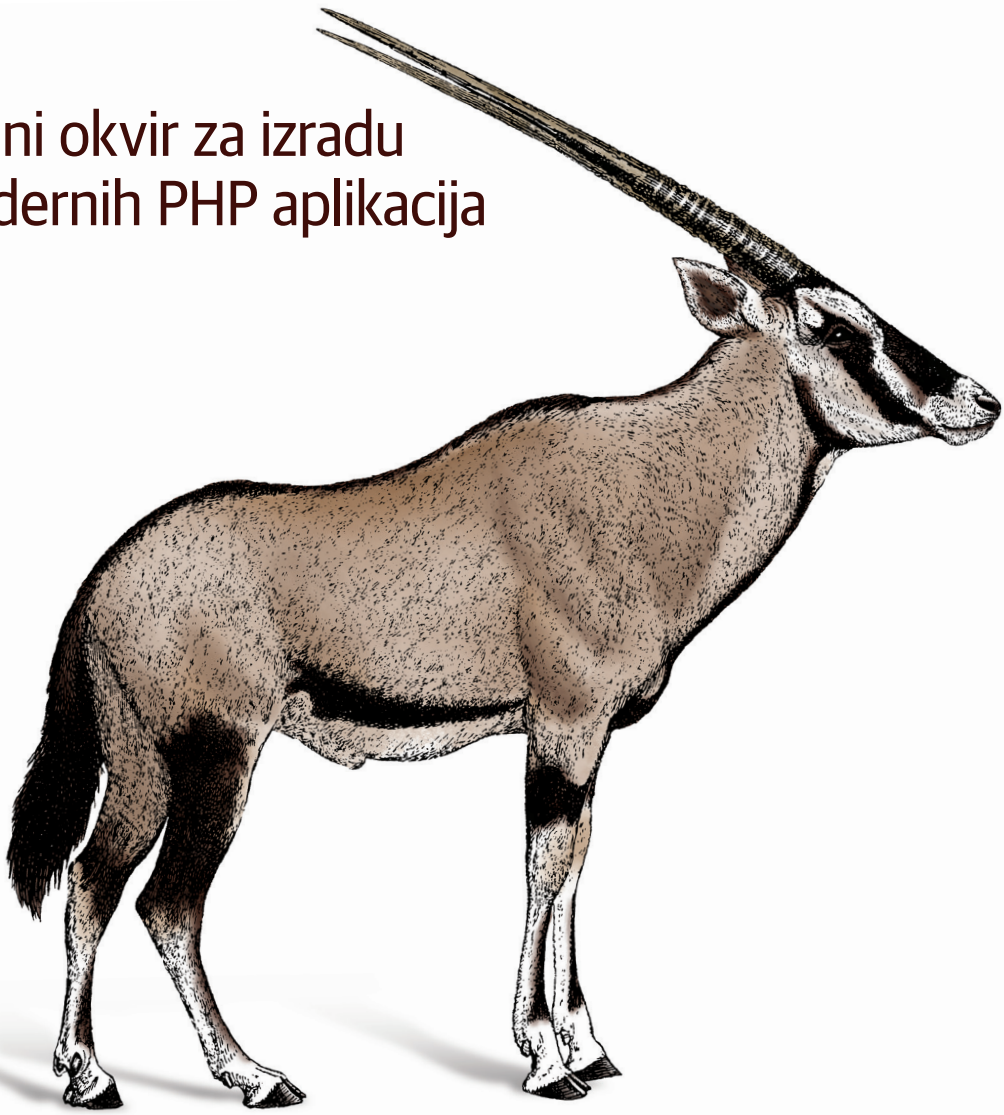
O'REILLY®

Prevod drugog izdanja

Edicija: **Up & Running**

Laravel

Radni okvir za izradu modernih PHP aplikacija



kompjuter biblioteka

Matt Stauffer

Laravel

Matt Stauffer

kompjuter biblioteka

O'REILLY®

Izdavač:
kompjuter biblioteka

Obalskih radnika 4a, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autor: Matt Stauffer

Prevod: Biljana Tešić

Lektura: Miloš Jevtović

Slog : Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Pekograf“ Zemun

Tiraž: 500

Godina izdanja: 2019.

Broj knjige: 517

Izdanje: Prvo

ISBN: 978-86-7310-540-6

Laravel: Up & Running

by Matt Stauffer
Published by O'Reilly Media, Inc

Cover Image by Rebecca Demarest

PREDGOVOR

Priča o tome kako sam počeo da koristim Laravel je uobičajena – pisao sam PHP godinama, ali sam bio na putu „ka vratima“, prateći snagu Railsa i drugih modernih veb okvira. Rails je imao aktivnu zajednicu, savršenu kombinaciju dogmatičnih podrazumevanih parametara i fleksibilnosti, ali i snagu Ruby Gemsa da bi bio iskorišćen unapred pripremljeni zajednički kod.

Nešto me je sprečavalo da „napustim brod“ i bilo mi je drago kada sam pronašao Laravel. On ima sve ono što me je privuklo Railsu, ali on nije samo Rails klon; on je inovativni okvir sa neverovatnom dokumentacijom, srdačnom zajednicom i jasnim uticajima mnogih jezika i radnih okvira.

Od tog dana uspeo sam da podelim sa drugima ono što sam naučio o Laravelu kroz pisanje blogova, podkasting i govore na konferencijama. Napisao sam desetine aplikacija u Laravelu za posao i sporedne projekte i sreo sam hiljade Laravelovih programera na Internetu i lično. Imam mnoštvo alatki u svom razvojnom kompletu, ali, iskreno, najsrećniji sam kada sednem ispred komandne linije i ukucam laravel new projectName.

ČEMU JE POSVEĆENA OVA KNJIGA

Ovo nije prva knjiga o Laravelu, a neće biti ni poslednja. Ne nameravam da ovo bude knjiga koja „pokriva“ svaku liniju koda ili svaki obrazac implementacije. Ne želim da ona zastari kada se objavi nova verzija Laravela. Umesto toga, ona je prvenstveno namenjena da obezbedi programerima pregled i konkretne primere na visokom nivou da bi naučili šta je potrebno da rade u Laravel bazi koda pomoću svake Laravel funkcije i podsistema. Umesto da „preslikavam“ dokumente, želim da čitaocima pomognem da shvate osnovne koncepte Laravela.

Laravel je moćan i fleksibilan PHP radni okvir. Ima naprednu zajednicu i širok ekosistem alatki, a, kao rezultat toga, njegova „privlačnost“ i domet se povećavaju. Ova knjiga je namenjena programerima koji već znaju kako da kreiraju veb stranice i aplikacije, pa žele da nauče kako to dobro da rade u Laravelu.

Dokumentacija za Laravel je detaljna i odlična. Ako mislite da neka određena tema nije „pokrivena“ dovoljno detaljno, preporučujem vam da pogledate dokumentaciju na Internetu.

Mislim da ćete u ovoj knjizi pronaći dobar balans između uvođenja na visokom nivou i konkretne primene, a na kraju bi trebalo da znate da napišete čitavu aplikaciju u Laravelu „od nule“. Ako sam dobro obavio svoj posao, bićete uzbuđeni da isprobate pisanje aplikacije.

KOME JE NAMENJENA OVA KNJIGA

U ovoj knjizi pretpostavljamo da poznajete osnovnu praksu objektno-orijentisanog programiranja, PHP (ili bar opštu sintaksu C porodice jezika), osnovne koncepte šablona Model -View-Controller (MVC) i izradu šablona. Ako nikada ranije niste kreirali veb sajt, možda je ovo previše za vas. Međutim, ako imate određeno programersko iskustvo, ne morate znati ništa o Laravelu pre nego što pročitate ovu knjigu – „pokrićemo“ sve što treba da znate, počev od najjednostavnijeg „Hello, world!“ programa.

