



Framework PHP : LARAVEL

Les contrôleurs

Mouhamed DIOP - Ingénieur de Conception en Informatique (ESP)

Objectifs spécifiques

A l'issue de cette séquence, l'étudiant devrait :

- Savoir ce que représente un contrôleur
- Pouvoir créer des contrôleurs
- Pouvoir définir des actions au niveau d'un contrôleur
- Pouvoir associer des routes aux contrôleurs créés



Qu'est-ce qu'un contrôleur?

Les contrôleurs sont destinés à regrouper, au sein d'une même classe, la logique de traitement des requêtes qui ont un certain lien

- Au lieu de définir toute la logique de gestion des requêtes en tant que **closures** dans les fichiers de routage, il est possible de la répartir sur plusieurs contrôleurs
- L'idée c'est par exemple de regrouper la gestion de toutes les requêtes concernant les clients (ajout, suppression, ...) dans un contrôleur, puis celle des produits dans un autre contrôleur, et ainsi de suite
- De ce fait, un contrôleur n'est rien d'autre qu'une classe capable de gérer un certain nombre de requêtes
- Les contrôleurs sont stockés dans le répertoire **app/Http/Controllers**
- Tous les contrôleurs héritent du contrôleur de base de Laravel (**Illuminate\Routing\Controller**)
 - Pas obligatoire, mais permet d'accéder à des fonctionnalités clé (middleware, etc.)

Création d'un contrôleur

La création d'un contrôleur se fait à l'aide de l'outil Artisan fourni par Laravel

- Pour ce faire, il suffit de se positionner dans la racine du projet et de taper la commande
php artisan make:controller nom_controleur

Exemple : **php artisan make:controller ArticleController** pour créer un contrôleur de gestion des articles

- Une fois la commande exécutée, le contrôleur sera créé et placé dans le bon répertoire
 - Pour l'exemple, le fichier **ArticleController.php** sera créé et placé dans le répertoire **app/Http/Controllers**

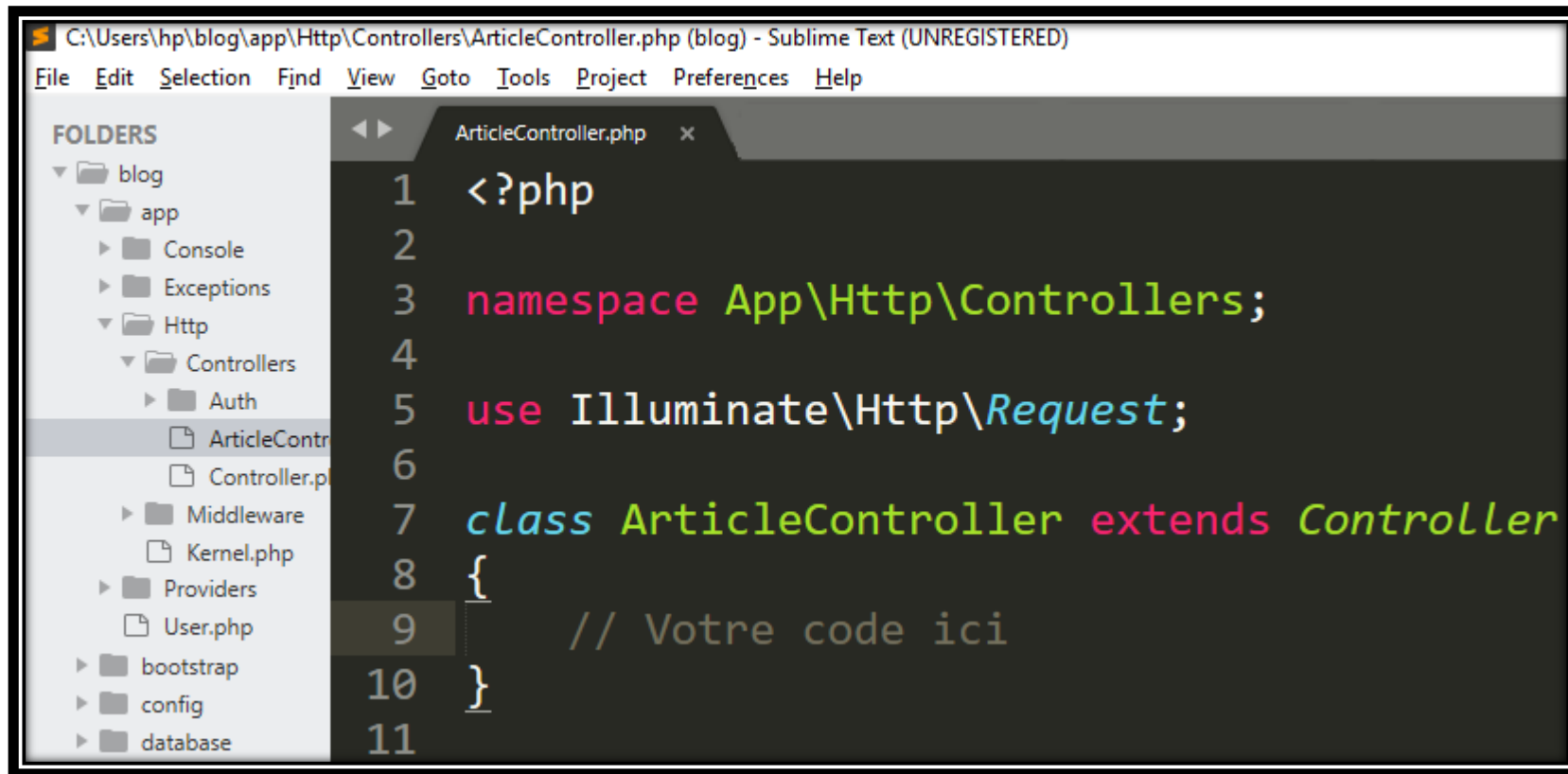
NB : Même si on passe généralement par **Artisan** pour créer un contrôleur, il est tout à fait possible de le créer manuellement (plus fastidieux quand même 😊)



