TP Formation JPA Hibernate

Pour Formations

Date 14/03/2021

Objet TP Formation JPA Hibernate

I) Préambule sur les travaux pratiques

Faire les différentes TP les uns à la suite des autres. N'hésitez pas à contacter le formateur en cas de besoin. Le but de ces TP est d'appliquer en pratique toutes les notions qui ont été abordé lors de la formation.

II) TP DAO Personne et Adresse (oneToOne)

Nous allons dans ce TP séparer les informations personnelles d'une personne avec son adresse.
On pourra considérer qu'une personne ne peut avoir une seule et unique adresse.
Nous allons utiliser le script SQL sur http://elhadji-gaye.fr/Formations/Elements-Dispo/DataBases/base_personnes_one_to_one.sql.

Créer le projet **Maven maven-dao-personne-one-to-one-jpa-hibernate**.

1) Créer la classe **com.cours.entities.Adresse** en y ajoutant les informations liées à l'adresse de la classe **com.cours.entities.Personne**.

2) Créer l'interface **IPersonneDao** avec les méthodes :

```
public interface IPersonneDao {

  public List<Personne> findAll();

  public Personne findById(Integer id);

  public Personne authenticate(String prenom, String nom);

  public Personne create(Personne person);

  public Personne update(Personne person);

  public Boolean delete(Personne person);

 // Créer un ensemble de personnes
 public List<Personne> createBatchPersonnes(List<Personne> persons);

// Supprimer un ensemble de personnes
 public Boolean deleteBatchPersonnes(List<Personne> persons);

  // chercher une Personne avec son prenom
  public List<Personne> findByPrenom(String prenom);
  public List<Personne> findByPrenomWithAdresses(String prenom);

  // chercher une Personne avec son nom
  public List<Personne> findByNom(String nom);
  public List<Personne> findByNomWithAdresses(String nom);
  // chercher des Personnes avec son prenom et son nom
  public List<Personne> findByPrenomNom(String prenom, String nom);
  public List<Personne> findByPrenomNomWithAdresses(String prenom, String nom);

  // chercher des personnes avec leur codePostal
  public List<Personne> findByCodePostal(String codePostal);
  public List<Personne> findByCodePostalWithAdresses(String codePostal);
```

```
    // chercher des personnes avec leur ville
    public List<Personne> findByVille(String ville);
    public List<Personne> findByVilleWithAdresses(String ville);


    // chercher les Personnes obese
    public List<Personne> findPersonnesObese();
    public List<Personne> findPersonnesObeseWithAdresses();

    // chercher les Personne obese
    public List<Personne> findPersonnesMaigre();
    public List<Personne> findPersonnesMaigreWithAdresses();
}
```

3) Créer l'interface **IAdresseDao** avec les méthodes **:**

```
public interface IAdresseDao {

    public List<Adresse> findAll();

    public Adresse findById(Integer id);

    // chercher des adresses avec leur codePostal
    public List<Adresse> findByRue(String rue);

    // chercher des adresses avec leur codePostal
    public List<Adresse> findByCodePostal(String codePostal);

    // chercher des adresses avec leur ville
    public List<Adresse> findByVille(String ville);

    public Adresse create(Adresse adr);

    public Adresse update(Adresse adr);

    public Boolean delete(Adresse adr);
}
```

4) Créer et compléter la classe **PersonneDao** qui implémentera l'interface **IPersonneDao**.

5) Créer et compléter la classe **AdresseDao** qui implémentera l'interface **IAdresseDao**.

6) Créer les classes de test unitaire **JUnitDao**, **JUnitPersonneDao** et **JUnitAdresseDao**. **JUnitDao** lancera automatiquement les tests unitaires **JUnitPersonneDao** et **JUnitAdresseDao**.

III) TP DAO Personne et Adresse (oneToOne avec table d'association)

Nous allons dans ce TP séparer les informations personnelles d'une personne avec son adresse.
On pourra considérer qu'une personne ne peut avoir une seule et unique adresse.
Nous allons utiliser le script SQL sur http://elhadji-gaye.fr/Formations/Elements-Dispo/DataBases/base_personnes_one_to_one_table_assoc.sql.

Créer le projet **Maven maven-dao-personne-one-to-one-table-assoc-jpa-hibernate**.

