Introduction

- How to write a program that fetches URL content

- How to write a program that submits GET or POST requests to a web server or a servlet
Start simply

- No programming involved!
  - Connect to server (telnet)
  - Send request (type in an http request)
  - Get input

- Using TCP Sockets
  - Connect to port 80
  - Send HTTP request (GET, POST, HEAD)
  - Read HTTP response

URL Programming In Java

- A URL is a name for a web resource (page, applet)
- Java URL programming allows Java programs to access web resources by sending protocol requests and receiving protocol responses.
- Related Classes:
  - URL class
  - URLConnection class
  - HttpURLConnection class
  - URLEncoder class
URL Class

- Represents a Uniform Resource Locator
  - scheme (protocol)
  - hostname
  - port
  - path
  - query string

Parsing a URL

- You can use a URL object as a parser:

  ```java
  URL u = new URL("http://www.cs.wmich.edu/index.html");
  System.out.println("Proto:" + u.getProtocol());
  System.out.println("File:" + u.getFile());
  ```
Retrieving URL contents

- There are a number of ways to do this:
  
  `Object getContent();`

  `InputStream openStream();`

  `URLConnection openConnection();`

Getting Header Information

- There are methods that return information extracted from response headers:

  `String getContentType();`

  `String getContentLength();`

  `long getLastModified();`
URLConnection Class

- Represents the connection (not the URL itself).
- More control than URL
  - can write to the connection (send POST data).
  - can set request headers.
- HttpURLConnection is a subclass of URLConnection that is specific to HTTP

Example: Reading from a URL

```java
// -- open connection
URL url = new URL("http://www.google.com/index.html");

HttpURLConnection conn = (HttpURLConnection)url.openConnection();
conn.connect();

// -- read in html ---------------
BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
String line;
String page = new String("");
while (null != ((line = in.readLine()))) {
    page += line;
}

in.close();
System.out.println(page);
```
Points to note

- An HttpURLConnection represents a connection to a particular page on a server, not just to the server.

- An HttpURLConnection object is NOT made by the usual use of new(). Instead it is created by the use of the Url’s openConnection() method.

More points to note

- The connect method (of HttpURLConnection) not only opens the connection but also makes the GET request!

- The code for reading in the HTML from the connection is identical to that for reading in a file.
Faking GET Form Submission

- Not all webpages are produced in response to a simple GET request
- Some are the output of a program which may require parameters
- passed using the GET method (via the URL), e.g.
  - http://www.cs.wmich.edu/servlet/convert?id=57&amount=10&units=pint

Faking POST Form Submission

- The POST request sends parameters (data) in the HTTP request body
- Any data to be sent must be encoded as a string of key-value pairs:
  id=57&amount=10&units=pint
**URLEncoder Class**

- Used to encode GET and POST parameters.

- Example:

  ```java
  String content = "action=" + URLEncoder.encode("100");
  content += "&zipcode=" + URLEncoder.encode("49008");
  content += "&name=" + URLEncoder.encode("Bill Somebody");
  content += "&passwd=" + URLEncoder.encode("cisco");
  ```

**Making GET Form Submissions**

- Step 1: Encode GET parameters

  ```java
  String content = "action=" + URLEncoder.encode("100");
  content += "&zipcode=" + URLEncoder.encode("49008");
  content += "&name=" + URLEncoder.encode("Bill Somebody");
  content += "&passwd=" + URLEncoder.encode("cisco");
  ```

- Step 2: Make an HttpURLConnection object in the usual way:

  ```java
  URL url = new URL("http://www.cs.wmich.edu/servlet/converter" + "?" + contents);
  ```

- Step 3:

  ```java
  HttpURLConnection conn = (HttpURLConnection)url.openConnection();
  ```

- Step 4:

  ```java
  conn.connect();
  ```

- Step 5: You can read the reply as explained before.
Making POST Form Submissions

- **Step 1:** Encode POST parameters
  String content = "action=" + URLEncoder.encode("100");
  content += "&zipcode=" + URLEncoder.encode("49008");
  content += "&name=" + URLEncoder.encode("Bill Somebody");
  content += "&passwd=" + URLEncoder.encode("cisco");

- **Step 2:** Make an HttpURLConnection object in the usual way:
  URL url = new URL("http://www.cs.wmich.edu/servlet/converter");

- **Step 3:**
  HttpURLConnection conn = (HttpURLConnection)url.openConnection();

Making POST Form Submissions (Contd.)

- **Step 4:** Tell it that you want to use the POST method
  conn.setRequestMethod("POST");
  conn.setDoOutput(true);

- **Step 5:** Build a Printer writer to send things out via the connection and send the data.
  PrintWriter out = new PrintWriter(conn.getOutputStream());
  out.println(content);
  out.close();

- **Step 6:** You can read the reply as explained before.