# Tp-myshop 项目一

## TP介绍

route.php 路由文件 在route文件下

路由文件可分admin 与后台 该框架自动调用

config 是配置文件

app.php 是应用设置

database.php是数据库设置

app

## TP shop商品项目步骤

### 1.解压TP5文件到项目目录中

### 2.配置win下的虚拟主机 wamp的环境配置

### 3.修改开发环境 修改3 true

app\_debug = true app\_trace=true url\_route\_must = true

4.调试路由开是否成功

在路由文件中 测试

### 5.cmd命令下创建后台模块

进入C跟目录 cd.. cd..

cd wamp64\www\myshop

php think build --module admin

6.将后台首页html文件复制到view文件目录中

在路由文件route.php中设置路由 ：Route::view('/','admin@index');

将后台的所有的css、js、images 等修饰HTML的文件目录复制到public/static目录下新建的admin文件夹中

修改后天首页的引入地址

测试

### 7.修改mysql文件 的数据库等等

### 8.键表、键模型、键控制器

建表:

create table shop\_users(

uid int unsigned primary key auto\_increment comment '主键',

uname char(20) not null unique comment '账号',

upwd char(32) not null comment '密码',

sex enum('w','m','x') default 'w' comment '性别',

uface varchar(200) default '' comment '头像',

tel varchar(15) default '' comment '联系电话',

addr varchar(200) default '' comment '收货地址',

auth int unsigned default 3 comment '权限 1超管 2普管 3普会',

regtime int unsigned default 0 comment '注册时间'

)engine=Innodb default charset=utf8;

键model

cmd命令 ：php think make:model User

键controller

cmd命令： php think make:controller admin/UserController

### 9.修改后台首页菜单

### 10. 配置菜单的路由 绑定控制器

URI ， 模块/控制器/方法

Route::rule('/admin/user/create','admin/UserController/create');

将URI为/admin/user/create 绑定为admin模块下UserController控制器的create方法

### 11.在default.html 中定义 可替换的模板

{block name="main"} {/block}

### 12.在控制器(路由绑定的方法)中引入添加用户页面

视图模板 user目录下 create.html

view('user/create');

### 13.在create.html 继承default.html 重写 可替换的模板 main

修改模板中之数据库一一对应

form action ="/admin/user/save" 配操作添加用户的路由(只要有跳转就要配置路由)

### 14.添加用户的数据发送到控制器save方法中

save 用形参->post来接收POST发送过来的数据

### 15.整理添加用户的数据

添加到数据库时为了不报错，可用try{}catch{}()语法

### 16.调用model User类来操作数据

User::create(整理的数据,true); (create 就是insert方法)

这里用User 就要将User的命名空间选择(use app\common\model\User;)出来

true 是开启数据库对字段的自动过滤

### 17.浏览用户 配置路由

在控制器index 方法中获取用户的所有数据

引入浏览用户的模板并将获取的数据传给模板用来遍历

view('user/index',['res'=>$array]);

['模板要引入的变量名' => 方法中获取的数据]

### 18.模板中变量

{foreach $res as $k=>$v}

{/foreach}

写原生函数也可用TP5规定的来写如上

{:date('Y-m-d H:i:s',$v->regtime)}

TP5中{:原生函数()} 冒号 (具体可参考手册)

### 19.删除用户RUI

href="/admin/user/delete/{$v->uid}"

URI 路由 /{$v->uid} 传入参数

### 20.绑定路由

Route::rule('/admin/user/delete/:id','admin/UserController/delete');

控制器方法delete 有形参可直接接受路由中绑定的id

### 21.model中destroy() 是删除方法(delete)

### 22.TP5中$this->error("", url);

错误方法 参数1：错误时显示的信息 参数2：要跳转的URL

$this->success() 一样

### 23.修改用户

修改路由传入要修改的id 设置路由

### 24.修改方法

用路由传过来的ID查询出来数据

引入修改模板并且传入查询出来的数据

### 25.修改 修改模板

value 显示 和路由的修改

### 26.设置操作修改的路由绑定修改方法

路由：Route::rule('/admin/user/:id/update','admin/UserController/update');

方法：User::update($request->post,['uid'=>$id],true);

