

Leçon 3 – Les beans de session

Les « beans de session » (« session beans » en anglais) complètent la couche métier déjà en partie réalisée par les entités.

1 MOTIVATION

Un « bean de session » est la technologie proposée par Java EE pour réaliser la logique métier dans le cadre d'une application distribuée. Par rapport à ce que vous avez codé l'année passée, vous comprenez que les entités ne suffisent pas : certaines logiques sont « inter-entités » et la notion de façade n'est pas encore présente.

Qu'apportent les beans de session ?

Les beans de session sont incorporés dans des conteneurs d'EJB¹ tout comme les Servlets et les JSP sont gérés par des conteneurs web. Par rapport à ce que vous pourriez coder avec de simples classes Java (comme l'année passée) cela apporte de nombreuses facilités

- gestion des transactions et de la sécurité
- gestion du cycle de vie des entités
- possibilité d'une utilisation à distance
- simplification de l'utilisation des entités qui leur sont étroitement liées

2 INTERFACE LOCALE ET DISTANTE

La logique métier que vous avez codé l'année passée n'était accessible que localement (par les servlets dans votre cas). On n'aurait pas pu l'utiliser à partir d'un client lourd². Les beans de session suppriment cette contrainte. Ils exposent deux interfaces : une locale et une distante³. Leur définition utilise une annotation.

Exemple

```
@Local
public interface EmpruntLocal {
    void emprunter( Lecteur lecteur, Livre livre );
    void retourner( Livre livre );
    Livre chercherLivreDisponible( Ouvrage ouvrage );
}

@Remote
public interface EmpruntRemote {
    //idem
}
```

¹EJB (Entity Java Bean) est un terme qui regroupe tous les types de beans liés aux applications distribuées (entité, session, ...)

²Une application graphique tournant sur un client et utilisant la logique métier tournant sur un serveur

³En pratique, elles peuvent être identiques mais cela permet d'offrir des fonctionnalités différentes en fonction du type de client.

3 AVEC OU SANS ÉTAT

Un bean de session peut-être avec ou sans état. Si il est « **avec état** » son état est conservé d'un appel à l'autre au sein d'une même « session »¹ (pensez à la session introduite pour les navigateurs). Au contraire, vous l'aurez compris, un bean « **sans état** » ne conserve rien entre chaque appel.

Le bean sans état est plus rapide et supporte mieux les montées en charge car le conteneur peut les réutiliser sans se soucier de sauver/récupérer l'état. C'est pourquoi il est à privilégier mais le bean avec état est pourtant incontournable lorsque la logique métier manipule des données non persistantes et donc non gérées par des entités.

Exemple

Dans un site de e-vente, le panier du client est une donnée de session non persistante (lors de la confirmation de l'achat, il deviendra persistant sous forme d'une commande). Une telle application implique la création d'un bean de session avec état appelé Panier.

Les annotations simplifient une fois encore leur écriture.

Exemple

```
@Stateless
public class EmpruntBean implements EmpruntLocal, EmpruntRemote {
    @PersistentContext
    private EntityManager em;
    public void retourner( Livre livre ) {
        em.merge(livre);
        // il faudrait vérifier que le livre est bien emprunté
        em.setLivlecteur(null);
        em.setDateemprunt(null);
    }
}

@Statefull
public class Panier implements PanierLocal, PanierRemote {
    @PersistentContext
    private EntityManager em;
    List<Achat> panier; // Achat représente un produit et une qté
    public Panier() {
        panier = new List<Achat>();
    }
    public void vider() {
        panier.clear();
    }
}
```

On remarque que :

- Le gestionnaire d'entité est créé et géré par le conteneur ce qui simplifie l'écriture
- Le conteneur gère également les transactions
- La méthode merge() apparaît; elle est liée au « détachement » des entités

¹De là son nom même si je le trouve réducteur et peu représentatif.

4 GÉRÉ OU DÉTACHÉ ? (OU LA FIN DES DTO)

Dans l'application de l'année passée, une méthode comme « retourner() » aurait reçu un DTO (Data Transfert Object) en paramètre. Nous n'en parlons plus ici. Pourquoi ?

Rappelons d'abord pourquoi les avoir introduits¹. La logique métier doit échanger des données avec la couche supérieure mais nous ne voulions pas lui retourner les objets que nous avons codé dans la couche inférieure (comme Lecteur) car cela lui aurait permis d'appeler directement les méthodes de l'objet et ainsi de contourner la couche métier. C'est pourquoi nous avons introduit les DTO qui sont une version ne contenant que les données sans aucune logique.

