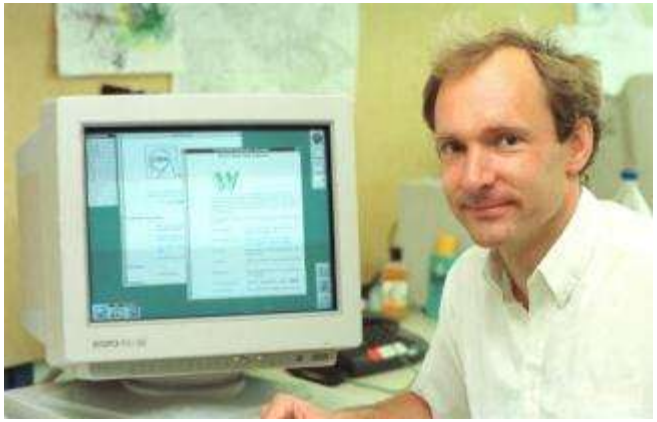
Web clients in Java

Overview

- The World Wide Web
  - History
  - Main components:
    - URLs, HTTP Protocol, HTML
- Web support in Java
  - Connecting and retrieving web data
    - Directly using socket connection on port 80
    - Using built-in Java classes
- Web services
  - Remote functionality via the web
  - REST web services

A short history of the web

- 1989 Tim Berners-Lee at CERN
- 1990 HTTP/0.9, HTML, URLs, first text-based browser
- 1993 Marc Andreesen releases NCSA Mosaic, graphical browser
- 1993 CERN agrees to release protocol royalty-free
- 1994 Andreesen forms Netscape
- 1994 W3C formed, standardizing protocols, encouraging interoperability



"In the Web's first generation, Tim Berners-Lee launched the Uniform Resource Locator (URL), Hypertext Transfer Protocol (HTTP), and HTML standards with prototype Unix-based servers and browsers.

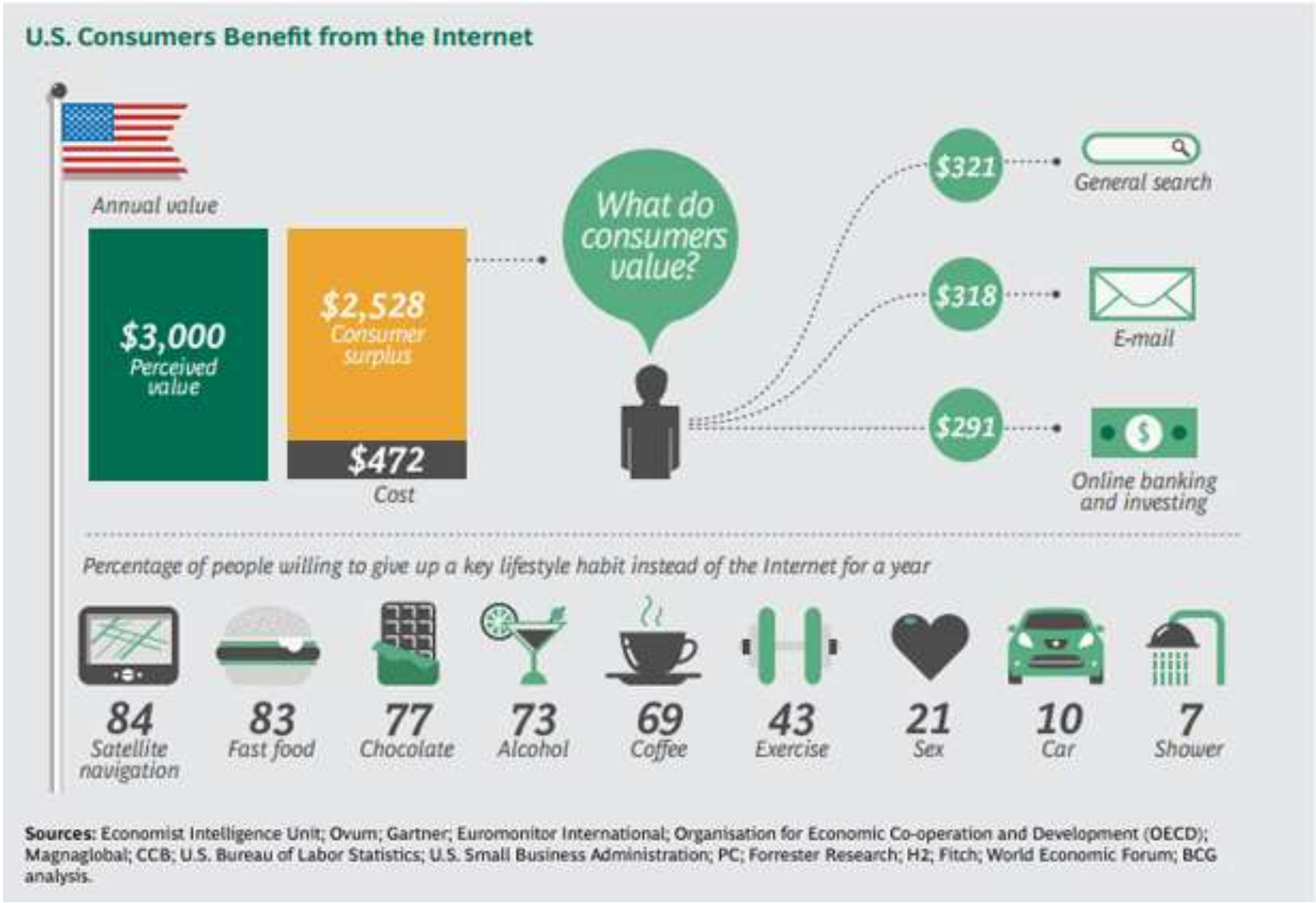
A few people noticed that the Web might be better than Gopher.

