HTTP Request/Response model

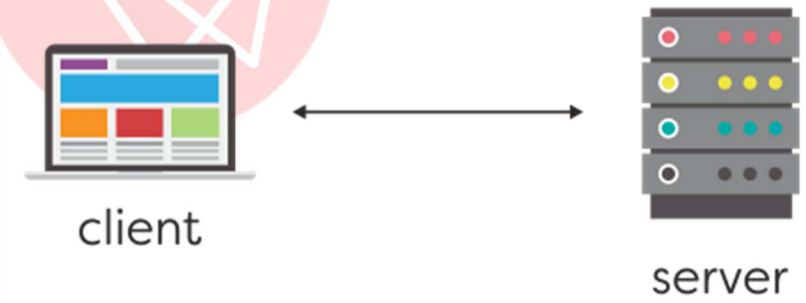
Osnovna namena frontend programiranja jeste razvijanje prezentacionih delova web sajtova i aplikacija. Kaže se prezentacionih, zato što je u pitanju deo web sajtova koji se prikazuje unutar web pregledača i sa kojim korisnici vrše direktnu interakciju. Ipak, potpuno je jasno da pojam prezentacionog dela za sobom povlači i postojanje određene logike koja je u pozadini, skrivena od očiju korisnika, i sa kojom korisnik ne može direktno da komunicira. Naravno, sasvim je moguće napraviti sajt koji uopšte ne poseduje pozadinsku logiku (*backend*), ali bi se u takvim situacijama govorilo o veoma jednostavnim sajtovima, koji pored statičkih informacija korisnicima ne bi mogli da pruže mnogo. Na primer, dovoljno je u priču uključiti kontakt formu i postojanje određene backend logike postaje neminovnost. U takvoj situaciji, podaci forme iz korisničkog web pregledača upućuju se backend logici na serveru, koja zatim obavlja njihovu dalju obradu (transformisanje, analiziranje, arhiviranje, slanje obaveštenja...). Može se rezimirati da se izvršavanje programske logike modernih web sajtova i aplikacija obavlja na dva različita nivoa – unutar web pregledača, ali i na serveru.

Postojanje dve nezavisne komponente koje čine jednu celinu za sobom povlači neminovnost njihove komunikacije. Komunikacija se ogleda u razmeni poruka, odnosno podataka, poštovanjem protokola za komunikaciju. U modulu koji je pred vama biće obrađene najznačajnije tehnike za postizanje komunikacije između frontend i backend logike web sajtova. Drugim rečima, biće prikazani različiti pristupi za slanje podataka i prijem podataka od servera.

Za početak, biće izneta neka osnovna načela komunikacije između klijenata i servera na webu.

Klijent–server model

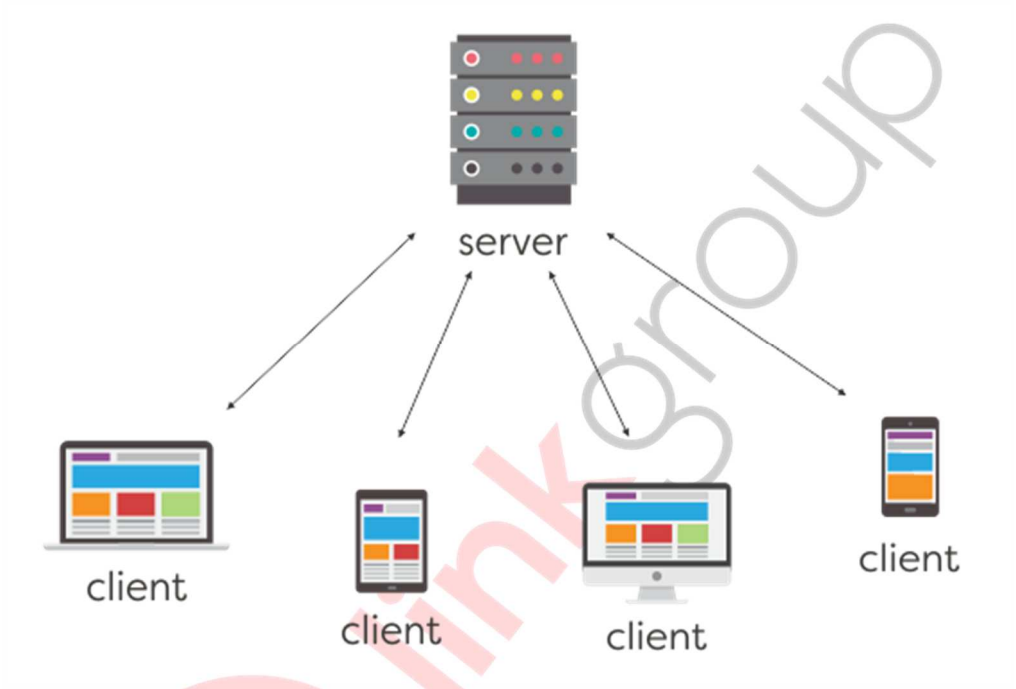
Web sajtovi i aplikacije funkcionišu na principima takozvanog klijent–server modela. Reč je o modelu koji definiše strukturu distribuiranih aplikacija, odnosno aplikacija čiji se programski kod izvršava na većem broju različitih kompjutera. Klijentski kompjuter je onaj sa koga se korišćenjem nekog web pregledača pregleda neki web sajt. Server je kompjuter na kome se web sajt fizički nalazi i koji klijentu isporučuje podatke na osnovu kojih pregledač formira prikaz sajta.



Slika 7.1 Klijent–server komunikacija

Slika 7.1. ilustruje uprošćenu komunikaciju između klijenta i servera na webu. Sliku možete sagledati kroz svakodnevno korišćenje weba. Na primer, prilikom otvaranja nekog web sajta korišćenjem web pregledača (npr. Chromea), vaš kompjuter se ponaša kao klijent, a kompjuter na kome se nalazi sajt kome pokušavate da pristupite kao server.

Bitno je razumeti da se web sajtovi tipično nalaze na samo jednom serverskom računaru. Sa druge strane, broj klijenata koji komuniciraju sa takvim serverom uglavnom je mnogo veći. Stoga se može zaključiti da klijent–server model na webu uglavnom podrazumeva jedan serverski kompjuter i veliki broj različitih klijentskih uređaja (slika 7.2).



