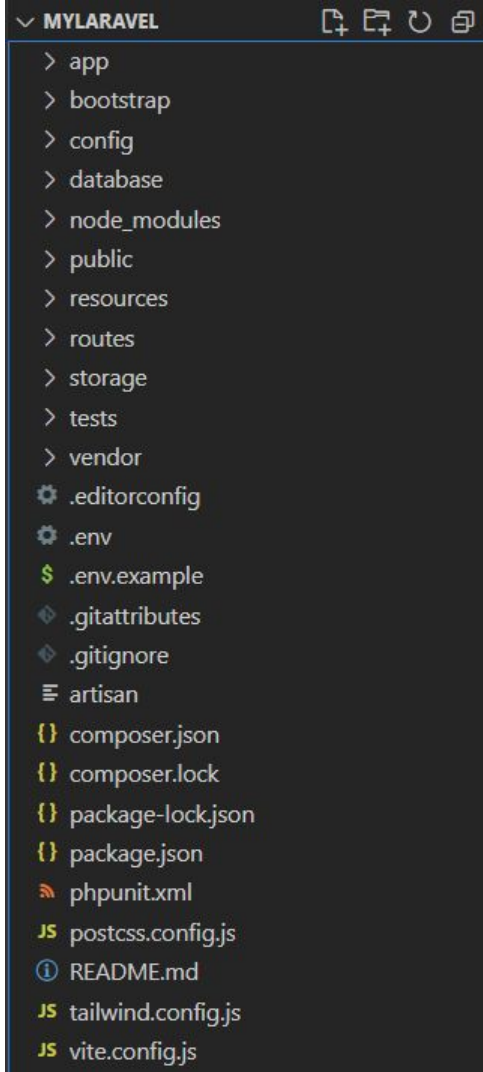


# Laravel

# Laravel 檔案結構說明

- App 目錄:應用程式的核心程式碼
- Bootstrap 目錄: 用來啟動框架和設定自動載入
- config 目錄: 所有應用程式的配置
- Database 目錄: 資料庫遷移與資料填入檔案
- Public 目錄:存放 index.php 檔案, 此檔案為 HTTP 請求的入口點。包含了前端資源檔(圖片、JavaScript、CSS)
- Resources 目錄:存放視圖、原始的資源檔 (LESS、SASS、CoffeeScript) 、語言檔



