Gruppo 17

Università degli Studi di
Bologna Scuola di Ingegneria

Esercitazione 7

Java RMI e Riferimenti Remoti RMI
Registry Remoto come pagine gialle

Corso di Reti di Calcolatori T

Daniele Magagnoli, Giulia Martina Bal,
Antonio Cassanelli, Gianmiriano Porrazzo

Anno Accademico 2021/2022

SPECIFICHE:

L’esercitazione chiedeva la realizzazione di un’applicazione C/S, con l’utilizzo di RMI, progettando un servizio di nomi RegistryRemoto che fornisce un servizio di pagine gialle ad utilizzatori su macchine diverse.

Il RegistryRemoto deve permettere:

- Ai server di registrarsi con nome del servizio e localizzazione del deployment. Può essere associato uno dei tag disponibili per avere la possibilità di ricercare una funzionalità in base al tag.

- Ai clienti di poter interrogare il RegistryRemoto attraverso i tag che descrivono i servizi

RegistryRemotoImpl

Il RegistryRemoto è realizzato come server RMI. Deve consentire ai clienti con un tag di ottenere dei nomi logici di server. Il costruttore crea una matrice in cui la prima colonna è formata da nomi, la seconda da riferimenti e la terza da tag.

```
*  Implementazione del Registry Remoto.

import java.rmi.Naming;

public class RegistryRemotoImpl extends UnicastRemoteObject implements
    RegistryRemotoTagServer {

    private static final String [] tags={
        "Medico", "Bar", "Dentista","Supermercato","Ristorante","Congresso"
    };
    //lista dei tag consentiti
    // num. entry [nomelogico][ref][tag]
    final int tableSize = 100;

    // Tabella: la prima colonna contiene i nomi, la seconda i riferimenti remoti e la terza il tag relativo
    Object[][] table = new Object[tableSize][3];

    //costruttore
    public RegistryRemotoImpl() throws RemoteException {
        super();
        for (int i = 0; i < tableSize; i++) {
            table[i][0] = null;
            table[i][1] = null;
            table[i][2] = null;
        }
    }
```

```java
// restituisco tutti i riferimenti che contengono il tag specificato
public synchronized String[] cercaTag (String[] tag) throws RemoteException{
    int dim = 0; // dimensione della lista restituita, ovvero il numero di servizi con il tag specificato
    String[] res =null; // lista che restituisco alla fine

    // inizio a contare quanti servizi hanno il tag specificato
    for(int i=0; i<tableSize && table[i][0]!=null; i++){ // Se table[i][0] == null non Ã¨ un Servizio
        int find=0; // conto i tag che fanno match
        String[] tmp = (String[])table[i][2]; //estraggo tutti i tag relativi al servizio
        for (int j=0; j<tmp.length;j++){// scorro i tag relativi al servizio
            for (int k=0; k<tag.length;k++){
                if(tmp[j].equals(tag[k]))
                    find++;
            }
        }
        if(find==tag.length){
            dim++;
        }
    }
    if (dim==0)
        throw new RemoteException("Nessuna corrispondenza trovata");

    res=new String[dim];// creo la lista della dimensione giusta
    dim=0; // per riusare la variabile

    for (int i=0; i<tableSize && table[i][0]!=null; i++){
        int find=0;
        String[] tmp = (String[])table[i][2];
        for (int j=0;j<tmp.length;j++){
            for (int k=0;k<tag.length;k++){
                if(tmp[j].equals(tag[k]))
                    find++;
            }
        }
        if(find==tag.length){
            res[dim]=(String)table[i][0];// salvo il nome logico nella lista
            dim++;
        }
    }
    return res;
}
```

CercaTag

La funzione cerca tag restituisce, in un array di stringhe, tutti i riferimenti che contengono i tag specificati dall'utente. Se non ci sono riferimenti con il tag indicato la funzione lancia una RemoteException, altrimenti crea una lista come risultato di dimensione pari al numero di corrispondenze trovate.

AssociaTag e avvio

```java
// associo i tag a nomi logici del servizio
public synchronized boolean associaTag(String nomeLogico, String[] tag)throws RemoteException{

    boolean res=false;
    // controllo che il tag sia presente tra quelli che posso scegliere
    for(String temp : tag){ //scorro i tag passati in ingresso
        for (String reg : tags){ //scorro i tag tra cui posso scegliere
            if(temp.equals(reg))
                res=true;
        }
    }
    if(res=false)
        throw new RemoteException("Tag non valido");
    // essendo sicuro che il tag esista, associo il tag al Servizio
    res=false;
    if((nomeLogico==null) || (tag==null))
        return res;
    for(int i=0; i<tableSize;i++){
        if(nomeLogico.equals((String) table[i][0])){
            table[i][2] = tag;
            res=true;
        }
    }
    return res;
}
```

