

# Route 路由控制

范聖佑 Shengyou Fan  
新北市樹林國小 (2015/07/07)

```
1  /**
2   * Display the specified resource.
3   *
4   * @param int $id
5   * @return Response
6   */
7  public function show($id)
8  {
9      $post = Post::with('comments')->where('id', $id)->first();
10
11      return View::make('post.show')->with('post', $post);
12  }
```

# 單元主題

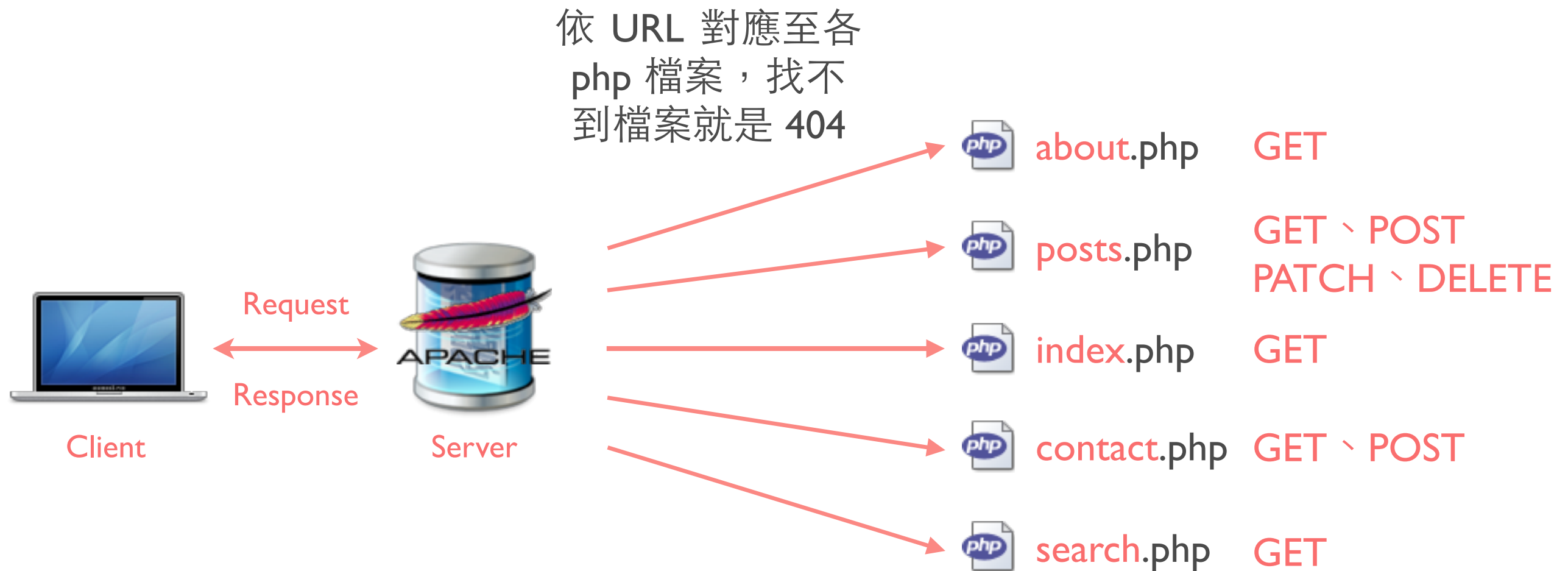
- 什麼是 Route 路由？
- 使用 Route 路由跟以往的作法有什麼不同？有什麼好處？
- 如何設定 Laravel 的 Route？
- 依照工作坊網站規劃書示範如何設定 Route