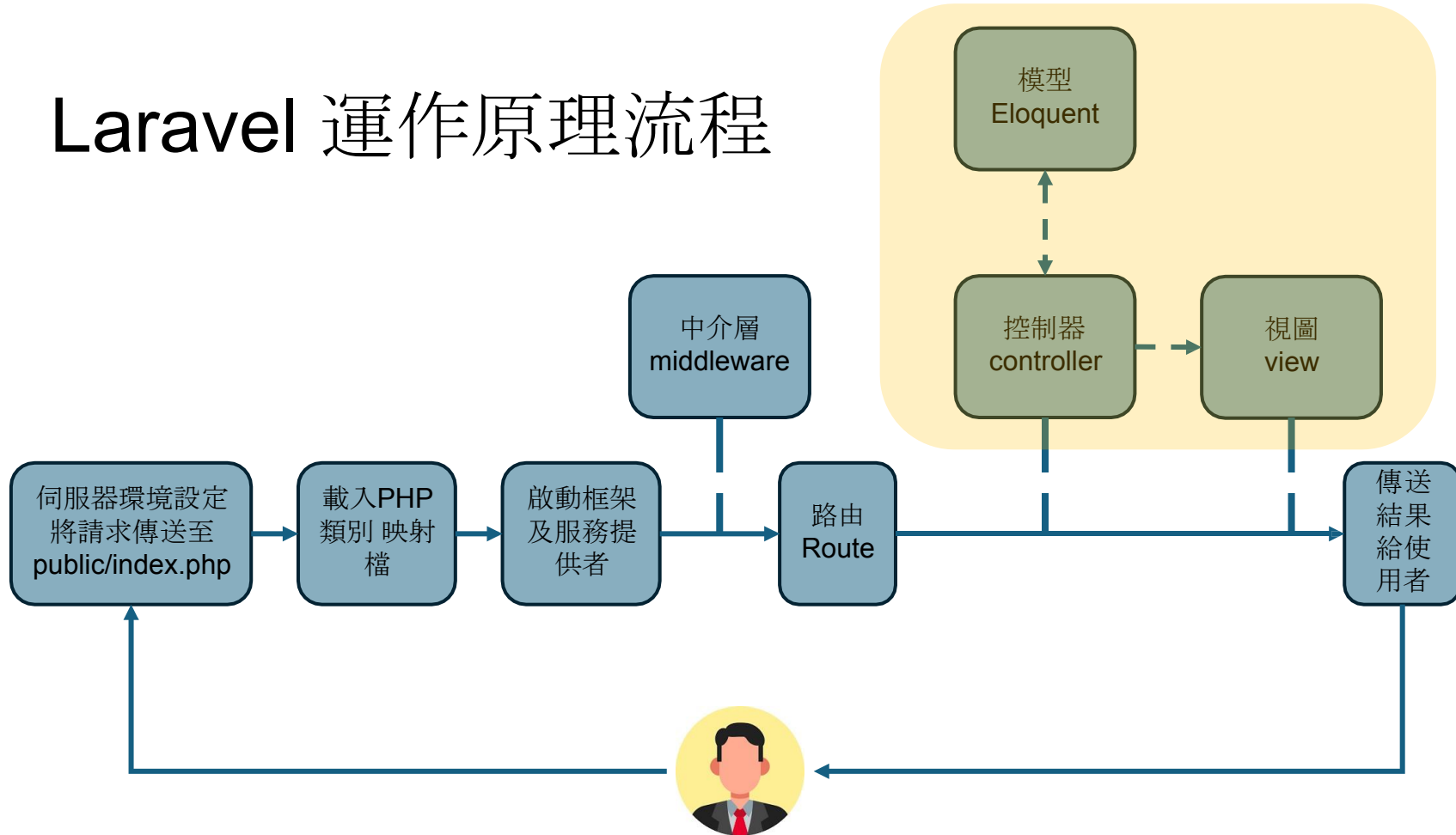
# Laravel 檔案結構說明

- **Routes** 目錄:包含應用的所有路由定義。Laravel 預設提供了三個路由檔案:auth.php、web.php、console.php (auth.php是原先的 api.php)
- **Storage** 目錄:編譯後的 **Blade** 模板、基於檔案的 session、檔案快取和其它框架生成的檔案。底下資料夾分隔成 app、framework, 及 logs 目錄。
  - app 目錄用於儲存應用程式使用的任何檔案
  - framework 目錄被用於儲存框架生成的檔案及快取
  - logs 目錄包含應用程式的日誌檔案。
- **Tests** 目錄: 自動化測試。並提供一個現成的 PHPUnit 範例
- **Vendor** 目錄: Composer 依賴模組

# 基本概念

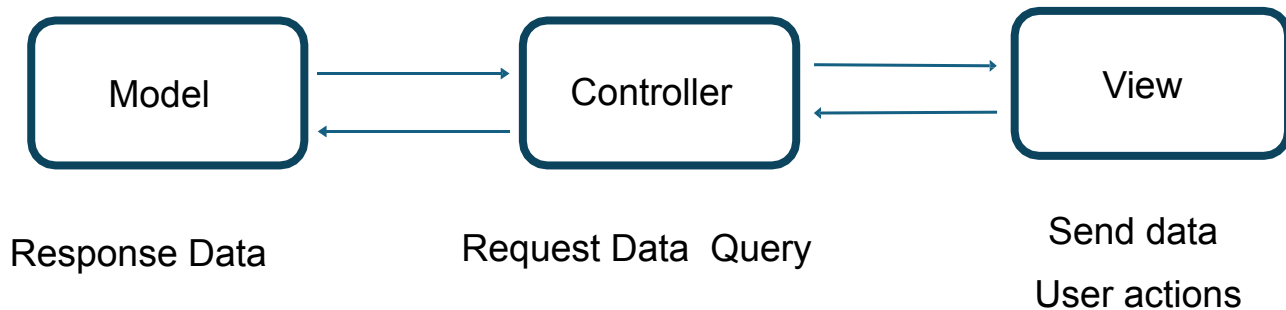
- **Artisan**是**Laravel**內建的命令列工具, 透過指令執行任務, 例如資料庫結構與資料的建立、檢視資料庫、檢查路由狀態、應用程式金鑰等等。
- 其基本使用方式
  - `php artisan` 指令 選項 參數
  - `php artisan list` 查看所有指令
  - `php artisan serve` 啟動伺服器
  - `php artisan cache:clear` 清除緩存
  - `php artisan migrate` 資料遷移
  - `php artisan make:(migration , model , controller)`

