# Föreläsning 2

Internetprogrammering
DD1389



## Innehåll

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



## Java trådar

- Vad är ett tråd?
- Varför trådar?
  - Icke blockering
    - Läsa stor fil i bakgrunden, klient-server anslutningar
  - Bakgrundsaktivitet
    - Hämta stor fil, spela upp musik, parallella beräkningar
  - Aktiva objekt
    - Klocka och timer
  - Oberoende uppgifter
    - En tjänst som kan göras för flera klienter simultant



## Använda trådar i java

Två sätt att skapa trådar:

- Implementera interfacet `Runnable`
- Ärva från (utöka) klassen `Thread`

Skillnad:

Egentligen inget men från ett objektorienterad synvinkel: ”man ska ärva en klass bara när man utökar klassen”



## Klassen Thread

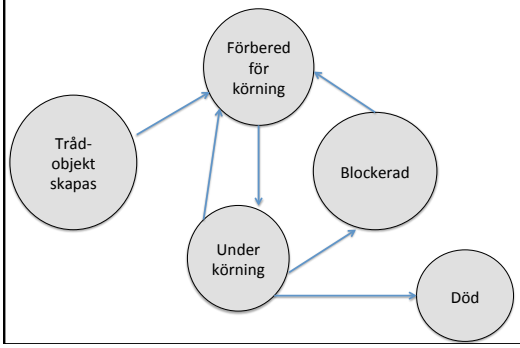
| Viktiga Metoder | Uppgift |
|---|---|
| Thread | Konstruktor: skapar objekt |
| run | Koden som ska köras parallell, ska stå i denna metod |
| start | Skapar en tråd och anropar trådens run metod |
| stop | Deprecated, Undvik! |
| sleep | Pausar trådet en viss tid |
| isAlive | true om trådet är under körning |



## Klassen Thread

| Viktiga metoder | Uppgift |
|---|---|
| join | Blockerar exekveringen av nästkommande kod tills trådet gjort sitt klar |
| currentThread | Returnerar en referens till aktuellt trådet |
| yield | Trådet ska få tillståndet färdigt för att köra |
| interrupt | Trådet avslutas |

## Ett tråds tillstånd




7

## Synkronisering

- Atomic: `a=10` och a = a*10+a*23
- Race condition
- Synchroniserad method
- Synchroniserad block
- wait/notify

8

## HTTP protokollet

- Applikationslagret
- Tillståndslös
- HTTP är specificerad av RFC, utvecklaren av webbläsaren och webbservern implementerar stöder för specifikationen och därmed finns smärre skillnader mellan olika webbläsare och mellan olika webbservrar.
- Webbläsaren skickar alltid förfrågan
- Webbservern skickar alltid svaret

9

## URL

- En URL (***U**niform **R**esource **L**ocator)* är en form av resurspekare.
- En resurs kan vara en statisk fil, det är konfigurerad för servern om filen ska skickas som den är eller om den ska den köras eller likanande
- Följande standardiserade format används:
  *protokoll://värd[:port]/resurs*

  ex:
  http://130.237.227.106/
  http://www.csc.kth.se/
  ftp://ftp.sunet.se
- default port för webbservrar är port 80 som alltså används om port uteblir i URL för HTTP t.ex http://www.csc.kth.se:80/

10

## HTTP-request

Formatet för en http-request är:

- initial line: `<metod> <sökväg> <version>`
- header1: `<variabel> : <värde>`
- header2: `<variabel> : <värde>`
- header3: `<variabel> : <värde>`
- *<blankrad>*
- message: Eventuell data...

11

| Metod | Uppgift |
|---|---|
| GET | frågar efter en resurs |
| HEAD | frågar efter information om en resurs, inte själva resursen |
| POST | skicka data till servern |
| PUT | skickar document till servern |
| DELETE | Ta bort en resurse |
| TRACE | när förfrågan måste passera genom en proxy, gateway etc |
| OPTIONS | listar tillåtna metoder |

12

## Viktiga headerfält

- Accept: vilka mime-type av resurs kan hanteras
- User-Agent: t.ex Mozilla/4.75 [en] (Win98)
- Host: serverns-url-adress
- Cookie: identifiering

13

## HTTP-request exempel

GET /~vahid/jag.jpg 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

14

## 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" webbläsaren.

15

## 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

16

## Viktiga headerfält

- Content-Type: typ av data (text, bild, ljudfil)
- Content-Length: storlek på data
- Server: namn på servern t.ex Apach
- Set-Cookie: skickar tokens till klienten
- Last-Modified: filen sista modifiering (caching)
- Date

17

## 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>
<html lang="en">.....

18

## 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



## 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"



## 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;"*



## 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/1.0\n");
    s.shutdownOutput();
    BufferedReader indata =new BufferedReader(
                        new InputStreamReader(s.getInputStream()));
    String str ;
    while( (str = indata.readLine()) != null){
       System.out.println(str);
    }
    s.close();
   }
}
```



## 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();
```

# 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.csc.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*



# 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()*



# Exempel 1/2

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

public class TinnyClient{
    public static void main(String[] args){
   URL url = null;
   try{
       url = new URL("http","www.csc.kth.se","/index.html");
   }
   catch(MalformedURLException e){
       System.out.println(e.getMessage());
   }
   HttpURLConnection con = null;
```



# Exempel 2/2

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



# 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">
```