Laravel može biti pokrenut na svakom operativnom sistemu, ali će biti nekih komandi „ljuske“ (shell) u knjizi koje je najlakše pokrenuti na operativnim sistemima Linux i macOS. Korisnici Windowsa će možda imati poteškoća sa ovim komandama i modernim PHP razvojem, ali, ako prate uputstva za pokretanje Homesteada (Linux virtuelne mašine), moći će da pokrenu sve komande iz te mašine.

KAKO JE OVA KNJIGA STRUKTURISANA

Ova knjiga je strukturisana prema hronološkom redosledu – ako izrađujete vašu prvu veb aplikaciju pomoću Laravela, početna poglavlja „pokrivaju“ osnovne komponente koje će vam biti neophodne da biste započeli rad, a kasnije poglavlja „pokrivaju“ funkcije koje su složenije, odnosno, koje su specijalizovane.

Svaki odeljak ove knjige se može čitati zasebno, ali za nekoga ko je nov u ovom radnom okviru strukturisao sam poglavlja tako da je zaista veoma razumno čitati od početka do kraja.

Gde god je to moguće, svako poglavlje će na kraju imati dva odeljka: Testiranje i TL;DR. Ako neko ne zna, TL;DR (too long; didn’t read) se odnosi na situacije kada neko ne želi da pročita određeni tekst zbog njegove dužine. U ova dva odeljka će biti prikazano kako se pišu testovi za funkcije koje su „pokrivene“ u svakom poglavlju i dat je pregled onoga što je „pokriveno“.

Knjiga je napisana za Laravel 5.8, ali će „pokriti“ funkcije i sintakse za Laravel 5.1.

O DRUGOM IZDANJU

Prvo izdanje knjige je objavljeno u novembru 2016. godine i „pokriva“ Laravel verzije od 5.1 do 5.3. U drugom izdanju je dodata „pokrivenost“ verzija od 5.4 do 5.8, Laravel Dusk i Horizon, a dodato je i 18. poglavlje o resursima zajednice i drugim sporednim Laravel paketima koji nisu „pokriveni“ u prethodnih 17 poglavlja.

KONVENCIJE

Konvencije korišćene u ovoj knjizi

Sledeće tipografske konvencije koriste se u ovoj knjizi:

Italic font

Ukazuje na nove termine, URL adrese, adrese e-pošte, nazive datoteka i ekstenzije datoteka.

`Font` sa konstantnom širinom svih karaktera

Koristi se za kod programa i u pasusima koji se odnose na programske elemente, kao što su nazivi promenljivih ili funkcija, baze podataka, tipovi podataka, okruženje promenljivih, iskazi i rezervisane reči.

`Podebljani font` sa konstantnom širinom svih karaktera

Ukazuje na komande ili drugi tekst koji treba da unese korisnik.

`Italic font` sa konstantnom širinom svih karaktera

Ukazuje na tekst koji treba zameniti korisničkim vrednostima ili vrednostima koje su utvrđene prema kontekstu.

{*Italic font u zagradama*}

Ukazuje na nazive datoteka ili putanje datoteka koje treba zameniti korisničkim vrednostima ili vrednostima koje su utvrđene prema kontekstu.

Ovaj element označava savet ili predlog.

Ovaj element označava generalnu napomenu.

Ovaj element označava upozorenje ili opomenu.

5.x Pošto ova knjiga „pokriva“ Laravel od verzije 5.1 do 5.8, videćete u njoj oznake koje ukazuju na komentare koji su specifični za određenu verziju. Uopšteno rečeno, oznaka ukazuje na verziju Laravela u koju je uvedena funkcija (tako da ćete videti 5.3 pored oznake koja je dostupna samo u Laravel 5.3 i novijoj verziji).

KAKO DA SA NAMA KONTAKTIRATE

Molimo vas da komentare i pitanja u vezi ove knjige šaljete izdavaču na sledeću adresu:

O’Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

800-998-9938 (za SAD ili Kanadu)

707-829-0515 (internacionalno ili lokalno)

707-829-0104 (faks)

Postoji veb prezentacija za ovu knjigu, gde su navedene greške, primeri i sve ostale dodatne informacije. Ovoj stranici možete da pristupite na adresi: http://bit.ly/laravel-up-and-running-2e.

Da biste komentarisali ili postavljali tehnička pitanja o ovoj knjizi, šaljite poruke elektronskom poštom na adresu bookquestions@oreilly.com.

Više informacija o našim knjigama, tečajevima, konferencijama i vestima možete pronaći na veb prezentaciji koja se nalazi na adresi http://www.oreilly.com.

Pronađite nas na Facebooku: http://facebook.com/oreilly

Pronađite nas na Twitteru: http://twitter.com/oreillymedia

Pronađite nas na YouTubeu: http://www.youtube.com/oreillymedia

ZAHVALNICE ZA PRVO IZDANJE

Ove knjige ne bi bilo bez velikodušne podrške moje divne supruge Tereve i bez razumevanja („tatinog druga u pisanju“) mog sina Malachija. Iako ona nije toga bila baš svesna, moja ćerka Mia je bila tu tokom skoro čitavog pisanja knjige, tako da je ova knjiga posvećena celoj porodici. Bilo je mnogo, mnogo dugih večernjih sati i odlazaka u Starbucks tokom vikenda koja su me razdvajali od moje porodice, pa ne mogu biti zahvalniji nego što jesam za njihovu podršku, ali i za njihovo prisustvo koje čini moj život divnim.

Osim toga, čitava „Tighten“ porodica me je podržavala i ohrabrivala tokom pisanja knjige, a nekoliko kolega je, čak, i uređivalo primere koda (na primer, izuzetan urednik Keith Damiani) i pomagalo u izazovnim kodovima (Adam Wathan, „kralj“ šablona Collection Pipeline). Dan Sheetz, moj partner u „Tighten zločinu“ (poduhvatu pisanja ove knjige), bio je dovoljno velikodušan da me nadgleda tokom dugih radnih sati koje sam proveo pišući ovu knjigu (pružao mi je podršku i ohrabrivao me je), a Dave Hicking, naš operativni menadžer, pomogao mi je da uskladim radne obaveze sa pisanjem knjige.

Taylor Otwell zaslužuje zahvalnost što je kreirao Laravel, a samim tim i što je omogućio toliko poslova i pomogao mnogim programerima da mnogo više zavole svoje živote. Zaslužuje zahvalnost, jer se fokusirao na to da progameri budu srećni i trudio se da saoseća sa njima i da izgradi pozitivnu i ohrabrujuću zajednicu. Zahvaljujem mu se što je bio ljubazan prijatelj koji me je ohrabrivao i koji je pun izazova.

Hvala Jeffreyu Wayu, jednom od najboljih učitelja na Internetu. On me je prvi „upoznao“ sa Laravelom i svaki dan „upoznaje“ sve više ljudi sa ovom platformom. On je fantastično ljudsko biće, koje sa ponosom mogu nazvati prijateljem. Hvala Jessu D’Amicu, Shawnu McCoolu, Ianu Landsmanu i Tayloru što su od samog početka u meni videli vrednog govornika na konferenciji i obezbedili mi platformu za podučavanje. Zahvaljujem se Dayleu Reesu zbog toga što mi je već na početku olakšao učenje Laravela.

