

O'REILLY®

Übersetzung  
der 2. US-Auflage

# Laravel

## Die umfassende Einführung

Das Framework für  
moderne PHP-Entwicklung



Matt Stauffer

Übersetzung von Jens Olaf Koch

Papier  
**plus<sup>+</sup>**  
PDF.

Zu diesem Buch – sowie zu vielen weiteren O'Reilly-Büchern – können Sie auch das entsprechende E-Book im PDF-Format herunterladen. Werden Sie dazu einfach Mitglied bei oreilly.plus<sup>+</sup>:

[www.oreilly.plus](http://www.oreilly.plus)

ÜBERSETZUNG DER 2. AMERIKANISCHEN AUFLAGE

---

# Laravel

## Die umfassende Einführung

*Das Framework für moderne PHP-Entwicklung*

*Matt Stauffer*

*Deutsche Übersetzung von  
Jens Olaf Koch*

**O'REILLY®**

Matt Stauffer

Lektorat: Ariane Hesse

Übersetzung: Jens Olaf Koch

Fachlicher Review: Mitarbeiter von mindtwo, [www.mindtwo.de](http://www.mindtwo.de)

Korrektur: Claudia Lötschert, [www.richtiger-text.de](http://www.richtiger-text.de)

Satz: III-satz, [www.drei-satz.de](http://www.drei-satz.de)

Herstellung: Stefanie Weidner

Umschlaggestaltung: Michael Oreal, [www.oreal.de](http://www.oreal.de)

Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN:

Print 978-3-96009-129-5

PDF 978-3-96010-374-5

ePub 978-3-96010-372-1

mobi 978-3-96010-373-8

1. Auflage 2020

Translation Copyright der deutschsprachigen Ausgabe © 2020 dpunkt.verlag GmbH

Wiebling Weg 17

69123 Heidelberg

Authorized German translation of the English edition of titled *Laravel: Up & Running*, 2E

ISBN 9781492041214 © 2019 Matt Stauffer.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«.

O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

#### *Hinweis:*

Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.



#### *Schreiben Sie uns:*

Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: [komentar@oreilly.de](mailto:komentar@oreilly.de).

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag noch Übersetzer können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

*Dieses Buch ist meiner Familie gewidmet.  
Mia, meiner kleinen Prinzessin, diesem Bündel voller Freude und Energie.  
Malachi, meinem kleinen Prinzen, Abenteuerer und Empathen.  
Tereva, meiner Inspiration, Ermutigerin, Upgraderin, Motivatorin, Rippe.*



<b>Vorwort</b> .....	<b>XXI</b>
<b>1 Warum Laravel?</b> .....	<b>1</b>
Warum ein Framework verwenden? .....	1
»Ich baue es einfach selbst« .....	2
Konsistenz und Flexibilität .....	2
Eine kurze Geschichte der Web- und PHP-Frameworks .....	2
Ruby on Rails .....	3
Eine Welle von PHP-Frameworks .....	3
Das Gute und das Schlechte an CodeIgniter .....	3
Laravel 1, 2 und 3 .....	4
Laravel 4 .....	4
Laravel 5 .....	4
Laravel 6 .....	5
Was ist so besonders an Laravel? .....	5
Die Philosophie von Laravel .....	5
Wie sich Laravel um die Zufriedenheit des Entwicklers verdient macht .....	6
Die Laravel-Community .....	8
Wie es funktioniert .....	8
Warum also Laravel? .....	10
<b>2 Eine Laravel-Entwicklungsumgebung einrichten</b> .....	<b>11</b>
Systemanforderungen .....	11
Composer .....	12
Lokale Entwicklungsumgebungen .....	12
Laravel Valet .....	12
Laravel Homestead .....	13

Ein neues Laravel-Projekt erstellen . . . . .	14
Installation von Laravel mit dem Laravel-Installationsprogramm . . . . .	14
Installation von Laravel mit dem create-project-Feature von Composer . . . . .	14
Das Helfer-Paket installieren . . . . .	15
Lambo: »laravel new« mit Düsenantrieb . . . . .	15
Die Verzeichnisstruktur von Laravel. . . . .	15
Die Ordner . . . . .	16
Die Dateien . . . . .	17
Konfiguration . . . . .	18
Die .env-Datei . . . . .	19
Achtung, fertig, los! . . . . .	21
Testen . . . . .	22
TL;DR. . . . .	22
<b>3 Routing und Controller . . . . .</b>	<b>23</b>
Eine kurze Einführung in MVC, HTTP-Verben und REST . . . . .	23
Was ist MVC? . . . . .	23
Die HTTP-Verben . . . . .	24
Was ist REST? . . . . .	25
Routendefinitionen . . . . .	26
Routing-Verben . . . . .	28
Routen-Handling . . . . .	28
Routenparameter . . . . .	29
Benannte Routen . . . . .	30
Routen gruppieren . . . . .	33
Middleware . . . . .	33
Pfad-Präfixe . . . . .	36
Fallback-Routen. . . . .	36
Subdomain-Routing. . . . .	36
Namensraum-Präfixe . . . . .	37
Namenspräfixe . . . . .	37
Signierte Routen . . . . .	38
Eine Route signieren . . . . .	38
Signierte Links zulassen . . . . .	39
Views. . . . .	40
Einfache Routen direkt mit Route::view() zurückgeben . . . . .	41
Verwendung von View Comosern, um Variablen für alle Views bereitzustellen . . . . .	41



