2.12 Travaux pratiques : Intégration Hibernate avec Spring

Exercice 1

On souhaite développer une application de gestion de stock pour un magasin de vente de produits informatiques.

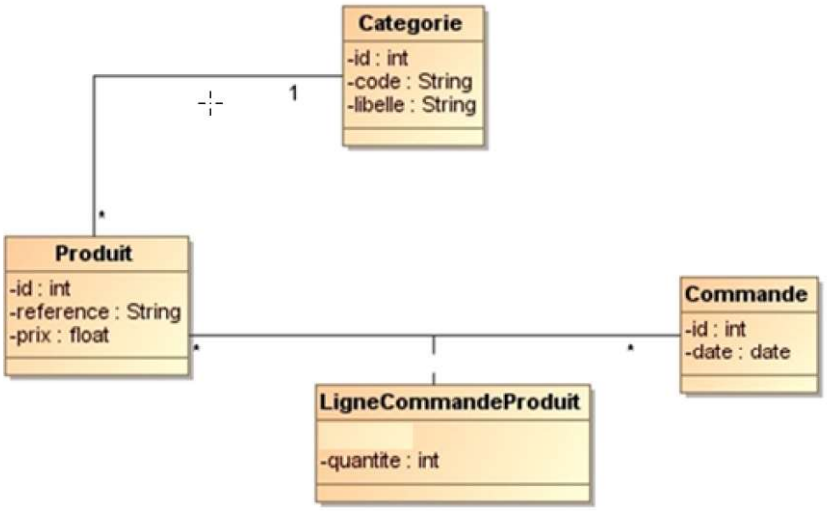

Figure 2.5: Diagramme de classe.

Couche persistance :

1. Développer les classes entités dans le package `ma.projet.classes`.
2. Créer le fichier de configuration `application.properties`.
3. Créer la classe `HibernateUtil` dans le package `ma.projet.util`.

Couche service :

1. Créer l'interface générique `IDao` dans le package `ma.projet.dao`.

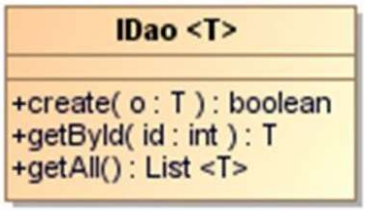

Figure 2.6: Diagramme de classe.

2. Créer les classes services : `ProduitService`, `CategorieService`, `CommandeService`, et `LigneCommandeService`, qui implémentent l'interface `IDao` dans le package `ma.projet.service`.
3. Créer une méthode permettant d'afficher la liste des produits par catégorie dans la classe `ProduitService`.
4. Créer une méthode permettant d'afficher la liste des produits commandés entre deux dates.
5. Créer une méthode permettant d'afficher les produits commandés dans une commande donnée.
   Exemple d'affichage :

```
Commande : 4     Date : 14 Mars 2013
Liste des produits :
Référence   Prix    Quantité
ES12        120 DH  7
ZR85        100 DH  14
EE85        200 DH  5
```

6. Créer une méthode permettant d'afficher la liste des produits dont le prix est supérieur à 100 DH dans la classe `ProduitService` en utilisant une requête nommée.
7. Créer des programmes pour tester les points ci-dessus.

Exercice 2

Un bureau d'études souhaite développer une application de gestion de projets pour calculer le temps passé dans chaque projet et l'imputer à son coût global.

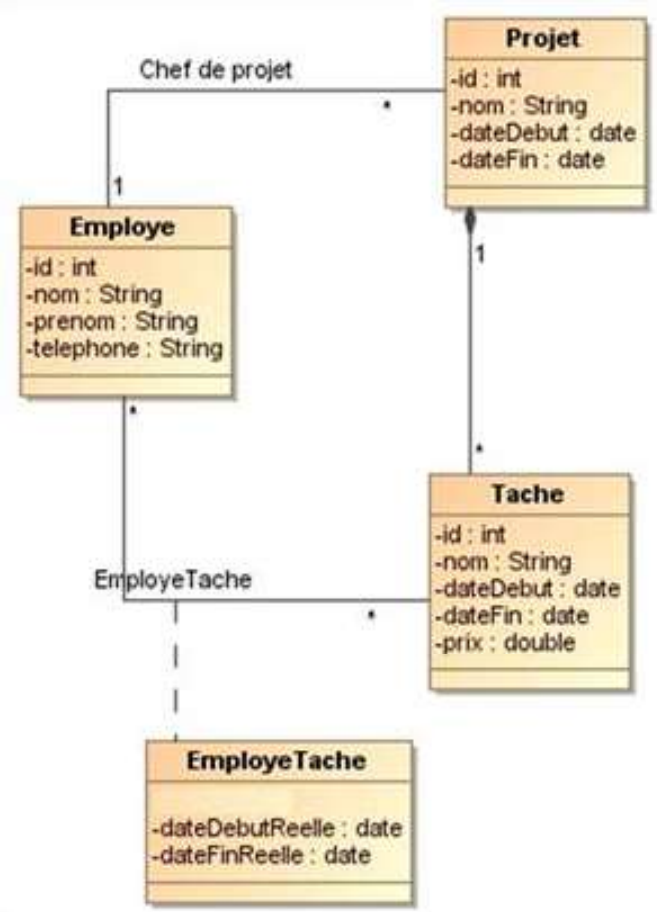

Figure 2.7: Diagramme de classe.