In the second generation, Marc Andreessen and Eric Bina developed NCSA Mosaic at the University of Illinois.

Several million then suddenly noticed that the Web might be better than sex.

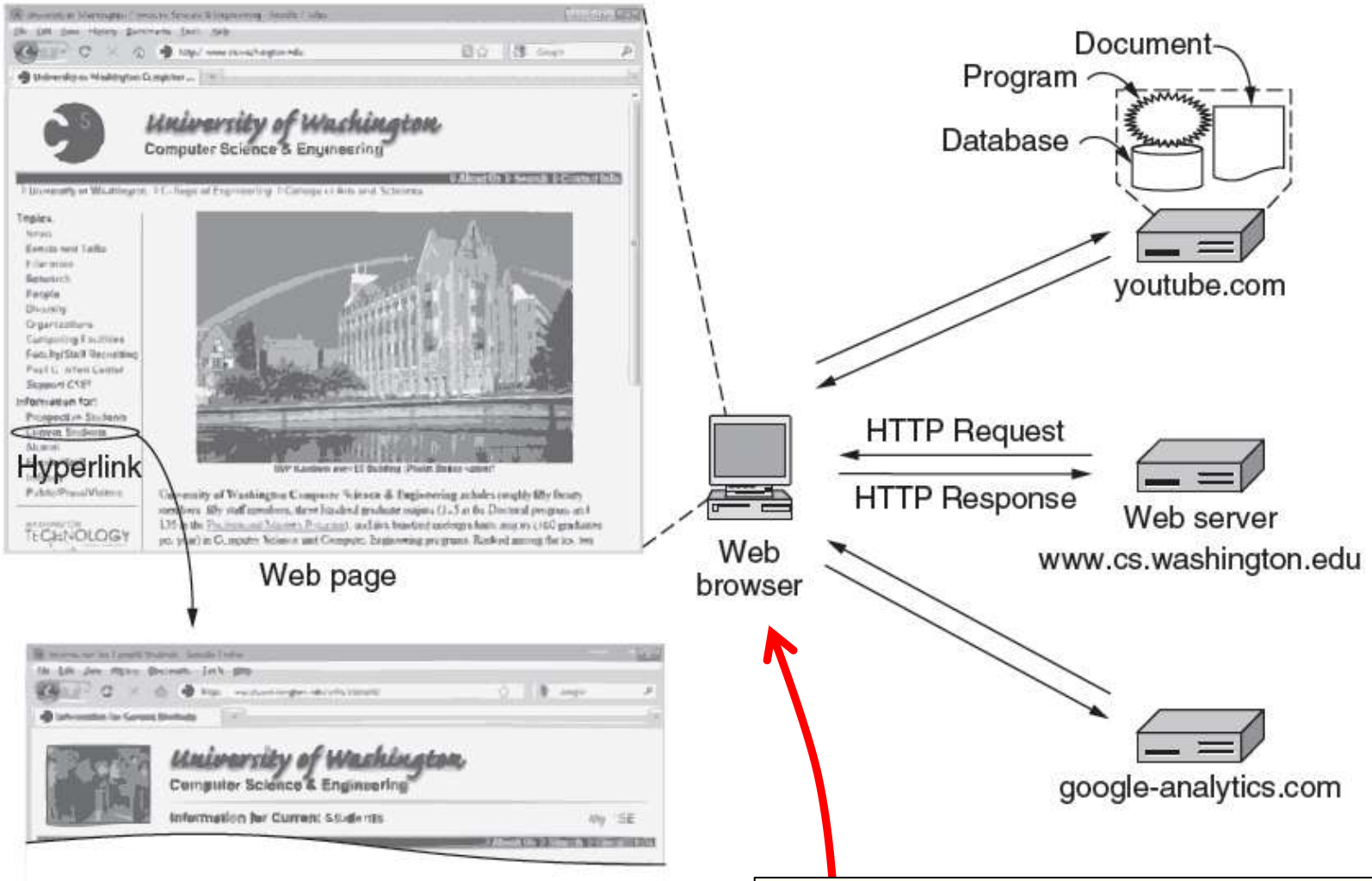
In the third generation, Andreessen and Bina left NCSA to found Netscape..."

-Bob Metcalfe, InfoWorld, August 21, 1995, Vol. 17, Issue 34.



