Postman

alat za testiranje i razvoj API interfejsa

PREVOD DRUGOG IZDANJA

DEJV VESTERVELD

kompjuter biblioteka

<packt>

Postman

alat za testiranje i razvoj API interfejsa

Dejv Vesterveld

Prevod II izdanja

kompjuter biblioteka

Packt›

Izdavač:

kompjuter biblioteka

Obalskih radnika 4a
Beograd, Srbija
Tel: 011/2520272
e-pošta: kombib@gmail.com
veb-sajt: www.kombib.rs

Za izdavača:
Mihailo J. Šolajić, direktor

Autor:
Dejv Vesterveld

Prevod: Nemanja Lukić
Recezent: Miroslav Ristić
Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Pekograf“, Zemun
Tiraž: 500
Godina izdanja: 2024.

Broj knjige: 579
Izdanje: Prvo
ISBN: 978-86-7310-602-1

Naslov originala:

API Testing and Development with Postman
Second Edition
ISBN 978-1-80461-790-8

Postman
alat za testiranje i
razvoj API interfejsa
Autorizovani prevod sa engleskog jezika.

SARADNICI

O AUTORU

Dejv Vesterveld strastveno deli svoje znanje i stručnost da pomogne testerima da ostanu relevantni u svetu softvera koji se stalno menja. Pomogao je mnogim testerima softvera svojim popularnim video kursevima i delio je svoje znanje na konferencijama, onlajn razgovorima i podkastima. Radi u softverskoj industriji dugi niz godina, na različitim pozicijama, uključujući uloge istraživačkog testera, programera za automatizaciju testova i programera za API integracije.

Želim da zahvalim svojoj ženi, Šarlin, na njenoj neizmernoj podršci u svemu što radim. Ti si stena koja mi omogućava da se predstavim svetu na način na koji to činim. Uvek si uzbuđena što delim svoje znanje sa svetom. Hvala ti na tvojoj ljubavi i podršci.

O RECENZENTIMA

Kristina Talajasingem je menadžer tima softverskih inženjera u kompaniji Northwestern Mutual sa 9 godina radnog iskustva u industriji. Ima diplomu iz softverskog inženjeringa i radila je za kompanije kao što su Sysco i Dassault Systèmes®. Strastveno voli testiranje, a pored svog svakodnevnog posla, aktivan je govornik na konferencijama i sastancima kao što su Star East, Women in Tech Global Conference i TestCon Europe.

Nil Makormik je menadžer za kontrolu kvaliteta koji radi u QA sferi od 1998. godine. Od početka 2000-ih se bavi API testiranjem. Učestvovao je u pionirskom radu na metodologijama testiranja u svojoj kompaniji, AAA, a 2020. godine je unapređen na svoju trenutnu poziciju. Veruje da testeri nikada ne bi trebalo da prestanu da uče, istražuju i, najvažnije, razvijaju se – kako na profesionalnom, tako i na ličnom planu.

Miroslav Ristić je redovni profesor na Prirodno-matematičkom fakultetu Univerziteta u Nišu, sa preko 25 godina iskustva u razvoju statističkog softvera. Posebno se ističe njegov rad na razvoju grafičkog korisničkog interfejsa R Commander za programski jezik R. Dugi niz godina recenzirao je značajan broj knjiga za izdavačku kuću Springer i časopis Journal of Applied Statistics. Od 2023. godine aktivno recenzira najaktuelnija izdanja izdavačke kuće "Kompjuter biblioteka". Nakon prevođenja, svako izdanje prolazi kroz njegovo stručno vrednovanje i recenziju prevoda, sa ciljem da se osigura da prevodi budu ne samo jasni, precizni i prilagođeni čitaocima, već i da održe visok kvalitet i stručnu relevantnost knjiga.

Predgovor

U svetu brze hrane, brze mode i strategija brzog plasiranja na tržište, da li je kvalitet uopšte važan? Pritisci sveta u kom živimo nas guraju ka traženju prečica, čak i kada je u pitanju proizvodnja softvera visokog kvaliteta. Ja dajem svoj doprinos u suprotstavljanju tom svetu. Mislim da je kvalitet važan. Ima dovoljno lako kvarljivih stvari u životu. Vreme je za više kvaliteta.

Ova knjiga je jedan mali ulog koji sam položio u nastojanju da pomognem svetu da vidi više softvera visokog kvaliteta. Bilo da ste profesionalni tester ili programer koji želi da sazna više o testiranju, možete da se pridružite pokušaju da unapredimo svet kroz kvalitetne softverske aplikacije.

API interfejsi postaju okosnica interneta. Pomažu kompanijama da ostvare eksternu komunikaciju, a takođe pružaju infrastrukturu za komunikaciju mnogih unutrašnjih delova modernih softverskih sistema. Brak održava dobra komunikacija, a isto važi i za internet. Dobra komunikacija između različitih usluga je važna za dobro funkcionisanje aplikacija. Zbog toga je testiranje API interfejsa važno za proizvodnju kvalitetnog softvera.

Na prvi pogled, ova knjiga se pretežno bavi alatom za testiranje API interfejsa, Postman, ali sam takođe pokušao da uključim primere i nastavne lekcije koje će vam pomoći da koristite taj alat na način koji će imati pravi uticaj na kvalitet. Ako pročitate ovu knjigu steći ćete temeljno poznavanje alata Postman, a takođe ćete imati solidne osnove da razmišljate o testiranju API interfejsa, uopšteno. Želim da steknete više od puke sposobnosti upravljanja alatom Postman. Takođe, želim da znate kada i kako da ga koristite, da biste bili efikasan učesnik kreiranja API interfejsa visokog kvaliteta.

Za koga je ova knjiga

Prva osoba za koju pišem sam ja. Mnoge ideje o kojima govorim u ovoj knjizi su stvari koje sam i sam učio pre nekoliko godina. U stvari, Postman tako često donosi nove mogućnosti da su neki delovi ovog drugog izdanja nove stvari koje sam naučio dok sam pisao knjigu.

Uvek se razvijam i učim i volim da delim ono što naučim s drugima da bih im pomogao na njihovom putu.

Testiranje API interfejsa može biti primamljivo. Međutim, to je sveobuhvatna praksa, što na početnike deluje obeshrabrujuće, pa sam ovu knjigu napisao prvenstveno za one testere softvera i programere koji kada treba da testiraju API interfejs ne znaju odakle da počnu. Trudio sam se da ne podrazumevam temeljno programersko iskustvo čitalaca, ali će vam poznavanje osnovnih programerskih pojmova svakako olakšati praćenje sadržaja knjige.

Ako radite kao tester softvera i zainteresovani ste da dopunite svoje veštine testiranjem API interfejsa, ova knjiga je svakako za vas. Ako ste programer koji želi da napreduje u smeru testiranja i kvaliteta, čestitam, na putu ste ka uspešnoj karijeri! Programeri koji znaju i razumeju kako se proizvodi kvalitetan softver će uvek biti traženi. U zavisnosti od vašeg obrazovanja možda ćete moći da preskočite neke delova ove knjige, ali ako joj posvetite potrebno vreme naučićete da koristite alat Postman i da dizajnirate i pišete dobre testove API interfejsa.

Šta obuhvata ova knjiga

Poglavlje 1, Terminologija i tipovi API interfejsa, upoznaje vas sa nekim osnovnim pojmovima i predstavlja različite tipove API interfejsa.

Poglavlje 2, Dokumentacija i dizajn API interfejsa, predstavlja principe dizajna koji se primenjuju pri kreiranju i testiranju API interfejsa, i odgovara na pitanja *kako* i *zašto* kreirati korisnu dokumentaciju.

Poglavlje 3, OpenAPI i API specifikacije, objašnjava šta su API specifikacije i kako se koriste u alatu Postman.

Poglavlje 4, Razmatranja za dobru automatizaciju API testova, uči vas kako kreirati i izvršavati vredne i dugotrajne API testove u alatu Postman.

Poglavlje 5, Koncept opcija autorizacije, predstavlja mnoge metode autorizacije API interfejsa koji su dostupni u alatu Postman.

Poglavlje 6, Kreiranje skriptova za validaciju testova, objašnjava kako se kreiraju i koriste skriptovi za testove u alatu Postman.

Poglavlje 7, Testiranje vođeno podacima, razmatra šta je testiranje vođeno podacima i kako ono služi za kreiranje skalabilnih testova u alatu Postman.

Poglavlje 8, Testiranje radnog toka, objašnjava šta su testovi radnog toka i kako se kreiraju tokovi u alatu Postman.

Poglavlje 9, Pokretanje API testova u CI procesu sa alatom Newman, pokazuje kako pokrenuti Postman API testove putem komandne linije sa alatom Newman.

Poglavlje 10, Praćenje API interfejsa pomoću alata Postman, objašnjava kako se prati upotreba API interfejsa pomoću alata Postman.

Poglavlje 11, Testiranje postojećeg API interfejsa, sadrži praktičan primer koji pokazuje kakve testove treba kreirati prilikom testiranja postojećeg API interfejsa.

Poglavlje 12, Kreiranje i korišćenje lažnih servera u alatu Postman, objašnjava šta su lažni serveri i kako ih postaviti i koristiti u alatu Postman.

Poglavlje 13, Korišćenje testiranja ugovora za verifikaciju API interfejsa, razmatra šta je testiranje ugovora i pokazuje kako ga kreirati i koristiti u alatu Postman.

Poglavlje 14, Testiranje bezbednosti API interfejsa, je kratak uvod u testiranje bezbednosti i primer podešavanja testiranja sa slučajnim podacima u alatu Postman.

Poglavlje 15, Testiranje performansi API interfejsa, predstavlja različite tipove testiranja performansi i neke od funkcija alata Postman koje mogu poslužiti za procenu performansi API interfejsa.

Da biste najbolje iskoristili ovu knjigu

Svrha ove knjiga je da vas opremi veštinama koje ćete moći da koristite odmah, u svom radu kao tester ili programer. Ako želite da izvučete najviše iz ove knjige, primenjujte naučeno čim budete mogli. Prođite sve vežbe u knjizi i pokušajte da primenite ideje koje ćete naučiti u „stvarnom životu".

Ne podrazumeva se obimno predznanje o API interfejsima, pa čak ni o principima razvoja i testiranja. Dokle god imate osnovno znanje o veb tehnologijama i razvoju softvera uopšte, trebalo bi da budete u mogućnosti da pratite knjigu i usvojite sve što je potrebno. Neki od test skriptova u alatu Postman koriste JavaScript, ali nije potrebno da mnogo znate o tome, iako je osnovno razumevanje vrlo korisno. Knjiga sadrži primere i izazove. Oni su važan deo knjige i da biste izvukli najviše iz nje, trebalo bi da im posvetite potrebno vreme.