# Route 簡介

# 什麼是 Route (路由)？

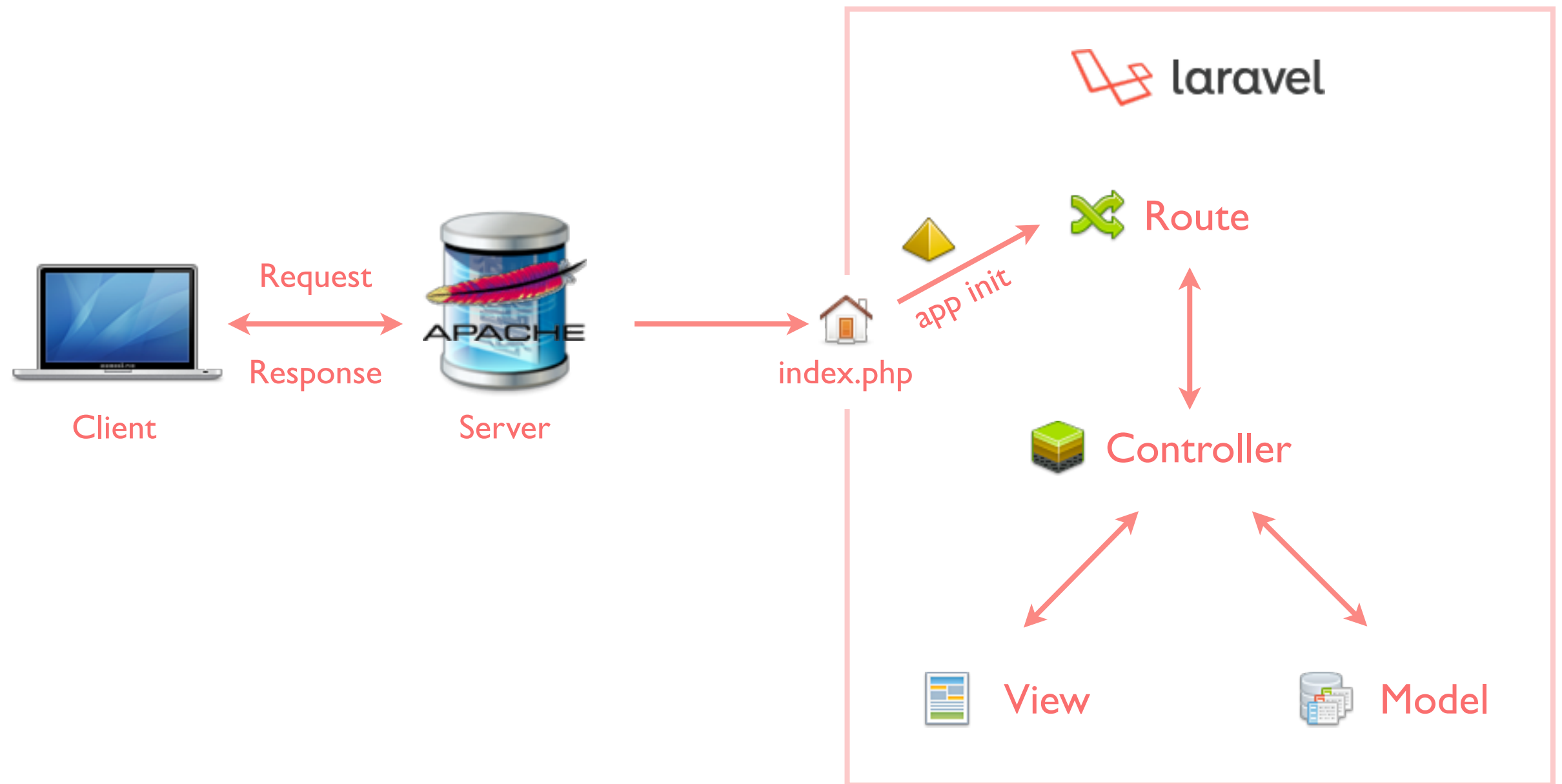
- 把它想像成是應用程式的大門管理員，每一個 Request 都要通過管理員，符合正確的條件才可以送到後方應用程式做處理，而管理員的放行原則，就是依照應用程式設定的 Route 規則來驗證
- 簡單來說，Route 就是應用程式的門禁規則，規則內指定每一個 Route 可以接受的動作 (GET、POST、PATCH、DELETE)、對應的網址 (uri) 及傳入的參數
- Route 會從規則表裡由上而下逐一比對，符合就會執行、若找不到就會回傳 404 (Not Found)

