TP JXTA

1 Introduction

L'objectif de ce TP est de vous faire découvrir les concepts de base de JXTA :

1. découverte dynamique des pairs présents sur le réseau ;
2. communication par canaux (pipes) entre les pairs ;
3. définition et mise en place d'un service JXTA.

2 Mise en place de l'environnement

Télécharger le fichier `http://pagesperso-systeme.lip6.fr/Sebastien.Monnet/tmp/tme-jxta.tar.bz2` dans votre répertoire `$HOME`, puis le décompresser pour créer le répertoire de travail pour ce TP .

Ce répertoire contient plusieurs sous-répertoires et un fichier :

- Le répertoire `PeerDiscovery` correspond à la première partie du TP.
- Le répertoire `PipeService` correspond à la seconde partie du TP.
- Le répertoire `JXTAService` correspond à la troisième partie du TP.
- Le fichier `useJXTA` comporte quelques lignes de configuration de votre environnement JXTA. Charger ce fichier : `source useJXTA` ou `. useJXTA`. Attention, il faut décommenter, voire modifier certaines lignes selon le type de shell que vous utilisez et afin d'indiquer le bon chemin d'accès. Ces lignes définissent quelques variables d'environnement :
  - La variable `JXTA_HOME` indique le répertoire qui contient les archives `.jar` de JXTA (il s'agit du sous-répertoire lib de votre répertoire de travail).
  - La variable `CLASSPATH` contient la liste des archives `.jar` utilisées par JXTA. La version de JXTA utilisée est la 2.5, qui nécessite Java 1.5 (minimum).

3 Première partie : découverte des pairs JXTA

Le but de cet exercice est de programmer un pair qui invoque le service de découverte pour trouver d'autres pairs. À l'initialisation (méthode `startJxta`), le pair instancie le groupe par défaut (`netPeerGroup`) et initialise la variable `discovery`, qui doit contenir une référence vers le service de découverte du groupe. Le pair enregistre un handler pour gérer la réception d'annonces et exécute ensuite une boucle infinie qui lance périodiquement des messages de découverte d'annonces (*Advertisement*) de pairs. Chaque réponse reçue déclenche l'invocation de la méthode `discoveryEvent`, qui affiche les noms des pairs découverts.

Étape 1: *Compléter le code du fichier `PeerDiscovery.java` en vous laissant guider par les commentaires. Vous pouvez vous servir de la documentation des classes JXTA disponible dans le sous-répertoire doc de votre répertoire de travail.*

Étape 2: *Compiler ce code :* `javac PeerDiscovery.java`

Étape 3: *Lancer le pair :* `java PeerDiscovery`

La première exécution du pair déclenche l'apparition d'une fenêtre de configuration du pair. L'utilisateur soit rentrer au moins le nom du pair, ainsi qu'un couple login/mot de passe. Il peut aussi configurer le pair pour supporter des fonctionnalités de type *rendez-vous* ou *relais*. Il peut encore initialiser le pair avec une liste de pairs de rendez-vous ou de relais à utiliser. Une fois la configuration finie, la fenêtre se ferme et un répertoire `.jxta` est créé dans le répertoire courant. Ce répertoire contient entre autres les informations de configuration. L'exécution suivante ne fera plus apparaître la fenêtre de configuration, mais juste une boîte de dialogue à deux champs (login/mot de passe). Pour reconfigurer le pair, il suffit d'effacer le répertoire `.jxta`.

Étape 4: *Pour le premier test, initialiser le nom du pair et le couple login/mot de passe. Désactiver HTTP. Laisser la case Multicast cochée. Garder les autres options par défaut (ne pas indiquer de pairs de rendez-vous, ne pas activer le mode rendez-vous, etc.). Observer la découverte dynamique des pairs qui tournent localement.*

Étape 5: *Arrêter le pair. Effacer le répertoire `.jxta` créé lors de l'exécution précédente. Reconfigurer le pair en désactivant le multicast. Garder les autres options à leur valeur par défaut. Observer le comportement.*

Étape 6: *Arrêter de nouveau le pair. Effacer le répertoire `.jxta` créé lors de l'exécution précédente. Reconfigurer le pair en désactivant le multicast et en indiquant le pair de rendez-vous local indiqué par le formateur. Garder les autres options à leur valeur par défaut. Observer le comportement.*

4 Deuxième partie : communication par canaux (pipes)

L'objectif de cet exercice est de programmer deux pairs et de les faire communiquer par un canal JXTA. On programmera séparément les deux pairs. Le premier pair (fichier `listener/PipeListener.java`) crée un *Input Pipe* en utilisant le *Pipe Advertisement* `examplepipe.adv` présent dans le répertoire `listener`. Ensuite il enregistre un handler qui sera appelé lors de chaque réception de message sur le pipe. Cet handler va afficher le message reçu, sous forme de chaîne de caractères.

