Sistem Terdistribusi

TIK-604

Husni.trunojoyo.ac.id

Pemrograman Socket dengan Java: Client

(Pemrograman Aplikasi Terdistribusi)

Husni

husni@trunojoyo.ac.id

Garis Besar Bahasan

- Ikhtisar
- Pemanfaatan PrintWriter dan BufferedReader
- Langkah-langkah menghubungkan client ke server
- Implementasi client jaringan generik
- Format dan parsing String
  - printf
  - StringTokenizer
  - String.split dan ekspresi reguler
- Masalah dengan pemblokiran I/O
- Pengambilan informasi pengguna dari server email
- Berbicara dengan Web server: Secara umum
- Berbicara dengan Web server: Client Java

Ikhtisar

Client vs. Server

- Definisi tradisional
  - Client: Pengguna layanan jaringan
  - Server: Penyedia atau penyalur layanan jaringan
- Masalah dengan definisi tradisional
  - Jika ada 2 program saling bertukar data, tidak jelas posisinya
  - Beberapa situasi (mis. X Windows) terlihat terbalik.
- Cara lebih mudah mengingat perbedaan
  - Server mulai terlebih dahulu. Server tidak menentukan host (cukup port).
  - Client berjalan berikutnya. Client menentukan host server (dan juga port).
- Analogi: jalur telepon perusahaan
  - Menginstal telepon seperti memulai server
  - Ekstensinya seperti port
  - Orang yang menelpon adalah client: dia menentukan host (nomor perusahaan) dan port (ekstensinya)

Client vs. Server (Lanj.)

- Jika server harus berjalan lebih dahulu, mengapa topik tentang client dibahas sebelum server?
  - Client sedikit lebih mudah
  - Client dapat diuji dengan menghubungkannya ke server yang sudah ada, siap di akses di Internet.
- Point: client yang dibuat dengan Java tidak harus (hanya dapat) berkomunikasi dengan server yang ditulis dalam Java.
  - Client dapat berkomunikasi dengan server apa saja yang menerima komuniksi socket (selama client tersebut mengetahui protokol komunikasi yang tepat)
  - ObjectInputStream & ObjectOutputStream memungkinkan program Java mengirimkan struktur data yang rumit.
    - Hanya bekerja untuk komunikasi Java-to-Java.

Bingung Istilah “Stream”?

- InputStream
  - Struktur data level rendah untuk pembacaan dari socket atau file atau sumber input lain. Kita akan membahas **BufferedReader** nanti, dan **ObjectInputStream** pada bagian berikutnya.
- OutputStream
  - Struktur data level rendah untuk pengiriman data ke socket atau file. Kita akan membungkus **PrintWriter** bersamanya, dan **ObjectOutputStream** pada bagian selanjutnya.
- Stream<T> (e.g., Stream<String>)
  - Pembungkus tingkat tinggi bagi array, list dan sumber data lain. Diperkenalkan pada Java 8.
- IntStream, DoubleStream, dll.
  - Spesialisasi primitif dari stream Java 8.

Pemanfaatan PrintWriter

- out.printf(formatString, val1, val2, ...);
  - Mengambil String-formatting-template & nilainya (satu nilai untuk setiap tempat %)
    - Lihat bagian “Utilitas lain” section
    - Di sini, “out” akan dihubungkan ke Socket, bukan ke output standard
- out.print(string) dan out.println(string)
  - Mengambil String tunggal
- String.format(formatString, val1, val2, ...);
  - **printf** *mengeluarkan* suatu String terformat

```
out.printf("Blah %s%n", 7);
```

  - **String.format** *mengembalikan* suatu String terformat

```
String s = String.format("Blah %s%n", 7);
out.print(s);
```

Pemanfaatan BufferedReader

- in.lines()
  - Mengembalikan suatu Stream<String>, dimana setiap entri di dalam Stream datang dari baris terpisah di dalam input. Secara umum sangat powerful dan cara efisien untuk membaca.
    - Jika socket tetap terbuka (open), kita akan menunggu jika mencoba untuk membaca ke “akhir” dari Stream
    - Lihat bagian File I/O untuk banyak contoh pemanfaatannya
- in.readLine()
  - Mengembalikan suatu String tunggal
    - Jika kita mengirimkan suatu String yang dihentikan dengan end-of-line (CR atau pasangan CR/LF pair), maka mengembalikan suatu String *non-null*
    - Jika socket tertutup, mengembalikan null
    - Jika socket terbuka tetapi bukan end-of-line, kita akan menunggu
- in.read()
  - Mengembalikan suatu karakter tunggal

Langkah-langkah Menghubungkan Client ke Server

Gagasan Penting

- Menghubungi Server adalah perkara mudah
  - Langkah-langkah dasarnya sama setiap waktu
- Tetapi berbicara dengan server mungkin cukup berat
  - Tergantung pada protokol yang didefinisikan, format request dan parsing respon mungkin sulit
- Langkah-langkah
  - Membuat suatu Socket
  - Sambungkan suatu PrintWriter ke Socket (untuk pengiriman data)
  - Ambil BufferedReader dari socket (untuk pembacaan data)
  - Lakukan I/O (ini hanya bagian yang berubah-ubah)
  - Tutup socket (otomatis dengan *try-with-resources*)

Langkah-langkah Implementasi Client

1. Buat suatu obyek Socket

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

2. Buat PrintWriter untuk mengirimkan data ke Socket

```
// Argumen terakhir true berarti autoflush -- flush stream
// ketika perintah println dipanggil
PrintWriter out = new PrintWriter(client.getOutputStream(), true);
```

3. Buat BufferedReader untuk membaca respon dari server

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

Langkah-langkah Implementasi Client (Lanj.)

