

Projet Bootstrap

Les mangas

Chapitre 3

Dans le développement d'une application l'interface représente toujours une grosse partie du travail. **Bootstrap** est une collection d'outils utile à la création de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option.

Configuration de l'application

On part de l'application ProjetTemplate qui sera renommée
On passe le projet sous PhpStorm

Configuration des fichiers root/composer.json et config/app.php

Remarque : à ne pas faire, car déjà réalisé avec ProjetTemplate

Pour pouvoir utiliser la classe Form, il faut commencer par mettre à niveau l'environnement de votre projet. Pour ce faire nous allons rajouter une ligne dans le fichier composer.json de notre projet, puis mettre à jour les références à l'aide de la commande : `composer update`
Modifier le fichier composer.json comme suit :

```
"require": {  
    "php": ">=5.5.9",  
    "laravel/framework": "5.2.*",  
    "laravelcollective/html": "5.2.*"  
},
```

Puis dans la console, saisir : `composer update`.

Config/app.php

Editez ce fichier et rajouter dans la partie providers la ligne en surbrillance

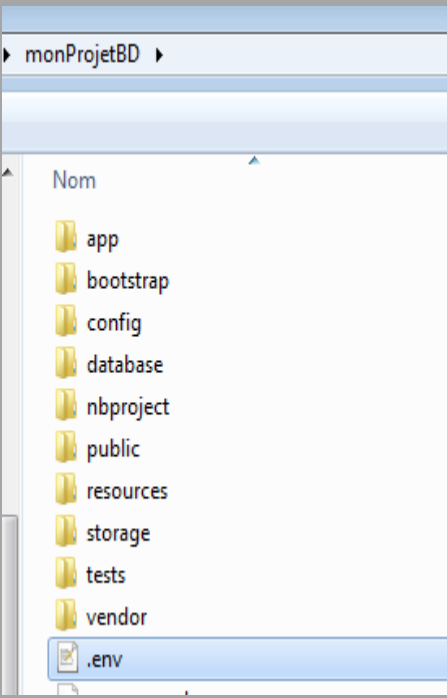
```
Illuminate\Session\SessionServiceProvider::class,  
Illuminate\Translation\TranslationServiceProvider::class,  
Illuminate\Validation\ValidationServiceProvider::class,  
Illuminate\View\ViewServiceProvider::class,  
Collective\Html\HtmlServiceProvider::class,  
  
/*  
 * Application Service Providers...  
*/
```

Ajoutez dans le tableau de la partie aliases les lignes Form et Html

```
'URL' => Illuminate\Support\Facades\URL::class,  
'Validator' => Illuminate\Support\Facades\Validator::class,  
'View' => Illuminate\Support\Facades\View::class,  
'Form' => 'Collective\Html\FormFacade',  
'Html' => 'Collective\Html\HtmlFacade',  
],  
];
```

Modification du fichier .env

Ce fichier est présent à la racine de votre projet, vous devez le modifier sous le gestionnaire de fichiers.

	<pre>APP_ENV=local APP_DEBUG=true APP_KEY=base64:ZdkgPdElRrTdXKKhFFAyHxVjZkZx Tjv6j4bA2vcq/9g= APP_URL=http://localhost DB_CONNECTION=mysql DB_HOST=192.168.225.100 DB_PORT=3306 DB_DATABASE=XXX_mangas DB_USERNAME=usersio DB_PASSWORD=sio CACHE_DRIVER=file SESSION_DRIVER=file QUEUE_DRIVER=sync REDIS_HOST=127.0.0.1 REDIS_PASSWORD=null REDIS_PORT=6379 MAIL_DRIVER=smtp MAIL_HOST=mailtrap.io MAIL_PORT=2525 MAIL_USERNAME=null MAIL_PASSWORD=null MAIL_ENCRYPTION=null</pre>
--	---

Utilisation de bootstrap

Le chargement des fichiers css se fait par l'appel suivant

```
{!! Html::style('assets/css/bootstrap.css') !!}
{!! Html::style('assets/css/mangas.css') !!}
{!! Html::style('assets/css/bootstrap.css') !!}
```

On remarque l'utilisation d'une syntaxe particulière que vous devez impérativement respecter. L'appel des fichiers de scripts se fait avec une syntaxe similaire

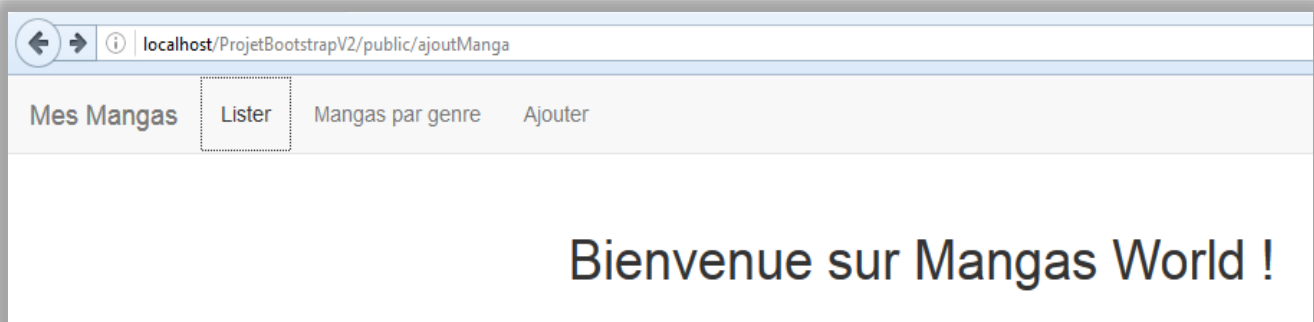
```
@yield('content')
{!! Html::script('assets/js/bootstrap.min.js') !!}
{!! Html::script('assets/js/jquery-2.1.3.min.js') !!}
{!! Html::script('assets/js/ui-bootstrap-tpls.js') !!}
{!! Html::script('assets/js/bootstrap.js') !!}
```

Remarque : Il est possible d'utiliser une autre syntaxe pour les appels

```
<link href="{{ asset('lib/bootstrap/css/bootstrap.css') }}" rel="stylesheet">
<link href="{{ asset('lib/bootstrap/css/mangas.css') }}" rel="stylesheet">
<link href="{{ asset('lib/bootstrap/css/bootstrap.css') }}" rel="stylesheet">
<script src="lib/jquery/jquery-2.1.3.min.js"></script>
<script src="lib/bootstrap/js/ui-bootstrap-tpls.js" type="text/javascript"></script>
<script src="lib/bootstrap/js/bootstrap.js"></script>
```