# Laravel 運作原理流程



# Model-View-Controller (MVC)架構

- MVC是一種軟體架構模式，把軟體系統分成三個部份
- 模型 (Model): 負責邏輯與資料處理
  - 封裝對資料的處理，主要是存取資料庫的操作
- 視圖 (View): 負責UI介面
  - 人機介面的溝通，藉由模型將資料以視覺方式呈現給使用者端
- 控制器 (Controller): 負責接收請求，協調M與V回應結果
  - 控制資料的處理流程，負責各項事件並做出相對回應



# MVC優勢

- 開發的過程中以「邏輯處理」與「資料呈現」分層處理, 明確的 區分各元件的功能, 可以提高系統的擴充性、可用性。
- 此外, 導入MVC更容易進行分工, 團隊每個人可以在各自負責的部份進行開發, 不會互相衝突或干擾。

## 優點

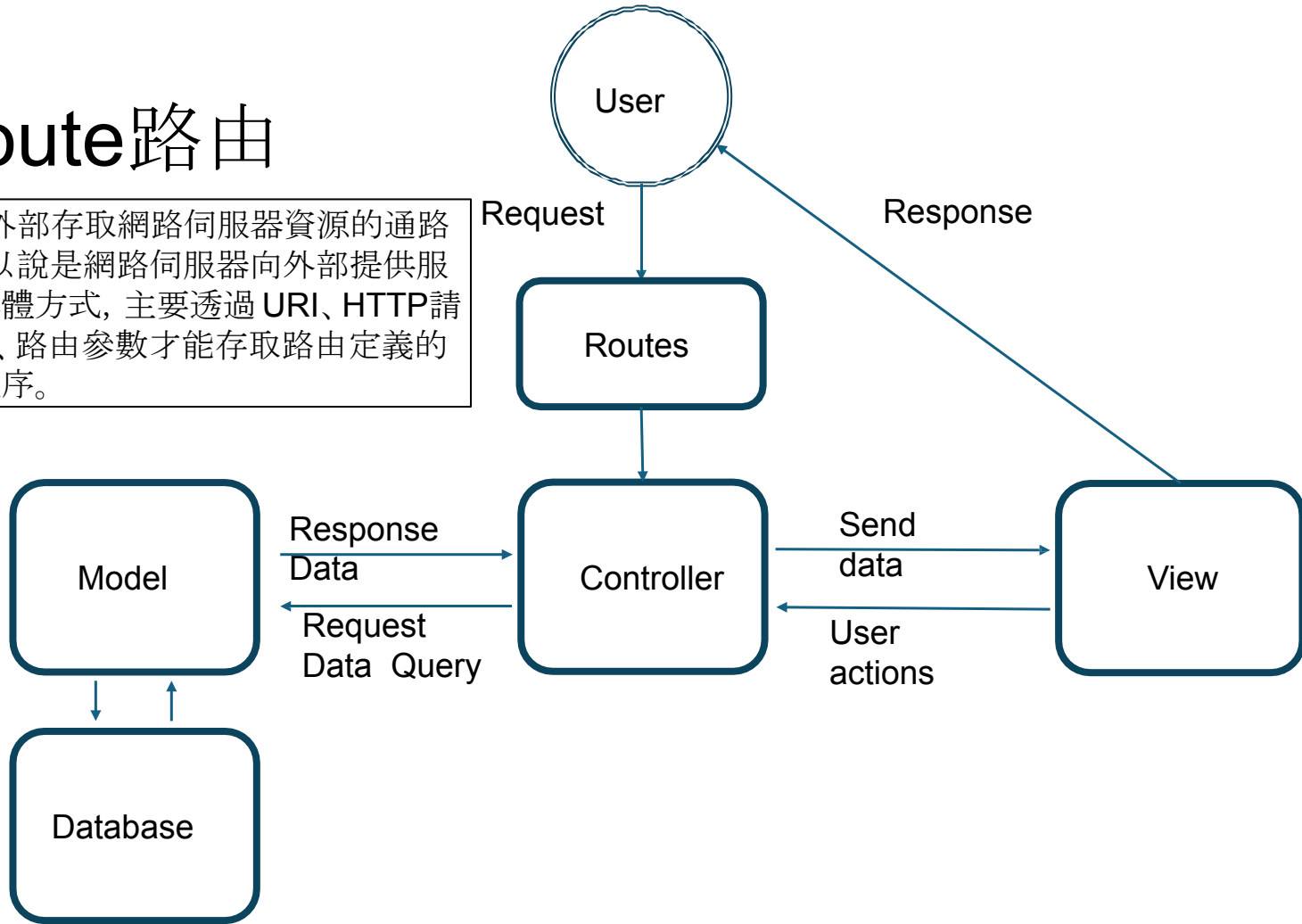
高擴充性 方便管理  
使程式結構更直覺  
有利於團隊分工

## 缺點

嚴謹系統規劃, 開發時間較長 不適合小專案  
系統資料較多, 所需效能較高

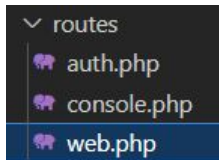
# Route路由

路由是外部存取網路伺服器資源的通路，也可以說是網路伺服器向外部提供服務的具體方式，主要透過 **URI**、**HTTP** 請求方法、路由參數才能存取路由定義的處理程序。





# 路由概念

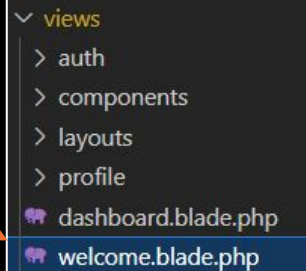


```
<?php
p
use
use Illuminate\Support\Facades\Route;
Route::get('/', function () {
    return view('welcome');
});
Route::get('/dashboard', function () {
    return view('dashboard');
})->middleware(['auth', 'verified'])->name('dashboard');
Route::middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'show']);
    Route::patch('/profile', [ProfileController::class, 'update']);
    Route::delete('/profile', [ProfileController::class, 'destroy']);
});
require __DIR__.'/auth.php';
```

定義路由

首頁路由

傳送welcome視圖



儀錶板路由



[Log in](#) [Register](#)



Search

VERSION



## Documentation

Laravel has wonderful documentation covering every aspect of the framework. Whether you are a newcomer or have prior experience with Laravel, we recommend reading our documentation from beginning to end. →



## Laracasts

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process. →



## Laravel News

Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials. →



