Applications web

TP3 - Compléments

Rappels

- Sujet et softs sur http://sd-127206.dedibox.fr/hagimont/resources-N7/teaching-N7.html
- IMPORTANT : à mettre dans .bashrc

  export JAVA_HOME=/mnt/n7fs/ens/tp_dh/jdk1.8.0_45

  export PATH=$JAVA_HOME/bin:$PATH

  export PATH=/mnt/n7fs/ens/tp_dh/eclipse:$PATH

  ou sous forme d'alias à exécuter dans le terminal concerné :

  alias awenv='' export JAVA_HOME=/mnt/n7fs/ens/tp_dh/jdk1.8.0_45; export PATH=$JAVA_HOME/bin:$PATH; export PATH=/mnt/n7fs/ens/tp_dh/eclipse:$PATH''

- Dans le terminal de lancement de jboss et d'eclipse, vérifier avec les commandes :
  - which java
  - which eclipse
- lancer jboss comme décrit dans le sujet du tp1, et vérifier : localhost :8080
- Créer une application web (Dynamic Web Projet), comme décrit dans le tp1
  - Sans runtime
  - En ajoutant les 3 jars indiqués
- Les fichiers html et jsp doivent être mis dans le dossier « WebContent »
- Exporter l'application en .war et la déposer dans dossier_ jboss/standalone/deployments

Spécificités du TP3

Même application annuaire qu'en TP1 et TP2, mais suppression des hash maps personnes et adresses. Les données seront stockées dans une bd interne à jboss, et gérées avec des entity beans :

- ajout de l'annotation @Entity pour la calsse Personne et la classe Adresse
- un champ clé primaire doit être annoté @ID, et s'il est entier généré automatiquement : @GeneratedValue(strategy=GenerationType.AUTO)
- Dans la Facade, une variable particulière permet de faire des appels à Jboss pour la gestion des objets persistants (dans la BD). Cette variable (em) est de type EntityManager. En mettant une annotation avant (@PersistenceContext), on a une

injection de dépendance (comme avec la variable Facade dans la servlet avec l'annotation @EJB), ce qui signifie que cette variable est initialisée par Jboss

- Lorsqu'on crée une instance d'un entity, pour que cet objet soit géré dans la BD, il faut appeler em.persist() en passant la référence à l'instance. L'objet est alors recopié dans la table associée à l'entity dans la BD
- Si on veut retrouver une instance qui est dans la base de donnée à partir de sa clé primaire, on appelle em.find() en passant la classe de l'entity et la clé primaire : par exemple em.find(Personne.class, personneId)
- On peut également retrouver par exemple une collection d'objets avec une requête, par exemple : return em.createQuery("from Adresse", Adresse.class).getResultList();
- Différents modes d'associations peuvent se faire entre les « entities ». Par exemple :
    - Une personne ←→ plusieurs adresses :
      - Un champ « proprietaire » est nécessaire dans adresse et doit être annoté : @ManyToOne // plusieurs adresse pour un propriétaire
      - Un champ « collection d'adresse » est nécessaire dans Personne, et doit être annoté : @OneToMany(mappedBy="proprietaire", fetch = FetchType.EAGER)
    - Plusieurs personnes ←→ plusieurs adresses :
      - Une collection « proprietaires » est nécessaire dans adresse et doit être annotée : @ManyToMany // plusieurs adresse pour un propriétaire
    - Une personne ←→ Une adresse : un champ « adresse » dans personne annoté @OneToOne(mappedBy="proprietaire", fetch = FetchType.EAGER)