



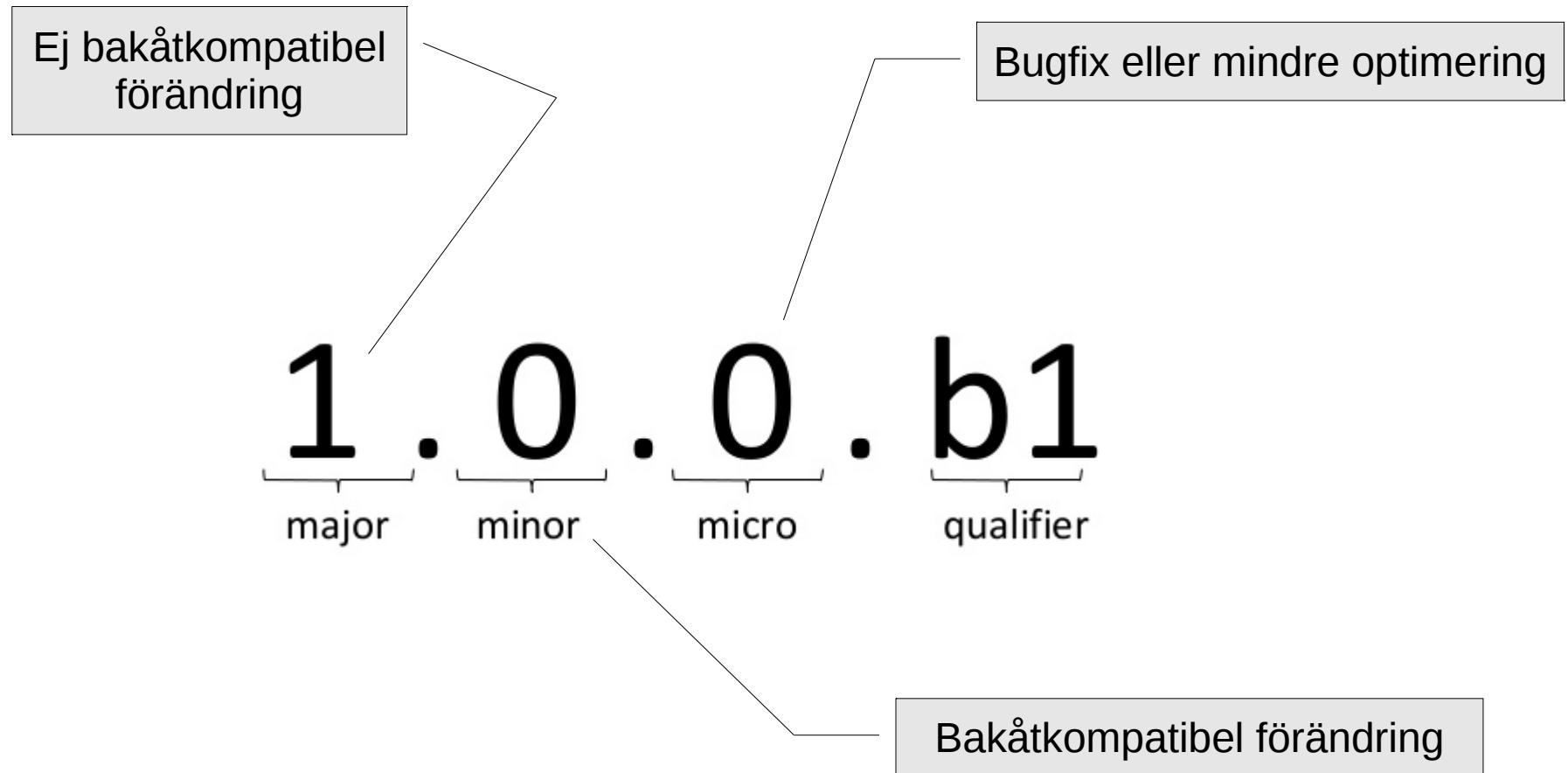
MALMÖ HÖGSKOLA

**Versionshantering med Git och GitFlow,
samt webbtjänster med HTTP och REST**

Dagens agenda

- Repetition från förra veckan
 - Semantisk versionering
- Versionshantering med Git
 - Verktyget Git
 - GitHub
 - GitFlow-modellen
- Webbtjänster med HTTP och REST
 - Introduktion till HTTP
 - Introduktion till REST

Semantisk versionering



Semantisk versionering: <http://semver.org/>
Lockart, *Modern PHP*, p. 61

Versionshantering med Git



Versionshantering – varför?

- Samarbete i teamet
 - Förenklar delning av kod och samtidigt arbete i projektet – även i samma fil (jämför med Google Docs)
 - Olika teammedlemmar kan utveckla flera olika features samtidigt!
- Tydlig historik och spårbarhet
 - Återgå till tidigare versioner av projektet
 - Gick något sönder när din kollega lade till kod? Ingen fara, den gamla koden finns kvar!
 - Jämför olika versioner av ett projekt, modul eller fil
 - Vem gjorde vad? Nu vet vi
- Minska risker
 - Låt inte all källkod leva på en enskild dator
 - Låt inte vem som helst förändra koden

Lockart, *Modern PHP*. p. 157



- Det för tillfället flitigast användna verktyget för versionshantering
- Välprövat
 - Hanterar ohemult stora projekt (exempelvis Linux)
 - Snabbt!
- Hanterar arbetsflöden på ett bra sätt
 - Byggt för att vara bra på att dela upp projekt och slå samman dem igen
 - Fork/pull request och branch/merge
- Distribuerat – ingen central server krävs
 - Pull/Push
- Open source!

