JAVA RMI

Remote Method Invocation

Contenu

- Événement de connexion/déconnexion d’un client.
- Gestion des connexions inversées (du « serveur » vers le « client »).

Événement connexions

```
private class ServListener implements IServerListener{
    public void clientConnected(Socket socket){
    System.out.println("Client connected: " + socket.getInetAddress());
    }
    public void clientDisconnected(Socket socket){
    System.out.println("Client disconnected: " +
    socket.getInetAddress());
    }
}
…

this.addServerListener(new ServListener());
```

Connexions inversées

- Le client est celui qui initie la première connexion vers le serveur.
- Le serveur est celui qui reçoit les connexions des clients.
- Souvent, on a besoin que le serveur contacte les clients de façon autonome (pas un callback).

Connexions inversées

- Lorsqu'un client se connecte au serveur, il peut donner un moyen au serveur de le contacter lorsque nécessaire.
- Observer (à distance)

Connexions inversées

1. On créé une nouvelle interface qui sera implémentée (et exécutée) du côté client et manipuler par le serveur. Disons MyServerObserver
2. Dans l'interface du serveur, on ajoute une fonction pour permettre d'ajouter un observeur sur notre serveur

Connexions inversées

3. Dans le code client, on enregistre notre interface d'écoute au module RMI

```
callHandler.registerGlobal(MyServerObserver.class, this);
```

4. Dans le code client (après 3) on enregistre notre interface d'écoute auprès du serveur

```
myServiceCaller.registerObserver(this);
```

Attention, la distinction entre Sérialiser this et copier l'objet sur le serveur VS envoyer un proxy du client au serveur pour lui permettre de nous rappeler se fait au niveau de l'enregistrement (ou non) [étape 3] auprès du service RMI avant le passage en paramètre.