core WEB programming

Network Programming: Clients

Agenda

- Creating sockets
- Implementing a generic network client
- Parsing data using StringTokenizer
- Retrieving files from an HTTP server
- Retrieving Web documents by using the URL class

Client vs. Server

- **Traditional definition**
  - Client: User of network services
  - Server: Supplier of network services
- **Problem with traditional definition**
  - If there are 2 programs exchanging data, it seems unclear
  - Some situations (e.g., X Windows) seem reversed
- **Easier way to remember distinction**
  - Server starts first. Server doesn't specify host (just port).
  - Client starts second. Client specifies host (and port).
- **Analogy: Company phone line**
  - Installing phone is like starting server
  - Extension is like port
  - Person who calls is the client: he specifies both host (general company number) and port (extension)

Client vs. Server (Continued)

- **If server has to start first, why are we covering clients before we cover servers?**
  - Clients are slightly easier.
  - We can test clients by connecting to *existing* servers that are already on the internet.
- **Point: clients created in Java need not communicate with servers written in Java.**
  - They can communicate with any server that accepts socket connections (as long as they know the proper communication protocol).
  - Exception: ObjectInputStream and ObjectOutputStream allow Java programs to send complicated data structures back and forth. Only works in Java, though.

Steps for Implementing a Client

1. **Create a Socket object**

```
Socket client = new Socket("hostname", portNumber);
```

2. **Create an output stream that can be used to send info to the Socket**

```
// Last arg of true means autoflush -- flush stream
// when println is called
PrintWriter out =
  new PrintWriter(client.getOutputStream(), true);
```

3. **Create an input stream to read the response from the server**

```
BufferedReader in =
  new BufferedReader
    (new InputStreamReader(client.getInputStream()));
```

Steps for Implementing a Client (Continued)

4. **Do I/O with the input and output Streams**
   - For the output stream, PrintWriter, use print and println, similar to System.out.println
     - The main difference is that you can create PrintWriters for different Unicode characters sets, and you can't with PrintStream (the class of System.out).
   - For the input stream, BufferedReader, you can call read to get a single character or an array of characters, or call readLine to get a whole line
     - Note that readLine returns null if the connection was terminated (i.e. on EOF), but waits otherwise
5. **Close the socket when done**

```
client.close();
```

- Also closes the associated input and

A Generic Network Client

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

/** A starting point for network clients. */

public class NetworkClient {
  protected String host;
  protected int port;

  public NetworkClient(String host, int port) {
    this.host = host;
    this.port = port;
  }

  public String getHost() {
    return(host);
  }

  public int getPort() {
    return(port);
  }
  ...
```

A Generic Network Client (Continued)

```
...

/** Establishes the connection, then passes the socket
 *  to handleConnection. */

public void connect() {
  try {
    Socket client = new Socket(host, port);
    handleConnection(client);
  } catch(UnknownHostException uhe) {
    System.out.println("Unknown host: " + host);
    uhe.printStackTrace();
  } catch(IOException ioe) {
    System.out.println("IOException: " + ioe);
    ioe.printStackTrace();
  }
}
...
```

A Generic Network Client (Continued)

```
/** This is the method you will override when
 *  making a network client for your task.
 *  This default version sends a single line
 *  ("Generic Network Client") to the server,
 *  reads one line of response, prints it, then exits.
 */

protected void handleConnection(Socket client)
    throws IOException {
  PrintWriter out =
    SocketUtil.getPrintWriter(client);
  BufferedReader in =
    SocketUtil.getBufferedReader(client);
  out.println("Generic Network Client");
  System.out.println
    ("Generic Network Client:\n" +
     "Made connection to " + host +
     " and got '" + in.readLine() + "' in response");
  client.close();
 }
}
```

SocketUtil – Simplifying Creation of Reader and Writer

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

public class SocketUtil {
  /** Make a BufferedReader to get incoming data.  */
  public static BufferedReader getBufferedReader
                      (Socket s) throws IOException {
    return(new BufferedReader(
      new InputStreamReader(s.getInputStream())));
  }

  /** Make a PrintWriter to send outgoing data.
   *  This PrintWriter will automatically flush stream
   *  when println is called.
   */
  public static PrintWriter getPrintWriter(Socket s)
      throws IOException {
    // 2nd argument of true means autoflush
    return(new PrintWriter(s.getOutputStream(), true));
  }
}
```