# 在使用 Route 之前...



各 php 檔裡要自行處理可接受的 HTTP 動作判斷

# Laravel 的 Route 機制



# 為什麼要用 Route ?

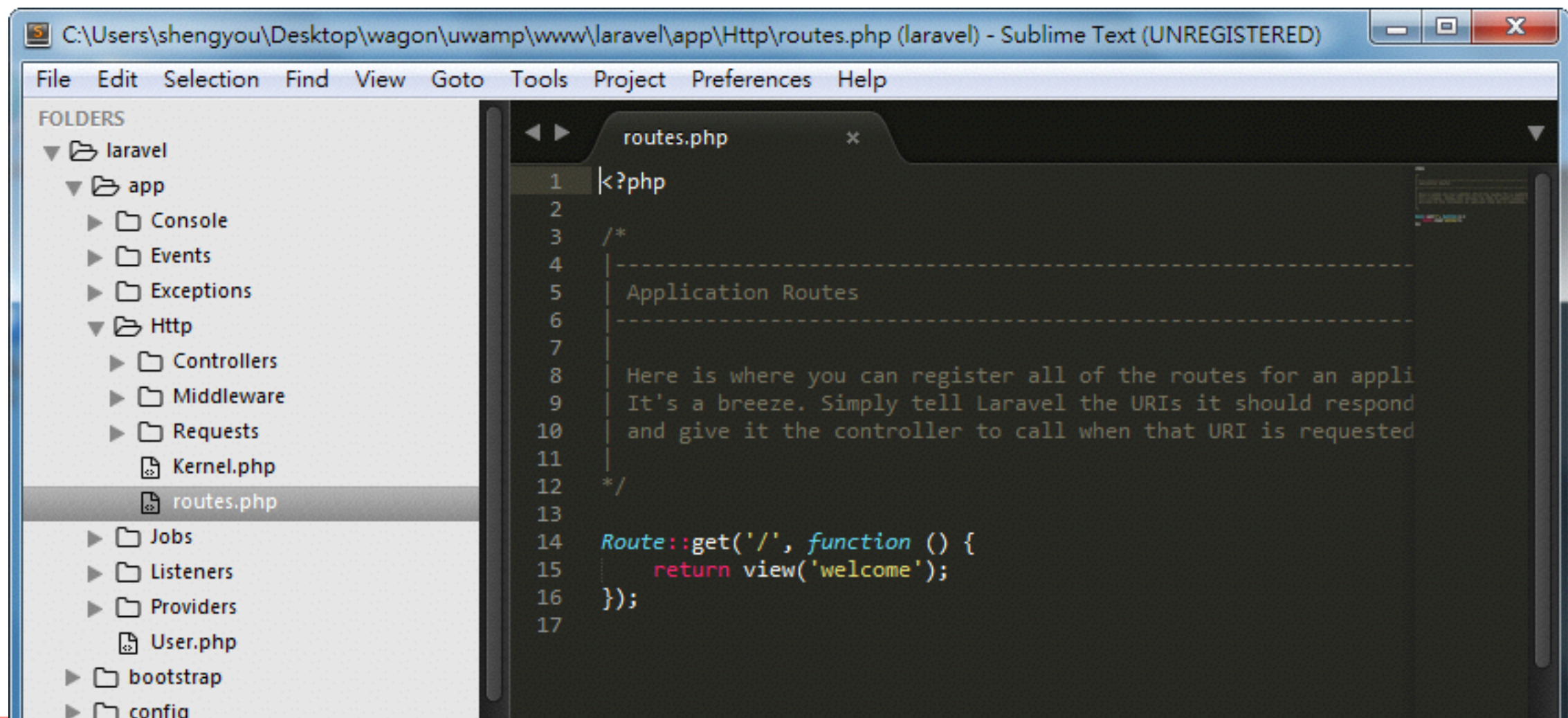
- HTTP 伺服器的 **Document Root** 指定在 **public**，透過框架進入點 (index.php) 統一管理所有路由，增加程式碼安全性
- 搭配 HTTP 伺服器的 **rewrite** 模組 (Apache 和 Nginx 都有對應)，可以讓網站的網址變得更好看也更符合 RESTful 語意
- 透過 **Route** 設計，應用程式內所有可使用的路徑就一目了然，增加團隊合作時的效率
- 在針對應用程式除錯時，可以透過統一的運作路徑逐一排除錯誤

# Route 語法



# Laravel 的 Route 設定檔

- 由於 Route 處理所有 HTTP 相關的 Request，因此 Laravel 的 Route 規則設定檔根據功能放置在：
  - `app/Http/routes.php`



