Projet de Données Reparties : Un service de partage d'objets répartis et dupliqués et robustes en Java Etape 1

Justin BARBIER
Tom BERTRAND
Maël MATHURIN

Département Sciences du Numérique
Deuxième année - Parcours Systèmes Logiciels
2022 - 2023

1 Ajouts par rapport au PODP étape 1

— Ajout de méthodes subscribe() et unsubscribe() dans la classe SharedObject permettant à un utilisateur de s'abonner / désabonner aux changements d'un objet.

— Ajout d'une liste contenant les Clients abonnés à un objet dans ServerObject. Lorsqu'une modification est faite sur l'objet, le ServerObject va utiliser la méthode call() du Callback sur tout les clients inscrits afin de notifier du changement. Le client va transmettre ensuite l'appel au SharedObject correspondant.

— Ajout d'une interface Callback qui doit être implémentée par l'application (ici Irc) et contenant une seule méthode call() appelée par le SharedObject quand une modification est faite sur sa valeur (quand le client appel le callback du SharedObject comme expliqué au dessus).

— Dans IRC : Ajout d'un indicateur coloré pour connaître l'état d'abonnement du Client et un compteur donnant le nombre de modification non lues apportées à l'objet une fois abonné. L'indicateur passe au orange si il y a une modification non lue. Il repasse au vert quand l'utilisateur lis l'objet et le compteur revient à 0.

2 Explication d'architecture

— Nous avons décidé de donner le choix au Client de s'abonner ou non, c'est un choix arbitraire, nous aurions pu l'abonner par défaut dès qu'il essaye de faire une lecture ou une écriture.

— Avoir une liste de Client abonnés stockée dans le ServerObject nous permet d'avoir un lien direct avec les Client concernés afin de faire appel au Callback qui lui est connu par le SharedObject correspondant. Le Callback est d'abord appelé par le ServerObject et propagé jusqu'au SharedObject. Cette implémentation nous semblait plus intuitive. Initialement nous avions prévu de mettre une liste dans le Server associant des identifiants à des listes de Clients mais cela rendait la méthode du Callback plus complexe à comprendre.

— Dans IRC : Les ajouts à l'interface graphique sont également arbitraires, nous avons choisi de combiner un indicateur coloré et un compteur afin d'avoir des informations suffisantes sur les changements apportés à l'objet.

— Le choix d'une interface Callback permet de donner une implantation adaptée à l'application qui l'utilise tout en donnant des restrictions sur la méthode à implanter (nom, paramètres, permissions et accès).

— Nous avons choisi de ne pas rendre la fonction unlock() bloquante car nous pensons qu'une fois qu'un écrivain a fini d'écrire (ou un lecteur a fini de lire), il n'a pas à attendre que tout le monde soit notifié pour pouvoir continuer à s'exécuter.

3 Démonstration

Dans cette section, nous présenterons des captures d'écran de notre démonstration accompagnées d'explications

FIGURE 1 – Situation initiale des 3 clients

Dans cette démonstration, nous utiliserons 3 clients lancés simultanément.

FIGURE 2 – 2 clients abonnés

Premièrement, les clients 1 et 2 s'abonnent à l'objet partagé. L'indicateur devient vert car ils sont à jour ou il n'y a pas eu de changement sur l'objet depuis qu'ils se sont abonnés.

FIGURE 3 – Client 1 écrit dans l'objet

Ensuite, Client 1 écrit dans l'objet, son indicateur reste vert puisque c'est lui qui a écrit et son compteur de modifications non lue reste à 0. Par contre, l'indicateur de Client 2 passe à l'orange pour lui signifier que des modifications on été apportées et son compteur passe à 1 pour lui signifier que l'objet a été modifié 1 fois depuis la dernière lecture/écriture par Client 2 ou depuis l'abonnement en l'occurrence. Client 3 n'est pas affecté puisqu'il n'est pas abonné.

FIGURE 4 – Client 2 lit l'objet

Client 2 effectue une lecture, ce qui lui permet d'afficher la dernière valeur de l'objet et change l'indicateur au vert, le compteur passe à 0.

FIGURE 5 – 3 écritures non lues

Ici, Client 1 écrit 3 fois consécutivement dans l'objet (on a effectué des lectures dans Client 1 pour que cela soit visible). On remarque que l'indicateur de Client 2 passe bien à l'orange et son compteur augmente de 3. Client 3 n'est pas affecté.

FIGURE 6 – Client 3 effectue une lecture

On constate que Client 3 est bien toujours fonctionnel même s'il n'est pas abonné.

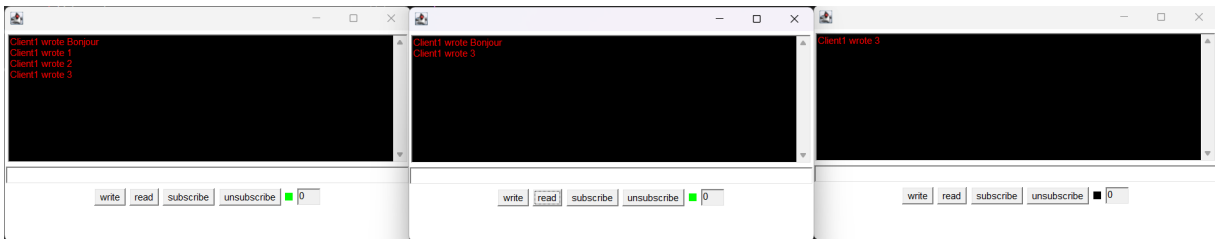

FIGURE 7 – Lecture par Client 2

Client 2 effectue une lecture, l'indicateur coloré et le compteur sont mis à jour.

FIGURE 8 – Client 3 écrit

Client 3 écrit dans l'objet, l'action est bien détectée par Client 1 et 2.

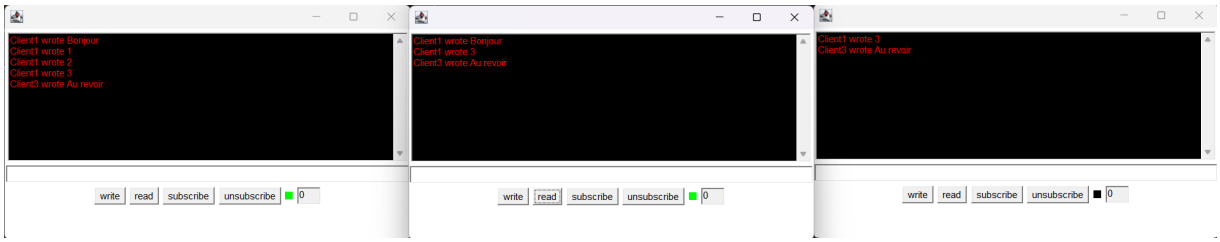

FIGURE 9 – Lectures par les clients

L'objet a bien enregistré les modifications apportées par 3, les indicateurs et compteurs sont mis à jour, ce qui termine notre démonstration.