Föreläsning 2

DD2390
Internetprogrammering
6 hp

URL

- En URL (***U**niform **R**esource **L**ocator)* är en form av resurspekare som märker ut en resurs och sättet (protokollet) att kommunicera med den. Resursen är ofta en fil, men bör betraktas mer generellt (en datamängd).
- Följande standardiserade format används:
  protokoll://värd[:port]/sökväg t e x
  http://localhost:8080/index.html
  http://www.kth.se/csc
  ftp://ftp.sunet.se
- Om port ej anges är 80 standard för http.

Innehåll

- HTTP-protokollet
- java.net.URL
- java.net.HttpURLConnection

HTTP-request

- Formatet för en http-request är:
  - initial line: <metod> <sökväg> <HTTP-version>
  - header1: <variabel> : <värde>
  - header2: <variabel> : <värde>
  - header3: <variabel> : <värde>
  - <*blankrad*>
  - message: Eventuell data...
- Vanligaste metoder är GET och POST. Med GET blir formulärparametrar en del av URL:en och med POST lagras dem i meddelandekroppen.
- I header:n lagras information om webläsaren, dess miljö och eventuella cookies som klienten skickar med till servern.
- Sökväg anges relativt serverns rot, t e x /index.html?file=meny.html
- Det finns HEAD, PUT, DELETE, TRACE, CONNECT

HTTP

- HTTP är ett protokoll på applikationlagernivå. Det är tillståndslöst då ingen förbindelse upprätthålls mellan två förfrågningar.
- Proceduren är följande:
  - Klienten öppnar en anslutning
  - Klienten skickar en förfrågan (= request)
  - Servern skickar ett svar (= response)
  - Servern stänger anslutningen
- HTTP-protokollet är en specifikation för hur denna kommunikation ska ske kan betraktas som en grammatisk kravspecifikation i två versioner
  - HTTP-request
  - HTTP-response
- Det är helt upp till utvecklaren av webbläsaren / webservern att implementera stödet för denna specifikation. Nuvarande version är 1.1.

Viktiga headerfält

- Accept
- User-Agent
- Host
- Cookie
- (Date)

HTTP-request (text)

GET /~stene/ HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: sv-SE
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: www.csc.kth.se
DNT: 1
Connection: Keep-Alive

Viktiga headerfält

- Content-Type
- Content-Length
- Server
- Set-Cookie
- (Date)

HTTP-response

- Formatet för en http-response är:
  - initial line: <http-version> <kod> <OK/Error>
  - header: <variabel> : <värde>
  - <blankrad>
  - message: Eventuell data...
- I header:n bör alltid *Content-Type* ingå för att webbläsaren ska kunna skilja text från binär information etc. Anges I MIME-format (t e x *text/plain, text/html, image/jpeg*) och specificerar innehållet i *message*.
- OBS!!! Det är endast *message* (*body*) som syns när man väljer “view source” i IE.

HTTP-response (text)

HTTP/1.1 200 OK
Date: Tue, 21 Jan 2014 15:27:34 GMT
Server: Apache
Last-Modified: Mon, 20 Jan 2014 13:57:32 GMT
ETag: "65d20858-15e5-45e86b00"
Accept-Ranges: bytes
Content-Length: 5605
Keep-Alive: timeout=3, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

Statuskoder

- 1xx = Informativ
  - 100 Continue (bekräftelse av header innan body)
- 2xx = Lyckad
  - 200 OK
  - 202 Accepted
  - 204 No Content
- 3xx = Omdirigering
  - 301 Moved Permanently
  - 302 Found
  - 304 Not Modified
- 4xx = Klientfel
  - 400 Bad request
  - 401 Unauthorized
  - 404 Not Found
- 5xx = Serverfel
  - 500 Internal Server Error

HTTP-response (binär)

HTTP/1.1 200 OK
Date: Tue, 21 Jan 2014 15:27:34 GMT
Server: Apache
Accept-Ranges: bytes
Content-Length: 2946
Keep-Alive: timeout=3, max=98
Connection: Keep-Alive
Content-Type: image/gif

GIF89a

Cache (request)

GET /~stene/ HTTP/1.1

Accept: text/html, application/xhtml+xml, */*

Accept-Language: sv-SE

User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko

Accept-Encoding: gzip, deflate

Host: www.csc.kth.se

If-Modified-Since: Mon, 20 Jan 2014 13:57:32 GMT

If-None-Match: "65d20858-15e5-45e86b00"

DNT: 1

Connection: Keep-Alive

HttpClient

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

public class HttpClient{

   public static void main(String[] args) throws Exception{
      String host = args[0];
      int port = Integer.parseInt(args[1]);
      String fil = args[2];
      Socket s =
         new Socket(host,port);

      PrintStream utdata =
         new PrintStream(s.getOutputStream());
      utdata.println("GET /" + fil + " HTTP/1.1");
      s.shutdownOutput();

