


Cours Laravel 6 – les bases – artisan et les contrôleurs

 laravel.sillo.org/cours-laravel-6-les-bases-artisan-et-les-contrôleurs/

bestmomo

August 28,
2019

Nous avons vu le cycle d'une requête depuis son arrivée, son traitement par les routes et sa réponse avec des vues qui peuvent être boostées par Blade. Avec tous ces éléments vous pourriez très bien réaliser un site web complet mais Laravel offre encore bien des outils performants que je vais vous présenter.

Pour correctement organiser son code dans une application Laravel il faut bien répartir les tâches. Dans les exemples vus jusqu'à présent j'ai renvoyé une vue à partir d'une route, vous ne ferez pratiquement jamais cela dans une application réelle (même si personne ne vous empêchera de le faire !). Les routes sont juste un système d'aiguillage pour trier les requêtes qui arrivent.

Mais alors qui s'occupe de la suite ?

Et bien ce sont les contrôleurs, le sujet de ce chapitre.

Nous allons aussi découvrir l'outil **Artisan** qui est la boîte à outil du développeur pour Laravel.

Artisan

Lorsqu'on construit une application avec Laravel on a de nombreuses tâches à accomplir, comme par exemple créer des classes, vérifier les routes...

C'est là qu'intervient Artisan, le compagnon indispensable. Il fonctionne en ligne de commande, donc à partir de la console. Il suffit de se positionner dans le dossier racine et d'utiliser la commande :

```
php artisan
```

pour obtenir la liste de ses possibilités, en voici un extrait :

```

λ php artisan
Laravel Framework 6.0-dev

Usage:
  command [options] [arguments]

Options:
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
      --ansi            Force ANSI output
      --no-ansi        Disable ANSI output
  -n, --no-interaction Do not ask any interactive question
      --env[=ENV]      The environment the command should run under
  -v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more details, 3 for debugging

Available commands:
  clear-compiled  Remove the compiled class file
  down           Put the application into maintenance mode
  env           Display the current framework environment
  help          Displays help for a command
  inspire       Display an inspiring quote
  list          Lists commands
  migrate       Run the database migrations
  optimize      Cache the framework bootstrap files
  preset        Swap the front-end scaffolding for the application
  serve         Serve the application on the PHP development server
  tinker        Interact with your application
  up           Bring the application out of maintenance mode
  auth
  auth:clear-resets Flush expired password reset tokens

```

Nous verrons peu à peu les principales commandes disponibles. Il y en a une pour connaître les routes prévues dans le code. Voici ce que ça donne avec une nouvelle installation :

```

λ php artisan route:list
+-----+-----+-----+-----+-----+-----+
| Domain | Method | URI      | Name | Action | Middleware |
+-----+-----+-----+-----+-----+-----+
|         | GET|HEAD | /        |      | Closure | web         |
| nombre 20 | GET|HEAD | api/user |      | Closure | api,auth:api |
+-----+-----+-----+-----+-----+-----+

```

On sait que la seule route au départ est celle-ci :

```

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

```

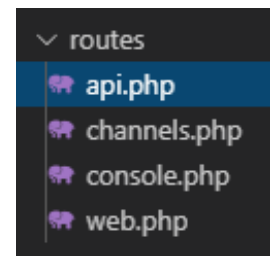
Elle correspond à la première ligne de la liste avec une closure.

Mais à quoi correspond la seconde route ?

On a vu qu'il y a 4 fichiers de routes, en particulier on a celui pour les API :

Lui aussi comporte une route par défaut :

```
Route::middleware('auth:api')->get('/user', function (Request $request) {  
    return $request->user();  
});
```



