

\*\*\*\*\*

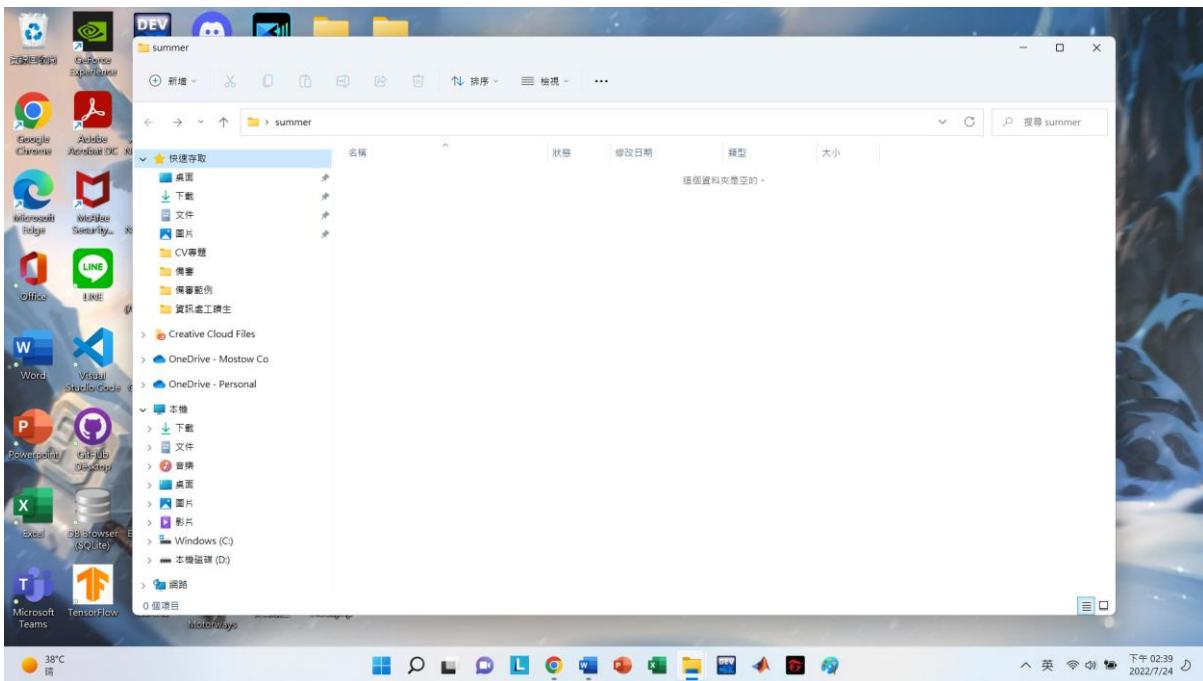
公告系統實作

廉價低薪苦工訓練班

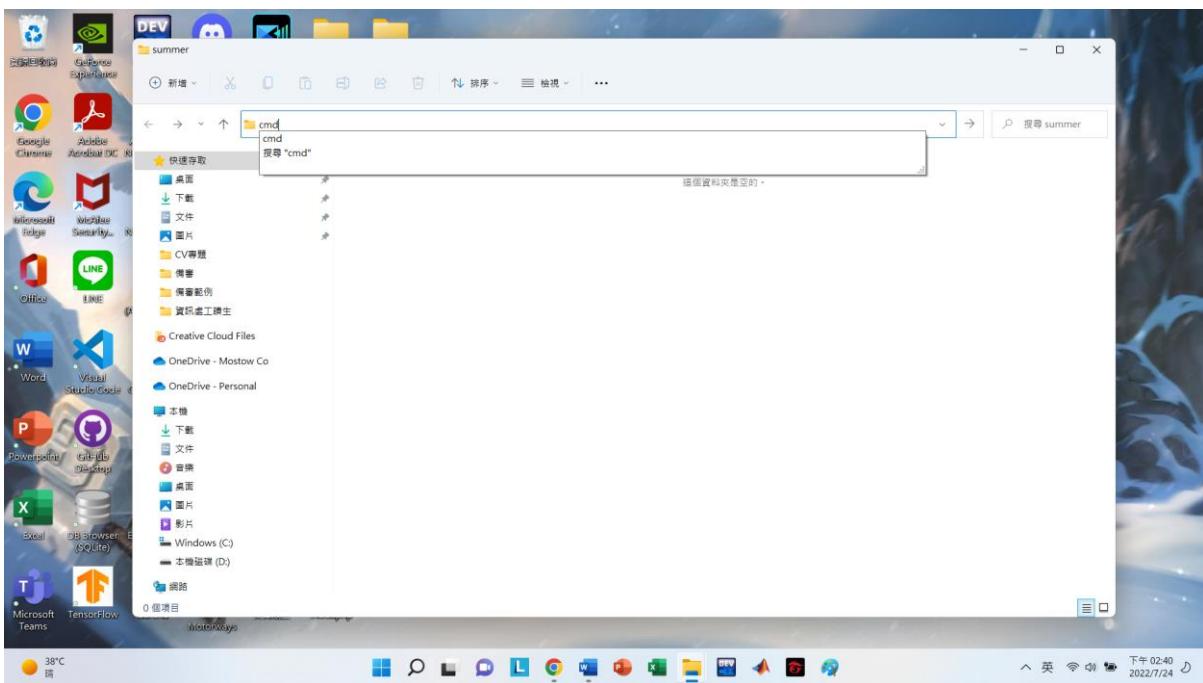
\*\*\*\*\*

## 1. 建立 CI4 專案

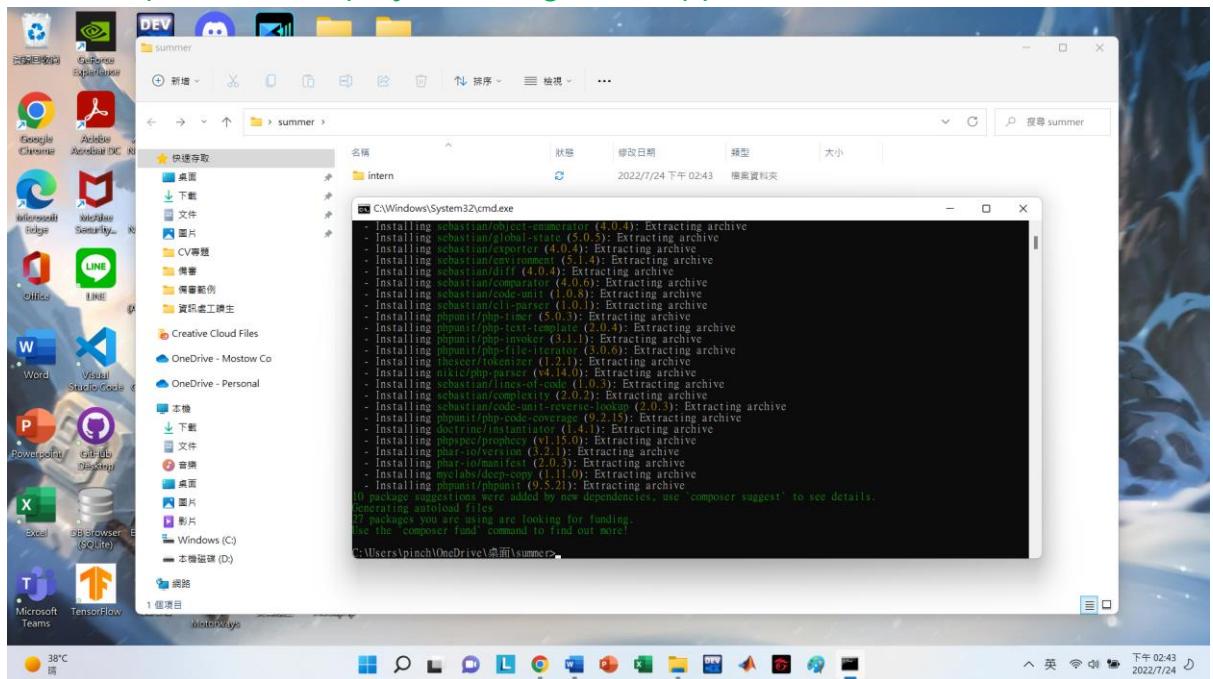
## 隨便建立一個資料夾



## 使用cmd

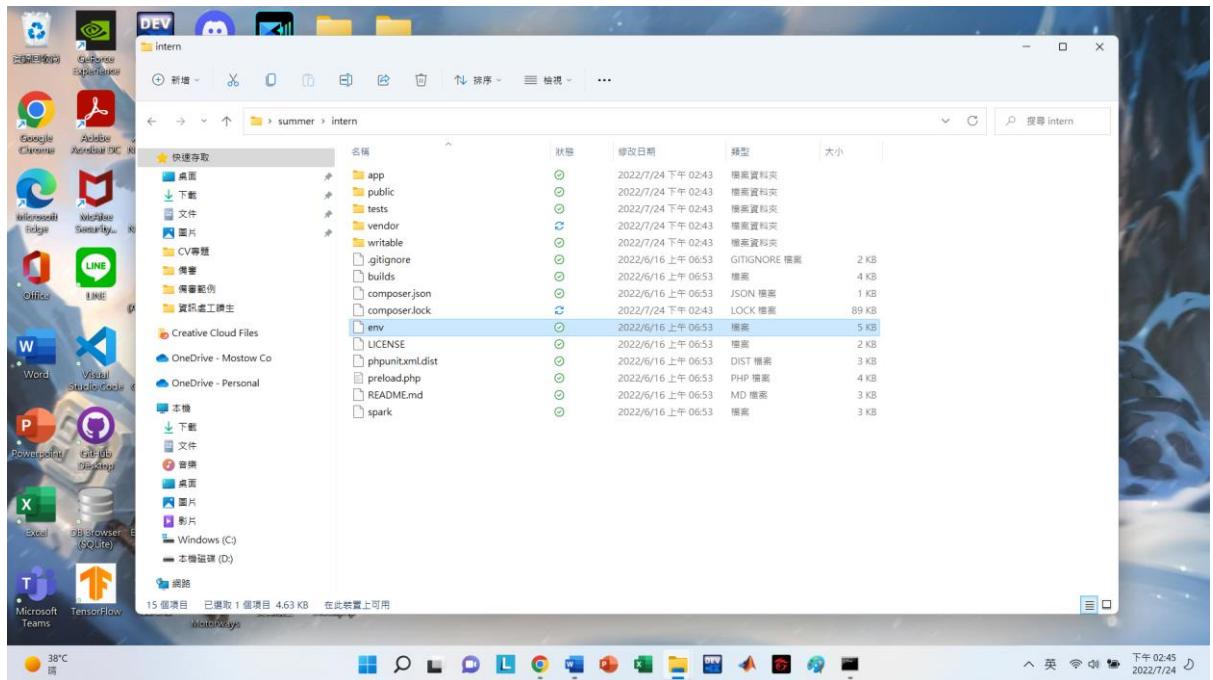


輸入 `composer create-project codeigniter4/appstarter intern` (專案名自己取) 指令

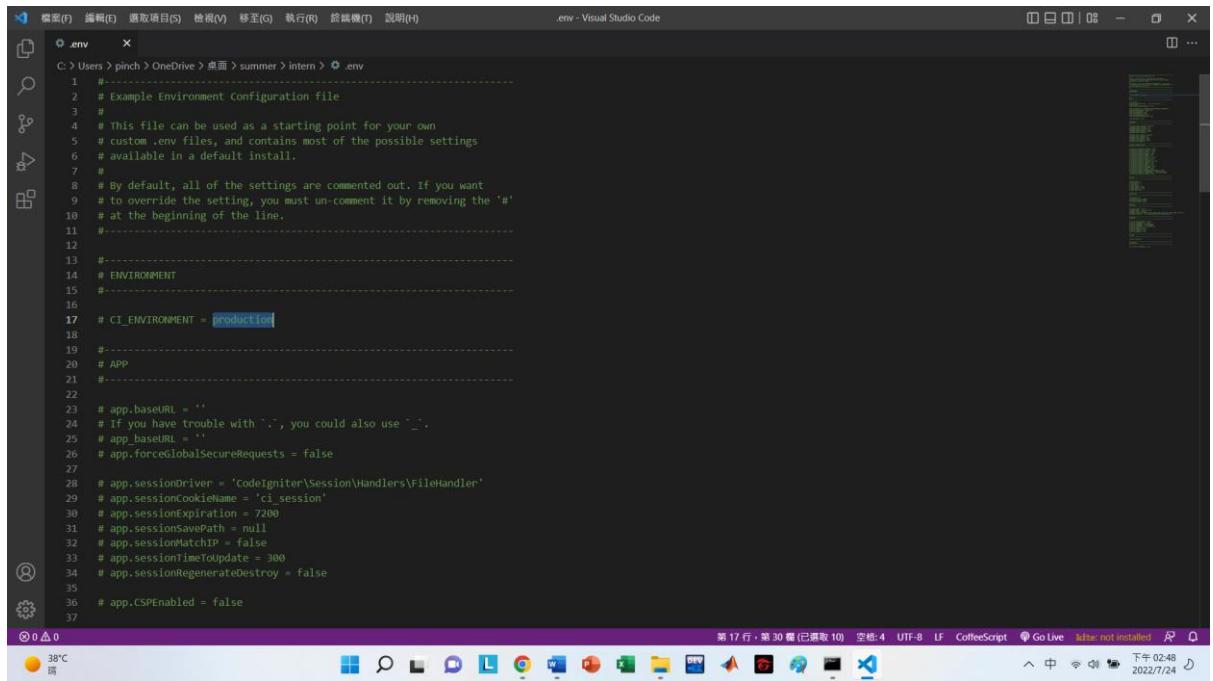


看到這畫面表示專案創建成功

打開專案把 env 檔改名為 .env 然後打開

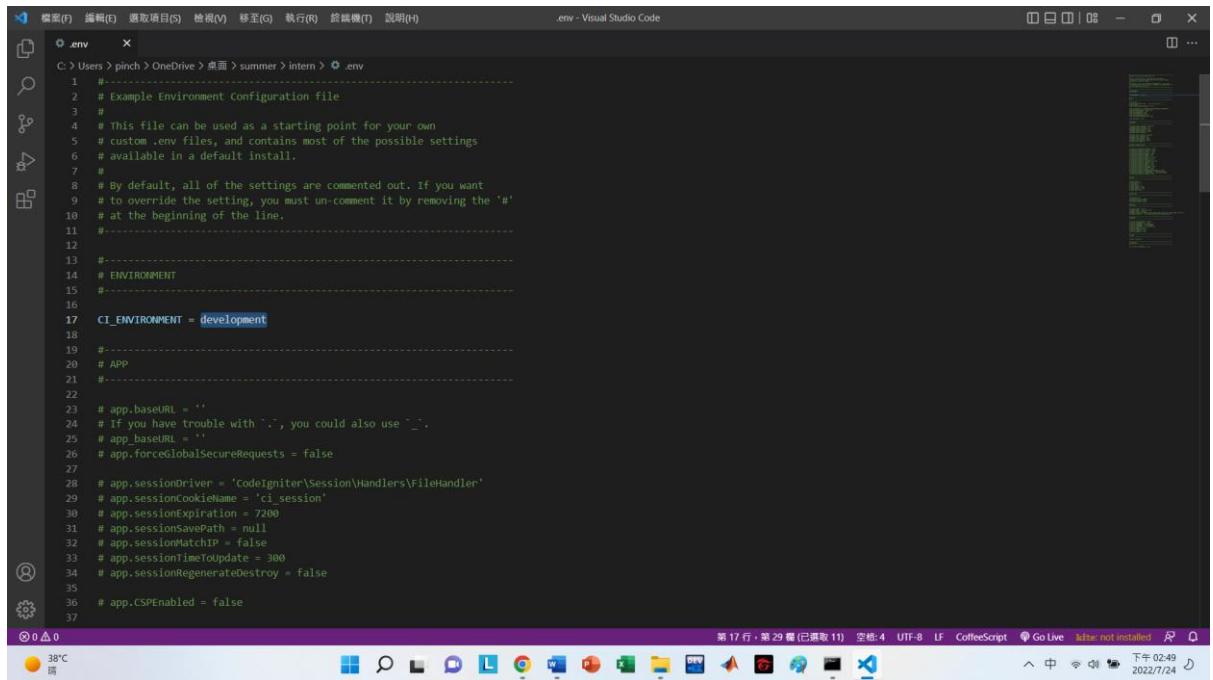


將 `# CI_ENVIRONMENT = production` 註解#拿掉，  
production 改成 development



```
C:\Users\pinch>OneDrive>桌面>summer>intern>.env
1 #
2 # Example Environment Configuration file
3 #
4 # This file can be used as a starting point for your own
5 # custom .env files, and contains most of the possible settings
6 # available in a default install.
7 #
8 # By default, all of the settings are commented out. If you want
9 # to override the setting, you must un-comment it by removing the '#'
10 # at the beginning of the line.
11 #
12 #
13 #
14 # ENVIRONMENT
15 #
16 #
17 # CI_ENVIRONMENT = production
18 #
19 #
20 # APP
21 #
22 #
23 # app.baseURL = ''
24 # If you have trouble with '.', you could also use '_'.
25 # app.baseURL = '_'
26 # app.forceGlobalSecureRequests = false
27 #
28 # app.sessionDriver = 'CodeIgniter\Session\Handlers\FileHandler'
29 # app.sessionCookieName = 'ci_session'
30 # app.sessionExpiration = 7200
31 # app.sessionSavePath = null
32 # app.sessionMatchIP = false
33 # app.sessionTimeToUpdate = 300
34 # app.sessionRegenerateDestroy = false
35 #
36 # app.CSPEnabled = false
37
```

