

19 Ottobre 2021



Bernardini Claudio
Corsetti Luca
Giardina Gianluca
Straccali Leonardo

Esercitazione 1

Socket in JAVA
(connectionless)

Obiettivi esercitazione

Hard skills

1

Reindirizzare le connessioni

2

Realizzazione di thread

3

Creazione e uso di Socket

Soft skills

1

Collaborazione per raggiungere un fine comune

2

Organizzazione del proprio lavoro in team

Introduzione

Client

1

Richiede al DS dove si trova il RS corrispondente al file (quale porta)

2

Chiede all'utente quale linee devono essere invertite

3

Richiede al RS lo swap delle due linee del file

DiscoveryServer

1

Crea thread RS (RowSwapServer) per ogni file disponibile

2

Si mette in ascolto di richieste da parte dei client

3

Reindirizza il client sulla porta del RS, se questo esiste

RowSwapServer

1

Si mette in ascolto delle richieste

2

Inverte le linee del file richieste dai client

3

Implementazione JAVA - DS 2

1

```
// check argomenti
if (((args.length % 2) == 0) || (args.length < 3)) {
    System.out.println("Usage: java DiscoveryServer serverPort " +
        "nomeFile1 port1" +
        "nomeFile2 port2" +
        " .. " +
        "nomeFileN portN");
    System.exit(-1);
}

// salvataggio porta del DiscoveryServer
port = Integer.parseInt(args[0]);

if (port < 1024 || port > 65535) {
    System.out.println("La porta " + port
        + " non è utilizzabile."
        + "Le porte devono essere comprese tra 1024 e 65535.");
    System.exit(-1);
}

int index = 0;

// salvataggio file & port nei rispettivi array
for (int i = 1; i < args.length; i += 2) {
    files[index] = args[i];
    ports[index] = Integer.parseInt(args[i + 1]);

    if (ports[index] < 1024 || ports[index] > 65535) {
        System.out.println("La porta " + ports[index]
            + " non è utilizzabile."
            + "Le porte devono essere comprese tra 1024 e 65535.");
        System.exit(-1);
    }
    index++;
}

// check duplicati tra porte e files salvati
if (hasDuplicatedFileNames(files)) {
    System.out.println("File duplicati trovati");
    System.exit(-1);
}

if (hasDuplicatedPorts(ports)) {
    System.out.println("Porte duplicate trovate");
    System.exit(-1);
}
```

3

```
// init dei RowSwapServer
index = 0;

for (int i = 1; i < args.length; i += 2) {
    File file = new File(args[i]);

    if (!file.exists() || file.isDirectory()) {
        System.out.println("Il file "
            + args[i]
            + " non è stato trovato oppure è una directory.");
        System.exit(-1);
    }

    new RowSwapServer(file, Integer.parseInt(args[i + 1])).start();
    index++;
}
```

```
String nomeFile = null;
StringTokenizer tokenizer = null;
// attesa di richieste dai Client
try {
    while (true) {
        buf = new byte[256];
        System.out.println("\n[DiscoveryServer] In attesa di richieste...");

        packet.setData(buf);
        socket.receive(packet);
        System.out.println("[DiscoveryServer] Richiesta ricevuta");

        // ricezione richiesta dal client del file x
        tokenizer =
            new StringTokenizer(ByteUtility.bytesToStringUTF(packet.getData()));
        nomeFile = tokenizer.nextToken();
        System.out.println("[DiscoveryServer] Richiesto file "
            + nomeFile);

        // ricerca della porta corrispondente al file richiesto
        int foundPort = -1;
        for (int i = 0; i < files.length && files[i] != null; i++) {
            if (files[i].equals(nomeFile)) {
                foundPort = ports[i];
            }
        }

        // invio della porta trovata del RowSwapServer al client
        // se non è stata trovata sarà inviato -1
        packet.setData(ByteUtility.intToBytes(foundPort));
        socket.send(packet);
    }
} catch (IOException e) {
    System.err.println("[DiscoveryServer] Problemi nella lettura: "
        + e.getMessage());
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}

System.out.println("[DiscoveryServer] termina...");
socket.close();
```