4. Lakukan operasi I/O
   - PrintWriter
     - printf atau println.
   - BufferedReader
     - lines atau readLine.

- Catatan
  - Ada cara standar untuk menghubungi server. Tidak ada cara standar untuk *carry on conversation* dengan server. So, masalah “networking” di Java kebanyakan berkaitan dengan masalah ***String formatting*** (untuk mengirimkan data) dan masalah ***String parsing*** (selama pembacaan data).
  - Untuk komunikasi Java-to-Java, masalahnya lebih simpel karena dapat menggunakan **ObjectInputStream** dan **ObjectOutputStream**.

Langkah-langkah Implementasi Client (Lanj.)

5. Tutup socket tersebut saat selesai

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

- Jika socket dideklarasikan menggunakan *try-with-resources*, ini terjadi secara otomatis, sehingga tidak harus memanggil `client.close` secara eksplisit
- Penutupan socket juga menutup stream input dan output.

Eksepsi

- UnknownHostException
  - Jika host (dilewatkan ke konstruktor) tidak dikenali DNS server
    - Kita dapat menggunakan string IP address sebagai host
- IOException
  - Waktu habis (*Timeout*)
  - Koneksi ditolak oleh server
  - Interupsi atau masalah lain yang tak diharapkan
    - Server menutup koneksi bukan karena terbaca error: null dikembalikan dari “readLine” dan “lines”
- Catatan
  - Socket mengimplementasikan **AutoCloseable**, sehingga dapat digunakan gagasan *try-with-resources* seperti pada operasi I/O.

```
try(Socket client = new Socket(...)) { ... }
```

Kelas Bantuan: SocketUtils

- Gagasan
  - Cara umum adalah membuat BufferedReader dan PrintWriter dari suatu Socket. Utilitas kecil SocketUtils akan menyederhanakan sintaks.
- Tanpa SocketUtils (untuk Socket s)

```
PrintWriter out = new PrintWriter(s.getOutputStream(), true);
BufferedReader in = new BufferedReader(new
  InputStreamReader(s.getInputStream()));
```

- Dengan SocketUtils (untuk Socket s)

```
PrintWriter out = SocketUtils.getWriter(s);
BufferedReader in = SocketUtils.getReader(s);
```

Kelas Pembantu: SocketUtils

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

public class SocketUtils {

  public static BufferedReader getReader(Socket s) throws IOException {
    return(new BufferedReader(new InputStreamReader(s.getInputStream())));
  }

  public static PrintWriter getWriter(Socket s) throws IOException {
    // Second argument of true means autoflush
    return(new PrintWriter(s.getOutputStream(), true));
  }

  private SocketUtils() {} // Uninstantiatable class
}
```

Kelas Basis dari Client Jaringan yang *Reusable*

Gagasan

- Menghubungi server hampir selalu sama
  - Panggil konstruktur Socket, gunakan blok try/catch untuk UnknownHostException dan IOException, tutup Socket
- Berbicara dengan server hampir selalu berbeda
  - Lakukan I/O dengan PrintWriter dan BufferedReader mengikuti protokol yang digunakan.
- Buat kelas NetworkClient yang *reusable*
  - Secara otomatis kerjakan langkah-langkah yang tidak berubah
  - Letakkan kode untuk operasi I/O sesungguhnya (yang melakukan perubahan)
- Langkah-langkah
  - Perluas NetworkClient
  - Perbaiki handleConnection(Socket s)
  - Instansiasi kelas dan panggil ***connect***

Client Jaringan Generik

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

public abstract class NetworkClient {

  private String host;
  private int port;

  public String getHost() { return(host); }
  public int getPort() { return(port); }

  /** Daftarkan host dan port.
    * Koneksi dibangun saat kita panggil connect. */

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

Client Jaringan Generik (Lanj.)

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

  /** Metode handleConnection() akan di-override
   *  ketika pembuatan client jaringan.  */
  protected abstract void handleConnection(Socket client) throws IOException;
}
```

Contoh Client

```
public class NetworkClientTest extends NetworkClient {
  public NetworkClientTest(String host, int port) {
    super(host, port);
  }

  @Override
  protected void handleConnection(Socket client) throws IOException {
    PrintWriter out = SocketUtils.getWriter(client);
    BufferedReader in = SocketUtils.getReader(client);
    out.println("Client Jaringan Generik");
    System.out.printf ("Client Jaringan Generik:%n" +
                       "Terhubung ke '%s' dan direspon '%s'.%n",
                        getHost(), in.readLine());
  }
```

Contoh Client (Lanj.)

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

Contoh Client: Hasil

```
java NetworkClientTest localhost 21
Client Jaringan Generik:
 Terhubung ke 'localhost' dan direspon '220-FileZilla Server version 0.9.41 beta'.
```

Baris pertama dari pesan welcome dari server FTP FileZilla di localhost

```
java NetworkClientTest localhost 110
Client Jaringan Generik:
 Terhubung ke 'localhost' dan direspon '+OK <576543687.6773@localhost>, POP3
  server ready.'.
```

```
java NetworkClientTest djxmmx.net 17
Client Jaringan Generik:
 Terhubung ke 'djxmmx.net' dan direspon '"Where's there's smoke, there's Snoop
  Dogg..."'.
```

