# 安装

1.安装器

composer global require laravel/installer

2.新建项目

laravel new blog

3.秘钥

php artisan key:generate

.env.example 文件重命名为 .env 文件

# 路由

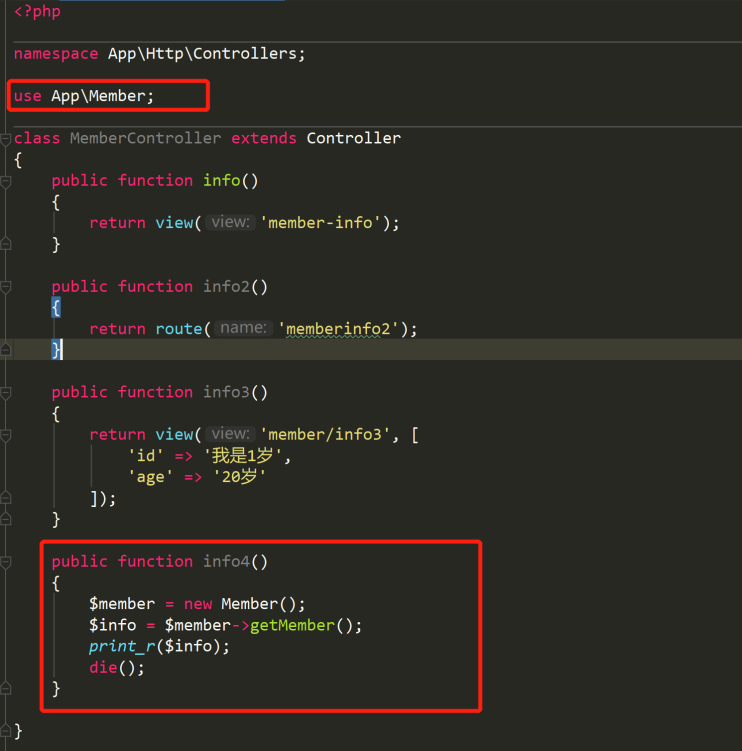
Get post match amy group 别名 群组 视图 自定义参数

<?php  
  
use function foo\func;  
  
  
Route::get('/post', function () {  
 return 'post';  
});  
  
Route::match(['get', 'post'], 'match', function () {  
 return 'match';  
});  
  
  
Route::any('any', function () {  
 return 'any';  
});  
  
Route::get('para/{id}', function (*$id*) {  
 return "para/{*$id*}";  
});  
  
//自定义的默认参数  
Route::get('para/i/{name2?}', function (*$name2* = 'sun') {  
 return "para/i/{*$name2*}";  
})->where('name2', '[A-Za-z]+');  
  
//自定义的参数  
Route::get('para/name/{name2?}', function (*$name2* = 'sun') {  
 return "para/name/{*$name2*}";  
})->where('name2', '[A-Za-z]+');  
  
//自定义两个参数  
Route::get('para/name2/{name2?}/{id?}', function (*$name2* = 'sun', *$id* = '111') {  
 return "para/name/{*$name2*}/{*$id*}";  
})->where(['name2' => '[A-Za-z]+', 'id' => '[0-9]+']);  
  
  
//路由别名:就是把别名加入了route组件中  
Route::get('route/direct', ['as' => 'tion', function () {  
 return array("tion", route('tion'));  
}]);  
  
//路由群组:目的是简化前缀一样的  
Route::group(['prefix' => 'group'], function () {  
 Route::get('route/1', ['as' => '1', function () {  
 return array("1", route('1'));  
 }]);  
 Route::get('route/2', ['as' => '2', function () {  
 return array("2", route('2'));  
 }]);  
});  
  
// 路由中输出视图  
Route::get('/', function () {  
 return view('welcome');  
});  
  
  
// 代码中会使用到  
Route::get('member/info', 'MemberController@info');  
  
//返回route的url  
Route::get('member/info2', [  
 'uses' => 'MemberController@info2',  
 'as' => 'memberinfo2'  
]);

# 视图

public function info3()  
{  
 return view('member/info3', [  
 'id' => '我是1岁',  
 'age' => '20岁'  
 ]);  
}