Git: <https://git-scm.com/>
Lockart, *Modern PHP*. p. 157



Terminologi

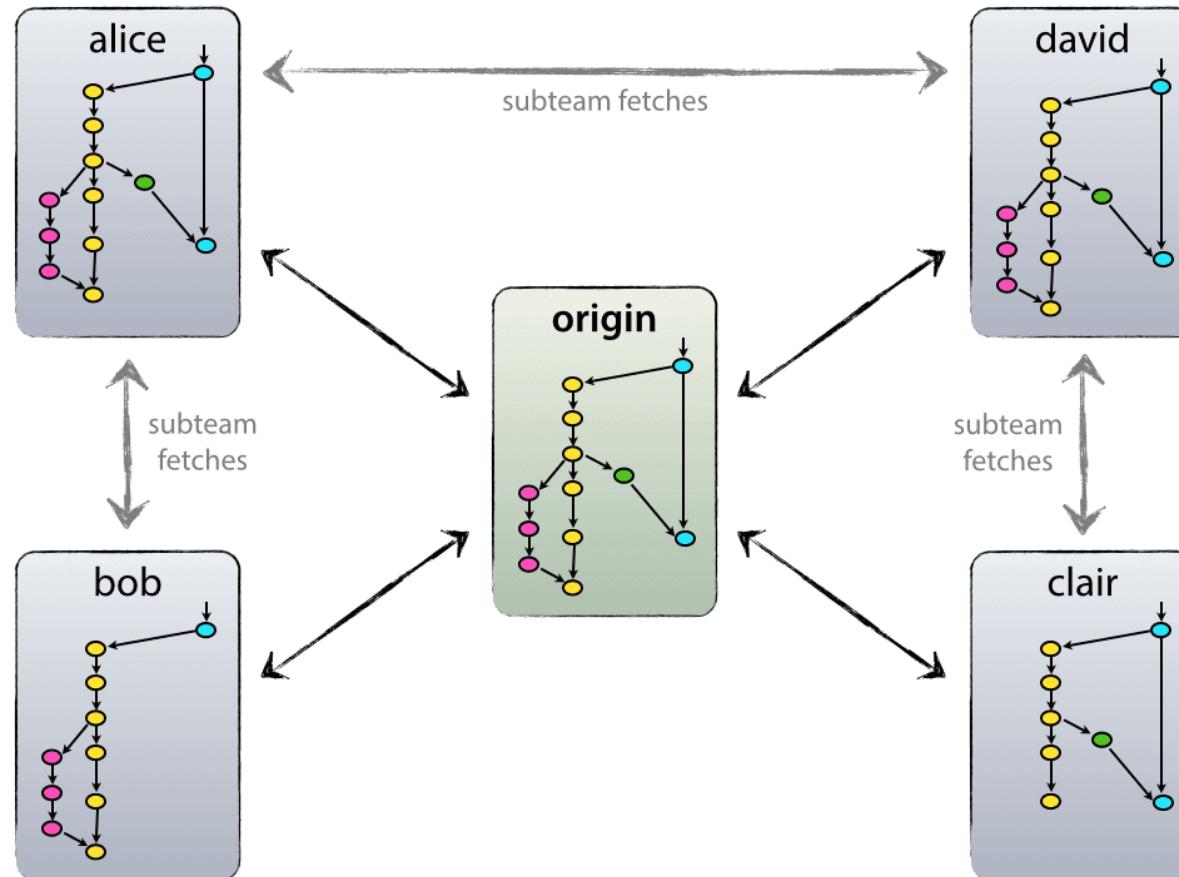
Repository: En plats för ett projekt, där all källkod finns samlad.

Branch: En avgrening av källkoden

Patch: En föreslagen kodförändring

Merge: Sammanslagning av två branches

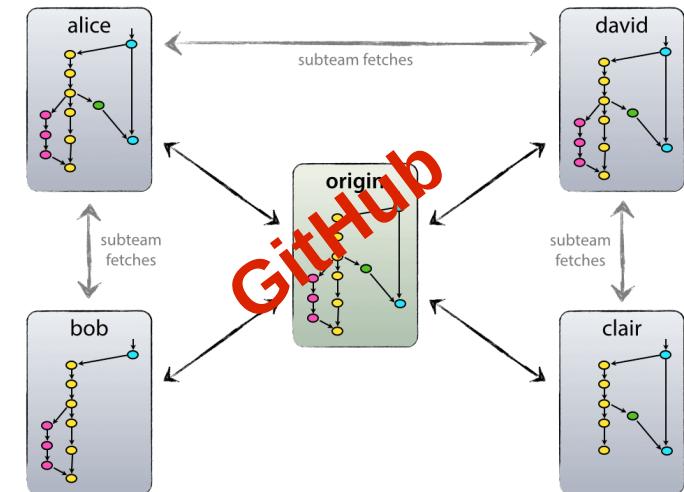
Git: <https://git-scm.com/>
Lockart, *Modern PHP*. p. 157



Git: <https://git-scm.com/>
Lockart, *Modern PHP*. p. 157

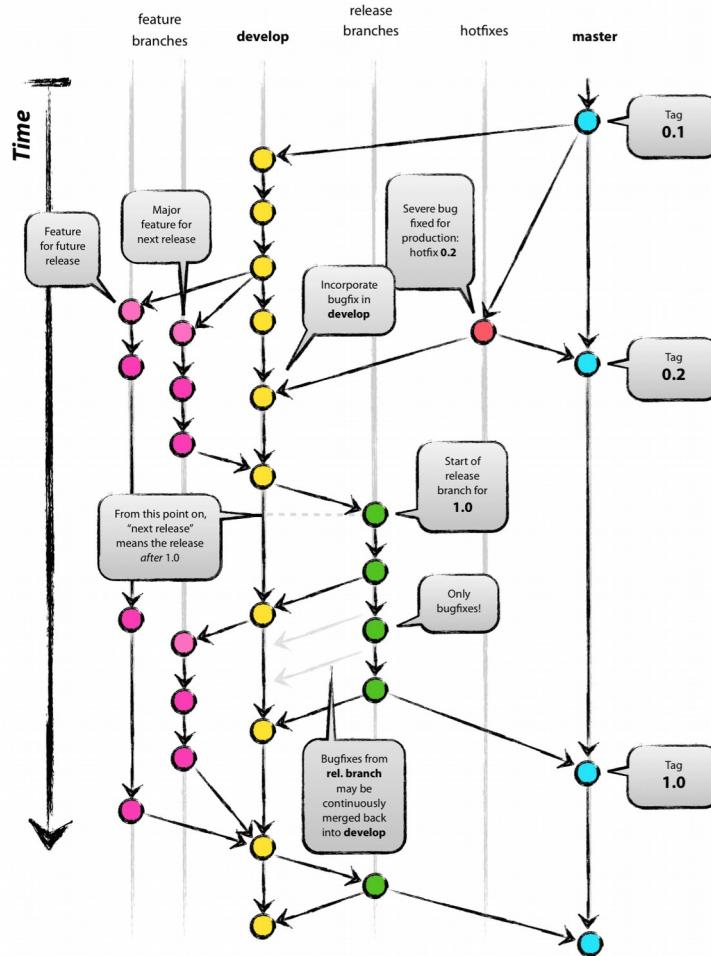


- En tjänst för lagring och delning av Git-repositories
- Stor i open source-världen
- Erbjuder även kringtjänster såsom wikis och viss ärendehantering



