Java API per URL

Laboratorio di Programmazione di Rete B

Daniele Sgandurra

Università di Pisa

08/10/2009

URL e HTTPS

Che Cos'è un URL

URL è un acronimo per Uniform Resource Locator:

- Un riferimento (un indirizzo) per una risorsa su Internet.
- Di solito un URL è il nome di un file su un host.
- Ma può anche puntare ad altre risorse:
  - una query per un database;
  - l'output di un comando.
- Es: `http://java.sun.com`
  1. `http`: identificativo del protocollo.
  2. `java.sun.com`: nome della risorsa.

Resource Name

Il nome di una risorsa è composto da:

- Host Name: il nome dell'host su cui si trova la risorsa.
- Filename: il pathname del file sull'host.
- Port Number: il numero della porta su cui connettersi (opzionale).
- Reference: un riferimento ad una specifica locazione all'interno del file (opzionale).

Nel caso del protocollo `http`, se il Filename è omesso (o finisce per `/`), il web server è configurato per restituire un file di default all'interno del path (ad es. `index.html`, `index.php`, `index.asp`).

URL e URI

- Un URI (Uniform Resource Identifier) è un costrutto sintattico che specifica, tramite le varie parti che lo compongono, una risorsa su Internet:
  - `[schema:]ParteSpecificaSchema[#frammento]`
  - dove `ParteSpecificaSchema` ha la struttura: `[//autorita']percorso][?query]`
- Un URL è un tipo particolare di URI: contiene sufficienti informazioni per individuare e ottenere una risorsa.
- Altre URI, ad es: `URN:ISBN:0-395-36341-1` non specificano come individuare la risorsa. In questo caso, le URI sono dette URN (Uniform Resource Name).

URL in JAVA

- In JAVA per creare un oggetto URL:

```
URL cli = new URL("http://www.cli.di.unipi.it/");
```

  che è un esempio di un URL assoluto.

- È anche possibile creare un URL relativo, che ha la forma `URL(URL baseURL, String relativeURL)`, ad es.

```
URL cli = new URL("http://www.cli.di.unipi.it/");
URL faq = new URL(cli, "faq");
```

  che risulterà puntare a `http://www.cli.di.unipi.it/faq`

- I protocolli gestiti da Java con gli URL sono `http`, `https`, `ftp`, `file` e `jar`.

- I costruttori possono lanciare una `MalformedURLException`.

Parsare un URL

Sono presenti metodi per ottenere informazioni sull'URL:

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

public class URLReader
{
   public static void main(String[] args) throws Exception
   {
      String url = "http://www.cli.di.unipi.it:80/faq";
      URL cli = new URL(url);

      System.out.println("protocol = " + cli.getProtocol());
      System.out.println("authority = " + cli.getAuthority());
      System.out.println("host = " + cli.getHost());
      System.out.println("port = " + cli.getPort());
      System.out.println("path = " + cli.getPath());
      System.out.println("query = " + cli.getQuery());
      System.out.println("filename = " + cli.getFile());
      System.out.println("ref = " + cli.getRef());
   }
}
```

Parsare un URL

Eseguendo l'esempio precedente si ottiene:

```
protocol = http
authority = www.cli.di.unipi.it:80
host = www.cli.di.unipi.it
port = 80
path = /faq
query = null
filename = /faq
ref = null
```

Leggere un URL

- Una volta creato un oggetto URL si può invocare il metodo `openStream()` per ottenere uno stream da cui poter leggere il contenuto dell'URL.
- Il metodo `openStream()` ritorna un oggetto `java.io.InputStream`: leggere da un URL è analogo a leggere da uno stream di input.

```
import java.net.*;
import java.io.*;
public class URLReader
{  public static void main(String[] args) throws Exception
   {
      String url = "http://www.cli.di.unipi.it/";
      URL cli = new URL(url);
      BufferedReader in = new BufferedReader(
         new InputStreamReader(cli.openStream()));
      String inputLine;
      while((inputLine = in.readLine()) != null)
         System.out.println(inputLine);
      in.close();
   }
}
```

Leggere un URL

Nell'esempio di prima, leggendo l'URL si ottiene:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html lang="it">
<head>
  <meta http-equiv="Content-Type" content="text/html;
                                   charset=iso-8859-1">
  <link rel="stylesheet" href="/cdc.css" type="text/css">
  <link rel="alternate" type="application/rss+xml"
                 title="Ultime notizie" href="/feed.php">
  <title>Home CdC </title>
</head>
<body bgcolor="#ced8e0">
....
```

Se necessario, settare il proxy con le proprietà:

```
java -Dhttp.proxyHost=proxyhost
    [-Dhttp.proxyPort=portNumber] URLReader
```

Connettersi ad un URL

- Nell'esempio precedente, la connessione all'URL veniva effettuata solo dopo aver invocato `openStream()`.
- In alternativa, è possibile invocare il metodo `openConnection()` per ottenere un oggetto `URLConnection`.
- Utile nel caso in cui si vogliano settare alcuni parametri o proprietà della richiesta prima di connettersi.
  - es: `cliConn.setRequestProperty("User-Agent", "Mozilla/5.0");`
- Successivamente, si invoca `URLConnection.connect()`.

```
...
String url = "http://www.cli.di.unipi.it/";
URL cli = new URL(url);
URLConnection cliConn = cli.openConnection();
cliConn.connect();
BufferedReader in = new BufferedReader(
      new InputStreamReader(
      cliConn.getInputStream()));
...
```

URL e HTTPS

Si possono usare le URL anche per connessioni sicure su HTTPS:

```
import java.net.*;
import java.io.*;
public class SecureClientUrl
{
   public static void main(String[] args)
    {
      try
      {
         URL url = new URL("https://www.verisign.com");
         URLConnection conn = url.openConnection();
         BufferedReader in = new BufferedReader(
                new InputStreamReader(conn.getInputStream()));
         String inputLine;
         while((inputLine = in.readLine()) != null)
            System.out.println(inputLine);
         in.close();
      }catch(Exception e){e.printStackTrace();}
    }
}
```