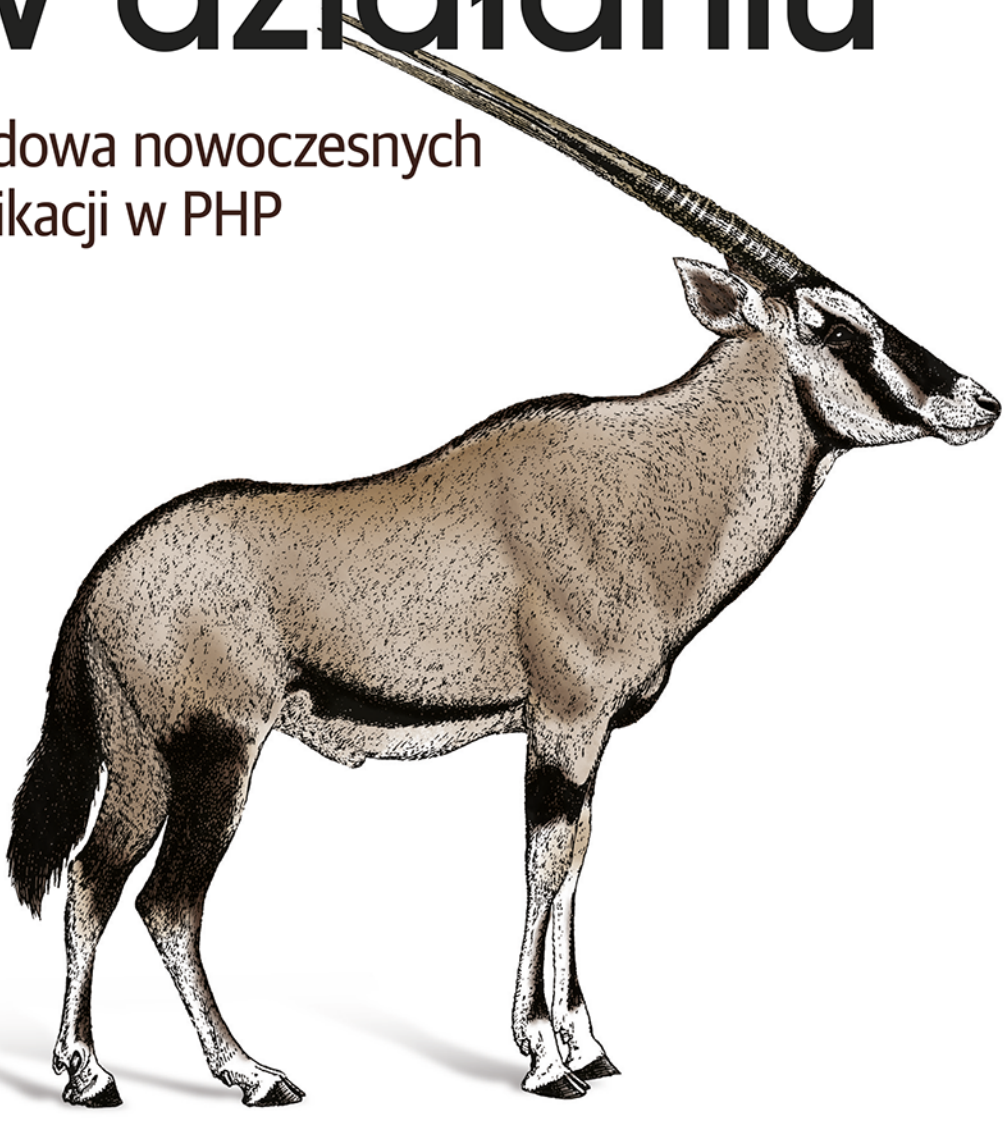
O'REILLY®

Wydanie II

Laravel w działaniu

Budowa nowoczesnych aplikacji w PHP



Helion

Matt Stauffer

Tytuł oryginału: Laravel Up & Running A Framework for Building Modern PHP Apps, 2nd Edition

Tłumaczenie: Piotr Rajca

ISBN: 978-83-283-7633-5

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/larav2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 1.

