



Technologia i rozwiązania

Laravel 4

Podstawy tworzenia aplikacji w PHP

Przewodnik dla początkujących!



Raphaël Saunier

[PACKT] open source*
PUBLISHING community experience distilled

Tytuł oryginału: Getting Started with Laravel 4

Tłumaczenie: Rafał Jońca

ISBN: 978-83-283-0298-3

Copyright © 2014 Packt Publishing.

First published in the English language under the title 'Getting Started with Laravel 4'.

© 2015 Helion S.A.

All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION

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)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/larave.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/larave>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

| | |
|---|-----------|
| Przedmowa | 11 |
| <hr/> | |
| Rozdział 1. Poznaj Laravel | 15 |
| <hr/> | |
| Potrzeba stosowania frameworków | 16 |
| Ograniczenia tworzonych przez siebie narzędzi | 16 |
| Laravel przybywa na ratunek | 16 |
| Nowe podejście do tworzenia aplikacji w języku PHP | 17 |
| Znacznie przyjaźniejsza obsługa HTTP | 17 |
| Wykorzystanie pełni możliwości języka PHP | 17 |
| Główne funkcje Laravel i źródła ich inspiracji | 18 |
| Prostota i zwieżłość | 20 |
| Odpowiedzialność, nazewnictwo i konwencje | 21 |
| Pomoc w staniu się lepszym programistą | 22 |
| Struktura aplikacji Laravel | 23 |
| Kontener aplikacji i cykl życia żądania | 24 |
| Poznanie Laravel | 24 |
| Migracja z wersji 3. do wersji 4. | 25 |
| Podsumowanie | 26 |
| <hr/> | |
| Rozdział 2. Narzędzie Composer | 27 |
| <hr/> | |
| Korzystanie z wiersza poleceń | 28 |
| Jak działa Composer? | 28 |
| Instalacja | 29 |
| Unix (Mac OS X, Linux) | 29 |
| Windows | 30 |
| Tworzenie nowej aplikacji Laravel | 30 |
| Odnajdywanie i instalacja nowych pakietów | 31 |
| Kilka dodatkowych rad | 32 |
| Podsumowanie | 33 |

| | |
|--|-----------|
| Rozdział 3. Pierwsza aplikacja | 35 |
| Tworzenie szkicu aplikacji | 36 |
| Encje, związki i atrybuty | 36 |
| Mapa aplikacji, czyli adresy URL | 36 |
| Uruchomienie aplikacji | 37 |
| Użycie wbudowanego serwera deweloperskiego | 38 |
| Utworzenie pierwszych ścieżek | 38 |
| Ograniczenie parametrów routingu | 39 |
| Wyłapywanie brakujących ścieżek | 39 |
| Obsługa przekierowań | 40 |
| Zwracanie widoków | 41 |
| Przygotowanie bazy danych | 41 |
| Tworzenie modeli Eloquent | 41 |
| Budowanie schematu bazy danych | 42 |
| Wstawienie danych początkowych | 43 |
| Szablony Blade | 44 |
| Wykonanie widoku głównego | 45 |
| Powrót do routingu i adresów URL | 46 |
| Strona podsumowania | 46 |
| Wyświetlenie strony konkretnego kota | 48 |
| Dodanie, edycja i usunięcie danych kota | 49 |
| Podsumowanie | 52 |
| Rozdział 4. Uwierzytelnianie i bezpieczeństwo | 53 |
| Uwierzytelnianie użytkowników | 53 |
| Tworzenie modelu użytkownika | 54 |
| Utworzenie niezbędnego schematu bazy danych | 54 |
| Widoki i ścieżki routingu związane z uwierzytelnieniem | 56 |
| Sprawdzanie danych wejściowych | 59 |
| Zabezpieczanie aplikacji | 60 |
| Atak typu CSRF | 61 |
| Atak typu XSS | 61 |
| Unikanie wstrzyknięcia kodu SQL | 62 |
| Ostrożne korzystanie z masowego przypisywania wartości | 63 |
| Pliki cookies — domyślnie bezpieczne | 63 |
| Wymuszenie protokołu HTTPS przy wymianie danych wrażliwych | 63 |
| Podsumowanie | 64 |
| Rozdział 5. Testy — to łatwiejsze, niż się wydaje | 65 |
| Zalety tworzenia testów | 66 |
| Anatomia testu | 66 |
| Testy jednostkowe PHPUnit | 67 |
| Definiowanie oczekiwanego wyniku za pomocą asercji | 68 |
| Przygotowanie sceny i wyczyszczenie obiektów | 68 |
| Przygotowanie się na wyjątki | 69 |
| Testowanie powiązanych ze sobą klas w pełnej izolacji | 69 |