GitHub: <https://github.com>

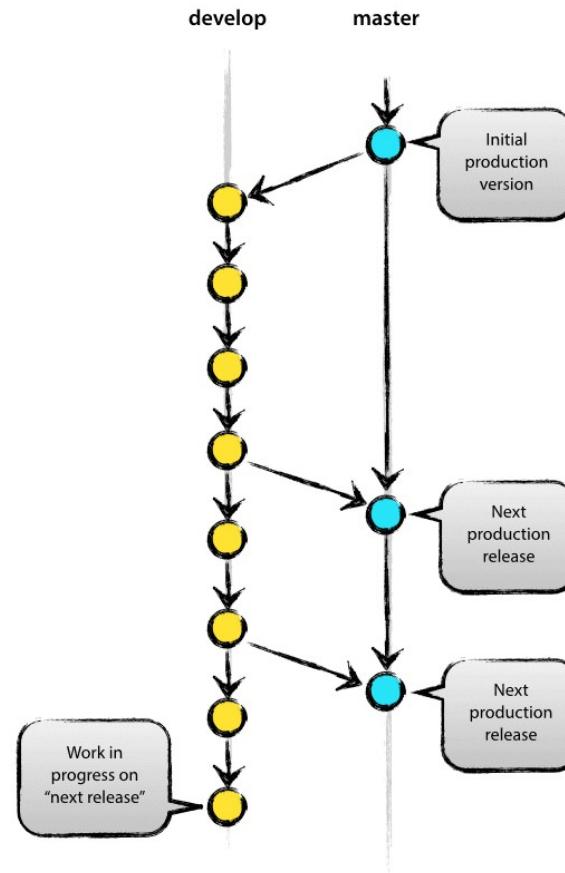
GitFlow-modellen



Driessen – A successful Git branching model: <http://nvie.com/posts/a-successful-git-branching-model/>
GitFlow: <https://github.com/nvie/gitflow>

GitFlow-modellen

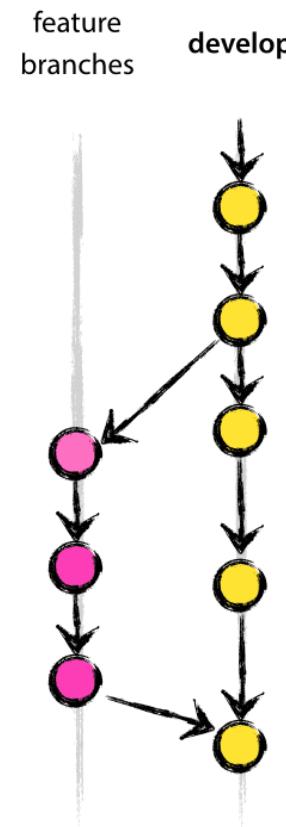
- Utgår från två branches:
 - **master** – kod i produktion
 - **develop** – kod under utveckling



Driessen – A successful Git branching model: <http://nvie.com/posts/a-successful-git-branching-model/>
GitFlow: <https://github.com/nvie/gitflow>

GitFlow-modellen

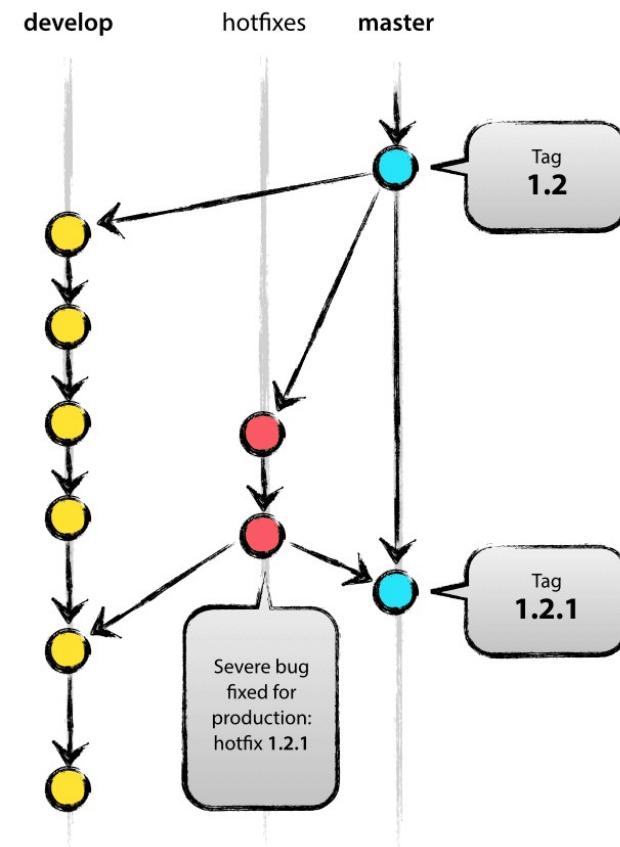
- Feature branching:
 - Avgrenas från *develop* eller annan feature-branch
 - Mergeas alltid med *develop*
 - Namnges i regel som *feature/name*



Driessen – A successful Git branching model: <http://nvie.com/posts/a-successful-git-branching-model/>
GitFlow: <https://github.com/nvie/gitflow>

GitFlow-modellen

- Feature branching:
 - Avgrenas från *master*
 - Mergeas med *develop* och *master*
 - Namnges i regel som *hotfix-name*



Driessen – A successful Git branching model: <http://nvie.com/posts/a-successful-git-branching-model/>
GitFlow: <https://github.com/nvie/gitflow>

Webbtjänster med HTTP och REST



Vad är HTTP?

- Det nätverksprotokoll som ligger till grund för kommunikation över World Wide Web.
 - Använder TCP för överföring
- Kan överföra data oavsett innehållstyp

HTTP-specifikationen: <https://tools.ietf.org/html/rfc2616>

Hypertext Transfer Protocol enligt Wikipedia:https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol



MALMÖ HÖGSKOLA

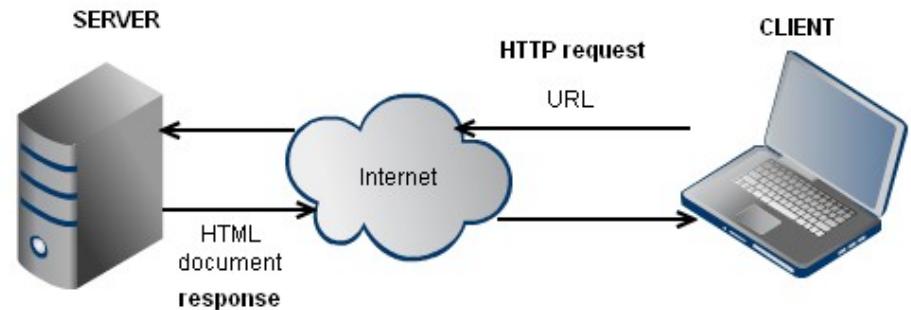
Terminologi

- **IP-adress** – en adress som identifierar en specifik maskin (fysisk eller virtuell) på ett nätverk.
- **TCP (Transmission Control Protocol)** – ett protokoll för överföring av data över ett nätverk. Protokollet garanterar att datan som sänds är korrekt överförd.
- **Portnummer** – ett nummer som läggs till en IP-adress för att adressera en specifik applikation på en maskin.
- **URL (Uniform Resource Locator)** – ett sätt att referera till en resurs på www.

