5 mai 2016

Léo Letourneur – LOIC GERLAND

INFO Répartie Projet Ergosum

# POLYTECH

Architecture de Spring

Spring est un framework de développement Java basé sur la notion de conteneur léger en opposition aux serveurs d'applications Java EE. Une application utilisant Spring est le plus souvent structurée en trois couches (MVC) :

La couche présentation : interface homme machine

La couche service : interface métier et traitement métier

La couche persistance : Accès aux données, recherche et persistance des objets en base de données.

Configuration

Maven est un outil de construction de projets qui facilite et automatise certaines tâches d'un projet Java :

* Automatiser certaines tâches : compilation, tests unitaires et déploiement des applications
* Gérer des dépendances de bibliothèques nécessaires au projet
* Générer des documentations

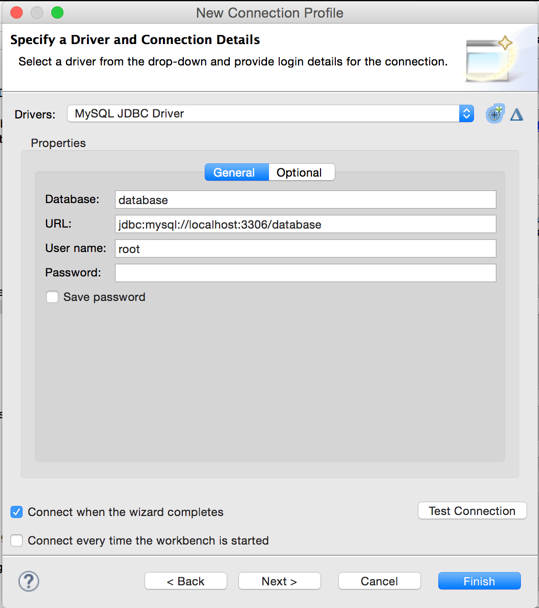
**Pom : project object model :**

C’est un fichier XML qui contient une description détaillée du projet, avec des informations concernant le versionning et la gestion des configurations, les dépendances, les ressources de l'application, les tests... C’est sur ce fichier que Maven s’appuie pour télécharger les dépendances d’un projet.

**Création des classes métier en utilisant Hibernate :**

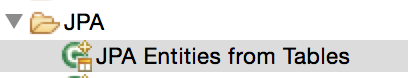
Configuration :

Avant tout chose, si l’on utilise un projet avec Maven, il faut ajouter la dépendance à Hibernate dans le pom.xml. Ensuite il faut installer une bibliothèque JPA au projet et la configurer avec la plateforme souhaitée et la connexion à la base de données.



Les classes métier :

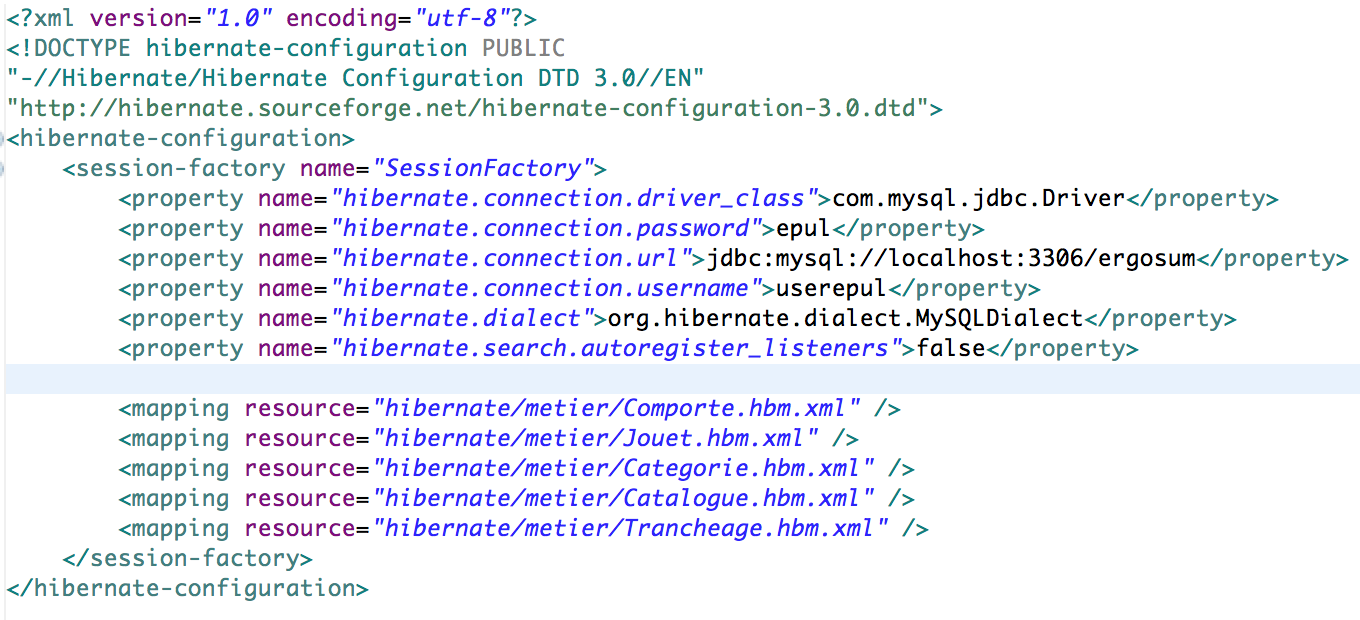
Lorsque la connexion et la configuration de Hibernate est effective on peut alors créer les classes métier. Pour cela, dans éclipse il faut faire new JPA Entities from Tables :



