Rapport TP JEE 3
application Web avec JPA

Benabbou Chaimaa

Le but de ce TP est de créer un Annuaire permettant d'ajouter des personnes et des adresses, de les associer et ainsi lister chaque personne avec son adresse (ses adresses) correspondante(s). Pour cela, on utilise des Entity beans, Session bean, des jsp, EJB et 3 relations bidirectionnelles différentes entres les Entity (OneToMany. ManyToMany, OneToOne).

Dans ce TP, on a :

➢ ***Personne.java*** :↠ une Entity bean représentant l'enregistrement de la table personnes de la BD
↠ le type de récupération de l'attribut adresses est ***FetchType.EAGER.*** Donc, à chaque fois qu'on récupère une personne à partir de Facade.java ,on garantie que *adresses* est récupérée de la BD.

➢ ***Adresse.java*** :↠ une Entity bean représentant l'enregistrement de la table adresses de la BD

➢ ***Facade.java*** : ↠ une Session bean EJB qui sert à manipuler ces entités, contenant les méthodes nécessaires à la gestion des données issues de la BD
↠ représentant l'enregistrement de la table personnes_adresses de la BD (dépend de la relation choisie)

➢ ***Servlet.java*** :↠ l'accès aux données se fait par l'intermédiaire d'EJB session
↠ permet la création de pages HTML dynamiques

➢ ***index.html*** :↠ code html qui permet de générer la première page web de notre annuaire

➢ **ajoutPersonne.jsp, ajoutAdresse.jsp, associer.jsp, lister.jsp** :↠ ces pages ont la particularité de pouvoir mélanger du code java et du code html qui génèrent du code html à son tour pour les différentes pages html dont on a besoin.

Dans ce projet, j'ai implémenté 3 versions avec différents types de relations entre les Entity beans:

➢ ***OneToMany*** (bidirectionnelle): en utilisant cette relation, on assure que chaque adresse a un unique propriétaire. Donc quand on associe une adresse déjà prise par une personne à un autre propriétaire, cette adresse n'appartient plus à la première personne mais seulement à la dernière à qui on l'a associée. Dans ce cas, la classe Personne a une liste d'adresses et la classe Adresse a une référence à son propriétaire unique.

➢ ***ManyToMany*** (bidirectionnelle): en utilisant cette relation, une personne peut avoir plusieurs adresses et une adresses peut avoir plusieurs propriétaires. Pour cela, une personne a une liste d'adresses et une adresse a une liste de owners.

➢ ***OneToOne*** (bidirectionnelle) : en utilisant cette relation, une personne ne peut avoir qu'une seule adresse et une adresse ne peut avoir qu'un seul propriétaire. Pour cela, dés qu'on associe une adresse A à personne P, on vérifie si le owner de A et l'adresse de P sont tous les deux *nulls*, si ce n'est pas le cas on les mets à null, et on fait la bonne association. Ainsi, on ne tient compte que de la dernière association faite.

Dans ce TP, j'ai choisi de ne tenir compte que de la dernière association faite. Mais j'aurais pu tenir compte que de la première association en vérifiant, *dans le cas de OneToOne*
que si le owner de A et l'adresse de P sont ne sont pas *null, on ne fait pas de nouvelles association et dans le cas de OneToMany, de ne pas faire d'association si l'adresse a déjà un propriétaire.*