Java RMI

Java RMI

Introduction

Java RMI

Introduction
Mise en œuvre

Java RMI

Introduction
Mise en œuvre
Objet accessible à distance

Java RMI

Introduction
Mise en œuvre
Objet accessible à distance
Premier exemple

Java RMI

Introduction
Mise en œuvre
Objet accessible à distance
Premier exemple
Garbage Collector Distribué

Java RMI

Introduction
Mise en œuvre
Objet accessible à distance
Premier exemple
Garbage Collector Distribué
Comparaison avec CORBA

Java RMI

Introduction
Mise en œuvre
Objet accessible à distance
Premier exemple
Garbage Collector Distribué
Comparaison avec CORBA

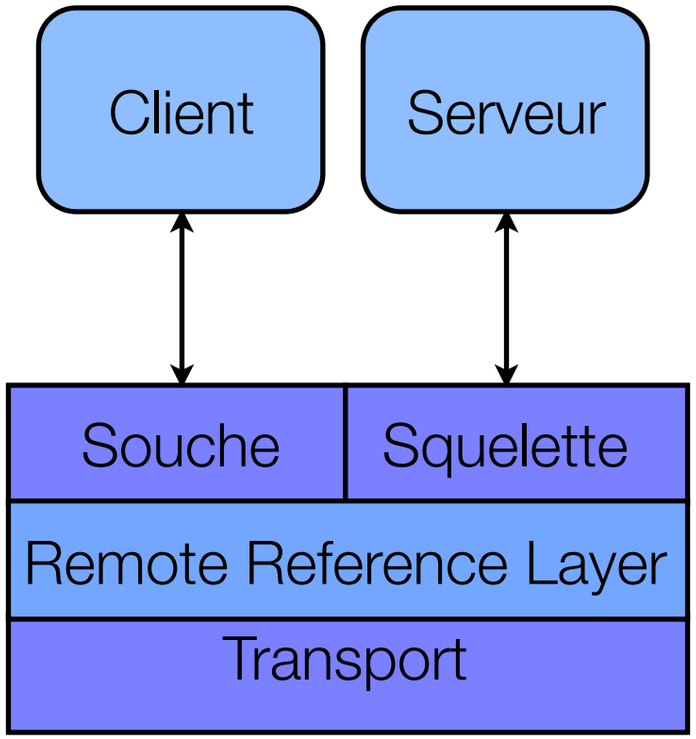
Introduction

- Solution simple et 100% Java : pas de nouveau langage IDL
- permet l'invocation de méthodes entre des objets Java qui s'exécutent dans des machines virtuelles distinctes et réparties
- sérialisation automatique des paramètres et résultats des méthodes même pour des objets Java complexes
- chargement dynamique des classes si nécessaires
- ramasse-miettes distribué
- annuaire de références d’objets RMI

Introduction

Application

RMI System



- Transparence de la communication entre client et serveur
- Bibliothèque RMI = couche de référence distante et couche de transport
- Souches et squelettes Java sont générés, par le compilateur rmic auparavant, à la volée maintenant

Mise en œuvre

- définir l'interface de l'objet distribué (OD)
- implanter l’OD
- générer le talon client et serveur appelés stub et skeleton RMI via le compilateur rmic. Cette étape est facultative maintenant : génération dynamique.
- côté serveur
  - instancier l'OD
  - enregistrer l’OD dans l’annuaire RMI
- côté client
  - obtenir l’OD via l’annuaire RMI
  - l’OD se manipule comme tout autre objet Java

Objet accessible à distance

- Pour qu'un objet Java soit accessible à distance via Java RMI, sa classe doit implanter au moins une interface Java RMI publique

- Une interface Java RMI est une interface Java « classique » héritant au moins de l'interface `java.rmi.Remote`

- Chaque méthode de l'interface doit lever au moins l'exception `java.rmi.RemoteException`

- Tout objet distant passé en paramètre doit être déclaré par son interface RMI

Premier exemple : une interface RMI

```
public interface Hello extends java.rmi.Remote
{
  public void afficher(String chaine)
         throws java.rmi.RemoteException;

  public Message getDernierMessage()
         throws java.rmi.RemoteException;
}
```

