoire WebContent/WEB-INF de votre projet web

Finalement, bouton droit sur votre projet-> properties-> Java Build Path -> Libraries -> add External Jars -> puis pointer vers les servlet*.jar et servlet*jsp*.jar qui se trouvent dans le répertoire plugin de votre éclipse.

Pour Hibernate

Télécharger la dernière version stable d'Hibernate Tools for Eclipse sur:
http://jboss.org/tools/download.html
Vous aurez le choix entre télécharger un zip ou bien de passer par le « update site » d'Eclipse (conseillé).

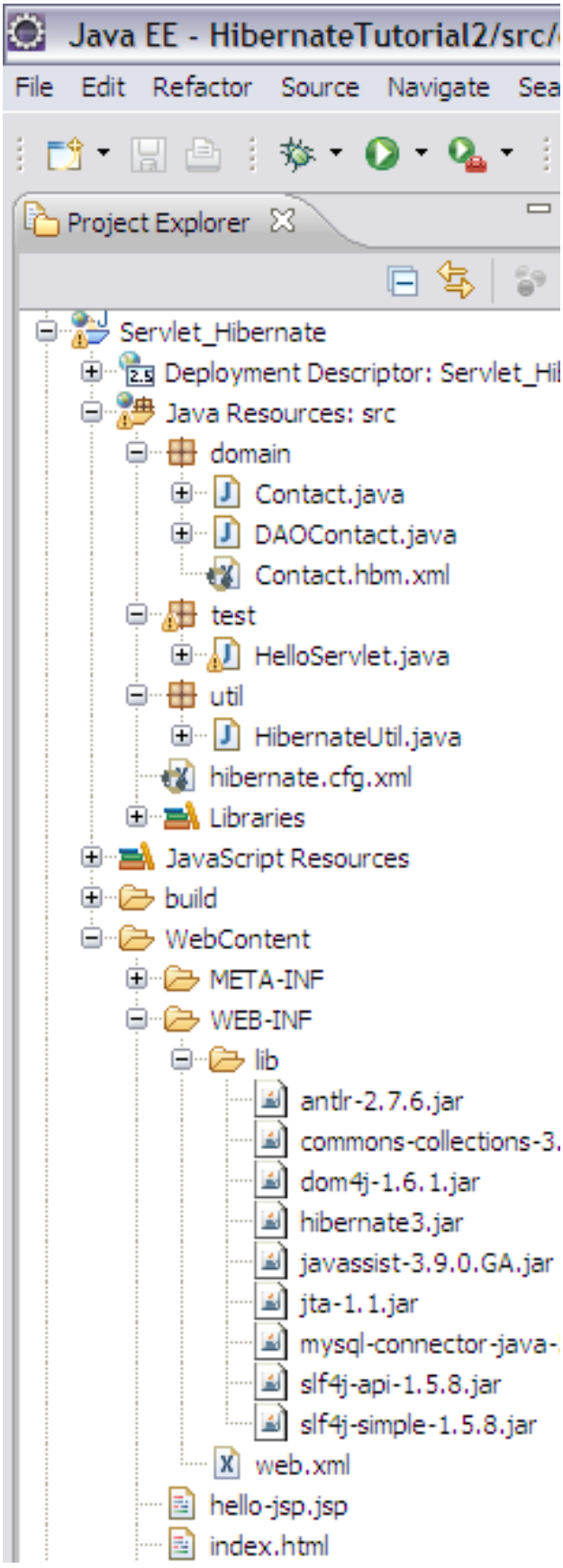
Pour l'installer, juste déziper et copie/coller le contenu du package « plugin » à destination du package du même nom d'Eclipse. Même chose pour le répertoire « features »
Attention : si vous passer via update site, ne téléchargez pas tout JBoss Tool, choisissez seulement Hibernate Tool