## Vibrant Ecosystem

Laravel's robust library of first-party tools and libraries, such as [Forge](#), [Vapor](#), [Nova](#), [Envoyer](#), and [Herd](#) help you take your projects to the next level. Pair them with powerful open source libraries like [Cashier](#), [Dusk](#), [Echo](#), [Horizon](#), [Sanctum](#), [Telescope](#), and more.

# 簡單修改welcome視圖

```
18 <body class="font-sans antialiased dark:bg-black dark:text-white/50">
19   <div class="bg-gray-50 text-black/50 dark:bg-black dark:text-white/50">
20     
21     <div class="relative min-h-screen flex flex-col items-center justify-center selection:bg-[#FF2D2D] selection:text-white">
22       <div class="relative w-full max-w-2xl px-6 lg:max-w-7xl">
23         <header class="grid grid-cols-2 items-center gap-2 py-10 lg:grid-cols-3">
24           .....<div class="flex lg:justify-center lg:col-start-2">
25             <svg class="h-12 w-auto text-white lg:h-16 lg:text-[#FF2D2D]" viewBox="0 0 62 65" fill="none" xmlns="http://www.w3
```

```
18 <body class="font-sans antialiased dark:bg-black dark:text-white/50">
19   <div class="bg-gray-50 text-black/50 dark:bg-black dark:text-white/50">
20     
21     <div class="relative min-h-screen flex flex-col items-center justify-center selection:bg-[#FF2D2D] selection:text-white">
22       <div class="relative w-full max-w-2xl px-6 lg:max-w-7xl">
23         <header class="grid grid-cols-2 items-center gap-2 py-10 lg:grid-cols-3">
24           <h1>Welcome My Laravel</h1>
25         <div class="flex lg:justify-center lg:col-start-2">
26           <svg class="h-12 w-auto text-white lg:h-16 lg:text-[#FF2D2D]" viewBox="0 0 62 65" fill="none" xmlns="http://www.w3
```