Example Client

```
public class NetworkClientTest {
  public static void main(String[] args) {
    String host = "localhost";
    if (args.length > 0)
      host = args[0];
    int port = 8088;
    if (args.length > 1)
      port = Integer.parseInt(args[1]);
    NetworkClient nwClient
      = new NetworkClient(host, port);
    nwClient.connect();
  }
}
```

Example Client, Result

```
> java NetworkClientTest ftp.netscape.com 21
Generic Network Client:
Made connection to ftp.netscape.com and got
‘220 ftp26 FTP server (UNIX(r) System V Release 4.0)
ready.’ in response
>
```

Aside: Parsing Strings Using StringTokenizer

- **Idea**
  - Build a tokenizer from an initial string
  - Retrieve tokens one at a time with `nextToken`
  - You can also see how many tokens are remaining (`countTokens`) or simply test if the number of tokens remaining is nonzero (`hasMoreTokens`)

```
StringTokenizer tok
  = new StringTokenizer(input, delimiters);
while (tok.hasMoreTokens()) {
  doSomethingWith(tok.nextToken());
}
```

StringTokenizer

- **Constructors**
  - StringTokenizer(String input, String delimiters)
  - StringTokenizer(String input, String delimiters, boolean includeDelimiters)
  - StringTokenizer(String input)
    - Default delimiter set is "`\t\n\r\f`" (whitespace)
- **Methods**
  - nextToken(), nextToken(String delimiters)
  - countTokens()
  - hasMoreTokens()
- **Also see methods in String class**
  - substring, indexOf, startsWith, endsWith, compareTo, …
  - JDK 1.4 has regular expressions in java.util.regex!

Interactive Tokenizer: Example

```
import java.util.StringTokenizer;

public class TokTest {
  public static void main(String[] args) {
    if (args.length == 2) {
      String input = args[0], delimiters = args[1];
      StringTokenizer tok
        = new StringTokenizer(input, delimiters);
      while (tok.hasMoreTokens()) {
        System.out.println(tok.nextToken());
      }
    } else {
      System.out.println
        ("Usage: java TokTest string delimiters");
    }
  }
}
```

Interactive Tokenizer: Result

```
> java TokTest http://www.microsoft.com/~gates/ :/.
http
www
microsoft
com
~gates


> java TokTest "if (tok.hasMoreTokens()) {" "(){. "
if
tok
hasMoreTokens
```

A Client to Verify Email Addresses

- **Talking to a mail server**
  - One of the best ways to get comfortable with a network protocol is to telnet to the port a server is on and try out commands interactively
- **Example talking to apl.jhu.edu's server**

```
> telnet apl.jhu.edu 25
Trying 128.220.101.100 ...Connected … Escape character …
220 aplcenmp.apl.jhu.edu Sendmail SMI-8.6/SMI-SVR4 ready …
expn hall
250 Marty Hall <hall@aplcenmp.apl.jhu.edu>
expn root
250 Gary Gafke <…>
250 Tom Vellani <…>
quit
221 aplcenmp.apl.jhu.edu closing connection
Connection closed by foreign host.
```

Address Verifier

```
/** Given an email address of the form user@host,
 *  connect to port 25 of the host and issue an
 *  'expn' request for the user. Print the results.
 */

public class AddressVerifier extends NetworkClient {
  private String username;

  public static void main(String[] args) {
    MailAddress address = new MailAddress(args[0]);
    AddressVerifier verifier
      = new AddressVerifier(address.getUsername(),
                            address.getHostname(),
                            25);
    verifier.connect();
  }
  ...
```

Address Verifier (Continued)

```
protected void handleConnection(Socket client) {
  try {
    PrintWriter out =
      SocketUtil.getPrintWriter(client);
    InputStream in = client.getInputStream();
    byte[] response = new byte[1000];
    // Clear out mail server's welcome message.
    in.read(response);
    out.println("EXPN " + username);
    // Read the response to the EXPN command.
    // May be multiple lines!
    int numBytes = in.read(response); // Can't use readLine!
    // The 0 means to use normal ASCII encoding.
    System.out.write(response, 0, numBytes);
    out.println("QUIT");
    client.close();
  } catch(IOException ioe) {
    System.out.println("Couldn't make connection: "
                       + ioe);
  }
}
...}
```