Slika 7.2. Klijent–server komunikacija (2)

HTTP(S)

Na slikama 7.1. i 7.2, linijama koje povezuju server i klijente dočarana je njihova međusobna komunikacija. Ipak, ono što još nije rečeno jeste da se komunikacija između klijenata i servera na webu obavlja poštovanjem jednog posebnog protokola. Reč je o protokolu HTTP.

Funkcionisanje weba u potpunosti je zasnovano na načelima HTTP, odnosno HTTPS protokola. Da je to tako – verujemo da ste i sami već primetili, koristeći neki web pregledač. Jednostavno, prilikom posete nekom web sajtu, vidljivo je da adresa sajta započinje sa `http` ili `https`:

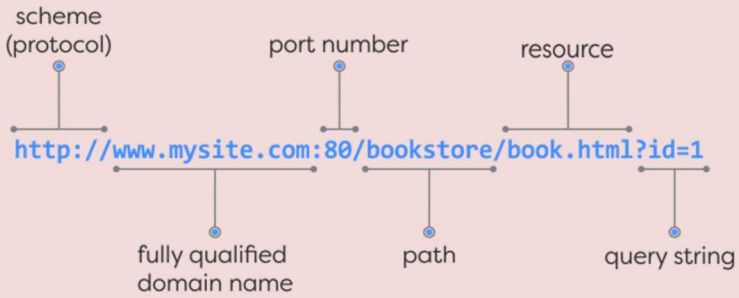
```
https://www.google.com/
```

Možete da primetite da adresa jednog od najpopularnijih web sajtova današnjice započinje odrednicom `https`, što govori da se za komunikaciju koristi HTTP, odnosno HTTPS protokol.

Adrese web sajtova i resursa na webu

Svi resursi na webu su jednoznačno određeni specijalnim adresama. Takve adrese omogućavaju razlikovanje resursa, a korisnicima obezbeđuju sistem za pristup. Na primer, upravo prikazani tekst `https://www.google.com` jeste primer jedne adrese web resursa.

Adresa jednog web resursa se drugačija naziva *Uniform Resource Locator* ili skraćeno **URL**. Prosto rečeno, URL je ono što kucate u web pregledaču kada pokušavate da pristupite nekom sajtu. Po svojoj strukturi, URL je hijerarhijska sekvenca komponenata (slika 7.3).



Slika 7.3. Struktura URL adrese

Slika 7.3. ilustruje različite delove od kojih može biti sačinjen jedan URL:

- **scheme** – naziv protokola koji se koristi za komunikaciju
- **fully qualified domain name (FQDN)** – definiše preciznu lokaciju kompjutera (hosta) unutar mreže
- **port number** – oznaka koja definiše određeni proces ili servis koji se izvršava na kompjuteru povezanom na mrežu
- **path** – putanja do određenog resursa na serverskom računaru
- **resource** – naziv resursa koji se od servera zahteva
- **query string** – opcioni parametri koji se u formi parova ključeva i vrednosti mogu proslediti serveru

URL adrese se mogu pojaviti u dva oblika i to kao:

- relativne
- apsolutne

URL adresa prikazana na slici 7.3 je apsolutna. **Apsolutne adrese** su kompletne, potpune adrese, koje sadrže sve relevantne informacije kako bi se moglo doći do nekog resursa.

Relativne URL adrese su znatno kraće od apsolutnih, ali unutar sebe ne poseduju sve delove koji mogu jednoznačno odrediti jedan resurs na webu. Zbog nedostatka određenih elemenata, relativne URL adrese se uvek odnose na lokaciju dokumenta unutar koga se nalaze. Primer jedne relativne adrese može da izgleda ovako:

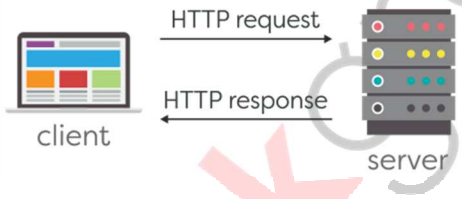
```
bookstore/book.html
```

Samo na osnovu upravo prikazane relativne URL adrese ne može se znati tačna lokacija `book.html` fajla. Zapravo, sve što na ovaj način možemo da vidimo jeste da se fajl `book.html` nalazi unutar foldera `bookstore`. Stoga se ovakav tip adrese oslanja na lokaciju dokumenta u kojem je naveden. Ukoliko se dokument sa ovakvom URL adresom nalazi na adresi `http://www.mysite.com`, onda će kompletna putanja na koju će upućivati prikazana relativna adresa biti:

```
http://www.mysite.com/bookstore/book.html
```

Prilikom kreiranja web sajtova, najčešća praksa jeste korišćenje relativnih adresa za povezivanja resursa koji se nalaze na istom serveru na kojem se nalazi i web sajt. Na taj način se prevazilazi problem koji može nastati usled promene domenskog imena preko koga je sajt dostupan. Za povezivanje svih eksternih resursa, koriste se apsolutne adrese.

HTTP je takozvani *request–response* protokol, što znači da njegov način funkcionisanja počiva na međusobnom smenjivanju zahteva i odgovora između dva kompjutera – servera i klijenta (slika 7.4).



Slika 7.4. Razmena zahteva i odgovora poštovanjem pravila HTTP-a

Iz priloženog se može zaključiti da su dva osnovna obrasca komunikacije između servera i klijenata na webu poštovanjem pravila HTTP protokola:

- HTTP zahtev (*HTTP request*) i
- HTTP odgovor (*HTTP response*).

HTTP zahtev

Osnovni obrazac komunikacije kojim se klijenti obraćaju HTTP serverima jeste HTTP zahtev. Najjednostavniji primer HTTP zahteva za jednom web stranicom može da izgleda ovako:

```
GET /index.html HTTP/1.1
host: www.mysite.com
```

Prikazani zahtev jeste ono što web pregledač upućuje nekom serveru, kada u polju adrese sajta upišete: *www.mysite.com.* On se sastoji iz nekoliko delova:

- **GET** – naziv HTTP metode
- **/index.html** – resurs koji se zahteva
- **HTTP/1.1** – verzija protokola
- **host: www.mysite.com** – naziv hosta

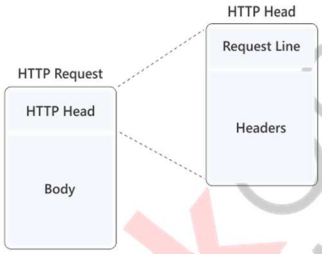
Prikazani HTTP zahtev predstavlja najosnovniji oblik jednog takvog zahteva. U stvarnosti se uglavnom koriste mnogo kompleksniji zahtevi, koji poseduju znatno više delova. Ipak, HTTP zahtev ne može imati proizvoljnu formu, već je prilikom njegovog formiranja potrebno pridržavati se nekih osnovnih pravila koja se tiču strukture HTTP zahteva.

