Opdracht1 week 3

Je gaat een Laravel app maken met een lijst met titels van boeken.

Normaal gesproken start je een app met “laravel new appnaam”. Deze keer ga je een bestaande app aanpassen. Hiervoor maak je een clone van een Laravel app op Github. Control + klik op de volgende link:

[summacollege/laravel1\_week3\_opdracht1: Opracht1 week3 module laravel1 SD Summa ICT (github.com)](https://github.com/summacollege/laravel1_week3_opdracht1)

Hier staat een repository die je gaat clonen (kopieren) naar je eigen omgeving. Zorg voor een plek op je eigen systeem waar je de app wilt bewaren en open daar de Git Bash.

Voer het volgende commando in op je systeem:

* git clone <https://github.com/summacollege/laravel1_week3_opdracht1.git>

De github repo wordt naar je systeem gekopieerd en er wordt een lokale repo aangemaakt. Daarnaast wordt een link gemaakt naar de github repo.

Optioneel:

Als je de lokale repo wilt koppelen aan je eigen github moet je eerst de remote verwijderen.

* Voer het volgende commando in op je systeem(in de map laravel1\_week3\_opdracht1):  
    
  Git remote remove origin
* Maak een github repo (zelf te kiezen naam) in je eigen github account. Kopieer de link (zoiets als: <https://github.com/jouwaccount/eigenrepo.git>)
* Voer het volgende commando in op je systeem(in de map laravel1\_week3\_opdracht1):  
    
  Git remote add origin <https://github.com/jouwaccount/eigenrepo.git>

En dan heb je een verbinding naar je eigen repo en kun je je code pushen.

Zorg dat je in de map staat (cd) die je zojuist hebt gedownload en geef het volgende commando (in de git bash):

* composer update

Met bovenstaand commando wordt alle benodigde software (pakketten) gedownload en geïnstalleerd in de laravel app . Voordat je het commando gaf had je geen Vendor map en daarna wel.

Standaard wordt het ".env" bestand niet meegenomen wanneer je een push uitvoert van een laravel app naar github. Gelukkig hebben we een ".env.example". Hierin staan enkele zaken al ingesteld die we kunnen gebruiken.

* Dus maak een kopie van ".env.example" met de naam ".env"

Als je het bestand ".env" opent mis je bijvoorbeeld een unieke key. Bij het aanmaken van een nieuwe Laravel app wordt die automatisch gegenereerd. Achteraf gebruiken we het commando " php artisan key:generate".

Voer het volgende commando in:

* php artisan key:generate

In het “.env” zie je bij database de naam " laravel1\_week3\_opdracht1" staan. Verander deze naam in:

* " laravel1\_opdrachten"

Deze database moet natuurlijk wel aanwezig zijn. Controleer ook de inlog gegevens voor je database server die in “.env” staan.

* Start (als dat nog niet is gebeurd) jouw MySQL server en maak een database met de naam "laravel1\_opdrachten"

In de applicatie is een extra commando gemaakt (geen standaard laravel) om een tabel aan te maken en te vullen in de database.

Voer het volgende commando in:

* php artisan import:titels

Controleer in bijvoorbeeld phpmyadmin of er inderdaad een tabel met de naam titles is aangemaakt.

Nu gaan we de data ophalen uit de tabel en in een variabele bewaren:

* open “web.php” in routes
* Voeg de volgende code toe:

*Route::get('/titels', function () {*

*$data = DB::select('select \* from titles');*

*dd($data);*

*});*

* Start de applicatie met "php artisan serve" en ga naar <http://localhost:8000/titels>

"DB" in de bovenstaande code staat natuurlijk voor database en de "::" betekent dat je een soort van static functions kunt gebruiken. De "DB::" aanduiding noemen ze een facade. De code is iets makkelijker en korter te schrijven dan dat je allemaal objecten moet instantieren.

Er is een tweede manier om de data op te halen. Hiervoor gebruiken we de zogenaamde querybuilder.

* open “web.php” in routes
* Vervang de code voor de route "/titels" met:

*Route::get('/titels', function () {*

*$data = DB::table('titles')->get();*

*dd($data);*

*});*

Tijd om de gegevens daadwerkelijk te tonen in een view.

* Maak een map in de "titels" in de map "resources/views"
* Maak een bestand in de map " resources/views /titels" met de naam “index.blade.php”
* Plaats de volgende code in dat bestand:

*@foreach($data as $title)*

*<div>{{ $title->title }}</div>*

*@endforeach*

De @foreach in bovenstaande code is een zogenaamde blade functie. Deze functie komt overeen met de foreach functie van php. Laravel zorgt er voor dat de blade functie bij het runnen wordt omgezet naar php. De schrijfwijze van blade is bedoelt om het werken met php makkelijker en overzichtelijker te maken. De "{{ *$title->title* }}" is in feite het title veld van elk record in de verzameling data. Door elke regel in een div te zetten krijg je een lijst met titels. Natuurlijk moeten we in de route nog aangeven dat we naar de view willen en dat de data moet meegenomen worden. Dat doen we door de tweede parameter van de return functie te vullen met een array.

* Open web.php en plaats(vervang) de volgende code:  
  *Route::get('/titels', function () {*  
   *$data = DB::table('titles')->get();*  
   *return view('titels.index',['data'=>$data]);*  
  *});*
* Controleer de werking in <http://localhost:8000/titels>

Selecteer nu enkel de records met de type "business".

* Pas de code aan:  
  *$data = DB::select('select \* from titles where type=?', [‘business’]);*
* Of:  
  $data = DB::table('titles')  
   ->where('type', '=', 'business')  
   ->get();

Maak de view iets uitgebreider:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>titels</title>  
 <style>  
 .container {  
 display: grid;  
 grid-template-columns: 1fr 1fr  
 }  
 </style>  
</head>  
<body>  
 <div class="container">  
 @foreach($data as $title)  
 <diV>{{ $title->title }}</div><div>{{ $title->type }}</div>  
 @endforeach   
 </div>  
</body>  
</html>

En we passen de route aan zodat alleen de kolommen "title" en"type" worden geselecteerd en de data wordt gefilterd op "type=business" en wordt gesorteerd in een aflopende volgorde:

*Route::get('/titels', function () {*   
 *$data = DB::table('titles')*   
 *->select('title', 'type')*   
 *->where('type', '=', 'business')*   
 *->orderByDesc('title')*   
 *->get();*   
 *// $data = DB::select('select title, type from titles where type=? order by title desc', ['business']);*   
 *return view('titels.index',['data'=>$data]);*   
*});*

De output:

