Laravel安裝：於安裝目錄輸入指令 composer create-project laravel/laravel **laravel 5.2.\*** --prefer-dist (粗體為**創建目錄** **版本號**)

一、路由控制器與視圖

|  |
| --- |
| 1.路由設置 (app/Http/routes.php)  //格式：Route::方法((方法補充),路徑,執行);  //方法可為：get post match(需要方法補充) any put delete，較複雜的有group controller resource  Route::get( 'test01', function () {return 'test01';});  Route::match(['get','post'], 'test02', function () {return 'test02';}); //含方法補充  //路徑可為：'一般字串/{正規則名}'，正規則最後需加入->where(['規則名'=>'[規則]']);  Route::get('test03/{preg?}', function ($preg) { //正規則名preg，最後加?表此規則可忽略  return $preg; //正規則名可當變數傳入(選擇性)  })->where(['preg' => '[0-9]+']); //指派正規則  //執行可為：呼叫控制器、匿名函數、[中間件指派,匿名函數]  Route::get('test04','test@index'); //呼叫控制器test之class@index  Route::controller('test05','test'); //呼叫控制器test之class@getIndex  Route::get('test06',['middleware'=>'auth' //中間件指派  ,function () {return 'test06';}]);  Route::resource('test07','test'); //依據請求方法呼叫不同的class：  //get / @index; get /create @create; get /{id} @show; get /{id}/edit @edit;  //post / @store; put/patch /{id} @update; delete /{id} @delete;  //其它  　 //中間件使用  Route::get('home',['middleware'=>'auth','uses'=>'HomeController@index']);  　 //群組指派  Route::group(['middleware'=>'auth'], function(){  Route::get('test08',function () {return 'test08';});  Route::get('test09',function () {return 'test09';});  });  　 //命名路由 (鍵名as為路由名稱，uses為路由執行)  Route::get('/welcome', [  'as' => 'Welcome',  'uses' => 'HomeController@index'  ]);  $url = action('FooController@method'); //透過路由執行取得網址  $url = route('name'); //透過路由名稱取得網址 |
| 2.控制器設置 (app/Http/controllers/Home.php (文件名自定))  <?php  namespace App\Http\Controllers;  use Illuminate\Routing\Controller as BaseController;  class Home extends BaseController  {  public function index() //控制器Home.php的class index  {  $data = ['a'=>'顯示資料']; //view只接受組合陣列，會使用extract拆開陣列  return view('Home', $data); //回傳視圖  }  } |

|  |
| --- |
| 3.視圖基本設置 (resources/views/Home.blade.php (Home為呼叫名，必須加.blade))  <html>  全是html語法，{{$test}} <--相當於echo $test，變數的顯示  全部的邏輯語法皆需要在前面加老鼠@，且不能敘述大括號，要用end做結尾  例如： @foreach ($test = $key=>$value)  變數{{$key}}=>{{$value }} <br>  **@endforeach**  </html> |
| 4.繼承視圖布局  概念：主架構**布局**文件(包括<header><main><footer>)、**子視**圖文件(僅main裡的section之類內容)   1. 任意引入視圖： @include('視圖呼叫名') 2. 子視繼承布局： @extends('布局呼叫名') 3. 布局呼叫子視區塊： @yield('子視區塊名') 4. 布局子視交叉區塊： (由子視覺定整體區塊排版)  |  |  | | --- | --- | | **布局：**  @section('同區塊名')  布局區塊內容  @show <!—布局區塊結束--> | **子視：**  @section('同區塊名')  子視區塊內容  @parent <!—布局區塊內容-->  子視區塊內容  @stop <!—子視區塊結束，stop同endsection--> |  1. 子視區塊格式： @extends('布局呼叫名') @section 子視為數個區塊所組成，透過布局安插 @stop |

