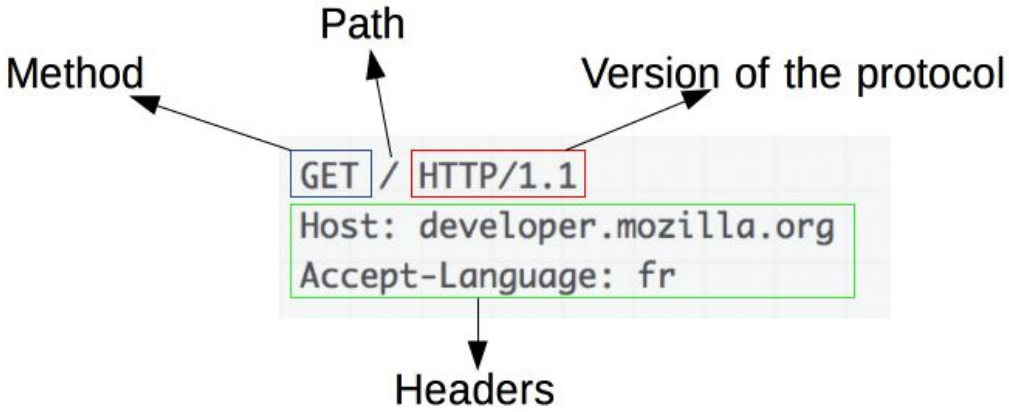
Föreläsning 2

HTTP, REST

Hypertext Transfer Protocol (HTTP)

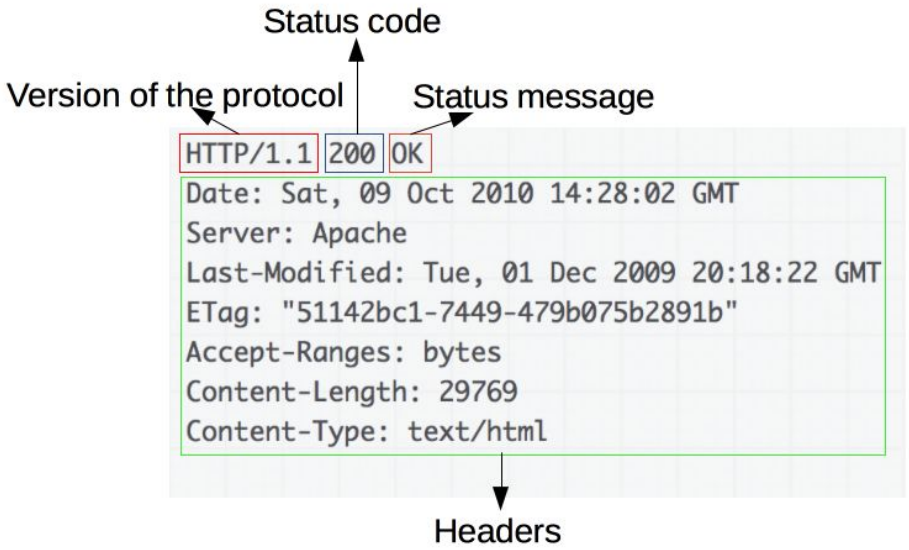
HTTP är det kommunikationsprotokoll som används för skicka data på internet.

En HTTP request som ut som följande



HTTP Response

En HTTP response ser ut som följande



HTTP body

Både requesten och responsen kan innehålla en body. En body är den data som skickas med anropet. Formatet på datan anges med `Content-Type` headern. Helst ska `Content-Length` också vara med när vi skickar data och ange hur mycket data som skickas.

Media typer (MIME)

Media typer (ofta kallat MIME typer) är standardiserade värden som beskriver formatet på data. Vi använder dessa typer i våra headers som berättar typen av data (t.ex. Content-Type). Vanliga typer är

- application/javascript
- application/json
- application/x-www-form-urlencoded
- text/css
- text/html
- text/plain
- image/jpeg
- image/png

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Complete_list_of_MIME_types

HTTP-statuskoder

Statuskoderna i HTTP är fördefinierade. En statuskod indikerar om servern lyckades hantera vår förfrågan. Vanliga statuskoder:

- 200 - OK
- 201 - Created
- 204 - No Content
- 301 - Moved Permanently
- 400 - Bad Request
- 401 - Unauthorized
- 404 - Not Found
- 500 - Server Error