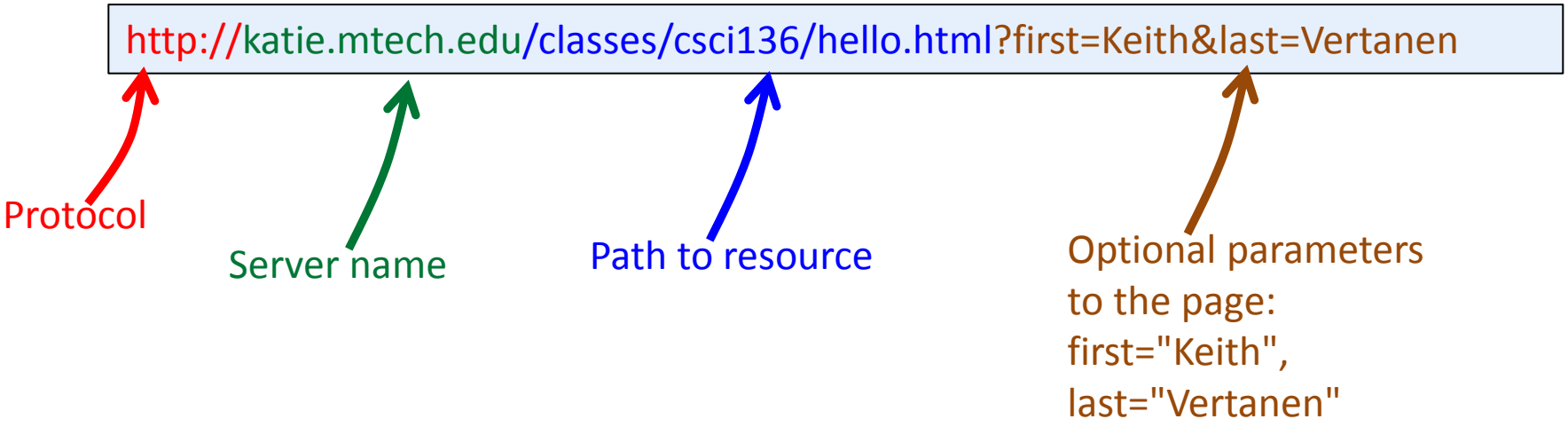
The Boston Consulting Group

Architecture of the web



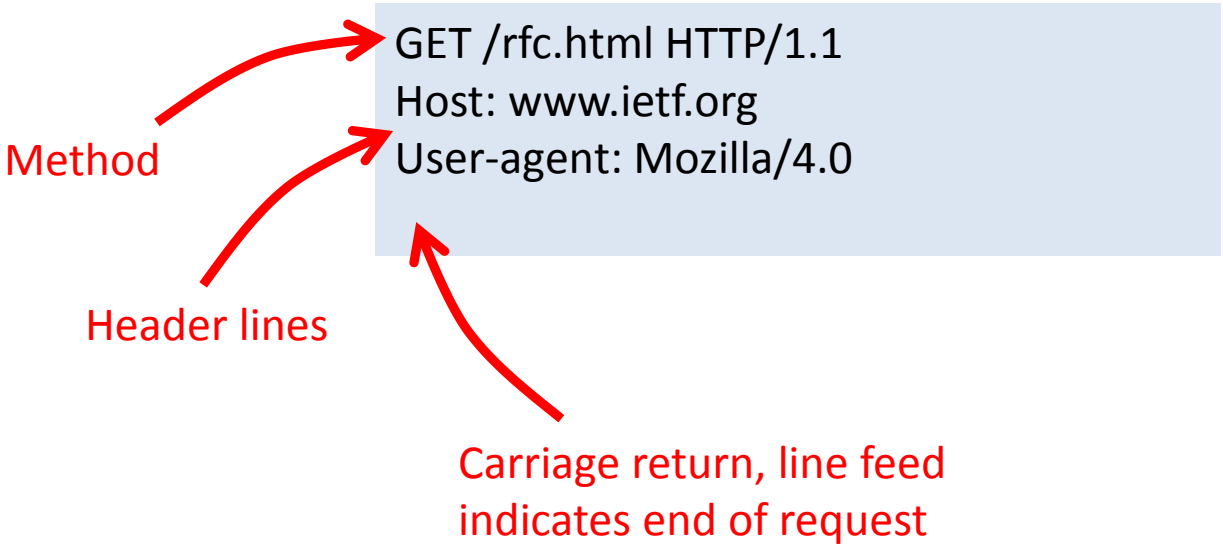
Web components: finding stuff

- Uniform Resource Locator (URL)
  - A page's worldwide name
  - Parts:
    - Protocol (scheme)
    - Domain Name System (DNS) name of machine
    - Hierarchical name that models a file directory structure
    - Optional "GET" name/value parameters sent to page



Web components: HTTP

- HyperText Transfer Protocol (HTTP)
  - Simple request-response protocol
  - Runs over a socket connection
    - TCP over port 80
  - ASCII format request and response headers



HTTP response

- Response from server
  - Status line: protocol version, status code, status phrase
  - Response headers: extra info
  - Body: optional data

