



Università degli Studi di Bologna
Scuola di Ingegneria

Corso di Reti di Calcolatori T

Esercitazione 7 (svolta)
Java RMI e Riferimenti Remoti
Un RMI Registry Remoto

Luca Foschini

Anno accademico 2013/2014

Esercitazione 7 1

Servizio di nomi distribuito

Si progettano **un servizio di nomi RegistryRemoto**, che sia capace di **facilitare la interazione tra clienti e servitori, fornendo il servizio a utilizzatori su macchine diverse che intendano usare come Clienti o Servitori RMI**, superando il problema della loro collocazione rispetto ad un registry di RMI.

Il RegistryRemoto deve permettere:

- ai servitori di registrare la propria **disponibilità di servizio**, tenendo traccia del **nome del servizio** e della **localizzazione di deployment**;
- ai clienti di ottenere i **riferimenti remoti necessari** per il servizio di cui hanno bisogno.

Il **RegistryRemoto** è realizzato come **server RMI** e deve poi consentire una invocazione dei servizi da parte dei clienti attraverso riferimenti remoti.

Esercitazione 7 2

Specifica: il RegistryRemoto

Si progetti un **servizio di naming remoto** (**RegistryRemoto**) che **consenta ai Client di recuperare i riferimenti ad oggetti remoti Server che si siano registrati**.

In particolare **RegistryRemoto** è realizzato come server RMI e implementa le seguenti operazioni per due tipologie di client.

Per i clienti:

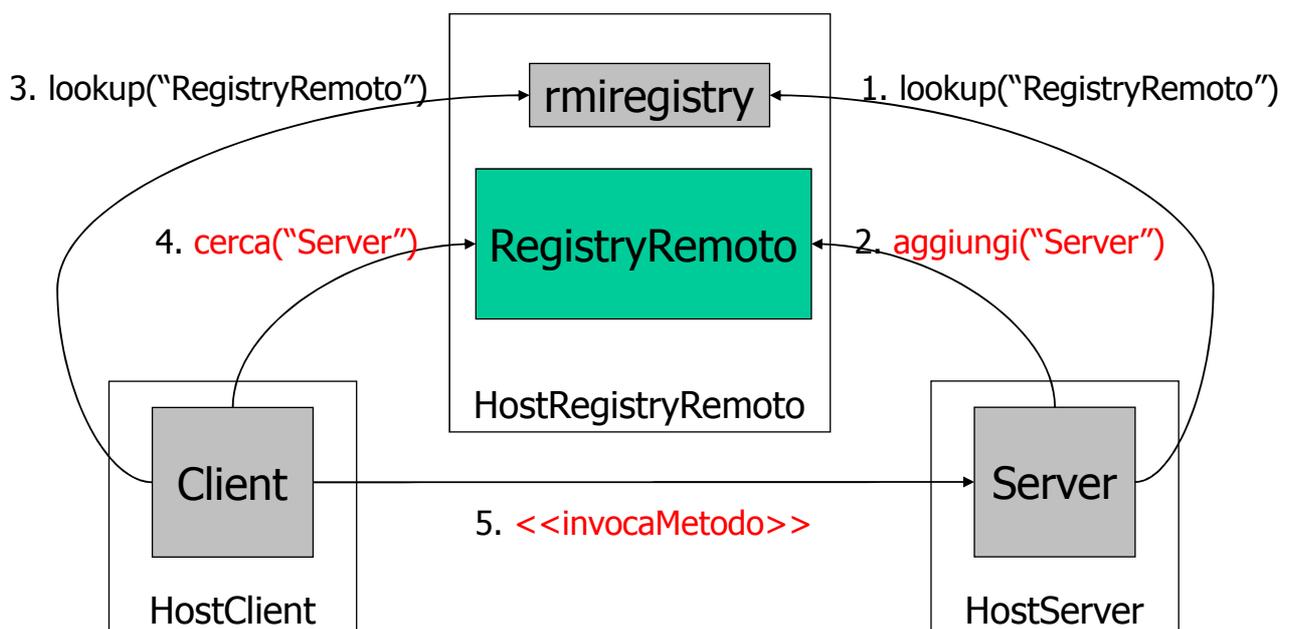
- **ricerca del primo riferimento** al server remoto registrato con il **nome logico** dato;
- **ricerca di tutti i riferimenti** ai server remoto registrati con lo stesso **nome logico**;

Per i fornitori di servizio, oltre alle funzioni offerte ai client:

- **ottenimento lista di tutte le coppie nome logico/riferimento** mantenute dal RegistryRemoto (servizio **senza parametri di ingresso**);
- **aggiunta** di un server remoto, dato il **nome logico** e il **riferimento remoto**;
- **eliminazione della prima** entry corrispondente al **nome logico** dato;
- **eliminazione di tutte le** entry registrate con il **nome logico**.