改完後如下



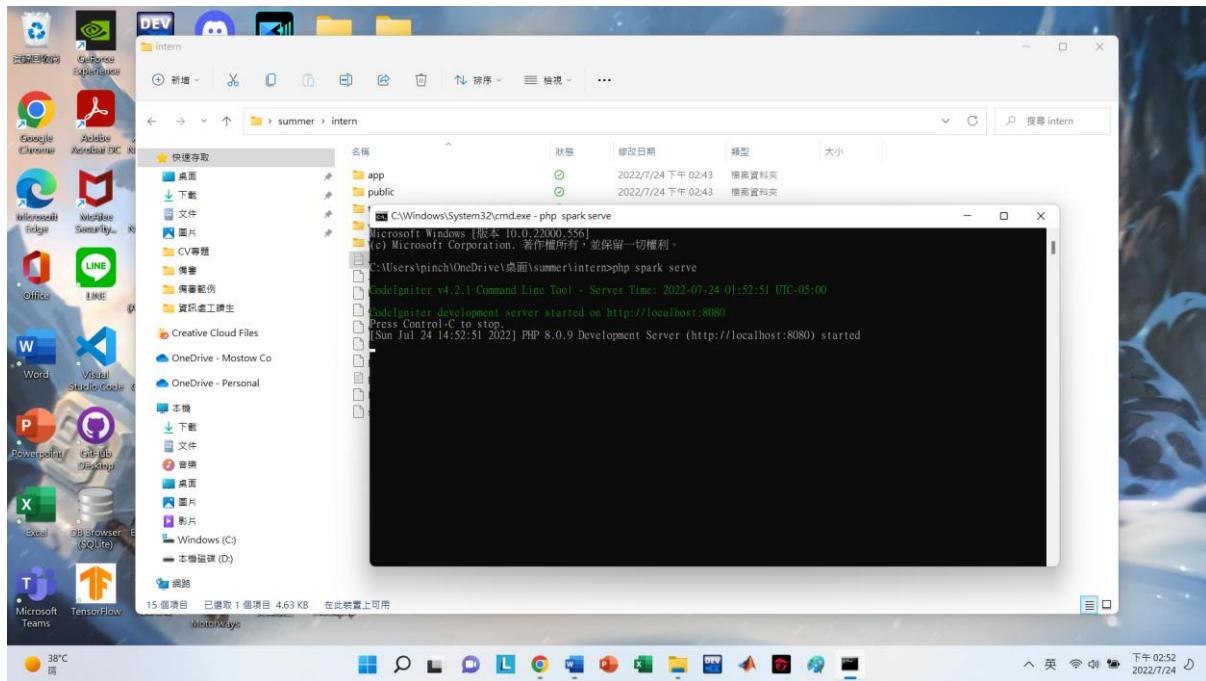
```
C:\Users\pinch>OneDrive>桌面>summer>intern>.env
1 #
2 # Example Environment Configuration file
3 #
4 # This file can be used as a starting point for your own
5 # custom .env files, and contains most of the possible settings
6 # available in a default install.
7 #
8 # By default, all of the settings are commented out. If you want
9 # to override the setting, you must un-comment it by removing the '#'
10 # at the beginning of the line.
11 #
12 #
13 #
14 # ENVIRONMENT
15 #
16 #
17 CI_ENVIRONMENT = development
18 #
19 #
20 # APP
21 #
22 #
23 # app.baseURL = ''
24 # If you have trouble with '.', you could also use '_'.
25 # app.baseURL = '_'
26 # app.forceGlobalSecureRequests = false
27 #
28 # app.sessionDriver = 'CodeIgniter\Session\Handlers\FileHandler'
29 # app.sessionCookieName = 'ci_session'
30 # app.sessionExpiration = 7200
31 # app.sessionSavePath = null
32 # app.sessionMatchIP = false
33 # app.sessionTimeToUpdate = 300
34 # app.sessionRegenerateDestroy = false
35 #
36 # app.CSPEnabled = false
37
```

這步驟是為了之後我們 code 出 bug 時 CI4 會告訴我們 bug 在哪

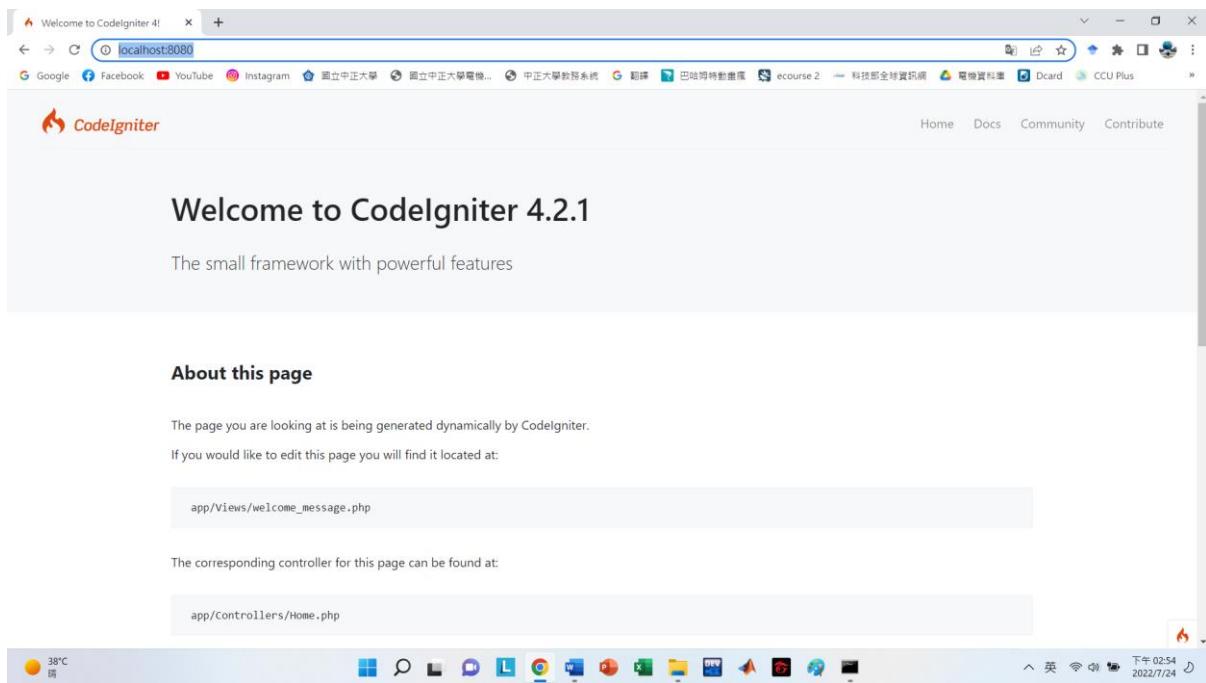
# 先去 P15

到專案裡打開 cmd 輸入 php spark serve .

記得確定 cmd 的位置在專案裡



打開瀏覽器輸入 localhost:8080



看到此畫面就代表你的專案創建成功

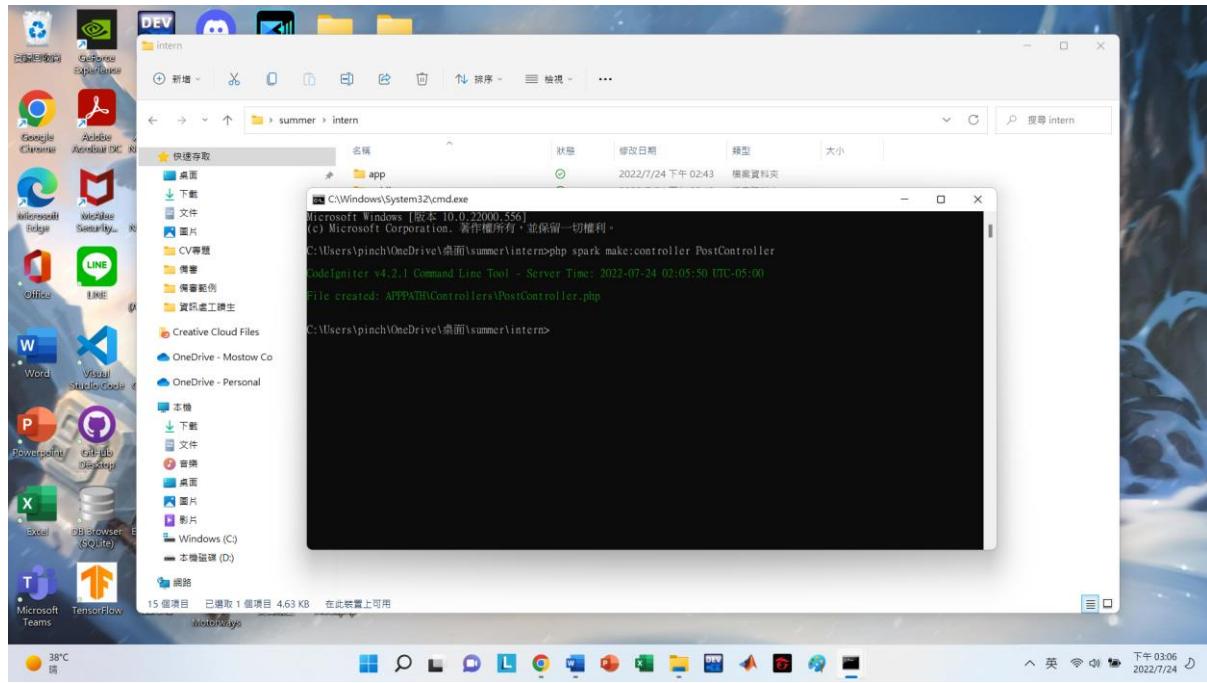
可以進入下一步開始寫網頁了

## 2. 建立 Controller & View

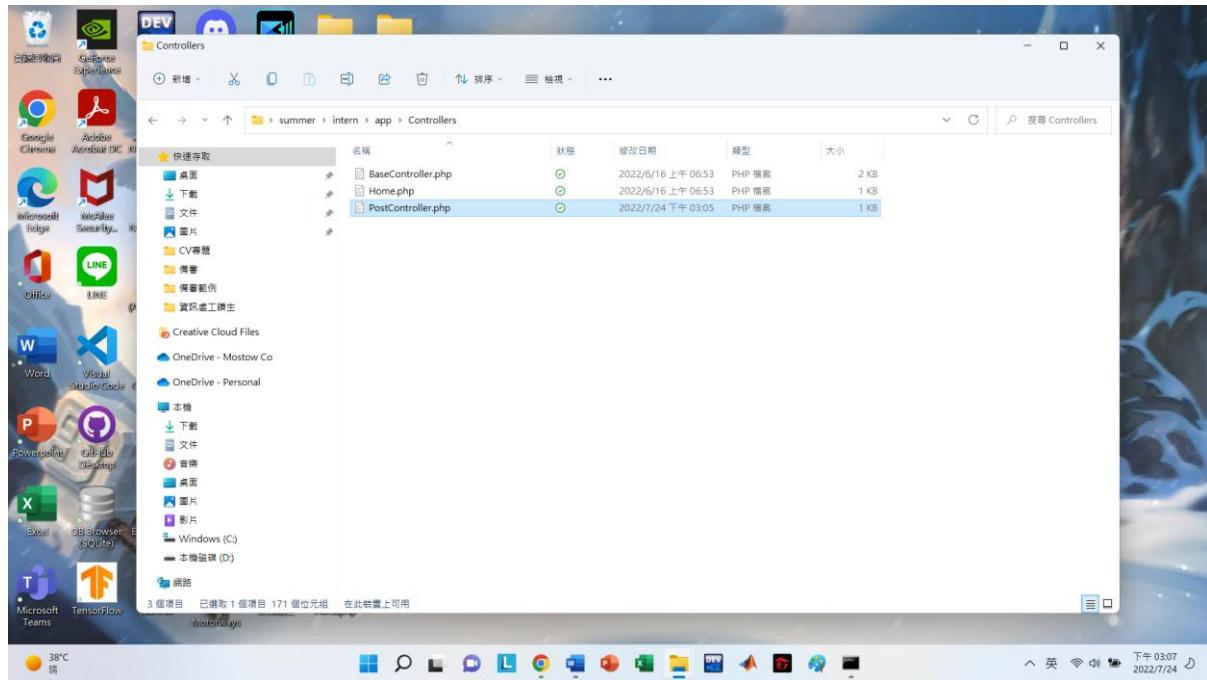
這時我們打開一個新的 cmd(一樣在專案的跟目錄打開) .

剛剛開 serve 的那個就不要動它了

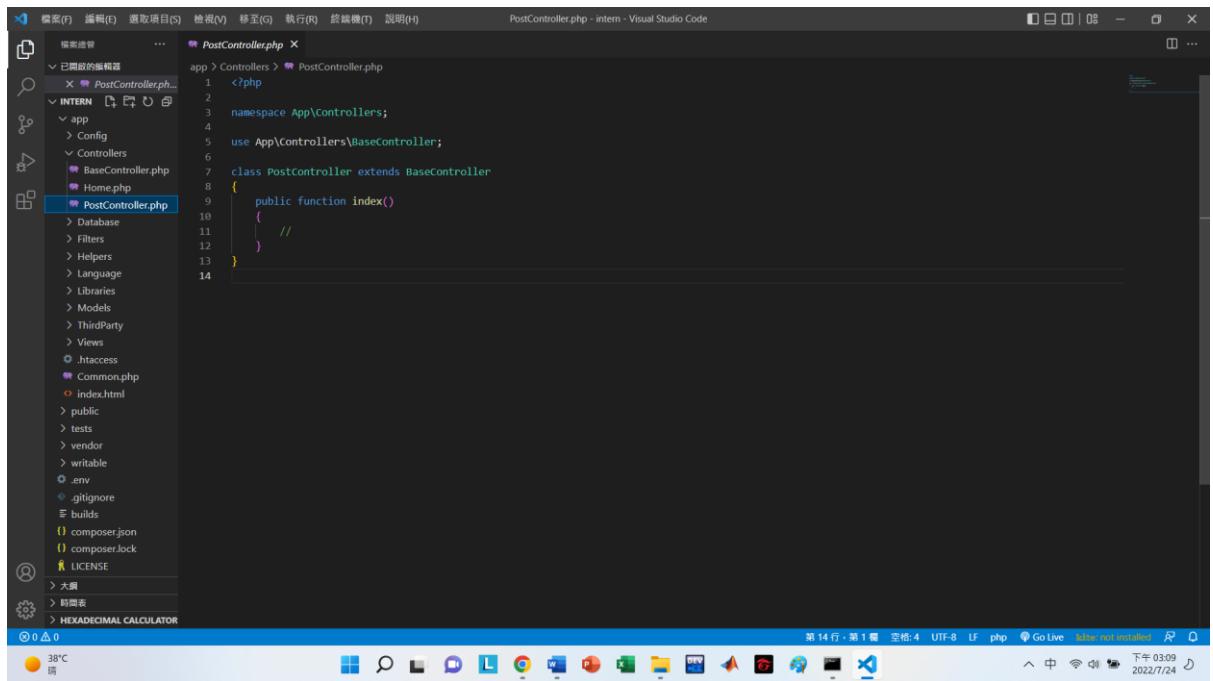
輸入 `php spark make:controller PostController` (Controller 的名字自己取)



