POGLAVLJE 9

Veb klijenti i serveri

Ako imate čitač veba iz CERN-ovog projekta WWW
(World Wide Web, distribuirani sistem hiperteksta), možete da pregledate
priručnik u WWW verziji s hipertekstom.
– Gvido van Rosum, novembra 1992.
(prvo pominjanje veba na Pythonovoj listi slanja)

U ovom poglavlju...

- Uvod
- Pythonove alatke za veb klijente
- Veb klijenti
- Veb (HTTP) serveri
- Srodni moduli

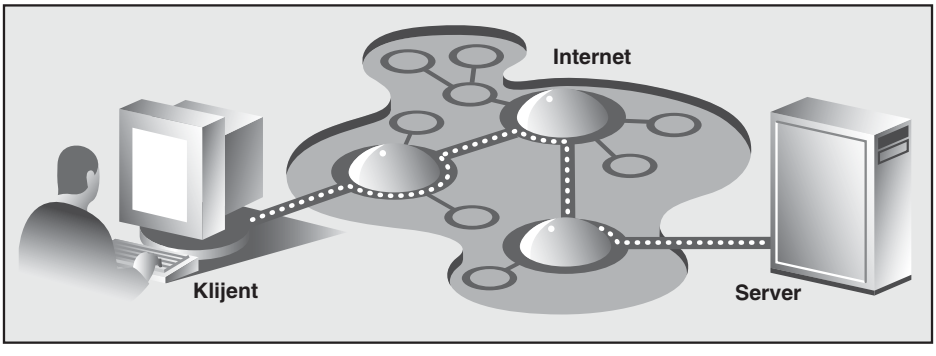
9.1 Uvod

Pošto se univerzum veb aplikacija neprestano širi, (re)organizovali smo ovu knjigu tako da omogućava čitaocima da se naročito fokusiraju na više aspekata proramiranja za veb kroz nekoliko poglavlja od kojih svako pokriva po jednu takvu temu.

U ovom poglavlju o veb programiranju, vraćamo se arhitekturi klijent/server, ali ovog puta iz perspektive veba. Ovo je solidna osnova za materijal iz preostalih poglavlja knjige.

9.1.1 Krstarenje vebom: klijent/server računarstvo

Krstarenje vebom spada pod isti kišobran arhitekture klijent/server koju smo razmatrali u više navrata. Međutim, ovog puta, *veb klijenti* su čitači veba (engl. *web browsers*) – to su aplikacije koje omogućavaju korisnicima da pregledaju dokumente na vebu. S druge strane su *veb serveri*, koji predstavljaju procese što se izvršavaju na računarima davalaca informacija. Ovi serveri čekaju na klijente i njihove zahteve za dokumentima, a potom vraćaju tražene podatke. Kao i većina servera u sistemu klijent/server, veb serveri su osmišljeni da rade neprestano. Iskustvo krstarenja vebom najbolje prikazuje slika 9-1. Vidimo kako korisnik izvršava program veb klijenta poput čitača veba, i povezuje se s veb serverom negde na internetu da bi dobio informacije.



Slika 9-1 Veb klijent i veb server na internetu. Klijent šalje zahtev preko interneta serveru, koji odgovara tako što šalje tražene podatke klijentu.

Klijenti mogu veb serverima da pošalju razne zahteve. U takve zahteve mogli bi da spadaju dobavljanje veb strane radi pregledanja ili slanje obrasca s podacima za obradu. Veb server potom obrađuje taj zahtev (a moguće i ostali sistemi), a klijent dobija odgovor u posebnom formatu, kako bi mogao da se prikaže.

Jezik kojim govore veb klijenti i serveri, standardni protokol za veb komunikaciju, naziva se protokol za prenos hiperteksta (engl. *HyperText Transfer Protocol, HTTP*). HTTP se piše povrh TCP i IP protokola, što znači da se oslanja na TCP i IP

koji za njega obavljaju komunikaciju nižeg nivoa. Ne preusmerava niti isporučuje poruke — za to se staraju TCP i IP — već odgovara na zahteve klijenata (šaljući i primajući HTTP poruke).

