Java URLs

Online References

Working with URLs:

- http://docs.oracle.com/javase/tutorial/networking/urls/index.html

Overview

- What is a URL
- Creating a URL
- Parsing a URL
- Reading Directly from a URL
- Connecting to a URL
- Reading from and Writing to a URL

What is a URL?

URL stands for *uniform resource locator*. URLs are references to resources on the internet.

What is a URL?

URLs have two components:

1. The *protocol identifier*: (http, sftp, imap, mailto and so on)
2. The *resource name*: everything after the ://

So if the URL is http://www.google.com, the protocol identifier is HTTP and the resource name is www.google.com

Resource Names

Resource names optionally commonly consist of:

1. host name (www.google.com)
2. filename (…/something.php)
3. port number(www.google.com:**4040/**)
4. reference (…/something.php**#reference)**

Creating URLs

Creating URLs

Using the java.net.URL class is the easiest way to create a URL:

URL myURL = new URL("http://google.com");

Creating URLs

URL myURL = new URL("http://google.com");

This is an *absolute URL*, because it contains all the information required to reach the desired resource. It is also possible to work with *relative URLs*, which do not.

Relative URLs

Some URLs are relative, for example in a webpage you might have:

```
<a href="PicturesOfMe.html">Pictures of Me</a>
<a href="MyKids.html">Pictures of My Kids</a>
```

where those addresses are *relative* to the page they're found on. You can use this relative information to make other URLs:

```
URL myURL = new URL("http://example.com/pages/");
URL page1URL = new URL(myURL, "page1.html");
URL page2URL = new URL(myURL, "page2.html");
```

Relative URLs

```
URL myURL = new URL("http://example.com/pages/");
URL page1URL = new URL(myURL, "page1.html");
URL page2URL = new URL(myURL, “page2.html");
```

This same concept also works for references:

```
URL page1BottomURL = new URL(page1URL,"#BOTTOM");
```

More URL Constructors

The following constructor:

```
new URL("http", "example.com", “/pages/page1.html");
```

Is the same as:

```
new URL("http://example.com/pages/page1.html");
```

It is also possible to construct with the port separately:

```
URL gamelan = new URL("http", "example.com", 80, “pages/page1.html”);
```

Is the same as:

```
URL gamelan2 = new URL(“http://example.com:80/pages/page1.html”);
```

Special Characters

Some URLs contain special characters (like spaces) that need to be converted. If you wanted to make a URL for the following address:

```
http://example.com/hello world/
```

It actually needs to be

```
URL url = new URL("http://example.com/hello%20world/");
```

java.net.URI will do this automatically for you:

```
URI uri = new URI("http", "example.com", "/hello world/", "");
URL url = uri.toURL();
```

MalformedURLException

Java will also not let you create bad URLs. In the case where the information passed to the URL object's constructor isn't correct it will throw a MalformedURLException:

```
try {
    URL myURL = new URL(...);
} catch (MalformedURLException e) {
    // exception handler code here
    // ...
}
```

Write-Once

URLs (like Strings), are *immutable*, or write-once. After you create a URL you cannot change any of it's values.

However you can get those values and use them to create other URLs.

Parsing URLs

Parsing URLs

Java URLs provide a set of accessor methods to get at the different parts of the URL:

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

public class ParseURL {
    public static void main(String[] args) throws Exception {
        URL aURL = new URL("http://example.com:80/docs/books/tutorial"
                           + "/index.html?name=networking#DOWNLOADING");

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

Parsing URLs

Which would print out:

```
protocol = http
authority = example.com:80
host = example.com
port = 80
path = /docs/books/tutorial/index.html
query = name=networking
filename = /docs/books/tutorial/index.html?name=networking
ref = DOWNLOADING
```

Reading Directly From a URL

Reading from a URL

It's very simple to read a webpage (or from some other resource) if you know the URL:

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

public class URLReader {
    public static void main(String[] args) throws Exception {

        URL oracle = new URL("http://www.oracle.com/");
        BufferedReader in = new BufferedReader(
                new InputStreamReader(oracle.openStream()));

        String inputLine;
        while ((inputLine = in.readLine()) != null) {
            System.out.println(inputLine);
        }
        in.close();
    }
}
```

Connecting to a URL

Connecting to a URL

You can get a URLConnection object (more on that in a bit) which will act as a connection to a URL (which you can then use to read and write through).

```
try {
    URL myURL = new URL("http://example.com/");
    URLConnection myURLConnection = myURL.openConnection();
    myURLConnection.connect();
}
catch (MalformedURLException e) {
    // new URL() failed
    // ...
}
catch (IOException e) {
    // openConnection() failed
    // ...
}
```

Connecting to a URL

Calling the openConnection() method **will open a new connection**.

This means if you've called it once on a URL, it will not return that connection but rather get a new one.

Also, the connection is only formed when the connect method is called. Note that other operations like getInputStream and getOutputStream will do this automatically, if necessary.

Reading from and Writing to a URLConnection

I/O with URLConnection

You can accomplish the same thing in the previous example (reading from a URL) with a URLConnection:

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

public class URLConnectionReader {
    public static void main(String[] args) throws Exception {
        URL oracle = new URL("http://www.oracle.com/");
        URLConnection yc = oracle.openConnection();
        BufferedReader in = new BufferedReader(new InputStreamReader(
                                yc.getInputStream()));
        String inputLine;
        while ((inputLine = in.readLine()) != null)
            System.out.println(inputLine);
        in.close();
    }
}
```

The difference is that URLConnections also allow you to write to the URL at the same time (as well as some other tasks).