二、服務的創建與使用

|  |
| --- |
| 1.概述：  Laravel的服務就是class，創建很容易，就是在路徑App\Services\下創造服務同名檔案，一個class一個檔案。  路由與控制器皆可以使用服務，方法有三種：  [A].使用use命名空間直接載入服務。  [B].透過自動依賴注入的方式，使用服務。  [C].透過bootstrap/app 將服務綁定至容器 (由make解析後使用，參見[E])。  [D].透過ServiceProvider將服務綁定至容器 (由make解析後使用，參見[E])。  [E].使用make解析服務，在router或是controller使用。 |
| 2.服務創造：app下創造服務資料夾/服務文件 (EX App\Services\GeneralService.php)  <?php  namespace App\Services;  class GeneralService{} |
| [A].使用use命名空間直接載入服務。  use App\Services\GeneralService;  $GeneralService = new GeneralService(); |
| [B].透過自動依賴注入的方式，使用服務。  function(\App\Services\GeneralService $generalService){  Sservice1 = new $generalService();  } |
| [C].透過bootstrap/app 將服務綁定至容器 (由make解析後使用，參見[E])。  　在bootstrap/app.php中，將服務綁至服務容器，綁定名稱可以隨意定義。  $app->bind(App\Services\GeneralService::class,function($app){  return new App\Services\GeneralService(); }); //普通回調函數綁定  $app->singleton(App\Services\GeneralService::class,function($app){  return new App\Services\GeneralService(); }); //單例回調函數綁定  $instance = new App\Services\GeneralService();  $app->instance('GeneralService',$instance); //實例對象服務綁定 |
| [D-1].透過ServiceProvider將服務綁定至容器 (由make解析後使用，參見[E])。  第一部分：註冊，在config/app.php中的'providers'進行提供者註冊，EX：  App\Providers\TestServiceProvider::class, |

|  |
| --- |
| [D-2].透過ServiceProvider將服務綁定至容器 (由make解析後使用，參見[E])。  第二部分：服務資料夾下創造提供綁定文件 (EX App\Providers\TestServiceProvider.php)  <?php  namespace App\Providers;  use Illuminate\Support\ServiceProvider;  use App\Services\GeneralService; //use服務  class TestServiceProvider extends ServiceProvider  {  protected $defer = true; //緩載，讓沒用到的服務不會載入  public function boot(){  //boot方法會在所有提供者的register完成後才註冊  }  public function register()  {  //綁定名稱可以隨意定義，注意下面的GeneralService繼承了use路徑  $this->app->bind(GeneralService::class,function($app){  return new GeneralService(); }); //普通回調函數綁定  /\*$this->app->singleton(GeneralService::class,function($app){  return new GeneralService(); }); //單例回調函數綁定  $instance = new GeneralService();  $this->app->instance('GeneralService',$instance); //實例對象服務綁定\*/  }  public function provides() //緩載時要提供所有的註冊名  {  return [  GeneralService::class,  /\*'GeneralService'\*/  ];  }  } |
| [E].使用make解析服務，在router或是controller使用。  　第一部分：在routes裡依據綁定名稱解析服務，解析方法有以下四種，效果相同。  $GeneralService = $this->app->make(App\Services\GeneralService::class); //透過服務容器make解析  $GeneralService = $this->app[App\Services\ServiceConTract::class]; //透過綁定接口解析  $GeneralService = app(App\Services\GeneralService::class); //透過全域函數app解析  $GeneralService = \App::make(App\Services\GeneralService::class); //透過Facades容器外觀解析  　第二部分：在controller裡依據綁定名稱解析服務。  use App;  $GeneralService = App::make(App\Services\GeneralService::class);  (p.s 自動依賴注入的綁定方式在Laravel5.2之後就沒反應，很奇怪...) |