HTTP je poznat kao *protokol bez uspostavljanja stanja* (engl. *stateless*) jer ne prati informacije od jednog do drugog zahteva klijenta, što je slično onome što smo do sada videli o arhitekturi klijent/server. Server radi neprestano, ali interakcije sa klijentima su jedinstveni događaji s takvom strukturom da kada se zahtev jednog klijenta usluži, klijent završava. Uvek se mogu poslati novi zahtevi, ali smatraju se zasebnim zahtevima za servisom. Zbog nedostatka konteksta po zahtevu, možda ćete primetiti da pojedine URL adrese mogu da sadrže dug niz promenljivih i vrednosti kao deo zahteva da se dobije neka vrsta informacije o stanju. Druga alternativa je primena *kolačića* (engl. *cookies*) — statičkih podataka koji se čuvaju na klijentskoj strani, a načelno sadrže i informacije o stanju. Kasnije u ovom poglavlju, osvrnućemo se na to kako koristiti i duge URL vrednosti i kolačiće da održimo informacije o stanju.

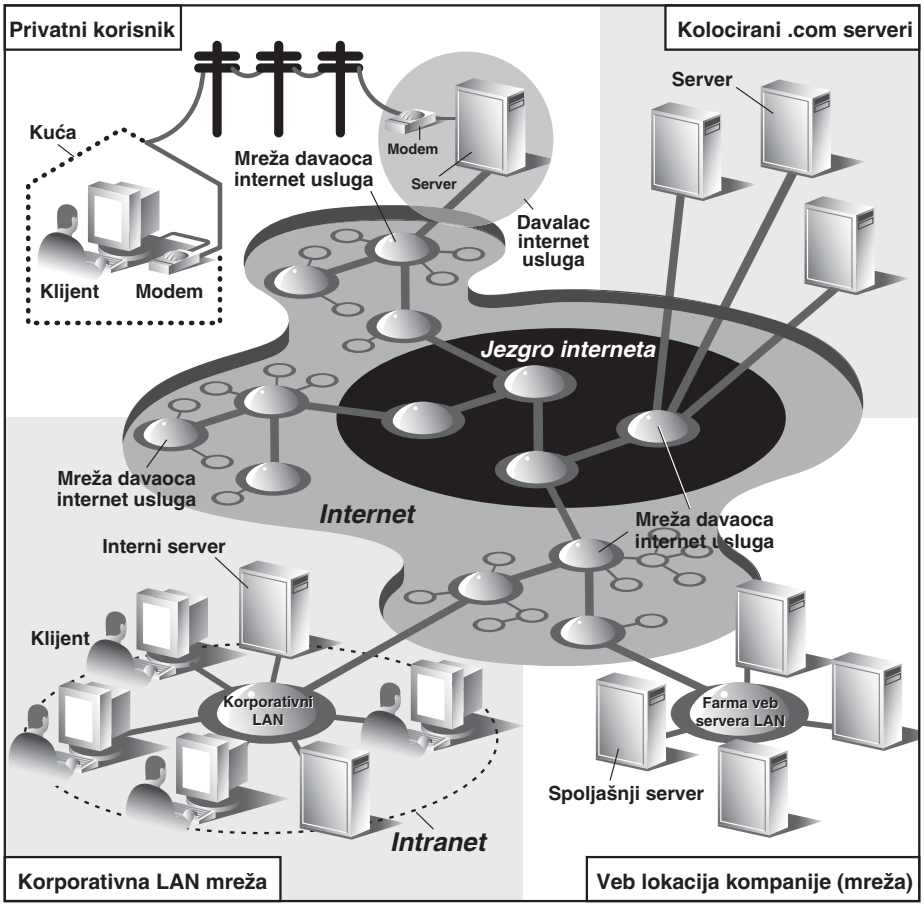
9.1.2 Internet

Internet je pokretni i promenljivi „oblak“ ili „jezero“ međusobno povezanih klijenata i servera razbacanih svuda po svetu. Metaforički govoreći, komunikacija od klijenta do servera sastoji se od niza konekcija od jednog do drugog lokvanja u jezeru, pri čemu je poslednji korak povezivanje sa serverom. Svi ovi detalji su skriveni od vas, klijenta korisnika. Apstrakcijom se stvara utisak direktne veze između vas (klijenta) i servera koji posećujete, ali ispod toga su skriveni protokoli HTTP, TCP i IP, koji obavljaju sav „prljav“ posao. Informacije o posredničkim *čvorovima* ionako nemaju važnost za običnog korisnika, pa je dobro što je implementacija skrivena. Slika 9-2 predstavlja proširenu prikaz interneta.

