⬅Revenir à la théorie.

Le Routage dans Laravel

- 1. Introduction
- 2. Une vue (view)
- 3. Routage sur une vue (view)
- 4. Routage retournant une string
- 5. Routage avec POST
- 6. Routage avec paramètres
- 7. Routage avec paramètres et contraintes
- 8. Routage retournant un tableau associatif
- 9. Routage sur un contrôleur
- 10. Nommage d'une route
- 11. Redirections
    - 11.1 Sur une route
    - 11.2 Sur une route nommée
    - 11.3 Sur une URL
- 11. Ordre des routes
- 12. Liste des routes via php artisan

1. Introduction

Dans l'arborescence de votre projet, vous pouvez voir qu'il y a un fichier qui s'appelle web.php dans le dossier routes.

Dans celui-ci on y décrit les différentes routes pour notre application.

Une route peut rediriger vers une vue mais ce n'est pas obligation.
La documentation de Laravel pour le routage se trouve à cette adresse https://laravel.com/docs/11.x/routing

2. Une vue (view)

Mais qu'est-ce qu'une vue ?
Il faut prendre cela comme on l'entend: une vue, c'est quelque chose que l'on voit. Et dans le monde du web ce que l'on voit c'est du HTML.
Une vue affichera donc du code HTML et d'autres gentilles choses que nous verrons au fil du temps. 😀
La définition de toutes nos vues se trouve dans le dossier: resources/views/
Par défaut, Laravel a une vue par défaut qui s'appelle 'welcome'.
On notera qu'elle se nomme welcome.blade.php
Le nom de fichier de nos vues auront donc cette syntaxe: nomdelavue.php ou bien nomdelavue.blade.php

3. Routage sur une vue (view)

Si l'on analyse ce fichier, nous y voyons une route qui est définie:

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

Ce code appelle la méthode get de la classe Route avec les paramètres suivants:

- Le premier paramètre est le chemin. Ici c'est la racine du site: /
- Le second paramètre est une méthode anonyme qui retourne une vue qui s'appelle 'welcome'.

Il ne faut pas mettre return view('welcome.blade.php');
Mettre le nom de la vue suffit: return view('welcome')

Ca veut dire:

> Hey Laravel, va dans le répertoire resources/views et prends la vue qui s'appelle welcome dont le nom de fichier est welcome.blade.php ou welcome.php

Notons encore que la méthode get suppose que la requête a été initiée par la méthode GET sinon on a aussi une méthode post dans la classe Route.

Cette route pourrait s'écrire aussi:

```
Route::view('/', 'welcome');
```

Ici, c'est un raccourci qui appellera la méthode view qui appellera la vue 'welcome' et retournera une route.

4. Routage retournant une string

```
Route::get('/salutation', function () {
    return "Salut ! Bienvenue sur ce site !";
});
```

Dans cette route, on va rediriger l'utilisateur si l'on essaie de charger l'url /salutation. Il sera affiché dans le navigateur: Salut ! Bienvenue sur ce site !

5. Routage avec POST

On a vu pour le moment le routage avec la méthode GET. (via l'url)

Voyons maintenant avec la métode POST.

```
Route::post('/chemin', function(){
    return "via post";
});
```

6. Routage avec paramètres

Avoir des paramètres dans des routes se gère de la manière suivante:

```
Route::get('article/{numero}', function($numero)
{
    return "Le numéro de l'article est $numero";
});
```

On voit que l'on doit mettre le paramètre entre accolades. Ensuite, la fonction anonyme pourra l'utiliser en reprenant le nom entre accolades en le préfixant de $.

On peut avoir autant de paramètres que l'on le souhaite.

```
Route::get('article/{numero}/{allee}', function($numero, $allee)
{
    return "Le numéro de l'article est $numero et il est rangé dans l'allée n°$allee.";
});
```

Mais on peut aussi tout regrouper dans une seule en rendant le deuxième paramètre optionnel :

```
Route::get('article/{numero}/{allee?}', function($numero, $allee = null)
{
    if(!is_null($allee)) return "Le numéro de l'article est $numero et il est rangé dans
    else return "Le numéro de l'article est $numero";
});
```

Quand on ajoute un point d'interrogation à la fin du paramètre, on le rend optionnel.