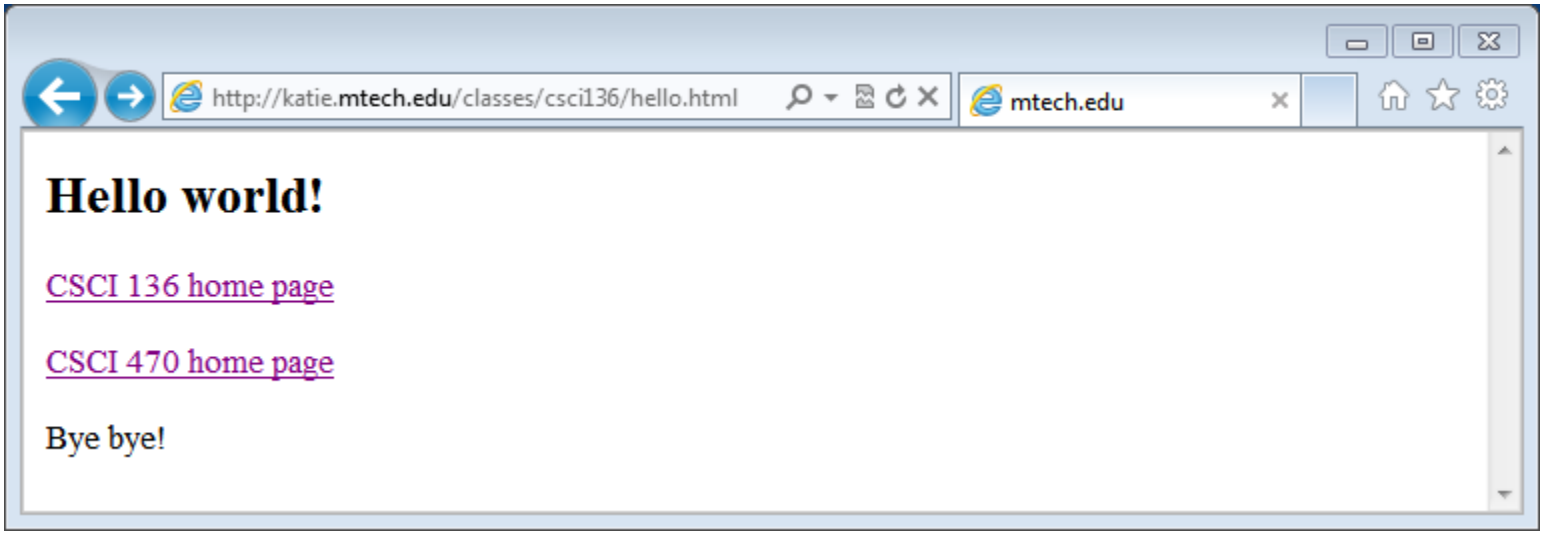
```
HTTP/1.1 200 OK
Date: Thu, 17 Nov 2011 15:54:10 GMT
Server: Apache/2.2.16 (Debian)
Last-Modified: Wed, 14 Sep 2011 17:04:27 GMT
Content-Length: 285

<html> …
```

Blank line denotes end of header, start of payload.

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Web components: HTML

- HyperText Markup Language (HTML)
  - Represents hypertext documents in ASCII form
  - Format text, add images, embed hyperlinks
  - Web browser renders
- Simple and easy to learn
  - Hack up in any text editor
  - Or use a fancy authoring program
- Web page
  - Base HTML file references objects
  - Each object has its own URL




```
<html>
  <body>
  <h2>Hello world!</h2>
    <a href="index.php">CSCI 136 home page</a> <br /><br />
    <a href="http://katie.mtech.edu/classes/csci470/">CSCI 470 home
page</a> <br /><br />
    Bye bye!
   </body>
</html>
```

```
vertanen@katie:~$ cd /home/classes/
vertanen@katie:/home/classes$ cd csci136/
vertanen@katie:/home/classes/csci136$ pwd
/home/classes/csci136
vertanen@katie:/home/classes/csci136$ ls -l hello.html
-rw-rwxr--+ 1 vertanen staff 208 Apr 26 13:11 hello.html
```

Programmatic web access

- Goal:
  - Retrieve from site specified by `args[0]`
  - Retrieve page specified by `args[1]`
  - Print out request and response to console
- Approach 1:
  - Use what we already know about socket communication
  - Establish connection on port 80
  - Construct HTTP request/response per protocol

```
GET /rfc.html HTTP/1.1
Host: www.ietf.org
User-agent: Mozilla/4.0
```

```
HTTP/1.1 200 OK
Date: Thu, 17 Nov 2011 15:54:10 GMT
Server: Apache/2.2.16 (Debian)
Last-Modified: Wed, 14 Sep 2011 17:04:27 GMT
Content-Length: 285

<html> …
```

```
public class GetPageSocket
{
   public static void main(String [] args)
   {
      try
      {
         // Construct the HTTP request
         String request = "";
         request += "GET " + "/" + args[1] + " HTTP/1.1\n";
         request += "Host: " + args[0] + "\n";
         request += "User-agent: Mozilla/4.0\n";
         System.out.println(request);

         // Establish and send request to the web server
         Socket sock = new Socket(args[0], 80);
         PrintWriter writer = new PrintWriter(sock.getOutputStream(), true);
         writer.println(request);
         ...
```