1) Créer la classe **com.cours.entities.Adresse** en y ajoutant les informations liées à l'adresse de la classe **com.cours.entities.Personne**.

2) Créer l'interface **IPersonneDao** avec les méthodes :

```
public interface IPersonneDao {

  public List<Personne> findAll();

  public Personne findById(Integer id);

  public Personne authenticate(String prenom, String nom);

  public Personne create(Personne person);

  public Personne update(Personne person);

  public Boolean delete(Personne person);

    // Créer un ensemble de personnes
  public List<Personne> createBatchPersonnes(List<Personne> persons);

// Supprimer un ensemble de personnes
  public Boolean deleteBatchPersonnes(List<Personne> persons);

  // chercher une Personne avec son prenom
  public List<Personne> findByPrenom(String prenom);
  public List<Personne> findByPrenomWithAdresses(String prenom);

  // chercher une Personne avec son nom
  public List<Personne> findByNom(String nom);
  public List<Personne> findByNomWithAdresses(String nom);
  // chercher des Personnes avec son prenom et son nom
  public List<Personne> findByPrenomNom(String prenom, String nom);
  public List<Personne> findByPrenomNomWithAdresses(String prenom, String nom);

  // chercher des personnes avec leur codePostal
  public List<Personne> findByCodePostal(String codePostal);
  public List<Personne> findByCodePostalWithAdresses(String codePostal);
```

```
    // chercher des personnes avec leur ville
    public List<Personne> findByVille(String ville);
    public List<Personne> findByVilleWithAdresses(String ville);


    // chercher les Personnes obese
    public List<Personne> findPersonnesObese();
    public List<Personne> findPersonnesObeseWithAdresses();

    // chercher les Personne obese
    public List<Personne> findPersonnesMaigre();
    public List<Personne> findPersonnesMaigreWithAdresses();
}
```

3) Créer l'interface **IAdresseDao** avec les méthodes **:**

```
public interface IAdresseDao {

    public List<Adresse> findAll();

    public Adresse findById(Integer id);

    // chercher des adresses avec leur codePostal
    public List<Adresse> findByRue(String rue);

    // chercher des adresses avec leur codePostal
    public List<Adresse> findByCodePostal(String codePostal);

    // chercher des adresses avec leur ville
    public List<Adresse> findByVille(String ville);

    public Adresse create(Adresse adr);

    public Adresse update(Adresse adr);

    public Boolean delete(Adresse adr);
}
```

4) Créer et compléter la classe **PersonneDao** qui implémentera l'interface **IPersonneDao**.

5) Créer et compléter la classe **AdresseDao** qui implémentera l'interface **IAdresseDao**.

6) Créer les classes de test unitaire **JUnitDao**, **JUnitPersonneDao** et **JUnitAdresseDao**. **JUnitDao** lancera automatiquement les tests unitaires **JUnitPersonneDao** et **JUnitAdresseDao**.

IV) TP DAO Personne et Adresse (oneToMany)

Nous allons dans ce TP séparer les informations personnelles d'une personne avec son adresse.
On pourra considérer qu'une personne peut plusieurs adresses.
Nous allons utiliser le script SQL sur http://elhadji-gaye.fr/Formations/Elements-Dispo/DataBases/base_personnes_one_to_many.sql.

Créer le projet **Maven maven-dao-personne-one-to-many-jpa-hibernate**.

1) Créer la classe **com.cours.entities.Adresse** en y ajoutant les informations liées à l'adresse de la classe **com.cours.entities.Personne**.

2) Créer l'interface **IPersonneDao** avec les méthodes :

```
public interface IPersonneDao {

  public List<Personne> findAll();

  public Personne findById(Integer id);

  public Personne authenticate(String prenom, String nom);

  public Personne create(Personne person);

  public Personne update(Personne person);

  public Boolean delete(Personne person);

   // Créer un ensemble de personnes
  public List<Personne> createBatchPersonnes(List<Personne> persons);

 // Supprimer un ensemble de personnes
  public Boolean deleteBatchPersonnes(List<Personne> persons);

  // chercher une Personne avec son prenom
  public List<Personne> findByPrenom(String prenom);
  public List<Personne> findByPrenomWithAdresses(String prenom);

  // chercher une Personne avec son nom
  public List<Personne> findByNom(String nom);
  public List<Personne> findByNomWithAdresses(String nom);
  // chercher des Personnes avec son prenom et son nom
  public List<Personne> findByPrenomNom(String prenom, String nom);
  public List<Personne> findByPrenomNomWithAdresses(String prenom, String nom);

  // chercher des personnes avec leur codePostal
  public List<Personne> findByCodePostal(String codePostal);
```

```
    public List<Personne> findByCodePostalWithAdresses(String codePostal);

    // chercher des personnes avec leur ville
    public List<Personne> findByVille(String ville);
    public List<Personne> findByVilleWithAdresses(String ville);


    // chercher les Personnes obese
    public List<Personne> findPersonnesObese();
    public List<Personne> findPersonnesObeseWithAdresses();

    // chercher les Personne obese
    public List<Personne> findPersonnesMaigre();
    public List<Personne> findPersonnesMaigreWithAdresses();
}
```

3) Créer l'interface **IAdresseDao** avec les méthodes **:**

```
public interface IAdresseDao {

    public List<Adresse> findAll();

    public Adresse findById(Integer id);

    // chercher des adresses avec leur codePostal
    public List<Adresse> findByRue(String rue);

    // chercher des adresses avec leur codePostal
    public List<Adresse> findByCodePostal(String codePostal);

    // chercher des adresses avec leur ville
    public List<Adresse> findByVille(String ville);

    public Adresse create(Adresse adr);

    public Adresse update(Adresse adr);

    public Boolean delete(Adresse adr);
}
```

4) Créer et compléter la classe **PersonneDao** qui implémentera l'interface **IPersonneDao**.

5) Créer et compléter la classe **AdresseDao** qui implémentera l'interface **IAdresseDao**.

6) Créer les classes de test unitaire **JUnitDao**, **JUnitPersonneDao** et **JUnitAdresseDao**. **JUnitDao** lancera automatiquement les tests unitaires **JUnitPersonneDao** et **JUnitAdresseDao**.

V) TP DAO Personne et Adresse (oneToMany avec table d'association)

Nous allons dans ce TP séparer les informations personnelles d'une personne avec son adresse.
On pourra considérer qu'une personne peut avoir plusieurs adresses.
Nous allons utiliser le script SQL sur http://elhadji-gaye.fr/Formations/Elements-Dispo/DataBases/base_personnes_one_to_many_table_assoc.sql.

Créer le projet **Maven maven-dao-personne-one-to-many-table-assoc-jpa-hibernate**.

1) Créer la classe **com.cours.entities.Adresse** en y ajoutant les informations liées à l'adresse de la classe **com.cours.entities.Personne**.

2) Créer l'interface **IPersonneDao** avec les méthodes :

```
public interface IPersonneDao {

  public List<Personne> findAll();

  public Personne findById(Integer id);

  public Personne authenticate(String prenom, String nom);

  public Personne create(Personne person);

  public Personne update(Personne person);

  public Boolean delete(Personne person);

    // Créer un ensemble de personnes
 public List<Personne> createBatchPersonnes(List<Personne> persons);

// Supprimer un ensemble de personnes
 public Boolean deleteBatchPersonnes(List<Personne> persons);

  // chercher une Personne avec son prenom
  public List<Personne> findByPrenom(String prenom);
  public List<Personne> findByPrenomWithAdresses(String prenom);

  // chercher une Personne avec son nom
  public List<Personne> findByNom(String nom);
  public List<Personne> findByNomWithAdresses(String nom);
  // chercher des Personnes avec son prenom et son nom
  public List<Personne> findByPrenomNom(String prenom, String nom);
  public List<Personne> findByPrenomNomWithAdresses(String prenom, String nom);

  // chercher des personnes avec leur codePostal
  public List<Personne> findByCodePostal(String codePostal);
  public List<Personne> findByCodePostalWithAdresses(String codePostal);
```

```
    // chercher des personnes avec leur ville
    public List<Personne> findByVille(String ville);
    public List<Personne> findByVilleWithAdresses(String ville);


    // chercher les Personnes obese
    public List<Personne> findPersonnesObese();
    public List<Personne> findPersonnesObeseWithAdresses();

    // chercher les Personne obese
    public List<Personne> findPersonnesMaigre();
    public List<Personne> findPersonnesMaigreWithAdresses();
}
```

3) Créer l'interface **IAdresseDao** avec les méthodes **:**

```
public interface IAdresseDao {

    public List<Adresse> findAll();

    public Adresse findById(Integer id);

    // chercher des adresses avec leur codePostal
    public List<Adresse> findByRue(String rue);

    // chercher des adresses avec leur codePostal
    public List<Adresse> findByCodePostal(String codePostal);

    // chercher des adresses avec leur ville
    public List<Adresse> findByVille(String ville);

    public Adresse create(Adresse adr);

    public Adresse update(Adresse adr);

    public Boolean delete(Adresse adr);
}
```

4) Créer et compléter la classe **PersonneDao** qui implémentera l'interface **IPersonneDao**.