| | |
|---|------------|
| Testy integracyjne | 70 |
| Testowanie — pobieranie bibliotek | 70 |
| Asercje dotyczące frameworka | 71 |
| Podszywanie się pod użytkowników | 72 |
| Testy z użyciem bazy danych | 72 |
| Sprawdzanie kodu HTML zwracanego przez widok | 73 |
| Podsumowanie | 74 |
| Rozdział 6. Artisan — narzędzie wiersza poleceń | 75 |
| Pobieranie najnowszych zmian | 75 |
| Interakcja i sprawdzanie aplikacji | 76 |
| Zabawa z wewnętrznymi elementami systemu | 77 |
| Tymczasowe wyłączenie aplikacji | 77 |
| Optymalizacja aplikacji | 78 |
| Instalacja poleceń innych twórców | 78 |
| Przyspieszenie prac programistycznych za pomocą generatorów | 78 |
| Wdrażanie aplikacji jednym poleceniem | 80 |
| Tworzenie własnych poleceń artisan | 81 |
| Tworzenie polecenia | 82 |
| Anatomia polecenia | 82 |
| Napisanie własnego polecenia | 83 |
| Podsumowanie | 85 |
| Rozdział 7. Projektowanie zaawansowanych aplikacji | 87 |
| Przejście z prostych funkcji routingu do rozbudowanych kontrolerów | 88 |
| Faworyzowanie jawnego routingu | 89 |
| Łatwe tworzenie adresów typu REST | 89 |
| Rozbudowa modeli | 90 |
| Sztuczki związane z wydajnością | 90 |
| Zabezpieczanie danych miękkimi usunięciami | 91 |
| Większa kontrola nad SQL | 92 |
| Nasłuchiwanie zdarzeń dotyczących modelu | 92 |
| Przydatna klasa paginacji | 93 |
| Łatwa konfiguracja środowiska | 93 |
| Artisan i środowiska | 94 |
| Dodawanie własnych ustawień w plikach konfiguracji | 95 |
| Stosowanie własnych klas | 95 |
| Wygodna współpraca z kodem po stronie klienta | 96 |
| Podsumowanie | 97 |
| Dodatek A. Arsenal narzędzi | 99 |
| Funkcje pomocnicze dla tablic | 99 |
| Przykłady użycia funkcji pomocniczych dla tablic | 100 |
| Obróbka tekstu | 101 |
| Funkcje boolowskie | 102 |
| Funkcje przekształcające | 102 |
| Funkcje odmiany | 102 |

| | |
|---|------------|
| Obsługa plików | 103 |
| Przesyłanie plików | 103 |
| Modyfikacja zawartości plików | 104 |
| Wysyłanie e-maili | 105 |
| Biblioteka Carbon, czyli łatwiejsza obsługa daty i czasu | 106 |
| Tworzenie obiektów Carbon | 106 |
| Wyświetlanie znaczników czasowych przyjaznych dla użytkownika | 107 |
| Metody zwracające wartości boolowskie | 107 |
| Obsługa Carbon we właściwościach DateTime modeli Eloquent | 107 |
| Kolejki, czyli wykonywanie długich zadań w tle | 108 |
| Utworzenie zadania i wysłanie go do kolejki | 108 |
| Nasłuchiwanie kolejki i wykonywanie zadań | 109 |
| Informacja o błędzie w wykonaniu zadania | 109 |
| Kolejka bez procesu działającego w tle | 109 |
| Co dalej? | 110 |
| Skorowidz | 111 |

Narzędzie Composer

W poprzednim rozdziale dowiedzieliśmy się, że Laravel powstał na bazie kilku pakietów niezależnych twórców. Zamiast dołączyć zewnętrzne zależności do własnego kodu źródłowego, Laravel korzysta z narzędzia **Composer**, zarządcy zależności, który pobiera i aktualizuje wszystkie pakiety. Z punktu widzenia narzędzia Laravel jest po prostu jeszcze jedną zależnością, którą należy zainstalować.

W tym rozdziale omówimy następujące tematy:

- problemy rozwiązywane przez zarządcę zależności,
- instrukcje instalacji zarządcy w systemach Windows oraz Unix (Mac OS X i Linux),
- tworzenie nowego projektu Laravel za pomocą Composer,
- znajdowanie i instalacja pakietów dodatkowych,
- ogólne wskazówki związane z korzystaniem z Composer.