三、表單使用

|  |
| --- |
| 1.路由設置 (app/Http/routes.php)  Route::match(['get','post'],'welcome','WelcomeController@index'); //將收到表單資料的文件修改為POST呼叫 |
| 2.表單發送 (控制器+視圖)  use Illuminate\Http\Request; //獲取請求函數  ..class..  public function \_\_construct (Request $request){} //從函數初始載入中獲取接收請求  $request->session()->save(); //記錄session狀態  <!--所有表單資料欄位都必須有name，表單用post發送-->  ..form..  {!! csrf\_field() !!} <!--post必須要加入csrf驗正參數，效果同下hidden行，選一即可-->  <input type="hidden" name="\_token" value="{{ csrf\_token() }}"> |
| 3.控制器接收請求 (控制器)  use Illuminate\Http\Request; //獲取請求函數  ..class..  public function \_\_construct (Request $request) //從函數初始載入中獲取接收請求  {  //表單內容資料  **$request->all(); //全部資料(含post、get)**  $request->input(); //檔案類型以外全部資料(含post、get)  $request->query(); //僅get全部資料  $request->input('name'); //單一名稱資料  $request->only('name1', 'name2'); //複數名稱資料  $request->except('name1', 'name2'); //除此之外(複數)名稱資料  //其他資料  $request->method(); //方法  $request->url(); //網址(不含#或get參數)  $request->fullUrl(); //完整網址(含#或get參數)  $request->path(); //網址僅路徑(不含網域、#或get參數)  } |
| 4.接收請求session一次存儲 (控制器)  use Illuminate\Http\Request; //獲取請求函數  //存儲方法  **$request->flash(); //所有資料存儲**  $request->flashOnly('name1', 'name2'); //(複數)名稱資料存儲  $request->flashExcept('name1', 'name2'); //除此之外(複數)名稱資料存儲  //獲取方法  **$request->old(); //所有資料獲取**  $request->old('name'); //單一資料獲取 |

四、模組資料庫使用：僅資料表數據管理 (Laravel存取的資料表強制必須要有一個AI的id)

|  |
| --- |
| 1.資料庫登入資訊：根目錄 .env 檔案  記得.env同層的.htaccess必須加入保護語法：<Files .env>deny from all</Files> |
| 2.創造資料表模組：app/Models/Student.php (默認檔案名稱隨資料表名稱，檔名首小寫+s = 資料表名(students)，一表一模組)  <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Model;  class Student extends Model  {  protected $table = 'students'; //自定表名 (若不隨默認表名對應設置)  public $timestamps = false; //關閉自動時間戳，時間戳為表內的DATETIME格式created\_at、updated\_at參數  protected $fillable = ['name', 'age']; //允許大量賦值 (迴圈更新資料必須開啟)  protected $guarded = ['name', 'age']; //拒絕大量賦值  } |
| 3.讀取資料 (控制器，若將讀取資料函數寫在模組，則把Student::改成$this->)  <?php  use App\Models\Student; //讀取資料表模組  ..class..  $student = Student::all(); //讀取所有數據 (**陣列**)  $student = Student::find(1); //讀取id=1資料  $student = Student::where('id','=','1')->get(); //讀取條件數據 (**陣列**) (當有first lists等最後位時，get會被取代)  $student = Student::whereRaw("(age >= 0) or (id > 200)")->get(); //讀取字串條件數據 (**陣列**)  //**陣列**可在最後加入 ->take(數量) ，限制提取之最大數量  //**陣列**可在最後加入 ->count() ，將會回傳資料筆數  //**陣列**可在最後加入 ->skip(數量) ，將會忽略資料  //**陣列**可在最後加入 ->first() ，將回傳第一筆資料  //可在最後加入　　 ->lists(V欄,K欄)，將會把K欄當鍵做組合陣列  //可在中間加入　　 ->select('a', 'b') ，將會只取a b兩欄的值  //可在中間加入　　 ->distinct() ，將會抓取不重複的值  //可在中加加入　　 ->orderBy('name', 'desc')，或是asc進行排序  //以下是屬性獲取方法  $data = $student->toArray(); //將資料以組合陣列顯示 |
| 4.寫入資料  <?php  //插入數據  $student = new Student;  $student->name = '小企鵝';  $student->age = '28';  $student->save(); //儲存插入法  **Student::create(['name'=>'John' , 'age'=>'30']); //創造插入法**  **$Student = Student::firstOrCreate(['name' => 'John']); //創造插入法，若重複則不插入，回傳第一筆發現重複的資料**  //修改數據  $user->touch(); //僅更新時間戳  $student = Student::find(1);  $student->name = 'john';  $student->save(); //儲存修改法  **student::find(3)->update(['name' => 'hahahaha']); //更新修改法 find**  **student::where('age','>','10')->update(['name' => 'aloha']); //更新修改法 where**  **student::whereRaw("name = 'aloha'")->update(['name' => 'kkr']); //更新修改法 whereRaw**    //刪除數據  $student = Student::find(1); $student->delete(); //刪除id=1資料  **student::destroy(1); student::destroy([1, 2, 3]); student::destroy(1, 2, 3); //刪除id符合資料**  **student::where('age','>','10')->delete(); //刪除條件數據**  **student::whereRaw("name = 'aloha'")->delete(); //刪除字串條件數據** |