Baris pertama quote dari server “quote of the day” (qotd) pada djxmmx.net

Temukan lebih banyak server qotd di https://en.wikipedia.org/wiki/QOTD19

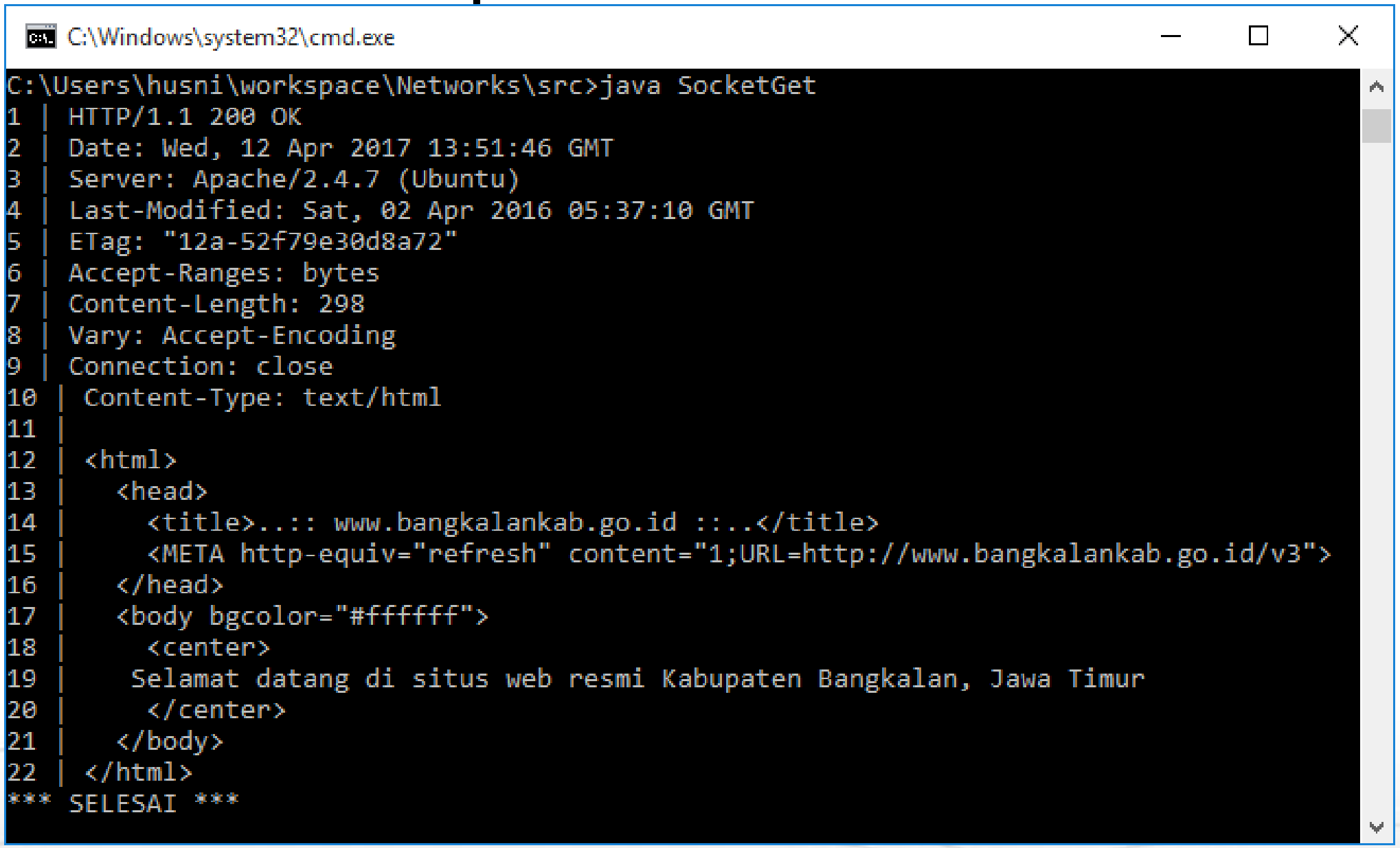
ClientTester.java (Simpel, Tidak *Reusable*)

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

public class ClientTester {
    public static void main(String[] args) {
        try {
            Socket client = new Socket("localhost", 110);                    //1
            Scanner input =new Scanner(client.getInputStream());             //2
            //PrintWriter output = new PrintWriter(client.getOutputStream(),true);  //3
            System.out.printf ("Client Jaringan Generik:%n" +                //4
               "Terhubung ke Localhost Port 21 %n" +
               "Respon Server: '%s'.%n", input.nextLine());
            client.close();                                                  //5.
        } catch(IOException ioEx){ ioEx.printStackTrace(); }
    }
}
```

Web Browser “Simpel”

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

public class SocketGet {
    public static void main(String[] args) throws Exception {
        try {
            Socket socket = new Socket("bangkalankab.go.id",80);
            PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())));
            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out.println("GET /index.html HTTP/1.0");
            out.println();
            out.flush();

            String inputLine;
            int count = 0;

            while ((inputLine = in.readLine()) != null) {
                count++;
                System.out.print(count + " | ");
                System.out.println(inputLine);
            }

            in.close(); socket.close();
            System.out.println("*** SELESAI ***");
        } catch (Exception e) { e.printStackTrace(); }
    }
}
```

Web Browser “Simpel”: Eksekusi



Tambahan:
Pembentukan dan Penguraian String

Strategi Pembentukan & Penguraian

- Gagasan
  - Mudah: Hubungkan ke server dan buat Reader/Writer
  - Sulit: Pembentukan request dan penguraian respon
- Pendekatan
  - Pembentukan request
    - Gunakan printf (dan String.format)
  - Penguraian respon: mudah tapi lemah
    - Gunakan StringTokenizer
  - Penguraian respon: lebih tangguh dan cukup rumit
    - Gunakan String.split dengan regular expressions
  - Penguraian respon: palig bagus dan sangat sulit
    - Gunakan Pattern dan pustaka regex penuh (tidak dibahas)

