# 1 电商项目部署

## 1.1项目部署步骤

### 1.配置虚拟主机

### 2.使用TP生成项目应用代码

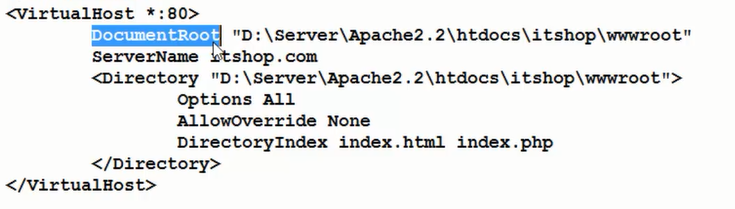
### 3.手动创建后台模块

## 1.2.虚拟主机的配置

1创建项目的根目录 jxshop

2 在jxshop下创建wwwroot目录 虚拟主机的域名解析到wwwroot下会更安全

3 配置虚拟主机

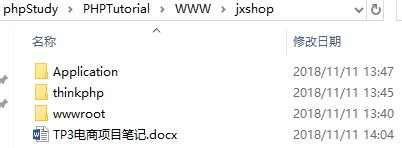


4 改hosts文件 ipconfig /flushdns

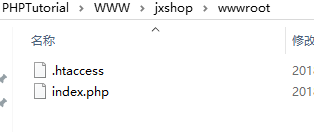
5 重启apache

## 1.3使用TP生成应用代码

1 拷贝TP框架源码到wwwroot同级目录



2 拷贝入口文件及重写规则文件到wwwroot



3 修改入口文件

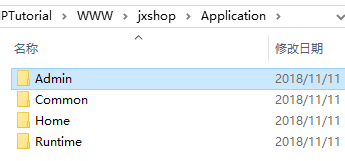


4 访问生成代码



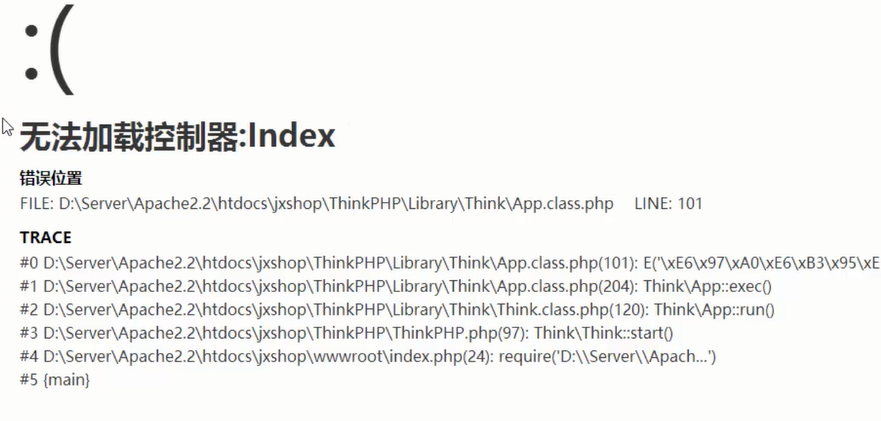
## 1.4 手动生成后台模块

1 复制home模块改名为admin模块



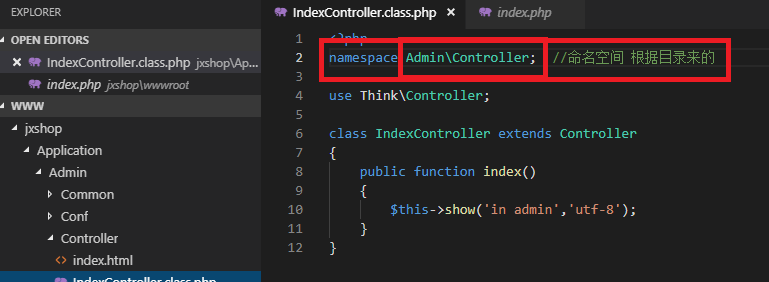
2 修改 admin/controller/IndexController.class.php