Création d'un contrôleur

Exemple d'un contrôleur de base créé par **Artisan**

- Il ne reste plus qu'à écrire les méthodes de gestion des requêtes



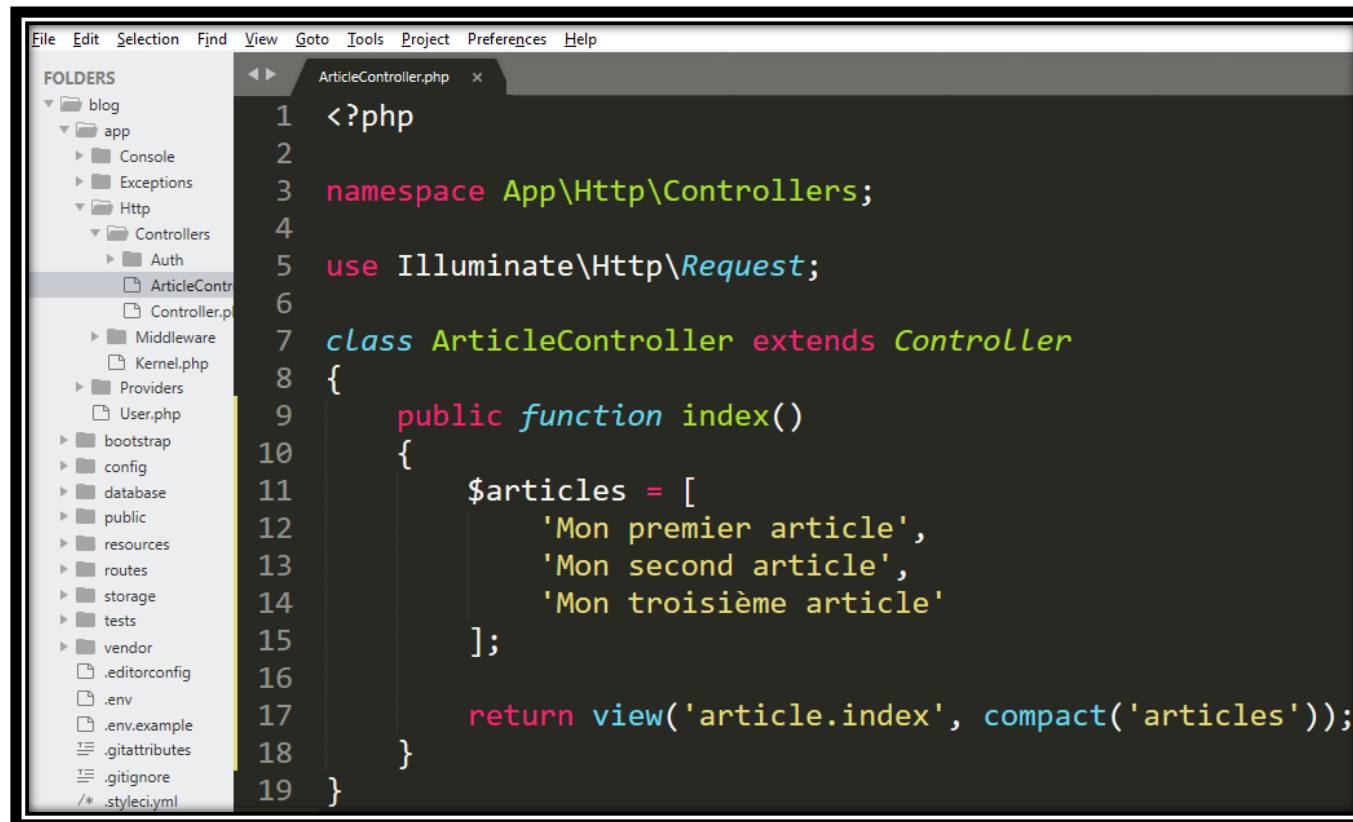
The screenshot shows the Sublime Text editor interface. On the left, the 'FOLDERS' sidebar displays the project structure: 'blog' (expanded) contains 'app' (expanded), which contains 'Console', 'Exceptions', 'Http' (expanded), 'Controllers' (expanded), 'Auth', 'ArticleContr...', 'Controller.p...', 'Middleware', 'Kernel.php', 'Providers', 'User.php', 'bootstrap', 'config', and 'database'. The main editor window shows the file 'ArticleController.php' with the following PHP code:

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ArticleController extends Controller
8 {
9     // Votre code ici
10 }
11
```

Ajout de méthodes de gestion à un contrôleur

Méthode gérant l'affichage de la liste des articles

- On peut ajouter autant de méthodes que de requêtes à gérer



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ArticleController extends Controller
8 {
9     public function index()
10    {
11        $articles = [
12            'Mon premier article',
13            'Mon second article',
14            'Mon troisième article'
15        ];
16
17        return view('article.index', compact('articles'));
18    }
19 }
```


Ajout de méthodes de gestion à un contrôleur

La vue **article/index.blade.php** pourrait ressembler à ceci

- La liste des articles (**\$articles**) lui est transmise par le contrôleur (voir slide précédent)

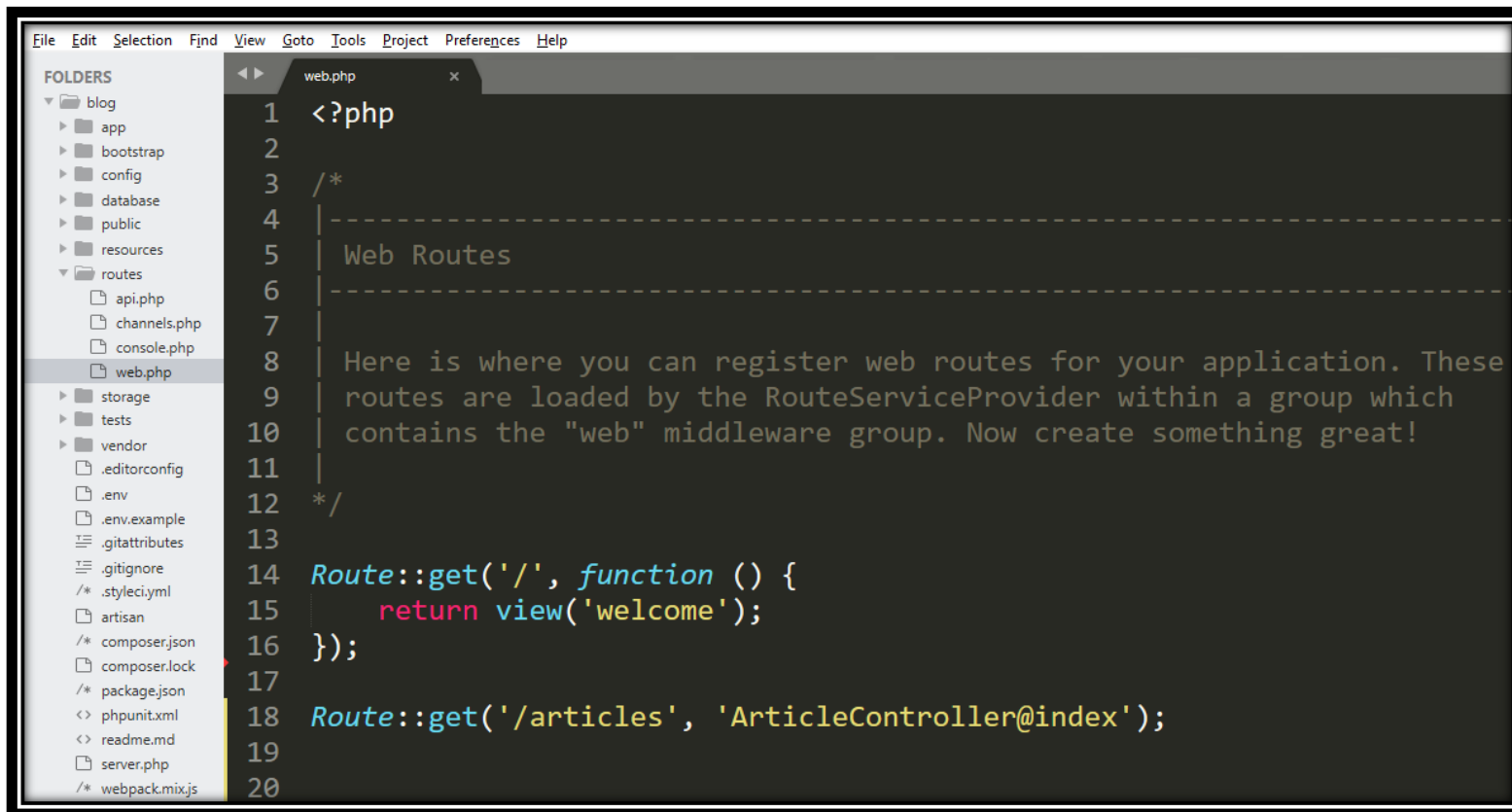


```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Liste des articles</title>
5 </head>
6 <body>
7     <ul>
8         @foreach ($articles as $article)
9             <li>{{ $article }}</li>
10        @endforeach
11    </ul>
12 </body>
13 </html>
```

Ajout de méthodes de gestion à un contrôleur

L'association de la méthode **index** du contrôleur avec une route se fait grâce au routage

- Il suffit d'éditer le fichier **routes/web.php** et d'y renseigner la nouvelle route



