Projet de Systèmes Concurrents

Jorge Gutierrez & Grégoire Martini

25 Janvier 2016

Table des matières

1 Etape 1

On doit implanter le service de gestion d'objets partagés répartis. Dans cette première version, les SharedObject sont utilisés explicitement par les applications.

Plusieurs applications peuvent accéder de façon concurrente au même objet, ce qui nécessite de mettre en œuvre un schéma de synchronisation globalement cohérent pour le service que l'on implante. On suppose que chaque application voulant utiliser un objet en récupère une référence (un SharedObject) en utilisant le serveur de nom. On ne gère pas le stockage de référence à des objets partagés dans des objets partagés pour l'instant.

1.1 Coté client

Classe Client

Classe SharedObject

1.2 Coté serveur

Classe Server

Classe ServerObject

2 Etape 2

3 Etape 3

On doit implanter un générateur de stub, destiné à soulager le programmeur de l'utilisation explicite des SharedObject.

Le générateur de stub est la classe StubGenerator.
La classe métier et l'interface de l'objet à partagé doivent être fournis par le programmeur.

Le stub étends la classe SharedObject et implémente l'interface de l'objet à partager ainsi que la classe Serializable. Il contient un constructeur qui est celui de SharedObject et l'ensemble des méthodes publiques de la classe métier. Il gère aussi les annotations @Read and @Write qui permettent de soulager le programmeur de la gestion explicite des verrous sur l'objet partagé dans son application. Il lui suffit d'annoter les méthodes de la classe métier selon qu'elles modifient (@Write) ou utilisent simplement l'état (@Read) de l'objet.

On utilise la classe File qui permet de créer un nouveau fichier et la classe FileWriter qui permet d'écrire dans un fichier. Le code générer est stocké dans un StringBuffer avant d'être écrit dans le fichier.
Toute l'architecture du stub est écrite dans des chaines de caractères statique et seules les parties dépendantes de la classe dont on génère le stub telles que le nom des méthodes ou des paramètres est récupérés au moment opportun.
On utilise l'introspection de Java pour accéder à la classe métier. Attention, l'utilisation de la méthode getParameter pour récupérer les paramètres d'une méthode nécessite l'utilisation de Java 8.

Lorsque le client à besoin du stub d'un objet, il utilise la méthode getStub en passant la classe de l'objet dont il veut le stub en paramètre et la méthode lui renvoi la classe du stub. Le code du stub est ainsi généré et compilé à la volée et instancié par le client de façon transparente pour le programmeur.

4 Etape 4