Rapport Middleware

M2 ALMA – 2018/2019 : Daniel AHMED, Pierre CAILLAUD, Demetre PHALAVANDISHVILI

Introduction

Dans le cadre du projet de l'unité d'enseignement Middleware, nous avons passé en revue deux projets effectués par des étudiants des années passées. Nous avons dû réaliser une critique de chacun de ces projets. Le but étant de déterminer les poins forts et les points faibles de chaque projet, ainsi que d'éventuelles pistes d'amélioration.

Nous avons ensuite modifié l'un de ces projets afin de l'améliorer en nous basant sur la critique précédemment effectuée. Après analyse, il nous a parut plus intéressant d'apporter des modifications au projet Gringott car il nous paraissait, à première vue, moins bien réalisé. Nous nous sommes par conséquent contentés de critiquer le projet Pay2bid sans le modifier.

Projet Pay2Bid : Critique du projet

Dans l'ensemble, le projet est plutôt bien structuré. Les choix des noms de classes et méthodes sont pertinents et les méthodes sont commentées. Ce qui rend la compréhension du projet relativement aisée et rapide.

Côté client, on observe que l'implémentation du timer pour les enchères est faite ici. On évite ainsi les problèmes liés à l'implémentation de ce dernier côté serveur (latence réseau, synchronisation ...). En revanche, du point de vue de la sécurité, on ouvre la porte à d'éventuelle modifications de l'utilisateur et aux conséquences qui en découlent. Il suffit que la méthode timeElapsed() ne soit jamais appelée chez un client pour provoquer le blocage d'une enchère.

Côté serveur, le choix a été fait de considérer le serveur entier comme un moniteur de Hoare. Ça a l'avantage d'empêcher d'éventuelles collisions. En revanche, cela rend impossible l'ajout d'un nouveau client lorsqu'une enchère est en cours.

Par ailleurs, le serveur n'est capable de gérer qu'une seule enchère à la fois. Les enchères suivantes sont placées dans une file d'attente le temps que l'enchère courante se finisse. Ça permet de grandement simplifier les système d'enchères mais ça limite aussi énormément l'utilisation de ce système par les clients.

En ce qui concerne les renchérissements, la méthode raiseBid() étant synchronized on ne peut pas avoir de collision sur une enchère. Par contre, côté client, le nom de celui qui a enchérit n'apparait pas. En tant que client, il m'est donc impossible donc de savoir si c'est bien moi qui détient l'enchère la plus haute ou un autre client qui aurait renchérit du même montant juste avant moi.

Pour conclure, le projet est plutôt solide du point de vue de sa conception, les quelques points négatifs sont essentiellement dus au nombre de fonctionnalités limité proposé par l'application.

Projet Gringott : Evolution du projet

Nous avons corrigé ce qui ne nous semblait pas bien implémenté dans l'ancien projet gringott. Notamment la synchronisation qui était réalisée côté client au lieu d'être faite côté serveur. Nous avons identifié les problèmes posés par cette décision, en particulier au moment où deux utilisateurs essayaient de se connecter le serveur ne faisait pas la différence et enregistrait deux utilisateurs avec le même pseudo. De plus dans l'implémentation de départ nous avons la possibilité d'enrichir sans arrêt tant que l'enchère n'est pas fini. Nous avons corrigé cette fonctionnalité et dans notre version on peut enrichir que lorsqu' un autre utilisateur à renchérit.

Nous avons opté d'utiliser Spring boot pour faciliter le développement et le déploiement de l'application. Ce qui nous a permis de faire la refonte de l'IHM en swing vers une IHM web réalisée en HTML5, CSS3 et angularjs. Cette implémentation en IHM web nous a permis de facilement ajouter les fonctionnalités qui manquaient dans l'implémentation de départ. En particuliers nous avons ajouté la possibilité de consulter nos enchères ajoutées.

Avec l'intégration des sockets web côté client, nous avons intégré de la réactivité dans l'application. Ceci nous a permis d'accéder au changement sur le serveur sans avoir besoin de recharger la page. Le scénario d'utilisation : un utilisateur A ajoute un nouvel objet à vendre et utilisateur B qui se trouve sur la page de liste des Enchères voit automatiquement l'objet ajouté.

Configuration du projet

Pré-requis du projet

Maven :

- version : 3.5.4 ou ultérieur

Eclipse:

- Dans marketplace, installer Spring tool 4
- Importer le projet
- Activer les annotations processing
- Avec le terminal dans le dossier racine du projet, il faut lancer mvn clean package

Intellij ultimate:

- Importer le projet
- Activer les annotations processing

Lancement du projet

Via IDE : Eclipse

Etape 1 : Lancement du serveur

On sélectionne le dossier du projet gringott-server et on exécute en tant que (Run as) Spring Boot App.

Etape 2 : Lancement du client

On sélectionne le dossier du projet gringott-client et on exécute en tant que (Run as) Spring Boot App

Etape 3 : Interface web

Une fois serveur et client lance pour accéder à l'IHM web il suffit d'ouvrir le navigateur web préféré sauf IE et accéder à IHM web via l'adresse suivante : http://localhost:3000/

Via IDE : Intellij Ultimate

Etape 1 : Lancement du serveur

Pour lancer serveur sur IDE Intellij il faudra regarder en haut a gauche de tableau de bord et sélectionner *GringottServerApplication* puis appuyer sur l'exécuter

Etape 2 : Lancement du client

Pour lancer serveur sur IDE Intellij il faudra regarder en haut à gauche du tableau de bord et sélectionner *GringottClientApplication* puis appuyer sur l'exécuter

Etape 3 : Interface web

Une fois serveur et client lancés, pour accéder à l'IHM web il suffit d'ouvrir son navigateur web préféré sauf IE et accéder à l'IHM web via l'adresse suivante : http://localhost:3000/

Via Terminal

Pour utiliser notre application, d'abord il faut lancer le serveur puis le client. Pour cela il faut suivre les étapes suivantes.

Etape 1 : Lancement du serveur

Ouvrir le terminal et se mettre dans le dossier gringott-server et taper la commande suivante

```
mvn spring-boot:run
```

Etape 2 : Lancement du client

ouvrir le terminal et se mettre dans le dossier gringott-client et taper la commande suivante

```
mvn spring-boot:run
```

Etape 3 : Interface web

Une fois serveur et client lancés, pour accéder à l'IHM web il suffit d'ouvrir son navigateur web préféré sauf IE et accéder à l'IHM web via l'adresse suivante : http://localhost:3000/.

Configurer Serveur chez client

Par défaut, si le client et le serveur s'exécutent sur la même machine, il n'y a pas besoin de configurer le serveur chez le client. Dans le cas contraire quand le client se trouve sur l'autre machine il faut modifier l'adresse du serveur chez le client. Pour cela dans le fichier `application.properties` se trouvant dans le dossier `middleware\gringott-client\src\main\resources` on doit modifier la première ligne en mettant l'adresse IP de la machine sur laquelle le serveur tourne.

Configurer plusieurs client sur la même machine

Par défaut, l'IHM web se trouve a l'adresse suivante : *http://localhost:3000*, mais nous avons la possibilité d'avoir plusieurs client qui tournent sur la même machine. Pour cela dans le fichier `application.properties` se trouvant dans le dossier `middleware\gringott-client\src\main\resources` on modifie l'option *server.port* en le mettant à 0. Ce changement nous donne uns *port* différent à l'exécution de l'application client pour accéder a l'IHM.