TDM – Remote Method Invocation en Java[1]

O. Aktouf

1. Rappels sur RMI

L'appel de méthode distante (Remote Method Invocation ou RMI) permet à un objet O1 de faire appel aux services d'un objet *distant* O2, en invoquant les méthodes publiques de l'objet O2. L'objet O2 est situé sur une JVM différente, sur la même machine que l'objet O1 ou sur une machine distante accessible à travers le réseau. On dit que l'objet O2 est le *serveur* alors que l'objet O1 est le *client*.

La figure 1 ci-dessous rappelle ce fonctionnement de base.

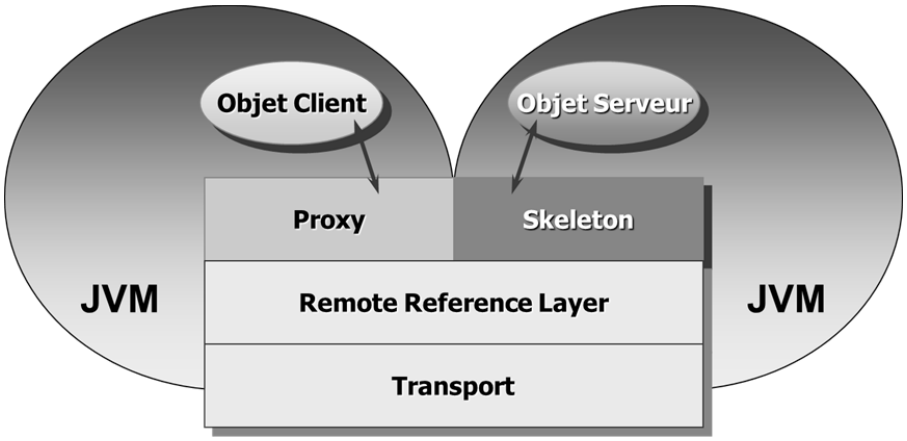

Fig. 1. Architecture de RMI [D. Hagimont]

[1] Ce TDM est inspiré du tutoriel d'Oracle sur le protocole RMI.

Objectif

Dans ce TDM, vous allez programmer votre première application utilisant le protocole RMI. L'objectif est de permettre à un objet client de demander à un objet serveur d'afficher le message « `Hello World!` ».

Pour cela, vous devez suivre les étapes décrites ci-dessous.

2. Ecriture du Serveur

2.1. L'interface du serveur

a. Un objet distant est « vu » par les autres objets à travers son interface. L'interface d'un objet distant doit étendre une interface de la bibliothèque Java. Quelle est cette interface ?
b. Quelle est l'exception spécifique à un appel distant qui doit être levée par toute méthode publique de l'interface distante ?
c. Ecrire l'interface de l'objet serveur. Cette interface sera nommée `Hello` et sera contenue dans un fichier `Hello.java`. Cette interface fournit une seule méthode publique :

```
public String sayHello() ;
```

La méthode `sayHello()` devra afficher le message suivant « `Hello World!` ». Attention, son implémentation sera écrite dans le serveur (voir ci-dessous).

2.2. L'implémentation du serveur

L'objet serveur doit implémenter l'interface distante et étendre `UnicastRemoteObject`.

a. Ecrire la classe `Server` qui implémente l'interface `Hello` et notamment la méthode `sayHello()`.

b. Instancier un objet serveur de type `Server` et l'enregistrer dans le `rmiregistry` en utilisant la méthode `rebind` et le nom `Hello`.

c. Compléter le code du serveur pour lui permettre, une fois lancé, d'afficher le message « `Server ready!` » sur la console.

3. Ecriture du Client

La classe `Client` doit effectuer les opérations suivantes :

- Demande la référence à l'objet serveur (stub) en évoquant la méthode `looklup` auprès du `rmiregistry` ;
- Appelle à distance la méthode `sayHello()`.

4. Compilation et exécution

Compiler et exécuter les classes `Server` et `Client`. Le message « `Hello world!` » devrait s'afficher sur la console.

5. Affichage en plusieurs langues

Modifier la méthode `sayHello()` en lui rajoutant un paramètre de type `int` indiquant la langue du message. Ce paramètre peut avoir les valeurs suivantes : `Fr`, `En`, `Sp` pour français, anglais et espagnol respectivement. Le message « `Hello World!` » sera alors affiché dans la langue choisie. La langue par défaut sera l'anglais.