**TCP/IP applikacioni nivo**

**HTTP**

*Hypertext Transfer Protocol* (*HTTP*) je protokol koji se koristi za pristup podacima na *World Wide Web* . Protokol prenosi podatke u formi punog teksta, *hypertext*-a, audio, video, i td.Ipak, Ovaj protokol se naziva *hypertext* transfer protokol jer njegova efikasnost obezbedjuje njegovo koriš**ć**enje u *hypertext* okruženju gde postoje nagli i brzi prelazi (skokovi) sa jednog dokumenta na drugi.

HTTP funkcioniše kao kombinacija FTP-a i SMPT-a. HTTP je sli**č**an FTP -u jer prenosi fajlove i koristi servise TCP-a. No, HTTP protokol je znatno jednostavniji od FTP-a jer koristi samo jednu TCP konekciju. Ne postoji posebna kontrolna (upravlja**č**ka) konekcija, a izmedju klijenta i servera se prenose samo podaci.

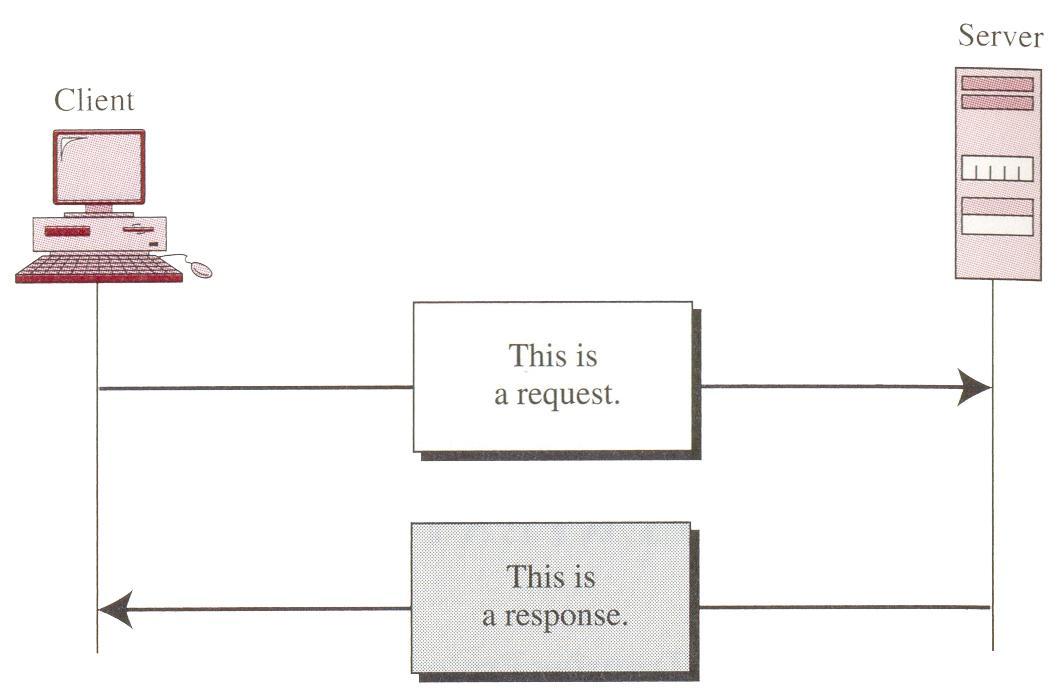
HTTP je sli**č**an SMPT-u iz razloga što preneti podaci izmedju klijenta i servera izgledaju kao SMPT poruke. Treba naglasiti da se HTTP razlikuje od SMPT-a po tome kako se poruke predaju od klijenta ka serveru, tako i kako se prenose od servera ka klijentu. Nasuprot SMPT-u, HTTP poruke nisu namenjene za **č**itanje od strane ljudi, one se **č**itaju i interpretiraju od strane HTTP servera i HTTP klijenta (*browser*-a). SMTP poruke se memorišu i prosledjuju dalje, dok se HTTP poruke isporu**č**uju neposredno.

Ideja HTTP-a je veoma jednostavna. Klijent šalje zahtev, koji izgleda kao e-mail poruka, i upu**ć**uje ga serveru. Server predaje odziv, koji izgleda kao e-mail odziv, i upu**ć**uje ga klijentu. Poruke tipa zahtev i odziv sadrže u sebi podatke u formi pisma formata tipa MIME (*Multi-purpose Internet Mail Extension*-dopuna SMTP-a koja omogu**ć**ava da se preko SMTP-a šaljui ne-ASCII kodirane podatke).

Komande od klijenta ka serveru se ugradjuju u poruke tipa zahtev. Sadržaji zahtevanih fajlova ili druge informacije se ugradjuju u odzivne poruke.

**HTTP transakcije**

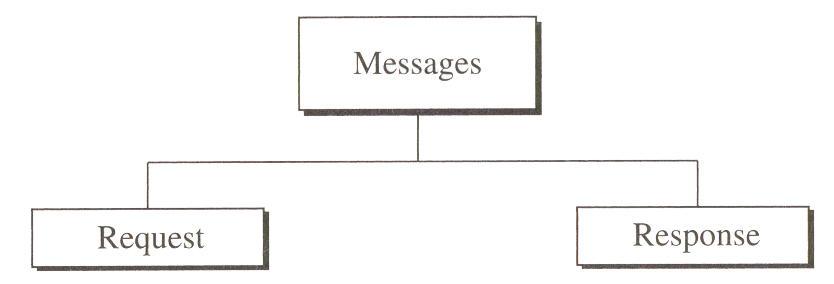
Na slici 1 prikazana je HTTP transakcija izmedju klijenta i servera. Klijent inicijalizira transakciju slanjem poruku tipa zahtev. Server se odaziva slanjem odgovora.



