Cours Laravel 6 – les données – les ressources (2/2)

laravel.sillo.org/cours-laravel-6-les-donnees-les-ressources-2-2/

bestmomo

September 1, 2019

Dans cet article nous allons terminer de coder la ressource que nous avons commencée dans le précédent. Il nous reste à voir comment créer et modifier un film.

Créer un film

Les routes

Pour la création d'un film on va avoir deux routes :

- pour afficher le formulaire de création
- pour soumettre le formulaire

```
POST       | films             | films.store    | App\Http\Controllers\FilmController@store    | web
GET|HEAD   | films/create      | films.create   | App\Http\Controllers\FilmController@create   | web
```

Le contrôleur

Dans le contrôleur ce sont les méthodes **create** et **store** qui sont concernées. On va donc les coder :

```
use App\Http\Requests\Film as FilmRequest;

...

public function create()
{
   return view('create');
}

public function store(FilmRequest $filmRequest)
{
   Film::create($filmRequest->all());

   return redirect()->route('films.index')->with('info', 'Le film a bien été créé');
}
```

La vue index

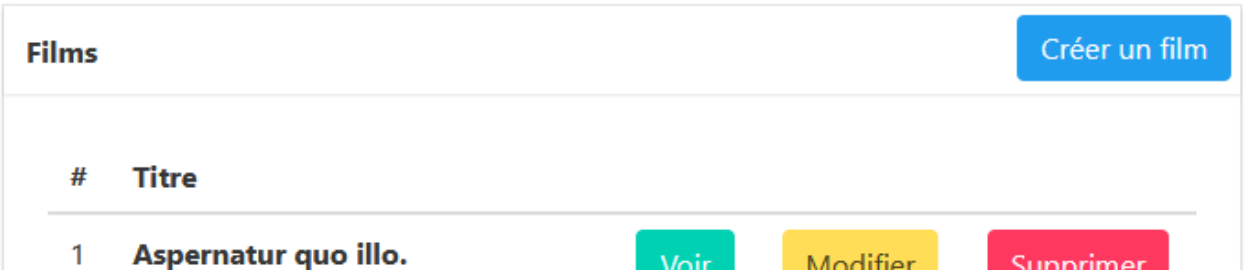
On a déjà codé la vue **index** mais on n'a pas prévu de bouton pour créer un film. Alors on va arranger ça (je remets tout le code pour que ce soit plus simple) :

```
@extends('template')
```

```
@section('css')
    <style>
        .card-footer {
            justify-content: center;
            align-items: center;
            padding: 0.4em;
        }
        .is-info {
            margin: 0.3em;
        }
    </style>
@endsection

@section('content')
    @if(session()->has('info'))
        <div class="notification is-success">
            {{ session('info') }}
        </div>
    @endif
    <div class="card">
        <header class="card-header">
            <p class="card-header-title">Films</p>
            <a class="button is-info" href="{{ route('films.create') }}">Créer un film</a>
        </header>
        <div class="card-content">
            <div class="content">
                <table class="table is-hoverable">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>Titre</th>
                            <th></th>
                            <th></th>
                            <th></th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach($films as $film)
                            <tr>
                                <td>{{ $film->id }}</td>
                                <td><strong>{{ $film->title }}</strong></td>
                                <td><a class="button is-primary" href="{{ route('film.show', $film->id)
}}">Voir</a></td>
                                <td><a class="button is-warning" href="{{ route('film.edit', $film->id)
}}">Modifier</a></td>
                                <td>
                                    <form action="{{ route('film.destroy', $film->id) }}" method="post">
                                        @csrf
                                        @method('DELETE')
                                        <button class="button is-danger" type="submit">Supprimer</button>
                                    </form>
                                </td>
                            </tr>
                        @endforeach
```

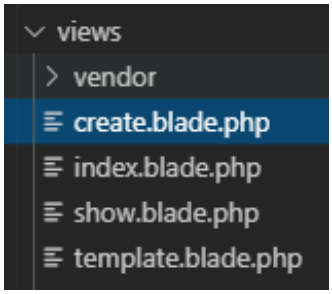
```
                </tbody>
            </table>
        </div>
    </div>
    <footer class="card-footer is-centered">
        {{ $films->links() }}
    </footer>
</div>
@endsection
```