Struktura HTTP zahteva

HTTP zahtev, koji klijent upućuje serveru, sastoji se iz dva osnovna dela i nekoliko poddelova:

- **glava** (*head*)
    - početna linija ili **linija zahteva** (*start line, request line*)
    - **zaglavlja** (*headers*)
- **telo** (*body*)

Ukoliko bismo strukturu HTTP zahteva prikazali grafički, ona bi izgledala kao na slici 7.5.



Slika 7.5. Struktura HTTP zahteva

Zahtev koji je prikazan nešto ranije bio je sačinjen iz linije zahteva i jednog zaglavlja. Pored zaglavlja koje govori kom hostu je potrebno uputiti zahtev, HTTP zahtev može sadržati i razna druga zaglavlja koja bliže određuju jedan zahtev. Tako primer jednog HTTP zahteva može da izgleda kao na slici 7.6.

Slika 7.6. Primer jednog HTTP zahteva

Slika 7.6. ilustruje primer jednog tipičnog HTTP zahteva. Zahtev započinje linijom zahteva (*request line*). Linija zahteva započinje navođenjem HTTP metode kojom se bliže određuje namena HTTP zahteva koji se upućuje serveru. Nakon linije zahteva, navedeno je nekoliko zaglavlja koja bliže određuju sam zahtev. Telo zahteva je prazno.

HTTP metode

Za kreiranje HTTP zahteva, klijentima je na raspolaganju nekoliko različitih obrazaca komunikacije, koji se drugačije nazivaju HTTP metode. Takve metode govore serveru koja je osnovna namera zahteva.

Neke od najznačajnijih HTTP metoda su:

- **GET** – zahteva podatke određenog resursa
- **HEAD** – zahteva samo HTTP zaglavlje, bez tela odgovora
- **POST** – prosleđuje nov resurs serveru
- **PUT** – prosleđuje serveru resurs kojim je potrebno zameniti već postojeći resurs
- **DELETE** – briše resurs sa servera
- **OPTIONS** – zahteva od servera do isporuči skup HTTP metoda koje podržava
- **TRACE** – koristi se za testiranje i debug, tako što server vraća klijentu zahtev koji je primio
- **PATCH** – vrši parcijalnu promenu postojećeg resursa

Pojam HTTP metoda možda nije najlakše razumeti, pogotovu ako se uzme u obzir da je reč o pojmovima koji nisu direktno vidljivi korisnicima. Zbog toga je za početnike teško da razumeju njihovu namenu. Stvarnost je takva da se prilikom jednostavnog surfovanja webom u pozadini najviše koriste zahtevi formirani GET metodom. Na primer, kada pokušavate da pročitate članak na nekom blogu, pregledač će u pozadini formirati HTTP zahtev korišćenjem GET metode.

U svakodnevnoj upotrebi na webu, često se koristi i POST metoda HTTP protokola. Iz pregleda HTTP metoda možete da pročitate da je reč o metodi kojom se serveru prosleđuje nov resurs. Na primer, kada kreirate nalog na nekoj društvenoj mreži, web pregledač će u pozadini iskoristiti upravo POST metodu, kako bi podatke vašeg novog naloga prosledio serveru. POST je i jedna od HTTP metoda čijim se korišćenjem upošljava i nešto ranije spomenuto telo HTTP zahteva. Naime, upotrebom POST metode, podaci se serveru prosleđuju upravo korišćenjem tela HTTP zahteva (slika 7.7).

Slika 7.7. Primer HTTP zahteva upućenog POST metodom

Slika 7.7. dočarava primer jednog HTTP zahteva upućenog POST metodom. Ovoga puta telo zahteva nije prazno, već se unutar njega nalaze podaci koji se prosleđuju serveru.

I preostale HTTP metode poseduju značajnu primenu na webu. Ipak, za početak ćemo pričati samo o GET i POST metodama.

Iz upravo prikazanih primera GET i POST HTTP zahteva može se videti da se telo zahteva može koristiti za smeštanje podataka koji se upućuju serveru. Ipak, u zavisnosti od namene zahteva, podaci unutar tela mogu, ali i ne moraju biti prisutni. Tako se GET zahtevi upućuju bez podataka unutar tela, za razliku od POST HTTP zahteva, kada se podaci smeštaju unutar tela.

HTTP odgovor

Po prijemu zahteva od klijenta, HTTP server vrši njegovu obradu i formira odgovor. Primer jednog jednostavnog HTTP odgovora može da izgleda ovako:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 500

<html>
<body>

      some server response...