访问 <http://jxshop.com/index.php?m=admin&c=index&a=index>

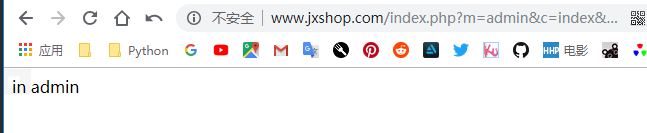


因为没有更改命名空间

所以,修改命名空间为Admin



3 访问测试



# 2.电商项目URL地址优化

## 2.1URL地址优化的具体方法

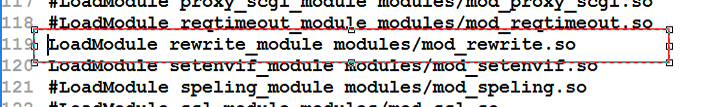
### 1.隐藏项目的入口文件

### 2.隐藏默认的home模块

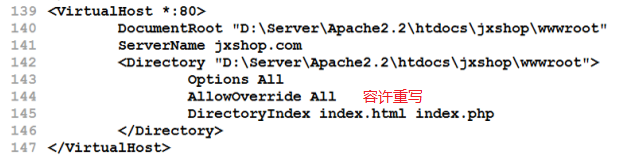
## 2.2隐藏项目的入口文件

1.开启apache的重写模块

Httpd.conf



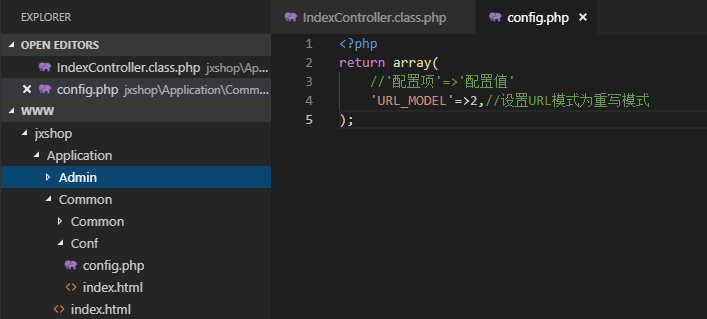
2.设置虚拟主机容许重写



3.拷贝TP提供的默认重写规则文件



