SI4 Polytech'Nice – UNS

UE « Applications Réparties », Partie « Objets » - Epreuve écrite individuelle

27 Mai 2014, durée 1h

Documents cours/TP NON autorisés. Comme d'habitude, il est conseillé de lire le sujet en entier avant de commencer !

F. Baude

Exercice 1 (barème approximatif : 8 points)

Le but de cet exercice est, grâce à une série de questions de synthèse, d'évaluer votre compréhension profonde du système Java RMI, ainsi que des autres technologies étudiées lors des séances. Quelques lignes de réponse (4 / 5 maximum) par question.

1. RMI impose que toute méthode d'une interface Remote (et donc son code d'implémentation) déclare jeter éventuellement une exception « java.rmi.RemoteException ». Quelle implication cela a-t-il sur le code qui invoque des méthodes RMI ? Il y a surement de bonnes raisons à imposer ceci, pourquoi à votre avis ?
2. RMIRegistry est lui-même une application Java RMI ; détailler ce qui en fait un serveur RMI (cad. quels sont les programmes qui sont clients de ce serveur ?). RMIRegistry est-il lui-même client d'autres objets RMI ?
3. Le téléchargement dynamique de .class est un mécanisme puissant de la plateforme Java RMI. Rappeler brièvement quels en sont les principes
4. Suite de 3. Rappeler dans quelle circonstance RMIRegistry peut être amené à utiliser ce mécanisme de téléchargement de classes
5. JNDI est une technologie que nous avons peu étudiée mais néanmoins utilisée, et qui peut être utile aussi bien en RMI qu'en JMS (et aussi en Corba d'ailleurs !): expliquer rapidement à quoi cela sert dans ces deux cas ; et du coup quelle est la valeur ajoutée si l'on s'en sert dans une application RMI ou JMS (ou Corba) plutôt qu'utiliser les services de nommage ad-hoc à chacune de ces technologies.
6. Expliquer brièvement la différence entre utiliser une queue JMS en Point-à-Point ou en Publish/Subscribe. Est-ce que cela change quelque chose du côté de celui qui produit les messages.
7. Expliquer comment avec JMS, réaliser simplement une application distribuée et concurrente, qui se base sur un modèle Producteur-Consommateur, avec la possibilité qu'il y ait P producteurs et C consommateurs, P et C >=1
8. Comment un Message-Oriented Middleware tel ActiveMQ (basé donc sur un principe de queue) fait-il pour assurer que tout message produit sera effectivement bien traité par ce(ux) qui le doivent. Et ce même dans les pires situations où n'importe quel élément participant pourrait tomber en panne.

Les 2 exercices qui suivent vont consister à travailler sur une application de « chat », dans 2 technologies différentes.

Exercice 2 (barème approximatif : 4 points) : une application « chat » en JMS

L'objectif de cet exercice est de spécifier (mais **pas** de programmer) selon le modèle JMS, une application permettant à un utilisateur de parler avec les autres utilisateurs connectés. Il doit donc avoir la possibilité d'envoyer des messages (ce seront des simples chaines de caractères), et de voir apparaitre « automatiquement » les messages que les autres utilisateurs écrivent. Les utilisateurs se connectent donc à une « chat room » dont le nom sera une chaine de caractères connue des participants.
Le fait de répondre aux questions ci-dessous va vous permettre d'esquisser les grandes lignes de votre application chat. J'attends donc en gros un pseudo code (algorithme), mais je n'exige pas les détails.

1. La chat room sera matérialisée par une queue JMS. Comment faire pour qu'un utilisateur se connecte à la room ?
2. Une fois connecté, comment chacun des utilisateurs verra arriver les messages postés dans la room ? Y'a-t-il un moyen éventuel de voir les messages qui avaient pu être échangés pendant qu'on n'était pas connecté à la chat room?
3. Décrire comment un utilisateur va poster un message dans la chat room
4. En cas de panne, y-a-t-il un risque qu'un message soit délivré plus d'une fois (et donc affiché plusieurs fois dans la chat room) ? Si oui, comment l'application pourrait déceler ce souci ? Dans ce cas, vaut-il mieux « rater » un message, que de l'afficher plusieurs fois ? Et pire, est-il possible que des messages arrivent de manière désordonnée, faisant ainsi perdre à celui qui les reçoit le fil de la conversation ?

Exercice 3 (barème approximatif : 8 points) : une application « chat » en RMI

L'objectif de cet exercice est de spécifier (et cette fois **de programmer**) selon le modèle RMI, une application permettant à un utilisateur de parler avec les autres utilisateurs connectés à la « chat room ». On supposera que cette chat room existe avant le lancement des codes des utilisateurs et que le nom qui lui est associé est connu de tous (public).

a) Un utilisateur doit donc avoir la possibilité d'envoyer des messages (le corps d'un message sera une simple chaine de caractères, … pour commencer voir plus bas), et de voir apparaitre « automatiquement » les messages que les autres utilisateurs écrivent. On supposera que les messages envoyés sont aussi reçus, en retour (en écho en quelque sorte). Ces messages seront simplement rangés dans une ArrayList sur chaque code utilisateur. Une interface graphique associée à chaque utilisateur a pour rôle de visualiser graphiquement et régulièrement le contenu de cette ArrayList. Vous ne devez pas programmer l'interface graphique ! La seule chose que vous devez spécifier et programmer à présent, ce sont les parties RMI.
b) Faites en sorte de fournir une solution qui ne présente pas de risque d'interblocage, et argumentez pourquoi c'est bien le cas.
c) Comme extension de votre programme donné en a+b) : on voudrait pouvoir utiliser une sous-classe des messages, qui soit d'un visuel plus agréable : la méthode toString de ces messages permettrait par exemple d'afficher une petite image en plus du simple texte. Expliquer comment modifier a minima le code déjà fourni afin que cette possibilité soit offerte, **mais ne soit pas obligatoire**. Dit autrement, expliquer comment profiter judicieusement du mécanisme de chargement dynamique de classes disponible en RMI, pour qu'un code utilisateur ne soit pas obligé de poster de tels messages plus sophistiqués, mais soit quand même capable d'en réceptionner et de les afficher.

d) Expliquer pourquoi vous ne risquez pas de désordonnancement dans l'ordre des messages reçus (comparé à leur ordre d'émission). Même si bien sûr, en cas de panne, vous pouvez avoir perdu des messages qui ont pu être échangés dans la room.