MailAddress

```
// Takes a string of the form "user@host" and
// separates it into the "user" and "host" parts.

public class MailAddress {
  private String username, hostname;

  public MailAddress(String emailAddress) {
    StringTokenizer tokenizer
      = new StringTokenizer(emailAddress, "@");
    this.username = getArg(tokenizer);
    this.hostname = getArg(tokenizer);
  }

  private static String getArg(StringTokenizer tok) {
    try { return(tok.nextToken()); }
    catch (NoSuchElementException nsee) {
      System.out.println("Illegal email address");
      return(null);
    }
  }...
}
```

Address Verifier: Result

```
> java AddressVerifier tbl@w3.org
250 <timbl@hq.lcs.mit.edu>

> java AddressVerifier timbl@hq.lcs.mit.edu
250 Tim Berners-Lee <timbl>

> java AddressVerifier gosling@mail.javasoft.com
550 gosling... User unknown
```

Brief Aside: Using the HTTP GET Command

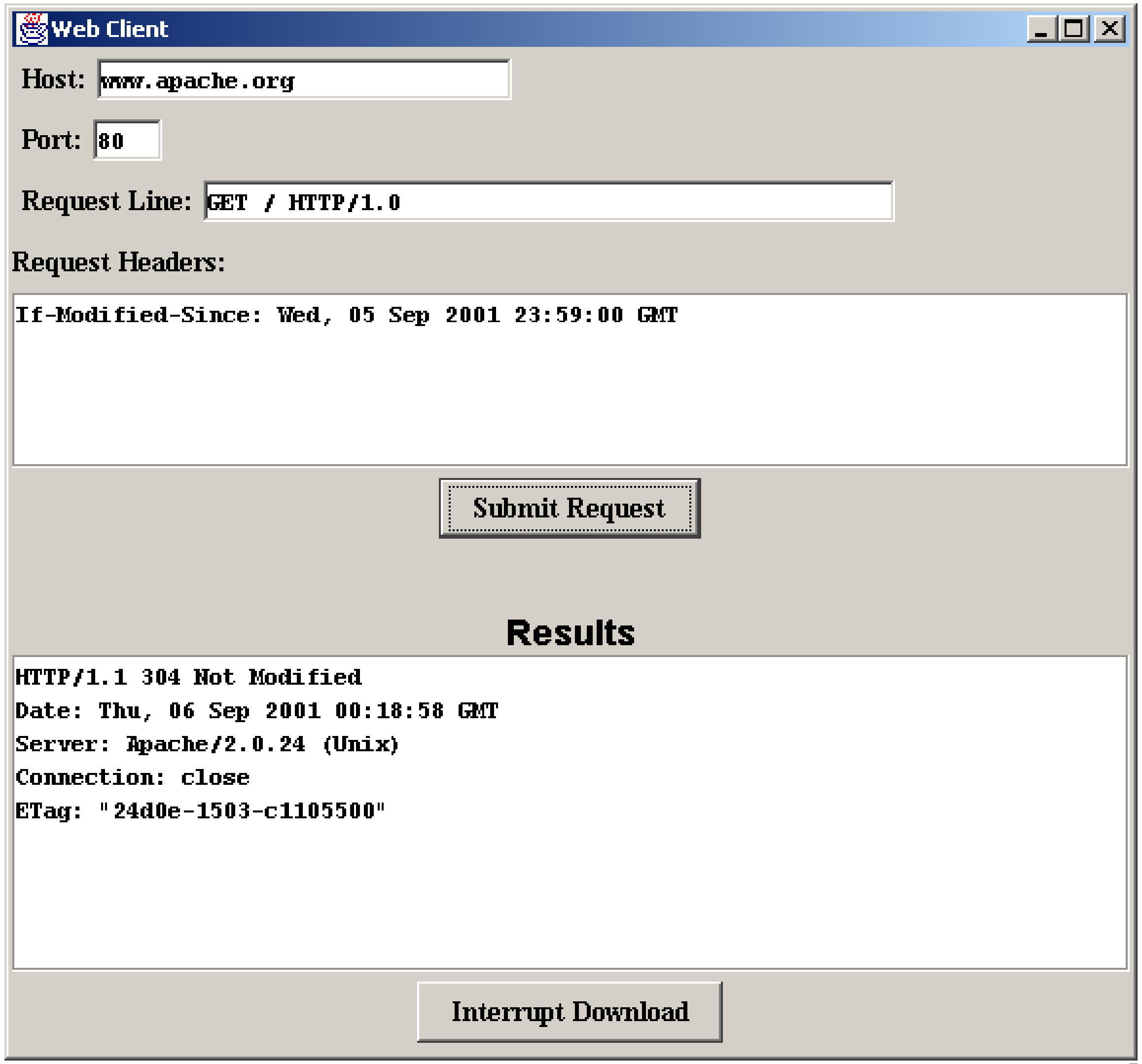
- For the URL http://www.apl.jhu.edu/~lmb/

```
Unix> telnet www.apl.jhu.edu 80
Trying 128.220.101.100 ...
Connected to aplcenmp.apl.jhu.edu.
Escape character is '^]'.
GET /~lmb/ HTTP/1.0

HTTP/1.0 200 Document follows
Date: Sat, 30 Jun 2001 14:34:58 GMT
Server: NCSA/1.5.2
Last-modified: Tue, 11 Jul 2001 15:13:56 GMT
Content-type: text/html
Content-length: 50479

<!DOCTYPE HTML PUBLIC
          "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
...
</HTML>Connection closed by foreign host.
Unix>
```

