# 額外模組

composer require laravelcollective/html

須注意 => 該package沒有繼續更新

下載完後即可使用模組的功能

<https://laravelcollective.com/docs/6.x/html#file-input>

## 更改form內容 - 1

@section('content')

    {{-- <form action="/posts" method="POST"> --}}

    {!! Form::open(['method'=>'POST', 'action'=>'App\Http\Controllers\PostsController@store'])!!}

        {{csrf\_field()}}

        <div class="form-group">

            {!! Form::label('title', 'Title', ['class' => 'form-label']) !!}

            {!! Form::text('title', null, ['class' => 'form-control']) !!}

            {!! Form::label('content', 'Content', ['class' => 'form-label']) !!}

            {!! Form::text('content', null, ['class' => 'form-control']) !!}

        </div>

        <div class="form-group">

            {!! Form::submit('Create post', ['class' => 'btn btn-primary']) !!}

        </div>

        {{-- <input type="text" name="title" placeholder="Enter title">

        <input type="text" name="content" placeholder="Enter content">

        <input type="submit" name="submit"> --}}

    {!! Form::close() !!}

    {{-- </form> --}}

@endsection

Form::open()開啟表單，第一個變數是傳遞的方法，第二個變數是傳遞的路徑=>這邊直接取用controllers的功能

Form::label()新增標籤，第一個變數是label綁定誰，第二個變數是label的內容，第三個變數給label使用class

Form::text()新增input:text，第一個變數是name和id，第二個變數是欄位內容，第三個變數給input:text使用class

Form::submit()提交表單，第一個變數是按鈕內容，第二個變數給按鈕class

Form::close()最後要關閉表單

更改form內容 - 2 - Model

    {{-- <form action="/posts/{{$post->id}}" method="POST"> --}}

    {!! Form::model($post, ['method'=>'PATCH', 'action'=>['App\Http\Controllers\PostsController@update', $post->id]]) !!}

        @csrf

        {{-- <input type="hidden" name="\_method" value="PUT"> --}}

        <div class="form-group">

            {!! Form::label('title', 'Title', ['class'=>'form-label']) !!}

            {!! Form::text('title', null, ['class'=>'form-controll']) !!}

            {!! Form::label('content', 'Content', ['class'=>'form-label']) !!}

            {!! Form::text('content', null, ['class'=>'form-controll']) !!}

            {{-- <input type="text" name="title" value={{$post->title}}>

            <input type="text" name="content" value={{$post->content}}> --}}

        </div>

        <div class="form-group">

            {!! Form::submit('Update', ['class'=>'btn btn-info']) !!}

            {{-- <input type="submit" name="submit" value="UPDATE"> --}}

        </div>

        {!! Form::close() !!}

    {{-- </form> --}}

Form::model()填充表單，第一個變數為填充的變數，第二個變數是陣列，陣列中的第一個變數是方法，第二個變數是執行的路徑

這邊使用Post模型，因此是填充$post參數進底下的title和content中

Validation

    public function store(Request $request)

    {

        //

        // Post::create($request->all());

        $this->validate($request, [

            'title' => 'required|min:1|max:255',

            'content' => 'required|min:5|max:255'

        ]);

        $post = new Post;

        $post->title = $request->title;

        $post->content = $request->content;

        $post->save();

        return redirect('posts');

    }

在App\Http\Controllers\Controller中，引入了ValidateRequest特性，使得其他的controller可以使用此方法。

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;

use Illuminate\Foundation\Validation\ValidatesRequests;

use Illuminate\Routing\Controller as BaseController;

透過$this->validate($request)去驗證使用者輸入的欄位

陣列中加入條件式去判斷使用者輸入的內容

未來可以使用$this->session()->的功能去顯示錯誤訊息

錯誤訊息

當validation驗證到錯誤時，會產生$errors的物件，裡面會儲存錯誤訊息

Route::group(['middleware'=>'web'], function() {

    Route::resource('/posts', PostsController::class);

});

因此要先到web.php中，使用該功能，建立middleware將需要使用到的路徑包覆進去

    @if (count($errors) > 0)

        <div class="alert alert-danger">

            <ul>

                @foreach($errors->all() as $error)

                    <li>{{$error}}</li>

                @endforeach

            </ul>

        </div>

    @endif



另一種驗證方式

php artisan make:request request\_name

會產生一個request檔案在App\Http\Requests\裡面，可以將Controller內部的驗證邏輯，放置到額外建立的request檔案中進行管理

class CreatePostRequest extends FormRequest

{

    /\*\*

     \* Determine if the user is authorized to make this request.

     \*/

    public function authorize(): bool

    {

        return true;

    }

    /\*\*

     \* Get the validation rules that apply to the request.

     \*

     \* @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array|string>

     \*/

    public function rules(): array

    {

        return [

            'title' => 'required|min:1|max:255',

            'content' => 'required|min:5|max:255'

        ];

    }

}

將有修改過的部分標記起來

use App\Http\Requests\CreatePostRequest;

    public function store(CreatePostRequest $request)

    {

        $post = new Post;

        $post->title = $request->title;

        $post->content = $request->content;

        $post->save();

        return redirect('posts');

    }

先前的$this->validate($request, []); 給刪除掉，引入創建的request檔案進行驗證

會產生與先前一樣的結果