Tecnologie Web L-A

Java e HTTP

Dario Bottazzi
Tel. 051 2093541,
E-Mail: dario.bottazzi@unibo.it,
SkypeID: dariobottazzi

Java e TCP/IP in a Nutshell

- java.net.InetAddress: rappresenta un indirizzo IP e consente di richiedere un indirizzo IP al DNS.
- java.net.Socket: rappresenta una socket TCP
- java.net.ServerSocket: rappresenta un server di socket in grado di rimanere in attesa di connessioni TCP da parte dei client

DNS Lookup

```
import java.net.*;

public class DomainName2IPNumbers {
  public static void main(String[] args) {
    try {
      InetAddress[] a = InetAddress.getAllByName(args[0]);
      for (int i = 0; i<a.length; i++)
        System.out.println(a[i].getHostAddress());
    } catch (UnknownHostException e) {
      System.out.println("Unknown host!");
    }
  }
}
```

Array con tutti gli indirizzi IP

Chiedi tutti gli indirizzo di tutti gli host che hanno il nome specificato

Stringa che rappresenta l'indirizzo IP

Raccogli l'eccezione se l'host non è stato trovato

http://java.sun.com/j2se/1.4.2/docs/api/java/net/InetAddress.html

TCP/IP: SimpleServer (1/2)

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

public class SimpleServer {
  public static void main(String[] args) {
    try {
      ServerSocket ss = new ServerSocket(Integer.parseInt(args[0]));
      while (true) {
        Socket con = ss.accept();
        InputStreamReader in =
          new InputStreamReader(con.getInputStream());
```

Crea una server socket in ascolto alla porta specificata da linea di comando

Rimane in attesa di una connessione. Quando arriva una connessione la accetta

Crea un InputStreamReader che riceve in ingresso l'InputStream della socket

L'InputStreamReader trasforma uno stream di byte in uno stream di caratteri usando una codifica specificata

TCP/IP: SimpleServer (2/2)

```
      StringBuffer msg = new StringBuffer();
       int c;
       while ((c = in.read())!=0)
         msg.append((char)c);
       PrintWriter out =
         new PrintWriter(con.getOutputStream());
       out.print("Simon says: "+msg);
       out.flush();
       con.close();
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Crea un buffer

Metti nel buffer tutti i caratteri letti fino allo 0

Crea una PrintWriter che prende in ingresso l'output stream della socket

Spedisci al client la risposta

Svuota il buffer, chiudi la connessione e passa alla nuova iterazione del ciclo while

TCP/IP: SimpleClient (1/2)

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

public class SimpleClient {
  public static void main(String[] args) {
    try {
      Socket con = new Socket(args[0], Integer.parseInt(args[1]));
      PrintStream out =
        new PrintStream(con.getOutputStream());
      out.print(args[2]);
      out.write(0);
      out.flush();
```

Crea una Socket indirizzata all'host specificato da arg[0] alla porta specificata da arg[1]

Crea un PrintStream che riceve in ingresso l'output stream della socket

Spedisci il carattere 0 e svuota il buffer

Spedisci la stringa specificata da arg[2]

TCP/IP: SimpleClient (2/2)

```
    InputStreamReader in =
      new InputStreamReader(con.getInputStream());
     int c;
     while ((c = in.read())!=-1)
       System.out.print((char)c);
     con.close();
    } catch (IOException e) {
     e.printStackTrace();
    }
  }
}
```

Crea un InputStreamReader che riceve in ingresso l'input stream della socket

Leggi tutti i caratteri fino alla fine dello stream (ovvero il "*carattere*" -1)

Chiudi la socket

Proviamo…

```
java SimpleServer 1234
```

```
java SimpleClient localhost 1234 "Hello World"

Simon says: Hello World
```

HTTP in Java

- Due approcci possibili
  - Si possono utilizzare le socket TCP e implementare a mano il protocollo HTTP. Scelta fortemente sconsigliata.
  - **Si possono usare** le classi di **Java** che consentono di lavorare con **HTTP**

HttpURLConnection

Rappresenta una connessione HTTP verso un host specificato

- setRequestMethod: imposta il metodo di request, GET (default) o POST
- setRequestProperty: imposta una coppia nome valore come proprietà della richiesta
- setDoInput: se è true (default) leggiamo dalla connessione
- setDoOutput: deve valere true se vogliamo scrivere output sulla connessione, tipicamente per richieste POST. Il valore di default è false

HttpURLConnection

- connect: stabilisce la connessione TCP. Non si usa praticamente mai dato che la connessione viene stabilita non appena si tenta di scrivere il corpo della richiesta o di leggere la risposta
- getOutputStream: fornisce uno stream di output per il corpo della richiesta POST
- getResponseCode: restituisce lo status code della risposta (es. 200 per OK)
- getHeaderField: restituisce un campo header della risposta
- getInputStream: fornisce uno stream di input per leggere il corpo della risposta

HttpURLConnection

- setInstanceFollowRedirects: per default le redirezioni sono abilitate e vengono gestite “automaticamente”. Questa caratteristica può essere disabilitata
- setUse-Caches: per default si memorizzano le risposte in una cache (vedi le slide sul caching). Questo comportamento può essere disabilitato.
- setAllowUserInteraction: se abilitato permette di gestire l’autenticazione tramite la classe Authenticator (vedi javadoc)
- getContent: a seconda della risorsa richiesta, il metodo permette di convertirla “automaticamente” in un oggetto java di tipo appropriato (vedi javadoc)

Esempio Google I'm Feeling Lucky

- Richiedo a Google una pagina che risponde ai determinati criteri di ricerca in modalità "I'm Feeling Lucky"
- Google mi risponde con un messaggio che mi redireziona alla pagina cui assegna massimo rank fra quelle che verificano i criteri di ricerca.
- La redirezione richiede che venga messo il riferimento alla pagina nello header del messaggio di response
- Questo semplice esempio **disabilita** la **redirezione** e **legge** la **locazione** a **cui sarei rediretto**

Come Faccio a Sentirmi Fortunato?

http://www.google.com/search?q=web&btnI=Mi+sento+fortunato

q=keyword di ricerca in URL encode

Voglio la modalità I'm feeling lucky

Provate a fare copia e incolla sul vostro browser e vedrete che funziona….

Esempio Google I'm Feeling Lucky (1/2)

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

public class ImFeelingLucky2 {
  public static void main(String[] args) {
    try {
      String req = "http://www.google.com/search?"+
        "q="+URLEncoder.encode(args[0], "UTF8")+"&"+
        "btnI="+URLEncoder.encode("I'm Feeling Lucky", "UTF8");
      HttpURLConnection con =
        (HttpURLConnection) (new URL(req)).openConnection();
      con.setRequestProperty("User-Agent", "IXWT");
      con.setInstanceFollowRedirects(false);
```

Prepara una stringa come spiegato alla slide precedente

Apri una connessione HTTP alla URL specificata

Setta lo User Agent

Disabilita le redirezioni automatiche

Esempio Google I'm Feeling Lucky (2/2)

```
      String loc = con.getHeaderField("Location");
      System.out.print("The prophet spoke thus: ");
      if (loc!=null)
        System.out.println("Direct your browser to "+loc+
          " and you shall find great happiness in life.");
      else
        System.out.println("I am sorry - my crystal ball is blank.");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Restituisce il valore del campo Location dell'header del response. La stringa è null se non ho trovato nulla