Talking to Web Servers Interactively

- **WebClient**
  - Simple graphical user interface to communicate with HTTP servers
  - User can interactively specify:
    - Host
    - Port
    - HTTP request line
    - HTTP request headers
  - HTTP request is performed in a separate thread
  - Response document is placed in a scrollable text area
  - Download all source files for WebClient from http://archive.corewebprogramming.com/Chapter17.html

WebClient: Example



Host: www.apache.org

Port: 80

Request Line: GET / HTTP/1.0

Request Headers:

```
If-Modified-Since: Wed, 05 Sep 2001 23:59:00 GMT
```

Results

```
HTTP/1.1 304 Not Modified
Date: Thu, 06 Sep 2001 00:18:58 GMT
Server: Apache/2.0.24 (Unix)
Connection: close
ETag: "24d0e-1503-c1105500"
```

A Class to Retrieve a Given URI from a Given Host

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

public class UriRetriever extends NetworkClient {
  private String uri;

  public static void main(String[] args) {
    UriRetriever uriClient
      = new UriRetriever(args[0],
                         Integer.parseInt(args[1]),
                         args[2]);
    uriClient.connect();
  }

  public UriRetriever(String host, int port,
                      String uri) {
    super(host, port);
    this.uri = uri;
  }
  ...
```

A Class to Retrieve a Given URI from a Given Host (Continued)

```
  // It is safe to use blocking IO (readLine) since
  // HTTP servers close connection when done,
  // resulting in a null value for readLine.

  protected void handleConnection(Socket uriSocket)
      throws IOException {
    PrintWriter out =
      SocketUtil.getPrintWriter(uriSocket);
    BufferedReader in =
      SocketUtil.getBufferedReader(uriSocket);
    out.println("GET " + uri + " HTTP/1.0\n");
    String line;
    while ((line = in.readLine()) != null) {
      System.out.println("> " + line);
    }
  }
}
```

A Class to Retrieve a Given URL

```
public class UrlRetriever {
  public static void main(String[] args) {
    checkUsage(args);
    StringTokenizer tok = new StringTokenizer(args[0]);
    String protocol = tok.nextToken(":");
    checkProtocol(protocol);
    String host = tok.nextToken(":/");
    String uri;
    int port = 80;
    try {
      uri = tok.nextToken("");
      if (uri.charAt(0) == ':') {
        tok = new StringTokenizer(uri);
        port = Integer.parseInt(tok.nextToken(":/"));
        uri = tok.nextToken("");
      }
    } catch(NoSuchElementException nsee) {
      uri = "/";
    }
```

A Class to Retrieve a Given URL (Continued)

```
    UriRetriever uriClient =
      new UriRetriever(host, port, uri);
    uriClient.connect();
  }

  /** Warn user if they forgot the URL. */
  private static void checkUsage(String[] args) {
    if (args.length != 1) {
      System.out.println("Usage: UrlRetriever <URL>");
      System.exit(-1);
    }
  }