Penguraian String dengan StringTokenizer

- Gagasan
  - Bangun suatu tokenizer dari string awal
  - Retrieve satu token pada satu waktu dengan nextToken
    - Token bermakna suatu bagian dari string tanpa *delimeters*
  - Kita dapat melihat berapa banyak token sisanya (countTokens) atau hanya menguji apakah jumlah token tersisa tidak nol (hasMoreTokens)

```
StringTokenizer tok = new StringTokenizer(input, delimiters);

while (tok.hasMoreTokens()) {
  doSomethingWith(tok.nextToken());
}
```

StringTokenizer

- Konstruktur
  - `StringTokenizer(String input, String delimiters)`
  - `StringTokenizer(String input, String delimiters, boolean includeDelimiters)`
  - `StringTokenizer(String input)`

    - Himpunan delimiter default: " \t\n\r\f" (whitespace)
- Metode-metode
  - `nextToken()`
  - `nextToken(String delimiters)`
  - `countTokens()`
  - `hasMoreTokens()`

Tokenizer Interaktif: Contoh

```
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 (“Penggunaan: java TokTest string delimiters");
    }
  }
}
```

Tokenizer Interaktif: Hasil

```
> Java TokTest http://www.detik.com/sepakbola/ :/.
http
www
detik
com
sepakbola

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

Penguraian String dengan Metode split

- Penggunaan dasar

```
String[] tokens = mainString.split(regexString);
```

- Perbedaan dari StringTokenizer
  - String lengkap adalah *delimiter* (bukan himpunan delimiter 1 karakter seperti dengan StringTokenizer)
    - "foobar".split("ob") mengembalikan {"fo", "ar"}
    - "foobar".split("bo") mengembalikan {"foobar"}
  - Kita dapat menggunakan regular expressions dalam delimiter
    - ^, $, *, +, ., dll untuk permulaan String, akhir String, 0 atau lebih, 1 atau lebih, karakter apa saja, dll.
    - Lihat http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html#sum
  - Kecuali jika memakai "+", string kosong dikembalikan antara delimiter berturutan
    - "foobar".split("o") mengembalikan {"f", "", "bar"}
    - "foobar".split("o+") mengembalikan {"f", "bar"}

Pentingnya Regular Expressions

- Gagasan
  - Untuk Strings, metode split, matches, dan replaceAll menggunakan regular expressions.
  - Banyak bahasa lain menggunakan regular expressions dengan sintaks mirip. Pemahaman sintaks regex merupakan skill penting bagi setiap programmer.

- Tutorial
  - http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html#sum
  - http://docs.oracle.com/javase/tutorial/essential/regex/
  - http://regexr.com/ (Bagus sekali, tidak khusus Java)

Dari Randall Munroe & xkcd.com

Splitter String Interaktif: Contoh

```
public class SplitTest {
  public static void main(String[] args) {
    if (args.length == 2) {
      String[] tokens = args[0].split(args[1]);
      for(String token: tokens) {
        if (token.length() != 0) { System.out.println(token); }
      }
    } else {
      System.out.println(“Pemanfaatan: java SplitTest string delimeters");
    }
  }
}
```

Splitter String Interaktif: Hasil

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

> java SplitTest http://www.microsoft.com/~gates/ :/.
http
www.microsoft.com/~gates/


> java SplitTest http://www.microsoft.com/~gates/ [:/.]+
http
www
microsoft
com
~gates
```

Ini adalah contoh sebelumnya (menggunakan StringTokenizer) untuk perbandingan

Masalah dengan Pemblokiran IO

Tambahan: Bicara Interaktif dengan Server

- Telnet
  - Banyak orang berpikir telnet sebagai tool untuk log in ke remote server pada port login default (23). Sesungguhnya ia lebih umum: tool untuk koneksi ke remote server pada port tertentu dan secara interaktif mengirimkan perintah dan melihat hasilnya
  - Sebelum menuliskan client Java untuk koneksi ke Server, sebaiknya telnet-lah server tersebut terlebih dahulu, coba beberapa perintah secara interaktif, dan lihat apakah protokol bekerja seperti yang dipikirkan
- Mengaktifkan telnet pada Windows 7, 8 atau 10
  - Mulai dari Windows Vista, telnet dinon-aktifkan secara default
    - Googling "install telnet windows" untuk mengetahui bagaimana mengaktifkannya
    - Kita mungkin pula perlu menyalakan local echo
    - Client telnet di Linux jauh lebih nyaman
    - Putty adalah client telnet gratis yang terkenal di Windows.

Verifikasi Email Addresses dengan Telnet

Contoh berbicara dengan SMTP server

```
> telnet apl.jhu.edu 25
Trying 128.220.101.100 ...Connected …
220 aplcenmp.apl.jhu.edu Sendmail …
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.
```

Client Java untuk Verifikasi Email Address

```
public class AddressVerifier extends NetworkClient {
  private String username;

  public AddressVerifier(String username, String hostname, int port) {
    super(hostname, port);
    this.username = username;
  }

  public static void main(String[] args) {
    if (args.length != 1) {   usage();   }
    MailAddress address = new MailAddress(args[0]);
    new AddressVerifier(address.getUsername(),  address.getHostname(), 25);
  }
```

Client Java untuk Verifikasi Email Address (Lanj.)

```
  protected void handleConnection(Socket client) throws IOException {
    PrintWriter out = SocketUtils.getWriter(client);
    InputStream rawIn = client.getInputStream();
    byte[] response = new byte[1000];
    // Clear out mail server's welcome message.
    rawIn.read(response);
    out.println("EXPN " + username);
    // Read the response to the EXPN command.
    int numBytes = rawIn.read(response);
    // The 0 means to use normal ASCII encoding.
    System.out.write(response, 0, numBytes);
    out.println("QUIT");
  }
  ...
}
```

