

Forms

Er zijn 7 restful controller acties (conventie/richtlijn):

`index` toont een lijst van iets (artikelen, gebruikers, etc)

`show` toont een specifiek item (artikel, gebruiker etc)

`create` toont een view om item aan te maken.

`store` slaat item op.

`edit` Toont een view om bestaand item aan te passen.

`update` slaat wijzigingen op.

`destroy` verwijdert item.

Met de artisan optie `-r` kunnen deze automatisch aangemaakt worden bij het maken van een controller. Met de `-m Modelnaam` optie erbij wordt een bijpassend database model aangemaakt.

Browsers ondersteunen eigenlijk allen GET en POST requests. Om toch een PUT of ander soort request te simuleren vanuit een form, stel het formulier in op POST en voeg een `@method('PUT')` toe in het formulier.

Bij het doen van een POST, bijvoorbeeld vanuit een form, kan een 419 error ontstaan als resultaat van bescherming tegen XSS exploits. Door het toevoegen van een `@csrf` statement in de view na het `<form>` element wordt deze fout voorkomen.

Algemeen

Alle resources (afb, JS, etc) in de `public` map.

Routing

Routes worden gedefinieerd in `routes/web.php`.

Een route wordt gedefinieerd met

Route: `get('url', $data);`

of

Route: `get('url', function():view {});`

of

Route: `get('/url/{variable}', function($variable):object {});`

of (om functie functionname in class controllerclass aan te roepen)

Route: `get('/url/{variable}', 'controllerclass@functionname');`

Data die naar de view moet worden gestuurd moet gedaan worden in de closure functie in de route
 return `view('viewnaam', ['key' => 'value']);`
 migrate:rollback

POST of GET argumenten kunnen worden opgevraagd met de `request($argumentnaam)`
 Database legen en migraties opnieuw uitvoeren
`migrate:refresh`

Foutafhandeling (404 pagina) wordt gestart met `abort(int $statusCode, string $message)`

Naast `Route::get('url', $data)` zijn er ook `Route::post`, `Route::put`, `Route::patch`, `Route::delete`, `Route::options`, `Route::any` en `Route::fallback`.
 Een migratie, factory en controller maken voor een project als volgt:
`Project -a`

Om toekomstbestendig te zijn kun je routes benoemen, als volgt:
 Route: `get('/articles/{article}', 'ArticlesController@show');`

In blade kan dan route worden weergegeven met
`@route('articles.show', $article)`

Artisan

Een nieuwe controller aanmaken:

`make:controller ControllerName`

Een eloquent model aanmaken:

`make:model Modelnaam`

Een migratie aanmaken:

`make:migration genereer_dat_abaan`
`make:migration genereer_dat_abaan`

Een migratie toepassen:

`migrate`

Een migratie rollback:

`migrate:rollback`

Database legen en migraties opnieuw uitvoeren

`migrate:refresh`

Een migratie, factory en controller maken voor een project als volgt:
`Project -a`

Blade language

Escape variabelen met

`@{{ $variabele }}`

Variabelen expliciet NIET escapen met

`{!! $variabele !!}`

Elementen in objecten weergeven met

`{{ $object->element }}`

Een template kan bouwen op een andere door

`@extends('bestanden/naam')` commando

Er kan een sectie uit een template-bestand worden

ingevoegd op de huidige positie met het

`@yield('sectienaam')`

commando.

Een sectie wordt gedefinieerd met

`@section('sectienaam')`

en geëindigd met

`@endsection`



By babipanghang

Not published yet.

Last updated 23rd January, 2020.

Page 1 of 3.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Blade language (cont)

Veel PHP is gewoon beschikbaar, zoals

```
Request::path()
```

Meerdere items zijn te plaatsen als

```
@foreach ($articles as $article)
<li>Hier invoegen met {{ article->
@endforeach
```

De asset map kan worden aangegeven met

```
p/ bes tan dsn naam.css
```

Foutmeldingen (zoals bij formvalidatiefouten) kunnen worden

weergegeven met

```
{{ $errors->first('validation') }};
```

Alleen weergeven als een variabele/attribuut bestaat.

```
@if ($errors->has('title'))<html><endhtml
```

Inline text plaatsen met conditie:

```
{{ $errors->has('title') ? 'tekst' : '' }}
```

Bij foutmelding iets weergeven:

```
@error ('validation') <b> uw tekst hier </b> @enderror
```

Een veldwaarde van een vorige submit invullen (zoals bij het veld

van formvalidatie):

```
{{ old('validation') }}
```

Migrations (Database)

Een migratie is een layout voor de

database, of een aanpassing daaraan.