  /** Tell user that this can only handle HTTP. */
  private static void checkProtocol(String protocol) {
    if (!protocol.equals("http")) {
      System.out.println("Don't understand protocol "
                         + protocol);
      System.exit(-1);
    }
  }}
```

UrlRetriever in Action

- No explicit port number

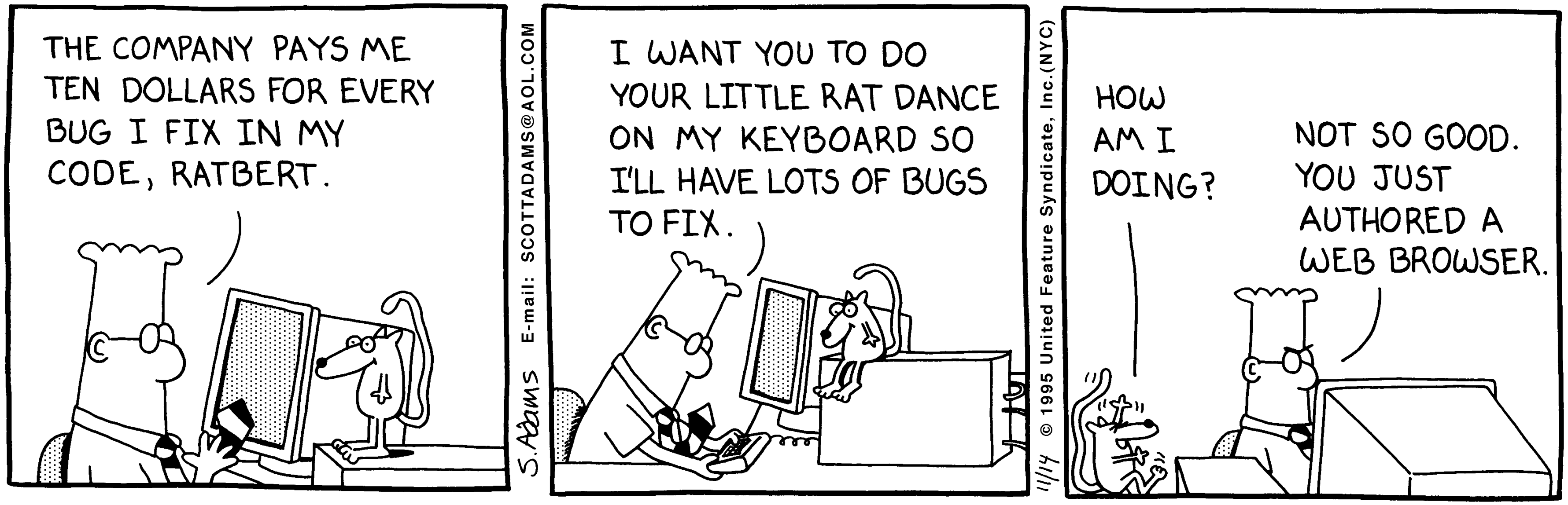
```
Prompt> java UrlRetriever
         http://www.microsoft.com/netscape-beats-ie.html
> HTTP/1.0 404 Object Not Found
> Content-Type: text/html
>
> <body><h1>HTTP/1.0 404 Object Not Found
> </h1></body>
```

UrlRetriever in Action (Continued)

- Explicit port number

```
Prompt> java UrlRetriever
      http://home.netscape.com:80/ie-beats-netscape.html
> HTTP/1.0 404 Not found
> Server: Netscape-Enterprise/2.01
> Date: Wed, 11 Jul 2001 21:17:50 GMT
> Content-length: 207
> Content-type: text/html
>
> <TITLE>Not Found</TITLE><H1>Not Found</H1> The requested
  object does not exist on this server. The link you
  followed is either outdated, inaccurate, or the server
  has been instructed not to let you have it.
```

Writing a Web Browser


THE COMPANY PAYS ME TEN DOLLARS FOR EVERY BUG I FIX IN MY CODE, RATBERT.
I WANT YOU TO DO YOUR LITTLE RAT DANCE ON MY KEYBOARD SO I'LL HAVE LOTS OF BUGS TO FIX.
HOW AM I DOING?
NOT SO GOOD. YOU JUST AUTHORED A WEB BROWSER.
S. Adams
E-mail: SCOTTADAMS@AOL.COM
11/14
© 1995 United Feature Syndicate, Inc.(NYC)

- **Wow! We just wrote a Web browser in 3 pages of code.**
  – Didn't format the HTML, but still not bad for 3 pages
  – But we can do even better…

Browser in 1 Page: Using URL

