LINKgroup

Distance Learning System

HTTP protokol

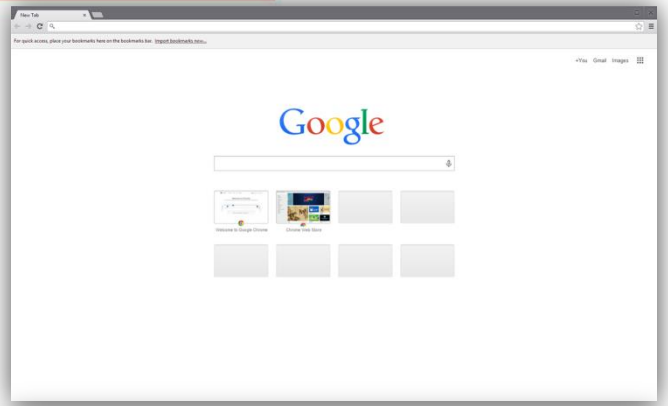
Python Network Programming

Izvršavanje web aplikacija

- Često se web serveri nazivaju i application serveri, zato što su u stanju da startuju i određene aplikacije prilikom obrade klijentskog zahteva.
- Onog trenutka kada određena web prezentacija (sajt) u svom postojanju upotrebi neku od aplikacija (funkcionalnosti) na serveru, ona, zapravo, postaje web aplikacija. Programski kod (program) koji se izvršava na serveru prilikom aktivacije klijentskog zahteva, naziva se **serverski kod**.
- Razlikuje se više načina izvršavanja serverskog koda, odnosno, više mogućnosti njegove implementacie. Kada web server primi klijentski zahtev, čita njegove karakteristike (sadržane u zaglavlju zahteva) i na osnovu njih izvršava određenu akciju. Ukoliko zahtev ne sadrži zahtev za aktivaciju web aplikacije, već samo potražuje neki dokument sa servera, server će pronaći dokument na fajl sistemu, a zatim ga proslediti klijentu kroz odgovarajući odgovor. Ovakav scenario podrazumeva preuzimanje statičkih sadržaja sa web servera (html dokumenata, slika i sl.).
- Ukoliko zahtev zahteva angažovanje serverskog koda, procedura je drugačija. Server, takođe, pronalazi dokument, ali ga prosleđuje **hendleru**, koja ga izvršava i prosleđuje klijentu.

Web klijent

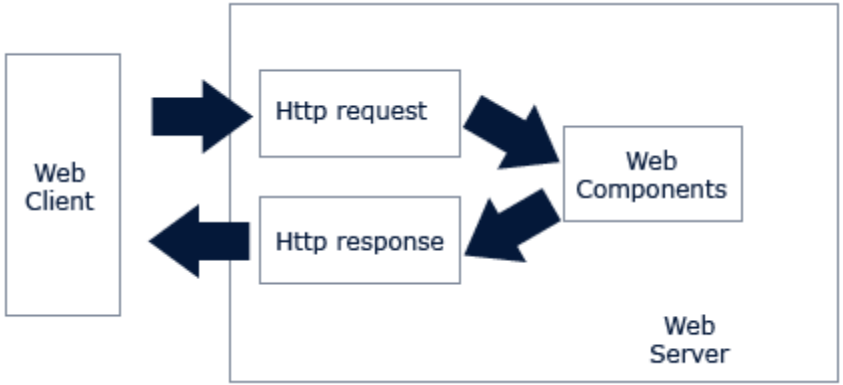
- Svaka web aplikacija ima dve strane u toku svog izvršavanja. Jedna strana je serverska, a druga klijentska. Klijentska strana je krajnja tačka izvršavanja web aplikacije i ona, obično, takođe, sadrži logiku koja je u stanju da pošalje zahtev serveru, kao i da preuzme i adekvatno pročita odgovor. Web pretraživač (mozilla, internet explorer, opera, safari...) je najčešći oblik klijentske web aplikacije. Ali konzument web aplikacije ne mora biti obavezno web pretraživač, već može biti bilo koja aplikacija, sve dok u svom izvršavanju podrazumeva i komunikaciju sa web serverom.