Preuzimanje datoteka primera koda

Paket kodova za knjigu se nalazi na GitHub platformi, na adresi `https://github.com/PacktPublishing/API-Testing-and-Development-with-Postman-Second-Edition`. Postoje i drugi paketi kodova iz našeg bogatog kataloga knjiga i video zapisa, dostupni na adresi `https://github.com/PacktPublishing/`. Pregledajte ih!

Preuzimanje slika u boji

Takođe, na raspolaganju je PDF datoteka koja sadrži slike u boji snimaka ekrana i dijagrama korišćenih u ovoj knjizi. Možete je preuzeti na adresi:
`https://packt.link/gbp/9781804617908`.

Konvencije

U ovoj knjizi koristili smo nekoliko tekstualnih konvencija.

`KodUTekstu`: Označava delove koda u tekstu, nazive tabela u bazi podataka, nazive direktorijuma, nazive datoteka, ekstenzije datoteka, putanje, lažne URL adrese, korisnički unos i Twitter naloge. Na primer: „Krajnja tačka `/product` daje informacije o proizvodima kojima pristupa ovaj API interfejs".

Blok koda je predstavljen na sledeći način:

```
openapi: 3.0.1
info:
    title: API interfejs za listu obaveza
    description: Upravlja zadacima liste obaveza
    version: „1.0"
servers:
    -url: https://localhost:5000/todolist/api
```

Kada želimo da skrenemo vašu pažnju na određeni deo bloka koda, relevantne linije ili stavke su podebljane:

```
/carts:
    post:
    get:
        queryParameter:
        username:
    /{cartId}:
        get:
        put:
```

Bilo koji unos ili izlaz iz komandne linije je napisan na sledeći način:

```
npm install -g newman
```

Podebljano: Označava novi termin, važnu reč ili reči koje vidite na ekranu. Na primer, reči u menijima ili okvirima za dijalog pojavljuju se u tekstu na sledeći način. Na primer: „Kliknite na dugme **Import** i odaberite opciju **OpenAPI**".

Saveti
ili
Važne napomene
Prikazani su ovako.

Postanite član Kompjuter biblioteke

Kupovinom jedne naše knjige stekli ste pravo da postanete član Kompjuter biblioteke. Kao član možete da kupujete knjige u pretplati sa 40% popusta i učestvujete u akcijama kada ostvarujete popuste na sva naša izdanja. Potrebno je samo da se prijavite preko formulara na našem sajtu.
Link za prijavu: kombib.rs/kblista.php

Skenirajte QR kod
registrujte knjigu
i osvojite nagradu

1

Terminologija i tipovi API interfejsa

Učenje nečeg novog može da podseća na pad sa broda. Sve je u pokretu i jedva držite glavu iznad vode. Čim shvatite kako nešto funkcioniše, nova saznanja se javljaju sa svih strana i sve kreće ispočetka. Nešto čvrsto, za šta se možete držati, daje vam priliku da se osvrnete i sagledate situaciju. To čini veliku razliku kada se uči nešto novo.

U ovom poglavlju želim da vam pružim tu čvrstu osnovu. Kao svaka specijalnost, testiranje i razvoj API interfejsa ima svoju terminologiju. Postoji mnogo termina koji imaju specijalizovana značenja kada radite sa API interfejsima. Koristiću neke od tih termina u ovoj knjizi i želim da izbegnem bilo kakav nesporazum.

Koristiću standardne definicije što je više moguće. Međutim, neki termini nemaju jasno određene definicije i objasniću kako nameravam da ih koristim i govorim o njima u ovoj knjizi. Budite svesni da ćete, dok čitate ili slušate materijale na internetu (ili komunicirate sa saradnicima), naići na ljude koji koriste malo drugačije termine.

Ova knjiga nije rečnik, tako da ne nameravam da pišem listu termina i njihove definicije. To bi bilo dosadno i verovatno ne bi bilo toliko poučno. Umesto toga, posvetiću malo vremena teoriji, šta je API interfejs i kako ga testirati. Objašnjenja i definicije važnih termina biće deo teksta.

Ovo poglavlje pokriva sledeće glavne teme:

- Šta je API interfejs?
- Tipovi API poziva
- Instaliranje alata Postman

- Struktura API zahteva
- Razmatranja za API testiranje
- Različiti tipovi API interfejsa

Do kraja ovog poglavlja bićete u stanju da koristite alat Postman za pravljenje API zahteva i imaćete dobro poznavanje osnovne API terminologije. Takođe ćete imati priliku da prođete vežbu koja će vam pomoći da utvrdite ono što ste naučili, što će vam omogućiti da počnete da koristite te veštine u svom svakodnevnom radu.

Šta je API interfejs?

NASA je 1969. godine objavila publikaciju pod naslovom *Computer Program Abstracts* koja sadrži sažetak programa za kontrolu prikaza u realnom vremenu, koji je prodavala kompanija IBM (samo 310 dolara! Dodatnih 36 dolara za dokumentaciju). Oglas kaže da je ovaj program dizajniran kao programski interfejs aplikacije za operatera – drugim rečima, kao API interfejs.

Programski interfejsi aplikacija (**API** interfejsi) postoje otprilike koliko i računarski kod. Konceptualno, to je samo način na koji dva različita dela koda (ili čovek i neki kod) međusobno komuniciraju. Klasa koja pruža određene javne metode koje drugi kod može pozvati ima API interfejs. Skript koji prihvata određene vrste unosa ima API interfejs. Upravljački program na vašem računaru koji zahteva da ga programi pozivaju na određeni način ima API interfejs.

Međutim, iako je internet porastao, pojam *API interfejsa* se suzio. Sad, gotovo uvek, kada neko govori o API interfejsu, misli na veb API interfejs. To je kontekst koji ću koristiti u ovoj knjizi. Veb API interfejs preuzima koncept interfejsa između dve stvari i primenjuje ga na odnos klijent/server na kojem je internet izgrađen. U veb API interfejsu, klijent je na jednoj strani interfejsa i šalje zahteve, dok je server (ili serveri) na drugoj strani interfejsa i odgovara na zahteve.

Vremenom se internet menjao i evoluirao i veb API interfejsi su se menjali i evoluirali zajedno s njim. Mnogi rani veb API interfejsi bili su izgrađeni za korporativnu upotrebu, sa strogim pravilima o načinu komunikacije dve strane. Za tu svrhu razvijen je tip API interfejsa nazvan **Jednostavan protokol za pristup objektima (SOAP)**. Međutim, početkom 2000-ih godina, veb je počeo da se orijentiše na potrošače. Neki od veb sajtova za elektronsku trgovinu, kao što su eBay i Amazon, počeli su da objavljuju API interfejse koji su bili transparentniji i fleksibilniji. Nakon toga, mnogi društveni mediji, uključujući Tviter, Fejsbuk i drugi, počeli su da koriste API interfejse. Mnogi od njih bili su izgrađeni po obrascu **Prenos stanja prikaza (REST)**.

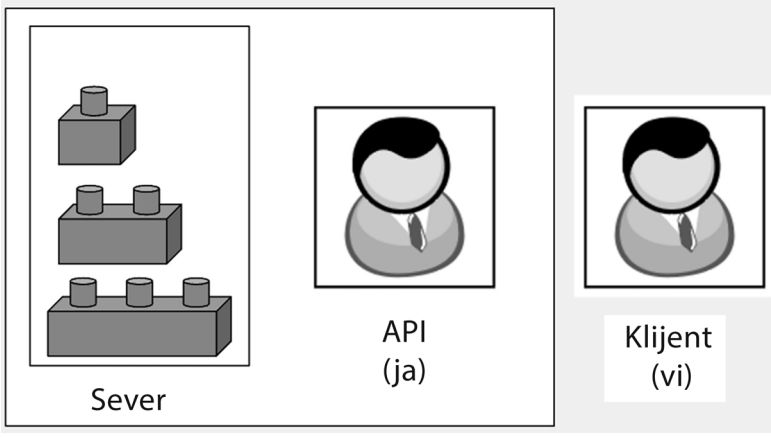
Taj pristup je fleksibilniji nego SOAP i izgrađen je direktno na osnovnim protokolima interneta.

Internet je nastavio da se menja, a kako je rasla popularnost mobilnih aplikacija i veb sajtova, tako je rasla i važnost API interfejsa. Neke kompanije su se suočile s izazovnim količinama podataka koje su želele da prenesu na mobilne uređaje, pa je Fejsbuk kreirao još jedan tip API interfejsa pod nazivom GraphQL. Ovaj tip API interfejsa definiše jezik upita koji pomaže da se smanji količina prenesenih podataka, dok istovremeno uvodi nešto rigidniju strukturu. Svaki od ovih različitih tipova API interfejsa dobro funkcioniše u nekim situacijama i to ću podrobnije objasniti kasnije u ovom poglavlju. Međutim, pre nego što uđemo u detalje svakog tipa API interfejsa, važno je razumeti neke od koncepata koji su osnova svih veb API poziva.

Tipovi API poziva

Neki API pozivi mogu da promene stanje na serveru, dok drugi vraćaju podatke bez promene. Termini **bezbedan** i **idempotentan** služe za opis različitih načina na koje API pozivi mogu uticati na podatke. Ovi termini možda zvuče zastrašujuće, pa ćemo ih bolje razumeti kroz ilustraciju koja koristi nešto što svi možemo da razumemo: LEGO kocke.

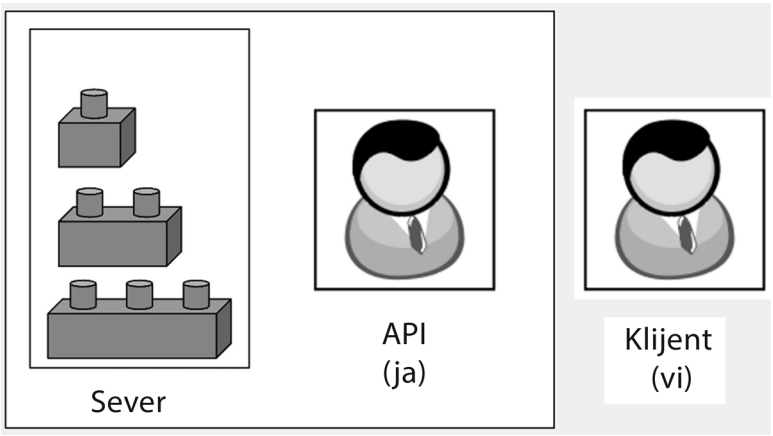
Zamislite sto sa nekoliko LEGO kocki na njemu i ja koji sedim za stolom. Ja predstavljam API interfejs, sto predstavlja server, a LEGO kocke predstavljaju objekte. Ako želite da komunicirate sa LEGO kockama, morate to učiniti preko mene. U ovoj ilustraciji, LEGO kocke predstavljaju objekte na serveru, ja predstavljam API interfejs, a vi predstavljate klijenta. Kao slika, to izgleda ovako:



Slika 1.1: *Predstavljanje servera i klijenta povezanih API interfejsom*

Vi ćete biti klijent, u ovom zamišljenom odnosu. To znači da od mene možete tražiti da radim nešto, sa LEGO kockama. Pitate me koliko je velika gornja LEGO kocka. Odgovaram da ima veličinu jedan. Ovo je primer API zahteva i odgovora koji je **bezbedan**. Bezbedan zahtev je onaj koji ne menja stanje na serveru. Traženjem informacija o tome šta se dešava na serveru, niste promenili ništa na samom serveru.