|  |
| --- |
| 5.高級語法  //join語法  //?use Illuminate\Database\Seeder; 用來extends? extends Seeder?  DB::table('users')  ->join('contacts', function($join)  {  $join->on('users.id', '=', 'contacts.user\_id')  ->where('contacts.user\_id', '>', 5) ->where('contacts.user\_id', '>', 5);  })  ->get();  DB::table("users")->delete(); //清空資料表 |

五、Session的使用

|  |
| --- |
| 1.Session啟用 (控制器)  use Illuminate\Http\Request; //獲取請求  ..class..  public function index(Request $request) {... //從函數的初始參數中載入請求 |
| 2.Session讀取 (控制器、視圖)  $session = $request->session()->all(); //讀取所有值  $session = $request->session()->get('\_token'); //讀取單一值  $session = session('\_token'); //讀取單一值  $session = $request->session()->has('\_token '); //是否有此鍵 |
| 3.Session修改 (控制器)  $request->session()->put('name','小芳'); //依鍵的值新增或修改 (值)  $request->session()->push('stack', '123'); //依鍵創造或新增陣列 (陣列)  $request->session()->forget('name'); //依鍵刪除  $value = $request->session()->pull('name', 'default'); //依鍵刪除，同時取出刪除資料 (若無刪除資料，回傳'default')  $request->session()->flush(); //清空Session  $request->session()->regenerate(); //重置整個Session ID (安全性考量)  $request->session()->save(); //\*\*必要\*\* 所有修改執行後必須儲存 |

六、authorize驗證模塊(middleware)使用

|  |
| --- |
| 1.視圖安裝&路由配置  CMD ->至laravel目錄->指令輸入：  php artisan make:auth |
| 2. migrations模組匯入 (laravel內建有users 與password\_resets兩個版)，於database/migrations  由於預設之users 與password\_resets符合auth之預設格式，因此直接執行artisan即可。(記得配置.env)  CMD ->至laravel目錄->指令輸入：  php artisan migrate |
| 3.控制器權限中間件，創建測試home控制器如app/Http/Controllers/HomeController.php  <?php  namespace App\Http\Controllers;  use App\Http\Requests;  use Illuminate\Http\Request;  use Illuminate\Support\Facades\Auth; //驗證模組  class HomeController extends Controller  {  public function \_\_construct()  {  $this->middleware('auth',['only'=>['index','index2']]); //only只有限制  //$this->middleware('auth',['except'=>['index','index2']]); //except除外限制  //$this->middleware('auth'); //完全限制  }  public function index()  {  return view('home');  /\*  if (Auth::check()){ //登入判斷限制  return Auth::user(); //使用者資料  //Auth::viaRemember() //使用者是否勾選記住我  //Auth::loginUsingId(1); //設定使用者為登入  //Auth::logout(); //設定使用者為登出  }  if (Auth::attempt(['email' => $email, 'password' => $password])) { //手動進行認證  //認證通過  }  \*/  }  } |
| 4.基本模板功能完成，始可進行測試：  首頁進行註冊/登入測試，Email若尚未設定完成則無法測試忘記密碼功能。  基本控制器為HomeController  視圖全部都在auth，emails資料夾內放的是忘記密碼信件內容。 |