% java GetPageSocket katie.mtech.edu classes/csci136/hello.html
GET /classes/csci136/hello.html HTTP/1.1
Host: katie.mtech.edu
User-agent: Mozilla/4.0

```
// Prepare to receive response form server
InputStreamReader stream = new InputStreamReader(sock.getInputStream());
BufferedReader    reader = new BufferedReader(stream);

// Read the response header, recording content length field
String response = "";
String line     = "";
final String CONTENT_LEN = "Content-Length: ";
int len = 0;
do
{
   line = reader.readLine();
   response += line + "\n";
   if (line.indexOf(CONTENT_LEN) != -1)
      len = Integer.parseInt(line.substring(CONTENT_LEN.length()));
}
while (line.length() > 0); // Continue until blank line

// Read the body of the response, i.e. the actual page
int count = 0;
do
{
   response += (char) reader.read();
   count++;
} while (count < len);
System.out.println(response);
```

GetPageSocket.java

```
// Prepare to receive res
InputStreamReader stream
BufferedReader    reader

// Read the response heac
String response = "";
String line     = "";
final String CONTENT_LEN
int len = 0;
do
{
   line = reader.readLine
   response += line + "\r
   if (line.indexOf(CONTE
      len = Integer.parse
}
while (line.length() > 0)

// Read the body of the r
int count = 0;
do
{
   response += (char) rea
   count++;
} while (count < len);
System.out.println(respor
```

```
% java GetPageSocket katie.mtech.edu classes/csci136/hello.html
GET /classes/csci136/hello.html HTTP/1.1
Host: katie.mtech.edu
User-agent: Mozilla/4.0

HTTP/1.1 200 OK
Date: Thu, 26 Apr 2012 20:02:56 GMT
Server: Apache/2.2.16 (Debian)
Last-Modified: Thu, 26 Apr 2012 19:39:17 GMT
ETag: "300a36-e0-4be9a24b74246"
Accept-Ranges: bytes
Content-Length: 224
Vary: Accept-Encoding
Content-Type: text/html
X-Pad: avoid browser bug

<html>
 <body>
  <h2>Hello world!</h2>
  <a href="index.php">CSCI 136 home page</a> <br /><br />
  <a href="http://katie.mtech.edu/classes/csci470/">CSCI 470
home page</a> <br /><br />
  Bye bye!
 </body>
</html>
```

Programmatic web access

- Approach 1:
  - A bit tedious
- Approach 2:
  - Use built-in Java support for retrieving pages
    - `URL` and `URLConnection` classes
  - Retrieve from full URL given as `args[0]`
  - Print page to console

```
public class GetPage
{
   public static void main(String [] args)
   {
      try
      {
         URL               url    = new URL(args[0]);
         URLConnection     conn   = url.openConnection();
         InputStreamReader stream = new InputStreamReader(conn.getInputStream());
         BufferedReader    reader = new BufferedReader(stream);

         String line     = "";
         String response = "";
         do
         {
            line = reader.readLine();
            if (line != null)
               response
         }
         while (line !=
         reader.close()
         System.out.pri
      }
      catch (MalformedUR
      catch (IOExceptior
   }
}
```

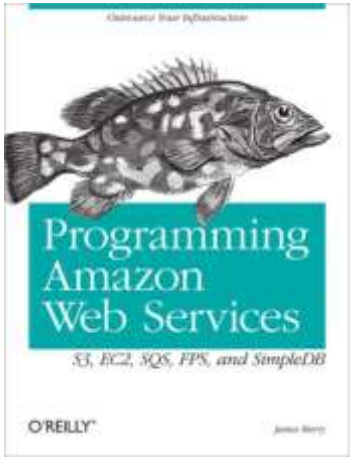
Note: Assumes URL is for a text document not binary data (e.g. an image)

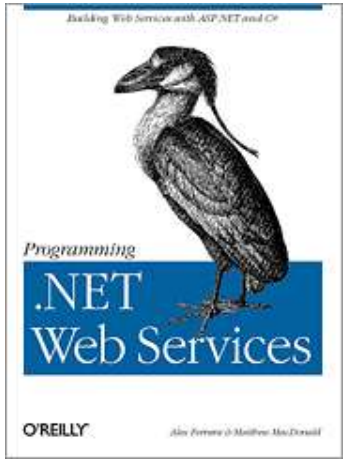
```
% java GetPage http://katie.mtech.edu/classes/csci136/hello.html
<html>
  <body>
    <h2>Hello world!</h2>
    <a href="index.php">CSCI 136 home page</a> <br /><br />
    <a href="http://katie.mtech.edu/classes/csci470/">CSCI 470
home page</a> <br /><br />
    Bye bye!
   </body>
</html>
```

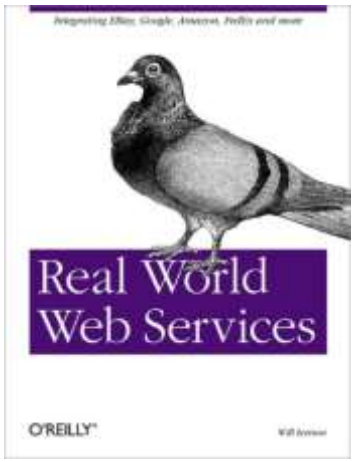
Web service

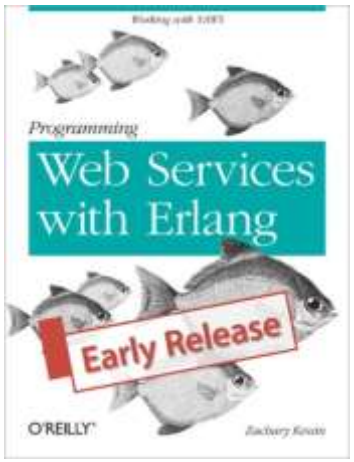
From Wikipedia, the free encyclopedia

A **Web service** is a method of communication between two electronic devices over the web.





