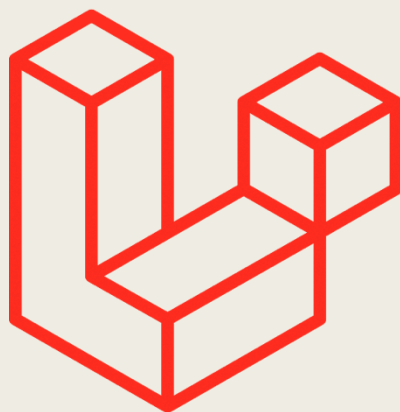


# 起手式

朱克剛



# 何謂 Laravel

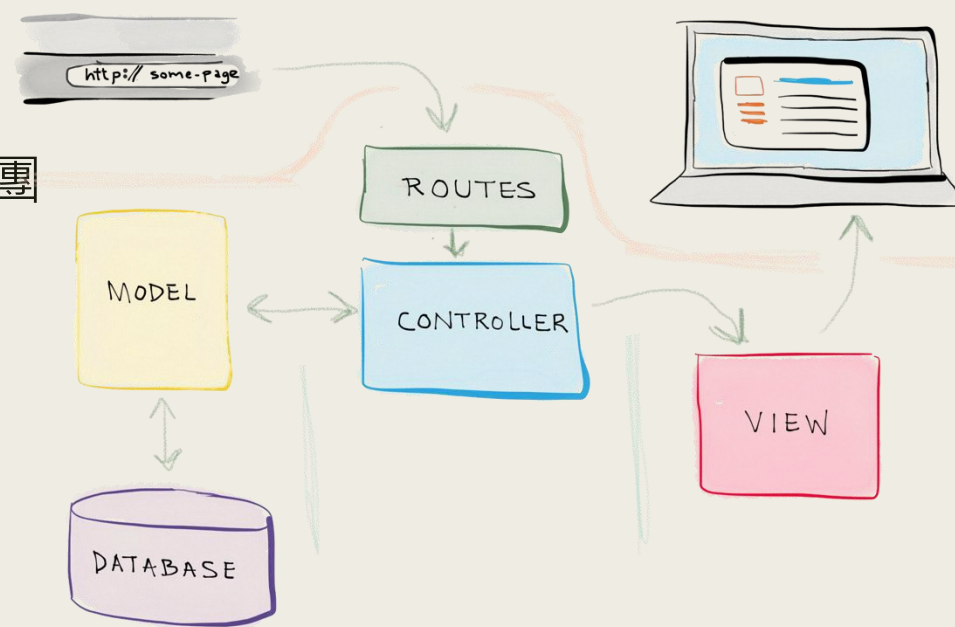
- 開源的 PHP 應用程式框架，2011 年釋出。
- 開發人員可以更快速地建立 Web 應用程式，例如路由、資料庫存取、視圖和控制器...等。
- Laravel 的設計重點在於簡潔、優雅，並提供開發人員開發高品質 Web 應用程式的便利性。

# 安裝

- Composer 是 php 的套件管理程式
  - 安裝方式：
    - <https://getcomposer.org/download/>
- 安裝 Laravel
  - 安裝方式：
    - `composer global require "laravel/installer"`
    - Windows 需要在 `php.ini` 中開啟 `zip extension`
    - 設定環境變數 `PATH`
      - 將 `.composer/vendor/bin` 加到 `PATH` 中
      - Windows 在 `AppData/Roaming/Composer` 中
      - Mac 在 `$HOME/.composer` 中
- 建立 Laravel 專案
  - `laravel new HelloLaravel`
- Mac / UNIX 需要修改專案下特定檔案具可寫入權限
  - `sudo chown -R daemon storage bootstrap/cache database`

# MVC

- MVC 全名為 Model-View-Controller，屬於軟體設計模型中的一種常見架構
- Model：負責資料處理與運算
- View：負責畫面顯示
- Controller：控制 View 與 Model 的運作
- 優點：程式架構清楚，各司其職，程式不會亂成一團
- 缺點：需有物件導向程式概念，學習門檻高



# ROUTER

# 設定路由

- 目的：決定哪個路徑 ( `http://hostname/PATH` ) 要做什麼事情
- 開啟 `routes/web.php` [在local端的資料夾裡找資料夾位置](#)
- 加上

```
Route::get('/hello', function() {  
    return 'Hello, Laravel!'  
});
```

- 網址輸入
  - `http://localhost/[專案名稱]/public/hello`

```
Route::get($uri, $callback);  
Route::post($uri, $callback);  
Route::put($uri, $callback);  
Route::patch($uri, $callback);  
Route::delete($uri, $callback);  
Route::options($uri, $callback);  
Route::any($uri, $callback);  
  
Route::match(['GET', 'POST'], $uri, $callback);
```

