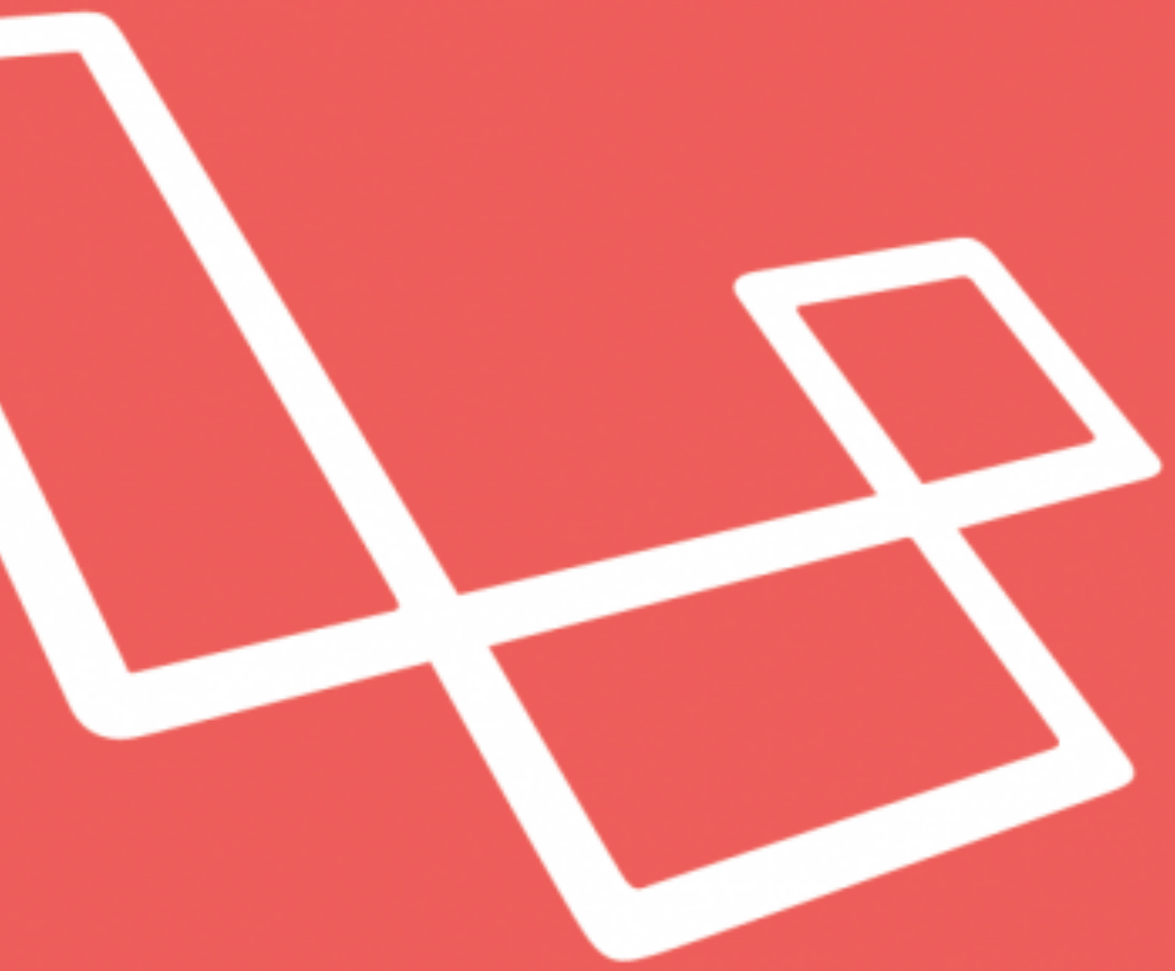


Laravel

Love beautiful code? We do too.