Composer został zainspirowany przez innych popularnych zarządców zależności z innych języków programowania, np. Bundler ze społeczności Ruby lub npm wykorzystywany przez **node.js** oraz przeglądarkowe projekty JavaScript, i stara się zapewnić podobną funkcjonalność językowi PHP. Domyślnie pakiety nie są instalowane globalnie — podstawowy sposób działania to instalacja pakietów na potrzeby konkretnego projektu. Jeśli pewne zależności trzeba zainstalować w całym systemie PHP, skorzystaj z **PEAR** — zarządcy pakietów dostarczanego domyślnie wraz z PHP.

Oto kilka podstawowych zalet korzystania z zarządcy zależności we własnym projekcie:

- obsługa staje się znacznie szybsza, bo nie trzeba samodzielnie szukać, pobierać i rozpakowywać poszczególnych pakietów;
- unika się konfliktów wersji w trakcie aktualizacji poszczególnych zależności;
- wczytywanie poszczególnych klas odbywa się automatycznie bez udziału programisty;
- odkrywanie i wybór odpowiednich pakietów okazuje się znacznie łatwiejsze dzięki centralnemu repozytorium.

Korzystanie z wiersza poleceń

Jeśli dopiero zaczynasz swoją przygodę z programowaniem, prawdopodobnie nigdy wcześniej nie miałeś do czynienia z **interfejsem wiersza poleceń (CLI)**. Korzystanie z narzędzia Composer, a w dalszej części książki również z narzędzia **Artisan** specyficznego dla Laravel, wymaga znajomości przynajmniej podstaw obsługi wiersza poleceń.

Oto opis sposobu uruchomienia wiersza poleceń w poszczególnych systemach.

1. W systemie Windows poszukaj programu *Wiersz polecenia*. Jeśli go nie znajdziesz, użyj polecenia *Start/Uruchom* (w systemie Windows 8.1 kliknij ikonę Windows prawym klawiszem myszy) i wpisz w polu tekst `cmd.exe`.
2. W systemie Mac OS X wiersz poleceń nosi nazwę *Terminal*. Znajdziesz go w folderze `/Applications/Utilities`.
3. W systemie Linux nazwa zależy od dystrybucji. Najczęściej nosi ona nazwę *Terminal* lub *Konsola*. Większość osób korzystających z tego systemu używa wiersza poleceń stosunkowo często.

Obsługa Laravel opisywana w książce nie wymaga żadnej zaawansowanej wiedzy na temat wiersza poleceń. Wystarczy znajomość sposobu przedostania się do odpowiedniego folderu w systemie plików przed wykonaniem wskazanych poleceń. W tym celu wpisz polecenie `cd`, a następnie podaj ścieżkę docelową.

W większości systemów można również wpisać polecenie `cd`, dodać znak spacji, a następnie przeciągnąć odpowiedni folder do okna terminalu.

W systemie Windows należy wykonać polecenie podobne do wskazanego poniżej.

```
> cd C:\ścieżka\do\folderu\z\kodem
```

Jeżeli jakiś przykład będzie dotyczył nie tylko systemu Windows, to w dalszej części książki do oznaczenia polecenia wykorzystamy znak `$`, a poszczególne foldery będą oddzielane znakiem ukośnika. Pamiętaj o samodzielnym dostosowaniu polecenia do systemu Windows.

Jak działa Composer?

Composer (<http://getcomposer.org/>) to plik wykonywalny PHP dodawany do zmiennej środowiskowej `PATH` (zawiera listę folderów, w których system szuka poleceń do wykonania). Po odpowiedniej instalacji z dowolnego folderu można wywołać polecenie `composer` uruchamiające zarządcę pakietów. Projekt wraz z zależnościami jest zdefiniowany w pliku JSON o nazwie `composer.json`. Composer odczytuje zawartość pliku i łączy się z internetowym repozytorium **Packagist** (<https://packagist.org/>), aby w sposób rekurencyjny pobrać oraz rozwiązać wszystkie zależności.

Następnie zależności są pobierane do lokalnego folderu (w Laravel jest to folder *vendor/*), a aktualny stan zależności zostaje zapisany w pliku *composer.lock*. Composer generuje również plik w folderze *vendor/*, który zawiera obsługę automatycznego wczytywania klas jako skrypt PHP (we właściwym kodzie aplikacji skrypt uruchamia polecenie `require 'vendor/autoload.php'`).

Instalacja

W systemach typu Unix i w systemie Windows instalacja Composer nie jest trudna, gdyż udostępniono instalator lub skrypty instalacyjne.