</body>
</html>
```

Prilikom formiranja HTTP odgovora, baš kao što je to bio slučaj i prilikom formiranja zahteva, postoje određena pravila, unapred definisana HTTP protokolom. Drugim rečima, može se reći da i HTTP odgovor poseduje određenu strukturu koja je veoma slična HTTP zahtevu.

Struktura HTTP odgovora

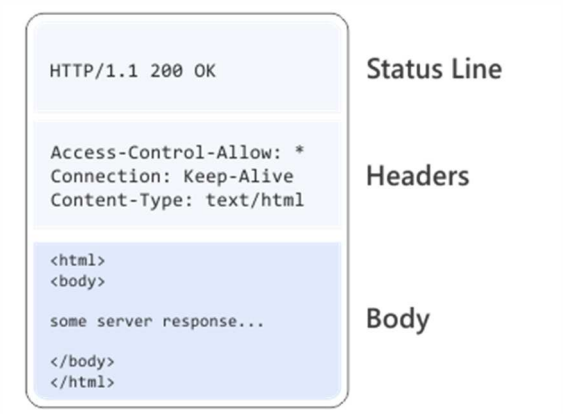
Struktura HTTP odgovora gotovo je identična strukturi HTTP zahteva. Čine je:

- **glava** (*head*)
    - početna linija ili **statusna linija** (*start line, status line*)
    - **zaglavlja** (*headers*)
- **telo** (*body*)

Osnovna i najznačajnija razlika između strukture HTTP zahteva i odgovora tiče se početne linije, koja se kod HTTP odgovora drugačije naziva statusna linija. Ona se sastoji od:

- verzije protokola (npr. HTTP/1.1)
- statusnog koda
- statusnog teksta

Primer jednostavnog HTTP odgovora, prikazanog iznad, vizuelno se može razložiti na celine kao na slici 7.8.



Slika 7.8. Primer HTTP odgovora

Na slici 7.8. se može videti da serverski odgovor započinje statusnom linijom koja sadrži informaciju o uspešnosti obrade zahteva koji je uputio klijent. Ovo je jedna od najznačajnijih informacija koje server isporučuje klijentu, naravno pored opcionih podataka unutar tela.

HTTP statusni kodovi

Za označavanje uspešnosti obrade klijentskog zahteva, HTTP protokol poznaje veliki broj statusnih kodova, koji su podeljeni u nekoliko grupa. U prikazanom primeru, server unutar statusne linije navodi kod 200, što znači da je klijentski zahtev uspešno obrađen. Statusni kod jeste osnovna informacija na osnovu koje klijent može znati kako je protekla obrada zahteva upućenog serveru. Različite grupe statusnih odgovora, koje se često nazivaju i statusne klase, jesu:

- **1xx** – informacije
- **2xx** – uspešno izvršavanje serverske logike pri obradi zahteva
- **3xx** – preusmeravanja (*redirections*)
- **4xx** – klijentske greške
- **5xx** – serverske greške

Neki od najznačajnijih konkretnih HTTP statusnih kodova su:

- **200 (OK)** – obrada zahteva je uspešna i server je isporučio odgovor
- **201 (Created)** – obrada zahteva je uspešna i pri tome je kreiran novi resurs
- **301 (Moved Permanently)** – resurs koji se zahteva je premešten na novu adresu
- **302 (Found)** – resurs koji se zahteva je privremeno premešten na drugu adresu
- **304 (Not Modified)** – resurs se nije promenio od prethodnog pristupa; koristi se prilikom keširanja resursa, kako bi klijent mogao da proveri da li postoji ažurnija verzija resursa
- **400 (Bad Request)** – server nije razumeo zahtev
- **401 (Unauthorized)** – pristup resursu zahteva autorizaciju
- **403 (Forbidden)** – pristup resursu je zabranjen