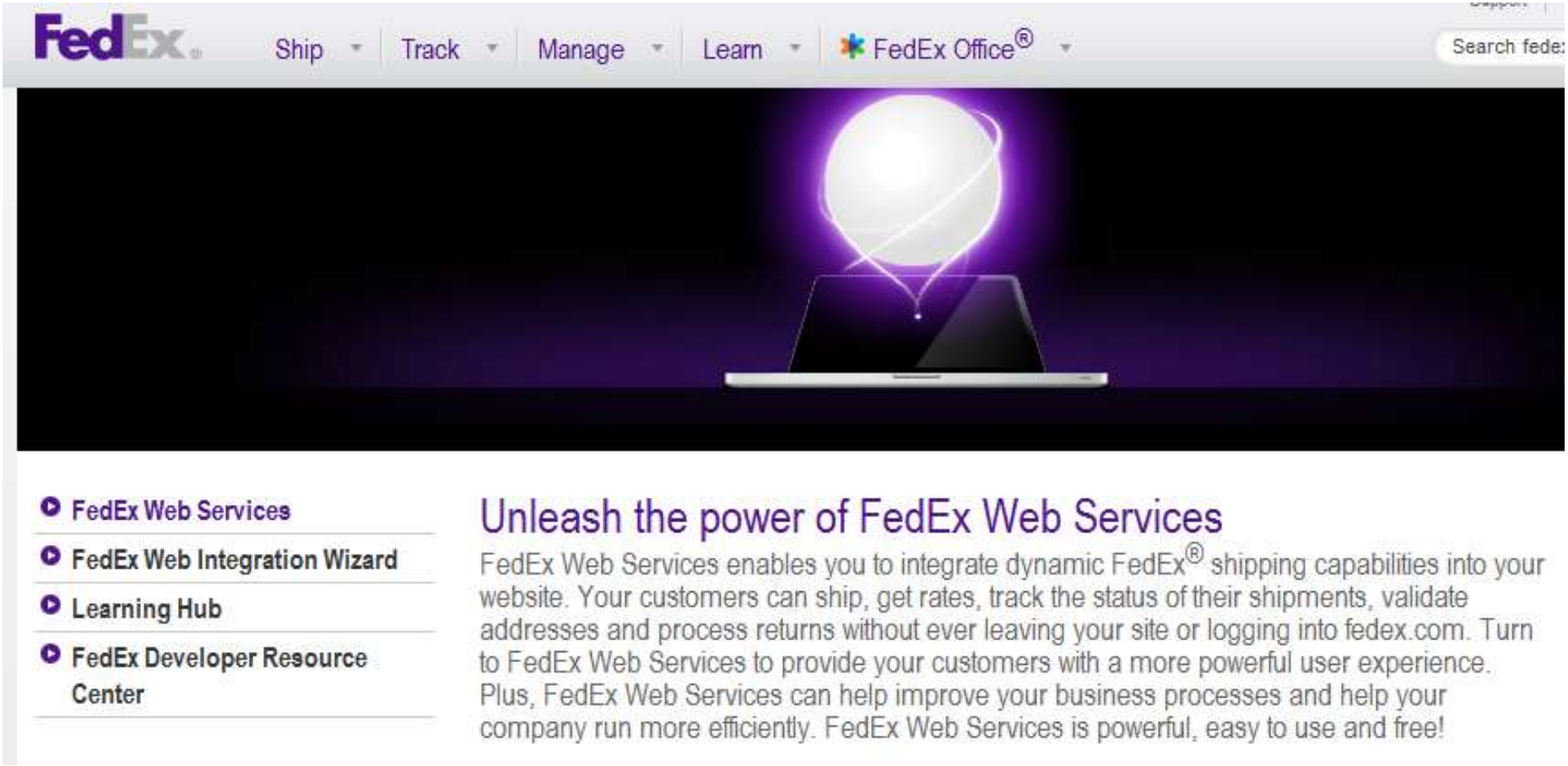




Web services

- Basic idea:
  - Allows others to make use of your:
    - Unique functionality, e.g. translating English to Spanish
    - Unique data, e.g. find out where a FedEx package is
  - Do this over the Internet
    - In a standard way using a known protocol (e.g. HTTP)
  - Possible business uses:
    - Within a company to integrate things
    - Between a company and partners
    - For free, promote your new fangled search engine (e.g. Bing)
    - For money, (e.g. $5/1000 search queries)

FedEx web services



Facebook web services

Google web services

Google Latitude API

Home Docs FAQ Forum Terms

What is the Google Latitude API?

The Google Latitude API allows for websites and programs to integrate with Google Latitude, enabling users to update and read their current location, their location history, and more!

How do I start?

