Java RMI et révisions

Benoît Barbot

Département informatique, Université Paris-Est Créteil, M1

Mardi 29 mars 2016, Cours 7 : Gestion des Threads de haut niveau

Plan

Java Remote Methode Invocation(RMI)

Rappel Executor/Thread

- Lancement d'une tache sur un autre thread
- Mémoire et code à exécuter partagé

Java Remote Methode Invocation(RMI)

Rappel Executor/Thread

- Lancement d'une tache sur un autre thread
- Mémoire et code à exécuter partagé

RMI

- Exécuter du code sur une machine distante
- Mémoire et code séparé
  $\Rightarrow$ Besoin d'échanger des objets et leurs classes

Principe

Côté serveur

- Construis et déclare des objets distants (Remote Object).
- Fournit une interface pour les objets distants.
- Construit un objet proxy de l'objet distant (stub).

Principe

Côté serveur

- Construis et déclare des objets distants (Remote Object).
- Fournit une interface pour les objets distants.
- Construit un objet proxy de l'objet distant (stub).

Côté client

1. Fais une requête pour un objet distant du serveur.
2. Reçoit un objet proxy.
3. Appelle les méthodes de l'objet distant en sérialisant ses arguments.
4. Reçoit la sérialisation des résultats des appels distants (possiblement un objet distant).

Principe

Côté serveur

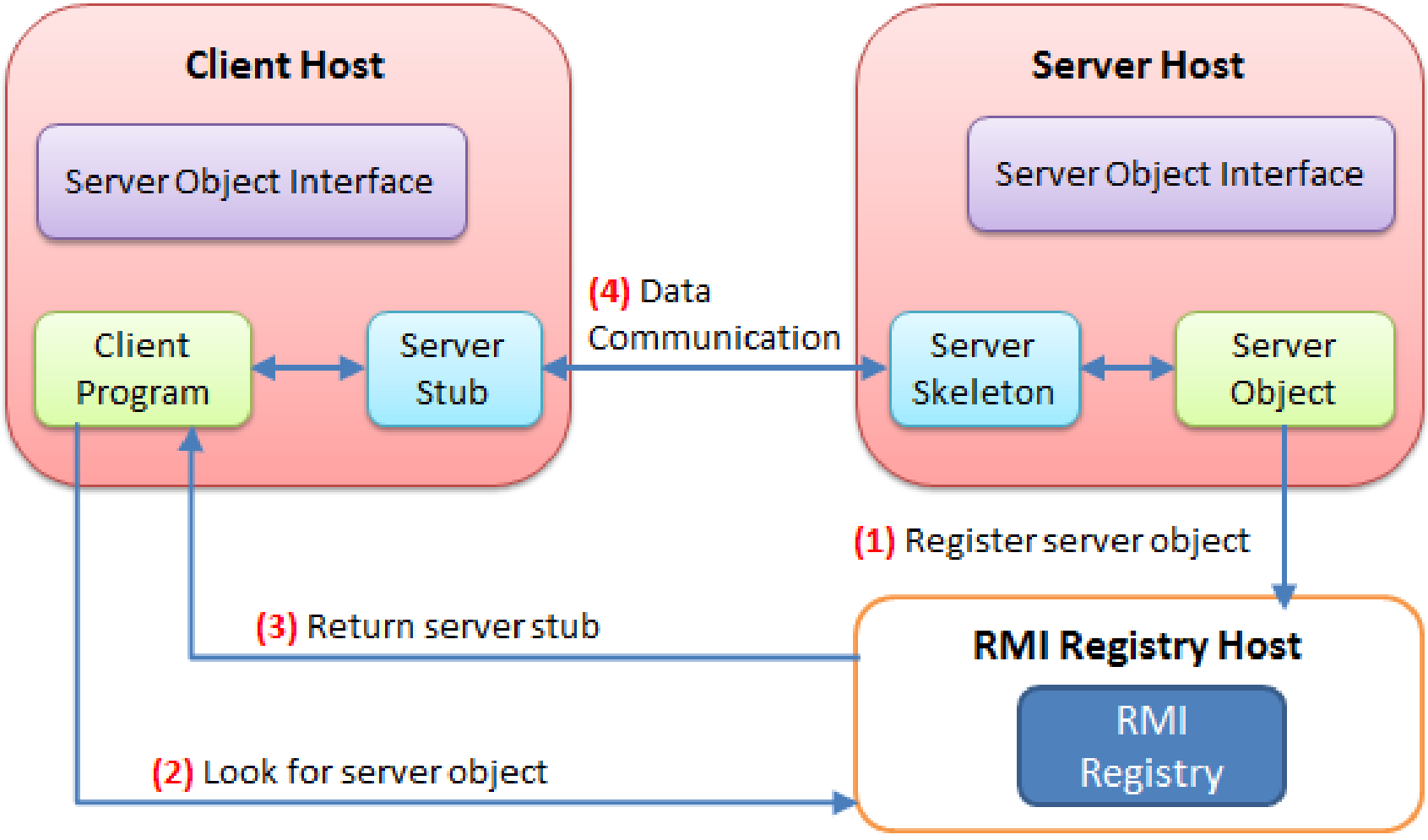
- Construis et déclare des objets distants (Remote Object).
- Fournit une interface pour les objets distants.
- Construit un objet proxy de l'objet distant (stub).

Côté client

1. Fais une requête pour un objet distant du serveur.
2. Reçoit un objet proxy.
3. Appelle les méthodes de l'objet distant en sérialisant ses arguments.
4. Reçoit la sérialisation des résultats des appels distants (possiblement un objet distant).