Premier exemple : implantation de l'interface

```
public class HelloImpl
        // protocole JRMP
        // extends java.rmi.server.UnicastRemoteObject
        // protocole IIOP, à préférer
        extends javax.rmi.PortableRemoteObject
        implements Hello
{
  protected Message leDernierMessage;

  public HelloImpl() throws java.rmi.RemoteException
  {
   // Appel du constructeur javax.rmi.PortableRemoteObject
   // soulevant l'exception java.rmi.RemoteException.
   super();
   leDernierMessage = new Message("");
  }
```

Premier exemple : implantation de l'interface

```
public void afficher(String chaine)
            throws java.rmi.RemoteException
   {
     System.out.println("chaine=" + chaine);
     leDernierMessage = new Message(chaine);
   }

   public Message getDernierMessage()
            throws java.rmi.RemoteException
   {
     return leDernierMessage;
   }
}
```

Premier exemple : objet Java sérialisable par RMI

- Pour qu'un objet Java soit sérialisable, sa classe doit implanter au moins l'interface `java.io.Serializable`
- Sérialisation de tous les attributs sauf ceux marqués `transient`
- Prise en compte récursive des graphes d'objets sérialisables ainsi que les cycles
- La plupart des classes Java de base sont sérialisables par ex. java.util.*
- Les objets associés à des ressources systèmes ne sont pas sérialisables `Thread`, `Process`, `File`, `AWT`, etc.

Premier exemple : objet Java sérialisable par RMI

```
public class Message implements java.io.Serializable
{
  protected java.util.Date date;
  protected String texte;

  // Le constructeur.
  public Message(String texte) {
    this.date = new java.util.Date();
    this.texte = texte;
  }

  // Pour afficher le message.
  public String toString()
  { return "Message[ . . . ]"; }
}
```

Premier exemple : l'annuaire Java RMI

- Java RMI fournit un annuaire gérant des associations entre noms symboliques et références d'objet RMI
  un serveur contenant 1 objet RMI annuaire

- Côté serveur
  - exporter la référence des objets RMI
  - `java.rmi.Naming.rebind("NomSymbolique",référenceObjetRMI);`

- Côté client
  - importer des références d'OD RMI
  - `référence=java.rmi.Naming.lookup,"//machineAnnuaire/NomSymbolique");`

Premier exemple : un serveur RMI

```
public class Serveur
{
 public static void main(String args[]) throws Exception
 {
   // Créer l'objet accessible par Java RMI.
   HelloImpl obj = new HelloImpl();

   // Enregistrer cet objet dans l'annuaire RMI.
   java.rmi.Naming.rebind("Hello", obj);

   // Ici le programme ne se termine pas !
   // Car attente des invocations distantes sur l'OD.
  }
}
```

Premier exemple : un client RMI

```
public class Client
{
 public static void main (String[] args) throws Exception
 {

   // Obtenir la souche sur l'OD via l'annuaire RMI.
   Hello helloDistant = (Hello)
            java.rmi.Naming.lookup("//localhost/Hello");

   // Invoquer des méthodes comme si l'objet était local.
   helloDistant.afficher("Hello world !");
   Message message = helloDistant.getDernierMessage();
   System.out.println("Le dernier message = " + message);
 }
}
```

Premier exemple : compilation et exécution

- Compiler les sources
  `> javac Hello.java Message.java HelloImpl.java Serveur.java Client.java`