建立完成好後到 app -> Controllers 打開它

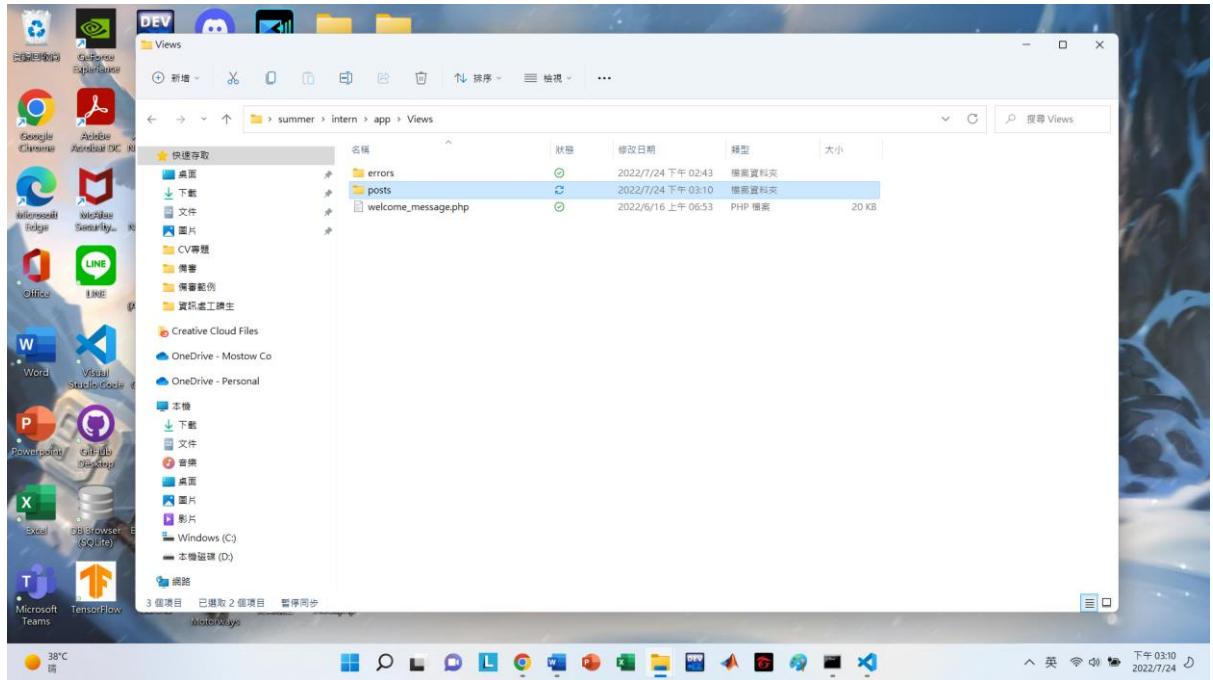


## 打開後先放著



這時我們開始創建 View ,

到 app -> Views 創建一個資料夾 posts ,



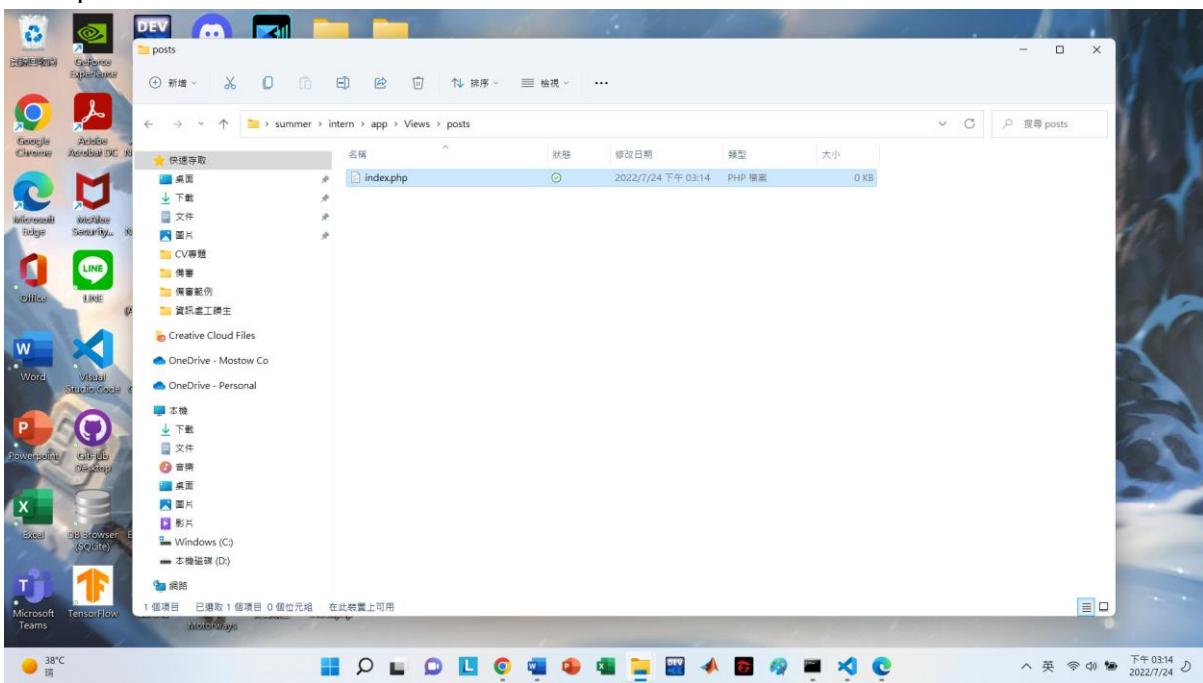
創建資料夾的原因在於你們之後開發時 ,

通常不會只有一個系統而且一個系統通常都會有不少頁面 ,

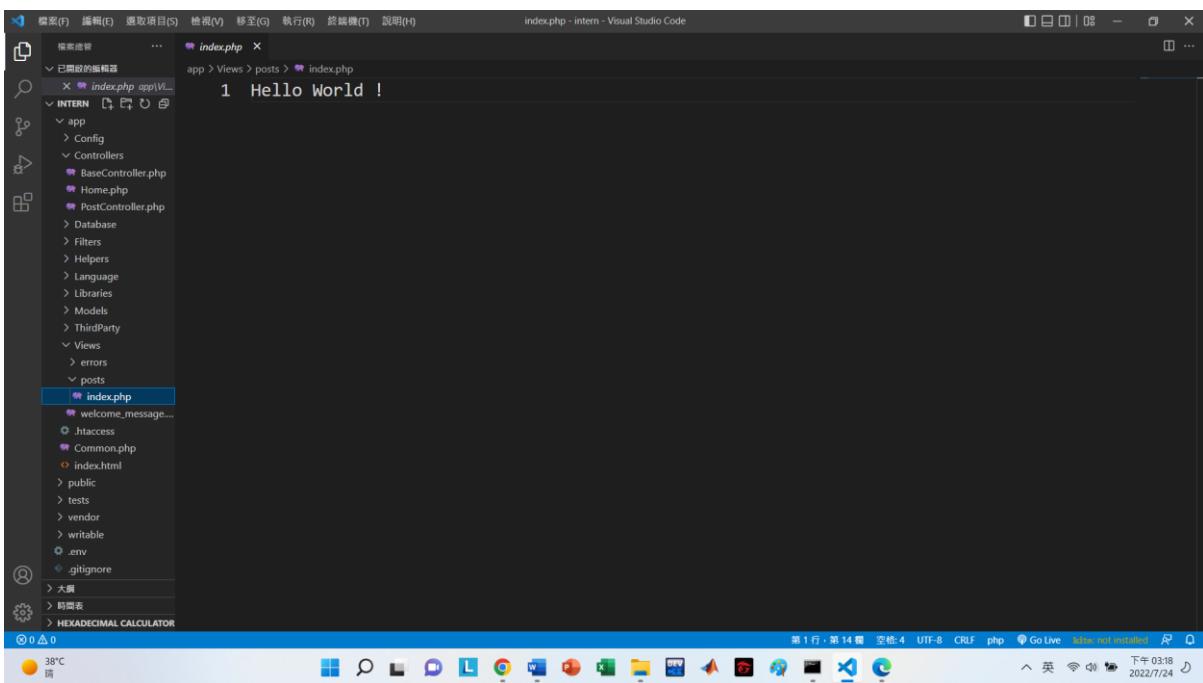
開發時就先整理好 ,

不然就等著幫你維護系統的人幹你!!

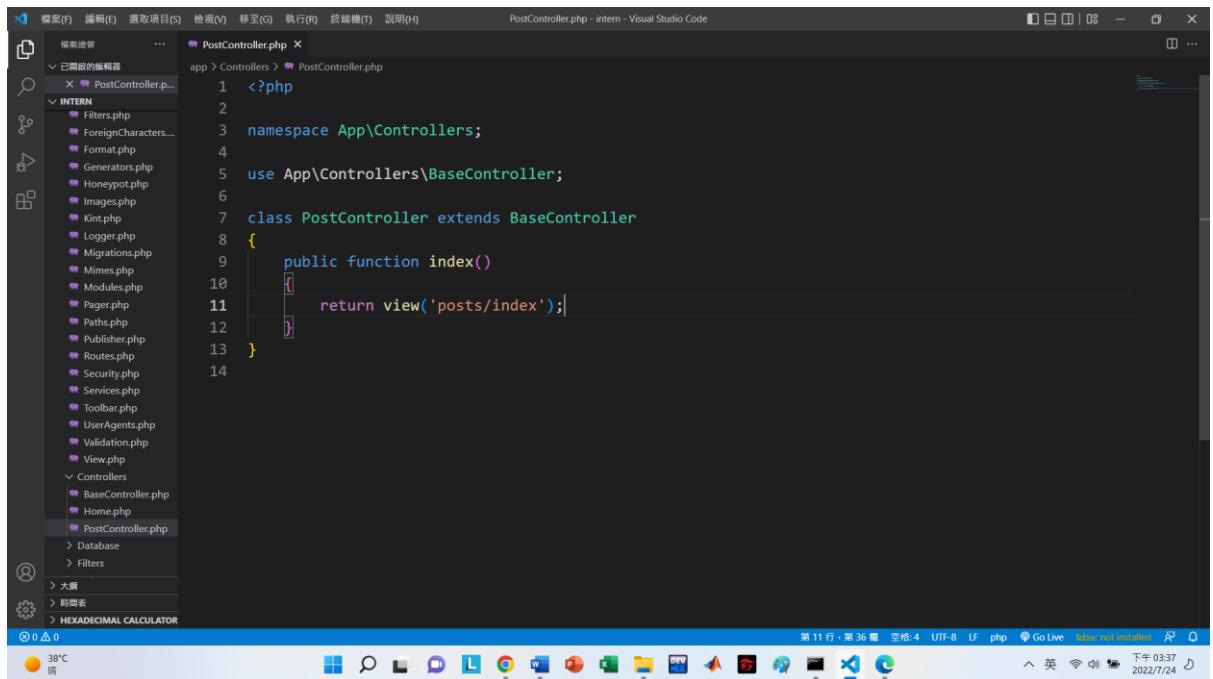
進入 posts 資料夾創建新的網頁頁面



新增內容



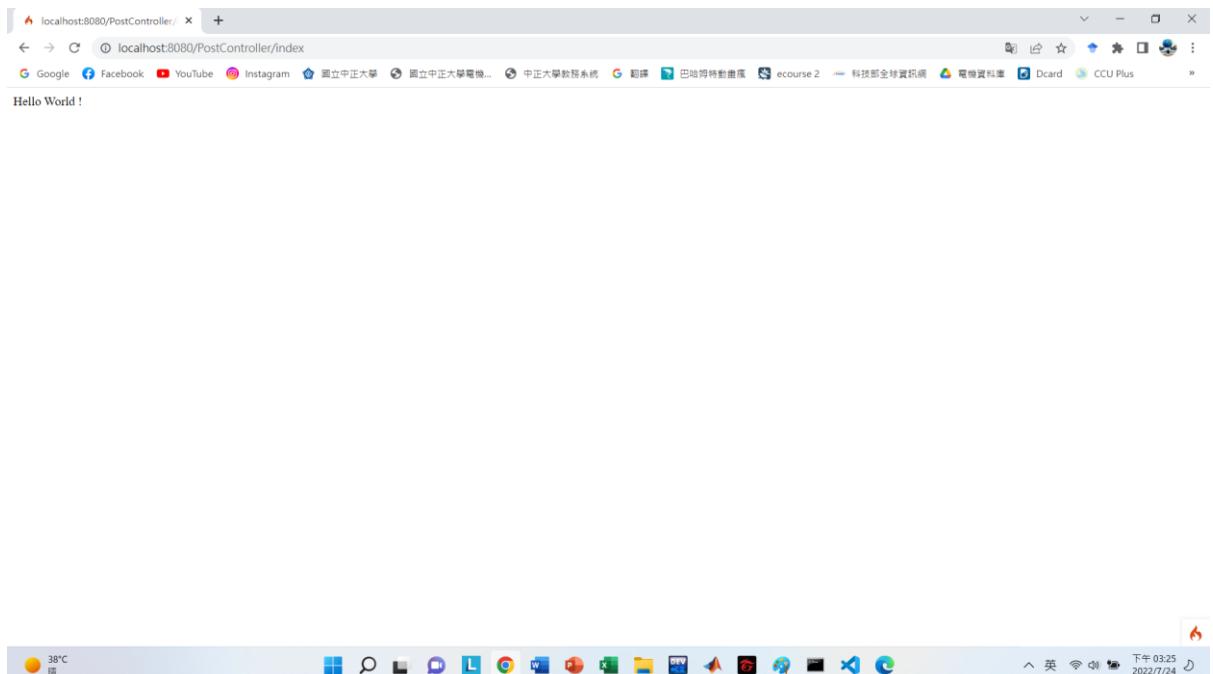
## 回到 PostController 刻 Function



The screenshot shows the Visual Studio Code interface with the file 'PostController.php' open. The code defines a class 'PostController' that extends 'BaseController'. It contains a single method 'index()' which returns a view named 'posts/index'. The code editor has syntax highlighting for PHP, and the status bar at the bottom right shows the date and time as '2022/7/24'.

```
<?php  
namespace App\Controllers;  
use App\Controllers\BaseController;  
  
class PostController extends BaseController  
{  
    public function index()  
    {  
        return view('posts/index');  
    }  
}
```

到瀏覽器輸入 localhost:8080/PostController/index 就可以看到網頁頁面



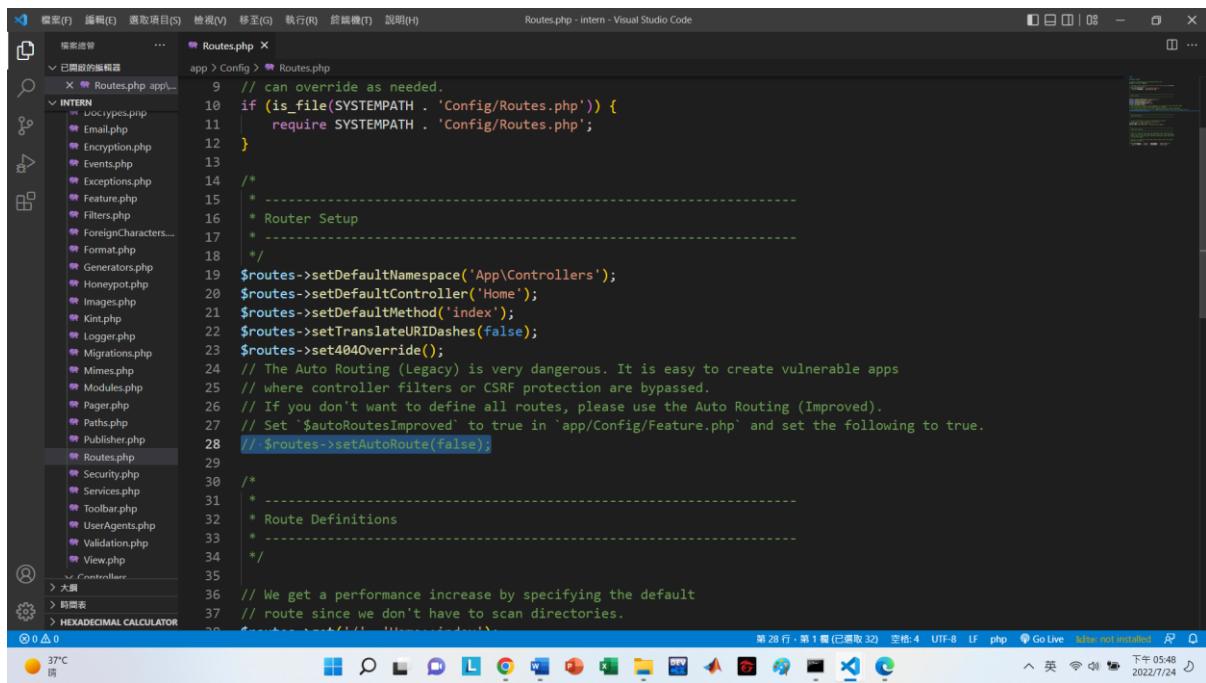
根據版本不同在這邊可能會發生問題，

若在這邊成功看到網頁頁面則可跳過下一页，

若看到 404 – File Not Found 請繼續往下看

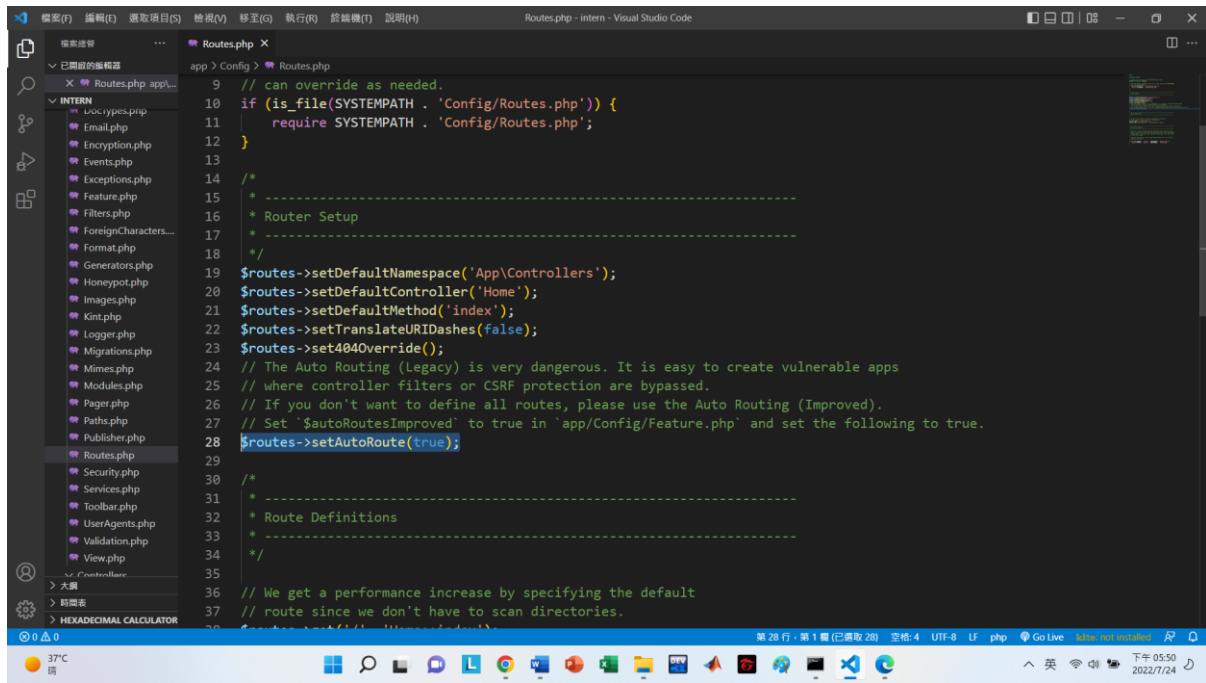
若看到的畫面顯示 404 – File Not Found ,  
這時可以到 app -> Config -> Routes.php 查看 ,  
找到第28行 ,