# 模型



# 原始sql

<?php  
namespace App\Http\Controllers;  
use Illuminate\Support\Facades\DB;  
class StudentController extends Controller  
{  
 public function info5()  
 {  
 $students = DB::select('select \* from student');  
 DB::insert("insert into student (id,name,age,sex) values (*?*,*?*,*?*,*?*)", ['2', '猪八戒', 20, 50]);  
 DB::update("update student set name=*?* where id=*?*", ['唐僧', 2]);  
 DB::delete("delete from student where id=*?*", [2]);  
 }  
}

# 查询构建器

//查询构建器  
public function info6()  
{  
 //增加  
 // $bool = DB::table('student')->insert(['id' => '2', 'name' => '沙僧']);  
  
 // $id = DB::table('student')->insertGetId(  
 // ['id' => '6', 'name' => '沙僧']);  
  
 //更新  
 // DB::table('student')->where('id',6)->update(['age'=>50]);  
  
 //自增  
 // $num = DB::table('student')->increment('age');  
 // $num = DB::table('student')->decrement('age',3);  
 // print\_r($num);  
  
 //删除  
 DB::table('student')->where('id','>=',6)->delete();  
 // DB::table('student')->truncate();  
  
}

public function info7()  
{  
 $count=DB::table('student')->count();  
 *print\_r*($count);  
 $age=DB::table('student')->max('age');  
 *print\_r*(" ");  
 *print\_r*($age);  
 $min=DB::table('student')->min('age');  
 *print\_r*(" ");  
 *print\_r*($min);  
 $avg=DB::table('student')->avg('age');  
 *print\_r*(" ");  
 *print\_r*($avg);  
  
}

# ORM

//orm  
public function info8()  
{  
 $student = Student::all();  
 // dd($student);  
  
 $every = Student::find(5);  
 // dd($every);  
  
 $get = Student::get();  
 // dd($get);  
  
 $info = Student::where('id', '>=', '5')->orderBy('age', 'desc')->first();  
 // dd(count($info));  
  
 Student::chunk(2, function (*$chunk*) {  
 // dd($chunk);  
 });  
  
 $count = Student::count();  
 // dd($count);  
}  
  
public function info9()  
{  
 //批量新增  
 $stu = Student::create([  
 'name' => 'li',  
 'age' => 20  
 ]);  
}  
  
public function info10()  
{  
 //如果没有就新增 有的话不变化  
 $stu=Student::firstOrCreate(  
 ['name'=>'info10']  
 );  
}  
  
public function info11()  
{  
 //如果没有就新增 有的话不变化  
 $stu = Student::firstOrNew(  
 ['name' => 'info11']  
 );  
 $stu->save();  
}

# 引擎

section是用来定义的

yield用来进行调用的

## Url

<a href**="**{{url('student/info13')}}**"**>**url()**</a>  
<br>  
<a href**="**{{action('StudentController@info13')}}**"**>**action()**</a>  
<br>  
<a href**="**{{route('info13')}}**"**>**route()**</a>

## 流程 循环 include 变量

@extends('layouts')  
  
@section('header')  
 {{-- 继承原来的--}}  
 @parent  
 **sidebar**@stop  
@section('content')  
 <p>{{$name}}</p>  
 <p>{{*time*()}}</p>  
 <p>{{isset($info)?'aaa':'bbb'}}</p>  
 <p>**@{{isset($info)?'aaa':'bbb'}}**</p>  
 {{--这种模板注释看不到信息--}}  
 @include('student.stu2',[  
 'message'=>'woshistu2'  
 ])  
  
 @if($name='sean')  
 **i am sean** @elseif($name='aa')  
 **i am aa** @else  
 **i am ss** @endif  
  
 @for($i=1;$i<3;$i++)  
 <p>{{$i}}</p>  
 @endfor  
  
 @foreach($num as $k=>$v)  
 <p>{{$v}}</p>  
 @endforeach  
   
@stop

页面

<body>  
<div class**="header"**>  
 @section('header')  
 **头部** @show  
</div>  
<div class**="main"**>  
 <div class**="sidebar"**>  
 @section('sidebar')  
 **侧边栏** @show  
 </div>  
 <div class**="content"**>  
 @yield('content', '主要内容区域')  
 </div>  
</div>  
<div class**="footer"**>  
 @section('footer')  
 **底部** @show  
</div>  
</body>

# Request

public function info14(Request *$request*)  
{  
 //取值  
 $name = *$request*->input('name');  
  