Slika 1 HTTP transakcija

**Poruke**

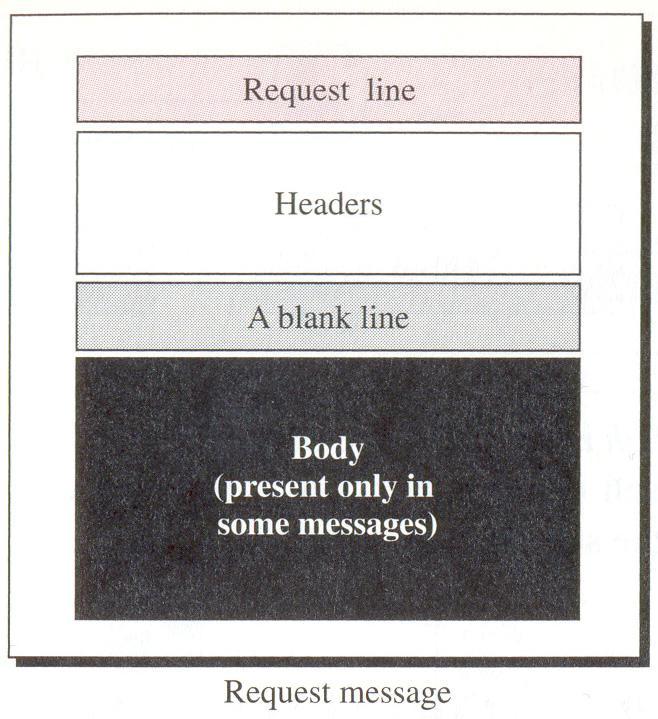
Kao što se vidi sa slike 2 postoje dva osnovna tipa HTTP poruka: zahtev i odziv. Oba tipa poruke imaju skoro isti format.



Slika 2 Kategorije poruka

**Poruke tipa zahtev**

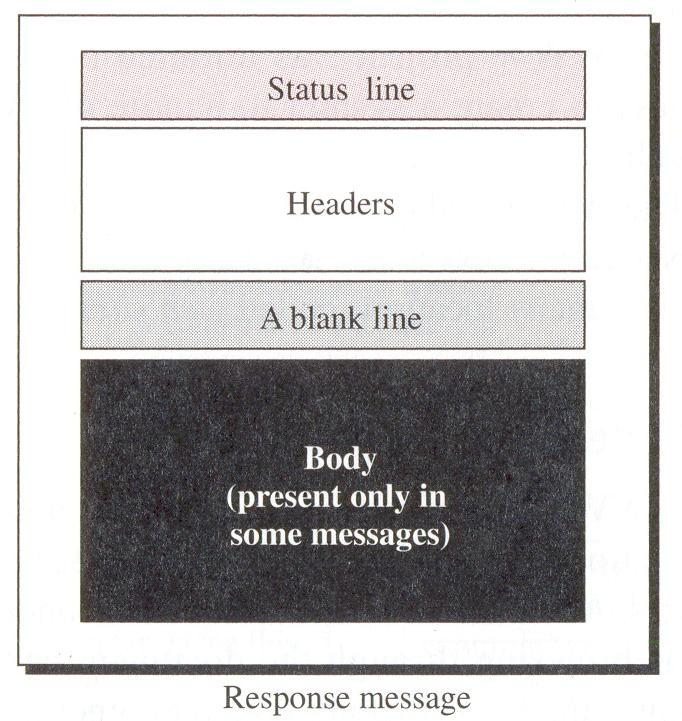
Poruke tipa zahtev **č**ini linija zahtev, zaglavlja, i ponekad telo (vidi sliku 3)



Slika 3 Poruke tipa zahtev

**Poruke tipa odziv**

Poruke tipa odziv **č**ini linija status, zaglavlja, i ponekad telo (vidi sliku 4)

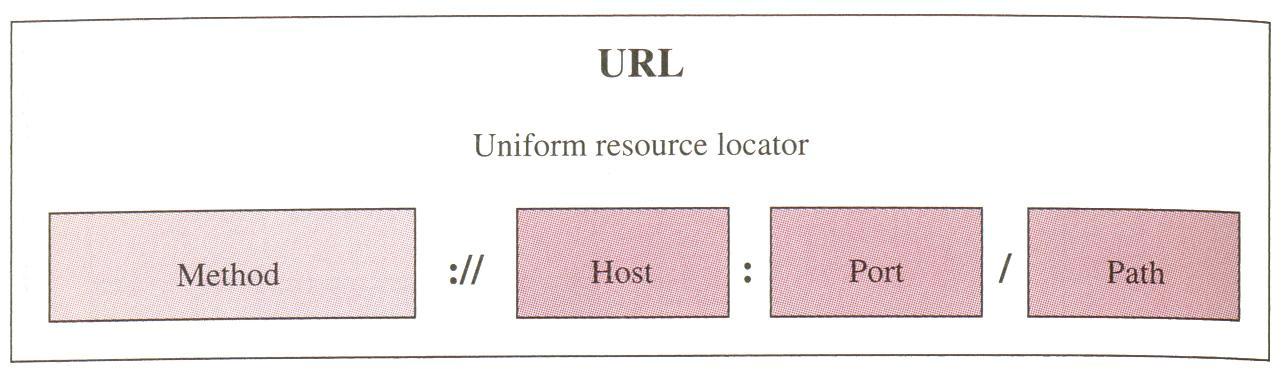


Slika 4 Poruka tipa odziv

**Uniformni lokator resursa (URL)**

Klijentu koji želi da pristupi dokumentu potrebna je adresa. Da bi olakšao pristup dokumentima koji su distribuirani po celom svetu , HTTP koristi koncept lokatora. Uniformni lokator resursa (Uniform Resource Locator- URL) predstavlja standard za specifikaciju bilo

kog tipa informacije na Internet-u. URL definiše **č**etiri stvari: metod, host ra**č**unar, port, i put (vidi sliku 5).