Međutim, postoje i druge vrste API poziva. Zamislite da mi date kocku veličine dva i tražite od mene da zamenim gornju kocku onom koju ste mi dali. Ja to radim i time menjam stanje servera. Gomila kocki sada se sastoji od kocke veličine tri i dve kocke veličine dva, kao što je prikazano na sledećem dijagramu:



Slika 1.2: *Novo stanje servera*

Pošto se stanje servera promenilo, to nije bezbedan zahtev. Međutim, ako mi date još jednu kocku veličine dva i tražite da ponovo zamenim gornju kocku onom koju ste mi upravo dali, ništa se neće promeniti na serveru. Gomila će i dalje biti sastavljena od kocke veličine tri i dve kocke veličine dva. To je primer **idempotentnog** poziva. API pozivi koji vraćaju isti rezultat, bez obzira na to koliko puta ih pozovete, poznati su kao idempotentni.

Zamislimo još jednu vrstu poziva. U ovom slučaju, dajete mi kocku i tražite da je dodam na vrh gomile. To uradim i sada imamo gomilu od četiri kocke. Ovo očigledno nije bezbedan poziv, jer se stanje servera promenilo, ali, da li je idempotentan?

Odgovor je ne, ali razmislite trenutak i uverite se da razumete zašto ovaj poziv nije idempotentan.

Ako vam je teško da shvatite zašto nije idempotentan, razmislite šta se dešava ako ponovite isti zahtev. Dajete mi još jednu kocku i tražite da je dodam na vrh gomile. Ako to uradim drugi put, da li je gomila kocki i dalje ista kao što je bila nakon prvog puta kada ste je dodali? Naravno da ne! Sada ima pet kocki i svaka dodatna kocka koju dodate na gomilu će je promeniti. Idempotentan poziv je onaj koji menja stvari samo prvi put kada ga izvršite i ne pravi nikakve promene prilikom sledećih poziva. Pošto ovaj poziv menja nešto svaki put, on nije idempotentan.

Bezbednost i idempotentnost su važni koncepti koje treba razumeti, posebno kada je reč o testiranju API interfejsa. Na primer, ako testirate pozive koji su bezbedni, možete pokretati testove paralelno, bez brige o njihovom međusobnom ometanju. Ali ako testirate pozive koji nisu bezbedni ili idempotentni, morate biti malo pažljiviji, koje testove pokrećete i kada ih pokrećete.

Postoji još nekoliko važnih pojmova koje treba da naučite, kao i strukturu API zahteva. Međutim, to će biti mnogo lakše ako postoji nešto konkretno, na šta možemo da se oslonimo, pa ćemo zato napraviti kratku pauzu da instaliramo alat Postman i pošaljemo naš prvi zahtev.

Instaliranje alata Postman

Postman može da se koristi kao veb ili kao desktop aplikacija. Funkcionalnost i dizajn su slični, a u ovoj knjizi ću uglavnom koristiti desktop aplikaciju, pa bih vam preporučio da i vi instalirate istu. Aplikacija je dostupna za Windows, Mac i Linux, a instalacija je ista kao i za bilo koji drugi program. Međutim, preporučio bih vam da napravite Postman nalog, ako ga već nemate. Kreiranje naloga je potpuno besplatno i olakšava vam upravljanje radom i deljenje vašeg rada. Besplatan Postman nalog stavlja na raspolaganje vrlo velikodušnu funkcionalnost. Postman ima neke napredne funkcije koje zahtevaju plaćeni nalog, ali svi primeri u ovoj knjizi će raditi i sa besplatnim nalogom. Međutim, ako nemate nalog, biće vam teško da pratite neke od primera, pa bih vam toplo preporučio da se registrujete:

1. Idite na adresu `https://postman.com`.
2. Izaberite opciju **Sign Up for Free**.
3. Napravite nalog i na pitanje kako želite da koristite Postman, izaberite opciju **Download Desktop App**, a zatim instalirajte aplikaciju kao i svaki drugi program.

Ako već imate Postman nalog, ali nemate desktop aplikaciju, možete preskočiti korake dva i tri i direktno je preuzeti sa početne Postman stranice.

Koristiću Postman verziju za Mac, ali osim eventualno neznatno drugačijeg izgleda ekrana, sve ostalo bi trebalo da bude isto bez obzira na platformu na kojoj koristite Postman.

Primarno ću koristiti desktop aplikaciju u ovoj knjizi, ali u neki situacijama je veb aplikacija izuzetno korisna. Preporučio bih vam da je podesite, takođe. Tokom procesa registracije možete preuzeti program Desktop Agent. Ovaj program omogućava veb verziji alata Postman da zaobiđe određena ograničenja koja veb pregledači postavljaju na veb zahteve. Ova ograničenja su vrlo važna za bezbednost na vebu, ali prilikom testiranja često moramo da ih zaobiđemo, a ovaj program će vam to omogućiti. Ponovo, možete ga preuzeti i instalirati kao i svaki drugi program.

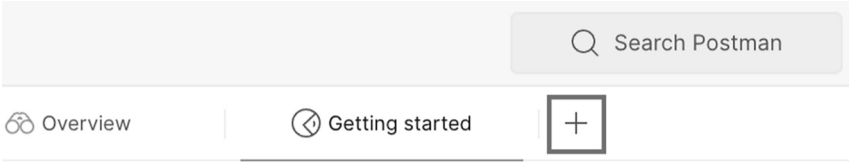
Pokretanje alata Postman

Kada instalirate Postman, otvorite aplikaciju. Prvi put kada otvorite Postman, moraćete da se prijavite. Kada se prijavite, videćete glavni ekran sa mnogo različitih opcija koje su vam na raspolaganju sa alatom Postman. Nemojte sada brinuti o svim tim opcijama. Pomenućemo ih sve (i još neke) u ovoj knjizi. Za sada samo zapamtite da postoje i da možete raditi mnogo zanimljivih stvari sa alatom Postman. Možda se čak i malo uzbudite, zbog saznanja šta ćete sve naučiti iz ove knjige!

Podešavanje zahteva u alatu Postman

Vreme je da postavimo API poziv da bismo ga razložili i videli kako sve funkcioniše:

1. Da biste postavili novi zahtev, kliknite na **+** blizu vrha aplikacije, da dodate novu karticu.



Slika 1.3: *Kreiranje nove kartice za zahtev*

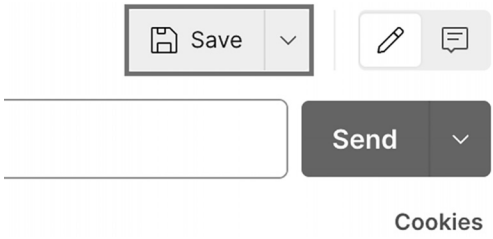
Za ovaj prvi primer, koristiću Postman Echo API. Ovo je API interfejs koji Postman pruža i koji se može koristiti za jednostavno testiranje.

2. U polje **Enter URL or paste text**, unesite sledeću URL adresu: `https://postman-echo.com/get`.
3. Kliknite na dugme **Send** desno od polja za URL adresu i trebalo bi da vidite odgovor od servera. Nemojte sada brinuti o podacima odgovora; za sada se samo pohvalite da ste poslali prvi zahtev sa alatom Postman!

Čuvanje zahteva

Sada kada ste kreirali zahtev, hajde da pogledamo kako se zahtevi čuvaju. Postman zahteva da se zahtevi čuvaju u nečemu što se zove kolekcije. Kolekcije su način da sakupite više zahteva koji pripadaju zajedno:

1. Kliknite na dugme **Save** iznad i desno od URL adrese zahteva.



Slika 1.4: *Čuvanje zahteva*

2. U iskačućem dijalog prozoru dajte zahtevu ime. Nazovite ga `Postman Echo GET`.
3. Kliknite na opciju **New Collection** na dnu iskačućeg prozora.
4. Imenujte kolekciju kao `Postman Echo Requests` i kliknite **Create**.
5. Kliknite na dugme **Save** na dijalogu. Tako ćete kreirati kolekciju i sačuvati zahtev u tu kolekciju.

Sada kada ste sačuvali zahtev, hajde da počnemo da istražujemo osnovnu strukturu API zahteva.

Struktura API zahteva

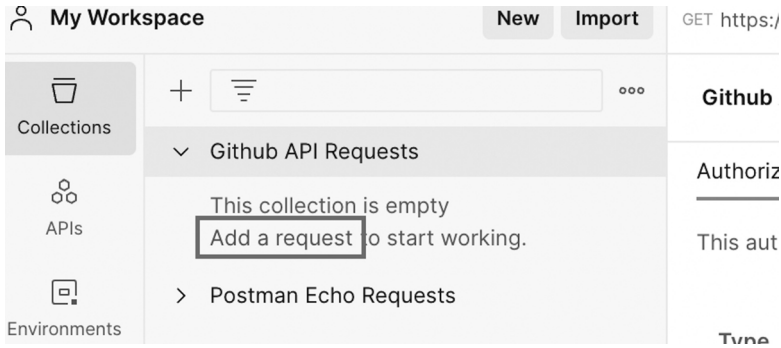
Kartica za zahteve pruža mnogo informacija o različitim delovima koji čine jedan API zahtev. Svaki od tih delova igra važnu ulogu u slanju i primanju podataka putem API interfejsa, pa ću vam predstaviti svaki od njih, redom. Neki delovi API zahteva su opcioni, zavisno od vrste zahteva i onoga što pokušavate da postignete, ali postoje tri dela potrebna za svaki API zahtev. Svaki API zahtev mora imati krajnju tačku, zaglavlja i akciju; pogledajmo to sledeće.

API krajnje tačke

Svaki zahtev zasnovan na vebu mora navesti **krajnju tačku**. U kartici **zahteva alata Postman**, od vas se traži da unesete URL adresu zahteva. Postman traži da unesete URL zato što je krajnja tačka API interfejsa upravo URL adresa. Termin *URL adresa* koristimo toliko često da ponekad zaboravimo šta on predstavlja. URL je skraćenica za **Uniformni lokator resursa**. Krajnja tačka API poziva određuje resurs, ili „R" iz skraćenice URL. Drugim rečima, krajnja tačka API interfejsa je uniformni lokator za određeni resurs sa kojim želite da komunicirate na serveru. URL adrese pomažu da locirate resurse na serveru, pa se koriste kao krajnje tačke API poziva.