4.修改项目的配置设置为重写模式



5.测试

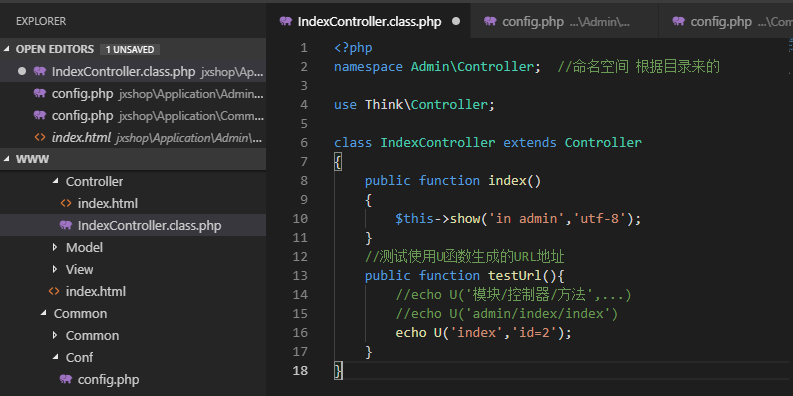


这里隐藏了index.php也依然可以正常访问

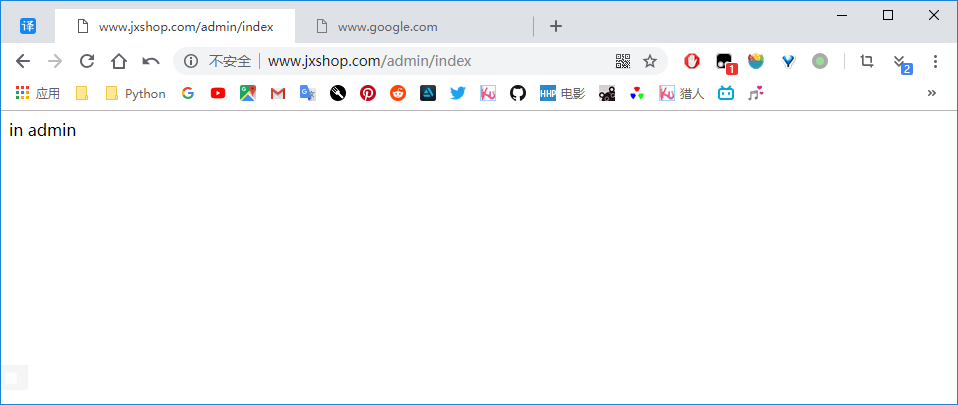
2.3 设置Home为默认的模块

1.修改配置项设置默认模块

2.增加设置容许访问的模块

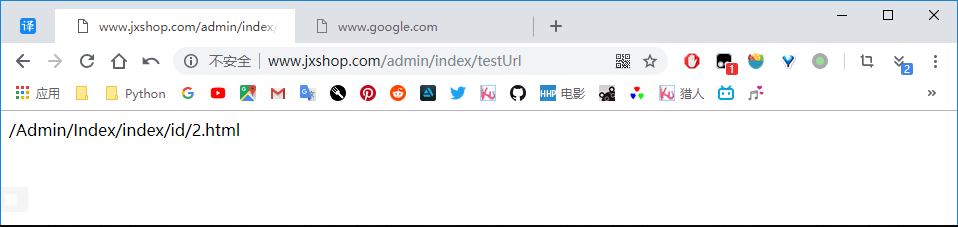


3.访问测试



4.测试使用U函数生成URL地址



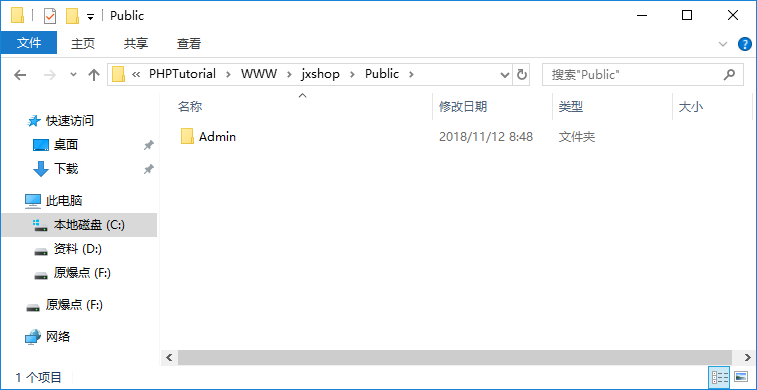


# 3 后台首页的展示

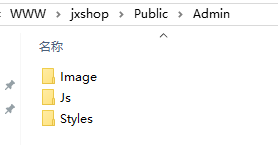
## 3.1后台资源文件拷贝

1.在域名对应的目录下创建Public目录 (用来存储各种资源)