Les entités ont la particularité d'être « détachés » lorsqu'ils quittent la couche métier²; ils ne sont plus gérés par le gestionnaire d'entités et toute modification apportée n'a aucune incidence sur la BD. Ils se comportent comme des DTO. C'est pourquoi le code de la méthode retourner() comporte un appel à « merge() » qui « ré-attache » l'entité, c'est-à-dire demande au gestionnaire de la reprendre en charge.

Insistons! Un appel à un setteur n'aura pas le même effet si l'entité en question est gérée (la BD est modifiée également) ou pas (on ne touche pas à la BD)

5 GESTION DES TRANSACTIONS

Avec un bean de session, la gestion des transactions peut être laissée au conteneur. Par défaut, le conteneur crée une transaction à chaque appel si elle n'existe pas encore (mais cela peut être facilement configuré). En pratique cela signifie que tout appel d'une méthode de la logique métier à partir de la couche supérieure crée une transaction qui se termine automatiquement quand cette méthode se termine. Nous verrons plus tard dans l'année comment raffiner ce comportement.

6 LES EXCEPTIONS

Dans toute logique métier, il peut y avoir des requêtes qui ne peuvent être satisfaites (mauvais paramètres par exemple). La logique devra générer une exception. Il est utile que cette exception soit annotée par « @ApplicationException ». Il sera alors possible d'indiquer au conteneur quelle politique appliquer lorsque cette application surgit (par exemple : quelle stratégie utiliser par rapport à la transaction).

7 LE POOL DE CONNEXION

Le système va devoir effectuer de nombreux accès à la BD. Rappelez-vous le fonctionnement de JDBC, il faut créer une connexion pour pouvoir exécuter une requête SQL. Le pool de connexion est un ensemble pré-crée de connexions facilement réutilisables. Cela permet d'accroître la réactivité de l'application. Netbeans permet d'automatiser cette création.

¹Revoyez votre code de l'année passée.

²Plus précisément, quand ils quittent le conteneur qui les contient.

8 BIBLIOTHÈQUE

Nous avons recodé pour vous une toute petite partie de la couche métier pour la bibliothèque. Faites tourner le projet fourni et puis reproduisez le.

Étapes : Voici, en résumé les différentes étapes à suivre

1. La BD est normalement déjà créée
2. Créez un projet de type 'entreprise application' (cochez la case demandant d'y ajouter un module client - cela servira pour le test - et décochez celle correspondant au module web)
3. Créez un pool de connexion (new File/ Sun resources / JDBC Connection Pool)
4. Dans le module 'ejb', créez une unité de persistance reliée à la BD (bien demander de ne pas créer la table puisqu'elle existe déjà)
5. Dans le module 'ejb', créez les entités à partir de la BD (soignez vos noms de package)
6. Lisez le code produit et apportez-y les éventuelles modifications opportunes (meilleurs noms par exemple). Ici, pour aller plus vite, vous pouvez aussi copier/coller le code fourni.
7. Copier/coller le bean de session « GestionLecteurFacadeBean » ainsi que les interfaces et exceptions liées.
8. Afin de le tester, copiez la classe de test du projet fourni. Il faut également ajouter au module client la librairie Toplink s'occupant de la persistance.
9. Compilez, déployez et exécutez le projet.

9 VENTE EN LIGNE

Inspirez-vous du projet « bibliothèque » pour coder la couche métier de la vente en ligne.

Étapes : Voici, en résumé les différentes étapes à suivre

1. Reproduisez les étapes 1 à 6 ci-dessus.
2. Créez le bean de session « PanierBean » (stateless ou statefull ?). Il doit permettre de créer un panier, et d'y ajouter un achat (Achat est un JavaBean simple)
3. Afin de le tester, vous pouvez créer une petite application cliente non graphique. Pour cela, il suffit de remplir le Main proposé dans le module « app-client ». Pour utiliser un bean de session : dans le source – clique-droit – Enterprise Ressource – Call Enterprise Bean. (n'oubliez pas Toplink).
4. Créez le bean de session « CommandeFacade » qui propose de passer une commande (à partir d'un panier) et d'informer sur une commande (à partir du numéro). Testez également cette partie.