Laboratoire : Création d'une Application avec Validation via FormRequest

Objectif

Créer une application web où un utilisateur peut soumettre un formulaire contenant son nom, son âge et son email. L'application utilisera une classe `FormRequest` personnalisée pour valider les données avant d'afficher un message de confirmation.

Étapes du Laboratoire

Étape 1 : Installation de Laravel

1. Installez Laravel via Composer si ce n'est pas déjà fait :

Créer projet;

2. Lancez le serveur de développement :
bash
Copier

```
php artisan serve
```

Accédez à `http://localhost:8000` pour vérifier que l'application fonctionne.

Étape 2 : Création des Routes

Ouvrez le fichier `routes/web.php` et ajoutez les routes nécessaires :
php
Copier

```
use App\Http\Controllers\FormController;

// Route pour afficher le formulaire
Route::get('/form', [FormController::class, 'showForm'])->name('form.show');

// Route pour traiter la soumission du formulaire
Route::post('/form/submit', [FormController::class, 'handleForm'])->name('form.submit');
```

Étape 3 : Création du Contrôleur

1. Créez un contrôleur nommé `FormController` :
   bash
   Copier
   ```
   php artisan make:controller FormController
   ```
2. Ouvrez le fichier `app/Http/Controllers/FormController.php` et ajoutez les méthodes suivantes :
   php
   Copier
   ```
   namespace App\Http\Controllers;

   use App\Http\Requests\FormRequest;

   class FormController extends Controller
   {
       // Afficher le formulaire
       public function showForm()
       {
           return view('form');
       }

       // Traiter la soumission du formulaire
       public function handleForm(FormRequest $request)
       {
           // Les données sont automatiquement validées grâce à
   FormRequest

           // Récupérer les données validées
           $validated = $request->validated();

           // Rediriger vers une page de confirmation
           return view('success', [
               'name' => $validated['name'],
               'age' => $validated['age'],
               'email' => $validated['email'],
           ]);
       }
   }
   ```

Étape 4 : Création de la Classe FormRequest

1. Créez une classe `FormRequest` personnalisée :
   bash
   Copier
   ```
   php artisan make:request FormRequest
   ```
2. Ouvrez le fichier `app/Http/Requests/FormRequest.php` et configurez les règles de validation et les messages d'erreur :
   php
   Copier
   ```
   namespace App\Http\Requests;

   use Illuminate\Foundation\Http\FormRequest;

   class FormRequest extends FormRequest
   {
       // Définir les autorisations (toujours true ici)
       public function authorize()
       {
           return true; // Autoriser toutes les requêtes
       }

       // Définir les règles de validation
       public function rules()
       {
           return [
               'name' => 'required|string|max:255',
               'age' => 'required|integer|min:1|max:120',
               'email' => 'required|email|max:255',
           ];
       }

       // (Optionnel) Personnaliser les messages d'erreur
       public function messages()
       {
           return [
               'name.required' => 'Le champ Nom est requis.',
               'age.min' => 'L\'âge doit être supérieur à 0.',
               'email.email' => 'Veuillez entrer une adresse
   email valide.',
           ];
       }
   }
   ```

Étape 5 : Création des Vues

1. **Vue du Formulaire (`resources/views/form.blade.php`)**
   Créez une nouvelle vue pour le formulaire :
   html
   Copier

```
<!DOCTYPE html>
<html>
<head>
    <title>Formulaire</title>
</head>
<body>
    <h1>Remplissez le formulaire</h1>

    <!-- Afficher les messages d'erreur -->
    @if ($errors->any())
        <div style="color: red;">
            <ul>
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
    @endif

    <form action="{{ route('form.submit') }}" method="POST">
        @csrf <!-- Token CSRF pour la sécurité -->

        <label for="name">Nom :</label>
        <input type="text" name="name" id="name" value="{{
old('name') }}" required><br><br>

        <label for="age">Âge :</label>
        <input type="number" name="age" id="age" value="{{
old('age') }}" required><br><br>

        <label for="email">Email :</label>
        <input type="email" name="email" id="email" value="{{
old('email') }}" required><br><br>

        <button type="submit">Soumettre</button>
    </form>
</body>
</html>
```

2. **Vue de Confirmation (`resources/views/success.blade.php`)**
   Créez une vue pour afficher les résultats du formulaire :
   html
   Copier

```
<!DOCTYPE html>
<html>
<head>
    <title>Résultats</title>
</head>
<body>
    <h1>Merci pour votre soumission !</h1>
    <p>Nom : {{ $name }}</p>
    <p>Âge : {{ $age }}</p>
    <p>Email : {{ $email }}</p>

    <a href="{{ route('form.show') }}">Retour au
formulaire</a>
</body>
</html>
```

Étape 6 : Tester l'Application

1. Accédez à `http://localhost:8000/form` dans votre navigateur.
2. Remplissez le formulaire avec des données valides (nom, âge et email) et soumettez-le.
3. Vous serez redirigé vers une page de confirmation affichant les données soumises.
4. Essayez de soumettre le formulaire avec des données invalides (par exemple, un âge négatif ou un email incorrect). Vous verrez des messages d'erreur correspondants.