Controller .....	41
Benutzereingaben .....	44
Abhängigkeiten in Controller einfügen. ....	46
Ressourcen-Controller .....	47
API-Ressourcen-Controller .....	48
Controller für eine einzelne Aktion .....	48
Routen-Modell-Bindung .....	49
Implizite Routen-Modell-Bindung .....	49
Benutzerdefinierte Routen-Modell-Bindung .....	50
Routen-Caching. ....	51
Methoden-Spoofing für Formulare .....	51
HTTP-Verben in Laravel. ....	52
HTTP-Methoden-Spoofing. ....	52
CSRF-Schutz .....	52
Umleitungen .....	54
redirect()->to() .....	55
redirect()->route(). ....	55
redirect()->back() .....	56
Andere Umleitungsmethoden. ....	56
redirect()->with() .....	57
Einen Request abbrechen .....	58
Gebräuchliche Response-Typen .....	59
response()->make(). ....	59
response()->json() und ->jsonp() .....	59
response()->download(), ->streamDownload() und ->file() .....	59
Testen .....	60
TL;DR .....	61
<b>4 Vorlagen erstellen mit Blade .....</b>	<b>63</b>
Daten ausgeben .....	64
Kontrollstrukturen. ....	65
Bedingungen .....	65
Schleifen .....	66
Vorlagen-Vererbung .....	67
Definieren von Abschnitten mit @section/@show und @yield. ...	68
Einbinden von Teilansichten .....	70
Verwendung von Stacks .....	72
Verwendung von Komponenten und Slots. ....	73
View Composer und Service Injection. ....	75
Daten mit View Composern an Views binden .....	76
Service Injection .....	78

Benutzerdefinierte Blade-Direktiven . . . . .	79
Parameter in benutzerdefinierten Blade-Direktiven . . . . .	81
Beispiel: Verwendung benutzerdefinierter Blade-Direktiven für eine mandantenfähige Anwendung . . . . .	81
Einfachere benutzerdefinierte Direktiven für »if«-Anweisungen . . . . .	82
Testen . . . . .	83
TL;DR. . . . .	84
<b>5 Datenbanken und Eloquent . . . . .</b>	<b>85</b>
Konfiguration . . . . .	85
Datenbankverbindungen . . . . .	86
Weitere Optionen zur Konfiguration von Datenbanken . . . . .	87
Migrationen . . . . .	88
Migrationen definieren. . . . .	88
Migrationen ausführen. . . . .	95
Seeding . . . . .	96
Eine Seeder-Klasse anlegen . . . . .	97
Modellfabriken. . . . .	98
Der Query Builder. . . . .	102
Grundlegender Einsatz der DB-Fassade. . . . .	103
Direktes SQL . . . . .	104
Verkettung mit dem Query Builder . . . . .	105
Transaktionen . . . . .	114
Einführung in Eloquent . . . . .	115
Erstellen und Definieren von Eloquent-Modellen . . . . .	116
Abrufen von Daten mit Eloquent. . . . .	118
Inserts und Updates mit Eloquent . . . . .	120
Löschen mit Eloquent . . . . .	124
Geltungsbereiche . . . . .	126
Anpassen von Feldinteraktionen durch Akzessoren, Mutatoren und Attribut-Casting . . . . .	130
Eloquent-Collections . . . . .	133
Serialisierung mit Eloquent . . . . .	135
Beziehungen mit Eloquent . . . . .	137
Aktualisierung von Zeitstempeln durch verknüpfte Datensätze . . . . .	150
Ereignisse in Eloquent. . . . .	153
Testen . . . . .	154
TL;DR. . . . .	156
<b>6 Frontend-Komponenten . . . . .</b>	<b>157</b>
Laravel Mix . . . . .	157
Verzeichnisstruktur von Mix . . . . .	159

Mix ausführen. . . . .	159
Was bietet Mix? . . . . .	160
Frontend-Frameworks und Auth-Scaffolding . . . . .	167
laravel/ui . . . . .	167
Frontend-Presets . . . . .	168
Auth-Scaffolding . . . . .	169
Paginierung . . . . .	169
Paginieren von Datenbank-Ergebnissen . . . . .	169
Paginator manuell erstellen . . . . .	170
Message Bags . . . . .	171
Benannte Error Bags . . . . .	173
Hilfsfunktionen für Strings, Pluralisierung und Lokalisierung . . . . .	173
Zeichenketten-Helfer und Pluralisierung . . . . .	173
Lokalisierung . . . . .	175
Testen . . . . .	179
Message und Error Bags testen . . . . .	179
Übersetzung und Lokalisierung . . . . .	179
TL;DR . . . . .	179
<b>7 Benutzereingaben erfassen und verarbeiten . . . . .</b>	<b>181</b>
Injizieren eines Anforderungsobjekts . . . . .	181
\$request->all() . . . . .	182
\$request->except() und \$request->only() . . . . .	182
\$request->has() . . . . .	183
\$request->input() . . . . .	183
\$request->method() und ->isMethod() . . . . .	184
Benutzereingaben in Array-Form . . . . .	184
JSON-Input (und \$request->json()) . . . . .	184
Routendaten . . . . .	186
Daten aus dem Request-Objekt extrahieren . . . . .	186
Daten aus Routenparametern . . . . .	186
Hochgeladene Dateien . . . . .	187
Validierung . . . . .	189
validate() auf das Anforderungsobjekt anwenden . . . . .	189
Manuelle Validierung . . . . .	191
Benutzerdefinierte Regeln . . . . .	192
Fehlermeldungen der Validierung anzeigen . . . . .	193
Form Requests . . . . .	194
Erstellen eines Form Requests . . . . .	194
Verwendung eines Form Requests . . . . .	195
Eloquent-Modelle und Massenzuweisung . . . . .	196
{!! und {!! . . . . .	197