Dlaczego Laravel?

W początkowych latach dynamicznie rozwijającego się internetu pisanie aplikacji internetowych wyglądało całkowicie inaczej niż obecnie. W tamtych czasach programiści byli odpowiedzialni nie tylko za pisanie kodu stanowiącego unikalną logikę biznesową aplikacji, lecz także za implementację komponentów powszechnie występujących na wszystkich witrynach, takich jak: uwierzytelnianie, weryfikacja danych wprowadzanych przez użytkowników, dostęp do baz danych, obsługa szablonów itd.

Obecnie programiści mają do dyspozycji dziesiątki frameworków oraz tysiące komponentów i bibliotek. Bardzo często można usłyszeć, że w czasie potrzebnym do nauczenia się korzystania z jednego frameworku pojawiają się trzy nowe (i rzekomo lepsze), które mają go zastąpić.

Stwierdzenie: „Dlatego, że istnieje" może być wystarczającym wyjaśnieniem dla wspinania się na jakiś górski szczyt, jeśli jednak chodzi o wybór konkretnego frameworku do zastosowania albo o podjęcie decyzji, czy w ogóle jakiegoś używać, istnieją lepsze uzasadnienia. Czy warto nawet zadawać pytanie, dlaczego używać frameworków? Albo bardziej konkretnie, dlaczego używać Laravela?

Po co używać frameworku?

Bardzo łatwo można odpowiedzieć na pytanie, jakie korzyści wynikają ze stosowania poszczególnych komponentów lub pakietów dostępnych dla programistów używających języka PHP. W przypadku pakietów za stworzenie i utrzymanie tych izolowanych fragmentów kodu o ściśle zdefiniowanym przeznaczeniu odpowiada ktoś inny i teoretycznie rzecz biorąc, ta osoba znacznie lepiej rozumie działanie tego komponentu niż my, wziąwszy pod uwagę czas, jaki możemy na to poświęcić.

Frameworki podobne do Laravela, takie jak Symfony, Lumen czy Slim, stanowią kolekcje komponentów przygotowanych przez niezależnych programistów, powiązanych ze sobą charakterystycznym dla danego frameworku „spoiwem" — plikami konfiguracyjnymi, dostawcami usług, narzuconą strukturą katalogów czy skryptami uruchamiającymi aplikacje. A zatem, ogólnie rzecz ujmując, zaleta korzystania z frameworków polega na tym, że ktoś za nas podejmuje decyzje nie tylko o tym, jakich komponentów użyć, lecz także *jak te komponenty powinny ze sobą współdziałać*.

„Zrobię to po swojemu"

Załóżmy, że podjęliśmy decyzję o rozpoczęciu prac nad nową aplikacją internetową, którą zamierzamy tworzyć bez korzystania z jakiegokolwiek frameworku. Gdzie musimy zacząć? No cóż… taka aplikacja pewnie będzie musiała obsługiwać żądania HTTP, więc będziemy musieli przejrzeć wszystkie dostępne biblioteki obsługujące żądania i odpowiedzi HTTP i wybrać jedną z nich. Następnie będziemy musieli wybrać mechanizm odpowiedzialny za trasowanie żądań HTTP. A przy okazji będziemy musieli wybrać jakiś sposób zapisu plików z informacjami o konfiguracji tras. A jakiej składni w nich używać? Gdzie je umieszczać? A co z kontrolerami? Gdzie mają być zapisywane i jak je wczytywać? No cóż, zapewne będziemy potrzebowali jakiegoś kontenera wstrzykiwania zależności, który pozwoliłby nam wybierać kontenery oraz obsługiwać ich zależności. Ale jaki to ma być kontener?

Co więcej, jeśli poświęcimy czas, by odpowiedzieć na te wszystkie pytania, i pomyślnie stworzymy aplikację, to jaki będzie wpływ podjętych przez nas decyzji na kolejnych programistów, którzy będą się tą aplikacją zajmować? Co się stanie, jeśli będziemy mieli na głowie cztery albo kilkanaście takich aplikacji opartych na samodzielnie skonstruowanych frameworkach i będziemy musieli pamiętać, gdzie w każdej z nich umieszczać kontrolery i jaka jest składnia plików z informacjami o trasach?