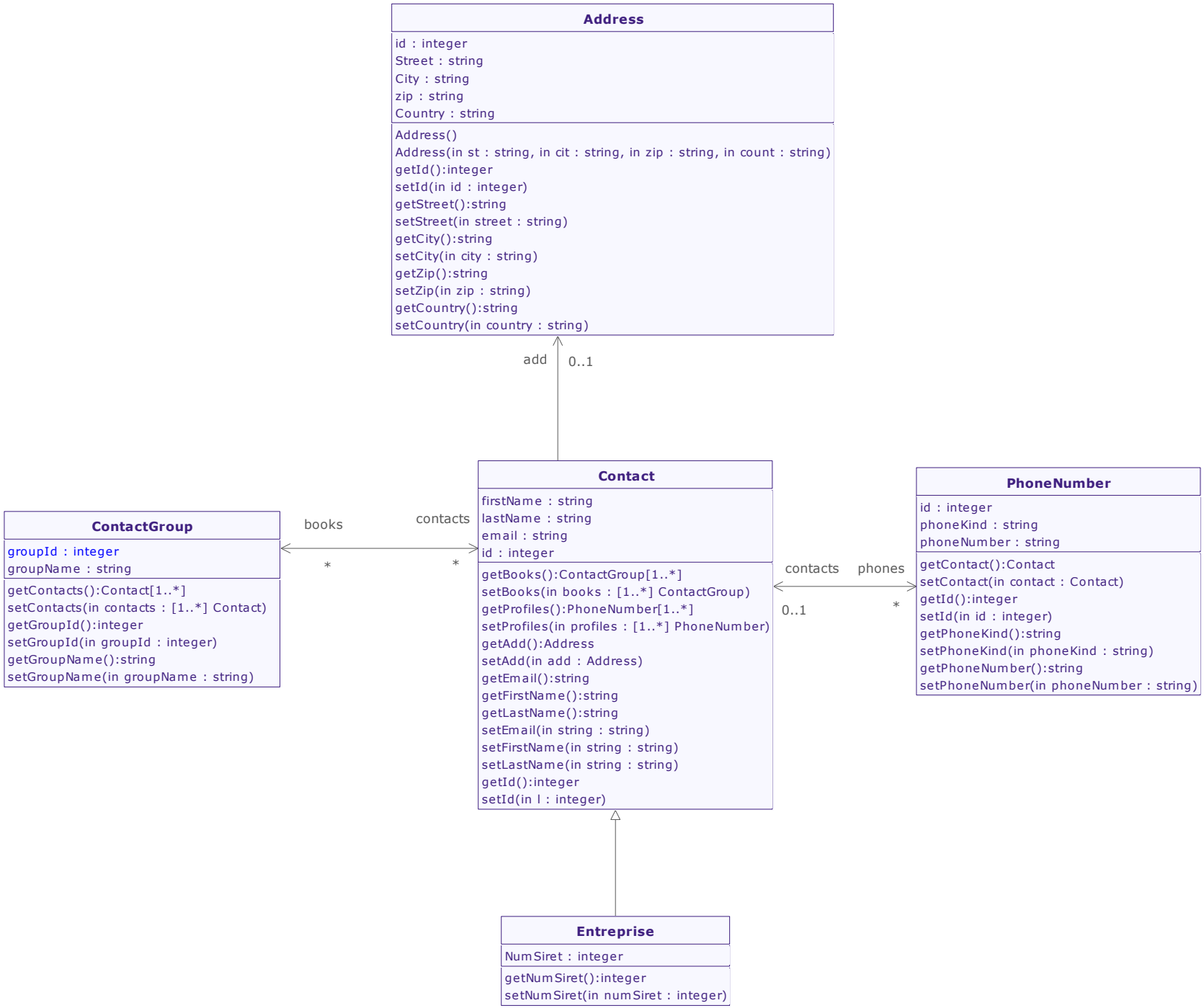
Télécharger et déziper la dernière version stable d'hibernate framework i.e. Hibernate – Core sur: http://www.hibernate.org/downloads
IMPORTANT: Assurez-vous que votre projet pointe vers le fichier hibernate(+num version).jar (qui se trouve dans le répertoire Hibernate-Core que vous venez juste de déziper). Pour cela : (votre projet->B. Droit->properties->Java Build Path->add external Jars->parcourir et pointer vers le hibernate(NumVersion).jar). **En plus :** Assurez-vous que votre projet pointe vers **TOUS** les fichiers .jar qui se trouvent dans les répertoires **lib/required** et **lib/jpa** du répertoire Hibernate Core.
ATTENTION : Si vous êtes dans le contexte d'un projet web i.e. Servlet/JSP (dans votre cas oui), vous pouvez sauter cette étape et à la place mettre tous les jars décrits précédemment dans le package **webcontent/WEB-INF/lib** de votre projet Web.

