Niclas Kahlmeier

Laravel 7

Ein Webmasters Press Lernbuch

Version 1.0.4 vom 06.07.2020

Autorisiertes Curriculum für das Webmasters Europe Ausbildungs- und Zertifizierungsprogramm

Inhaltsverzeichnis

Basisarchitektur von Laravel

3

Du lernst in dieser Lektion
➤ die Architektur und den Arbeitsablauf von Laravel kennen. ➤ wie die MVC-Architektur in Laravel umgesetzt ist. ➤ welche Namenskonventionen in Laravel üblich sind.

Bevor es mit dem Programmieren richtig losgeht, gebe ich dir einen kurzen Überblick über die Architektur von Laravel. Du wirst jetzt vielleicht mit den Augen rollen … ;-) Ich denke aber, es wird dir dabei helfen, Struktur und Konzept von Laravel zu verstehen.

3.1 MVC in Laravel

Die Model-View-Controller-Architektur, kurz MVC, teilt die Applikation in die drei namensgebenden Teile:

Das ***Model*** repräsentiert alle Daten und die Datenverwaltung der Anwendung. Es antwortet außerdem auf Informationsanfragen und reagiert auf Anforderungen, Daten zu ändern. Es registriert alle Datenänderungen und benachrichtigt, sobald sich Informationen ändern.

Die ***View*** repräsentiert die Benutzeroberfläche. Sie rendert auch die Daten aus dem Model in eine Form, die für die Benutzeroberfläche geeignet ist. Außerdem erhält die View Nutzer-Input und verarbeitet diesen oder leitet ihn weiter.

Der ***Controller*** ist die Schaltzentrale. Er erhält die Nutzeranfragen und stellt beim Model Anfragen nach Daten. Die entsprechenden Modelobjekte und Views werden aufgerufen und die notwendigen Daten übergeben.

Merksatz: Das Model steuert die Datenbank. Die View kümmert sich um das HTML. Der Controller ist der »Mittelsmann« zwischen Model und View.

3.2 Warum verwendet man die MVC-Architektur?

Die erste Frage, die du dir vielleicht gestellt hast, ist: »Warum macht man das überhaupt?« Wenn man eine kleine Codebase hat, ist es noch nicht notwendig. Bei größeren Projekten ist es jedoch enorm wichtig, den Überblick zu behalten. Die MVC-Architektur verleiht dem Code mehr Struktur und erleichtert dadurch die Arbeit.

Die MVC-Architektur hat weitere Vorteile:

- Die Rollenverteilung im Team wird einfacher. Frontend-Entwickler kümmern sich um Views und Backend-Entwickler um Controller und Models.
- MVC zwingt Entwickler, ihre Dateien in logische Verzeichnisse zu unterteilen. Dadurch lassen sich Dateien und auch ihre Funktionen leichter erkennen und finden.
- Die Models und Views können separat voneinander bearbeitet werden, da diese nicht direkt voneinander abhängig sind. Dieses Prinzip nennt man ***Separation of Concerns*** (auf Deutsch in etwa »Trennung der Zuständigkeiten«).

