1

Exemple élémentaire de client/serveur RMI

(F. Boyer, Polytech/RICM4)

- Etapes du développement
  1- Spécifier et écrire l'interface des objets Remote
  2- Programmer les classes implémentant ces interfaces
  3- Ecrire le serveur * qui instancie l'objet Remote, puis exporte sa référence distante (*stub)* auprès du *registry*
  4- Ecrire le client qui récupère la référence de l'objet Remote auprès du *registry* et l'invoque

- Etapes de l'exécution
  1- Lancer le *registry* (si celui-ci n'est pas embarqué dans le serveur)
  2- Lancer le serveur
  3- Lancer le client

  ** par simplicité, on suppose qu'il n'y a qu'un serveur et que celui-ci n'instancie qu 'un seul objet Remote*

2

Interface d'un objet Remote

- Contraintes
  - L'interface étend **java.rmi.Remote**
  - Les méthodes lèvent l'exception **java.rmi.RemoteException**

```
public interface FrenchToEnglish extends java.rmi.Remote{
  // translate the given word from french to english
  String translate(String word) throws java.rmi.RemoteException;
}
```

3

Implémentation d'un objet Remote

- Contraintes
  - La classe étend **java.rmi.server.UnicastRemoteObject** (sinon utiliser le modèle par délégation, plus complexe)

```
public class FrenchToEnglishImpl
     extends java.rmi.server.UnicastRemoteObject
     implements FrenchToEnglish {
  public FrenchToEnglishImpl () throws java.rmi.RemoteException{
     super();
  }
  public String translate(String word) throws java.rmi.RemoteException {
     String result = ...
     return result;
  }}
```

4

Implémentation du serveur

- Crée l'objet distant et l'enregistre auprès du registry

```
public class FrenchToEnglishServer {
  public static void main(String args[]) {
      // Crée et Installe un Security Manager
      // nécessaire seulement  si le code est téléchargé
      if (System.getSecurityManager() == null) {
         System.setSecurityManager(new java.rmi.RMISecurityManager()); }

      try {
         // instancie l'objet Remote
         FrenchToEnglish obj = new FrenchToEnglishImpl();

         // enregistre l'objet auprès du registry
         java.rmi.Naming.bind("//"+args[0]+"/FrenchToEnglishTranslator", obj);

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

5

Implémentation du client

- Récupère le stub de l'objet Remote auprès du registry et invoque des méthodes sur ce stub

```
public class FrenchToEnglishClient {
public static void main(String args[]) {

   try {
      FrenchToEnglish obj = (FrenchToEnglish )
               java.rmi.Naming.lookup("//" + args[0] + "/FrenchToEnglishTranslator");

      String  word= obj.translate(args[0]);
      System.out.println(word);

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

6

Lancement et usage du registry

- Lancement en stand-alone (port par défaut = 1099)
  > rmiregistry 2001
- Lancement embarqué dans le serveur
  - Utiliser la classe LocateRegistry: **LocateRegistry.createRegistry(2001)**
- La classe Naming
  - Exemples d'URL
    `//machineX:2001/FrenchToEnglishTranslator`
    `//:2001/FrenchToEnglishTranslator`
    `/FrenchToEnglishTranslator`
  - Méthodes Statiques
    - **bind(String url, Remote r), rebind(String url, Remote r), unbind(String url)**
      enregistre/désenregistre un stub auprès du serveur
    - **Remote lookup(String url)**
      retourne un stub

7

Lancement (manuel) du serveur

- *Les classes requises doivent être accessibles via le CLASSPATH ou téléchargeables via HTTP ou file:*
- *Le fichier .policy autorise l'usage du accept et du connect sur les Sockets si les classes doivent être téléchargées*

- Par défaut
  - Java **FrenchToEnglishServer** localhost:1099
- Si téléchargement de code
  - java
    -Djava.security.policy=./.policy
    -Djava.rmi.server.codebase=http://hostwww/FrenchToEnglich/myclasses/
    FrenchToEnglishServer localhost:1099

  fichier .policy:

```
grant { permission java.net.SocketPermission "*:1024-65535","connect,accept";
        permission java.net.SocketPermission "*:80", "connect";
        // permission java.security.AllPermission; /*autorise tout */ };
```

8

Lancement (manuel) du client

- *Les classes requises doivent être accessibles via le CLASSPATH ou téléchargeables via HTTP ou file:*
- *Le fichier .policy autorise l'usage du accept et du connect sur les Sockets si les classes doivent être téléchargées*

- Par défaut
  - Java **FrenchToEnglishClient** localhost:1099
- Si téléchargement de code
  - java
    -Djava.security.policy=./.policy
    -Djava.rmi.server.codebase=http://hostwww/FrenchToEnglich/myclasses/
    FrenchToEnglishClient localhost:1099

  fichier .policy:

```
grant { permission java.net.SocketPermission "*:1024-65535","connect ";
        permission java.net.SocketPermission "*:80", "connect";
        // permission java.security.AllPermission; /*autorise tout */ };
```