Forms & HTML

乾太 Kantai

2018/01/24

你還沒準備好！

那我們從頭開始。

伺服器需求

- PHP >= 5.5.9
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension

簡單一句話，去裝 XAMPP

https://www.apachefriends.org/zh_tw/index.html

然後去安裝 Composer

https://getcomposer.org

安裝 Laravel

你必須透過終端機來執行 Composer 指令。

透過 Composer 安裝 Laravel 有兩種安裝方式：

1. 先下載 Laravel 安裝包，再獨立安裝：

```
$> composer global require "laravel/installer"
$> laravel new {project-name}
```

2. 下載 Laravel 同時直接安裝：

```
$> composer create-project --prefer-dist laravel/laravel {project-name}
```

如果你是 Linux 或 macOS 作業系統，安裝 Laravel 之後，你必須設定一些權限。
`storage` 與 `bootstrap/cache` 目錄中的目錄必須讓你的伺服器有寫入權限。

```
$> chmod -R 777 {path}
```

現在你們準備好了！

那我們現在開始。

超好用 artisan 指令集(1/2)

啟動內置的 Laravel 開發伺服器

```
$> php artisan serve
```

列出所有的路由表

```
$> php artisan route:list
```

開啟與關閉維護模式

```
$> php artisan down
$> php artisan up
```

產生 Controller 範本

```
$> php artisan make:controller {controller-name}
```

產生乾淨 Controller 範本

```
$> php artisan make:controller {controller-name} --plain
```

產生 RESTful Controller 範本

```
$> php artisan make:controller {controller-name} --resource
```

超好用 artisan 指令集(2/2)

產生 Model 範本

```
$> php artisan make:model {model-name}
```

資料庫遷移

```
$> php artisan migrate
```

資料庫遷移及資料

```
$> php artisan migrate:refresh --seed
```

產生 Middleware 範本

```
$> php artisan make:middleware {activity-name}
```

產生 Jobs 範本

```
$> php artisan make:job {job-name}
```

產生 Requests 範本

```
$> php artisan make:request {request-name}
```

案例一：

我們想要寫一個 Form 表單，假如是讓使用者輸入天氣資訊的網頁好了，填寫資料並且送出後，把資料送到後台去處理，並且將結果儲存於資料庫當中。

分析：

會有一個 View 負責顯示 Form 表單。
會有一個 Controller 負責處理 Form 表單。
會有一個 Model 負責儲存資料。

http:// some-page

ROUTES

MODEL

CONTROLLER

VIEW

DATABASE

所以我們要先寫 Routing

Path: /routes/web.php

建立路由：

```
Route::method('{url}', 'ControllerName@ActionName');
```

範例：

```
// 首頁，撰寫天氣資訊的地方
Route::get('/weather', 'WeatherController@index');
// 表單傳入後台的路由
Route::post('/weather', 'WeatherController@store');
```

補充 Routing Method

Actions Handled By Resource Controller

Verb	Path	Action	Route Name
GET	`/photo`	index	photo.index
GET	`/photo/create`	create	photo.create
POST	`/photo`	store	photo.store
GET	`/photo/{photo}`	show	photo.show
GET	`/photo/{photo}/edit`	edit	photo.edit
PUT/PATCH	`/photo/{photo}`	update	photo.update
DELETE	`/photo/{photo}`	destroy	photo.destroy

```php
/* !! This is ‘/routes/web.php’.
 *
 * Frontend Routes
 * Namespaces indicate folder structure
 */
Route::group(['namespace' => 'Frontend', 'as' => 'frontend.'], function () {
    include_route_files(__DIR__.'/frontend/');
});
```

```php
/* !! This is ‘/routes/frontend/weather.php’.
 *
 * Frontend Access Controllers
 * All route names are prefixed with 'frontend.weather'
 */
Route::group(['namespace' => 'Weather', 'as' => 'weather.'], function () {
    /*
     * These routes require no user to be logged in
     */
    Route::group(['middleware' => 'guest'], function () {
        /*
         * Default home page
         */
        Route::get('/', 'WeatherController@index')->name('home');

        /*
         * Creating a new data routes
         */
        Route::post('/', 'WeatherController@store')->name('insert');
    });
});
```

http:// some-page

ROUTES

MODEL

CONTROLLER

VIEW

DATABASE

建立 Controller

Path: /app/Http/Controllers

你可以直接用 artisan 來減少你的工作量：

```
$> php artisan make:controller {controller-name} --resource
```

然後我們只需要把

```
public function index() { ... }
public function store() { ... }
```

這兩個 function 留下就好，其餘的註解掉。

這時候我們先開始寫 index()，我們要讓 Controller 回傳 View

```
public function index() {
    return view('weather.index');
}
```