Testen . . . . .	198
TL;DR. . . . .	199
<b>8 Artisan und Tinker . . . . .</b>	<b>201</b>
Eine Einführung in Artisan . . . . .	201
Grundlegende Artisan-Befehle . . . . .	202
Optionen . . . . .	202
Befehle nach Gruppen . . . . .	203
Benutzerdefinierte Artisan-Befehle . . . . .	206
Ein Beispielbefehl . . . . .	207
Argumente und Optionen . . . . .	208
Benutzereingaben verwenden . . . . .	210
Eingabeaufforderungen . . . . .	212
Ausgaben . . . . .	213
Schreiben von Closure-basierten Befehlen . . . . .	214
Aufruf von Artisan-Befehlen in normalem Anwendungscode . . . . .	214
Tinker . . . . .	215
Laravels Dump-Server. . . . .	216
Testen . . . . .	217
TL;DR. . . . .	218
<b>9 Authentifizierung und Autorisierung . . . . .</b>	<b>219</b>
User-Modell und -Migration. . . . .	220
Verwendung des globalen auth()-Helfers und der Auth-Fassade . . . . .	223
Die Auth-Controller . . . . .	224
RegisterController . . . . .	224
LoginController . . . . .	225
ResetPasswordController. . . . .	227
ForgotPasswordController. . . . .	227
VerificationController . . . . .	227
ConfirmPasswordController . . . . .	227
Auth::routes() . . . . .	228
Das Auth-Gerüst. . . . .	230
»Remember Me«: Die Erinnerungsfunktion. . . . .	231
Manuelle Authentifizierung von Benutzern . . . . .	233
Manuelles Abmelden eines Benutzers. . . . .	233
Invalidierung von Sitzungen auf anderen Geräten. . . . .	233
Auth-Middleware . . . . .	234
E-Mail-Verifizierung . . . . .	235
Blade-Direktiven zur Authentifizierung . . . . .	236

Guards . . . . .	236
Ändern des Standard-Wächters . . . . .	237
Verwendung anderer Guards ohne Änderung des Standards . . . . .	237
Hinzufügen eines neuen Guards . . . . .	237
Closure Request Guards . . . . .	238
Erstellen eines benutzerdefinierter Providers . . . . .	238
Benutzerdefinierte Provider für nicht-relationale Datenbanken . . . . .	239
Authentifizierungs-Ereignisse . . . . .	239
Autorisierung (ACL) und Rollen . . . . .	240
Berechtigungsregeln definieren . . . . .	241
Die Gate-Fassade (und wie man Gate injiziert) . . . . .	242
Gate für Ressourcen . . . . .	243
Die Authorize-Middleware . . . . .	243
Autorisierung per Controller . . . . .	244
Überprüfen einer Instanz des User-Modells . . . . .	245
Überprüfungen mit Blade . . . . .	246
Abfangen von Prüfungen . . . . .	246
Richtlinien . . . . .	247
Testen . . . . .	249
TL;DR . . . . .	252
<b>10 Request, Response und Middleware . . . . .</b>	<b>253</b>
Der Lebenszyklus des Request-Objekts . . . . .	253
Bootstrapping der Anwendung . . . . .	254
Service Provider . . . . .	255
Das Request-Objekt . . . . .	256
Zugriff auf das Request-Objekt in Laravel . . . . .	257
Informationen aus einem Request erhalten . . . . .	258
Das Response-Objekt . . . . .	262
Response-Objekte in Controllern erzeugen und verwenden . . . . .	262
Spezialisierte Antworttypen . . . . .	263
Laravel und Middleware . . . . .	269
Eine Einführung in Middleware . . . . .	269
Benutzerdefinierte Middleware erstellen . . . . .	270
Middleware binden . . . . .	272
Parameter an die Middleware übergeben . . . . .	275
Vertrauenswürdige Proxys . . . . .	276
Testen . . . . .	277
TL;DR . . . . .	278

<b>11</b>	<b>Der Container</b> .....	<b>279</b>
	Eine kurze Einführung in die Injektion von Abhängigkeiten .....	279
	Abhängigkeitsinjektion und Laravel .....	281
	Der globale Helfer app() .....	282
	Wie ist der Container verdrahtet? .....	283
	Klassen an den Container binden .....	284
	Bindung mittels Closure. ....	284
	Bindung von Singletons, Aliasen und Instanzen .....	285
	Binden einer konkreten Instanz an ein Interface .....	286
	Kontextuelle Bindung .....	287
	Konstruktor-Injektion in Laravel-Framework-Dateien .....	287
	Methoden-Injektion .....	288
	Fassaden und Container .....	289
	Wie Fassaden funktionieren .....	290
	Echtzeit-Fassaden .....	291
	Service Provider .....	292
	Testen .....	292
	TL;DR .....	293
<b>12</b>	<b>Testen</b> .....	<b>295</b>
	Grundlagen des Testens .....	296
	Tests benennen .....	300
	Die Testumgebung .....	301
	Vier spezielle Traits beim Testen .....	302
	RefreshDatabase .....	302
	WithoutMiddleware .....	302
	DatabaseMigrations .....	302
	DatabaseTransactions .....	303
	Einfache Unit-Tests .....	303
	Anwendungstests: So funktionieren sie .....	304
	Die TestCase-Klasse .....	304
	HTTP-Tests .....	305
	Testen von Standardseiten mit \$this->get() und anderen HTTP-Aufrufen .....	305
	Testen von JSON-APIs mit \$this->getJson() und anderen JSON-HTTP-Aufrufen .....	306
	Behauptungen bezüglich \$response .....	307
	Authentifizierung von Antworten .....	309
	Weitere Anpassungen für HTTP-Tests .....	310
	Behandlung von Ausnahmen in Anwendungstests .....	310