- **404 (Not Found)** – server ne može da pronađe traženi resurs
- **500 Internal Server Error** – greška na serveru prilikom obrade zahteva

Pitanje

Koji HTTP statusni kod se koristi da označi uspešno obrađen zahtev?

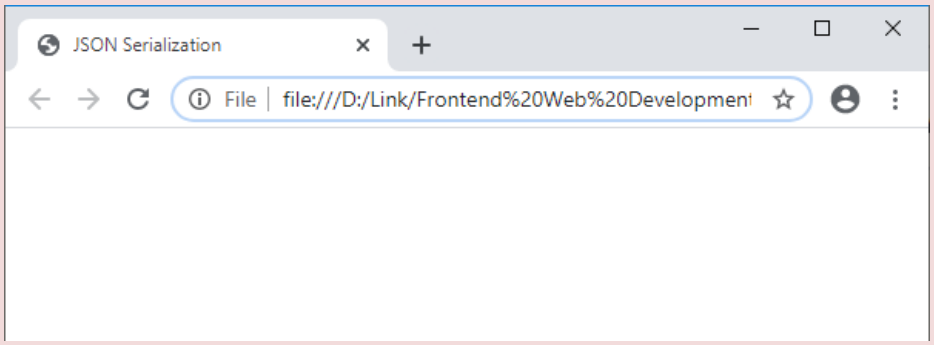
a) 404
b) 201
c) 200
d) 300

Objašnjenje:

Statusni kod 200 koji HTTP server isporučuje klijentu označava da je obrada zahteva bila uspešna i da je server isporučio odgovor.

Instalacija lokalnog HTTP servera

Dosadašnji tok lekcije razjasnio je pojmove klijenta i servera i HTTP protokola koji se koristi za njihovu međusobnu komunikaciju. Mogli ste da pročitate da najosnovniji način na koji se jedan web pregledač može *naterati* da napravi HTTP zahtev jeste otvaranje nekog sajta. Ipak, za prikaz svih HTML dokumenata u dosadašnjem toku ovoga kursa, web pregledači uopšte i nisu koristili HTTP protokol. Razlog je vrlo jednostavan – kako bi se uputio HTTP zahtev, neophodno je da fajl koji se zahteva bude smešten na nekom HTTP serveru.



Slika 7.9. Pregled HTML dokumenta koji se nalazi na fajl sistemu korisnika

Slika 7.9 ilustruje pregled jednog od HTML dokumenata iz ovog kursa unutar web pregledača. Bitno je da primetite da adresa iz adresne trake uopšte ne započinje odrednicom `http://` ili `https://` i da je sa leve strane adrese jasno ispisano *File*. S obzirom na to da se unutar web pregledača pregleda fajl koji se nalazi direktno na klijentskom fajl sistemu, ovakvo ponašanje je potpuno očekivano. Web pregledač za čitanje sadržaja fajla koristi file, a ne HTTP protokol.

Kako bi se uposlio HTTP protokol, o kome je bilo reči u dosadašnjem toku lekcije, HTML dokument je potrebno smestiti na HTTP server. Odlična stvar je ta što bilo koji kompjuter ujedno može biti i klijent i server. Postojanje web pregledača čini ga HTTP klijentom, a kako bi postao i server, neophodno je obaviti instalaciju programa koji predstavlja lokalni HTTP server.

Danas postoji nekoliko HTTP servera koje je moguće veoma lako instalirati. Mi ćemo se odlučiti za Apache HTTP Server koji se može dobiti u okviru paketa XAMPP. XAMPP je moguće preuzeti sa sledeće adrese:

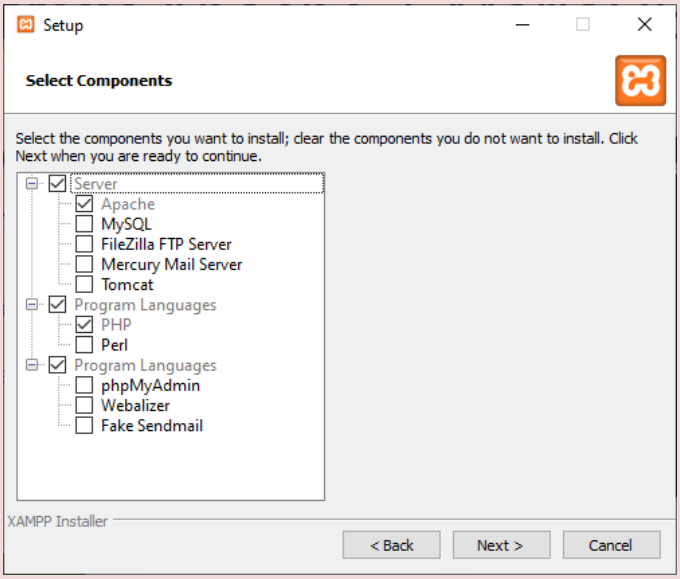
https://www.apachefriends.org/index.html

XAMPP je dostupan u tri različite verzije, odnosno za Windows, Linux i macOS operativne sisteme (slika 7.10).

Slika 7.10. Zvanični web sajt XAMPP-a i ujedno lokacija sa koje je moguće preuzeti instalacioni fajl

Sa zvaničnog XAMPP web sajta potrebno je da obavite preuzimanje instalacionog fajla odgovarajuće verzije, u zavisnosti od operativnog sistema koji koristite. Preuzeti fajl je potrebno da pokrenete i da pratite instalacione korake.

