Esercizi su Callback RMI

Esercitazione di Laboratorio di Programmazione di Rete A

Daniele Sgandurra

Università di Pisa

03/12/2008

Esercizio Settimana Scorsa

Sviluppare un'applicazione RMI per la gestione di un'elezione. Il server esporta un insieme di metodi:

- `public void vota(String nome)`: accetta come parametro il nome del candidato. Non restituisce alcun valore. Registra il voto di un candidato in una struttura dati opportunamente scelta.
- `public int risultato(String nome)`: accetta come parametro il nome di un candidato e restituisce i voti accumulati da tale candidato fino a quel momento.
- un metodo che consente di ottenere i nomi di tutti i candidati, con i rispettivi voti, ordinati rispetto ai voti ottenuti.

Esercizio Votazione con Callback

- Modificare l'esercizio in modo che il server notifichi ogni nuovo voto ricevuto a tutti i client che hanno votato fino a quel momento.
- Il server deve tenere in un'opportuna struttura dati una lista dei client registrati. La registrazione dei client avviene nel momento del voto.
- Il server ha bisogno di un metodo per permettere ai client di registrarsi e di de-registrarsi. Dato che i client devono chiamare questi metodi sul server, questi i metodi sono remoti.

Esercizio Votazione con Callback

- Ogni client deve pubblicizzare i propri metodi remoti per essere notificato dal server: crea un oggetto che implementa un'interfaccia remota chiamabile dal server.
- Lato client non si esegue l'rmiregistry: infatti, quando un client si registra sul server passa come parametro un riferimento ad un proprio oggetto remoto, per cui il server non ha bisogno di fare una lookup su quell'oggetto.
- Ogni client, per terminare deve: (i) invocare la de-registrazione; (ii) fare l'operazione inversa della export, per rimuovere l'oggetto remoto dagli oggetti esportati (altrimenti rimane "appeso" e va terminato con `CTRL+C`).

Esempio Client: l'Oggetto Remoto per Callback

L'oggetto remoto passato al server per effettuare le callback:

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

public interface ClientCallbackInterface extends Remote
{
      public ... notify(...) throws RemoteException;
}

public class ClientCallbackImpl implements ClientCallbackInterface
{
      public ... notify(...)
      {
         ...
      }
}
```

Esempio Client: la Registrazione

Il client passa uno stub dell'oggetto remoto al server:

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;

public class VotazioneClient
{
   public static void main(String[] args)
   {
         ...
         Registry registry = LocateRegistry.getRegistry(server);
         ... stub = (...) registry.lookup("...");
         ClientCallbackImpl client = new ClientCallbackImpl();
         ClientCallbackInterface callbackStub = (ClientCallbackInterface)
                              UnicastRemoteObject.exportObject(client, 0);
         stub.vota(nomeCandidato, callbackStub);
         ....
         stub.unregisterClient(callbackStub);
         UnicastRemoteObject.unexportObject(client, true);
      }catch(Exception e){e.printStackTrace();}
   }
}
```

Esempio Server: Metodi per (De)Registrarsi

Il server esporta metodi per registrarsi e de-registrarsi per le callback:

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

public interface Votazione extends Remote
{
   void vota(String nome, ClientCallbackInterface client) throws RemoteException;
   ...
   void unregisterClient(ClientCallbackInterface client) throws RemoteException;
}
```

Esempio Server: Notifiche ai Client

Ogni volta che arriva un voto, il server notifica tutti i client registrati:

```
private synchronized void notifyClients(Candidato c)
{
   for(...)
   {
      ClientCallbackInterface client = ...;
      try
      {
         client.notify(c);
      }catch(RemoteException e){System.out.println(e);}
   }
}

public void unregisterClient(ClientCallbackInterface client)
{ .... }

...
```

Soluzioni

Inviare la soluzione degli esercizi (solo i file .java) a :

`ricci@di.unipi.it`
`sgandurra@di.unipi.it`

Tra due settimane saranno disponibili le soluzioni.