Explications des Composants

1. **Routes** :
- Les routes définissent les points d'entrée de l'application et associent chaque URL à une méthode spécifique du contrôleur.
2. **Contrôleur** :
- Le contrôleur gère les requêtes entrantes et utilise la classe `FormRequest` pour valider les données du formulaire.
- La méthode `handleForm` reçoit directement une instance de `FormRequest`, ce qui garantit que les données sont validées avant d'atteindre le contrôleur.
3. **Classe FormRequest** :

- La classe `FormRequest` centralise la logique de validation et permet de personnaliser facilement les règles et les messages d'erreur.
- Elle est spécifiquement conçue pour encapsuler la validation des formulaires.

4. **Vues** :

- `form.blade.php` : Affiche le formulaire et affiche les messages d'erreur en cas de validation échouée.
- `success.blade.php` : Affiche les données validées après soumission réussie.

Code Final

Contrôleur (`FormController.php`)

php
Copier

```
namespace App\Http\Controllers;

use App\Http\Requests\FormRequest;

class FormController extends Controller
{
    public function showForm()
    {
        return view('form');
    }

    public function handleForm(FormRequest $request)
    {
        // Les données sont automatiquement validées grâce à
FormRequest
        $validated = $request->validated();

        return view('success', [
            'name' => $validated['name'],
            'age' => $validated['age'],
            'email' => $validated['email'],
        ]);
    }
}
```

Classe FormRequest (`FormRequest.php`)

php
Copier

```
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class FormRequest extends FormRequest
{
    public function authorize()
    {
        return true; // Autoriser toutes les requêtes
    }

    public function rules()
    {
        return [
            'name' => 'required|string|max:255',
            'age' => 'required|integer|min:1|max:120',
            'email' => 'required|email|max:255',
        ];
    }

    public function messages()
    {
        return [
            'name.required' => 'Le champ Nom est requis.',
            'age.min' => 'L\'âge doit être supérieur à 0.',
            'email.email' => 'Veuillez entrer une adresse
email valide.',
        ];
    }
}
```

Vue du Formulaire (`form.blade.php`)

html
Copier

```
<!DOCTYPE html>
<html>
<head>
    <title>Formulaire</title>
</head>
<body>
    <h1>Remplissez le formulaire</h1>

    <!-- Afficher les messages d'erreur -->
    @if ($errors->any())
        <div style="color: red;">
            <ul>
                @foreach ($errors->all() as $error)
```

```
                    <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
    @endif

    <form action="{{ route('form.submit') }}" method="POST">
        @csrf

        <label for="name">Nom :</label>
        <input type="text" name="name" id="name" value="{{
old('name') }}" required><br><br>

        <label for="age">Âge :</label>
        <input type="number" name="age" id="age" value="{{
old('age') }}" required><br><br>

        <label for="email">Email :</label>
        <input type="email" name="email" id="email" value="{{
old('email') }}" required><br><br>

        <button type="submit">Soumettre</button>
    </form>
</body>
</html>
```

Vue de Confirmation (`success.blade.php`)

html
Copier

```
<!DOCTYPE html>
<html>
<head>
    <title>Résultats</title>
</head>
<body>
    <h1>Merci pour votre soumission !</h1>
    <p>Nom : {{ $name }}</p>
    <p>Âge : {{ $age }}</p>
    <p>Email : {{ $email }}</p>

    <a href="{{ route('form.show') }}">Retour au
formulaire</a>
</body>
</html>
```

Avantages de l'Utilisation de FormRequest

1. **Centralisation de la Validation** :
- Toutes les règles de validation sont regroupées dans une classe dédiée, rendant le code plus propre et maintenable.
2. **Réutilisabilité** :
- La classe `FormRequest` peut être réutilisée pour d'autres formulaires similaires.
3. **Personnalisation** :
- Vous pouvez facilement personnaliser les messages d'erreur et ajouter des règles complexes sans encombrer le contrôleur.

Améliorations Possibles

1. **Ajout de Règles Avancées** :
- Ajoutez des règles comme `unique` pour vérifier que l'email n'est pas déjà utilisé.
2. **Stylisation** :
- Appliquez des styles CSS pour rendre l'interface utilisateur plus attrayante.
3. **Internationalisation** :
- Utilisez les fichiers de traduction pour gérer les messages d'erreur dans plusieurs langues.

Conclusion

Ce laboratoire vous a permis de créer une application Laravel utilisant une **classe FormRequest** pour valider les données soumises via un formulaire. Cette approche est recommandée pour les applications plus complexes car elle sépare la logique de validation du contrôleur, rendant le code plus modulaire et maintenable. Vous pouvez maintenant explorer davantage Laravel en intégrant des fonctionnalités avancées comme les bases de données, les sessions ou l'authentification.