Internet protocol suite enligt Wikipedia: https://en.wikipedia.org/wiki/Internet_protocol_suite

Hur HTTP-kommunikation fungerar

- Två program – en klient (exempelvis en webbläsare) och en server (exempelvis Apache).
- Klienten anropar servern över TCP-koppling med hjälp av HTTP-anrop.



En HTTP-session



IP-adress: 192.168.1.42
Port: 50222

http://www.mah.se



www.mah.se
IP-adress: okänd
Port: 80

?

En HTTP-session



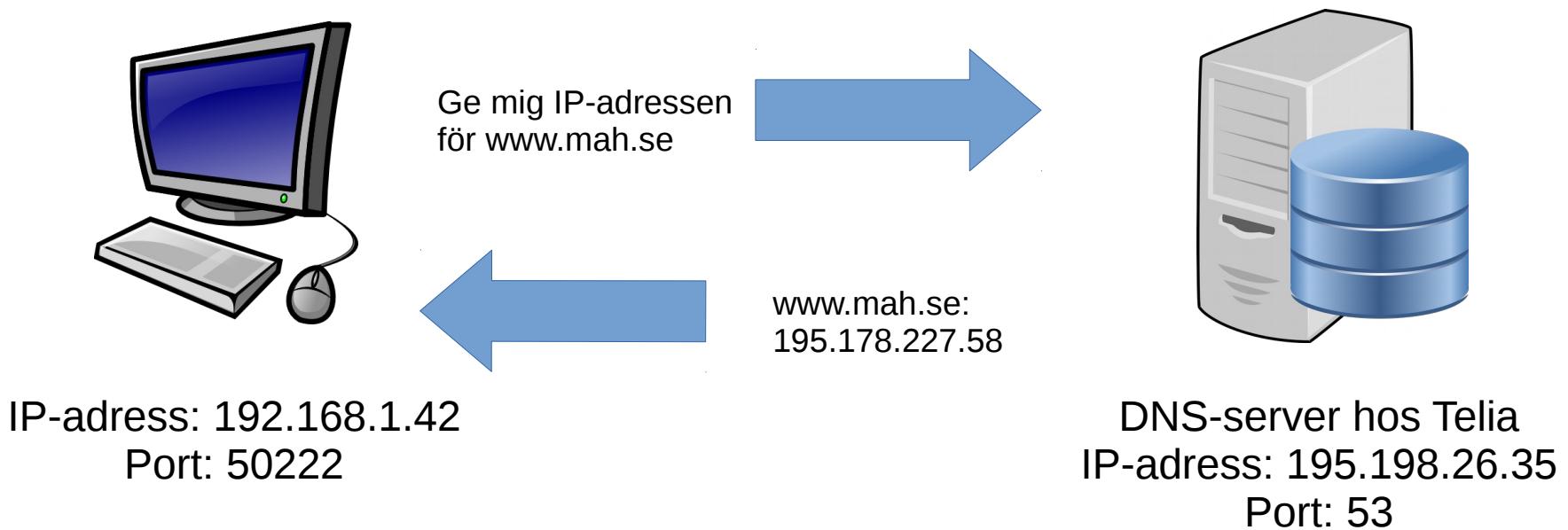
Ge mig IP-adressen
för www.mah.se



IP-adress: 192.168.1.42
Port: 50222

DNS-server hos Telia
IP-adress: 195.198.26.35
Port: 53

En HTTP-session



En HTTP-session



IP-adress: 192.168.1.42
Port: 50222



www.mah.se
IP-adress: 195.198.26.35
Port: 80

http://www.mah.se

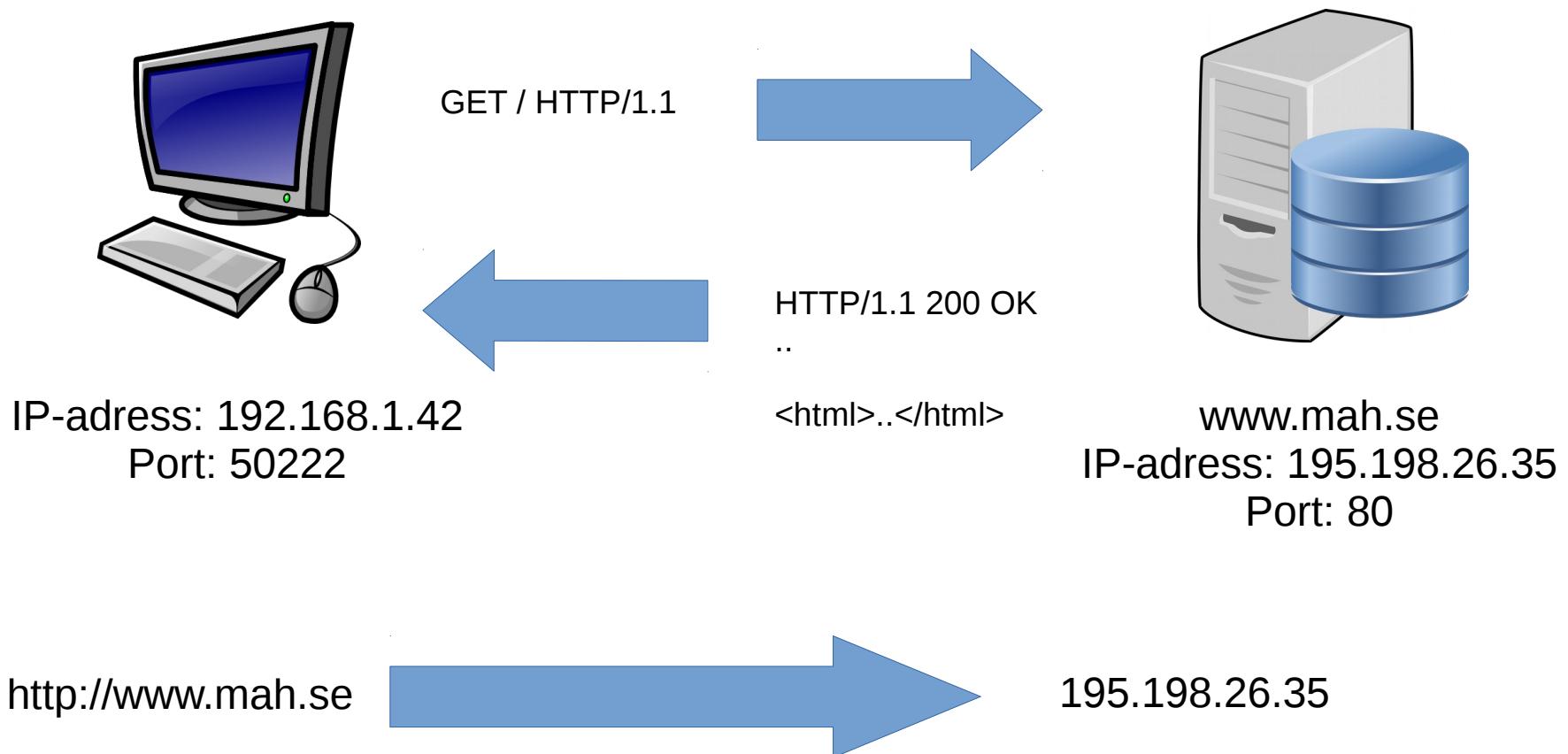


195.198.26.35



MALMÖ HÖGSKOLA

En HTTP-session



HTTP-verb/metoder

Verb/metod	Vad händer?
GET	Läser en resurs
POST	Skapar en ny resurs
PUT	Skapar en ny resurs eller uppdaterar en befintlig resurs
DELETE	Raderar en resurs
OPTIONS	Listar godkända operationer för en resurs
HEAD	Returnerar endast headers vid anrop

