# 安装

**注意composer中使用的php是系统环境中的PHP版本**

1. **安装composer**
2. **配置中文镜像**

composer config -g repo.packagist composer https://packagist.phpcomposer.com

1. **使用composer安装laravel**

composer create-project laravel/laravel [laravel] [5.5.\*]

---->laravel：项目名称

---->5.5.\*: 版本

1. **必须执行**

// 自动加载才生效，它会把，这个自动加载写入  自动加载的配置文件中

composer install --optimize-autoloader

1. 优化：

php artisan key:generate

1. 优化

php artisan config:cache

1. 优化

php artisan route:cache

# 使用

## 1、Http请求

**（1）GET/POST配置路由**

/routes/web.php

Route::get('#1', "#2**@**#3"); //GET

Route::post(“#1”, “#2**@**#3”); //POST

Route::get('/user', 'Home\UserController**@**index');

#1 url中的请求路径，#2 控制器名 ，#3 类中的方法

# 访问controller目录下的Home文件夹下的UserController控制器

# 控制器必须设置命名空间

**（2）新建文件**

/app/Http/Controllers/新建控制器

**规则**

**1：文件名 *与* 类名 *相同***

**2：文件名必须为 \*\*\*Controller.php**

**--------------------------------------------------------**

**说明：**

类继承的是当前文件下的Controller.php文件

<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

class IndexController extends Controller

{

    public function index()

    {

        print\_r("api\_index");

    }

}

**（3）通过url访问**

## API

The page has expired due to inactivity. Please refresh and try again.

## 3、路由参数

（1）从路由返回

Route::get('/', function () {

//可以添加 reder

return view('greeting', ['name' => 'James']);

});

（2）请求到控制器

Route::get('#1', "#2@#3");

Route::post(“#1”, “#2@#3”);

#1 url中的请求路径，#2 控制器名 ，#3 类中的方法

### 方法接收

public function index( $id ){ }

（3）多请求路由

Route::match([‘get’, ‘post’]， “name”, “#2@#3”)

Route::any(“name”, “#2@#3”)

（4）路由参数

Route::get(“user/{info}”, “#2**@**#3”)

（5）正则路由

Route::get(“user/{id}/{name}”, “#2**@**#3”)

->where( [‘id’=>”.\*”, “name”=>”.\*” ] )

1. 匿名路由

Route::get(“user/mm”, [‘as’ => ‘center’, “#2@#3”])

1. 路由群组

Route::group( [ ‘prefix’=> ‘member’ , function(){

Route::get( ‘#1’, “#2@#3”);

Route::get( ‘#1’, “#2@#3”);

}

访问都需要添加member/

## 视图

**路径：**

/resources/views 视图文件路径

**传值：**

*/app/Http/Controller/IndexController-IndexContrller-index()*

return view('index', ['aa'=>'1234']);

//使用/resources/views/member/index

return view(“member/index”, [“a”=>”123”]);

*/resources/views*

<h1>{{ $aa }}</h1>

## 返回json

return response()->json([

'name' => 'Abigail',

'state' => 'CA'

]);

## 模型

1. 填写命名空间
2. use Illuminate\Database\Eloquent\Model;
3. DB::table()->select()->....

## 表单验证

required->不能为空

**（1）简单验证**

# 进行验证-验证参数查看手册

# 简单验证失败会自动返回验证，并重定向到请求页面

public function select(Request $request)

$validatedData = $request->validate([

"id" => "required"

]);

**（2）自定义验证**

use Validator; //引入

// 构建一个验证器

$validator = Validator::make($request->all(), [

'id' => 'required',

]);

// 验证失败-返回

if ($validator->fails()) {

return response()->json([

"info"=> "id不能为空"

]， **404**);

};

## 跨域请求

1. **添加中间件**

php artisan make:middleware [name]

1. **设置响应头**

public function handle($request, Closure $next)

{

// 处理请求前执行中间件

return $next($request);

}

public function handle($request, Closure $next)

{

$response = $next($request);

// 执行请求后处理中间件

// 这里设置跨域，返回响应头等信息

return $response;

}

## 全局变量

php artisan config:cache

## 测试

php artisan serve

## 上传文件

**（1）配置文件**

config/filesystems.php

disks:可配置的磁盘

->local:本地存储

|->driver:存储的位置

|->root:路径（public\_path(“/”)），

public\_path:/public/

storage\_path:/storage/app/

->public:

->s3:亚马逊云

1. **代码**

# uploads目录名称

$path = $request->file('image')->store("uploads");

#

$file = $request->file(“image”);