//参考model类中update ctrl+B

### 27.搜索

浏览用户表单 搜索 method=get

修改搜索表单 发送到index方法

### 28.实现搜索

定义数据用于拼接搜索条件

判断搜索条件是否为空不为空则拼接

$condition[] = ['sex','=',$\_GET['sex'] ]; //sex ='s' TP5中采用的是预处理绑定数据 因此GET的单引号可以省略

$condition[] = ['uname','like',"%{$\_GET['uname']}%"]; // 拼接姓名需要模糊查询

### 29.分页

只用将原来的select改成paginate(3)

模板{$引入的数组->render(); |raw} / {:$引入的数组->render(); } （这是两种模板中解析PHP函数的方法）

### 30.因有查询需要将查询的条件也传给分页

appends($\_GET); //即可

### 31.键分类表、模型、控制器 (参考以上)

补充：

1.路由的另一种定义：

Route::group(['name'=>'admin/cate','prefix'=>'admin/CateController/'],function() {

Route::rule('[:id]/create','create',get) @param 3 可写可不写

});

2.分类排序是用orderRaw('concat(path,pid)') 该方法

3. 添加子分类时模板和路由都绑定了id 控制器也需要id 直接给控制器放一个形参$id会报错

虽然路由绑定的[:id]可有可无，但在没有用ID是会报错 此时需要给形参ID一个默认值

$id=''/$id=0

4.删除用时需要判断一下该类别是否有子类 用find来查找(查找到一条就返回),不必用select

### 32.键商品表、模型、控制器 (参考以上)

补充：

1.上传图片时需要在app.php 中配置保存路径

上传文件保存路径

'save\_path'=>Env::get('root\_path').'public/static/uploads/',

显示上传图片路径

'disp\_path'=>'/static/uploads/',

Tp5中config 获取和设置配置参数

2.整理上传图片

2.1先获取上传的图片

$file = $requset->file('模板上传表单的name值');

2.2将其移动到自己定义的路径中

$info = $file->move(config('save\_path'));

2.3获取到移动到文件夹图片的名称 //需要长传到数据库

$goods['gpic'] = $info->getSaveName();

3.制作缩略图

因为获取来图片的名称 需要将\ 转义成/sm\_

$sl = str\_replace('\\','/sm\_',$goods['gpic']);

think\Image::open($file)->thumb(100,100)->save(config('save\_path').$sl);

4.model 中显示 获取器的重写 gpic是库中图片字段

4.1模板中 $v->smgpic 数据库中没有smgpic

public function getSmgpicAttr(){

return str\_replace('\\','/sm\_',$this->gpic);

}

4.模板中$v->gpic 数据库中也有gpic

public function getGpicAttr($value){

//$value值就是数据库的值

}

5.上下架 redirect(); 重定向 局部刷新

### 33.登陆 验证码 验证

1.登陆引入登陆模板

2.添加验证码

表与图片对齐 JS点击刷新 /code定义路由

<img style="vertical-align:top;" onclick="this.src='/code?a='+Math.random()" src="/code"/>

Route::rule('/code','admin/LoginController/verify');//验证码路由路由

Captcha() 类与缩略图类 一样都需要安装 果然都对验证码显示不出来 则可用ob\_clean()；清空缓存

类中构造函数来配置验证码。类中entry() 获取验证码 类中check(模板输入的验证码) 是验证 验证码是否正确

3.验证登录 $user = User::where(['uname'=>$login['uname']])->where(['upwd'=>md5($login['upwd'])])->find();

验证成功设置session 用于验证

### 34.行为，后置行为

1.创建application\admin\behavior

2.创建CheckLogin.php 文件

**<?php  
namespace** app\admin\behavior;  
  
**class** CheckLogin  
{  
 **use** \traits\controller\Jump; //为了可以使用error()方法  
 **public function** run()  
 {  
 **if**(**empty**(session('adminLogin')))  
 {  
 **return** $this->error("未登陆",'/admin/login');  
 }  
}  
}  
**?>**

3.所有路由组后面添加验证

->after('\app\admin\behavior\CheckLogin');