1. Find out the basics on Getting Started.
2. Find out how to use REST to invoke the Google Latitude API by reading the Developer's Guide.
3. Browse the Google Latitude API Reference Guide.
4. Get community support. Join our community and participate in our discussion group.

Google Calendar APIs and Tools

Home Docs FAQ

What are the Google Calendar APIs and Tools?

Google calendar

Life's important events all in one place. Google Calendar offers many ways to create and share content other than the web interface that we all know and love.

Calendar API

The Calendar API lets you incorporate Calendar functionality into your own application or website. You can edit calendars, create and delete events, send invitations, and more.

Calendar Gadgets

Calendar Gadgets let you extend Google Calendar to give your users a custom, content-rich experience. Create status displays, interactive events, and custom user interface controls.

Twitter web services

Home → Documentation

Getting Started

Updated on Wed, 2011-09-28 16:40

Twitter is an information network and communication mechanism that produces more than 200 million tweets a day. The Twitter platform offers access to that corpus of data, via our APIs. Each API represents a facet of Twitter, and allows developers to build upon and extend their applications in new and creative ways. It's important to note that the Twitter APIs are constantly evolving, and developing on the Twitter Platform is not a one-off event.

Accessing a web service

- REST (Representational State Transfer)
  - Make an HTTP request to a specific URL
  - The URL determines what you want from the web service
  - HTTP response in some format
    - e.g. plaintext, XML, JSON

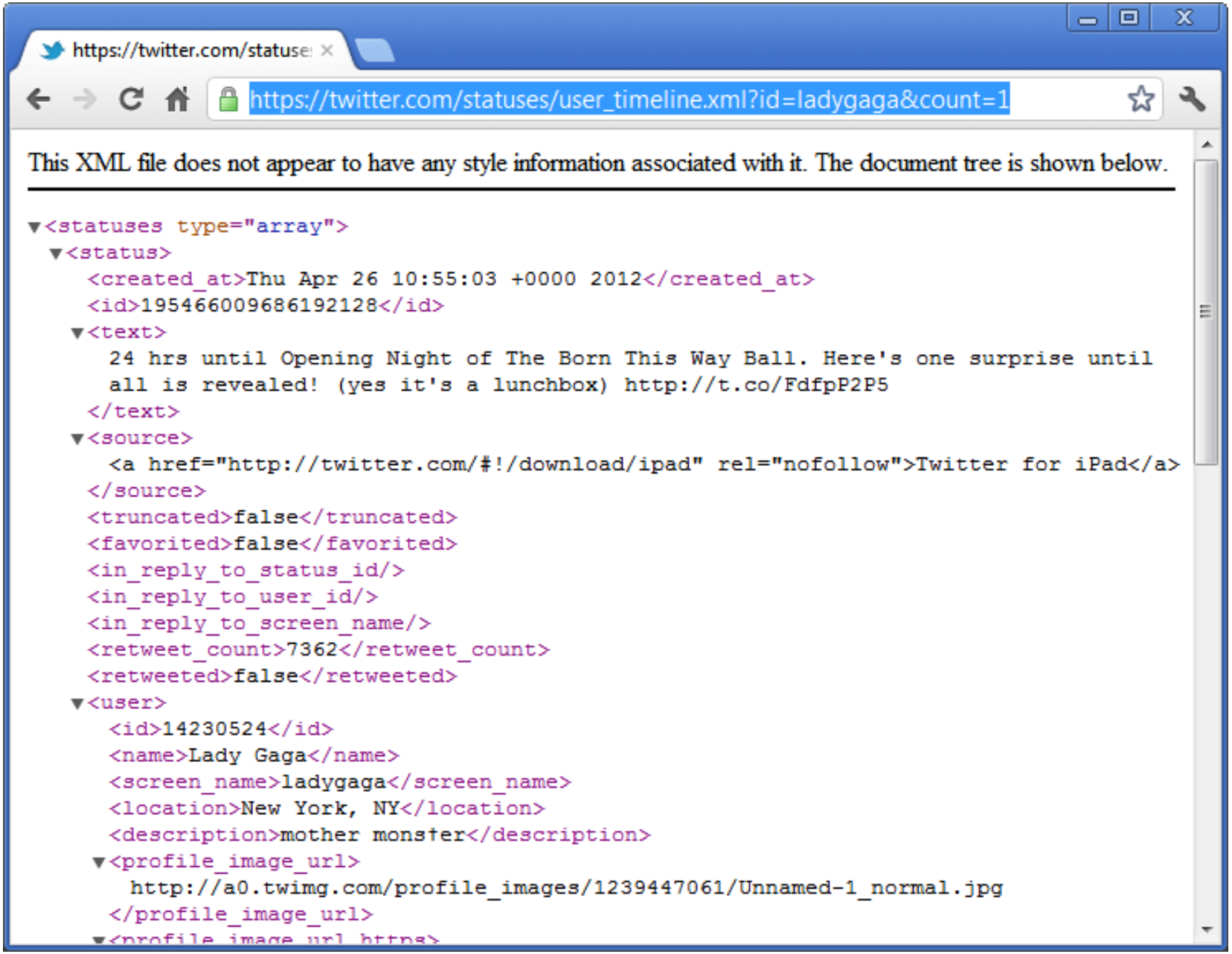
Get images of Lady Gaga
```
http://api.bing.net/xml.aspx?AppId=MYID&Query=Lady+Gaga&Sources=Image
```

