Projet Données Réparties
Un service de partage d'objets répartis et dupliqués en Java

SNIDI Nabil
EL HABTI Ouassel
KARMAOUI Oussama

Département Sciences du Numérique - Deuxième année
2022-2023

Contents

1 Introduction

Dans ce rapport, nous présentons l'implémentation d'un schéma publier/s'abonner pour la gestion d'objets partagés. L'objectif de cette extension est de permettre aux clients de suivre les changements d'état d'un objet partagé sans avoir à scruter régulièrement l'objet partagé. Nous décrivons les changements apportés à l'interface du serveur pour permettre aux clients de s'abonner et de se désabonner aux changements d'état d'un objet partagé, ainsi que la méthode utilisée pour notifier les abonnés des changements d'état. Nous présentons également les choix de conception que nous avons faits pour cette extension, et argumentons nos décisions.

2 Implémentation

L'implémentation de l'extension proposée se divise en deux parties. D'abord, il s'agit d'ajouter les fonctionnalités de s'abonner et de se désabonner aux changements d'état de l'objet partagé côté serveur. Ensuite, il faut ajouter la possibilité pour le client de notifier le serveur de ses changements d'états locaux.

2.1 Abonnement et désabonnement côté serveur

Pour permettre aux clients de s'abonner et de se désabonner aux changements d'état d'un objet partagé, nous avons étendu l'interface du ServeurObject en ajoutant deux nouvelles méthodes :

- `subscribe(Client client, Callback callback)` : cette méthode permet à un client de s'abonner aux changements d'état de l'objet partagé `Shared Object`. Le client fournit un objet de rappel (`callback`) qui sera appelé par le serveur à chaque changement d'état de l'objet partagé.
- `unsubscribe(Client client, Callback callback)` : cette méthode permet à un client de se désabonner des changements d'état de l'objet partagé `Shared Object`. Le client doit fournir le même objet de rappel (`callback`) que celui fourni lors de l'abonnement.

Ces deux nouvelles méthodes sont implémentées en utilisant une liste d'abonnés par objet partagé. Cette liste est mise à jour à chaque nouvelle écriture dans l'objet partagé. À chaque mise à jour, le ServeurObject itère sur la liste d'abonnés correspondante et appelle le rappel fourni par le client.

2.2 Notification des changements d'états locaux

Pour permettre aux clients de notifier le serveur de leurs changements d'états locaux, nous avons ajouté une nouvelle méthode à l'interface du client :

- `NotifyCallBack()` : cette méthode permet au client de notifier le serveur d'un changement d'état de l'objet partagé. Le nouvel état de l'objet partagé est fourni sous forme de message transferé au autres clients.

Lorsqu'un client appelle la méthode `publish`, le serveur met à jour l'état de l'objet partagé correspondant et itère sur la liste d'abonnés pour cet objet partagé afin d'appeler le rappel fourni par chaque client abonné.(On voulais utiliser un Callback et on est en trains de formaliser ce cas d'utilisation)

3 IRC

Nous avons ajouté deux boutons (Subscribe et Unsubscribe) à l'interface de l'IRC ainsi qu'un panel pour les notifications et un `LightUpPanel` qui s'agit qu'une zone rouge qui devient vert lors de la réception d'une notification.

3.1 Vue sur l'application IRC

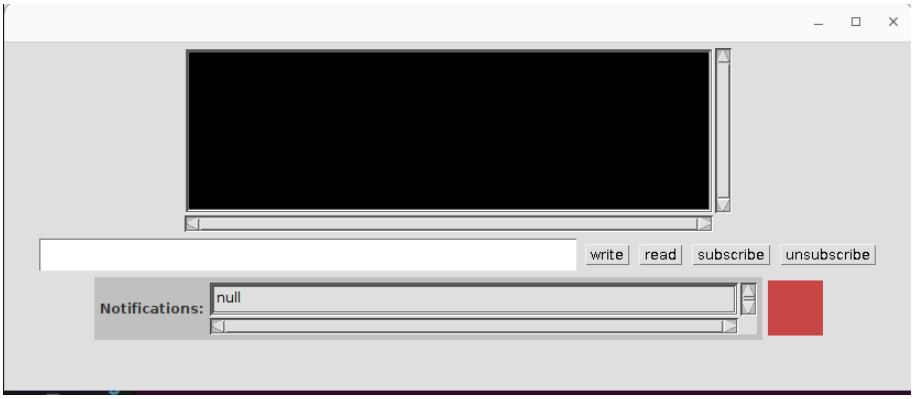


Figure 1: Vue globale sur IRC

3.2 Subscribe/ Unsubscribe

Vous Pouvez voir que lorsqu'on fait un abonnement à un fichier ça apparait sur les terminal associés qui exécutent l'application Irc . De meme pour un désabonnement.
Ceci notifie tout les clients que vous pouvez le voir sur le terminal (I'm notified)