拿掉註解//並設置為true ,  
目的是為了讓網站可以自動找到網頁路徑



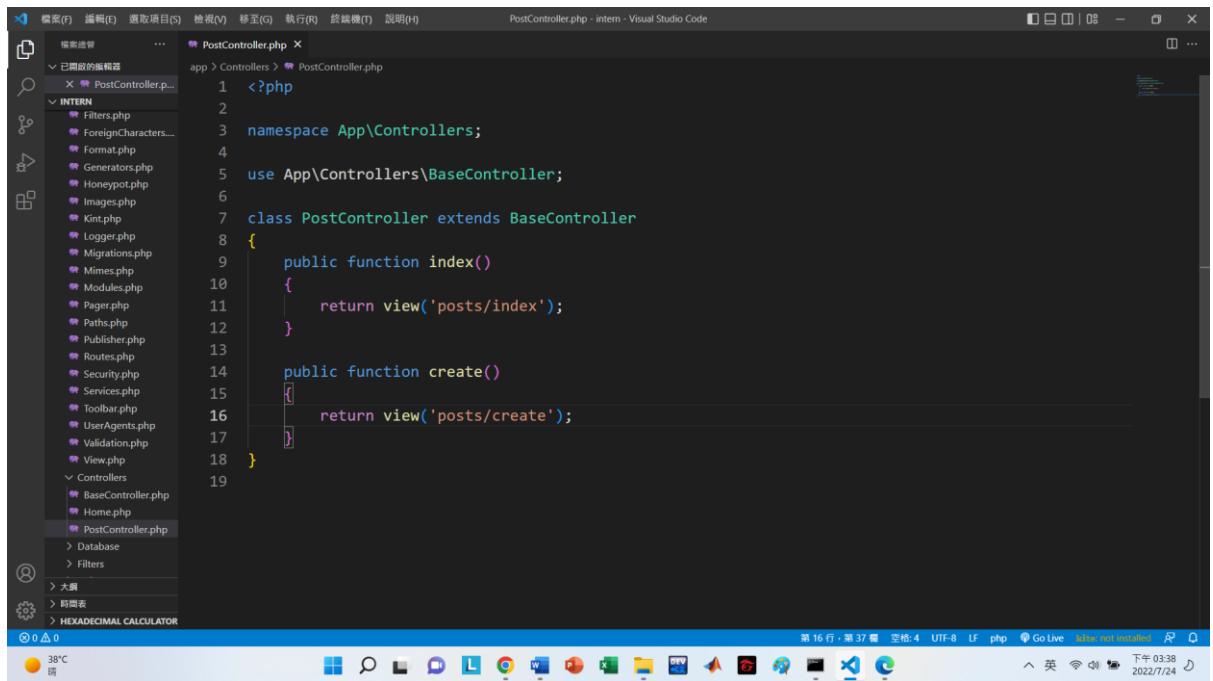
```
9 // can override as needed.
10 if (is_file(SYSTEMPATH . 'Config/Routes.php')) {
11 require SYSTEMPATH . 'Config/Routes.php';
12 }
13
14 /*
15 * -----
16 * Router Setup
17 * -----
18 */
19 $routes->setDefaultNamespace('App\Controllers');
20 $routes->setDefaultController('Home');
21 $routes->setDefaultMethod('index');
22 $routes->setTranslateURIDashes(false);
23 $routes->set404Override();
24 // The Auto Routing (Legacy) is very dangerous. It is easy to create vulnerable apps
25 // where controller filters or CSRF protection are bypassed.
26 // If you don't want to define all routes, please use the Auto Routing (Improved).
27 // Set '$autoRoutesImproved' to true in 'app/Config/Feature.php' and set the following to true.
28 // $routes->setAutoRoute(false);
29
30 /*
31 * -----
32 * Route Definitions
33 * -----
34 */
35
36 // We get a performance increase by specifying the default
37 // route since we don't have to scan directories.
```

改完後如下



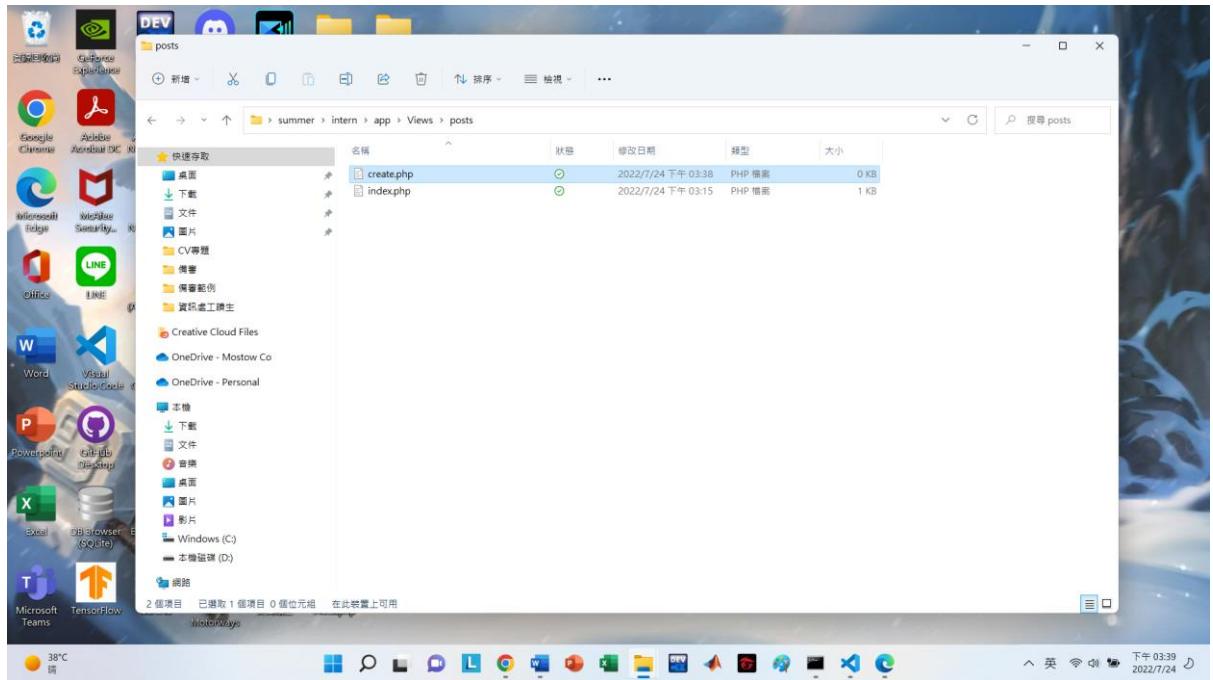
```
9 // can override as needed.
10 if (is_file(SYSTEMPATH . 'Config/Routes.php')) {
11 require SYSTEMPATH . 'Config/Routes.php';
12 }
13
14 /*
15 * -----
16 * Router Setup
17 * -----
18 */
19 $routes->setDefaultNamespace('App\Controllers');
20 $routes->setDefaultController('Home');
21 $routes->setDefaultMethod('index');
22 $routes->setTranslateURIDashes(false);
23 $routes->set404Override();
24 // The Auto Routing (Legacy) is very dangerous. It is easy to create vulnerable apps
25 // where controller filters or CSRF protection are bypassed.
26 // If you don't want to define all routes, please use the Auto Routing (Improved).
27 // Set '$autoRoutesImproved' to true in 'app/Config/Feature.php' and set the following to true.
28 $routes->setAutoRoute(true);
29
30 /*
31 * -----
32 * Route Definitions
33 * -----
34 */
35
36 // We get a performance increase by specifying the default
37 // route since we don't have to scan directories.
```

順利看到網頁頁面後，  
我們就可以準備刻給使用者寫文章的頁面，  
新增 create function 顯示新增文章頁面

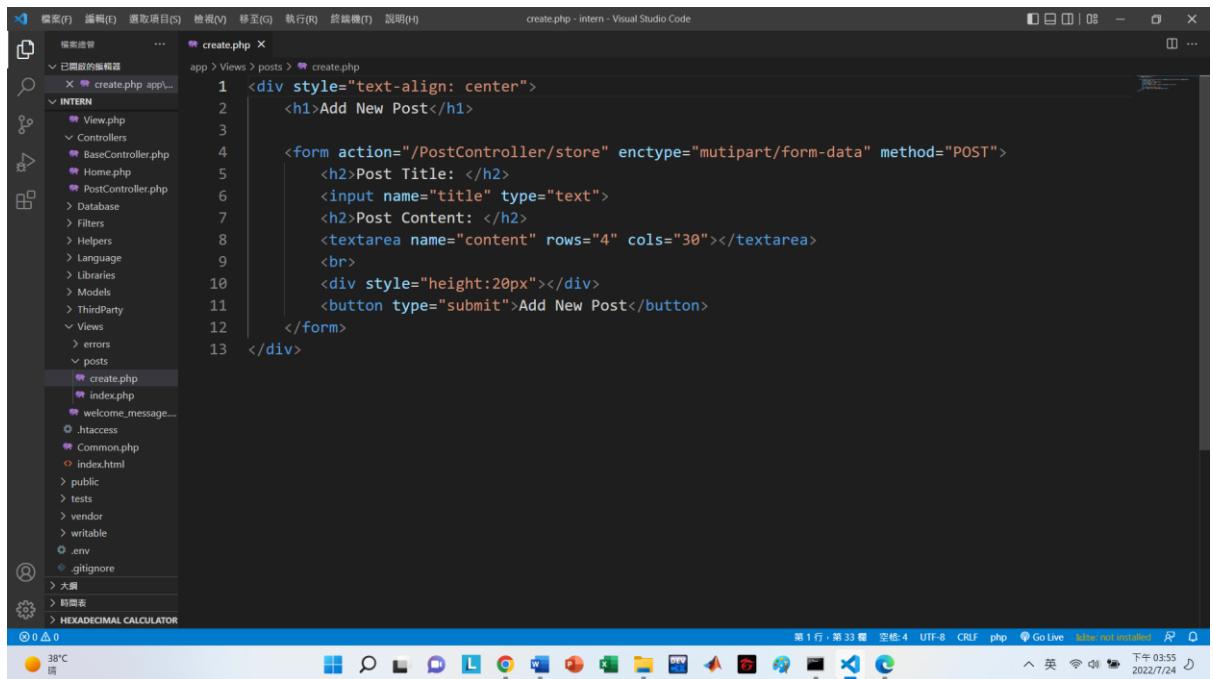


```
PostController.php
1 <?php
2
3 namespace App\Controllers;
4
5 use App\Controllers\BaseController;
6
7 class PostController extends BaseController
8 {
9     public function index()
10    {
11        return view('posts/index');
12    }
13
14    public function create()
15    {
16        return view('posts/create');
17    }
18 }
19
```

到 app -> Views -> posts 中創建 create.php



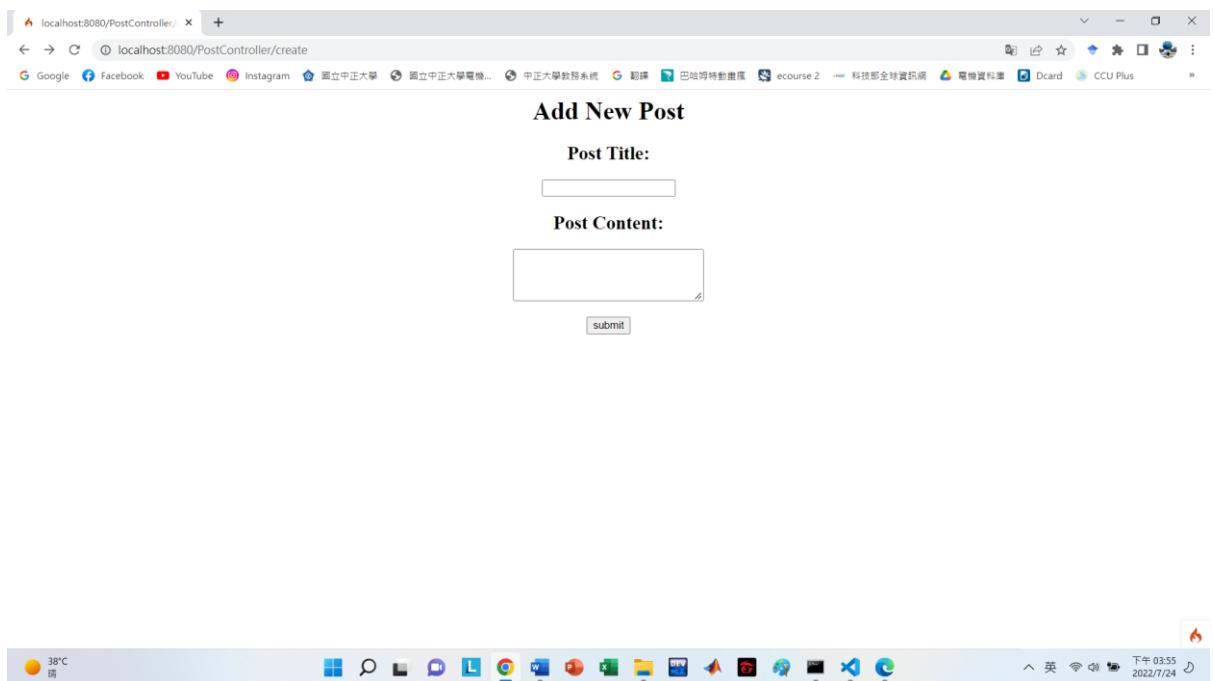
接著就可以用各位所學的 HTML、CSS、JS 開始刻前端頁面



A screenshot of Visual Studio Code showing the file structure and code for a PHP file named 'create.php'. The code is for a form to add a new post, including fields for title and content.

```
<div style="text-align: center">
    <h1>Add New Post</h1>
    <form action="/PostController/store" enctype="multipart/form-data" method="POST">
        <h2>Post Title:</h2>
        <input name="title" type="text">
        <h2>Post Content:</h2>
        <textarea name="content" rows="4" cols="30"></textarea>
        <br>
        <div style="height:20px"></div>
        <button type="submit">Add New Post</button>
    </form>
</div>
```

刻出來的結果長這樣



A screenshot of a web browser displaying a form titled 'Add New Post'. The form has two input fields: 'Post Title:' and 'Post Content:', both represented by empty text input boxes. A 'submit' button is located below the content box.

localhost:8080/PostController/

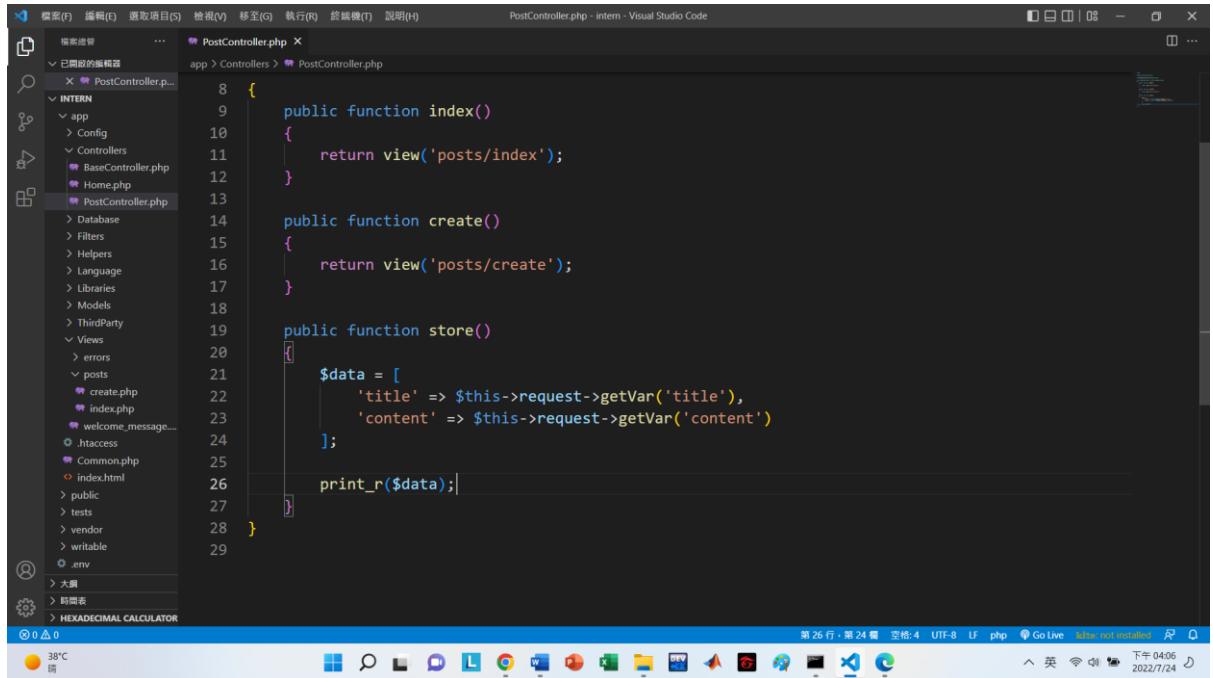
Add New Post

Post Title:

Post Content:

submit

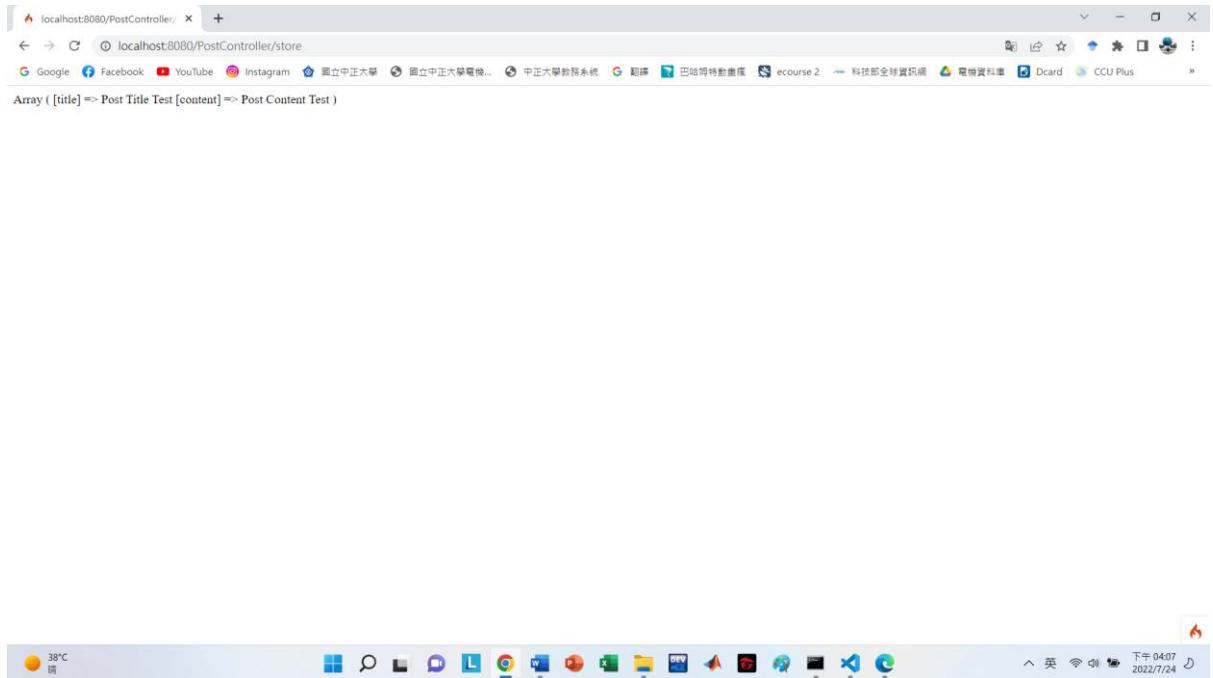
之後我們就建立一個存使用者寫的文章的 function，  
先確定使用者寫的東西有從頁面傳過來，  
所以我們先 print 一下



