Laboratoire n°2 de Systèmes distribués

Madani, Mounawar
mounawar.madani@hepl.be

Caprasse, François
francois.caprasse@hepl.be

Thiernesse,Cédric
cedric.thiernesse@hepl.be

8 octobre 2012

L'énoncé en bref

Il s'agit de programmer un jeu d'échecs en réseau. Deux joueurs lancent le visuel du jeu sur leur ordinateur respectif et jouent comme s'ils étaient l'un en face de l'autre. On simplifiera les régles dans un premier temps(pas de rock par exemple). Remarque : il ne s'agit pas ici d'implémenter un algorithme pour que l'ordinateur joue automatiquement contre un joueur.

Table des matières

1 Consignes

Reportez-vous aux modalités d'évaluation pour les consignes générales. Le laboratoire est à rendre par mail pour la date précisée dans le planning du cours(semaine du 26 novembre 2012).

2 Principales fonctionnalités à programmer

De la liste ci-dessous peut découler des sous fonctionnalités qu'il faudra implémenter. On supposera que les joueurs sont déjà enregistrés dans le système.

- Ajout d'un nouvel échiquier physique avec ses pièces(positionnées correctement au départ)[Administrateur]
- Création d'une nouvelle partie [User]. Vous pouvez considérer que les deux joueurs se sont mis d'accord au préalable et qu'un des deux crée la partie. Sinon, il s'agira ici de mettre en place un protocole de manière à ce que les deux joueurs puissent déclarer qu'ils souhaitent jouer ensemble. Pour ce faire, utilisez les techniques qui vous semblent le plus approprié(Appel synchrnone de type RPC, messages asynchrones, combinaison des deux, ...)

– Autorisation du déplacement d'un pion, stockage du coup joué et mise à jour de l'interface graphique[User]

3 Le jeu d'échecs : les règles et mouvements de base des pièces

Des explications sur le jeu d'échecs peuvent être trouvées facilement sur le net. Par exemple : `http://fr.wikipedia.org/wiki/Echecs` [4].

4 Techniques à utiliser

Les technologies à utiliser pour réaliser le laboratoire sont celles de la plateforme JEE. Principalement, on utilisera donc des EJB(stateless ou/et stateful, JMS et les entités pour la réalisation de l'énoncé.

– Configurer certaines relations d'entités avec le mode cascade pour la création des objets(par exemple, si on crèe un échiquier, on peut créer aussi les cases et les pièces).
– Utiliser la génération automatique des id pour les clés primaires.
– Configuration d'une stratégie d'héritage sur les entités qui héritent d'une classe.
– Utilisation des annotations relatives aux rôles pour l'accès aux fonctionnalités.

5 Pour démarrer ...

Cette section a pour but de vous permettre de démarrer rapidement. Elle ne se substitue en rien à la recherche individuelle sur le net, dans les tutoriaux et autres guides de développement. Si vous travailllez avec Netbeans, il faut créer différents projets : dans les grandes lignes, il s'agit donc de créer une enterprise application, qui contiendra un module jar contenant les différents EJB(pas de module war ici donc n'oubliez pas de décocher la case), ainsi qu'une enterprise application cliente que l'on peut ajouter dans le projet de l'enterprise application.

5.1 Etapes à réaliser pour créer les différentes applications

1. Fichier → New project → Java EE → Enterprise Application.
2. Fichier → New project → Java EE → Enterprise Application client.
3. Sur le projet EAR, click à droite → Add Java EE Module...
4. Fichier → New project → Java → New Java Class Library. Ce jar est destiné à contenir les interfaces de l'EJB.

5.2 Ajouter des EJB

1. Ajout d'un EJB session : click à droite sur le module EJB -> New ... SessionBean.
2. Ajout d'une méthode business : click à droit dans le code de l'EJB session → Insert code → Add business method.

5.3 Ajouter une référence à un EJB dans du code client

Si le code client est managé(container client ou container d'EJB), on peut utiliser l'injection des dépendances pour initialiser une variable permettant de contacter un EJB. Le code généré ci-dessous fonctionne dans un container client(variable en static) :

```
@EJB
private static SessionBeanRemote sessionBean;
```

Pour générer ce code : dans la classe du main, click à droite → Insert Code → Call Enterprise Bean. Il faut que l'EJB expose une interface Remote pour que cela fonctionne. La démarche est la même si vous voulez appeler un EJB depuis un autre EJB.

5.4 Générer un message JMS

1. Dans le package du composant concerné, click à droite → New → Other → GlassFish → JMS Resource.
2. Dans le code source, click à droite → Insert Code → Send JMS Message.

Normalement, un fichier glassfish-resources.xml est créé lorsque vous ajoutez une ressource JMS. Lors du déploiement, les objets (Connection Factory ou Destination) sont créés sur le serveur.

Si vous ajouter des properties à votre message, vous pourriez par exemple écrire le code suivant :

```
TextMessage tm = session.createTextMessage();
tm.setStringProperty("filtre","valeur");
tm.setText(messageData);
...
```

Il faudra éventuellement ajouter cette information sur le serveur dans la configuration des destinations.