Exemple de menu

Nous allons construire une page nommée master.blade.php à mettre dans le répertoire layouts. Cette page sera fixe et contiendra le menu de l'application



Voici le code de cette page.

```

<!doctype html>
<html lang="fr">
  <head>
    <title>Mangas</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    {!! Html::style('assets/css/bootstrap.css') !!}
    {!! Html::style('assets/css/mangas.css') !!}
    {!! Html::style('assets/css/bootstrap.css') !!}
  </head>
  <body class="body">
    <div class="container">
      <nav class="navbar navbar-default navbar-fixed-top" role="navigation">
        <div class="container-fluid">
          <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#navbar-collapse-target">
              <span class="sr-only">Toggle navigation</span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="{{ url('/') }}">Mes Mangas</a>
          </div>
          <div class="collapse navbar-collapse" id="navbar-collapse-target">
            <ul class="nav navbar-nav">
              <li><a href="{{ url('/listMangas') }}" data-toggle="collapse" data-target=".navbar-collapse.in">Lister</a></li>
              <li><a href="{{ url('/listMangasGenre') }}" data-toggle="collapse" data-target=".navbar-collapse.in">Mangas par genre</a></li>
              <li><a href="{{ url('/ajouterManga') }}" data-toggle="collapse" data-target=".navbar-collapse.in">Ajouter </a></li>
            </ul>
          </div>
        </div><!--/.container-fluid -->
      </nav>
    </div>
    <div class="container">
      @yield('content')
    </div>
    {!! Html::script('assets/js/bootstrap.min.js') !!}
    {!! Html::script('assets/js/jquery-2.1.3.min.js') !!}
    {!! Html::script('assets/js/ui-bootstrap-tpls.js') !!}
    {!! Html::script('assets/js/bootstrap.js') !!}
  </body>

```

<html>

PageMenu .php

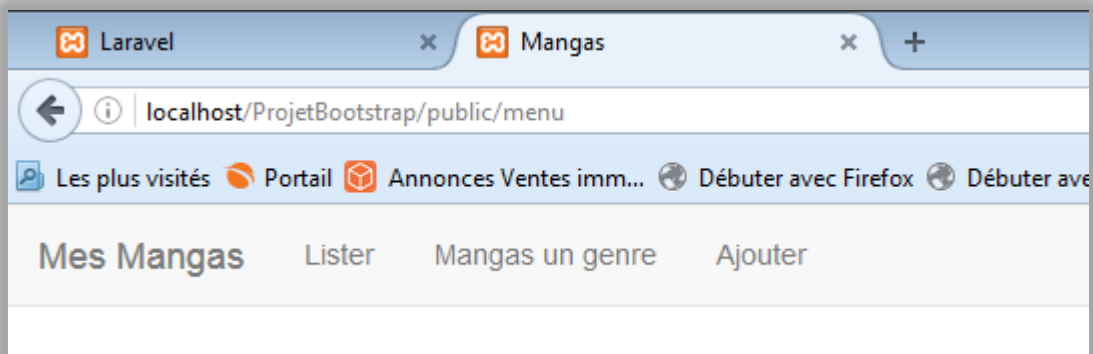
Vous construisez ensuite la page menu :

```

@extends('layouts.master')
@section('content')
  <div>
    <h1 class="bvn"> Bienvenue sur Mangas World ! </h1>
  </div>
@stop

```

Vous devez obtenir :



Dans le menu, seul l'item Ajouter appelle une page.

Routeur

Il faut mettre la route pour appeler pageMenu sur le lancement de l'application

```
Route::get('/', function () {
    return view('pageMenu');
});
```

On ajoute une deuxième ligne qui nous servira plus tard.

```
Route::get('/pageMenu', function () {
    return view('pageMenu');
});
```

Affichage de la liste des mangas

Id	Titre	Genre	Dessinateur	Scénariste	Prix	Modification
1	Bleach	1	1	1	12.50	
2	One Punch Man	3	4	4	11.80	
3	Dragon Ball Z	1	3	3	7.50	
4	Parasyte	4	6	6	9.90	
5	Hunter X Hunter	1	8	8	12.25	
6	Death Note	6	7	7	12.50	
7	Goldorak	4	3	3	45.00	
8	Barakamon	3	4	4	50.00	
9	tset	2	5	5	50.00	

Voici une première version de l'affichage des mangas

Code pour le routeur

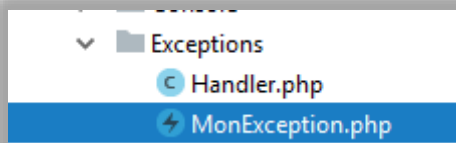
On appelle une méthode du contrôleur par l'intermédiaire du routeur

```
Route::get('/listerMangas', 'MangaController@listerMangas');
```

Construction du contrôleur

On crée le contrôleur comme une classe PHP dans le répertoire http du projet

La classe MonException est à récupérer sur le réseau. Vous devez ensuite la mettre dans :



Son code est :

```
/*...*/  
  
namespace App\Exceptions;  
  
use Exception;  
  
class MonException extends Exception  
{  
    protected $message = 'Unknown exception'; // Message d'erreur  
    private $string; // inconnu  
    protected $code = 0; // Code d'erreur de l'utilisateur  
    protected $file; // fichier source  
    protected $line; // ligne de l'erreur  
    private $trace; // inconnu  
  
    public function __construct($message, $code = 0, Exception $previous = null)  
    {  
        if (!$message) {  
            throw new $this('Unknown '. get_class($this));  
        }  
        parent::__construct($message, $code);  
    }  
  
    public function __toString()  
    {  
        return get_class($this) . " '{${this->message}}' in {${this->file}}({${this->line}})\n"  
            . "{${this->getTraceAsString()}}";  
    }  
}
```

Méthode listerMangas

Cette méthode est ajoutée au contrôleur MangaControleur

```
<?php

namespace App\Http\Controllers;

use App\Manga;
use App\Genre;
use App\Scenariste;
use App\Dessinateur;

use Illuminate\Support\Facades\Input;

use App\Exceptions\MonException;

use Request;

class ControllerMangas extends Controller
{
    public function listerMangas() {
        try {

            $unMangas = new Manga();
            $mesMangas = $unMangas->getListeMangas();

            foreach($mesMangas as $manga) {
                if(!file_exists( filename: public_path() . '/images/' . $manga->couverture)) {
                    $manga->couverture = 'erreur.png';
                }
            }

            return view( view: 'vues.listerMangas', compact( varname: 'mesMangas' ));

        } catch (MonException $e) {
            $monErreur = $e->getMessage();
            return view( view: 'vues.pageErreur', compact( varname: 'monErreur' ));
        } catch (\Exception $ex) {
            $monErreur = $ex->getMessage();
            return view( view: 'vues.pageErreur', compact( varname: 'monErreur' ));
        }
    }
}
```

