Sérialisation

Laboratoire n°4 Christophe Peretti Samuel Darcey

Table des matières

[1 Introduction 2](#_Toc449698341)

[2.1 XML 2](#_Toc449698342)

[2.2 JSON 4](#_Toc449698343)

[3 Conclusion 5](#_Toc449698344)

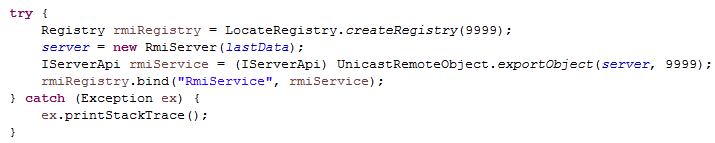
# 1 Introduction

Le laboratoire SER – Plex consistant à simuler les activités d’un complexe cinématographique, est constitué de 4 laboratoires constituant les différentes parties du projet. Le présent document porte sur le quatrième laboratoire : Echange d’informations entre serveurs et clients grâce à RMI.

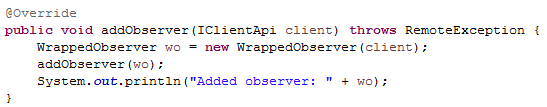
Dans ce laboratoire, il s’agissait concevoir plusieurs classes java, afin de mettre en place un serveur RMI du côté du World Film Center (WFC) afin de pouvoir mettre à jour les bases de données des différents gestionnaires de cinémas (PlexAdmin). De plus, ce PlexAdmin fait aussi office de serveur RMI afin que les différents médias (PlexMedia) puissent venir chercher les informations des différentes projection de film.

# 2 WFC

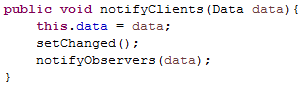
L’applicatif instancie un RmiServer, qui est une classe implémentant l’interface IServerApi (disponible dans le WFC\_Interface. On crée aussi le Registre lié à cette classe, qui va être disponible sur le réseau avec le nom « RmiService ». Pour ce serveur, le port 9999 est utilisé.



Le serveur met à disposition une classe permettant à un client de s’enregistrer comme observer.



De plus, nous avons implémenté une fonction public permettant à l’applicatif de notifier les différents clients (et donc appeler leur fonction update).

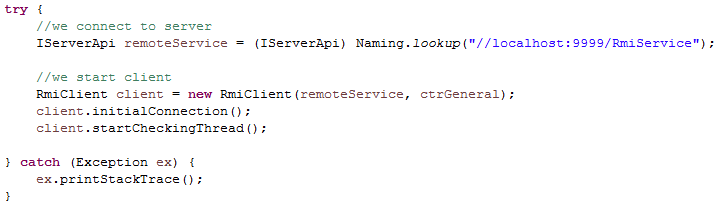


Lorsque l’utilisateur choisi une option, les dernières données sont transmises au serveur, et donc notifiées aux clients (PlexAdmin).

# 3. PlexAdmin

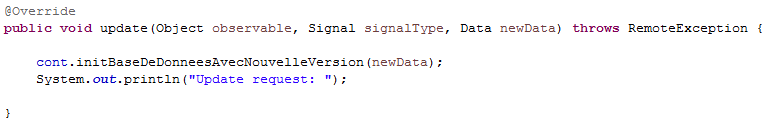
## 3.1 Client RMI

Le PlexAdmin fait office de client RMI : lorsque le serveur envoie les nouvelles données, le client doit automatiquement vider sa base de donnée de projections, et mettre à jour la liste des films avec les nouvelles données. Dans le ControlleurWFC, nous avons instancié un client RMI, après avoir récupéré le RmiService du serveur.



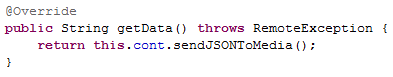
Le RmiClient reçoit en paramètre le remoteService (afin de pouvoir s’y enregistrer en tant qu’observer), ainsi que le ControleurGeneral de l’applicatif, qui va pouvoir mettre à jour les données reçues.

Ainsi, lorsque le serveur met à jour ses données, cela notifie le client, appelle donc la fonction update, qui elle appelle la fonction de mise à jour des données du client.



## 3.2 Serveur RMI

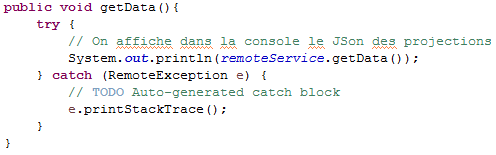
L’applicatif PlexAdmin, fait aussi office de serveur RMI. Son fonctionnement est un peu différent de celui de WFC. Ici, ce n’est pas le serveur qui force les clients de faire une action, mais il met simplement à disposition une méthode qui leur transmet la liste des projections au format JSON. Cette méthode est appelée grâce à RMI, directement par les clients eux-mêmes, lorsqu’ils en ont envie.



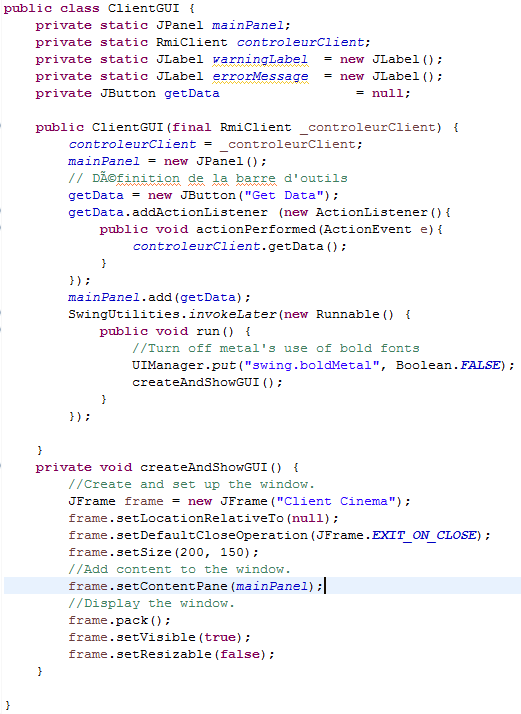
Nous avons dû modifier l’interface du serveur RMI, afin que la méthode getData() retourne un String. Cela permet d’envoyer directement le JSON à travers cette méthode. Nous avons donc aussi dû modifier de deux manières la fonction sendJSONToMedia() : Tout d’abord, celle-ci ne s’exécute plus dans un Thread à part, afin d’être sûr que le JSON à la fin de la méthode est bien finit d’être écrit dans le fichier, et ensuite, ce même JSON devient la valeur de retour de la fonction.

# 4. PlexMedia

L’applicatif PlexMedia, fait office de client RMI, qui s’enregistre en tant qu’observer auprès de PlexAdmin, tout comme celui-ci s’enregistre auprès du WFC. Pour ce média, nous avons créé une vue ClientGUI, qui ouvre une petite fenêtre JFrame, avec un bouton unique au centre. Lors que le bouton est pressé, cela appelle la fonction getData() du RmiClient, qui fait office de relai et appelle la fonction à distance getData(), du RMIServer PlexAdmin.



Et voici la vue du média :



# 5. Aperçu de l’applicatif PlexMedia

# 6 Conclusion

Grâce à ce laboratoire, nous avons eu un aperçu de la puissance de RMI. Pouvoir appeler des méthodes à distance est très pratique, et permet de s’affranchir d’une communication par Socket qui nécessite une sérialisation et désérialisation de chaque côté, parfois très couteuse est lourde à mettre en place.