**ROUTING**

Routing is een essentieel concept in webontwikkeling dat **verwijst naar het proces van het bepalen van hoe een webapplicatie reageert op verschillende URL's die worden aangevraagd door de gebruiker**. Het bepaalt **welke code of actie moet worden uitgevoerd** op basis van de URL die wordt opgevraagd door de browser van de gebruiker.

**Http-requests en URL's.**

Met een http-request (get of post) wordt een http-server **gevraagd een bestand** (html, css, jpg, js etc) **terug te geven of een actie uit te voeren gevolgd door een respons**. Dat kan zijn: het ophalen van gegevens in de vorm van een web-pagina of json; of het wegzetten van ingevoerde gegevens; of het wijzigen van de status van bepaalde gegevens.

Een http-request bestaat uit:

* een **http-methode**: get, post, put, delete, etc.;
* een **url**: bijvoorbeeld: https://autonet.nl/cars/edit/2531 of http://clients/view/19283?detailed=yes&lang=EN.

en kan ook bevatten:

* een **payload** in geval van een form. Payload wil zeggen de verzameling inputs uit de form met hun ingevoerde values;
* **clientside** **cookies**;
* eventuele **extra request-headers**.

**Van bestandgerichte requests (vroeger).**

In het verleden werd met een http-request-URL altijd **rechtstreeks een script-bestand opgevraagd**, die de benodigde actie uitvoerde en een bijbehorende respons teruggaf. Toen zagen URL’S er zo uit:  
https://doctor.com/editclients.php?id=92018&profile=1

En URL's voor afbeeldingen, stylesheets en javascriptbestanden zo:  
https://doctor.com/style/general.css

**Naar taakgerichte requests (nu).**

Tegenwoordig worden request-URL's veelal gestructureerd **per soort gegeven (resource) en taak**. Een moderne, professionele request-URL heeft in het algemeen de volgende opbouw:  
https://doctor.com/client/edit/92018/?profile=1&format=json

In de resource zul je tabellen van de database herkennen. In de task zul je CRUD taken ontdekken, naast andere taken. De identifiers wijzen naar specifieke records binnen de resource en de parameters zijn extra aanwijzingen voor de server over de gewenste afhandeling van de request of over de gewenste content of format van de response.

**Hoe werkt dan Routing?**

Routing is de techniek in de webapplicatie die uitgaande van een bepaalde URL regelt welke script/function/methode de http-request zal afhandelen. Maar hoe gaat dat in zijn werk?

* **Documentroot**:
  + De documentroot van de server is **gelijkgesteld aan public map van Laravel**. Dat betekent dat **de server voor de afhandeling van een http-request gaat zoeken naar een document of bestand te beginnen bij de public map**. Deze public map bevat de openbare resources zoals images, stylesheets en javascriptbestanden.   
    **En let op: het bevat ook de index.php**. De index.php is de start-script voor alle taakgerichte http-requests (dus requests die niet simpel om een afbeelding, stylesheet of javascript verzoeken).
* **Request rewrite**:
  + Moderne taakgerichte URL's bevatten geen script bestandsnaam meer. Hoe weet de server dan welke PHP-bestand de request moet gaan afhandelen?   
    In Laravel moet dat index.php zijn**. Dat wordt opgelost door zogeheten 'request rewriting' op de server**. In de public map (documentroot) waar de server het eerst een bestand gaat zoeken staat een configuratiebestand met de naam: .htaccess. Daarin staan **instructies (rewrite directives) die de request URL herschrijven naar een script bestandsnaam**. De server zal eerst deze instructies uitvoeren.

Dus een op de server binnenkomende URL als:  
- http://autonet.nl/cars/view/11827?detailed=yes&pictures=no  
**WORDT** (in framework Laravel\*) door de http-server volgens de .htaccess directives herschreven naar een URL:  
- http://autonet.nl/index.php

Dus de server gaat na de 'request rewrite' **de index.php laden en uitvoeren**. Maar wacht even... de informatie over resource, task en parameters is helemaal uit de URL verdwenen. Gaat dat wel goed? Geen nood, de s**erver heeft die info netjes voor de PHP-script bewaard in**:

**$\_SERVER['REDIRECT\_URL']** (bevat in het voorbeeld: /cars/view/11827)

**$\_SERVER['REDIRECT\_QUERY\_STRING']** (bevat in het voorbeeld: ?detailed=yes&pictures=no)

* **URL analyse en matching**:
  + In een Laravel webapplicatie moet je (als developer**) zelf expliciet definiëren welke soorten URL's je laat afhandelen door welke classes, methodes en functions**. Die definities voeg je toe **in routes/web.php** in een aanroep van $router->get(<patroon>,<actie>). Er zijn heel veel mogelijkheden om te definiëren welke soort URL welke afhandeling krijgt. Patronen kan je ook definiëren met een regularexpression.

Laravel zal de binnenkomende 'redirected URL' **proberen te matchen met een van een patroon van een van de definities**. Als dat lukt dan wordt de bijbehorende methode of function aangeroepen.

Laravel zal ook proberen om **gegevens** uit de 'redirected URL' en de 'redirected query **string' toe te voegen als argumenten** van de aanroep van de betrokken methode of function.