Thomas Duchatelle
11 février 2013

TP – Spring

– Initialisation aux Frameworks –

Table des matières

I - Sujet

1 Objectifs

Dans ce TP, nous reprendrons *la bibliothèque* du précédent TP afin de lui rajouter une couche métier (business). Il se décomposera en 2 parties :

- Séparation des préoccupations et injection de dépendances
- Coupage de *Spring* à *Hibernate* pour réaliser la couche DAO.

2 Modèle de données

Le modèle de données, présenté figure 2 de la présente page, est le même que dans le TP sur Hibernate (section : *pour aller plus loin*). Lui sont ajoutés l'aspect d'exemplaire (`BookCopy`), et d'emprunt (`Customer`).

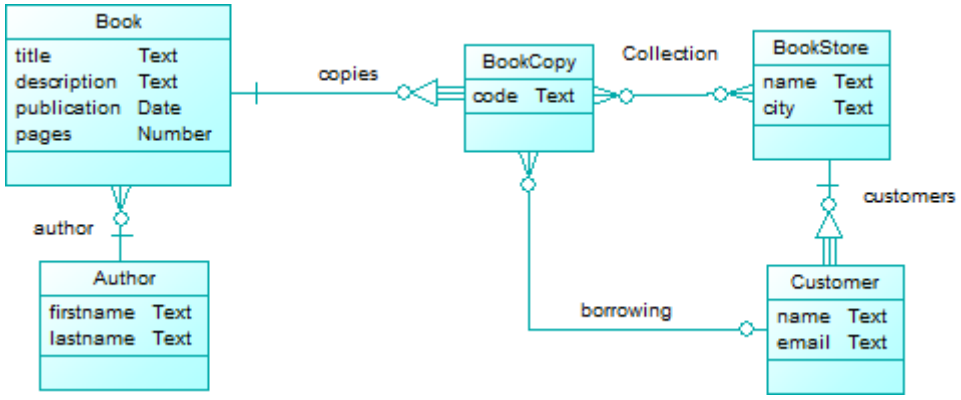

Figure 1 – Modèle de données

3 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 la référence est mise à jour

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 simulé 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 sont recherchés en base, ils doivent exister.
3. vérifications : l'exemplaire n'est pas déjà emprunté, 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'exemplaires autorisés doit être configuré dans un fichier *properties*.

4 Architecture

L'architecture logicielle suit le principe de la *Séparation des Préoccupations*. Elle est présentée figure 4 page 9.

Les briques logicielles sont :

BookManager contient les règles métiers pour réaliser les 2 cas étudiés

CopyCodeGenerator Générateur des codes exemplaires, utilisant le `BookStoreDAO` pour trouver un nouveau code.

BookDAO couche d'accès aux données `Book` : recherche de livre à partir du code isbn, sauvegarde et suppression

BookStoreDAO couche d'accès aux données `BookStore` : recherche de la librairie par son nom

BookCopyDAO couche d'accès aux données `BookCopy` : génération d'un code unique, recherche d'un copie par son code

CustomerDAO couche d'accès aux données `Customer` : recherche par ID.

5 Composants fournis

Afin de gagner du temps, certains composant ont déjà été créés.

a. 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 comme l'était la `SessinFactory` d'Hibernate lors du premier TP.

b. Mapping relationnel

Le modèle présenté figure 2 page 3 est déjà mappé à la base de données. Il vous faudra peut-être supprimer les tables existant déjà dans votre base locale.

c. Interfaces

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

d. 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`
Elle est pré-activée dans le socle du TP fourni.

Pour remplir la base de données et lister son contenu, la classe `net.yvesrocher.training.frameworks.dao.ut` est présente. En la configurant comme un *singleton*, la base sera automatiquement remplie au démarrage du contexte *Spring*.

Elle expose une méthode `printAll()` qui écrit dans la console l'intégralité de la BDD.

II - Première Partie : Injection de Dépendances

1 Périmètre de la première partie

Dans cette première partie, nous nous intéresserons au principe d'*Inversion de Contrôle* :
- Déclarer un *bean*
- Modifier sa *portée* (ou *scope*)
- Injecter des dépendances

Nous travaillerons donc uniquement sur la *couche business*, et utiliseront la couche DAO bouchonnée.

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 dans la méthode `main`

1. Rappel : un *bean* est le nom d'une "brique applicative" pour Spring.

Vous pouvez récupérer plusieurs `IBookManager` et vérifier que ce soit la même instance[2] qui est renvoyée.

Modifier ensuite 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.

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

Vérifier qu'en appelant une méthode de `BookManager`, ce dernier puisse accéder au `CopyCodeGenerator`.

c. Première configuration

Ajouter un fichier de propriétés : `src/main/resources/config/books.properties` avec la propriété : `borrow.limit`. Récupérez cette valeur (en l'injectant) dans le `BookManager`.

3 Implémenter les beans

a. BddMonitor

Configurez le `BddMonitorImpl` pour être un bean de type singleton. Récupérez le dans la méthode `main` afin d'afficher le contenu de la base de données.

b. CopyCodeGenerator

Implémentez les méthodes du bean `CopyCodeGeneratorImpl` :
– appeler le `CopyBookDAO` pour obtenir un nouveau code

c. BookManager

Implémentez la méthode `BookManagerImpl.insertReference` pour remplir le cahier des charges partie I 3 a..