Étape 1: *Compléter le code du fichier `PipeListener.java` en vous laissant guider par les commentaires.*

Le deuxième pair (fichier `listener/PipeSender.java`) crée un *OutputPipe* à partir d'une copie du même Pipe Advertisement `examplepipe.adv`, présent dans le répertoire `sender`.(En situation réelle, cet *Advertisement* serait découvert grâce au protocole de découverte. Pour simplifier cet exercice, on suppose que le deuxième pair dispose déjà de l'*Advertisement*). Lors de la résolution du pipe (méthode `outputPipeEvent`), le pair crée un message et l'envoie sur le pipe.

Étape 1: *Compléter le code du fichier `PipeSender.java` en vous laissant guider par les commentaires.*

Étape 2: *Lancer les deux pairs en local, sur des ports différents de la même machine (utiliser le configurateur pour le choix des ports, par exemple 9701 et 9702). Désactiver HTTP, laisser le multicast actif et ne pas indiquer de pairs de rendez-vous. Observer leur communication.*

Étape 3: *Lancer plusieurs senders et plusieurs listeners sur différents ports, toujours en local. Pour créer plusieurs senders ou listeners, recopier le répertoire sender (respectivement listener). (Ne pas oublier d'effacer le répertoire* `.jxta` *avant la recopie.) Configurer des numéros de ports différents au lancement de chaque pair. Laisser HTTP toujours désactivé et le multicast actif. Arrêter et démarrer dynamiquement les clients et les serveurs et observer le comportement.*

Étape 4: *Refaire l'expérience en désactivant le multicast et en configurant les pairs avec le pair de rendez-vous indiqué par le formateur. Observer le comportement.*

5 Troisième partie : définition d'un service

L'objectif de cet exercice est de définir un service JXTA. Il faudra programmer deux pairs : un pair va instancier le service en tant que serveur, le deuxième va l'invoquer en tant que client. Le serveur se mettra en écoute sur le canal du service. Le client enverra sur le canal du service une chaîne de caractères avec son nom. Le service consistera à afficher cette chaîne de caractères transmise par le client.

Le pair qui joue le rôle du serveur (fichier `server/Server.java`) doit tout d'abord créer un *Input Pipe* à partir du *Pipe Advertisement* (`examplepipe.adv` présent dans le répertoire `server`) afin de traiter les requêtes qui lui sont adressées. Ensuite il doit publier le service qu'il rend. Pour cela, il doit créer et publier et un *Module Spec Advertisement*. Il faut ajouter au *Module Spec Advertisement* le précédent *Pipe Advertisement* afin que les clients puissent joindre le serveur.

Le pair qui joue le rôle du client (fichier `client/Client.java`) doit boucler jusqu'à trouver le *Module Spec Advertisement* du service qu'il souhaite invoquer. Puis extraire le pipe du *Module Spec Advertisement* afin de créer un *Output Pipe* et ainsi s'adresser au serveur pour lui soumettre sa requête.

Étape 1: *Compléter le code des fichiers* `Client.java` *et* `Server.java` *en vous laissant guider par les commentaires.*

Étape 2: *Lancer les deux pairs en local, sur des ports différents de la même machine (utiliser le configurateur pour le choix du port). Désactiver HTTP et ne pas indiquer de pairs de rendez-vous.*

Étape 3: *Lancer plusieurs clients et plusieurs servers sur différents ports, toujours en local. Laisser HTTP toujours désactivé. Arrêter et démarrer dynamiquement les clients et les serveurs et observer le comportement.*

Étape 4: *Refaire l'expérience en configurant les pairs avec le pair de rendez-vous indiqué par le formateur. Observer le comportement.*

Étape 5: *(Optionnel) Modifier le service pour faire exécuter par le serveur le produit scalaire de deux vecteurs. Le service devra cette fois-ci renvoyer un résultat au client à travers un pipe que celui aura indiqué dans sa requête. Attention, les vecteurs doivent être encodés sous forme d'un chaîne de caractères (voir la documentation de la classe Base64 de l'implémentation de JXTA).*