Datenbank-Tests . . . . .	311
Verwendung von Modellfabriken . . . . .	312
Seeding in Tests . . . . .	312
Testen anderer Laravel-Features . . . . .	312
Ereignisse faken . . . . .	312
Bus- und Warteschlangen-Fakes . . . . .	314
Mails faken . . . . .	315
Benachrichtigungen faken . . . . .	316
Dateioperationen faken . . . . .	317
Mocking . . . . .	317
Eine kurze Einführung ins Mocken . . . . .	318
Eine kurze Einführung in Mockery . . . . .	318
Andere Fassaden faken . . . . .	321
Artisan-Befehle testen . . . . .	322
Behauptungen bezüglich der Artisan-Befehlssyntax . . . . .	322
Browser-Tests . . . . .	323
Auswahl des Werkzeugs . . . . .	323
Testen mit Dusk . . . . .	324
TL;DR . . . . .	335
<b>13 APIs schreiben . . . . .</b>	<b>337</b>
Die Grundlagen REST-ähnlicher JSON-APIs . . . . .	337
Controller-Organisation und JSON-Antworten . . . . .	339
Header lesen und senden . . . . .	342
Response-Header senden . . . . .	343
Request-Header lesen . . . . .	343
Paginierung . . . . .	343
Sortieren und Filtern . . . . .	345
Sortieren der API-Ergebnisse . . . . .	345
Filtern der API-Ergebnisse . . . . .	347
Ergebnisse transformieren . . . . .	348
Schreiben eines eigenen Transformators . . . . .	348
Verschachtelung und Beziehungen mit benutzerdefinierten Transformatoren . . . . .	349
API-Ressourcen . . . . .	351
Erstellen einer Ressourcen-Klasse . . . . .	352
Ressourcen-Collections . . . . .	353
Verschachtelte Beziehungen . . . . .	354
Paginierung in API-Ressourcen verwenden . . . . .	355
Bedingtes Anwenden von Attributen . . . . .	356
Weitere Anpassungen für API-Ressourcen . . . . .	356

API-Authentifizierung mit Laravel Passport . . . . .	357
Eine kurze Einführung in OAuth 2.0 . . . . .	357
Passport installieren . . . . .	357
Die Passport-API . . . . .	359
Passports Grant-Typen . . . . .	360
Clients und Tokens mit der Passport-API und Vue-Komponenten verwalten . . . . .	368
Passport-Scopes . . . . .	370
Bereitstellen von Passport . . . . .	372
API-Token-Authentifizierung . . . . .	373
Benutzerdefinierte 404-Antworten . . . . .	374
Triggern der Fallback-Route . . . . .	374
Testen . . . . .	374
Passport testen . . . . .	375
TL;DR. . . . .	375
<b>14 Daten speichern und abrufen . . . . .</b>	<b>377</b>
Lokale und Cloud-basierte Datei-Manager. . . . .	377
Konfiguration des Dateizugriffs . . . . .	377
Verwendung der Storage-Fassade . . . . .	378
Zusätzliche Flysystem-Provider hinzufügen. . . . .	380
Grundlagen von Datei-Uploads und -Handhabung . . . . .	380
Einfache Datei-Downloads . . . . .	381
Sessions. . . . .	382
Zugriff auf die Session . . . . .	382
Methoden, die für Session-Instanzen verfügbar sind . . . . .	383
Flash-Sitzungsspeicher . . . . .	385
Cache . . . . .	385
Zugriff auf den Cache . . . . .	386
Methoden, die für Cache-Instanzen verfügbar sind . . . . .	386
Cookies . . . . .	388
Cookies in Laravel . . . . .	388
Auf Cookies zugreifen . . . . .	388
Logging . . . . .	391
Wann und warum man Logs verwenden sollte . . . . .	392
In Logs schreiben . . . . .	392
Log-Kanäle . . . . .	393
Volltextsuche mit Laravel Scout . . . . .	396
Scout installieren . . . . .	396
Ein Modell für die Indexierung kennzeichnen . . . . .	396
Einen Index durchsuchen . . . . .	397
Warteschlangen und Scout . . . . .	397



Operationen ohne Indizierung durchführen. . . . .	397
Bedingt ausgeführte Indizierung. . . . .	398
Manuelles Auslösen der Indizierung im Code . . . . .	398
Manuelles Auslösen der Indizierung über die Kommandozeile . . . . .	398
Testen . . . . .	398
Dateien speichern . . . . .	399
Session . . . . .	400
Cache . . . . .	401
Cookies . . . . .	401
Logging . . . . .	403
Scout . . . . .	403
TL;DR . . . . .	403
<b>15 E-Mail und Benachrichtigungen . . . . .</b>	<b>405</b>
E-Mail . . . . .	405
»Klassische« E-Mail . . . . .	406
E-Mails als »Mailable« . . . . .	406
E-Mail-Vorlagen . . . . .	408
In build() verfügbare Methoden . . . . .	409
Anhänge und Inline-Bilder . . . . .	410
Markdown-Mailables . . . . .	411
Darstellung von Mailables im Browser . . . . .	413
Warteschlangen . . . . .	413
Lokale Entwicklung . . . . .	414
Benachrichtigungen . . . . .	415
Definieren der via()-Methode für die zu benachrichtigenden	
Empfänger. . . . .	418
Senden von Benachrichtigungen. . . . .	419
Benachrichtigungen in Warteschlangen stellen . . . . .	419
Laravels integrierte Benachrichtigungstypen . . . . .	420
Testen . . . . .	424
E-Mail . . . . .	424
Benachrichtigungen . . . . .	424
TL;DR . . . . .	425
<b>16 Queues, Jobs, Events, Broadcasting und der Scheduler . . . . .</b>	<b>427</b>
Warteschlangen . . . . .	427
Warum Warteschlangen? . . . . .	428
Grundlegende Warteschlangen-Konfiguration. . . . .	428
Warteschlangen-Jobs . . . . .	429
Ausführen eines Queue-Workers . . . . .	432
Fehlerbehandlung. . . . .	433

