

Podatki na spletu

Pridobivanje podatkov iz spleta

- ▶ Na spletu so podatki na voljo kot:
 - ▶ vsebina spletne strani (HTML)
 - ▶ datoteka za nalaganje (ang. download)
 - ▶ klici preko aplikacijskega programskega vmesnika API (XML, JSON)
- ▶ Spletne strani pregledujemo s spletnim brskalnikom
- ▶ Protokol HTTP:
 - ▶ IP naslov ali URL strežnika (193.2.67.103, <http://www.fmf.uni-lj.si>)
 - ▶ tekstovni protokol
 - ▶ zahtevek in odgovor (ang. request, response)
- ▶ Pregled prometa, vsebine:
 - ▶ Firbug na Firefoxu (add-on)
 - ▶ "Inspect element" na Google Chrome

Protokol *HTTP*

- ▶ Tekstovni protokol nad protokolom TCP/IP
- ▶ Klient pošlje poizvedbo
- ▶ Strežnik odgovori
- ▶ V vsakem pogovoru najprej nekdo vpraša (klient), drugi pa odgovori (strežnik)
- ▶ Dva računalnika lahko izmenjujeta vlogi
- ▶ URL - ang. Uniform Resource Locator - identifikator vira

Protokol *HTTP*

- ▶ Primer: `http://www.fmf.uni-lj.si:80/pot/do/vira?a=1&b=2`
- ▶ protokol `http://`
- ▶ ime strežnika `www.fmf.uni-lj.si`
 - ▶ vsako ime se preslika v IP številko
 - ▶ za preslikavo skrbijo DNS strežniki (del TCP/IP omrežja)
- ▶ lokalna pot do vira na strežniku `/pot/do/vira/`
 - ▶ lahko je to dejanska pot na disku od neke mape dalje
 - ▶ lahko je pač samo simbolična pot (ang. route), ki sproži neko funkcijo, ki vrne odgovor
- ▶ parametri za poizvedbo oblike `ključ=vrednost`, za znakom `'?'`, pari so ločeni z znakom `'&'`

Zahtevek (ang. *request*)

- ▶ Oblika (tekstovna):
- ▶ Začetna vrstica (ukaz + pot do vira + št. protokola, npr. GET /pot/do/vira/ HTTP/1.0)
 - ▶ ukazi: GET, POST, DELETE, ...
- ▶ Ena ali več vrstic glave (ang. Header)
 - ▶ vrstice oblike: ključ: vrednost1, vrednost2
 - ▶ ena ali več tekstovnih vrednosti ločenih z vejico
 - ▶ primer: Last-Modified: Fri, 31 Dec 1999 23:59:59 GMT
- ▶ Prazna vrstica
- ▶ Opcijsko telo zahtevka (poljuben tekst)
 - ▶ poljuben tekst
 - ▶ za pravilno interpretacijo imamo v glavi dva ključa
 - ▶ Content-Type - določa tip vsebine, oz. ang. MIME type (MIME = Multipurpose Internet Mail Extensions)
 - ▶ Content-Length - število bytov v telesu
 - ▶ Primer uporabe telesa: nalaganje datoteke

Kaj pošilja spletni brskalnik?

- ▶ V Chrome vklopimo Inspect element, preklopimo na Network
- ▶ Vpišemo v naslovno vrstico: `http://httpbin.org/get`
- ▶ Ko gremo na neko spletno stran preko naslova, brskalnik vedno uporabi ukaz GET:

```
GET /get HTTP/1.1
Host: httpbin.org
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/4
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cookie: _ga=GA1.2.2024541521.1447802935; _gat=1
```

Odgovor (ang. *response*)

- ▶ Oblika (tekstovna) je podobna obliki zahtevka
 - ▶ začetna vrstica (št. protokola + koda odgovora + opis odgovora)
 - ▶ primer: HTTP/1.0 200 OK
 - ▶ ena ali več vrstic glave (ang. Header)
 - ▶ prazna vrstica
 - ▶ opcijsko telo
- ▶ Kode odgovora:
 - ▶ 1xx - vsebina je zgolj informacijska
 - ▶ 2xx - uspešno pridobljena vsebina (npr. 200 OK)
 - ▶ 3xx - prevezava (ang. redirect, npr. 301 Moved Permanently, 302 Moved Temporarily)
 - ▶ 4xx - napaka na strani klienta (npr. 404 Not Found, vir ne obstaja)
 - ▶ 5xx - napaka na strani strežnika (npr. 500 Server Error, strežnik se je pokvaril, ...)