Maintenant on a un bouton :

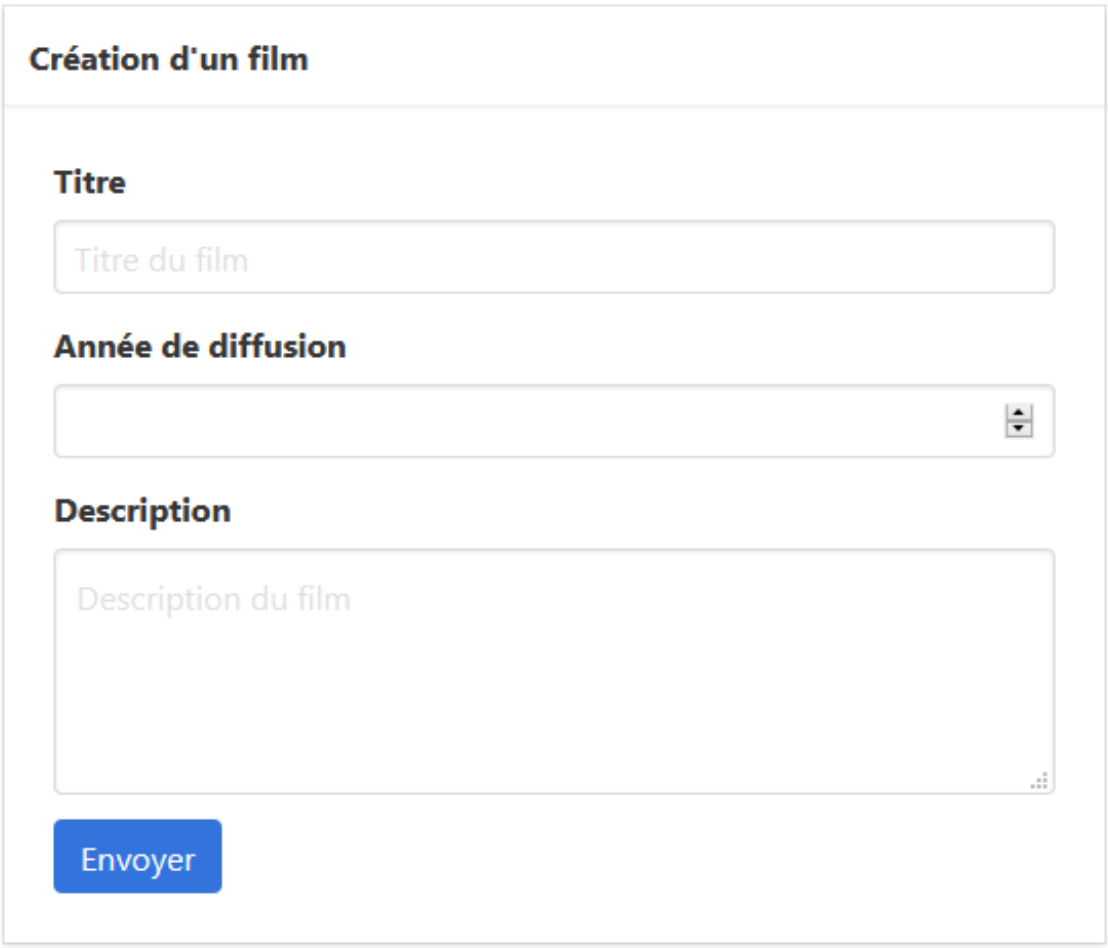