Problemi u web programiranju

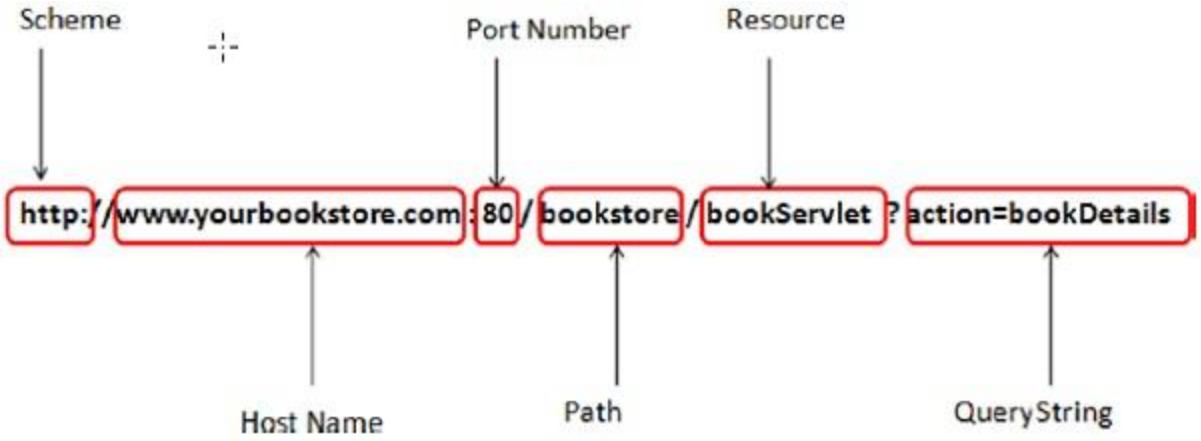
- Koncept klijent/server je izuzetno problematičan sa stanovišta
  - Bezbednosti
  - Brzine transporta informacija

- Postoje razne tehnike kojima se ovi problemi zaobilaze, ali je i dalje, baš zbog njih, kreiranje web aplikacija jedan od najvećih programerskih izazova danas



Osnove HTTP-a

- HTTP je request-response protokol, što znači da njegov način funkcionisanja počiva na međusobnom smenjivanju zahteva i odgovora između klijenta i servera. Kada klijent zatraži neki resurs, taj zahtev obično sadrži identifikaciju resursa koji je zahtevan, i to u formi Uniform Resource Locator-a (URL). URL je hijerarhijska sekvenca komponenti, struktuirana kao na slici



HTTP zahtevi i odgovori (zahtev)

- HTTP komunikacija između klijenta i servera započinje HTTP zahtevom klijenta
- Zahtev sadrži u sebi i HTTP metod, odnosno obrazac koji veb-server treba da ispoštuje prilikom obrade zahteva i rukovanja resursima. Metod bi najbolje mogao da se okarakteriše kao najobičnija naredba. Ako bismo npr. otkucali u pretraživaču www.mysite.com/index.html, mogli bismo biti sigurni da će negde u našem zahtevu postojati linija:

  GET /index.html HTTP/1.1

  Kao i linija:

  host: www.mysite.com
- gde je:

  GET naziv metoda,
  /index.html adresa koja se zahteva,
  HTTP/1.1 verzija protokola i
  host: www.mysite.com naziv top domena za taj sajt

HTTP zahtevi i odgovori (odgovor)

- Na osnovu ovih informacija web server formira odgovor i prosleđuje ga klijentu.
- Na zahtev formatiran na pomenuti način, server formira odgovor i prosleđuje ga nazad pošiljaocu (a server u svakom trenutku zna IP adresu i port, gde se od njega očekuje da prosledi odgovor).
- Odgovor servera sadrži status – **200** ako je sve u redu i mnoge druge brojeve grešaka, ukoliko nešto nije kako treba. Brojevi sa kojima se najčešće susreće su **404** (tražena strana ne postoji) i **500** (greška u veb-aplikaciji). Zapravo, statusi se po brojevima dele na određene grupacije: **1xx za informacije, 2xx sve vrste uspešnih apstrakcija, 3xx redirekcije, 4xx klijentske greške, 5xx serverske greške**.