配置路由运行CheckLogin 类里面的run方法 运行完后互通路由

### 35.前面模板的部署

1.整体和后台部署差不多 参考以上

2.前台模板的另一种include的引入模板

{include file="common/head"} common文件夹/head公共头部

3.公共头部可选

<?php if(empty($show)): ?>

<link href="/static/home/css/hidemenu.css" rel="stylesheet" type="text/css" />

<?php endif;?>

模板那个需要该css样式可定义show为true即可

4.前台首页类别的遍历 在cate数据库中添加用递归获取类别的方法

//前台模板 获取类别  
**static public function** getSubCate($cate=[],$pid=0){  
 **if**(**empty**($cate)){  
 //为空获取数据  
 $cate = **self**::*select*();  
 }  
 $arr = [];  
 **foreach** ($cate **as** $k=>$v){  
 **if**($v->pid == $pid){  
 //递归循环遍历出当前类下的所有子类  
 $v->sub = **self**::*getSubCate*($cate,$v->cid);  
 $arr[] = $v;  
 }  
 }  
 **return** $arr;  
}

5.前台点击列表可搜索出该类下所有的子类商品可利用path中的逗号

$sousu = [];  
**if**(!**empty**($id)){  
 $arr\_cid = Cate::*where*('path','like',"%,$id,%")->column('cid');  
 $arr\_cid[] = $id;  
 $sousu[] =['cate\_cid','in',$arr\_cid];  
}

1. 点击到加入购物车时要生成session

$goods = Goods::*get*($id);  
$goods->cnt = $request->post('cnt');//把购物数据添加到商品对象中  
//将所有的数据添加都session中 二维数组的session  
session("cart.$id",$goods); //$\_SESSION['cart'][$id] = $goods  
//提示加入购物车成功  
**return** view('cart/add',['goods'=>$goods]);

### 使用中间件验证是否登陆

1.定义中间件可以通过命令行指令快速生成中间件

php think make:middleware CheckLogin

2.在application目录下生成http/middleware/CheckLogin.php

3.在hangle方法中书写中间想要执行的内容

4.最后return $next($request); 执行完中间件代码后将归还与原来预先定义好的处理方法

**public function** handle($request, \Closure $next)  
{  
 **if**(**empty**(session('homeLogin'))){  
 //为空就让返回去登陆  
 session('back',$\_SERVER['REQUEST\_URI']);  
 **return** $this->error("请先去登陆...",'/login');  
 }  
 //自己代码执行完后，归还给原来预先定义好的处理方法  
 **return** $next($request);  
}

5.更改路由

->middleware('CheckLogin');

### 鉴定完后生成订单表(主)和订单详情表(附)

1.添加订单表(主表)后生成订单详情(附表) 是一对多的关系先有一个订单下可以有很多的商品，因此需要将两表绑定

2.在Orders.php model中用hsaMany(‘附表表名’,’外键’,’主键’);对附表进行关联

//主表关联附表 一对多 主表定义 hasMany 附表belongsTo关联  
**public function** details(){  
 **return** $this->hasMany('Details','orders\_id','oid');  
}

3.在Details.php model中绑定主表 belongsTo(‘主表表名’,’外键’,’主键’);

//建立订单与订单详情两个表的关联 一对多  
**public function** orders(){  
 //附表关联  
 **return** $this->belongsTo('Orders','orders\_id','oid');  
}

### 使用事件ini来对商品表的销量和库存进行改变

1.当订单详情表 新增数据时就需要在商品表中更改库存和销量因此使用事件来更改

2.因为是详情表新增后，所以在Details.php 中添加一个事件方法

//事件 当订单详情表生成时表示交易成功需要将商品表中的该商品库存减去买的 同时销售要加上  
**static public function** init()  
{  
 **self**::*event*('after\_insert',**function**($detail){  
 Goods::*get*($detail->gid)->setDec('stock',$detail->cnt);  
 Goods::*get*($detail->gid)->setInc('salecnt',$detail->cnt);  
 });  
}

3.dedails 表中存有商品表的id gid和购买的个数cnt

4.setDec 减少 setInc新增

5.after\_insert在数据表添加之后（手册）

### 39.使用订单详情(多)来获取商品表的数据(一)

1.前台遍历个用户的订单后，每个订单下还要将所买的商品名和图片遍历出来 因此需要在详情表中绑定商品表

2.因为是从多来找一，所以Goods.php model中不需要使用hasMany来关联之间在Details.php model中belongsTo来绑定

//建立订单详情与商品两个表的关联 多对一  
**public function** goods(){  
 //多 关联  
 **return** $this->belongsTo('Goods','gid','gid');  
}