Da biste to bolje razumeli, hajde da podesimo konkretan primer:

1. Kreirajte drugu kolekciju klikom na dugme **New** iznad navigacionog panela i odaberite **Collection** u iskačućem prozoru.
2. Imenujte kolekciju kao `GitHub API Requests`.
3. Ako proširite prikaz kolekcije, videćete poruku da je kolekcija prazna i link **Add a request**. Kliknite na taj link.



Slika 1.5: *Dodavanje zahteva u kolekciju*

4. Imenujte zahtev `Get Repos` i popunite polje za URL adresu zahteva sledećom URL adresom:
   `https://api.github.com/users/djwester/repos`.
5. Kliknite na dugme **Send** da vidite odgovor.

   Ova krajnja tačka će vam dati informacije o mojim javnim GitHub spremištima. Ako imate GitHub nalog, možete uneti svoje korisničko ime u delu URL adrese gde piše `djwester` i dobićete podatke za svoja spremišta.

Često ćete videti da je krajnja tačka API interfejsa navedena bez osnovnog dela ovog API interfejsa. Na primer, ako pogledate GitHub API dokumentaciju, ona navodi ovu krajnju tačku kao `/users/:username/repos`. Svi GitHub API pozivi počinju istom osnovnom URL adresom (drugim rečima, `https://api.github.com`), tako da se ovaj deo krajnje tačke često izostavlja, kada se govori o krajnjoj tački. Ako vidite da krajnje tačke počinju sa `/` umesto sa `http` ili `www`, zapamtite da morate pronaći osnovnu URL adresu API interfejsa za krajnju tačku, da biste mogli da uputite poziv.

API akcije

Svaki API poziv mora navesti resurs sa kojim radimo. Taj resurs je krajnja tačka, ali postoji još jedna stvar koja je potrebna svakom API pozivu. API interfejs mora nešto da uradi sa navedenim resursom. Ono što želimo da API interfejs uradi navodimo pomoću API **akcija**. Ove akcije ponekad nazivamo **glagoli**, jer govore API pozivu šta očekujemo da uradi sa resursom koji smo naveli. Za neke resurse su validne samo određene akcije, dok za druge može biti više različitih validnih API akcija.

U alatu Postman, možete odabrati željenu akciju iz padajućeg menija pored tekstualnog polja gde ste uneli URL adresu. Podrazumevano, Postman postavlja akciju na **GET**, ali ako kliknete na padajući meni, možete videti da postoji mnogo drugih akcija dostupnih za API pozive. Neke od ovih akcija su specijalizovane za određene aplikacije, tako da ih nećete često viđati. U ovoj knjizi ću koristiti samo **GET**, **POST**, **PUT** i **DELETE**. Mnogi API interfejsi koriste i **PATCH**, **OPTIONS** i **HEAD**, koje su veoma slične ovima koje ću ja koristiti, tako da ćete lako moći da ih koristite kada na njih naiđete. Ostatak akcija na ovoj listi se ne koristi često i verovatno ih nećete često viđati u aplikacijama koje testirate i kreirate.

Četiri akcije (**GET**, **POST**, **PUT** i **DELETE**) su nekad predstavljene skraćenicom **CRUD**. To znači Kreiraj (**C**), Čitaj (**R**), Ažuriraj (**U**) i Obriši (**D**). U API interfejsu, akcija **POST** služi za kreiranje novih objekata, akcija **GET** za čitanje informacija o objektima, akcija **PUT** za promenu (ili ažuriranje) postojećih objekata, i (iznenađenje, iznenađenje) akcija **DELETE** služi za brisanje objekata. U praksi, sa API interfejsom koji podržava sve CRUD aspekte imate mogućnost da uradite gotovo sve što je potrebno, i zbog toga ćete najčešće koristiti ove četiri akcije.

API akcije i krajnje tačke su potrebne za sve veb API interfejse, ali postoji nekoliko drugih važnih delova API zahteva koje ćemo razmotriti.

API parametri

API parametri služe za stvaranje strukture i reda API interfejsa. Oni organizuju slične stvari zajedno. Na primer, u API pozivu koji smo ranije napravili dobili smo GitHub spremišta određenog korisnika. Postoji mnogo GitHub korisnika, i možemo koristiti istu API krajnju tačku da bismo dobili listu spremišta bilo kog od njih jednostavnom promenom korisničkog imena u krajnjoj tački. Deo krajnje tačke koji prihvata različita korisnička imena je **parametar**.

Parametri zahteva

Parametar korisničkog imena u API krajnjoj tački za GitHub spremišta je poznat kao **parametar zahteva**. Možete zamisliti parametar zahteva kao promenljivu nisku u API krajnjoj tački. Oni su veoma uobičajeni za veb API interfejse. Videćete ih predstavljene na različite načine u dokumentaciji različitih API interfejsa. Na primer, GitHub dokumentacija koristi dvotačku ispred parametra zahteva kao oznaku da je to parametar zahteva, a ne samo još jedan deo krajnje tačke. U GitHub dokumentaciji krajnje tačke su određene na sledeći način:

```
/users/:username/repos.
```

U drugim API interfejsima, umesto toga, parametri zahteva su zatvoreni u vitičaste zagrade. U tom slučaju, krajnja tačka izgleda ovako:

```
/users/{{username}}/repos.
```

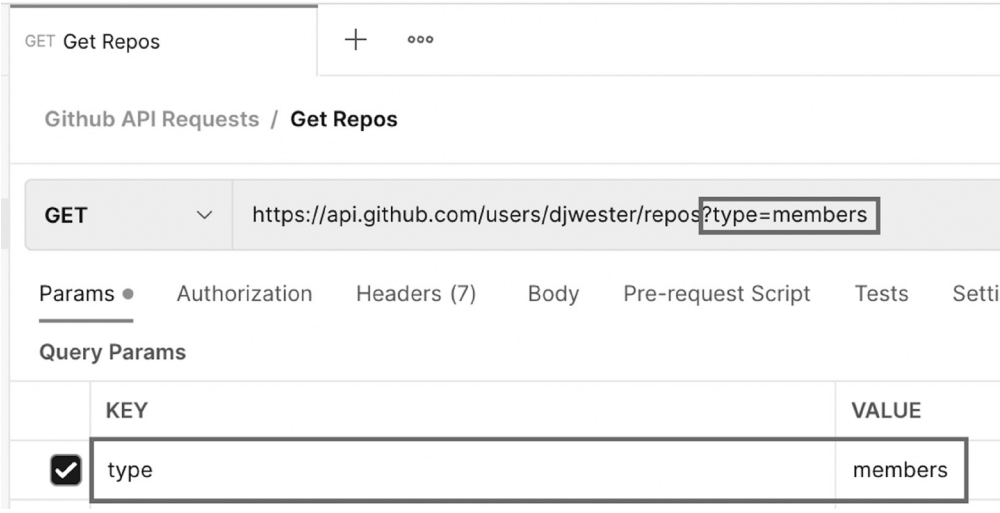
Bez obzira na format, poenta parametara zahteva je da dobijete informacije o različitim objektima koji su istog tipa. Već smo videli kako možete to da uradite sa ovom krajnjom tačkom, zamenom mog korisničkog imena vašim korisničkim imenom (ili bilo kojim drugim GitHub korisničkim imenom).

Parametri upita

Postoji još jedan tip parametra u API krajnjoj tački. To je **parametar upita**, malo komplikovaniji za upotrebu. Parametar upita često deluje kao vrsta filtera ili dodatne akcije koju možete primeniti na krajnju tačku. Predstavljen je znakom pitanja i određen ključem, što je stavka koju pretražujete, i vrednošću, što je ono što želite da pretraga vrati.

Sve je to vrlo apstraktno, pa hajde da vidimo to sa GitHub zahtevom koji smo ranije poslali. Ova krajnja tačka podržava nekoliko različitih parametara upita. Jedan od njih je parametar `type`. Da biste dodali parametre u API krajnju tačku u alatu Postman, izaberite karticu **Params**, a zatim unesite naziv parametra upita u polje **Key** i vrednost u polje **Value**. U ovom slučaju, koristićemo parametar `type`, pa unesite tu reč u polje **Key**.

Za ovu krajnju tačku, parametar `type` nam omogućava filtriranje na članove i vlasnika spremišta. Podrazumevano, krajnja tačka će vratiti samo ona spremišta čiji sam ja vlasnik, ali ako želim da vidim sva spremišta čiji sam član, uneću `member` u polje **Value**. U ovom trenutku, zahtev bi trebalo da izgleda otprilike ovako:



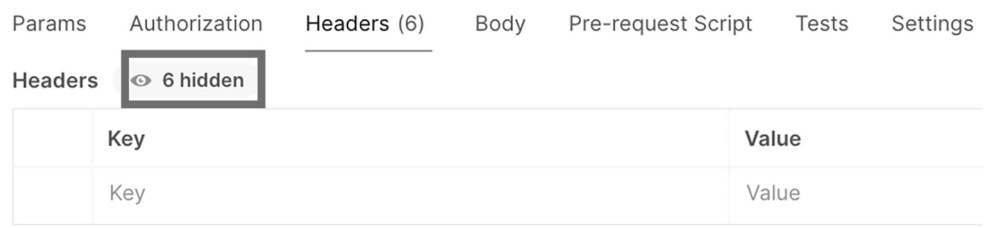
Slika 1.6: *Parametar upita type u API pozivu*

Kada navedete parametre upita u kartici **Params**, Postman ih automatski dodaje u URL adresu zahteva. Ako pošaljem ovaj zahtev, dobiću nazad sva spremišta čiji sam član, a ne samo ona čiji sam vlasnik. Parametri su moćna API paradigma, ali postoje još neki osnovni delovi API strukture o kojima nisam govorio. Sledeća stvar koju ćemo razmotriti su API zaglavlja.

API zaglavlja

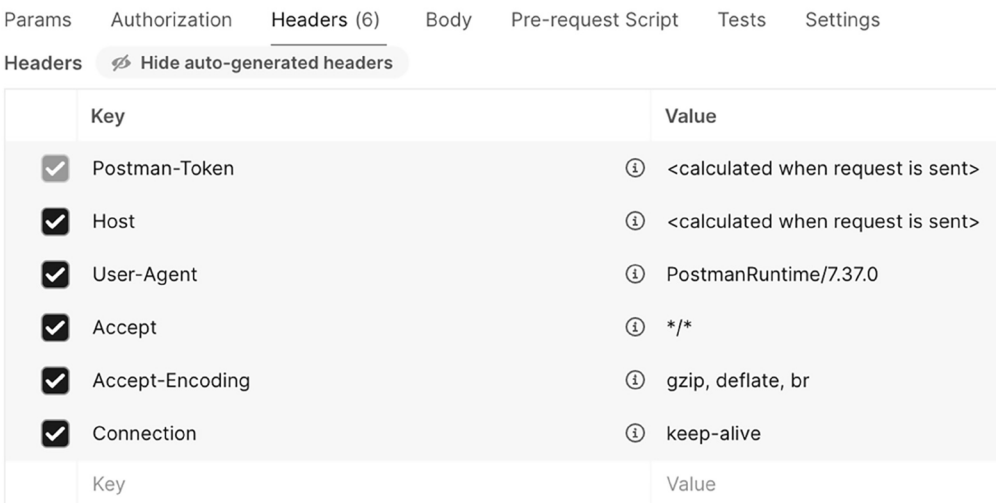
Svaki API zahtev obuhvata neko **zaglavlje**. Zaglavlja sadrže deo pozadinskih informacija koje često nisu toliko važne ljudskim korisnicima, ali serveru pružaju određene informacije o klijentu koji šalje zahtev. Ponekad ćemo morati da izmenimo ili dodamo određena zaglavlja da bi API interfejs radio ono što želimo, ali uglavnom možemo jednostavno da pustimo alat koji koristimo da pošalje podrazumevana zaglavlja, bez brige o tome.

