Cours : Web Services

DIC3-M2GL&SI/DGI/ESP/UCAD/SN

TP0: Java RMI

On veut développer un logiciel utilisant la technologie RMI. Pour cela on se propose de mettre en oeuvre un logiciel de chat. Un tel logiciel permet simultanément à plusieurs personnes de discuter ensemble. Chaque message posté par un utilisateur est immédiatement envoyé et présenté aux autres utilisateurs.

RMI : définition des interfaces distantes

La mise en oeuvre complète du logiciel de discussion en ligne, va se découper en trois parties distinctes. Premièrement, il va falloir nous entendre sur comment nos différents objets distribués vont communiquer entre eux. En second lieu, il nous faudra voir comment coder un objet distant. Enfin, il faudra voir comment utiliser un objet distant. Des éléments de la première étape sont fournis dans la suite.

Les interfaces de notre logiciel de discussion

Pour définir nos interfaces, il nous faut en premier lieu savoir de combien d'objets distants nous aurons besoin : il en découle naturellement le nombre d'interfaces distantes. Prendre quelques minutes de réflexion avant de poursuivre la lecture du document (...).

Il semble que deux types d'objets nous sont nécessaires. Il faut un objet pour représenter un client : en effet, chaque utilisateur hébergera sur sa machine un objet distant le représentant. Tous ces objets seront inscris à une salle de discussion qui prendra en charge la propagation des messages à envoyer : il s'agit donc de notre second type d'objets distants.

Il en résulte les deux interfaces distantes dont le code vous est présenté dans l'exemple suivant. Des explications sur chacune des méthodes proposées suivront.

```
import java.rmi.*;

public interface ChatRoom extends Remote {
    public void subscribe(ChatUser user, String
pseudo) throws RemoteException;
    public void unsubscribe(String pseudo) throws
RemoteException;

    public void postMessage(String pseudo, String
message) throws RemoteException;
}
```

Fichier "ChatRoom.java"

```
import java.rmi.*;

public interface ChatUser extends Remote {
    public void displayMessage(String message) throws
RemoteException;
}
```

Fichier "ChatUser.java"

Quelques explications

En fait, les choses sont très simples. L'objet central du système à concevoir est la salle de discussion (l'objet *ChatRoom*). Tous les clients se connectent initialement à cet objet pour faire partie du groupe de discussion : pour ce faire ils utilisent la méthode *subscribe*. Celle-ci prend deux paramètres : une référence distante associée au client considéré et une chaîne de caractères (le pseudo).

Le passage de ces deux derniers paramètres, lors de l'invocation de la méthode *subscribe*, se réalise de manière différente. En effet, le pseudo est un objet Java normal : il sera donc simplement sérialisé lors de l'appel de la méthode. Par contre, il en va différemment pour le paramètre *user*. Celui-ci est de type *java.rmi.Remote*, par le jeux subtile de l'héritage d'interface : il s'agit donc d'un objet distant. Il doit être clair que l'objet client ne va pas être sérialisé sur le réseau : soit quoi, au final, il ne sera plus distant. Seul sa référence distante (son *stub*) sera sérialisé.

Une fois les clients connectés à la salle de discussion, imaginons que l'un deux envoie un message. Pour ce faire, il utilisera la méthode *postMessage* de la salle de discussion. Cet objet, ayant reçu le nouveau message, se chargera de le renvoyer à tous les utilisateurs. C'est donc bien un objet de type *ChatRoom* que sera le coeur de notre système. Or, pour la salle de discussion, les clients sont bien entendu aussi des objets distants. Elle devra donc invoquer, sur chaque client la méthode *displayMessage*. Cette dernière aura pour tâche de mettre à jour, avec ce message, toutes les fenêtres des utilisateurs.

Au terme d'un certain délai, un utilisateur pourra décider de quitter la salle de discussion. Dans ce cas, il invoquera la méthode *unsubscribe* en lui fournissant son pseudo. Cela permettra à la salle de discussion de retrouver la référence distante associée et de la supprimer.