 $all = *$request*->all();  
 // print\_r($all);  
  
 $method=*$request*->method();  
 // print\_r($method);  
  
 $isMethod=*$request*->isMethod('GET');  
 // print\_r($isMethod);  
  
 $ajax=*$request*->ajax();  
  
 $is=*$request*->is('student/\*');  
 // print\_r($is);  
  
 $url=*$request*->url();  
 *print\_r*($url);  
  
}

# Session

## 路由

Route::group(  
 ['middleware'=>['web']],  
 function(){  
 Route::get('student/info15', [  
 'uses' => 'StudentController@info15',  
 'as' => 'info15'  
 ]);  
 Route::get('student/info16', [  
 'uses' => 'StudentController@info16',  
 'as' => 'info16'  
 ]);  
 }  
);

## 函数

Request组件 Session组件

put设置 push当做数组向里面推

Flash只生效一次

Get进行获取

Forget删除一个

Flush删除所有

All获取所有

Has进行判断

public function info15(Request *$request*)  
{  
 *$request*->session()->put('key', 'value');  
  
 session()->put('key2','value2');  
  
 Session::put('key3','value3');  
  
 Session::push('key4','value4');  
  
 //生效一次  
 Session::flash('key7','value7');  
}  
  
public function info16(Request *$request*)  
{  
 $session = *$request*->session()->get('key');  
 // print\_r($session);  
  
 $value2=session()->get('key2');  
 // print\_r($value2);  
  
 $value3=Session::get('key3');  
 // print\_r($value3);  
  
 $value4=Session::get('key4');  
 // print\_r($value4);  
  
 //取出并且进行删除  
 // $value5=Session::pull('key4');  
 // print\_r($value5);  
  
 if(Session::has('key4')){  
 echo "key4";  
 }  
  
 //删除  
 Session::forget('key4');  
 //删除所有  
 Session::flush();  
 //所有信息  
 Session::all();  
  
 Session::get('key7');  
  
}

# Response

redirect使用完整路由

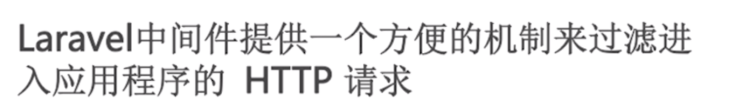
Action使用@符号的

Route使用别名

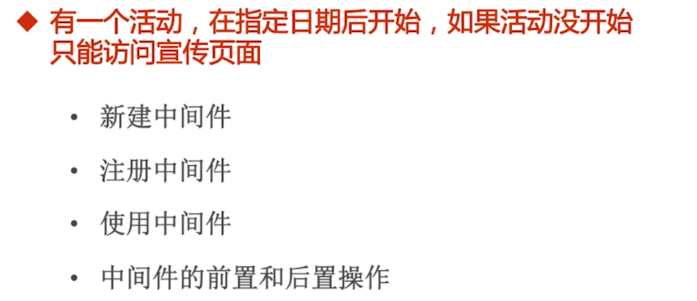
public function info17(Request *$request*)  
{  
 $data=array('a','b','c');  
  
 $data=response()->json($data);  
  
 // return $data;  
  
 // return redirect('student/info7');  
 // return redirect()->action('StudentController@info7');  
 // return redirect()->route('studentinfo7');  
 // return redirect()->back();  
}

# 中间件

## 意义



## 场景



## 路由

//宣传  
Route::get('student/act0', [  
 'uses' => 'StudentController@act0',  
 'as' => 'act0'  
]);  
  
//活动中  
Route::group(  
 ['middleware' => ['act']],  
 function () {  
 Route::get('student/act1', [  
 'uses' => 'StudentController@act1',  
 'as' => 'act1'  
 ]);  
  
 Route::get('student/act2', [  
 'uses' => 'StudentController@act2',  
 'as' => 'act2'  
 ]);  
 }  
);

## Act中间件

<?php  
  
namespace App\Http\Middleware;  
  
use Closure;  
  
class Act  
{  
 //前置操作  
 public function handle(*$request*, Closure *$next*)  
 {  
 if (*time*() < *strtotime*('2026-09-09')) {  
 return redirect('student/act0');  
 }  
  
 return *$next*(*$request*);  
 }  
  
 //后置操作  
 // public function handle($request, Closure $next)  
 // {  
 //  
 // $response = $next($request);  
 // echo $response;  
 //  
 // echo '后置操作';  
 //  
 // }  
}