|  |
| --- |
| 5.客製化註冊資訊  1.模組資訊：config/auth.php  'defaults' => [  'guard' => 'web',  'passwords' => 'users',  ],  //↑預設資訊，守門員為web，重置密碼為提供者users  //守門員：控制器或路由的'auth'認證，可加入:更換守門員，默認為auth:web  //重置密碼：會優先重置預設的提供者  /\*更改設定的方式：在AuthController中，加入  protected $guard = 'admin';  protected $passwords = 'account';\*/  'guards' => [  'web' => [  'driver' => 'session',  'provider' => 'users',  ],  ],  //↑守門員設置，driver可以為'session'(會話)或是'token'(令牌)  'providers' => [  'users' => [  'driver' => 'eloquent',  'model' => App\User::class,  ],  /\*'users' => [ //使用database引擎  'driver' => 'database',  'table' => 'users',  ],\*/  ],  //↑守門員的提供者，設置引擎與資料庫模組  'passwords' => [  'users' => [ //預設資訊內抓取的重設設定名  'provider' => 'users', //提供者會對應到資料庫  'email' => 'auth.emails.password', //帳號cols => 重置信件視圖  'table' => 'password\_resets', //資料庫內的重置暫存表  'expire' => 60, //重置動作存活時間(分)  ],  ],  //↑密碼重置設置資訊  2. migrations模組匯入：依據模組資訊修改migrations資料夾檔案，並使用php artisan migrate  3.模組配置：默認檔案為app/User.php (依據一般模組檔修改) |
| 4.控制器配置：app/Http/Controllers/Auth/AuthController.php  資訊配置：  protected $redirectTo = '/'; //重導路徑  protected $guard = 'admin'; //守門員  protected $passwords = 'account'; //重置密碼  驗證配置：  'name' => 'required|max:255',  'email' => 'required|email|max:255|unique:users',  'password' => 'required|confirmed|min:6',  /\* 驗證法 (用I隔開)：  email 信箱格式限制  image 圖片格式限制  **integer 整數字格式限制**  numeric 數字格式限制  ip 格式為IP限制  mimes:format 檔案格式限制，例如mimes:doc,xls  exists:cols\_name 值必須存在於table所有資料之限定cols中，例如exists:users,name  **digits:num 數字，且字元吻合num，例如digits:10**  digits\_between:n1,n2 數字，且字元介於n1n2之間，例如digits\_between:2,10  **between:n1,n2 數字，且數值介於n1n2之間，例如between:0,999**  max:num min:num 數字，且數值受極值num限制  in:str 字串，內容必須包含任意一個str，例如in:abc,bcd  **confirmed 驗證，表單須存在另一個同名\_confirmation欄位**  **regex:/preg/ 字串正規則吻合**  **required 必須填欄位**  **unique:table 值於該table內不重複**  \*/  創建配置：  return User::create([ //修改模組名或是資料表，此區都要修改或新增  'name' => $data['name'],  'email' => $data['email'],  'password' => bcrypt($data['password']),  ]);  5.視圖配置：修改auth 資料夾內視圖之欄位 |

七、composer安裝package套件包

|  |
| --- |
| 1.套件資源取得  官方套件資源，整理於<https://packagist.org/> ，也可以從composer官網找到Packages進入。  或者是從<http://packalyst.com/> 中，取得別人整理好的for Laravel套件。 |
| 2.套件資源安裝  在根目錄之composer.json檔案中，找到require，將要安裝的資源套件名稱加入  例如套件intervention/image，安裝設置如： (冒號後可指定版本號)  "require": {  "php": ">=5.5.9",  "laravel/framework": "5.2.\*",  **"intervention/image": "\*"**  }, |
| 3.透過composer下載套件  修改完成composer.json後，使用composer update語法即可 (更新已安裝包版本也是如此)  CMD ->至laravel目錄->指令輸入：  composer update |
| 4.使用套件包  使用方法要從套件資源網站內尋找，如本image範例中，官方的靜態使用方法範例如下： (如法炮製即可)  use Intervention\Image\ImageManagerStatic as Image;    // open an image file  $img = Image::make('public/mistake.png');  // resize image instance  $img->resize(320, 240);  // insert a watermark  $img->insert('public/mark.png');  // save image in desired format  $img->save('public/mistake2.png'); |