Artisan-View

如果這時你以為，建立 View 應該也是下 artisan 指令，像是：

```
$> php artisan make:view {view-name}
```

那你就大錯特錯了。

因為 Laravel 還沒支援到這東西，但也別灰心！網上有人意識到這問題，立馬寫了這樣的東西出來！

https://github.com/svenluijten/artisan-view

只要安裝一下就可以使用上面的指令了。

```
$> composer require sven/artisan-view:^2.0 –dev
```

建立 View

然後你只要下個

```
$> php artisan make:view {view-name}
     --resource –extends={extend-name}
     --section=“title:{title-name}”
     --section={section-name}
```

或者

```
$> php artisan make:view {view-name}
```

就建立好 View 了。

Path: /resoures/views

建立 Form

```
{{ html()->form('POST', route('weather.store'))->open() }}
    <p>日期</p>
    {{ html()->text('date')
        ->placeholder('請輸入您的日期')
        ->required() }}

    <p>天氣</p>
    {{ html()->text('weather')
        ->placeholder('請輸入天氣狀況')
        ->attribute('maxlength', 8)
        ->required() }}

    {{ form_submit('送出') }}
{{ html()->form()->close() }}
```

```
{{ html()->form('POST', route('frontend.weather.store'))->open() }}
    <div class="row">
        <div class="col">
            <div class="form-group">
                {{ html()->label(__('validation.attributes.weather.date'))->for('date') }}

                {{ html()->text('date')
                    ->class('form-control')
                    ->placeholder(__('validation.attributes.weather.date'))
                    ->attribute('maxlength', 19) // ex: YYYY.MM.DD-hh:mm:ss
                    ->required() }}
            </div><!--form-group-->
        </div><!--col-->
    </div><!--row-->

    <div class="row">
        <div class="col">
            <div class="form-group">
                {{ html()->label(__('validation.attributes.weather.weather'))->for('weather') }}

                {{ html()->text('weather')
                    ->class('form-control')
                    ->placeholder(__('validation.attributes.weather.weather'))
                    ->attribute('maxlength', 8)
                    ->required() }}
            </div><!--form-group-->
        </div><!--col-->
    </div><!--row-->

    <div class="row">
        <div class="col">
            <div class="form-group clearfix">
                {{ form_submit(__('labels.frontend.weather.submit')) }}
            </div><!--form-group-->
        </div><!--col-->
    </div><!--row-->
{{ html()->form()->close() }}
```

http:// some-page

ROUTES

MODEL

CONTROLLER

VIEW

DATABASE

處理 Form 表單資料

Path: /app/Http/Controllers/WeatherController

```php
    public function store(Request $request) {
        // 取得 Request 的資料
        $input_date    = $request->input('date');
        $input_weather = $request->input('weather');

        // 寫入資料到資料庫
        // ... ???
    }
```

