CS 254 - Computer Networks

Lab 5: Network Programming in Java - URL's

Due: Thursday, February 20, 2014 (class-time)

Objective:

In this lab you will become familiar with Java's URL class and how a Java program can connect to a URL over the Internet.

Background:

(This material is taken primarily from Chapters 21 and 22 of *The Java Tutorial*.)

The `java.net` package which is included in the Java class library provides several classes that allow Java programs to communicate over a network. This particular lab examines the URL class and the URLConnection class. (We have already encountered the Socket class.)

The URL class represents an Internet Uniform Resource Locator. A URL object can be downloaded with a single call or streams can be opened to read or write the URL. The URLConnection class provides additional methods that allow one to work with URL's in more sophisticated ways.

A URL consists of several components. In particular, one must specify a protocol (usually `http` although it could be `ftp` or even `telnet`) and a host name, e.g. `www.sbu.edu`. In addition, one can optionally specify the port number and the file. For example, the URL
`http://www.cs.sbu.edu:80/contest/index.html`
specifies the `http` protocol, the host `www.cs.sbu.edu`, port 80, and the file `contest/index.html`

Instructions:

1. Obtain a copy of the Java program `URLParser.java`. Examine this program. It creates a URL object and then displays the object.

   First modify the program to display the protocol, host, filename and port number, with labels, using the methods `getProtocol()`, `getHost()`, `getFile()`, and `getPort()`. Also modify the program to obtain the URL string from the user instead of having it hardwired in the code.

Print a copy of the execution of the program with the URL string:
`http://www.cs.sbu.edu:80/contest/index.html`
and hand it in with your lab write-up.

Execute the program again with the URL string: `http://www.cs.sbu.edu/` Print the results you obtain.

Document the program with the modifications you made and hand in a hard copy of the modified program with your lab.

2. In this exercise you will use the URL member function `openStream()` to open a URL as an input stream.

Obtain a copy of the Java program `FileDisplay.java`. Create a text file with several lines of text and store it in your Eclipse project. Compile and execute the program providing the name of the text file you just created when asked for a file name. The program displays a text file named by the user. Note that the program uses the file name to create a File object, uses this File object to create a FileReader object, and then converts the FileReader object into a BufferedReader so that the file can be read one line at a time using the BufferedReader method `readLine()`. (This could be done with the Scanner class but here we use the BufferedReader.)

You need to modify this program so that instead of displaying a text file it displays an `html` file (which is a text file) obtained from a web server using Java's URL class. The modifications should proceed as follows:

- Import the `java.net` package.
- Obtain the URL string from the user.
- Open the URL as in step 1.
- Remove the line that creates the File object *f*.
- Replace the FileReader object *fr* with an InputStreamReader object, passing the constructor `aURL.openStream()`.
- Pass the InputStreamReader object to the BufferedReader constructor.
- Handle the two exceptions: `MalformedURLException` and `IOException`.
- Rename the class with a more descriptive name.

Compile and execute the program on the URL strings:

- `http://www.cs.sbu.edu/contest`
- `http://www.google.com/`

Describe the results you obtain in each case.

Document the program with the modifications you made and hand in a hard copy of the modified program with your lab.

3. Further modify the program from Step 2 to *open a network connection* to the URL object. The URL member function `openConnection()` returns a URLConnection. Once a URLConnection is opened a program can read or write across the connection.

   The steps are:
   1) Create a URL.
   2) Open a connection to the URL.

   To open the connection do the following:

   ```
   URLConnection connection = aURL.openConnection();
   ```

   3) Get an InputStream from the connection by calling the method `getInputStream()`. Create an InputStreamReader from the InputStream and then create a BufferedReader as before.
   4) Close the input stream after the reading has completed.

   Modify your program from Step 2 as described above by opening a URLConnection and then displaying the contents of the `html` file.

   Compile and execute your program to make sure it works. Describe the results you get. Document the program with the modifications you made and hand in a hard copy of the modified program with your lab.

4. In the next step we will use a URLConnection to write to a web page ("post" data). Before we do that, open a browser and browse the URL
   http://aspnet2/networks/piggy.html

   Enter a phrase and look at the results. (Pretty awsome, huh?)

5. Now you will write a Java program to do the same thing, that is post a phrase and then capture the resulting web page.

   First copy the program `FormPoster.java` to your Eclipse project. This is the shell of the program that will do the job for you. You will need to complete this program.

   Add code to accomplish the following in the main method:
   1) Create a String variable that stores the phrase you want translated.
   2) Encode this String into a form recognized by http software by calling the method encode() with parameters "phrase" and the phrase you want translated. Store the result of this call in a variable named *formData*.
   3) Add a try-catch and catch the exceptions MalformedURLException and IOException.
   4) Within the try block create a URL object from the string `"http://aspnet2/networks/pig.aspx"`.
   5) Open a connection (URLConnection class) to the URL as you did in Step 3 of the lab.
   6) Set the connection for output by calling the (URLConnection) method `setDoOutput(true)`.
   7) Create an OutputStreamWriter passing the constructor two parameters: `connection.getOutputStream()` and `"ASCII"`.
   8) Create a PrintWriter object passing the constructor the OutputStreamWriter as a parameter and passing *true* as a second parameter.

9) Post the data by calling the println method of the PrintWriter passing *formData* as the parameter.
11) Close the PrintWriter.
12) Now you need to get a reader to read the results of the posting. Create an InputStreamReader passing the constructor the parameter connection.getInputStream().
13) Create a BufferedReader from the InputStreamReader.
14) Add the while loop to read the lines from the BufferedReader and display them on the console.
15) Finally, close the BufferedReader.

Note that steps 12-15 in the procedure above are the same as what you used in Step 3 of the lab to display the source of a web page.

Once you have written this code, run it. Once it runs correctly print the results of an execution and turn it in with your lab.

Document the program indicating the modifications you made and including your name(s) and the date. Print the program and turn it in with your lab.

Extra Credit:

Modify the code from Step 5 above so that when the resulting web page is read from the resulting site you can extract ("scrape") the string with the translated phrase. So the program should only display the translated phrase at the console not the entire web page.

Again document your program appropriately and turn in the source code and a sample execution.

Extra Credit:

Explain why the translated phrase in the resulting HTML is not on one line.

Help Policy:

Help Policy in Effect for This Assignment: Group Project with Limited Collaboration

In particular, you may discuss the assignment and concepts related to the assignment with the following persons, in addition to an instructor in this course: any member of your group; any St. Bonaventure Computer Science instructor; and any student enrolled in CS 254.

You may use the following materials produced by other students: materials produced by members of your group.