Esercitazione 7 3

Architettura di riferimento



Esercitazione 7 4

Progetto e sue parti

Il progetto RMI si compone, oltre alle classi già viste nell'esercitazione 6, delle ulteriori classi :

- Un'interfaccia remota **RegistryRemotoClient** (contenuta nel file *RegistryRemotoClient.java*) in cui vengono definiti i metodi invocabili dai clienti (cerca, cercaTutti);
- Un'interfaccia remota **RegistryRemotoServer** (contenuta nel file *RegistryRemotoServer.java*) che estende **RegistryRemotoClient** aggiungendo i metodi invocabili dai server (restituisceTutti, aggiungi, eliminaPrimo, eliminaTutti);
- Una classe per la realizzazione del **RegistryRemoto** (**RegistryRemotoImpl** contenuta nel file *RegistryRemotoImpl.java*), che implementa i metodi di **RegistryRemotoServer** invocabili in remoto.

NOTA: possibilità di usare interfacce remote diverse con scope diversi in base al **ruolo** di utilizzo dell'oggetto remoto!!

Sarà inoltre necessario **modificare opportunamente Server e Client dell'esercitazione 6** in modo che effettuino la registrazione e la ricerca del riferimento all'oggetto Server, presso il RegistryRemoto (invece che sull'rmiregistry locale).

Esercitazione 7 5

Deployment

Il progetto RMI si compone delle tre parti **RegistryRemoto**, **Cliente** e **Servitore**, che sono sotto il controllo utente e da attivare, e anche della parte di supporto resa necessaria ad RMI per il sistema di nomi, il registry da attivare sul nodo del RegistryRemoto.

Il **RegistryRemoto** presenta l'interfaccia di invocazione:

```
java -Djava.security.policy=rmi.policy  
RegistryRemotoImpl [rmiregistryPort]
```

Il **Server** presenta l'interfaccia di invocazione:

```
java -Djava.security.policy=rmi.policy  
ServerCongressoImpl NomeHostRegistryRemoto [rmiregistryPort]
```

Il **Client** viene attivato con:

```
java -Djava.security.policy=rmi.policy  
ClientCongresso NomeHostRegistryRemoto [rmiregistryPort]
```

L'ordine di attivazione è prima il registry RMI, poi il RegistryRemoto (sullo stesso nodo), poi il server, infine la parte cliente.

Esercitazione 7 6

Interfaccia RegistryRemotoClient

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

public interface RegistryRemotoClient extends Remote {

    public Remote cerca(String nomeLogico)
                        throws RemoteException;
    public Remote[] cercaTutti(String nomeLogico)
                        throws RemoteException;

}
```

Esercitazione 7 7

Interfaccia RegistryRemotoServer

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

public interface RegistryRemotoServer
                        extends RegistryRemotoClient {

    // Tabella: la prima colonna contiene i nomi, la seconda i riferimenti remoti
    public Object[][] restituisciTutti()
                        throws RemoteException;
    public boolean aggiungi(String nomeLogico, Remote riferimento)
                        throws RemoteException;
    public boolean eliminaPrimo(String nomeLogico)
                        throws RemoteException;
    public boolean eliminaTutti(String nomeLogico)
                        throws RemoteException;

}
```

Esercitazione 7 8

RegistryRemoto 1/5

```
public class RegistryRemotoImpl extends UnicastRemoteObject
                                implements RegistryRemotoServer
{ final int tableSize = 100;
  // Tabella: la prima colonna contiene i nomi, la seconda i riferimenti remoti
  Object [][] table = new Object[100][2];

  // Costruttore
  public RegistryRemotoImpl() throws RemoteException
  { super();
    for( int i=0; i<tableSize; i++ )
      { table[i][0]=null; table[i][1]=null; }
  }

  public synchronized Remote cerca(String nomeLogico)
                                throws RemoteException
  { Remote risultato = null;
    if( nomeLogico == null ) return null;
    for( int i=0; i<tableSize; i++ )
      if( nomeLogico.equals((String)table[i][0]) ){
        risultato = (Remote) table[i][1];
        break;
      }
    return risultato;
  }
}
```

Esercitazione 7 9

RegistryRemoto 2/5

```
public synchronized Remote[] cercaTutti(String nomeLogico)
                                throws RemoteException
{ int cont = 0;
  if( nomeLogico == null ) return new Remote[0];
  for( int i=0; i<tableSize; i++ )
    if( nomeLogico.equals((String)table[i][0]) ) cont++;
  Remote[] risultato = new Remote[cont];
  // Ora lo uso come indice per il riempimento
  cont=0;
  for( int i=0; i<tableSize; i++ )
    if( nomeLogico.equals((String)table[i][0]) )
      risultato[cont++] = (Remote)table[i][1];
  return risultato;
}
```