我們似乎缺了 Model 這東西。

建立 Model

Path: /app/Http/Models

你依舊可以直接用 artisan 來減少你的工作量：

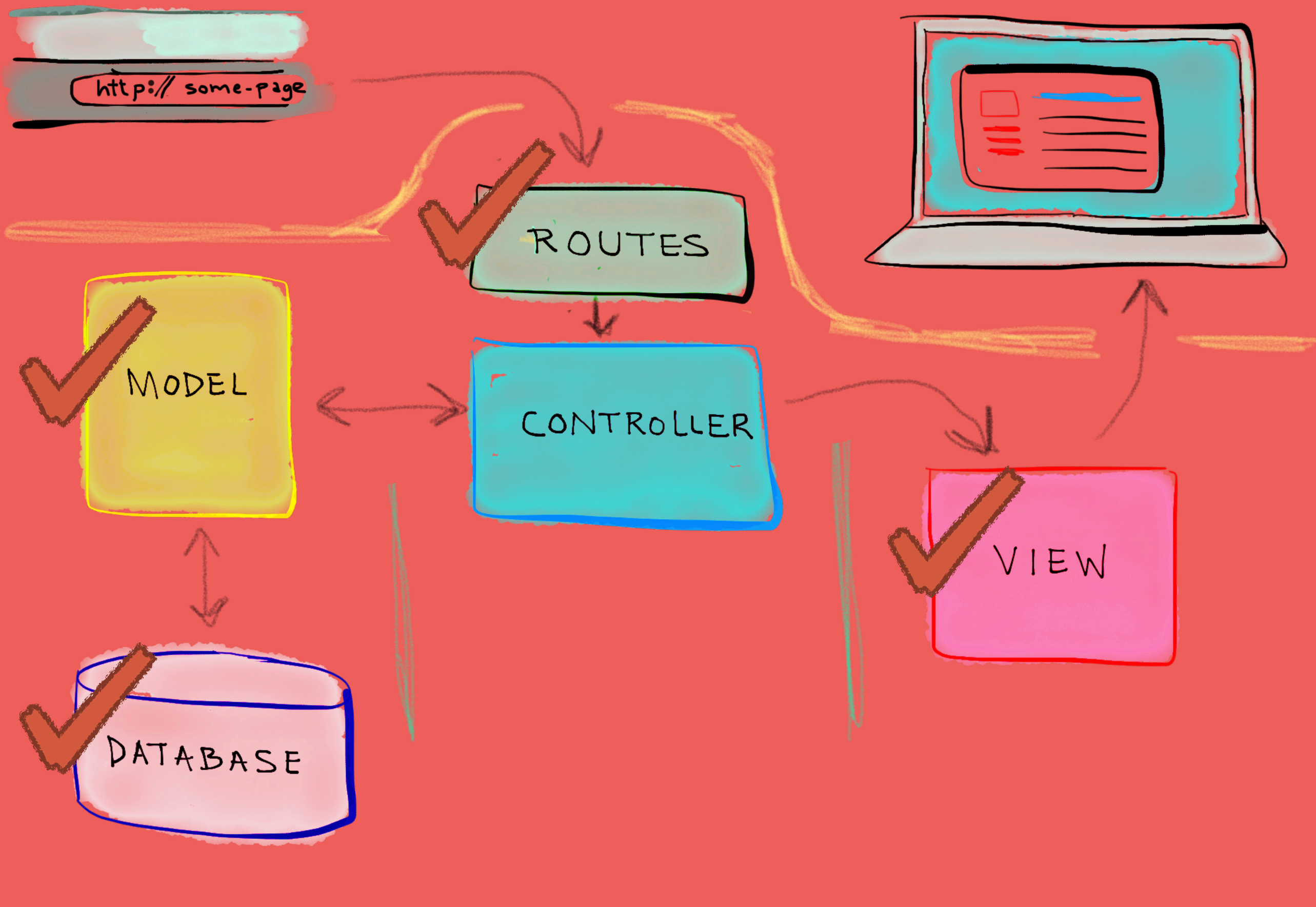
```
$> php artisan make:model {model-name}
```

我們只需要在 Model 當中設定幾些基本的資料即可，例如：

```
// (必填)與模型關聯的資料表。
protected $table    = 'your_table_name';

// (選填)資料表的主鍵。 @default = ‘id’
public $primaryKey = 'your_primary_key_name';

// (選填)時間戳記，自動寫入與更新 created_at 和 updated_at 欄位。
public $timestamps = Bool;
```


http:// some-page
ROUTES
MODEL
CONTROLLER
VIEW
DATABASE

繼續處理 Form 表單資料

Path: /app/Http/Controllers/WeatherController

```php
// 引入你的 Model
use App\Models\{model-name};

// ...

public function store(Request $request) {
    // 取得 Request 的資料
    $input_date    = $request->input('date');
    $input_weather = $request->input('weather');

    // 寫入資料到資料庫
    $model = new {model-name};
    $model->date    = $input_date;
    $model->weather = $input_weather;
    $model->save();

    // 將網頁重新導向
    return retirect()->route('weather.index');
}
```

```php
    /**
     * @param Request $request
     *
     * @return mixed
     */
    public function store(Request $request) {
        try {
            $__date           = $request->input('date');
            $__weather        = $request->input('weather');

            $weather          = new Weather;
            $weather->date    = $__date;
            $weather->weather = $__weather;
            $weather->save();

            return redirect()
                ->route('frontend.weather.index')
                ->withFlashSuccess(__('strings.weather.store.success'));
        } catch (LogicExcpetion $e) {
            return redirect()
                ->route('weather.index')
                ->withFlashDanger(__('strings.weather.store.logic_danger'));
        } catch (RuntimeException $e) {
            return redirect()
                ->route('weather.index')
                ->withFlashDanger(__('strings.weather.store.database_danger'));
        } catch (Exception $e) {
            return redirect()
                ->route('weather.index')
                ->withFlashDanger(__('strings.weather.store.other_danger'));
        }
    }
```

