De student heeft:

* Composer geïnstalleerd (zie week1)
* Laravel installer geïnstalleerd (zie week1)

Opdracht

* Maak een nieuw laravel project met de naam week2. Zorg voor een lokale Git Repo.

We willen een route waarbij we een parameter kunnen meegeven zodat iets getoond word.

Opdracht

* Maak een route met onderstaande info:

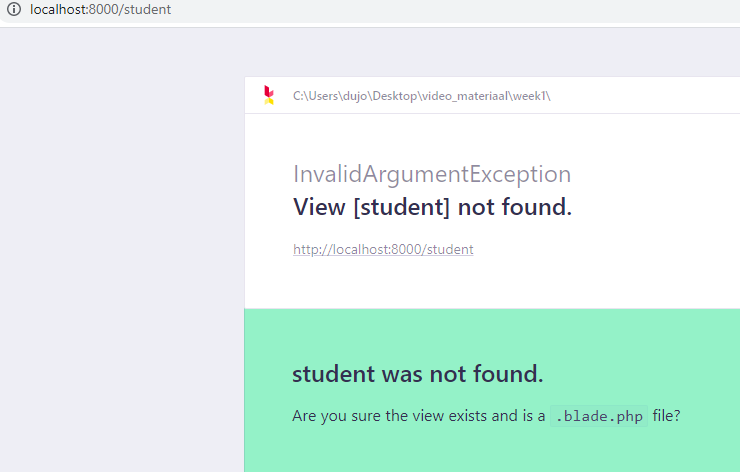
*Route::get('/student', function () {*

*return view('student', ['naam' => 'Sara']);*

*});*

Opdracht

* Test deze route in de browser en je krijgt deze melding.



Je ziet in het groene gedeelte dat Laravel zich afvraagt of er wel een view "student" wel aanwezig is. En dat is natuurlijk niet zo.

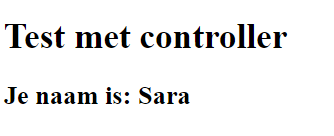
Opdracht

* Maak de view student met een standaard HTML layout en plaats in de body de volgende code:

*<h1>Test met controller</h1>*

*<h2>Je naam is: {{ $naam }}</h2>*

Het resultaat is:



Stel je wilt dit verhaal dynamisch maken waarbij iemand in de url een naam kan meegeven die wordt weergegeven in het resultaat.

Opdracht

* Maak een nieuwe route met onderstaande info:

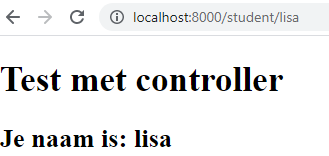
*Route::get('/student/{naamUitUrl}', function ($naamUitUrl) {*

*return view('student', ['naam' => $naamUitUrl]);*

*});*

Opdracht

* Test de route in een browser. In de url type: "localhost:8000/student/lisa"



Stel we willen de naam in hoofdletters schrijven. Dan kan dit natuurlijk in de view. Maar het kan ook in het bestand web.php. Echter begint er nu een probleem te ontstaan. Het aantal routes zal waarschijnlijk flink gaan oplopen en als we in alle routes functies gaan gebruiken en daarin weer functies, zal het bestand onoverzichtelijk worden.

Om dit te voorkomen kunnen we gebruik maken van controllers. In controllers regelen we alle logica met betrekking tot gegevens. In de controller kun je daarna beslissen waar de data naar toe gaat. Bijvoorbeeld naar een view. Dus we willen in de browser ergens naar toe. Laravel ontvangt een request dat wordt afgehandeld in de web.php. In een route bepalen we welke controller en functie gebruikt gaat worden voor verdere verwerking. In de controller wordt bepaald welke view de output gaat tonen.

Opdracht

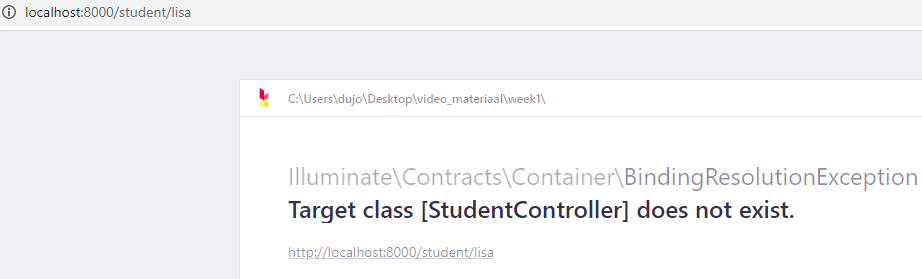
* Verander de laatste route in web.php:

Route::get('/student/{naamUitUrl}',[StudentController::class, 'show']);

De tweede parameter van de "get" functie bestaat nu uit een array met de naam van de controller die je wilt benaderen en de naam van de methode (functie) in die controller die je wilt gebruiken. De :: voor het woord class heet een "Scope resolution operator". Lees meer hierover op het web. In feite kunnen we de methode "index" gebruiken zonder een instantie van de class te maken.

Opdracht

* Voer de laatste test in de browser opnieuw uit.



Het resultaat geeft duidelijk aan dat er geen controller(class) is met de naam StudentController. Die moet je dus nog maken.

Opdracht

* Geef in een console de opdracht: *php artisan make:controller StudentController*
* Open het bestand in "app/Http/Controllers/StudentController.php"
* Voeg de volgende code toe aan de class:

public function show($naamUitUrl)

{

$naamUitUrl = Str::upper($naamUitUrl);

return view('student', ['naam' => $naamUitUrl]);

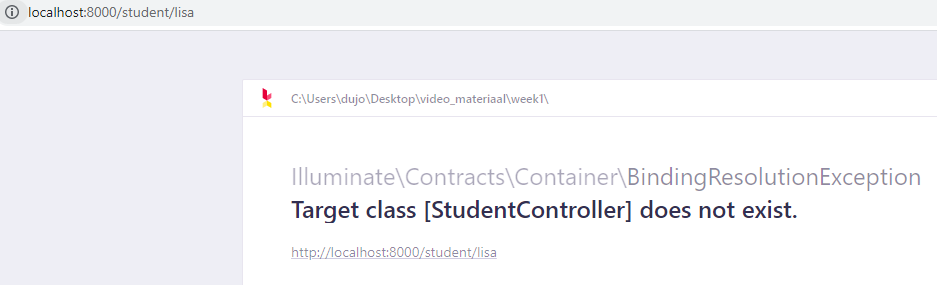
}

* Boven de class voeg je nog deze code toe:

*use Illuminate\Support\Str;*

Opdracht

* Test opnieuw in browser en je krijgt nog steeds een foutmelding



Dit komt doordat in het bestand "web.php" wij de controller "StudentController" gebruiken, maar we hebben geen verwijzing naar de controller.

Opdracht

* Maak in bestand "web.php" bovenin een verwijzing naar de "StudentController"

*use App\Http\Controllers\StudentController;*

Opdracht

* Test opnieuw.
* Werk je Lokale Git Repo bij met een commit.