5) Créer et compléter la classe **AdresseDao** qui implémentera l'interface **IAdresseDao**.

6) Créer les classes de test unitaire **JUnitDao**, **JUnitPersonneDao** et **JUnitAdresseDao**. **JUnitDao** lancera automatiquement les tests unitaires **JUnitPersonneDao** et **JUnitAdresseDao**.

VI) TP DAO Personne et Adresse (ManyToMany)

Nous allons dans ce TP séparer les informations personnelles d'une personne avec son adresse.
On pourra considérer qu'une personne peut avoir une seule ou plusieurs adresses.
Nous allons utiliser le script SQL sur http://elhadji-gaye.fr/Formations/Elements-Dispo/DataBases/base_personnes_many_to_many.sql.

Créer le projet **Maven maven-dao-personne-many-to-many-jpa-hibernate**.

1) Créer la classe **com.cours.entities.Adresse** en y ajoutant les informations liées à l'adresse de la classe **com.cours.entities.Personne**.

2) Créer l'interface **IPersonneDao** avec les méthodes :

```
public interface IPersonneDao {

   public List<Personne> findAll();

   public Personne findById(Integer id);

   public Personne authenticate(String prenom, String nom);

   public Personne create(Personne person);

   public Personne update(Personne person);

   public Boolean delete(Personne person);

      // Créer un ensemble de personnes
  public List<Personne> createBatchPersonnes(List<Personne> persons);

 // Supprimer un ensemble de personnes
  public Boolean deleteBatchPersonnes(List<Personne> persons);

   // chercher une Personne avec son prenom
   public List<Personne> findByPrenom(String prenom);
   public List<Personne> findByPrenomWithAdresses(String prenom);

   // chercher une Personne avec son nom
   public List<Personne> findByNom(String nom);
   public List<Personne> findByNomWithAdresses(String nom);
   // chercher des Personnes avec son prenom et son nom
   public List<Personne> findByPrenomNom(String prenom, String nom);
   public List<Personne> findByPrenomNomWithAdresses(String prenom, String nom);

   // chercher des personnes avec leur codePostal
   public List<Personne> findByCodePostal(String codePostal);
```

```
    public List<Personne> findByCodePostalWithAdresses(String codePostal);

    // chercher des personnes avec leur ville
    public List<Personne> findByVille(String ville);
    public List<Personne> findByVilleWithAdresses(String ville);


    // chercher les Personnes obese
    public List<Personne> findPersonnesObese();
    public List<Personne> findPersonnesObeseWithAdresses();

    // chercher les Personne obese
    public List<Personne> findPersonnesMaigre();
    public List<Personne> findPersonnesMaigreWithAdresses();
}
```

3) Créer l'interface **IAdresseDao** avec les méthodes **:**

```
public interface IAdresseDao {

    public List<Adresse> findAll();

    public Adresse findById(Integer id);

    // chercher des adresses avec leur codePostal
    public List<Adresse> findByRue(String rue);

    // chercher des adresses avec leur codePostal
    public List<Adresse> findByCodePostal(String codePostal);

    // chercher des adresses avec leur ville
    public List<Adresse> findByVille(String ville);

    public Adresse create(Adresse adr);

    public Adresse update(Adresse adr);

    public Boolean delete(Adresse adr);
}
```

4) Créer et compléter la classe **PersonneDao** qui implémentera l'interface **IPersonneDao**.

5) Créer et compléter la classe **AdresseDao** qui implémentera l'interface **IAdresseDao**.

6) Créer les classes de test unitaire **JUnitDao**, **JUnitPersonneDao** et **JUnitAdresseDao**. **JUnitDao** lancera automatiquement les tests unitaires **JUnitPersonneDao** et **JUnitAdresseDao**.