La vue create

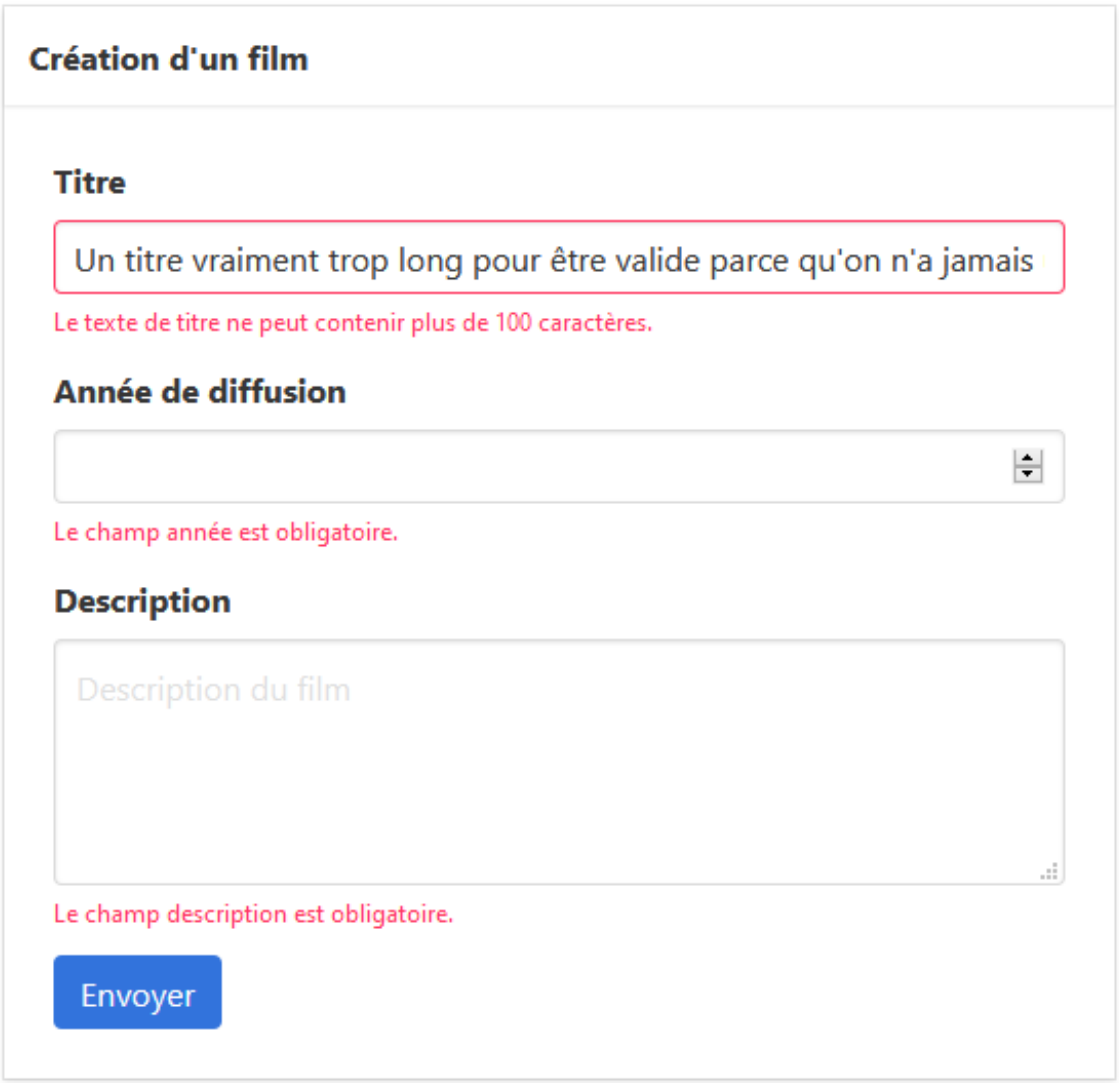
On crée la vue **create** :



