ESERCITAZIONE 6

Java RMI

OBIETTIVI

Sviluppare un'applicazione C/S che fornisca due servizi:

- Contare le righe che contengono un numero di parole superiore ad un intero.
- Eliminare una riga da file remoto, inviando il nome del file e il numero di riga.

Davide di Molfetta, Mirko Legnini, Daniele Nanni Cirulli, Natanaele Stagni, Lorenzo Venerandi

SPECIFICHE

METODI REMOTI: Sono CONTA_RIGHE e ELIMINA_RIGA.

Entrambi accettano come parametro il nome di un file remoto ed un intero.

CONTA_RIGHE:

- Restituisce il numero delle righe che contengono un numero di parole maggiore dell'intero inviato.

ELIMINA_RIGA:

- se il file esiste e se ha un numero di righe almeno pari all'intero inviato dal cliente, restituisce il nome del file modificato e un intero che indica le righe presenti.

In caso di errore, si solleva un'eccezione remota.

CLIENTE

Il Cliente è realizzato come un filtro che chiede a linea di comando quale operazione eseguire.
Prima di avviare il ciclo di richieste si connette al Registry e istanzia un oggetto remoto Server RMI.
Gestiamo qualsiasi errore Server Side tramite una Remote Exception.

```
while ((service = stdIn.readLine()) != null) {

    if (service.equals("Count")) {
        int min = 0;
        System.out.print("Nome file? ");
        fileName=stdIn.readLine();
        System.out.println("Numero minimo parole");
        try{
            min= Integer.parseInt(stdIn.readLine());
        }
        catch(NumberFormatException e) {
            System.out.println("Il numero di parole deve essere un intero");
            System.out.print("Servizio (Count=conta linee, Cancel=cancella linea): ");
            continue;
        }
        System.out.println("Righe con pi\ufffd di "+ min + "parole= " + serverRMI.conta_righe(fileName, min));
    }// Count=conta linee

    else if (service.equals("Cancel")) {
        int line= 0;
        Risposta r;
        System.out.print("Nome file? ");
        fileName=stdIn.readLine();
        System.out.println("Numero linea da eliminare");
        try{
            line= Integer.parseInt(stdIn.readLine());
        }
        catch(NumberFormatException e) {
            System.out.println("Il numero di parole deve essere un intero");
            System.out.print("Servizio (Count=conta linee, Cancel=cancella linea): ");
            continue;
        }
        if ((r=serverRMI.elimina_riga(fileName, line)).getRighe()>=0)
            System.out.println(r.toString());
        else
            System.out.println("Numero riga maggiore del massimo o file inesistente");

    } //

    else
        System.out.println("Servizio non disponibile");

    System.out.print("Servizio (Count=conta linee, Cancel=cancella linea): ");
} // while (!EOF), fine richieste utente
```

CLASSI DI APPOGGIO

```java
public class Risposta implements Serializable {

    static final long serialVersionUID=1L;
    private String nomeFile;
    private int nRighe;
    public Risposta(String nomeFile, int nRighe) {
        super();
        this.nomeFile = nomeFile;
        this.nRighe = nRighe;
    }

    public int getRighe(){
        return this.nRighe;
    }
    @Override
    public String toString() {
        return "Risposta [nomeFile=" + nomeFile + ", nRighe=" + nRighe + "]";
    }

}
```

Oggetto risposta serializzabile

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

public interface RemOp extends Remote {

    public int conta_righe(String fileName, int max) throws RemoteException;
    public Risposta elimina_riga(String fileName, int line) throws RemoteException;
}
```

Interfaccia Server

SERVER

```java
public synchronized int conta_righe(String fileName, int min) throws RemoteException {
    try {
        int res = 0;
        String linea;
        File f = new File(fileName);
        BufferedReader br = new BufferedReader(new FileReader(f));
        while ((linea = br.readLine()) != null) {
            if (linea.split("[ \t]+").length > min)
                res++;
        }
        br.close();
        return res;
    } catch (Exception e) {
        throw new RemoteException("Rilancio eccezione: " + e.getMessage(), e);
    }
}
```

Il metodo conta righe legge una linea per volta e conta le parole tramite una Split.

Il metodo elimina riga legge le righe una per volta, contandole, e scrive su un file d'appoggio tutte le righe esclusa quella da eliminare.

```java
public synchronized Risposta elimina_riga(String fileName, int line) throws RemoteException {
    try {
        int curLine = 1;
        String linea;
        File f = new File(fileName), tmp = new File("temp");
        BufferedReader br = new BufferedReader(new FileReader(f));
        PrintWriter pw = new PrintWriter(tmp);
        while ((linea = br.readLine()) != null) {
            if (curLine !=line) {
                pw.println(linea);
            }
            curLine++;
        }
        br.close();
        pw.close();
        Files.move(tmp.toPath(), f.toPath(), StandardCopyOption.REPLACE_EXISTING);
        if(line>=curLine) throw new Exception("Numero linea maggiore dimensione file");
        return new Risposta(fileName, curLine - 2);
    } catch (Exception e) {
        throw new RemoteException("Rilancio eccezione: " + e.getMessage(), e);
    }
}
```

Eventuali eccezioni in apertura o lettura del file vengono rilanciate in una Remote Exception

CONCLUSIONI

- La realizzazione dei servizi tramite RMI è più veloce e versatile rispetto al semplice utilizzo di Socket
- Tutti i metodi del Server sono stati implementati synchronized per evitare ambiguità nelle operazioni sui file