Konsekwencja i elastyczność

Frameworki rozwiązują ten problem, udostępniając dogłębnie przemyślaną odpowiedź na pytanie: „Którego komponentu należy użyć w tym miejscu?" oraz zapewniając, że wszystkie wybrane komponenty będą ze sobą dobrze współdziałać. Co więcej, frameworki dostarczają konwencji pozwalających zmniejszyć ilość kodu, który musi zrozumieć programista dopiero zaczynający pracę nad danym projektem — jeśli będziemy rozumieć, jak działa określanie tras w jednym projekcie napisanym w Laravelu, będziemy rozumieli, jak ono działa także w innych projektach używających tego frameworku.

Kiedy ktoś nakazuje przygotowanie nowego frameworku dla każdego nowego projektu, tak naprawdę postuluje zapewnienie możliwości *kontrolowania* tego, co wejdzie w skład fundamentu, na którym będziemy budowali te aplikacje. To oznacza, że najlepsze frameworki będą nam zapewniać nie tylko solidną podstawę, lecz także swobodę w dostosowywaniu ich do własnych potrzeb. I jak pokażę w dalszej części tej książki, właśnie te cechy sprawiają, że Laravel jest tak unikalny.

Krótka historia WWW i frameworków PHP

Ważnym elementem udzielenia odpowiedzi na pytanie: „Dlaczego Laravel" jest zrozumienie historii tego frameworka oraz rozumienie historii jego poprzedników. Nim Laravel zyskał popularność, istniało wiele innych frameworków, i to nie tylko pisanych w języku PHP, lecz także w środowiskach internetowych.

Ruby on Rails

David Heinemeier Hansson udostępnił pierwszą wersję frameworku Ruby on Rail w roku 2004 i od tamtego czasu trudno jest znaleźć jakikolwiek framework do tworzenia aplikacji internetowych, który nie byłby w jakimś stopniu wzorowany na Ruby on Rails.

Framework ten spopularyzował zastosowanie wzorca MVC, API typu RESTful korzystających z formatu JSON, konwencji konfiguracyjnych, wzorca ActiveRecord, jak również wielu innych narzędzi i konwencji, które wywarły ogromny wpływ na sposób, w jaki programiści obecnie tworzą aplikacje internetowe — zwłaszcza pod względem szybkości ich tworzenia.

Zalew frameworków PHP

Dla większości programistów było oczywistym, że Rails oraz inne podobne frameworki do tworzenia aplikacji były powiewem przyszłości, a frameworki PHP, w tym takie, które jawnie naśladowały Rails, faktycznie zaczęły powstawać bardzo szybko.

Pierwszym z nich był CakePHP udostępniony w 2005 roku, szybko po nim pojawiły się framework Symfony, CodeIgniter, Zend Framework oraz Kohana (stanowiąca modyfikację CodeIgnitera). W roku 2008 pojawił się framework Yii, a w 2010 Aura i Slim. W roku 2011 pojawiły się FuelPHP i Laravel, które nie tyle były modyfikacjami CodeIgnitera, ile stanowiły dla niego alternatywy.

Niektóre z tych frameworków były bardziej podobne do Ruby on Rails, koncentrowały się na mechanizmach odwzorowań obiektowo-relacyjnych (ORM), strukturach MVC oraz innych rozwiązaniach mających zapewnić szybkość tworzenia aplikacji. Inne, takie jak Symfony i Zend, koncentrowały się bardziej na korporacyjnych wzorcach projektowych i zastosowaniach związanych z handlem elektronicznym.

Dobre i złe strony CodeIgnitera

CakePHP oraz CodeIgniter były dwoma wczesnymi frameworkami PHP, które najbardziej otwarcie podchodziły do tematu stopnia, w jakim były wzorowane na Ruby on Rails. CodeIgniter szybko zdobył popularność i w 2010 roku był prawdopodobnie najpopularniejszym z niezależnych frameworków PHP.