2.在Public目录下创建Admin目录

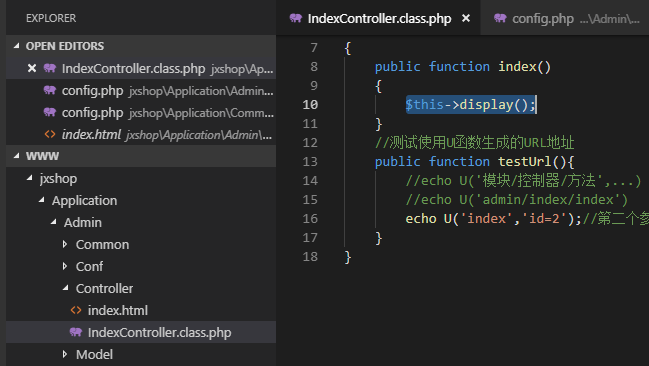


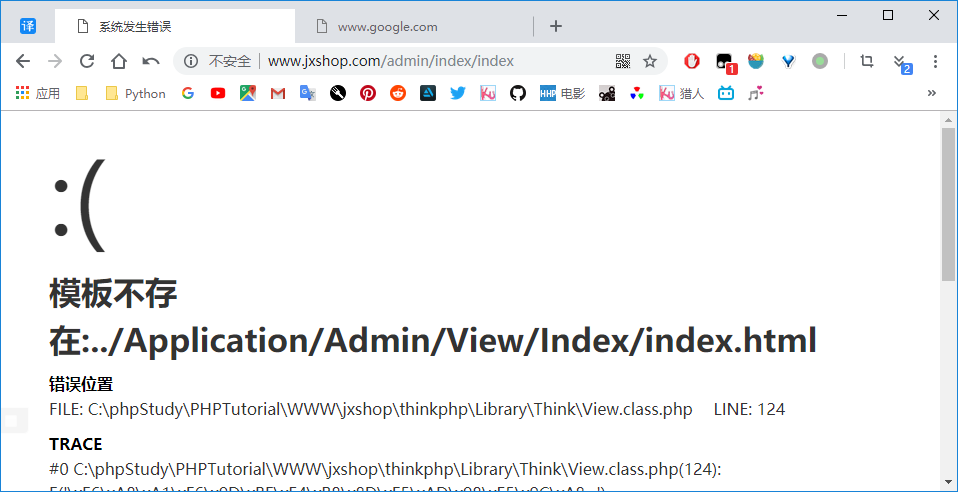
3.将资源文件进行拷贝



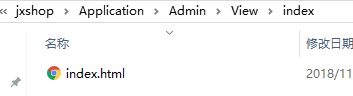
## 3.2展示后台页面

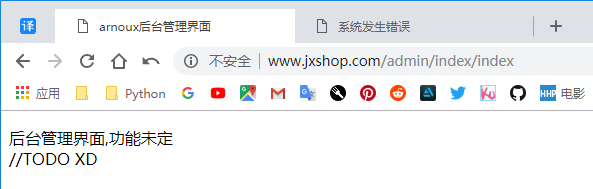
1.修改代码加载模板



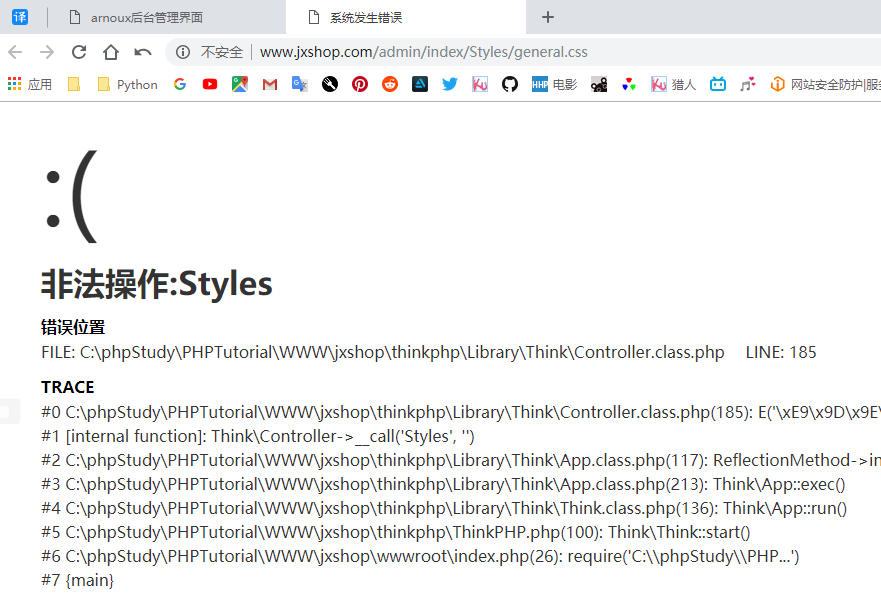


2.在视图目录下创建目录和页面于控制器同名





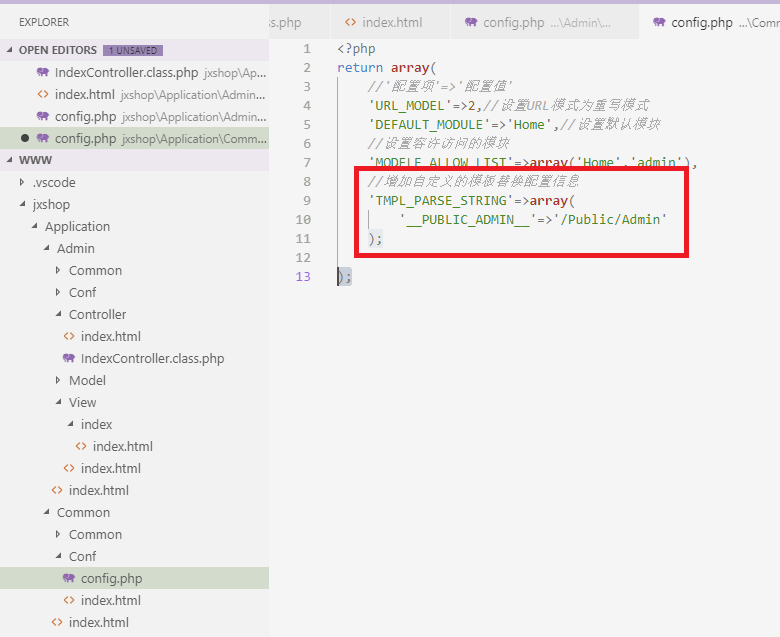
可访问,但没有加载css样式



## 3.3解决现实错乱问题

3.修改配置项增加自定义模板替换配置信息

在jxshop/application/Common/conf/config.php 中添加配置信息



在模板文件中 替换

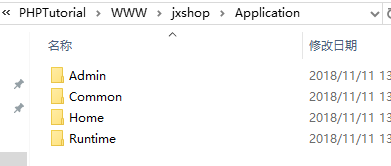


访问后成功加载css



小结:其实,在TP框架中 WWW\jxshop\wwwroot目录下只是存放了 资源文件文件夹, .htaccess index.php 这三个

在直接访问网址时,由前面写的默认跳转的路由功能转移到了WWW\jxshop\Application下的某个模块



例如这里有四个模块,其中admin是通过复制home文件夹改名得到的

如果访问 域名/admin/index/index

将直接调用 WWW\jxshop\wwwroot\Admin\Controller下的名为IndexController.class.php的模块(命名规则为网址的第二个参数+Controller.class)



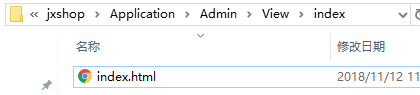
看网址第三个参数的函数(这里的模块就是MVC中的C,控制类.控制类会去调用V,也就是视图)

会调用index函数

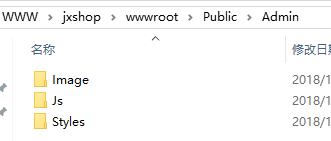
$this->display()