Poin penting: readLine dapat digunakan hanya jika:

- Diketahui jumlah baris data akan dikirimkan (panggil readLine sebanyak itu)
- Server akan menutup koneksi ketika selesai, begitu pula server HTTP (panggil readLine sampai diperoleh null)

Kelas Bantuan: MailAddress

```
public class MailAddress {
  private String username, hostname;

  public MailAddress(String emailAddress) {
    String[] pieces = emailAddress.split("@");
    if (pieces.length != 2) {
      System.out.println("Illegal email address");
      System.exit(-1);
    } else {
      username = pieces[0];
      hostname = pieces[1];
    }
  }

  public String getUsername() { return(username); }
  public String getHostname() { return(hostname); }
}
```

Verifikasi Alamat Email: Hasil

```
> 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
```

Berbicara dengan Web Server

Pendekatan Umum

Perintah “GET” HTTP

untuk URL http://somehost/somepath

HTTP 1.0	HTTP 1.1
Terhubung ke somehost pada port 80 GET /somepath HTTP/1.0 Baris kosong	Terhubung ke somehost pada port 80 GET /somepath HTTP/1.1 Host: somehost Connection: close Baris kosong

Catatan:

- Server menutup koneksi ketika ia selesai mengirimkan halaman. Ini memudahkan Java membaca halaman: cukup gunakan lines() dan proses Stream<String> atau readLine() sampai diperoleh null.
- Alasan bagi header Host dalam HTTP 1.1 adalah agar terkoneksi ke host yang menggunakan virtual hosting, yaitu situs yang menjadi host bagi banyak nama domain pada mesin yang sama. Sebagian besar situs kecil dan sedang menggunakan virtual hosting.
- Alamat setelah “GET” dinamakan URI.
- Top-level URL: untuk URL http://somehost, browser memperlakukannya sebagai http://somehost/, dan mengirimkan / sebagai URI untuk request GET.