Kontrolle der Warteschlange . . . . .	435
Warteschlange für andere Funktionen. . . . .	436
Laravel Horizon . . . . .	436
Ereignisse . . . . .	437
Ein Ereignis auslösen . . . . .	438
Nach einem Ereignis »lauschen« . . . . .	439
Broadcasting von Ereignissen über WebSockets und Laravel Echo . . . .	442
Konfiguration und Einrichtung . . . . .	443
Übertragung eines Ereignisses . . . . .	444
Empfang der Nachricht . . . . .	446
Fortgeschrittene Broadcasting-Werkzeuge . . . . .	449
Laravel Echo (die JavaScript-Seite) . . . . .	452
Scheduler . . . . .	457
Verfügbare Aufgabentypen . . . . .	457
Verfügbare Zeitangaben. . . . .	458
Definieren von Zeitzonen für geplante Befehle . . . . .	460
Blockierung und Überlappung. . . . .	460
Output von Aufgaben handhaben . . . . .	460
Aufgaben-Hooks . . . . .	461
Testen . . . . .	462
TL;DR. . . . .	463
<b>17 Helfer und Collections . . . . .</b>	<b>465</b>
Helfer . . . . .	465
Arrays . . . . .	465
Zeichenketten. . . . .	467
Anwendungspfade . . . . .	469
URLs . . . . .	470
Verschiedenes. . . . .	472
Collections . . . . .	475
Die Grundlagen . . . . .	475
Ein paar Methoden . . . . .	477
TL;DR. . . . .	482
<b>18 Das Laravel-Ökosystem . . . . .</b>	<b>483</b>
Tools, die in diesem Buch behandelt werden . . . . .	483
Valet . . . . .	483
Homestead. . . . .	483
Der Laravel-Installer. . . . .	484
Mix. . . . .	484
Dusk. . . . .	484
Passport . . . . .	484

Horizon.....	484
Echo .....	485
Tools, die in diesem Buch nicht behandelt werden.....	485
Forge.....	485
Envoyer.....	485
Cashier .....	486
Socialite.....	487
Nova .....	487
Spark.....	487
Lumen.....	487
Envoy .....	488
Telescope .....	488
Vapor .....	488
Weitere Ressourcen .....	489
<b>Glossar .....</b>	<b>491</b>
<b>Index .....</b>	<b>499</b>



---

# Vorwort

Der Weg, der mich zu Laravel geführt hat, ist ganz typisch: Ich hatte jahrelang in PHP programmiert, war aber bereits dabei, mich davon zu verabschieden und mich den Möglichkeiten von Rails und anderen modernen Web-Frameworks zu widmen. Insbesondere Rails besaß eine lebendige Community und bot eine perfekte Kombination aus meinungsstarken Vorgaben und Flexibilität sowie das Potenzial von Ruby Gems, vorgefertigte Packages mit Programmen und Bibliotheken zu nutzen.

Aber etwas hielt mich noch davon ab, PHP endgültig hinter mir zu lassen, und ich war froh darüber, als ich auf Laravel stieß. Laravel bot alles, was mich an Rails anzog, aber es war mehr als bloß ein Rails-Klon. Es war ein innovatives Framework mit unglaublich guter Dokumentation, einer einladenden Community und deutlichen Einflüssen verschiedener Sprachen und Frameworks.

Seitdem habe ich in Blogs, Podcasts und Vorträgen auf Konferenzen geteilt, was ich auf meiner Reise mit Laravel gelernt habe; ich habe Dutzende von Laravel-Apps für Arbeits- und Nebenprojekte geschrieben; und ich habe Tausende von Laravel-Entwicklern online und persönlich getroffen. Es gibt viele Tools in meinem Entwicklungs-Werkzeugkasten, aber ich bin ehrlich gesagt am glücklichsten, wenn ich mich vor eine Kommandozeile setzen und `laravel new projektName` eingeben kann.

## Worum es in diesem Buch geht

Dies ist nicht das erste Buch über Laravel, und es wird nicht das letzte sein. Es soll kein Buch sein, das jede Zeile Code oder jedes mögliche Implementierungsmuster abdeckt. Und es soll kein Buch sein, das automatisch veraltet ist, sobald eine neue Version von Laravel veröffentlicht wird. Stattdessen geht es in erster Linie darum, Entwicklern einen wirklichen Überblick und konkrete Beispiele zu geben, damit sie wissen, was sie benötigen, um in allen Laravel-Releases mit jedem einzelnen Feature und Subsystem arbeiten zu können. Anstatt einfach die Dokumentation nachzuerzählen, möchte ich Ihnen helfen, die grundlegenden Konzepte zu verstehen, auf denen Laravel aufbaut.

Laravel ist ein leistungsfähiges und flexibles PHP-Framework. Es gibt eine florierende Community und ein umfassendes Ökosystem an Werkzeugen, was dazu beiträgt, dass Laravels Attraktivität und Reichweite ständig wachsen. Dieses Buch richtet sich an Entwickler, die bereits wissen, wie man Websites und Anwendungen erstellt, und die jetzt lernen wollen, wie man genau das erfolgreich auch mit Laravel macht.

Die Laravel-Dokumentation ist umfassend und ausgezeichnet. Wenn Sie feststellen, dass ich ein bestimmtes Thema Ihrem Geschmack zufolge nicht gründlich genug abdecke, empfehle ich Ihnen, die Onlinedokumentation (<https://laravel.com/docs>) zu besuchen und tiefer in dieses spezielle Thema einzutauchen.

In diesem Buch erwartet Sie eine angenehme Balance zwischen einer Einführung auf hohem Niveau und konkreter Anwendung, und am Ende werden Sie sich hoffentlich dabei wohlfühlen, wenn Sie eine komplette Anwendung in Laravel von Grund auf neu schreiben. Und wenn ich meinen Job gut gemacht habe, werden Sie auch heiß darauf sein.

## Für wen dieses Buch gedacht ist

Dieses Buch setzt Kenntnisse grundlegender objektorientierter Programmierpraktiken, PHP (oder zumindest der allgemeinen Syntax der Sprachen der C-Familie), grundlegender Konzepte des Entwurfsmusters »Model-View-Controller« (MVC, auf Deutsch »Modell-Präsentation-Steuerung«) und der Arbeit mit Template Engines voraus. Wenn Sie zuvor noch nie selbst eine Website entworfen haben, wird es vielleicht etwas zu anspruchsvoll sein. Aber solange Sie über Programmiererfahrung verfügen, müssen Sie vor der Lektüre dieses Buchs kein Vorwissen zu Laravel haben – wir decken alles ab, was Sie wissen müssen, angefangen beim einfachsten »Hallo, Welt!«.