3.3 Life Cycle

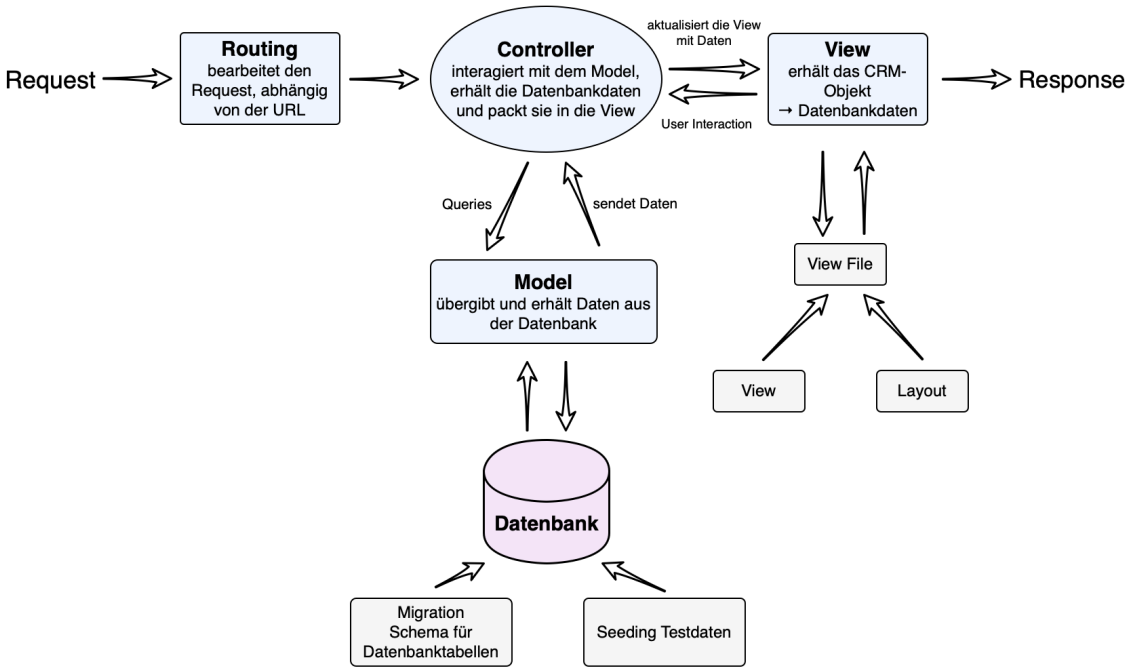

Abb. 1 *Workflow in Laravel*

In Abb. 1 siehst du eine Grafik zum Workflow: Der Aufruf einer Webseite, die eine eindeutige Adresse im Web hat (Uniform Resource Locator, URL) geschieht mit Hilfe des HTTP-Protokolls (HTTP Request). Die Anfrage wird dann im Routing verarbeitet. Abhängig von der URL wird der entsprechende Controller aufgerufen. Der Controller kommuniziert mit der Datenbank, um z. B. Daten daraus abzurufen oder Daten in die Datenbank zu schreiben. Das Model leitet die entsprechenden Daten an den Controller weiter. Der Controller ruft die entsprechende View auf und gibt die Daten weiter. Die View kann sich auch aus mehreren Dateien zusammensetzen. Nachdem die View die notwendigen Daten erhalten hat, wird sie gerendert. Nach dem Rendern wird eine Antwort (HTTP Response) an den Webbrowser gesendet. Dies ist dann die für den Nutzer sichtbare Webseite.

Die gerade schon besprochene MVC-Architektur habe ich in der Grafik (Abb. 2) rot markiert.

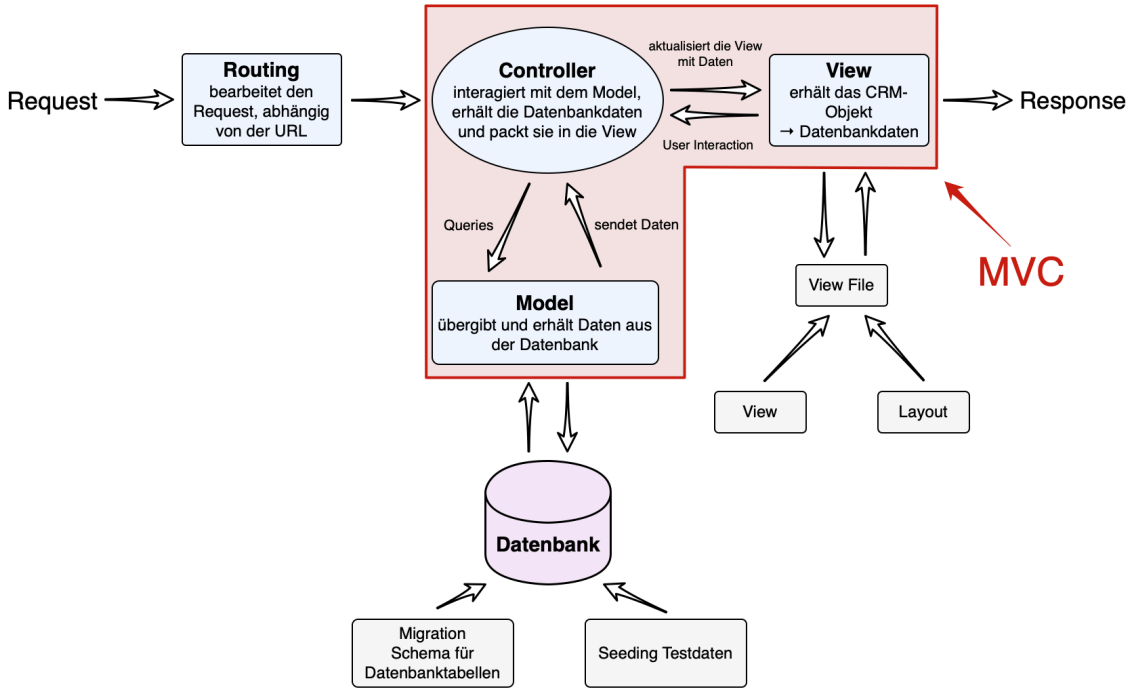

Abb. 2 *MVC-Architektur innerhalb des Workflow*

3.4 Namenskonventionen (Naming Conventions)

Laravel wird von Tausenden Entwicklern für die verschiedensten Websites und Applikationen genutzt. Dabei orientieren sie sich an etablierten Regeln zur Namensgebung von Funktionen, Klassen und Variablen. Dies hilft anderen Entwickler dabei, sich in dem von anderen geschriebenen Code zurechtfinden. Ich gehe im Folgenden die verschiedenen Bezeichnungen durch. Vielleicht sagen dir die genannten Begriffe jetzt noch nichts, aber wir werden diese im weiteren Verlauf alle kennenlernen.