Hvala svim osobama koje su uložile vreme i trud u pisanje postova na blogu o Laravelu od samog početka, a to su Eric Barnes, Chris Fidao, Matt Machuga, Jason Levis, Ryan Tablada, Dries Vints, Maks Surguy i još mnogo drugih.

Hvala i čitavoj zajednici prijatelja na Twitteru, IRC-u i Slacku koji su komunicirali sa mnom tokom godina. Voleo bih da mogu navesti svako ime, ali nekog bih zaboravio i onda bih se osećao užasno. Svi su briljantni i počastvovan sam što mogu redovno komunicirati sa njima.

Zahvaljujem se uredniku izdavačke kuće „O’Reilly“ Ally MacDonaldu i svim tehničkim urednicima: Keithu Damianiju, Michaelu Dyryndi, Adamu Fairholmu i Mylesu Hysonu.

Naravno, zahvaljujem se ostatku porodice i prijateljima koji su me podržavali direktno ili indirektno kroz ovaj proces – mojim roditeljima i rođacima, zajednici „Gainesville“, drugim vlasnicima kompanija i autorima, drugim govornicima na konferencijama i jedinstvenom DCB-u. Moram da prestanem da nabrajam, jer neću imati dovoljno prostora u knjizi, a na kraju ću se zahvaliti i baristima iz Starbucksa.

ZAHVALNICE ZA DRUGO IZDANJE

Drugo izdanje je veoma slično prvom, tako da sve prethodne zahvalnosti i dalje važe. Međutim, ovoga puta sam imao pomoć nekoliko novih ljudi. Tehnički lektori su bili Tate Penaranda, Andy Swick, Mohamed Said i Samantha Geitz, a nova urednica izdavačke kuće „O'Reilly" je Alicia Young, koja mi je pomogla u mnogim promenama u mom životu i Laravel zajednici tokom prošle godine. Matt Hacker iz „Atlas" tima odgovorio je na sva moja „glupa" pitanja o AsciiDoc formatiranju, uključujući i pitanja o iznenađujuće teškom formatiranju za metod __().

Ne bih uspeo da završim proces pisanja drugog izdanja bez pomoći i mog istraživačkog asistenta Wilbura Powerya. Wilbur je bio voljan da ispita evidencije izmena do kojih je došlo u poslednjih nekoliko godina, da izdvoji zahteve i obaveštenja i da uskladi svaku funkciju sa trenutnom strukturom knjige, pa je čak i testirao svaki primer koda za Laravel 5.7 (a kasnije i za 5.8) da bih mogao svoje ograničeno vreme i energiju da posvetim pisanju novih i ažuriranih segmenata.

Moja kćerka Mia sada je van majčinog stomaka. Dakle, mojoj listi izvora inspiracije dodaćemo i njenu radost, energiju, ljubav, ljupkost i avanturistički duh.

POGLAVLJE 1

Zašto Laravel?

U ranim danima dinamičnog Veba pisanje veb aplikacije izgledalo je mnogo drugačije nego danas. Programeri su tada bili odgovorni za pisanje koda ne samo za jedinstvenu poslovnu logiku naših aplikacija, već i za svaku komponentu koja je toliko uobičajena na svim sajtovima, kao što su autentifikacija korisnika, validacija unosa, pristup bazi podataka, izrada šablona i još mnogo štošta.

Danas programeri imaju desetine radnih okvira za razvoj aplikacija i hiljade lako dostupnih komponenata i biblioteka. Uobičajeno je da se, dok programeri uče jedan okvir, pojave već tri nova (i navodno bolja) koji će ga zameniti.

„Samo zato što radni okvir postoji“ može da bude valjano opravdanje za korišćenje, ali postoje bolji razlozi zbog kojih treba da izaberete određeni radni okvir ili da uopšte koristite radni okvir. Vredi postaviti pitanje: „Zašto radni okviri?“. Tačnije, zašto Laravel?

ZAŠTO KORISTITI RADNI OKVIR?

Lako je videti zašto je pogodno koristiti pojedinačne komponente ili pakete koji su dostupni PHP programerima. Kada se koriste paketi, neko drugi je odgovoran za razvoj i održavanje izolovanog dela koda koji ima dobro definisan zadatak, a teoretski ta osoba ima više vremena da stekne bolje razumevanje ove pojedinačne komponente, nego što vi imate vremena za to.

Radni okviri, kao što su Laravel, Simfony, Lumen i Slim, pripremaju za pakovanje kolekciju komponenata nezavisnih proizvođača, zajedno sa „lepkom“ prilagođenog radnog okvira, kao što su konfiguracione datoteke, dobavljači usluga, propisane strukture direktorijuma i aplikacije za pokretanje sistema.

Dakle, pogodnost upotrebe radnog okvira je u suštini da je neko doneo odluke ne samo o pojedinačnim komponentama, već i o tome *kako te komponente treba spojiti.*

„Sam ću izraditi radni okvir“

Recimo da želite da kreirate novu veb aplikaciju bez pogodnosti okvira. Odakle ćete da počnete njeno kreiranje? Verovatno će biti izvršeno rutiranje HTTP zahteva, pa morate da procenite sve dostupne HTTP zahteve i biblioteke odgovora i da izaberite jednu biblioteku. Zatim ćete morati da izaberete ruter. Verovatno ćete morati da podesite neki oblik konfiguracione datoteke za rute. Koju sintaksu treba koristiti? Gde treba smestiti tu sintaksu? Šta je sa kontrolerima? Gde „žive“ i kako se učitavaju? Verovatno će vam biti potreban kontejner za injektovanje zavisnosti da biste razrešili kontrolere i njihove zavisnosti. Koji kontejner?

Štaviše, ako odvojite vreme da odgovorite na sva ova pitanja i uspešno kreirate aplikaciju, kako će to uticati na sledećeg programera? Šta se dešava kada imate četiri ili nekoliko desetina takvih prilagođenih aplikacija zasnovanih na radnom okviru i morate da zapamtite gde kontroleri „žive“ u svakoj od njih ili koja je sintaksa rutiranja?

Konzistentnost i fleksibilnost

Radni okviri rešavaju problem konzistentnosti i fleksibilnosti, tako što obezbeđuju pažljivo razmotren odgovor na pitanje „Koju komponentu treba da koristimo ovde?“ i osiguravaju da posebno izabrane komponente dobro funkcionišu zajedno. Osim toga, okviri obezbeđuju konvencije koje smanjuju količinu koda koji novi programer treba da razume – ako razumete kako rutiranje funkcioniše u jednom Laravel projektu, razumećete kako funkcioniše u svim Laravel projektima.

Kada neko prepisuje vaš radni okvir za svaki novi projekat, ono za šta se taj neko zalaže je mogućnost da *kontroliše* šta se događa i da ne „ulazi“ u osnovu vaše aplikacije. To znači da će najbolji radni okviri ne samo obezbediti solidnu osnovu, već će vam dati i slobodu da ih prilagodite prema vašim potrebama. Konzistentnost i fleksibilnost, kao što ću vam prikazati u ostatku ove knjige, predstavljaju deo onoga što Laravel čini posebnim.

KRATKA ISTORIJA VEBA I PHP RADNIH OKVIRA

Da bismo mogli da odgovorimo na pitanje „Zašto Laravel?“, važno je da razumemo istoriju Laravela i ono što je bilo pre njega. Pre Laravelovog porasta popularnosti, postojali su različiti radni okviri i drugi pokreti u PHP-u i drugim prostorima za razvoj Veba.