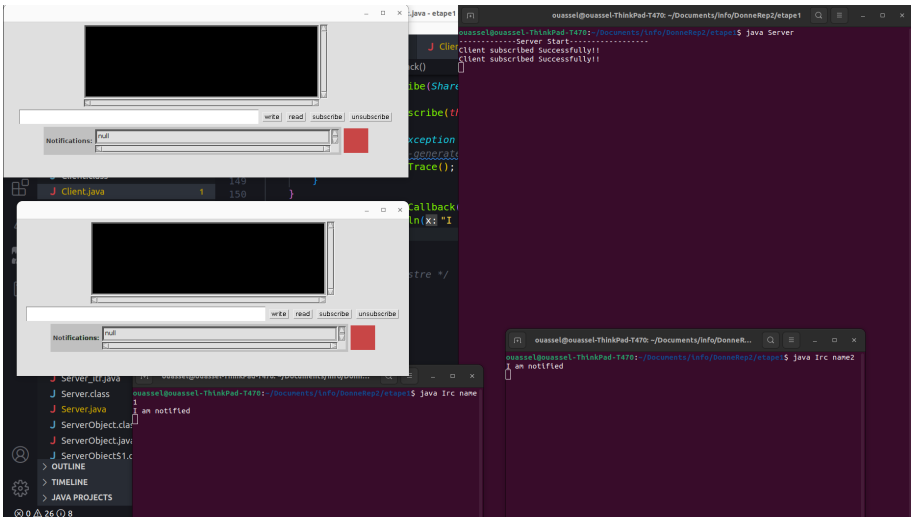
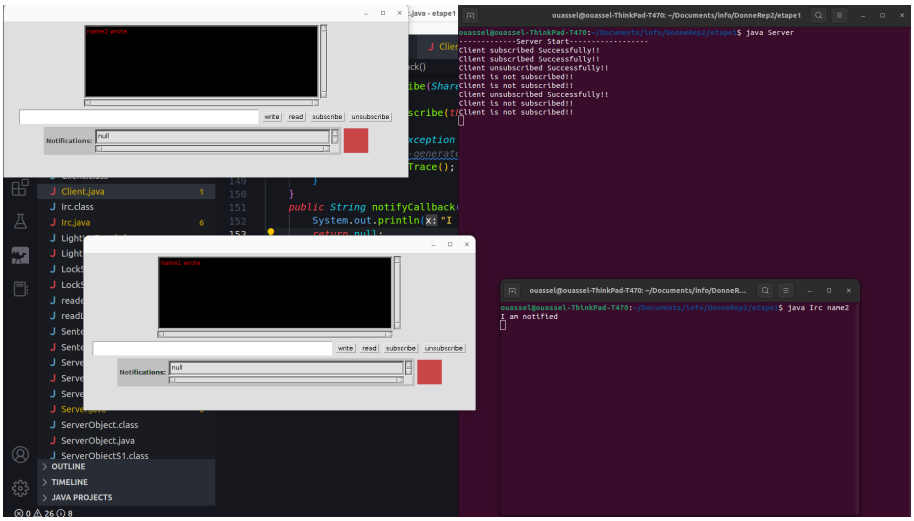


Figure 2: Subscribe—Unsubscribe

Après un enchaînement d'opérations subscribe/unsubscribe, vous pouvez voir que lorsqu'on fait subsribe et on veut le faire une autre fois ça remonte ceci avec un affichage, de meme pour unsubscribe.
De meme, lorsqu'on fait un write dans un client ça notifie les autres clients qui sont abonnés au SharedObject.



3.3 Conclusion

En étendant l'interface du serveur avec les méthodes d'abonnement et de désabonnement, ainsi qu'en ajoutant la méthode de notification des changements d'état, nous avons pu implémenter un schéma publier/s'abonner pour la gestion des objets partagés dans notre application de chat. Ce schéma permet aux clients de recevoir des notifications uniquement lorsqu'un changement d'état est détecté, améliorant ainsi l'efficacité de l'application.

En ce qui est difficultés, nous avons trouvé pas mal de problème pour notifier les autres clients abonnés à l'objet partagé lorsqu'on fait un abonnement ou un désabonnement ainsi que write par client donné, d'où le fait qu'on a fait une notification basique (I'm notified) et on va essayer d'étendre ça pour l'étape prochaine.