# 設定 Route 動作

- 接收 GET (讀取)

```
Route::get('{uri}', function() { // Closure });
```

- 接收 POST (寫入)

```
Route::post('{uri}', function() { // Closure });
```

- 接收 PATCH (更新)

```
Route::patch('{uri}', function() { // Closure });
```

- 接收 DELETE (刪除)

```
Route::delete('{uri}', function() { // Closure });
```

# Route 接收參數

- 接收必要參數

```
Route::get('posts/{id}', function($id)
{
    return 'Post: ' . $id;
});
```

- 接收選擇性參數

```
Route::get('users/{name?}', function($name = 'John')
{
    return 'My name is ' . $name;
});
```

# 限制參數格式

- 使用 where 限制

```
Route::get('posts/{id}', function($id)
{
    return 'Post: ' . $id;
})
->where('id', '[0-9]+');
```

- 使用 pattern 限制

```
Route::pattern('id', '[0-9]+');
Route::get('posts/{id}', function($id)
{
    return 'Post: ' . $id;
});
```

# 有哪些 routes ?

- 當應用程式愈寫愈大、Route 也因此愈設定愈多時，尤其綜合使用很多 Route 特異功能時，如何知道目前的應用程式有哪些 routes 呢？

答：`artisan` 有一個指令，可以將目前所有 routes 規則整理成表格後印出，透過這個表格就可以知道到底有多少條規則被設定了

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

(列出目前所有的 route 規則)