Valja pomenuti da se može desiti da bi u mnoštvu podataka koji se prenose internetom, neki mogli biti osetljiviji. Ne postoji podrazumevano dostupan servis za šifrovanje, te standardni protokoli podatke prenose onako kako ih aplikacije šalju. Dopunski nivo bezbednosti dodaje se običnim utičnicama, takozvani *sloj bezbednih utičnica* (engl. *secure socket layer, SSL*), radi šifrovanja svog prenosa koji se obavlja preko date utičnice. Na programerima je da odrede da li hoće tu dodatnu bezbednosnu komponentu.

Gde su klijenti i serveri

Kao što možete da zaključite na osnovu slike 9-2, internet čini više međusobno povezanih mreža koje funkcionišu donekle složno). Leva polovina dijagrama predstavlja veb klijente — korisnike koji su kod kuće, povezani preko svog davaoca internet usluga, ili rade preko lokalne LAN mreže svoje kompanije. Sa dijagrama su izostavljeni mehanizmi posebne namene (i popularni) kao što su zaštitne barijere (engl. *firewalls*) i posrednički serveri (engl. *proxy servers*).



Slika 9-2 Detaljniji prikaz interneta. Leva strana pokazuje gde ćete naći veb klijente, dok desna otkriva gde se obično nalaze veb serveri.

Zaštitne barijere doprinose borbi protiv neovlašćenog pristupa korporativnoj (ili kućnoj) mreži tako što blokiraju ulazne tačke te mreže. Bez njih, računari sa serverima mogli bi da dozvole uljezima da uđu preko nezaštićenog ulaza i steknu pristup sistemu. Administratori mreže smanjuju verovatnoću hakerskog upada tako što sve zaključavaju, a otvaraju samo ulaze za dobro poznate servise poput veb servera i pristupanja kroz bezbedno okruženje (engl. *secure shell, SSH*) koje se zasniva na pomenutom SSL sloju.

Posrednički serveri su još jedna korisna alatka koja može da se upotrebi zajedno sa zaštitnim barijerama (ili možda ne može). Administratorima mreže ponekad više odgovara da samo određen broj računara ima pristup internetu, možda da bi bolje pratili ulazni i izlazni saobraćaj njihovih mreža. Od koristi je i ako posrednički server može da kešira podatke. Evo primera: ako Linda pristupa veb stranic koja je kešira-

rana preko posredničkog servera, onda će se ta strana, kada je kasnije poseti njena koleginica Heder, brže učitati. Njen čitač veba nije morao da se muči da ide do veb servera, već je sve što mu treba dobio od posredničkog servera. Povrh toga, IT osoblje u njihovoj kompaniji sada zna da su barem dva zaposlena posetila tu veb stranu, kao i kada (a verovatno i ko tačno). Takvi serveri su poznati i kao *istureni posrednici* (engl. *forward proxies*), zbog onog što rade.

Sličan tip računara je *povučeni posrednik* (engl. *reverse proxy*). Ovakvi serveri funkcionišu otprilike suprotno isturenim posrednicima. (Zapravo, moguće je konfigurisati jedan računar tako da se ponaša i kao povučeni i kao istureni posrednik.) Povučeni posrednik radi poput servera s kojim klijenti mogu da se povežu. Verovatno će pristupati *pozadinskom serveru* (engl. *back-end server*) da dobije informacije koje traže klijenti. Povučeni posrednici takođe mogu da keširaju takve podatke od servera i da ih direktno vraćaju klijentu kao da su pozadinski serveri.

Dakle, umesto da keširaju umesto korisnika, „žive bliže" klijentima i služe ih, povučeni zastupnici žive bliže (pozadinskim) serverima. Postupaju u ime servera, vrlo verovatno keširajući podatke umesto njih, raspoređujući opterećenje, itd. Povučene zastupnike možete koristiti i kao zaštitne barijere ili da šifrujete podatke (SSL, HTTPS, Secure FTP (SFTP) itd.). Veoma su korisni, i vrlo je verovatno da ćete, krstareći vebom tokom dana, naići na više od jednog povučenog posrednika. Razmotrimo sada gde se neki od tih pozadinskih veb servera nalaze.

