MVC

路由:

基礎路由:

Route::get('login', 'Auth\LoginController@Login')->name('login');

* 遇到login路徑執行Auth資料夾底下的LoginController的Login函數且此路由命名為login

Route::get('login',function(){ return 'hello'; });

* 遇到login路徑執行匿名含數

路由參數:

Route::get('/users/{user}', 'UsersController@show')

* {user}可以為字串且以user的名稱傳入UsersController給show用

Route::get('/users/{user?}', function($user='jason'){

return 'hello'. $user; });

* 會去加上?會去判斷有無{user}若無會默認為jason

Route::get('/users/{user?}', function($user='jason'){

return 'hello'. $user; })->where('user', '[A-Za-z]+');

* 加上where('user', '[A-Za-z]+')會去判斷user是否符合正則表達是的規範

Route::get('/users/{id}/{user?}', function($id , $user='jason'){

return 'hello'. $id. $user;

})-> where(['id'=> '[1-9]+','name'=>'[A-Za-z]+');

路由+Controller:

基礎配置:

Web.php

Route::get('/users/{user}', 'UsersController@info) ->where('user', '[A-Za-z]+');

UserController.php

public function info($user){return $user;}

* Controller可以直接用相同名稱接收Route的{ }變數Route後面一樣可以用正則表達式限制符號

***NOTE:*一個控制器對應的視圖最好創一個資料夾區隔，例如MemberController會有Member的資料夾放MemberController控制的視圖**

路由+Controller+view:

傳值配置:

Web.php

Route::get('/users/{user}', 'UsersController@info)

UserController.php

public function info($user){

return view('User',['name'=>$user]);

}

User.blade.php

{{$name}}

* 路由將變數已user的名稱傳入Contorller，Controller再將變數包成名稱name傳進User.blade.php也可以用compact把user傳入

public function info($user){

$x=$user;

return view('User',compact('x'));

}

public function info(User $user){

$x=$user;

return view('User',compact('x'));

}

* 若是以Model來接收路由的{user}則laravel會自動用{user}的去匹配Model的主鍵，若有則直接傳入匹配的數據，又稱隱性路由模型綁定

Modle+Controller

基本配置:

User.php

class Sender extends Model

{

public static function getUser(){

return 'user...';

}

}

UserController.php

public function info(){

return User::getUser();

}

Database

Laravel查詢構造器

新增:

DB::table('user')->insert(

['name'=>'jason','age'=>25]

);

* 選擇user這張表新增name 跟 age

$id = DB::table('user')->insertGetId(

['name'=>'jason','age'=>25]

);

* 新增完資料後會返回所新增資料的ID放入$id

更新:

DB::table('user')

->where('id',12)

->update(['name'=>'jason','age'=>25]);

* 選擇user這張表且找尋ID為12的數據，更新他的name跟age
* update一定要搜尋指定的數據

刪除:

DB::table('user')

->where('id',12)

->delete();

* 選擇user這張表且找尋ID為12的數據，刪除搜尋的數據
* delete一定要搜尋指定的數據

查詢:

$user = DB::table('student')->first();

* 用first()獲取數值並返回一個陣列儲存表student的第一筆內容給user

$user = DB::table('student')->get();

* 用get()獲取數值並返回一個陣列儲存表student的全部內容給user
* 可加上where限制抓取內容

$user = DB::table('student') ->where('id', '>=',12)->get();

$user = DB::table('student') ->pluck('name');

* 搜尋表student並只返回name這個欄位

$user = DB::table('student')->select('id','name');

* 搜尋表student並只返回id,name這個欄位

$user = DB::table('student')->chunk(2, function($students){

var\_dump($students);

});

* 搜尋表student並一次只返回一2組數據，然後用匿名含數顯示出來

Eloquent ORM

簡介:

* 簡潔的ActiveRecord實現，用來實驗數據庫操作，每個數據表都有一個相對應的Model用於和數據表交互

Model建立:

* 一般的model在laravel裡會自動對應到復數名稱的資料表
* 例如:model叫User他就會自動對應到叫Users的數據表
* 如果不想內定則在model內指定

protected $table = 'student';

* 在Model中輸入這行指定參考到student
* Model默認id為主建如需修改一樣在Model內加入

protected $primaryKey = 'id';

public $timestamps = false;

* 關閉創立時間跟更新時間

protected function getDateFormat(){return time();}

* GetDateFormat是Model本身就有的方法若再寫一次在Model就會複寫原
* 本的方法進而修改到時間格式

protected $fillable = ['name', 'age'];

* 允許批量賦值得字段

使用Model搜尋:

* 上面建立了一個Model我們讓他對應到student的table然後用下列指令
* 即可搜尋

$user=User::all();

* User::為取用user model ，all()為取model的全部數據
* ORM查詢會返回一個object屬性放在object的attributes裡

$user=User::find(1);

* 查詢主鍵id為1的數據

***NOTE:*查詢構造器的指令也能用在ORM**

使用Model新增:

$user = new User();

$user->name = 'jason';

$user->save();

* 創立一個新的user object然後指定給object的name一個值，最後save。

$user = User::create(['name' => 'imooc','age' => 18]);

* 新增的第二種方法，會自動save但是必須在model裡允許批量賦值

使用Model更新:

$user=User::find(1);

$user->name='andy';

$user->save();

* 直接查找，獲取後直接覆蓋在儲存就可以更新了

$user=User::where('id','>',1019)->update([

'age'=>25

]);

* 批量更新表裡面id小於1019的數據的age

使用Model刪除:

$user=User::find(1);

$user->delete();

* 查找到後直接使用delete刪除

User::destroy(1,2);

* 直接刪除主鍵為1跟2的數據

User::where('id','>',1019)->delete();

* 用where搜尋特定條件後刪除所有符合的數據

Blade模板引擎

Blade

簡介:

* 是Laravel提供的一個概念簡單強大個模板引擎，不限制你在view中使用
* php原生代碼

模板繼承:

Layout.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title> @yield('title','Laravel') </title>

</head>

<body>

<div class='header'>

@section('header')

top

@show

</div>

<div class='sidebar'>

@section('sidebar')

side

@show

</div>

</body>

</html>

Home.blade.php

@extends('layout')

@section('title')

titletitle

@stop

@section('header')

@parent

toptop

@endsection

@section('sidebar')

sideside

@endsection

* 用extends引入layout模板，然後用section選定模板的區域並更改成此模板想要的值，@parent為此section的原始設定值，而layout的section跟yield的差別在section可以擴充而yield只能修改有的部分

模板中使用PHP代碼:

<p>{{ time() }}</p>

<p>{{ date('Y-M-D H:i:s',time()) }}</p>

* 使用{{ }} 即可輸入PHP指令第二行date指令前面變數為輸出格式，後面變
* 數為輸入時間，time()會取用當前時間

<p>{{ isset($name) ? $name : 'this'}}</p>

<p>{{$name or 'this'}} </p>

* isset()用來查詢變數$name是否存在存在就輸出$name不存在就輸出’this’
* 第2行為相同功能簡略寫法

<p>@{{$name}}</p>

* 加上@會原樣輸出上述範例會輸出{{$name}}不會去解析PHP語法

{{-- --}}

<!-- -->

* 上面的註釋不會顯示在原碼中只能在原始程式看到，下面的註釋在網頁檢查
* 原碼時還能看到

@include('layout')

@extends('layout')

* Include會直接去找layout.blade.php的檔案，並把檔案內容寫入我們引入的
* 位置上
* extends是用於不同視圖引入時必須有不同設定，extends要搭配@yield跟
* @section一起用