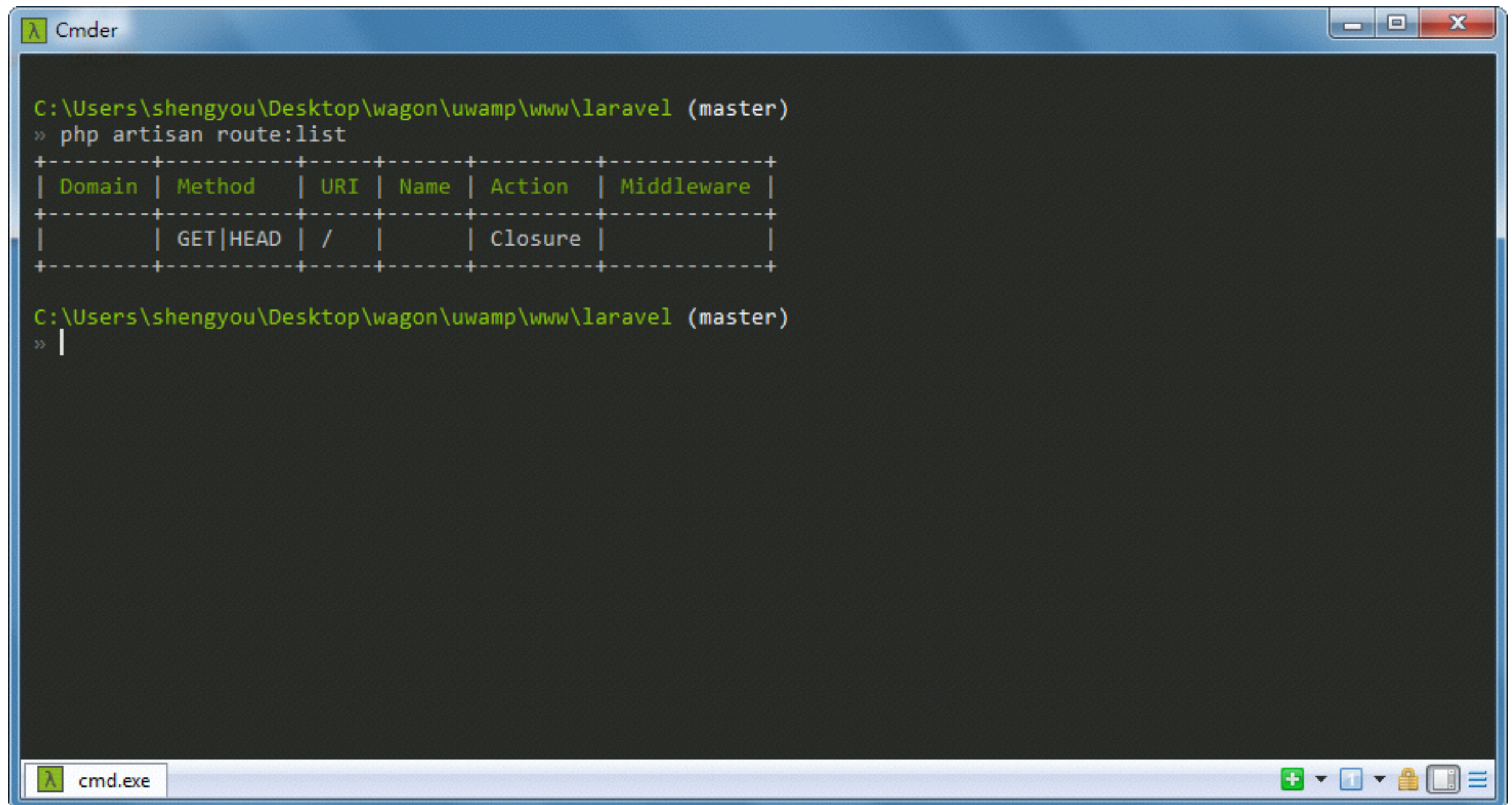
# artisan route:list

- 印出所有 route 表格
  - 呼叫這個指令，artisan 會把目前應用程式內所有的 route 規則整理成表格印出在畫面上
  - --name={...} 依名字搜尋 route
  - --path={...} 依路徑搜尋 route
- 範例：  

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

# 檢查 Route 設定

用 artisan 指令列出目前的 Route 設定



```
C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> php artisan route:list
```

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/		Closure	

```
C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> |
```



# 進階 Route 設定

# 進階 Route 設定

- Laravel 還可搭配進階技巧做出更多彈性的設計
  - Named Routes
  - Route 群組
    - Route Prefixing
    - Sub-Domain Routing

# 為 Route 命名

- 可以為每一個 Route 取一個名字

```
Route::get('post/{id}', ['as' => 'posts.show', function()  
{  
    //  
}]);
```

- 之後在 view 上面就可以直接用 helper 產生 url

```
<a href="{{ route('posts.show', $id) }}"></a>
```

# Route 群組

- 將一系列有相同設定的 Route 規則組成一個群組來一同設定，除了少打一些程式碼外，在管理上也比較方便

```
Route::group(['{function}' => '{setting}'], function()  
{  
    Route::get('{uri}', function()  
    {  
  
    });  
});
```

# Route Prefixing

- 設定某些 Route 前都有共同的前置 uri

```
Route::group(['prefix' => 'admin'], function()  
{  
    Route::get('users', function()  
    {  
        // 實際上的 URL 會是 "/admin/users"  
    });  
});
```