# 路由中的參數

## ■ 試試這個路由

```
Route::get('/hello/{name?}', function($name='Laravel') {  
    return 'Hello, '.$name;  
});
```

## ■ 網址試試這兩個

- `http://localhost/[專案名稱]/public/hello`
- `http://localhost/[專案名稱]/public/hello/World`

# 在路由端轉到另外一個路由

- `Route::redirect('/here', '/there');`
- `Route::redirect('/apple', 'https://www.apple.com');`



# 命名

- 可以給路由取一個名字

```
Route::get('/', function () {  
    return view('welcome');  
})->name('home');
```

# 進到路由後轉到另外一個路由

## ■ 作法一：

```
use Illuminate\Support\Facades\Redirect;  
return Redirect::route('路由名稱');
```

## ■ 作法二：

```
return redirect()->intended('路由設定的第一個參數');  
return redirect()->intended(route('路由名稱'));
```

## ■ 作法三：

```
return to_route('路由名稱');
```



# VIEW

# 建立 View

- View 就是前端網頁，網頁內容原則上不應該有後端 PHP 程式碼，位置在 `resources/view` 目錄中，檔名結尾必須是 `.blade.php`
- 現在建立一個簡單的 View，名稱為 `home`，使用 `artisan` 建立
  - `php artisan make:view home`

- 設定路由

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



- 試試網址

- `http://localhost/[專案名稱]/public/home`

```
Route::view('/home', 'home');
```

# 傳資料到 View

## ■ 修改路由

```
Route::get('/home/{str}', function($str) {  
    return view('home', ['data' => $str]);  
});
```



另一種語法

## ■ 修改 View

```
<body>  
    Hi {{ $data }}  
</body>
```

```
Route::get('/home/{str}', function($str) {  
    return view('home'  
        ->with('data', $str);  
});
```

# CONTROLLER

# 建立 Controller

- Controller 用來控制 View，並且回應使用者在 View 上產生的需求
- 建立方式建議使用 artisan 指令，Controller 與 View 的名稱建議一致
  - *php artisan make:controller HomeController*
- 會在 app/Http/Controllers 目錄下會產生 HomeController.php 檔案，自行新增 \_\_invoke() 函數。

```
class HomeController extends Controller
{
    public function __invoke($str) {
        return view('home', ['data' => $str]);
    }
}
```

# 連接路由與 Controller

- 記得在 web.php 中加上 namespace 否則會找不到指定的 Controller

```
namespace App\Http\Controllers;  
Route::get('/home/{str}', HomeController::class);
```

呼叫\_\_invoke()

- 連結到 Controller 中指定的函數，例如 redirect()

```
Route::get('/to/{id}', [HomeController::class, 'redirect']);
```

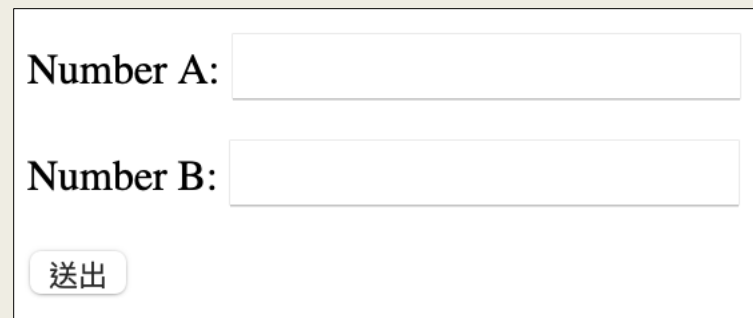
```
public function redirect($urlId) {  
    switch ($urlId) {  
        case 1:  
            $url = 'https://www.google.com'; break;  
        case 2:  
            $url = 'https://tw.yahoo.com'; break;  
        default:  
            $url = 'https://www.chainhao.com.tw';  
    }  
    return redirect()->away($url);  
}
```



# 表單

- 設計一個表單，檔名為 form.blade.php

```
<form method="post">  
    @csrf  
    <p><label>Number A: </label><input name="a"></p>  
    <p><label>Number B: </label><input name="b"></p>  
    <input type="submit">  
</form>
```



Number A:

Number B:

# FormController

- php artisan make:controller FormController

```
class FormController extends Controller
{
    public function action(Request $request) {
        $a = $request->a;
        $b = $request->b;
        $answer = $a + $b;
        return view('answer', [
            'a' => $a,
            'b' => $b,
            'answer' => $answer
        ]);
    }
}
```

# MODEL

# 建立 Model

```
class MyModel extends Model
{
    use HasFactory;

    public function add($request) {
        $a = $request->a;
        $b = $request->b;
        return [
            'a' => $a,
            'b' => $b,
            'answer' => $a + $b
        ];
    }
}
```

- 模型用來處理資料，不涉及任何使用者介面，例如資料庫存取
- 使用 artisan 快速產生 model 檔，位置在 app/Models 目錄下
  - *php artisan make:model MyModel*
- 自訂一個函數 add()

