Chapitre 6
Organiser grâce aux contrôleurs

1. Définition et usage

1.1 Qu'est-ce qu'un contrôleur ?

Dans le modèle MVC (modèle-vue-contrôleur), le contrôleur contient la logique concernant les actions effectuées par l'utilisateur. En pratique, dans une application Laravel, l'utilisation de contrôleurs permet de libérer les routes du code qu'elles contiennent dans leurs fonctions de rappel.

Un contrôleur est matérialisé par une classe et chacune de ses méthodes représente une action. Une action correspond généralement à une route.

1.2 Déplacer le code des fonctions de rappel des routes

Les routes créées dans les chapitres précédents sont surchargées de code pour traiter les actions dans leurs fonctions de rappel. L'objectif est de les libérer de ce code pour obtenir un fichier de routes propre qui fait simplement le lien entre les URI de l'application et les actions à effectuer pour chacune de ces URI.

Le fichier des routes `routes/web.php` est ainsi allégé, et devient un index des différentes URI avec leurs actions associées.

Exemple de fichier de routes sans fonction de rappel

```
Route::get('videos', 'VideoController@index');

Route::get('videos/creer', 'VideoController@create');

Route::post('videos/creer', 'VideoController@store');

Route::get('videos/{id}', 'VideoController@show');
```

Le fonctionnement des routes avec les contrôleurs est détaillé dans la suite de ce chapitre.

2. Créer et utiliser un contrôleur

2.1 Créer un contrôleur avec Artisan

Comme pour de nombreux autres éléments, Artisan, l'outil en ligne de commande fourni avec Laravel, permet de créer rapidement un fichier contenant la structure de base d'un contrôleur :

Créer un contrôleur

```
php artisan make:controller ProductController
```

Les contrôleurs sont créés dans le dossier `app/Http/Controllers`.

Remarque

Par convention, les contrôleurs sont en notation Camel Case et terminent tous avec le mot ***`Controller`****. Cela permet de les différencier des services et des modèles auxquels ils font référence.*

2.2 Structure d'un contrôleur

Un contrôleur est une classe qui étend la classe de base **`Controller`** et dont chaque méthode publique représente généralement une action qui correspond à une route.

Fichier `app/Http/Controllers/ProductController.php`

```
<?php

namespace App\Http\Controllers;

class ProductController extends Controller
{

    public function home()
    {
        return 'Bienvenue sur la page des produits';
    }

    public function show($id)
    {
        return 'Page du produit : ' . $id;
    }
}
```

Une méthode d'un contrôleur retourne une réponse, exactement comme dans les fonctions de rappels du chapitre « Créer des routes » à propos des routes. Dans l'exemple précédent, les méthodes **`home`** et **`show`** retournent une réponse sous la forme d'une chaîne de caractères, mais il aurait aussi bien pu s'agir d'une vue ou d'un objet JSON. Le code ci-dessous présente le même contrôleur, dont les méthodes retournent cette fois des vues.

Fichier `app/Http/Controllers/ProductController.php`

```
<?php

namespace App\Http\Controllers;

use App\Product;

class ProductController extends Controller
{
```

```
    public function home()
    {
        return view('products.home');
    }

    public function show($id)
    {
        return view('products.show', [
            'product' => Product::find($id)
        ]);
    }
}
```

2.3 Lier une route à un contrôleur

Dans le fichier `routes/web.php`, à la place d'utiliser des fonctions de rappel comme cela a été le cas dans les précédents chapitres, l'organisation avec des contrôleurs permet de simplement signaler au routeur quel contrôleur aller chercher et quelle action effectuer. La syntaxe à utiliser est **`NomDuControl-ler@action`** sous forme d'une chaîne de caractères.

Lier une route à un contrôleur

```
Route::get('products', 'ProductController@home');

Route::get('products/{id}', 'ProductController@show');
```

L'action correspond au nom de la méthode du contrôleur à exécuter.

2.4 Variables

2.4.1 Passer un paramètre à un contrôleur

Pour passer une variable à un contrôleur, il faut la déclarer dans le chemin de la route entre accolades et mettre la variable en paramètre de la méthode qui représente l'action à effectuer, comme dans l'exemple ci-dessous.

Fichier `routes/web.php`

```
Route::get('products/{id}', 'ProductController@show');
```

Fichier app/Http/Controllers/ProductController.php

```
class ProductController extends Controller
{

    public function show($id)
    {
        return 'Page du produit : ' . $id;
    }
}
```

2.4.2 Valeurs optionnelles

Les paramètres optionnels (qui permettent par exemple d'associer une même action aux URI `products` et `products/17`) doivent être déclarés avec un point d'interrogation dans le chemin de la route et avec une valeur par défaut à **`null`** dans l'action du contrôleur.

Fichier routes/web.php

```
Route::get('products/{id?}', 'ProductController@show');
```

Fichier app/Http/Controllers/ProductController.php

```
class ProductController extends Controller
{

    public function show($id = null)
    {

        if ($id === null) {
            return Product::all();
        } else {
            return Product::findOrFail($id);
        }

    }
}
```

3. Dépendances

3.1 Injection de dépendance

3.1.1 Fonctionnement

Il est possible grâce aux fonctions assistantes et au système de façades fourni par Laravel d'accéder à de nombreux éléments dans les contrôleurs comme la requête de l'utilisateur, le cache, les sessions, etc.

Il est également possible d'injecter les dépendances directement dans les méthodes des contrôleurs, simplement en donnant l'indication d'un type connu par Laravel à un paramètre de la méthode représentant l'action.

Par exemple, pour récupérer une instance de la classe **`Illuminate\Http\Request`**, on peut passer à une méthode un premier paramètre **`$request`** préfixé par son type.

Affichage de l'URL dans une méthode de contrôleur

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller
{

    public function home(Request $request)
    {
        return 'L'URL de la page est :' . $request->url();
    }
}
```

Remarque

*La classe **`Request`** et la fonction assistante pour y accéder seront étudiées dans la suite de ce chapitre et au chapitre « Traiter des formulaires », les sessions et le cache au chapitre « Les sessions et le cache ». En attendant, on retiendra simplement qu'on peut transmettre d'autres éléments du framework à une méthode du contrôleur.*