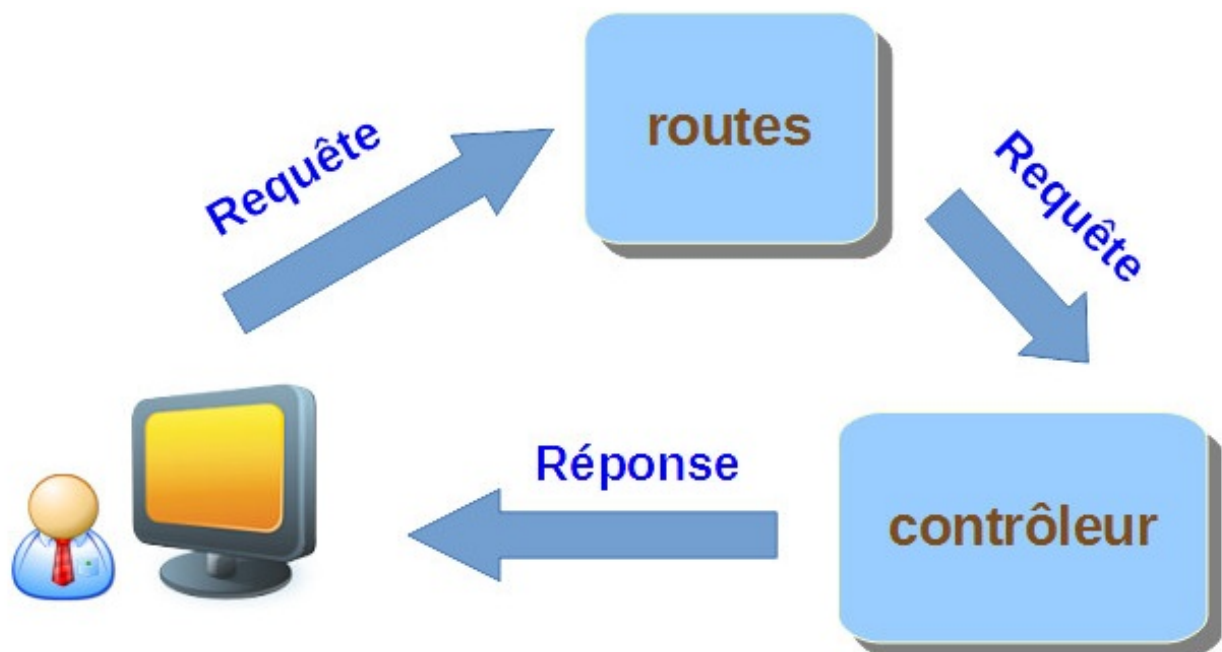
Une API est un système sans état qui se contente de fournir des ressources. je ne vais pas évoquer cette possibilité dans ce cours. Vous pouvez supprimer cette route pour éviter de polluer votre liste de routes.

Je parlerai des middlewares dans un prochain chapitre.

Les contrôleurs

Rôle

La tâche d'un contrôleur est de réceptionner une requête (qui a déjà été sélectionnée par une route) et de définir la réponse appropriée, rien de moins et rien de plus. Voici une illustration du processus :



Constitution

Pour créer un contrôleur nous allons utiliser Artisan. Dans la console entrez cette commande :

```
php artisan make:controller WelcomeController
```

Si tout se passe bien vous allez trouver le contrôleur ici :

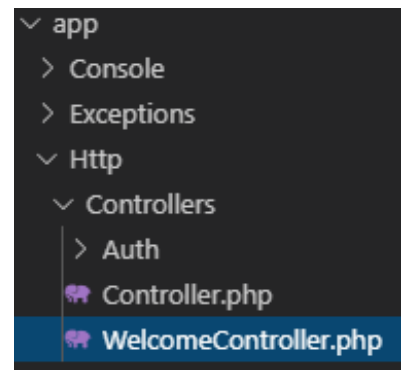
Avec ce code :

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class WelcomeController extends Controller
{
    //
}
```



Ajoutez la méthode **index** :