# 在 Controller 中使用 Model

- Controller 將複雜的運算交給 Model 去處理，處理完再將結果丟給 View 去呈現

```
<!-- 此為 result.blade.php -->
<body>
{{ $a }} + {{ $b }} = {{ $answer }}
</body>
```

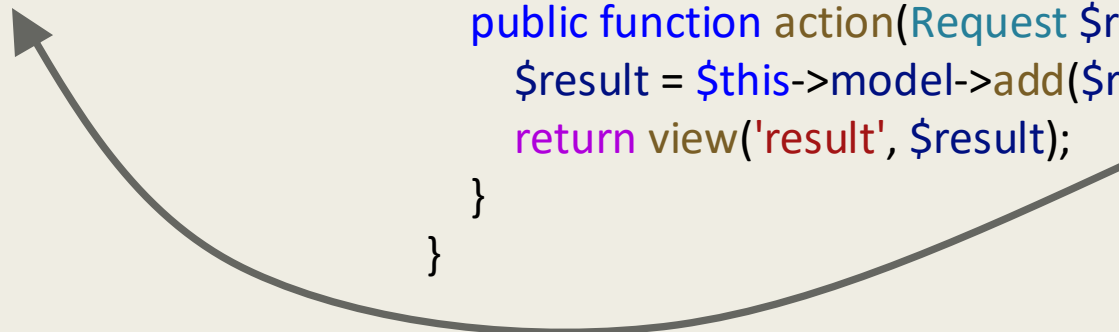
```
use App\Models\MyModel;
```

```
class FormController extends Controller
{
```

```
    private $model;
```

```
    function __construct() {
        $this->model = new MyModel();
    }
```

```
    public function action(Request $request) {
        $result = $this->model->add($request);
        return view('result', $result);
    }
}
```



# 路由設定

```
Route::view('/form', 'form');  
Route::post('/form', [FormController::class, 'action']);
```

# MIDDLEWARE

# 用途

- 在 http request 進入指定的路由前先做一些事情，例如檢查前端傳過來的資料是否正確
- 產生 middleware 指令，會在 app/Http/Middleware 目錄中產生 MyCheck.php
  - `php artisan make:middleware MyCheck`
- 修改 bootstrap/app.php
  - `$middleware->append(MyCheck::class);`
- 若不想修改 app.php，在路由中使用時加上
  - `use App\Http\Middleware\MyCheck;`



# MyCheck 內容

- 檢查 token 的第三字元是否為 5

```
public function handle(Request $request, Closure $next): Response
{
    if ($request->token[2] != 5) {
        die("error");
    }
    return $next($request);
}
```

# 修改路由

- 載入 form 之前先呼叫中介層 check 做一些處理

```
Route::view('/main', 'main')->middleware('check');
```

或者

```
Route::view('/main', 'main')->middleware(MyCheck::class);
```

- 輸入網址
  - `http://localhost/laravel/public/main?token=1234`

# BLADE模版

# 顯示PHP的變數內容

- 原有語法
  - `<?php echo $data; ?>`
- Blade 語法
  - `{{ $data }}`

# 條件判斷

```
Route::get("/main/{n}", function($n) {  
    return view('main', ['n' => $n]);  
});
```



<http://localhost/laravel/public/main/1>

```
<body>  
@if ($n == 1)  
    Hello, 1  
@elseif ($n == 0)  
    Hello, 0  
@else  
    Others  
@endif  
</body>
```

# 迴圈

## ■ For 迴圈

```
<ul>  
@for ($i = 0; $i < 10; $i++)  
    <li> {{ $i }} </li>  
@endfor  
</ul>
```

## ■ Foreach 迴圈

```
@foreach ($arr as $el)  
    <p>{{ $loop->index }} : {{ $el }}</p>  
@endforeach
```

# PHP 區段

- 多行時

```
@php  
    $counter = 1;  
@endphp
```

- 只有一行時

```
@php($counter = 1)
```

# 表單

- 防止跨域攻擊 ( 看到 419 Page Expired 錯誤時 )

```
<form ...>  
    @csrf  
</form>
```

- 額外設定 PUT、DELETE...隱藏欄位

```
<form action="do" method="post">  
    @method('PUT')  
</form>
```



其他

# 跳轉到一般網頁

- 在 public 目錄中有一個 test.html 檔案，此檔案並非屬於 Laravel 架構中的 View
- 如要在路由中設定跳轉到該網頁，設定如下

```
use Illuminate\Support\Facades\File;

Route::get('/test', function() {
    return File::get(public_path() . '/test.php');
});
```

# 跳轉到外部網站

- 在 Router 或 Controller 中可以跳轉到外部網站

```
Route::get("/go", function() {  
    return redirect()->away("https://www.google.com");  
});
```