

Controllers

Inleiding

Je hebt een nieuwe View en Route gemaakt.

De view laat statische data zien (= niet uit de database). Als we data uit de database willen tonen (=dynamische data) dan hebben we net als in Yii de **controller** nodig.

Dit is een lang verhaal waarin MVC en het nut van een controller wordt uitgelegd. We hebben dit ook gehad bij Yii, maar het is belangrijk dat je precies weet wat een controller en een view doet. Lees dit stuk dus goed door.

Wat is een Controller?

Een Controller zorgt ervoor dat de **View** requests kan ontvangen en/of versturen naar de **database**. Deze structuur noemen we ook wel een ModelViewController (**MVC**).

Wat is een MVC?

De MVC is een basis software architectuur waardoor je makkelijk en overzichtelijk projecten kunt bouwen. Het MVC model maakt elke code los om een specifieke taak uit te voeren. Daardoor is bekend wat elk onderdeel doet en bij bugs is het makkelijk te zien waar het probleem zit.

De MVC architectuur wordt ook gebruikt in andere frameworks.

Wat doet een MVC?

De acties bij een MVC kun je vergelijken met een restaurant.

Scenario: Het restaurant (MVC):

Het restaurant is de **MVC** architectuur. Zij zorgen er allemaal voor dat het restaurant (in webdevelopment, de site) soepeltjes verloopt en dat alle bestellingen worden uitgevoerd.

Image-1656834908474.jpg

De klant (View)

De klant is de **View**. Zij zijn technisch gezien de front-end en zijn verantwoordelijk wat jij op jouw browser te zien krijgt.

Image-1656831284319.jpg

Dit zijn de klanten van het restaurant. Zij willen graag **pizza** en **pasta** bestellen. Zij voeren een **GET** request uit:

Request - GET: items: {'pizza', 'pasta'}

Onthouden! Alle **Views** staan in het mapje `/resources/views/` met als filenaam `filenaam.blade.php`.

De ober (Controller)

De ober is de **Controller**. Zij zijn verantwoordelijk voor het communiceren tussen de **View** (klant) en **Model** (keuken). Je kan ze zien als de tussenpersoon.

Image-1656831459034.jpg

Dit is de ober. De ober heeft de bestelling (**GET** request) ontvangen en zorgt ervoor dat dit gecommuniceerd wordt naar de **keuken**.

Hieronder zie je hoe een Controller te werk gaat in Laravel:

```
//De class Controller is de ober
class OberController extends Controller {

    //Een ober heeft als functie om gerechten op te halen van de Keuken
    public function getDishes(Request $request){

        //De GET request (pizza & pasta) is wat de Controller (ober) gekregen heeft van de View (klant)
        //(zie dit hetzelfde als een $_GET)
        $bestellingen = $request->query('items');

        //De Controller (ober) vraagt aan de Model (keuken) om de gerechten te maken en het uit te voeren
        $gerechten = Keuken::maakGerechten($bestellingen)->get();

        //De Controller (ober) geeft het gerecht aan de klant (View)
        return view('klant', compact('gerechten'));
    }
}
```

De keuken (Model)

De keuken is de **Model**. Zij zijn verantwoordelijk dat ze de taken van de **controller** uitvoeren en dit te communiceren met de **database**.



Dit is de keuken. Ze hebben een taak van de ober ontvangen om de gerechten te maken en te geven aan de ober..

Hoe maak je een Controller?

Nu je weet wat een controller doet, gaan we er eentje maken. Dat gaat heel simpel!

Een controller maken kan je makkelijk met een commando via de terminal maken:

- `php artisan make:controller JouwvoornaamController`
Let op dat Jouwvoornaam met ene hoofdletter begint!

Voer het commando uit. Jouw Controller is nu aangemaakt! Je kan jouw gemaakte Controller terugzien in de map `app/Http/Controllers` met de naam `JouwvoornaamController.php`

Als je de file opent, zie je deze code:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class JouwvoornaamController extends Controller
{
    // Hier maak je jouw functions.
}
```

Dit is jouw Controller. Alleen is het leeg. In de class maak je alle logica en CRUD's aan.

Jouw View tonen vanuit de Controller

We gaan nu code maken die de view die we eerder hebben gemaakt aanroept vanuit de controller.

Om dat te kunnen doen moet je eerst een `public function functieNaam(){} in de class` maken. Daarin plaats je

`return view('viewNaam');` in de codeblok van `public function functieNaam(){} in de class`. Hieronder een voorbeeld:

```
public function functieNaam(){
    return view('viewNaam');
}
```

Gefeliciteerd! Je hebt nu een Controller gemaakt die een view laat zien! Alleen laat hij dat niet zien, omdat hij niet weet naar welk URL hij moet gaan...

Daarom moet je hem nog via de **Routes** naar de Controller verwijzen.

Jouw Controller verwijzen in de Routes

Om jouw Controller te verwijzen in de Routes moet je `routes/web.php` aanpassen.

Je maakt een route naar de controller op de volgende manier:

```
// plaats dit bovenaan in de routing file (web.php)
use App\Http\Controllers\UrlController;

// Plaats vervolgens je routing informatie
Route::get('/url', [UrlController::class, 'functionName']);
```

/url: dit is de URL die je koppelt aan de controller.

UrlController: de naam van de Controller

functionName: de naam van de van de functie die Controller zit

Een voorbeeld van de Routes bij het restaurant scenario (op basis van de code bij "*De ober (Controller)*"):

```
Route::get('/dishes', '[OberController::class, 'getDishes');
```

Dit zorgt ervoor dat de URL */dishes* wordt gekoppeld aan de functie *getDishes* die staat in het bestand *App\Http\Controllers\OberController*

Opdracht - Controller maken en linken aan de Route

1. Maak een controller aan via de terminal en geef de ControllerNaam `homeController`
2. Ga naar de controller en maak in de `class` een `public function` aan die `index()` heet. Toon jouw eerder gemaakte View in de Controller.
3. Ga naar jouw `web.php` en zoek naar de `Route::get` die de URL `/` heeft. Vervang de verwijzing i.p.v. naar de view nu naar de `homecontroller`. Ook moet hij verwijzen naar de `index` functie.

De output is hetzelfde als in de vorige opdracht, maar dit keer ga je via de **controller** naar de view.

[image-1666297596665.png](#)

Inleveren

1. Een screenshot van de terminal die aantoont dat je een Controller hebt gemaakt
2. Een screenshot van de code van je Controller
3. Een screenshot van de code van je Routes

Revision #7

Created 20 October 2022 20:28:45 by Max

Updated 28 October 2022 19:01:47 by Max