5.5 Réception d'un message JMS

De nouveau, il est possible d'utiliser l'injection des dépendances pour obtenir la connection factory et la destination.

```
@Resource(mappedName = "jms/Topic")
private static Topic topic;
@Resource(mappedName = "jms/ConnectionFactory")
private static ConnectionFactory myFactory;
```

Dans l'exemple ci-dessous, la réception du message se fait de manière synchrone(le client bloque sur un receive). Le client ne désire les messages dont la propriété filtre a pour valeur 'valeur'.

```
... receiveJMSMessageToMyQueue() throws JMSException {
        ...
        Connection connection = null;
        Session session = null;
        TextMessage m =null;
        connection = myFactory.createConnection();
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

        connection.start();
       MessageConsumer consumer = session.createConsumer(topic,"filtre='valeur'");
        m = (TextMessage) consumer.receive();
        ...
        return m.getText();
    }
```

5.6 Création des entités et de la BD dans Netbeans

Aucun outil particulier n'est imposé donc si vous le souhaitez, vous pouvez utiliser une autre base de données que celle indiquée ci-dessous.

Un diagramme de classes pour le jeu est donné en figure 5.6.

1. Dans la vue des services, Services → Databases → Create database.
2. Create entities dans la vue du projet pour ajouter une nouvelle classe entité.
3. Si nécessaire, ajout du driver de la BD dans le projet. Votre projet → Libraries → click à droite → Add ...
4. Ajout d'une entité : Votre package → New Entity Class.
5. Ajout à la main des properties dans la classe entité précédement créée. Il faut régler le warning (point d'exclamation jaune) et configurer la relation(monodirectionnelle ou bidirectionnelle, ...).
6. Choisir la stratégie à appliquer sur la persistence unit(create, drop and create, none). Pour cela → fichier persistence.xml de votre projet. La base pourra donc par exemple être recréée au prochain déploiement sur base des entités du projet.

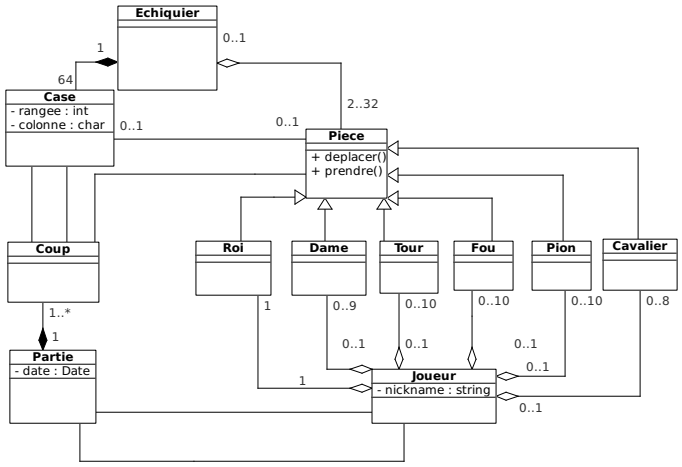

FIGURE 1 – Diagramme de classes simplifié d'un jeu d'échecs

5.7 Création du GUI

Les quelques fragments de code suivants sont destinés à créer rapidement et manuellement un GUI pour représenter le plateau, les cases et les pièces du jeu. Vous pouvez évidement utiliser/créer d'autres classes si vous le souhaitez. Par exemple, on pourrait utiliser autre chose qu'un tableau à deux dimensions pour représenter l'ensemble des cases du jeu.

Sur un JPanel, vous pouvez fixer la couleur comme ceci : `casejeu.setBackground(Color.DARK_GRAY);` Cela permet de spécifier la couleur d'une case du jeu. Pour créer le plateau et les cases :

```
public class GameGui extends javax.swing.JFrame {

    private CaseGui[][] cases = new CaseGui[8][8];
    private JPanel echiquier = new JPanel(new GridLayout(8, 8));
    ...
  public GameGui(){
    ...
    case[x][y].setBackground(Color.DARK_GRAY);
    ...
    echiquier.add(cases[i][j]);
   }
}
```

Les cases peuvent être par exemple des JPanel. L'ajout d'une pièce sur une case peut se faire de la manière suivante :

```
public class CaseGui extends JPanel
{
  public ... addPiece(...){

  add(new JLabel(new ImageIcon(filename)));
  ...
  }
}
```

Pour gérer la sélection et le déplacement des pièces, vous pouvez utiliser la technique du drag and drop [3].

6 Critères intervenant dans l'évaluation

– L'architecture doit pouvoir être justifiée (DAO, Service Layer,...).

- Le code généré inutile doit être retiré(dans la pratique on ne le retirerait pas mais le but ici est de vérifier que vous comprenez ce qui est généré pour vous.
- La recherche de la meilleure solution. Par exemple, on ne charge pas l'ensemble d'une table en mémoire pour ensuite la filtrer soit-même ! De la même manière, on ne calculera pas soit même un nouvel id sur base du plus grand id déjà présent dans un table donnée.
- Utilisation adéquate des différents méchanismes de rappel(callbacks sur les EJB, sur les entités, ...).
- ...

Liens utiles

[1] Tutorial Java EE : `http ://docs.oracle.com/javaee/6/tutorial/doc/`.

[2] Tutorial Java SE : `http ://docs.oracle.com/javase/tutorial/`.

[3] Swing Drag and Drop : `http ://docs.oracle.com/javase/tutorial/uiswing/dnd/intro.html`.

[4] Règles du jeu d'échecs : `http ://fr.wikipedia.org/wiki/Echecs`.