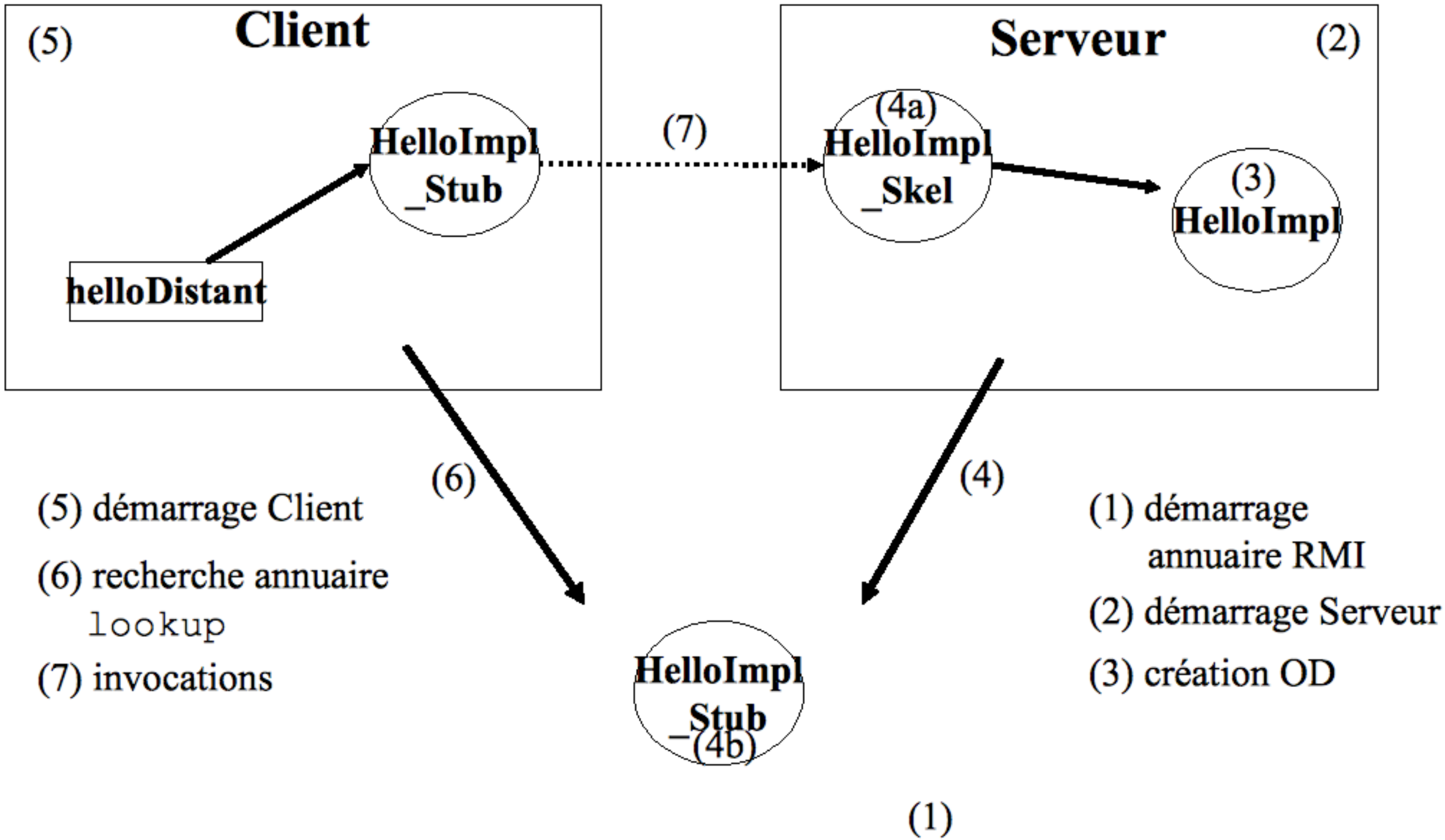
- Générer les stub et les skeletons RMI (optionnel)
  `> rmic HelloImpl`
  produit `HelloImpl_Stub.class` et `HelloImpl_Skel.class`

- Lancer l'annuaire RMI
  `> rmiregistry &`

- Lancer le serveur de l'OD
  `> java Serveur &`

- Lancer le client de l'OD
  `> java Client`

Premier exemple : vision d'ensemble



Distributed Garbage Collector

- Le DGC interagit avec les GC locaux et utilise un mécanisme de reference-counting
- Lorsqu'un OD est passé en paramètre à un autre OD : ref_count++
- Lorsqu'un stub n'est plus référencé : weak reference
  quand le GC du client libère le stub, sa méthode finalize est appelée et informe le DGC : ref_count--
- Lorsque le nombre de références d'un OD = 0 : weak reference
  Le GC du serveur peut alors libérer l'OD
- Le client doit régulièrement renouveler son bail au près du DGC
- Si l’on fait référence à un OD libéré : RemoteException

Comparaison de Java RMI avec CORBA

- RMI est propriétaire SUN / CORBA est une norme internationale
- RMI moins ambitieux que CORBA
- RMI n'offre pas interopérabilité avec des objets d'autres langages
- RMI est beaucoup plus lent que les implantations de CORBA
+ RMI est une implantation standard et gratuite (incluse dans JDK)
+ RMI est object-oriented / CORBA est object-based
+ RMI est plus simple à mettre en œuvre que CORBA
+ Pas besoins d'IDL : seulement l'interface de l'OD et son URL
+ Les OD se manipulent comme les OL (grâce au DGC)
+ RMI supporte le passage d'objets locaux par copie
+ RMI permet une gestion de la sécurité (RMISecurityManager, ACL, ...)