Był prosty, łatwy w użyciu, dysponował niesamowitą dokumentacją i prężną społecznością. Jednak dość wolno adaptował nowoczesne technologie i wzorce projektowe, a wraz z rozwojem frameworków do tworzenia aplikacji internetowych oraz wszelkiego rodzaju narzędzi wspomagających pisanie w języku PHP CodeIgniter powoli zaczął odstawać od konkurentów pod względem zaawansowania technologicznego i wbudowanych możliwości. W odróżnieniu od wielu innych frameworków rozwojem CodeIgnitera zajmowała się firma, której stosunkowo dużo czasu zajęło podjęcie decyzji o wykorzystaniu nowych możliwości języka PHP 5.3, takich jak przestrzenie nazw, przeniesienie bazy kodu do serwisu GitHub, a następnie zapewnienie możliwości korzystania z Composera. W 2010 roku Taylor Otwell, twórca Laravela, był już na tyle niezadowolony z CodeIgnitera, że zdecydował się na stworzenie własnego frameworku.

Laravel 1, 2 i 3

Pierwsza wersja beta Laravela 1 została udostępniona w czerwcu 2011 roku i w całości napisano ją od podstaw. Była wyposażona we własny mechanizm ORM (Eloquent), mechanizm trasowania oparty na domknięciach (wzorowany na Ruby Sinatra), nowoczesny system rozszerzeń i funkcje pomocnicze wspomagające tworzenie formularzy, weryfikację poprawności danych, uwierzytelnianie itd.

Prace nad rozwojem frameworka postępowały szybko i jego kolejne wersje, Laravel 2 i Laravel 3, pojawiły się odpowiednio w listopadzie 2011 roku i lutym 2012 roku. Wprowadzono w nich kontrolery, obsługę testów jednostkowych, narzędzia wiersza poleceń, kontroler odwrócenia sterowania (Ioc), obsługę związków oraz migracji w mechanizmie Eloquent.

Laravel 4

Tworząc wersję Laravel 4, Taylor od podstaw przepisał cały framework. W tamtym czasie Composer, obecnie wszechobecny menedżer pakietów PHP, był na najlepszej drodze, by stać się standardem, a Taylor zauważył zalety, jakie da przepisanie frameworka w postaci kolekcji komponentów dostarczanych i scalanych przy użyciu Composera.

Taylor stworzył więc kolekcję komponentów o nazwie kodowej *Illuminate* i w maju 2013 roku udostępnił framework Laravel 4 o całkowicie nowej strukturze. Kod tej nowej wersji Laravela nie był udostępniany w formie pakietu, który programista pobierał w całości, lecz raczej sam framework pobierał większość swych komponentów z Symfony (kolejnego frameworka udostępniającego swoje pakiety tak, że mogły być pobierane przez inne rozwiązania) i Illuminate, używając do tego celu Composera.

W Laravelu 4 zostały także wprowadzone kolejki, komponent poczty elektronicznej, fasady (*facades*) i wstępna inicjalizacja baz danych. A ponieważ od tego momentu Laravel był zależny od komponentów Symfony, ogłoszono, że także on będzie udostępniany w sześciomiesięcznym cyklu wydawniczym.

Laravel 5

W listopadzie 2014 roku miała zostać udostępniona wersja 4.3 Laravela, jednak wraz z postępem prac stało się oczywiste, że znaczenie wprowadzanych w nim zmian zasługiwało na nadanie tej wersji frameworka nowego numeru, i tak w lutym 2015 roku został udostępniony Laravel 5.

W Laravelu 5 wprowadzono nową strukturę katalogów, usunięto funkcje pomocnicze do tworzenia formularzy i kodu HTML, wprowadzono interfejsy kontraktu, nowe widoki, mechanizm Socialite do uwierzytelniania w mediach społecznościowych, mechanizm Elixir do kompilacji zasobów dodatkowych, mechanizm Scheduler do uproszczenia korzystania z systemowego narzędzia cron, dotenv do ułatwienia zarządzania środowiskiem, wprowadzono żądania formularzy i zupełnie nowe narzędzie REPL (pętla odczyt-przetworzenie-wyświetlenie, *read-evaluate-print loop*). Od tamtego czasu Laravel zyskał jeszcze nowe możliwości i dojrzał, jednak nie wprowadzono w nim już żadnych poważniejszych zmian.