## 控制器

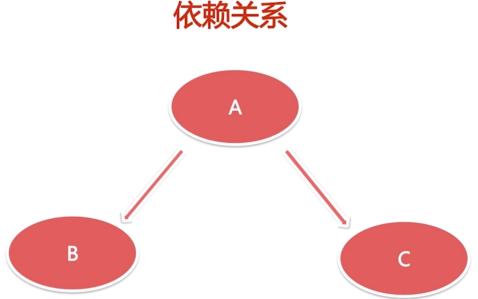
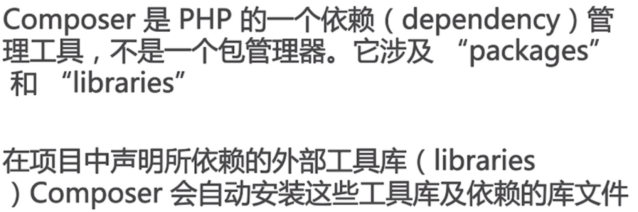
public function act0()  
{  
 return '活动快要开始了';  
}  
  
public function act1()  
{  
 return '进行中1';  
}  
  
public function act2()  
{  
 return '进行中2';  
}

## 设置act

protected $routeMiddleware = [  
 'auth' => \App\Http\Middleware\Authenticate::class,  
 'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,  
 'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,  
 'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,  
 'can' => \Illuminate\Auth\Middleware\Authorize::class,  
 'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,  
 'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,  
 'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,  
 'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,  
 'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,  
 'act' => \App\Http\Middleware\Act::class  
];

