Rapport de projet de Persistance des objets

Paraita Wohler (M1 IFI)
groupe HMABWE

16 novembre 2011

Mon travail dans le groupe a été de m'occuper de la liaison entre le code *métier* et la base de données. J'ai fait la conception des différents DAO, de l'objet Connexion, et de la persistance de la configuration (modification du fichier persistence.xml). Mon binôme Swan Engilberge s'est occupé d'une partie de l'abstraction des DAO pour le code de l'application ainsi que de l'implémentation de certaines méthodes. Nous avons tout les deux conçus la javadoc accompagnant les DAO ainsi que les requêtes qui allaient être utilisées. Pour rappel, l'application devait pouvoir communiquer avec une base de données, en utilisant un certain type de persistance, en l'occurrence JPA. L'application devait aussi pouvoir fonctionner en utilisant un autre type de persistance, en impactant le moins possible le code de l'application. La principale difficulté pour moi a été de concevoir une *couche* d'abstraction entre le code métier et le type de persistance. J'ai du travailler avec Anthony Biga qui s'est occupé de la création des objets métier (les entités), et nous avons occulté au reste du groupe le type de persistance. Les autres membres du groupe devaient voir les entités comme des objets manipulable pour la récolte d'informations localement, et toute requête dans la base de données se ferait aux DAO.
Il était nécessaire de factoriser le code tant que possible, d'une part pour éviter la redondance de code, mais aussi pour garder les DAO en un tout cohérent. C'est pourquoi j'ai crée un type DAO générique, qui me permettrait d'avoir le même fonctionnement pour n'importe quel DAO d'un objet métier, quelque soit le type de la clé primaire. J'ai utilisé des interfaces pour occulter le type *réel* des DAO, et grâce au pattern *Fabrique Abstraite* j'ai pu totalement *cacher* le type d'un DAO. On aura ainsi, par exemple, le code de l'application qui demanderait un DaoEtudiant au lieu d'un JpaDaoEtudiant ou JdbcDaoEtudiant. J'ai fait de même pour rendre occulter la connexion, dépendante du type de persistance. Ainsi dans le cas de JPA, l'application manipule une Connexion et non un EntityManager.
Enfin j'ai mis en place une gestion des exceptions ou j'attrape toute erreur interne aux DAO que je *masque* en erreur DAO, encore une fois pour cacher le type de persistance utilisé. Pour la modification du fichier *persistence.xml*, j'ai réutilisé un morceau de code que l'on avait vu en cours de Technos XML, qui charge le fichier en DOM, je le modifie avec la saisie de l'utilisateur, puis j'enregistre l'arbre ainsi modifié. L'abstraction du type de persistance par les DAO, la Connexion et les DaoException me parait convenable, car le reste du groupe qui devait travailler sur le code de l'application a effectivement réussi à manipuler les DAO sans à aucun moment chercher à faire des requêtes, ou encore s'occuper du cache des objets métier.