The screenshot shows the Visual Studio Code interface with the file `PostController.php` open. The code contains three functions: `index()`, `create()`, and `store()`. The `store()` function uses `print_r($data)` to output the received data. The code is as follows:

```
8  {
9      public function index()
10     {
11         return view('posts/index');
12     }
13
14     public function create()
15     {
16         return view('posts/create');
17     }
18
19     public function store()
20     {
21         $data = [
22             'title' => $this->request->getVar('title'),
23             'content' => $this->request->getVar('content')
24         ];
25
26         print_r($data);
27     }
28 }
```

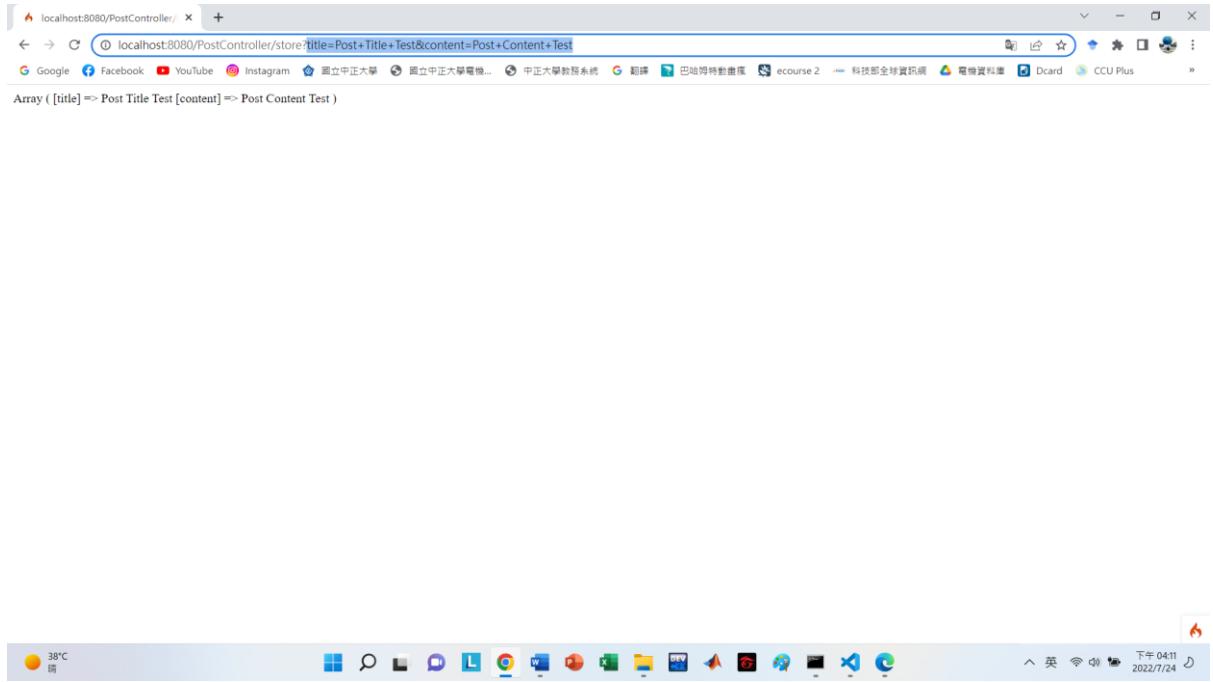
看到有 print 出東西就代表你成功將表單資料傳到 Controller 了



The screenshot shows a browser window displaying the output of the `print_r` function. The URL is `localhost:8080/PostController/store`. The output is an array containing two elements: `[title] => Post Title Test` and `[content] => Post Content Test`.

```
Array ( [title] => Post Title Test [content] => Post Content Test )
```

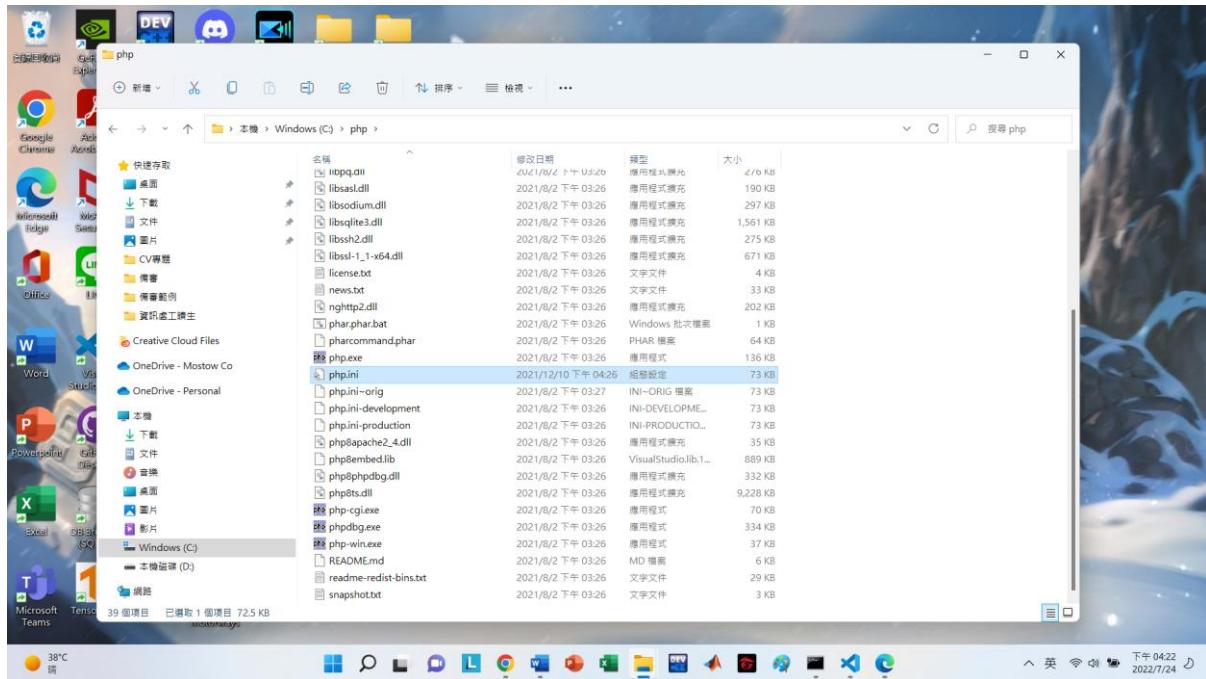
傳送表單的方式可以透過 GET 與 POST，  
如果利用 GET 傳送，  
會在網址的地方看到使用者輸入的內容



在特定情況下用 GET 傳送表單可能會讓資料被有心人士看到，  
所以傳送表單請盡量使用 POST

### 3. 建立資料庫 & Migrations & Models

檢查一下你們目前用的那個 php · 的環境配置檔 (php.ini) 有沒有開啟 sqlite3



若沒 php.ini 就複製一份 php.ini-development 把名字改成 php.ini ·

再打開修改把註解拿掉

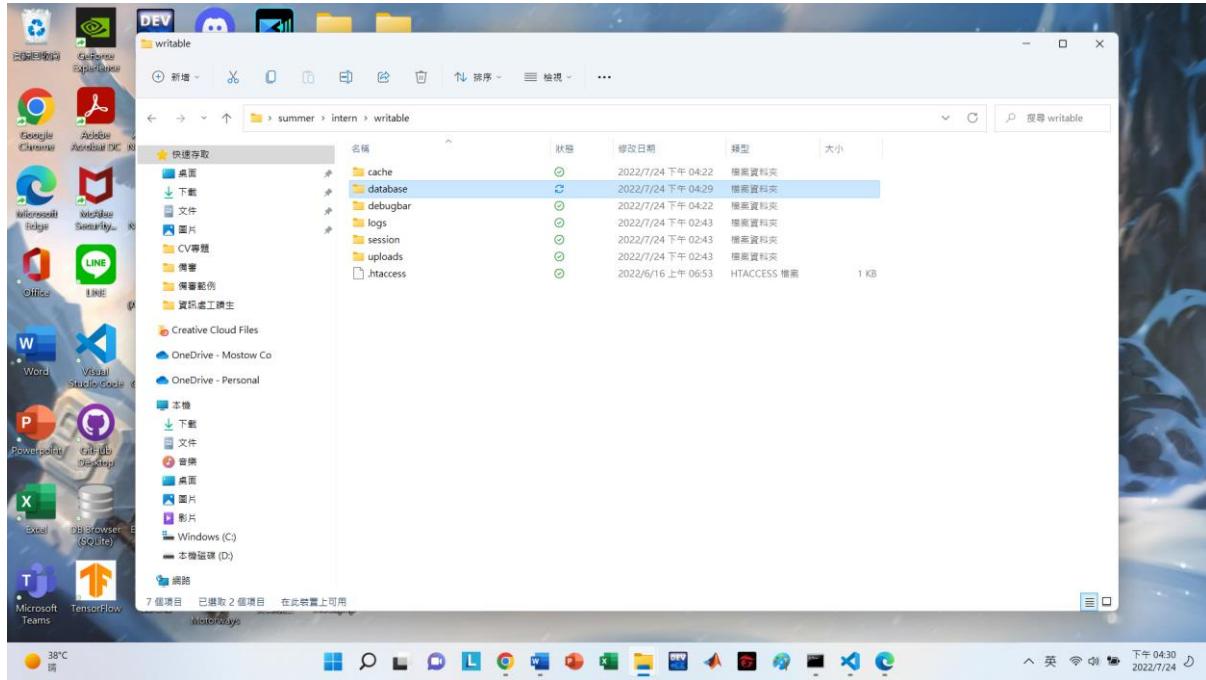
```
C:\> php > E:\php\php.ini
920 ;extension=bz2
921 extension=curl
922 ;extension=ffi
923 ;extension=ftp
924 extension=fileinfo
925 extension=gd
926 ;extension=gettext
927 ;extension=gmp
928 extension=intl
929 ;extension=imap
930 ;extension=ldap
931 extension=mbstring
932 ;extension=exif      ; Must be after mbstring as it depends on it
933 extension=mysqli
934 extension=oci8_12c ; Use with Oracle Database 12c Instant Client
935 extension=oci8_19  ; Use with Oracle Database 19 Instant Client
936 ;extension=odbc
937 extension=openssl
938 ;extension=firebird
939 extension=pdo_mysql
940 ;extension=pdo_oci
941 ;extension=pdo_odbc
942 extension=pdo_pgsql
943 extension=pdo_sqlite
944 ;extension=pgsql
945 ;extension=shmop
946
947 ; The MIBS data available in the PHP distribution must be installed.
948 ; See http://www.php.net/manual/en/snmp.installation.php
949 ;extension=snmp
950
951 ;extension=soap
952 ;extension=sockets
953 ;extension=sodium
954 extension=sqlite3
955 ;extension=tidy
956 ;extension=xsl
```

上面如果有的註解還沒拿掉的話 ·

也可以一起拿掉

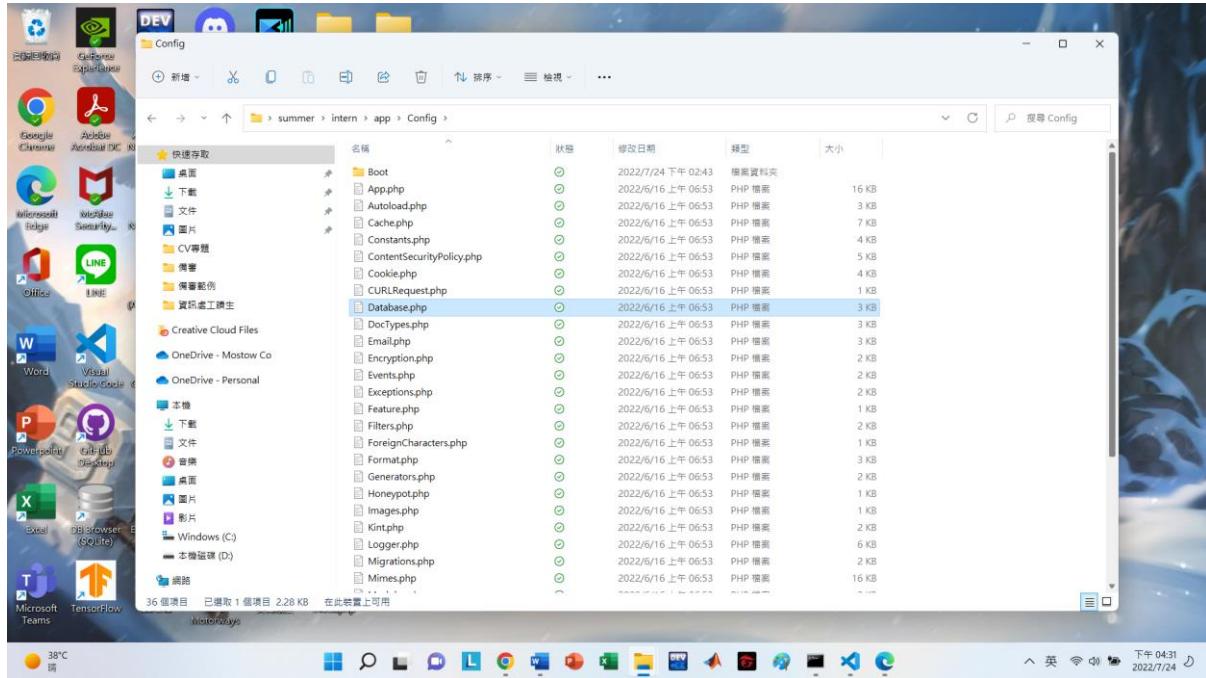
接著我們來建立資料庫，

先到 writable 創建一個資料夾 database

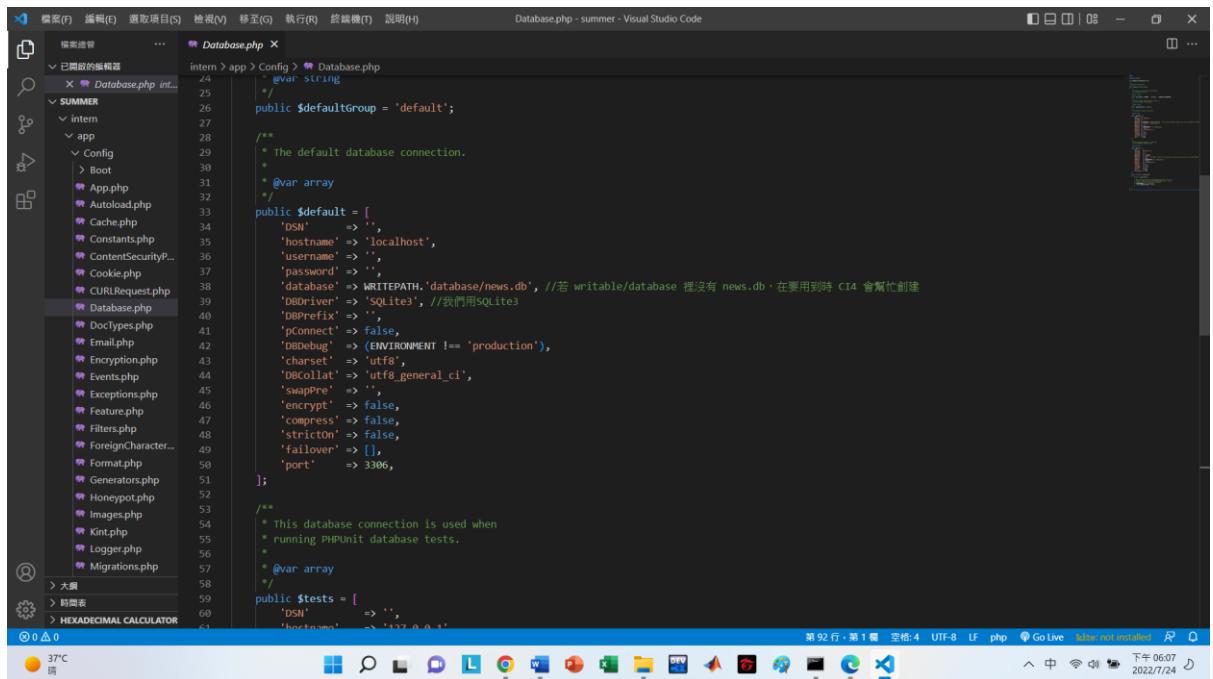


接著我們去配置 database 的藍圖，

到 app -> Config -> database.php



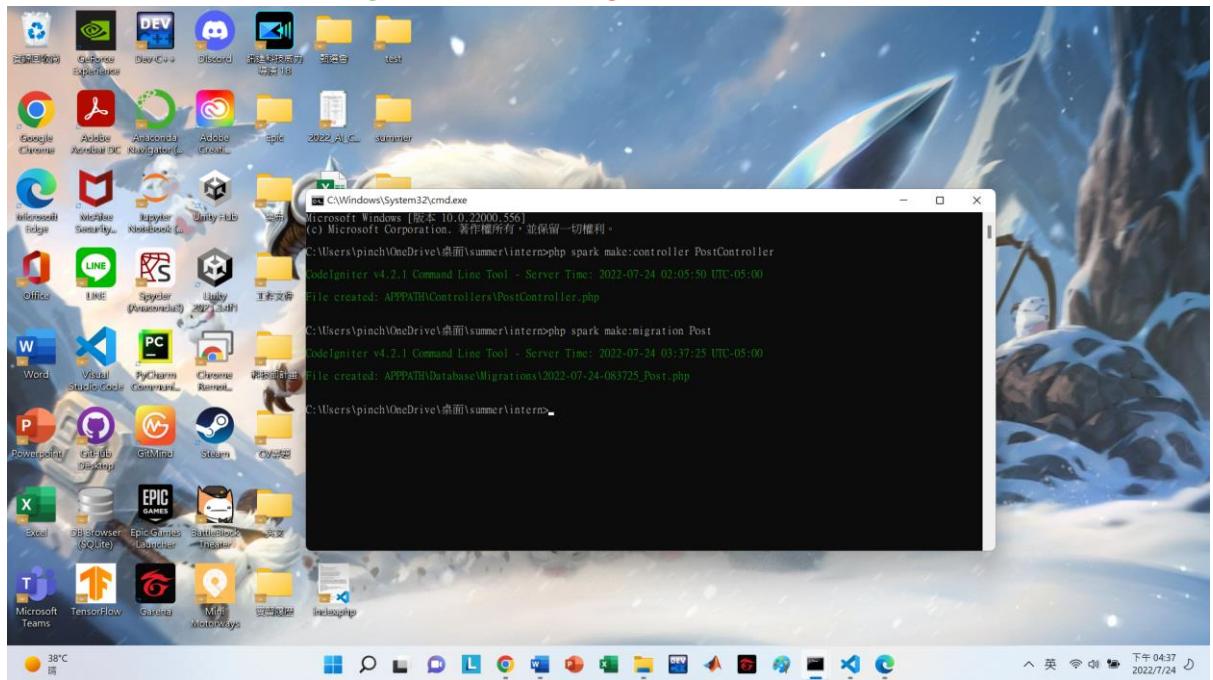
改下面這張圖的地方就好



```
Database.php
24     * @var string
25     */
26    public $defaultGroup = 'default';
27
28    /**
29     * The default database connection.
30     *
31     * @var array
32     */
33    public $default = [
34        'DSN' => '',
35        'hostname' => 'localhost',
36        'username' => '',
37        'password' => '',
38        'database' => WRITEPATH.'database/news.db', //若 writable/database 裡沒有 news.db，在要用到時 CI4 會幫忙創建
39        'DBDriver' => 'SQLite3', //我們用SQLite3
40        'DBPrefix' => '',
41        'pconnect' => false,
42        'DBDebug' => (ENVIRONMENT !== 'production'),
43        'charset' => 'utf8',
44        'DBCollat' => 'utf8_general_ci',
45        'swapPre' => '',
46        'encrypt' => false,
47        'compress' => false,
48        'strictOn' => false,
49        'failover' => [],
50        'port' => 3306,
51    ];
52
53    /**
54     * This database connection is used when
55     * running PHPUnit database tests.
56     *
57     * @var array
58     */
59    public $tests = [
60        'DSN' => '',
61        'hostname' => '127.0.0.1',
62    ];
63
```

