Projet Java RMI Program

19 avril 2018

Implémentation des web hooks en Java RMI

Rappels

Toutes les interfaces partagés entre client et serveur doivent hériter de `Remote` et tous les objets partagés doivent implémenter ces interfaces ainsi que hériter de la classe `UnicastRemoteObject`, tout ceci afin de pouvoir s'appeler entre elles.

Pour mon architecture j'ai donc décomposé le problème en 2 modules, un qui va correspondre au client et l'autre qui va correspondre au serveur.
Je vais vous détailler comment chacun des deux fonctionne :

Le client

Le main est contenu dans la classe `LanceurClient` laquelle va s'occuper en premier lieu de récupérer l'interface `ServeurDistantInterface` laquelle est enregistré dans le registre RMI, cette dernière est définie et implémentée au niveau du module du serveur.

Grâce à la récupération de cet objet serveur, nous allons pouvoir abonner notre client à celui-ci et ainsi récupérer les messages envoyés par le serveur.

Le module Client contient l'interface `InterfaceClient`, celle-ci bénéficiera au serveur pour pouvoir envoyer des messages à l'ensemble des clients.

Le serveur

Le main est contenu dans la classe `LanceurServeur` lequel va lancer le LocateRegistry sur un port particulier, enregistrer le `ServeurDistantInterface` dans le registre RMI pour que le client puisse appeler l'instance de cet objet.

Enfin c'est la classe Serveur qui s'occupera de recevoir les abonnements des clients, et elle s'occupera d'envoyer les données aux dits clients.

Choix spécifiques

Les données sont envoyées depuis le terminal à chaque fois qu'un message est rentré, et ceci vers tous les clients connectés. Les clients reçoivent également le nombre de personnes qui ont reçu ce message, et peuvent se désabonner du serveur à tout instant.

Implémentation du protocole HTTP streaming en Java RMI

Principe du protocole HTTP streaming

Contrairement au protocole HTTP normal, le protocole HTTP streaming ne ferme pas la connexion TCP en fin de cycle d'une requête GET/POST, mais va transmettre régulièrement les nouvelles données grâce au transfert par bloc.

Avec HTTP streaming, lors de la première méthode GET pour le flux vidéo, on a une réponse contenant un en tête "Transfer-Encoding : chuncked" qui nous montre qu'un transfert de bloc a lieu entre serveur et client sans que l'on sache la taille des données échangées.

Le problème étant qu'avec Java RMI on ne peut pas laisser ouvert la connexion, elle est fermée à chaque appel de méthode, une bonne implémentation sera alors impossible.

Tentative d'implémentation

- On pourrait simuler ceci avec une méthode contenu par un objet partagé par le serveur. Cette méthode non pas un message, mais un tableau d'informations ou "octets" qui serait lu récursivement par le client pour "feinter" la manière dont procède la technique du HTTP Streaming.
- On pourrait au travers d'une méthode d'un objet distant appeler une connection TCP et déclarer les échanges en keep-alive, cela permettrait de garder la connexion active.