⬅Revenir à la théorie.

Les Contrôleurs

- 1. Présentation
- 2. Création d'un contrôleur
- 3. Création d'une méthode publique
- 4. Création d'une route vers le contrôleur
- 5. Contrôleur avec paramètres
    - 5.1 Premier exemple
    - 5.2 Second exemple
        - 5.2.1 Utilisation de la Liste
        - 5.2.2 Utilisation de la recherche sur le nom
        - 5.2.3 Utilisation de la recherche sur le prénom
        - 5.2.4 Utilisation de la recherche sur le nom et/ou prénom
- 6. Route nommée vers un contrôleur
- 7. Création d'un contrôleur avec des méthodes prédéfinies

1. Présentation

Un contrôleur est une classe qui contient des méthodes qui seront utilisées dans notre application. Elles y contiennent la logique des actions demandées par l'utilisateur.

Chaque méthode représente une action. Et ces actions ont généralement une route dédiée.

La tâche principale d'un contrôleur est de recevoir une requête venant d'une route et de définir la réponse à apporter.
Cela peut se résumer de cette manière:

1. L'utilisateur fait une requête.
2. Laravel sélectionne la route.
3. La route envoie une requête au contrôleur.
4. Le contrôleur fournit une réponse.

Le contrôleur peut évidemment appeler une vue.

2. Création d'un contrôleur

Ici, je vais vous montrer comment créer un contrôleur via l'outil artisan. Vous pourriez le faire à la main mais autant utiliser artisan.

On ouvre une fenêtre de commandes dans le répertoire de l'application.
Si on veut créer un contrôleur nommé 'PersonController', on tapera:

```
php artisan make:controller PersonController
```

Artisan va créer le contrôleur dans le répertoire app/Http/Controllers. Vous verrez qu'un fichier PersonController.php a bien été créé.

Par convention, on utilise du Camel Case pour nommer un controller. Le nom se terminer par Controller: DataController, ChildController, DeviceController, etc.

3. Création d'une méthode publique

On va y ajouter une méthode publique nommée 'hi' à notre contrôleur:

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PersonController extends Controller
{
    public function hi(){
        return "Bonjour cher visiteur !";
    }
}
```

4. Création d'une route vers le contrôleur

Maintenant créons une route 'Hi' dans le fichier web.php

```
Route::get('Hi', [PersonController::class, 'hi']);
```

Si nous testons notre route 'Hi', nous allons avoir une erreur:

```
Target class [PersonController] does not exist.
```

Il suffit d'ajouter au début de notre fichier web.php la ligne suivante:

```
use App\Http\Controllers\PersonController;
```

On pourra dès lors utiliser la classe PersonController dans web.php

5. Contrôleur avec paramètres

5.1 Premier exemple

On va passer les paramètres de notre route à notre contrôleur. En l'occurence l'identité de l'utilisateur.
On ajoutera une contrainte de n'accepter que des caractères alphabétiques.

```
Route::get('users/Hi/{name}', [PersonController::class, 'hi'])->where('name','[A-z]+');
```

Dans le contrôleur PersonController

```
class PersonController extends Controller
{
    public function hi($name){
        return view('Hi')->with('name', $name);
    }
}
```

Dans la view Hi.blade.php

```
Bonjour cher {{ $name }} !
```

Utilisation: localhost:8000/users/Hi/Johnny
Résultat: Bonjour cher Johnny !

5.2 Second exemple

Dans cet exemple, nous allons avoir un contrôleur qui va permettre de saluer un utilisateur, afficher la liste des utilisateurs, rechercher des utilisateurs.

Le contrôleur redirigera vers la vue Results.

Comme nous verrons plus tard l'utilisation des bases de données, nous utiliserons un tableau de users dans notre contrôleur qui contiendra tous nos users. Nous utiliserons ce tableau pour rechercher des utilisateurs ou afficher la liste des utilisateurs.

Dans web.php on va ajouter quatre routes:

```
Route::get('users/List', [PersonController::class, 'getUsersList']);
Route::get('users/SearchByName/{name}', [PersonController::class, 'getUsersByName']);
Route::get('users/SearchByFirstname/{firstname}', [PersonController::class, 'getUsersByFi
Route::get('users/SearchBySomething/{thing}', [PersonController::class, 'getUsersBySometh
```

Dans le contrôleur PersonController:

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PersonController extends Controller
{
    private array $users = [
        [
            "name" => "Piette",
            "firstname" => "Johnny"
        ],
        [
            "name" => "Piette",
            "firstname" => "Gabriel"
        ],
        [
            "name" => "Dupont",
            "firstname" => "Philip"
        ],
        [
            "name" => "Colin",
            "firstname" => "Stéphane"
        ],
        [
            "name" => "Jacques",
            "firstname" => "Véronique"
        ],
        [
            "name" => "Larock",
            "firstname" => "Jacques"
        ]
    ];

    public function hi($name)
    {
        return view("Hi", ['name' => $name]);
    }

    public function getUsersList()
```

```
    {
        return view('results', ['users' => $this->users]);
    }

    public function getUsersByName($name)
    {
        $array = $this->Search('name', $name);
        return view('results', ['users' => $array]);
    }

    public function getUsersByFirstname($firstname)
    {
        return view('results', ['users' => $this->Search('firstname', $firstname)]);
    }

    public function getUsersBySomething($thing)
    {
        $arrayNames = $this->Search('name', $thing);
        $arrayFirstNames = $this->Search('firstname', $thing);
        $array = array_merge($arrayNames, $arrayFirstNames);
        return view('results', ['users' => $array]);
    }

    private function search($champ, $valeur): array
    {
        $array = [];
        foreach ($this->users as $user) {
            if (strtolower($user["$champ"]) === strtolower($valeur)) {
                array_push($array, $user);
            }
        }
        return $array;
    }
}
```

Dans la view Results.blade.php:

```
<h1>Résultat</h1>

@forelse ($users as $user)
- Nom: {{ $user['name'] }} <br/>
- Prénom: {{ $user['firstname'] }}
<hr/>
@empty
Aucun utilisateur n'a été trouvé.
@endforelse
```

5.2.1 Utilisation de la Liste

URL: http://localhost:8000/users/List
Résultat:

```
Résultat
- Nom: Piette
- Prénom: Johnny
- Nom: Piette
- Prénom: Gabriel
- Nom: Dupont
- Prénom: Philip
- Nom: Colin
- Prénom: Stéphane
- Nom: Jacques
- Prénom: Véronique
- Nom: Larock
- Prénom: Jacques
```

5.2.2 Utilisation de la recherche sur le nom

URL: http://localhost:8000/users/SearchByName/Piette
Résultat:

```
Résultat
- Nom: Piette
- Prénom: Johnny
- Nom: Piette
- Prénom: Gabriel
```

5.2.3 Utilisation de la recherche sur le prénom

URL: http://localhost:8000/users/SearchByFirstname/Jacques
Résultat:

```
Résultat
- Nom: Larock
- Prénom: Jacques
```

5.2.4 Utilisation de la recherche sur le nom et/ou prénom

URL: http://localhost:8000/users/Search/Jacques
Résultat:

```
- Nom: Jacques
- Prénom: Véronique
- Nom: Larock
- Prénom: Jacques
```

6. Route nommée vers un contrôleur

Rien de nouveau, on a déjà vu qu'on sait nommer une route. On peut donc nommer une route qui pointe vers un contrôleur:

```
Route::get('bonjour/{n}', [PersonController::class, 'hi'])->where('n','[A-z]+')->name("He
```

7. Création d'un contrôleur avec des méthodes prédéfinies

Il est possible de créer automatiquement avec artisan un contrôleur avec des méthodes classiques: consultation, mise à jour, ajout, etc. Cela permet de gagner du temps et donne une cohérence dans l'utilisation des noms de méthode.

Le corps de ces méthodes sont vides mais la structure est créée pour vous.

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

Résultat:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class TutuController 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)
    {
        //
    }
```

```php
    /**
     * 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)
    {
        //
    }
}
```

⬅Revenir à la théorie.