Metadata – Request Headers

```
1 GET http://www.w3.org/Protocols/rfc2616/rfc2616.html HTTP/1.1
2 Host: www.w3.org
3 Accept: text/html,application/xhtml+xml,application/xml; ...
4 User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 ...
5 Accept-Encoding: gzip,deflate,sdch
6 Accept-Language: en-US,en;q=0.8,hi;q=0.6
```

```
1 HTTP/1.1 200 OK
2 Date: Sat, 23 Aug 2014 18:31:04 GMT
3 Server: Apache/2
4 Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT
5 Accept-Ranges: bytes
6 Content-Length: 32859
7 Cache-Control: max-age=21600, must-revalidate
8 Expires: Sun, 24 Aug 2014 00:31:04 GMT
9 Content-Type: text/html; charset=iso-8859-1
10 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/1999/xhtml"
11 <html xmlns='http://www.w3.org/1999/xhtml'>
12 <head><title>Hypertext Transfer Protocol -- HTTP/1.1</title></head>
13 <body>
14 ...
```

Svarskoder

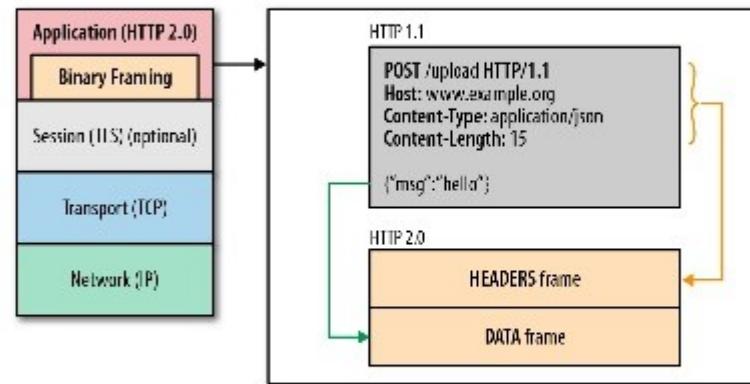
Exempel:

- 200 – OK
- 201 – Created
- 404 – Not Found
- 405 – Method Not Allowed
- 500 – Internal Server Error
- https://sv.wikipedia.org/wiki/Lista_%C3%B6ver_HTTP-statuskoder

HTTP/2 – den moderna versionen

HTTP/2 in one slide...

- One TCP connection
- Request → Stream
 - Streams are multiplexed
 - Streams are prioritized
- Binary framing layer
 - Prioritization
 - Flow control
 - Server push
- Header compression

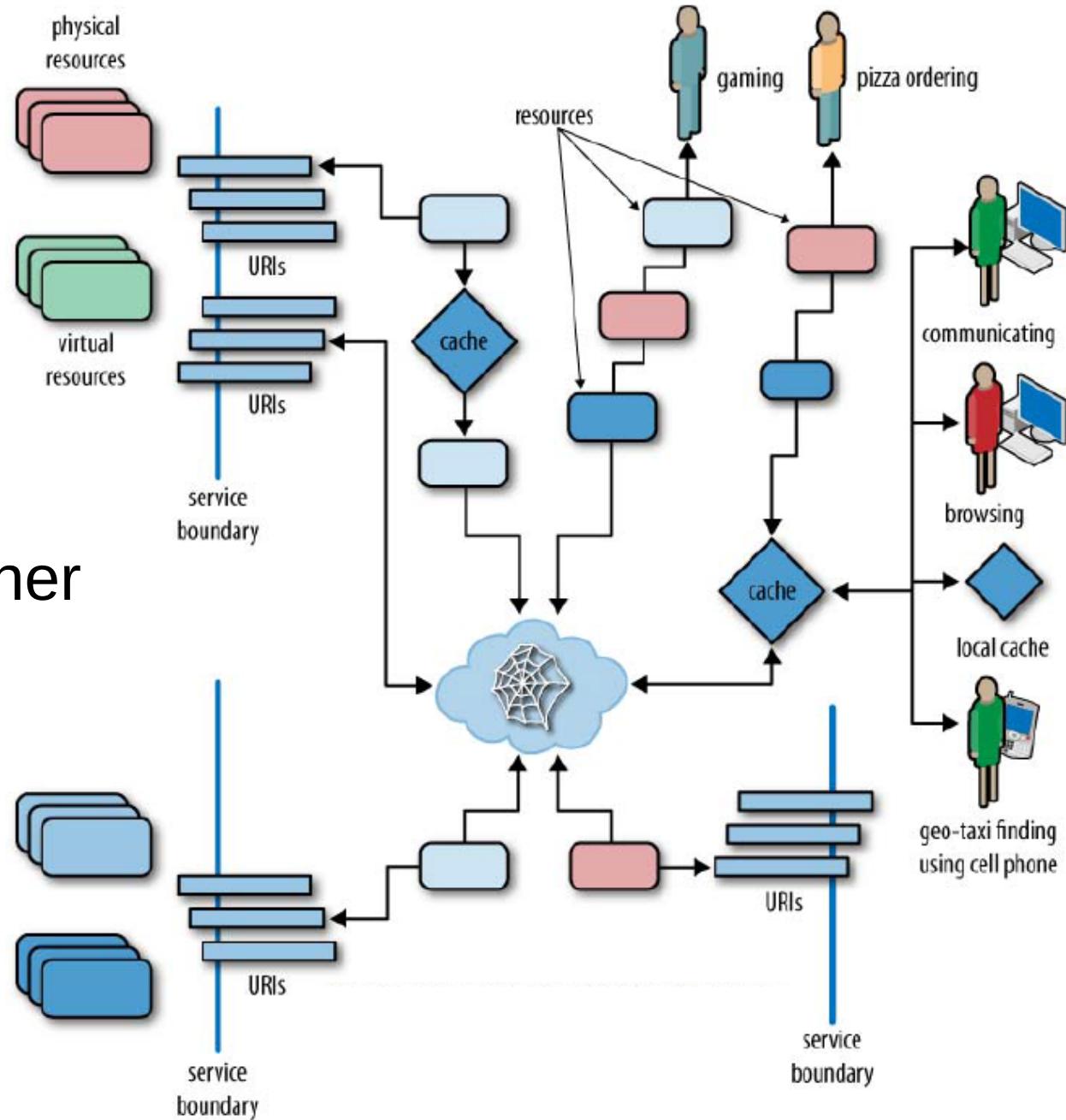


HTTP/2-specifikationen: <https://tools.ietf.org/html/rfc7540>
HTTP/2 enligt Wikipedia: <https://en.wikipedia.org/wiki/HTTP/2>