回到 cmd 創建 migration 檔，

輸入 `php spark make:migration Post` (migration 的名字自己取)



migration 檔案簡單來說它就是一個資料庫裡 table 的模板，

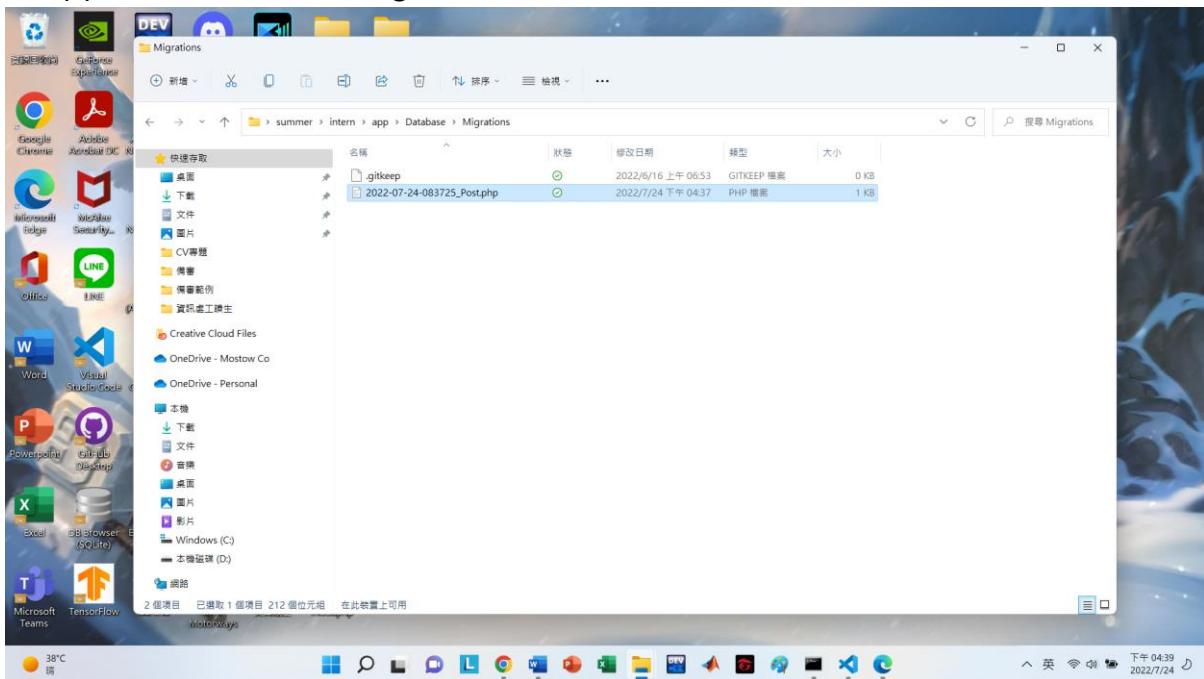
當一個團隊開發時你要建 table 時，

別人也要創一個一樣的 table，

這時比起跟他說自己的 table 有什麼東西在哪邊講半天，

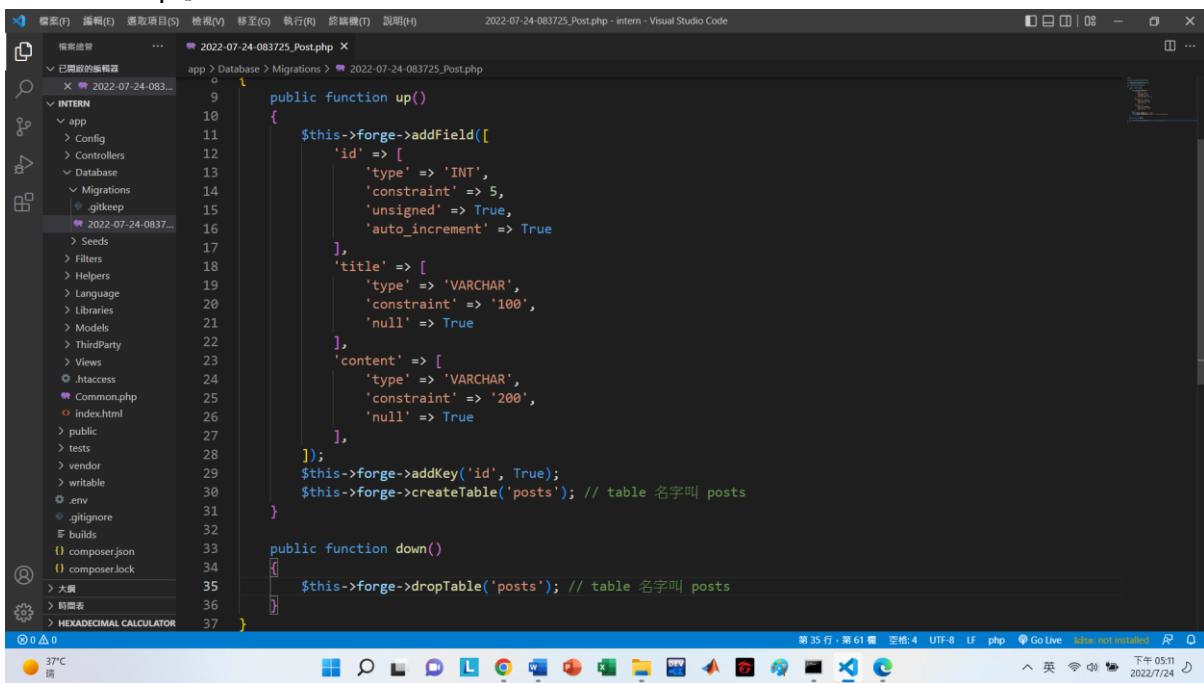
不如直接傳一個模板給他印就好

到 app -> Database -> Migrations 打開



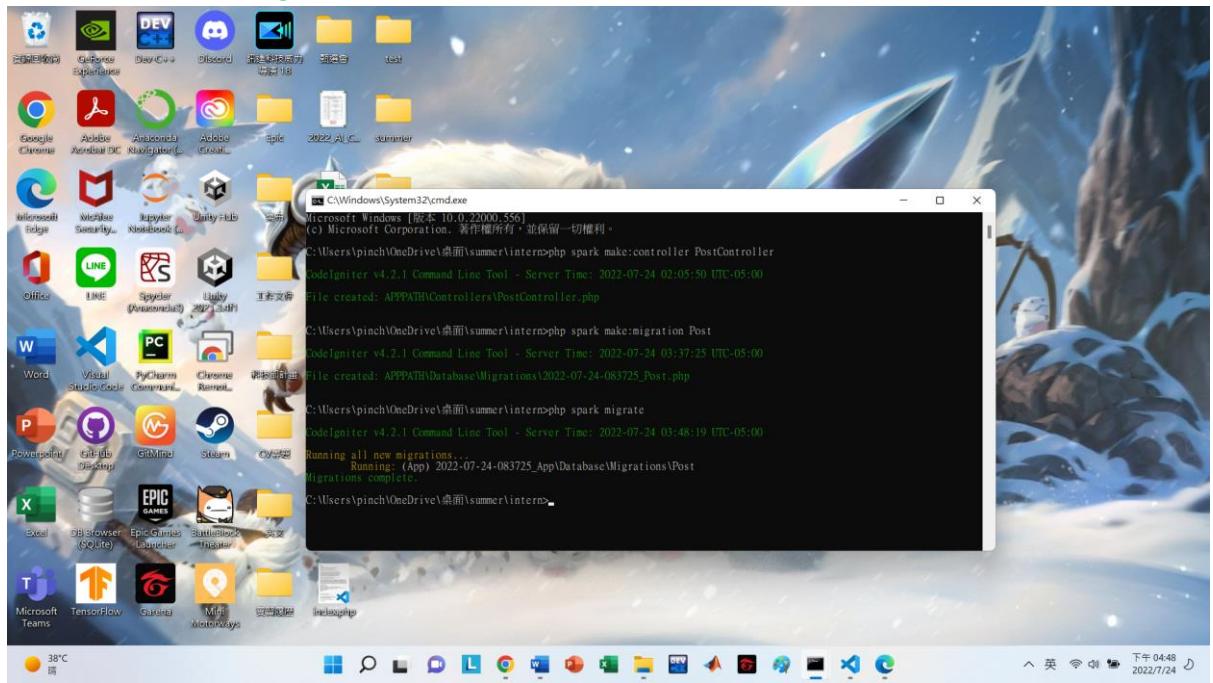
修改裡面的 function up() 和 function down()

function up() 就是放你之後 table 要長怎樣

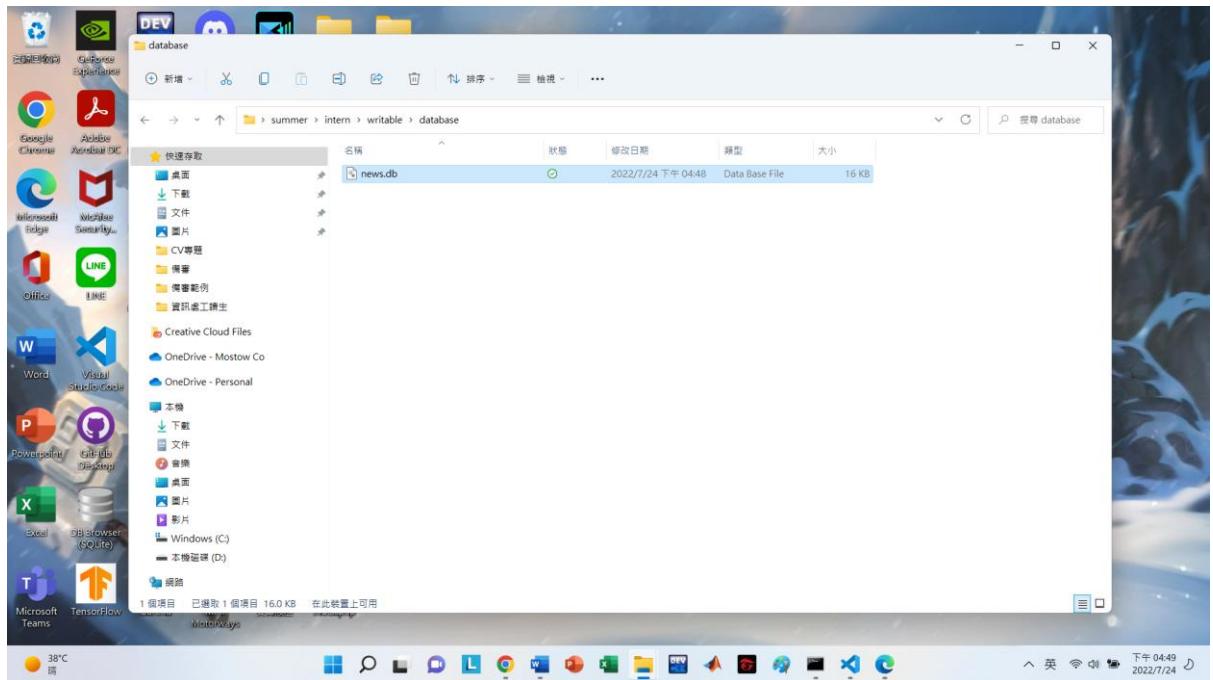


寫完後再回到 cmd 把 table 模板印到資料庫裡

輸入 `php spark migrate`



這時我們再去 writable -> database 檢查一下資料庫有沒有建好

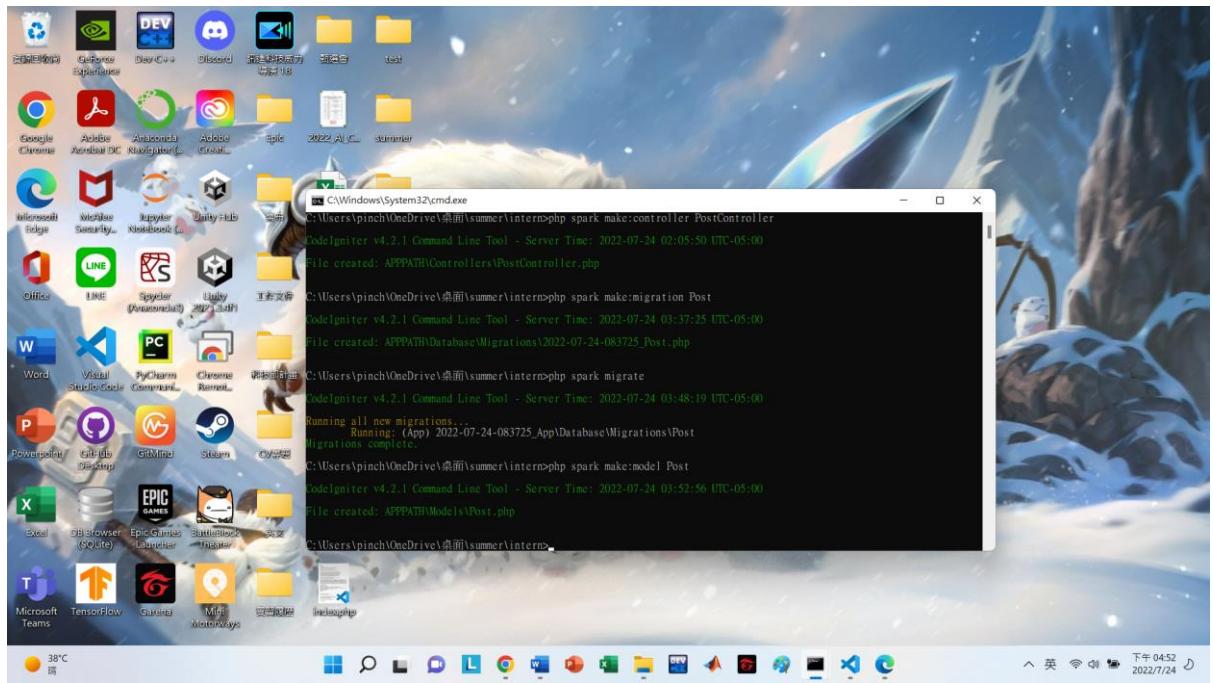


有看到 news.db 的話 ·

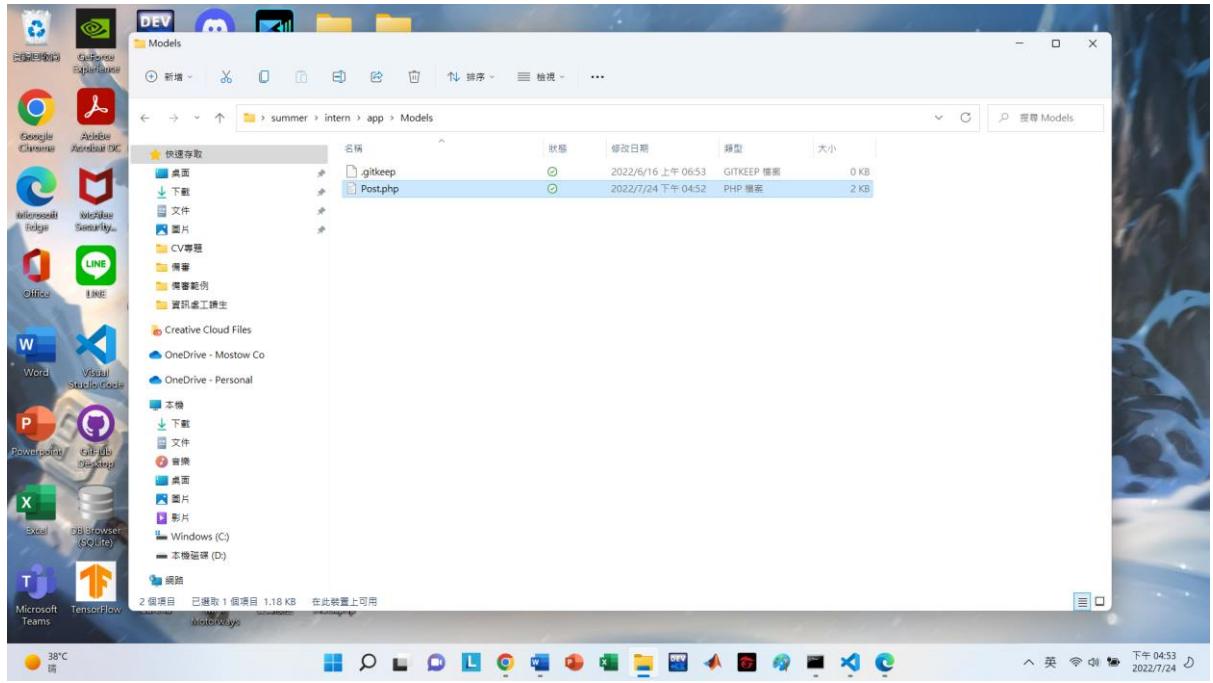
資料庫應該是沒太大的問題了

接著來創建 model

回到 cmd 輸入 `php spark make:model Post` (名字自己取)