Laravel kann mit jedem Betriebssystem eingesetzt werden, aber es wird einige Kommandozeilen-Befehle im Buch geben, die am einfachsten unter Linux/macOS ausgeführt werden können. Windows-Benutzer werden es mit diesen Befehlen und mit moderner PHP-Entwicklung möglicherweise etwas schwerer haben, aber wenn Sie Homestead (eine virtuelle Linux-Maschine) zum Laufen bringen, können Sie alle Befehle von dort aus ausführen.

## Wie dieses Buch aufgebaut ist

Dieses Buch ist in gewisser Weise chronologisch gegliedert: Wenn Sie Ihre erste Webanwendung mit Laravel erstellen, decken die ersten Kapitel die grundlegenden Komponenten ab, die Sie zum Einstieg benötigen, während die späteren Kapitel weniger grundlegende bzw. etwas ausgefallene Funktionen behandeln.

Jeder Abschnitt des Buchs kann für sich allein gelesen werden, aber für alle, die dieses PHP-Framework zum ersten Mal einsetzen, habe ich versucht, die Kapitel so zu strukturieren, dass es sinnvoll ist, alles in der vorgegebenen Reihenfolge zu lesen.

Wenn es passt, schließt ein Kapitel mit zwei speziellen Abschnitten: »Testen« und »TL;DR.« Falls Sie es nicht kennen: »TL;DR« steht für »too long; didn't read«, also »zu lang, nicht gelesen«. Diese letzten Abschnitte zeigen, wie Sie Tests für die im Kapitel behandelten Funktionen schreiben können, und fassen sehr kompakt zusammen, um was es im betreffenden Kapitel geht.

Dieses Buch ist für Laravel 6 geschrieben, deckt aber in der Regel Funktionen und Syntaxänderungen von Laravel 5.1 an ab – auch um die permanente Weiterentwicklung des Frameworks lebendig darzustellen.

## Über die zweite Ausgabe

Die erste Ausgabe von *Laravel: Up & Running* erschien im November 2016 und umfasste die Laravel-Versionen 5.1 bis 5.3. Diese zweite Ausgabe deckt die Versionen 5.4 bis 6.6 sowie Laravel Dusk und Horizon ab, zudem ist ein 18. Kapitel über Community-Ressourcen und andere Laravel-Pakete hinzugekommen, die nicht zum Kern des Frameworks gehören und in den ersten 17 Kapiteln nicht behandelt wurden.

## Vorbemerkung zur deutschen Ausgabe

Laravel gibt es nur in einer englischsprachigen Version, und das gilt auch für nahezu alle zusätzlichen Module, Programmpakete und Dienstleistungsangebote des gesamten Laravel-Ökosystems.

Wir haben deshalb bei der Übersetzung dieses Buchs versucht, eine angenehme Balance zwischen englischen Originalbegriffen und deutschen Fachausdrücken zu erreichen. Zumal es für viele der in Programmierung und Anwendungsentwicklung benutzten Begrifflichkeiten gar keine deutsche Entsprechung gibt. Deshalb benutzen wir mal die englischen, mal die deutschen Begriffe, mit einer Tendenz zum Englischen. Da Sie sich als angehender Laravel-Entwickler überwiegend im englischsprachigen Umfeld bewegen werden, wäre es eher kontraproduktiv, würden wir zu viele Begriffe oder Teile von Programmlistings eindeutschen.

Dazu kommt, dass Laravel von sich aus einige Funktionen mitbringt, die selbstständig Pluralformen (zum Beispiel von Modell-Namen) erstellen. Dabei wird standardmäßig ein »s« benutzt und angehängt. Obwohl man dieses Verhalten individuell übersteuern kann, liest sich der Code insgesamt geschmeidiger, und man hat weniger Arbeit, wenn man Laravel seinen »Willen« lässt.

Der Schöpfer von Laravel, Taylor Otwell, gibt sich extrem viel Mühe, den Code so lesbar und so nah an natürlicher Sprache zu halten, wie es nur geht. Und es liest

sich natürlich besser, wenn eine Datenbank-Migration `create_customers_table` heißt und nicht `create_kundens_table` (ohne Übersteuerung des Standardverhaltens) oder `create_kunden_table` (mit angepasster Pluralform).

## Konventionen, die in diesem Buch verwendet werden

Die folgenden typografischen Konventionen werden in diesem Buch verwendet:

### *Kursiv*

Zeigt neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateierweiterungen an.

### Nichtproportional

Wird für Programm-Listings verwendet, aber auch innerhalb von Absätzen, um dort auf Programmelemente wie Variablen- oder Funktionsnamen, Datenbanken, Datentypen, Umgebungsvariablen, Anweisungen und Schlüsselwörter zu verweisen.

### **Nichtproportional fett**

Zeigt Befehle oder anderen Text an, der vom Benutzer wortgetreu eingegeben werden muss.

### *Nichtproportional kursiv*

Zeigt Programmcode an, der durch Benutzereingaben oder durch kontextabhängige Werte ersetzt werden soll.

### *{Kursiv in Klammern}*

Zeigt Dateinamen oder Dateipfade an, die durch Benutzereingaben oder durch kontextabhängige Werte ersetzt werden sollen.



Dieses Element weist auf einen Tipp oder Vorschlag hin.



Dieses Element kennzeichnet einen allgemeinen Hinweis.



Dieses Element weist auf eine Warnung hin.



## O'Reilly Online-Lernen



Seit fast 40 Jahren bietet *O'Reilly Media* Technologie- und Business-Training, Wissen und Einsichten, um Unternehmen zum Erfolg zu verhelfen.