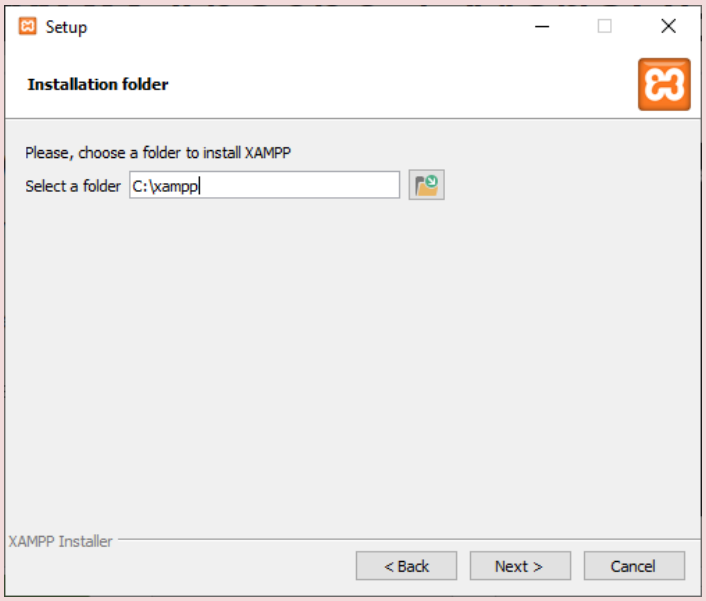
Tokom instalacije je potrebno obratiti pažnju na dva ključne tačke. Prva je odabir komponenata koje će biti instalirane (slika 7.11).



Slika 7.11. Odabir XAMPP komponenata koje će biti instalirane

Nama je sasvim dovoljan XAMPP sa osnovnim skupom funkcionalnosti. Takav skup podrazumeva Apache HTTP server sa podrškom za PHP jezik. Iako nama PHP nije potreban, on se, kao što možete videti sa slike 7.11, ne može isključiti iz skupa minimalnih funkcionalnosti. Sve ostalo je moguće isključiti i nije potrebno za praćenje ovog kursa.

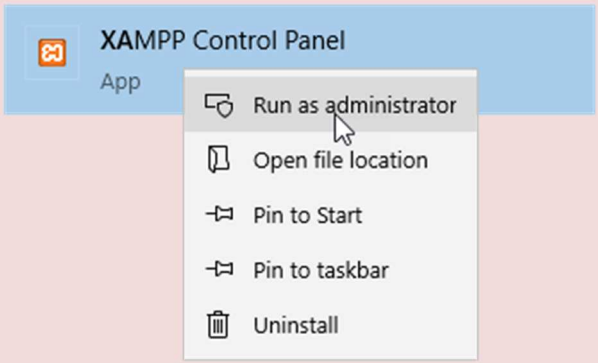
Sledeći bitan detalj XAMPP instalacije tiče se odabira lokacije na kojoj će se naći njegovi fajlovi (slika 7.12).



Slika 7.12. Podrazumevana XAMPP instalaciona putanja

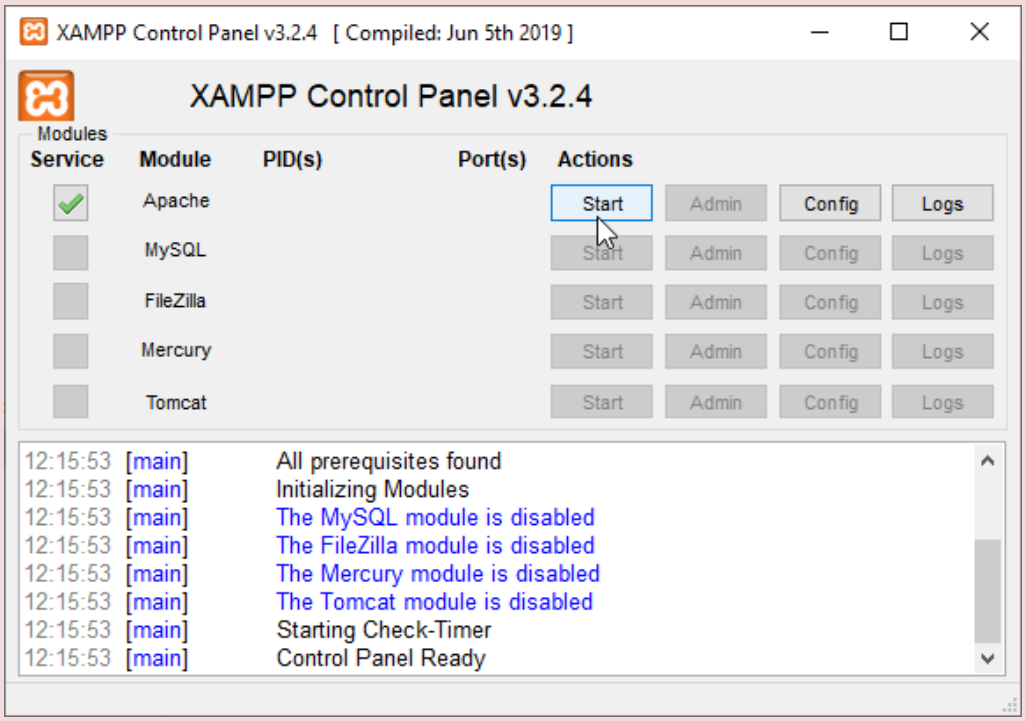
XAMPP se podrazumevano instalira na putanji `C:\xampp` i savet je da se ova putanja ne menja.

Nakon instalacije, dobija se aplikacija XAMPP Control Panel, koja se može koristiti za pokretanje i konfiguraciju pojedinačnih servera. Preporuka je da se XAMPP uvek pokreće sa administratorskim privilegijama (slika 7.13).



Slika 7.13. Pokretanje XAMPP-a sa admin privilegijama na Windows operativnom sistemu

Kada pokrenete XAMPP Control Panel, dočekaće vas prozor kao na slici 7.14.



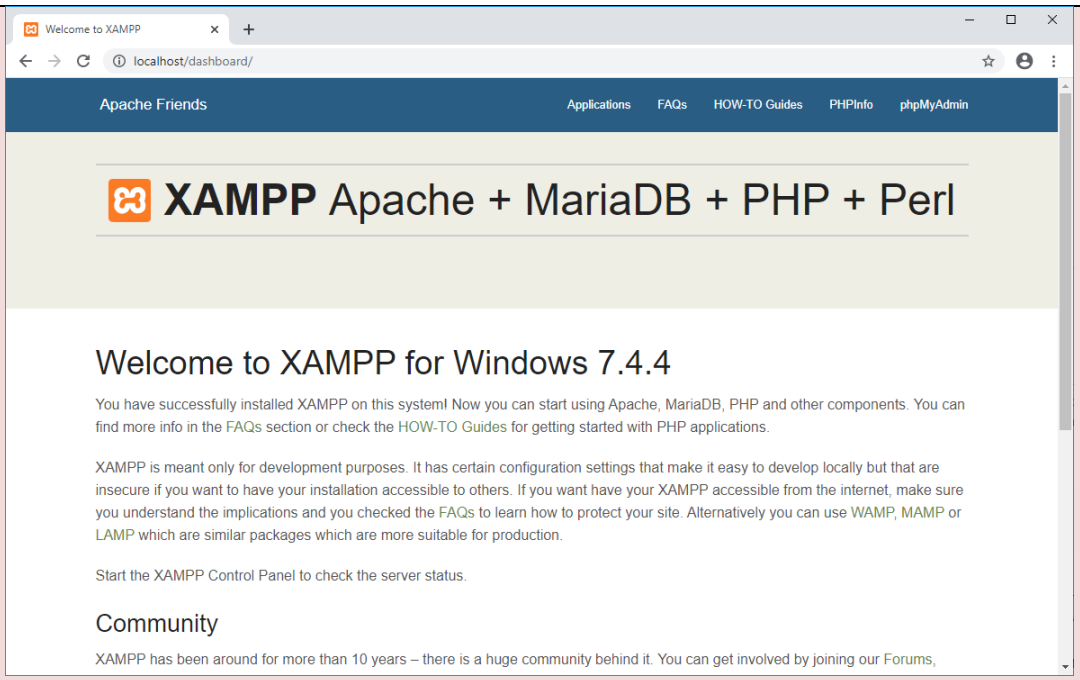
Slika 7.14. XAMPP Control Panel

XAMPP Control Panel omogućava pojedinačno pokretanje i konfigurisanje svih instaliranih servera. Nama je potreban Apache server; stoga je za njegovo pokretanje dovoljno kliknuti na dugme *Start*. Uspešno pokretanje Apache HTTP servera propraćeno je odgovarajućom vizuelnom reprezentacijom – natpis *Apache* dobija zelenu pozadinu, u koloni Port(s) se ispisuje broj porta, a u polju sa porukama – *[Apache] Status change detected: running*.

Obavljanjem koraka koji su opisani u prethodnim redovima, vaš kompjuter postaje HTTP server. Takav server je moguće uposliti slanjem HTTP zahteva na adresu lokalnog servera:

- IP adresa lokalnog servera je 127.0.0.1
- naziv hosta lokalnog servera je localhost

Ovo praktično znači da je dovoljno da otvorite neki web pregledač i u polje za adresu uneste ili `127.0.0.1` ili `localhost` (slika 7.15).



Slika 7.15. Podrazumevana početna stranica XAMPP Apache HTTP servera

Ono što dobijate navigacijom na `localhost` adresu jeste parsirani sadržaj fajla koji se nalazi na serveru. Reč je o podrazumevanoj stranici koja se na lokalni server automatski postavlja prilikom instalacije.

Folder čiji sadržaj predstavlja ono što se nalazi na lokalnom serveru podrazumevano se nalazi na putanji (naravno, pod uslovom da niste menjali instalacionu putanju XAMPP-a):