Unix (Mac OS X, Linux)

Przed wszystkim musimy się upewnić, że plik wykonywalny `php` jest dostępny z poziomu wiersza poleceń. W tym celu otwórz okno terminalu i wpisz następujące polecenie.

```
$ php -v
```

Powinna pojawić się informacja na temat wersji PHP aktualnie zainstalowanej w systemie. Jeśli pojawi się błąd informujący o nieznanym poleceniu lub też wersja zainstalowanego PHP okaże się zbyt niska (minimum to wersja 5.3.7), skorzystaj ze wskazówek dotyczących instalacji PHP w danym systemie dostępnych na stronie <http://php.net>. Jeśli w systemie Mac OS do instalacji użyto **MAMP** lub **Homebrew**, upewnij się, że zmienna środowiskowa `PATH` zawiera w pierwszej kolejności zainstalowaną wersję zamiast domyślnej wersji z systemu operacyjnego.

Gdy mamy pewność, że odpowiednia wersja PHP jest dostępna z poziomu wiersza poleceń, należy zainstalować Composer za pomocą poniższego polecenia:

```
$ curl -sS https://getcomposer.org/installer | php
```

Aby instalacja była dostępna z poziomu dowolnego folderu, przenieś ją do katalogu znajdującego się w zmiennej `PATH`. Jeśli pojawi się błąd dostępu, wybierz inny folder, do którego masz prawo zapisu. Ewentualnie skorzystaj z polecenia `sudo`.

```
$ sudo mv composer.phar /usr/local/bin/composer
```

Aby mieć pewność, że wszystko zostało zainstalowane prawidłowo, ponownie otwórz okno terminalu i wpisz następujące polecenie:

```
$ composer
```

Spowoduje to wyświetlenie listy wszystkich dostępnych poleceń narzędzia.

Windows

Aby zainstalować Composer w systemie Windows, wystarczy pobrać plik *Composer-Setup.exe* dostępny pod adresem <https://getcomposer.org/download/>. Oczywiście zadziała on tylko wtedy, gdy jest zainstalowana odpowiednia wersja PHP.

Instalator poprosi o wskazanie lokalizacji pliku wykonywalnego PHP. Oto kilka typowych lokalizacji:

- domyślna — *C:\PHP5\php.exe* lub *C:\PHP\php.exe*,
- przy użyciu XAMPP — *C:\xampp\php\php.exe*,
- przy użyciu WAMP — *C:\wamp\bin\php\php5.x.x\php.exe*.

Jeśli plik wykonywalny nie znajduje się w przedstawionych lokalizacjach, ale masz pewność, że został zainstalowany w systemie, użyj narzędzia wyszukiwania, aby znaleźć plik *php.exe*.

Instalator zajmie się pobraniem i zainstalowaniem narzędzia Composer, doda także polecenia `php` i `composer` do zmiennej środowiskowej `PATH`.

Aby się upewnić, że instalacja przebiegła pomyślnie, otwórz nowy wiersz poleceń i wpisz dwa polecenia przedstawione poniżej.

```
> php -v
> composer
```

Oba polecenia powinny wykonać się bez błędów, a następnie wyświetlić informację o wersji oraz dostępnych poleceniach narzędzia.

Tworzenie nowej aplikacji Laravel

Po zainstalowaniu narzędzia Composer utworzenie nowego projektu Laravel jest dziecinnie proste. Po przejściu do folderu, w którym chcesz utworzyć projekt, wykonaj następujące polecenie:

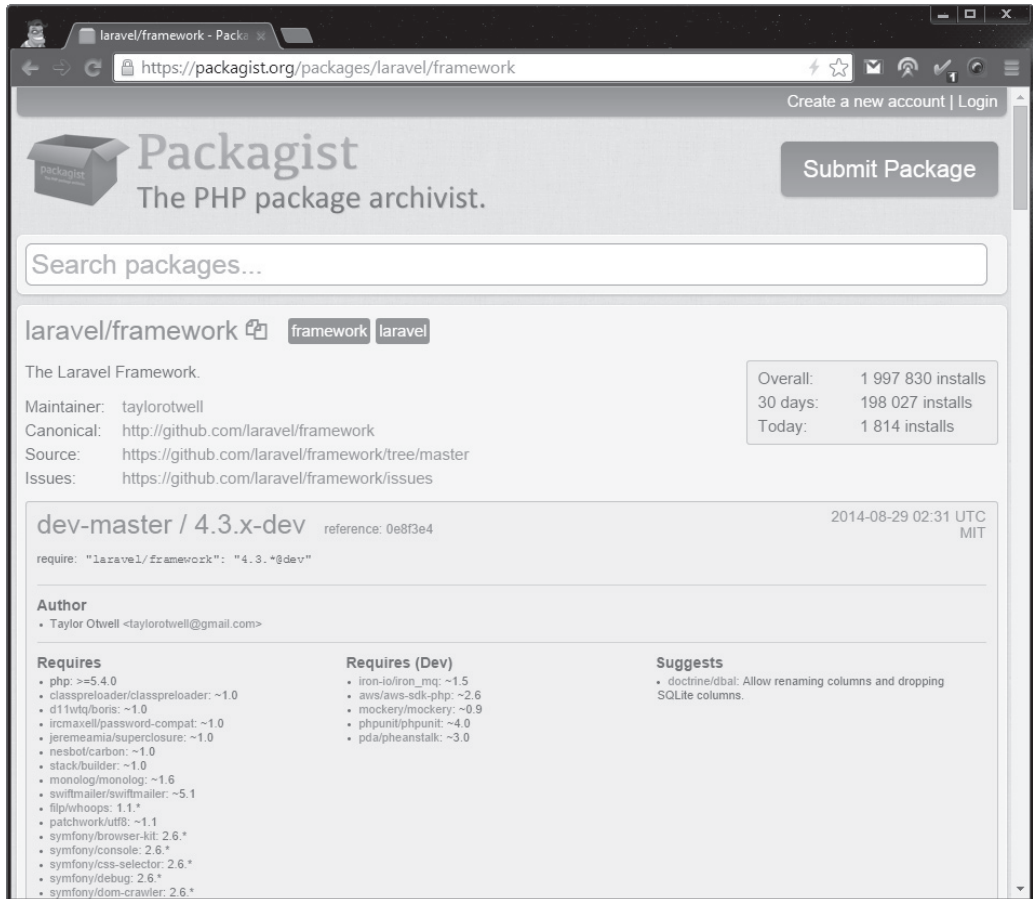
```
$ composer create-project laravel/laravel --prefer-dist
```

Spowoduje to pobranie najnowszej wersji Laravel wraz z zależnościami. Ponieważ szybkość łącza internetowego i moc procesora wpływają na pobranie i instalację, ta może zająć nawet kilka minut. Po zakończeniu całego procesu w folderze pojawi się cała struktura katalogów opisana w poprzednim rozdziale oraz gotowa do uruchomienia projektu.

Jeśli korzystasz z systemu kontroli wersji Git, warto w tym momencie uruchomić polecenie `git init`. Główny folder zawiera odpowiednio przygotowany plik *.gitignore*, a puste foldery zawierają pliki *.gitkeep*.

Odnajdywanie i instalacja nowych pakietów

Strona wyszukiwania dostępna pod adresem <http://packagist.org> ułatwia odnalezienie i dodanie pakietów oferujących przydatne funkcje, np. obróbkę obrazów lub generowanie plików PDF. Wskaźnikami, czy pakiet wart jest uwagi, są nie tylko liczba pobrań i liczba gwiazdek w serwisie GitHub, ale również jakość dokumentacji, pokrycie testami lub ogólna aktywność projektu. Przed dodaniem pakietu można na stronie Packagist sprawdzić jego różne wersje i zależności.



The screenshot shows the Packagist website interface for the `laravel/framework` package. The page includes a search bar, a 'Submit Package' button, and detailed information about the package. The package is identified as 'The Laravel Framework' and is maintained by Taylor Otwell. It lists the canonical URL, source code repository, and issues page. The current version shown is `dev-master / 4.3.x-dev` with a reference hash of `0e8f3e4`. The package requires PHP `>=5.4.0` and lists various dependencies under 'Requires' and 'Requires (Dev)'. It also suggests the `doctrine/dbal` package. Installation statistics are provided: Overall (1 997 830 installs), 30 days (198 027 installs), and Today (1 814 installs). The package was last updated on 2014-08-29 02:31 UTC by MIT.

W trakcie prac programistycznych można z powodzeniem stosować wersję `dev-master`, ale przed udostępnieniem wersji produkcyjnej warto wskazać konkretny numer wersji, aby uniknąć potencjalnych problemów ze zgodnością.

Aby zainstalować pakiet, w edytorze tekstu otwórz plik `composer.json` i w obiekcie `require` wstaw nazwę pakietu oraz wymaganą wersję.

```
"require": {
    "laravel/laravel": "v4.1.*",
    "intervention/image": "dev-master"
}
```

Ponieważ jest to plik JSON, uważaj, aby przypadkiem nie pozostawić na końcu żadnych zbędnych znaków przecinka. Użyj polecenia `composer validate`, aby sprawdzić, czy plik `composer.json` nie zawiera żadnych błędów.

