TD 4. programmation objet répartie : CORBA)
durée 3h
M1 Formation Initiale

Melliti Tarek

14 Mai 2007

Exercice - 1 *Prise en main : un serveur d'horloge*

---

nous allons implanter une horloge qui sera accessible à distance. L'interface IDL de l'horloge que nous vous proposons d'implanter est la suivante :

```
module exo1corba {
interface Horloge {
readonly attribute string localite;
string getTime( );
string getTimeZone( );
};};
```

Commencer par générer les fichiers d'amorce pour le serveur et le client de la façon suivante : *idlj -fall Horloge.idl*

Cette commande place les fichiers générés dans paquetage générés du module *td*1. Ainsi, ***HorlogePOA.java*** sera placé dans le répertoire *./td*1/.
1- Vérifiez que le fichier ***HorlogePOA.java*** a bien été généré et implantez l'objet servant (***td1.HorlogeImpl***) en héritant de cette classe. Dans le champ localite, placez le nom de la machine locale (utilisé la classe ***java.net.InetAdress***). Pour manipuler les heures utilisez la classe ***GregorianCalendar*** du paquetage **java.util** (consultez la doc).
2- Écrivez une classe ***td1.HorlogeServeur*** qui effectue les étapes suivante suivantes :
– initialisation de l'ORB :
ORB.init(...) ;
– récupération de la référence de l'adaptateur d'objets racine :
POAHelper.narrow(orb.resolve initial references("RootPOA")) ;
– instanciation de l'objet servant horloge :
HorlogeImpl obj=new HorlogeImpl() ;
– enregistrement et activation du servant par l'adaptateur d'objets :
rootPOA.activate_ object(obj) ;
– récupération du context racine (l'objet Context) du serveur de nom.
NamingContextExt c=NamingContextExtHelper.narrow(orb.resolve_ initial_ references("NameService")) ;
– génération d'un nom pour l'objet et le relier au context racine
c.bind(c.to_ name("horloge"),obj) ;
– activation de l'adaptateur d'objet :
rootPOA.the_ POAManager().activate() ;
– mise en attente de requêtes de l'ORB :
orb.run().
Compilez votre code puis
– lancez votre serveur de nom *tnameserv*
– executez votre serveur
3- Écrivez une classe ***HorlogeClient*** qui localise l'objet horloge de votre voisin et qui l'invoque.

Exercice - 2 *Un forum de discussion*

L'objectif de cet exercice est de développer une application de forum interrogeable à distance, permettant de poster des messages ou encore faire des commentaire sur des messages déjà postés. voici le fichier IDL de l'application :

```
module exo2corba {
exception NoSuchCommentFound {string message;string val;};
exception AlreadyExist {string message;}; struct Comment{ string author; string date; string body; };
typedef sequence<Comment> comments;
interface Message {
readonly attribute string title;
readonly attribute string author;
readonly attribute string date;
readonly attribute string body;
readonly attribute comments rep;
boolean addComment(in Comment m) raises(AlreadyExist);
boolean removeComment(in string author) raises(NoSuchCommentFound) };
exception NoSuchMessageFound { string message; string val;};
interface Forum {
readonly attribute string theme;
readonly attribute string moderator;
boolean postMessage(inout Message m) raises(AlreadyExist)
Message getMessage(in string title) raises(NoSuchMessageFound)
boolean removeMessage(in string title) raises(NoSuchMessageFound)
};};
```

Un objet forum est déployé pour chaque thème. Les objets forum sont également publié sur le serveur de nom sous le contexte "FORUM" et comme alias le thème. Les clients peuvent alors déposer des messages. Chaque dépositaire de message reste propriétaire de son message (ce qui implique que le Message une fois déposé devient lui même objet distant). Les autres client peuvent interroger l'objet forum pour récupérer la référence vers un objet message et peuvent y rajouter des commentaires.

1- Compilez le fichier IDL. Ouvrez le fichier "***ForumOperations***" que constatez vous à propos de la signature de la méthode *postMessage(...)*. Expliquez cette différence par rapport à l'IDL et rappeler le role des fichier ***...Holder*** générés.

2- Implémentez les objets servant de ***Forum*** et ***Message*** (***ForumImpl*** et ***MessageImpl***)

3- Écrivez une classe ***ForumServer*** qui déploie un objet **ForumImpl** dont le thème est "CORBA" et le moderator "Melliti".

4- Écrivez une classe ***ForumClient1*** qui poste un message dont le corps est : " c'est quoi IDL ? "

5- Écrivez une classe ***ForumClient2*** qui récupère le Message posté et rajoute un commentaire don le corps est une réponse à la question du message

6- Écrivez une classe ***ForumClient3*** qui récupère le Message posté et qui rajoute un autre commentaire avant d'afficher le contenue de discution comme suit :

Message deposé par Tarek le Lundi sur CORBA :
c'est quoi IDL ?
————————**REPONSES**————————
comentaire de Melliti le Lundi :
c'est un system de typage abstrait pour les objets
corba implmentés dans des languages différents
————————————————————————-
comentaire de LOL le Lundi : *non je ne suis pas d'accord ... ;-)*
————————————————————————