S'il reste du temps, implémenter la méthode `BookManagerImpl.borrowBook`.

d. Tester

Vérifier le contenu de la base de données avant après pour valider le bon fonctionnement des méthodes.

2. Réalisation d'un classe (exemple : voiture). A ne pas confondre avec une classe qui correspondrait aux plans de la voiture.

4 Pour aller plus loin

Déclarer un autre bean répondant à l'interface `ICopyCodeGenerator`. Que ce passe-t-il à la création du bean `BookManager` ?

Nommer les générateurs et définissez celui qui doit être injecter dans le `BookManager`. Récupérez l'autre dans la méthode `main`.

III - Seconde Partie : Couche d'Accès aux Données

1 Périmètre de la seconde partie

Dans cette seconde partie, nous nous intéressons à la couche d'accès au données. Nous sohaitons la connecter à une base de données SQL à l'aide d'*Hibernate*.

2 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-persistence.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-daomock.xml` de la liste des fichiers de configuration.

3 Ré-implémenter la couche DAO

Implémenter chacune des briques DAO à partir de son interface :
- IBookCopy
- IBookDAO
- IBookStoreDAO
- ICustomerDAO

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.*

Rappel, les transaction sont gérées avec l'annotation `@Transactionnal`

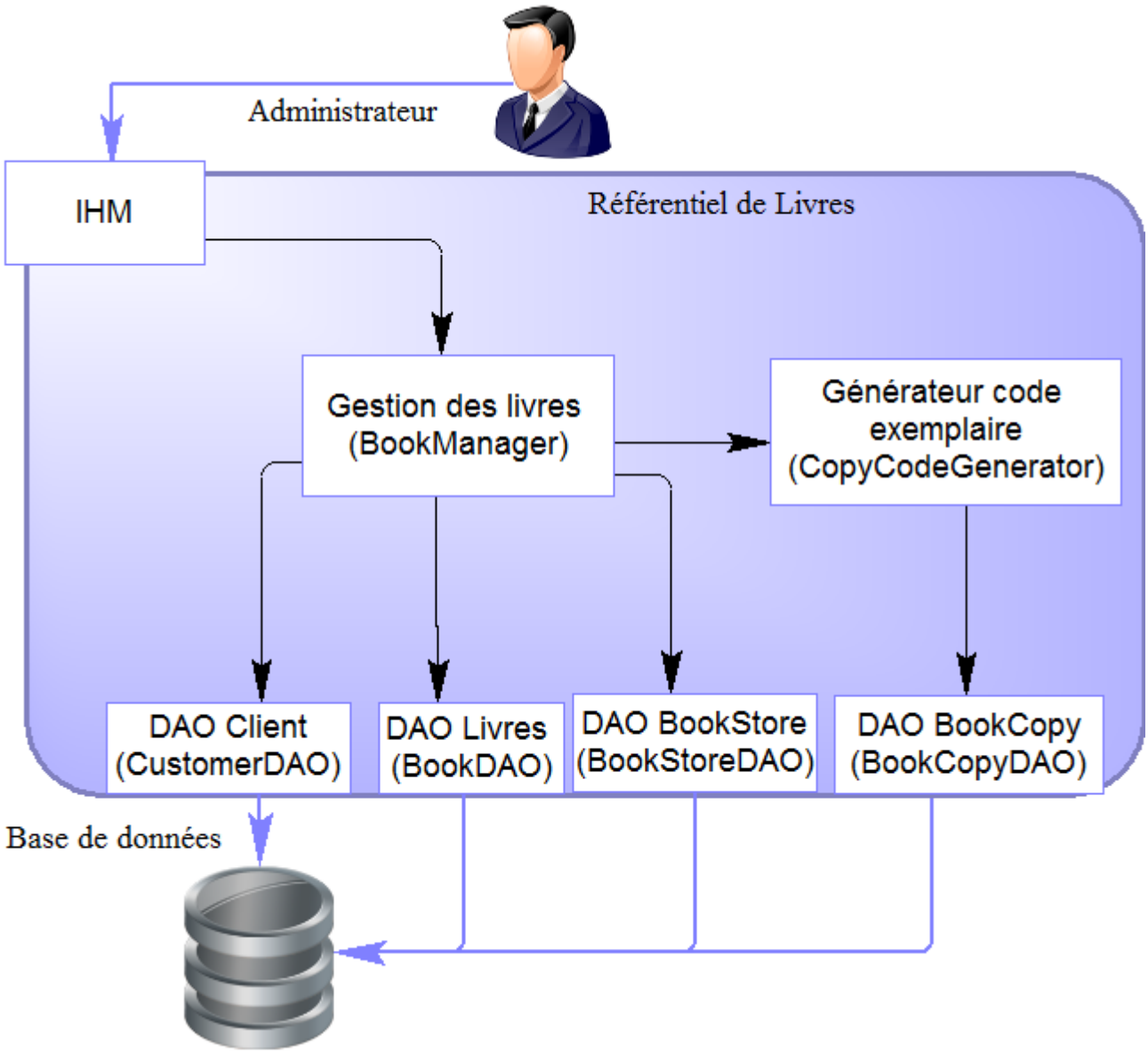

FIGURE 2 – Architecture des objets métiers