Contoh: Telnet ke Web Server

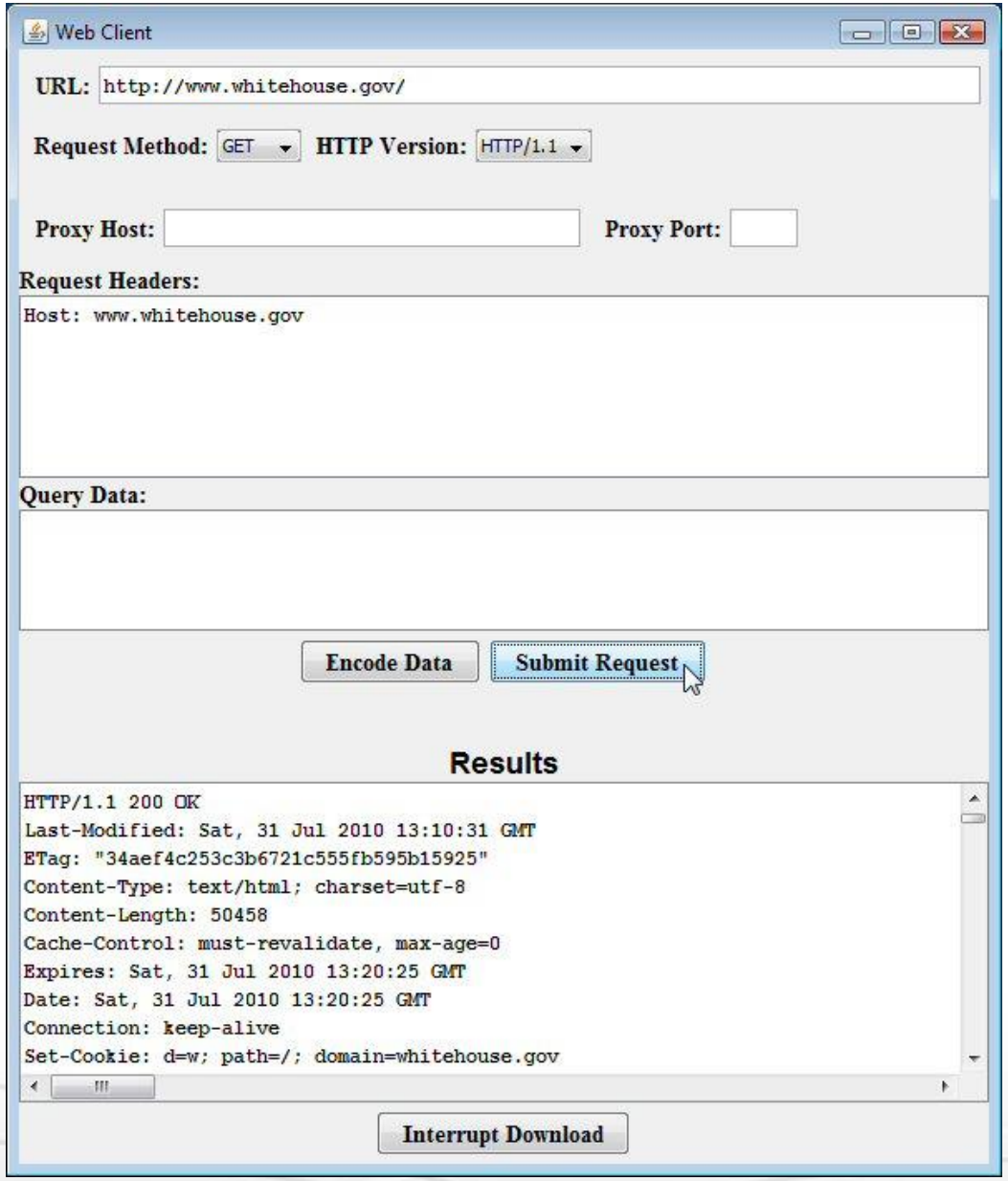
URL: http://www.apl.jhu.edu/~hall/

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

HTTP/1.1 200 OK
...
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<!DOCTYPE HTML>
<html>
...
</html>Connection closed by foreign host.
Unix>
```

Berbicara Interaktif dengan Web Server

- Masalah
  - Client Telnet Windows tidak bagus kerjanya untuk urusan ini
    - Client Telnet di Linux, Solaris dan MacOS bekerja bagus untuk ini.
- Solusi: WebClient
  - GUI simpel untuk komunikasi dengan HTTP server
  - Pengguna dapat secara interaktif menentukan:
    - URL dengan host, port dan URI
    - HTTP request headers
  - Request HTTP dikerjakan dalam thread tersendiri
  - Dokumen respon diletakkan dalam suatu *scrollable text area*
  - Download semua file sumber untuk WebClient dari home page tutorial.

WebClient: Contoh



Berbicara dengan Web Server:
Client Java

Mendapatkan URI dari Host

```
public class UriRetriever extends NetworkClient {
  private String uri;

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

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

Mendapatkan URI dari Host

```
  // Aman menggunakan in.lines() atau loop dan
  // lakukan in.readLine() karena Web server menutup
  // koneksi setelah mengirimkan data.

  protected void handleConnection(Socket client) throws IOException {
    PrintWriter out = SocketUtils.getWriter(client);
    BufferedReader in = SocketUtils.getReader(client);
    out.printf("GET %s HTTP/1.1\r\n", uri);
    out.printf("Host: %s\r\n", getHost());
    out.printf("Connection: close\r\n\r\n");
    in.lines().forEach(System.out::println);
  }
}
```

Meng-Echo-kan Semua Baris: Java 8 vs. Java 7

- Java 8 (slide sebelumnya)

```
in.lines().forEach(System.out::println);
```

- Java 7

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

Kelas Bantuan: Penguraian URL (URLParser)

```
public class UrlParser {
  private String host;
  private int port = 80;
  private String uri;

  public UrlParser(String url) {
    StringTokenizer tok = new StringTokenizer(url);
    String protocol = tok.nextToken(":");
    checkProtocol(protocol);
    host = tok.nextToken(":/");
    try {
      uri = tok.nextToken("");
      if (uri.charAt(0) == ':') {
        tok = new StringTokenizer(uri);
        port = Integer.parseInt(tok.nextToken(":/"));
        uri = tok.nextToken("");
      }
    } catch(NoSuchElementException nsee) {
      uri = "/";
    } ...
```

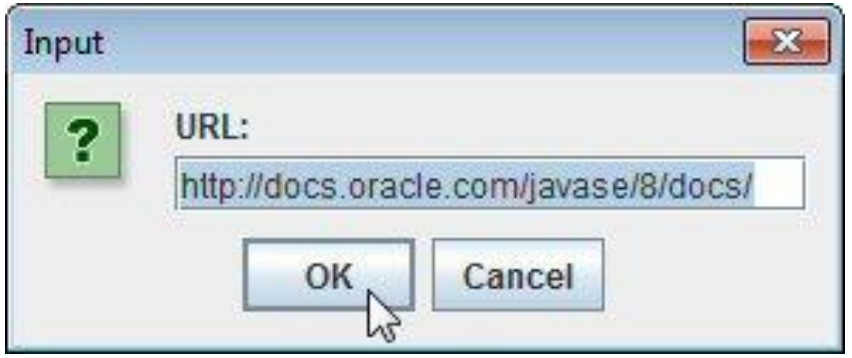
Kelas Pe-retrieve URL (URLRetriever)

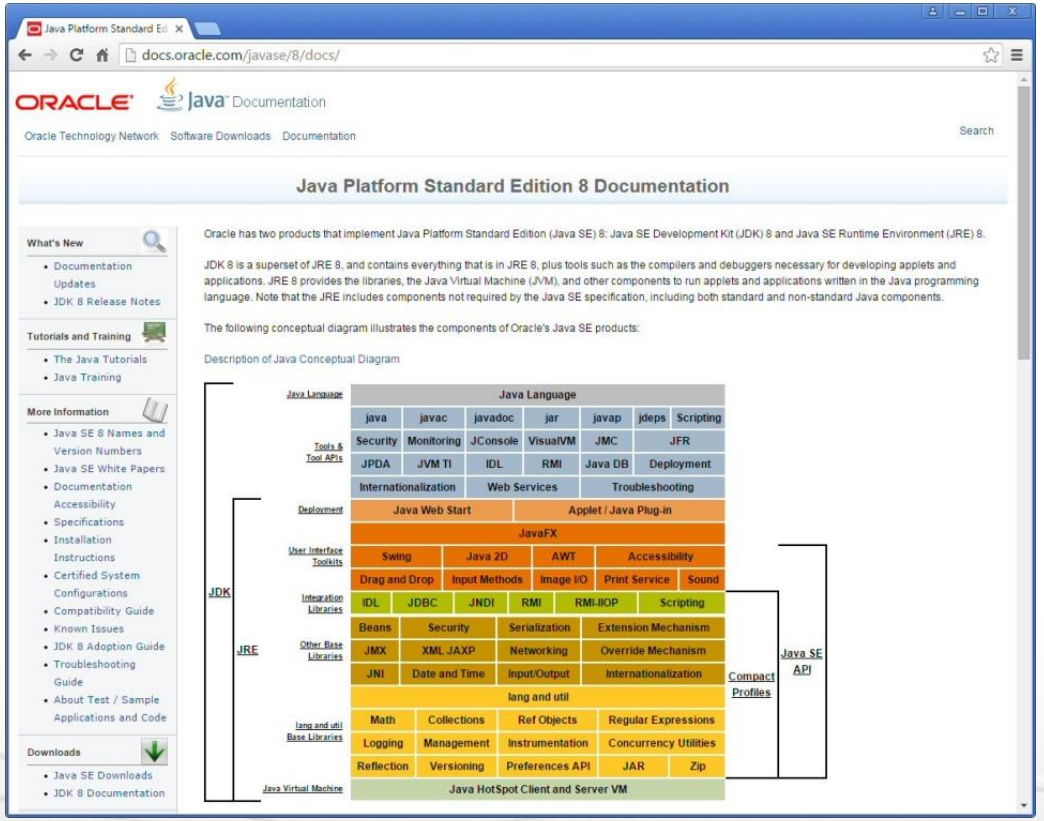
```
public class UrlRetriever {
  public static void main(String[] args) {
    String defaultAddress = "http://docs.oracle.com/javase/8/docs/";
    String address = JOptionPane.showInputDialog("URL:", defaultAddress);

    if (address == null) { address = defaultAddress; }
    UrlParser parser = new UrlParser(address);

    UriRetriever uriClient =
      new UriRetriever(parser.getHost(), parser.getPort(), parser.getUri());
    uriClient.connect();
  }
}
```