Bestanden met migraties staan in database/migrations

Voor development kunnen het beste

bestaande migraties gewijzigd worden. Voor

wijzigingen aan live servers maak een

nieuwe migratie.

Controllers

Controllers staan in Http/Controllers

Request data kan worden opgevraagd met de request()

dnaam) functie.

De request() functie (zonder argumenten) retourneert een

object met validate(array \$options) functie, zoals dit:

```
request()->validate([
```

```
'name' => ['required', 'min:3', 'max:255']
```

```
,
```

```
'body' => 'required',
```

```
'body' => 'required'
```

```
]);
```

De validate() functie retourneert bovendien een array met valide

elementen die gerefereerd kunnen worden bij verdere verwerking

van de formuliergegevens.

In controllers wordt vaak gegevens uit een DB gehaald met

functies als

```
$article = Article::findOrFail($id);
```

er vanuitgaande dat er een eloquent model Article bestaat. In

plaats daarvan kunnen we ook Laravel om het eloquent model

op te halen.

```
public function show(Article $article) {
```

```
//code hier
```

```
}
```

De variabelenaam moet hierbij overeenkomen met een wildcard

uit de route.

Model (Database)

Configuratie in config/database.php

en in .env file in de root

Model (Database) (cont)

Eloquent modellen zijn classes die een database

kunnen uitgevoerd worden via deze class. Deze

functies om die tabel te manipuleren. Het resultaat

model is gerepresenteerd in diezelfde class.

Indien een eloquent model is aangemaakt kun

model importeren zoals

```
use App\Modelnaam
```

```
en dan
```

```
Modelnaam::where('kolomnaam',
```

Queries zien eruit als:

```
\DB::table('tabelnaam')->wh
```

```
e->first();
```

```
>firstOrFail() haalt 1 waarde op of
```

```
findOrFail().
```

Eloquent modellen bevinden zich in app/

de all() functie haalt alle records uit een tab

functie (eerste resultaat), take(int \$count)

opnemen en paginatie(int \$currentPage)

opnemen).

```
De latest('datumveld-record') {
```

```
record op. find($id) haalt een record op me
```



By babipanghang

Not published yet.

Last updated 23rd January, 2020.

Page 2 of 3.

Sponsored by [CrosswordCheats.com](https://crosswordcheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Model (Database) (cont)

Wildcards in routes worden standaard gematched op het id veld in een database. Om de wildcard naar een ander veld te laten matchen, definiëer de functie

```
public function getRouteKeyName() {
    return 'keynaam';
}
```

in het eloquent model.

Een nieuw record aanmaken kan als volgt:

```
$article = new EloquentModelNaam();
$article->title = request('title');
$article->excerpt = request('excerpt');
```

Maar ook als:

```
EloquentModelNaam::create([
    'title' => request('title'),
    'excerpt' => request('excerpt')
]);
```

Bij deze laatste methode moet er opgepast worden: de array zou onbedoeld meer velden kunnen bevatten dan bedoeld. Daarom moet er bij Laravel in het eloquent model worden aangegeven welke velden met een publiek form in te vullen zijn door een class property \$fillable te definiëren:

```
protected $fillable = ['title', 'excerpt'];
```

Alle velden expliciet **niet** bewaken kan met

```
protected $guarded = [];
```

Relaties tussen eloquent models kunnen worden gedefinieerd in een functie in een eloquent model. Bijvoorbeeld

```
return $this->hasMany(Article::class);
```

geeft een relatie aan met meerdere articles.

Views

Views worden opgeslagen in `resources /views`

Gegenererde resources worden opgeslagen in de `resources` map. Overige in de `public` map.

In de `/webpack.mix.js` wordt aangegeven wat voor bestanden gecompileerd moeten worden, zoals SASS en enkele JS bestanden. Dit wordt gedefinieerd met een regel zoals

```
mix.sass('resources/sass/beans.css', 'public/development');
```

Het bestand `/package.json` definieert dependencies. Deze dependencies moeten met de hand geïnstalleerd worden met npm (op te halen via nodejs.org), met het commando `npm install`.

Resources compileren wordt in gang gezet met `npm dev`.

`npm run watch` houdt resources in de gaten en compileert ze automatisch.



By babipanghang

cheatography.com/babipanghang/

Not published yet.

Last updated 23rd January, 2020.

Page 3 of 3.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>