Corso di Reti di Calcolatori

UNICAL – Facoltà di Ingegneria – a.a. 2002/2003

Esercitazione sul networking in Java (1ª parte)

paolo.trunfio@deis.unical.it

java.net.InetAddress

- static InetAddress getByName(String host)
  restituisce l'indirizzo IP di un dato host
- static InetAddress[] getAllByName(String host)
  restituisce tutti gli indirizzi IP di un dato host
- static InetAddress getLocalHost()
  restituisce l'indirizzo IP di localhost
- byte[] getAddress()
  restituisce l'indirizzo IP sotto forma array di byte
- String getHostAddress()
  restituisce l'indirizzo IP sotto forma di stringa (p.es. "192.168.35.75")
- String getHostName()
  restituisce l'host name associato all'indirizzo IP

Lookup di indirizzi locali e remoti

```
static void printLocalAddress () {
    try {
      InetAddress myself = InetAddress.getLocalHost ();
      System.out.println ("My name : " + myself.getHostName ());
      System.out.println ("My IP : " + myself.getHostAddress ());
    } catch (UnknownHostException ex) {
      System.out.println ("Failed to find myself:");
    }
}

static void printRemoteAddress (String name) {
    try {
      InetAddress machine = InetAddress.getByName (name);
      System.out.println ("Host name : " + machine.getHostName ());
      System.out.println ("Host IP : " + machine.getHostAddress ());
    } catch (UnknownHostException ex) {
      System.out.println ("Failed to lookup " + name);
    }
}
```

java.net.Socket

- Socket(String host, int port)
- Socket(InetAddress address, int port)
- void close()
- InetAddress getInetAddress()
  indirizzo a cui è connesso il socket
- int getPort()
  porta remota a cui è connesso il socket
- int getLocalPort()
  porta locale del socket
- InputStream getInputStream()
- OutputStream getOutputStream()
- void setSoTimeout(int timeout)
  timeout per operazioni di lettura dal socket. Genera InterruptedIOException
- String toString() restituisce una rappresentazione del socket del tipo:
  “Socket[addr=hostname/192.168.35.75,port=3575,localport=1026]”

Un semplice client

```
import java.io.*;
import java.net.*;
public class ClientTest {
  public static void main(String[] args) {
    try {
      Socket s = new Socket("www.deis.unical.it",80);
      PrintWriter out = new PrintWriter (s.getOutputStream(),true);
      BufferedReader in =
        new BufferedReader (new InputStreamReader(s.getInputStream()));
      out.println ("GET / HTTP/1.0"); out.println ();
      boolean more = true;
      while (more) {
        String line = in.readLine();
        if (line == null) more = false;
        else System.out.println(line);
      }
    } catch (IOException e) { System.out.println("Error"+e); }
  }
}
```

Timeout di socket

```
Socket s = new Socket (…);
s.setSoTimeout(10000);
```

Le operazioni di lettura che seguono generano una InterruptedIOException quando è stato raggiunto il timeout.

```
try {
  String line;
  while (line = in.readLine()) != null) {
     process line
  }
} catch (InterrupedIOException e)
{
  react to timeout
}
```

Questo timeout può essere settato su un socket già istanziato. Tuttavia si potrebbe verificare un blocco indefinito già in fase di creazione del socket, fino a quando non si stabilisce la connessione con l'host.

SocketOpener (1)

```
import java.io.*;
import java.net.*;

public class SocketOpenerTest {
  public static void main(String[] args) {
    String host = "www.deis.unical.it";
    int port = 80;
    int timeout = 10000;
    Socket s = SocketOpener.openSocket(host, port, timeout);
    if (s == null)
      System.out.println("The socket could not be opened.");
    else
      System.out.println(s);
   }
}

class SocketOpener extends Thread {
  private String host;
  private int port;
  private Socket socket;
```

SocketOpener (2)

```
  public static Socket openSocket (String host, int port, int timeout) {
    SocketOpener opener = new SocketOpener(host, port);
    opener.start();
    try {
      opener.join(timeout);
    } catch (InterruptedException e) { System.err.println (e); }
    return opener.getSocket();
  }

  public SocketOpener(String host, int port) {
    this.host = host; this.port = port; socket = null;
  }

  public Socket getSocket() { return socket; }

  public void run() {
    try {
      socket = new Socket(host, port);
    } catch (IOException e) { System.err.println (e); }
  }
}
```

java.net.ServerSocket

- ServerSocket(int port)
- Socket accept()
  rimane in attesa di una connessione, e restituisce un socket tramite il quale si effettua la comunicazione
- void close()
- InetAddress getInetAddress()
  indirizzo locale di questo server socket
- int getLocalPort()
  porta locale di questo server socket