Metodi e classi di utility

Check delle porte e file duplicati

```
private static boolean hasDuplicatedFileNames(String[] fileNames) {
    if (fileNames.length <= 1) {
        return false;
    }

    for (int i = 0; i < fileNames.length && fileNames[i] != null; i++) {
        for (int j = i + 1; j < fileNames.length && fileNames[i] != null; j++) {
            if (fileNames[i].equals(fileNames[j])) {
                return true;
            }
        }
    }

    return false;
}

private static boolean hasDuplicatedPorts(int[] ports) {
    if (ports.length <= 1) {
        return false;
    }

    for (int i = 0; i < ports.length && ports[i] != 0; i++) {
        for (int j = i + 1; j < ports.length && ports[j] != 0; j++) {
            if (ports[i] == ports[j]) {
                return true;
            }
        }
    }

    return false;
}
```

Classe ByteUtility

```
public class ByteUtility {
    public static int bytesToInt(byte[] bytes) throws IOException {
        ByteArrayInputStream byteInputStream = new ByteArrayInputStream(bytes);
        DataInputStream dataInputStream = new DataInputStream(byteInputStream);
        return dataInputStream.readInt();
    }

    public static byte[] intToBytes(int integer) throws IOException {
        ByteArrayOutputStream byteOutputStream = new ByteArrayOutputStream();
        DataOutputStream dataOutputStream = new DataOutputStream(byteOutputStream);
        dataOutputStream.writeInt(integer);
        return byteOutputStream.toByteArray();
    }

    public static String bytesToStringUTF (byte[] bytes) throws IOException {
        ByteArrayInputStream byteInputStream = new ByteArrayInputStream(bytes);
        DataInputStream dataInputStream = new DataInputStream(byteInputStream);
        return dataInputStream.readUTF();
    }

    public static byte[] stringUTFToBytes (String string) throws IOException {
        ByteArrayOutputStream byteOutputStream = new ByteArrayOutputStream();
        DataOutputStream dataOutputStream = new DataOutputStream(byteOutputStream);
        dataOutputStream.writeUTF(string);
        return byteOutputStream.toByteArray();
    }
}
```

Implementazione JAVA - RS

```
int aLineIndex = -1;
int bLineIndex = -1;
StringTokenizer tokenizer = null;
// attesa di richieste
try {
    while (true) {
        buf = new byte[256];
        System.out.println("[RowSwapServer-"
                            + port
                            + "] In attesa di richieste ... ");

        packet.setData(buf);
        socket.receive(packet);

        // ricezione richiesta con le righe da swappare
        tokenizer = new StringTokenizer(ByteUtility.bytesToStringUTF(packet.getData()), "-");
        aLineIndex = Integer.parseInt(tokenizer.nextToken());
        bLineIndex = Integer.parseInt(tokenizer.nextToken());

        System.out.println("[RowSwapServer-"
                            + port
                            + "] Ricevuta richiesta di swap di righe: swapping riga "
                            + aLineIndex + " con riga " + bLineIndex);

        // swap delle righe
        int status = LineUtility.swapLines(this.file, aLineIndex, bLineIndex);
        System.out.println("[RowSwapServer-"
                            + port
                            + "] Esito swap: " + (status < 0 ? "errore" : "successo")
                            + ". Invio risposta al client..");

        packet.setData(ByteUtility.intToBytes(status));
        socket.send(packet);
    }
} catch (IOException e) {
    System.err.println("[RowSwapServer-"
                        + port
                        + "] Problemi nella lettura: "
                        + e.getMessage());

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

System.out.println("[RowSwapServer-" + port + "] termina..");
socket.close();
```


Classe LineUtility

1

```
static final int EXCEPTION_THROWN = -1;
static final int SAME_INDEX = -2;
static final int LINE_NOT_FOUND = -3;

static int swapLines(File file, int aIndex, int bIndex) {
    if (aIndex == bIndex) {
        return SAME_INDEX;
    }

    int i = 1;
    String a = null;
    String b = null;
    String line = null;

    try {
        BufferedReader reader = new BufferedReader(new FileReader(file));

        while (((line = reader.readLine()) != null) && (a == null || b == null)) {
            if (i == aIndex) {
                a = line;
            } else if (i == bIndex) {
                b = line;
            }
            i++;
        }

        reader.close();

        if (a == null || b == null) {
            return LINE_NOT_FOUND;
        }
    }
```