Slika 5 URL

*Metod* je protokol za izbavljanje dokumenata, na primer HTTP.

*Host* je ra**č**unar kod koga je locirana informacija, mada ime ra**č**unara može da bude u alijazi.Web stranice su obi**č**no zapam**ć**ene u ra**č**unarima, a ra**č**unarima su data *alias* imena koja obi**č**no po**č**inju sa karakterima *www*. Ovo nije obavezno, ipak, kao host može da bude bilo koje ime dato ra**č**unaru koje se odnosi na tu web stranicu.

URL može opciono da sadrži *broj porta* servera. Ako je port uklju**č**en, on treba da se ubaci izmedju hosta i puta, i treba da bude izdvojen simbolom **״** : **״**.

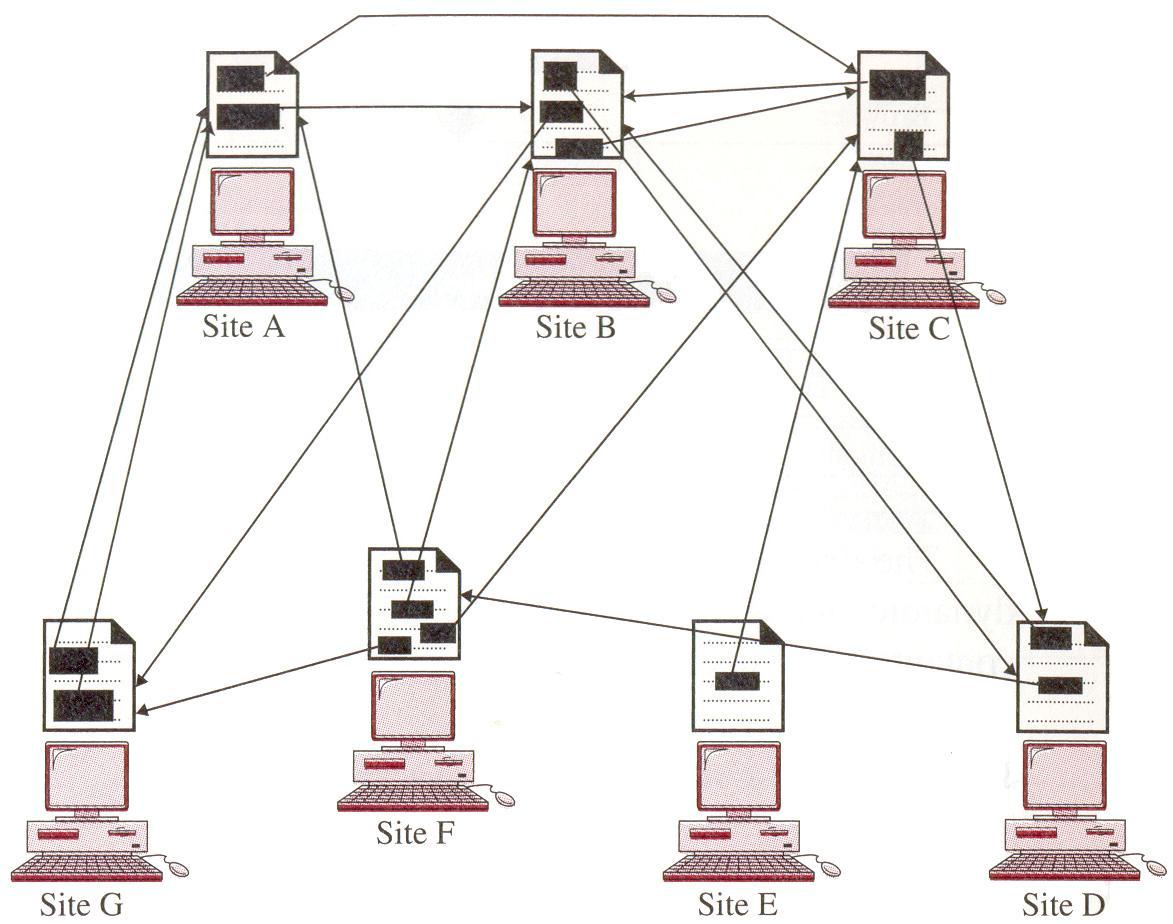
*Put* se odnosi na ime-putanje fajla, a ukazuje gde je ta informacija locirana. Naglasimo da putmože u sebi da sadrži kose crte **״** / **״** koje kod Unix operativnog sistema izdvajaju direktorijume od subdirektorijuma i fajlova.

**World Wide Web (www)**

*World Wide Web*, ili web, predstavlja skladište informacije rasejane po celom svetu, alizajedni**č**ki povezane u jednu celinu. WWW je jedistven po kombinaciji koju karakterišu osobine fleksibilnosti, prenosivosti, i *user-friendly* (bliskost sa korisnikom). Ove osobine **č**ine da www, u odnosu na druge servise koji se nude od strane Internet-a, bude po mnogo **č**emu razli**č**itit i specifi**č**an.

WWW projekat je bio iniciran od strane CERN-a (*European Laboratory for Particle Physics*) sa ciljem da se kreira sistem koji **ć**e za potrebe nau**č**nih istraživanja manipulisati sa distribuiranim resursima.