EchoServer (1)

```
import java.io.*;
import java.net.*;

public class EchoServer {

  public static void main(String[] args ) {
    try {
      ServerSocket s = new ServerSocket(8189);
      Socket incoming = s.accept( );
      BufferedReader in = new BufferedReader
        (new InputStreamReader(incoming.getInputStream()));
      PrintWriter out = new PrintWriter
        (incoming.getOutputStream(), true /* autoFlush */);
      out.println( "Hello! Enter BYE to exit." );
```

EchoServer (2)

```
      boolean done = false;
      while (!done) {
        String line = in.readLine();
        if (line == null)
          done = true;
        else {
           out.println("Echo: " + line);
           if (line.trim().equals("BYE"))
              done = true;
        }
      } // while
      incoming.close();
    } catch (Exception e) { System.err.println(e); }
  } // main

} // class
```

Gestione di connessioni multiple

```
while (true) {
  Socket incoming = s.accept();
  Thread t = new ThreadedEchoHandler(incoming);
  t.start();
}

class ThreadedEchoHandler extends Thread {
 …
 public void run () {
  try {
    BufferedReader in = new BufferedReader
      (new InputStreamReader(incoming.getInputStream()));
    PrintWriter out = new PrintWriter (incoming.getOutputStream(), true);
    String line;
    while ((line=in.readLine()) != null) {
      process line
    }
    incoming.close();
  } catch (Exception e) { handle exception }
 }}
```

Trasmissione di oggetti serializzati (1)

```
import java.io.*;

public class Studente implements Serializable {

  private int matricola;

  private String nome, cognome, corsoDiLaurea;

  public Studente (int matricola, String nome, String cognome,
                   String corsoDiLaurea) {
    this.matricola = matricola; this.nome = nome;
    this.cognome = cognome; this.corsoDiLaurea = corsoDiLaurea;
  }

  public int getMatricola () { return matricola; }

  public String getNome () { return nome; }

  public String getCognome () { return cognome; }

  public String getCorsoDiLaurea () { return corsoDiLaurea; }

}
```

Trasmissione di oggetti serializzati (2)

```
import java.io.*;
import java.net.*;
public class SendObject {
  public static void main (String args[]) {
    try {
      ServerSocket server = new ServerSocket (3575);
      Socket client = server.accept();
      ObjectOutputStream output =
        new ObjectOutputStream (client.getOutputStream ());
      output.writeObject("<Welcome>");
      Studente studente =
        new Studente (14520,"Leonardo","da Vinci","Ingegneria Informatica");
      output.writeObject(studente);
      output.writeObject("<Goodbye>");
      client.close();
      server.close();
    } catch (Exception e) { System.err.println (e); }
  }
}
```

Trasmissione di oggetti serializzati (3)

```
import java.io.*;
import java.net.*;
public class ReceiveObject {
  public static void main (String args[]) {
    try {
      Socket socket = new Socket ("localhost",3575);
      ObjectInputStream input =
        new ObjectInputStream (socket.getInputStream ());
      String beginMessage = (String)input.readObject();
      System.out.println (beginMessage);
      Studente studente = (Studente)input.readObject();
      System.out.print (studente.getMatricola()+" - ");
      System.out.print (studente.getNome()+" "+studente.getCognome()+" - ");
      System.out.print (studente.getCorsoDiLaurea()+"\n");
      String endMessage = (String)input.readObject();
      System.out.println (endMessage);
      socket.close();
    } catch (Exception e) { System.err.println (e); }
  } }
```

HttpWelcome (1)

```
import java.io.*;
import java.net.*;
import java.util.*;

public class HttpWelcome {

  private static int port = 80;

  private static String HtmlWelcomeMessage () {
    return "<html>\n"+
           "  <head>\n"+
           "    <title>UNICAL - Facoltà di Ingegneria</title>\n"+
           "  </head>\n"+
           "  <body>\n"+
           "    <h2 align=\"center\">\n"+
           "      <font color=\"#0000FF\">Benvenuti al Corso di"+
           " Reti di Calcolatori</font>\n"+
           "    </h2>\n"+
           "  </body>\n"+
           "</html>";
  }
```

HttpWelcome (2)

```
public static void main (String args[]) {
  try {
    ServerSocket server = new ServerSocket(port);
    System.out.println("HTTP server running on port: "+port);
    while (true) {
      Socket client = server.accept();
      BufferedReader in = new BufferedReader
        (new InputStreamReader(client.getInputStream()));
      PrintWriter out = new PrintWriter
        (new OutputStreamWriter(client.getOutputStream()));
      String request = in.readLine();
      System.out.println("Request: "+request);
      StringTokenizer st = new StringTokenizer(request);
```