Unser einzigartiges Netzwerk von Experten und Innovatoren teilt sein Wissen und seine Expertise in Büchern, Artikeln, Konferenzen und auf unserer Online-Lernplattform. Die Online-Lernplattform von O'Reilly bietet Ihnen On-Demand-Zugriff auf Live-Schulungen, detaillierte Lernpfade, interaktive Codierumgebungen und eine umfangreiche Sammlung von Texten und Videos von O'Reilly und über 200 anderen Verlagen. Für weitere Informationen besuchen Sie bitte <https://www.oreilly.com>.

## Englischsprachige Website zu diesem Buch

Es gibt eine englischsprachige Website zu diesem Buch, auf der wir Errata, Beispiele und alle weiteren Informationen aufführen. Sie können diese Seite unter <https://bit.ly/laravel-up-and-running-2e> aufrufen.

## Danksagung für die erste Ausgabe

Dieses Buch wäre ohne die verständnisvolle Unterstützung meiner wunderbaren Frau Tereva und meines Sohnes Malachi (»Papa schreibt, Buddy!«) nicht möglich gewesen. Und obwohl sie es selbst nicht wirklich bewusst miterlebt hat, war meine Tochter Mia fast über die gesamte Zeit der Entstehung des Buchs dabei, sodass dieses Buch der ganzen Familie gewidmet ist. Es gab viele lange Abendstunden und Arbeitsbesuche bei Starbucks am Wochenende, sodass ich oft nicht bei meiner Familie war, und ich könnte nicht dankbarer sein für ihre Unterstützung und vor allem für ihr Da-Sein, das mein Leben einfach bereichert.

Darüber hinaus hat mich die gesamte Tighthen-Familie während des Schreibens des Buchs unterstützt und ermutigt, und mehrere meiner Kollegen haben Codebeispiele bearbeitet (Keith Damiani, »Editor Extraordinaire«) und mir bei schwierigen Projekten geholfen (Adam Wathan, König der Collection-Pipeline). Dan Sheetz, mein Partner bei Tighthen, war so freundlich, mir viele Stunden nachzusehen, die ich mit der Arbeit an diesem Buch verbracht habe, und war dabei immer eine Stütze und Ermutigung; und Dave Hicking, unser Betriebsleiter, half mir, meine Zeiteinteilung und Verantwortlichkeiten rund um meine Schreibzeiten zu organisieren.

Taylor Otwell verdient Dank und Anerkennung als Schöpfer von Laravel – und damit auch für die Schaffung vieler Jobs und dafür, dass so viele Entwickler ihr Leben noch mehr lieben. Er verdient Anerkennung dafür, wie sehr er sich auf die Zufriedenheit der Entwickler konzentriert und wie hart er gearbeitet hat, um mit

Empathie für uns Entwickler eine positive und ermutigende Community aufzubauen. Aber ich möchte ihm auch dafür danken, dass er ein liebenswürdiger, unterstützender und anregender Freund ist. Taylor, du bist ein Held.

Vielen Dank an Jeffrey Way, der einer der besten Lehrer im Internet ist. Er hat mich mit Laravel bekannt gemacht und bringt Laravel heute noch jeden Tag vielen Menschen näher. Er ist auch, wenig überraschend, ein fantastischer Mensch, den ich gerne Freund nenne.

Vielen Dank an Jess D'Amico, Shawn McCool, Ian Landsman und Taylor, die mich schon früh als Konferenzsprecher geschätzt haben und mir eine Plattform bieten, von der aus ich wirken kann. Vielen Dank an Dayle Rees, der in den frühen Tagen von Laravel so vielen Menschen dabei geholfen hat, es zu erlernen.

Vielen Dank an alle Menschen, die ihre Zeit und Mühe dem Schreiben von Blog-Posts über Laravel gewidmet haben, besonders in den Kindertagen von Laravel: Eric Barnes, Chris Fido, Matt Machuga, Jason Lewis, Ryan Tablada, Dries Vints, Maks Surguy und viele mehr.

Und vielen Dank an die gesamte Community von Freunden auf Twitter, IRC und Slack, die über die Jahre mit mir kommuniziert haben. Ich wünschte, ich könnte jeden Namen nennen, aber ich würde einige vergessen und mich dann dafür schämen. Ihr seid alle brilliant, und ich fühle mich geehrt, regelmäßig mit euch in Kontakt zu stehen.

Vielen Dank an meine O'Reilly-Lektorin, Ally MacDonald, und alle meine technischen Redakteure: Keith Damiani, Michael Dyrinda, Adam Fairholm und Myles Hyson.

Und natürlich danke an den Rest meiner Familie und meine Freunde, die mich direkt oder indirekt unterstützt haben – meine Eltern und Geschwister, die Gainesville-Community, Geschäftspartner und Autoren, andere Konferenzredner und die einzigartige DCB. Ich muss aufhören, weiter zu schreiben, bevor ich noch meinen Starbucks-Baristas danke.

## Danksagung für die zweite Ausgabe

Die zweite Ausgabe ist der ersten sehr ähnlich, sodass alle vorherigen Danksagungen noch gültig sind. Aber ich habe diesmal Hilfe von ein paar weiteren Personen bekommen. Meine technischen Korrekturleser waren Tate Peñaranda, Andy Swick, Mohamed Said und Samantha Geitz, und meine neue O'Reilly-Lektorin war Alicia Young, die mich im letzten Jahr über viele Veränderungen in meinem Leben und innerhalb der Laravel-Community hinweg bei der Arbeit gehalten hat. Matt Hacker vom Atlas-Team beantwortete alle meine dummen AsciiDoc-Formatierungsfragen, auch zur überraschend schwierigen Formatierung der `__()`-Methode.

Und ohne die Hilfe meines Assistenten Wilbur Powery hätte ich es nicht geschafft, diese zweite Ausgabe zu schreiben. Wilbur war bereit, die Changelogs,

Pull-Requests und Ankündigungen der letzten Jahre zu durchsuchen und jedes Feature der aktuellen Struktur des Buchs zuzuordnen, und testete tatsächlich jedes einzelne Codebeispiel, damit ich meine begrenzte Zeit und Energie auf das Schreiben der neuen und aktualisierten Abschnitte konzentrieren konnte.

