Sockets in Java

Lorenzo Gallucci

Sockets

- La libreria Java dispone di un'API per la gestione dell'I/O di rete
- Il package di riferimento è java.net
- Vari compiti:
  - Gestione dell'I/O su socket (sia TCP che UDP)
  - Gestione di un sottoinsieme delle funzionalità protocollo HTTP
  - Gestione di URL e URI
- In questo corso, siamo maggiormente interessati all'I/O su socket
  - → si basa sull'I/O su stream visto la volta scorsa!

Sockets

- Classi principali per l'I/O su socket in modalità TCP:
  - ServerSocket: consente di gestire un punto d'accesso per i client
    - Ogni connessione genera un socket, sia lato client, che lato server
  - Socket: incapsula il concetto di connessione bidirezionale, associato ad un socket TCP
    - Può essere creato lato server in risposta ad un tentativo di connessione
    - Può essere creato a richiesta, lato client

Classe ServerSocket

- Utilizzata tipicamente nel codice che esegue sulla macchina che ha il ruolo di “server”
- Alcuni dei metodi disponibili in ServerSocket sono:
  - ServerSocket(int port) → costruttore che inizializza un “server socket” in ascolto sulla specifica porta TCP
  - Socket accept() → accetta la prossima richiesta di connessione, generando un socket lato server (se necessario, attende l’arrivo di una richiesta → OPERAZIONE BLOCCANTE!)
  - void close() → chiude il server socket

Classe ServerSocket

- Ve ne sono molti altri, che consentono di personalizzare il server socket quanto a:
  - Prestazioni
  - Modalità di bufferizzazione
  - Riutilizzo di server socket esistenti
- Al solito:
  - TUTTI i metodi possono lanciare una IOException!!

Classe Socket

- Utilizzata nel codice che esegue sulla macchina che ha il ruolo di “server”, ma anche in quella che il ruolo di “client”
- Alcuni dei metodi disponibili in Socket sono:
  - Socket(String host, int port) → costruttore che inizializza un socket TCP connesso alla porta ‘port’ del server ‘host’ (→ utilizzato nel client)
  - InputStream getInputStream() → apre uno stream di input che legge dati dal socket
  - OutputStream getOutputStream() → apre uno stream di output che scrive dati sual socket
  - void close() → chiude il socket

Classe Socket

- Ve ne sono molti altri, che consentono di personalizzare un socket quanto a:
  - Prestazioni (urgent data, traffic class, ecc.)
  - Modalità di bufferizzazione
  - Riutilizzo di oggetti Socket esistenti
- Al solito:
  - TUTTI i metodi possono lanciare una IOException!!

Utilizzo di socket e server socket

- Tipicamente, un ServerSocket (usato sul lato server) viene:
  - Inizializzato con una specifica porta
  - Utilizzato in un ciclo, attivo finchè è attivo il server, che richiama il metodo accept()
  - Al socket restituito da quest'ultimo viene richiesto di creare due stream, uno di input, l'altro di output, per interagire con il client appena connesso
  - Al termine, il socket generato può venir chiuso (con close()) e si ritorna al ciclo di accept()

Utilizzo di socket e server socket

- Lato client:
  - Si inizializza un Socket indicando una specifica porta ed un indirizzo (del server)
  - Al socket così ottenuto viene richiesto di creare due stream, uno di input, l'altro di output, per interagire con il server appena connesso
  - Al termine, il socket generato può venir chiuso (con close()) e il client può proseguire con altre operazioni

Esempio con i Socket

- Fattorizziamo le funzionalità di I/O dall'esempio dell'altra volta:

```
package sistemidielaborazioneinrete.io;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class Utilità
{
   static void scriviStringa(String stringaDaScrivere, OutputStream outputStream) throws
     IOException
   {
      DataOutputStream dataOutputStream = new DataOutputStream(outputStream);
      dataOutputStream.writeUTF(stringaDaScrivere);
   }
```

Esempio con i Socket

- (continua)

```
static String leggiStringa(InputStream inputStream) throws IOException
   {
      DataInputStream dataInputStream = new DataInputStream(inputStream);
      String stringaLetta = dataInputStream.readUTF();
      return stringaLetta;
   }
}
```

Esempio con i Socket

- Dall'esempio della volta scorsa, l'I/O in memoria si può fare così:

```
private static void leggiEScriviInMemoria(String stringaDaScrivere) throws
   IOException
{
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    Utilità.scriviStringa(stringaDaScrivere, baos);

    ByteArrayInputStream bais = new ByteArrayInputStream(baos.toByteArray());
    String readUTF = Utilità.leggiStringa(bais);

    System.out.println(readUTF);
}
```

Esempio con i Socket

- Volendo invece scrivere su un socket:

```
private static void leggiEScriviAttraversoUnSocket(String stringaDaScrivere, int
  porta) throws IOException
{
    // Assicurati che la prima connessione sia subito accettata
    final Server server = new Server(porta);
    server.start();

    Socket socket = new Socket("localhost", porta);
    OutputStream outputStream = socket.getOutputStream();
    Utilità.scriviStringa(stringaDaScrivere, outputStream);
}
```

- Si noti la dipendenza da una classe “Server”, che implementa lo schema cui si è accennato in precedenza

Esempio con i Socket

- La classe Server è dunque

```
package sistemidielaborazioneinrete.io;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Server extends Thread
{
   private final ServerSocket serverSocket;
   private final int          port;

   public Server(int port) throws IOException
   {
      this.port = port;
      this.serverSocket = new ServerSocket(port);
      setDaemon(true);
   }
```

Esempio con i Socket

- (continua)

```
@Override
   public void run()
   {
      try
      {
        while (true)
        {
           Socket socket = serverSocket.accept();
           String readUTF = Utilità.leggiStringa(socket.getInputStream());
           System.out.println(readUTF + " ( attraverso il socket " + port + ")");
        }
      }
      catch (IOException e)
      {
         throw new RuntimeException("Impossibile accettare la connessione", e);
      }
   }
}
```