Następnie wykonaj polecenie `composer install`, a Composer pobierze pakiet i wszystkie jego zależności. Aby zaktualizować plik `composer.lock` i zapisać dokładne numery wersji poszczególnych zależności, wykonaj polecenie `composer update`. W momencie wdrożenia lub rozpowszechniania aplikacji plik z wersjami umożliwi innym osobom pobranie za pomocą polecenia `composer install` dokładnie tych samych wersji pakietów. Polecenie `update` zawsze poszukuje najnowszych wersji każdego pakietu, co w przypadku systemu produkcyjnego może doprowadzić do błędów związanych ze zgodnością między pakietami.

Kilka dodatkowych rad

Przed rozpoczęciem tworzenia pierwszej aplikacji Laravel warto zaznajomić się z kilkoma wskazówkami pomagającymi w codziennym korzystaniu z narzędzia Composer.

- Umieść plik `composer.lock` w systemie kontroli wersji, aby uniknąć pobierania niezgodnych ze sobą wersji pakietów i upewnić się, że wszyscy programiści korzystają z pakietów w dokładnie tych samych wersjach.
- Nie trzeba umieszczać zawartości folderu `vendor/` w systemie kontroli wersji. Domyślnie jest on wyłączony z systemu kontroli wersji Git za pomocą odpowiedniego wpisu w pliku `.gitignore`. Dodanie go do systemu kontroli wersji zwiększyłoby jedynie rozmiar repozytorium bez żadnych istotnych korzyści. Dopóki repozytorium zawiera pliki `composer.json` i `composer.lock`, każdy będzie mógł odtworzyć tę samą zawartość folderu z pakietami.
- Nie należy edytować plików znajdujących się w folderze `vendor/`, ponieważ mogą zostać automatycznie nadpisane przy następnym użyciu polecenia `composer install`.
- Composer w trakcie instalacji pakietu udostępnia dwie opcje — `--prefer-source` i `--prefer-dist`. Różnica między nimi polega na tym, że w przypadku drugiej z nich Composer będzie preferował pobieranie wydań stabilnych i w miarę możliwości nie będzie pobierał całej historii z repozytorium Git.
- Łączny rozmiar folderu `vendor/` będzie oscylował w granicach 25 MB w przypadku opcji `--prefer-dist` i mniej więcej trzykrotnie więcej w przypadku opcji `--prefer-source`, ponieważ zostanie pobrana pełna historia z repozytorium Git. Po umieszczeniu aplikacji Laravel na serwerze należy na nim uruchomić polecenie `composer install`. Jeśli jedyną dostępną opcją jest przesyłanie wszystkich plików za pomocą protokołu

FTP, skorzystaj z pakietów takich jak `barryvdh/laravel-vendor-cleaner`, które usuną wszystkie zbędne pliki przed umieszczeniem aplikacji na serwerze.

- Polecenie `diagnose` narzędzia Composer oraz opcje dodatkowych informacji (`-v`, `-vv` i `-vvv`) pomogą zidentyfikować typowe problemy i poprosić o pomoc na forum, IRC lub w witrynie **Stack Overflow**.

Podsumowanie

W tym rozdziale wyjaśniliśmy, jakie problemy rozwiązuje użycie zarządcy zależności. Zainstalowaliśmy narzędzie Composer i utworzyliśmy pierwszy projekt Laravel. Dowiedzieliśmy się również, w jaki sposób znaleźć i zainstalować pakiet oraz jak uniknąć typowych błędów związanych z korzystaniem z narzędzia Composer.

W następnym rozdziale rozpocznie się prawdziwa zabawa! Po uzyskaniu w pełni działającej wersji narzędzia Composer przejdziemy przez wszystkie kroki niezbędne do utworzenia w pełni działającej aplikacji Laravel.

Skorowidz

A

- adresy
 - typu REST, 89
 - URL, 36, 46
- akceleratory PHP, 78
- anatomia
 - polecenia, 82
 - testu, 66
- API, 25, 89
- aplikacje zaawansowane, 87
- Artisan, 28, 74, 85, 94
- asercje, 68
- asercje dotyczące frameworka, 71
- atak
 - typu CSRF, 61
 - typu XSS, 61
- atrapy, 69
- atrybuty, 36

B

- baza danych, 41
- bezpieczeństwo, 53
- bezpośredniość PHP, 16
- biblioteka
 - Carbon, 17, 106
 - Doctrine, 17
 - Mockery, 70
 - SwiftMailer, 17
- Blade, 44
- Breed, 42

C

- Carbon, 17, 106
- certyfikat SSL, 63
- CLI, interfejs wiersza poleceń, 28
- CodeIgniter, 15
- Composer, 25–29, 71
- CRUD, Create-Retrieve-Update-Delete, 36
- CSRF, Cross-Site Request Forgery, 61
- cykl życia żądania, 24

