1. 将默认的web路由改成api路由

.\app\Providers\RouteServiceProvider.php 中做如下修改即可：

**protected function** mapWebRoutes()  
{  
 Route::*prefix*(**'web'**)  
 ->**middleware**(**'web'**)  
 ->**namespace**($this->**namespace**)  
 ->group(base\_path(**'routes/web.php'**));  
}

**protected function** mapApiRoutes()  
{  
 Route::**middleware**(**'api'**)  
 ->**namespace**($this->**namespace**)  
 ->group(base\_path(**'routes/api.php'**));  
}

1. 常用命令：

创建仓库+Model+migrate数据迁移：php artisan make:repository Product

创建整套(谨慎使用，生成的默认控制器好像不能用)：php artisan make:entity ProductDp

创建控制器验证：php artisan make:request AdminLoginRequest

创建控制器+资源：php artisan make:controller ArticleController --resource

命令行操作模型：php artisan tinker

创建模型+数据库：php artisan make:model Admin -m

创建工厂：php artisan make:seeder UsersTableSeeder

操作工厂：php artisan db:seed --class=ArticlesTableSeeder

批量操作工厂：php artisan db:seed

给数据表添加字段：php artisan make:migration --table=users adds\_api\_token\_to\_users\_table

执行程序到数据表生效：php artisan migrate

安装laravel5.5系统：php composer create-project --prefer-dist laravel/laravel trade "5.5.\*"

将安装包发布config配置文件到config目录：php artisan vendor:publish 然后选择对应的编号

查看所有路由：php artisan route:list

1. 资源路由

| **动作** | **URI** | **行为** | **路由名称** |
| --- | --- | --- | --- |
| GET | /photos | index | photos.index |
| GET | /photos/create | create | photos.create |
| POST | /photos | store | photos.store |
| GET | /photos/{photo} | show | photos.show |
| GET | /photos/{photo}/edit | edit | photos.edit |
| PUT/PATCH | /photos/{photo} | update | photos.update |
| DELETE | /photos/{photo} | destroy | photos.destroy |

1. 队列任务

//执行指定队列shop2的任务，最大尝试次数3 超期时间60S

php artisan queue:work --queue=shop2 --tries=3 –-timeout=60

//后台守护进程运行

php artisan queue:work --queue=shop2 --daemon

//查看所有失败任务，命令会列出所有任务的 ID、连接、队列以及失败时间

php artisan queue:failed

//重试一个 ID 为 5 的失败任务

php artisan queue:retry 5

//队列重启

php artisan queue:restart

1. 文档生成

第一步：将routers/api.php文件中“**'middleware'** => **'auth.admin'**”这句暂时删除

第二步：执行：php artisan api:generate --routePrefix=admin/v1/\*

第三步：将routers/api.php文件中“**'middleware'** => **'auth.admin'**”这句恢复

参考文档：http://laravelacademy.org/post/4578.html

1. 小技巧

* **//检查数据表是否存在**

**if**(\Schema::*hasTable*(**'fanli\_statementdd'**)){  
 **echo "存在"**;  
}**else**{  
 **echo "不存在"**;  
}**exit**;

*//获取当前明细表名***echo** $this->**model**->getTable();**exit**;

*//连接指定数据库并设置指定表名*$mx = Mingxi::*on*(**'mysql\_read'**)->getModel()->setTable(**'2018-08'**);  
$res = $mx->first();

* 更新加载类：

php composer dump-autoload

* 传参

*$name = $request->query('name', 'Helen');//只能用于查询get的参数*$name = $request->input(**'name'**, **'Sally'**);//所有参数都可以接收

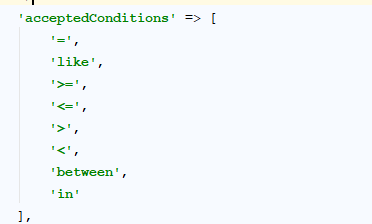
* 联表、别名

$this->**model**->from(**'user as u'**)  
 ->select(**'u.\*'**, **'i.id as i\_id'**)  
 ->leftJoin(**'info as i'**, **'i.user\_id'**, **'='**, **'u.id'**)  
 ->get();

* 其他

1. **repository模式下对between、in等关系的查询支持**

首先：将config/repository.php中**'acceptedConditions'增加对between、in的支持，如下：**



接下来：将https://github.com/clms2/l5-repository/blob/master/src/Prettus/Repository/Criteria/RequestCriteria.php文件复制并替换到\vendor\prettus\l5-repository\src\Prettus\Repository\Criteria\RequestCriteria.php

然后：解析请求参数，解析的是特定形式的查询参数，如: /users?search=user\_name:133;created\_at:2018-01-01,2018-01-17&searchFields=user\_name:like;created\_at:between