On récupère la collection de mangas qui sera passée à la page d'affichage.

```
public function listerMangas() {  
    try {  
        $unMangas = new Manga();  
        $mesMangas = $unMangas->getListeMangas();  
        foreach($mesMangas as $manga) {  
            if(!file_exists(public_path() . '/images/' . $manga->couverture)) {  
                $manga->couverture = 'erreur.png';  
            }  
        }  
        return view('vues.listerMangas', compact('mesMangas'));  
    } catch (MonException $e) {  
        $monErreur = $e->getMessage();  
        return view('vues.pageErreur', compact('monErreur'));  
    } catch (\Exception $ex) {  
        $monErreur = $ex->getMessage();  
        return view('vues.pageErreur', compact('monErreur'));  
    }  
}
```

Classe metier Manga

Vous devez ajouter une classe métier nommée Manga. Cette classe est en liaison avec la table Manga de la base de données. Elle contient des accesseurs/mutateurs pour chaque propriété et les traitements métiers comme

- Recherche d'un manga
- Recherche de tous les mangas


```
<?php

namespace App;

use DB;
use Illuminate\Database\Eloquent\Model;
use App\Exceptions\MonException;
use Illuminate\Database\QueryException;

class Manga extends Model{

    protected $table = 'manga';
    protected $primaryKey = 'id_manga';

    public $timestamps = false;
    protected $fillable = [
        'id_manga',
        'id_dessinateur',
        'id_scenariste',
        'prix',
        'titre',
        'couverture',
        'id_genre'
    ];
}
```

Page liste de mangas

Code de la page d'affichage avec la requête qui utilise les jointures

```

@extends('layouts.master')
@section('content')
<div>
    <br> <br>
    <br> <br>
    <div class="container">
        <div class="blanc">
            <h1>Liste de mes Mangas</h1>
        </div>
        <table class="table table-bordered table-striped">
            <thead>
                <tr>
                    <th>Id</th>
                    <th>Titre</th>
                    <th>Genre</th>
                    <th>Dessinateur</th>
                    <th>Scénariste</th>
                    <th>Prix</th>
                    <th>Modification </th>
                </tr>
            </thead>
            @foreach($mesMangas as $unManga)
                <tr>
                    <td> {{ $unManga->id_manga }} </td>
                    <td> {{ $unManga->titre }} </td>
                    <td> {{ $unManga->lib_genre }}</td>
                    <td>
                        {{ $unManga->nom_dessinateur }}
                    </td>
                    <td>
                        {{ $unManga->nom_scenariste }}
                    </td>
                    <td> {{ $unManga->prix }} </td>
                    <td style="text-align:center;"><a href="{{ url('/modifierManga') }}/{{ $unManga->id_manga }}">
                        <span class="glyphicon glyphicon-pencil" data-toggle="tooltip" data-placement="top" title="Modifier"></span></a></td>
                </tr>
            @endforeach
            <BR> <BR>
        </table>
    </div>
</div>
@stop

```

Code de la page avec la requête sans les jointures.

```

<table class="table table-bordered table-striped">
    <thead>
        <tr>
            <th>Id</th>
            <th>Titre</th>
            <th>Genre</th>
            <th>Dessinateur</th>
            <th>Scénariste</th>
            <th>Prix</th>
            <th>Modification </th>
        </tr>
    </thead>
    @foreach($mesMangas as $unManga)
        <tr>
            <td> {{ $unManga->id_manga }} </td>
            <td> {{ $unManga->titre }} </td>
            <td> {{ $unManga->id_genre }}</td>
            <td>
                {{ $unManga->id_auteur }}
            </td>
            <td>
                {{ $unManga->id_scenariste }}
            </td>
            <td> {{ $unManga->prix }} </td>
            <td style="text-align:center;"><a href="{{ url('/modifierManga') }}/{{ $unManga->id_manga }}">
                <span class="glyphicon glyphicon-pencil" data-toggle="tooltip" data-placement="top" title="Modifier"></span></a></td>
        </tr>
    @endforeach
    <BR> <BR>
</table>
</div>
</body>
@extends('layouts.footer')
</html>

```

Voici deux fonctions avec une requête simple et une autre avec des jointures

Requête simple	Requête avec les noms
<pre> public function getListeManga() { \$mesMangas = DB::table('manga') ->Select() ->get(); return \$mesMangas; } </pre>	<pre> public function getListeMangas () { try{ \$mesMangas = DB::table('manga') ->Select() ->join('genre', 'manga.id_genre', '=', 'genre.id_genre') ->join('dessinateur', 'manga.id_dessinateur', '=', 'dessinateur.id_dessinateur') ->join('scenariste', 'manga.id_scenariste', '=', 'scenariste.id_scenariste') ->get(); return \$mesMangas; } catch (QueryException \$e) { throw new MonException(\$e->getMessage(), 5); } } </pre>

Affichage avec les noms

Mes Mangas
Lister
Mangas un genre
Ajouter

Liste de mes Mangas

Id	Titre	Genre	Dessinateur	Scénariste	Prix	Modification
1	Bleach	Aventure	TITE	TITE	12.50	✎
2	One Punch Man	Action	YUSUKE	ONE	11.80	✎
3	Dragon Ball Z	Aventure	TORIYAMA	TORIYAMA	7.50	✎
4	Parasyte	Science-fiction	IWAAKI	IWAAKI	9.90	✎
5	Hunter X Hunter	Aventure	TOGASHI	TOGASHI	12.25	✎
6	Death Note	Policier	OBATA	OBA	12.50	✎
7	Goldorak	Science-fiction	TORIYAMA	OBA	45.00	✎
8	Barakamon	Action	YUSUKE	OBA	50.00	✎
9	tset	Tanche-de-vie	OBA	OBA	50.00	✎
10	Jojo's bizarre adventure - Saison 5 - Golden Wind	Aventure	TORIYAMA	TORIYAMA	45.00	✎

Ajout d'un manga

Ce traitement est plus complexe, car il demande de :

- Construire un formulaire qui sera traité par un contrôleur
- D'initialiser des boîtes déroulantes

Pour bien le comprendre, ce traitement sera construit pas à pas

Voici la page

Mes Mangas Lister Mangas un genre Ajouter

Ajout d'un manga

Titre :

Genre :

Couverture : Aucun fichier sélectionné.