# Composer

## 定义



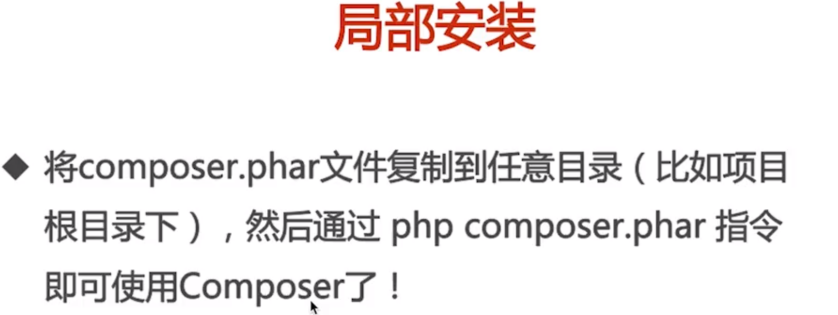
## Composer.phar

Composer.pha是composer的二进制文件

## 下载composer.phar

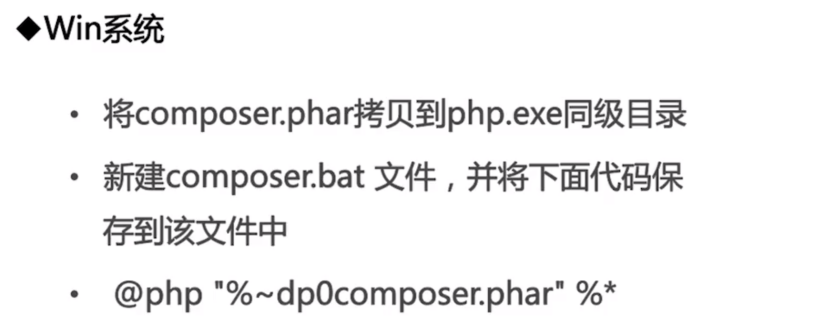


## 局部安装

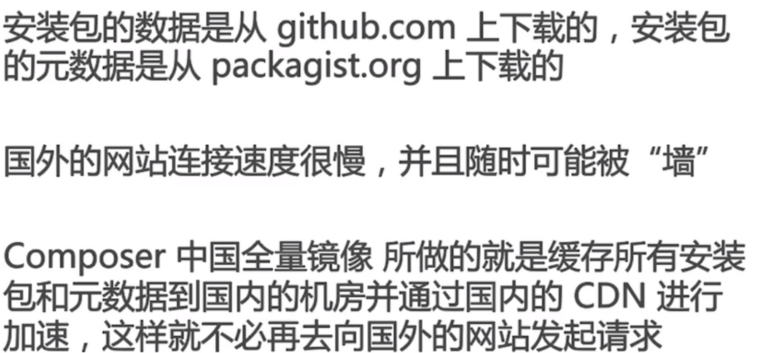


## 全局安装

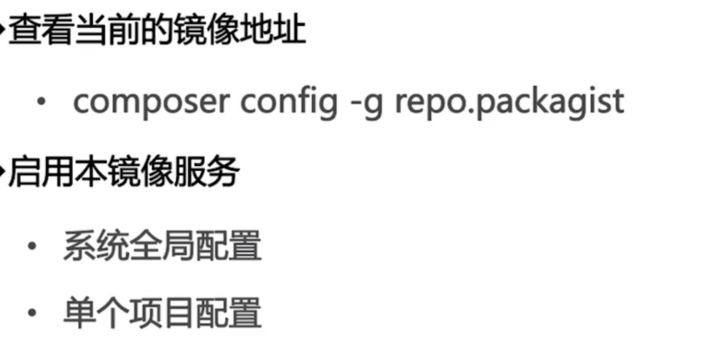




## 中国镜像



## 配置



## Composer init

Composer init是进行初始化的，建立composer.json文件

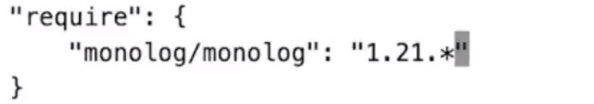
## Composer search

C:\Users\Administrator\Desktop\a>composer search monolog

## Composer show

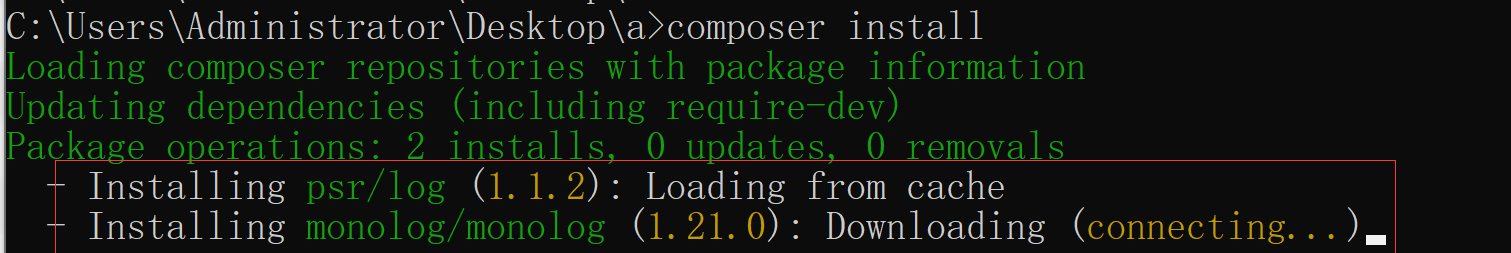
C:\Users\Administrator\Desktop\a>composer show --all monolog/monolog

## 修改json



## Composer install

进行安装

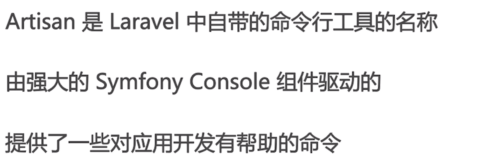


## Composer update

如果在json文件里面进行删除json数据之后，进行执行update后，就会按照json的包配置

# 命令行

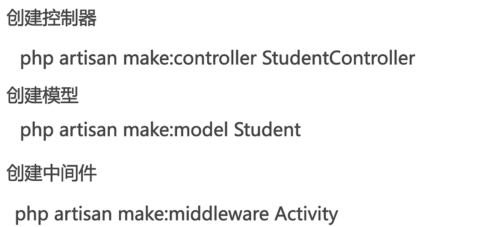
## 定义



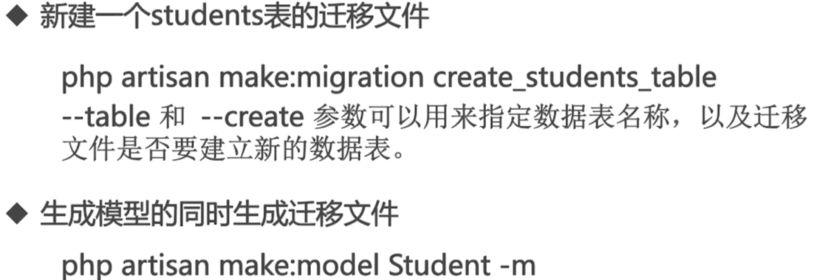
## 命令



## 命令使用



## 命令参数



## 实例

1.建立迁移文件migration

D:\phpstudy\PHPTutorial\WWW\Laravel>php artisan make:migration create\_student\_table --create=students