会返回View下模块名下函数名文件

WWW\jxshop\wwwroot\Admin\View\indexindex.html

这个文件

如果这个文件中使用了css img js 都会从wwwroot\Public\Admin

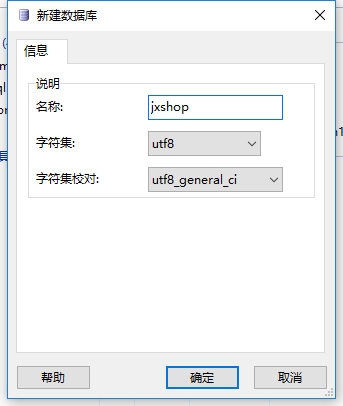


访问控制的流程大概就是这样

# 4 商品分类管理

## 4.1数据库配置

### 4.1.1 创建项目数据库配置



### 4.1.2 创建分类的数据表

1.创建数据表常用原则

有主键 尽量不允许为空且有默认值 尽量小的类型

2.数据表分类

大概代码

CREATE TABLE `itable` (

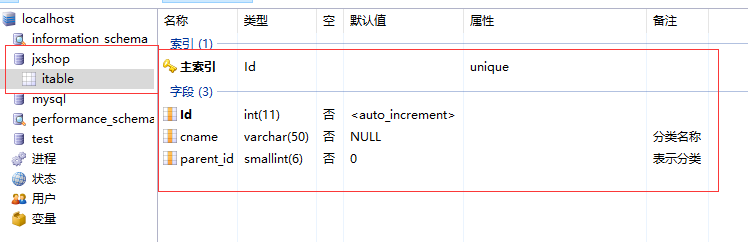
`Id` int(11) NOT NULL AUTO\_INCREMENT,

`cname` varchar(50) NOT NULL DEFAULT 'NULL' COMMENT '分类名称',

`parent\_id` smallint(6) NOT NULL DEFAULT '0' COMMENT '表示分类',

PRIMARY KEY (`Id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;