Kaj bi pa odgovoril strežnik?

- ▶ Na prejšnji zahtevek strežnik odgovori takole

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 17 Nov 2015 23:29:16 GMT
Content-Type: application/json
Content-Length: 585
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
.
{
  "args": {},
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
    "Accept-Encoding": "gzip, deflate, sdch",
    "Accept-Language": "en-US,en;q=0.8",
    "Cache-Control": "max-age=0",
    "Cookie": "_ga=GA1.2.2024541521.1447802935; _gat=1",
    "Host": "httpbin.org",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) ",
  },
  "origin": "84.20.227.68",
  "url": "http://httpbin.org/get"
}
```


Knjižnica *httr*

- ▶ Enostavna knjižnica za uporabo protokola HTTP
- ▶ Opis: <https://github.com/hadley/httr/blob/master/vignettes/quickstart.Rmd>



```
library(httr)
r <- GET("http://httpbin.org/get")
status_code(r)
```

```
## [1] 200
```

- ▶ Glava:

```
headers(r)
```

```
## $`access-control-allow-credentials`
## [1] "true"
##
## $`access-control-allow-origin`
## [1] "*"
##
## $`content-encoding`
## [1] "gzip"
##
## $`content-type`
```

Vsebina

- ▶ Za testiranje komunikacije bomo uporabili strežnik `http://httpbin.org` in vire na njemu
- ▶ Niz v JSONU, ki ga vrne vir `/get` - opis zahtevka

```
content(r, "text")
```

```
## No encoding supplied: defaulting to UTF-8.
```

```
## [1] "{\n  \"args\": {}, \n  \"headers\": {\n    \"Accept\": \"application/json, text/xml, application/\"
```

```
str(content(r))
```

```
## List of 4
```

```
## $ args : Named list()
```

```
## $ headers:List of 4
```

```
## ..$ Accept : chr "application/json, text/xml, application/xml, */*"
```

```
## ..$ Accept-Encoding: chr "gzip, deflate"
```

```
## ..$ Host : chr "httpbin.org"
```

```
## ..$ User-Agent : chr "libcurl/7.54.0 r-curl/3.2 httr/1.3.1"
```

```
## $ origin : chr "188.230.128.78, 188.230.128.78"
```

```
## $ url : chr "https://httpbin.org/get"
```

Obravnava vsebine

- ▶ Branje tekstovne vsebine v kodni tabeli

```
content(r, "text", encoding = "ISO-8859-1")
```

- ▶ Detekcija kodne tabele iz besedila (načeloma bi morala kodna tabela biti napisana v glavi)

```
stringi::stri_enc_detect(content(r, "raw")).
```

- ▶ Obravnava binarnih datotek

```
bin <- content(r, "raw")  
writeBin(bin, "datoteka.txt")
```

Parametri

- ▶ Parametri so mehanizem za pošiljanje parametrov na vir (podobno kot parametri pri funkcijah)
- ▶ Vir na strežniku (lahko) uporabi parametre
- ▶ Parametri so zakodirani v nizih

```
r <- GET("http://httpbin.org/get",  
  query = list(key1 = "value1", key2 = "value2")  
)  
content(r)$args
```

- ▶ Če so v seznamu vrednosti NULL se ignorirajo

“Kukiji” (ang. Cookies)

- ▶ Mehanizem, ki ga uporabljajo brskalniki in strežniki za komunikacijo stanja
- ▶ Slovar vrednosti (ključ=vrednost)
- ▶ Strežnik poda slovar v glavi odgovora pod ključem `Set-Cookie`:
- ▶ Brskalnik si v profilu uporabnika shrani ključe zadnjega odgovora za določen URL strežnika
- ▶ Vsakič ko strežniku pošilja zahtevek, priloži slovar v glavi pod ključem `Cookie`:
- ▶ Vrednosti v slovarju so lahko neberljive (dešifrira jih lahko le strežnik)
 - ▶ številka seje
 - ▶ zašifrirani podatki, ki jih razume samo strežnik
 - ▶ avtentifikacijski žetoni (ang. token)
- ▶ Kukiji imajo lahko rok trajanja (brskalnik pretečene kukije ne pošilja v zahtevkih)

Kukiji in ključi v glavi

- ▶ Primer uporabe na testnem strežniku <http://httpbin.org>

```
r <- GET("http://httpbin.org/cookies/set", query = list(a =  
cookies(r)
```

- ▶ Na viru je funkcija, ki prebere parametre in vrne odgovor, kjer zahteva od brskalnika nastavitev vrednosti v cookie za ta URL strežnika
- ▶ Če večkrat zaženemo zgornjo kodo z različnimi parametri vidimo, da se vrednosti hranijo med zahtevki

Kukiji in ključ v glavi

► Nastavljanje ključev v glavi