```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  blog
  app
  bootstrap
  config
  database
  public
  resources
  routes
    api.php
    channels.php
    console.php
    web.php
  storage
  tests
  vendor
    .editorconfig
    .env
    .env.example
    .gitattributes
    .gitignore
    .styleci.yml
    artisan
    composer.json
    composer.lock
    package.json
    phpunit.xml
    readme.md
    server.php
    webpack.mix.js
web.php
1 <?php
2
3 /*
4  -----
5  | Web Routes
6  | -----
7  |
8  | Here is where you can register web routes for your application. These
9  | routes are loaded by the RouteServiceProvider within a group which
10 | contains the "web" middleware group. Now create something great!
11 |
12 */
13
14 Route::get('/', function () {
15     return view('welcome');
16 });
17
18 Route::get('/articles', 'ArticleController@index');
19
20
```


Ajout de méthodes de gestion à un contrôleur

Méthode gérant l'affichage d'un seul article

- Il faudra ajouter la méthode correspondante au niveau du contrôleur (**ArticleController**)

```
public function show($id)
{
    // le code ici...
}
```

- Ensuite, définir une route d'accès à cette méthode

```
Route::get('/articles/{id}', 'ArticleController@show');
```

- Désormais, lorsqu'une requête correspond à l'URL de la route spécifiée, la méthode **show** de la classe **ArticleController** sera exécutée. Les paramètres de la route seront également transmis à la méthode.