Außerdem hat meine Tochter Mia jetzt den Weg aus dem Bauch ihrer Mutter gefunden. Also lassen Sie mich einfach noch ihre Freude und Energie und Liebe und Niedlichkeit und ihren Abenteuergeist der Liste meiner Inspirationsquellen hinzufügen.



# Warum Laravel?

In den frühen Tagen des dynamischen World Wide Web sah das Schreiben einer Anwendung ganz anders aus als heute. Die Entwickler waren damals nicht nur dafür verantwortlich, den Code für die jeweils spezielle Anwendungslogik zu schreiben, sondern auch für all jene Komponenten, die immer wieder für Websites benötigt werden: für die Authentifizierung von Benutzern, die Validierung von Eingaben, Datenbankzugriffe, die Erstellung von Vorlagen und vieles mehr.

Heutzutage gibt es Dutzende von Frameworks für die Anwendungsentwicklung und Tausende von Komponenten und Bibliotheken, die allen Programmierern leicht zugänglich sind. Es ist gängige Rede unter Entwicklern, dass, kaum habe man ein Framework gelernt, es bereits drei neuere (und angeblich bessere) Frameworks gebe, die es gerne ersetzen möchten.

»Weil er halt existiert«, mag eine sinnvolle Rechtfertigung dafür sein, einen Berg zu besteigen. Aber es gibt bessere Gründe als »Weil es halt existiert«, um sich für ein bestimmtes Framework zu entscheiden – oder überhaupt eines zu verwenden. Stellen wir uns also die berechtigte Frage: Wozu sind Frameworks eigentlich gut? Genauer gesagt: Wozu ist Laravel gut?

## Warum ein Framework verwenden?

Es ist leicht nachzuvollziehen, warum es hilfreich ist, die einzelnen Komponenten und Pakete zu verwenden, die es für PHP-Entwickler gibt. Bei Packages ist jemand anderes für die Entwicklung und Wartung eines isolierten Stücks Programmcode verantwortlich, mit dem eine bestimmte Aufgabe gelöst wird, und theoretisch sollte diese Person ein tieferes Verständnis für diese einzelne Komponente besitzen, als Sie sich selbst in angemessener Zeit aneignen können.

Frameworks wie Laravel – und Symfony, Lumen und Slim – versammeln eine Reihe Komponenten von Drittanbietern und bündeln diese mit Framework-eigenem »Klebstoff« wie Konfigurationsdateien, Service Providern, vordefinierten Verzeichnisstrukturen und Anwendungs-Boostraps. Ganz allgemein besteht der Vorteil, ein Framework zu verwenden, darin, dass jemand für Sie nicht nur bereits über einzelne

Komponenten entschieden hat, sondern auch darüber, wie diese Komponenten zusammenarbeiten sollen.

## »Ich baue es einfach selbst«

Nehmen wir an, Sie beginnen mit einer neuen Webapplikation, ohne die Vorteile eines Frameworks zu nutzen. Womit fangen Sie an? Ziemlich sicher müssen HTTP-Requests geroutet werden, sodass Sie sich alle verfügbaren HTTP-Request- und Response-Bibliotheken anschauen und danach eine auswählen müssen. Dann brauchen Sie einen Router. Und wahrscheinlich benötigen Sie auch irgendeine Art Konfigurationsdatei für die Routen. Welche Syntax soll dabei verwendet werden? An welcher Stelle soll die Konfigurationsdatei abgelegt werden? Und was ist mit Controllern? Wo sollen die liegen und wie sollen sie geladen werden? Wahrscheinlich benötigen Sie einen Dependency Injection Container, um die Controller und ihre Abhängigkeiten aufzulösen. Aber welchen?

Wenn Sie sich die Zeit nehmen, all diese Fragen zu beantworten und Ihre Anwendung erfolgreich zu erstellen, was bedeutet das dann für einen möglichen späteren Entwickler? Und was machen Sie, wenn Sie beispielsweise vier dieser Anwendungen mit solch einem Framework Marke Eigenbau programmiert haben – oder gar ein ganzes Dutzend – und Sie sich jeweils daran erinnern müssen, wo sich die Controller befinden oder wie genau die Routing-Syntax lautet?

## Konsistenz und Flexibilität

Frameworks lösen dieses Problem, indem sie eine sorgfältig durchdachte Antwort auf die Frage »Welche Komponente sollen wir verwenden?« geben und zudem sicherstellen, dass die jeweils gewählten Komponenten auch gut zusammenarbeiten. Darüber hinaus bieten Frameworks feste Konventionen, die die Menge an Code reduzieren, die ein Entwickler, der neu zu einem Projekt stößt, verstehen muss: Sobald Sie begriffen haben, wie Routing in *einem* Laravel-Projekt funktioniert, wissen Sie sofort, wie es in *allen* Laravel-Projekten funktioniert.

Wenn jemand vorschreibt, dass sein eigenes Framework für jedes neue Projekt benutzt werden muss, dann geht es letztlich darum, zu *kontrollieren*, was in die Grundlage seiner Applikation einfließt und was nicht. Das bedeutet, dass Ihnen die besten Frameworks nicht nur ein solides Fundament bieten, sondern auch die Freiheit geben, praktisch alles Ihren eigenen Wünschen entsprechend anzupassen. Und genau das ist es, was Laravel so besonders macht und was ich Ihnen im weiteren Verlauf dieses Buchs zeigen möchte.

## Eine kurze Geschichte der Web- und PHP-Frameworks

Will man die Frage »Warum Laravel?« beantworten, muss man seine Geschichte kennen – und seine Vorläufer. Bevor Laravel populär wurde, gab es bereits eine