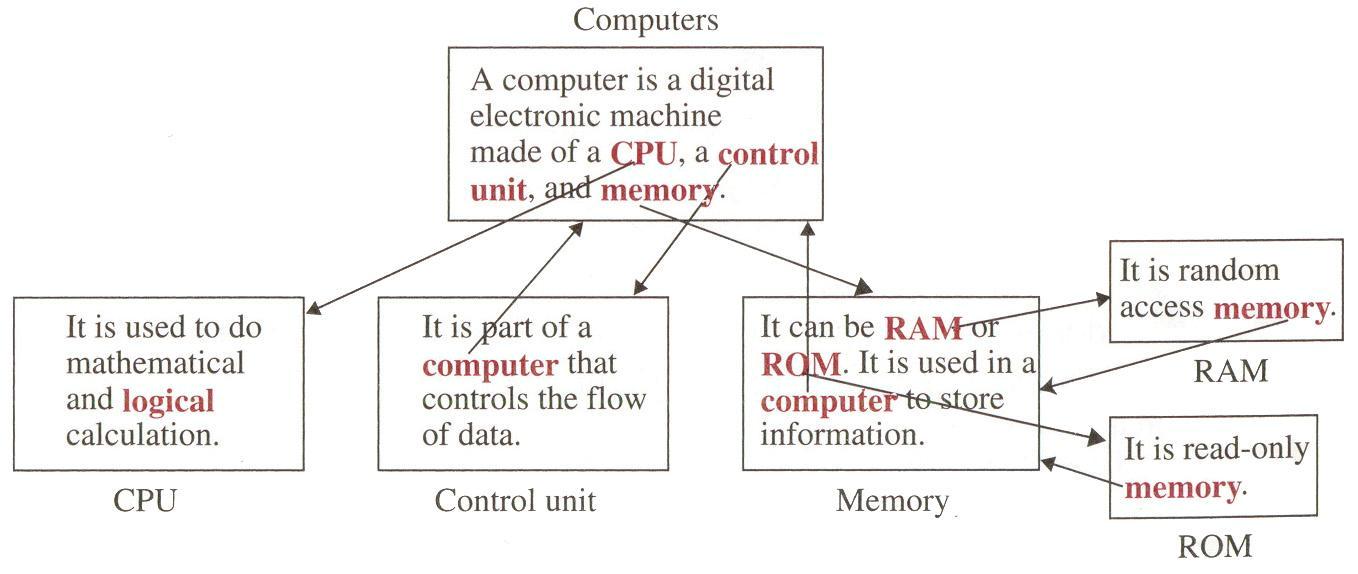
Danas, WWW je distribuirani klijent-server servis, kod koga klijent koriste**ć**i *browser* može da pristupi servisu koriste**ć**i server. Ipak treba naglasiti da je dostupni servis distribuiran na ve**ć**i broj lokacija koje se nazivaju *web sites* (vidi sliku 6).



Slika 6 Distribuirani servisi

**Hypertext i Hypermedia**

WWW koristi koncept *hypertext*-a i *hypermedia*. Kod *hypertext* okruženja, informacija je zapam**ć**ena u skupu dokumenata koji su zajedni**č**ki povezani koriste**ć**i koncept pokaziva**ć**a. Neka stavka se može pridružiti drugom dokumentu koriste**ć**i pokaziva**ć** (*pointer*). **Č**italac koji pretražuje po dokumentu može da predje na drugi dokument izborom (klikom) na stavke koje su povezane sa drugim dokumentima. Na slici 7 prikazan je koncept *hypertext*-a. Dok *hypertext* dokumenti sadrže samo tekst, *hypermedia* dokumenti mogu da sadrže slike, grafike,i zvu**č**ne tonove.



Slika 7 Koncept hypertext-a

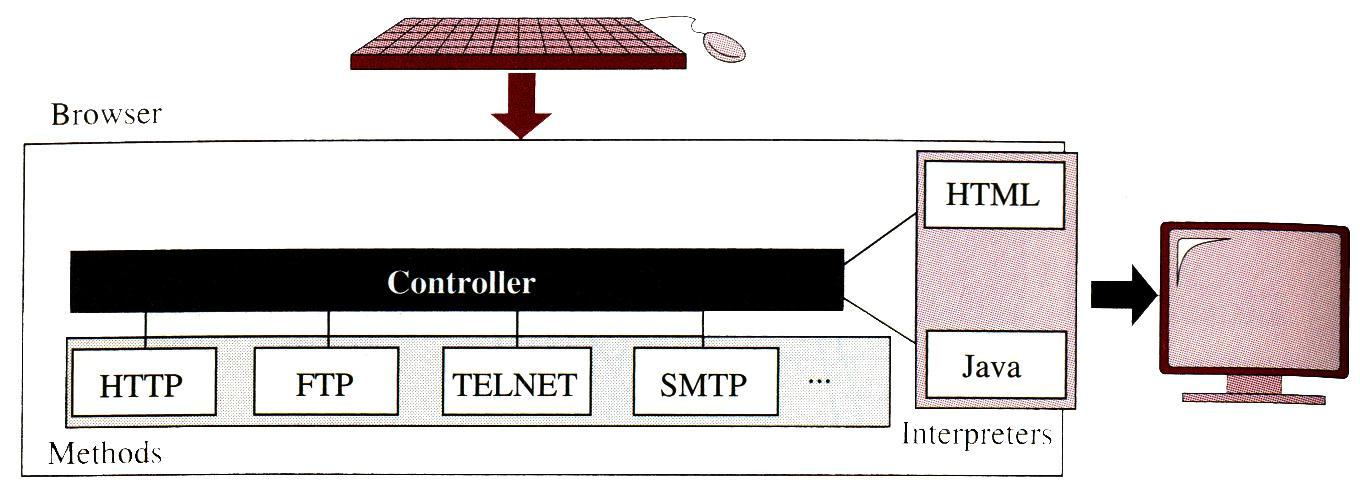
Jedinica *hypertext*-a ili *hypermedia* koja je dostupna na Web-u naziva se stranica (*page*). Glavna stranica organizacije ili individue zove se *home-page*.

**Arhitektura pretraživaća (browser-a)**

Ve**ć**i broj proizvodja**ć**a nudi komercijalne pretraživa**ć**e koje interpretiraju i prikazuju web dokumente. Obi**č**no svi pretraživa**č**i koriste skoro istu arhitekturu. Svaki browser standardno **č**ine slede**ć**e tri celine: kontroler, klijent programi, i interpreteri.