Dessinateur :

Scenariste :

Prix :

Appel du traitement

L'item du menu appelle le contrôleur par l'intermédiaire du routeur

```
/* On appelle le contrôleur */  
Route::get('/ajouterManga', 'MangaController@ajoutManga'  
);
```

AjoutManga

Formulaire formMangaAjout

Ce formulaire complexe comprend des textBox et des listes déroulantes. Nous commencerons par la première textBox pour saisir le titre.

L'item du menu appelle le contrôleur par l'intermédiaire du routeur

Code pour le routeur

On appelle une méthode du contrôleur par l'intermédiaire du routeur

```

    /* On appelle le contrôleur */
    Route::get('/ajouterManga', 'MangaController@ajoutManga'
    );

```

Méthode ajoutManga

La fonction du contrôleur qui va appeler le formulaire d'ajout est :

```

public function ajoutManga() {
    try {
        $unGenre = new Genre();
        $mesGenres = $unGenre->getListeGenres();
        $unScenariste = new Scenariste;
        $mesScenaristes = $unScenariste->getListeScenaristes();
        $unDessinateur = new Dessinateur;
        $mesDessinateurs = $unDessinateur->getListeDessinateurs();
        return view( view: 'vues.formMangaAjout', compact( varname: 'mesGenres', 'mesScenaristes',
            'mesDessinateurs' ));
    } catch (MonException $e) {
        $monErreur = $e->getMessage();
        return view( view: 'vues.pageErreur', compact( varname: 'monErreur' ));
    } catch (\Exception $ex) {
        $monErreur = $ex->getMessage();
        return view( view: 'vues.pageErreur', compact( varname: 'monErreur' ));
    }
}

```

On récupère toutes les collections qui vont nous servir à initialiser les listes déroulantes.

Entête du formulaire

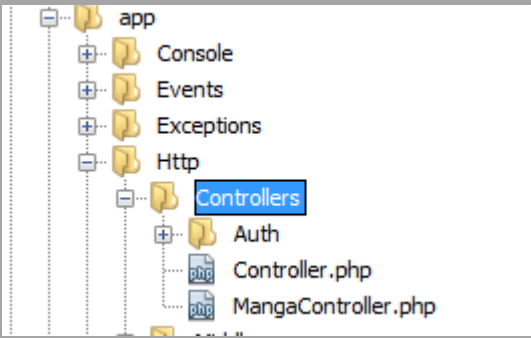
```

<@extends('layouts.master')
@section('content')
<div>
    <br> <br>
    <br> <br>
    <div class="container">
        <div class="blanc">
            <h1>Liste des Mangas d'un genre </h1>
        </div>
        <div class="well">
            {!! Form::open(['url' => 'ajoutManga', 'files' => true]) !!}

```

Ce formulaire est couplé avec un contrôleur nommé MangaController. Il contient la méthode postajouterManga.

Il faut donc écrire la fonction `postajouterManga` dans la classe `MangaController`.

	<p>Ce contrôleur va récupérer les données saisies et les insérer dans la base de données.</p>
---	---

Code de la fonction `postajouterManga` dans le contrôleur

use Illuminate\Support\Facades\Input;

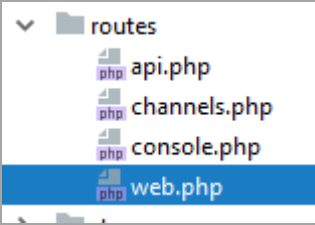
```
public function postAjouterManga() {

    try {
        $code_d = Request::input('cbDessinateur');
        $prix = Request::input('prix');
        $id_scenariste = Request::input('cbScenariste');
        $titre = Request::input('titre');
        $couverture = Input::file('couverture');
        $code_ge = Request::input('cbGenre');
        // Vérification de la couverture
        if (isset($couverture)) {
            $imageName = $couverture->getClientOriginalName();
            Input::file('couverture')->move(public_path() . '/images/', $imageName);
        } else {
            // Si il n'y a pas d'image on ne met pas de couverture
            $imageName = 'erreur.png';
        }
        $unManga = new Manga();
        $unManga->ajoutManga($code_d, $prix, $titre, $imageName, $code_ge, $id_scenariste);
        return view('vues.pageMenu');
    } catch (MonException $e) {
        $monErreur = $e->getMessage();
        return view('vues.pageErreur', compact('monErreur'));
    } catch (\Exception $ex) {
        $monErreur = 'Taille maximum dépassé !';
        return view('vues.pageErreur', compact('monErreur'));
    }
}
```

Code du traitement d'ajout dans la classe métier Manga

```
public function ajoutManga($code_d, $prix, $titre, $couverture, $code_ge, $id_scenariste){
    try{
        DB::table('manga')->insert(
            [
                'id_dessinateur' => $code_d, 'prix' => $prix,
                'titre' => $titre, 'couverture' => $couverture, 'id_genre' => $code_ge,
                'id_scenariste' => $id_scenariste
            ]
        );
    } catch (QueryException $e) {
        throw new MonException($e->getMessage(), code: 5);
    }
}
```

Pour appeler ce contrôleur il faut ajouter une route dans notre routeur qui se trouve dans le répertoire http. C'est une route avec la méthode Post (formulaire)

	<pre>Route::post('/ajoutManga', ['as' => 'postajouterManga', 'uses' => 'MangaController@postajouterManga']);</pre>
--	--

On peut à présent commencer à écrire les parties de notre formulaire

Entête & Item titre

On écrit un simple code HTML en utilisant Bootstrap

```
{!! Form::open(['url' => 'ajoutManga', 'files' => true]) !!}
<div class="col-md-12 well well-sm">
    <div class="form-group">
        <label class="col-md-3 control-label">Titre : </label>
        <div class="col-md-3">
            <input type="text" name="titre"
                value="Ajout Manga" class="form-control" required autofocus>
        </div>
    </div>
</div>
<BR> <BR>
```

