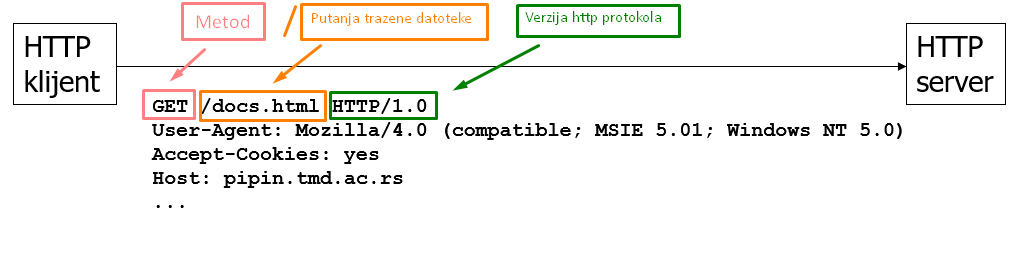
# Teorija iz web programiranja

## HTTP Protokol

Kada god naš browser zatraži neki sadržaj od web servera, on to radi tako što uputi **zahtev** i dobije **odgovor**. A taj zahtev i odgovor su formirani po **HTTP protokolu**.

Komunikacija je zasnovana na **zahtev/odgovor** principu koji su *nezavisni* od ostalih parova zahtev/odgovor.

##### Http zahtev



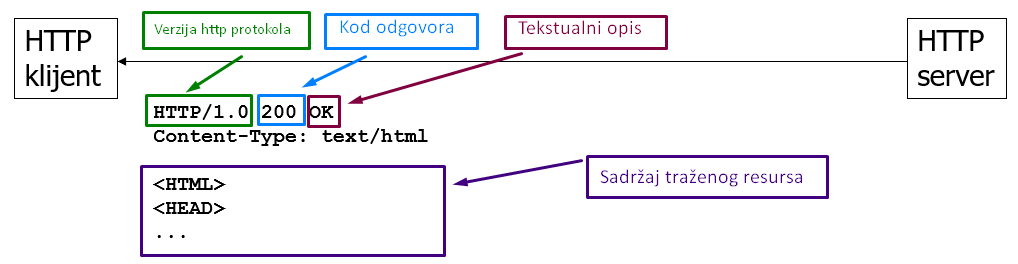
**Slika 1.** Primer http zahteva

**Prvi red** u zahtevu je **obavezan** i formata je **metod** /**putanja\_trazene\_datoteke** **verzija\_http** kao što vidimo na slici 1. Posle prvog reda idu opcioni atributi oblika **ime\_atributa**: **vrednost.** Posle tih opcionih atributa ide prazan red koji označava kraj http zahteva.

##### Http odgovor

Format odgovora u **prvom redu** je **obavezan** i oblika: **verzija\_http** **code** **txt\_opis** a u sledećim redovima idu opcioni atributi istog formata kao i opcioni atributi u http zahtevu (slika 1). Posle opcionih atributa ide prazan red, nakon kog ide **sadržaj traženog resursa**. Taj sadržaj je onakav kakva je ta datoteka, ako je tekstualni, ide txt ako je binarni idu binarni podaci.

**Kodovi** su podeljeni u opsege, **100** – unapredjenje protokola , **200** – uspešnost , **300** – redirekcija, **400** – greška na *frontu*, **500** – greška na *beku*



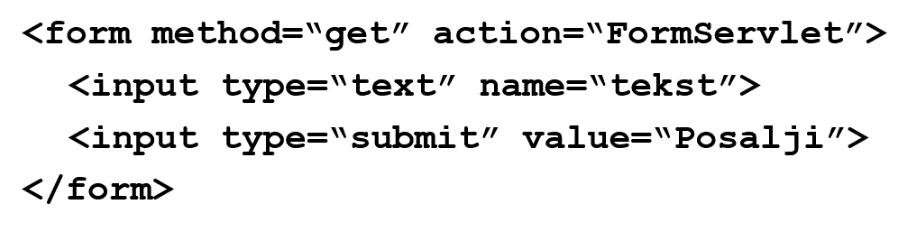
**Slika 2**. Primer http odgovora

**Sadržaj** traženog resursa može biti **statički**(unapred uskladišten) i **dinamički**(generisan po zahtevu).

##### Preuzimanje podataka sa formi

Na slici 3 vidimo primer kako mi preko forme preuzimamo podatke. Te podatke možemo poslati na 2 načina, preko metode **get** i metode **post**. Akcija ( ***action*** parametar forme) predstavlja url koda koji će pokupiti parametre forme i obraditi ih.

