Série 6 : RMI

Exercice 1 : Initiation à Java RMI

Lors de cet exercice, nous allons écrire une application client/serveur utilisant le package RMI de Java, faisant les mêmes applications que nous avons développées lors du TP précédent basée sur les Sockets.

Question 1 :

On vous demande d'établir une communication entre le client et le serveur qui échangent des messages sous forme de lignes de texte. Le scénario est le suivant :

- Coté Serveur :
  1. Définir une interface héritant de l'interface *java.rmi.Remote* et déclarant une méthode qui doit faire l'écho d'un message donné en paramètre.
  2. Définir une classe implémentant l'interface déjà créée et héritant de la classe *UnicastRemoteObject* qui contient les différentes méthodes autorisant l'invocation de méthodes à distance.
  3. Créer une instance de cette classe dans la méthode *main* et l'enregistrer dans l'annuaire des objets accessibles de l'extérieur.
- Coté Client :

  On vous demande d'écrire une classe Client à qui on passe en paramètre l'URL du serveur précédent et qui :
  1. Lit une ligne tapée au clavier
  2. L'envoie au serveur qui la transforme en majuscule
  3. Affiche la réponse que celui-ci envoie
  4. Reboucle en 1 et s'arrête lorsque la ligne tapée est 'END'.

Question 2 :

On vous demande maintenant de créer deux programmes client et serveur simples permettant d'établir les services suivants :

1. Le client doit appeler une méthode du serveur permettant d'afficher la date actuelle.

2. Le serveur reçoit un entier positif de la part du client, et doit signaler qu'il est premier ou non.
3. Le serveur doit afficher tout les nombres premiers inférieurs à un entier envoyé par le client.
4. Le client envoie au serveur une liste d'entiers strictement positifs : le serveur indique si ces nombres sont premiers entre eux.

Exercice2 :

Nous disposons des services implantés lors des exercices du premier TP: un qui réalise des opérations sur un tableau trié, l'autre qui permet d'identifier si un mot est un palindrome ou non et le dernier qui permet de découper une ligne de chaînes de caractère.

1. On souhaite rendre chacune de ces méthodes accessibles à distance. Donnez alors la structure des interfaces qui seront partagées par le serveur et le client sachant que chaque méthode appartient à un objet distinct.
2. Compléter le code de ces méthodes afin qu'elles puissent gérer les erreurs dues à leur appel à distance.
3. Sachant que toute méthode Java appelée par un programme Java distant doit appartenir à un objet accessible à distance. Donnez la structure des classes Java qui vont représenter respectivement les objets créés.

Exercice 3 : Gestion d'un compte bancaire

Nous disposons d'un service qui offre les opérations suivantes de gestion d'un compte bancaire *debiter()*, *crediter()* et *lireSolde()*.

1. On souhaite rendre chacune de ces méthodes accessibles à distance de manière à ce qu'elles définissent l'interface entre le client et le serveur. Ecrire cette interface.

2. Déduire la classe qui matérialise le service qui offre les opérations *debiter()*, *crediter()* et *lireSolde().*

3. Ecrire le programme du serveur qui doit permettre l'enregistrement du service auprès de *RMI Registry*.

4. Ecrire le programme du client qui doit être lancé à partir d'un autre répertoire ou d'une autre machine.

Exercice 4 : Partage d'ensembles de coordonnées 2D

On souhaite stocker sur un serveur de données un ensemble de coordonnées 2D décrivant les positions d'un certain nombre de véhicules dans le plan.

Ce serveur devra inclure les fonctionnalités suivantes:

- accès au nombre de positions
- accès au nom d'une position
- accès en lecture aux positions
- création d'une nouvelle position et nommage
- destruction d'une position
- accès en écriture aux positions

1. Ecrire l'application serveur.
2. Ecrire une application cliente ayant pour but de :
   - ✓ créer une nouvelle position.
   - ✓ détruire une position.
   - ✓ modifier une position.
   - ✓ lire et afficher toutes les positions.

Exercice 5 : Calcul parallèle

On désire concevoir une application client-serveur à même de réaliser un calcul parallèle pour la multiplication d'un ensemble de n vecteurs à 4 dimensions par une matrice 4x4.

1. Ecrire une application serveur permettant de réaliser le produit d'un vecteur par une matrice et rendant le résultat.

2. Ecrire une application serveur permettant de réaliser le produit de nv vecteurs par une matrice M et rendant les résultats.

3. Ecrire une application cliente initialisant n vecteurs et distribuant leurs produits par une matrice M sur n serveurs disponibles.