Internetprogrammering
DD1386

Föreläsning 2

Innehåll

- HTTP-protokollet
- HttpClient
- HttpServer
- Hantering tillstånd i HTTP-kommunikation
  - Headerfältet Set-cookie
  - Headerfältet Cookie

HTTP protokollet

- Applikationslagret
- Tillståndslös
- Proceduren är följande:
  - Klienten öppnar en anslutning
  - Klienten skickar en förfråga (request)
  - Servern skickar ett svar (response)
  - Servern stänger anslutningen
- HTTP-protokollet är en specifikation för hur denna kommunikation ska ske. Det kan betraktas som en grammatisk kravspecifikation för två rollerna klient och server
  - HTTP-request (skickas alltid från en klient till en server som en begäran)
  - HTTP-response (skickas alltid från en server till en klient som svar till requesten)
- Det är helt upp till utvecklaren av webbläsaren / webbservern att implementera stödet för denna specifikation.

URL

- En URL (***U**niform **R**esource **L**ocator)* är en form av resurspekare (adress till en resurs över internet)

- Vad är en resurs?
  - En statisk fil som ska skickas bit för bit till klienten.
  - Eller en exekverbar fil som ska exekveras hos servern och utdata från exekveringen ska skickas till klienten. De filer som ska exekveras förs har konfigurerats hos servern så att servern inte skickar den exekverbara filen till klienten.

URL

- Exempel på url:
  - http://localhost:8080/index.html
  - http://www.eecs.kth.se/
  - ftp://ftp.sunet.se

- Standardiserat format:
  protokoll://värd[:port]/sökvägen till resursen
  t.ex:
- default port för webbservrar är 80 vilket betyder att det behöver inte anges i urlen.

HTTP-request

```
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…
Accept-Encoding: gzip, deflate
Host: www.csc.kth.se
Connection: close
```

HTTP-request

- Formatet för en http-request är:
  - Start line: <metod> <sökväg> <HTTP-version> ↲
  - Header 1: <variabel> : <värde> ↲
  - Header 2: <variabel> : <värde> ↲
  - Header 3: <variabel> : <värde> ↲
  - ...
  - Header N: <variabel> : <värde> ↲
  - ↲
  - Payload: Eventuell data...

OBS väldigt viktigt!! Två efterföljande radbrytningar innan payload börjar

- 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 webbläsaren, dess miljö och eventuella cookies som klienten skickar med till servern.
- Sökväg anges relativt serverns rot, t.ex `/index.html?file=meny.html`

HTTP-metoder

Viktiga Metoder	**Beskrivning**
GET	Frågar efter en resurs
POST	Skicka data till servern

Mindre viktiga Metoder	**Beskrivning**
HEAD	Frågar efter information om en resurs
PUT	Skicka dokument till servern
DELETE	Ta bort en resurs
TRACE	När förfrågan måste passera genom en proxy, gateway etc
OPTIONS	Frågar efter tillåtna (tillgängliga) metoder

Viktiga headerfält i en request

- Accept: vilka mime-type av resurs kan hanteras
- User-Agent: t.ex Mozilla/4.75[en] (Win98)
- Host: serverns domän namn och portnummer
- Cookie: används för identifiering
- Date: när paketet skapades

HTTP-response (binär exempel)

HTTP/1.1 200 OK↵

Date: Sun, 21 Jan 2018 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

HTTP-response

- Formatet för en http-response är:
  - Start line: <http-version> <statuskod> <OK/Error>↲
  - header: <variabel> : <värde> ↲
  - …
  - header: <variabel> : <värde> ↲
  - ↲
  - payload: data...

OBS väldigt viktigt!! Två efterföljande radbrytningar innan payload börjar

- I header:n bör alltid *Content-Type* ingå för att webbläsaren ska kunna skilja text från binär information etc. Anges I MIME-format (t.ex *text/plain, text/html, image/jpeg*) och specificerar innehållet i *message*.
- OBS!!! Det är endast *payload* (*body*) som syns när man väljer “view source” i webbläsaren.

Viktiga headerfält i en response

- Content-Type: mimetype
- Content-Length: storlek av data
- Server: info om server, t.ex: Apache
- Set-Cookie: skickar kaka
- Date: datum då servern skapade responsen

Hanteraing av tillstånd

- Applikationslagret
- HTTP är tillståndslös, d.v.s. det finns inte stöd för att servern ska komma ihåg något om tidigare request
- Hantering av tillstånd:
  - Headern Set-Cookie i responsen
  - Headern Cookie i request för att hantera tillstånd

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=751668B0C61F226904B; Path=/*↵
- Om klienten vid ett senare tillfälle försöker komma åt samma domän/sökväg skickas cookie:n med i request-header:n. – *"Cookie: JSESSIONID=751668B0C61F226904B;"*

Response med set-cookie

```
HTTP/1.1 200 OK↵
Content-type: text/html↵
Set-Cookie: status = morgonpigg↵
Set-Cookie:  likes = 1038↵
↵
[Payload]
```

Request med Cookie

```
GET /step1 HTTP/1.1
Host: u-shell.csc.kth.se
Cookie: vardagar = morgonpigg;
  likes = 1038
```

Format

- Set-Cookie: <name>=<value> ⟵ raderas när man stänger av webbläsaren
- Set-Cookie: <name>=<value>; Expires=<date>
- Set-Cookie: <name>=<value>; Max-Age=<non-zero-digit>
- Set-Cookie: <name>=<value>; Domain=<domain-value>
- Set-Cookie: <name>=<value>; Path=<path-value>

  …

Exempel:

- Set-Cookie: id= 47w123
- Set-Cookie: id= 47w123; Max-Age=3600
- Set-Cookie: id= 47w123; Expires=Fri, 21 Jan 2022 09:00:00 GMT;
- Set-Cookie: id= 47w123; Domain=kth.se

Exempel

Vi går igenom Form.java och lägger till Set-Cookie för den.