HTTP Request / response

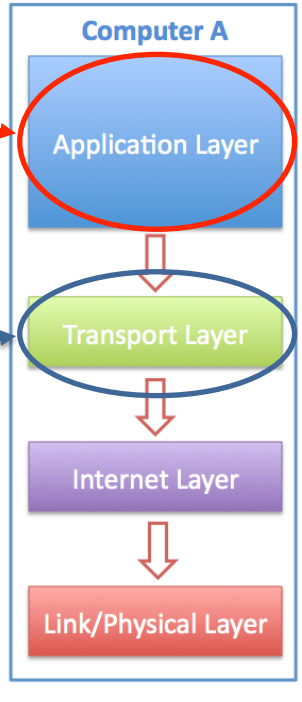
- Korisnik unosi adresu:
  - http://www.mysite.com/index.html
- User Agent otvara Socket preko porta 80 i šalje zahtev preko njega:
  - GET /path/file.html HTTP/1.0
    host: www.mojsajt.com
- Server, nakon što prihvati zahtev, na isti port, preko istog Socketa, šalje sledeći odgovor:

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 500

<html>
        <body>
                some server response...
        </body>
</html>
```

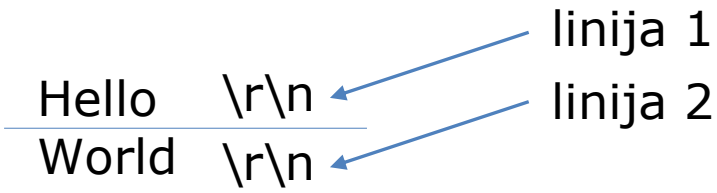
HTTP infrastruktura

- HTTP je aplikativni protokol u četvoroslojnom, IP modelu
- Ovo znači da nije bitno na kom će transportnom protokolu počivati
- Praksa je ipak, da se za HTTP protokol, koristi **TCP**
- Sistem komunikacije treba da bude takav da dva sagovornika "govore" naizmenično a ne "u glas"



Način razmene informacija

- U HTTP-u, nakon uspostavljene veze, uvek započinje program koji je inicirao komunikaciju (ovaj program se zove **klijent**, dok se program koji očekuje komunikaciju, naziva **server**)
- Tokom komunikacije, sagovornici koriste bajtove, koji predstavljaju tekst (strimove karaktera), tako da je kompletan "razgovor" razumljiv i ljudskom oku
- Kompletan "razgovor" podeljen je na celine, od kojih je najmanja celina - **red**.
- Red je serija karaktera (bajtova) koja se završava oznakom za novi red (ovo je bajt 0A) ili oznakom za povratak na početak reda i novi red (carriage return, bajt 0D).



HTTP poruke

- HTTP podrazumeva slanje pitanja serveru i dobijanje odgovora od servera
- Ova pitanja i odgovori imaju istu strukturu i zovu se **HTTP poruke**
- Jedna HTTP poruka se sastoji od redova teksta koji se dele u dve grupacije: **zaglavlje** i **telo**
- Zaglavlje je prvi, obavezni deo poruke, i ono se sastoji od **opisa** i **zaglavlja** (mn.)
- Opis sadrži ključne podatke zahteva, dok zaglavlja sadrže dodatne, manje važne informacije

```
GET /mypage HTTP/1.1
Host: myhost.com
Connection: Close
…
…
```

Opis HTTP zahteva

- Opis HTTP zahteva sadrži tri informacije bitne serveru
  - HTTP **metod**
  - **Putanju** do traženog dokumenta
  - **Verziju** HTTP protokola

GET /mypage HTTP/1.1

- Metod je **GET**, traženi dokument je **/mypage** a verzija protokola **1.1**
- Ovo za server jednostavno znači: *dostaviti fajl mypage*

HTTP metod

- HTTP metod je prvo što server dobija od klijenta
- Ovaj metod govori serveru na koji način treba da tretira zahtev
- HTTP metod može biti jedan od ponuđenih:

Za većinu standardnih web aplikacija, metodi GET i POST su dovoljni

GET
POST
PUT
DELETE
PATCH
HEAD
OPTIONS
TRACE
CONNECT

HTTP metodi GET i POST

- Metodi GET i POST se razlikuju kontekstualno ali i fizički
- **Kontekstualno** razlika je u tome što metod GET služi za dobavljanje podataka dok se metod POST najčešće koristi za unos i ažuriranje podataka
- **Fizički**, POST se razlikuje od GET metode po tome što **može sadržati telo**

