Routes en Read

In deze les leren we wat een virtuele app is en hoe we deze in Apache configureren.

Daarna gaan we een route in ons Laravel project opzetten en laten we de inhoud van onze table links die we in de vorige les hebben gemaakt, met een blade template afdrukken. We gaan de R van CRUD maken.

Document Root

Even eenreminder. De *document root* is de directory waar je webserver begint. De document root is een configuratie parameter in de webserver config file. De parameter verwijst naar een directory op je file systeem en deze directory is waar de webserver zijn web pagina's verwacht.

Virtual App

Het opzetten van een virtuele app onder XAMPP blijkt niet goed te werken. Ik moet nog een keer uitzoeken waarom niet. Voor nu kan dit hoofdstuk over Virtual App worden overgeslagen.

De nieuwe Laravel App staat onder localhost/links/public waarbij links/public in de document root staat.

Op de file: C:\xampp\apache\conf\extra\httpd-vhosts.conf en plaats de volgende configuratie:

```
<VirtualHost *:80>
    ServerName links.local
    DocumentRoot "C:/xampp/htdocs/links/public"
    <Directory C:/xampp/htdocs/links/public>
        Require all granted
        AllowOverride All
    </Directory>
</VirtualHost>
```

Restart de Apache webserver. Wat er nu is ingesteld is dat als de webserver een verzoek krijgt van het domein links.local de webserver's document root verandert naar c:/xampp/htdocs/links.

Nu moeten we alleen nog zorgen dat links.local naar het ip adress 127.0.0.1 resolved. Dat kan door de hosts file onder windows aan te passen. Open notepad in admin mode en open C:\Windows\System32\drivers\etc\hosts

Voeg de volgende regel toe:

```
127.0.0.1 links.local
```

Deze regel verteld jouw computer dat als je links.local in tikt hierbij het ipaddres 127.0.0.1 hoort. Je computer gaat dan niet meer verder naar de DNS van internet om te vragen wat het ip address van links.local is.

Routes

Een route is een manier voor de webserver om te weten welke pagina hij moet laden. In een traditionele web app komt het pas in de URL overeen met het path in je document root.

Bijvoorbeeld als je gaat naar localhost/myproject/b/c/input.php

Dan kun je in je *document root* een directory vinden die myproject heet en daarin staat een directory die b heet en daarin staat dan weer de directory c waarin uiteindelijk de file input.php staat.

In Laravel werkt dit anders.

Via een slimme truuk wordt alles wat naar Laravel verwijst opgevangen en het path uit de url wordt door de webserver en Laravel ontleed. Stel myproject was een Laravel project in het vorige voorbeeld dan wordt de rest van het path b/c/input.php als een parameter aan Laravel doorgegeven en kun je in Laravel door middel van het definiëren van een route aangeven wat er moet gebeuren.

Ga naar de file routes/web.php en open deze. Plaats de volgende code:

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

Ga nu terug naar je Laravel project in de browser en controleer of je (standaard) web pagina nu ander /welcome staat. Je hebt nu de standaard Laravel welcome pagina verhuisd van / naar /welcome .

Laten we een route toevoegen, door het volgende toe te voegen:

```
Route::get('/showlinks', function () {
    $links = \App\Link::all();
    echo "<pre>";
    print_r($links);
    echo "</pre>";
```

```
});
```

Je ziet dat $links een (Laravel) object is. Als je goed kijkt herken je de resultaten van de query en van de tabel links.

Maak nu in resources/views een nieuwe view aan. Je ziet daar de welcome.blade.php file staat. Dat is de eerste standaard pagina, die we al zagen. Maak nu een nieuwe template aan en noem die links.blade.php

Plaats nu eerst de volgende route in de web.php file.

```
Route::get('/links', function () {
    $links = \App\Link::all();
    return view('links', ['links' => $links]);
});
```

Wat hier staat is het volgende:

1. als je naar de link /links gaat dan
2. haal je alle links op en zet je deze in een object, dan
3. geeft je dit object mee als value van een assiociative array aan de view 'links' en dan
4. return de output van deze view (naar de browser).