La funzione associa tag controlla che il tag sia presente tra quelli scelti e lo inserisce nella matrice. Il RegistryRemoto viene avviato e si collega alla porta 1099 (di default) .

```java
// Avvio del Server RMI
public static void main(String[] args) {

    int registryRemotoPort = 1099;
    String registryRemotoHost = "localhost";
    String registryRemotoName = "RegistryRemoto";

    // Controllo dei parametri della riga di comando
    if (args.length != 0 && args.length != 1) {
        System.out.println("Sintassi: ServerImpl [registryPort]");
        System.exit(1);
    }
    if (args.length == 1) {
        try {
            registryRemotoPort = Integer.parseInt(args[0]);
        } catch (Exception e) {
            System.out
            .println("Sintassi: ServerImpl [registryPort], registryPort intero");
            System.exit(2);
        }
    }

    // Impostazione del SecurityManager
    if (System.getSecurityManager() == null)
        System.setSecurityManager(new RMISecurityManager());

    // Registrazione del servizio RMI
    String completeName = "//" + registryRemotoHost + ":" + registryRemotoPort
            + "/" + registryRemotoName;
    try {
        RegistryRemotoImpl serverRMI = new RegistryRemotoImpl();
        Naming.rebind(completeName, serverRMI);
        System.out.println("Server RMI: Servizio \"" + registryRemotoName
                + "\" registrato");
    } catch (Exception e) {
        System.err.println("Server RMI \"" + registryRemotoName + "\": "
                + e.getMessage());
        e.printStackTrace();
        System.exit(1);
    }
}
```

```java
public class Server extends UnicastRemoteObject implements ServerInt{
  public Server() throws RemoteException {
    super();
  }
  // Avvio del Server RMI
  public static void main(String[] args) {
    int registryRemotoPort = 1099;
    String registryRemotoName = "RegistryRemoto";
    String serviceName = ""; // "ServerCongresso";
    String[] tag=null;
    boolean rp=false;// per vedere se c'Ã¨ la porta o meno

    // Controllo dei parametri della riga di comando
    if (args.length <2) {
      System.out
          .println("Sintassi: ServerCongressoImpl NomeHostRegistryRemoto [registryPort] almeno un tag");
      System.exit(1);
    }
    String registryRemotoHost = args[0];
    try {
        registryRemotoPort = Integer.parseInt(args[1]);
        rp=true;
    }catch (Exception e) {
        rp=false;
    }
    if(rp && args.length>=3){
        tag=new String[args.length-2];
        for(int i=2;i<args.length;i++)
            tag[i-2]=args[i];
    }
    else if(args.length>=2 && !rp){
        tag=new String[args.length-1];
        for(int i=1; i<args.length;i++)
            tag[i-1]=args[i];
    }

    serviceName="Server"+tag[0];
```

Server

```java
// Registrazione del servizio RMI
String completeRemoteRegistryName = "//" + registryRemotoHost + ":"
    + registryRemotoPort + "/" + registryRemotoName;

try {
  RegistryRemotoTagServer registryRemoto = (RegistryRemotoTagServer) Naming
      .lookup(completeRemoteRegistryName);
  Server serverRMI = new Server();
  registryRemoto.aggiungi(serviceName, serverRMI);
  //associo il tag al nome logico
  registryRemoto.associaTag(serviceName,tag);
  System.out.println("Server RMI: Servizio \"" + serviceName
      + "\" registrato");
} catch (Exception e) {
  System.err.println("Server RMI \"" + serviceName + "\": "
      + e.getMessage());
  e.printStackTrace();
  System.exit(1);
}
```

Il server si collega alla porta del registry e successivamente effettua i controlli sui parametri. In questo caso deve essere inserito almeno un tag tra quelli scelti. Dopo l'avvio procede per la registrazione del servizio e associa i tag tramite la funzione associaTag.

```
//  Impostazione del SecurityManager
if (System.getSecurityManager() == null)
    System.setSecurityManager(new RMISecurityManager());

while(true){
// Connessione al servizio RMI remoto
try {
    String completeRemoteRegistryName = "//" + registryRemotoHost + ":"
            + registryRemotoPort + "/" + registryRemotoName;

    RegistryRemotoTagClient registryRemoto =
            (RegistryRemotoTagClient) Naming.lookup(completeRemoteRegistryName);
    /*Chiedo all'utente di inserire un tag*/
    System.out.println("Inserire un tag tra quelli consentiti(Inserire uno o piÃ¹ e premere invio)");
    String[] tag=stdIn.readLine().split(" ");

    String[] res=registryRemoto.cercaTag(tag);
    System.out.println("ClientRMI: trovati "+res.length+" server con i tag richiesti\n");
    for (String s : res)
        System.out.println(s+" ");


    serviceName=res[0];

    ServerInt serverRMI = (ServerInt) registryRemoto.cerca(serviceName);
    System.out.println("\nClientRMI: Servizio \"" + serviceName + "\" connesso");
} catch (Exception e) {
    System.err.println("ClientRMI: " + e.getMessage());
    e.printStackTrace();
    System.exit(2);
```

Client

Il Client deve chiedere un servizio remoto al Server al quale si connette attraverso la funzione lookup(). Successivamente chiede all'utente di inserire un tag tra quelli consentiti e tramite la funzione cercaTag implementata in RegistryRemotoImpl ottiene il numero server con i tag inseriti.

CONSIDERAZIONI FINALI

Per far funzionare il server in cartelle differenti abbiamo aggiunto nella cartella i file .class , il file di testo rmi.policy e la classe Java in cui viene implementato il server.

GRAZIE
PER L’ATTENZIONE