```
GET /mypage HTTP/1.1
Host: myhost.com
Connection: Close
…
…
```

```
POST /mypage HTTP/1.1
Host: myhost.com
Connection: Close
…

Hellooooooooo!!!!!
```

Zaglavlja HTTP zahteva

- Zaglavlja (mn.) su serija parova ključeva i vrednosti u okviru zaglavlja zahteva
- Ključevi i vrednosti se odvajaju oznakom **:**
- Iako ni jedno zaglavlje nije ključno za server, postoji zaglavlje koje se smatra obaveznim u web aplikacijama (Host zaglavlje)
- Nakon ispisanih svih zaglavlja, ispis se prekida sa praznim redom

```
GET /mypage HTTP/1.1
Host: localhost
Connection: close
```

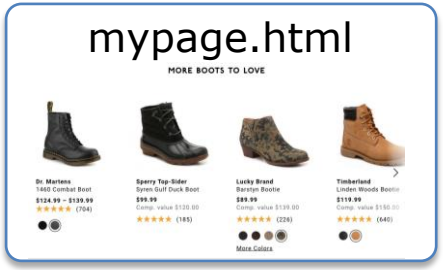
Server zna da je kraj zahteva po praznom redu

Host zaglavlje

- Na osnovu host zaglavlja, server može preciznije da grupiše fajlove i tako hostuje više različitih domena

c:/sites/mysite.com/

mypage.html



GET /mypage.html HTTP/1.1
Host: mysite.com
Connection: close

c:/sites/yoursite.com/

mypage.html

GET /mypage.html HTTP/1.1
Host: yoursite.com
Connection: close

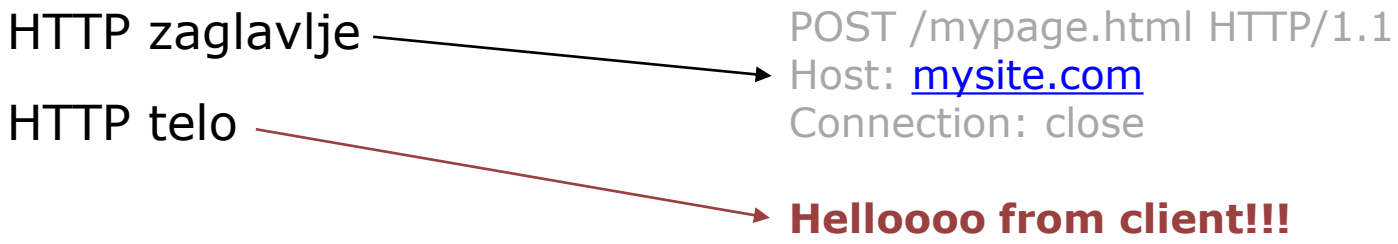
Još interesantnih zaglavlja

- Osim pomenutog host zaglavlja, postoji još dosta zaglavlja koji se u HTTP-u mogu koristiti u kontekstu HTTP zahteva
- Neki od interesantnijih (ali narevno ne svi) su:
- **Connection**
  Da li konekcija treba da bude zadržana ili prekinuta nakon uspešno obavljene komunikacije
- **Accept**
  Koji tip podataka je dozvoljen u odgovoru
- **Content-Length**
  Dužina tela u bajtovima
- **Content-Type**
  Tip sadržaja

Pogledaj još hedera i njihovih opisa:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

Telo HTTP zahteva - HTTP body

- HTTP poruka sadrži i telo
- U slučaju HTTP zahteva, ovo telo može, u zavisnosti od metode i konteksta poruke, biti prisutno ili izostati
- Telo počinje onog trenutka kada ispišemo prazan red u zahtevu (odnosno, kada ispišemo dva nova reda)



Telo HTTP zahteva - HTTP body

- Telo HTTP zahteva nije dozvoljeno za svaki HTTP metod

