MIGRATION

Une migration dans Laravel, c'est un fichier PHP qui décrit comment créer ou modifier une table dans la base de données.

À quoi ça sert ?

- Créer des tables (ex : users, courses…)
- Modifier la structure (ajouter/supprimer des colonnes)
- Garder un historique des changements (comme Git mais pour la base de données)
- Partager facilement la structure de la base avec toute l'équipe

Comment ça marche ?

- Tu écris la structure de ta table dans un fichier de migration (ex : quels champs, quels types, quelles relations…)
- Tu exécutes la commande `php artisan migrate`
- Laravel lit tous les fichiers de migration et applique les changements à la base de données

Exemple simple

php

Apply

```
public function up()

{

    Schema::create('courses', function (Blueprint $table) {

        $table->id();

        $table->string('title');

        $table->timestamps();
```

```
    });


}
```

Ici, on dit à Laravel : “Crée une table courses avec un id, un titre, et les dates de création/modification.”

Avantages

- Tu peux versionner ta base de données comme ton code
- Tu peux facilement revenir en arrière (`php artisan migrate:rollback`)
- Tu peux déployer la même structure sur plusieurs serveurs sans erreur

En résumé :Une migration, c’est un script qui décrit la structure de ta base de données, versionné et exécutable à tout moment pour créer ou modifier tes tables de façon propre et collaborative.Veux-tu voir comment exécuter une migration ou comment la modifier ?

POLICY

Qu’est-ce qu’une Policy dans Laravel ?

Une Policy (ou “politique” en français) est une classe qui centralise les règles d’autorisation pour un modèle donné (par exemple, qui a le droit de modifier ou supprimer un cours).But :Sécuriser ton application en s’assurant qu’un utilisateur ne peut agir que sur les ressources auxquelles il a droit.

Exemple concret

Imaginons que tu as plusieurs formateurs.Tu veux que :

- Un formateur ne puisse modifier/supprimer QUE ses propres cours
- Un admin puisse tout faire

Au lieu de mettre la logique d’autorisation partout dans tes contrôleurs, tu la mets dans une Policy.

Comment ça marche ?

1. Création d’une Policy

2. bash
3. Apply
4. Run
5. ```
   php artisan make:policy CoursePolicy --model=Course
   ```

Cela crée une classe `CoursePolicy` dans `app/Policies/`.

1. Exemple de méthode dans la Policy
2. php
3. Apply
4. ```
   public function update(User $user, Course $course)
   ```
5. ```
   {
   ```
6. ```
       // Seul le formateur propriétaire ou un admin peut modifier
   ```
7. ```
       return $user->id === $course->teacher_id || $user->role === 'admin';
   ```

Qu’est-ce qu’une Policy dans Laravel ?

Une Policy (ou “politique” en français) est une classe qui centralise les règles d’autorisation pour un modèle donné (par exemple, qui a le droit de modifier ou supprimer un cours).But :Sécuriser ton application en s’assurant qu’un utilisateur ne peut agir que sur les ressources auxquelles il a droit.

Exemple concret

Imaginons que tu as plusieurs formateurs.Tu veux que :

- Un formateur ne puisse modifier/supprimer QUE ses propres cours
- Un admin puisse tout faire

Au lieu de mettre la logique d’autorisation partout dans tes contrôleurs, tu la mets dans une Policy.

Comment ça marche ?

1. Création d’une Policy
2. bash
3. Apply
4. Run
5. ```
   php artisan make:policy CoursePolicy --model=Course
   ```

Cela crée une classe `CoursePolicy` dans `app/Policies/`.

1. Exemple de méthode dans la Policy
2. php
3. Apply
4. ```
   public function update(User $user, Course $course)
   ```
5. ```
   {
   ```
6. ```
       // Seul le formateur propriétaire ou un admin peut modifier
   ```
7. ```
       return $user->id === $course->teacher_id || $user->role === 'admin';
   ```
8. ```
   }
   ```
9. Utilisation dans le contrôleur
10. php
11. Apply
12. ```
    $this->authorize('update', $course);
    ```

Laravel va automatiquement appeler la méthode `update` de la Policy.

Avantages

- Centralisation : toutes les règles sont au même endroit
- Lisibilité : ton contrôleur reste propre
- Sécurité : impossible de “sauter” une vérification

En résumé :Une Policy, c’est une classe qui dit “qui a le droit de faire quoi” sur un modèle (ici, Course).Laravel s’en sert pour vérifier les droits avant chaque action sensible.

1. Générer la Policy

Dans ton terminal, exécute :

bash

Apply

Run

```
php artisan make:policy CoursePolicy --model=Course
```

Cela va créer le fichier :`app/Policies/CoursePolicy.php`

---

2. Implémenter les règles dans la Policy

Ouvre ce fichier et remplace le contenu par :

php

Apply

```php
<?php

namespace App\Policies;

use App\Models\Course;

use App\Models\User;

class CoursePolicy

{

    /**

     * Détermine si l'utilisateur peut voir le cours.

     */

    public function view(User $user, Course $course)

    {

        // Le formateur propriétaire ou un admin peut voir

        return $user->id === $course->teacher_id || $user->role === 'admin';

    }

    /**

     * Détermine si l'utilisateur peut modifier le cours.

     */

    public function update(User $user, Course $course)

    {

        // Seul le formateur propriétaire ou un admin peut modifier

        return $user->id === $course->teacher_id || $user->role === 'admin';

    }

    /**

     * Détermine si l'utilisateur peut supprimer le cours.
```

```
     */
    public function delete(User $user, Course $course)
    {
        // Seul le formateur propriétaire ou un admin peut supprimer
        return $user->id === $course->teacher_id || $user->role === 'admin';
    }
}
```

3. Enregistrer la Policy

Ouvre le fichier :`app/Providers/AuthServiceProvider.php`Dans la propriété `$policies`, ajoute la ligne suivante :

php

Apply

```
protected $policies = [
    \App\Models\Course::class => \App\Policies\CoursePolicy::class,
    // ... autres policies si besoin
];
```

Exemple complet :

php

Apply

```
protected $policies = [
    \App\Models\Course::class => \App\Policies\CoursePolicy::class,
];
```

---

4. Utilisation dans le contrôleur

Dans ton `CourseController`, tu peux maintenant utiliser :

php

Apply

```
$this->authorize('update', $course);

$this->authorize('delete', $course);

$this->authorize('view', $course);
```

Laravel vérifiera automatiquement les règles définies dans ta Policy.

Résumé :

- On crée la Policy
- On définit qui a le droit de voir, modifier, supprimer un cours
- On enregistre la Policy dans `AuthServiceProvider`
- On utilise `$this->authorize()` dans le contrôleur pour sécuriser les actions