增加這一行而已

Welcome My Laravel



[Log in](#) [Register](#)



Search

VERSIONS



## Documentation

Laravel has wonderful documentation covering every aspect of the framework. Whether you are a newcomer or have prior experience with Laravel, we recommend reading our documentation from beginning to end. →



## Laracasts

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process. →



## Laravel News

Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials. →



## Vibrant Ecosystem

Laravel's robust library of first-party tools and libraries, such as [Forge](#), [Vapor](#), [Nova](#), [Envoyer](#), and [Herd](#) help you take your projects to the next level. Pair them with powerful open source libraries like [Cashier](#), [Dusk](#), [Echo](#), [Horizon](#), [Sanctum](#), [Telescope](#), and more.

# Route 路由

管理並設定使用者的請求

# 路由概念

- 路由扮演入口的角色，根據使用者指定入口，進而傳送視圖或檔案或服務。
- 大部分路由是`routes/web.php`定義，也可以自記擴充自訂，但需要與`app`綁定連結，才能生效。
- 基礎路由

```
Route::get('/', function () {  
    return view('welcome');  
});
```
- `Route`是路由物件，`get`是http網路協議中取得資料的方法，`return`是回傳`welcome`視圖

# 路由概念 - http方法與路由動作

- http方法
  - GET:取得資料
  - POST:新增一筆資料
  - PUT:新增一筆資料, 會覆蓋原先存在資料
  - PATCH:在現存資料再附加新的資料
  - DELETE:刪除資料

```
Route::get($uri, $callback);  
Route::post($uri, $callback);  
Route::put($uri, $callback);  
Route::patch($uri,  
$callback);  
Route::delete($uri,  
$callback);
```

# 路由 其餘介紹

- match方法, 在同一路由限定或加入不同HTTP方法

```
Route::match(['get', 'post'], '/', function(){  
    // 此路由可同時接收get與post方法, 也可看成僅接收get和post方法  
});
```

- any方法

```
Route::any('/', function(){  
    // 此路由接受全部HTTP方法  
});
```



# 路由參數

- 比如網址uri設計如下
  - students/[學號]
  - students/[學號]/information

- 路由設定web.php

- 新增

```
Route::get('students/{sid}', function ($sid) {  
    return '學號: ' . $sid;  
});
```

- 參數使用{ }刮起來



# 選擇性路由參數

- 若定義的路由參數不是必填的情況下，只需要在參數後面加上問號即可。

```
Route::get('students/{sid}/score/{subject?}', function ($sid, $subject = null) {  
    return '學號: ' . $sid . '的' . (is_null($subject)) ? '全部科目' : $subject . '分數'  
    '  
});
```

- 請注意使用選擇性路由參數，後方接收變數的預設值需要設定，此範例預設值設計為null，這樣方面後面直接使用is\_null進行判斷

# 使用正規表達式客製化參數形式

```
// 使用正規表達式客製化參數形式
```

```
Route::get('students/{sid}', function ($sid) {  
    return '學號: ' .  
}) -> where(['sid' =>
```

```
's[0-9]{10}']);
```

- 設定參數規則為10位數，開頭為s，後面為10個數字
  - <http://127.0.0.1:8000/students/s1101234567> 可正常顯示
  - <http://127.0.0.1:8000/students/s110123456> 404 NOT FOUND
  - <http://127.0.0.1:8000/students/a1101234567> 404 NOT FOUND

- 若是科目則寫成，請注意( [ 條件a, 條件b ])

```
Route::get('students/{sid}', function ($sid)  
{ return '學號: ' . $sid;  
}) -> where(['sid' => 's[0-9]{10}', 'subject' =>  
'(chinese|mathematics|english|physics)']);
```

