# Refactoring to MVC

*Aan het eind van deze les heb je een control file en heb je de complete routing voor het object links opgezet. De functies Read and Delete lopen aan het eind van deze les via de controller en we hebben het raamwerk opgezet op de ander CRUD functies ook op te zetten via de controller.*

We gaan onze code een beetje opnieuw bouwen, dit heet ook wel refacturing. Het doel is dat we onze code opschonen en het MVC model wat duidelijker vorm geven in onze code. Hierdoor wordt code overzichtelijker en sluiten we beter aan bij de Laravel standaard.

## Routing and controllers

Tot nu te hebben we onze routing opgezet en in de routing hebben we tevens de controls opgenomen. Bijvoobeeld bij het toevoegen van een nieuwe link hebben we alle code in de routing file gestopt. Dit werkt maar is op den duur, als onze code meer wordt niet meer overzichtelijk. In onze routing houden we alleen de routing bij en de control-code gaat in een aparte file. Hiermee gaan we een duidelijk onderscheid maken tussen de routing en de controls.

We kunnen al onze CRUD routes voor het object Link in één keer aanpassen met één route:

```
Route::resource('links','LinksController');
```

Plaats deze route en plaats de andere routes voorlopig in commentaar.

Deze ene regel zorgt voor routes naar alle CRUD-functies:

| Method | URI | Action (function in controller) | functie |
|---|---|---|---|
| GET | /links | index | standaard pagina voor indes |
| GET | /links/create | create | create form |
| POST | /links | store | create form post |
| GET | /links/{id} | show (één item) | laat één item zien |
| GET | /links/{id}/edit | edit | edit form |
| PUT/PATCH | /links/{id} | update | edit form post |
| DELETE | /links/{id} | destroy | delete |

zie ook https://laravel.com/docs/6.x/controllers of gebruik het commando:

```
 php artisan route:list
```

In de directory app/http/Controllers staan al onze controlers en we maken een nieuwe file en noemen die LinkController.php

Tip: er is ook een artisan command om een standaard controller file te maken:

```
php artisan make:controller LinksController
```

De routing zal naar deze file verwijzen. In deze php file maken we de volgende 'standaard' code:

```
<?php
namespace App\Http\Controllers;
use App\Link;
use Illuminate\Http\Request;

class LinkController extends Controller
{
    protected function index (){
 □□//
    }

    protected function create(){
       //
    }

    public function edit($id){
        //
    }

    public function show($id){
    □//
    }

    public function update(Request $request, $id) {
□□//
   }

    protected function store(Request $request){
         //
    }
```

```
    protected function destroy($del_id){
□□//
    }
}
```

Nu hebben we alle standaard methods voor de routing, we moeten nu alleen de code nog per functie invullen.

Plaats in elke control een echo die laat zien dat in welke control je zit. Dus in de index control plaats je `echo "index";`, in de create control plaats je `echo "create";` , etc.

> Check: Test alle GET functies (zie tabel hierboven) en check of de juiste controls worden aangeroepen.

# Read

Als alles goed werkt dan kunnen we de code bij de index method in de liinksController invullen. Deze code hadden we al eerder gemaakt, maar die stond in de routing (web.php). Verplaats deze code naar de index method in de linksController. Onze Read van CRUD hebben we op deze manier via de linksController laten lopen

# Delete

Voor de delete hebben we nu nog een aparte control in de web.php staan.

Als je in de tabel kijkt dan zie je dat de method destroy in de linksController wordt aangeroepen als je de HTTP header DELETE meesturen. Dat kan niet via een link of via een button. Dat kan alleen als je een form gebruikt. We moeten dus kiezen: bij een button of een link moeten we de route zoals we die al in web.php hebben gemaakt laten staan. Als we de delete via de route::resource willen laten lopen dan moeten we een form gebruiken.

```
<form method="POST" action="/links/{{ $link->id }}">
  {{ csrf_field() }}
  {{ method_field('DELETE') }}
  <input type="submit" class="btn btn-danger" value="Delete">
</form>
```

In dit form wordt de method DELETE in de header van het HTTP request meegestuurd. Hierdoor 'weet' de resource control dat die de method destroy moet aanroepen.

Als dit werkt gaan we de andere controls later invullen; we gaan eerst de templates aanpassen zodat alles er iets beter uit ziet.

--

---

Revision #13
Created 2019-10-27 11:46:02 UTC by Admin
Updated 2020-07-01 21:56:04 UTC by Max