```
C:\xampp\htdocs
```

Unutar foldera na ovoj putanji sada je moguće smeštati različite fajlove kojima želimo da pristupimo korišćenjem HTTP protokola. Na primer, ukoliko unutar prikazanog foldera smestite folder `my-site` sa fajlom `index.html`, njemu ćete upošljavanjem lokalnog HTTP servera moći da pristupite kada u web pregledaču otkucate:

```
http://localhost/my-site/index.html
```

ili

```
http://localhost/my-site
```

Savet je da sve primere koji budu prezentovani u nastavku ovoga kursa smeštate na lokalni HTTP server i da ih pregledate upravo upošljavanjem takvog servera, navođenjem `localhost` naziva hosta.

Na koji način se mogu razmenjivati podaci između frontenda i backenda?

Ukoliko se pitate zbog čega je dosadašnji tok lekcije bio posvećen HTTP protokolu, odnosno HTTP zahtevima i odgovorima – razlog je vrlo jednostavan. Reč je o osnovnom mehanizmu koji omogućava razmenu podataka između klijenata i servera na webu. S obzirom na to da je osnovni komunikacioni protokol na webu HTTP, podaci koji se razmenjuju između klijenata i servera mogu se naći ili unutar zaglavlja ili unutar tela HTTP zahteva, odnosno HTTP odgovora. Stoga je vrlo bitno razumeti način na koji funkcioniše *request–response* model komunikacije na webu.

Za upućivanje HTTP zahteva, pa samim tim i za slanje određenih podataka serveru, frontend programer na raspolaganju ima nekoliko pristupa:

- query string
- forme
- AJAX

U nastavku ove lekcije biće ilustrovani najjednostavniji pristupi za upućivanje podataka serveru, koji ne zahtevaju korišćenje JavaScript jezika. Naredne lekcije baviće se pojmom AJAX, odnosno korišćenjem kombinacije JavaScript jezika i ugrađenih funkcionalnosti web pregledača za postizanje komunikacije se serverom.

Prosleđivanje podataka serveru korišćenjem Query stringa

Nešto ranije u ovoj lekciji definisan je pojam URL-a. Reč je, dakle, o skraćenici za *Uniform Resource Locator*, što je zapravo adresa jednog web resursa. Drugim rečima, URL je ono što kucate u web pregledaču kada pokušavate da pristupite nekom sajtu.

URL se može sastojati iz brojnih delova, koji omogućavaju da se precizno definiše lokacija na kojoj se nalazi traženi resurs. Ipak, pored takvih delova, URL omogućava da se serveru proslede i određeni podaci u formi parova ključeva i vrednosti (slika 7.16).

Slika 7.16. Query string deo URL adrese

Slika 7.16. ilustruje strukturu URL adrese sa posebnim akcentom na njen poslednji segment – query string. Query string je deo URL adrese koji omogućava da se određene vrednosti pridruže parametrima i na taj način postanu sastavni deo zahteva koji klijent upućuje serveru. Podaci definisani query stringom na taj način postaju dostupni HTML serveru, odnosno serverskoj logici koja obrađuje zahtev koji je klijent uputio.

Query string se unutar linka navodi nakon karaktera upitnik (?). Između ključeva i vrednosti u paru postavlja se karakter jednako (=), dok se različiti parovi ključeva i vrednosti međusobno razdvajaju karakterom ampersend (&) ili tačkom sa zapetom (;).

Slika 7.17. Primer URL adrese sa query stringom

Unutar URLa prikazanog slikom 7.17, query string je:

```
id=1&mode=user
```

Na ovaj način, serveru se zajedno sa HTTP zahtevom upućuju i dva para ključeva i vrednosti, odnosno dva parametra i dve vrednosti. Ovakvi parametri naći će se unutar glave HTTP zahteva.

Query string je moguće formulisati prilikom pisanja bilo kog URL-a – direktno unutar polja za adresu web pregledača ili prilikom formulisanja vrednosti `href` atributa linkova:

```
<a href="get-data.php?id=1&mode=user">Get user data</a>
```

Ovo je sada primer jednog linka, odnosno a elementa, čiji `href` atribut poseduje vrednost sa URL-om koji poseduje query string. Query stringom se, klikom na link, HTTP serveru prosleđuju dva para parametara i njihove vrednosti.

Prosleđivanje podataka serveru korišćenjem query stringa moguće je postići uvek, odnosno bez obzira na HTTP metodu koja se koristi za formulisanje zahteva. Prikazani primeri su proizvodili GET HTTP zahteve, ali s obzirom na to da je putanja sastani deo svakog HTTP zahteva, query string podatke je moguće definisati i kada se zahtev upućuje korišćenjem POST, PUT, DELETE i ostalih HTTP metoda.

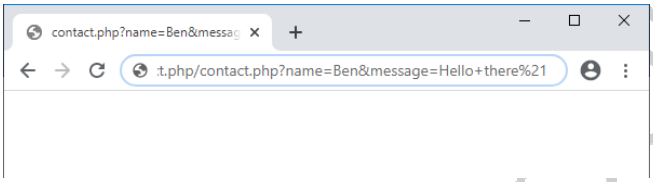
Prosleđivanje podataka serveru korišćenjem HTML formi

Još jedan od izvornih načina za komunikaciju klijenta i servera na webu jesu web forme. Prosleđivanjem HTML forme, njeni podaci se šalju serveru, i to na URL adresu koja je definisana kao vrednost `action` atributa:

```
        <form action="contact.php">
            <label for="name">Name: </label>
            <input type="text" name="name" id="name">
            <label for="message">Message: </label>
            <textarea name="message" id="message" cols="30"
rows="10"></textarea>
            <input type="submit" value="Submit">
        </form>
```

Kod ilustruje jednostavnu HTML formu. Forma poseduje dva elementa, pomoću kojih korisnik može da unese svoje ime i poruku. Forma se prosleđuje klikom na `input` element tipa `submit` i tada se podaci forme prosleđuju na adresu definisanu vrednošću atributa `action`. Stoga se podaci upravo prikazane forme prosleđuju na adresu `contact.php`. S obzirom na to da je reč o relativnoj URL adresi, podaci ovakve forme biće prosleđeni fajlu `contact.php`, koji se na serveru nalazi na identičnoj putanji kao i fajl sa ovakvom formom. Na primer, ukoliko se ovakva forma nalazi na HTML stranici koja se dobija adresom `http://www.mysite.com`, ovakva forma će biti prosleđena na URL adresu: `http://www.mysite.com/contact.php`.

Kada se forma koja se dobija ovakvim kodom popuni i prosledi klikom na *submit* dugme, ukoliko pogledate unutar trake sa adresom u web pregledaču, moći ćete da vidite zanimljiv efekat (slika 7.18).



Slika 7.18. Podaci forme podrazumevano se prosleđuju kroz URL

Slika 7.18 ilustruje efekat koji se dobija nakon prosleđivanja prikazane forme, pri čemu je u polje za ime uneto `Ben`, a u polje za poruku `Hello there!`. Možete da vidite da se podrazumevano, podaci forme serveru prosleđuju upravo korišćenjem query string-a, kao parovi ključeva i vrednosti, pri čemu su nazivi ključeva vrednosti `name` atributa koji su definisani na `input` elementima forme.

Na način prosleđivanja podataka formi, moguće je uticati definisanjem atributa `method`, koji može imati vrednosti:

- `get`
- `post`

Prosleđivanje podataka forme GET HTTP metodom

`get` je podrazumevana vrednost i u takvoj situaciji prosleđivanjem forme serveru se upućuje jedan GET zahtev, dok se podaci forme smeštaju unutar URL-a, baš kao u prikazanom primeru (slika 7.19).