U alatu Postman, na kartici **Headers** možete videti koja će zaglavlja biti poslata sa vašim zahtevom. Kada prvi put odete na tu karticu, možda ćete misliti da nijedno zaglavlje nije navedeno. U tom slučaju bi, verovatno, trebalo da promenite prikaz da biste prikazali automatski generisana zaglavlja:



Slika 1.7: *Zaglavlja su skrivena*

Kada su zaglavlja otkrivena, trebalo bi da vidite sledeću listu:



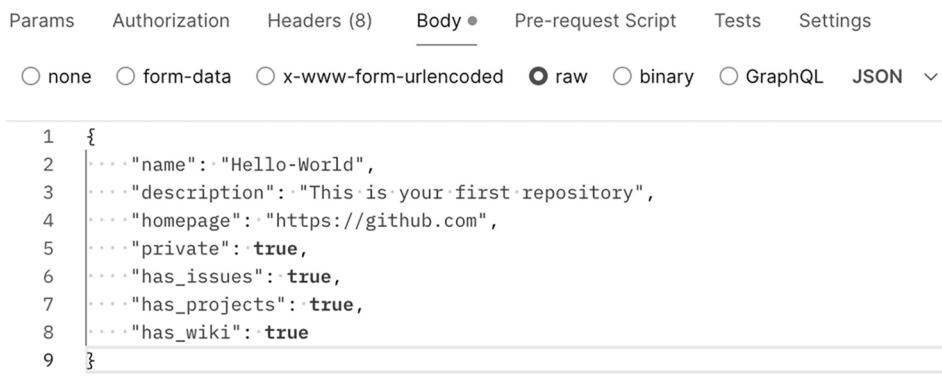
Slika 1.8: *Lista zaglavlja*

Možete modifikovati automatski generisana zaglavlja i dodavati nova, po potrebi. U narednim poglavljima ću detaljnije objasniti kako zaglavlja funkcionišu i kako da ih koristite, pa sad nemojte previše razmišljati o njima. Sada sam ih pomenuo samo da biste upoznali terminologiju. Usmerićemo pažnju na telo API zahteva.

API telo

Ako želite da kreirate ili izmenite resurse pomoću API interfejsa, serveru su potrebne određene informacije o osobinama resursa koje želite. Ta vrsta informacija se obično određuje u **telu** zahteva.

Telo zahteva ima mnogo oblika. Ako kliknete na karticu **Body** u Postman zahtevu, videćete različite vrste podataka koje možete poslati. Možete poslati podatke obrasca, šifrovane podatke obrasca, neobrađene podatke, binarne podatke, pa čak i GraphQL podatke. Kao što možete zamisliti, postoji mnogo detalja prilikom slanja podataka u telu zahteva. Na primer, ako biste pokušali da napravite novo spremište pomoću GitHub API interfejsa, možda biste uneli telo koje izgleda ovako:



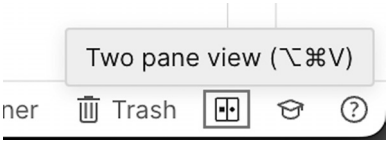
Slika 1.9: *Telo zahteva*

Uglavnom, `GET` zahtevi ne zahtevaju da navedete telo. Druge vrste zahteva, kao što su `POST` i `PUT`, koje to zahtevaju, često zahtevaju neku vrstu autorizacije, jer vam omogućavaju da izmenite podatke. Više ćemo naučiti o autorizaciji u *Poglavlju 5, Koncept opcija autorizacije*. Kada budete mogli da autorizujete zahteve, biće mnogo više primera stavki koje biste mogli da navedete u telu API zahteva.

API odgovor

Do sada smo mnogo vremena posvetili priči o raznim delovima API zahteva, a postoji jedna veoma važna stvar koju sam malo ignorisao. API interfejs je dvosmerna ulica. Šalje podatke serveru u zahtevu, ali onda server obrađuje taj zahtev i šalje nazad odgovor.

Podrazumevani prikaz koji Postman koristi je odgovor na dnu stranice zahteva. Takođe, možete izmeniti prikaz da vidite zahtev i odgovor u panelima jedan pored drugog. Ako želite, možete promeniti prikaz klikom na ikonu **Two pane view** pri dnu ekrana aplikacije, kao što je prikazano na sledećem snimku ekrana:



Slika 1.10: *Promena prikaza*

Postoji nekoliko različitih vidova odgovora. Najočigledniji je telo odgovora. Uglavnom se tu nalazi većina informacija koje tražite. U zahtevima za GitHub spremišta koja ste napravili, liste spremišta će se pojaviti u telu odgovora i Postman će ih prikazati u toj kartici.

API odgovor može sadržati i nekoliko drugih stvari, kao što su kolačići i zaglavlja. To mogu biti veoma važni tragovi onoga šta se dešava kada testirate ili kreirate API interfejse, o kojima će još biti reči u ovoj knjizi.

Obradili smo mnogo materijala, kada je u pitanju funkcionisanje API zahteva. Videli smo da API zahtev može da sadrži više različitih delova. Ovi jednostavni delovi se spajaju u moćan alat. Sada imate predstavu o osnovnim delovima koji čine API poziv i kako da ih koristite u alatu Postman. Vreme je da naučite kako sve to služi za testiranje API interfejsa, ali pre nego što se u to upustimo, želim da se na trenutak zaustavimo i primenimo stečeno teorijsko znanje.

Učenje kroz praksu – pravljenje API poziva

Knjige su sjajan izvor za razvoj i učenje. Čitate ovu knjigu, pa mislim da nema potrebe da vas ubeđujem! Međutim, kad pročitate knjigu (ili čak tri ili četiri knjige) ne znači da ste savladali temu. Teorija je jedno, a praktična primena je nešto sasvim drugo. To su dve vrlo različite stvari i, kada čitate, vi osećate da poznajete temu, ali to je samo osećaj, a ne stvarnost.

Ako želite da to bude stvarnost i ne želite da ova knjiga bude samo još jedan deo teorijskog znanja unutar vaše glave, morate primeniti u praksi stvari koje učite.

Praktične vežbe možda usporavaju čitanje, ali ćete uz njih učiti brže i izvući više iz ove knjige. Ali, dosta priče. Hajde da pređemo na vežbe!

Da bih vam pomogao u sticanju praktičnog iskustva, napravio sam jednostavnu API aplikaciju sa kojom možete stupiti u interakciju. Imajte na umu da je ova aplikacija samo *igračka*, što znači da radi, neke stvari, jednostavnije nego *pravi* API interfejs. Međutim, nadam se da će biti korisna za teme ove knjige.

Podešavanje testiranja aplikacije

Da biste koristili ovu aplikaciju, potrebno je malo podešavanja. Postoje dve opcije. Možete pokrenuti verziju na GitPod platformi, ili je možete pokrenuti na svom kućnom računaru. Preporučujem GitPod, jer automatski postavlja sve zavisnosti. To možete uraditi na sledeći način:

1. Pre svega, potreban vam je GitHub nalog. Ako ga nemate, možete se prijaviti na adresi `https://github.com`.
2. Sa nalogom, idite na adresu `https://gitpod.io/#https://github.com/djwester/todo-list-testing` u svom pregledaču.
3. Kliknite na **Continue with GitHub**.
4. Prihvatite podrazumevana podešavanja radnog prostora i nastavite.
5. Kada se učitavanje završi, idite na terminal, ukucajte komandu `make run-dev` i pritisnite *Enter*.
6. Usluga bi trebala da se pokrene i trebalo bi da vidite poruku obaveštenja sa nekoliko opcija. Kliknite na opciju **Make Public**.
7. Kliknite na karticu **Ports** i videćete javnu URL adresu veb sajta za testiranje dostupnog za kopiranje.

Znajte da će GitPod zatvoriti ovaj veb sajt nakon nekoliko minuta neaktivnosti, pa ako ga niste koristili neko vreme i dobijete poruku o grešci prilikom pravljenja API poziva, možda ćete morati da ponovite postupak i ponovo pokrenete veb sajt. Takođe, svaki put kada ponovo pokrenete veb sajt, imaće malo drugačiju URL dresu, pa nemojte zaboraviti da ažurirate URL adresu svih API poziva koji upućuju na ovaj veb sajt.

Ako ipak želite da pokrenete veb sajt na svom kućnom računaru, ima nekoliko preduslova. Prvo, potrebno je da imate instaliran Python 3.11:

1. Idite na adresu `https://www.python.org/downloads/`.
2. Potrebno je da preuzmete Python 3.11, pa kliknite na link za platformu koju koristite, ispod dugmeta **Download Python**.

3. Na sledećoj stranici pronađite verziju 3.11 (podverzija nije bitna), preuzmite je i instalirajte.

Takođe ćete morati da instalirate menadžer paketa poetry. To možete uraditi pokretanjem sledeće komande u komandnoj liniji:

```
curl -sSL https://install.python-poetry.org | python3 -
```

Kada ispunite ove preduslove, potrebno je da preuzmete veb sajt sa GitHub spremišta. Uverite se da ste ispratili uputstva za instaliranje Git sistema sa adrese `https://github.com/git-guides/install-git`. Zatim možete napraviti kopiju GitHub spremišta tako što ćete otići u komandnu liniju, odrediti direktorijum u koji želite da ga preuzmete i pokrenuti sledeću komandu:

```
git clone https://github.com/djwester/todo-list-testing.git
```

Nakon preuzimanja, otvorite direktorijum:

```
cd todo-list-testing
```

Pokrenite aplikaciju pozivanjem komande `make run-dev`. Sada možete pristupiti aplikaciji na adresi `https://localhost:8000`.

Sada kada ste pokrenuli aplikaciju, naučićete da pošaljete poziv ka njoj.