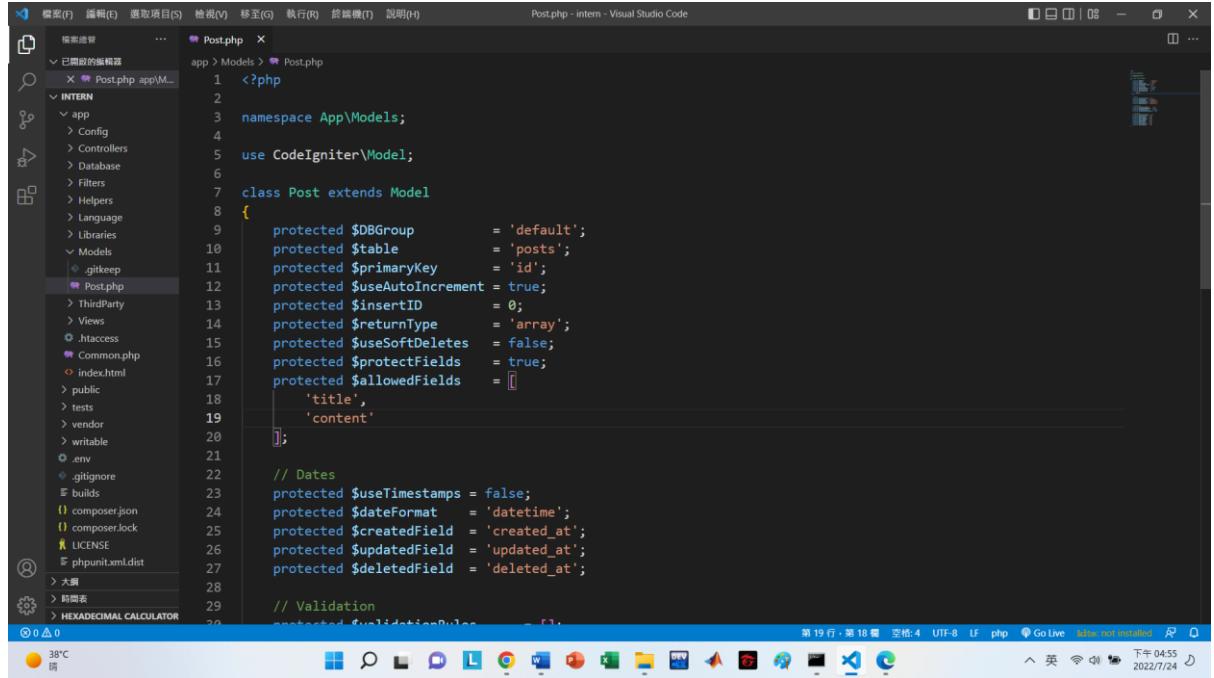
到 app -> Models 修改剛剛創建的 model (Post.php)



\$allowedFields 新增 table 允許修改的欄位，

然後檢查一下\$table 的名稱與資料庫的中的 table 名稱是否一致

(CI4 會自動用你 model 的名字取，但有時還是會出錯)



```
<?php
namespace App\Models;
use CodeIgniter\Model;

class Post extends Model
{
    protected $DBGroup      = 'default';
    protected $table        = 'posts';
    protected $primaryKey   = 'id';
    protected $useAutoIncrement = true;
    protected $insertID     = 0;
    protected $returnType   = 'array';
    protected $useSoftDeletes = false;
    protected $protectFields = true;
    protected $allowedFields = [
        'title',
        'content'
    ];

    // Dates
    protected $useTimestamps = false;
    protected $dateFormat    = 'datetime';
    protected $createdField  = 'created_at';
    protected $updatedField  = 'updated_at';
    protected $deletedField  = 'deleted_at';

    // Validation
    protected $validationRules = [
        ...
    ];
}
```

這裡科普一下，

我們在用 MVC 架構的時候，

取 Model 的名字和資料庫對應的 table 有一些約定成俗的習慣：

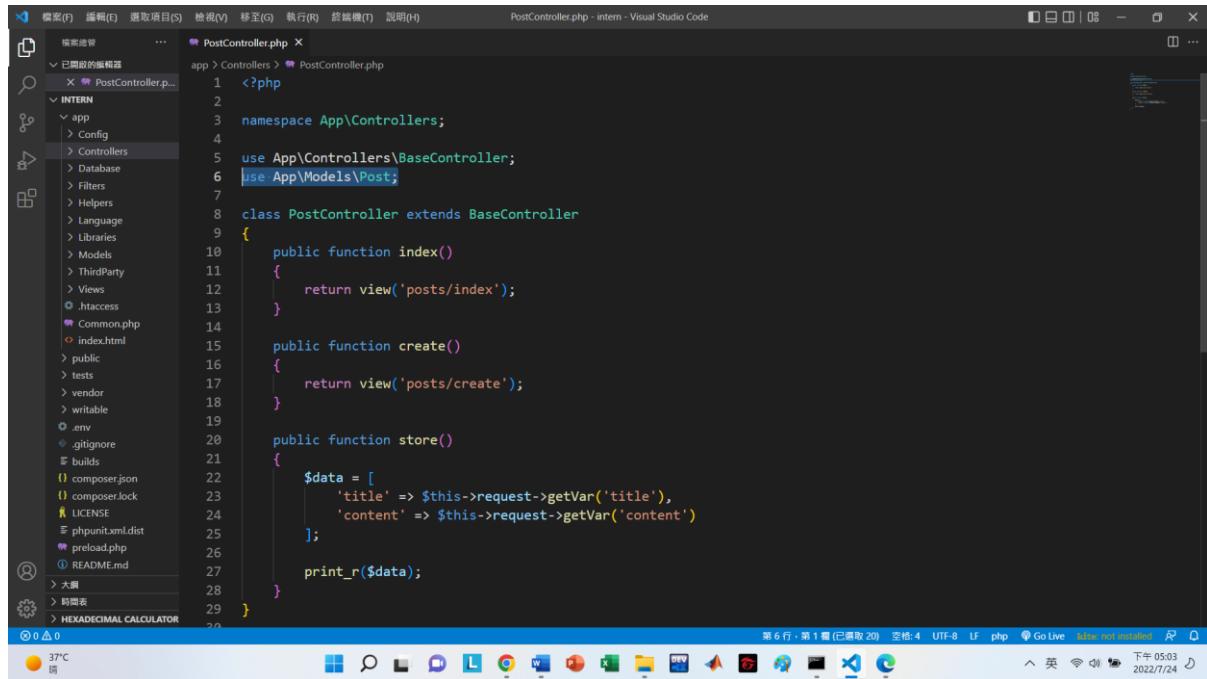
每個學校或公司的方法會有些許不同，

在這裡我們的 Model 第一個字大寫然後取單數 table 小寫且取複數

Model	table
Post	posts

## 4. 使用者寫文章

到 PostController 中引入剛剛的 Model Post



```
<?php
namespace App\Controllers;
use App\Controllers\BaseController;
use App\Models\Post;

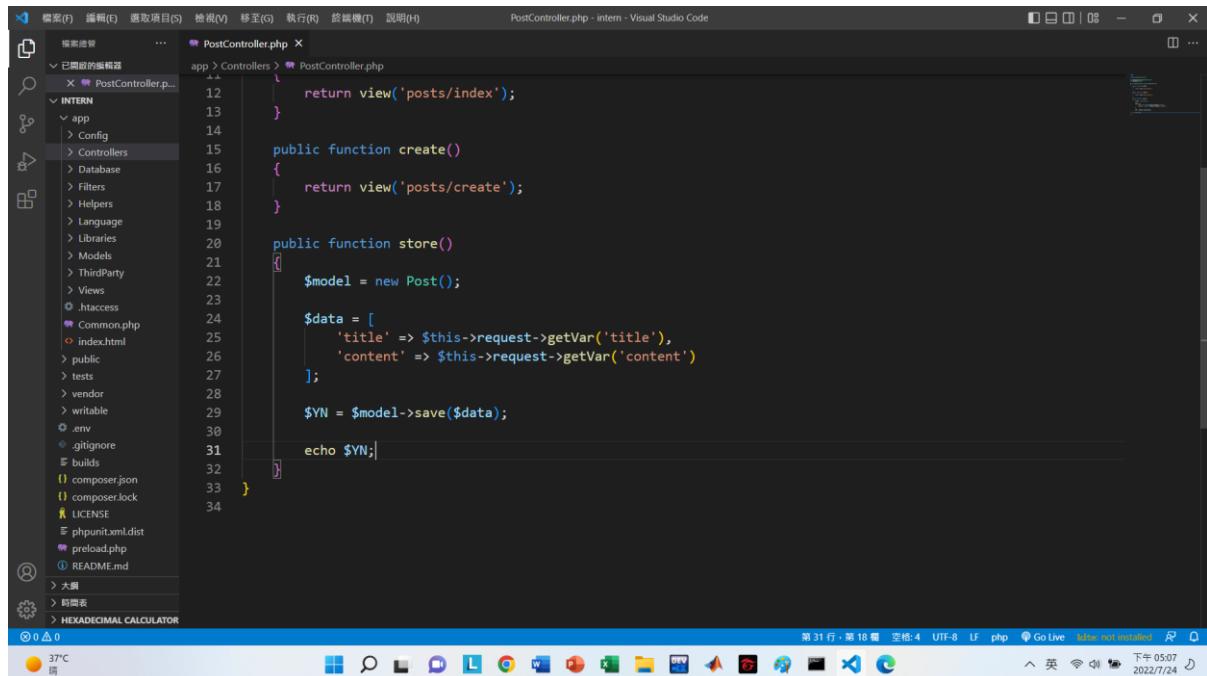
class PostController extends BaseController
{
    public function index()
    {
        return view('posts/index');
    }

    public function create()
    {
        return view('posts/create');
    }

    public function store()
    {
        $data = [
            'title' => $this->request->getVar('title'),
            'content' => $this->request->getVar('content')
        ];

        print_r($data);
    }
}
```

修改 PostController 中的 store function 如下



```

public function store()
{
    $model = new Post();

    $data = [
        'title' => $this->request->getVar('title'),
        'content' => $this->request->getVar('content')
    ];

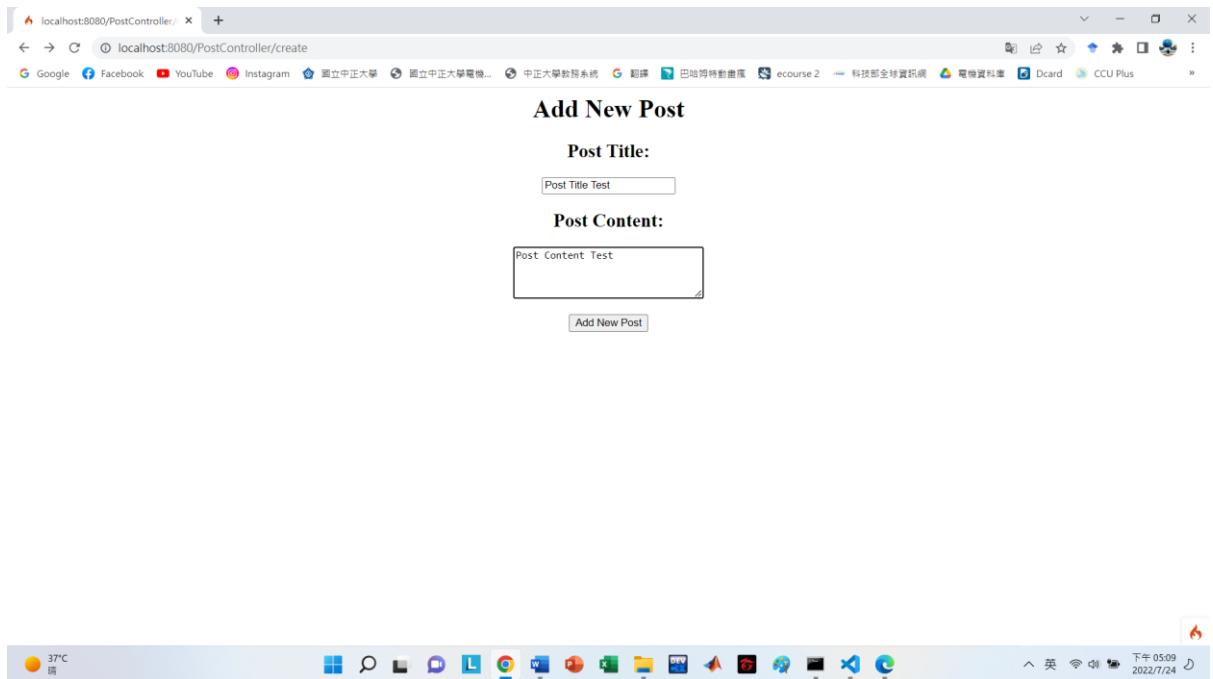
    $YN = $model->save($data);

    echo $YN;
}
```

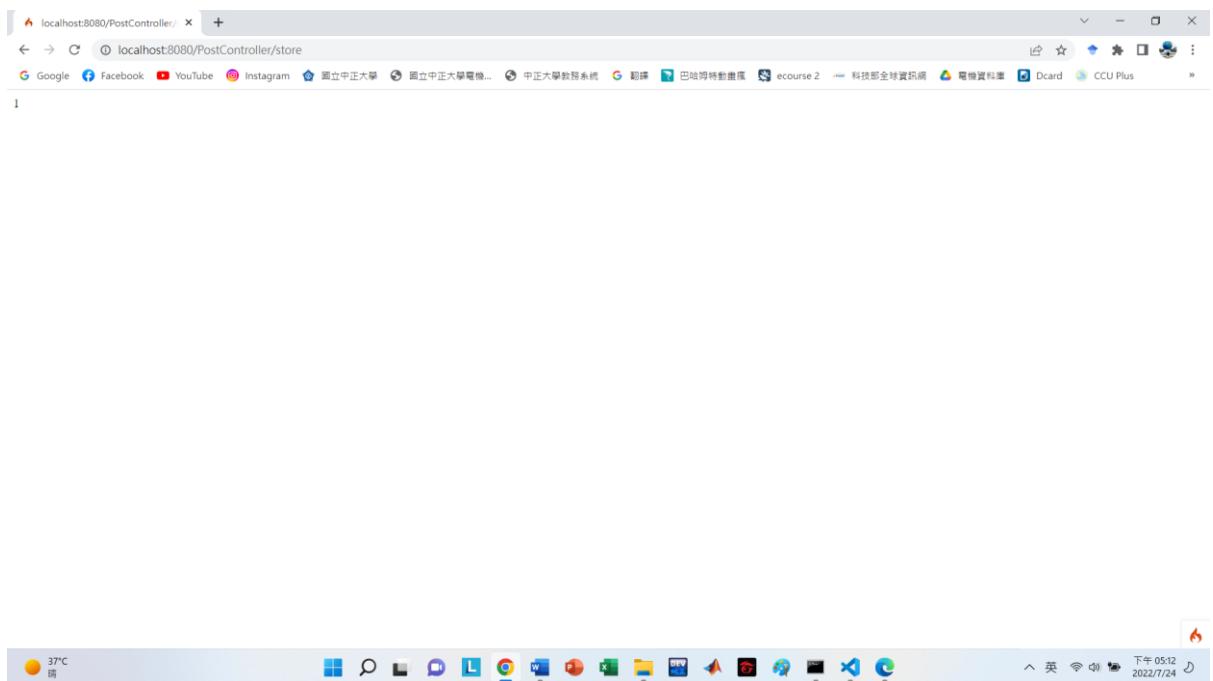
我們現在已經可以將頁面的資料傳到 Controller 了 ·

之後我們要做的就是把傳到 Controller 的資料存到資料庫

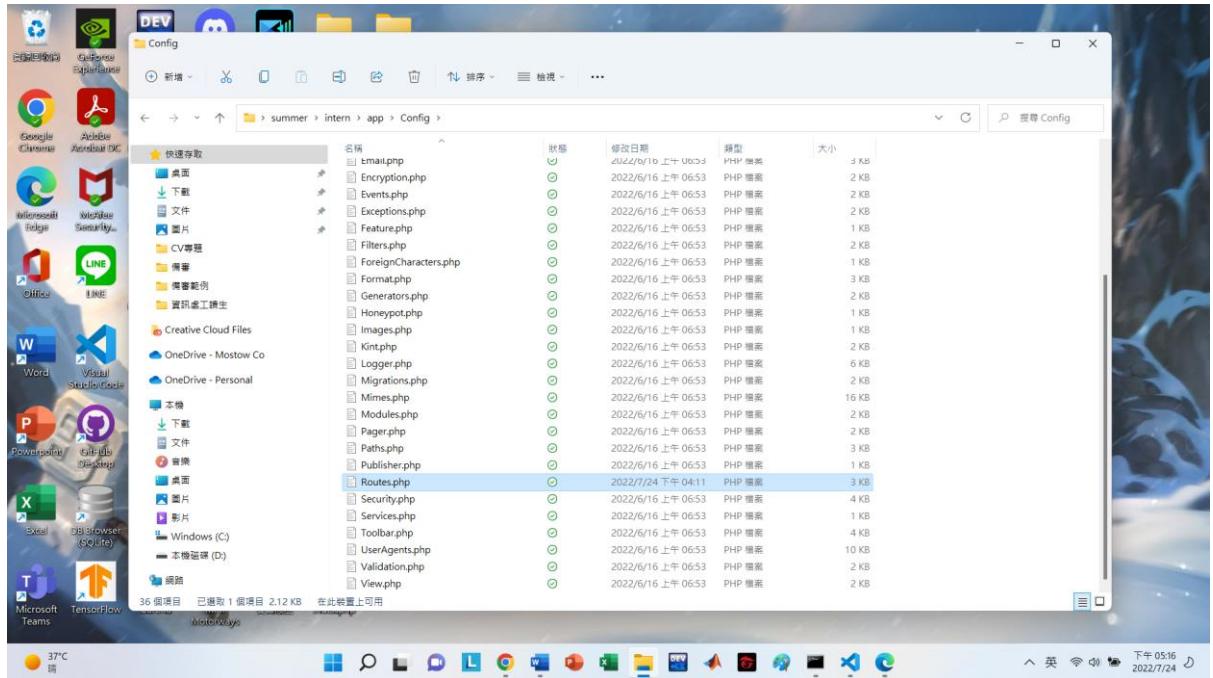
我們先試試看存不存得進去



若出現以下圖片表示有存進去



到這裡我們讓使用者寫文章 and 存檔的功能就做好了，  
接著就是要讓東西存完後系統自動跳到別的畫面，  
我們先到 app -> Config 的 Routes 新增路徑



