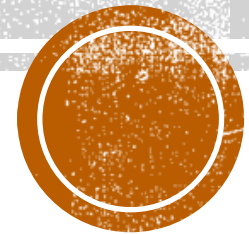




# Module: Développer en back-end

## 2. Manipulation des contrôleurs



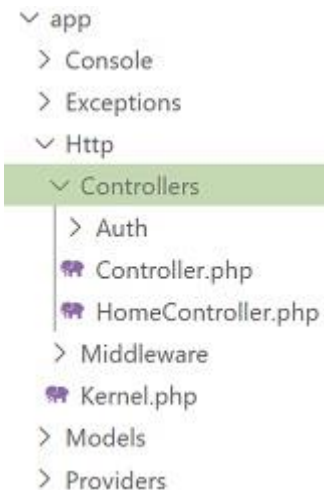
***Filière: Développement digital – option web full stack***

*Formatrice: Asmae YOUALA*

*ISTA AL ADARISSA*

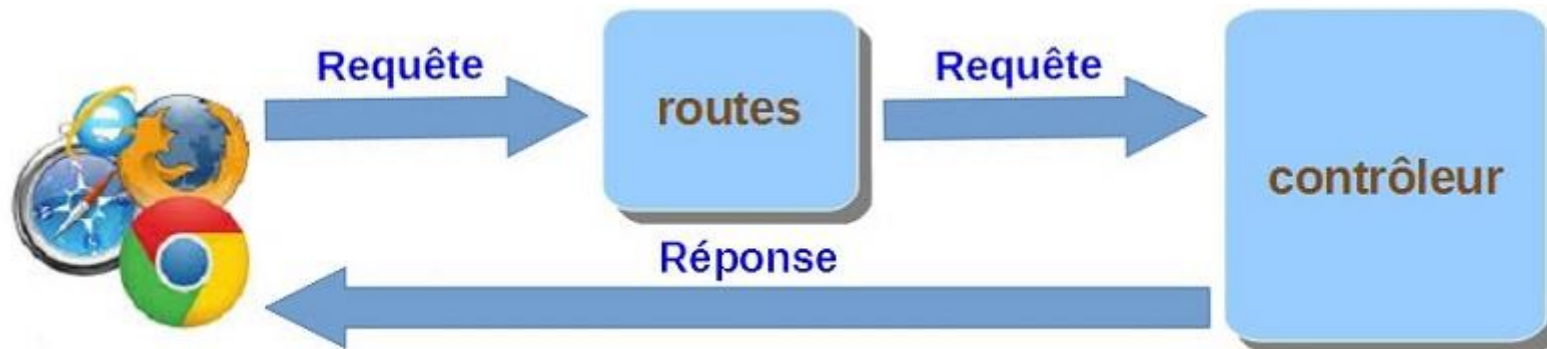
## Qu'est-ce qu'un contrôleur ?

- 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.
- Les contrôleurs sont destinés à regrouper la logique de traitement des demandes associée dans une seule classe.
- Dans votre projet Laravel, ils sont stockés dans le répertoire **app/Http/Controllers**.



## Rôle d'un contrôleur

- La tâche d'un **contrôleur** est de recevoir 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 :



## Créer un contrôleur sous Laravel

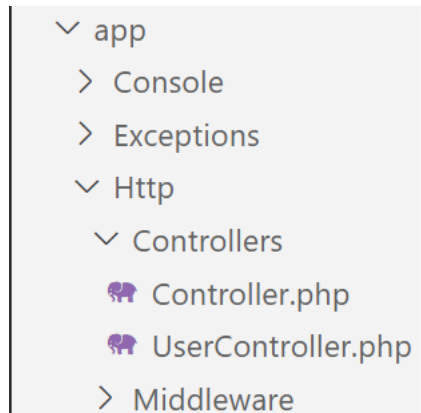
- Pour créer un contrôleur à partir de la fenêtre du terminal, ouvrez le terminal et changez le répertoire vers votre dossier racine laravel. Une fois que vous y êtes, vous pouvez exécuter l'une des commandes suivantes pour créer un contrôleur.
- **Artisan**, l'outil en ligne de commande fourni avec Laravel, permet de créer rapidement un fichier contenant la structure de base d'un contrôleur.