Choix d'un genre

Pour le réaliser, il faut initialiser la liste déroulante avec des données qui viennent de la base mangas et qui sont passées au formulaire. Il faut donc écrire une classe nommée Genre avec une méthode qui renvoie tous les genres.

```
<?php

namespace App\metier;

use Illuminate\Database\Eloquent\Model;
use DB;

class Genre extends Model {

    //On déclare la table manga

    protected $table = 'genre';
    protected $primaryKey = 'id_genre';
    public $timestamps = false;
    protected $fillable = [
        'id_genre',
        'lib_genre'
    ];

    /**
     * Get Identifiant Manga
     * @return [int] id_manga
     */
    public function getIdGenre() {
        return $this->getKey();
    }

    public function getListeGenres() {
        $query = DB::table('genre')->get();
        return $query;
    }

}

?>
```

Cette méthode renvoie une collection de données qui sera accessible dans le code HTML

Code de la saisie du genre


```

<div class="form-group">
  <label class="col-md-3 control-label">Genre : </label>
  <div class="col-md-3">
    <select class='form-control' name='cbGenres' required>
      <OPTION VALUE=0>Sélectionner un genre</option>
      @foreach ($mesGenres as $unG)
      {
        <OPTION VALUE =" {{ $unG->id_genre}} "> {{ $unG->lib_genre}}</option>
      }
    </select>
  </div>
</div>
<BR> <BR>

```

On reçoit la liste des genres dans une collection nommée \$mesGenres. On la balaye avec une boucle @foreach dans la partie HTML. On accède aux propriétés avec
\$mesGenres->nom attribut dans la base.

La liste déroulante est initialisée. Vous pouvez remarquer l'absence de code php.

Item couverture

```

<br><br>
<div class="form-group">
  <label class="col-md-3 control-label">Couverture : </label>
  <div class="col-md-3">
    <input type="hidden" name="MAX_FILE_SIZE" value="204800"/>
    <input id='couv' type="file" name="couverture" class="btn btn-default pull-left" />
  </div>
</div>
<br><br>

```

Cette partie est semblable à la saisie du titre. Il ne doit pas poser de problème.

Item dessinateur

```

<div class="form-group">
  <label class="col-md-3 control-label">Dessinateur : </label>
  <div class="col-md-3">
    <select class='form-control' name='cbDessinateur' required>
      <OPTION VALUE=0>Sélectionner un Dessinateur </option>
      @foreach ($mesDessinateurs as $unD)
      {
        <OPTION VALUE="{{ $unD->id_dessinateur }}" > {{ $unD->nom_dessinateur }} </option>
      }
      @endforeach
    </select>
  </div>
</div>
<BR> <BR>

```

Pour le réaliser il faut écrire une classe Dessinateur avec une méthode qui renvoie la liste des auteurs de dessin. Cette classe contient une méthode getListeDessinateurs.

Classe Dessinateur

```

<?php
namespace App\metier;

use Illuminate\Database\Eloquent\Model;
use DB;

class Dessinateur extends Model {

    //On déclare la table Dessinateur

    protected $table = 'dessinateur';
    protected $primaryKey = 'id_dessinateur';
    public $timestamps = false;
    protected $fillable = [
        'id_dessinateur',
        'nom_dessinateur',
        'prenom_dessinateur'
    ];

    /**
     * Get Identifiant Manga
     * @return [int] id_manga
     */
    public function getIDessinateur() {
        return $this->getKey();
    }

    public function getListeDessinateurs() {
        $query = DB::table('dessinateur')->get();
        return $query;
    }

    public function getAuteur($id) {
        $query = DB::table('dessinateur')
            ->select()
            ->where('id_dessinateur', '=', $id)
            ->first();
        return $query;
    }
}
?>

```

Item Scenariste

La réalisation de cet item est semblable à l'item Dessinateur. Il faut créer une classe Scenariste avec une méthode qui renvoie la liste des auteurs de scénario.

```
<div class="form-group">
  <label class="col-md-3 control-label">Scenariste : </label>
  <div class="col-md-3">
    <select class='form-control' name='cbScenariste' required>
      <OPTION VALUE=0>Sélectionner un Scenariste</option>
      @foreach ($mesScenaristes as $unS)
      {
        <OPTION VALUE="{{ $unS->id_scenariste }}"> {{ $unS->nom_scenariste }} </option>
      }
      @endforeach
    </select>
  </div>
</div>
<BR> <BR>
<div class="form-group">
```

Classe Scenario

```
<?php

namespace App\metier;

use Illuminate\Database\Eloquent\Model;
use DB;

class Scenariste extends Model {
    //On déclare la table scenariste

    protected $table = 'scenariste';
    protected $primaryKey = 'id_scenariste';
    public $timestamps = false;
    protected $fillable = [
        'id_scenariste',
        'nom_scenariste',
        'prenom_scenariste'
    ];

    /**
     * Get Identifiant Manga
     * @return [int] id_manga
     */
    public function getIScenariste() {
        return $this->getKey();
    }

    public function getListeScenaristes() {
        $query = DB::table('scenariste')->get();
        return $query;
    }

    public function getScenariste ($id) {
        $query = DB::table('scenariste')
            ->select()
            ->where('id_scenariste', '=', $id)
            ->first();
        return $query;
    }
}

?>
```

Item Prix

```

<BR> <BR>
<div class="form-group">
  <label class="col-md-3 control-label">Prix : </label>
  <div class="col-md-3">
    <input type="text" name="prix" value="0" class="form-control" >
  </div>
</div>
<BR> <BR> <BR>

```

Ce traitement permet la saisie d'un prix.

Boutons de fin

```

<div class="form-group">
  <div class="col-md-6 col-md-offset-3">
    <button type="submit" class="btn btn-default btn-primary"><span class="glyphicon glyphicon-ok"></span> Valider</button>
    &nbsp;
    <button type="button" class="btn btn-default btn-primary"
      onclick="{ window.location = '{{url('/pageMenu')}}'; }">
      <span class="glyphicon glyphicon-remove"></span> Annuler</button>
  </div>
</div>

```

Le bouton Valider va appeler le contrôleur et le bouton annuler revient à la page de départ.

Modification d'un manga

Il faut commencer par définir une nouvelle route pour appeler notre page qui permettra la modification d'un manga. Lors de l'appel de la route, il faut passer le numéro de l'appel à la page de modification pour qu'elle charge les données. Voici le code présent dans la page qui affiche la liste des mangas.

```

<td> {{ $manga->prix }} </td>
<td style="text-align:center;"><a href="{{ url('/modifierManga') }}/{{ $manga->id_manga }}">
  <span class="glyphicon glyphicon-pencil" data-toggle="tooltip" data-placement="top" title="Modifier"></span></a></td>

```

On redirige vers le routeur en passant l'id du manga à modifier

```
Route::get('/modifierManga/{id}', ['uses' => 'MangaController@modification']);
```

On appelle la méthode nommée modification du contrôleur en lui passant comme paramètre l'id d'un manga.

