Database Migration

In deze les leren we

- *hoe via Laravel een websever kunnen opstarten en;*
- *hoe we met Laravel een database aanmaken en vullen met testdata.*

Aan het eind van deze les hebben we ee nieuwe database die is gevuld met test data.

Laravel web server

We zagen in de vorige les dat we naar ons nieuw Laravel project gaan via de link 127.0.0.1/links/public. Uiteindelijk als we productie gaan draaien willen we een eigen domain name zoals www.links.com. Voor nu kunnen we een 'virtual' host op een andere poort aanmaken met het commando:

```
php artisan serve
```

Nadat je php artisan server heb gestart zie je het volgende:

```
Laravel development server started: http://127.0.0.1:8000
```

Dit betekent dat je Laravel project draait op `http://127.0.0.1:8000`

Als we naar 127.0.0.1:8000 gaan dan gaan we dus naar een nieuwe document root, namelijk c:/xampp/httpdocs/links/public

Deze directory is de (Laravel) project directory en als we via de Laravel webserver nasar 127.0.0.1:8000 gaan dan komen we rechtstreeks in de directory c:/xampp/httpdocs/links/public

We kunnen XAMPP naast de Laravel server laten draaien. Dat kan ook handig zijn om phpmyadmin te kunnen gebruiken. XAMPP en Laravel serve zitten elkaar niet in de weg, omdat ze beide andere poorten gebruiken.

Artisan

Artisan (ambachtsman) is de command line tool die bij Laravel hoort. Je kunt er allemaal handige dingen mee doen.

>

> Let op het uitvoeren van een artisan command doen we vanuit de project directory.

Database Migratie

Met Artisan maken we een tabel, eerst voeren we het volgende in de command prompt uit.

```
// Check of je in de juiste directory staat
php artisan make:migration create_links_table --create=links
```

Dit commando maakt een file. In de output van dit commando staat de naam van de file die is gemaakt. Deze staat in `database/migrations/` en heet `{{datetime}}_create_links_table.php`

Open deze file.

> tip: ga naar de direcory waar deze file in staan in je command prompt en type *notepad* gevolgd door de naam van de file.

In het script staan twee functies up() and down(). up() wordt gebruikt om een nieuwe database te maken en down kan worden gebuikt om deze database weer te verwijderen.

Migratie scripts kunnen worden gebruikt om een bestaande database aan te passen (migrate) of een nieuwe database aan te maken.

Verander de function up() zodat die er als volgt uit ziet (niet alles kopiëren; alleen de functie/method aanpassen).

```
public function up()
{
    Schema::create('links', function (Blueprint $table) {
        $table->increments('id');
        $table->string('title');
        $table->string('url')->unique();
        $table->text('description');
        $table->timestamps();
    });
}
```

Om het database script te draaien moet Laravel toegang hebben tot de database. Edit hiervoor de `.env` file in de *files* directory.

In de .env file staan configuratie parameters zoals de database naam en userid en password van de database.

In de .env.example file staat een complete voorbeeld config waar alle mogelijke configuratie-parameters in staan. Voor nu beperken we ons even tot de database connectie.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=links
DB_USERNAME=root
DB_PASSWORD=
```

Maak nu met phpMyAdmin de database `links`.

Voer de migratie uit:

```
php artisan migrate
```

De output moet zijn: Migration table created successfully.

Bij het uitvoeren van een migratie van een product dat je via githib hebt 'ge-cloned' ,moet je mogelijk nog alle vendor packages downloaden en de dependencies controleren, dat kan met:

```
composer update --no-scripts
```

Composer zorgt er dan voor dat alle benodigde software wordt geïnstalleerd. Dit hoeft dus niet bij een 'verse' installatie van Laravel.

Als je een foutmelding krijgt die iets over key lengte zegt, kijk dan naar het commentaar aan het eidne van deze les.

In alle andere gevallen moet je de foutmelding bestuderen en met behulp van Google uitvogelen wat er aan de hand is. Het meest aannemelijke is dat er op één of andere manier geen verbinding met de database kan worden gemaakt.

Kijk nu via phpmyadmin welke of de tabel is aangemaakt.

Nu volgen nog wat tips over de migration, als het niet goed is gegaan bijvoorbeeld, hoe kun je het dan terug draaien. Indien je voor een tweede keer deze les doorloopt kan (hoeft niet) je doorgaan naar 'Tabel Vullen' hieronder.

Draai nu de migratie terug met het commando:

```
php artisan migrate:rollback
```

Kijk nu weer via phpmyadmin wat er is gebeurd.

Voer tenslotte de migratie nog een keer uit (probeer zelf te bedenken met welk commando).

Met het commando:

```
 php artisan help migrate
```

Zie je alle migratie opties. Kijk zelf in de opties hoe je een commando kan geven waarbij er niets wordt uitgevoerd maar waarbij er wel wordt getoond wat er *zou* worden uitgevoerd. Dit kun je de *net-als-of* optie noemen.

met het commando:

```
php artisan migrate:refresh
```

Combineer je een rollback en een migrate.

Opdracht

Bestudeer de help van de migrate en kijk met deze *net-alsof-optie* wat er wordt uitgevoerd bij een migrate en een rollback.

Tabel vullen

De tabel is nu aangemaakt in de database links. NU gaan we de tabel vullen.

Om aan de slag te gaan is het handig als we onze tabel met wat data kunnen vullen. Ook daar zijn handige commando's voor.

```
php artisan make:model --factory Link
```

Dit commando maakt een model (van MVC) voor het object link en het maakt een file in de directory databases/factories die nodig is om de database te vullen..

Nadat je dit commando hebt uitgevoerd, open je de file LinkFactory.php in de genoemde directory.

Pas de code in deze file aan op de volgende manier:

```
$factory->define(Link::class, function (Faker $faker) {
    return [
    'title' => substr($faker->sentence(2), 0, -1),
'url' => $faker->url,
'description' => $faker->paragraph,
    ];
```

```
});
```

De $faker->sentence(2) method maakt een random zin van 2 woorden en met de substring functie halen we het laatste karakter weg, omdat dat een punt is die we er niet bij willen hebben.

De $faker->url genereert een random url.

En de $faker->paragraph, je raadt het al, maakt een stukje tekst, een paragraaf.

Nu moeten we een script maken dat met deze omschrijving de tabellen vult, dat doen we met:

```
php artisan make:seeder LinksTableSeeder
```

(denk eraan dat je in de juiste directory staat als je een Artisan commando uitvoert)

In de database/seeds directory staat de file LinksTableSeeder.php die we net hebben aangemaakt. Open deze en verander deze.

```
public function run()
{
factory(App\Link::class, 5)->create();
}
```

De 5 geeft aan dat we 5 rijen willen vullen met data.

Open vervolgens de file DatabaseSeeder.php indezelfde directory en plaats de volgende code in deze file:

```
public function run()
{
    $this->call(LinksTableSeeder::class);
}
```

De code in deze file zorgt ervoor dat de LinksTableSeeder.php wordt uitgevoerd.

Let op dat de naam *LinksTableSeeder* case sensitivie is en verwijst naar de file die je in de vorige stap hebt gemaakt.

We gaan een nieuwe migration uitvoeren en zorgen er voor dat tijdens deze migratie de tabellen gevuld worden met data op de manier zoals we zojuist hebben gedefinieerd.

```
php artisan migrate:refresh --seed
```

Controleer met phpmyadmin of de tabel links is gevuld.

Ga door met de volgende les, als er data in de tabel staat.

Samengevat

Er zijn nog wat andere handige commando's die tijdens het ontwikkelen van een applicatie goed van pas kunnen komen, alle handige migratie commando's op een rijtje:

Commando	Beschrijving
php artisan migrate	**voer de migratie uit**
php artisan migrate:rollback	maak de migratie weer ongedaan
php artisan migrate:refresh	**doe een rollback en een migrate**
php artisan migrate:refresh --seed	doe een rollback en een migrate en vul de tabellen met data
php artisan migrate:fresh	drop alle tabellen en doe opnieuw een migrate
php artisan help	alle commando's
php artisan migrate help	help over het commando migrate

We zijn de volgende files tegengekomen:

files (vanuit project directorie)	Beschrijving
database/migrations/<create_table>.php	definieer hoe de tabel moet worden gemaakt
.env	file met configuratie opties zoals database opties
database/factories/<table>Factory.php	Definieer hoe tabel te vullen met test data
database/seeds/<table>TableSeeder.php	Roep functie X keer aan om rijen te genereren
database/seeds/DatabaseSeeder.php	Vanuit hier dient de TableSeeder te worden aangeroepen

Comment

(door Rocco)

voor de error met Artisan migrate die met length te maken heeft
ga naar de AppServiceProvider.php file in je project folder en pas dit aan

```
<?php


namespace App\Providers;
```

```
use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Schema;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }

    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        Schema::defaultStringLength(191);
    }
}
```

Revision #27
Created 2019-10-19 15:29:15 UTC by Admin
Updated 2020-08-29 10:15:44 UTC by Max