# 路由群組

- 群組化的目的是共同這些資料的部分屬性部分，可以減少繁複過程。
- 通常應用於路由前綴(prefix)、控制器命名空間、中介層等等，是否要使用群組化端看個人撰寫習慣。

以下重複'student'，將其提出

```
// 使用正規表達式客製化參數形式
Route::get('student/{sid}', function ($sid)
{
    return '學號: ' . $sid;
})->where(['sid' => 's[0-9]{10}']);

// 使用正規表達式客製化參數形式
Route::get('students/{sid}/score/{subject?}', function ($sid, $subject = null) {
})->where(['sid' => 's[0-9]{10}', 'subject' =>
    '(chinese|mathematics|english|physics)');
});
```

# 路由群組 範例改寫

```
// 群組化students格式
Route::pattern('sid', 's[0-9]{10}');
Route::group(['prefix' => 'students'], function () {
    { Route::get('{sid}', function ($sid) {
        return '學號: ' . $sid;
    });
    Route::get('{sid}/score/{subject?}', function ($sid, $subject = null) {
        return '學號: ' . $sid . '的' . (is_null($subject)) ? '全部科目' : $subject . '分數';
    });
});
});
```

# 查看路由表

- 當專案到一定規模後，其路由會越來越複雜，可以透過artisan的指令查看當前所有路由的定義以及HTTP方法。
- 指令 `php artisan route:list`

```
GET|HEAD students/students/{sid}/score/{subject?}
GET|HEAD students/{sid} .....
GET|HEAD students/{sid}/score .....
```

```
PS C:\xampp\htdocs\myLaravel> php artisan route:list
```

```
GET|HEAD / .....
POST _ignition/execute-solution ..... ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
GET|HEAD _ignition/health-check ..... ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST _ignition/update-config ..... ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET|HEAD confirm-password ..... password.confirm > Auth\ConfirmablePasswordController@show
POST confirm-password ..... Auth\ConfirmablePasswordController@store
GET|HEAD dashboard ..... dashboard
POST email/verification-notification ..... verification.send > Auth\EmailVerificationNotificationController@store
POST forgot-password ..... password.request > Auth\PasswordResetLinkController@create
POST forgot-password ..... password.email > Auth\PasswordResetLinkController@store
GET|HEAD login ..... login > Auth\AuthenticatedSessionController@create
POST login ..... Auth\AuthenticatedSessionController@store
POST logout ..... logout > Auth\AuthenticatedSessionController@destroy
PUT password ..... password.update > Auth\PasswordController@update
GET|HEAD profile ..... profile.edit > ProfileController@edit
PATCH profile ..... profile.update > ProfileController@update
DELETE profile ..... profile.destroy > ProfileController@destroy
GET|HEAD register ..... register > Auth\RegisteredUserController@create
POST register ..... Auth\RegisteredUserController@store
POST reset-password ..... password.store > Auth\NewPasswordController@store
GET|HEAD reset-password/{token} ..... password.reset > Auth\NewPasswordController@create
GET|HEAD students/students/{sid}/score/{subject?} .....
GET|HEAD students/{sid} .....
GET|HEAD students/{sid}/score .....
GET|HEAD up .....
GET|HEAD verify-email ..... verification.notice > Auth\EmailVerificationPromptController
GET|HEAD verify-email/{id}/{hash} ..... verification.verify > Auth\VerifyEmailController
```