Méthode modification

Voici le code de cette méthode, on récupère le manga à modifier que l'on passe au formulaire nommé formMangaModif. Comme pour l'ajout, il faut lui passer les collections qui serviront à l'initialisation des listes déroulantes.

```
public function modification($id) {

    try {
        $unManga = new Manga();
        $manga = $unManga->getManga($id);

        $unGenre = new Genre();
        $mesGenres = $unGenre->getListeGenres();

        $unScenariste = new Scenariste();
        $mesScenaristes = $unScenariste->getListeScenaristes();

        $unDessinateur = new Dessinateur();
        $mesDessinateurs = $unDessinateur->getListeDessinateurs();

        return view('vues.formMangaModif', compact('manga', 'mesGenres',
            'mesScenaristes', 'mesDessinateurs'));
    } catch (MonException $e) {
        $monErreur = $e->getMessage();
        return view('vues.pageErreur', compact('monErreur'));
    } catch (\Exception $ex) {
        $monErreur = $ex->getMessage();
        return view('vues.pageErreur', compact('monErreur'));
    }
}
```

Il faut ajouter à la classe Manga une méthode qui retourne un manga sur son id.

```
/**
 * Get Identifiant Manga
 * @return [int] id_manga
 */
public function getIdManga() {
    return $this->getKey();
}

public function getManga($idManga) {

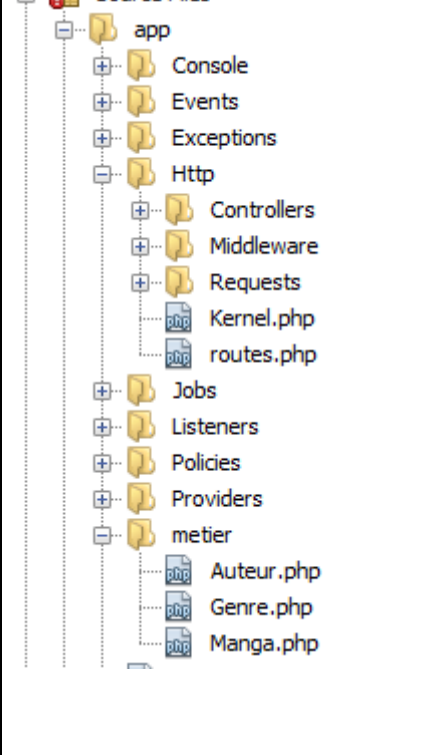
    $manga = DB::table('manga')
        ->select()
        ->where('id_manga', '=', $idManga)
        ->first();

    return $manga;
}
```

formMangaModif

Cette page contient une fonction de recherche d'un manga. Elle récupère le numéro du manga passé en paramètre.

Cette fonction est écrite dans la classe Manga. Pour l'appeler, on utilise la méthode suivante présente dans la classe Manga :

	<p>Pour forcer un retour sous forme d'objet, il faut utiliser la fonction first () à la place de get ()</p> <p>On utilise la méthode first pour rendre un objet et non une collection</p> <pre>public function getManga (\$idManga){ try{ \$manga = DB::table('manga') ->select() ->where('id_manga', '=', \$idManga) ->first(); return \$manga; } catch (QueryException \$e) { throw new MonException(\$e->getMessage(), code: 5); } }</pre>
---	---

Formulaire de modification

La formulaire formModifmanga est similaire à la page ajoutManga. A la différence, elle possède des valeurs pour le manga à modifier.

Ce formulaire sera chaîné sur une nouvelle route qui recevra l'id du manga à mettre à jour. Les différents champs contiendront les valeurs à modifier.

La route appellera une méthode de mise à jour qui sera écrite dans le contrôleur de notre manga. Ce dernier possède déjà une méthode d'ajout.

Entête du formulaire

```
@extends('layouts.master')
@section('content')
<div>
    <br> <br>
    <br> <br>
    <div class="container">
        <div class="blanc">
            <h1>Modification d'un manga </h1>
        </div>
    </div>
    <div class="well">
```

Ce formulaire va s'insérer dans la page master.blade.

```
{!! Form::open(array('route' => array('postmodifierManga',$unManga->id_manga), 'method' => 'post')) !!}
<div class="col-md-12 well well-sm">
    <div class="form-group">
        <label class="col-md-3 control-label">Titre : </label>
        <div class="col-md-3">
            <input type="hidden" name="id_manga" value="{{ $unManga->id_manga or '' }}" />
            <input type="text" name="titre" value="{{ $unManga->titre or '' }}"
                class="form-control" required autofocus >
        </div>
    </div>
</div>
<BR> <BR>
```

Contrairement à l'ajout, l'appel du contrôleur ne se fait pas directement, on passe par une route appelée avec postmodifierManga avec le paramètre \$UnManga->id_manga. La méthode utilisée est POST

Route postmodifierManga

Voici la route utilisée

```
Route::post('/postmodifierManga/{id}',
[
    'as' => 'postmodifierManga',
    'uses' => 'MangaController@postmodifierManga'
]);
```

La fonction postmodifierManga recevra le paramètre ID qui contient le numéro du manga à modifier.

Voici le code de cette fonction présente dans le contrôleur. On utilise une requête de mise à jour.


```

public function postModifierManga() {
try{
    // Récupération des données
    $code = Request::input('id_manga');
    $code_d = Request::input('cbDessinateur');
    $prix = Request::input('prix');
    $code_sc = Request::input('cbScenariste');
    $titre = Request::input('titre');
    $code_ge = Request::input('cbGenres');
    $couverture = Input::file('couverture');
    // Vérification de la couverture
    if (isset($couverture)) {
        $imageName = $couverture->getClientOriginalName();
        Input::file('couverture')->move(public_path() . '/images/', $imageName);
    } else {
        // Si il n'y a pas d'image on ne met pas de couverture
        $imageName = 'erreur.png';
    }
    // Modification d'un manga
    $unManga = new Manga();
    $unManga->modificationManga($code, $code_d, $prix, $titre, $imageName, $code_ge, $code_sc);
    return redirect(to: '/listerMangas');
} catch (MonException $e) {
    $monErreur = $e->getMessage();
    return view( view: 'vues.pageErreur', compact( varname: 'monErreur' ));
} catch (\Exception $ex) {
    $monErreur = 'Taille maximum dépassé !';
    return view( view: 'vues.pageErreur', compact( varname: 'monErreur' ));
}
}

```

Il faut rajouter la méthode de modification dans la classe métier Manga

```

public function modificationManga ($code, $code_d, $prix, $titre, $couverture,
    $code_ge, $scenariste ) {
    try{
        DB::table('manga')
            ->where ('id_manga', $code)
            ->update([
                'id_dessinateur' => $code_d,
                'prix' => $prix,
                'titre' => $titre,
                'couverture' => $couverture,
                'id_genre' => $code_ge,
                'id_scenariste' => $scenariste,
            ]);
    } catch (QueryException $e) {
        throw new MonException($e->getMessage(), code: 5);
    }
}

```

Item genre du manga

Ce traitement est à étudier avec attention. Il faut charger la liste des genres dans la liste déroulante et sélectionner celui du manga que l'on veut modifier.