$file->isValid() // 判读文件是否上传成功

$file->getClientOriginalExtension(); // 文件类型

$file->getRealPath(); // 文件路径

$file->store("uploads");// 保存

## API

routes/api.php

默认目录为\app\HTTP\controller,如果添加了home，Api目录，需要这么写

Route::post("/", "Api\IndexController@reponse\_info");

1. **注册中间件**

如果你希望中间件在应用处理每个 HTTP 请求期间运行。只需要在 app/Http/Kernel.php 中的 $middleware 属性中列出这个中间件

# 数据库

laravel能轻易实现读写分离

DB facade/查询构造器/Eloquent ORM

## 配置

/config/database.php

/.env 中配置

## 2、DB facade执行原生sql语句

配置完成后在controller中使用

use Illuminate\Support\Facades\DB;

DB::select() / DB::insert() / DB::updata() / DB::delete()

## 2、查询构造器、可预防sql注入

DB::table(‘ name ‘) -> insert(

[‘key’ => ‘value’, ‘key’ => ‘value2’ ]

)

**// 获取插入的自增id**

$id = DB::table(‘ name ‘) -> insertGetID(

[‘key’ => ‘value’, ‘key’ => ‘value2’ ]

)

**//更新**

DB::table( ‘name’ )->where(

‘id’,2 //id 为2

) ->update( [‘key’ => ‘value’] )

**// 自增 ->where()加条件**

DB::table(‘name’)->increment(‘age’, 4) //给某个字段自增，默认加1,加4

DB::table(‘name’)->decrement(‘age’, 4, [‘key’=>’value’]) //自减,修改其他字段

**//删除**

DB::table(‘ name’)->where(‘id’, 4) ->delete() //删除id为4的

DB::table(‘ name’)->where(‘id’,’>=’, 4) ->delete() //删除id为大于等于4的

DB::table(‘ name’)->truncate() //清空表

**//获取表的数据**

DB::table(‘ name’ )->get() //获取所有数据

DB::table(“ name’ )->first() //一条，第一条

orderBy(‘id’,desc) //排序

**//添加多个条件**

->whereRaw(“id > ? and name = ?”, [3,’aa’])

->pluck(“ age ”) //取得哪个字段

->lists(‘age’, ‘id’) //指定那个键做下标

->select( ‘id’, ‘name’) 指定查询哪些字段

**//防止一次拿出太多数据**

DB::table(“ name”)->chunk(2, function($a){

//在这操作数据，每次查询2条，一直到查询完

dd($a) //

if ( ..){

return false ;//暂停

}

})

**//聚合**

DB::table(‘name’)->count(); 统计

->max(‘ age’ ) //获取age中最大的 值

->min(‘ age’ ) //获取age中最小值

->avg(‘ age’ ) //平均值

->sum( ‘age’) //和

## 事务

DB::beginTransaction();

try {

DB::commit();

return 'ok';

} catch (\Exception $e) {

DB::rollBack();

return 'error';

}

**Eloquent ORM**

新建文件

namespace ...

use Illuminate\Database\Eloquent\Model

class Student extend Model

{

protected $table = ‘表名’

protected $primaryKey = “”

protected $fillable = [‘ key1 ‘, ‘key2’] //可同时添加这2个字段

public $timestamps = false //自动维护时间戳

/\*\* 修改保存时间 \*/

protected function getDateFormat () { return time() }

/\*\* 关闭自动格式化时间 \*/

protected function asDateTime($val) { return $val }

}

/app/Http/controller/indeController.php

Student::XXXXX;

新增数据

Student::create( [‘name’ => ‘xxxx’ ]) //在新建表中添加

Student::firstOrCreate( [‘name’=>’xx’]) //没有增加，有就返回

$a = Student::firstOrNew( [‘name’=>’xxx’]) //如果没有则创建新的实例，

$a->save(); //新创的保存

更新

$stu = Students::find(1230);

$stu->name = ‘222’ //修改

$stu->save() // 修改，并维护时间戳

删除

$stu = Students::find(123)

$bool = $stu->delete(); //删除

通过主键删除

Student::destroy(1) //删除id为1的

Student::destroy( [ 1,2,3 ] ) //删除id为1,2,3的

# blade模板引擎

模板继承

layourts.blade.php 主页面

@section(‘head’) @section

@yield(‘content’, ‘字模板’)

1/继承主模板

@extends(‘layouts’) //继承主模板，位置为/resources/views

1. 新建字模板

@section(‘head”)

@parent //显示父模板

xxx //字模板内容

@stop

3\@include(‘student.commit’) //引入一个模板

@if ( $name == ‘sean’)

iam

@elseif($name == ‘inmm’)

ddd

@else

dfdsf

@endif

------------------

@unless( $nam == ‘ss’) ------- @unless //相当于if取反

模板中的url

url() /

action(“className@functionName”) /

route( “url” ) 通过别名

# controller

## Request