A63 – Persistance des données – TP1

1. Mise en place

Dans note cas nous allons travailler avec une base de données mySQL. On crée alors une base de données nommée Toplink. Celle-ci aura une table « Personne » avec un « id », un « nom » et un « prénom ».

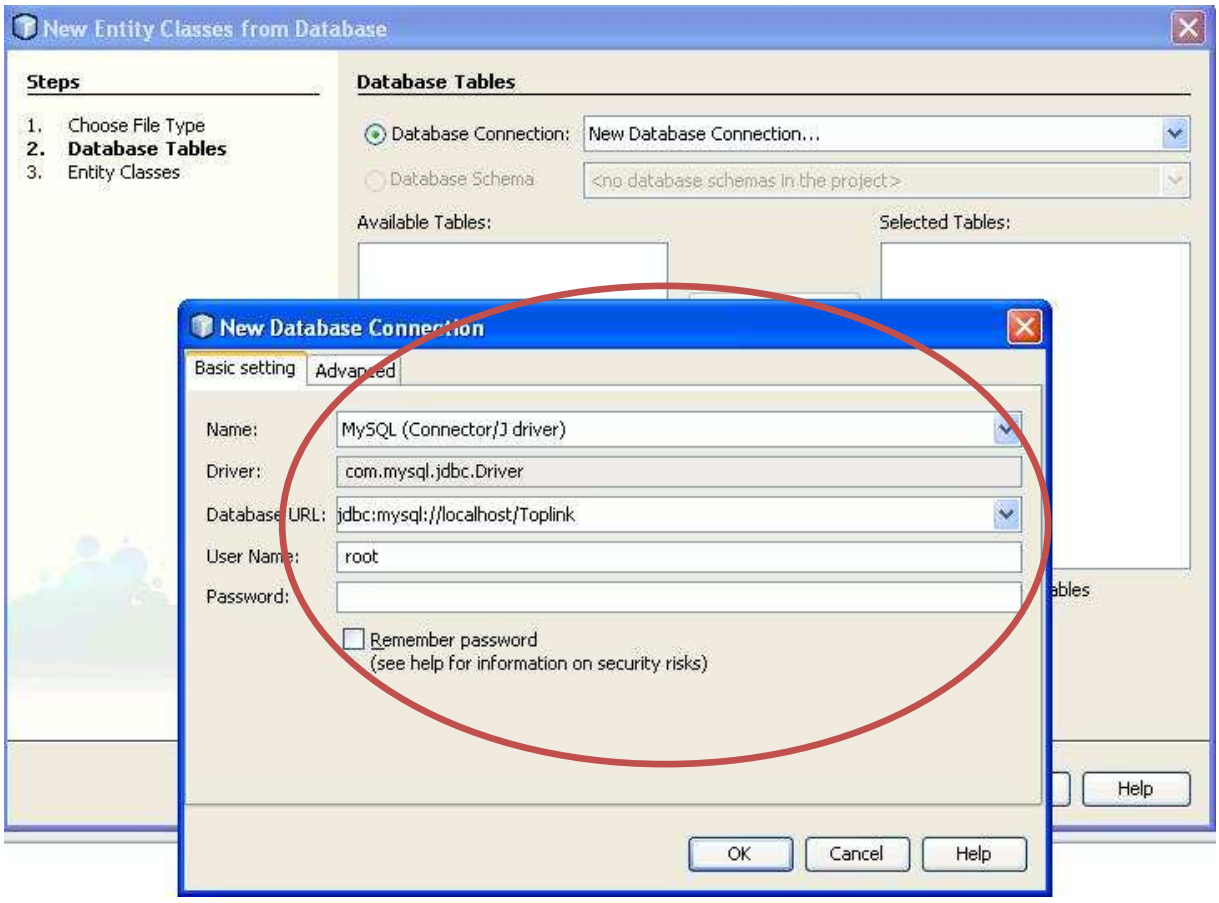
Après, il faut télécharger le JAR permettant la mise en place de persistance des données, ici nous allons utiliser Toplink JPA qui se trouve à l'adresse suivante : http://www.oracle.com/technology/products/ias/toplink/jpa/download.html

On décompresse le fichier téléchargé pour former un JAR.

On ouvre ensuite netbeans et on créer un projet. On ajoute ce JAR aux librairies du projet.