Il faut donc découper la partie Option pour mettre à jour l'option selected lorsque le genre du manga correspond à celui qui doit être modifié. Laravel nous permet d'utiliser des structures de contrôle foreach et if précédées de @.

On peut récupérer la collection de genres et accéder directement aux property (attributs de la classe)

```
<div class="form-group">
  <label class="col-md-3 control-label">Genre : </label>
  <div class="col-md-3">
    <select class='form-control' name='cbGenres' required>
      <OPTION VALUE=0>Sélectionner un genre</option>
      @foreach ($mesGenres as $unG)
        selected=""
        <option value="{{ $unG->id_genre }}"
          @if ( $unG->id_genre == $unManga->id_genre )
            selected="selected"
          @endif
        > {{ $unG->lib_genre}} </option>
      @endforeach
    </select>
  </div>
</div>
<BR> <BR>
```

Item Couverture

```
<div class="form-group">
  <label class="col-md-3 control-label">Couverture : </label>
  <div class="col-md-3">
    <input type="hidden" name="MAX_FILE_SIZE" value="204800"/>
    <input name="couverture" type="file" class="btn btn-default pull-left"/>

    @if ($unManga->id_manga=='')
      /* Code à écrire */

    @endif

  </div>
</div>
<BR> <BR>
```

Le code php permet d'initialiser le champ couverture.

Item Scenariste

Le code est semblable à celui du genre.

```
<br /><br />
<div class="form-group">
  <label class="col-md-3 control-label">Scenariste : </label>
  <div class="col-md-3">
    <select class="form-control" name="cbScenariste" required>
      <option value="0">Sélectionner un Scenariste</option>

      @foreach($mesScenaristes as $unS)
        <option value="{{ $unS->id_scenariste }}"
          @if($unS->id_scenariste == $manga->id_scenariste )
            selected
          @endif
        >{{ $unS->nom_scenariste }}</option>
      @endforeach
    </select>
  </div>
</div>
<br /><br />
```

Item Dessinateur

Le code est semblable au précédent

```

<br /><br />
<div class="form-group">
  <label class="col-md-3 control-label">Dessinateur : </label>
  <div class="col-md-3">
    <select class="form-control" name="cbDessinateur" required>
      <option value="0">Sélectionner un Dessinateur</option>

      @foreach($mesDessinateurs as $unD)
        <option value="{{ $unD->id_dessinateur }}"
          @if($unD->id_dessinateur == $manga->id_dessinateur )
            selected
          @endif
        >{{ $unD->nom_dessinateur }}</option>
      @endforeach
    </select>
  </div>
</div>
<br /><br />

```

Item Prix

```

<div class="form-group">
  <label class="col-md-3 control-label">Prix : </label>
  <div class="col-md-3">
    <input type="text" name="prix" value="{{ $unManga->prix or '' }}"
      class="form-control" >
  </div>
</div>

```

Ce code ne présente aucune difficulté.

Boutons du formulaire

```

<div class="form-group">
  <div class="col-md-6 col-md-offset-3">
    <button type="submit" class="btn btn-default btn-primary"><span class="glyphicon glyphicon-ok"></span> Valider</button>
    &nbsp;
    <button type="button" class="btn btn-default btn-primary"
      onclick="{ window.location = '{{url('/pageMenu')}}';}">
      <span class="glyphicon glyphicon-remove"></span> Annuler</button>
  </div>
</div>

```

On revient à la page qui va afficher la liste des mangas

Exécution d'une modification

The screenshot shows a web application interface for modifying a manga. At the top, there is a navigation bar with links: 'Mes Mangas', 'Lister', 'Mangas un genre', and 'Ajouter'. The main heading is 'Modification d'un manga'. Below this, there are several form fields:

- Titre :** A text input field containing 'Dragon Ball Z'.
- Genre :** A dropdown menu with 'Aventure' selected.
- Couverture :** A file upload area with a 'Parcourir...' button, the text 'Aucun fichier sélectionné.', and a file named 'oiseau3.PNG'.
- Scenariste :** A dropdown menu with 'TITE' selected.
- Dessinateur :** A dropdown menu with 'OBATA' selected.
- Prix :** A text input field containing '7.50'.

At the bottom of the form, there are two buttons: '✓ Valider' and '✕ Annuler'.

Recherche des mangas par genre

Ce traitement consiste à choisir un genre de manga dans une liste déroulante et d'afficher les titres correspondants.



The screenshot shows a web application interface for searching mangas by genre. The main heading is 'Liste des Mangas d'un genre'. Below this, there is a form with the following elements:

- Genre :** A dropdown menu with 'Action' selected.

At the bottom of the form, there are two buttons: '✓ Valider' and '✕ Annuler'.