Contrôleurs et espaces de noms

Lorsqu'un contrôleur ne se trouve pas directement dans le dossier de base des contrôleurs, il faudra veiller à adapter le nom dans les routes

- Le chemin adapté devra être défini relativement à `app/Http/Controllers`
- Par exemple, si `ArticleController` se trouvait dans un sous dossier `article`
 - Son chemin est alors `app/Http/Controllers/article/ArticleController.php`
 - Le nom complet de sa classe devient `App\Http\Controllers\Article\ArticleController`
- Par conséquent, la définition de la route devient:
 - `Route::get('/articles', 'Article\ArticleController@index');`



Contrôleurs à action unique

Il est possible de définir un contrôleur ne gérant qu'une seule action

- L'action sera définie sur le contrôleur par l'unique méthode `__invoke`

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ShowProfile extends Controller
{
    // Méthode d'affichage du profil d'un utilisateur
    public function __invoke($id)
    {
        return view('user.profile', ['user' => User::findOrFail($id)]);
    }
}
```

Contrôleurs à action unique

L'enregistrement d'une route pour un contrôleur à action unique ne nécessite pas la spécification de la méthode

- Le seul nom du contrôleur va suffire

```
Route::get('/users/{id}', 'ShowProfile');
```

- Il est possible de générer un contrôleur à action unique avec **Artisan**
 - Il suffit de spécifier l'option **--invokable**

php artisan make:controller ShowProfile --invokable



Contrôleurs de ressources

Il est possible de générer avec **Artisan** des contrôleurs capables de gérer toutes les opérations (**CRUD**) sur une ressource

- Par exemple, il est possible de créer un contrôleur chargé de la gestion de toutes les requêtes concernant les Produits (création, suppression, etc.)
 - Il suffit de taper la commande suivante :

php artisan make:controller ProductController --resource

- Cette commande générera un contrôleur dans **app/Http/Controllers/ProductController.php**
 - Ce contrôleur contiendra une méthode pour chacune des opérations possibles sur les produits



Contrôleurs de ressources

Il est possible de définir une route d'accès à un contrôleur de ressources

- Elle se définit au niveau du fichier de routage (`routes/web.php`)
`Route::resource('products', 'ProductController');`
- Cette simple déclaration crée plusieurs routes pour gérer diverses actions sur la ressource
 - Le contrôleur généré aura déjà des squelettes de méthodes pour chacune de ses actions
 - Des commentaires sur les verbes HTTP et les URI accompagneront la définition des méthodes générées
- Plusieurs routes de contrôleurs de ressources peuvent être déclarées en même temps
`Route::resources(['articles' => 'ArticleController', 'products' => 'ProductController']);`



Contrôleurs de ressources

Extrait d'un contrôleur de ressources généré par **Artisan**

```
class ProductController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        //
    }
}
```


Contrôleurs de ressources

Par défaut, les actions gérées par les contrôleurs de ressources sont :

- ***index*** : chargée de l'affichage de la liste des éléments
- ***create*** : chargée de l'affichage du formulaire d'ajout d'un nouvel élément
- ***store*** : chargée de l'ajout d'un élément dans le système de stockage
- ***show*** : chargée de l'affichage de l'élément spécifié
- ***edit*** : chargée de l'affichage du formulaire de modification d'un élément
- ***update*** : chargée de la mise à jour de l'élément spécifié
- ***destroy*** : chargée de la suppression de l'élément spécifié



Contrôleurs de ressources

Exemple : Actions et routes générées pour le contrôleur de ressources ProductController

Verbe HTTP	URI	Action	Nom de la Route
GET	/products	index	products.index
GET	/products/create	create	products.create
POST	/products	store	products.store
GET	/products/{product}	show	products.show
GET	/products/{product}/edit	edit	products.edit
PUT/PATCH	/products/{product}	update	products.update
DELETE	/products/{product}	destroy	products.destroy



Liens utiles

Pour aller plus loin...

- <https://laravel.com/docs/6.x/controllers>
- <https://laracasts.com/series/laravel-from-scratch-2018/episodes/6>
- <https://www.w3schools.in/laravel-tutorial/controllers/>