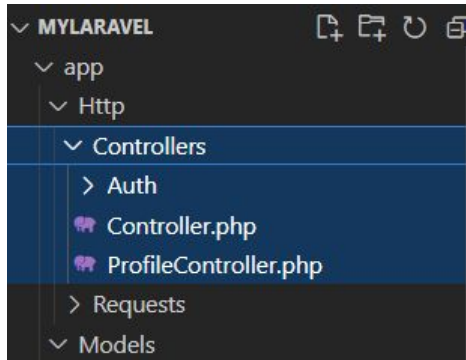
Showing [27] routes

# Controller 控制器

處理程式流程

# 建立控制器controller

- 目前專案已經事先建立好各種控制器

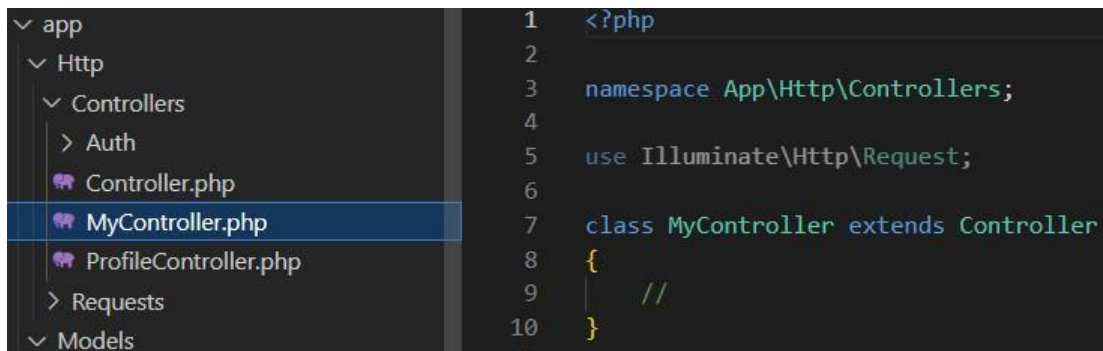


- 自己建立可以使用指令, **artisan make:controller** 名稱
  - 例如
    - php artisan make:controller IndexController
    - php artisan make:controller MyController



```
PS C:\xampp\htdocs\myLaravel> php artisan make:controller MyController
```

```
INFO Controller [C:\xampp\htdocs\myLaravel\app\Http\Controllers\MyController.php] created successfully.
```



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class MyController extends Controller
8 {
9     //
10 }
```

# 路由與控制器關係

- 路由相對應要執行的動作，改交由控制器來負責執行，可以簡化路由的寫法以及管理。
- 現在我們改寫先前首頁路由的寫法
- web.php

web.php

原先

```
// 路由參數
Route::get('students/{sid}/score', function ($sid)
{
    return '學號: ' . $sid . '總分是';
});
```

修改後

```
<?php
namespace
use
use
use
use Illuminate\Support\Facades\Route;
// Route::get('/', function ()
//     return view('welcome');
//
//
//
// 改寫
Route::get('/', [MyController::class,
    'index']);
```

index是MyController類別裡面的一個叫做index的方法

MyController.php

```
<?php
namespace
App\Http\Controllers:
use
Illuminate\Http\Request:
class MyController extends
{
    //將路由首頁的執行動作放在這
    public function
    index()
    {
        return
    }
    view('welcome');
}
```

# 練習建立StudentsController

- 指令 `php artisan make:controller StudentsController`

StudentsController.php

```
<?php

namespace
App\Http\Controllers;
use
Illuminate\Http\Request;
class StudentsController extends
{
    // 撰寫兩個取得學生學號和取得成績資料兩個方
    public function getStudentsID($sid)
        return '學號: ' . $sid;
    }
    public function getStudentsScore($sid, $subject = null)
        return '學號: ' . $sid . '的' . ((is_null($subject) ? '全部科目' : $subject) . '分數
    }
};
}
```

# 練習建立StudentsController

web.php

原先

```
// 群組化students格式 --- 沒有控制器
Route::pattern('sid', 's[0-9]{10}');
Route::group(['prefix' => 'students'], function () {
    Route::get('{sid}', function ($sid) {
        return '學號: ' . $sid;
    });
    Route::get('{sid}/score/{subject?}', function ($sid, $subject = null) {
        return '學號: ' . $sid . '的' . (is_null($subject)) ? '全部科目' : $subject . '分數';
    }->where(['subject' => '(chinese|mathematics|english|physics)']));
});
```

web.php

修改後

```
// 群組化 --- 使用控制器
Route::group(['prefix' => 'students'], function () {
    Route::get('{sid}', [
        'as' => 'students',
        'uses' => [StudentsController::class, 'getStudentsID']
    ])
    Route::get('{sid}/score/{subject?}', [
        'as' => 'students.score',
        'uses' => [StudentsController::class, 'getStudentsScore']
    ]->where(['subject' => '(chinese|mathematics|english|physics)']));
});
```

as是對路由作命名  
uses使用轉移給要使用的控制器

# 路由命名也可以是用name()