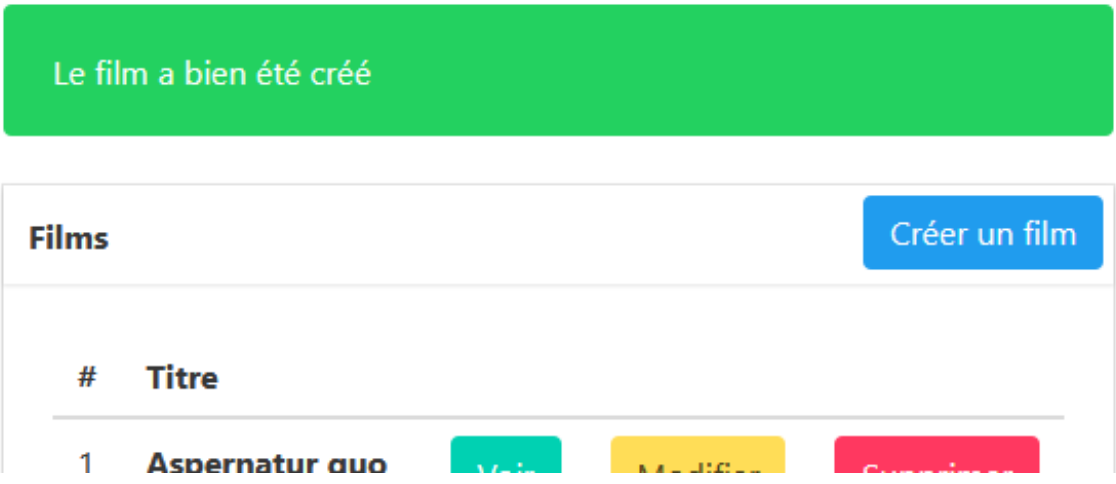
```
@extends('template')

@section('content')
   <div class="card">
      <header class="card-header">
         <p class="card-header-title">Création d'un film</p>
      </header>
      <div class="card-content">
         <div class="content">
            <form action="{{ route('films.store') }}" method="POST">
               @csrf
               <div class="field">
                  <label class="label">Titre</label>
                  <div class="control">
                    <input class="input @error('title') is-danger @enderror" type="text" name="title"
value="{{ old('title') }}" placeholder="Titre du film">
                  </div>
                  @error('title')
                     <p class="help is-danger">{{ $message }}</p>
                  @enderror
               </div>
               <div class="field">
                  <label class="label">Année de diffusion</label>
                  <div class="control">
                    <input class="input" type="number" name="year" value="{{ old('year') }}"
min="1950" max="{{ date('Y') }}">
                  </div>
                  @error('year')
                     <p class="help is-danger">{{ $message }}</p>
                  @enderror
               </div>
               <div class="field">
                  <label class="label">Description</label>
                  <div class="control">
                     <textarea class="textarea" name="description" placeholder="Description du
film">{{ old('description') }}</textarea>
                  </div>
                  @error('description')
                     <p class="help is-danger">{{ $message }}</p>
                  @enderror
               </div>
               <div class="field">
                  <div class="control">
                    <button class="button is-link">Envoyer</button>
                  </div>
               </div>
            </form>
         </div>
      </div>
   </div>
@endsection
```



Comme on a prévu la validation on va vérifier que ça marche :



Lorsqu'un film a été créé on retourne à la vue **index** avec un message :



Modifier un film

Les routes

Pour la modification d'un film on va avoir deux routes :

- pour afficher le formulaire de modification
- pour soumettre le formulaire

```
PUT|PATCH | films/{film}      | films.update  | App\Http\Controllers\FilmController@update  | web
DELETE    | films/{film}      | films.destroy | App\Http\Controllers\FilmController@destroy | web
GET|HEAD  | films/{film}/edit | films.edit    | App\Http\Controllers\FilmController@edit    | web
```

Le contrôleur

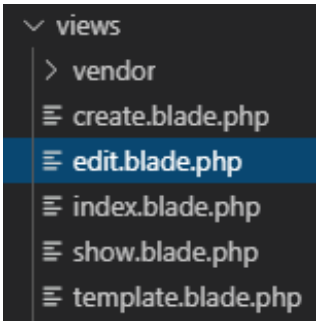
Dans le contrôleur ce sont les méthodes **edit** et **update** qui sont concernées. On va donc les coder :

```
public function edit(Film $film)
{
   return view('edit', compact('film'));
}

public function update(FilmRequest $filmRequest, Film $film)
{
   $film->update($filmRequest->all());

   return redirect()->route('films.index')->with('info', 'Le film a bien été modifié');
}
```