```
POST /mypage.html HTTP/1.1
Host: mysite.com
Connection: close

Helloooo from client!!!
```

```
GET /mypage.html HTTP/1.1
Host: mysite.com
Connection: close

Helloooo from client!!!
```

Sadržaj tela HTTP zahteva

- Telo može imati bilo koji sadržaj i njegova veličina može (hipotetički) biti neograničena
- Obzirom na širok dijapazon tipova sadržaja, klijent je dužan da, ukoliko šalje neki sadržaj, obavesti server o njegovom tipu
- Ovo se obavlja pomoću zaglavlja **Content-Type**.

```
POST /mypage.html HTTP/1.1
Host: mysite.com
Content-type: image/jpeg

Hello
```

```
POST /mypage.html HTTP/1.1
Host: mysite.com
Content-type: text/html

Hello
```

Iako se u tela zahteva identična, različiti tipovi će učiniti da ih različito tretiramo. Ne zaboravite, da je u telu zapravo jedino napisano:

01001000 01100101 01101100 01101100 01101111

MIME tipovi

- **M**ultipurpose **I**nternet **M**ail **E**xtensions types
- Način na koji se tretira sadržaj zavisi u HTTP-u od njegovog MIME tipa
  - MIME tip se označava sledećom sintaksom:

    tip/podtip *ili* ***tip/podtip;parametri***
- Lista postojećih MIME tipova

  https://www.iana.org/assignments/media-types/media-types.xhtml
- Lista postojećih MIME tipova

Dužina tela HTTP zahteva

- Slanje tela u HTTP poruci zahteva dolazi sa probemom njegove dužine
- U telu nije moguće obaviti delimiterom, jer sve što je definisano kao delimiter, može se pojaviti i u sadržaju tela
- Ovaj problem nije toliko izražen u telu odgovora, jer njemu može uslediti kraj konverzacije koja se može označiti prekidom konekcije
- Klijent mora poslati tačnu dužinu sadržaja tela serveru, kako bi server znao kada da prekine sa slušanjem/čitanjem, a započne upis
- Dužina tela se predstavlja zaglavljem **Content-Length**, dužinom u bajtovima

```
POST /mypage.html HTTP/1.1
Host: mysite.com
Content-Length: 8

Helloooo
```

Dužina tela HTTP zahteva

- Poslata dužina mora da odgovara stvarnoj dužini tela
- Ukoliko je dužina tela veća od poslate dužine, server samo prekida čitanje i počinje upis
- Ukoliko je dužina tela kraća od poslate dužine, server može ostati na čekanju na ostatak sadržaja