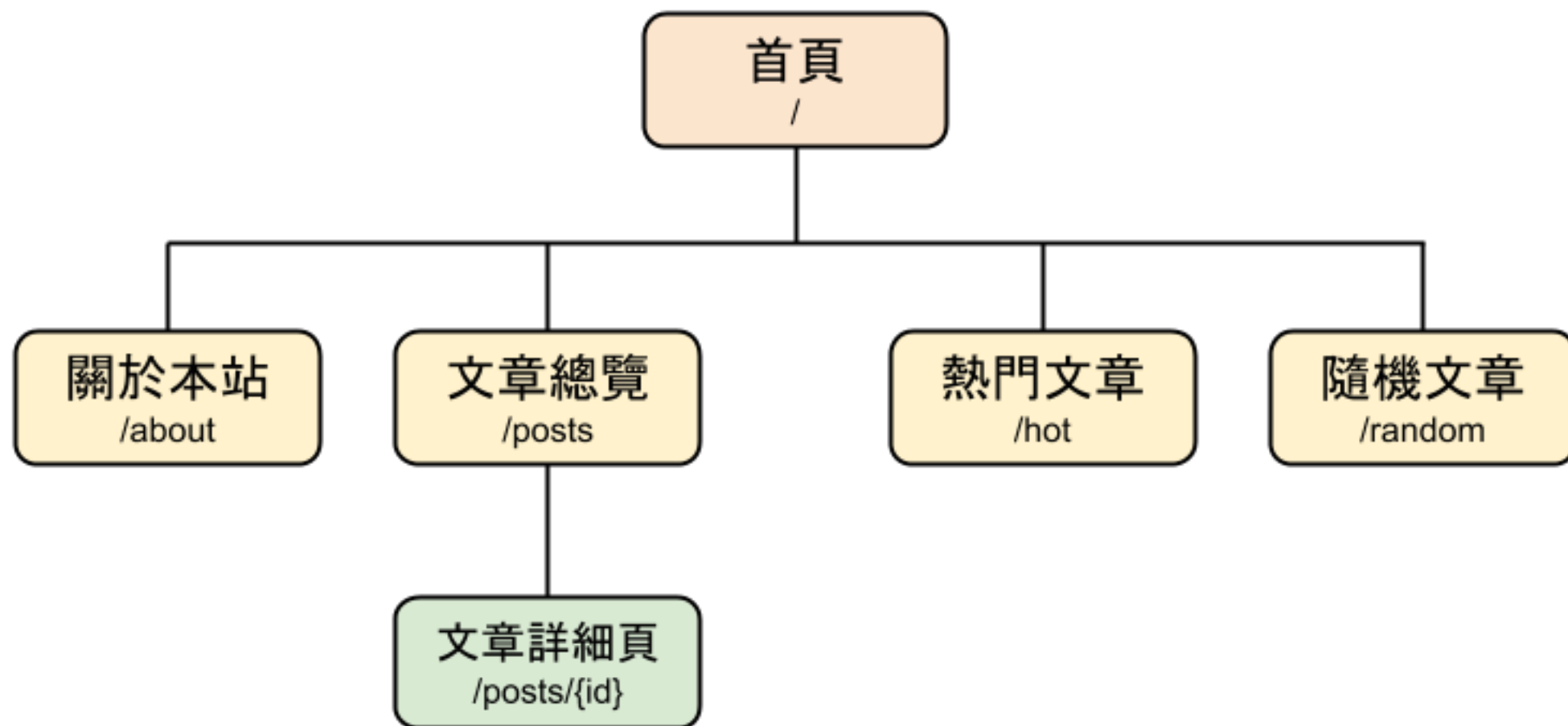
# Sub-Domain Routing

- 限制 Route 只能在某個 Sub-Domain 才接受

```
Route::group(['domain' => '{account}.myapp.com'], function()
{
    Route::get('user/{id}', function($account, $id)
    {
        // URL 若是 {account}.myapp.com 就會進入這裡
    });
});
```

設定專案 Route

# 專案網站架構圖





# 預定完成的 Route

頁面名稱	Method	網址	路由名稱	樣板
首頁 (精選文章)	GET	/?page={page}	home.index	/posts/index.blade.php
關於本站頁	GET	/about	about.index	/about/index.blade.php
文章總覽頁	GET	/posts?page={page}	posts.index	/posts/index.blade.php
熱門文章頁	GET	/hot?page={page}	posts.hot	/posts/index.blade.php
隨機文章頁	GET	/random	posts.random	/posts/show.blade.php
文章詳細頁	GET	/posts/{id}	posts.show	/posts/show.blade.php
新增文章頁	GET	/posts/create	posts.create	/posts/create.blade.php
儲存文章	POST	/posts	posts.store	-
編輯文章頁	GET	/posts/{id}/edit	posts.edit	/posts/edit.blade.php
更新文章	PATCH	/posts/{id}	posts.update	-
刪除文章	DELETE	/posts/{id}	posts.destroy	-
新增回覆	POST	/posts/{id}/comment	posts.comment	-

