TD LARAVEL

Afin de vous familiariser avec le framework Laravel , nous allons réaliser en cours une micro application où les utilisateurs pourront poster de simple messages.

PARTIE 1 – INSCRIPTION / LOGIN

Par chance Laravel nous mâche le travail en ce qui concerne la partie Inscription et connexion.

Lancer la commande *php artisan migrate* pour créer la table users déjà préétablie par Laravel (cf. dossier database/migration).

Puis lancer la commande *php artisan make :auth pour créer les templates dans le dossier ressources/views.*

Nous aurons ainsi un système clé d'inscription / connexion clé en main :

Lien vers la page d'inscription :
localhost/platform/public/index.php/register

Lien vers la page de connexion :
localhost/platform/public/index.php/login

PARTIE 2 – La classe Post (Model)

Importer la table posts de la base de données « projet » dans la base de données « platform ».

Dans le dossier app/models créer le fichier Post.php :

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model as Eloquent;

class Post extends Eloquent
{

      protected $with = ['author'];  // A chaque fois qu'on récupérera un post on joindra les infos de son auteur

      public function author()
      {

                return $this->belongsTo('App\User','user_id');  // Laravel fera une jointure de table pour récupérer les infos de l'auteur associé à la clé étrangère user_id (définie en second paramètre de la méthode belongsTo.

      }


 }
```

Cela n'est pas obligatoire mais dans la class User (app/User.php) il est possible d'établir la relation inverse :

< ?php namespace App ;

…

class User extends Model {

    …

    public function posts(){

        return $this->hasMany('App\Models\Post') ;

    }

}

Ceci pourra être utile lorsque l'on souhaitera récupérer tous les posts d'un utilisateur bien précis : $posts = $user->posts ;

PARTIE 4 – Les templates (Vue)

1) Le template global « app.blade.php »

Il s'agit du template parent dont tous les autres templates de notre projet hériteront.

Nous choisirons le template layouts/app.blade.php créé par Laravel que nous pourrons modifier utltérieurement.

2) Le template enfant « create.blade.php»

Créer un dossier post dans le dossier ressources/views puis ajoutez-y le fichier create.blade.php avec ce code :

```
< !—app/ressources/views/post/create.blade.php -->

@extends('layouts.app') < !—hérite du template situé dans
ressources/views/layouts/app.blade.php -->

@section('content')

{!! Form::open() !!}

    {{ csrf_field() }}

    {!! Form::textarea('title',{{ old('title') }} ) !!}

    {!! Form::textarea('content', {{ old('content') }}) !!}

    {!! Form::submit('Envoyer !') !!}

{!! Form::close() !!}

@endsection

< !—fin fichier create.blade.php -->
```

NB : la fonction old('nom_champ_formulaire') permet de garder les anciennes valeur pour éviter un reset du formulaire en cas d'erreurs sur l'un des champs lors de la soumission.

NB² : Tout le contenu entre les 2 sections 'content' en bleues sera inclus à la place du @yield('content') du template layouts/app.blade.php

Pour en savoir plus sur la classe Form :
https://laravelcollective.com/docs/5.2/html

PARTIE 4 – PostController

1) Créer dans le dossier app/Http/Controllers le fichier PostControllers.php :

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

use Illuminate\Routing\Controller;

class PostController extends Controller {


}
```

2) Charger la classe Post

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

use Illuminate\Routing\Controller;
use App\Models\Post;

class PostController extends Controller {


}
```

3) Charger la classe Request qui nous permettra de récupérer les données de formulaire :

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

use Illuminate\Routing\Controller;
use App\Models\Post;
use Illuminate\Http\Request;

class PostController extends Controller {


}
```

En savoir plus sur l'objet Request :
https://laravel.com/docs/5.2/requests

4) Charger la classe Auth qui nous permettra de récupérer les informations de l'utilisateur connecté, et donc de l'auteur du post :

```
<?php namespace App\Http\Controllers ;
    use Illuminate\Routing\Controller;
    use App\Models\Post;
    use Illuminate\Http\Request;
    use Auth ;

    class PostController extends Controller {

    }
```

5) Charger la classe Validator qui nous permettra de vérifier que l'utilisateur n'envoie pas n'importe quoi lorsqu'il soumet un formulaire :

```
<?php namespace App\Http\Controllers ;
    use Illuminate\Routing\Controller;
    use App\Models\Post;
    use Illuminate\Http\Request;
    use Auth ;
    use Validator ;

    class PostController extends Controller {

    }
```