### 4.1.3 配置TP与数据库交互信息

在ThinkPhp文件夹下配置文件拷贝并配置到自己的应用下

*/\* 数据库设置 \*/*

'DB\_TYPE' => 'mysql', *// 数据库类型*

'DB\_HOST' => '127.0.0.1', *// 服务器地址*

'DB\_NAME' => 'jxshop', *// 数据库名*

'DB\_USER' => 'root', *// 用户名*

'DB\_PWD' => 'root', *// 密码*

'DB\_PORT' => '3306', *// 端口*

'DB\_PREFIX' => 'jx\_', *// 数据库表前缀*

# 4.2分类控制器及模型创建

## 4.2.1 修改控制器的继承关系

原来的:A继承C

改成链式继承:A继承B继承C

现在只要更改B就可以达成某种功能而不用变动其他的了

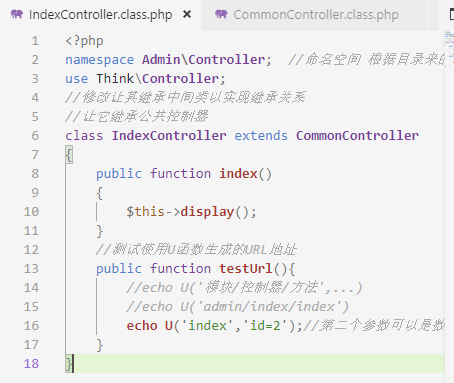
1. 创建公共控制器

在IndexController.class.php同目录下创建CommonController.class.php

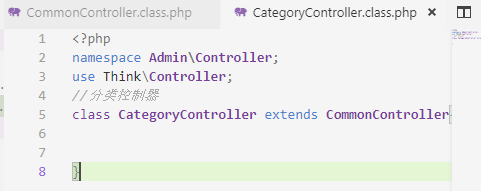


1. 修改index控制器继承关系

修改让其继承中间类以实现链式继承关系

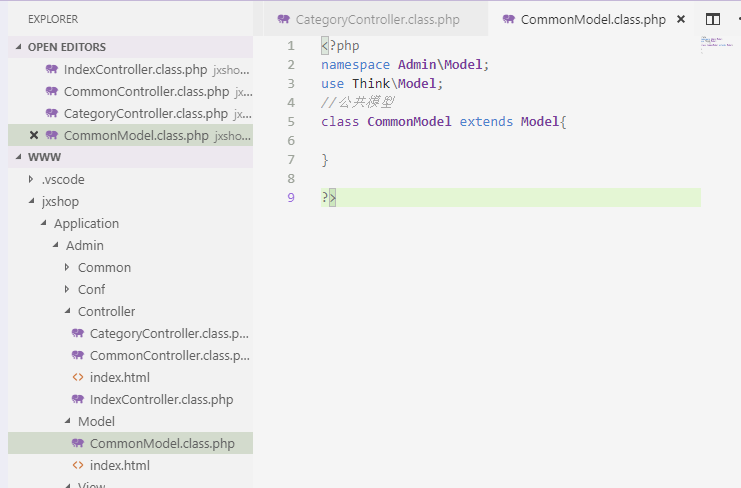


1. 创建分类的控制器

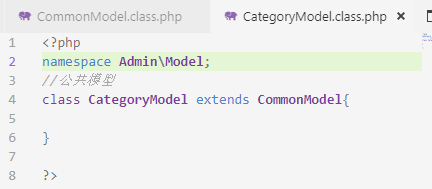


## 4..2.2 创建模型

1.创建公共模型



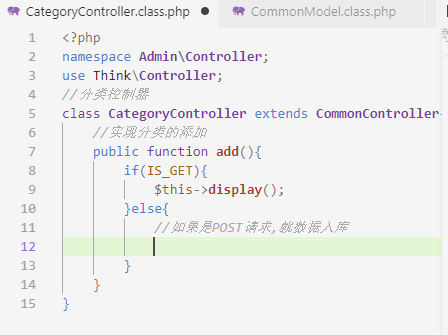
2.创建分类模型



## 4.3 商品分类的添加

### 4.3.1 展示出添加页面

1.创建方法展示模板



2.创建 admin/view/category/add.html

这个html页面可以发送post请求以向数据库添加数据,是一个后台管理界面

### 4.3.2数据入库

1.修改CategoryModel.class.php添加自定义字段

<?php

namespace **Admin\Model**;

*//公共模型*

class **CategoryModel** extends **CommonModel**{

*//自定义字段*

protected $fields=**array**('id','cname','parent\_id');

}?>

2.自动验证

<?php

namespace **Admin\Model**;

*//公共模型*

class **CategoryModel** extends **CommonModel**{

*//自定义字段*

protected $fields=**array**('id','cname','parent\_id');

*//自动验证*

protected $\_validate=**array**(

**array**('cname','require','分类名称必须填写'),

);

}

?>

### 4.4 实现可以添加为子分类

### 1.控制器中调用模型获取数据

if(IS\_GET){

*//获取格式化之后的分类xinxi*

$model=**D**('Category');

$cate=$model->**getCateTree**();*//获取格式化分类信息*

*//将信息复制给模板*

$this->**assign**('cate',$cate);

$this->**display**();

### 2.在模型中创建getCaetTree方法获取数据

1. *//获取格式化之后的数据*
2. public function **getCateTree**(){
3. *//先获取所有的分类信息*
4. $data=$this->**select**();
5. *//再对获取的信息进行格式化*
6. $list=$this->**getTree**($data);
7. **dump**($list); *//虽然不懂原理,该行直接显示所有取出的数据*
8. return $list;
9. }

### 3.在模型中创建getTree方法格式化数据

略,用不到

### 4.模板中对数据进行展示

在html模板中写法为:

<volist *name*="cate" *id*="vo">*<!-- 该volist标签可以被TP框架检测并 功能为自动把cate当作vo遍历 -->*

{$vo.id}<b>{$vo.name}</b>{$vo.comment}</br> *<!-- vo就是cate数组的一个元素 -->*

</volist>

用这种方式可以遍历名为cate的变量 if(IS\_GET){

*//获取格式化之后的分类xinxi*

$model=**D**('Category');

$data=$model->**getCateTree**();*//获取格式化分类信息*

*//将信息复制给模板*

$this->**assign**('cate',$data); //控制类中则是这么写

Html中的值 数据实际的值

小记:在使用类似 {$vo.comment} 进行变量输出的时候有可用函数:

{$vo.lev|str\_repeat=”--‘,###}

变量名 替换功能:把’—‘重复变量本身次

<img src=”\_\_PUBLIC\_ADMIN\_\_/Images/<eq name=”vo.isrec” value=”1”>yes.gif<else />no.gif</eq>” />