```
POST /mypage.html HTTP/1.1
Host: mysite.com
Content-Length: 3

Helloooo
```

Server čita tri bajta
a zatim odgovara

```
POST /mypage.html HTTP/1.1
Host: mysite.com
Content-Length: 20

Helloooo
```

Server čita osam bajtova
a zatim čeka na još 12 bajtova
koji nikada ne stižu

HTTP Odgovor

- Odgovor servera je strukturalno isti kao i pitanje klijenta, osim što se umesto opisa zahteva, pravi opis odgovora

Opis odgovora →

HTTP/1.1 200 OK
Connection: close

Helloooo from server!

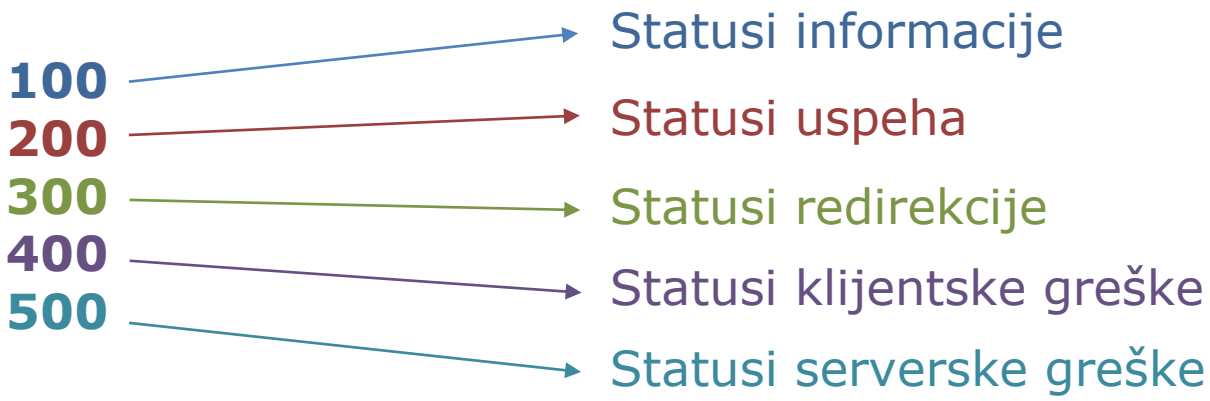
Opis odgovora se sastoji od tri dela:

Statusne poruke → OK
Statusnog koda → 200
Verzije HTTP protokola → HTTP/1.1

HTTP Statusni kodovi

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

- Statusni kod i poruka uvek idu u kombinaciji a definisani su specifikacijom HTTP-a
- U HTTP specifikaciji razlikujemo pet kategorija statusnih kodova
- Svaka kategorija počinje stotinom između 100 i 500
- Kategorije statusnih kodova su:



Statusni kod 100 - Informacija

- Statusni kod 100 obaveštava klijenta o nečemu vezanom za zahtev
- Ovaj statusni kod se može pojaviti u poruci pre slanja konačnog odgovora (dakle, može se očekivati još neka poruka)
- Statusni kod 100 podrazumeva sledeće kodove / poruke:

100 Continue

101 Switching Protocol

102 Processing

103 Early Hints

Statusni kod 200 - Ispravno procesiranje

- Statusni kod 200 predstavlja varijacije uspešne obrade zahteva

200 OK
201 Created
202 Accepted
203 Non-Authoritative Information
204 No Content
205 Reset Content
206 Partial Content

Statusni kod 300 - Redirekcije

- Statusni kod 300 obaveštava klijenta da treba da redirektuje zahtev

300 Multiple Choice
301 Moved Permanently
302 Found
303 See Other
304 Not Modified
305 Use Proxy
306 unused
307 Temporary Redirect
308 Permanent Redirect

Statusni kod 400 - Klijentske greške