Bezeichnung von Controllern

Namen für Controller werden im Singular und großgeschrieben, enthalten keine Leerzeichen und enden auf *Controller*.

Beispiel: *UserController*, *AuthController*, *RoleController*

Diese Schreibweise nennt man ***Pascal Case***.

Controller-Methoden

Alle Methoden versehen wir in Laravel mit der sogenannten ***Camel-Case-Notation***. Das erste Zeichen wird kleingeschrieben.

Beispiel: `get()`, `getAll()` , `deleteSelected()`

Außerdem sollten im Controller für normale Create-Update-Delete-Operationen, kurz CRUD, folgende Methoden genutzt werden:

- `index()`
- `create()`
- `store()`
- `show()`
- `edit()`
- `update()`

- `destroy()`

Benennung von Datenbankelementen

Datenbanktabellen werden im Plural und kleingeschrieben. Worttrennungen kennzeichnen wir mit Unterstrichen – dies nennt man ***Snake Case***. Die Bezeichnung sollte im Plural sein.

Beispiel: *users*, *user_pictures*, *awarded_certificates*

Tabellenspalten werden kleingeschrieben und nutzen ebenfalls Snake Case. Die Tabellenbezeichnung wird nicht erwähnt.

Beispiel: *id*, *created_at*, *updated_at*, *name*, *email*

Der **Primärschlüssel** sollte *id* heißen.

Fremdschlüssel enden auf *_id*. Beispiel: *user_id*, *post_id*

Variablen

Variablen werden im Singular mit ***Camel Case*** bezeichnet.

Beispiel: *$activeUser*

Ausnahme: Sobald die Variable mehrere Items, ein Array oder eine Collection enthält, wird die Variable im Plural bezeichnet.

Models

Models sollten großgeschrieben im Singular bezeichnet werden und die Pascal-Case-Schreibweise nutzen.

Beispiel: *User*, *Role*, *UserImage*

Relationships

Wie bei allen Methoden wird die Camel-Case-Schreibweise verwendet.

Beispiel: *email()*, *postComment()*

Views

Views werden in der Snake-Case-Schreibweise und kleingeschrieben, außerdem schließen sie immer ab mit der Dateiendung *.blade.php*.

Beispiel: *register.blade.php*, *profile.blade.php*

Die verschiedenen Case Styles findest du hier: https://medium.com/better-programming/string-case-styles-camel-pascal-snake-and-kebab-case-981407998841

3.5 Teste dein Wissen!

1. Die View …

   Bitte ankreuzen:

   ☐ repräsentiert die Benutzeroberfläche.

- ☐ repräsentiert die Schnittstelle zwischen Datenverwaltung und Benutzeroberfläche.
- ☐ repräsentiert das Request Handling.
- ☐ repräsentiert die Datenverwaltung.

2. Warum verwendet man die MVC-Architektur?

   Bitte ankreuzen:

   - ☐ Die Rollenverteilung im Team wird einfacher.
   - ☐ Die MVC-Architektur verleiht dem Code mehr Struktur.
   - ☐ Die Models und Views können separat voneinander bearbeitet werden, da diese nicht direkt voneinander abhängig sind.
   - ☐ Das Prinzip der Separation of Concerns (auf Deutsch »Trennung der Zuständigkeiten«) wird umgesetzt.

3. Die Schreibweise `UserController` bezeichnet man als …

   Bitte ankreuzen:

   - ☐ Pascal Case
   - ☐ Snake Case
   - ☐ Camel Case
   - ☐ Kebab Case

4. Die Schreibweise `deleteSelectedFile()` bezeichnet man als …

   Bitte ankreuzen:

   - ☐ Camel Case
   - ☐ Snake Case
   - ☐ Pascal Case
   - ☐ Kebab Case

```
$request-> session()->get('name', "Lorem Ipsum") ;
```

4. **Prüfe, ob es in der Session einen Key *'name'* mit zugeordnetem Wert gibt.**

```
session( )->has('name') ;
```

Index

M

N

O

P

Q

R

S

T

U

V

W

X