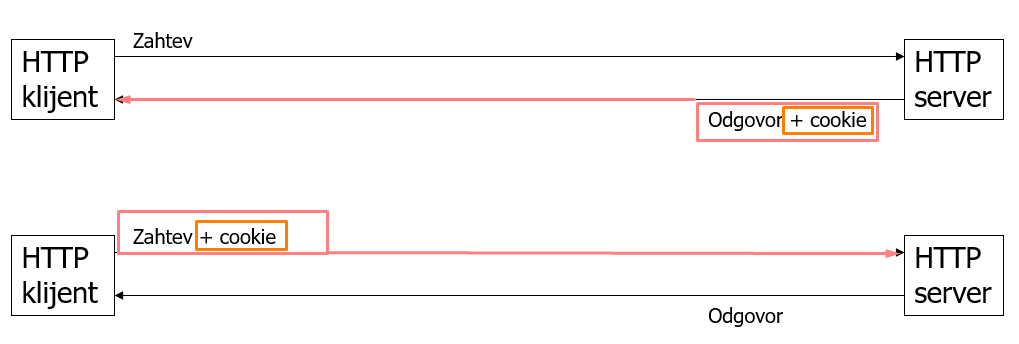
U slučaju metode **GET**, parametri forme se nalaze u ***adresnoj liniji*** i svima su vidljivi što nije bezbedan način. U slučaju metode **POST**, parametri forme se nalaze u ***telu zahteva***, što definitivno predstavlja bezbedniji način za slanje parametara.

**Slika 3**. Primer forme preko koje šaljemo podatke

##### Praćenje sesije korisnika

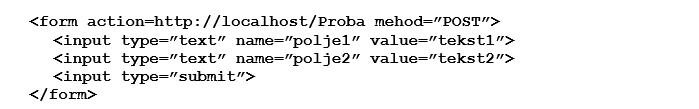
Http protocol ne prati sesiju, pošto se veza klijenta i servera zatvara po isporuci resursa. Alternativa koja rešava taj problem je ***cookie******mehanizam***.

**Cookie mehanizam** funkcioniše tako što *server* **šalje** ***cookie*** klijentu u okviru http **odgovora**, *klijent* **čuva** primljeni ***cookie*** i **šalje** ga uz svaki http **zahtev**.

**Slika 4**. Primer cookie mehanizma

### Pitanja sa ispita

#### HTTP Protokol

**1. Napisati prvi i poslednji red HTTP zahteva (HTTP verzija 1.1) koji se dobija kada se klikne na submit forme (a ništa se ne unese u polja) u sledećoj formi:**

###### Odgovor:

Prvi: **POST** **/http://localhost/Proba** **HTTP/1.1**

Poslednji: polje1=tekst1**&**polje2=tekst2

###### Napomena:

Voditi računa da je u pitanju bio POST zahtev pa su zbog toga parametri forme zapravo činioci poslednjeg reda, što nije u slučaju GET zahteva.

U **POST** metodi postoji atribut **Content length: dužina** ( atribut koji nam govori koliko još **bajtova** treba da **učitamo** **nakon praznog reda**), dok kod **GET** metode kraj označava **prazan red**.

**2. Šta je to “Permanent connection” u HTTP protokolu verzije 1.1?**

Klijent od servera zahteva se konekcija ne zatvara odmah po slanju odgovora. U zaglavlju http zahteva i odgovora se stavlja atribut **Connection = Keep-Alive**

**3. Navesti nazive atributa u HTTP zahtevu i HTTP odgovoru, a koji omogućava praćenje sesije u HTTP protokolu.**

###### Odgovor:

Pošto ne postoji način za praćenje sesije, uvodi se mehanizam zvani ***cookie* mehanizam**. Kada se prvi put šalje zahtev na server, nemamo atribut vezan za praćenje sesije. Nakon prvog zahteva, u odgovoru servera se nalazi atribut **Set-Cookie: ime=vrednost** . Pošto je klijent sada dobio svoj cookie, u sledećem zahtevu on ima atribut **cookie: ime=vrednost** koji ga jedinstveno identifikuje. Na taj način je omogućeno praćenje sesije u http protokolu.