Vous devez sélectionner la connexion précédemment créée et les tables correspondantes à vos classes. Vous pouvez alors choisir les liens entre les tables.

|  |  |
| --- | --- |
| Capture%20d’écran%202016-05-05%20à%2010.48.51.png | Capture%20d’écran%202016-05-05%20à%2010.49.08.png |
|  |  |

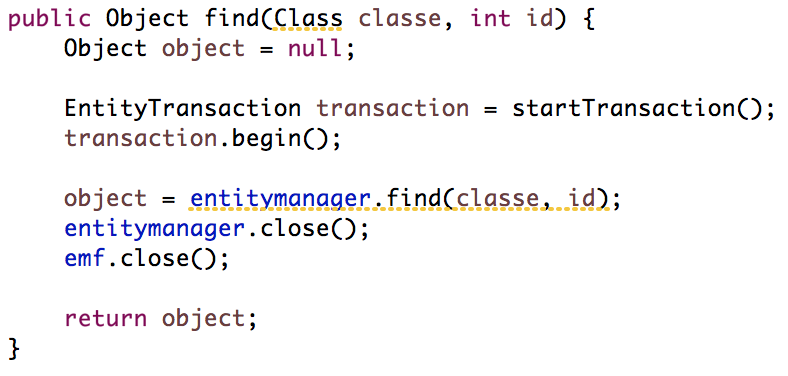
Enfin, il faut compléter le fichier hibernate.properties ou hibernate.cfg.xml avec les propriétés de la base de données et les classes générées.



Les classes métier sont alors générées. Il y a ici deux possibilités, soit les classes comporte des annotations de mapping, soit les classes sont associées à des fichiers hbm.xml qui comportent les annotations.

|  |  |
| --- | --- |
| Annotations | Hbm.xml |
| Capture%20d’écran%202016-05-05%20à%2011.00.00.png | Capture%20d’écran%202016-05-05%20à%2010.59.40.png |

Il est ensuite possible de créer des services très épurés comme ci-dessous :



Rôle du fichier hibernate.cfg.xml :

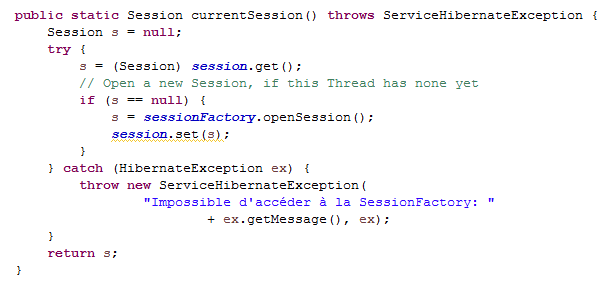
En remplacement ou en complément au fichier hibernate.properties, le fichier hibernate.cfg.xml permet de définir les propriétés concernant la connexion à la base de données. Les propriétés sont alors définies par un tag <property>. Le nom de la propriété est définie grâce à l'attribut « name » et sa valeur est fournie dans le corps du tag. Il permet aussi de définir le mapping entre les classes et les fichiers hbm.xlm comme par exemple :

<mapping resource="Adherents.hbm.xml"/>

**La couche service avec la classe service :**

Quelle est son rôle :

Le but de la couche de service est de définir les fonctions qui permettent d’interagir avec la base de données. Pour cela, elle s’appuie sur les classes métier générées par Hibernate et sur une classe qui va instancier une session afin de dialoguer avec la base de données. Généralement on définit deux fonctions dans cette classe qui vont permettre l’ouverture et la fermeture de la session. Par exemple le code pour ouvrir une session :