Pozivanje testiranja aplikacije

Hajde da provežbamo deo teorije koju smo dosad prošli. Prvo, otvorite aplikaciju u svom veb pregledaču. Trebalo bi da vidite stranicu koja izgleda ovako:

Slika 1.11: *Aplikacija lista zadataka*

Upišite zadatak u polje i kliknite `Add` da biste ga dodali na listu. Trebalo bi da se pojavi na listi ispod polja. Sada kada na veb sajtu postoji nešto, hajde da pristupimo tome pomoću API interfejsa:

1. Kreirajte novu kolekciju u alatu Postman i nazovite je `TodoList`.
2. Dodajte zahtev u kolekciju i nazovite ga `Get Todo List Item`.
3. U polje za URL adresu unesite URL adresu veb sajta koji ste napravili, a zatim dodajte `/tasks` na kraj.
4. Pošaljite zahtev.

Trebalo bi da dobijete odgovor koji prikazuje stavku koju ste kreirali na veb sajtu.

Izazov

Dobili ste puno postupnih uputstava za pravljenje API poziva, a sada imamo izazov za vas. Želimo da pokušate to da uradite bez tačnih uputstava.

Dokumentacija za ovaj veb sajt liste zadataka kaže da postoji krajnja tačka koja izgleda ovako:

```
/tasks/{task_id}
```

Možete li pozvati tu krajnju tačku u alatu Postman za ovu stavku liste zadataka koju ste napravili? Ne zaboravite da vitičaste zagrade označavaju da je `task_id` parametar zahteva.

Koristiću ovaj veb sajt za još nekoliko primera i izazova, ali sada se slobodno igrajte sa API interfejsom i isprobajte neke od stvari koje smo učili.

Razmatranja za API testiranje

Počeli smo od mehanike pravljenja API zahteva, ali ovo je knjiga o testiranju API interfejsa, pa sada kada znate neke osnovne funkcionalnosti API zahteva, hajde da vidimo šta treba uzeti u obzir kada se testira API interfejs. Važan aspekt testiranja je istraživačko testiranje.

Početak istraživanja

Još uvek se jasno sećam kada sam prvi put video savremeni veb API interfejs u akciji. Kompanija za koju sam radio gradila je centralizovanu platformu za izveštavanje o svim automatizovanim testovima, a ja sam bio zadužen da pomognem u testiranju te platforme za izveštavanje. Jedan od ključnih aspekata ove platforme bila je mogućnost čitanja i manipulacije podacima putem veb API interfejsa. Čim sam počeo da testiram ovaj sistem, shvatio sam koliko je to moćan pristup.

Drugi deo mog posla u to vreme bio je rad na prilagođenom sistemu za automatizaciju testova. Ovaj sistem je bio prilično drugačiji od standardnijeg radnog okvira za automatizaciju testova koji su mnogi drugi u kompaniji koristili.

Međutim, činjenica da nova platforma za izveštavanje ima API interfejs značila je da moj prilagođeni sistem za automatizaciju testova može da unosi podatke u ovu platformu za izveštavanje, iako je radio veoma drugačije od drugih sistema za automatizaciju testova. Aplikacija za izveštavanje o testovima nije morala da zna ništa o tome kako moj sistem, ili bilo koji drugi, radi. Ovo je bio veliki pomak za mene i verovatno ključni korak na putu do pisanja ove knjige. Ipak, još nešto što sam primetio dok sam testirao ovaj sistem je to da je API interfejs imao mane i nedostatke.

Možda je primamljivo misliti da sva testiranja API interfejsa moraju biti programski napravljena, ali ja tvrdim da treba početi istraživanjem. Kada sam testirao API interfejs za tu platformu za izveštavanje o testovima, jedva sam znao kako da koristim API interfejs, a kamoli kako da automatizujem testove za njega, a ipak sam našao mnoge probleme koje smo uspeli da ispravimo. Ako želite da poboljšate kvalitet API interfejsa, morate razumeti šta on radi i kako funkcioniše. Morate ga istražiti.

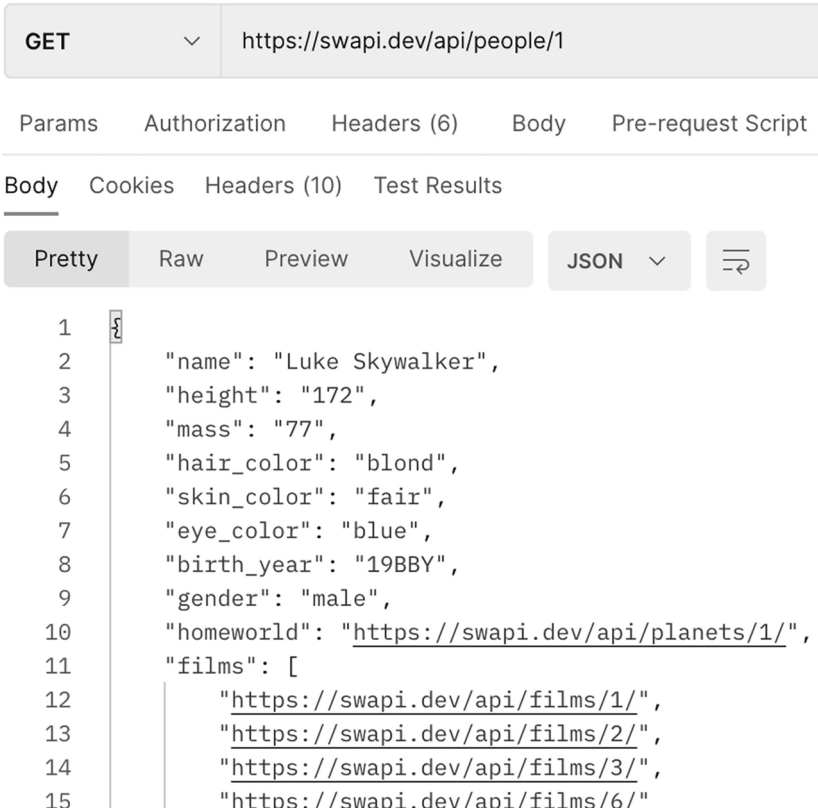
Ali kako to učiniti?

Srećom, Postman je jedan od najboljih alata za to. Sa alatom Postman možete lako isprobati mnoge različite krajnje tačke i upite, i možete dobiti trenutne povratne informacije. Postman olakšava igranje sa API interfejsom i prelazak sa jednog mesta na drugo. Istraživanje podrazumeva praćenje tragova koji su vam pred nosom. Kad dobijete rezultate od zahteva, razmišljate o pitanjima koja ti rezultati donose i pokušavate da odgovorite narednim pozivima. Sve je to jednostavno sa alatom Postman. Radi ilustracije, proćićemo studiju slučaja kratke istraživačke sesije pomoću alata Postman.

Za ovaj deo koristiću `https://swapi.dev/` API. Ovo je zabavan mali API interfejs koji izlaže podatke o filmovima iz serijala Ratovi zvezda. Ne brinite ako niste ljubitelj serijala Ratovi zvezda. Predznanje nije neophodno!

Studija istraživačkog slučaja

Hajde da probamo sledeće. Prvo, kreirajte novu kolekciju pod nazivom *Star Wars API* i dodajte zahtev pod nazivom *Get People*. Unesite `https://swapi.dev/api/people/1/` u URL polje i pošaljite taj zahtev. Trebalo bi da dobijete odgovor sa nekim podacima o liku *Luke Skywalker*:



Slika 1.12: *Odgovor na zahtev Star Wars API*

U odgovoru su linkovi ka drugim delovima API interfejsa. Odgovor koji sadrži linkove ka drugim relevantnim resursima poznat je kao **Hypermedia API** interfejs. Prvi link na listi filmova će nam dati informacije o filmu sa ID brojem 1. Pošto istražujemo, hajde da pogledamo taj link i vidimo šta radi. Možete samo kliknuti na njega i Postman će otvoriti novu karticu sa zahtevima sa već učitanim linkom. U ovom trenutku, znate šta treba da uradite: samo kliknite na **Send** i Postman će vam dati podatke o tom prvom filmu. Na spisku likova tog filma ćete videti da postoji više različitih likova, uključujući, naravno, link do `/people/1`.

To izaziva zaključak da postoji još nešto za proveru u `/people` API interfejsu, pa hajde da se vratimo i istražimo taj deo malo bolje. Kliknite na karticu **Get People** da se vratite na taj zahtev i promenite URL adresu da uklonite `/1` sa kraja. Sada možete kliknuti na **Send** da pošaljete zahtev na krajnju tačku, `https://swapi.dev/api/people`. Ovaj odgovor sadrži listu različitih ljudi u bazi podataka. Možete videti da na vrhu odgovora stoji broj 82.

Mi smo u režimu istraživanja, pa postavljamo pitanje: „Šta bi se desilo ako pokušam da zatražim osobu pod rednim brojem 83?" Ovaj API odgovor ukazuje da postoje samo 82 osobe, pa možda nešto neće biti u redu ako pokušamo da dobijemo osobu koja je van okvira ovog skupa podataka. Da bismo to proverili, dodajte `/83` na kraj URL adrese i ponovo kliknite na **Send**. Zanimljivo (ako se API interfejs nije promenio od kada sam ovo napisao), dobijamo podatke za lika iz filma Ratovi zvezda. Izgleda kao da je brojanje pogrešno, ili je možda neki lik uklonjen u nekom trenutku. Verovatno smo upravo naišli na grešku!

U svakom slučaju, ilustrovali smo moć istraživanja. Kasnije će doći na red neke moćne funkcije alata Postman, za automatizaciju testiranja, ali nemojte prevideti očigledno. API testiranje je testiranje. Kada testiramo, pokušavamo da pronađemo nove informacije ili probleme koji su mogli da promaknu. Ako odmah pređemo na automatizaciju testiranja, mogli bismo da propustimo neke važne stvari. Što pre – to bolje, odvojite vreme da istražite i upoznate API interfejse.

Istraživanje je ključni deo svakog testiranja, ali podrazumeva mnogo više od pukog postavljanja raznih zahteva u aplikaciji. Dobro testiranje, takođe, zahteva sposobnost povezivanja posla i poslovne vrednosti.

Potraga za poslovnim problemima

Kada razmatrate testiranje i dizajn API interfejsa, važno je uzeti u obzir poslovni problem koji API interfejs rešava. API interfejs nije u vakuumu. On je tu da pomogne kompaniji da zadovolji poslovne potrebe. Poznavanje tih potreba pomoći će vam da usmerite pristup i strategije testiranja. Na primer, API interfejs za internu upotrebu vrši različite funkcije od javnog API interfejsa, kom pristupaju klijenti, spolja.

Kada testirate API interfejs, tražite poslovne probleme. Ako pronađete probleme koji sprečavaju API interfejs da radi ono što je potrebno, otkrivaćete vredne probleme i unaprediti kvalitet aplikacije. Nisu sve greške jednako važne.