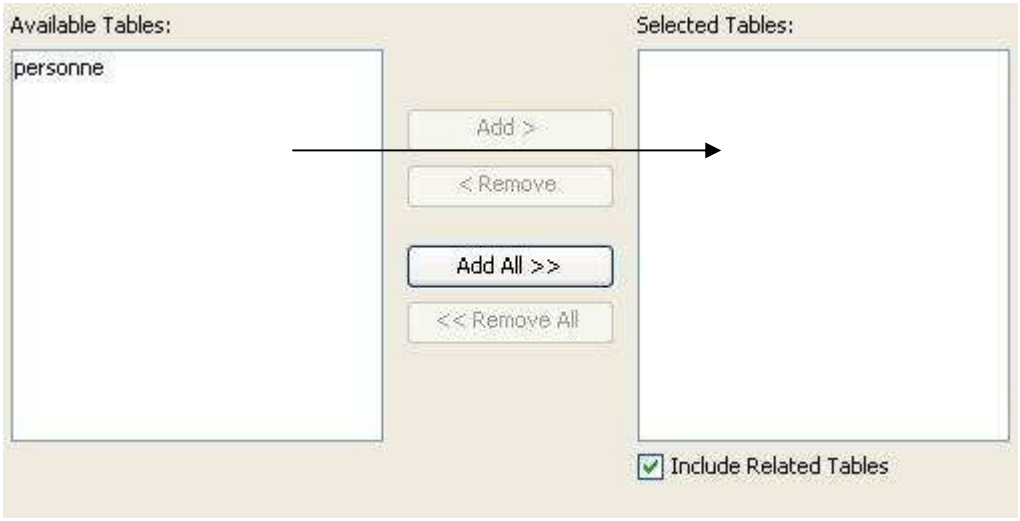
Il faut aussi chercher le driver JDBC correspondant à mySQL si celui-ci n'est pas encore présent.

On ajoute une nouvelle « Entity class from database ». Dans la fenêtre qui s'ouvre, on choisit « new database connection ». On voit alors ceci :



On choisit driver le nom « MySQL », on remplit la « Database URL » telle que sur l'image, et le User Name qui est « root ».

On choisit ensuite une table à ajouter au projet :



Il faut aussi cliquer sur « Create Persistence Units », il créera alors le fichier xml et une Persistence Unit.

Netbeans crée ensuite automatiquement une classe correspondant à la table Personne :

```
package toplinkjpa;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

/**
 *
 * @author Lionel
 */
@Entity
@Table(name = "personne")
@NamedQueries({@NamedQuery(name = "Personne1.findById", query = "SELECT p FROM
Personne1 p WHERE p.id = :id"), @NamedQuery(name = "Personne1.findByNom", query = "SELECT p
FROM Personne1 p WHERE p.nom = :nom"), @NamedQuery(name = "Personne1.findByPrenom",
query = "SELECT p FROM Personne1 p WHERE p.prenom = :prenom")})
public class Personne1 implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Column(name = "id", nullable = false)
```

```
private Integer id;
@Column(name = "nom", nullable = false)
private String nom;
@Column(name = "prenom", nullable = false)
private String prenom;

public Personne1() {
}

public Personne1(Integer id) {
   this.id = id;
}

public Personne1(Integer id, String nom, String prenom) {
   this.id = id;
   this.nom = nom;
   this.prenom = prenom;
}

public Integer getId() {
   return id;
}

public void setId(Integer id) {
   this.id = id;
}

public String getNom() {
   return nom;
}

public void setNom(String nom) {
   this.nom = nom;
}

public String getPrenom() {
   return prenom;
}

public void setPrenom(String prenom) {
   this.prenom = prenom;
}

@Override
public int hashCode() {
   int hash = 0;
   hash += (id != null ? id.hashCode() : 0);
   return hash;
}
```

```
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Personne1)) {
            return false;
        }
        Personne1 other = (Personne1) object;
        if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "toplinkjpa.Personne1[id=" + id + "]";
    }

}
```

Et le fichier xml associé au projet :

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="Toplink_JPAPU" transaction-type="RESOURCE_LOCAL">
    <provider>oracle.toplink.essentials.PersistenceProvider</provider>
    <class>toplinkjpa.Personne</class>
    <properties>
      <property name="toplink.jdbc.url" value="jdbc:mysql://localhost/Toplink"/>
      <property name="toplink.jdbc.user" value="root"/>
      <property name="toplink.jdbc.driver" value="com.mysql.jdbc.Driver"/>
      <property name="toplink.jdbc.password" value=""/>
    </properties>
  </persistence-unit>
</persistence>
```

Avant de pouvoir travailler sur la base de données, il faut importer les différentes classes nécessaires dans la classe Main :

```
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
```

et déclarer les différents en dessous de la déclaration de la classe Main:

```
private static EntityManagerFactory emf;
private static EntityManager em;
private static EntityTransaction trans;
```

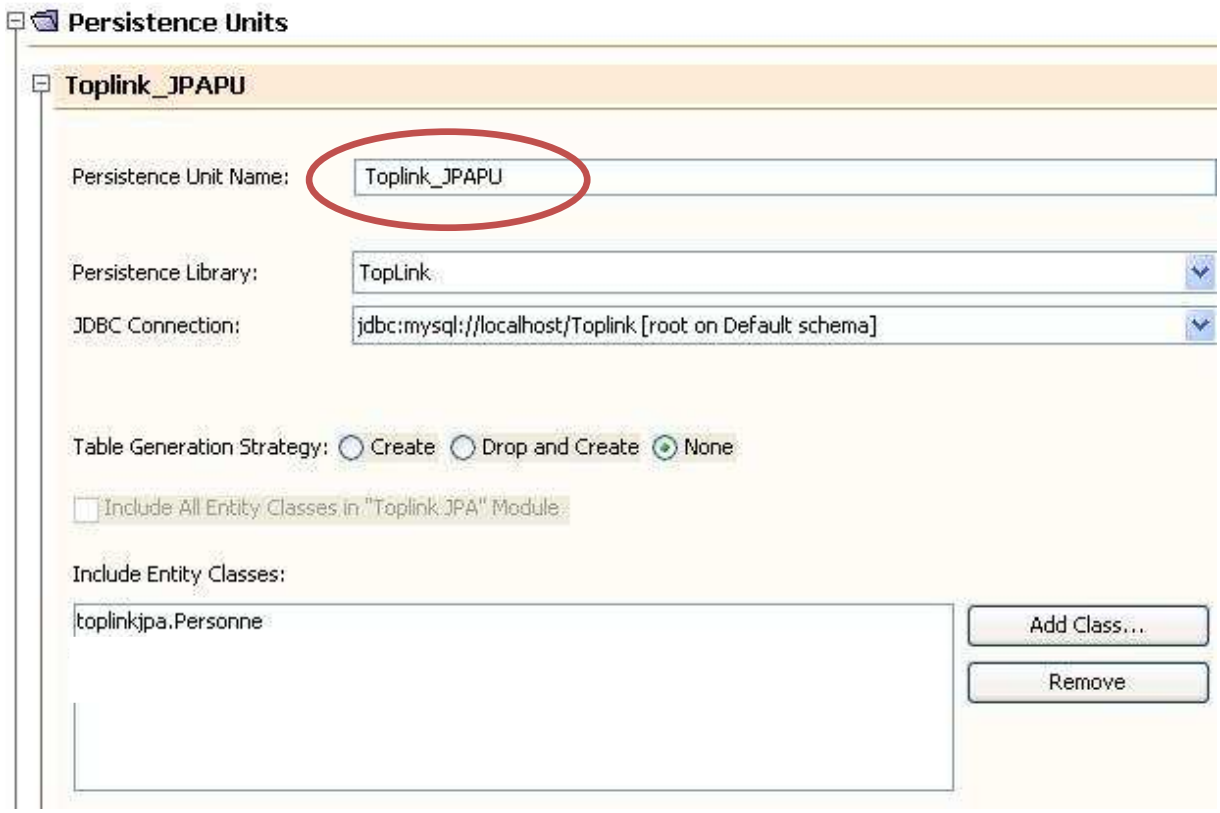
Avant de commencer les opérations sur les tables, il introduit ce bout de code au début de programme :

```
emf = Persistence.createEntityManagerFactory("Toplink_JPAPU");
 em = emf.createEntityManager();
trans = em.getTransaction();
```

Et celui-ci juste avant la fermeture de l'application :

```
em.close();
emf.close();
```

Ici le String « Toplink JPAPU » correspond au nom de la Persistence Unit dans le fichier persistence.xml



2. Ajouter un enregistrement

Avant chaque opération, on débute l' « Entity Transaction » trans avec trans.begin(); qui récupère la transaction de l' « Entity Manager » em. Lorsqu'une opération est prête à être soumise à la base on exécute la transaction avec trans.commit();

Voici le bout de code permettant de créer un nouvel enregistrement dans la base de données :

```
p = new Personne(nom, prenom);

trans.begin();
em.persist(p);
trans.commit();
```

Ici les variables nom et prenom sont égales au champs de deux JTextField. Personne est le nom de l'objet relié à la base de données.

3. Supprimer un enregistrement

Pour pouvoir supprimer un enregistrement, il faut tout d'abord récupérer un enregistrement de la table sous la forme d'un objet dans java :

```
p = em.find(Personne.class, id);
```

Ici « p » est un objet Personne et id est l'identifiant d'une personne.

On peut ensuite exécuter la suppression :

```
trans.begin();
em.remove(p);
trans.commit();
```

4. Modifier un enregistrement

On récupère à nouveau une Personne comme dans la partie « Supprimer un enregistrement » :

```
p = em.find(Personne.class, id);
```

On execute ensuite :

```
trans.begin();
p.setNom(jNomUpdate.getText());
p.setPrenom(jPrenomUpdate.getText());
trans.commit();
```

Les méthodes SetNom et SetPrenom servent à enregistrer respectivement un nom et un prénom pour une personne. Ici on leur passe en paramètre le Text de deux JTextField.

5. Gestion des associations

Ici, on décrit l'association entre les tables Personne et une nouvelle table Localité. Cette portion de code est à insérer dans la classe Personne car c'est la table Personne qui reçoit la clé étrangère de la table Localité.

```
@OneToOne(fetch = FetchType.EAGER, cascade = {CascadeType.PERSIST, CascadeType.REMOVE})
@JoinColumn(name = " localite_fk", nullable = false)
 private Localite address;
```

6. Avantages et inconvénients

Toplink JPA est un moyen très simple de travailler avec des bases de données en java, puisque c'est presque comme si l'on travaillait directement sur un objet. On n'a ainsi pas besoin de taper des requêtes sql et écrire tout le code nécessaire à leur exécution.

La programmation et la gestion de base de données dans une application se trouvent ainsi facilitées.

7. Sources

http://www.devx.com/Java/Article/33650/0