```
<?php
...
class WelcomeController extends Controller
{
    public function index()
    {
        return view('welcome');
    }
}
```

Analysons un peu le code :

- on trouve en premier l'espace de nom (**App\Http\Controllers**),
- le contrôleur hérite de la classe **Controller** qui se trouve dans le même dossier et qui permet de factoriser des actions communes à tous les contrôleurs,
- on trouve enfin la méthode **index** qui renvoie quelque chose que maintenant vous connaissez : une vue, en l'occurrence « welcome » dont nous avons déjà parlé.
Donc si j'appelle cette méthode je retourne la vue « welcome » au client.

Liaison avec les routes

Maintenant la question qu'on peut se poser est : comment s'effectue la liaison entre les routes et les contrôleurs ?

Ouvrez le fichier des routes et entrez ce code :

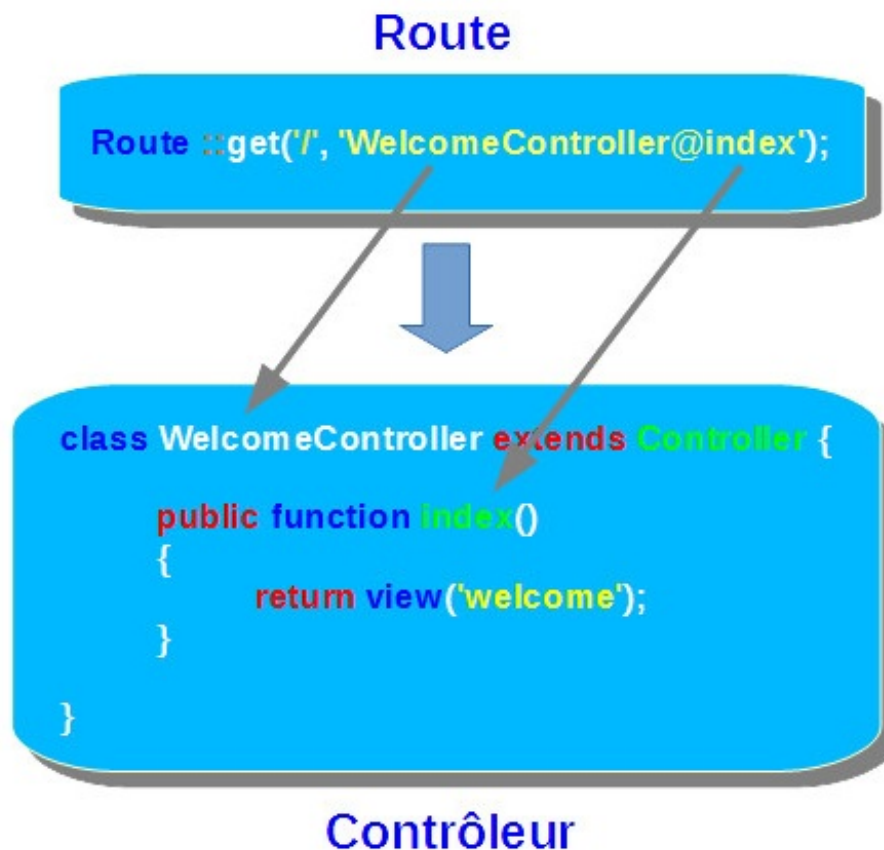
```
Route::get('/', 'WelcomeController@index');
```

Maintenant avec l'url de base vous devez retrouver la page d'accueil de Laravel :

Laravel

[DOCS](#)[LARACASTS](#)[NEWS](#)[BLOG](#)[NOVA](#)[FORGE](#)[VAPOR](#)[GITHUB](#)

Voici une visualisation de la liaison entre la route et le contrôleur :



On voit qu'au niveau de la route il suffit de désigner le nom du contrôleur et le nom de la méthode séparés par @.

Si vous êtes attentif au code vous avez sans doute remarqué qu'au niveau de la route on ne spécifie pas l'espace de noms du contrôleur, on peut légitimement se demander comment on le retrouve. Laravel nous simplifie la syntaxe en ajoutant automatiquement cet espace de nom.

Route nommée

De la même manière que nous pouvons nommer une route classique on peut aussi donner un nom à une route qui pointe une méthode de contrôleur :

```
Route::get('/', 'WelcomeController@index')->name('home');
```

Si on utilise Artisan pour lister les routes :

```
λ php artisan route:list
```

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/	home	App\Http\Controllers\WelcomeController@index	web

On voit bien que l'action est faite par le contrôleur avec précision de la méthode à utiliser. On trouve aussi le nom de la route.

Utilisation d'un contrôleur

Voyons maintenant un exemple pratique de mise en œuvre d'un contrôleur. On va conserver notre exemple avec les articles mais maintenant traité avec un contrôleur. On conserve le même template et les mêmes vues :

On va créer un contrôleur (entraînez-vous à utiliser Artisan) pour les articles :

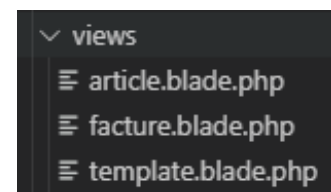
```
<?php
```

```
namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
```

```
use App\Http\Requests;
```

```
class ArticleController extends Controller
{
    public function show($n)
    {
        return view('article')->with('numero', $n);
    }
}
```



Dans ce contrôleur on a une méthode **show** chargée de générer la vue. Il ne nous reste plus qu'à créer la route :

```
Route::get('article/{n}', 'ArticleController@show')->where('n', '[0-9]+');
```

Voici une illustration du fonctionnement avec ce contrôleur :

Route

```
Route ::get('article/{n}', 'ArticleController@show')->where('n', '[0-9]+');
```



```
class ArticleController extends BaseController {  
    public function show($n)  
    {  
        return View::make('article')->with('numero', $n);  
    }  
}
```

Contrôleur

Notez qu'on pourrait utiliser la méthode « magique » pour la transmission du paramètre à la vue :

```
return view('article')->withNumero($n);
```

En résumé

- Les contrôleurs servent à réceptionner les requêtes triées par les routes et à fournir une réponse au client.
- Artisan permet de créer facilement un contrôleur.
- Il est facile d'appeler une méthode de contrôleur à partir d'une route.
- On peut nommer une route qui pointe vers une méthode de contrôleur.