http:// some-page

ROUTES

MODEL

CONTROLLER

VIEW

DATABASE

那麼問題來了，剛剛好像有看到像是

```
...->withFlashSuccess()
...->withFlashDanger()
```

這些將後端處理的資訊傳入 View 當中，難道 Laravel 那麼厲害，連這些後端傳到前端的 Message 都會自動幫你處理？

```blade
@if ($errors->any())
    <div class="alert alert-danger" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>

        @foreach ($errors->all() as $error)
            {!! $error !!}<br/>
        @endforeach
    </div>
@elseif (session()->get('flash_success'))
    <div class="alert alert-success" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>

        @if(is_array(json_decode(session()->get('flash_success'), true)))
            {!! implode('', session()->get('flash_success')->all(':message<br/>')) !!}
        @else
            {!! session()->get('flash_success') !!}
        @endif
    </div>
@elseif (session()->get('flash_warning'))
    <div class="alert alert-warning" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>

        @if(is_array(json_decode(session()->get('flash_warning'), true)))
            {!! implode('', session()->get('flash_warning')->all(':message<br/>')) !!}
        @else
            {!! session()->get('flash_warning') !!}
        @endif
    </div>
@elseif (session()->get('flash_info'))
    <div class="alert alert-info" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>

        @if(is_array(json_decode(session()->get('flash_info'), true)))
            {!! implode('', session()->get('flash_info')->all(':message<br/>')) !!}
        @else
            {!! session()->get('flash_info') !!}
        @endif
    </div>
@elseif (session()->get('flash_danger'))
    <div class="alert alert-danger" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>

        @if(is_array(json_decode(session()->get('flash_danger'), true)))
            {!! implode('', session()->get('flash_danger')->all(':message<br/>')) !!}
        @else
            {!! session()->get('flash_danger') !!}
        @endif
    </div>
@elseif (session()->get('flash_message'))
    <div class="alert alert-info" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>

        @if(is_array(json_decode(session()->get('flash_message'), true)))
            {!! implode('', session()->get('flash_message')->all(':message<br/>')) !!}
        @else
            {!! session()->get('flash_message') !!}
        @endif
    </div>
```

答案是自幹啊！

案例一 結束

No machine was harmed.

案例二：

既然我們已經寫好了輸入天氣資料的網頁應用程式，那我們應該也需要一個顯示天氣列表的人機介面。

分析：

會有一個 View 負責顯示天氣列表(Blade 切版)。

會有一個 Controller 負責把 Model 拿去餵 View。

Routing

Path: /routes/web.php

首先，我們要為路由多加一個路徑：

```php
// 顯示天氣資訊列表的地方
Route::get('/weather/list', 'WeatherController@list');
```

Controller

Path: /app/Http/Controllers

我們需要為 WeatherController 多建立一個新的 Action
並且將資料庫的天氣資料，透過 Model 彙整起來丟到 View 當中：

```php
    /**
     * @return mixed
     */
    public function list() {
        $__weather = new Weather::all();
        return view(
          'weather.list',
          ['weathers' => $__weather]
        );
    }

    // 或者 return view('view-name')->with($response);
```

View

Path: /resoures/views

```blade
<table>
    <thead>
        <tr>
            <th>編號</th>
            <th>紀錄日期</th>
            <th>天氣狀況</th>
        </tr>
    </thead>
    <tbody>
        @foreach ($weaters as $weater)
            <tr>
                <td>{{ $weater->id }}</td>
                <td>{{ $weater->date }}</td>
                <td>{{ $weater->weater }}</td>
            </tr>
        @endforeach
    </tbody>
</table>
```

案例二 結束

No machine was harmed.