Na desnoj strani slike 9-2 fokus je na veb serverima; prikazano je gde se mogu naći. Korporacije s većim veb lokacijama uglavnom imaju čitavu *farmu veb servera* lociranu kod njihovog davaoca internet usluga. Takvo fizičko pozicioniranje naziva se *kolokacija,* što znači da se serveri kompanije nalaze kod davaoca internet usluga zajedno s računarima ostalih korporativnih korisnika. Moguće je da svi ti serveri prosleđuju različite informacije klijentima ili da su deo redundantnog sistema s dupliranim informacijama namenjenog velikoj potražnji (velikom broju klijenata). Manje kompanijske veb lokacije možda ne zahtevaju toliko hardvera i mrežne opreme, te bi mogli da imaju samo jedan ili nekoliko kolociranih servera kod svojih davalaca internet usluga.

U svakom slučaju, većina kolociranih servera smeštena je kod velikih davalaca internet usluga u *mrežnoj okosnici* (engl. *network backbone*), što znači da imaju širu i, verovatno, bržu vezu sa internetom — bliži su jezgru interneta, takoreći. To dozvoljava klijentima da brzo pristupe serverima — to što su deo okosnice mreže znači da klijenti ne moraju da skaču po velikom broju mreža kako bi pristupili serveru, što omogućava da se u datom vremenskom intervalu usluži više klijenata.

Internet protokoli

Trebalo bi da imate na umu sledeće: iako se internet najčešće koristi za krstarenje, to nije jedina njegova primena, a svakako ne najstarija. Internet prethodi vebu gotovo tri decenije. Pre veba, internet se uglavnom koristio u obrazovne i istraživačke svrhe, a mnogi od izvornih internet protokola, kao što su FTP, SMTP i NNTP, i danas su tu.

Pošto je Python od početka bio prepoznat kao jezik za internet programiranje, on podržava sve vrste pomenutih protokola, i mnoge druge. Razlikujemo „internet programiranje“ i „veb programiranje“: ovo drugo se odnosi samo na aplikacije napravljene specijalno za veb, kao što su veb klijenti i serveri, što je fokus ovog poglavlja.

Internet programiranje pokriva širi opseg aplikacija, uključujući aplikacije koje koriste neke od prethodno pomenutih internet protokola, uz mrežno programiranje i programiranje utičnica u načelu – sve to smo razmatrali u prethodnim poglavljima ove knjige.

9.2 Pythonove alatke za veb klijente

Treba imati na umu da je čitač veba samo jedan tip veb klijenta. Svaka aplikacija koja šalje zahtev za podacima veb serveru smatra se klijentom. Da, moguće je napraviti druge klijente koji dobavljaju dokumente ili podatke sa interneta. Jedan od važnijih razloga zašto se to radi jeste to što čitač veba ima ograničen kapacitet; koristi se prevashodno za pregledanje i interakciju s veb lokacijama. Klijentski program, s druge strane, ima mogućnost da radi više — ne samo da može da preuzme podatke, već i da ih sačuva, upravlja njima, ili možda čak i da ih prenese na drugu lokaciju ili drugoj aplikaciji.

Aplikacije koje koriste modul `urllib` za preuzimanje informacija na vebu ili pristupanje njima (pomoću funkcije `urllib.urlopen()` ili `urllib.urlretrieve()`) mogu se smatrati jednostavnim veb klijentom. Potrebno je samo da navedete ispravnu veb adresu.