即可

参考文档：<https://segmentfault.com/a/1190000012987608>

1. 获取某段代码的sql语句

DB::connection()->enableQueryLog();#开启执行日志

$count = DB::table('test') //执行查询

->whereNull('deleted\_at')

->where('id', '=', 3)

->where('Name', '=', '测试')

->count();

print\_r(DB::getQueryLog()); //获取查询语句、参数和执行时间

1. Repository模式findWhere用法

$this->**repository**->findWhere(

[

**'taocj'**=>0,

**'paicj'**=>0,

**'mallcj'**=>0,

[**'date'**,**'>='**,**'2015-05'**]

],

[**'id'**,**'date'**]

);

1. 控制台命令

1.统计上个月月报

php artisan statement:bymonth

2.统计每日报表

今日数据: php artisan report:daily tenMinute

昨日数据: php artisan report:daily yesterday

历史数据：php artisan report:daily timeofDay --start=2018-10-01 --end=2018-10-08

1. Laravel打印sql语句

laravel如何打印orm封装的sql语句

$query = CdbForumSellthreadSearch::where($params)->orderBy("$orderby", "$ascDesc")->offset($start)->limit($limit);

$list = CdbForumSellthreadSearch::where($params)->orderBy("$orderby", "$ascDesc")->offset($start)->limit($limit)->get()->toArray();

//打印出的sql是预编译的sql语句，带有问号占位符的sql

print\_r($query->toSql());

//下面打印出占位符对应的变量的值，数组里面的顺序和占位符的顺序是一致的

echo '<pre>';

print\_r($query->getBindings());

exit;

1. 测试环境仓库地址

以下Laravel测试环境和git已搭建好：

新平台订单api： git@120.79.133.35:/srv/trade.git

测试域名：http://trade.xiaofei178.net

新平台用户api：git@120.79.133.35:/srv/member.git

测试域名：http://member.xiaofei178.net

新平台其他api：git@120.79.133.35:/srv/other.git

测试域名：http://other.xiaofei178.net

新平台后台前端：git@120.79.133.35:/srv/admin.git

测试域名：http://admin.xiaofei178.net

订单：

const checkedOptions = [

{ key: 0, name: '未认领' },

{ key: 1, name: '待审核' },

{ key: 2, name: '已认领' },

{ key: 3, name: '已返现' },

{ key: -1, name: '退款' },

]

const statusOptions = [

{ key: 1, name: '订单创建' },

{ key: 2, name: '订单付款' },

{ key: 3, name: '订单成功' },

{ key: 4, name: '订单失效' },

{ key: 5, name: '订单结算' },

{ key: 6, name: '订单部分退款' },

]

13. 队列任务执行时被killed的解决方案

[leo](https://learnku.com/users/5320)

研发总监 @ 普瑞生

通常情况下是 OOM 了，php artisan queue:work --memory=1024。如果还是被 Kill 则有可能是超出了虚拟机本身的内存，调出 virtualbox 把虚拟机内存调高

1年前

[@leo](https://learnku.com/users/5320) 我在 job 里面有写 ini\_set('memory\_limit', '1024M') 虚拟机本身的内存我给了4个G，按理说不应该的，请问 php artisan queue:work --memory=1024 和直接在 job 中设置内存上限这两种做法有什么区别吗？

1年前

研发总监 @ 普瑞生

[@lybc](https://learnku.com/users/1232)queue:work 会启动一个 deamon 进程，再有 deamon 进程去启动 worker 进程来处理 job，deamon 进程会监控 worker 进程的内存占用情况，如果超出了 --memory 的设置就会将其 kill 掉，而你的 ini\_set('memory\_limit', '1024M') 并不会影响 deamon 进程的这个行为。

例如：/usr/local/php/bin/php artisan queue:work --queue=tuijianren --memory=3000

查看进程是否启用：ps -fe|grep tuijianren |grep -v gre

14. first（）结果转数组的解决方案

使用 get\_object\_vars（）可以将他抓转为数组

get\_object\_vars — 返回由对象属性组成的关联数组；

在LARAVEL中其实还可以用 toArray(); json\_decode()但前两者在某些情况下不起作用，比如链接数据库查询使用了 first() 这时候就可以使用 get\_object\_vars（）了；

15. updateOrCreate用法

updateorcreate 如同:

$list= User::where(array('name'=>'php'))->get()->toArray();

if(empty($list)){

User::create(array('name'=>'php','email'=>'php@qq.com'));

}esle{

$list->update(['email'=>'php@qq.com']);

}

16. Kafka队列使用

先安装日志扩展

php composer require psr/log 1.0.2

再安装kafka扩展

php composer require nmred/kafka-php

再配置文件