**4. Čime se označava kraj HTTP zahteva u GET metodi?**

###### Odgovor:

Praznim redom.

###### Napomena:

Ovo **ne važi** i za **POST** metod, za njega je kraj zahteva određen atributom **Content length**( atribut koji nam govori koliko još **bajtova** treba da **učitamo** **nakon praznog reda**) !

**5. Navesti osnovnu razliku između GET i POST zahteva.**

###### Odgovor:

U slučaju GET metode, podaci forme se nalaze u adresnoj liniji, dok se kod POST metode podaci forme nalaze u telu zahteva.

**GET /FormServlet?tekst=asdf HTTP/1.1**

**...**

**POST /FormServlet HTTP/1.1**

**...**

**Content-length: 10**

**...**

**tekst=asdf**

**6. Navesti ključne elemente HTTP odgovora koji se koriste kod redirekcije. Dati primer takvog HTTp odgovora.**

###### Odgovor:

Redirekcija se svodi na slanje poruke: **302 Object moved** i postavljanja atributa **location: nova\_adresa** u http odgovoru.

Primer:

**HTTP/1.1 302 Object moved**

**Location: neki\_novi\_url**

**7. Ukratko napiši sadržaj HTTP zahteva.**

###### Odgovor:

**METOD /putanja HTTP/verzija**

Dodatni redovi koji sadrže attribute u obliku: **ime\_atributa: vrednost**

**Prazan** **red** na kraju

**8. Ukratko napiši sadržaj HTTP odgovora.**

###### Odgovor:

**HTTP/verzija** **code** **txt\_opis**

Dodatni redovi koji sadrže attribute u obliku: **ime\_atributa: vrednost**

**Prazan** **red**

**Sadržaj**

**9. Napisati prvi red HTTP zahteva i odgovora.**

###### Odgovor:

Zahtev: **METOD /putanja HTTP/verzija**

Odgovor: **HTTP/verzija** **code** **txt\_opis**

**10. Napisati prvi red HTTP odgovora u HTTP protokolu verzije 1.0 za kod 404.**

###### Odgovor:

**HTTP/1.0** **404** **Not found**

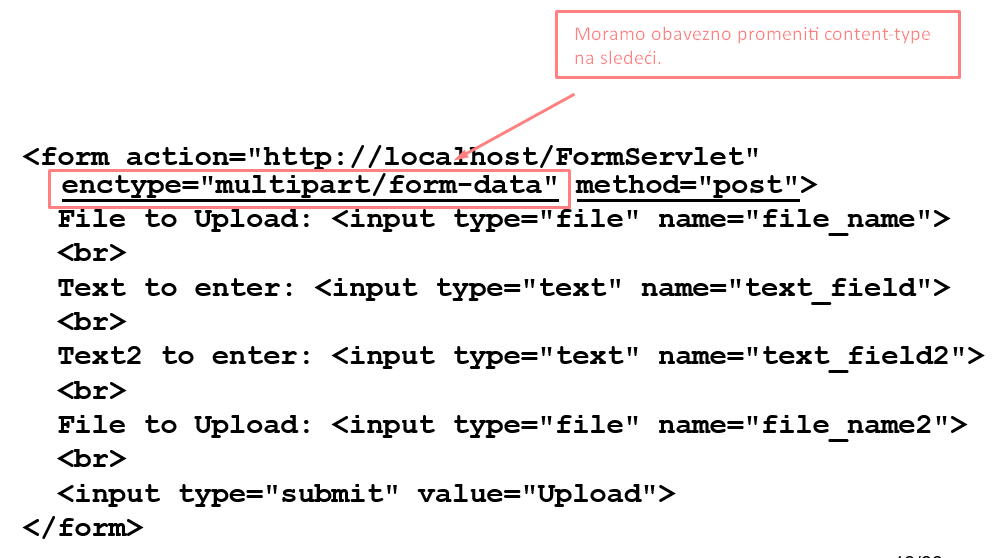
**11. Navesti metode HTTP zahteva i šta radi koji.**

