TP : Implémentation d'un service RMI en Java

Ce TP consiste à créer une application Java permettant à un client d'invoquer une méthode à distance sur un serveur. Nous allons suivre les étapes standards de RMI.

1. Objectif du TP

- Comprendre le fonctionnement de RMI (Remote Method Invocation).
- Implémenter un service distribué simple (exemple : un service de calcul d'âge).
- Manipuler le registre RMI (rmiregistry).
- Déployer un client et un serveur interagissant via RMI.

2. Étapes du développement

Nous allons réaliser les étapes suivantes :

1. **Définir une interface distante**
2. **Implémenter l'interface distante dans une classe serveur**
3. **Générer le Stub (client) et Skeleton (serveur)**
4. **Créer un serveur RMI**
5. **Créer un client RMI**
6. **Lancer le registre RMI**
7. **Exécuter le serveur et le client**

3. Développement du service RMI

Étape 1 : Définir l'interface distante

L'interface distante déclare les méthodes accessibles à distance. Elle doit :

- Étendre java.rmi.Remote.

- Déclarer que ses méthodes peuvent lever une exception RemoteException.

```
package rmi;

import java.rmi.Remote;

import java.rmi.RemoteException;


public interface CalculAge extends Remote {
   int calculerAge(int anneeNaissance) throws RemoteException;
}
```

Étape 2 : Implémentation de l'interface distante

Le serveur implémente cette interface et étend UnicastRemoteObject.

```
package rmi;

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

import java.util.Calendar;


public class CalculAgeImpl extends UnicastRemoteObject implements
CalculAge {


   protected CalculAgeImpl() throws RemoteException {
      super();
   }


   @Override
   public int calculerAge(int anneeNaissance) throws RemoteException {
      int anneeActuelle = Calendar.getInstance().get(Calendar.YEAR);
```

```
        return anneeActuelle - anneeNaissance;
    }
}
```

Étape 3 : Générer le Stub

On génère le stub et le squelette avec l'outil rmic.

rmic rmi.CalculAgeImpl

Étape 4 : Création du serveur RMI

Le serveur RMI :

- Crée une instance de l'implémentation distante.
- L'enregistre dans le registre RMI.

```
package rmi;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.Naming;

public class Serveur {
    public static void main(String[] args) {
        try {
            // Création de l'instance distante
            CalculAgeImpl obj = new CalculAgeImpl();

            // Démarrage du registre RMI (si non lancé séparément)
            LocateRegistry.createRegistry(1099);
```

```
            // Enregistrement de l'objet dans le registre RMI
            Naming.rebind("rmi://localhost/CalculAgeService", obj);

            System.out.println("Serveur prêt...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Étape 5 : Création du client RMI

Le client RMI :

- Se connecte au registre RMI.
- Récupère une référence de l'objet distant.
- Appelle une méthode distante.

```
package rmi;
import java.rmi.Naming;

public class Client {
    public static void main(String[] args) {
        try {
            // Recherche de l'objet distant
            CalculAge stub = (CalculAge)
Naming.lookup("rmi://localhost/CalculAgeService");

            // Appel de la méthode distante
```

```
            int age = stub.calculerAge(2000);
            System.out.println("Âge calculé : " + age + " ans");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4. Exécution du TP

Étape 6 : Lancer le registre RMI

Avant d'exécuter les programmes, il faut démarrer le registre RMI avec :

rmiregistry 1099

Étape 7 : Démarrer le serveur

Dans un terminal :

java rmi.Serveur

Étape 8 : Exécuter le client

Dans un autre terminal :

java rmi.Client

5. Explication des éléments de code

1. **java.rmi.Remote** : Interface mère des objets distants.
2. **UnicastRemoteObject** : Permet d'exporter un objet distant.
3. **RemoteException** : Indique un problème réseau possible.

4. **Naming.lookup()** : Permet de récupérer un objet distant via son URL.
5. **LocateRegistry.createRegistry(1099)** : Démarre un registre RMI local.
6. **rmiregistry** : Programme permettant d'associer un nom à un objet distant.