REST på fem fingrar

- **Resurser**, snarare än tjänster, som identifiereras med URLer
- **En webb av resurser**, där en resurs inte behöver ge svar på allt utan glatt kan länka vidare till andra resurser
- **Klient – Server**-modell
- **Tillståndslöst (stateless)**, varje anrop är idempotent
- **Cachningsbara resurser**

Resurser Identifierare



HTTP

Resurser

Vad är en resurs?



**“A resource is anything we expose to the Web,
from a document or video clip
to a business process or device.
From a consumer’s point of view,
a resource is anything with
which that consumer interacts
while progressing toward some goal.”**

Webber, J., Parastratidis, S., Robinson, I. REST in Practice

Webber et al. Rest in Practice

Resurser på webben

URI

`http://weather.example.com/oaxaca`

Resource

Oaxaca Weather Report

Representation

Metadata:
Content-type:
application/xhtml+xml

Data:
<!DOCTYPE html PUBLIC "....
"http://www.w3.org/....
<html xmlns="http://www....
<head>
<title>5 Day Forecast for
Oaxaca</title>
...
</html>

Identifies

Represents



MALMÖ HÖGSKOLA

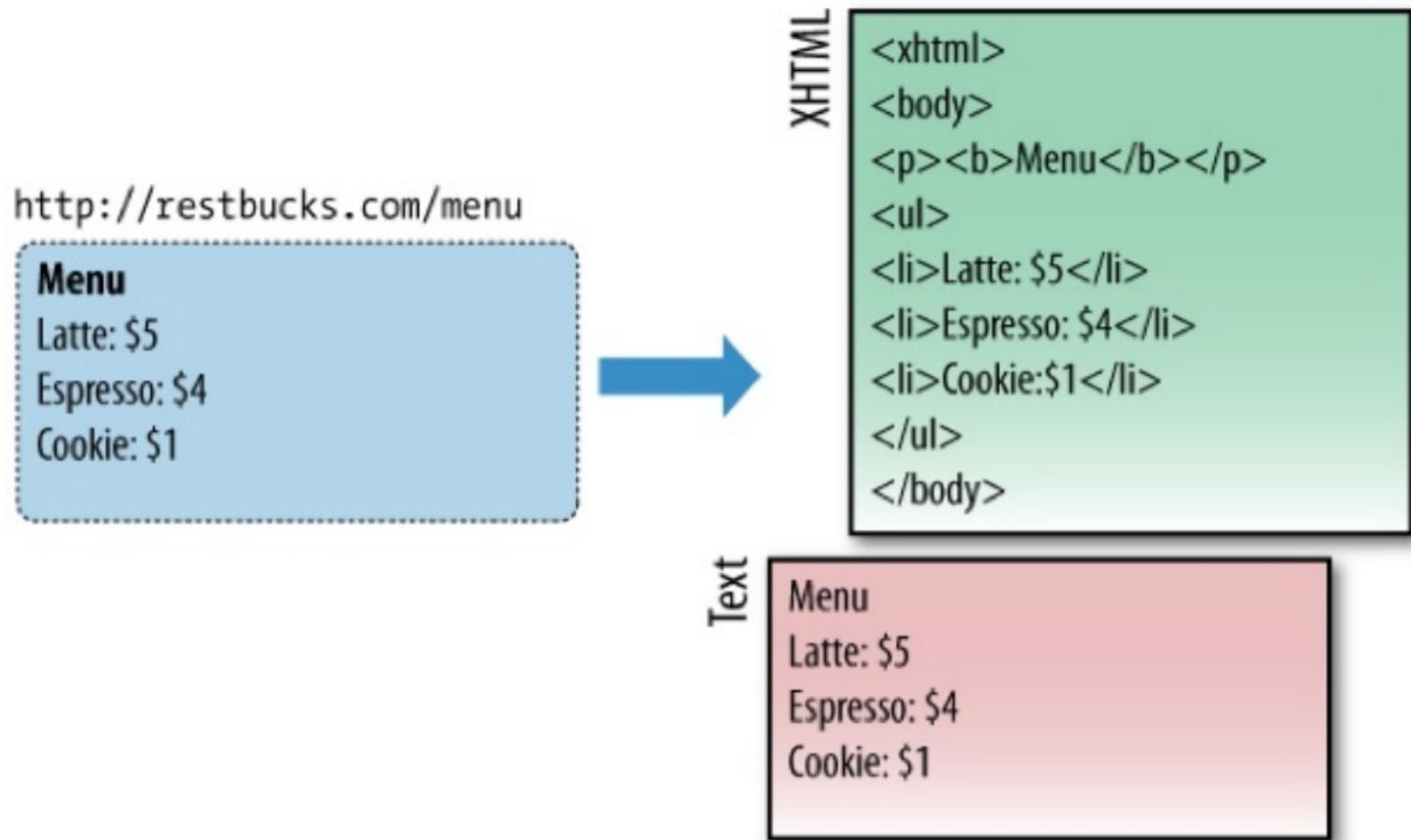


Figure 1-3. Example of a resource and its representations

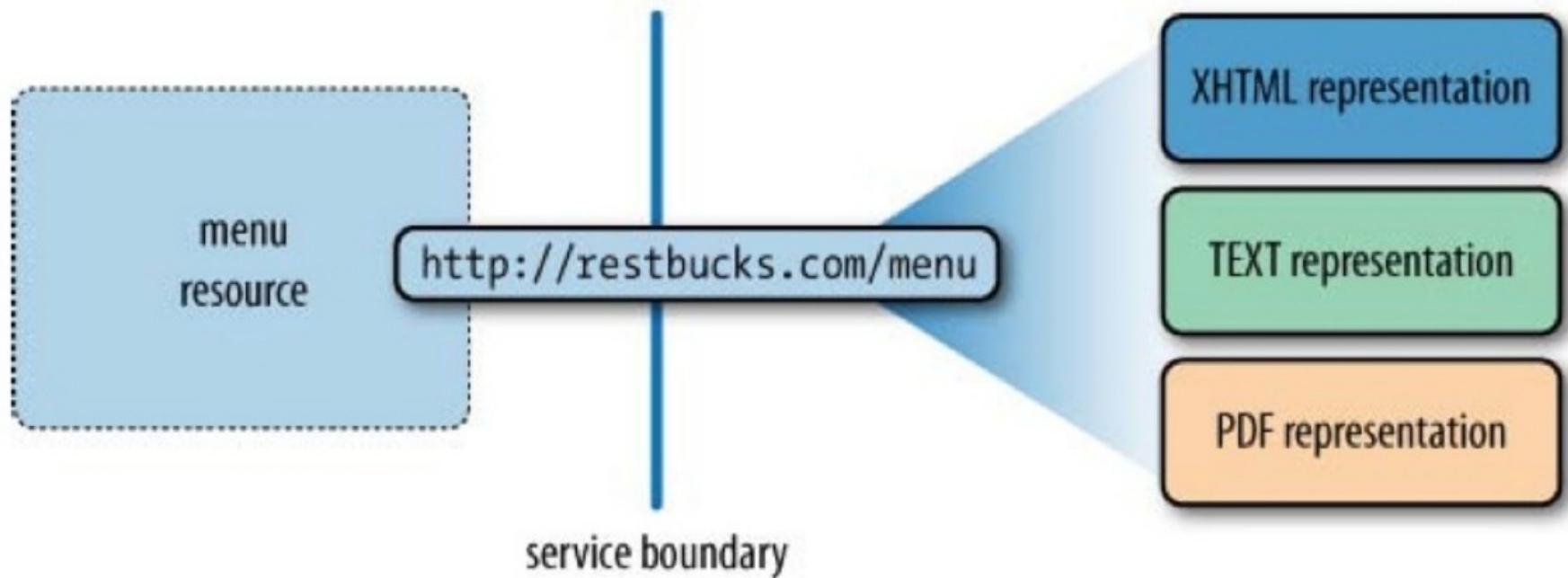


Figure 1-5. *Multiple resource representations addressed by a single URI*

Identifierare

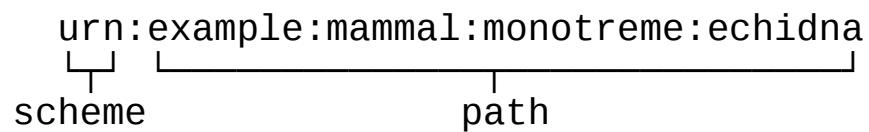
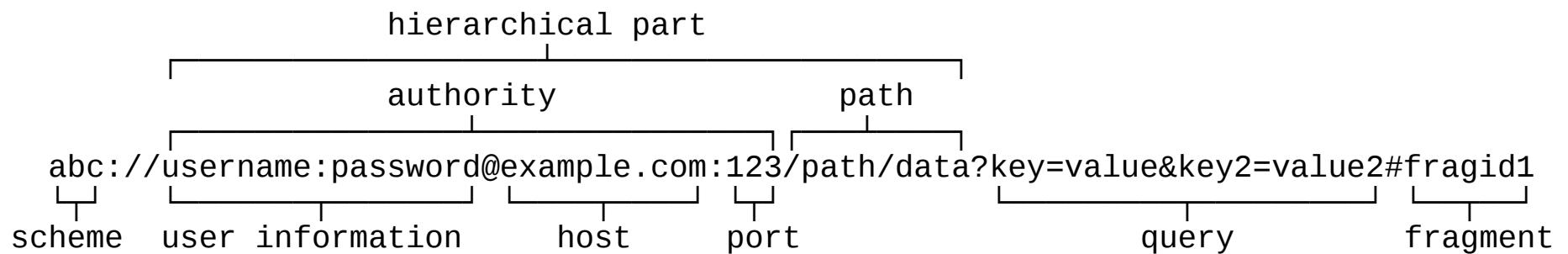
URI	Uniform Resource Identifier
URL	Uniform Resource Locator



URI

<scheme>:<scheme-specific-structure>

URI

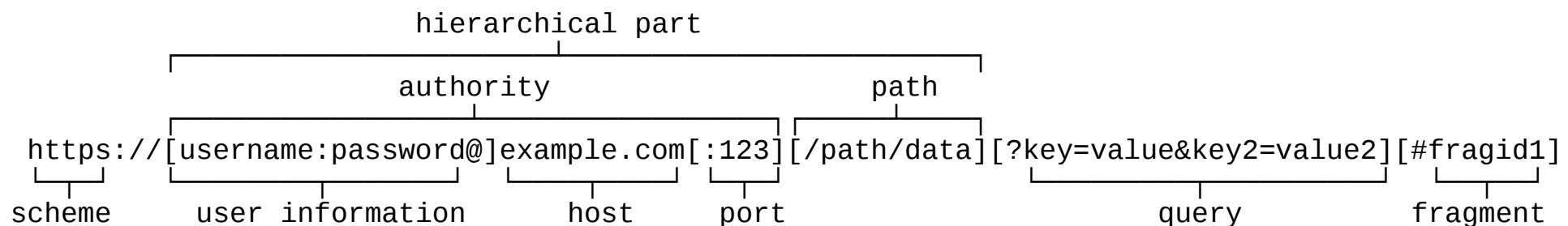


Exempel på URI:er

- ISBN
 - isbn:978-0-33-050811-7
- Mailto Scheme URI (e-post)
 - mailto:johan.holmberg@mah.se
- Tel URI (telefoni)
 - tel:+421-2-529-263-67
- Spotify URI
 - spotify:user:idioti.se:playlist:4xvCr0XKcPCJqzAYTHIR3E
- HTTP Scheme URI (URL)
 - http://www.mah.se/dk

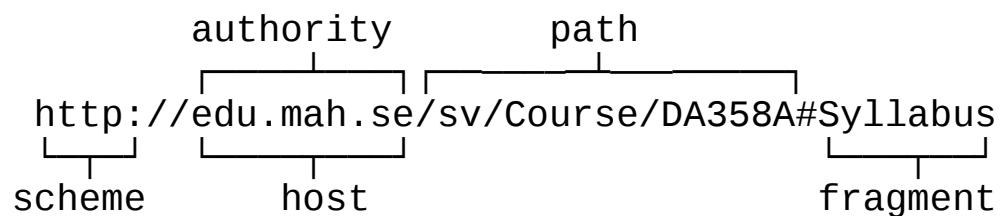
URL

HTTP-schemat används för att lokalisera resurser på nätverket via HTTP-protokollet.