I/O with URLConnection

Many webpages have *forms* which require user input via text fields, buttons or a GUI that let you enter data to send to the server.

After you enter this information, your browser will write this data to the URL over the network. The server will receive the data, process it and respond with something (usually another webpage).

I/O with URLConnection

This is commonly done with the HTTP POST method, so writing to a URL is sometimes called *posting* to a URL. Web servers recognize post commands and forwards the data from the client to the appropriate place (a script or something of that nature).

I/O with URLConnection

Java programs can do this automatically by:

1. Creating a URL object.
2. Getting a URLConnection object from the URL.
3. Setting output capability on the URLConnection.
4. Opening a connection to the resource.
5. Getting an output stream from the connection.
6. Writing to the output stream.
7. Closing the output stream.

Reverse Example

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

public class Reverse {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage:  java Reverse "
                + "http://<location of your servlet/script>"
                + " string_to_reverse");
            System.exit(1);
        }

        String stringToReverse = URLEncoder.encode(args[1], "UTF-8");

        URL url = new URL(args[0]);
        URLConnection connection = url.openConnection();
        connection.setDoOutput(true);

        OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
        out.write("string=" + stringToReverse);
        out.close();

        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String decodedString;
        while ((decodedString = in.readLine()) != null) {
            System.out.println(decodedString);
        }
        in.close();
    }
}
```

The reverse example assumes that there is a reverse servlet up and running on a remote web server.

http://docs.oracle.com/javase/tutorial/networking/urls/examples/ReverseServlet.java

Reverse Example

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

public class Reverse {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage:  java Reverse "
                + "http://<location of your servlet/script>"
                + " string_to_reverse");
            System.exit(1);
        }

        String stringToReverse = URLEncoder.encode(args[1], "UTF-8");

        URL url = new URL(args[0]);
        URLConnection connection = url.openConnection();
        connection.setDoOutput(true);

        OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
        out.write("string=" + stringToReverse);
        out.close();

        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String decodedString;
        while ((decodedString = in.readLine()) != null) {
            System.out.println(decodedString);
        }
        in.close();
    }
}
```

Write an error message if we don't have the right arguments.

Reverse Example

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

public class Reverse {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage:  java Reverse "
                + "http://<location of your servlet/script>"
                + " string_to_reverse");
            System.exit(1);
        }

        String stringToReverse = URLEncoder.encode(args[1], "UTF-8");

        URL url = new URL(args[0]);
        URLConnection connection = url.openConnection();
        connection.setDoOutput(true);

        OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
        out.write("string=" + stringToReverse);
        out.close();

        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String decodedString;
        while ((decodedString = in.readLine()) != null) {
            System.out.println(decodedString);
        }
        in.close();
    }
}
```

We need to encode the string we're sending to the reverse server (eg., this will convert ' ' to '%20' and so on).

Reverse Example

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

public class Reverse {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage:  java Reverse "
                + "http://<location of your servlet/script>"
                + " string_to_reverse");
            System.exit(1);
        }

        String stringToReverse = URLEncoder.encode(args[1], "UTF-8");

        URL url = new URL(args[0]);
        URLConnection connection = url.openConnection();
        connection.setDoOutput(true);

        OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
        out.write("string=" + stringToReverse);
        out.close();

        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String decodedString;
        while ((decodedString = in.readLine()) != null) {
            System.out.println(decodedString);
        }
        in.close();
    }
}
```

Create a URL targeting our reverse server. The open a connection and set the output flag to true.

Reverse Example

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

public class Reverse {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage:  java Reverse "
                + "http://<location of your servlet/script>"
                + " string_to_reverse");
            System.exit(1);
        }

        String stringToReverse = URLEncoder.encode(args[1], "UTF-8");

        URL url = new URL(args[0]);
        URLConnection connection = url.openConnection();
        connection.setDoOutput(true);

        OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
        out.write("string=" + stringToReverse);
        out.close();

        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String decodedString;
        while ((decodedString = in.readLine()) != null) {
            System.out.println(decodedString);
        }
        in.close();
    }
}
```

Open a stream that we can write to and the send the string over to the URL.

Reverse Example

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

public class Reverse {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage:  java Reverse "
                + "http://<location of your servlet/script>"
                + " string_to_reverse");
            System.exit(1);
        }

        String stringToReverse = URLEncoder.encode(args[1], "UTF-8");

        URL url = new URL(args[0]);
        URLConnection connection = url.openConnection();
        connection.setDoOutput(true);

        OutputStreamWriter out = new OutputStreamWriter(connection.getOutputStream());
        out.write("string=" + stringToReverse);
        out.close();

        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String decodedString;
        while ((decodedString = in.readLine()) != null) {
            System.out.println(decodedString);
        }
        in.close();
    }
}
```

Read the response and print it to the screen.