UrlRetriever Beraksi

- Nomor port tidak disebutkan

```
HTTP/1.1 200 OK
Server: Apache
...

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE HTML …>
<html ...>
<head>...</head>
<body>
...
</body>
</html>
```

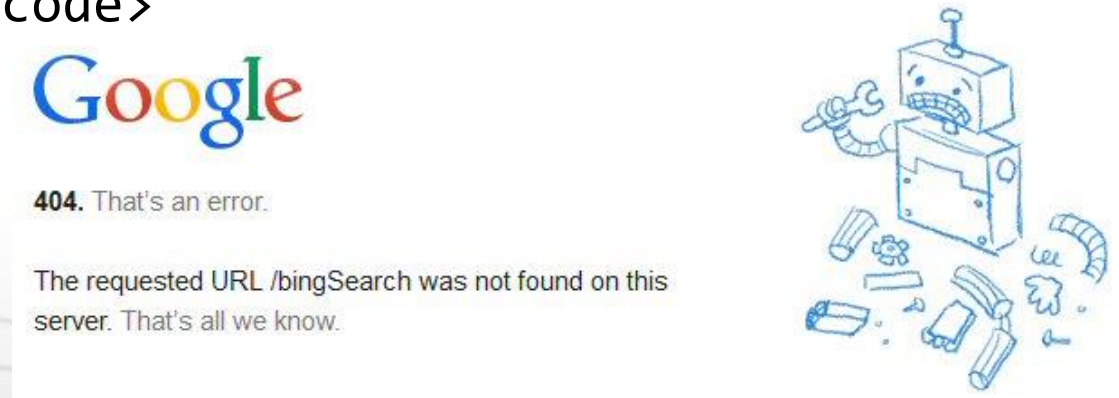




UrlRetriever Beraksi (Lanj.)

- Nomor port disebutkan


```
HTTP/1.1 404 Not Found
Content-Type: text/html
...

<!DOCTYPE html>
<html lang=en>
...
<p><b>404.</b> <ins>That’s an error.</ins>
<p>The requested URL <code>/bingSearch</code>
was not found on this server.
<ins>That’s all we know.</ins>
...
```



Berbicara dengan Web Server:

Menggunakan Kelas URL

Menulis Web Browser

- Kita telah memuat browser dalam 3 halaman
  - Bagus, tidak pasti, karena kita hanya menampilkan HTML kasar, tidak menformatnya atau menangani interaksi dengan *end-user*
  - Masih, tidak bagus untuk kode singkat

- Dapat dijadikan lebih baik
  - Penting memahami bagaimana menghubungi server secara eksplisit, karena dalam hidup nyata kemungkinan kita menuliskan server tertentu dengan protokol tertentu.
  - Tetapi, untuk kasus khusus koneksi ke Web server, Java punya dukungan bawaan
    - Buat suatu URL
    - Panggil openConnection
    - Bungkuskan *BufferedReader* terhadap *InputStreamReader*
    - Gunakan lines atau readLine

Metode Bantuan: Mendapatkan Baris Halaman

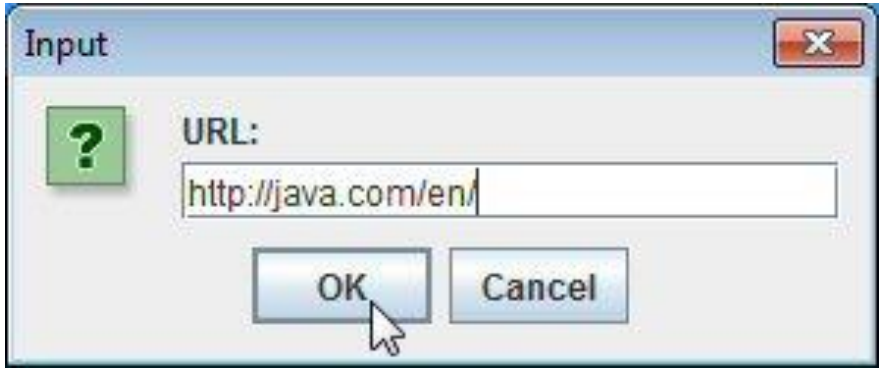
```
public class WebUtils {
  public static Stream<String> lines(String address) {
    try {
      URL url = new URL(address);
      BufferedReader in =
        new BufferedReader(new InputStreamReader(url.openStream()));
      return(in.lines());
    } catch(IOException ioe) {
      System.err.println(ioe);
      return(Stream.empty());
    }
  }
  ...
}
```

