Conception et développement d'applications Internet MASTER 2 - SITN Bekhouche Abdesslem/Sobral Diogo	TP3 : EJB

Exercice 1: Entity Bean

L'objectif de cet exercice est de créer un entity bean permettant de mapper une table de la base de données locale.

1. Créer une table LIVRES dans la base de données local "sample" (Onglet Services -> Database -> JavaDB-> Sample sous NetBeans).
La table aura un identifiant qui sera généré automatiquement, un titre, un auteur.

Exemple/Correction : Table Livres (auteur *varchar(64)*, titre *varchar(64)*, numero *number*)

2. Créer un entity bean correspondant à la table LIVRES.

Exemple/Correction :

```
@Entity
@Table (name="LIVRES")
public class Livres implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @Column(name = "numero")
    private Integer numero;

    @Column(name = "titre")
    private String titre;

    @Column(name = "auteur")
    private String auteur;

...
```

3. Modifier (ou le créer) le fichier "persistence.xml" permettant de configurer l'entity manager.

<!-- nom de la persistance unit et la transaction type JTA. Si JTA faut que la data source soit déclarée au niveau serveur. Aller dans la console d'administration et déclarer sa data source et son connection pool-->

```
<persistence-unit name="PU_NomADefinir" transaction-type="JTA">

      <!-- l'implémentation de jpa, ici eclipse -->
       <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
<!-- L'entity quon gère -->
       <class>metier.entity.Livres</class>
       <properties>


<!-- les propriétés de connexion à votre base de données, ici une base de données derby,
attention le driver client derby doit être contenu dans en tant que librairie dans votre projet
EJB, ou autre client si vous utilisez une autre BDD -->
          <property name="serverName" value="localhost"/>
          <property name="portNumber" value="1527"/>
          <property name="databaseName" value="derby"/>

          <property name="javax.persistence.jdbc.user" value="app"/>
          <property name="javax.persistence.jdbc.password" value="app"/>

          <property name="javax.persistence.jdbc.url"
value="jdbc:derby://localhost:1527/sample"/>
          <property name="driverClass" value="org.apache.derby.jdbc.ClientDataSource"/>
          <property name="javax.persistence.jdbc.driver"
value="org.apache.derby.jdbc.ClientDataSource"/>

<!-- pour avoir des logs au niveau du serveur -->
          <property name="eclipselink.logging.level" value="FINE"/>
       </properties>
   </persistence-unit>
```

4. Utiliser un client lourd en utilisant un entity manager permettant d'insérer plusieurs livres en base.

Exercice 2: Accès via servlet et jsp

L'objectif de cet exercice est de créer, lister et supprimer des livres à partir d'un client léger.

1. Côté "métier" :

- Le session bean

```
@Stateless
public class GestionLivre implements GestionLivreLocal {

   @PersistenceContext(unitName="PU_NomADefinir")
```

EntityManager em;

…

//Utilisation de l'entity Livres

- L'interface GestionLivreLocal

2. Côté serveur Web : Servlet : on déclare l'EJB à utiliser et on l'utilise

- @EJB

private GestionLivreLocal manager;

- Puis on utilise l'EJB en appelant ses méthodes

Il faut faire pareil si l'interface est Remote au lieu de local modulo les annotations.