D

- dane wrażliwe, 63
- Doctrine, 17
- dodawanie danych, 49
- dokumentacja, 25
- domknięcia, closure, 18

E

- edycja danych, 49
- elementy systemu, 77
- Eloquent, 41
- e-mail, 105
- encje, 21, 36

F

- fasada, 25
- filtry routingu, 53
- format JSON, 96
- formularze, 50, 80

framework

CodeIgniter, 15

Laravel, 15

Symfony, 15

funkcje

anonimowe, 18

boolowskie, 102

Laravel, 18

odmiany, 102

pomocnicze dla tablic, 99

przekształcające, 102

systemu Eloquent, 42

G

generatory, 78

generowanie

formularzy HTML, 80

migracji, 79

plików, 80

H

Homebrew, 29

I

identyfikator użytkownika, 59

instalacja

Composer, 29

pakietów, 31

PHPUnit, 71

poleceń, 78

interfejsy, 18

J

jawny tekst, 63

język

PHP, 16

szablonów, 44

K

klasa paginacji, 93

klucze obce, 55

kod

404, 40

po stronie klienta, 96

kolejki, 20, 108

kompozytor widoku, 51

komunikat o błędzie, 57, 109

konfiguracja środowiska, 93

konfigurowanie baz danych, 72

kontener aplikacji, 24

kontrakt, 54

kontroler, 88

kontroler zasobów, 21

konwencja PSR-0, 18

konwencje, 21

konwencje kodowania, 25

L

Laravel, 15, 18

kolejki, 20

łatwość testowania, 18

mechanizm szablonów, 19

modularność, 18

obsługa e-maili, 19

ORM, 19

Redis, 19

routing, 19

schemat bazy, 19

uwierzytelnianie, 19

zapytania, 19

zarządzanie konfiguracją, 19

logowanie, 57

Ł

łatwość testowania, 18

M

MAMP, 29
 mapa aplikacji, 36
 masowe przypisywanie wartości, 63
 mechanizm
 przygotowanych zapytań, 62
 szablonów, 19
 metoda
 back(), 57
 be(), 72
 call(), 71
 cats(), 54
 check(), 56
 count(), 93
 fails(), 60
 find(), 48
 setUp(), 68
 metody zwracające wartości boolowskie, 107
 miękkie usunięcia, 91
 migracje, 79
 model, 21, 36, 90
 Breed, 42
 Eloquent, 41
 User, 54
 Mockery, 70
 modularność, 18
 modyfikacja zawartości plików, 104
 możliwości PHP, 17
 MVC, Model-View-Controller, 15, 89

N

narzędzie
 Artisan, 28, 74, 85, 94
 Composer, 25–29, 71
 wiersz poleceń, 28, 75
 nasłuchiwanie
 kolejki, 109
 zdarzeń, 92
 nazewnictwo, 21
 nazwy metod, 26

O

obróbka tekstu, 101
 obsługa
 Carbon, 107
 daty i czasu, 106
 e-maili, 19
 formatu JSON, 96
 HTTP, 17
 plików, 103
 przekierowań, 40
 odpowiedzialność, 21
 ograniczenie
 parametrów routingu, 39
 wartości, 55
 optymalizacja aplikacji, 78
 ORM, odwzorowanie obiektowo-relacyjne, 19

P

pakiety, 25, 31
 parametry routingu, 39
 PDO, PHP's Data Object, 62
 PEAR, 27
 PHPUnit, 71
 pierwsza aplikacja, 35
 plik
 blade.php, 50
 composer.json, 95
 database.php, 72
 filter.php, 57
 phpunit.xml, 70
 routes.php, 51
 SupportStrTest.php, 68
 UserController.php, 88
 pliki
 cookies, 63
 konfiguracji, 95
 routingu, 46
 polecenie routes, 76
 pomoc, 22
 proces działający w tle, 109
 profilowanie zapytań, 91

projektowanie zaawansowanych aplikacji, 87
 protokół HTTPS, 63
 przeciążanie, 18
 przekierowania, 40
 przestrzenie nazw, 17
 przesyłanie plików, 103
 przygotowanie bazy danych, 41

R

redis, 19
 repozytorium Packagist, 28
 REST, 22, 89
 routing, 19, 21, 39, 46
 routing jawny, 89
 rozbudowa modeli, 90

S

schemat bazy danych, 19, 36, 42, 54
 serwer deweloperski, 38
 składnia dla tablic, 18
 sprawdzanie
 danych, 59
 funkcji aplikacji, 66, 76
 kodu HTML, 73
 SQL, 92
 stosowanie własnych klas, 95
 strona
 błędu, 39
 podsumowania, 46
 struktura aplikacji Laravel, 23
 SwiftMailer, 17
 Symfony, 15
 szablony, 22
 szablony Blade, 44