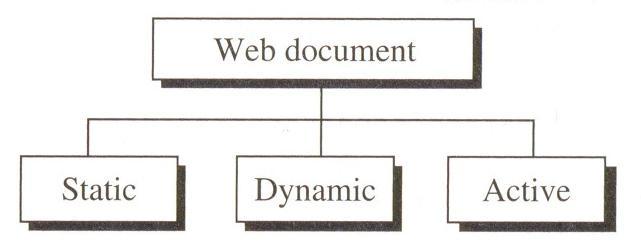
Kontroler prihvata ulaze sa tastature ili miša i koristi klijent programe da bi pristupio dokumentu. Nakon što se pristupilo dokumentu, kontroler koristi jedan od interpretera da bi

na ekranu prikazao dokumenat. Klijent programi mogu da prestavljaju jedan od metoda (protokola) koje smo prethodno analizirali, kakvi su HTTP, FTP, ili TELNET. Interpreter može biti HTML ali Java, sve u zavisnosti od tipa dokumenta (vidi sliku 8).



Slika 8 Browser arhitektura

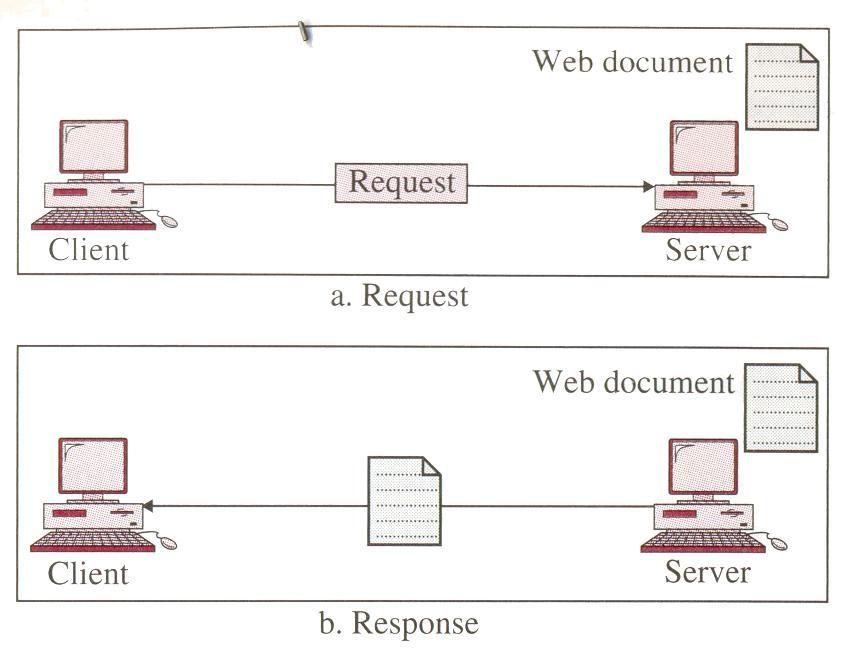
Dokumenti kod www-a mogu biti grupisani u slede**ć**e tri velike kategorije: stati**č**ke, dinami**č**ke, i aktivne (vidi sliku 9). Tip kategorije se zasniva na trenutku kada se odredjuju sadržaji dokumenta.



Slika 9 Kategorije web dokumenata

**Statički dokumenti**

Stati**č**ki dokumenti su dokumnti fiksnog-sadržaja koji se kreiraju i pamte u serveru. Klijent može da dobije samo kopiju tog dokumenta. Drugim re**č**ima, sadržaj fajla se odredjuje u trenutku kada se fajl kreira, a ne kada se koristi. Naravno da se sadržaj fajla u serveru može da se promeni, ali korisnik nije taj koji to može da uradi. Kada klijent pristupi dokumentu, predaje mu se kopija dokumenta. Korisnik zatim može da koristi program za pretraživanje (browsing program) sa ciljem da prikaže dokumenat (vidi sliku 10)

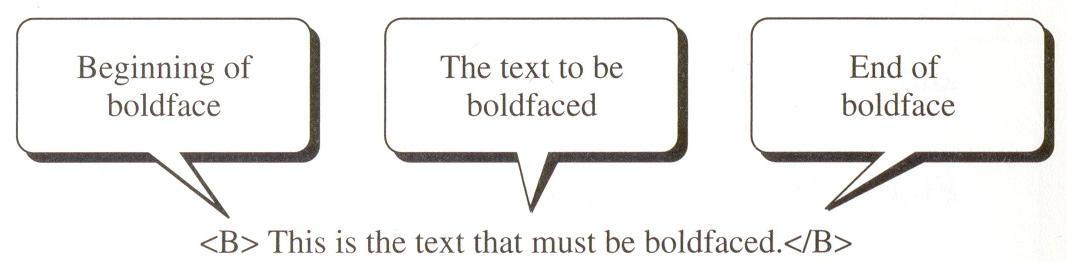


Slika 10 Stati**č**ki dokumenat

**HTML**

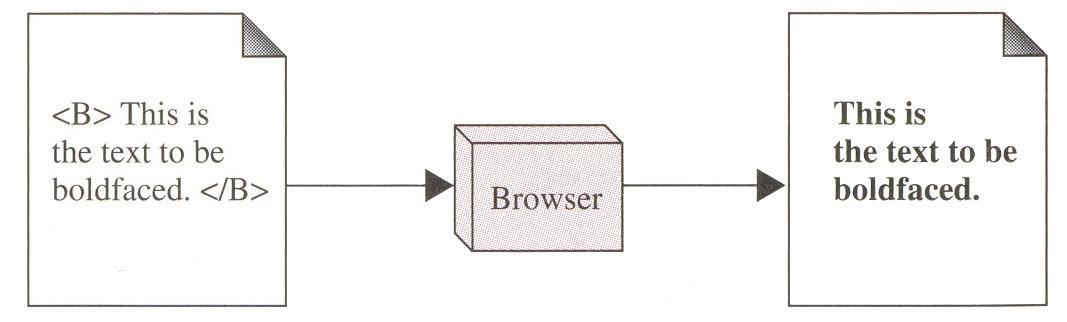
*Hypertext Markup Language* (HTML) je jezik za kreiranje web stranica. Termin *markup language* je pozajmljen od industrije za publikovanje knjiga. Pre nego što se knjiga složi ištampa, tehni**č**ki urednik **č**ita rukopis i daje zamerke. Na osnovu ovih zamerki slovo-slaga**č** zna kako da formatira tekst. Tako na primer, ako tehni**č**ki urednik ukaže da deo linije treba da bude štampan *bold*, on podvla**č**i talasasto taj deo linije. Na isti na**č**in podaci web stranice se formatiraju radi interpretacije od strane *browser*-a.