```java
public class UrlRetriever2 {
  public static void main(String[] args) {
    try {
      URL url = new URL(args[0]);
      BufferedReader in = new BufferedReader(
                                new InputStreamReader(
                                  url.openStream()));
      String line;
      while ((line = in.readLine()) != null) {
        System.out.println("> " + line);
      }
      in.close();
    } catch(MalformedURLException mue) { // URL c'tor
      System.out.println(args[0] + "is an invalid URL: "
                             + mue);
    } catch(IOException ioe) { // Stream constructors
      System.out.println("IOException: " + ioe);
    }
  }
}
```

UrlRetriever2 in Action

```
Prompt> java UrlRetriever2 http://www.whitehouse.gov/
> <HTML>
> <HEAD>
> <TITLE>Welcome To The White House</TITLE>
> </HEAD>
> ... Remainder of HTML document omitted ...
> </HTML>
```

Useful URL Methods

- **openConnection**
  - Yields a `URLConnection` which establishes a connection to host specified by the URL
  - Used to retrieve header lines and to supply data to the HTTP server
- **openInputStream**
  - Returns the connection’s input stream for reading
- **toExernalForm**
  - Gives the string representation of the URL
- **getRef, getFile, getHost, getProtocol, getPort**
  - Returns the different components of the URL

Using the URL Methods: Example

```
import java.net.*;

public class UrlTest {
  public static void main(String[] args) {
    if (args.length == 1) {
      try {
        URL url = new URL(args[0]);
        System.out.println
          ("URL: " + url.toExternalForm() + "\n" +
           "   File:       " + url.getFile() + "\n" +
           "   Host:       " + url.getHost() + "\n" +
           "   Port:       " + url.getPort() + "\n" +
           "   Protocol:   " + url.getProtocol() + "\n" +
           "   Reference: " + url.getRef());
      } catch(MalformedURLException mue) {
        System.out.println("Bad URL.");
      }
    } else
      System.out.println("Usage: UrlTest <URL>");
  }
}
```

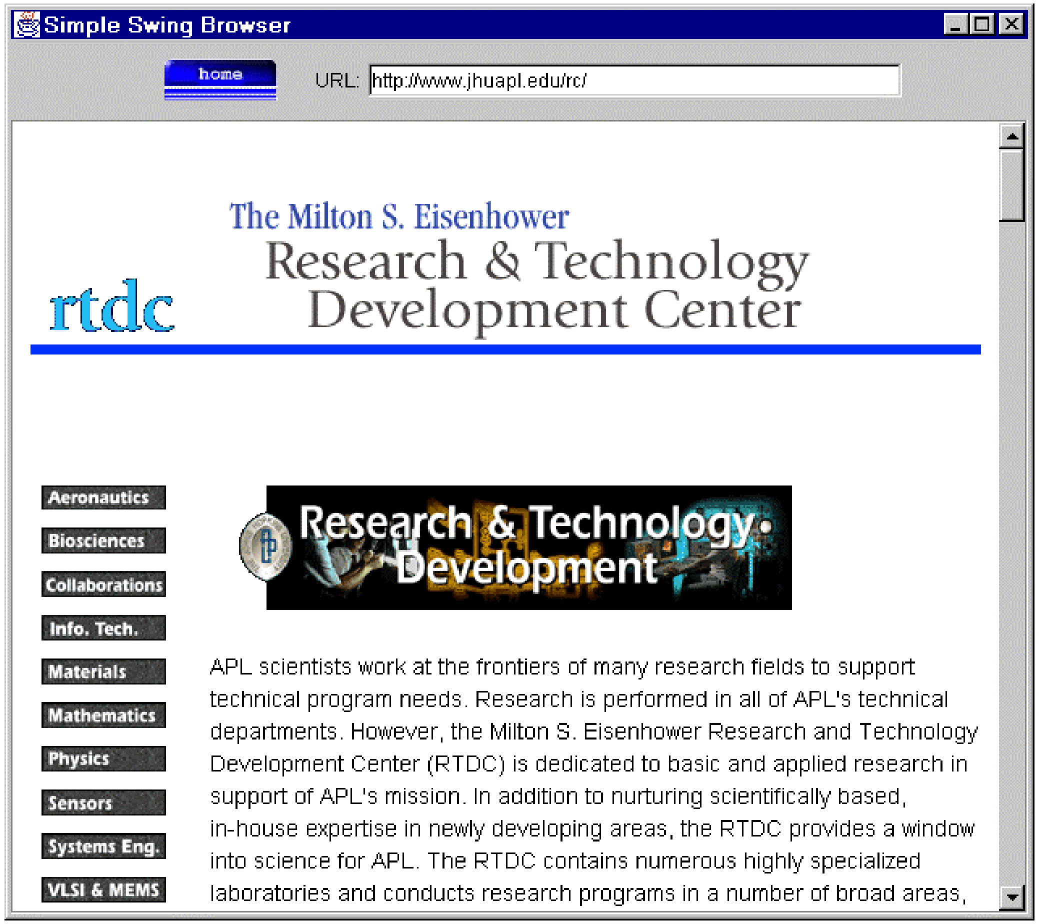
Using the URL Methods, Result