Ruby on Rails

David Heinemeier Hansson je objavio prvu verziju radnog okvira Ruby on Rails 2004. godine. Od tada je bilo teško pronaći radni okvir za veb aplikacije koji na neki način nije bio pod uticajem Railsa.

Rails je popularizovao MVC, RESTful JSON API-e, konvenciju nad konfiguracijom, Active-Record i mnoge druge alatke i konvencije koje su imale značajan uticaj na način na koji su veb programeri pristupali svojim aplikacijama – posebno kada je reč o brzom razvoju aplikacija.

„Navala“ PHP radnih okvira

Većini programera je bilo jasno da su Rails i slični radni okviri veb aplikacija bili „talas“ budućnosti, a PHP okviri, uključujući i one koji imitiraju Rails radne okvire, počeli su da se pojavljuju brzo. CakePHP je bio prvi radni okvir 2005. godine, a nedugo iza njega su usledili Simfony, CodeIgniter, Zend Framework i Kohana (izdanak radnog okvira CodeIgniter). Radni okvir Yii je objavljen 2008. godine, a Aura i Slim dve godine kasnije, dok je 2011. godina „donela“ radne okvire FuelPHP i Laravel, koji nisu bili u potpunosti izdanci radnog okvira CodeIgniter, već su predloženi kao njegove alternative. Neki od ovih okvira su bili sličniji Railsu, fokusirajući se na objektno-relaciono mapiranje baza podataka (ORM), MVC strukture i druge alatke čiji je cilj brzi razvoj. Drugi radni okviri, kao što su Simfony i Zend, više su se fokusirali na projektne obrasce preduzeća i elektronsku trgovinu.

Dobre i loše strane radnog okvira CodeIgniter

CakePHP i CodeIgniter su prva dva PHP radna okvira koja su najviše bila inspirisana Railsom. CodeIgniter je brzo postao slavan, a do 2010. godine je nesumnjivo bio jedan od najpopularnijih nezavisnih PHP radnih okvira.

CodeIgniter je bio jednostavan, lak za upotrebu i hvaljen zbog odlične dokumentacije i snažne zajednice. Međutim, njegova upotreba moderne tehnologije i šablona napredovala je sporo. Pošto je „svet“ radnog okvira rastao i PHP-ove alatke napredovale, CodeIgniter je počeo da zaostaje u pogledu tehnološkog napretka i mogućnosti izvan njegovih funkcija. Za razliku od mnogih drugih radnih okvira, CodeIgniter je bio prespor u odnosu na novije funkcije radnog okvira PHP 5.3, kao što su imenski prostori i prelazak na Git-Hub, a kasnije i na Composer. Taylor Otwell, tvorac Laravela, bio je nezadovoljan radnim okvirom CodeIgniter, pa je 2010. godine počeo da piše svoj radni okvir.

Laravel 1, 2 i 3

Prvi beta Laravel 1 je realizovan u junu 2011. godine; napisan je potpuno „od nule“. Sadržao je prilagođeni ORM (Eloquent), rutiranje zasnovano na zatvaranju (inspirisano radnim okvirom Ruby Sinatra), modularnim sistemom za proširenje i „pomoćnicima“ za obrasce, validaciju, autentifikaciju i još mnogo štošta.

Rani razvoj Laravela je brzo napredovao, a Laravel 2 i 3 su objavljeni u novembru 2011, odnosno u februaru 2012. godine. Uvedeni su kontroleri, testiranje jedinica, alatka komandne linije, kontejner za inverziju kontrole (IoC), Eloquent veze i migracije.

Laravel 4

U Laravelu 4 Taylor je ponovo napisao čitav radni okvir „od nule“. Do tada je Composer, PHP-ov sada sveprisutniji menadžer paketa, pokazivao znakove da će postati industrijski standard, a Taylor je uvideo vrednost prepisivanja radnog okvira kao kolekcije komponenata koje je distribuirao i grupisao Composer.

Taylor je razvio skup komponenata pod kodnim imenom Illuminate i u maju 2013. godine objavio je Laravel 4, sa potpuno novom strukturom. Umesto da većinu svog koda stavi kao preuzimanje, Laravel je sada većinu svojih komponenata izdvojio iz Simfonyja (još jednog radnog okvira čije komponente mogu da koriste ostali radni okviri) i Illuminate komponenata pomoću Composera.

Laravel 4 je takođe uveo redove za čekanje, komponentu e-pošte, obrasce Facade i dodavanje baze podataka. A pošto se Laravel sada oslanja na Simfony komponente, najavljeno je da će Laravel biti preslikavanje (ne odmah, ali ubrzo) šestomesečnog rasporeda objavljivanja koji prati Simfony.

Laravel 5

Laravel 4.3 je trebalo da bude objavljen u novembru 2014. godine, ali je, pošto je razvoj napredovao, postalo jasno da značaj njegovih promena zaslužuje glavno izdanje, a Laravel 5 je objavljen u februaru 2015. godine.

Laravel 5 ima preuređenu strukturu direktorijuma, uklanjanje obrasca i HTML „pomoćnika“, uvođenje interfejsa ugovora, niz novih prikaza, Socialite za autentifikaciju društvenih medija, Elixir za kompilaciju resursa, Scheduler za pojednostavljenje planiranja poslova, dotenv za pojednostavljeno upravljanje okruženjem, zahteve za obrasce i potpuno novi REPL (read-evaluate-print loop). Laravel 5 je rastao (kad je reč o funkcijama) i „sazreo“, ali nije bilo velikih promena kao u prethodnim verzijama.

ŠTA JE POSEBNO U LARAVELU?

Šta razlikuje Laravel od drugih PHP radnih okvira? Zašto je vredno imati više od jednog PHP radnog okvira u bilo kom trenutku? Svi radni okviri koriste komponente kompanije Simfony, zar ne? Sada ćemo malo razgmotriti šta to čini Laravel posebnim.

Filozofija Laravela

Treba samo da pročitate Laravel marketinški materijal i README datoteke da biste videli vrednosti Laravela. Taylor koristi reči koji se odnose na svetlost, kao što su „Illuminate“ i „Spark“, ali i reči „Artisans“, „Elegant“, „Breath of fresh air“ i „Fresh start“ i na kraju „Rapid“ i „Warp speed“.

Dve vrednosti radnog okvira koje se najčešće pominju su povećanje brzine i sreća programera. Taylor je opisao „Artisan“ kao jezik koji se namerno suprotstavlja vrednostima koje su više utilitarne. Možete videti genezu ovakvog njegovog razmišljanja iz 2011. godine o StackExhangeu, kada je izjavio: „Ponekad provedem mnogo vremena (sate) da bih uložio napor da kod ‚izgleda lepo‘ - samo radi boljeg pregleda samog koda“. Često je govorio da programeri lakše i brže ostvare svoje ideje, oslobađajući se nepotrebnih prepreka prilikom kreiranja odličnih proizvoda.

Laravel je, u svojoj osnovi, opremanje i omogućavanje programera. Cilj je da se pomoću njega obezbedi jasan, jednostavan i lep kod, a i funkcije koje pomažu programerima da brzo nauče, pokrenu, razviju i napišu kod koji je jednostavan, jasan i trajan.