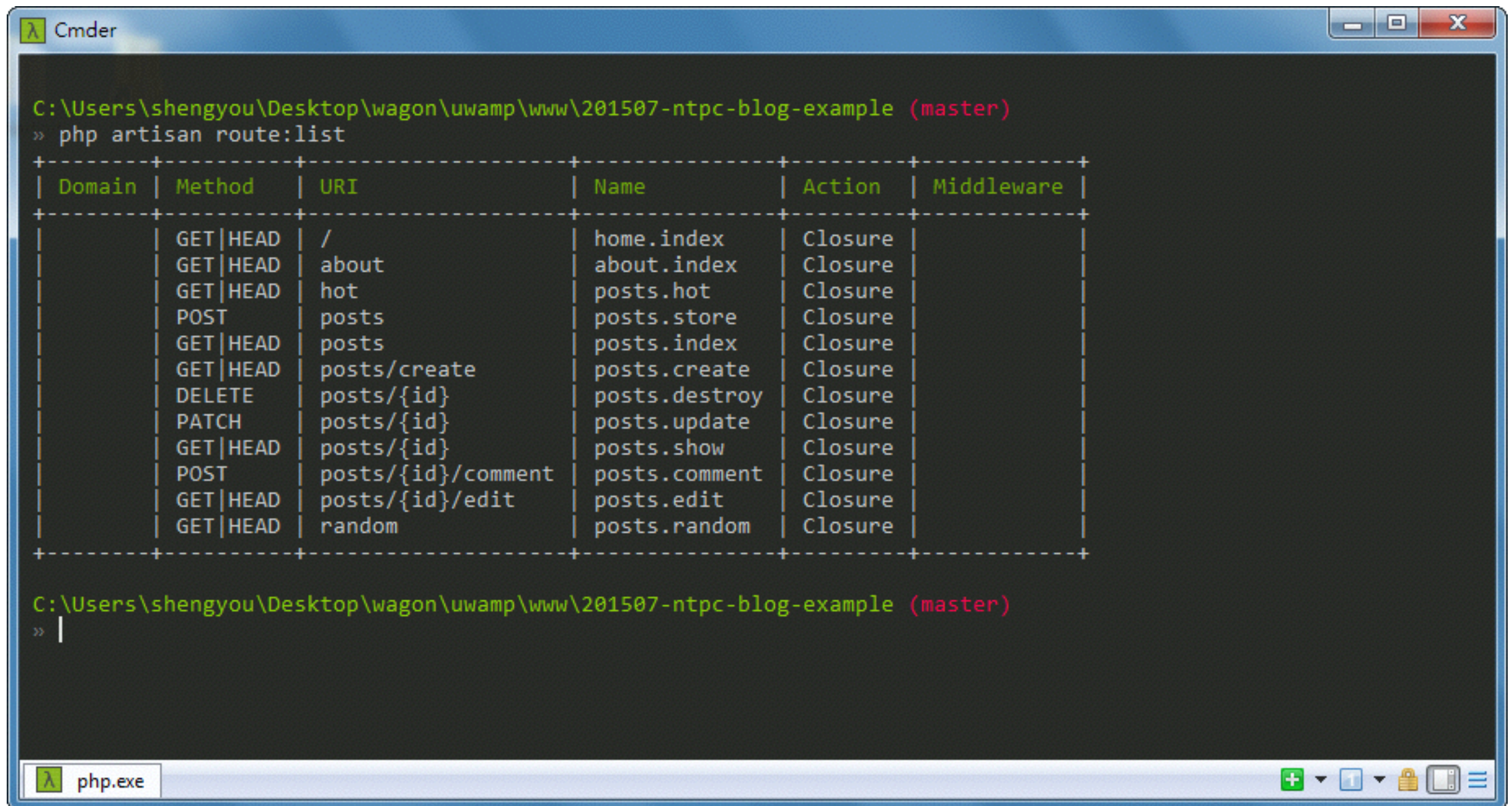
# 直接回傳字串

- 在這個階段的實作裡，我們先直接在 **Route** 裡回傳字串，先確認把通道打開即可
- 在瀏覽器裡預覽時，只要有看到字串顯示在畫面上就表示成功了
- 若發現打中文是亂碼，是因為我們送出的內容裡沒有編碼設定

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

# Route 規則表格

設定完 routes.php 後，執行 artisan route:list 檢查結果



```
C:\Users\shengyou\Desktop\wagon\uwamp\www\201507-ntpc-blog-example (master)
>> php artisan route:list
```

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/	home.index	Closure	
	GET HEAD	about	about.index	Closure	
	GET HEAD	hot	posts.hot	Closure	
	POST	posts	posts.store	Closure	
	GET HEAD	posts	posts.index	Closure	
	GET HEAD	posts/create	posts.create	Closure	
	DELETE	posts/{id}	posts.destroy	Closure	
	PATCH	posts/{id}	posts.update	Closure	
	GET HEAD	posts/{id}	posts.show	Closure	
	POST	posts/{id}/comment	posts.comment	Closure	
	GET HEAD	posts/{id}/edit	posts.edit	Closure	
	GET HEAD	random	posts.random	Closure	

```
C:\Users\shengyou\Desktop\wagon\uwamp\www\201507-ntpc-blog-example (master)
>> |
```

# 存檔點

- 試著把現在已經可以運作的程式碼加入版本控制內
- 流程提醒：
  - working directory > staging area > commit

# 單元總結

- 在這個單元裡我們學到了些什麼？
  - 什麼是 Route 以及使用 Route 的好處
  - Laravel 的 Route 語法
  - 設定專案應用程式的 Route



歡迎提問討論