Registre

- Besoin d'un annuaire des objets partagé par le serveur
- *rmiregistry* lance un serveur avec cet annuaire
- Le serveur enregistre ses objets dans l'annuaire


Client Host
Server Object Interface
Client Program
Server Stub
(4) Data Communication
Server Host
Server Object Interface
Server Skeleton
Server Object
(1) Register server object
(3) Return server stub
RMI Registry Host
RMI Registry
(2) Look for server object

Interface d'objet distant

Spécifications

- Hérite de l'interface Remote
- Toutes les méthodes lèvent RemoteException
- Arguments et valeur de retour implémentent Serializable
- Si pas respecté $\Rightarrow$ java.rmi.server.ExportException : remote object implements illegal remote interface ;

Interface d'objet distant

Spécifications

- Hérite de l'interface Remote
- Toutes les méthodes lèvent RemoteException
- Arguments et valeur de retour implémentent Serializable
- Si pas respecté ⇒ java.rmi.server.ExportException : remote object implements illegal remote interface ;

Exemple "Hello world"

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Welcomer extends Remote {
  String hi(String s) throws RemoteException ;
}
```

Objet distant, Server

Spécifications

- Implémente une interface d'objet distant
- Dans le main
  1. Créé l'objet
  2. Construis un serveur servant le proxy vers l'objet distant avec : UnicastRemoteObject.exportObject(/* *Objet* */, /* *port* */)
  3. Enregistre le proxy dans un registre

Exemple “Hello world”xs

```
public class Server implements Welcomer {
  @Override
  public String hi(String s)
      throws RemoteException{
    return "Bonjour␣"+ s + "!";
  }

  public static void main(String[] args)
      throws RemoteException {
    Welcomer w = new Server();
    Welcomer stub = (Welcomer)
      UnicastRemoteObject.exportObject(w, 0);
    Registry registry=LocateRegistry.getRegistry();
    registry.rebind("Welcome", stub);
    System.out.println("Welcomer␣bound");
  }
}
```

Exemple “Hello world”xs

```
public class Server implements Welcomer {
  @Override
  public String hi(String s)
      throws RemoteException{
    return "Bonjour␣"+ s + "!";
  }

  public static void main(String[] args)
      throws RemoteException {
    Welcomer w = new Server();
    Welcomer stub = (Welcomer)
      UnicastRemoteObject.exportObject(w, 0);
    Registry registry=LocateRegistry.getRegistry();
    registry.rebind("Welcome", stub);
    System.out.println("Welcomer␣bound");
  }
}
```

Demo + Ou sont les classes ?

Client

Spécification

- Récupère le proxy grâce à un registre.
- Utilise le proxy comme un objet local.

Client

Spécification

- Récupère le proxy grâce à un registre.
- Utilise le proxy comme un objet local.

Exemple Hello world

```
public class Client {
  public static void main(String args[])
      throws RemoteException, NotBoundException {
    String name = "Welcome";
    Registry registry =
      LocateRegistry.getRegistry("localhost");
    Welcomer welcomer =
      (Welcomer) registry.lookup(name);
    System.out.println(welcomer.hi("UPEC"));
  }
}
```

Motivation

Problème

Le serveur doit connaître toutes les classe qu'il manipule :

- Client défini B héritant de A
- Serveur à une méthode f : A $\rightarrow$ C
- f(B) $\Rightarrow$ Erreurs

Motivation

Problème

Le serveur doit connaître toutes les classe qu'il manipule :

- Client défini B héritant de A
- Serveur à une méthode f : A $\rightarrow$ C
- f(B) $\Rightarrow$ Erreurs

Chargement dynamique des classes

- Le client met à disposition ses classes
- Le serveur télécharge les classes qu'il ne connaît pas

$\Rightarrow$ Problème de sécurité !

Motivation

Problème

Le serveur doit connaître toutes les classe qu'il manipule :

- Client défini B héritant de A
- Serveur à une méthode f : A $\rightarrow$ C
- f(B) $\Rightarrow$ Erreurs

Chargement dynamique des classes

- Le client met à disposition ses classes
- Le serveur télécharge les classes qu'il ne connaît pas

$\Rightarrow$ Problème de sécurité !

Demo Pb

Security Manager

Propriété

- Limite ce qu'un programme java peut exécuter
- Lève une SecurityException si une méthode viole une règle.
- Règle : accès fichier, accès réseau, accès à la réflexion, à l'interface graphique ...
- Règle spécifiée dans un fichier .policy de politique de sécurité

Security Manager

Propriété

- Limite ce qu'un programme java peut exécuter
- Lève une SecurityException si une méthode viole une règle.
- Règle : accès fichier, accès réseau, accès à la réflexion, à l'interface graphique ...
- Règle spécifiée dans un fichier .policy de politique de sécurité

Création d'un nouveau Security Manager

```
if (System.getSecurityManager() == null) {
  System.setSecurityManager(new SecurityManager());
}
```

Security Manager

Propriété

- Limite ce qu'un programme java peut exécuter
- Lève une SecurityException si une méthode viole une règle.
- Règle : accès fichier, accès réseau, accès à la réflexion, à l'interface graphique ...
- Règle spécifiée dans un fichier .policy de politique de sécurité

Création d'un nouveau Security Manager

```
if (System.getSecurityManager() == null) {
  System.setSecurityManager(new SecurityManager());
}
```

Fichier de politique de sécurité

Spécifié au lancement par : −Djava.security.policy={fichier}.policy

Mettre les classes à disposition du serveur

Serveur Web

- Un serveur web contient toutes les classes
- Le serveur les télécharge au besoin
- Le serveur contient une liste d'URL ou aller chercher des classes : −Djava.rmi.server.codebase= URL

ByteBuffer et BufferUnderflowException

- Segmentation des messages TCP
- Méthode avec taille de serialisation
- Méthode avec rattrapage d'exceptions