Objasnimo ideju na jednom primeru. Da bi u**č**inili deo teksta koji se u HTML-u prikazuje boldiran, neophodno je da na po**č**etku i kraju teksta postavimo marker, kako je to prikazano na slici 11.



Slika 11 Markeri za boldiranje

Oba markera <B> i </B> predstavljaju instrukcije za browser. Kada *browser* uo**č**i dva markera, on zna da tekst izmedju ta dva markera treba da bude boldiran (vidi sliku 12).



Slika 12 Efekat markera za boldiranje

HTML nam dozvoljava da koristimo samo ASCII karaktere, kako za glavni tekst tako i instrukcije za formatiranje. Na ovaj na**č**in, svaki ra**č**unar može da prihvati ceo dokumenat kao ASCII dokumenat. Glavni tekst su podaci, a instrukcije za formatiranje se mogu koristiti od strane *browser*-a za formatiranje podataka.

**Struktura web stranice**

Web stranicu **č**ine dva dela: zaglavlje i telo.

**Zaglavlje**

Zaglavlje (*head*) predstavlja prvi deo web stranice. Zaglavlje sadrži naslov stranice i druge parametre koje *browser* koristi.

**Telo**

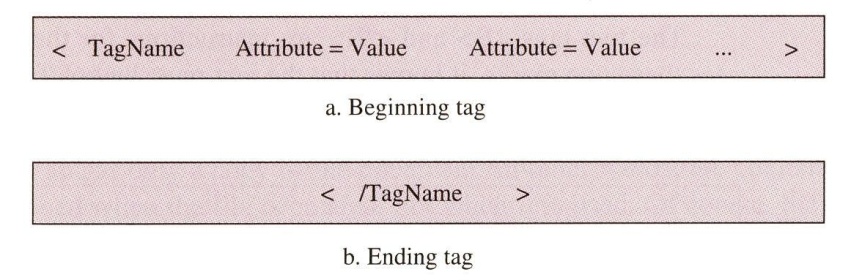
Aktuelni sadržaji stranice nalaze se u telu, koje sadrži tekst i markere. Dok tekst predstavlja aktuelna informacija koja se sadrži u okviru stranice, markeri (tagovi) definišu pojavljivanje (izgled) dokumenta. Svaki HTML marker predstavlja ime iza koga sledi opciona lista atributa, sve zatvoreno u okviru zagrade manje od i ve**ć**e od, tj.< >.

Nakon atributa, ako je prisutan, sledi znak jednakosti, i vrednost atributa. Neki od markera mogu da se koriste kao samostalni, dok se drugi koriste u parovima. Oni koji se koriste u parovima nazivaju se po**č**etni (startni) i krajnji markeri, respektivno. Startni marker može da ima atribute i vrednosti. Krajnji marker ne može imati atribute ili vrednosti, ali mora da ima simbol "/" ispred imena.

**Markeri**

*Browser* donosi odluku o strukturi teksta na osnovu markera, a to su obi**č**no markeri ugradjeniu tekst. Marker (*tag*) se zatvara sa dve zagrade < i >, i obi**č**no se markeri javljaju u parovima. Po**č**etni marker startuje imenom markera, a završni marker po**č**inje sa "/" iza **č**ega sledi ime markera.