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

HTTP Headers

Vi kan ange vad vi vill i våra headers - men det finns många standarder som vi ska följa. Vanliga request headers är

- `Content-Type` - Formatet på body datan om någon skickas
- `Content-Length` - Hur mycket data som skickas om någon
- `Accept` - Vilket format vi vill ha på datan i svaret
- `Cookie` - En cookie som skapats av servern tidigare
- `Host` - Namnet på servern vi skickar en request till. Obligatorisk header
- `User-Agent` - Vad det är för typ av klient som gör requesten

HTTP Headers

Vanliga response headers är:

- `Content-Type` - Formatet på body datan om någon skickas
- `Content-Length` - Hur mycket data som skickas om någon
- `Cache-Control` - Information om cachen
- `Location` - Vart man ska gå vid en redirection
- `Set-Cookie` - En cookie som ska sparas och därefter skickas i cookie headern

En mer fullständig lista hittar ni på

https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

HTTP metoder

En HTTP metod (eller verb) är en identifikation som berättar vad det är för typ av request vi gör.

Metoderna som finns är

- GET
- HEAD
- POST
- PUT
- DELETE
- TRACE
- OPTIONS
- CONNECT
- PATCH

REST

- **REST** står för Representational state transfer
- Det är det absolut vanligaste sättet att designa API:er för webben
- I REST används HTTP-verben tillsammans med URL:er för att utföra olika operationer

REST - GET

- I REST används verbet GET för att hämta en lista av objekt eller ett specifikt objekt
- Om vi har en resurs som vi kallar “movies” kan vi använda GET på följande vis

```
GET /movies
```

- Servern kommer att svara med en lista över filmer
- Om vi tänker att alla filmer har ett unikt id kan vi även hämta en specifik film med

```
GET /movies/star-wars
```

REST - POST

- POST används för att lägga till en ny resurs i ett REST-API
- Själva datan brukar skickas som JSON men i vissa fall som XML
- Så för att lägga till en ny film används

```
POST /movies
```

- Datan som skickas till API:et brukar kallas för “payload”
- När datan skickas brukar API:et svara med statuskoden 201 (Created) för att berätta för klienten att allt gick bra
- Om servern inte accepterar det som skickats svarar den med statuskod 400 (Bad request)

REST - PUT

- PUT används för att uppdatera en viss resurs. Datan som skickas är samma som för post
- Så om vi vill uppdatera en viss film kan vi göra

```
PUT /movies/star-wars
```

- Precis som med POST kommer servern att svara med statuskoden 400 (Bad request) om det är något problem med datan som skickats
- Om allt går bra svarar servern med statuskod 200

REST - PATCH

PATCH fungerar snarlikt till PUT.

Skillnaden är att om vi gör ett PUT anrop till en resurs /movies/star-wars så ska resursern som ligger på routen skrivas över men den nya datan.

Medans om vi gör ett PATCH anrop ska enbart de fält vi skickar med i anropet uppdateras på resursen.

PUT skriver över en resurs, medans PATCH uppdaterar en eller flera delar av en resurs.

REST - DELETE

- För att ta bort resurser i ett REST-API används DELETE

```
DELETE /movies/star-wars
```

- Om servern lyckas ta bort resursen svarar den oftast med statuskod 200 (Accepted), eller i vissa fall 204 (No Content)

Övning - REST

Du ska skapa en server som hanterar en specifik resurs (t.ex. filmer). Servern ska hantera att skapa resurser, hämta resurser (alla och på specifikt id), uppdatera resurser och ta bort resurser (GET, POST, PUT, DELETE). Resurserna kan du enkelt spara i minnet på servern (t.ex. som en global array). Det är rekommenderat att jobba med all data som JSON (glöm inte en JSON parsing middleware).

Bonuspoäng - Lägg till validering på data och returnera logiska statuskoder (inte bara 200/500).

Bonuspoäng 2 - Lägg även till PATCH

Node skapa filer

I node.js använder vi oss av `fs` modulen för att hantera filer. Vi kan inte bara läsa filer med modulen, utan vi kan också skapa och skriva till filer.

```
fs.writeFile('myfile.json', JSON.stringify(data), function(err) => {
    ...
});
```

Övning - enkel databas

- Lägg till enkel databas logik i den förra övningen.
- När man ändrar/skapar nya resurser så säg till att du också sparar datan till en fil (enklast som JSON).
- När man startar servern ska du kolla om det existerar en fil med data redan och isåfall ladda in den.