Manipulation de contrôleur de ressource :

1. Contrôleur invocable

- 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 **__invoke** dans le contrôleur :

```
<?php
namespace App\Http\Controllers;

use App\Models\Student;
use Illuminate\Http\Request;

class StudentController extends Controller
{

                public function  __invoke($id){
                 $user=Student::findOrFail($id);

                 return $user->name;


                        }
}
```

- Lors de l'enregistrement d'itinéraires pour des contrôleurs à action unique, vous n'avez pas besoin de spécifier une méthode de contrôleur. Au lieu de cela, vous pouvez simplement transmettre le nom du contrôleur au routeur :

```
use App\Http\Controllers\Student;
Route::get('show/{id}', StudentController::class);
```

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

```
php artisan make:controller ProvisionServer --invokable
```

2. Contrôleur Middleware

- Un middleware peut être affecté aux routes du contrôleur dans vos fichiers de routes :

```
Route::get('show/{id}', [StudentController::class,'show'])->middleware('terminate');
```

- Vous trouverez peut-être pratique de spécifier un middleware dans le constructeur de votre contrôleur. En utilisant la méthode middleware dans le constructeur de votre contrôleur, vous pouvez affecter un middleware aux actions du contrôleur :

```
class StudentController extends Controller

{
    public function __construct()
{
//Appliquer ce middleware à toutes les actions du contrôleur
$this->middleware('terminate');
//Affecter le middleware à toutes les actions sauf l'action 'show'
$this->middleware('terminate')->except('show');
//Affecter le middleware seulement à l'action 'create
$this->middleware('terminate')->only('create');

}
}
```

Les contrôleurs vous permettent également d'enregistrer un middleware à l'aide d'une fermeture. Cela fournit un moyen pratique de définir un middleware en ligne pour un seul contrôleur sans définir une classe complète de middleware :

```
$this->middleware(function ($request, $next)
{
 return $next($request);
 });
```

3. Contrôleur de ressources

La route de ressources de Laravel permet aux routes classiques "CRUD" pour les contrôleurs d'avoir une seule ligne de code. Ceci peut être créé rapidement en utilisant la commande make : controller (commande Artisan) quelque chose comme ceci".

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

Le code ci-dessus produira un contrôleur dans l'emplacement **app/Http/Controllers/** avec le nom de fichier **StudentController.php** qui contiendra une méthode pour toutes les tâches disponibles des ressources.

Ensuite, vous pouvez enregistrer une route de ressources qui pointe vers le contrôleur :

```
use App\Http\Controllers\StudentController;
Route::resource('students', studentController::class);
```

Cette déclaration de route unique crée plusieurs routes pour gérer diverses actions sur la ressource. Le contrôleur généré aura déjà des méthodes 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

php artisan route:list --name=students

```
PS C:\xampp\htdocs\tp1> php artisan route:list --name=students

  GET|HEAD  students ..... students.index › StudentController@index
  POST      students ..... students.store › StudentController@store
  GET|HEAD  students/create ..... students.create › StudentController@create
  DELETE    students/{student} ..... students.destroy › StudentController@destroy
  GET|HEAD  students/{student} ..... students.show › StudentController@show
  PUT       students/{student} ..... students.update › StudentController@update
  GET|HEAD  students/{student}/edit ..... students.edit › StudentController@edit
```

Actions gérées par le contrôleur de ressources

Verb	URI	Action	Route Name
GET	/students	index	students.index
GET	/students/create	create	students.create
POST	/students	store	students.store
GET	/students/{student}	show	students.show
GET	/students/{student}/edit	edit	students.edit
PUT	/students/{student}	update	students.update
DELETE	/students/{student}	destroy	students.destroy

Personnalisation du comportement du modèle manquant

En règle générale, une réponse HTTP 404 est générée si un modèle de ressource implicitement lié n'est pas trouvé. Cependant, vous pouvez personnaliser ce comportement en appelant la méthode **missing** lors de la définition de votre route de ressource. La méthode **missing** accepte une fermeture qui sera invoquée si un modèle lié implicitement ne peut être trouvé pour aucune des routes de la ressource :

```
Route::resource('students', StudentController::class)->missing(function
(Request $request) {
  return redirect()->route('student.index');
  });
```

Les développeurs Laravel ont également la liberté d'enregistrer plusieurs contrôleurs de ressources à la fois en passant un tableau à une méthode de ressource, quelque chose comme ceci :

```
Route::resources([
    'password' => 'PasswordController',
    'picture' => 'DpController'
]
);
```

Travail à faire :

Refaire le TP de la séance 5 en utilisant un contrôleur ressource. Remplacez les sept routes que nous avons définies manuellement par la route suivante.

```
Route::resource('students', StudentController::class)
```