Projet de Systèmes Concurrents

Jorge Gutierrez & Grégoire Martini

25 Janvier 2016

Table des matières

1 Etape 1

2 Etape 2

3 Etape 3

On doit implanter un générateur de stub, destiné à soulager le programmeur de l'utilisation explicite des SharedObject. Nous prenons les hypothèses suivantes :

un objet partagé est une classe sérialisable (par exemple Sentence). Il s'agit de la classe métier de l'objet partagé.

l'objet partagé est utilisable à partir de variables de type une interface (par convention, l'interface pour utiliser un objet partagé de classe Sentence s'appellera Sentenceitf) qui inclut les méthodes métier de Sentence auquel on ajoute les méthodes de verrouillage en héritant de SharedObjectitf. Mais Sentence n'implémente pas Sentenceitf, car la classe métier ne définit pas les méthodes de verrouillage (c'est le stub qui le fait).

un stub est généré, appelé Sentencestub. Ce stub hérite de SharedObject (donc des méthodes de verrouillage) et il implémente l'interface Sentenceitf. Avec ces hypothèses, les interfaces de la classe Client (notamment pour le serveur de nom) restent les mêmes. Et pour utiliser un objet partagé de la classe Sentence, on fera : SentenceItf s = (SentenceItf)Client.lookup ("MySentence") ; s.lockread() ; s.meth() ; s.unlock() ; On pourra également étudier l'annotation des méthodes dans les interfaces afin de leur associer un mode de verrouillage (@read ou @write). Ainsi, le verrouillage de l'objet partagé est réalisé par le stub généré et on obtient la séquence d'appel suivante : SentenceItf s = (SentenceItf)Client.lookup ("MySentence") ; s.meth() ;

4 Etape 4