Plateformes de Développement d'Applications Distribuées

Travaux Dirigés N° 03

Persistance

Préliminaires : installation de J2SE et de NetBeans

L'installation de l'environnement de développement **NetBeans 6.7.1 + J2SDK 1.6** sur les machines de l'ENIS se fera par les étudiants sur les postes Ubuntu. Il s'agit simplement de récupérer une archive précompilée préparée pour les TDs. Cette archive peut aussi être utilisée sur tout PC exécutant le système d'exploitation Ubuntu (8.04, 8.10 ou 9.04).

Récupération et extraction de l'archive

Après avoir récupéré l'archive ENIS_J2EE_Package.tgz, il faudra l'extraire. Ensuite, dans un terminal, saisir les commandes suivantes :

```
~% tar xzvvf /chemin/vers/ENIS_J2EE_Package.tgz
~% cd  ENIS_J2EE_Package
~% ./install.sh
```

À l'issue des deux commandes ci-dessus, on disposera d'un répertoire **`${HOME}/J2EE`** contenant l'installation de **NetBeans** ainsi que des outils de développement et d'exécution **JAVA 6** de SUN. On disposera aussi d'un fichier nommé **`env.sh`** qui permet de placer correctement les variables d'environnement nécessaires pour la compilation, le déploiement et l'exécution des applications réparties. Ce fichier doit être **"chargé" <u>chaque fois</u>** qu'un nouveau terminal (shell) est ouvert. Le chargement du fichier **`env.sh`** s'effectue, **<u>dans le nouveau terminal</u>**, à l'aide des commandes suivantes :

```
~% source ${HOME}/J2EE/env.sh
```

Pour lancer **NetBeans**, il suffit de saisir la commande `netbeans&` dans un terminal.

L'objectif de ces TD est de se faire familiariser avec L'API de persistance JAVA. Il s'agit de créer un système de gestion d'enseignement dans un établissement semblable à l'ENIS.

Exercice 01 : Préparation des entités

L'objectif de cet exercice est de créer un ensemble d'entités JAVA représentant

- Un étudiant caractérisé par : (Table `EJB_ENSEIGNEMENT_ETUDIANT`)
  - Son numéro de carte d'identité nationale : `cin` (Clef primaire de type chaîne de caractères)
  - Son nom de famille : `nom`
  - Son prénom : `prenom`
  - Sa date de naissance : `date` (de type `java.util.Date`)
  - Son niveau : `niveau` (entier de type `Integer`)
- Un cours caractérisé par : (Table `EJB_ENSEIGNEMENT_COURS`)

  - Son nom : `titre` (Clef primaire)
- Un enseignant caractérisé par : (Table `EJB_ENSEIGNEMENT_ENSEIGNANT`)
  - Son nom : `nom`
  - Son prénom : `prenom`
  - Son grade : `grade` (*assistant*, *maitre-assistant*, *maitre de conférences* ou *professeur*)
  - **La clef primaire est la combinaison des champs nom et prénom**
- Un département caractérisé par : (Table `EJB_ENSEIGNEMENT_DEPARTEMENT`)
  - Son nom : `nom` (Clef primaire)

Question 1 : Création de la base de données et de la connexion à cette base

L'outil **Netbeans** permet de faciliter la création de bases de données et d'automatiser la connexion à ces bases. Dans la section *Services → Data Bases*, cliquer avec le bouton droit sur **Java DB** et choisir *" Create Database "*. Choisir les caractéristiques suivantes de votre base :

- Nom : `Enseignement`
- Utilisateur : `admin`
- Mot de passe : `admin`

Question 2 : Création des entités

Créer un nouveau projet `gestion-enseignement` composé d'un module EJB ainsi que d'un module Client JAVA classique

Q2.1 : Dans le module EJB ajouter les 4 classes `Etudiant`, `Cours`, `Enseignant`, et `Departement` correspondant aux entités décrites plus haut. Ajouter aussi la classe `EnseignantPK` qui représente la clef primaire composite pour l'entité Enseignant.

N.B 1 : Lors de la création de la première entité, **Netbeans** va demander de créer une **unité de persistance**. Il faut créer cette unité et la connecter à la base de données créée dans la question précédente.