      BufferedReader indata =
         new BufferedReader(new InputStreamReader(s.getInputStream()));
      String str = "";
      while( (str = indata.readLine()) != null){
         System.out.println(str);
      }
      s.close();
   }
}
```

Cache (response)

HTTP/1.1 304 Not Modified
Date: Tue, 21 Jan 2014 15:33:09 GMT
Server: Apache
Connection: Keep-Alive
Keep-Alive: timeout=3, max=100
ETag: "65d20858-15e5-45e86b00"

HttpServer 1 / 2

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

public class HttpServer{

   public static void main(String[] args) throws IOException{
      ServerSocket ss = new ServerSocket(80);
      while(true){
         Socket s = ss.accept();
         BufferedReader request =
        new BufferedReader(new InputStreamReader(s.getInputStream()));
         String str = request.readLine();
         System.out.println(str);
         StringTokenizer tokens =
        new StringTokenizer(str," ?");
         tokens.nextToken();
         String requestedDocument = tokens.nextToken();
         while( (str = request.readLine()) != null && str.length() > 0){
        System.out.println(str);
         }
         s.shutdownInput();
```

Cookies

- Då man i en webserverapplikation vanligtvis vill kunna skilja de olika klienterna från varandra och detta inte stöds av HTTP låter man servern spara en s k *cookie* hos klienten.
- En cookie har ett namn och ett värde och sparas i webbläsarens minne. Servern definierar cookie:n i response header:n
  - Set-Cookie: < namn > = < värde >;
  - expires= < datum >;
  - domain= < domän >;
  - path= < bibliotek >
    - *"Set-Cookie: JSESSIONID=751668BA70BF145F52370C61F226904B; Path=/"*
- Om klienten vid ett senare tillfälle försöker komma åt samma domän/sökväg skickas cookie:n med i request-header:n.
  - *"Cookie: JSESSIONID=751668BA70BF145F52370C61F226904B;"*

HttpServer 2 / 2

```
         PrintStream response =
        new PrintStream(s.getOutputStream());
         response.println("HTTP/1.0 200 OK");
         response.println("Server : Slask 0.1 Beta");
         if(requestedDocument.indexOf(".html") != -1)
        response.println("Content-Type: text/html");
         if(requestedDocument.indexOf(".gif") != -1)
        response.println("Content-Type: image/gif");

         response.println("Set-Cookie: clientId=1; expires=Wednesday,31-Dec-07 21:00:00 GMT");

         response.println();
         File f = new File("."+requestedDocument);
         FileInputStream infil = new FileInputStream(f);
         byte[] b = new byte[1024];
         while( infil.available() > 0){
        response.write(b,0,infil.read(b));
         }
         s.shutdownOutput();
         s.close();
      }
   }
}
```

java.net.URL

- En klass för att beskriva en URL

```
try{
   URL url = new URL("http","www.nada.kth.se","index.html");
}
catch(MalformedURLException e){
   System.out.println(e.getMessage());
}
```

- Det finns ett flertal konstruktorer
- I detta fall blir porten den som är default för protokollet
- Viktigaste metod att känna till är
  - *public URLConnection **openConnection**() throws IOException*

Exempel 2/2

```
try{
   con.connect();
}
catch(IOException e){
   System.out.println(e.getMessage());
}
BufferedReader infil = null;
try{
   infil = new BufferedReader(new InputStreamReader(con.getInputStream()));
}
catch(IOException e){
   System.out.println(e.getMessage());
}
String rad = null;
try{
   while( (rad=infil.readLine()) != null){
   System.out.println(rad);
   }
}
catch(IOException e){
   System.out.println(e.getMessage());
}
System.out.println(con.getHeaderField("Content-Type"));
}
}
```

java.net.HttpURLConnection

- Med hjälp av denna klass kan simulera en HTTP-klient
- Är en subklass till URLConnection
- En instans av denna klass kan sägas ha två tillstånd
  - Ett innan man skickar iväg en http-request, för att manipulera headerfälten.
  - Ett efter man skickat iväg en http-request och innehåller det som servern skickar = http-response
  - Metoden som delar tillstånden heter *connect()*

HTTP request/response

```
GET /~stene/ HTTP/1.1
User-Agent: Mozilla/5.0        ← skrivs med setRequestProperty()
Host: www.csc.kth.se
Accept: text/html, application/xhtml+xml, */*
Connection: keep-alive


HTTP/1.1 200 OK
Date: Tue, 21 Jan 2014 15:27:34 GMT
Server: Apache
Last-Modified: Mon, 20 Jan 2014 13:57:32 GMT
ETag: "65d20858-15e5-45e86b00"
Accept-Ranges: bytes
Content-Length: 5605
Keep-Alive: timeout=3, max=100
Connection: Keep-Alive
Content-Type: text/html        ← läses med getHeaderField()

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">


<html lang="en">
```

Exempel 1/2

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

public class Slask{
   public static void main(String[] args){
      URL url = null;
      try{
         url = new URL("http","www.nada.kth.se","/index.html");
      }
      catch(MalformedURLException e){
         System.out.println(e.getMessage());
      }
      HttpURLConnection con = null;
      try{
         con = (HttpURLConnection)url.openConnection();
      }
      catch(IOException e){
         System.out.println(e.getMessage());
      }
      con.setRequestProperty("User-Agent","Mozilla");
```