9.2.1 Jedinstvene adrese resursa

Obično krstarenje vebom podrazumeva korišćenje veb adresa zvanih *jedinstvene adrese resursa* (*Uniform Resource Locators* – URL). Takve adrese se koriste da se locira dokument na vebu ili da se pozove CGI program da generiše dokument za vašeg klijenta. URL adrese su deo šireg skupa identifikatora poznatih kao *jedinstveni identifikatori resursa* (*Uniform Resource Identifiers* – URI). Ovaj nadskup napravljen je u očekivanju drugih konvencija imenovanja koje tek treba da se definišu. URL je, naprosto, URI koji koristi postojeći protokol ili šemu (to jest, http, ftp itd.) kao deo svog adresiranja. Da bismo upotpunili ovu sliku, dodaćemo da se URI identifikatori koji nisu URL adrese ponekad zovu *jedinstvena imena resursa (Uniform Resource Names* – URN), ali kako su URL adrese jedine URI vrednosti koje se danas koriste, i ne morate da slušate mnogo o URI ili URN vrednostima, sem, možda o XML identifikatorima.

Veb adrese imaju određenu strukturu – baš kao ulične adrese. U Americi, ulična adresa obično ima format „broj/ulica“ – na primer, 123 Main Street. U drugim zemljama važe drugačija pravila. URL koristi naredni format:

```
prot_sch://net_loc/path;params?query#frag
```

Tabela 9-1 opisuje svaku od komponenata formata.

Tabela 9-1 Komponente veb adrese

URL komponenta	Opis
prot_sch	Mrežni protokol ili šema preuzimanja
net_loc	Lokacija servera (i možda informacija o korisniku)
path	Putanja do datoteke ili CGI aplikacije, razgraničena kosim crtama (/)
params	Opcioni parametri
query	Skup parova ključ/vrednost razdvojenih ampersendom (&)
frag	Fragment vezan za dato sidre unutar dokumenta

net_loc se može razložiti na nekoliko komponenata od kojih su neke obavezne, a druge opcione. Znakovni niz *net_loc* izgleda ovako:

```
user:passwd@host:port
```

Ove komponente opisane su u tabeli 9-2.

Tabela 9-2 Komponente lokacije mreže

Komponenta	Opis
user	Korisničko ime
passwd	Korisnička lozinka
host	Ime ili adresa računara koji izvršava veb server (obavezno)
port	Broj ulaza (ukoliko nije podrazumevani, 80)

Od četiri komponente, najvažnije je ime `host`. Broj `port` je neophodno samo ako se veb server izvršava na ulazu s brojem koji nije podrazumevan. (Ako niste sigurni šta je broj ulaza, pročitajte poglavlje 2, „Mrežno programiranje“.)

Korisnička imena, možda i lozinke, koriste se samo prilikom uspostavljanja FTP konekcija, a čak i tada obično nisu neophodni, jer je većina tih konekacija anonimna.

Python ima dva modula za rad sa URL adresama pomoću potpuno različitih funkcionalnosti i alatki. Jedan je `urlparse`, a drugi `urllib`. Sada ćemo se nakratko osvrnuti na neke od njihovih funkcija.

9.2.2 Modul urlparse

Modul urlparse pruža osnovnu funkcionalnost za upravljanje URL znakovnim nizovima. U te funkcije spadaju urlparse(), urlunparse() i urljoin().

urlparse.urlparse()

urlparse() razlaže URL znakovni niz u neke od prethodno opisanih glavnih komponenata. Ima ovakvu sintaksu:

```
urlparse(urlstr, defProtSch=None, allowFrag=None)
```

urlparse() razlaže znakovni niz *urlstr* na šestorku (prot_sch, net_loc, path, params, query, frag). Svaka od ovih komponenata je opisana ranije. *defProtSch* određuje podrazumevani mrežni protokol ili šemu preuzimanja, u slučaju da nisu navedeni u znakovnim nizu urlstr. *allowFrag* je indikator koji označava da li je deo URL adrese koji opisuje fragment dozvoljen. Evo ispisa rezultata funkcije urlparse() pošto joj se prosledi URL:

```
>>> urlparse.urlparse('http://www.python.org/doc/FAQ.html')
('http', 'www.python.org', '/doc/FAQ.html', '', '', '')
```

urlparse.urlunparse()

urlunparse() se ponaša upravo suprotno od funkcije urlparse() — objedinjuje šestorku (prot_sch, net_loc, path, params, query, frag) — *urltup*, koja bi mogla biti rezultat funkcije urlparse(), u jedan URL znakovni niz koji vraća. Prema tome, možemo da uspostavimo narednu jednakost:

```
urlunparse(urlparse(urlstr)) ≡ urlstr
```