N.B 2 : Il faut demander à l'assistant de création de l'unité de persistance de créer automatiquement les tables au déploiement. Ceci se traduira par l'ajout des lignes suivantes dans le fichier `persistance.xml` :

```
<provider>oracle.toplink.essentials.PersistenceProvider</provider>
  <jta-data-source>jdbc/enseignement</jta-data-source>
  <properties>
    <property name="toplink.ddl-generation" value="drop-and-create-tables"/>
  </properties>
```

N.B 3 : Il ne faut pas hésiter à utiliser les menus de **NetBeans** pour créer automatiquement les propriétés persistantes des entités. Ceci réduira considérablement le temps de développement.

Q2.2 : Modifier les entités créées pour refléter les liens suivants :

- Un cours est suivi par un ensemble d'étudiants (On utilisera pour cela une table additionnelle qui sera générée automatiquement :
  `@ManyToMany`

```
        @JoinTable(
            name = "EJB_ENSEIGNEMENT_COURS_ETUDIANT",
            joinColumns = @JoinColumn(name = "NOM_COURS",
                                      referencedColumnName = "TITRE"),
            inverseJoinColumns = @JoinColumn(name = "CIN_ETUDIANT",
                                             referencedColumnName = "CIN")
        )
    )
```

- Un cours est enseigné par un enseignant
- Un département est composé d'un ensemble d'enseignants
- Un département est composé d'un ensemble de cours
- Un enseignant donne un ensemble de cours
- Un enseignant est affilié à un département
- Un étudiant suit un ensemble de cours.

Question 3 : Création de l'EJB

Q3.1 : Créer 4 classes Java classiques (`EtudiantInfo`, `EnseignantInfo`, `CoursInfo` et `DepartementInfo`) reflétant chacune les détails des entités leur correspondant **sans pour autant contenir des informations sur les liens entre entités**. Ces classes seront utilisées pour répondre aux requêtes des clients.

Q3.2 : Créer un **EJB session avec état** exposants le méthodes suivantes :

- `void creerEtudiant`
  `(String cinEtu,`
  `String nomEtu,`
  `String prenomEtu,`
  `Date naissanceEtu,`
  `Integer niveauEtu);`
- `void creerEnseignant(EnseignantInfo infoEns);`
- `void creerCours(String nomCours);`
- `void creerDepartement(String nomDep);`

Compléter le code de ces méthodes dans la classes de l'EJB

N.B 1 : On utilisera la fonctionnalité de log présente dans l'API Java pour pouvoir tracer l'invocation de chacune des méthodes. Exemple :

```
private static final Logger logger =
    Logger.getLogger("enseignement.requete.EnsRequeteBean");
...
@PersistenceContext
private EntityManager em;
...
public void creerEtudiant(String cinEtu, String nomEtu, String prenomEtu,
        Date naissanceEtu, int niveauEtu) {
```



```
        logger.info("creerEtudiant: entrée");
        try {
            Etudiant e = ...
            em....
        } catch (Exception ex) {
            throw new EJBException(ex);
        }
        logger.info("creerEtudiant: sortie");
    }
```

Exercice 02 : Ajout d'un client JAVA

Ajouter un client Java permettant d'invoquer les méthodes de l'EJB. On utilisera ce client pour créer un ensemble d'étudiants, enseignants, cours, et départements.

Question 1 : Opérations basiques

Q1.1 : Ajouter la méthode suivante à l'EJB :

- `void ajouterEtudiant(String etudiantCIN, String coursTitre);`

Cette méthode permet l'ajout d'un étudiant à un cours donné. On utilisera la méthode `find` du persistance manager pour l'implanter.

Q1.2 : Ajouter dans la classe Etudiant une clause `NamedQueries` contenant une requête statique (nommée `"enseignement.entities.Etudiant.trouverAvecCours"`) permettant de trouver les étudiant suivant un cours donné.

Q1.2 : Ajouter la méthode suivante à l'EJB :

- `List<EtudiantInfo> trouverEtudiantSuivantCours(String coursTitre);`

Cette méthode utilise la requête créée précédemment pour trouver tous les étudiants suivant un cours donné. Tester cette méthode avec le client pour afficher du coté client son résultat.

Question 2 : Enrichissement

Par analogie la méthode `ajouterEtudiant`, ajouter les méthodes :

- `ajouerEnseignant` qui permet d'ajouter un enseignant à un département donné
- `ajouterCours` qui permet d'ajouter un cours à un enseignant donné

Par analogie à la méthode `trouverEtudiantSuivantCours`, ajouter les requêtes et les méthodes métier :

- `trouverEtudiantsSuivantEnseignant` qui permet de trouver les étudiants d'un enseignant donné
- `trouverEnseignantsSuivantEtudiant` qui permet de trouver les enseignants d'un étudiant donné