Čudni pokušaji

Neće svaki korisnik koristiti API interfejs na najbolji mogući način, za koji je API interfejs napisan. Svi smo ograničeni sopstvenom perspektivom, a teško je ući u tuđi um i videti stvari iz tuđe perspektive. Ne možemo pretpostaviti svaki mogući postupak korisnika našeg sistema, ali postoje strategije koje vam mogu pomoći da sagledate problem iz više uglova. Pokušajte da uradite nešto jednostavno čudno ili neobično. Probajte različite ulaze i vidite šta se dešava. Eksperimentišite sa stvarima za koje se čini da ih ne bi trebalo dirati. Radite stvari koje vam izgledaju čudno. Uglavnom se neće ništa dogoditi, ali povremeno će se desiti nešto zanimljivo, što vam inače ne bi palo na pamet.

Testiranje nije besmisleno ponavljanje istih test slučajeva. Koristite maštu. Pokušajte neobične i zanimljive stvari. Vidite šta možete da saznate. Svrha ove knjige je da naučite da koristite novi alat. Činjenica da ste kupili ovu knjigu govori da želite da učite. Prenesite taj stav u vaše testiranje. Pokušajte nešto neobično da vidite šta će se desiti.

Naravno, testiranje je mnogo više od ovih nekoliko ideja koja sam naveo. Međutim, to su važni, osnovni principi testiranja. Opisaću mnogo različitih načina upotrebe alata Postman u ovoj knjizi, a većina će biti primeri primene ovih strategija. Pre nego što počnemo da koristimo Postman, sledi kratak pregled različitih tipova API interfejsa na koje ćete naići.

Različiti tipovi API interfejsa

Postoji nekoliko tipova API interfejsa koji se često koriste na internetu. Pre nego što uronimo u detalje upotrebe alata Postman, valja upoznati različite tipove API interfejsa i znati po čemu se razlikuju i kako se testiraju. U sledećim odeljcima su kratki opisi tri tipa API interfejsa koje ćete najčešće viđati na internetu.

REST API interfejsi

Počećemo od verovatno najčešće korišćenog tipa API interfejsa na modernom vebu, a to je **RESTful API**. REST je skraćenica za **Prenos stanja prikaza** i odnosi se na arhitektonski stil koji ukazuje na to kako treba da kreirate API interfejse. Neću ulaziti u detalje o svojstvima koje bi RESTful API interfejs trebalo da ima (možete ih naći na Vikipediji, ako želite, na adresi `https://en.wikipedia.org/wiki/Representational_state_transfer`), ali postoji nekoliko nagoveštaja koji vam mogu pokazati da verovatno testirate RESTful API interfejs.

Pošto se RESTful API interfejsi zasnivaju na skupu smernica, nisu svi isti. Ne postoji zvaničan standard koji definiše tačne specifikacije koje odgovor mora da ispuni. To znači da mnogi API interfejsi za koje mislimo da su RESTful ne prate strogo sve REST smernice. REST je, generalno, fleksibilniji nego protokol zasnovan na standardima, kao što je **SOAP** (obrađen u sledećem odeljku), ali to omogućava mnogo raznovrsnosti u definisanju i upotrebi REST API interfejsa.

Dakle, kako da znate da li je API interfejs RESTful?

Prvo, koje vrste zahteva su obično definisane? Većina REST API interfejsa ima pozive `GET`, `POST`, `PUT` i `DELETE`, i možda još nekoliko drugih. U zavisnosti od potreba, API interfejs možda neće koristiti sve ove akcije, ali to su one uobičajene, koje ćete verovatno videti ako API interfejs jeste RESTful.

Još jedan nagoveštaj je u vrstama zahteva ili odgovora koje API interfejs dozvoljava. Često će REST API interfejs koristiti JSON podatke u svojim odgovorima (iako mogu koristiti tekst ili čak XML). Generalno govoreći, ako podaci u odgovorima i zahtevima API interfejsa nisu XML, postoji velika verovatnoća da imate posla sa nekom vrstom API interfejsa u REST stilu. Postoji mnogo primera REST API interfejsa na vebu, a zapravo, svi koje smo do sada pomenuli u ovoj knjizi jesu RESTful.

SOAP API interfejsi

Pre nego što se pojavio REST protokol, bio je SOAP protokol. SOAP protokol je nastao mnogo pre nego što je Roj Filding osmislio koncept REST API interfejsa. Sada se ne koristi toliko za veb (posebno za manje aplikacije), ali godinama je bio podrazumevan način za pravljenje API interfejsa, pa još uvek ima mnogo SOAP API interfejsa.

SOAP je, zapravo, protokol sa **W3C** definicijom standarda. To znači da je njegova upotreba mnogo strože definisana u poređenju sa REST protokolom, koji je arhitektonska smernica, za razliku od strogo definisanog protokola.

Ako marite za lagano štivo, pogledajte W3 primer SOAP protokola (`https://www.w3.org/TR/soap12-part0/`). Tvrdi da je to:

> *nenormativni dokument namenjen da pruži lako razumljiv vodič o karakteristikama SOAP verzije 1.2.*

Dobro, štivo možda nije toliko lako kao što sam rekao, ali kad vidite neke od tih primera lakše ćete shvatiti zašto su REST API interfejsi postali toliko popularni. SOAP API interfejs zahteva visoko strukturiranu XML poruku da bi poslao zahtev. Pošto su izgrađeni u XML jeziku, ovi zahtevi nisu toliko čitljivi za ljude i njihovo kreiranje je vrlo kompleksno. Naravno, postoje mnogi alati (kao što je SoapUI) koji mogu pomoći, ali generalno, SOAP API interfejsi su previše složeni za početnike. Potrebno je da imate više informacija (kao što je struktura poruke).

Ali kako znati da API interfejs jeste SOAP API interfejs?

Najvažnije pravilo je da li zahtev mora biti u strukturiranom XML formatu? Ako je odgovor da, to je SOAP API interfejs. Pošto ovi API interfejsi obavezno prate W3C specifikaciju, moraju koristiti XML i moraju navesti stvari kao što su `env:Envelope` XML elementi. Ako API interfejs zahteva da se navede XML, i taj XML sadrži **Envelope** element, u pitanju je SOAP API interfejs.

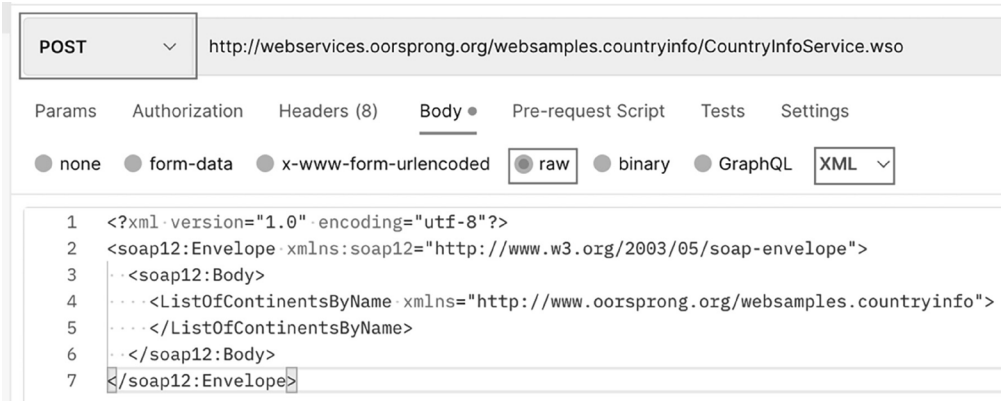
Još jedan pokazatelj SOAP API interfejsa je prisustvo **datoteke opisa veb usluge** (**WSDL**). Ova datoteka služi za definisanje ugovora o tome kako API interfejs funkcioniše. Namenjena je da je korisnici koriste programski, kao pomoć za pozivanje API interfejsa. Nema je svaki SOAP API interfejs, ali ako postoji, možete biti sigurni da je pred vama SOAP API interfejs.

Primer SOAP API interfejsa

Pogledajmo primer kako izgleda pozivanje SOAP API interfejsa. Ovo je malo teže od slanja GET zahteva na krajnju tačku, ali Postman će nam pomoći. U ovom primeru koristiću uslugu za informacije o zemljama da bih dobio listu kontinenata po imenu. Početna stranica za tu uslugu je: `http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso`. Da bismo pozvali ovaj API interfejs u alatu Postman, moraćemo da podesimo nekoliko stavki. Naravno, potrebno je kreirati zahtev u alatu Postman. Međutim, u ovom slučaju, umesto da metod zahteva bude `GET`, potrebno je postaviti metod zahteva na `POST` i zatim uneti URL adresu navedenu iznad. SOAP zahtevi se obično šalju `POST` metodom umesto `GET` metodom, jer moraju poslati XML podatke u telu zahteva. Obično ne šaljete podatke u telu GET zahteva, pa većina SOAP usluga zahteva da se zahtevi šalju `POST` protokolom.

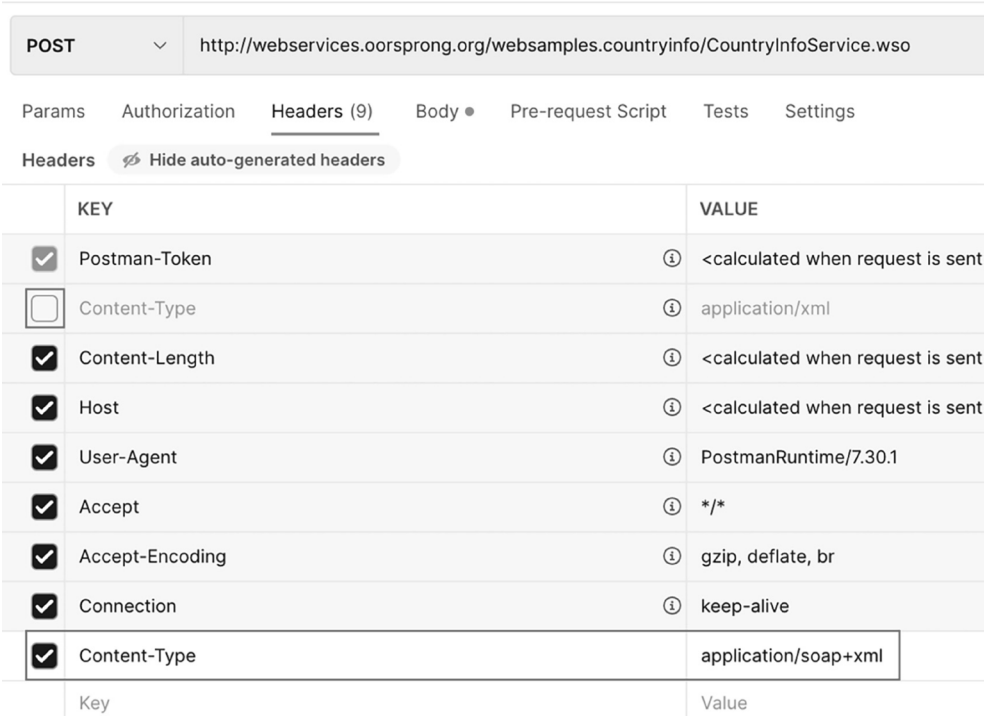
Međutim, **nemojte** još da kliknete na **Send**. Pošto je to SOAP API interfejs, potrebno je poslati i neke XML informacije. Želimo da dobijemo listu kontinenata po imenima, pa ako odete na veb stranicu `CountryInfoServices`, možete kliknuti na prvu vezu na listi, koja će vam pokazati XML definicije za tu operaciju. Koristite primer za SOAP 1.2 na toj stranici i kopirajte XML za to.