Get last 20 tweets by Lady Gaga
```
https://twitter.com/statuses/user_timeline.xml?id=ladygaga
```

Get last tweet by Lady Gaga
```
https://twitter.com/statuses/user_timeline.xml?id=ladygaga&count=1
```

Translate "I love Lady Gaga" into Spanish
```
http://api.bing.net/xml.aspx?AppId=MYID&Query=I+love+Lady+Gaga&Sources=T
ranslation&Translation.SourceLanguage=En&Translation.TargetLanguage=Es
```

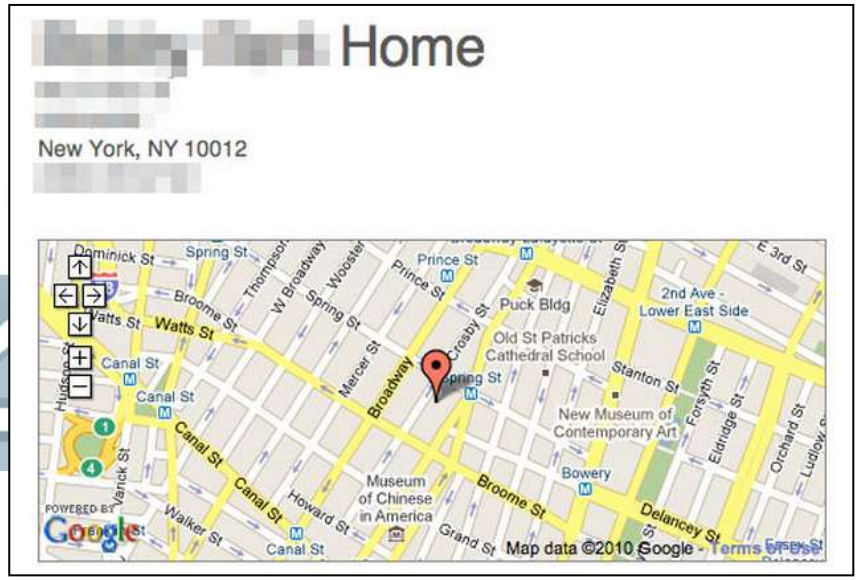
Accessing Twitter via normal web browser interface.



Retrieving last tweet of Lady Gaga via REST web services API.

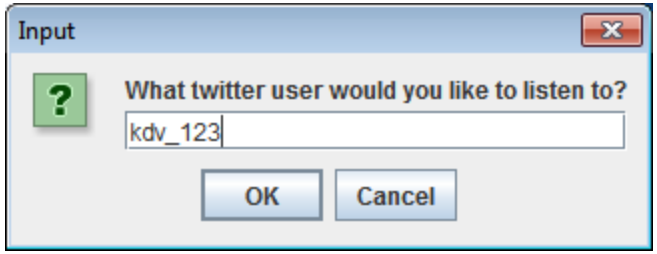
Mashups

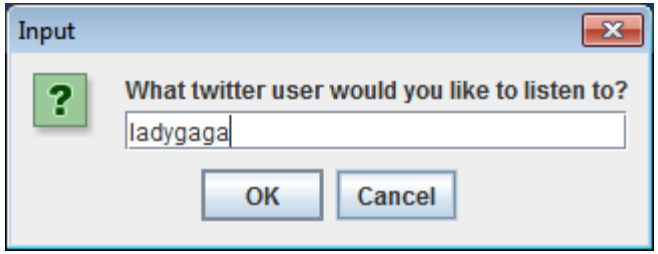
- Mashups
  - A web application hybrid
    - Combine the functionality or data from several web sites
  - Frequently done using web services
    - e.g. Combine Google Maps API with Twitter API to show :

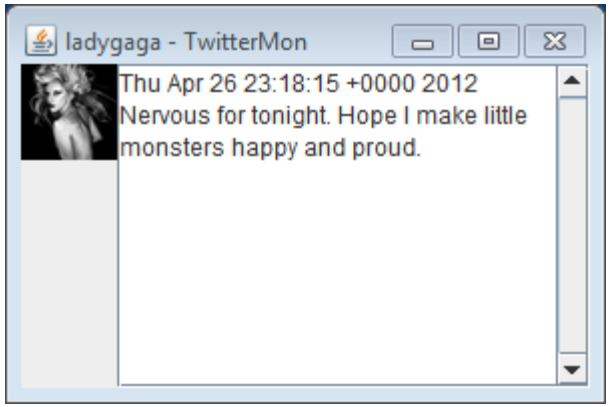


Twitter Monitor

- Goal:
  - GUI application that follows a twitter user
  - Twitter User specified via a dialog box
  - Shows a scrolling text box with tweets
  - Profile image of the user







Summary

- The World Wide Web
  - URL: where to go
  - HTTP: protocol web clients and servers agree to use
  - HTML: how information is formatted (for humans)
- Web support in Java
  - Using `Socket` class
  - Using `URL` and `URLConnection` classes
- Web services
  - Integrate with external knowledge, functionality, etc.
  - REST web services, retrieve data from specific URL
    - Exactly how depends on service provider