Télécharger et déziper la dernière version stable de slf4j sur : http://www.slf4j.org/download.html hibernate utilise cette API pour les logs.
Assurez vous que votre projet pointe sur les jars suivants : slf4j-api-numVersion.jar (il se trouvait déjà dans lib/required d'hibernate core mais ce n'est pas grave) et slf4j-simple-numVer.jar (votre projet->B. droit ->properties->Java Build Path->add external Jars->parcourir et pointer les 2 jars).
ATTENTION : Si vous êtes dans le contexte d'un projet web i.e. Servlet/JSP (dans votre cas oui), vous pouvez sauter cette étape et à la place mettre tous les jars décrits ci-dessus dans le package webcontent/WEB-INF de votre projet Web.

Optionnellement vous pouvez utiliser Toad pour visualiser votre base de donnés. Beaucoup plus pratique que la simple vue Dos.
http://www.quest.com/toad-for-mysql/

Pour l'arborascence du projet dans eclipse et pour l'emplacement des fichiers, .jar, etc. voir la figure ci-dessous :


Java EE - HibernateTutorial2/src/
File Edit Refactor Source Navigate Sea
Project Explorer
Servlet_Hibernate
Deployment Descriptor: Servlet_Hi
Java Resources: src
domain
Contact.java
DAOContact.java
Contact.hbm.xml
test
HelloServlet.java
util
HibernateUtil.java
hibernate.cfg.xml
Libraries
JavaScript Resources
build
WebContent
META-INF
WEB-INF
lib
antlr-2.7.6.jar
commons-collections-3.
dom4j-1.6.1.jar
hibernate3.jar
javassist-3.9.0.GA.jar
jta-1.1.jar
mysql-connector-java-
slf4j-api-1.5.8.jar
slf4j-simple-1.5.8.jar
web.xml
hello-jsp.jsp
index.html


Address
id : integer
Street : string
City : string
zip : string
Country : string
Address()
Address(in st : string, in cit : string, in zip : string, in count : string)
getId():integer
setId(in id : integer)
getStreet():string
setStreet(in street : string)
getCity():string
setCity(in city : string)
getZip():string
setZip(in zip : string)
getCountry():string
setCountry(in country : string)
add
0..1
Contact
firstName : string
lastName : string
email : string
id : integer
getBooks():ContactGroup[1..*]
setBooks(in books : [1..*] ContactGroup)
getProfiles():PhoneNumber[1..*]
setProfiles(in profiles : [1..*] PhoneNumber)
getAdd():Address
setAdd(in add : Address)
getEmail():string
getFirstName():string
getLastName():string
setEmail(in string : string)
setFirstName(in string : string)
setLastName(in string : string)
getId():integer
setId(in l : integer)
ContactGroup
groupId : integer
groupName : string
getContacts():Contact[1..*]
setContacts(in contacts : [1..*] Contact)
getGroupId():integer
setGroupId(in groupId : integer)
getGroupName():string
setGroupName(in groupName : string)
books
contacts
*
*
contacts
phones
0..1
*
PhoneNumber
id : integer
phoneKind : string
phoneNumber : string
getContact():Contact
setContact(in contact : Contact)
getId():integer
setId(in id : integer)
getPhoneKind():string
setPhoneKind(in phoneKind : string)
getPhoneNumber():string
setPhoneNumber(in phoneNumber : string)
Entreprise
NumSiret : integer
getNumSiret():integer
setNumSiret(in numSiret : integer)