```
> java UrlTest http://www.irs.gov/mission/#squeezing-them-dry
URL: http://www.irs.gov/mission/#squeezing-them-dry
  File:       /mission/
  Host:       www.irs.gov
  Port:       -1
  Protocol:   http
  Reference:  squeezing-them-dry
```

Note: If the port is not explicitly stated in the URL, then the standard port for the protocol is assumed and **`getPort`** returns –1

A Real Browser Using Swing

- The `JEditorPane` class has builtin support for HTTP and HTML



APL scientists work at the frontiers of many research fields to support technical program needs. Research is performed in all of APL's technical departments. However, the Milton S. Eisenhower Research and Technology Development Center (RTDC) is dedicated to basic and applied research in support of APL's mission. In addition to nurturing scientifically based, in-house expertise in newly developing areas, the RTDC provides a window into science for APL. The RTDC contains numerous highly specialized laboratories and conducts research programs in a number of broad areas,

Browser in Swing: Code

```
import javax.swing.*;
import javax.swing.event.*;
...

public class Browser extends JFrame implements HyperlinkListener,
                                               ActionListener {
  private JEditorPane htmlPane;
  ...

  public Browser(String initialURL) {
    ...
    try {
        htmlPane = new JEditorPane(initialURL);
        htmlPane.setEditable(false);
        htmlPane.addHyperlinkListener(this);
        JScrollPane scrollPane = new JScrollPane(htmlPane);
        getContentPane().add(scrollPane, BorderLayout.CENTER);
    } catch(IOException ioe) {
       warnUser("Can't build HTML pane for " + initialURL
                + ": " + ioe);
    }
```

Browser in Swing (Continued)

```
    ...
    Dimension screenSize = getToolkit().getScreenSize();
    int width = screenSize.width * 8 / 10;
    int height = screenSize.height * 8 / 10;
    setBounds(width/8, height/8, width, height);
    setVisible(true);
  }

  public void actionPerformed(ActionEvent event) {
    String url;
    if (event.getSource() == urlField)
      url = urlField.getText();
    else // Clicked "home" button instead of entering URL
      url = initialURL;
    try {
      htmlPane.setPage(new URL(url));
      urlField.setText(url);
    } catch(IOException ioe) {
      warnUser("Can't follow link to " + url + ": " + ioe);
    }
  }
```

Browser in Swing (Continued)

```
...
public void hyperlinkUpdate(HyperlinkEvent event) {
  if (event.getEventType() ==
               HyperlinkEvent.EventType.ACTIVATED) {
    try {
      htmlPane.setPage(event.getURL());
      urlField.setText(event.getURL().toExternalForm());
    } catch(IOException ioe) {
       warnUser("Can't follow link to "
                + event.getURL().toExternalForm() +
                ": " + ioe);
    }
  }
}
```

Summary

- **Opening a socket requires a hostname (or IP address) and port number**
- **A PrintWriter lets you send string data**
  - Use autoflush to send the full line after each println
- **A BufferedReader allows you to read the input one line at a time (readLine)**
  - The `readLine` method blocks until a response is sent
  - For a typical GET request, after the HTTP server sends the response the connection is closed and `readLine` returns `null`
- **StringTokenizer provides simple parsing**
- **The URL and URLConnection classes simplify communication with Web servers**

core WEB programming

Questions?