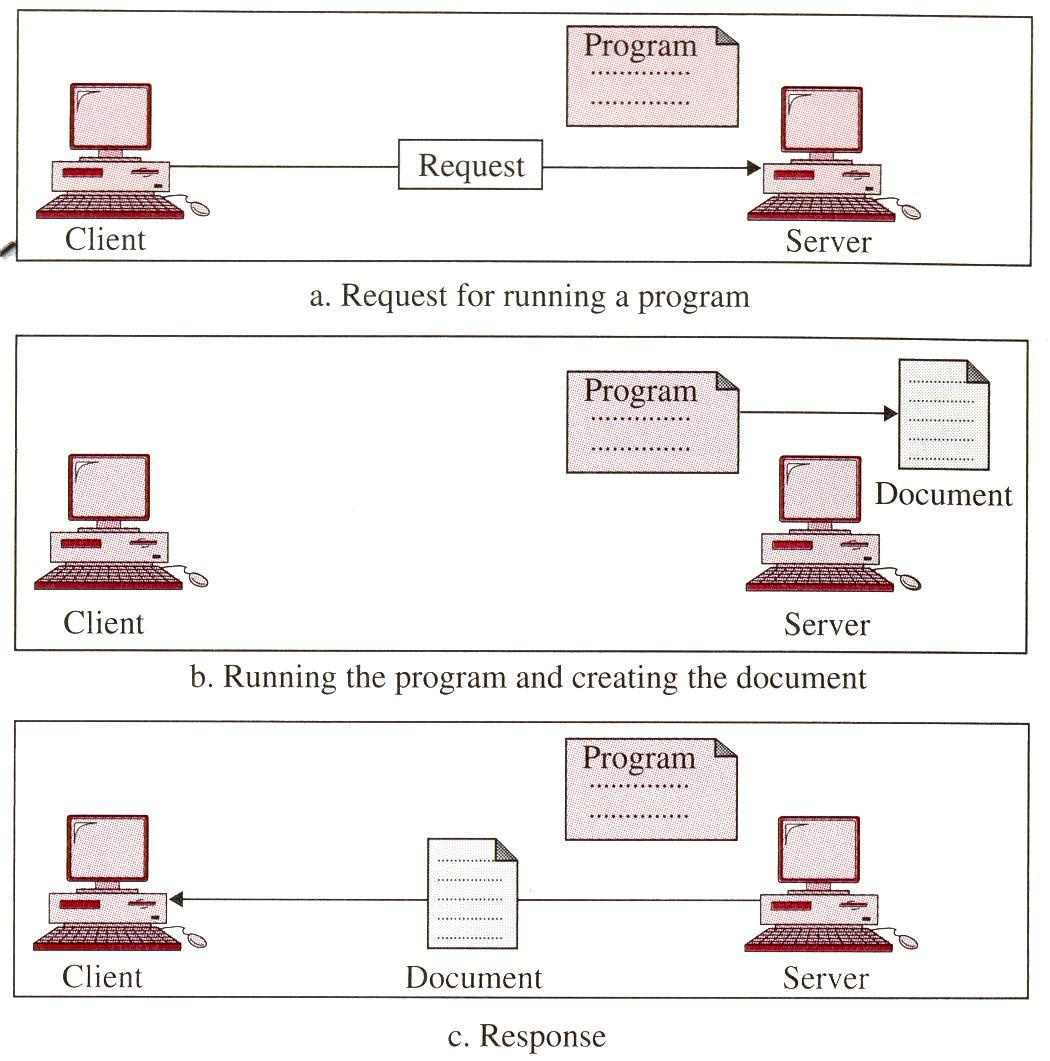
Marker može da ima listu atributa, pri **č**emu iza svakog atributa sledi znak jednakosti i vrednost koja se pridružuje tom markeru. Na slici 13 prikazan je format markera.



Slika 13 Po**č**etni i krajnji markeri.

**Dinamički dokumenti**

Dinami**č**ki dokumenti nemaju unapred definisani format. Umesto toga, dinami**č**ki dokumenat se kreira od strane web servera, uvek kada *browser* zahteva dokumenat. Kada zahtev pristigne web server izvršava aplikacioni program koji kreira dinami**č**ki dokumenat. Server vra**ć**a izlaz programa kao odziv *browser*-u koji je zahtevao dokumenat. S obzirom da se "taze" dokumenat kreira kod svakog zahteva, sadržaji dinami**č**kog dokumenta mogu da variraju od jednog zahteva do drugog. Veoma jednostavan primer dinami**č**kog dokumenta predstavlja dobijanje vremena i datuma od strane servera. Vreme i datum su tipovi informacije koje su dinami**č**ke po tome što se menjaju od trenutka do trenutka. Klijent može da zahteva od servera da izvršava program, kakav je program "*date*" kod Unix-a, i da predaje rezultat tog programa klijentu. Na slici 14 prikazani su koraci slanja i odazivanja na dinami**č**ki dokumenat.



Slika 14 Dinami**č**ki dokumenat

Server koji manipuliše sa dinami**č**kim dokumentima obavlja slede**ć**e korake:

1. Server ispituje URL kako bi odredio da li on definiše dinami**č**ki dokumenat
2. Ako URL definiše dinami**č**ki dokumenat, server izvršava program
3. Server predaje (šalje) izlaz programa klijentu (*browser*-u)

**Zajednički *gateway* interfejs**

Zajedni**č**ki *gateway* interfejs (*Common Gateway Interface – CGI*) je tehnologija kojom se kreiraju i manipuliše se dinami**č**kim dokumentima. CGI predstavlja skup standarda koji definišu kako treba napisati dinami**č**ki dokumenat, kako ulazne podatke dostaviti programu, i kako treba koristiti izlazne podatke.

CGI nije novi jezik, umesto toga, on dozvoljava programerima da koriste neki od slede**ć**ih jezika kakvi su C, C++, Bourne Shell, C Shell, TcL, ili Perl. Jedina stvar koju CGI definiše je skup pravila i naziva kojih programer mora da se pridržava.