```
r <- GET("http://httpbin.org/get",  
        add_headers(Name = "Hadley"))  
str(content(r)$headers)
```

```
## List of 5  
## $ Accept      : chr "application/json, text/xml, application/xml, */*"  
## $ Accept-Encoding: chr "gzip, deflate"  
## $ Host        : chr "httpbin.org"  
## $ Name        : chr "Hadley"  
## $ User-Agent   : chr "libcurl/7.54.0 r-curl/3.2 http/1.3.1"
```

POST

- ▶ ukaz *POST* se tipično uporablja, ko pošljamo vsebino in pričakujemo, da bo vir z njo nekaj naredil
- ▶ Obnašanje je podobno kot pri klicu funkcije
- ▶ Primeri uporabe
 - ▶ forme v HTML
 - ▶ nalaganje datotek
 - ▶ API vmesniki, ki rabijo zahtevnejšo vsebino (XML, JSON)

POST

```
url <- "http://httpbin.org/post"  
body <- list(a = 1, b = 2, c = 3)
```

- ▶ Kodiranje slovarja v obliki HTML forme

```
r <- POST(url, body = body, encode = "form")
```

- ▶ Večdelno sporočilo:

https://en.wikipedia.org/wiki/MIME#Multipart_messages

```
r <- POST(url, body = body, encode = "multipart")
```

- ▶ Parametri v telesu v obliki JSON-a

```
r <- POST(url, body = body, encode = "json")
```

- ▶ Pregled zahtevka

```
POST(url, body = body, encode = "form", verbose())
```

Format XML

- ▶ XML - ang. Extensible Markup Language
- ▶ Podoben kot HTML, le strožje pogoji za sintakso
 - ▶ značke alfanumerične
 - ▶ vse značke se zapirajo (<a> ... ,
)
 - ▶ značke pravilo vgnezdene
 - ▶ med pripadajočima značkama je vsebina (tekst) ali več poddreves XML
 - ▶ samo ena korenska značka v celém dokumentu
 - ▶ parametri v značkah nujno v dvojnih navednicah
 - ▶ nekaj preddefiniranih značk (< = <, > = >, & = &, ' = ', " = ")
 - ▶ UTF-8, uvodna vrstica

Primer

- ▶ <http://www.w3schools.com/xml/note.xml>

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Prednosti formata

- ▶ Posplošeni bralniki (razčlenjevalniki)
 - ▶ DOM Parser - prebere celo drevo, po katerem lahko brskamo
 - ▶ SAX - zaporedni razčlenejevalnik (sprožanje dogodkov)
- ▶ Možnost formalne definicije sintakse
 - ▶ sheme: DTD, XSD shema
 - ▶ avtomatično preverjanje pravilnosti sintakse in gnezdenja značk
- ▶ Enolična struktura
 - ▶ jezik za poizvedbe: XPATH
- ▶ Slabost: predolg, a lahko kompresiramo (zip)
- ▶ Npr. Wordove datoteke .xlsx so zazipane XML datoteke

JSON

- ▶ Izgleda kot struktura iz vgnezenih slovarjev in seznamov v Pythonu
- ▶ Odgovor na XML, ki je tipično predolg
 - ▶ Še bolj ekstremen format: Apache AVRO ()
- ▶ Primer: odgovor vira: <http://httpbin.org/get>

```
{  
  "args": {},  
  "headers": {  
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",  
    "Accept-Encoding": "gzip, deflate",  
    "Accept-Language": "en-US,en;q=0.5",  
    "Cookie": "_ga=GA1.2.273164132.1447798188; a=1; b=4; _gat=1",  
    "Host": "httpbin.org",  
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:41.0) Gecko/20100101 Firefox/41.0"  
  },  
  "origin": "90.157.220.250",  
  "url": "http://httpbin.org/get"  
}
```

XPATH

- ▶ Jezik za poizvedbe po XML dokumentu (DOM drevesu)
- ▶ DOM - Document object model
 - ▶ značke so vozlišča drevesa (node)
 - ▶ parametri so atributi vozlišč
 - ▶ besedila med značkami so tekstovna vozlišča (listi drevesa)

Primer

- ▶ Oglejmo si spletno stran: <http://www.fmf.uni-lj.si/si/iskanje?q=.&cmd=+I%C5%A1%C4%8Di+&m=any&wm=beg>