HttpWelcome (3)

```
      if ((st.countTokens()>=2) && st.nextToken().equals("GET")) {
        String message = HtmlWelcomeMessage();
        // Start of response headers
        out.println ("HTTP/1.0 200 OK");
        out.println ("Content-Length: "+message.length());
        out.println ("Content-Type: text/html");
        out.println ();
        // End of response headers
        out.println (message);
      } else {
        out.println("400 Bad Request");
      }
      out.flush();
      client.close();
    }
   } catch (Exception e) { System.err.println(e); }
  } // main
} // class
```

HttpServer (1)

```
import java.io.*;
import java.net.*;
import java.util.*;

public class HttpServer {
  private static final int port = 3575;
  public static void main (String args[]) throws IOException {
    try {
      ServerSocket server = new ServerSocket(port);
      System.out.println("HTTP server running on port: "+port);
      while(true) {
        Socket client = server.accept();
        ThreadedServer cc = new ThreadedServer(client);
      }
    } catch(IOException e) {
      System.err.println(e);
    }
  }
}
```

HttpServer (2)

```
class ThreadedServer extends Thread {
  private Socket client;
  private BufferedReader is;
  private DataOutputStream os;
  public ThreadedServer (Socket s) {
    client = s;
    try {
      is = new BufferedReader
        (new InputStreamReader(client.getInputStream()));
      os = new DataOutputStream (client.getOutputStream());
    } catch (IOException e) {
      try {
        client.close();
      } catch (IOException ex) { System.err.println(""+ex); }
      return;
    }
    this.start();
  }
```

HttpServer (3)

```
public void run() {
  try { // get a request and parse it.
    String request = is.readLine();
    System.out.println("Request: "+request);
    StringTokenizer st = new StringTokenizer(request);
    if ((st.countTokens()>=2) && st.nextToken().equals("GET")) {
      if ((request = st.nextToken()).startsWith("/"))
        request = request.substring( 1 );
      if (request.endsWith("/") || request.equals(""))
        request = request + "index.html";
      //   per impedire che si possa scrivere: http://hostname/../../../etc/passwd
      //   oppure: http://hostname//etc/passwd
      if ((request.indexOf("..") != -1) || (request.startsWith("/"))) {
        os.writeBytes("403 Forbidden. "+
        "You do not have enough privileges to read: "+request+"\r\n");
      } else {
        File f = new File(request); reply (os, f);
      }
    }
```

HttpServer (4)

```
      else {
        os.writeBytes("400 Bad Request\r\n");
      }
      client.close();
    } catch (IOException e1) { System.err.println("I/O error: "+e1); }
      catch (Exception e2) { System.err.println("Exception: "+e2); }
  }

  public static void reply (DataOutputStream out, File f) throws Exception {
    try {
      DataInputStream in = new DataInputStream(new FileInputStream(f));
      int len = (int)f.length(); byte buf[] = new byte[len];
      in.readFully(buf);
      out.writeBytes("HTTP/1.0 200 OK\r\n");
      out.writeBytes("Content-Length: "+buf.length+"\r\n");
      out.writeBytes("Content-Type: text/html\r\n\r\n");
      out.write(buf); out.flush(); in.close();
    } catch (FileNotFoundException e) { out.writeBytes("404 Not Found\r\n"); }
  }
} // class ThreadedServer
```

MailSender (1)

```
import java.net.*;
import java.io.*;
public class MailSender {
  private static PrintWriter out; private static BufferedReader in;
  public static void main(String[] args) {
    try {
      BufferedReader console =
        new BufferedReader(new InputStreamReader(System.in));
      System.out.print ("SMTP server: "); String smtp = console.readLine ();
      System.out.print ("From: "); String from = console.readLine ();
      System.out.print ("To: "); String to = console.readLine ();
      System.out.println ("Message: "); String message = "";
      boolean flag = true;
      do {
        String line = console.readLine ();
        if (line.length() > 0 && Character.getType(line.charAt(0)) ==
            Character.CONTROL) flag = false;
        else message += line+"\n";
      } while (flag);
```

MailSender (2)

```
    Socket s = new Socket(smtp, 25);
    out = new PrintWriter(s.getOutputStream());
    in = new BufferedReader(new InputStreamReader(s.getInputStream()));
    String hostName = InetAddress.getLocalHost().getHostName();
    send (null);
    send ("HELO "+hostName);
    send ("MAIL FROM: "+from);
    send ("RCPT TO: "+to);
    send ("DATA");
    out.println (message);
    send ("."); s.close ();
   } catch (Exception e) {  System.err.println (e); }
  }
  public static void send (String s) throws IOException {
    if (s != null) { System.out.println(">> "+s); out.println(s); out.flush(); }
    String line = in.readLine();
    if (line != null) System.out.println("<< "+line+"\n");
  }
} // class
```