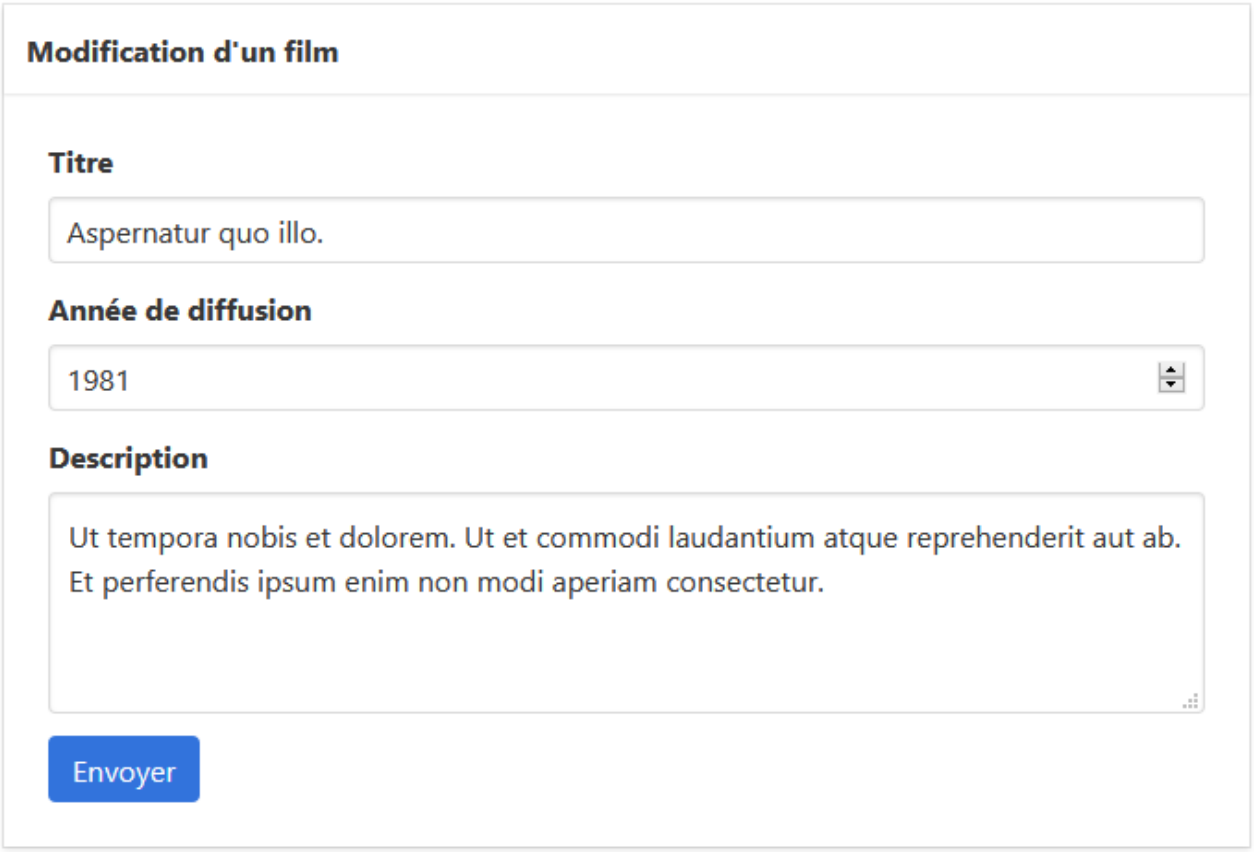
Le code ressemble beaucoup à celui vu ci-dessus pour la création et c'est normal.