Co jest tak wyjątkowe w Laravelu?

A zatem co takiego odróżnia Laravela od konkurentów? Dlaczego warto używać jednocześnie więcej niż jednego frameworka PHP? Przecież większość z nich i tak używa komponentów pochodzących z Symfony, nieprawdaż? Zastanówmy się więc przez chwilę, co sprawia, że Laravel jest taki wyjątkowy.

Filozofia Laravela

Wystarczy rzucić okiem na materiały reklamowe i pliki README Laravela, żeby zacząć doceniać jego wartość. Taylor używa słów związanych ze światłem, takich jak „Illuminate" (oświecać, rozjaśniać, być źródłem natchnienia) i „Spark" (iskra). Ale stosuje także takie określenia jak „Artisans" (rzemieślnicy), „Elegant" (elegancki) i dalej: „Breath of fresh air" (powiew świeżego powietrza), „Fresh start" (nowy początek). I w końcu: „Rapid" (szybki) i „Warp speed" (prędkość nadświetlna).

Dwiema wartościami frameworku, na które kładzie się największy nacisk, są: szybkość tworzenia aplikacji oraz zadowolenie programistów. Taylor opisał język „Artisan" jako celowo odróżniający się na tle innych praktycznych zalet. Genezę tego sposobu myślenia można zauważyć w pytaniu zadanym przez niego w serwisie StackExchange (*https://softwareengineering.stackexchange.com/questions/90954/is-there-any-benefit-to-obsession-with-making-code-look-pretty*), w którym stwierdził: „Czasami spędzam absurdalne ilości czasu (godziny), starając się zapewnić, by «kod wyglądał ładnie»" — stanowiącym próbę znalezienia uzasadnienia dla wkładania wysiłku w dążenie do uzyskania lepszych doznań zapewnianych przez sam wygląd kodu. Oprócz tego często wspominał o znaczeniu, jakie ma ułatwianie programistom realizowania swoich pomysłów oraz eliminacji niepotrzebnych barier na drodze do tworzenia wspaniałych produktów.

Laravel w swojej istocie jest frameworkiem, który daje programistom wielkie możliwości i potrzebne narzędzia. Jego celem jest dostarczanie przejrzystego, prostego i pięknego kodu oraz możliwości, które programiści będą mogli szybko poznać, opanować i zastosować do tworzenia własnego kodu — przejrzystego, prostego i trwałego.

Koncepcja tworzenia z myślą o programistach jest wyraźnie zauważalna w materiałach dotyczących Laravela. W dokumentacji frameworku napisano, że „najlepszy kod tworzą szczęśliwi programiści". Przez jakiś czas nieoficjalnym sloganem frameworku było „szczęście programistów od momentu rozpoczęcia pisania aż do wdrożenia". Oczywiście, twórcy każdego narzędzia i frameworku będą twierdzić, że zależy im na szczęściu programistów. Jednak nadawaniu temu zagadnieniu *pierwszoplanowego*, nie drugorzędnego znaczenia miało ogromny wpływ na styl i rozwój Laravela. O ile dla innych frameworków pierwszoplanowym celem może być czystość architektury lub zgodność z celami i wartościami korporacyjnych zespołów programistycznych, w przypadku Laravela podstawowym celem jest służenie pojedynczym programistom. Nie oznacza to wcale, że Laravel nie pozwala na tworzenie aplikacji o czystej architekturze lub gotowych do zastosowań korporacyjnych, jednak w jego przypadku uzyskanie tych celów nie będzie się odbywać kosztem czytelności i łatwości analizy bazy kodu.

Jak Laravel zapewnia programistom szczęście?

