Thomas Duchatelle
31 janvier 2013

TP – Initialisation aux Frameworks

– Spring –

Table des matières

I - Sujet

1 Présentation du contexte

Pour ce TP, nous reprendrons le sujet du précédent : *la bibliothèque*. Il se décompose en 2 partie :
- Séparation des préoccupations et injection de dépendances
- Coupage de *Spring* à *Hibernate* pour réaliser la couche DAO.

2 Cas étudiés

Nous allons étudier 2 cas fonctionnels :
- Création d'une nouvelle référence et ajout d'exemplaires
- Emprunt d'un exemplaire par un client

a. Nouvelle référence

L'ajout d'une nouvelle référence et d'exemplaires liés suis la procédure ci-après :

1. L'administrateur renseigne l'identifiant de la libraire et les données sur le livre qu'il souhaite ajouter (code ISBN du livre, titre, description, nombre de pages), ainsi que le nombre d'exemplaires.
2. Recherche de la librairie par son identifiant, elle doit exister
3. Vérification que la référence n'existe pas déjà (par son code ISBN), sinon une exception sera levée.
4. Génération d'un code pour chacun des exemplaires
5. Sauvegarde de la référence `Book`, et des exemplaires, en BDD.

Le point 1 ne fait pas partie du TP. Il sera simuler dans la fonction `main`.

b. Emprunt d'un exemplaire

Pour emprunter un exemplaire, le processus est :

1. L'utilisateur indique son code client et le code de l'exemplaire qu'il souhaite emprunter
2. le client et l'exemplaire son rechercher en base, ils doivent exister.
3. vérifications : l'exemplaire n'est pas déjà emprunter, le client ne doit pas avoir plus de 3 emprunts en cours
4. ajouter l'exemplaire à la liste des emprunt en cours du client

Le nombre d'exemplaire autorisés doit être configuré dans un fichier *properties*.

3 Architecture

L'architecture logicielle suit le principe de la *Séparation des Préoccupations*. Elle est présentée figure 3 page suivante.

Les briques logicielles sont :