Sintaksa funkcije urlunparse() može se izraziti na sledeći način:

```
urlunparse(urltup)
```

urlparse.urljoin()

Funkcija urljoin() je korisna kada je potreban veći broj srodnih URL adresa – na primer, URL adresa za skup stranica koje se generišu za veb lokaciju. Evo sintakse funkcije urljoin():

```
urljoin(baseurl, newurl, allowFrag=None)
```

Funkciji urljoin() prosleđuje se adresa *baseurl*, da bi spojila njenu osnovnu putanju (*net_loc* sa punom putanjom do datoteke na kraju, ali ne uključujući datoteku) sa vrednošću *newurl*. Na primer:

```
>>> urlparse.urljoin('http://www.python.org/doc/FAQ.html',
... 'current/lib/lib.htm')
'http://www.python.org/doc/current/lib/lib.html'
```

Kratak pregled funkcija modula urlparse dat je u tabeli 9-3.

Tabela 9-3 Glavne funkcije modula `urlparse`

Funkcije modula `urlparse`	Opis
`urlparse(`*`urlstr`*`, `*`defProtSch`*`=None, `*`allowFrag`*`=None)`	Raščlanjuje znakovni niz *`urlstr`* u pojedinačne komponente, koristeći *`defProtSch`* ako protokol ili šema nisu navedeni u znakovnim nizu *`urlstr`*; *`allowFrag`* određuje da li je URL fragment dozvoljen
`urlunparse(`*`urltup`*`)`	Objedinjuje n-torku URL podataka (*`urltup`*) u jedan URL znakovni niz
`urljoin(`*`baseurl`*`, `*`newurl`*`, `*`allowFrag`*`=None)`	Spaja osnovni deo URL adrese *`baseurl`* sa vrednošću *`newurl`* da bi se formirala potpuna URL adresa; *`allowFrag`* ima isto značenje kao za `urlparse()`

9.2.3 Modul/paket `urllib`



MODUL: `urllib` u Pythonu 2 i Pythonu 3

Osim ako planirate da pišete mrežni klijent nižeg nivoa, modul `urllib` ima svu funkcionalnost koja vam je potrebna. `urllib` ima biblioteku za veb komunikaciju visokog nivoa, podržavajući osnovne veb protokole – HTTP, FTP i Gopher – i omogućavajući pristup lokalnim datotekama. Funkcije modula `urllib` osmišljene su tako da preuzimaju podatke (sa interneta, lokalne mreže ili računara) pomoću pomenutih protokola. Ako koristite ovaj modul, načelno nema potrebe za modulima `httplib`, `ftplib` i `gopherlib` sem ako želite funkcionalnost niskog nivoa. U tom slučaju, ovi moduli se mogu smatrati alternativama. (Napomena: većina `*lib` modula namenjena je pisanju klijenata odgovarajućih protokola. Međutim, nije uvek tako, jer bi `urllib` onda trebalo nazvati „internetlib“ ili nešto slično!)

3.x

U Pythonu 3, načinjen je korak u smeru objedinjavanja modula *`urllib`*, *`urlparse`*, *`urllib2`* i drugih srodnih modula iz Pythona 2 u paket pod imenom *`urllib`*, tako da ćete segmente modula *`urllib`* i *`urllib2`* naći u modulu *`urllib.request`*, dok je *`urlparse`* postao *`urllib.parse`*. Paket *`urllib`* u Pythonu 3 takođe obuhvata podmodule *`response`*, *`error`* i *`robotparser`*. Imajte na umu ove promene dok čitate ovo poglavlje i pregledate primere ili rešavate vežbanja.

Modul `urllib` sadrži funkcije za preuzimanje podataka sa datih URL adresa kao i za kodiranje i dekodiranje znakovnih nizova da bi se podesili tako da mogu da budu deo ispravnih URL znakovnih nizova. U narednom odeljku razmatraćemo funkcije `urlopen()`, `urlretrieve()`, `quote()`, `unquote()`, `quote_plus()`, `unquote_plus()` i `urlencode()`. Razmotrićemo i neke metode dostupne za objekte nalik datotekama koje vraća funkcija `urlopen()`.