2.执行迁移文件migrate

D:\phpstudy\PHPTutorial\WWW\Laravel>php artisan migrate

3.进行填充数据文件seeder

D:\phpstudy\PHPTutorial\WWW\Laravel>php artisan make:seeder StudentTabSeeder

4.在seeder文件里面进行添加数据 对应着进行中添加数据

D:\phpstudy\PHPTutorial\WWW\Laravel>php artisan db:seed --class=StudentTabSeeder

## 迁移文件配置文件

'mysql' => [  
 'driver' => 'mysql',  
 'url' => env('DATABASE\_URL'),  
 'host' => env('DB\_HOST', '127.0.0.1'),  
 'port' => env('DB\_PORT', '3306'),  
 'database' => env('DB\_DATABASE', 'test3'),  
 'username' => env('DB\_USERNAME', 'root'),  
 'password' => env('DB\_PASSWORD', 'root'),  
 'unix\_socket' => env('DB\_SOCKET', ''),  
 'charset' => 'utf8',  
 'collation' => 'utf8\_unicode\_ci',  
 'prefix' => '',  
 'prefix\_indexes' => true,  
 'strict' => true,  
 'engine' => null,  
 'options' => *extension\_loaded*('pdo\_mysql') ? *array\_filter*([  
 PDO::*MYSQL\_ATTR\_SSL\_CA* => env('MYSQL\_ATTR\_SSL\_CA'),  
 ]) : [],  
],

# 文件上传

<form id**="dddd"** name**="dddd"**enctype**="multipart/form-data"** method**="POST"** action**="**{{url('upload')}}**"**>  
 <input id**="id"** type**="file"** name**="button"** value**="id"**/>  
 {{csrf\_field()}}  
 <input type**="submit"** name**="button"** value**="button"**/>  
</form>

$file=*$request*->file();

# 邮件发送

## 配置

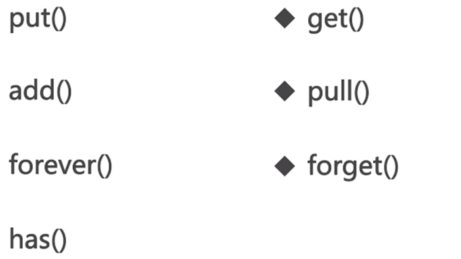
MAIL\_DRIVER=smtp  
MAIL\_HOST=smtp.163.com  
MAIL\_PORT=465  
MAIL\_USERNAME=jimsun7117@163.com  
MAIL\_PASSWORD=jim7117  
MAIL\_ENCRYPTION=ssl  
MAIL\_FROM\_ADDRESS=null  
MAIL\_FROM\_NAME="${APP\_NAME}"

## 代码

public function email(Request *$request*)  
{  
 Mail::raw('邮件内容',function (*$message*){  
 *$message*->from('jimsun7117@163.com','email function');  
 *$message*->subject("主题");  
 *$message*->to("862890248@qq.com");  
 });  
}  
  
public function email2(Request *$request*)  
{  
 Mail::send("email.email",['name'=>'email2'],function (*$message*){  
 *$message*->from('jimsun7117@163.com','email function');  
 *$message*->subject("主题");  
 *$message*->to("862890248@qq.com");  
 });  
}

# 缓存

## 命令



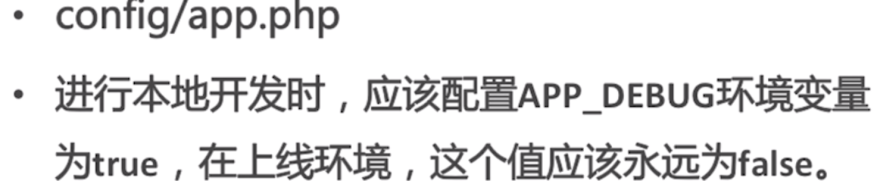
'path' => storage\_path('framework/cache/data'),