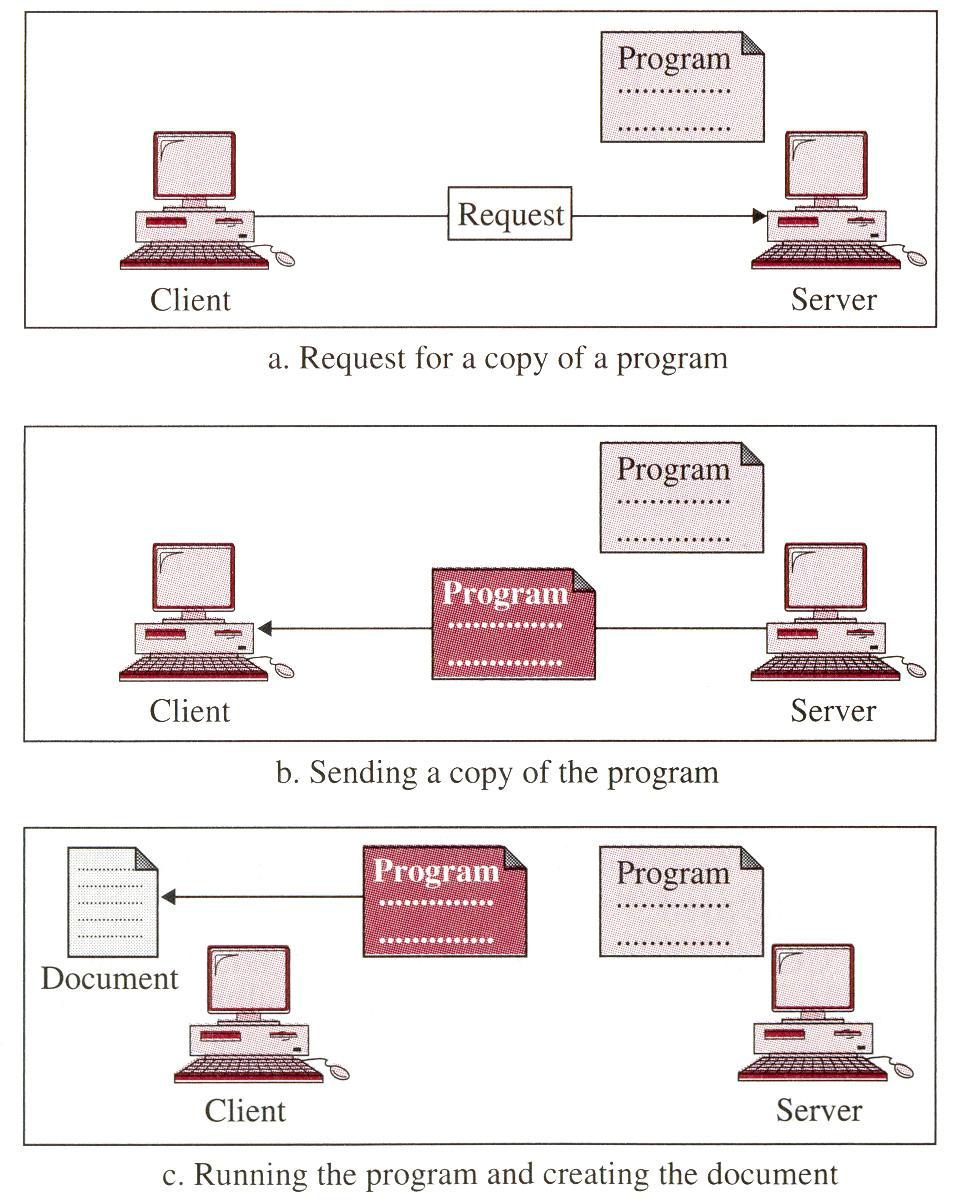
Koriš**ć**enje *common* kod CGI-a ukazuje da standard drefniše skup pravila koja su zajedni**č**ka za bilo koji jezik ili platformu. Termin *gateway* u ovom slu**č**aju zna**č**i da je CGI program *gateway* tipa koji se može koristiti za pristup drugim resursima kakve su baze podataka,grafi**č**ki paketi, i dr. Termin *interfejs* u ovom slu**č**aju zna**č**i da postoji skup unapred definisanih termina, promenljive, pozivi, i drugo, koje se može koristiti od strane bilo kog CGI programa.

**CGI program**

CGI program u svojoj najednostavnijoj formi je kod napisan na jednom od jezika koji podržavaju CGI. Bilo koji programer koji može kodirati sekvencu zamisli (ideja) u programu, a zna sintaksu jednog od prethodno'pomenutih jezika može kreirati jednostavan CGI program.

**Aktivni dokumenti**

Kod velikog broja aplikacija, neophodno je da postoji program koji **ć**e se izvršavati na strani klijenta. Ove aplikacije se nazivaju *aktivni-dokumenti*. Tako na primer, pretpostavimo da želimo da izvršimo program koji kreira animiranu grafiku na ekranu, ili interaguje sa korisnikom. Programu je definitivno potrebno da se izvršava na strani klijenta gde se odvija animacija ili interakcija. Kada *browser* zahteva aktivni dokumenat, server šalje kopiju dokumenta u formi bajt-koda. Dokument se zatim izvršava na strani klijenta (*browser*-a) kako je to prikazano na slici 15.



Slika 15 Aktivni dokumenat

Na strani servera aktivni dokumenat je memorisan u formi binarnog koda. Treba naglasiti, da se ne stvara *overhead* (ne unosi dodatno režijsko vreme) na strani servera kakav je to slu**č**aj sa dinami**č**kim dokumentima. I pored toga što se aktivni dokumenat ne izvršava od strane servera, on se memoriše kao binarni dokumenat koji se izvla**č**i od strane klijenta. Kada klijent primi dokumenat, on ga može zapamtiti u svoju memorijsku oblast. Na ovaj na**č**in, klijent može ponovo da izvršava dokumenat bez da izda drugi zahtev.

Aktivni dokumenat se prenosi od servera ka klijentu u binarnoj formi. To zna**č**i da dokumenat može da bude komprimovan na strani servera, a dekomprimovan na strani klijenta, pri **č**emu se štedi kako na propusnosti tako i na vremenu prenosa.

**Java**

Java predstavlja kombinaciju HLL-a, izvršenja u okruženju, i klasu biblioteke koja omogu**ć**ava programeru da napiše aktivni dokumenat, a *browser*-u da ga izvrši. Java se može takodje koristiti kao samostalni program bez koriš**ć**enja *browser*-a. Ipak, Java se naj**č**eš**ć**e koristi za kreiranje apleta (*applet*), tj malih aplikacionih programa.

Java je objektno-orijentisani jezik, koji je, sintakti**č**ki i semanti**č**ki, veoma sli**č**an C++. No on nije tako složen kao C++ i nema te kompleksnosti koje se odnose na operator *overloading* ili višestruko nasledjivanje (*multiple inheritance*). Java je takodje hardverski nezavisna i ne koristi pokaziva**č**e. Kod Java, kao i kod drugih objektno-orijentisanih jezika, programer definiše skup objekata i postavlja operacije (metode) koje operišu nad tim objektima. On je *typed* jezik, što zan**č**i da programer mora da deklariše tip bilo kog dela podatka pre nego štoga po**č**ne koristiti.

[www.maturski.org](http://www.maturski.org/)