Stwierdzenie, że zależy nam na szczęściu programistów, to jedna sprawa, ale zupełnie czym innym jest faktycznie o nie zadbać — wymaga to bowiem znalezienia odpowiedzi na pytanie, jakie cechy frameworku w największym stopniu uprzykrzają programistom życie, a które będą ich uszczęśliwiać. Laravel stara się ułatwiać pracę programistom na kilka sposobów.

Przede wszystkim Laravel jest frameworkiem służącym do szybkiego tworzenia aplikacji. Oznacza to, że koncentruje się on na zapewnieniu łagodnej (płytkiej) krzywej trudności nauki oraz na minimalizacji kroków, jakie trzeba wykonać od rozpoczęcia prac nad nową aplikacją do jej opublikowania. Wszystkie czynności najczęściej wykonywane podczas tworzenia aplikacji internetowych, zaczynając od interakcji z bazami danych, przez uwierzytelnianie i obsługę poczty elektronicznej, a kończąc na wykorzystaniu pamięci podręcznej, są upraszczane przez komponenty udostępniane przez framework. Jednak komponenty Laravela nie są tak wspaniałe same z siebie; zapewniają one spójny API i przewidywalne struktury w obrębie całej aplikacji. Oznacza to, że kiedy będziemy próbowali zrobić w Laravelu coś nowego, najprawdopodobniej skończy się to stwierdzeniem „… i to po prostu działa".

Jednak nie chodzi o sam framework. Laravel udostępnia cały ekosystem narzędzi służących do budowania i uruchamiania aplikacji. Do prac programistycznych służą Homestead i Valet, do zarządzania serwerem — Forge, a do zaawansowanego wdrażania — Envoyer. Do tego dochodzi zestaw dodatkowych pakietów: Cashier do obsługi płatności i subskrypcji, Echo do obsługi technologii WebSockets, Scout do obsługi wyszukiwania, Passport do obsługi API służących do uwierzytelniania, Dusk do testowania interfejsów użytkownika, Socialite do obsługi logowania w mediach społecznościowych, Horizon do monitorowania kolejek, Nova do budowania paneli administracyjnych oraz Spark do ułatwiania tworzenia rozwiązań typu SaaS[1]. Laravel stara się umożliwić programistom uniknięcie wykonywania powtarzających się czynności, tak by mogli skoncentrować się na tym, co jest unikalne.

Oprócz tego Laravel stosuje podejście „konwencje ponad konfigurację". Oznacza to, że jeśli będziemy chcieli ograniczyć się do domyślnego sposobu działania Laravela, będziemy musieli zrobić znacznie mniej niż w przypadku stosowania innych frameworków, które zmuszają do deklarowania wszystkich ustawień niezależnie od tego, czy programista chce użyć zalecanej konfiguracji. Stworzenie projektu korzystającego z Laravela zajmuje mniej czasu niż stworzenie analogicznego rozwiązania korzystającego z większości innych frameworków PHP.

Co więcej, Laravel w bardzo dużym stopniu koncentruje się na zapewnieniu prostoty. Jeśli chcemy, nic nie stoi na przeszkodzie, by korzystać ze wstrzykiwania zależności, atrapy, repozytoriów, wzorców Data Mapper, Command Query Responsibility Segregation oraz wszelkich innych bardziej zaawansowanych wzorców architektonicznych. Jednak w odróżnieniu od innych frameworków, które mogą sugerować stosowanie takich rozwiązań we wszystkich projektach, Laravel, jego dokumentacja i społeczność preferują rozpoczynanie od najprostszych możliwych implementacji — globalnej

[1] *Software as a Service* — oprogramowanie jako usługa — *przyp. tłum.*

funkcji tu, fasady tam, wzorca ActiveRecord gdzieś indziej. W ten sposób programiści mogą tworzyć możliwie najprostsze aplikacje zaspokajające ich potrzeby, bez ograniczania możliwości ich stosowania w bardziej złożonych środowiskach.