2

```
        reader = new BufferedReader(new FileReader(file));
        StringBuilder builder = new StringBuilder();
        while ((line = reader.readLine()) != null) {
            if (line.equals(a)) {
                builder.append(b);
            } else if (line.equals(b)) {
                builder.append(a);
            } else {
                builder.append(line);
            }
            builder.append(System.lineSeparator());
        }
        reader.close();

        BufferedWriter writer = new BufferedWriter(new FileWriter(file));
        writer.write(builder.toString());
        writer.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
        return EXCEPTION_THROWN;
    } catch (IOException e) {
        e.printStackTrace();
        return EXCEPTION_THROWN;
    }

    return 0;
}
```

Implementazione Client

```
1 // controllo args e salvataggio parametri
try {
    if (args.length != 3) {
        System.out.println("Usage: java Client serverIP serverPort fileName");
        System.exit(-1);
    }

    addr = InetAddress.getBy_name(args[0]);
    port = Integer.parseInt(args[1]);
    fileName = args[2];
} catch (UnknownHostException e) {
    System.out
        .println("Problemi nella determinazione dell'endpoint del server : ");
    e.printStackTrace();
    System.out.println("Client: interrompo ... ");
    System.exit(-1);
}

2 // invio richiesta al DiscoveryServer con il filename
try {
    packet.setData(ByteUtility.stringUTFToBytes(fileName));
    socket.send(packet);
    System.out.println("Richiesta inviata al DiscoveryServer " + addr + ", " + port);
} catch (IOException e) {
    System.out.println("Problemi nell'invio della richiesta: ");
    e.printStackTrace();
    System.exit(-1);
}

3 int rowSwapServerPort = -1;
// ricezione risposta dal DiscoveryServer con l'eventuale porta del corrispondente RowSwapServer
try {
    rowSwapServerPort = ByteUtility.bytesToInt(packet.getData());

    if (rowSwapServerPort < 0) {
        System.out.println("Il server non ha trovato il file specificato..");
        System.exit(-1);
    }

    System.out.println("RowSwapServer trovato alla porta " + rowSwapServerPort + "!");
} catch (IOException e) {
    System.out.println("Problemi nella lettura della risposta: ");
    e.printStackTrace();
    System.exit(-1);
}
```

```
4 // finita l'inizializzazione leggiamo l'input dall'utente per le righe da swappare
// da inviare al RowSwapServer designato
BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
String readInput = null;
// matcha solo stringhe con questo formato: '22-5'
Pattern patternChecker = Pattern.compile("^\\d+(-\\d+)$");

System.out.println("^D(Unix)/^Z(Win)+invio per uscire.");
System.out.println("Inserisci righe da scambiare (separate da '-'): ");
try {
    while ((readInput = stdIn.readLine()) != null) {
        // check dell'input dell'utente
        if (!patternChecker.matcher(readInput).matches()) {
            System.out.println("Gli indici devono essere numerici e devono essere separati da un solo '-!');
            System.out.println("Inserisci righe da scambiare (separate da '-'): ");
        } else {
            // invio richiesta al RowSwapServer con le righe da swappare
            packet.setPort(rowSwapServerPort);
            packet.setData(ByteUtility.stringUTFToBytes(readInput));
            socket.send(packet);
            System.out.println("Richiesta inviata al server [RowSwap-"
                                + rowSwapServerPort + "]);

            packet.setData(buf);
            socket.receive(packet);

            // ricezione esito dello swap dal RowSwapServer
            int requestStatus = ByteUtility.bytesToInt(packet.getData());
            System.out.println("Il server ha risposto con un codice di "
                                + (requestStatus < 0 ? "errore" : "successo"));
            System.out.println("Inserisci righe da scambiare (separate da '-'): ");
        }
    }
} catch (IOException e) {
    System.out.println("Problemi nella lettura / invio della richiesta: ");
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}

System.out.println("Client: termino ... ");
socket.close();
```


[FINE]

<https://github.com/ilcors-dev/gruppo-reti>

Gestione progetto

Coding



git



Team