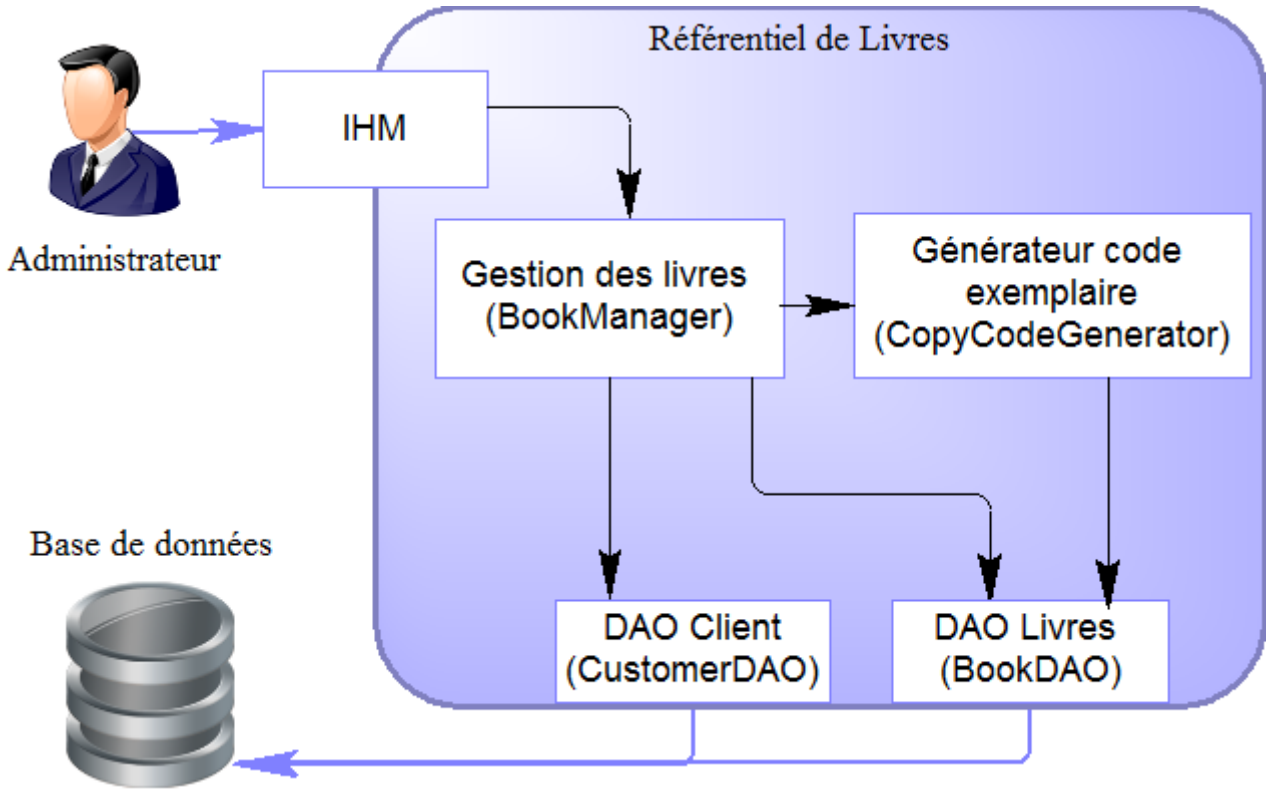

FIGURE 1 – Architecture des objets métiers

BookManager contient les règles métiers telles qu'elles sont décrites dans le cas présentés.

CopyCodeGenerator Générateur des codes exemplaires

BookDAO couche d'accès aux données `Book` de la BDD.

CustomerDAO couche d'accès aux données `Customer` de la BDD.

II - PREMIÈRE PARTIE : INJECTION DE DÉPENDANCES

1 Composants déjà présents

Afin de gagner du temps, certains composant ont déjà été créés. Si vous souhaitez continuer du TP sur Hibernate, copier

a. Interfaces

Les interfaces de chaque brique ont été crées. Comme dans la convention de nommage utilisée dans le Référentiel Coordonnées, elle commence par un "*I*" ("i" majuscule).

b. Contexte Spring

La configuration principale de Spring est dans le fichier : `src/main/resources/spring/books-context.xml`.

Dans la fonction `main`, le contexte est déjà instancié et est prêt à l'utilisation.

c. Couche DAO bouchonnée

Une fausse implémentation de la couche DAO est présente dans le package `net.yvesrocher.training.framewo`
Celle-ci utilise des `HashMap` en interne pour simuler une véritable base de données.

Pour activer cette couche bouchonnée, il faut ajouter le fichier de configuration *Spring* : `src/main/resources/s`

2 Déclaration des beans et injection

a. Premier bean

Dans cette partie :

1. Créez une implémentation à l'interface `IBookManager`
2. Déclarez la comme un *Bean*[1]
3. Récupérez une instance à partir du contexte Spring

Vous pouvez récupérer plusieurs `IBookManager` et vérifier que ce soit la même instance[2] qui est renvoyée.
Il est possible de modifier le scope en `singleton`, puis en `prototype` pour constater des différences de comportement.

b. Première dépendance

Le *BookManager* ne peut pas travailler seul. Il a besoin du générateur de code d'exemplaire, et de la couche d'accès aux données des livres et des clients.

Dans un premier temps, nous ne nous intéresseront qu'au générateur :

1. Créez une implémentation de l'interface `ICopyCodeGenerator`
2. Déclarez la comme un *Bean*
3. Injectez un ICopyCodeGenerator dans le *BookManager*

3 Implémenter les beans

a. CopyCodeGenerator

Implémentez les méthodes du bean `CopyCodeGeneratorImpl`. Il doit utiliser le `BookDAO` pour trouver un identifiant disponible.

1. Rappel : un *bean* est le nom d'une "brique applicative" pour Spring.
2. Réalisation d'un classe (exemple : voiture). A ne pas confondre avec une classe qui correspondrait aux plans de la voiture.

b. BookManager

Implémentez la méthode `BookManagerImpl.createNewBook`. Il doit utiliser le `CopyCodeGenerator` et le `BookDAO`.

c. Tester

Des méthodes statiques sont présentes sur les DAO afin de lister le contenu de la base de données.

III - Seconde Partie : Couche d'Accès aux Données

1 Configuration du contexte Spring

Avant tout, il faut configurer Spring pour qu'il puisse injecter la `SessionFactory`. Un fichier de configuration est prêt : `src/main/resources/spring/books-dao-hibernate.xml`. Ajoutez le à la liste lors de la création du contexte Spring.

Les nouveaux beans utilisant hibernate risquent de rentrer en conflit avec les bouchons. Retirez le fichier `src/main/resources/spring/books-dao-mock.xml` de la liste des fichiers de configuration.

2 Ré-implémenter la couche DAO

Ré-implémenter la couche DAO en utilisant Hibernate.

a. Injection de la session factory

La session factory est maintenant injectable :

```
1 @Named
2 public class EmployeeDAOImpl implements IEmployeeDAO {
3
4   @Inject
5   private SessionFactory factory;
6 }
```

b. Ouverture de la session

La session est ouverte automatiquement lors de l'ouverture d'une transaction. On respecte le pattern : *1 transaction = 1 session.*