U alatu Postman, potrebno je postaviti tip ulaznog tela na **raw**, izabrati **XML** iz padajućeg menija, a zatim uneti kopirane podatke u Envelope elemente koje ste kopirali sa stranice za dokumentaciju. Trebalo bi da izgleda ovako:



Slika 1.13: *SOAP API informacije*

Specifično za ovaj API interfejs, takođe treba da izmenimo zaglavlje `Content-Type`. Idite na tab **Headers** i kliknite na dugme **hidden**. Videćete da je Postman automatski dodao Content-Type zaglavlje sa vrednošću `application/xml`. Međutim, ovaj API interfejs zahteva da se tip sadržaja postavi na `application/soap+xml`. Ne možemo direktno izmeniti automatski kreirano zaglavlje, ali možemo dodati neko koje će ga zameniti. Na dnu liste, unesite `Content-Type` u polje **Key** i postavite vrednost na `application/soap+xml`. Postman će automatski precrtati automatski generisano zaglavlje, što znači da ga neće koristiti, već će umesto toga koristiti ono koje ste naveli. Za svaki slučaj, možete ga i ukloniti sa liste.



Slika 1.14: *Zaglavlje Content-Type postavljeno na application/soap+xml*

Sada ste konačno spremni da kliknete na **Send**. Trebalo bi da dobijete listu kontinenata. Kao što vidite, pozivanje SOAP API interfejsa je mnogo komplikovanije. Ova složenost je jedan od faktora pada popularnosti ovih API interfejsa, a takođe ukazuje na tip API interfejsa. REST API interfejsi, naravno, mogu takođe imati složena tela zahteva, ali ako zahtev mora da bude u XML formatu, kao i prisustvo **Envelope** elementa ukazuju na SOAP API interfejs.

GraphQL API interfejsi

SOAP protokol je stariji od REST protokola, a na mnogo načina, REST protokol je dizajniran da reši neke nedostatke SOAP protokola. Naravno, u svetu softvera unapređenja su stalna, tako da sada imamo tip API interfejsa poznat kao GraphQL. GraphQL je jezik za upite koji je dizajniran da reši neke nedostatke koje ima REST API interfejs. RESTful API interfejs nije svestan koje tačne informacije možda tražite, pa kada pozovete REST API krajnju tačku, on vam daje sve informacije koje ima. To može da znači da dobijate i informacije koje vam nisu potrebne, ili može da znači da ne dobijate sve potrebne informacije i da morate pozvati više krajnjih tačaka da biste dobili ono što želite. Bilo koji od ovih slučajeva vas usporava, a za velike aplikacije, sa velikim brojem korisnika, to je problem. GraphQL je dizajnirao Fejsbuk da bi rešio pomenute probleme.

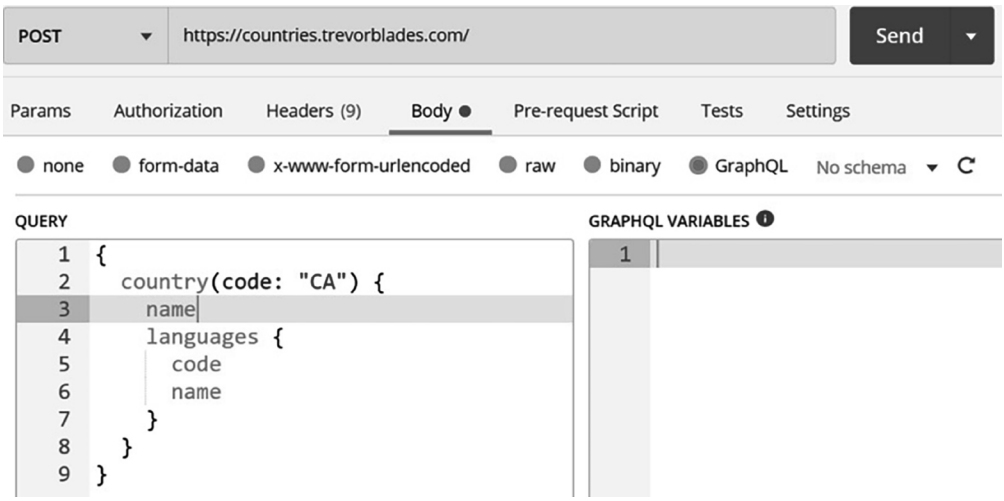
GraphQL je jezik za upite za API interfejse, pa zahteva od vas da navedete u upitu šta tražite. Sa REST API interfejsima, obično će vam biti potrebno samo da znate razne krajnje tačke da biste došli do informacija koje tražite, ali sa GraphQL API interfejsom, jedna krajnja tačka sadrži veći deo, ili sve informacije koje su vam potrebne, a na raspolaganju su vam i upiti da filtrirate te informacije samo na one koje vas interesuju. To znači da morate znati šemu ili strukturu podataka da biste znali kako pravilno da ih zahtevate, umesto da znate sve krajnje tačke.

Kako da znate da API interfejs jeste GraphQL API interfejs?
Pa, ako dokumentacija govori o vrstama upita koje morate pisati, gotovo sigurno je GraphQL API interfejs. Na neki način, GraphQL API interfejs je sličan kao SOAP API interfejs, u smislu da usluzi morate reći koje vas informacije interesuju. Međutim, SOAP API interfejs će uvek koristiti XML i pratiti strogu definiciju u pozivima, dok je GraphQL API interfejs obično malo jednostavniji i nije definisan u XML formatu. Takođe, kod GraphQL API interfejsa, način na koji je šema definisana može da varira, jer se ne prati strogo definisan standard.

Primer GraphQL API interfejsa

Pogledajmo primer poziva GraphQL API interfejsa u realnoj situaciji, da bismo ga bolje razumeli. Ovaj primer koristi verziju API interfejsa `countries` koju možete naći na adresi `https://countries.trevorblades.com/`. Informacije o šemi možete naći na GitHub spremištu, na adresi: `https://github.com/trevorblades/countries`. Možete kreirati upite na ponuđenom okruženju, ali za ovaj primer, pogledajmo kako da to postavimo u alatu Postman.

Slično kao i kod poziva SOAP API interfejsa, moraćemo da odredimo uslugu koju želimo i da uradimo `POST` zahtev umesto `GET` zahteva. GraphQL upiti se mogu raditi kao `GET`, ali je mnogo lakše navesti upit u telu `POST` zahteva, pa se većina GraphQL API poziva šalje tim metodom. Zapravo, kada odaberete GraphQL opciju za telo vašeg zahteva u alatu Postman, primetićete da se tip automatski menja u `POST`, jer se očekuje da je to potrebno za GraphQL zahtev. Dakle, nastavite i uradite to; izaberite **GraphQL** opciju na kartici **Body** i unesite upit koji želite:



Slika 1.15: *GraphQL upit*

Dok unosite upit, možda ćete primetiti da Postman nudi opcije za automatsko popunjavanje. To je moguće jer je struktura podataka poznata. Postman će automatski pročitati GraphQL šemu umesto vas i iskoristiće je kao pomoć pri konstruisanju upita.

Kao što možete videti, u ovom primeru, zatražio sam ime i jezike Kanade. Kada sam naveo ove informacije, mogu da kliknem na **Send**, i dobiću nazad JSON sa imenom zemlje i spiskom službenih jezika. Ako želim dodatne informacije (recimo, ime glavnog grada), mogao bih modifikovati upit da uključim zahtev za te informacije i ponovo ga poslati na istu krajnju tačku, i dobio bih novi skup informacija koje sam zatražio.

U ovom trenutku, očigledno je da sam u mogućnosti da dam vrlo kratak uvod za svaki od ovih API interfejsa. Ako odaberete GraphQL ili SOAP API interfejse, možda ćete morati da izdvojite određeno vreme da biste ih bolje upoznali pre nego što nastavite da čitate ovu knjigu. Za većinu primera u ostatku ove knjige koristićemo RESTful API interfejs.

Međutim, koncepti API testiranja uglavnom su isti, bez obzira na tip API testiranja koji radite. Trebalo bi da budete u mogućnosti da iskoristite stvari koje ćete naučiti u ovoj knjizi i primenite ih, bez obzira na tip API interfejsa sa kojim radite svakodnevno.

Rezime

Zastanimo na trenutak da razmislimo o svemu što smo prošli u ovom poglavlju. Instalirali ste Postman i već napravili nekoliko API zahteva. Naučili ste kako API zahtevi funkcionišu i kako se pozivaju. Objasnio sam vam osnovne aspekte testiranja i strategije koje možete da primenite u svakodnevnom radu. Takođe ste napravili pozive GraphQL, SOAP i REST API interfejsima i naučili dobar deo API terminologije.

Sada imate čvrstu osnovu na koju možete da se oslonite za ostatak ove knjige. Uvešću vas duboko u mnoge teme vezane za testiranje i dizajn API interfejsa i pomoći vam da izvučete maksimum iz alata Postman. Međutim, da biste izvukli najviše iz ove knjige i da se ne biste osećali frustrirano, bilo bi dobro da se uverite da razumete teme ovog poglavlja.

Zastanite na trenutak i postavite sebi sledeća pitanja:

- Da li bih mogao/la komforno da pročitam članak o testiranju API interfejsa? Da li bih mogao/la da pratim terminologiju koja se koristi?
- Koje su osnovne strategije i pristupi za testiranje API interfejsa?
- Ako bih dobio/la API krajnju tačku, da li bih mogao/la da pošaljem zahtev ka njoj u alatu Postman? Šta bih morao/la da uradim da pošaljem taj zahtev?
- Ako bih dobio/la neku API dokumentaciju, da li bih mogao/la da shvatim o kakvom se API interfejsu radi i da šaljem zahteve ka njemu?

Ako znate odgovore na ova pitanja, sigurno imate osnovu koja vam je potrebna da nastavite da čitate ovu knjigu. Ako niste potpuno sigurni, možda bi trebalo da se vratite na relevantne odeljke u ovom poglavlju i uverite se da ste ih dobro razumeli.

Već ste mnogo naučili! U sledećem poglavlju ćete upoznati neke od principa dizajna API interfejsa i naučiti da ih primenite u praksi, prilikom kreiranja API interfejsa, pomoću alata Postman.

Pridružite se našoj zajednici na platformi Discord

Pridružite se našoj zajednici na platformi Discord, za diskusije sa autorom i drugim čitaocima:

`https://discord.com/invite/nEN6EBYPq9`