Couche persistance :

1. Développer les classes entités dans le package `ma.projet.classes`.
2. Créer le fichier de configuration `application.properties` .
3. Créer la classe `HibernateUtil` dans le package `ma.projet.util`.

Couche service :

1. Créer l'interface générique `IDao` dans le package `ma.projet.dao`.
2. Créer les classes services : `ProjetService`, `TacheService`, `EmployeService`, et `EmployeTacheService` qui implémentent l'interface `IDao`.
3. Créer une méthode permettant d'afficher la liste des tâches réalisées par un employé dans la classe `EmployeService`.
4. Créer une méthode permettant d'afficher la liste des projets gérés par un employé dans la classe `EmployeService`.
5. Créer une méthode permettant d'afficher la liste des tâches planifiées pour un projet dans la classe `ProjetService`.

6. Créer une méthode permettant d'afficher la liste des tâches réalisées dans un projet avec les détails de début et fin réels.
   Exemple d'affichage :

```
Projet : 4      Nom : Gestion de stock     Date début : 14 Janvier 2013
Liste des tâches:
Num Nom            Date Début Réelle   Date Fin Réelle
12  Analyse        10/02/2013          20/02/2013
13  Conception     10/03/2013          15/03/2013
14  Développement  10/04/2013          25/04/2013
```

7. Créer une méthode permettant d'afficher la liste des tâches dont le prix est supérieur à 1000 DH dans la classe `TacheService` en utilisant une requête nommée.
8. Créer une méthode permettant d'afficher la liste des tâches réalisées entre deux dates dans la classe `TacheService`.
9. Créer des programmes pour tester les points ci-dessus.

Exercice 4

Cet exercice consiste à développer une application de gestion de l'état civil des citoyens d'une province.

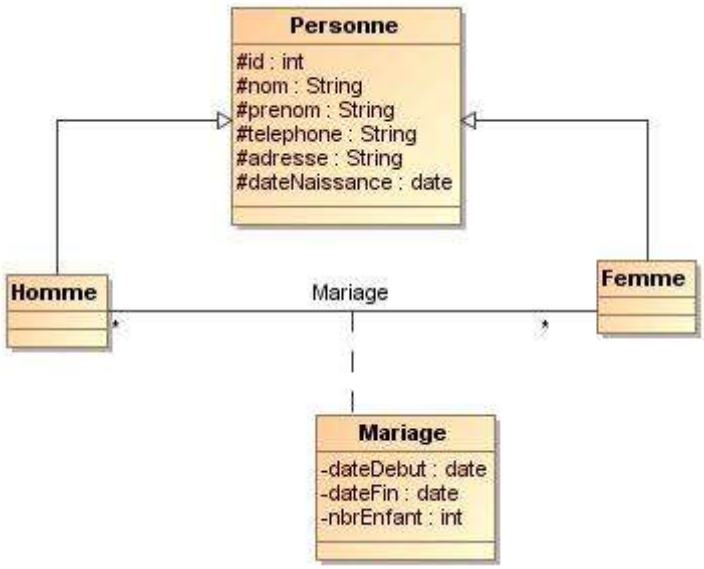

Figure 2.8: Diagramme de classe.

Couche persistance :

1. Développer les classes entités dans le package `ma.projet.beans` en utilisant toutes les annotations vues en classe.
2. Créer le fichier de configuration `application.properties` .
3. Créer la classe `HibernateUtil` dans le package `ma.projet.util`.
4. Générer la base de données sous MySQL.

Couche service :

1. Créer l'interface générique `IDao` dans le package `ma.projet.dao`.
2. Créer les classes services : `HommeService`, `FemmeService`, et `MariageService`, qui implémentent l'interface `IDao`.
3. Créer une méthode permettant d'afficher les épouses d'un homme passé en paramètre entre deux dates dans la classe `HommeService`.

4. Créer une requête native nommée pour renvoyer le nombre d'enfants d'une femme donnée entre deux dates.
5. Créer une méthode dans la classe `FemmeService` pour appeler la requête de la question précédente.
6. Créer une requête nommée pour renvoyer les femmes mariées deux fois ou plus.
7. Créer une méthode dans la classe `FemmeService` pour appeler la requête de la question précédente.
8. Créer une méthode pour renvoyer le nombre d'hommes mariés à quatre femmes entre deux dates en utilisant l'API `CRITERIA`.
9. Créer une méthode pour renvoyer les mariages d'un homme donné en paramètre avec les détails des épouses, des dates et du nombre d'enfants.
   Exemple d'affichage :

```
Nom : SAFI SAID
Mariages En Cours :
1. Femme : SALIMA RAMI   Date Début : 03/09/1990    Nbr Enfants : 4
2. Femme : AMAL ALI      Date Début : 03/09/1995    Nbr Enfants : 2
3. Femme : WAFA ALAOUI   Date Début : 04/11/2000    Nbr Enfants : 3

Mariages échoués :
1. Femme : KARIMA ALAMI  Date Début : 03/09/1989
           Date Fin : 03/09/1990    Nbr Enfants : 0
```

10. Créer un programme de test avec divers scénarios :
    - Créer 10 femmes et 5 hommes,
    - Afficher la liste des femmes,
    - Afficher la femme la plus âgée,
    - Afficher les épouses d'un homme passé en paramètre,
    - Afficher le nombre d'enfants d'une femme passée en paramètre entre deux dates,
    - Afficher la liste des femmes mariées deux fois ou plus,
    - Afficher les hommes mariés à quatre femmes entre deux dates,
    - Afficher les mariages d'un homme passé en paramètre.