```
GET /contact.php?name=Ben&message=Hello+there%21
HTTP/1.1

Host: www.mysite.com
Cache-Control:no-cache
Connection:Kepp-Alive

Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

Slika 7.19. Uprošćena struktura HTTP zahteva prilikom prosleđivanja kontakt forme korišćenjem GET metode

URL encoding

U upravo prikazanom primeru prosleđivanja forme korišćenjem GET HTTP metode, mogli ste da vidite da se unutar URL-a određeni parametri ne prosleđuju u obliku koji smo definisali popunjavanjem polja forme. Razlog je vrlo jednostavan – određeni karakteri se ne mogu naći unutar URL-a.

Određeni karakteri unutar URL adresa imaju specijalno značenje, pa se ne mogu naći u svom izvornom obliku. Na primer, unutar URL-a se u izvornom obliku ne može naći karakter kosa crta (/), zato što je on jedan od osnovnih karaktera koji se koriste za izgradnju URL adresa. Dalje, unutar URL adrese ne može se naći ni karakter razmak (space), karakter hash (#) ni još mnogi drugi. Kako bi se ovakvi specijalni karakteri ipak mogli navesti unutar URL adresa, pribegava se URL enkodingu, koji se često drugačije naziva i *percent-encoding*.

Percent-encoding zasniva se na prikazu rezervisanih karaktera korišćenjem vrednosti koje započinju karakterom procenat (%). Otuda i naziv percent-encoding. Nakon karaktera procenat navode se dva karaktera koji predstavljaju ASCII vrednost karaktera u heksadecimalnom formatu.

Rezervisani URL karakter	Percent-encoding
razmak	%20
!	%21
#	%23
$	%24
%	%25
&	%26
'	%27
(	%28
)	%29

Tabela 7.1. Percent-encoding rezervisanih URL karaktera

Pored korišćenja koda `%20`, za predstavljanje razmaka može se koristiti i karakter `+`.

Unutar URL-a mogu se direktno naći sva velika i mala slova, svi brojevi, ali i karakteri: tilda (~), srednja crta (-), tačka (.) i donja crta (_). Svi ostali karakteri se moraju definisati u upravo prikazanom obliku, korišćenjem karaktera procenat i ASCII heksadecimalnog koda.

Na osnovu priče o *percent-encodingu*, možete razumeti zbog čega su podaci iz forme unutar URL-a predstavljani u obliku koji je prikazan na slici 7.18 – razmak je zamenjen karakterom plus (+), a karakter uzvičnik kodom `%21`.

Prosleđivanje podataka forme POST HTTP metodom

Kada se za vrednost atributa `method` postavi `post` (ili `POST`), prosleđivanje forme proizvodi POST HTTP zahtev, a tada se podaci smeštaju u telo takvog zahteva:

```
        <form action="contact.php" method="POST">
            <label for="name">Name:</label>
            <input type="text" name="name" id="name">
            <label for="message">Message:</label>
            <textarea name="message" id="message" cols="30"
rows="10"></textarea>
            <input type="submit" value="Submit">
        </form>
```

Sada je na `form` elementu postavljen atribut `method` sa vrednošću `POST`. Zbog toga će prosleđivanje ove forme rezultovati HTTP POST zahtevom, a podaci će biti prosleđeni u njegovom telu (slika 7.20).

Slika 7.20. Uprošćena struktura HTTP zahteva prilikom prosleđivanja kontakt forme korišćenjem POST metode

Kada se podaci forme prosleđuju korišćenjem POST HTTP metode, oblik u kome će se oni naći unutar tela HTTP zahteva definisani su vrednošću `enctype` atributa koji je moguće postaviti na `form` element:

- `application/x-www-form-urlencoded` – podrazumevana vrednost; podaci se definišu u identičnom obliku kao i prilikom smeštanja unutar URL-a; drugim rečima, oni se navode kao parovi ključeva i vrednosti, pri čemu se primenjuje nešto ranije opisani principi percent-encodinga;
- `multipart/form-data` – vrednost koja se koristi kada se formom prosleđuju fajlovi;
- `text/plain` – vrednost kojom se definiše da će podaci unutar tela HTTP zahteva biti definisani u izvornom obliku; uglavnom se koristi samo u procesu testiranja i otklanjanja grešaka.

Rezime

- Izvršavanje programske logike modernih web sajtova i aplikacija obavlja se na dva različita nivoa – unutar web pregledača, ali i na serveru.
- Web sajtovi i aplikacije funkcionišu na principima takozvanog klijent–server modela.
- Klijentski kompjuter je onaj sa koga se korišćenjem nekog web pregledača pregleda neki web sajt.
- Server je kompjuter na kome se web sajt fizički nalazi i koji klijentu isporučuje podatke na osnovu kojih web pregledač formira prikaz web sajta.
- Funkcionisanje weba, u potpunosti je zasnovano na načelima HTTP, odnosno HTTPS protokola.
- Adresa jednog web resursa se drugačija naziva Uniform Resource Locator ili skraćeno URL.
- HTTP je takozvani *request–response* protokol, što znači da njegov način funkcionisanja počiva na međusobnom smenjivanju zahteva i odgovora između servera i klijenata.
- Osnovni obrazac komunikacije kojim se klijenti obraćaju HTTP serverima jeste HTTP zahtev.
- Po prijemu zahteva od klijenta, HTTP server vrši njegovu obradu i formira HTTP odgovor.
- HTTP zahtevi i odgovori sastoje se iz zaglavlja i tela.
- Za kreiranje HTTP zahteva, klijentima je na raspolaganju nekoliko različitih obrazaca komunikacije, koji se drugačije nazivaju HTTP metode.
- IP adresa lokalnog servera je 127.0.0.1 ili localhost.
- Tri osnovna načina na koje klijenti i server mogu da razmenjuju podatke su: URL query string, HTML forme i AJAX.
- Query string omogućava da se serveru proslede određeni podaci u formi parova ključeva i vrednosti kroz URL adresu.
- Prosleđivanjem HTML forme, njeni podaci se šalju serveru, i to na URL adresu koja je definisana kao vrednost `action` atributa.