7. Routage avec paramètres et contraintes

Les contraintes vont permettre de filtrer notre résultat.
Ici nous allons rediriger uniquement si le paramètre est bien un nombre en utilisant ->whereNumber('variable')

```
Route::get('article/{numero}', function($numero)
{
    return "Le numéro de l'article est $numero.";
})->whereNumber('numero');
```

On aurait pu l'écrire aussi de la sorte avec ce qu'on appelle une expression régulière (appelée aussi regex) dans une contrainte where:

```
Route::get('article/{numero}', function($numero)
{
    return "Le numéro de l'article est $numero.";
})->where('numero', '[0-9]+');
```

Si l'on doit faire une contrainte qui porte sur plusieurs paramètres, on utilisera un tableau dans la clause where:

```
Route::get('article/{numero}/{allee?}', function($numero, $allee = null)
{
    if(!is_null($allee)) return "Le numéro de l'article est $numero et il est rangé dans
    else return "Le numéro de l'article est $numero";
})->where(array('numero' => '[0-9]+', 'allee' => '[A-z0-9]+'));
```

Pour le paramètre 'numero' on accepte que des chiffres et pour le paramètre 'allee', on n'accepte que lettres et chiffres.
Elle aurait pu s'écrite de la sorte :

```
Route::get('article/{numero}/{allee}', function($numero, $allee)
{
    return "Le numéro de l'article est $numero et il est rangé dans l'allée n°$allee.";
})->whereNumber('numero')->whereAlphaNumeric('allee');
```

Il existe un ensemble de contraintes déjà définies:

- whereAlpha('toto'): vérifie que toto ne contient que des lettres.
- whereNumber('toto'): vérifie que toto est un nombre.
- whereAlphaNumeric('toto'): vérifie que toto contient du texte et/oui des chiffres.
- whereUuid: vérifie que c'est un identifiant unique universel.

Pour information, Laravel permet de créer des identifiants uniques (Uuid) assez simplement:

```
$uuid = (string)Str::uuid();
```

Laravel fournit des fonctions globales d'aide (Helper). Plusieurs de ces fonctions sont utilisées par le framework mais vous pouvez aussi les utiliser: https://laravel.com/docs/11.x/helpers

8. Routage retournant un tableau associatif

Lorsqu'on retourne un tableau associatif comme ceci:

```
Route::get('/json', function(){
    return ["foo"=>"bar"];
});
```

Voici ce que reçoit et affiche un navigateur:

```
{"foo":"bar"}
```

C'est du json (Javascript Object Notation)

9. Routage sur un contrôleur

On ne va pas trop s'étendre dessus car nous verrons plus tard les contrôleurs.
Evidemment, une route peut rediriger vers une contrôleur. Celui-ci fera ce qu'il a à faire et reverra vers une vue ou chaine, tableau json, etc.

```
Route::get('/str', [WelcomeController::class,'index']);
```

Ce contrôleur pourrait ressembler à ceci:

```
class WelcomeController extends Controller
{
    public function index()
    {
        return 'Hello, World!';
    }
}
```

10. Nommage d'une route

Par facilité, on peut aussi nommer une route par exemple pour générer une url ou pour effectuer une redirection:

```
Route::get('/home', function () {
    return "Je suis la page d'accueil";
})->name('welcome');
```

11. Redirections

Laravel fournit la possibilité de facilement effectuer des redirections sur des routes et url.

11.1 Sur une route

On fait une redirection de la route /accueil vers la route nommée 'welcome'.

```
Route::get('/accueil',function(){
    return redirect('home');
});
```

11.2 Sur une route nommée

On fait une redirection de la route /accueil vers la route nommée 'welcome'.

```
Route::get('/accueil',function(){
    return redirect()->route('welcome');
});
```

On constate qu'on appelle après la méthode redirect() la méthode route avec comme paramètre le nom de la route 'welcome'.

11.3 Sur une URL

Il est possible aussi de faire une redirection sur une url au lieu d'une route.

```
Route::get('/google',function(){
    return redirect("http://www.google.be");
});
```

Quand on appelera la route /google on sera redirigé vers le site de Google.

11. Ordre des routes

Attention, il est important de bien faire attention à l'ordre de nos routes.
Prenons un exemple

```
Route::get('/{str}',function($str){
    return "Je suis la page $str.";
});

Route::get('/toto',function(){
    return "Je suis la page super page de toto !";
});
```

Que va-t-il s'afficher si l'on va sur la page http://site/toto ?

12. Liste des routes via php artisan

On peut lister les routes définies via la commande

```
php artisan route:list
```

Pour plus d'informations: https://stackoverflow.com/questions/22118598/laravel-routes-gethead

⬅Revenir à la théorie.