### Créer un simple contrôleur vierge

- Pour créer un contrôleur vous pouvez directement entrer la commande : **php artisan make:controller NomDeMonController**

#### Exemple:

**php artisan make:controller UserController**



## Créer un contrôleur sous Laravel

- Dans l'exemple précédent nous avons créer le controller **UserController**.
- Le contrôleur crée est sous le dossier **app/Http/Controllers**
- Ajoutons la méthode **index** au controlleur **UserController** :

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class UserController extends Controller
{
    //
    public function index() {
        return view("welcome");
    }
}
```

## Appel depuis la route:

- Maintenant nous devons faire le lien entre notre fichier **web.php** et notre Controller.
- Nous devons donc changer notre deuxième paramètre :

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

- A partir de Laravel 8.x, l'appel d'une action d'un contrôleur s'effectue en passant un tableau comme deuxième paramètre de la fonction; Le tableau comprend le nom de la classe contrôleur et le nom de la méthode à appeler.

## Appel depuis la route: Utilisation des paramètres

Ajoutons à notre contrôleur la méthode **get** prenant en argument un paramètre \$id:

```
class UserController extends Controller
{
    public function get($id) {
        return "page $id";
    }
}
```

Ajoutons dans le fichier **web.php** la route suivante:

```
Route::get('/get/{id}', [UserController::class, 'get']);
```

## Contrôleurs à simple action

Si une action de contrôleur est particulièrement complexe, vous trouverez peut-être pratique de dédier une classe de contrôleur entière à cette action unique. Pour ce faire, vous pouvez définir une seule méthode magique `__invoke` dans le contrôleur;

- Vous pouvez générer un contrôleur invocable en utilisant l'option `--invokable` de la commande `Artisan make:controller` :

**php artisan make:controller HomeController --invokable**

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller
{
    /**
     * Handle the incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function __invoke(Request $request)
    {
        return "Bienvenue à la page home";
    }
}
```



## Contrôleurs à simple action

Pour appeler ce type de contrôleur via la route, il suffit d'indiquer la classe du contrôleur sans évoquer une méthode spécifique:

```
Route::get('/home', HomeController::class);
```

## Contrôleur de ressources

- Les contrôleurs de ressources Laravel fournissent les routes CRUD au contrôleur en une seule ligne de code.
- Un contrôleur de ressources est utilisé pour créer un contrôleur qui gère toutes les requêtes http stockées par votre application (get, post, update, delete ...).
- Pour commencer, nous pouvons utiliser l' option **make:controller** de la commande Artisan **--resource** pour créer rapidement un contrôleur pour gérer ces actions :

```
php artisan make:controller PostController --resource
```

- Cette commande générera un contrôleur sous **app/Http/Controllers/PostController.php**. Le contrôleur contiendra une méthode pour chacune des opérations de ressources disponibles.

## Contrôleur de ressources

La classe PostController créée a l'allure suivante:

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PostController 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)
    {
        //
    }

    /**
     * Display the specified resource.
     *
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }

    /**
     * Show the form for editing the specified
    resource.
     *
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        //
    }
}
```

```
    /**
     * Update the specified resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request,
    $id)
    {
        //
    }

    /**
     * Remove the specified resource from
    storage.
     *
     * @param int $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        //
    }
}
```

## Contrôleur de ressources

- La méthode **resource** () de la classe **Route** est une fonction statique comme la méthode **get** () qui donne accès aux différentes routes que nous pouvons utiliser dans un contrôleur.
- Syntaxe de la méthode resource() :

**Route::resource("/posts", PostController::class);**

- Cette déclaration de route unique crée plusieurs routes pour gérer diverses actions sur la ressource.
- Le contrôleur généré contient déjà les méthodes souhaitées pour chacune de ces actions.
- N'oubliez pas que vous pouvez toujours obtenir un aperçu rapide des routes de votre application en exécutant la commande Artisan **route:list**.

En exécutant la commande ci-après, on obtient:

**php artisan route:list --except-vendor**

GET HEAD	posts .....	posts.index › PostController@index
POST	posts .....	posts.store › PostController@store
GET HEAD	posts/create .....	posts.create › PostController@create
GET HEAD	posts/{post} .....	posts.show › PostController@show
PUT PATCH	posts/{post} .....	posts.update › PostController@update
DELETE	posts/{post} .....	posts.destroy › PostController@destroy
GET HEAD	posts/{post}/edit .....	posts.edit › PostController@edit

## Contrôleur de ressources

- Vous pouvez même enregistrer plusieurs contrôleurs de ressources à la fois en passant un tableau à la méthode **resources** :

```
Route::resources([  
  'photos' => PhotoController::class, 'posts' => PostController::class,  
]);
```