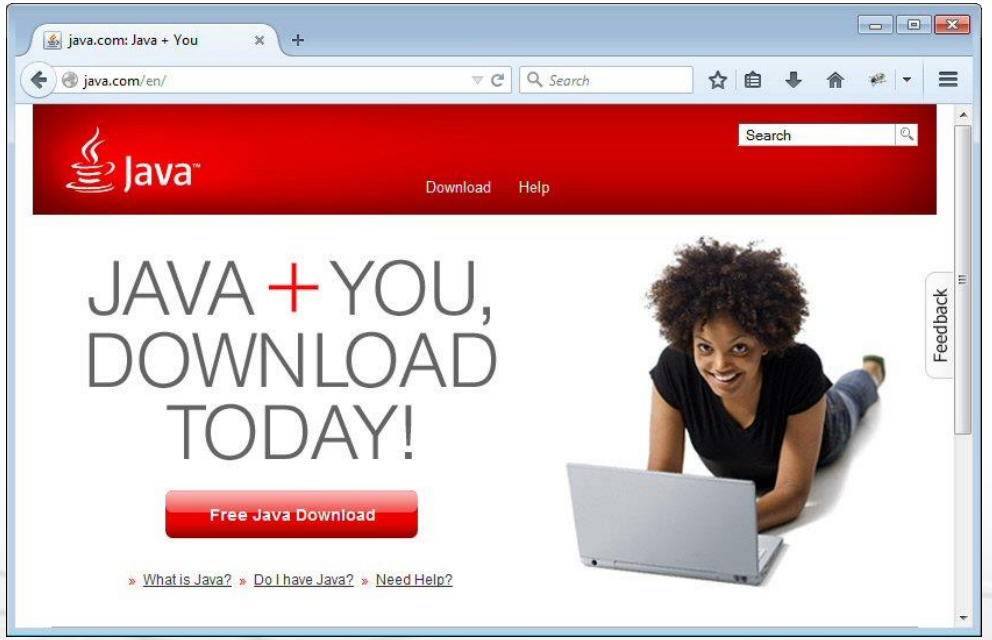
Browser 1 Halaman: Pemanfaatan URL

```
public class UrlRetriever2 {
  public static void main(String[] args) {
    String defaultAddress = "http://docs.oracle.com/javase/8/docs/";
    String address = JOptionPane.showInputDialog("URL:", defaultAddress);

    if (address == null) {
      address = defaultAddress;
    }

    WebUtils.lines(address).forEach(System.out::println);
  }
}
```

UrlRetriever2 Beraksi

```
<!DOCTYPE HTML ...>
<html lang="en-US" xml:lang="en-US">
...
<h1>Java<em>+</em>You, Download Today!</h1>
...
</html>
```





Metode-metode URL

- openConnection
  - Menghasilkan suatu **`URLConnection`** yang membangun koneksi ke host yang ditentukan dengan URL
  - Digunakan untuk me-retrieve baris-baris header dan memasok data ke server HTTP
- openInputStream
  - Mengembalikan stream input koneksi untuk pembacaan (*reading*)
- toExernalForm
  - Memberikan representasi string dari URL
- getRef, getFile, getHost, getProtocol, getPort
  - Mengembalikan komponen-komponen dari URL

Pemanfaatan Metode dari URL: Contoh

```
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>");
  }
}
```

Pemanfaatan Metode dari URL: Hasil

```
> 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
```

Catatan:
Jika port tidak dinyatakan secara eksplisit dalam URL,
maka dianggap port standard dari protokol yang digunakan,
dan getPort mengembalikan -1

Browser Nyata Menggunakan Swing

- Kelas JEditorPane sudah bawaan untuk HTTP & HTML

Browser dengan Swing: *Source 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 dengan Swing (Lanj.)

```
    ...
    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 url = initialURL; // klik “home”, tidak harus masukkan URL

    try {
      htmlPane.setPage(new URL(url));
      urlField.setText(url);
    } catch(IOException ioe) {  warnUser("Can't follow link to " + url + ": " + ioe);
  }
}
```

Browser dengan Swing (Lanj.)

```
...
 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);
     }
   }
 }
```

Pertanyaan?

Rangkuman

- Membuka suatu Socket

```
new Socket("hostname-or-IP-Address", port)
```

- Mendapatkan suatu PrintWriter untuk pengiriman data ke server

```
new PrintWriter(client.getOutputStream(), true)
```

- Mendapatkan suatu BufferedReader untuk pembacaan data dari server

```
new BufferedReader
      (new InputStreamReader(client.getInputStream()))
```

- Catatan
  - Blokir lines dan readLine sampai data diterima atau koneksi ditutup
    - Jika koneksi ditutup, lines hanya melengkapkan Stream dan readLine mengembalikan *null*
  - Server HTTP normalnya menutup koneksi setelah pengiriman data, sehingga aman menggunakan lines atau readLine
  - String.split dan StringTokenizer membantu penguraian string.

Contoh: Pemanfaatan InetAddress

```
import java.net.InetAddress;
import java.net.UnknownHostException;

public class GetIP {
   public static void main(String[] args) {
      InetAddress address = null;
      try {
         address = InetAddress.getByName("www.detik.com");
      } catch (UnknownHostException e) {
         System.exit(2);
      }
      System.out.println(address.getHostName() + "=" + address.getHostAddress());
      System.exit(0);
   }
}
```