- Statusni kod 400 obaveštava klijenta da postoji greška u njegovom zahtevu

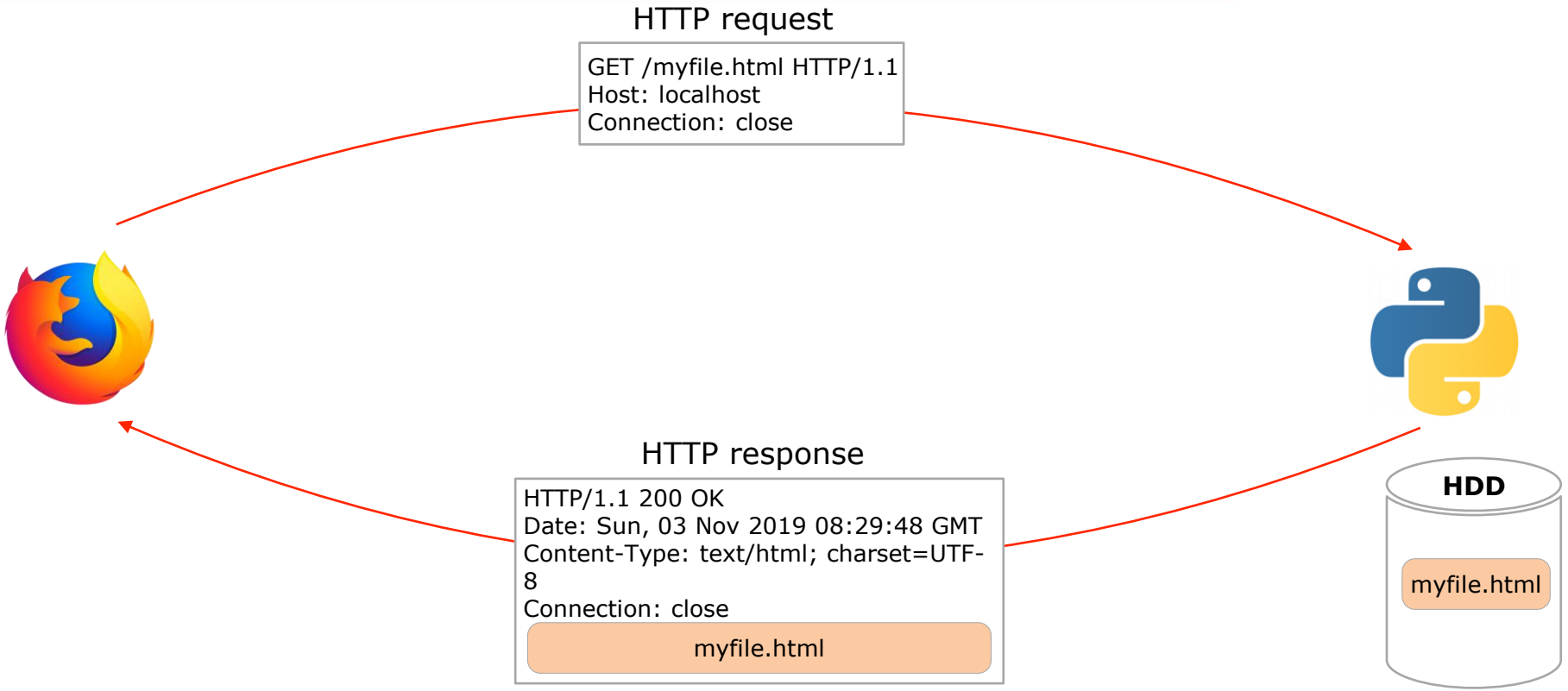
400 Bad Request
401 Unauthorized
402 Payment Required
403 Forbidden
404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout
409 Conflict
410 Gone
411 Length Required
412 Precondition Failed
413 Payload Too Large
414 URI Too Long
415 Unsupported Media Type
416 Requested Range Not Satisfiable
417 Expectation Failed
418 I'm a teapot
421 Misdirected Request
422 Unprocessable Entity
423 Locked (WebDAV)
424 Failed Dependency
425 Too Early
426 Upgrade Required
428 Precondition Required
429 Too Many Requests
431 Request Header Fields Too Large
451 Unavailable For Legal Reasons

Statusni kod 500 - Redirekcije

- Statusni kod 500 ukazuje na grešku uzrokovanu programom na serveru

500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported
506 Variant Also Negotiates
507 Insufficient Storage
508 Loop Detected
510 Not Extended
511 Network Authentication Required

Kompletna procedura HTTP request/response



Vežba 1 (pnp-ex02 hellohttp)

- Kreirati jednostavan HTTP server koji procesira zahtev tako što klijentu u odgovoru vraća naziv traženog fajla (ne fajl, nego samo naziv)
- Na primer, ako je zahtev:

  GET /file.txt HTTP/1.1
  Host: localhost

- Odgovor će biti:

  HTTP/1.1 200 OK

  file.txt

- Testirati server pomoću web pregledača

Vežba 2 (pnp-ex02 tickets)

- Kreirati HTTP server za registraciju karata
- Server prihvata dva tipa zahteva
  - Prvi tip zahteva generise id karte i čuva ga u fajlu
  - Drugi tip zahteva pronalazi id karte u fajlu i, ako postoji, markira ga kao iskorišćen
  - Svaka karta se u fajlu predstavlja jednim redom sledećeg sadržaja:
    - id_karte|datum izdavanja|validna
    - Na primer:

      1|2019-10-01 18:03:45.476064|1

      2|2019-10-03 14:05:23.382341|0

      3|2019-10-05 19:00:03.234564|1