Laravel

Deze les

- Wat en waarom Laravel?
- Installatie
- Mappenstructuur
- Console
- Router en Controllers
- Request en Response
- Blade

Wat en waarom Laravel?

- PHP framework
- Leesbare code(OOP)
- Goede documentatie
- Open source
- Laatste trends
- Database migrations
- Active record

```php
$school = new School;

$school->name = 'Noorderpoort';
$school->location = 'Verzetstrijderslaan 2';

$school->save();
```

Installatie

PHP in je path zetten

Composer installeren
Voor Windows is er een installer(*https://getcomposer.org/Composer-Setup.exe*).

Voor linux/mac moet je composer.phar handmatig in je /usr/local/bin zetten:

```
curl -sS https://getcomposer.org/installer | php
```

Installeer Laravel:

```
composer create-project laravel/laravel --prefer-dist
```

Mappenstructuur

Console

De laravel console maakt het makkelijk om nieuwe bestanden aan te maken, database **migrations** en database **seeds** uitvoeren. Een aantal veel gebruikte commands zijn:

`php artisan make:seeder`	Create a new seeder class
`php artisan make:controller`	Create a new resource controller class
`php artisan make:migration`	Create a new migration file
`php artisan db:seed`	Seed the database with records
`php artisan migrate`	Run the database migrations
`php artisan list`	Lists all commands

Router

http://laravel.com/docs/5.1/routing

In de Router defineer je naar welke URL's je applicate moet luisteren, vervolgens koppel je hier je code. Dit kan door middel van een callback of via een Controller. Daarnaast kun je middlewares toevoegen aan routes, Middlewares worden uitgevoerd voor dat de code die gekoppeld is aan je route word uitgevoerd.

```
Route::get('/', function () {
    return view('index');
});


Route::post('product', 'ProductController@save');
Route::put('product/{id}', 'ProductController@save');
Route::delete('product/{id}', 'ProductController@save');


Route::group(['prefix' => 'api/v1'], function()
{
    Route::resource('post', 'PostController'); //REST API
});


Route::get('admin/profile', ['middleware' => 'auth', function () {
    echo 'authenticated';
}]);
```

Requests en Responses

In laravel kun je gebruik maken van de Request en Response objecten. Het Request object maakt het ophalen van data uit de `$_GET`, `$_POST` en `$_SESSION` variablen gemakkelijk. Het Response object maakt het gemakkelijk om data(JSON, tekst of bestanden) terug te sturen naar de client. Daarnaast kun je headers aan je response toevoegen.

```php
public function store(Request $request)
{
    $name = $request->input('name');
    $uri = $request->path();
    $url = $request->url();
    $method = $request->method();
    $input = $request->all();
    return $request->has('name');
}

return (new Response('done', 200))->header('Content-Type', 'text/plain');
return response('done', 200)->header('Content-Type', 'text/plain');

return response()->json(['workshop' => 'Laravel', 'date' => '4-11-2015']);

return response()->download(storage_path('app/uploads/8493jf98hadsifuhih892.pdf' ));
```

Blade

Blade is de template engine van Laravel(zoals Smarty en Twig). Blade bestanden worden opgeslagen in `resources/views` en moeten worden opgeslagen met de extensie `.blade.php`

```
return view('index', ['name' => 'Foo']);
return view('index')->with('name', 'Foo');

<p>{{$name}}</p> // Escaped html
<p><{!!$name}}</p> // RAW text


@foreach ($items as $item)
 @if($item->id)
   <p>ID: {{ $item->id }}</p>
  @endif
@endforeach
```

Blade

Blade templates kun je ook hergebruiken. Dit kun je doen door middel van extenden of includen.

```
// resources/views/base.blade.php
<html>
 <head>
   <title>@yield('title')</title>
 </head>
 <body>
   <div class="container">
     @yield('content')
   </div>
 </body>
</html>
-------------------------------------------------------------------------------------------------
-----
// resources/views/index.blade.php

@extends('index')

@section('title', 'Application')

@section('content')
 <p>Main content</p>
@endsection
```

Blade

Blade templates kun je ook hergebruiken. Dit kun je doen door middel van extenden of includen.

```
// CSRF field
<form action="{{url('post')}}" method="POST">
 {{ csrf_field() }}
 {{ method_field('DELETE') }}

 <button>Delete Post</button>
</form>
```

Opdracht

Maak de volgende applicatie
Een pagina met een formulier, het formulier moet het 2 invoer vakken en 1 selectbox bevatten. Vervolgens moet door middel van een POST request een som berekend worden. In de selectbox moet je de keuze hebben uit delen, keer, minus en plus. Vervolgens laat je het antwoord van de som zien in een aparte blade view.

Maak gebruik van het volgende
- Routes (GET en POST)
- Controllers
- Request object
- Blade(met extends)

Vervolg opdracht
Maak een “REST API” die precies hetzelfde doet maar dan JSON returned in plaats van het antwoord returned in een blade view.

Volgende les

- Sessions
- Migrations
- Query builder
- Eloquent
- Authentication