Koncept usmeravanja programera je jasan u dokumentima o Laravelu. U dokumentaciji je zapisano da „srećni programeri pišu najbolji kod“. „Sreća programera od preuzimanja do implementiranja“ je bio nezvanični slogan među tvorcima radnih okvira neko vreme. Naravno, cilj je da svaka alatka ili radni okvir učine programera srećnim. Međutim, kada je sreća programera *primarna* briga, a ne sekundarna, to ima ogroman uticaj na Laravelov stil i napredak u donošenju odluka. U slučajevima kada su drugi radni okviri možda usmereni na čistotu arhitekture kao na svoj primarni cilj ili na kompatibilnost sa ciljevima i vrednostima timova za razvoj preduzeća Laravel se prvenstveno fokusira da opsluži individualnog programera. To ne znači da u Laravelu ne možete da napišete aplikacije čiste arhitekture ili aplikacije koje su spremne za preduzeća, ali ne po cenu čitljivosti i razumljivosti vaše baze koda.

Kako Lavarel čini programera srećnim

Jedno je reći da želite da programeri budu srećni. Usrećiti programere je nešto sasvim drugo i od vas se traži da preispitate šta je to u radnom okviru što će najverovatnije učiniti programere nesrećnima i srećnima. Postoji nekoliko načina na koje Laravel pokušava da olakša život programerima.

Prvo, Laravel je radni okvir za brzi razvoj aplikacija. To znači da se fokusira na površnu (jednostavnu) krivu učenja i na minimiziranje koraka između pokretanja i objavljivanja nove aplikacije. Svi zadaci koji se najčešće koriste za izradu veb aplikacija, od interakcija baza podataka, do autentifikacije, pa sve od redova čekanja, do e-pošte i keširanja, pojednostavljuju komponente koje obezbeđuje Laravel.

Međutim, Laravel komponente nisu odlične same po sebi; one obezbeđuju konzistentni API i predvidljive strukture u čitavom radnom okviru. To znači da je, kada pokušate da uradite nešto novo u Laravelu, više nego verovatno da ćete na kraju reći: „…to jednostavno funkcioniše“.

Sve ovo se ne završava samim radnim okvirom. Laravel obezbeđuje čitav ekosistem alatki za izradu i pokretanje aplikacija. Postoje Homestead i Valet za lokalni razvoj, Forge za upravljanje serverom i Envoyer za naprednu implementaciju, a tu je i komplet dodatnih paketa: Cashier za plaćanja i pretplate, Echo za WebSockete, Scout za pretraživanje, Passport za API autentifikaciju, Dusk za frontend testiranje, Socialite za prijavu na društvenim mrežama, Horizon za praćenje redova čekanja, Nova za izradu administratorskih panela i Spark za pokretanje SaaS-a. Laravel pokušava da „izbaci“ poslove koji se ponavljaju da bi programeri mogli da urade nešto jedinstveno.

Drugo, Laravel se fokusira na „konvenciju nad konfiguracijom“ – to znači da morate, ako ste voljni da koristite Laravelove podrazumevane vrednosti, da obavite mnogo manje posla nego u drugim radnim okvirima koji zahtevaju da deklarišete sve svoje postavke, čak i ako koristite preporučenu konfiguraciju. Za projekte izgrađene u Laravelu potrebno je manje vremena nego za projekte koji su izgrađeni u većini drugih PHP radnih okvira.

Laravel se, takođe, detaljno fokusira na jednostavnost. U njemu možete koristiti injektovanje zavisnosti i simuliranje, šablon i spremišta, Command Query Responsibility Segregation i sve druge složenije arhitektonske šablone. Međutim, dok drugi radni okviri mogu da predlože korišćenje tih alatki i struktura u svakom projektu, Laravel i njegova dokumentacija i zajednica se oslanjaju na početak najjednostavnije moguće implementacije – globalne funkcije ovde, obrasce Facade tamo, ActiveRecord onamo. Ovo omogućava programerima da kreiraju najjednostavniju moguću aplikaciju kao rešenje za svoje potrebe, bez ograničavanja njene korisnosti u složenim okruženjima.

Zanimljivo je da se Laravel razlikuje od ostalih PHP radnih okvira po tome što su tvorac Laravela i Laravel zajednica više povezani i inspirisani radnim okvirom Ruby on Rails i funkcionalnim programskim jezicima, nego Javom.

U modernom PHP-u postoji jaka struja koja se oslanja na obimnost i složenost, prihvatajući aspekte PHP-a koji su sličniji Javi. Međutim, Laravel je obično na „drugoj strani“, prihvatajući izražajnu, dinamičnu i jednostavnu praksu kodiranja i jezičke funkcije.

Laravel zajednica

Ako ste prvi put „izloženi“ Laravel zajednici u ovoj knjizi, možete očekivati nešto posebno. Jedan od prepoznatljivih elemenata Laravela koji je doprineo njegovom rastu i uspehu je srdačna zajednica za podučavanje koja ga okružuje. Od Laracasts video uputstava Jeffreya Waya, Laravel vesti, Slacka i IRC-a i Discord kanala, od prijatelja sa Twittera do blogera, pa sve do podkasta Laracon konferencije, Laravel ima „bogatu“ i aktivnu zajednicu punu ljudi, od kojih su neki već odavno prisutni, a neki tu tek „prvi dan“. I to nije slučajno:

„Od samog početka Laravela hteo sam da se ljudi osećaju kao da su deo nečega. Prirodni instinkt ljudi je da žele pripadati i biti prihvaćeni u grupi drugih istomišljenika. Dakle, ‚ubrizgavanjem ličnosti' u veb radni okvir i aktivnošću u zajednici ta vrsta osećanja u njoj može rasti".

Taylor Otwell, *intervju podrške za proizvod*

Taylor je od ranih dana Laravela shvatio da su za uspešan projekat otvorenog koda potrebne dobra dokumentacija i srdačna zajednica. A one su sada obeležja Laravela.

KAKO LARAVEL FUNKCIONIŠE

Sve što sam podelio sa čitaocima u ovom poglavlju bilo je potpuno apstraktno. Pitate se šta je sa kodom. Sada ćemo „zaroniti" u jednostavnu aplikaciju (primer 1-1) da biste mogli da vidite kako se Laravel koristi iz dana u dan.

Primer 1-1 „Hello World" u datoteci routes/web.php

```
<?php
Route::get(,/', function () {
    return ,Hello, World!';
});
```

Najjednostavnija moguća radnja koju možete da preduzmete u aplikaciji Laravel je da definišete rutu i vratite rezultat u bilo kom trenutku kada neko poseti tu rutu. Ako inicijalizujete potpuno novu Laravel aplikaciju na vašoj mašini, definišite rutu u primeru 1-1, a zatim opslužite sajt iz *javnog* direktorijuma i dobićete potpuno funkcionalan „Hello, World" primer (pogledajte sliku 1-1) .

Slika 1-1 Vraćanje rezultata „Hello, Worl!" pomoću Laravela

Ovo izgleda slično kao u kontroleru, kao što možete videti u primeru 1-2.

Primer 1-2 „Hello, World“ pomoću kontrolera

```
// File: routes/web.php
<?php

Route::get(‘/’, ‘WelcomeController@index’);


// File: app/Http/Controllers/WelcomeController.php
<?php

namespace App\Http\Controllers;

class WelcomeController extends Controller
{
    public function index()
    {
        return ‘Hello, World!’;
    }
}
```

Ako skladištite pozdrave u bazi podataka, ovo će izgledati prilično slično (pogledajte primer 1-3).