```
Route::group(['prefix' => 'students'], function () {  
    Route::get('{sid}', [StudentsController::class, 'getStudentsID'])  
        ->name('students');  
    Route::get('{sid}/score/{subject?}', [StudentsController::class, 'getStudentsScore'])  
        ->where(['subject' => '(chinese|mathematics|english|physics)'])  
        ->name('students.score');  
});
```

- 别忘了加上使用

```
1  <?php  
2  
3  use App\Http\Controllers\ProfileController;  
4  use App\Http\Controllers\MyController;  
5  use App\Http\Controllers\StudentsController;  
6  use Illuminate\Support\Facades\Route;
```

# 測試



- 雖然有無使用控制，結果都是相同，但這樣的分開控制寫法，可以將路由、控制器、執行內容三者之間的職責與彼此之間的關係。
- 藉由分工模組可以劃分每個角色及其職權，可以讓我們更容易管理每個路由後續到底要執行什麼內容，以及如何處理後續資料。



資源控制器

Resource Controller

# 基本概念

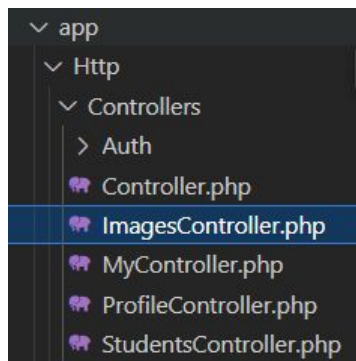
- 我們想要建立一個照片控制器，其基本功能具有新增、瀏覽、編輯、刪除四大功能，也就是**CRUD**。
- **Laravel**針對新增、瀏覽、編輯三功能各別分切成兩個步驟執行，總共具有七個動作
  - Index
  - Create
  - Store
  - Show
  - Edit
  - Update
  - Delete (destroy)

# 建立資源控制器

- 指令
  - `php artisan make:controller ImagesController --resource`
- 此指令可以快速建立7個動作的控制器

```
PS C:\xampp\htdocs\myLaravel> php artisan make:controller ImagesController --resource
```

```
INFO Controller [C:\xampp\htdocs\myLaravel\app\Http\Controllers\ImagesController.php] created successfully.
```



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ImagesController extends Controller
8 {
9     /**
10      * Display a listing of the resource.
11      */
12     public function index()
13     {
14         //
15     }
16
17     /**
18      * Show the form for creating a new resource.
19      */
20     public function create()
21     {
22         //
23     }
24
25     /**
26      * Store a newly created resource in storage.
27      */
28     public function store(Request $request)
29     {
30         //
31     }
32
33     /**
34      * Display the specified resource.
35      */
36     public function show(string $id)
37     {
38         //
39     }
40
41     /**
42      * Show the form for editing the specified resource.
43      */
44     public function edit(string $id)
45     {
46         //
47     }
48
49     /**
50      * Update the specified resource in storage.
51      */
52     public function update(Request $request, string $id)
53     {
54         //
55     }
56
57     /**
58      * Remove the specified resource from storage.
59      */
60     public function destroy(string $id)
61     {
62         //
63     }
64 }
```

# 根據控制器編寫路由

```
// ImagesController 建立路由, 群組化基本寫法
Route::prefix('images')->group(function () {
    Route::get('/', 'ImagesController@index')->name('images.index');
    Route::get('/create', 'ImagesController@create')->name('images.create');
    Route::post('/', 'ImagesController@store')->name('images.store');
    Route::get('/{image}', 'ImagesController@show')->name('images.show');
    Route::get('/{image}/edit', 'ImagesController@edit')->name('images.edit');
    Route::match(['put', 'patch'], '/{image}', 'ImagesController@update')->name('images.update');
    Route::delete('/{image}', 'ImagesController@destroy');
});
```

- 寫完非常冗長且有點繁雜, 簡單寫法如下

```
Route::resource('images', 'ImagesController');
```

# 資源控制器提供的七個路由

```
Route::resource('images', 'ImagesController');
```

HTTP方法	URL	控制器方法	對映路由的自訂名稱
GET	images	index()	images.index
GET	images/create	create()	images.create
POST	images	store()	images.store
GET	images/{image}	show()	images.show
GET	images/{image}/edit	edit()	images.edit
PUT/PATCH	images/{image}	update()	images.update
DELETE	images/{image}	destroy()	images.destroy