Ś

ścieżki, 38
 ścieżki routingu, 56

T

test PHPUnit, 67
 testowanie powiązanych klas, 69
 testy, 65
 integracyjne, 70
 jednostkowe, 67
 z użyciem bazy danych, 72
 tworzenie
 adresów REST, 89
 API, 89
 aplikacji, 17, 35
 aplikacji Laravel, 30, 32
 kodu SQL, 92
 modeli Eloquent, 41
 modelu użytkownika, 54
 obiektów Carbon, 106
 polecenia, 82
 schematu bazy, 42, 54
 szkicu aplikacji, 36
 ścieżek, 38
 testów, 66
 układu graficznego, 45
 własnych poleceń, 81
 zadania, 108
 tymczasowe
 dane, 45
 wyłączenie aplikacji, 77

U

upiększanie PHP, 20
 uruchomienie aplikacji, 37
 User, 54
 usuwanie danych, 49
 uwierzytelnianie, 19, 53, 56

W

wartości boolowskie, 107
 wdrażanie aplikacji, 80, 81
 widok, 22, 41, 56
 główny, 45
 logowania, 57
 podrzędny, 46

- wiersz poleceń, 28, 75
- własne
 - polecenia, 83
 - ustawienia, 95
- wstawienie danych, 43
- wstrzyknięcie
 - zależności, 25
 - kodu SQL, 62
- wydajność, 90
- wyjątki, 39, 69
- wykonywanie zadań, 109
- wyłączenie aplikacji, 77
- wymuszenie protokołu HTTPS, 63
- wynik
 - oczekiwany, 68
 - rzeczywisty, 68

- wyprzedzające wczytywanie rekordów, 90
- wysyłanie e-maili, 105
- wyświetlanie
 - znaczników czasowych, 107
 - strony, 48

Z

- zabezpieczanie
 - aplikacji, 60
 - danych, 91
- zadanie, 108
- zapytania, 19
- zarządzanie konfiguracją, 19
- zdarzenia modelu, 92
- znaczniki czasowe, 107
- związki, 36
- zwracanie widoków, 41

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

Laravel 4

Podstawy tworzenia aplikacji w PHP

Laravel to szkielet aplikacji dla języka PHP, zdobywający w ostatnim czasie ogromną popularność. Powody są dwa: obszerna i przejrzysta dokumentacja oraz podstawowe założenie autorów, że tworzenie aplikacji z wykorzystaniem Laravela ma być po prostu przyjemne. I tak jest! Łatwe zarządzanie bazą danych, przyjazny system szablonów, intuicyjna konfiguracja to tylko niektóre z zalet tego szkieletu.

Jeżeli chcesz poznać w pełni jego potencjał i wykorzystać go w trakcie tworzenia kolejnej strony WWW lub aplikacji internetowej, to trafiłeś na doskonałą książkę. Sięgnij po nią i już wkrótce bez problemu przygotujesz swoje środowisko pracy, zainstalujesz najnowszą wersję za pomocą narzędzia Composer, a następnie rozpoczniesz prace nad swoim projektem. Przekonaj się, jak tworzyć ścieżki, konfigurować bazę danych oraz zabezpieczać dostęp do wybranych stron. Dzięki lekturze kolejnych rozdziałów nauczysz się tworzyć testy automatyczne oraz korzystać z narzędzia Artisan, które ułatwi Ci codzienne zadania! Ponadto dowiesz się, w jaki sposób tworzyć serwisy typu REST oraz generować formularze. Książka ta jest idealną pozycją dla wszystkich programistów PHP szukających frameworka, który spełni ich oczekiwania!



Dzięki tej książce:

- poznasz narzędzie Composer
- pobierzesz najnowszą wersję szkieletu Laravel
- przygotujesz ścieżki dostępu do stron
- przetestujesz stworzony kod
- błyskawicznie zbudujesz pierwszą aplikację WWW

Tworzenie aplikacji w PHP jeszcze nigdy nie było takie proste!

[PACKT] open source
PUBLISHING community experience distilld

Helion

27583 numer katalogowy

księgarnia internetowa

<http://helion.pl>

zamówienia telefoniczne

☎ 0 801 339900

☎ 0 601 339900

Sprawdź najnowsze promocje:

● <http://helion.pl/promocje>

Książki najczęściej czytane:

● <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

● <http://helion.pl/nawosci>

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 43
e-mail: helion@helion.pl
<http://helion.pl>

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 978-83-283-0298-3



9 788328 302983

Informatyka w najlepszym wydaniu

cena: 34,00 zł