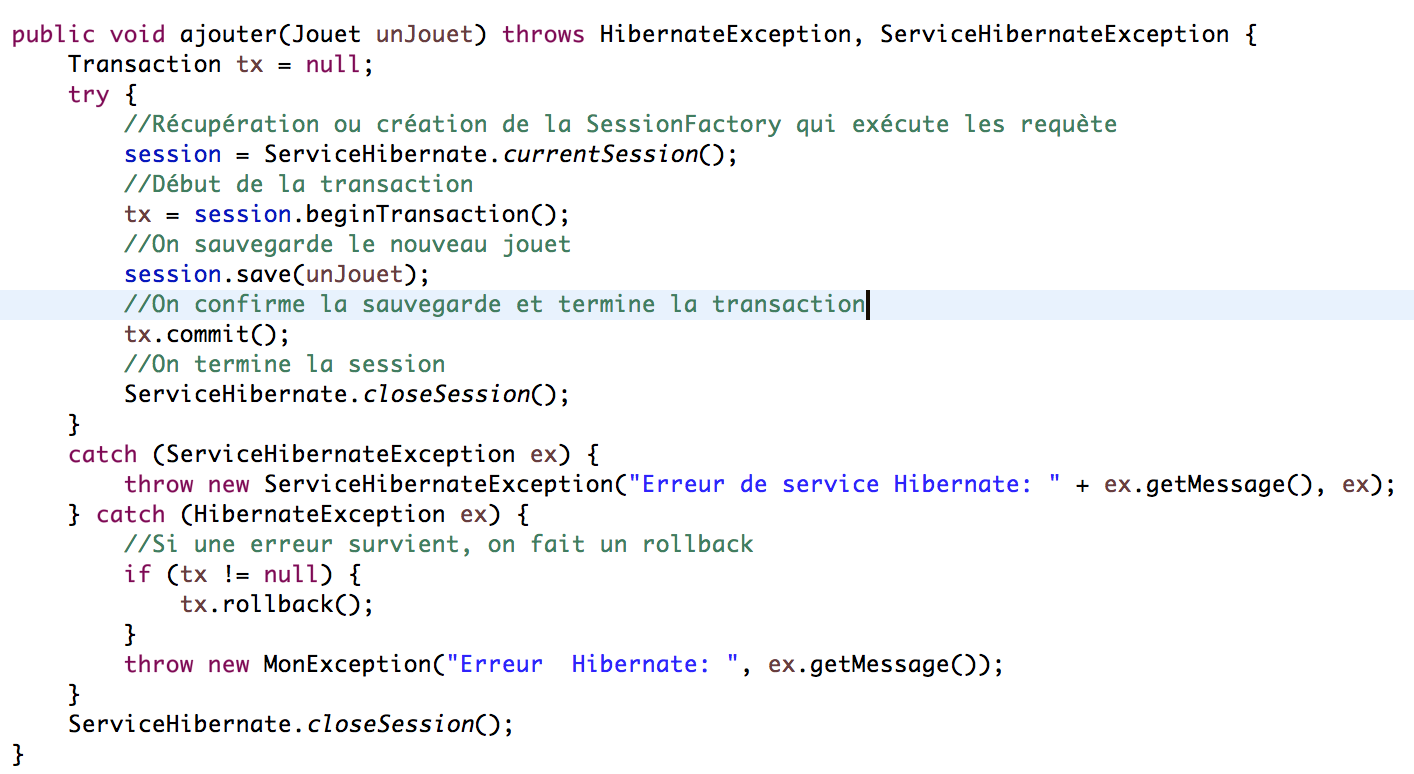
Quel est le code qui lit le fichier hibernate.cfg.xml :



La ligne de code ci-dessus est celle qui permet de configurer le sessionFactory à partir de la configuration présente dans le fichier hibernate.cfg.xml. Ainsi à partir de ce sessionFactory on peut instancier des sessions afin de communiquer avec la base de données.

La couche métier

La classe GestionErgosum permet d’exprimer les fonctions du contrôleur en requêtes que le ServiceHibernate va exécuter sur la base de données. Prenons comme exemple la méthode ajouter(Jouet unJouet) qui est appelée dans la fonction sauverJouet() du contrôleur. Le code a été commenté pour répondre à la question.



Les requêtes sont écrites en HQL (Hibernate Query Language). Hibernate propose son propre langage HQL dans le but d’offrir un langage d'interrogation commun à toutes les bases de données. Son intérêt est d'être indépendant de la base de données sous jacente : la requête SQL sera générée par Hibernate à partir du HQL en fonction de la base de données précisée via un dialect.