新增這行如下圖

```

25 // where controller filters or CSRF protection are bypassed.
26 // If you don't want to define all routes, please use the Auto Routing (Improved).
27 // Set '$autoRoutesImproved' to true in 'app/Config/Feature.php' and set the following to true.
28 $routes->setAutoRoute(true);

30 /*
31 * -----
32 * Route Definitions
33 * -----
34 */

35 // We get a performance increase by specifying the default
36 // route since we don't have to scan directories.
37 $routes->get('/', 'Home::index');
39 $routes->get('PostController', 'PostController::index');

40 /*
41 * -----
42 * Additional Routing
43 * -----
44 */

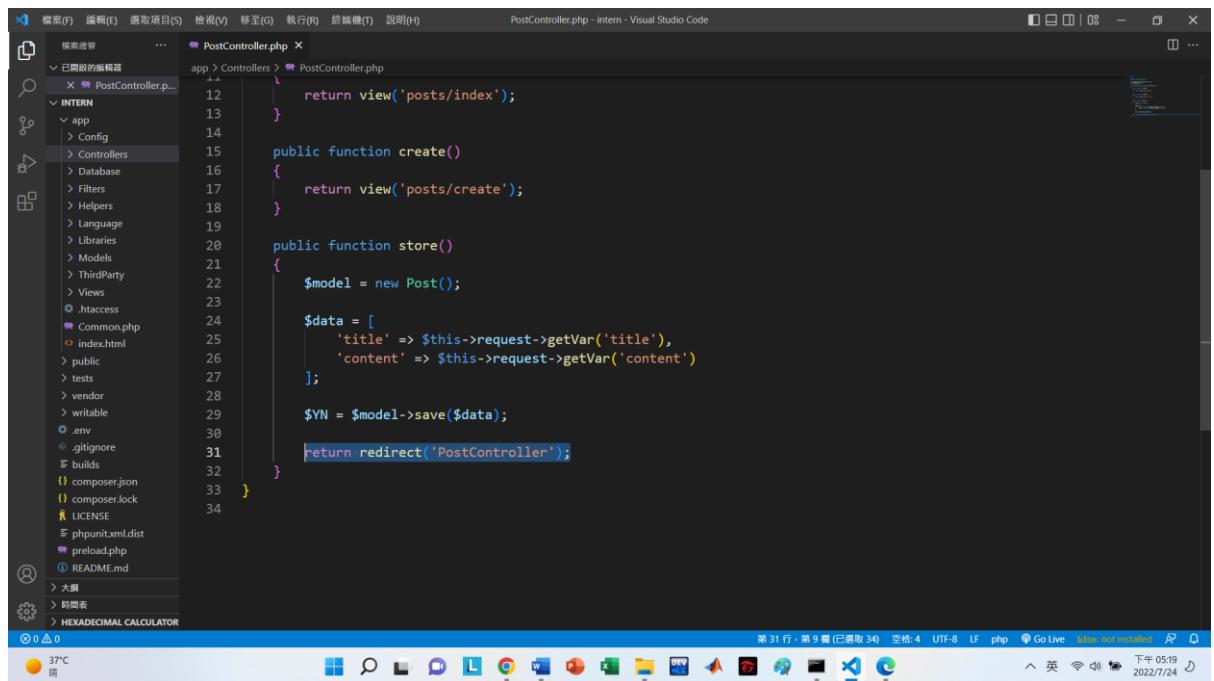
45 /*
46 * There will often be times that you need additional routing and you
47 * need it to be able to override any defaults in this file. Environment
48 * based routes is one such time. require() additional route files here
49 * to make that happen.
50 */
51 /*
52 * You will have access to the $routes object within that file without
53 * needing to reload it.
54 */

```

這意思是當你輸入在瀏覽器輸入網址的地方輸入localhost:8080/PostController 時，  
CI4 會幫你呼叫 PostController 中的 index function

之後在回到 PostController 修改一下 store function ,

讓它存完可以自己導回 index 頁面



The screenshot shows the Visual Studio Code interface with the file `PostController.php` open. The code is as follows:

```
12     return view('posts/index');
13 }
14
15 public function create()
16 {
17     return view('posts/create');
18 }
19
20 public function store()
21 {
22     $model = new Post();
23
24     $data = [
25         'title' => $this->request->getVar('title'),
26         'content' => $this->request->getVar('content')
27     ];
28
29     $YN = $model->save($data);
30
31     return redirect('PostController');
32 }
33 }
34 }
```

The code defines three methods: `index()`, `create()`, and `store()`. The `store()` method creates a new `Post` model, sets its title and content from the request variables, saves it, and then redirects back to the `PostController`.

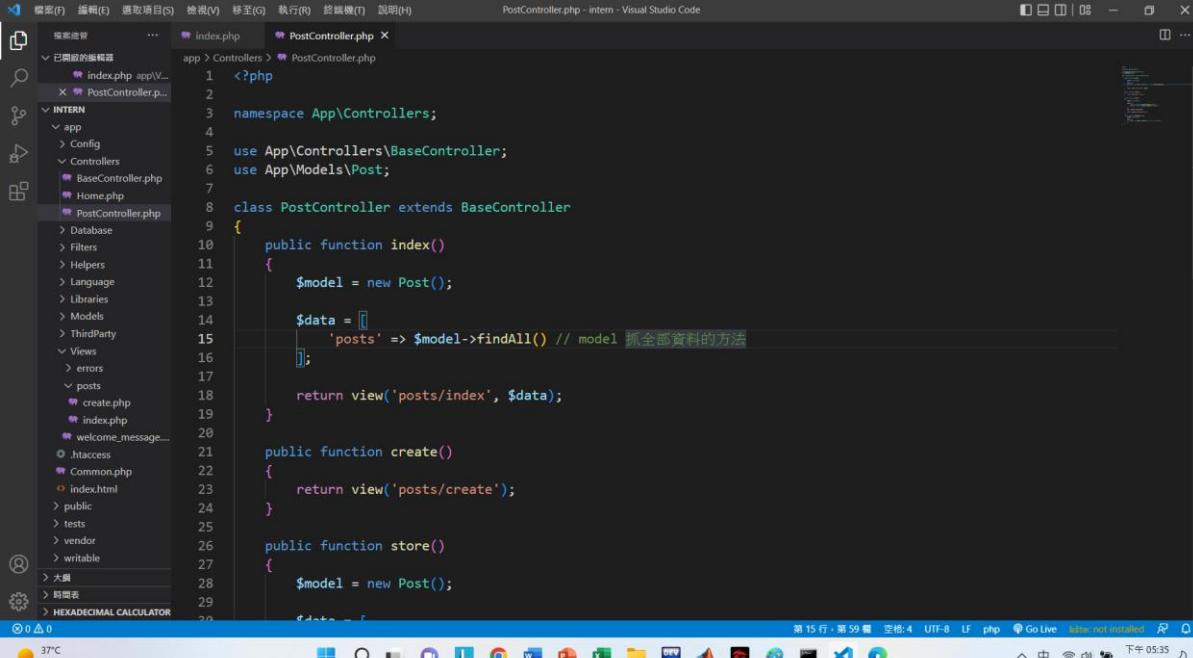
## 5. 使用者讀文章

接下來我們要做的就是從資料庫撈資料在從 Controller 傳資料到 View

首先我們先做一個目錄頁面(只顯示文章標題) ,

這邊是用 index 當目錄頁面 ,

想要創建新的頁面也可以



```
<?php
namespace App\Controllers;
use App\Controllers\BaseController;
use App\Models\Post;

class PostController extends BaseController
{
    public function index()
    {
        $model = new Post();

        $data = [
            'posts' => $model->findAll() // model 摘全部資料的方法
        ];

        return view('posts/index', $data);
    }

    public function create()
    {
        return view('posts/create');
    }

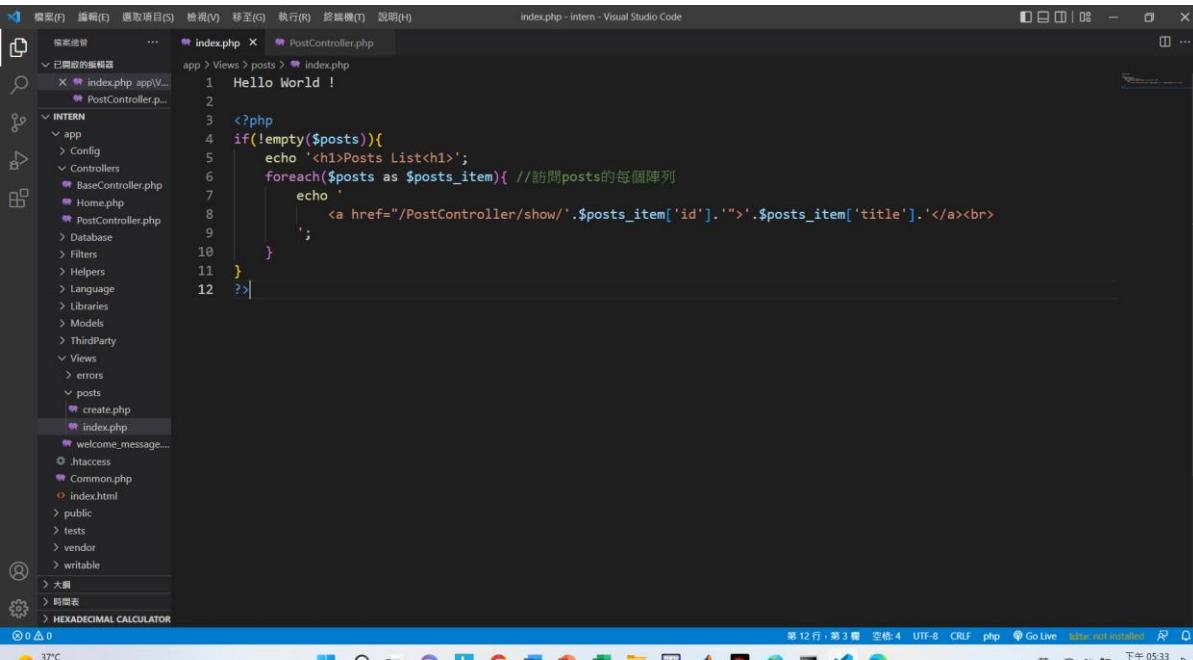
    public function store()
    {
        $model = new Post();
    }
}
```

CI4 Model 訪問資料的用法 :

<https://codeigniter4.github.io/userguide/models/model.html>

我們在去修改 app ->views->posts->index.php .

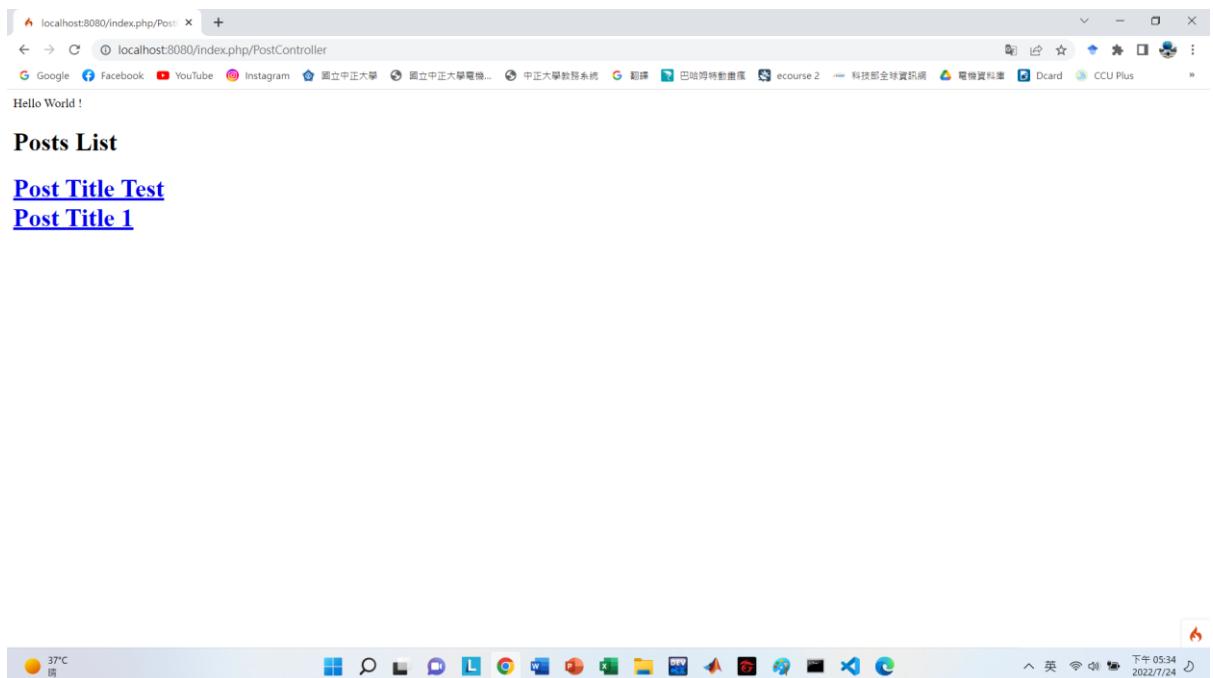
把收到的資料 print 出來



```
Hello World !

<?php
if(!empty($posts)){
    echo '<h1>Posts List<h1>';
    foreach($posts as $posts_item){ //訪問posts的每個陣列
        echo '
            <a href="/PostController/show/'.$posts_item['id'].'">'.$posts_item['title'].'
        </a><br>
    ';
}
?>
```

## 來看看結果



現在目錄做好了，

接著就是要看文章詳細內容的頁面。

先來刻 PostController 中的 show function

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "Intern". The "posts" folder is expanded, showing files like "create.php", "index.php", and "show.php".
- Code Editor (Center):** Displays the content of "PostController.php". The code handles POST requests to save new posts and GET requests to show existing posts.
- Status Bar (Bottom):** Shows the current file is "PostController.php - intern - Visual Studio Code", the line count is "第 47 行, 第 42 檔", the character count is "變態: 4", and the encoding is "UTF-8". It also includes icons for Go To Line, Find, and Replace.

```
index.php PostController.php
app > Controllers > PostController.php

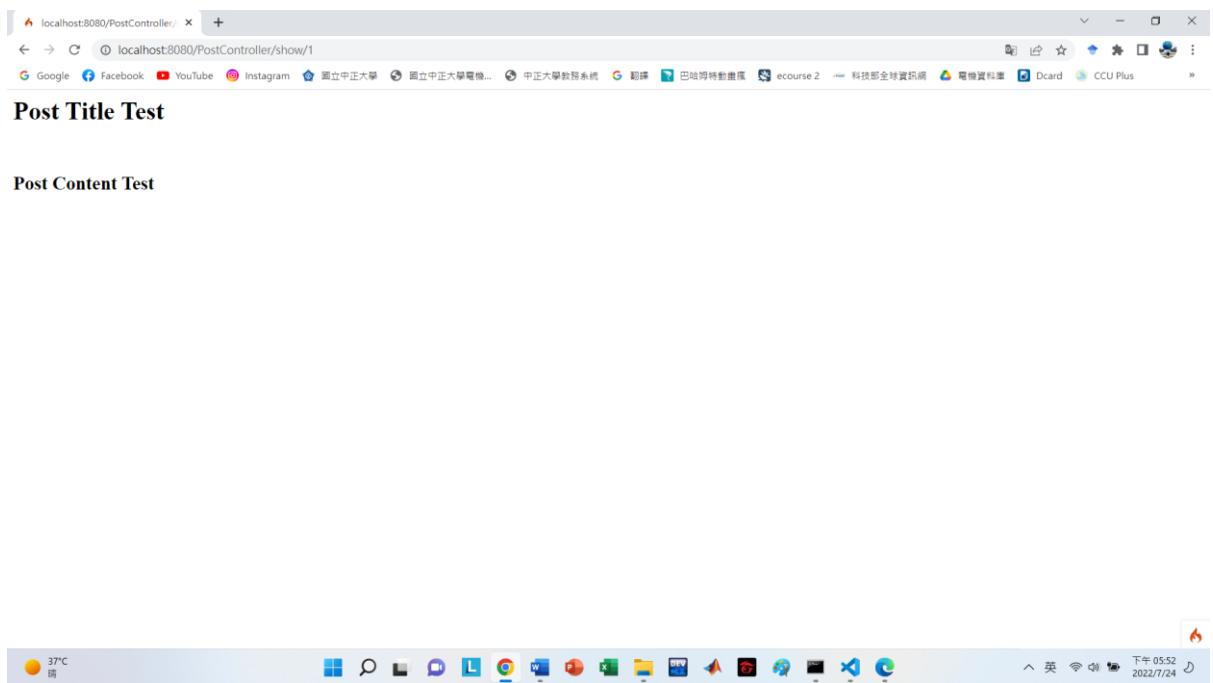
29
30     $data = [
31         'title' => $this->request->getVar('title'),
32         'content' => $this->request->getVar('content')
33     ];
34
35     $YN = $model->save($data);
36
37     return redirect('PostController');
38 }
39
40 public function show($post_id){
41     $model = new Post();
42
43     $data = [
44         'posts' => $model->find($post_id) // model 抓特定資料的方法
45     ];
46
47     return view('posts/show', $data);
48 }
49 }
```

再來刻某一文章詳細內容的頁面

```
show.php x PostController.php
app > Views > posts > show.php
1 <h3><?php echo $posts['title']?></h3>
2 <br>
3 <h5><?php echo $posts['content']?></h5>
```

The screenshot shows the Visual Studio Code interface with the file structure on the left and the code for 'show.php' on the right. The code prints the title and content of a post from a database.

詳細內容頁面如下



恭喜你！！！

公告系統的基礎功能：寫文章、讀文章都做完拉