Interesującym zagadnieniem pokazującym, w jaki sposób Laravel różni się od innych frameworków PHP, jest to, że jego twórca i społeczność używających go programistów są bardziej związani i zainspirowani językiem Ruby, frameworkiem Ruby on Rails oraz językami programowania funkcyjnego, niż językiem Java. Obecnie w społeczności programistów PHP istnieje silny nurt dążenia do złożoności, tworzenia bardziej rozbudowanego kodu i wykorzystania tych aspektów języka, które przypominają Javę. Laravel wydaje się znajdować po przeciwnej stronie i stara się wykorzystywać bardziej ekspresyjne, dynamiczne i prostsze praktyki programistyczne i możliwości języka.

Społeczność użytkowników Laravela

Jeśli niniejsza książka stanowi pierwszy kontakt Czytelnika ze społecznością użytkowników Laravela, to może oczekiwać czegoś naprawdę szczególnego. Jednym z czynników wyróżniających ten framework, który w wielkim stopniu przyczynił się do jego rozwoju i popularności, jest życzliwa i chętna do pomocy społeczność jego użytkowników. Zaczynając od wideoporadników *Laracast* Jeffreya Waya (*https://laracasts.com/*), przez serwis Laravel News (*https://laravel-news.com*), kanały Slack, IRC i Discord poświęcone Laravelowi, znajomych na Twitterze, blogerów, po podcasty na konferencjach Laracon — Laravel dysponuje bogatą i żywą społecznością, pełną osób, które używały go od samego początku jego istnienia, i takich, które dopiero zaczynają go poznawać. I bynajmniej nie jest to żaden przypadek:

> Od samego początku istnienia Laravela przyświecała mi idea, że wszyscy chcą się czuć czegoś częścią. To naturalny ludzki instynkt, że chcemy należeć i być akceptowanymi przez jakąś grupę podobnie myślących osób. A zatem dzięki zapewnieniu frameworkowi pewnej osobowości i aktywnej działalności w ramach społeczności, takie poczucie może się przekształcić w społeczność.
>
> — Taylor Otwell, wywiad dla Product and Support

Taylor od samego początku rozumiał, że projekt typu *open-source* do odniesienia sukcesu potrzebuje dwóch rzeczy: dobrej dokumentacji i przyjaznej społeczności. I te dwie rzeczy są cechami charakterystycznymi Laravela.

Jak działa Laravel?

Jak na razie wszystko, co napisałem, było całkowicie abstrakcyjne. Można by się zapytać: „A co z kodem?”. Przyjrzyjmy się zatem prostej aplikacji (Listing 1.1.), tak byśmy mogli zobaczyć, jak w praktyce wygląda codzienne stosowanie frameworku Laravel.

Listing 1.1. „Witaj, świecie!” w pliku routes/web.php

```
<?php

Route::get('/', function () {
    return 'Witaj, świecie!';
});
```

Najprostszą rzeczą, jaką można zrobić w aplikacji Laravela, jest zdefiniowanie trasy i zwracanie wyniku za każdym razem, kiedy ktoś się do tej trasy odwoła. Jeśli zainicjujemy na komputerze zupełnie nową aplikację Laravela, zdefiniujemy trasę przedstawioną na listingu 1.1, a następnie uruchomimy aplikację z poziomu katalogu *public*, uzyskamy w pełni funkcjonalną aplikację typu „Witaj, świecie!" (patrz rysunek 1.1).

Rysunek 1.1. Zwracanie łańcucha „Witaj, świecie!" w aplikacji Laravela

W przypadku wykorzystania kontrolerów aplikacja wygląda całkiem podobnie, co pokazałem na listingu 1.2.

