RMI

mardi 15 octobre 2024 15:25

Utilisé que par le langage java

Motivations
- Tirer parti des bonnes propriétés de l'orientation objet (encapsulation, modularité, réutilisation, polymorphisme, composition)
- Objet : unité de désignation et de distribution

Objets "langage"
- Représentation propre au langage : instance d'une classe
- Ex : java RMI

Objets "système"
- Représentation "arbitraire" définie par l'environnement d'exécution
- Interopérabilité entre objets écrits dans des langages différents
- Exemple : CORBA / gRPC

Éléments d'une "invocation"
- Référence d'objet ("pointeur" universel)
- Identification d'une méthode
- Paramètres d'appel et de retour (y compris signal d'exception) => passage par valeur : types élémentaires et types construits

Java RMI :
- Gère tous ces détails automatiquement
- Fonctionne sur le modèle client-serveur
- Étend aux objets le principe de RPC

Caractéristiques étendues :
- Invoquer une méthode d'un objet se trouvant sur une autre machine (objet distant: OD) exactement comme s'il se trouvait sur la même machine
  objetDistant.methode();
- Utiliser un OD sans savoir où il se trouve en demandant à un service "dédié" de renvoyer son adresse
  objetDistant = ServiceDeNoms.recherche("monObjet");
- Pouvoir passer un OD en paramètre d'appel à une méthode locale ou distante
  resultat = objetLocal.methode(oobjetDistant);
  resultat = objetDistant.methode(autreObjetDistant);
- Pouvoir récupérer le résultat d'un appel distant sous forme d'un nouvel objet qui aurait été créé sur la machine distante
  NouvelObjetDistant = ObjetDistant.methode();

Quand le client fait appel à une méthode il fait appel au stub qui représente l'objet distant …

Principe :
Mécanisme permettant l'appel de méthodes entre objets Java s'exécutant sur des machines virtuelles différentes (espaces d'adressage distincts), sur le même ordinateur ou sur des ordinateurs distants reliés par un réseau
- Utilise directement les sockets
- Code ses échanges avec un protocole propriétaire RMP (Remote Method Protocol)

Objectifs :

- Rendre transparent l'accès à des objets distribués sur un réseau
- Faciliter la mise en œuvre et l'utilisation d'objets distants java
- Préserver la sécurité (inhérent à l'environnement Java)
    - RMISecurityManager
    - Distributed Garbage Collector (DGC)

Objet distant (objet serveur) :
- Ses méthodes sont invoquées depuis une autre JVM
    - Dans un processus différent (même machine)
    - Dans une machine distante (via réseau)
- Son comportement (ensemble des méthodes de l'objet) est décrit par une ou plusieurs interface distante
- Interface au sens Java du terme (ces interfaces correspondent au contrat)
    - Déclare les méthodes distantes utilisables par le client
- Se manipule comme un objet local

Invocation distante (RMI) :
Action d'invoquer une méthode d'une interface distante d'un objet distant (même syntaxe qu'une invocation sur un objet local)

Mode opératoire :
Côté serveur :
1. L'objet serveur s'enregistre auprès du Naming de sa JVM (méthode rebind)
2. L'objet skeleton est créé : celui-ci crée le port de communication et maintient une référence vers l'objet serveur
3. Le Naming enregistre l'objet serveur et le port de communication utilisé auprès du serveur de noms

L'objet serveur est prêt à répondre à des requêtes

Côté client :
4. l'objet client fait appel au Naming pour localiser l'objet serveur (méthode lookup)
5. Le Naming récupère les "références" vers l'objet serveur, crée l'objet Stub et rend sa référence au client
6. Les client effectue l'appel au serveur par appel à l'objet Stub

Les amorces (stub/skeleton)
- Programmes jouant le rôle d'adaptateurs pour le transport des appels distants
    - Réalisent les appels sur la couche réseau
    - Pliage / dépliage des paramètres
- A une référence d'OD manipulée par un client correspondant à une référence d'amorce
- Les amorces sont générée par le compilateur d'amorces : rmic

L'amorce client (stub) :
- Représentant local de l'OD qui implante ses méthodes "exportées"
    - Transmet l'invocation distante à la couche inférieure : Remote Reference Layer
    - Réalise le pliage ("marshalling") des arguments des méthodes distantes
    - Inversement, réalise les dépliage ("unmarshalling") des valeurs de retour
- Il utilise pour cela la sérialisation des objets

L'amorce serveur (skeleton) :
- Réalise le dépliage des arguments reçus par le flux de pliage
- Fait appel à la méthode de l'objet distant
- Réalise le pliage de la valeur de retour

Les couches de "bases" de l'architecture RMI :

- La couche des références distantes
  - Permet l'obtention d'une référence d'objet distant à partir de la référence locale au Stub
  - Ce service est assuré par le lancement du programme rmiregistry
    - À ne lancer qu'une seule fois par JVM, pour tous les objets à distribuer
    - Une sorte de service d'annuaire pour les objets distants enregistrés
- La couche de transport
  - Connecte les 2 espaces d'adressage (JVM)
  - Suit les connexions en cours
  - Écoute et répond aux invocations
  - Construit un table des OD disponibles
  - Réalise l'aiguillage des invocations

Etapes de développement d'une application RMI :

1. Définir une interface Java pour l'OD (serveur -> client) => écriture du contrat
2. Créer et compiler une implémentation de cette interface (serveur)
3. Créer les classes stub et skeleton (rmic) (serveur) => étape qui n'est plus nécessaire aujourd'hui car effectuée lors de la compilation
4. Créer et compiler une application serveur RMI (serveur)
5. Démarrer rmiregistry et lancer l'application serveur (serveur) => à faire une seule fois pour pouvoir publier les objets distants
6. Créer, compiler et lancer le programme client accédant à l'OD du serveur (client)

Toutes les interfaces distantes doivent implémenter java.rmi.Remote
Les méthodes de l'interface doivent lever l'exception java.rmi.RemoteException dans le cas où les appels n'aboutissent pas
Si une classe qui implémente l'interface n'hérite d'aucune classe, elle doit hériter de UnicastRemoteObject. Dans le cas où elle hérite ce cette classe, elle doit surcharger le constructeur en levant l'exception RemoteException.

```
Naming.rebind("rmi://www.sitedistant.fr:1099/Calculator", c);
              Adresse où se trouve le serveur   Identifiant      Objet Distant
```