###### Odgovor:

**GET** – **zahteva** resurs od web servera, **POST** - **šalje** parametre forme i traži odgovor, **HEAD** – **zahteva** samo HTTP odgovor, bez slanja samog resursa, **PUT** – omogućava klijentu da **pošalje** **datoteku** na web server, **OPTIONS** – od web servera se **traži** spisak **metoda** koje podržava, **DELETE** – omogućava klijentu da **obriše** resurs sa web servera.

**12. Kako izgleda HTTP zahtev u kom se prosleđuje fajl?**

###### Uvod:

****Običan POST zahtev ima atribut **Content-Type: application/x-www-form-urlencoded** ali kada je u pitanju POST zahtev koji prosleđuje fajl priča se komplikuje.

**Slika 1.** Primer HTML koda na klijentu pri slanju datoteke

Prvo u formi na frontu moramo imati podešen **enctype** na **multipart/form-data.** Tek kada smo enctype podesili, onda u **input** atributima možemo koristiti **type=”file”.** Zbog ovoga imamo totalno drugačiji donji deo HTTP zahteva ( onaj posle praznog reda, tj tamo gde dolaze parametri forme ).

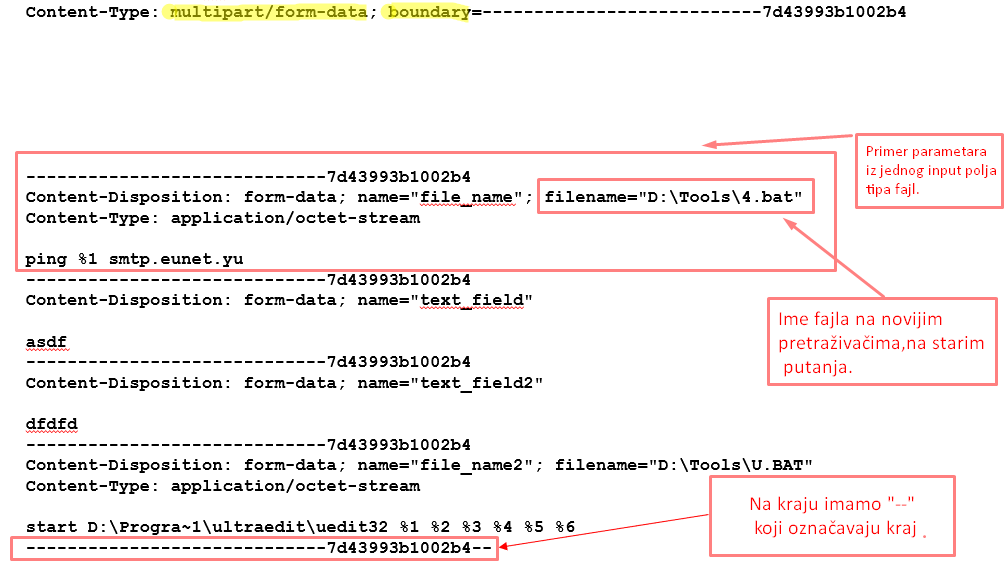
###### Odgovor:

U odnosu na klasični POST zahtev **Content-Type** atribut dobija i nastavak zvani **boundary** koji služi za definisanje granice između pojedinih elemenata forme.

Primer:

**Content-Type: multipart/form-data; boundary=---------------------------7d43993b1002b4**

Sada se menja i format parametara posle praznog reda. Na slici 2 vidimo da kada se prosleđuje fajl, imamo “***novo zaglavlje***” koje ima attribute **Content-Disposition** koji dobija još dodatni deo **filename** kada je input polje bilo tipa file. A takođe ima i **Content-Type** atribut u koliko je input polje bilo tipa fajl, to polje nam govori tačan tip fajla. Nakon toga ide prazan red i bajtovi fajla.

**Slika 2.** Primer HTTP zahteva u kom se prosleđuje fajl