On sélectionne un genre de manga

Liste de mes Mangas

Id	Titre	Genre	Dessinateur	Scénariste	Prix	Modification
2	One Punch Man	Action	YUSUKE	ONE	11.80	
8	Barakamon	Action	YUSUKE	OBA	50.00	

On affiche la liste

Route : appel Formulaire

Le choix d'un genre de manga va se faire dans un formulaire qui sera appelé sur un item du menu. Il faut donc définir une route.

```
<ul class="nav navbar-nav">
  <li><a href="{{ url('/listerMangas') }}" data-toggle="collapse" data-target=".navbar-collapse.in">Lister</a></li>
  <li><a href="{{ url('/listerMangasGenre') }}" data-toggle="collapse" data-target=".navbar-collapse.in">Mangas par genre</a></li>
  <li><a href="{{ url('/pageMangaAjout') }}" data-toggle="collapse" data-target=".navbar-collapse.in">Ajouter</a></li>
</ul>
</div>
<!--/.container-fluid -->
```

La route sera

```
/* On appelle le contrôleur */
Route::get('/listerMangasGenre', 'MangaController@listerGenre'
);
```

La fonction du contrôleur sera :

```
public function listerGenre ()
{
    $unGenre = new Genre();
    $mesGenres = $unGenre->getListeGenres();
    return view('vues/fomChoixMangaGenre', compact('mesGenres') );
}
```

Formulaire : formChoixMangaGenre

On construit le formulaire qui va nous permettre de choisir un genre de manga. L'entête de ce formulaire est chaîné sur un contrôleur qui va récupérer le choix sélectionné :

Entête du formulaire

```
<@extends('layouts.master')
@section('content')
<div>
    <br> <br>
    <br> <br>
    <div class="container">
        <div class="blanc">
            <h1>Liste des Mangas d'un genre </h1>
        </div>

        <div class="well">
            {!! Form::open(array('route' => array('postAfficherManga','cbGenres'), 'method' => 'post')) !!}
        </div>
    </div>
</section>
```

On appelle une route `postAfficherManga` avec un paramètre `cbGenres` (issu de la liste déroulante) par la méthode `post`.

```
// post genre
Route::post('/postAfficherMangasGenres',
[
    'as' => 'postAfficherMangasGenres',
    'uses' => 'ControllerMangas@listerMangasGenre'
]);
```

La méthode appelée est `listerMangaGenre` qui appartient au contrôleur `MangaController`. Cette méthode récupère le choix du genre et lance une requête sur la base de données pour obtenir une collection de données qui sera passée à une vue : `pageMangaGenre`

```
public function listerMangasGenre() {  
    try {  
        $unMangas = new Manga();  
        $idGenre = Request::input('cbGenres');  
        $mesMangas = $unMangas->getListeMangasGenre($idGenre);  
        return view('vues.listerMangas', compact('varname: 'mesMangas'));  
    } catch (MonException $e) {  
        $monErreur = $e->getMessage();  
        return view('vues.pageErreur', compact('varname: 'monErreur'));  
    } catch (\Exception $ex) {  
        $monErreur = $ex->getMessage();  
        return view('vues.pageErreur', compact('varname: 'monErreur'));  
    }  
}
```

La requête dans la classe Manga est

```
public function getListeMangasGenre($id_genre) {  
    try{  
        $mesMangas = DB::table('manga')  
            ->select()  
            ->join('genre', 'manga.id_genre', '=', 'genre.id_genre')  
            ->join('dessinateur', 'manga.id_dessinateur', '=', 'dessinateur.id_dessinateur')  
            ->join('scenariste', 'manga.id_scenariste', '=', 'scenariste.id_scenariste')  
            ->where('genre.id_genre', '=', $id_genre)  
            ->get();  
        return $mesMangas;  
    } catch (QueryException $e) {  
        throw new MonException($e->getMessage(), code: 5);  
    }  
}
```

Retour au formulaire : liste déroulante

Cette partie concerne le choix d'un genre. Pour alimenter la liste déroulante, on récupère la collection de genres passée en paramètre.


```

<@extends('layouts.master')
@section('content')
<div>
  <br> <br>
  <br> <br>
  <div class="container">
    <div class="blanc">
      <h1>Liste des Mangas d'un genre </h1>
    </div>

    <div class="well">
      {!! Form::open(array('route' => array('postAfficherManga','cbGenres'), 'method' => 'post')) !!}
      <div class="form-group">
        <label class="col-md-3 control-label">Genre : </label>
        <div class="col-md-3">
          <select class="form-control" name='cbGenres' required>
            <OPTION VALUE=0>Sélectionner un genre</option>
            @foreach ($mesGenres as $unG)
              <option value="{{ $unG->id_genre }}" > {{ $unG->lib_genre }} </option>
            @endforeach
          </select>
        </div>
      </div>
      <br/><br/><br/>
      <div class="form-group">
        <div class="col-md-6 col-md-offset-3">
          <button type="submit" value="Envoyer" class="btn btn-default btn-primary"/><span class="glyphicon glyphicon-ok">

          </span> Valider</button>
          &nbsp;
          <button type="button" class="btn btn-default btn-primary"
            onclick="{ window.location = '{{url('/pageMenu')}}'; }">
            <span class="glyphicon glyphicon-remove"></span> Annuler</button>
        </div>
        {!! Form::close() !!}
      </div>
    </div>
  </div>
</div>
@stop

```

Affichage de la page : pageMangaGenre

```

@extends('layouts.master')
@section('content')
<div>
    <br> <br>
    <br> <br>
    <div class="container">
        <div class="blanc">
            <h1>Liste de mes Mangas</h1>
        </div>
        <table class="table table-bordered table-striped">
            <thead>
                <tr>
                    <th>Id</th>
                    <th>Titre</th>
                    <th>Genre</th>
                    <th>Dessinateur</th>
                    <th>Scénariste</th>
                    <th>Prix</th>
                    <th>Modification </th>
                </tr>
            </thead>
            <tbody>
                @foreach($mesMangas as $unManga)
                <tr>
                    <td> {{ $unManga->id_manga }} </td>
                    <td> {{ $unManga->titre }} </td>
                    <td> {{ $unManga->lib_genre }} </td>
                    <td>
                        {{ $unManga->nom_dessinateur }}
                    </td>
                    <td>
                        {{ $unManga->nom_scenariste }}
                    </td>
                    <td> {{ $unManga->prix }} </td>
                    <td style="text-align:center;"><a href="{{ url('/modifierManga') }}/{{ $unManga->id_manga }}">
                        <span class="glyphicon glyphicon-pencil" data-toggle="tooltip" data-placement="top" title="Modifier"></span></a></td>
                </tr>
                @endforeach
            </tbody>
        </table>
    </div>
</div>
@stop

```

Affichage de la couverture


Un problème se pose, il faut gérer le fichier image avec le nom de la couverture

Liste de mes Mangas

Bleach

Genre : Aventure
 Dessinateur : TITE
 Scénariste : TITE
 Prix : 12.50

[Modification](#)



Ajout d'un manga

Dans la vue, On va chercher l'image correspondante

```
// Vérification de la couverture
if (isset($couverture)) {
    $imageName = $couverture->getClientOriginalName();
    Input::file('couverture')->move(public_path() . '/images/', $imageName);
} else {
    // Si il n'y a pas d'image on ne met pas de couverture
    $imageName = 'erreur.png';
}
```