La vue edit

Là aussi on va avoir pratiquement le même code que la vue **create**. La différence c'est qu'il faut renseigner les contrôles du formulaire au départ. On crée la vue **edit** :



```
@extends('template')

@section('content')
   <div class="card">
      <header class="card-header">
         <p class="card-header-title">Modification d'un film</p>
      </header>
      <div class="card-content">
         <div class="content">
            <form action="{{ route('films.update', $film->id) }}" method="POST">
               @csrf
               @method('put')
               <div class="field">
                  <label class="label">Titre</label>
                  <div class="control">
                    <input class="input @error('title') is-danger @enderror" type="text" name="title"
value="{{ old('title', $film->title) }}" placeholder="Titre du film">
                  </div>
                  @error('title')
                     <p class="help is-danger">{{ $message }}</p>
                  @enderror
               </div>
               <div class="field">
                  <label class="label">Année de diffusion</label>
                  <div class="control">
                    <input class="input" type="number" name="year" value="{{ old('year', $film-
>year) }}" min="1950" max="{{ date('Y') }}">
                  </div>
                  @error('year')
                     <p class="help is-danger">{{ $message }}</p>
                  @enderror
               </div>
               <div class="field">
                  <label class="label">Description</label>
                  <div class="control">
                     <textarea class="textarea" name="description" placeholder="Description du
film">{{ old('description', $film->description) }}</textarea>
                  </div>
                  @error('description')
                     <p class="help is-danger">{{ $message }}</p>
                  @enderror
               </div>
               <div class="field">
                  <div class="control">
                    <button class="button is-link">Envoyer</button>
                  </div>
               </div>
            </form>
         </div>
      </div>
   </div>
@endsection
```



Là encore on a la validation qui fonctionne et le message qui informe de la modification avec la redirection :

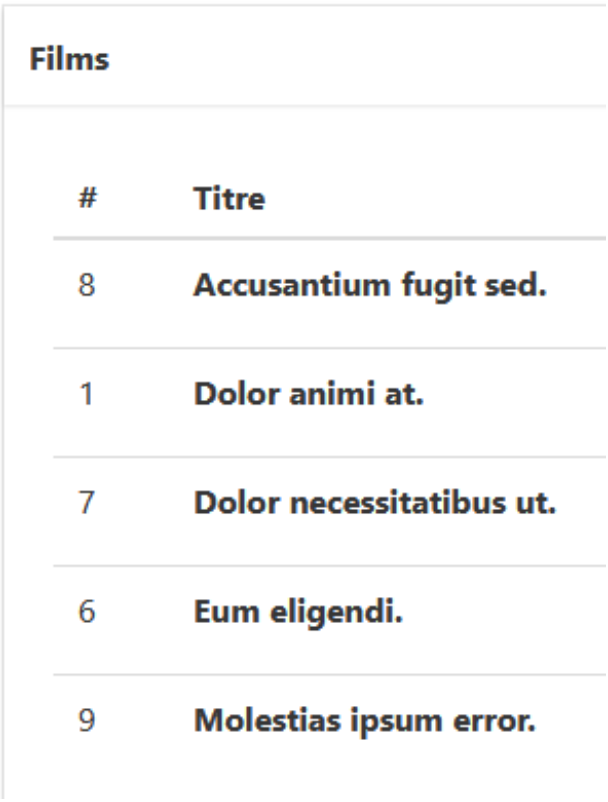
Ordonner les films

Pour le moment les films sont listés dans l'ordre de leur identifiant, ce qui n'est pas très heureux. Il serait plus judicieux d'avoir la liste dans l'ordre alphabétique des noms de films. Pour le faire il suffit d'intervenir dans le contrôleur :

```
public function index()
{
   $films = Film::oldest('title')->paginate(5);
```

Maintenant on a bien les films dans l'ordre désiré :

Finalement aussi quel intérêt de faire apparaître l'identifiant ? Surtout maintenant qu'ils sont dans le désordre, on va donc modifier la vue **index** :




```
<thead>
   <tr>
      <th>Titre</th>
      <th></th>
      <th></th>
      <th></th>
   </tr>
</thead>
<tbody>
   @foreach($films as $film)
      <tr>
         <td><strong>{{ $film->title }}
</strong></td>
         <td><a class="button is-primary"
href="{{ route('films.show', $film->id)
}}">Voir</a></td>
         <td><a class="button is-warning"
href="{{ route('films.edit', $film->id)
}}">Modifier</a></td>
         <td>
            <form action="{{ route('films.destroy', $film->id) }}" method="post">
               @csrf
               @method('DELETE')
               <button class="button is-danger" type="submit">Supprimer</button>
            </form>
         </td>
      </tr>
   @endforeach
</tbody>
```

C'est beaucoup mieux :

Soft delete

Telle qu'on a prévue la suppression des films ils sont définitivement retirés de la base. Laravel propose une autre façon de procéder avec le soft delete. Dans un premier temps on se contente de marquer le film comme supprimé en renseignant une colonne **deleted_at**. Dans un deuxième temps on peut faire la suppression définitive. En gros on crée une corbeille.