6) **Ajouter la méthode validator qui se chargera de vérifier les données de formulaire envoyé par l'utilisateur lorsqu'il cherche à créer ou à éditer un post :**

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

use Illuminate\Routing\Controller;
use Illuminate\Http\Request;
use App\Models\Post;
use Auth;
use Validator ;

class PostController extends Controller {

    protected function validator(array $data){

        return Validator::make($data, ['content' => 'required|max:140',
                                       'title' => 'min:5|max:100']);

    }

}
```

7) **Ajout de la méthode create qui aura simplement le rôle d'afficher le formulaire de création de post :**

```
public function  create()

{

   return view('post/create') ;

}
```

8) **Création de la route dans le fichier app/http/routes.php** qui permettra d'afficher le formulaire de création de post lorsque l'on tapera l'url localhost/platform/public/post/create :

```
Route::get('/post/create', 'PostController@create');
```

9) Ajouter la méthode store qui se chargera uniquement d'enregistrer le post envoyé en base de données :

```
public function  store(Request $request)

{
        $validation = $this->validator($request->all()) ;

        If( !$validation->fails()) {

                $post = new Post ;
                $post->title = $request->input('title') ;
                $post->content = $request->input('content') ;
                $post->user_id = Auth ::Id() ;
                $post->save() ;
                return redirect('/') ; // On redirige sur la page d'accueil

        }
        else
        {
                return redirect('/post/create')->withErrors($validation)->withInput() ;

                // En cas d'erreur on redirige sur le formulaire de création de post. La méthode
withErrors permet de transmettre au template les erreurs rencontrées lors de la validation. La
méthode withInput permet de concervé les précédentes valeurs envoyé pour éviter de réinitialiser
tout le formulaire.

        }

 }
```

10) Création de la route associée à la méthode store :

```
// app/http/routes.php

Route :: post('/post/create', 'PostController@store') ;
```

11) Création de la méthode edit qui aura pour rôle d'afficher le formulaire d'édition d'un post :

```
public function edit(Post $post){

    return view('post.edit',[ 'post' => $post]) ;


}
```

12) Création de la route associée à la méthode edit :

```
Route::get('/post/{post}/edit', 'PostController@edit');  // {post} correspond au paramètre Post $post définit dans la méthode edit de la classe PostController.

// Lorsque l'on tape localhost/public/post/2/edit  alors le paramètre $post sera une instance de la classe Post correspondant au post ayant l'id 2 dans la table posts
```

13) Création du template d'édition du post :

```
< !—ressources/views/post/edit.blade.php -->

@extends('layouts.app')


@section('content')

{!! Form::model($post) !!} <!-- la méthode model prend en paramètre une instance de model (exemple instance de la classe Post) et elle se chargera de remplir automatiquement tous les champs de formulaire qui lui sont associés -->

{{ csrf_field() }}

{!! Form::text('title') !!}

{!! Form::textarea('content') !!}

{!! Form::submit('Envoyer !') !!}

{!! Form::close() !!}

@endsection
```

14) Création de la méthode update qui permettra de modifier le post en base de données:

```
public function update(Post $post, Request $request){ // $post sera une instance de la classe post définie par l'url tapée.

        $validator = $this->validator($request->all());


        if($validator->fails() == false){  // S'il n'y pas d'erreur dans nos champs de formulaire on enregistre en base de données

                $post->title = $request->input('title');

                $post->content = $request->input('content');

                $post->save();  // On applique la méthode save pour modifier en base de données le post dans la table "posts"

                return redirect('/post/'.$post->id.'/edit'); //  On redirige l'utilisateur sur la page de création des posts.

        }

        else

        {

            return redirect('/post/'.$post->id.'/edit')->withErrors($validator)->withInput();

        }

}
```

15) Créer la route associée à la méthode edit :

```
Route::post('/post/{post}/edit','PostController@store');
```

Fichiers finaux :

Contrôleur : https://codeshare.io/TD_LARAVEL_class_PostController

Model : Class Post : https://codeshare.io/TD_LARAVEL_class_Post

Vues :

- layouts/app.blade.php : https://codeshare.io/TD_LARAVEL_view_app
- post/create.blade.php : https://codeshare.io/TD_LARAVEL_view_create
- post/edit.blade.php : https://codeshare.io/TD_LARAVEL_view_edit

Routes :

App/http/routes.php : https://codeshare.io/TD_LARAVEL_routes.php