Exemple : SELECT t FROM Trancheage AS t

Exemple : Nombre de maison dans un pays (Plusieurs maisons peuvent être à la même adresse)

|  |  |
| --- | --- |
| SQL | HQL |
| SELECT COUNT (m.\*)  FROM maison m  JOIN adresse a ON m.idAdresse=a.id  JOIN pays p ON a.idPays=p.id  GROUP BY p.id | SELECT COUNT(m.id)  FROM maison as m  JOIN m.adresse as a  JOIN a.pays as p  GROUP BY p.id |

La couche Contrôle

- Spring et le contrôleur

* Etude de sa mise en place
* Rôle des fichiers

La couche Présentation

**Quelles sont les technologies utilisées :**

La couche présentation est surtout constituée de vues, de ressources comme les images et de fichiers JavaScripts. Cet ensemble d’éléments va définir le design du site internet.

L’élément important de cette couche sont les vues. Elles sont basées sur la technologie JSP (JavaServer Pages) qui permet de créer dynamiquement des pages HTML à partir de balises personnalisées. Pour rendre ces pages plus dynamiques et apporter des fonctionnalités supplémentaires, on utilise des fichiers JavaScripts dans lesquels nous pouvons écrire des fonctions afin de rajouter de traitements coté client.

**Les ressources images, le css :**

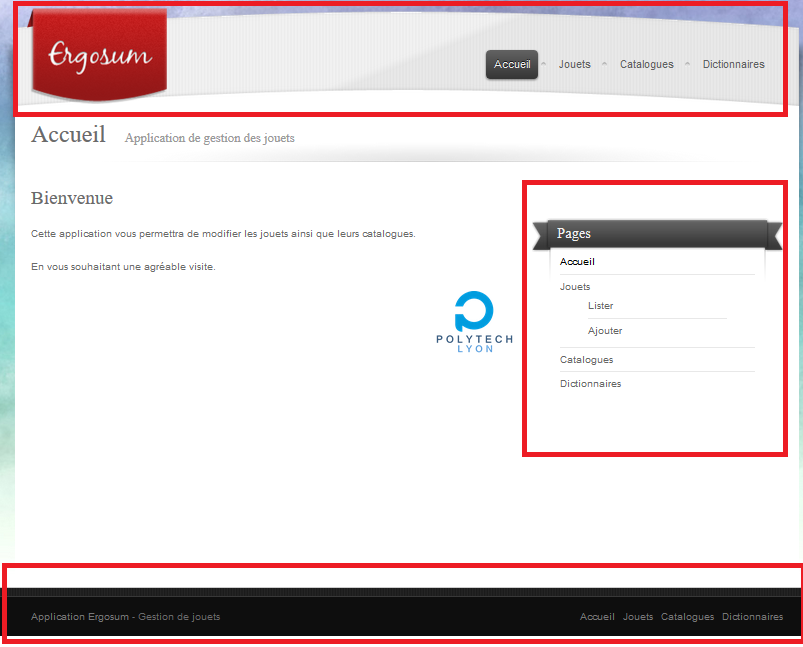
Les images et le css sont deux ressources qui définissent exclusivement l’aspect visuel des vues. Dans le projet Ergosum, il y a beaucoup d’images qui représentent plusieurs éléments du site comme des icônes, des boutons ou encore le background. Pour parfaire, la mise en page du site et l’insertion des images, on utilise des fichiers css (Cascading Style Sheets) pour définir spécifiquement l’aspect des éléments. Par exemple la taille d’un image, l’alignement d’une section, etc.

**Le rôle de la couche jstl :**

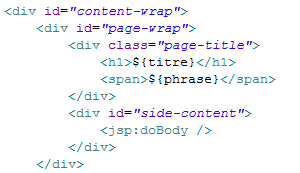
**???**

* Modification de traitements supplémentaires
* Point 1
* Layout

Toutes les pages du site ont une mise en forme commune qui est composé d’un header, d’une barre de navigation latérale sur la droite et d’un footer.



Cependant quand nous avons reçu le projet, tout le code de ce layout était écrit plusieurs fois dans les vues. Cela pose un gros problème de modularité car si on veut changer un des éléments du layout nous devons changer le code dans toutes les vues. Pour remédier à ce problème, nous avons donc mis en place un layout. Pour cela, nous utilisons une fonctionnalité du JSP qui nous permet de créer des balises personnalisées. Ainsi, dans un fichier layout.tag nous allons écrire le code du layout en rajoutant une balise spéciale <jsp :doBody> qui va indiquer l’endroit où le code des vues va être insérée.



Ensuite, il suffit de rajouter ce tag dans le code des vues pour indiquer qu’il faut le remplacer par le code du fichier.



Ici on peut remarquer deux attributs titre et phrase qui permette de passer des paramètres au layout pour dans notre cas changer le titre et la phrase qui est écrite dans le header en fonction de la vue.

* Root

Point 2

Présentation des jouets

* par catégorie
* tranche d’âge
* par quantité de distribution