La migration

Dans la migration de la table films on ajoute la colonne avec la méthode **softDeletes** :

```
public function up()
{
   Schema::create('films', function (Blueprint $table) {
      ...
      $table->softDeletes();
   });
}
```

On relance la migration avec la population :

```
php artisan migrate:fresh --seed
```

Le modèle

Au niveau du modèle Film on ajoute le trait **SoftDeletes** :

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Film extends Model
{
    use SoftDeletes;

    protected $fillable = ['title', 'year', 'description'];
}
```

Maintenant quand on supprime un film on le conserve dans la base et on renseigne la colonne **deleted_at** :

id	title	deleted_at
1	Delectus eos sit.	2019-09-01 20:24:29
2	Vel dolorem.	*NULL*
3	Sint et.	*NULL*

Les routes

On ajoute deux routes pour les deux nouvelles actions :

```
Route::delete('films/force/{film}', 'FilmController@forceDestroy')->name('films.force.destroy');
Route::put('films/restore/{film}', 'FilmController@restore')->name('films.restore');
```

```
DELETE    | films/force/{film}   | films.force.destroy | App\Http\Controllers\FilmController@forceDestroy | web
PUT       | films/restore/{film} | films.restore       | App\Http\Controllers\FilmController@restore      | web
```

Le contrôleur

Dans le contrôleur il faut déjà modifier la méthode **index** pour inclure les films de la corbeille :

```php
public function index()
{
    $films = Film::withTrashed()->oldest('title')->paginate(5);

    return view('index', compact('films'));
}
```

Dans la méthode **destroy** il faut changer le texte du message :

```php
return back()->with('info', 'Le film a bien été mis dans la corbeille.');
```

Enfin il faut créer les deux nouvelles méthodes :

```
public function forceDestroy($id)
{
    Film::withTrashed()->whereId($id)->firstOrFail()->forceDelete();

    return back()->with('info', 'Le film a bien été supprimé définitivement dans la base de données.');
}

public function restore($id)
{
    Film::withTrashed()->whereId($id)->firstOrFail()->restore();

    return back()->with('info', 'Le film a bien été restauré.');
}
```

La vue index

Il ne reste plus qu'à modifier la vue **index** :

```
@foreach($films as $film)
    <tr @if($film->deleted_at) class="has-background-grey-lighter" @endif>
        <td><strong>{{ $film->title }}</strong></td>
            <td>
                @if($film->deleted_at)
                    <form action="{{ route('films.restore', $film->id) }}" method="post">
                        @csrf
                        @method('PUT')
                        <button class="button is-primary" type="submit">Restaurer</button>
                    </form>
                @else
                    <a class="button is-primary" href="{{ route('films.show', $film->id) }}">Voir</a>
                @endif
            </td>
            <td>
                @if($film->deleted_at)
                @else
                    <a class="button is-warning" href="{{ route('films.edit', $film->id) }}">Modifier</a>
                @endif
            </td>
        <td>
            <form action="{{ route($film->deleted_at? 'films.force.destroy' : 'films.destroy', $film->id)
}}" method="post">
                @csrf
                @method('DELETE')
                <button class="button is-danger" type="submit">Supprimer</button>
            </form>
        </td>
    </tr>
@endforeach
```

Fonctionnement

Maintenant quand on clique sur le bouton **Supprimer** d'un film ça le met dans la corbeille et il prend cette apparence avec un fond grisé et des boutons différents :

On peut alors soit le supprimer définitivement avec le bouton **Supprimer**, soit le restaurer avec le bouton **Restaurer**.

J'ai prévu un ZIP récupérable ici qui contient le projet complet pour les deux articles sur les ressources.

En résumé

- Un contrôleur de ressource permet de normaliser les opérations CRUD. Sa création et son routage sont simples.
- Le soft delete permet de mettre en place une corbeille pour les enregistrements.