Primer 1-3 Više „Hello, World“ pozdrava pomoću pristupa bazi podataka

```
// File: routes/web.php
<?php

use App\Greeting;

Route::get(‚create-greeting‘, function () {
    $greeting = new Greeting;
    $greeting->body = ‚Hello, World!‘;
    $greeting->save();
});

Route::get(‚first-greeting‘, function () {
    return Greeting::first()->body;
});
// File: app/Greeting.php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Greeting extends Model
{
    //
}
```

```
// File: database/migrations/2015_07_19_010000_create_greetings_table.php
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateGreetingsTable extends Migration
{

    public function up()
    {

        Schema::create('greetings', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('body');
            $table->timestamps();

        });
    }
public function down()
{

    Schema::dropIfExists('greetings');
}
}
```

Primer 1-3 može biti malo komplikovaniji, pa, ako jeste, samo ga preskočite. Naučićete o svemu što se ovde događa u kasnijim poglavljima, ali već vidite da pomoću samo nekoliko redova koda možete podesiti migracije baze podataka i modela i izdvojiti zapise. To je veoma jednostavno.

Zašto Laravel?

DAKLE, ZAŠTO LARAVEL?

Zato što vam Laravel pomaže da ostvarite svoje ideje bez gubitka koda, koristeći moderne standarde kodiranja, okruženi aktivnom zajednicom, sa osnaženim ekosistemom alatki.

I zato što vi, dragi programeri, zaslužujete da budete srećni.

POGLAVLJE 2

Podešavanje Laravel razvojnog okruženja

Deo uspeha PHP-a je posledica činjenice da je teško pronaći veb server koji *ne može* da opsluži PHP. Međutim, moderne PHP alatke imaju striktnije zahteve od onih iz prošlosti. Najbolji način za razvoj Laravela je da se osigura konzistentno lokalno i udaljeno serversko okruženje za kod; srećom, ekosistem Laravel ima nekoliko alatki za to.

ZAHTEVI SISTEMA

Sve ono što ćemo „pokriti“ u ovom poglavlju možemo prikazati pomoću Windows računara, ali biće vam potrebne desetine stranica sa prilagođenim instrukcijama i upozorenjima. Prepustiću te instrukcije i upozorenja korisnicima Windowsa, pa će primeri u ovom poglavlju i ostatku knjige biti fokusirani na Unix/Linux/macOD programere.

Bez obzira da li ćete odlučiti da opslužujete vaš veb sajt instaliranjem PHP-a i drugih alatki na lokalnom računaru, da opslužujete razvojno okruženje iz virtuelne mašine pomoću Vagranta ili Dockera ili da se oslonite na alatku kao što je MAMP/WAMP/XAMPP, u vaše razvojno okruženje morate da instalirate sve što je navedeno u nastavku da bi Laravel sajtovi bili opsluženi:

- PHP >= 7.1.3 za Laravel verzije od 5.6 do 5.8, PHP >= 7.0.0 za verziju 5.5, PHP >= 5.6.4 za verziju 5.4, PHP između verzija 5.6.4 i 7.1.* za verzije 5.3 ili PHP >= 5.5.9 za verzije 5.2 i 5.1
- OpenSSL PHP ekstenzija
- PDO PHP ekstenzija
- Mbstring PHP ekstenzija
- Tokenizer PHP ekstenzija

- XML PHP ekstenzija (Laravel 5.3 i noviji)
- Ctype PHP ekstenzija (Laravel 5.6 i noviji)
- JSON PHP ekstenzija (Laravel 5.6 i noviji)
- BCMath PHP ekstenzija (Laravel 5.7 i noviji)

Composer

Bez obzira na računar na kome ćete programirati, potrebno je da Composer bude instaliran globalno. Composer je alatka koja je u osnovi najmodernijeg PHP razvoja. On je menadžer zavisnosti za PHP, slično kao NPM za Node ili RubyGems za Ruby. Međutim, baš kao i NPM, Composer je takođe osnova mnogih naših testiranja, učitavanja lokalnih skriptova, instalacionih skriptova i još mnogo čega. On je potreban da biste instalirali i ažurirali Laravel i uneli neke spoljne zavisnosti.

LOKALNA RAZVOJNA OKRUŽENJA

Za mnoge projekte dovoljan je hosting razvojnog okruženja u kome se koristi jednostavniji skup alatki. Ako ste na vašem sistemu već instalirali MAMP, WAMP ili XAMPP, to će verovatno biti dovoljno da pokrenete Laravel. Takođe možete pokrenuti Laravel pomoću PHP-ovog ugrađeng veb servera, pod pretpostavkom da je vaš sistemski PHP odgovarajuće verzije.

Da biste započeli rad u Laravelu, treba samo da pokrenete PHP. Sve ostalo zavisi od vas.

Laravel sadrži Valet i Homestead, dve dragocene alatke za lokalni razvoj. Ako niste odlučni koju da upotrebite, preporučujem vam da koristite Valet i da samo ukratko upoznate Homestead.

Laravel Valet

Ako želite da koristite PHP-ov ugrađeni veb server, najjednostavnija opcija je da opslužite svaki sajt pomoću URL-a *lokalnog veb servera*. Ako pokrenete `php -S localhost:8000 -t public` iz osnovne fascikle vašeg Laravel sajta, PHP-ov ugrađeni veb server opslužiće vaš veb sajt na adresi *http://localhost:8000/*. Takođe možete pokrenuti `php artisan serve` nakon što podesite vašu aplikaciju za jednostavno pokretanje ekvivalentnog servera.

Ako ste zainteresovani da povežete svaki vaš sajt sa određenim razvojnim domenom, moraćete da upoznate datoteke hosta operativnog sistema i da koristite alatku kao što je dnsmask. Međutim, pokušaćemo da uradimo nešto jednostavnije.

Ako ste korisnik Maca (postoje i nezvanična izdanja Laravel Valeta za Windows i Linux), u Laravel Valetu nećete morati da povežete vaše domene sa fasciklama aplikacija. Valet instalira dnsmask i niz PHP skriptova koji omogućavaju da ukucate `laravel new miapp && open maypp.test` i da sve *jednostavno funkcioniše*. Moraćete da instalirate nekoliko alatki koje koriste Homebrew kroz koji će vas „voditi“ dokumentacija, ali koraci od početne instalacije do opsluživanja aplikacija su mali i jednostavni.

Instalirajte Valet (pogledajte dokumente za najnovija uputstva za instalaciju) i preusmerite ga na jedan ili više direktorijuma u kojima će biti smešteni vaši sajtovi. Pokrenuo sam `valet park` iz direktorijuma *~/Sites*, u koji sam stavio sve svoje aplikacije za razvoj. Sada možete samo da dodate *.test* na kraju naziva direktorijuma i da ga posetite u vašem pregledaču.

Valet olakšava opsluživanje svih fascikli u konkretnoj fascikli kao *{foldername}.test* (koristeći `valet park`), opsluživanje samo jedne fascikle (koristeći `valet link`), otvaranje domena koji opslužuje Valet (koristeći `valet open`), opsluživanje Valet sajta pomoću HTTPS-a (koristeći `valet secure`) i otvaranje tunela ngrok, tako da biste mogli da delite vaš sajt, koristeći `valet share`.

Laravel Homestead

Homestead je još jedna alatka koju možete koristiti za postavljanje lokalnog razvojnog okruženja. To je alatka za konfiguraciju, koja se nalazi na vrhu Vagranta (alatke za upravljanje virtuelnim mašinama) i obezbeđuje unapred konfigurisanu sliku virtuelne mašine koja je savršeno podešena za razvoj Laravela i predstavlja najčešće proizvodno okruženje u kojem su pokrenuti mnogi Laravel sajtovi. Homestead je, verovatno, i najbolje lokalno razvojno okruženje za programere koji koriste Windows računare.

Dokumenti Homesteada su robusni i stalno se ažuriraju, tako da ću vas samo uputiti na njih da biste naučili kako Homestead funkcioniše i kako se podešava.

Vessel

Ovo nije zvanični Laravel projekat, već je Chris Fidao sa sajtova „Servere for Hackers“ i „Shipping Docker“ napravio jednostavnu alatku za kreiranje Docker okruženja za Laravel razvoj, pod nazivom Vessel. Pogledajte dokumentaciju za Vessel da biste saznali više informacija.

KREIRANJE NOVOG LARAVEL PROJEKTA

Postoje dve opcije za kreiranje novog projekta Laravel - obe se pokreću iz komandne linije. Prva je globalno instaliranje Laravel instalacione alatke (korišćenjem Composera), a druga je upotreba Composerove funkcije za kreiranje projekta. Detaljnije informacije o ove dve opcije možete saznati na stranici sa dokumentacijom za instalaciju, ali preporučujem Laravel instalacionu alatku.

Instaliranje Laravela pomoću Laravel instalacionih alatki

Ako ste instalirali Composer globalno, instaliranje Laravel instalacione alatke je jednostavno, baš kao i pokretanje sledeće komande:

```
composer global require „laravel/installer“
```

Nakon što instalirate Lavarel instalacionu alatku, pokretanje novog Laravel projekta je jednostavno. Samo pokrenite ovu komandu iz komandne linije:

```
laravel new projectName
```

Biće kreiran novi poddirektorijum trenutnog direktorijuma pod nazivom *{projectName}* i instaliran osnovni Lavarel projekat u tom poddirektorujumu.

Instaliranje Laravela pomoću Composerove funkcije create-project

Composer takođe obezbeđuje funkciju pod nazivom create-project za kreiranje novih projekata sa određenom osnovnom strukturom. Da biste koristili ovu alatku za kreiranje novih Laravel projekata, izvršite sledeću komandu:

composer create-project laravel/laravel projectName

Baš kao i instalaciona alatka, ova funkcija će kreirati poddirektorijum vašeg trenutnog direktorijuma pod nazivom {projectName} sa osnovnom strukturom za Laravel instalaciju, koja je spremna za programiranje.

Lambo: super moćni laravel new

Pošto često preduzimam isti niz koraka nakon kreiranja novog projekta Laravel, kreirao sam jednostavni skript pod nazivom Lambo, koji automatizuje te korake uvek kada kreiram novi projekat.

Lambo pokreće `laravel new` i unosi kod u Gitu, postavlja *.env* akreditive sa razumnim podrazumevanim vrednostima i otvara projekat u pregledaču; opciono, otvara projekat u uređivaču i preduzima nekoliko drugih korisnih koraka za izradu.

Možete instalirati Lambo pomoću Composerove komande `global require`:

```
composer global require tightenco/lambo
```

Možete da koristite Lambo kao `laravel new` :

```
cd Sites
lambo my-new-project
```

Laravelova struktura direktorijuma

Kada otvorite direktorijum koji sadrži osnovnu Laravel aplikaciju, videćete sledeće datoteke i direktorijume:

```
app/
bootstrap/
config/
public/
resources/
routes/
storage/
tests/
vendor/
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package.json
phpunit.xml
readme.md
server.php
webpack.mix.js
```

Različite alatke za izradu u Laravelu pre verzije 5.4

U projektima koji su kreirani pre Laravela 5.4 verovatno ćete videti gulpfile.js, umesto webpack.mix.js; to znači da je u projektu pokrenut Laravel Elixir, umesto Laravel Mix.

Sada ćemo pregledati jednu po jednu fasciklu da biste ih upoznali.

Fascikle

Osnovni direktorijum podrazumevano sadrži sledeće fascikle:

app

U ovoj fascikli će biti smešten veći deo stvarnih aplikacija - modeli, kontroleri, komande i PHP kod domena.

bootstrap

Ova fascikla sadrži datoteke koje Laravel radni okvir koristi za pokretanje uvek kada se izvršava.

config

U ovoj fascikli „žive“ sve konfiguracione datoteke.

database

U ovoj fascikli „žive“ migracije baze podataka, početne vrednosti i „fabrike“.

public

Ovo je direktorijum na koji server ukazuje kada opslužuje veb sajt. U njemu se nalazi *index.php*, tj. prednji kontroler koji aktivira proces pokretanja i usmerava sve zahteve na odgovarajući način. Ovde se nalaze i datoteke sa javnim postavkama, kao što su slike, opisi, stilovi, skriptovi ili sadržaj za preuzimanje.

resources

U ovoj fascikli „žive“ datoteke koje su potrebne za druge skriptove - prikazi, jezičke datoteke i (opcione) Sass/Less/source CSS i izvorne JavaScript datoteke.

routes

U ovoj fascikli „žive“ sve definicije za HTTP rute i „rute konzole“, odnosno Artisan komande.

storage

U ovoj fascikli „žive“ keš memorije, evidencije i kompajlirane sistemske datoteke.

tests

U ovoj fascikli „žive“ testovi jedinice i integracije.

vendor

U ovoj fascikli Composer instalira svoje zavisnosti. Git je ignoriše, jer se očekuje da se Composer izvršava kao deo procesa implementacije na udaljenim serverima.

Slobodne datoteke

Osnovni direktorijum sadrži sledeće datoteke:

.editorconfig

Daje uputstva IDE-u/uređivaču teksta o Laravelovim standardima za kodiranje (na primer, veličina uvlačenja, skup znakova i da li treba skratiti prateći razmak). Videćete ovu fasciklu u svakoj Laravel aplikaciji u kojoj je pokrenuta verzija 5.5 i novija.

.env i .env.example

„Diktira" promenljive okruženja (za koje se očekuje da će biti različite u svakom okruženju, pa zato nisu namenjene upravljanju verzijama). *.env.example* je šablon koji svako okruženje treba da duplira da bi kreiralo sopstvenu *.env* datoteku koju Git ignoriše.

.gitignore i .gitattributes

Ovo su Git konfiguracione datoteke.

artisan

Omogućava da pokrenete Artisan komande (pogledajte Poglavlje 8) iz komandne linije.

composer.json i composer.lock

Ovo su konfiguracione datoteke za Composer; *composer.json* može uređivati korisnik, a *composer.lock* ne može. U ovoj fascikli datoteke dele neke osnovne informacije o projektu i takođe definišu njegove PHP zavisnosti.

package.json

Ovo je ista datoteka kao *composer.json*, ali za front-end elemente i zavisnosti sistema za izradu; ukazuje NPM-u na kojim JavaScript zavisnostima treba da se zaustavi.

phpunit.xml

Ovo je konfiguraciona datoteka za PHPUnit, koju Laravel alatka koristi za uobičajeno testiranje.

readme.md

Ovo je Markdown datoteka koja obezbeđuje osnovne instrukcije za Laravel. Nećete videti ovu datoteku ako koristite Laravel instalacionu alatku.

server.php

Ovo je server za kreiranje rezervnih kopija koji omogućava slabijim serverima da i dalje pregledaju Laravel aplikaciju.

webpack.mix.js

Ovo je (opciona) konfiguraciona datoteka za Mix. Ako koristite Elixir, videćete *gulpfile.js*. Ove datoteke služe da bi sistemu za izradu davale uputstva kako da kompajliraju i obrade front-end elemente.