Listing 1.2. Aplikacja typu „Witaj, świecie!" napisana przy użyciu kontrolera

```
// Plik: routes/web.php
<?php
Route::get('/', 'WelcomeController@index');

// Plik app/Http/Controllers/WelcomeController.php
<?php

namespace App\Http\Controllers;

class WelcomeController extends Controller
{
    public function index()
    {
        return 'Witaj, świecie!';
    }
}
```

A jeśli powitanie będzie przechowywane w bazie danych, aplikacja także będzie wyglądać podobnie (jak pokazałem na listingu 1.3):

Listing 1.3. Aplikacja typu „Witaj, świecie!" z wieloma powitaniami pobieranymi z bazy danych

```
// Plik routes/web.php
<?php

use App\Greeting;

Route::get('create-greeting', function () {
    $greeting = new Greeting;
    $greeting->body = 'Witaj, świecie!';
    $greeting->save();
});
```

```
Route::get('first-greeting', function () {
    return Greeting::first()->body;
});

// Plik app/Greeting.php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Greeting extends Model
{
    //
}

// Plik database/migrations/2020_07_19_010000_create_greeting_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');
    }
}
```

Przykład przedstawiony na listingu 1.3 może być nieco onieśmielający — jeśli faktycznie tak jest, po prostu go pomiń. O wszystkim, co dzieje się w tych przykładach, dowiesz się z kolejnych rozdziałów książki; jednak już na pierwszy rzut oka widać, że dzięki wykorzystaniu zaledwie kilku wierszy kodu możliwe jest przygotowanie migracji bazy danych, modelu oraz pobranie z bazy rekordów. To naprawdę jest aż tak proste.

Dlaczego Laravel?

A zatem dlaczego Laravel?

Gdyż pozwala nam urzeczywistniać pomysły bez pisania niepotrzebnego kodu, przy użyciu do tego celu nowoczesnych standardów, oddając nam do dyspozycji aktywną społeczność oraz zestaw narzędzi o ogromnych możliwościach.

Jak również dlatego, Drogi Programisto, że zasługujesz, by być szczęśliwym.

Laravel: framework godny profesjonalisty!

Framework Laravel cieszy się rosnącą popularnością wśród programistów aplikacji w języku PHP. Jest szybki, potężny i elastyczny, można się go błyskawicznie nauczyć, a generowany przez niego kod cechują przejrzystość i czytelność. Co więcej, Laravel skupia wokół siebie zaangażowaną społeczność, która z jednej strony systematycznie tworzy nowe narzędzia i pakiety wzbogacające ten framework, z drugiej — chętnie śpieszy z pomocą w rozwiązywaniu problemów. Jest to więc świetny, dynamicznie rozwijający się zestaw narzędzi, który spodoba się każdemu profesjonalnemu programiście PHP, ceniącemu efektywność i wysoką jakość tworzonego kodu.

Ta książka stanowi praktyczne i kompletne wprowadzenie do Laravela (zawiera informacje dotyczące Laravela 5.8). Dzięki niej osoby dysponujące pewnym doświadczeniem w tworzeniu kodu PHP błyskawicznie rozpoczną tworzenie znakomitych aplikacji. Znalazły się tutaj zarówno ogólne informacje na temat korzystania z tego frameworka, jak i przykłady szczegółowych zastosowań. Omówiono także mnóstwo narzędzi i bibliotek wzbogacających funkcjonalność Laravela: Dusk, Horizon, Artisan, Mix czy Passport. Opisano też interfejsy pozwalające na dostęp do systemu plików, sesji, ciasteczek, pamięci podręcznej i wyszukiwania oraz narzędzia do korzystania z kolejek, implementacji zadań, zdarzeń i publikowania zdarzeń WebSocket.

W książce między innymi:

- tworzenie szablonów w Blade
- generowanie, walidacja, normalizacja oraz filtrowanie danych użytkownika
- praca z bazami danych za pomocą Eloquent
- testowanie kodu PHP: PHPUnit, Mockery oraz Dusk
- tworzenie API typu RESTful
- inne narzędzia i biblioteki Laravela

Matt Stauffer — programista i trener, uważany za jednego z najlepszych znawców Laravela. Pracuje w firmie konsultingowej Tighten, w której zajmuje stanowisko dyrektora technicznego. Uwielbia rozmawiać o programowaniu i blogować, jest też twórcą i gospodarzem *The Five-Minute Geek Show*. Często występuje jako prelegent na branżowych konferencjach.

ISBN 978-83-283-7633-5
9 788328 376335