## 代码

public function cache1(Request *$request*)  
{  
 Cache::put('key1','val1',10);  
 Cache::add('key2','val2',10);  
 Cache::forever('key3','val3');  
 Cache::has('key1');  
}  
  
public function cache2(Request *$request*)  
{  
 $value=Cache::get('key1');  
  
 //获取之后赋值 然后进行删除  
 $value2=Cache::pull('key1');  
  
 // 删除forget  
 $value3=Cache::forget('key1');  
}

# Debug

## 配置

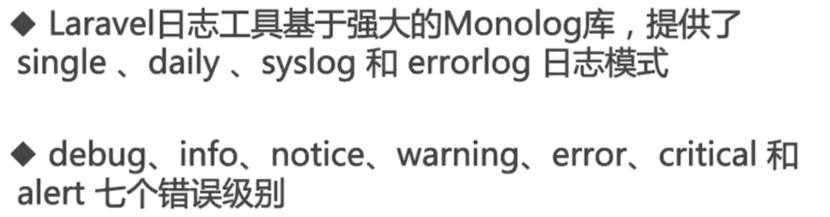


## 函数

Abort()函数调用error文件夹里面的页面

# 日志

## 基础



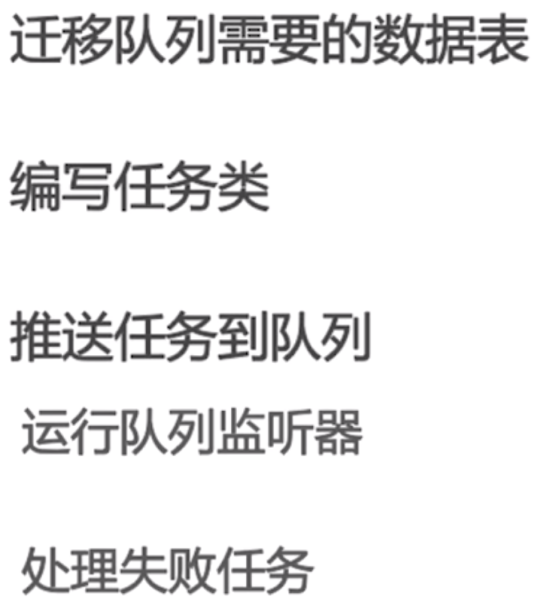
## 代码

APP\_NAME=Laravel  
APP\_ENV=local  
APP\_LOG=single  
APP\_KEY=base64:6AYKw8LAu3+fzFGh0tyVKCsxPVCP95/j6SL5XFA6GBk=  
APP\_DEBUG=true  
APP\_URL=http://la.la.la

Log::info("info log");

# 队列

## 步骤



## 第一步：配置env

BROADCAST\_DRIVER=log  
CACHE\_DRIVER=file  
QUEUE\_CONNECTION=database  
SESSION\_DRIVER=file  
SESSION\_LIFETIME=120  
QUEUE\_DRIVER=database

## 第二部：配置queue.php

'default' => env('QUEUE\_CONNECTION', 'database'),

'database' => [  
 'driver' => 'database',  
 'table' => 'jobs',  
 'queue' => 'default',  
 'retry\_after' => 90,  
],

## 迁移

D:\phpstudy\PHPTutorial\WWW\Laravel>php artisan queue:table

public function up()  
{  
 Schema::create('jobs', function (Blueprint *$table*) {  
 *$table*->bigIncrements('id');  
 *$table*->string('queue')->index();  
 *$table*->longText('payload');  
 *$table*->unsignedTinyInteger('attempts');  
 *$table*->unsignedInteger('reserved\_at')->nullable();  
 *$table*->unsignedInteger('available\_at');  
 *$table*->unsignedInteger('created\_at');  
 *$table*->charset = 'utf8';  
 *$table*->collation = 'utf8\_unicode\_ci';  
 *$table*->engine = 'InnoDB';  
 });  
}

## Job

D:\phpstudy\PHPTutorial\WWW\Laravel>php artisan make:job SendEmail

## 监控

D:\phpstudy\PHPTutorial\WWW\Laravel>php artisan queue:listen