KONFIGURACIJA

Osnovne postavke aplikacije Laravel (postavke veze baze podataka, postavke reda za čekanje i e-pošte itd.) „žive“ u datotekama u fascikli *config*. Svaka od ovih datoteka vraća kao rezultat PHP niz, a svaka vrednost u nizu je dostupna pomoću konfiguracionog ključa, koji se sastoji od naziva datoteke i svih ključeva „potomaka“, razdvojenih tačkama (.). Dakle, ako kreirate datoteku u konfiguracionoj datoteci *config/services.php* koja izgleda ovako:

```
// config/services.php
<?php
return [
    ‚sparkpost‘ => [
        ‚secret‘ => ‚abcdefg‘,
    ],
];
```

možete pristupiti promenljivoj konfiguracije, koristeći `g(‚services.sparkpost.secret‘).`

Svaka promenljiva konfiguracije koja treba bude različita za svako okruženje (i zbog toga nije namenjena kontroli izvora) „živeće“ u vašim *.env* datotekama. Ako želite da koristite različit Bugsnag API ključ za svako okruženje, podesićete konfiguracionu datoteku tako da biste mogli da povučete vrednost iz *.env* datoteke:

```
// config/services.php
<?php
return [
    ‚bugsnag‘ => [
        ‚api_key‘ => env(‚BUGSNAG_API_KEY‘),
    ],
];
```

Ova `env()`pomoćna funkcija povlači vrednost iz vaše *.env* datoteke sa istim ključem. Dakle, sada dodajte taj ključ datotekama *.env* (postavkama za ovo okruženje) i *.env.example* (šablonu za sva okruženja):

```
# In .env
BUGSNAG_API_KEY=oinfp9813410942

# In .env.example
BUGSNAG_API_KEY=
```

Vaša *.env* datoteka već sadrži veliki broj promenljivih za okruženje koje su potrebne za radni okvir, kao što je upravljački program e-pošte koji ćete koristiti, i postavke osnovne baze podataka.

Korišćenje poziva env() izvan konfiguracionih datoteka

Određene funkcije u Laravelu, uključujući neke funkcije keširanja i optimizacije, nisu dostupne ako koristite pozive env() bilo gde izvan konfiguracionih datoteka.

Najbolji način za povlačenje promenljivih okruženja je podešavanje onoga što želite da bude specifično za okruženje. Omogućite da te stavke konfiguracije pročitaju promenljive okruženja, a zatim referencirajte promenljive konfiguracije bilo gde u okviru vaše aplikacije:

```
// config/services.php
return [
    ‚bugsnag‘ => [
        ‚key‘ => env(‚BUGSNAG_API_KEY‘),
    ],
];

// In controller, or whatever
$bugsnag = new Bugsnag(config(‚services.bugsnag.key‘));
```

Datoteka .env

Sada ćemo da pogledamo podrazumevani sadržaj datoteke *.env*. Tačni ključevi će se razlikovati, u zavisnosti od verzije Laravela koju koristite, ali u primeru 2-1 ćete videti kako ti ključevi izgledaju u verziji 5.8.

Primer 2-1 Podrazumevane promenljive okruženja u Laravelu 5.8

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
```

```
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Nećemo razmatrati sve ključeve, zato što je većina njih samo grupa informacija o autentifikaciji za različite servise (Pusher, Redis, DB, Mail). Međutim, ove dve važne promenljive okruženja treba da znate:

APP_KEY

nasumično generisani znakovni niz koji se koristi za šifrovanje podataka. Ako je ovo prazno, možete naići na grešku „No application encryption key has been specified". U tom slučaju samo pokrenite komandu `php artisan key:generate` i Laravel će generisati jedan ključ.

APP_DEBUG

Bulova vrednost koja utvrđuje da li korisnici ove instance vaše aplikacije treba da vide greške debagovanja – odlično je za lokalna i scenska okruženja, a loše za proizvodnju.

Ostalim postavkama koje se odnose na autentifikaciju (`BROADCAST_DRIVER, QUEUE_CONNECTION` itd.) date su podrazumevane vrednosti koje funkcionišu uz minimum oslanjanja na spoljne servise, što je savršeno za početak rada. Kada pokrenete prvu aplikaciju Laravel, verovatno ćete želeti da za većinu projekata izvršite promenu samo u postavkama konfiguracije baze podataka. Koristim Laravel Valet, pa sam promenio `DB_DATABASE` u naziv mog projekta, `DB_USERNAME` u `root` i `DB_PASSWORD` u prazan znakovni niz:

```
DB_DATABASE=myProject
DB_USERNAME=root
DB_PASSWORD=
```

Zatim sam kreirao bazu podataka koja ima isti naziv kao projekat u mom omiljenom MySQL klijentu i spreman sam da nastavim dalje.

POČETAK RADA

Sada ste spremni da počnete rad u praznoj Laravel instalaciji. Pokrenite `git init`, izvršite prazne datoteke pomoću komandi `git add.` i `git commit` i spremni ste da započnete pisanje koda. To je sve!

Ako koristite Valet, možete da pokrenete sledeće komande i da momentalno vidite vaš sajt „uživo“ u pregledaču:

```
laravel new myProject && cd myProject && valet open
```

Uvek kada pokrenem novi projekat, preduzimam ove korake:

```
laravel new myProject
cd myProject
git init
git add .
git commit -m „Initial commit“
```

Sve svoje veb sajtove čuvam u fascikli *~/Sites*, koju sam postavio kao primarni Valet direktorijum da bih u ovom slučaju odmah mogao pristupiti testu *myProject.test* u pregledaču, bez dodatnog posla. Mogu da uređujem datoteku *.env* i da je usmerim na određenu bazu podataka, da dodam tu bazu podataka u MySQL aplikaciju i spreman sam za početak pisanja koda. Ako koristite Lambo, svi ovi koraci su već automatski izvršeni.

TESTIRANJE

Na kraju svakog poglavlja u odeljku „Testiranje“ prikazano je kako se pišu testovi za funkcije koje su razmatrane. Pošto ovo poglavlje ne „pokriva“ funkciju koja se može testirati, ukratko ćemo razmotriti testove (više informacija o pisanju i pokretanju testova u Laravelu naći ćete u Poglavlju 12).

Laravel uvodi PHPUnit kao zavisnost i konfigurisan je da pokrene testove u svakoj datoteci u direktorijumu *testova* čiji se naziv završava sa *Test.php* (na primer, *tests/UserTest.php*).

Dakle, najjednostavniji način za pisanje testova je kreiranje datoteke u direktorijumu testova sa nazivom koji se završava sa *Test.php*. Testove ćete najlakše pokrenuti ako pokrenete komandu `./vendor/bin/phpunit` iz komandne linije (u korenu projekta). Ako testovi zahtevaju pristup bazi podataka, obavezno ih pokrenite sa računara na kome se nalazi baza podataka, pa ako hostujete bazu podataka u Vagrantu, proverite da li je *ssh* u Vagrant radnom okviru da biste odatle pokrenuli testove. Više informacija o testovima i još mnogo čemu možete saznati u Poglavlju 12.

U nekim odeljcima za testiranje biće korišćene sintaksa testiranja i funkcije koje još uvek niste upoznali (ukoliko prvi put čitate ovu knjigu). Ako vam je kod u nekim odeljcima za testiranje nejasan, samo ga preskočite i vratite se na njega nakon što pročitate poglavlje o testiranju.

TL;DR

Pošto je Laravel PHP radni okvir, veoma je jednostavno opslužiti ga lokalno. Laravel obezbeđuje i dve alatke za upravljanje lokalnim razvojem: jednostavniju alatku pod nazivom Valet, koja koristi lokalni računar za obezbeđivanje zavisnosti, i unapred konfigurisanu Vagrant postavku, pod nazivom Homestead. Laravel koristi Composer i može se instalirati pomoću njega sa nizom fascikli i datoteka koje odražavaju konvencije i vezu sa drugim alatkama otvorenog koda.