Exempel på URL

Lägg märke till att endast *scheme* och *host* är obligatoriska delar



Representationer



REST och representation

- Resurser och hur vi når dem är RESTs fokus
- Representationen är ett sätt att beskriva informationen hos en resurs
- Resursen kan bestå eller bero på andra resurser
 - Representationen bör kunna hantera detta
- Resursens representation är inte definierad i någon standard
 - En resurs kan representeras av flera olika format

Olika sätt att representera en resurs

Ada Lovelace

Ada Lovelace [redigera](#) [redigera wikitext](#)

Ada Lovelace, egentligen *Augusta Ada King*, grevinna av Lovelace, född 10 december 1815 i London, död 27 november 1852 i London, var en brittisk matematiker och skribent. Hon är mest hägkommen för sitt arbete med Charles Babbage:s maskin. Lovelace innehåller den första algoritmen som är avsedd att bearbetas med maskinen.

Lovelace var enda barn till poeten Lord Byron och dennes fru Anne Isabella Millett. Hon föddes i England fyra månader senare; han dog till slut i sjukdom under Greklandkriget. Lovelace främjade dottterns intresse för matematik och logik i ett försök att förhindra att sitt intresse för sin far och begravdes enligt sin sista vilja vid sidan av honom.

Ada Lovelace beskrev sitt arbetsätt som "poetisk vetenskap"^[5] och sätter sig i talanger henne fram till ett långvarigt samarbete med en annan brittisk matematiker, Charles Babbage. Åren 1842–43 översatte hon en artikel om maskinen skriven av Babbage om "en notapparat". Dessa noter innehåller vad många^[vilka?] anser vara det första programmet som producerar siffror.^[7] Hennes "poetiska vetenskap" drev henne fram till att hon i sambhället å ena sidan och tekniken som ett redskap å den andra.^[8]

Innehåll [dölj]

1 Biografi

1.1 Barndom

1.2 Vuxna år

42 2021

Ada Lovelace

Inledning

Ada Lovelace, egentligen Augusta Ada King, född 10 december 1815 i London, död där 27 november 1852, var en brittisk skribent. Hon är mest hägkommen för sitt arbete som förstörde den analytiska maskinen. Hennes anteckningar om den analytiska maskinen innehöll den första algoritmen som är avsedd att bearbetas av en dator, och därmed som historiens första datorprogrammerare.

Lovelace var enda barn till poeten Lord Byron. Makarna separerade en månad efter att dottern föddes. Lovelace föddes i maj 1815 och dog i oktober samma år. Hon var åtta år gammal. Adas mor var bitter på grund av att hennes man inte visste om hennes intresse för matematik och logik i ett försök att utveckla den galenskap som hon såg hos barnen. Lovelace föddes i en familj där hennes far och hennes farfar var författare och hennes farfar var en författare och hennes farfar var en författare.

Biografi

Barndom

Ada Lovelace föddes 10 december 1815 som dotter till baronen av Byron, och Anne Isabella "Annabel" George Byron förväntade sig att hans nya ätt



Vanliga format

- HTML
 - Används oftast för att ge ett visuellt svar – en webbsida
- XML
 - Används ofta när detaljer, såsom datatyper, är viktiga
 - Mycket vanligare förr i tiden
- JSON (**JavaScript Object Notation**)
 - Mer kompakt än XML, sparar bandbredd
 - Ofta mer läsbart för människor
- CSV, Excel, PDF, PNG, etc.

XML och JSON

```
<?xml version="1.0"?>
<unicorn>
  <id>4</id>
  <name>Tordönsenhörning</name>
  <description>Det här är en irriterande
    tvivelaktiga näjet att stifta bek.
    .</description>
  <reportedBy>Johan</reportedBy>
  <spottedWhere>
    <name>Söderbärke, Sverige</name>
    <lat>59.9801</lat>
    <lon>15.4446</lon>
  </spottedWhere>
  <spottedWhen>2010-05-08 00:05:00</spottedWhen>
  <image>http://unicorns.idioti.se/bilder/tordon.jpg</image>
</unicorn>
```

```
[{"id": "4",
  "name": "Tordönsenhörning",
  "description": "Det här är en irriterande
    tvivelaktiga näjet att stifta bekantskap med.
  "reportedBy": "Johan",
  "spottedWhere": {
    "name": "Söderbärke, Sverige",
    "lat": "59.9801",
    "lon": "15.4446"
  },
  "spottedWhen": {
    "date": "2010-05-08 00:00:00.000000",
    "timezone_type": 3,
    "timezone": "UTC"
  },
  "image": "http://unicorns.idioti.se/bilder/tordon.jpg"
}]
```



Att begära ett format

De flesta REST-tjänster som erbjuder mer än ett format gör det på något av följande sätt:

- Genom sökvägen (del av URLen, usch):
 - `http://www.example.com/json/horses`
- Genom queries (söksträngar i URLen, meh):
 - `http://www.example.com/horses?format=json`
- Genom headers (helt klart vackrast):
 - `http://www.example.com/horses`
 - **Accept: application/json**

Exempel på olika format

Vi använder Johans eminenta enhörnings-API

Vad är REST?



Vad är REST?

- **Representational State Transfer**
- En arkitektonisk stil för att designa nätverksbaserade applikationer
- Syftar till att förenkla datautbyte mellan olika applikationer
 - Använd naken HTTP istället för exempelvis SOAP eller mer komplexa tekniker
 - HTTP-standarden definierar metoderna *Get*, *Post*, *Put* och *Delete* för CRUD-operationer.
 - HTTP-standarden definierar ett antal statuskoder för att förmedla resursens tillstånd



REST som webbtjänst

- Som de flesta andra tekniker för webbtjänster (exempelvis Web Services) är REST
 - plattformsoberoende
 - språkoberoende
 - byggt på vedertagna standarder (HTTP)
- REST erbjuder exempelvis inte säkerhet, kryptering eller sessionshantering, men detta kan realiseras:
 - Autentiering med REST görs ofta med tokens
 - Kryptering med REST görs lättast med HTTPS
 - Sessionshantering görs ofta med cookies och tokens

REST som webbtjänst (forts)

- **ST** i REST står för *State Transfer*, vilket innebär
 - Varje anrop är självständigt och innehåller all information som behövs för att genomföra en förfrågan → det är idempotent

REST – Meddelanden

- Klient och server pratar med varandra genom meddelanden
 - Det är klienten som initierar kommunikationen
- Klienten skickar en förfrågan till servern genom att
 - Data skickas Request Body
 - Metadata skickas som Request Headers



Mer komplexa REST-anrop

- Förra exemplet använde bara en parameter (ett användar-id)
 - Det är enkelt att bygga in fler parameterar i sina anrop

```
http://www.acme.com/phonebook/UserDetails?firstName=John&lastName=Doe
```

- Behöver man skicka många parametrar eller mycket data (exempelvis en fil) används metoden POST
- Tumregler:
 - För att läsa data – använd GET, som inte ska förändra tillståndet hos en resurs
 - För att skriva data – använd POST, PUT eller DELETE.