We gaan nu naar de zo net aangemaakte pagina resources/view/links.blade.php en maken de volgende code:

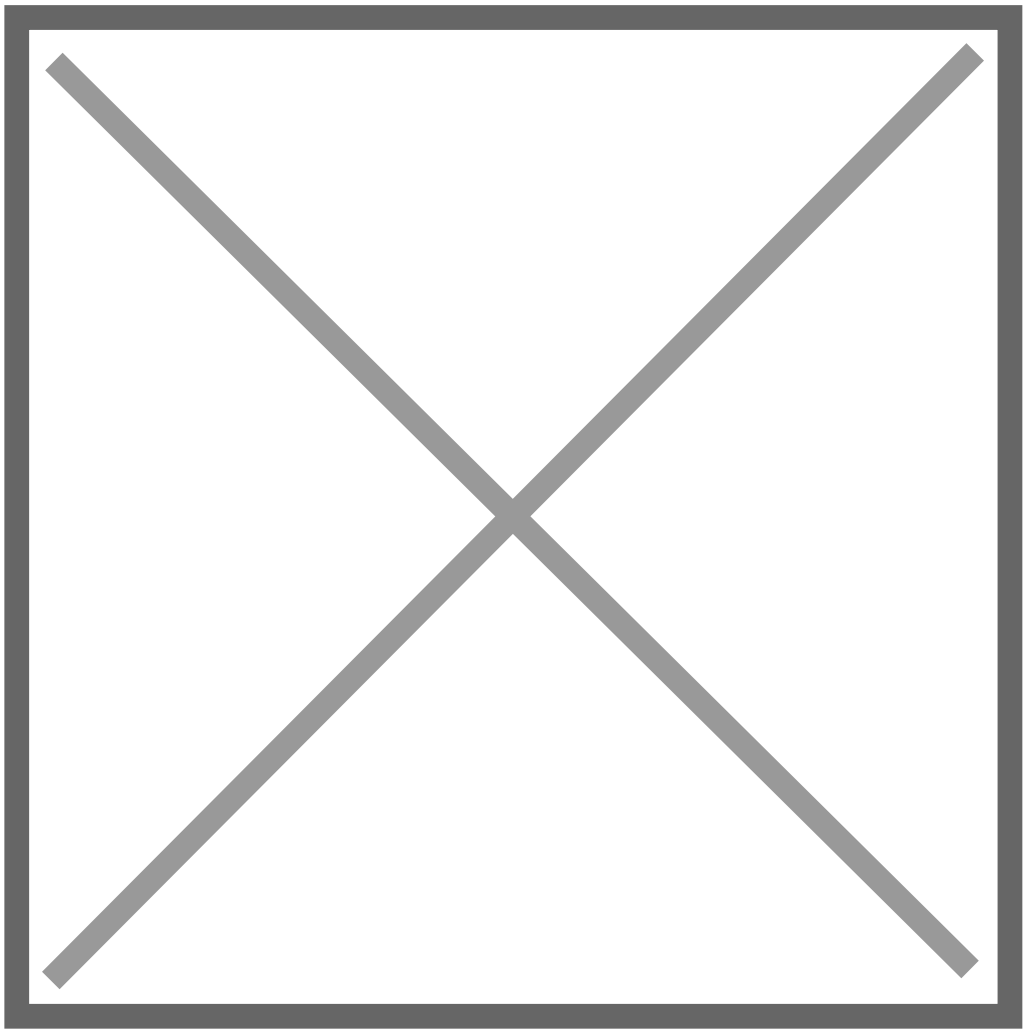
```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <title>Laravel</title>
    </head>

    @foreach ($links as $link)
      <a href="{{ $link->url }}">{{ $link->title }}</a><br>
    @endforeach

</html>
```

Dit is een Blade template. In principe is het HTML maar er zitten speciale statements in. Zo zie je een @foreach loop die door het object $links loopt. Een beetje zoals we eerder hebben geoefend met assiociative arrays (2) in een tabel afdrukken.

Ook zien we in de balde template fat de url en title uit het links object worden afgedrukt. We hebben ook nog een description in de links tabel, weet je nog?

Opdracht

Pas nu de code in de blade template aan zodat de links en de description in een tabel worden afgedrukt. De output moet er als volgt komen uit te zien.



--

Revision #12
Created 2019-10-21 12:54:23 UTC by Admin
Updated 2020-07-01 21:53:25 UTC by Max