八、migrations數據庫遷移與填充

|  |
| --- |
| 0.artisan使用方法  CMD ->至laravel目錄->指令輸入：  php artisan 其他指令 |
| 1.官方遷移文件範例 (位置database/migrations/Y\_m\_d\_His\_create\_TABLENAME\_table.php)  遷移文件可以直接透過artisan語法在資料庫中創造資料表，創造時會一併記錄日期版本號，可用於資料表版本管理。  遷移文件創造方法：  1.直接複製資料夾範例文件，再修改檔名與內容  2.php artisan make:migration creat\_TABLENAME\_table  遷移文件使用方法：  1.創造(up)資料表php artisan migrate  2.刪除(down)資料表php artisan migrate:rollback |
| 2.遷移文件內容與常用功能  //此為官方範例2014\_10\_12\_100000\_create\_password\_resets\_table，對應資料表為password\_resets  use Illuminate\Database\Schema\Blueprint;  use Illuminate\Database\Migrations\Migration;  class CreatePasswordResetsTable extends Migration //class名為：資料表名將字首與底線後字大寫，頭尾加入Create與Table  {  public function up()  {  Schema::create('password\_resets', function (Blueprint $table) { //創造  $table->string('email')->index();  $table->string('token')->index();  $table->timestamp('created\_at');  /\* $table常用創造規則：  ->string('colsName') 創造varchar(255)  ->string('colsName',num) 創造varchar(num)  ->integer('colsName') 創造int  ->float('colsName') 創造float  ->increments('colsName') 創造AI的primary key，colsName通常為id  ->text('colsName') 創造text  ->date('colsName') 創造date  ->dateTime('colsName') 創造dateTime  ->time('colsName') 創造time  ->boolean('colsName') 創造boolean  ->timestamps() 創造兩個timestamp，colsName為created\_at與updated\_at  ->rememberToken() 創造varchar(100)的remember\_token，記錄令牌  索引與修飾字段 (加在上方創造規則之後)  ->index() 欄位設定為索引  ->unique() 欄位設定為獨特索引  ->primary() 欄位設定為主鍵  ->first() 欄位放至於表首  ->after('colsName') 欄位接至於欄後  ->nullable() 欄位可為null  ->default('val') 欄位默認值  ->unsigned() 欄位為正整數(限int欄)  \*/  });  }  public function down()  {  Schema::drop('password\_resets'); //刪除  }  } |

|  |
| --- |
| 3.文件種子填充  此功能依據模組，將會透過artisan填充資料表。  修改填充文件後，使用指令：**php artisan db:seed**  填充文件(database/seeds/DatabaseSeeder.php)原碼如下：  <?php  use Illuminate\Database\Seeder;  use App\User; //根據模組文件進行填充，模組文件必須先有且設定$fillable欄位  class DatabaseSeeder extends Seeder  {  public function run()  {  //呼叫seeds資料夾內其他填充類別  // $this->call(UsersTableSeeder::class);  //依據use模組填充$fillable欄位  DB::table("users")->delete(); //清空  User::create([  'name'=>'測試填充名稱',  'email'=>'test@test.com',  'password'=>'18A9S5DW3E5R9D9FD9S5GG6FSDGDGERWDS2A2D12A'  ]);  echo "users Seeder Compelete!";  }  } |

零、其他功能使用

|  |
| --- |
| 1.其它功能函數  dd(變數); //return顯示變數的結構  url(get路徑) //return完整http路徑，常用於視圖 |
| 2.重導向函數  use Illuminate\Http\RedirectResponse;  ..class..  redirect()->action('HomeController@index');  redirect('http://www.google.com'); |