Controller

Controller深談:

Controller之Request:

public function request1(Request $request){

//取值

echo $request->name;

echo $request->input('name','未知');

echo $request->all();

}

* 第一行用request取name的值
* 第二行用request取name的值但多了若沒有顯示未知的功能
* 第三行用request取所有的變數

public function request1(Request $request){

//判斷請求

echo $request->method();

echo $request->ismethod('GET');

echo $request->is('student/\*');

echo $request->url();

}

* 第一行查詢request類型
* 第二行查詢request是否為特定類型
* 第三行查詢此方法是否在此路由底下
* 第四行查詢此方法的路由

Controller之Session:

**簡介:**

* http協定是stateless的所以Session提供一種保存用戶數據的方法
* 有3種使用方法
* 1. http request類的session()方法
* 2. session()輔助函數
* 3. Session facade

Route::group(['middleware' => ['web']] , function(){

Route::any('/session1','StudentController@session1')->name('session1');

Route::any('/session2','StudentController@session2')->name('session2');

});

* 使用Session之前要sessionstart在laravel中將sessionstart寫在web的中間件裡所以我們要套用web來開啟session的路由

public function session1(Request $request){

//1. HTTP request session()

$request->session()->put('key1','value1');

echo $request->session()->get('key1');

}

* put()將一個名為key1且數值為value1的數據放入session
* get()將一個名為key1的數據從session取出

public function session1(Request $request){

//2. session()

session()->put('key2','value2');

echo session()->get('key2');

}

* 同上只不過直接用session()

public function session1(Request $request){

//3. session facade

Session::put('key3','value3');

echo Session::get('key3');

}

* 同上只不過直接用façade的session()

Session::put(['key3'=>'value3','key4'=>'value4']);

* 多數據產生

echo Session::get('key4','default');

* 取出key4，若無則預設值為default

Session::push('key5','value5');

Session::push('key5','value55');

* 將數值放進key5，則key5會產生一個陣列裡面有value5跟value55

***NOTE:*session的put所產生的數據為單一變數而push產生的數據為陣列，所以put出 來的變數，無法再用push去新增變成陣列**

Session::pull('key77')

* 取完數此數據後刪除

Session::all()

* 取出所有數據

session()->has('key3');

* 判斷是否有key3.

session()->forget('key3');

* 刪除key3.

session()->flush();

* 清空session

session()->flash('key5','value5');

* 產生一個key5且有訪問一次即刪除的特性

Controller之Response:

**簡介:**

* 常見類型
* 1.字符串
* 2.視圖
* 3.JSON
* 4.重定向

public function response(Request $request){

$data=[

'errCode' => 0,

'errMsg' =>'success',

'data' =>'jason'

];

response()->json($data);

}

* 把$data轉成JSON後回應

public function response(Request $request){

return redirect('session2');

}

* 用rdeirect()把function response重新導向function session2

return redirect()->route('home');

Controller之Middleware:

**簡介:**

* Laravel中間件提供一個方便的機制來過濾進入應用程序的http請求

Controller

public function activity0(){

return 'activity0';

}

public function activity1(){

return 'activity1';

}

Route

Route::any('/activity0','StudentController@activity0')->name('activity0');

Route::group(['middleware' => ['activity']] , function(){

Route::any('activity1','StudentController@activity1')->name('activity1');

});

Middleware

<?php

namespace App\Http\Middleware;

use Closure;

class Activity

{

public function handle($request,Closure $next){

if(time() < strtotime('2019-01-05')){

return redirect('activity0');

}

return $next($request);

}

}

Kernel

protected $routeMiddleware = [

'activity' => \App\Http\Middleware\Activity::class,

];

* 設定2個Controller function 然後新增一個Middleware叫Activity且設定if的判斷成立則導向activity0不然就繼續當前的請求，創好Middleware後必須去Kernel註冊，註冊後即可使用在路由中

***NOTE:*kernel裡的middleware註冊會套用到所有request，routeMiddleware註冊必須自己去路由定義，middlewareGroups註冊把多個middleware定義成單個名稱**