Esercitazione 7 10

RegistryRemoto 3/5

```
public synchronized Object[][] restituisciTutti()
    throws RemoteException
{ int cont = 0;
  for (int i = 0; i < tableSize; i++)
    if (table[i][0] != null) cont++;
  Object[][] risultato = new Object[cont][2];
  // Ora lo uso come indice per il riempimento
  cont = 0;
  for (int i = 0; i < tableSize; i++)
    if (table[i][0] != null) {
      risultato[cont][0] = table[i][0];
      risultato[cont][1] = table[i][1];
    }
  return risultato;
}

public synchronized boolean aggiungi(String nomeLogico,
    Remote riferimento) throws RemoteException
{ boolean result = false; // Cerco la prima posizione libera e la riempio
  if((nomeLogico == null)|| (riferimento == null)) return risultato;
  for(int i=0; i<tableSize; i++)
    if( table[i][0] == null ){
      table[i][0]= nomeLogico; table[i][1]=riferimento;
      result = true; break;
    }
  return result;
}
```

Esercitazione 7 11

RegistryRemoto 4/5

```
public synchronized boolean eliminaPrimo
    (String nomeLogico) throws RemoteException
{ boolean risultato = false;
  if( nomeLogico == null ) return risultato;
  for( int i=0; i<tableSize; i++ )
    if( nomeLogico.equals( (String)table[i][0]) )
      { table[i][0]=null;
        table[i][1]=null;
        risultato=true;
        break;
      }
  return risultato;
}

public synchronized boolean eliminaTutti
    (String nomeLogico) throws RemoteException
{ boolean risultato = false;
  if( nomeLogico == null ) return risultato;
  for( int i=0; i<tableSize; i++ )
    if( nomeLogico.equals((String)table[i][0]) )
      { if( risultato == false ) risultato = true;
        table[i][0]=null;
        table[i][1]=null;
      }
  return risultato;
}
```

Esercitazione 7 12

RegistryRemoto 5/5

```
public static void main (String[] args) {
    int registryRemotoPort = 1099;
    String registryRemotoHost = "localhost";
    String registryRemotoName = "RegistryRemoto";

    if (args.length != 0 && args.length != 1) // Controllo argomenti
    { System.out.println("..."); System.exit(1); }
    if (args.length == 1)
    { try { registryRemotoPort = Integer.parseInt(args[0]); }
      catch (Exception e) {...}
    }

    // Registrazione RegistryRemoto presso rmiregistry locale
    String completeName = "/" + registryRemotoHost + ":" +
    registryRemotoPort + "/" + registryRemotoName;
    try
    { RegistryRemotoImpl serverRMI = new RegistryRemotoImpl();
      Naming.rebind(completeName, serverRMI);
    } catch (Exception e) {...}
}
}
```

Esercitazione 7 13

Server

```
public class ServerCongressoImpl extends UnicastRemoteObject
    implements ServerCongresso
{
    // Riportiamo solo il main, il resto del codice è uguale
    public static void main(String[] args)
    {
        prog = new Programma[3]; // creazione programma
        for (int i = 0; i < 3; i++) prog[i] = new Programma();
        int registryRemotoPort = 1099; // default
        String registryRemotoName = "RegistryRemoto";
        String serviceName = "ServerCongresso";

        if (args.length != 1 && args.length != 2) {...} // Controllo argomenti
        String registryRemotoHost = args[0];
        if (args.length == 2)
        { try { registryRemotoPort = Integer.parseInt(args[0]); }
          catch (Exception e) {...} } // if

        // Registrazione servizio presso RegistryRemoto
        String completeRemoteRegistryName = "/" + registryRemotoHost +
        ":" + registryRemotoPort + "/" + registryRemotoName;
        try
        { RegistryRemotoServer registryRemoto =
        (RegistryRemotoServer) Naming.lookup(completeRemoteRegistryName);
          ServerCongressoImpl serverRMI = new ServerCongressoImpl();
          registryRemoto.aggiungi(serviceName, serverRMI);
        } catch (Exception e) {...}
    } /* main */ ... } // ServerCongressoImpl
}
```

Esercitazione 7 14

Client

class ClientCongresso

```
{ public static void main(String[] args) // Riportiamo solo inizio main
{ int registryRemotoPort = 1099;
  String registryRemotoName = "RegistryRemoto";
  String serviceName = "ServerCongresso";
  BufferedReader stdIn =
    new BufferedReader(new InputStreamReader(System.in));

  if (args.length != 1 && args.length != 2) {...}
  String registryRemotoHost = args[0];
  if (args.length == 2)
  { try {registryRemotoPort = Integer.parseInt(args[0]); }
    catch (Exception e) {...}
  }

  // Recupero il riferimento al servizio remoto presso il RegistryRemoto
  try
  { String completeRemoteRegistryName = "/" +
    registryRemotoHost + ":" + registryRemotoPort + "/" +
    registryRemotoName;
    RegistryRemotoClient registryRemoto = (RegistryRemotoClient)
    Naming.lookup(completeRemoteRegistryName);
    ServerCongresso serverRMI =
    (ServerCongresso) registryRemoto.cerca(serviceName);
  }
  ... // Il resto del codice è uguale
```