在输出文本中进行if-else判断

更多请查阅手册 http://document.thinkphp.cn/manual\_3\_2.html#use\_function

$this->**display**();

控制类中则是这么写

顺便用js和boot strip写了个界面,还算方便

<!DOCTYPE *html*>

<html *lang*="en">

<head>

<meta *charset*="UTF-8">

<meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

<meta *http-equiv*="X-UA-Compatible" *content*="ie=edge">

<title>comment</title>

*<!-- 新 Bootstrap 核心 CSS 文件 -->*

<link *rel*="stylesheet" *href*="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.css">

*<!-- jQuery文件。务必在bootstrap.min.js 之前引入 -->*

<script *src*="http://cdn.bootcss.com/jquery/1.11.1/jquery.min.js"></script>

*<!-- 最新的 Bootstrap 核心 JavaScript 文件 -->*

<script *src*="http://cdn.bootcss.com/bootstrap/3.3.0/js/bootstrap.min.js"></script>

<script *src*="\_\_PUBLIC\_ADMIN\_\_/js/jquery.js"></script>

<script>*//发送评论*

function **postComment**()

{

var val1 = document.**getElementById**("commontTextarea").value;

var val2 = document.**getElementById**("nameInput").value;

$.**post**(window.location.href, {'name':val1, 'comment':val2});

location.**reload**();

}

</script>

</head>

<body>

<p>add</p>

<volist *name*="cate" *id*="vo">*<!-- 该volist标签可以被TP框架检测并 功能为自动把cate当作vo遍历 -->*

{$vo.id}<b>{$vo.name}</b>{$vo.comment}</br> *<!-- vo就是cate数组的一个元素 -->*

</volist>

<div *class*="form-group" >

<div *class*="" *style*="padding: 10px 10px 10px;">

<textarea *id*="commontTextarea" *class* ="form-control" *type*="text" *name*="comment" *placeholder*="有什么想说的吗"></textarea>

<input *id*="nameInput" *class* ="form-control" *type*="textarea" *name*="name" *placeholder*="昵称"/>

<button *onclick*="**postComment**()" *class* ="form-control" *type*="submit" *value*="提交" >发送</button>

</div>

</div>

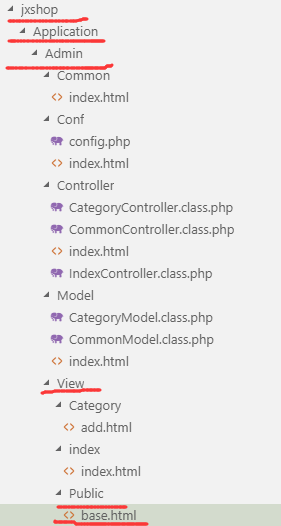
</body>

</html>

## 4.5模板继承使用

### 1.在视图目录下创建public目录(专门存储各种公共模板)

### 2.复制category中add的模板改名为base



### 3.对公共模板进行修改(只留下所有页面都会用到的代码)

<!DOCTYPE *html*>

<html *lang*="en">

<head>

<meta *charset*="UTF-8">

<meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

<meta *http-equiv*="X-UA-Compatible" *content*="ie=edge">

<title><block *name*="title"></block></title>

*<!-- 新 Bootstrap 核心 CSS 文件 -->*

<link *rel*="stylesheet" *href*="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.css">

*<!-- jQuery文件。务必在bootstrap.min.js 之前引入 -->*

<script *src*="http://cdn.bootcss.com/jquery/1.11.1/jquery.min.js"></script>

*<!-- 最新的 Bootstrap 核心 JavaScript 文件 -->*

<script *src*="http://cdn.bootcss.com/bootstrap/3.3.0/js/bootstrap.min.js"></script>

</head>

<body>

<block *name*="description"></block>*<!-- 说明该页面功能 -->*

<block *name*="body"></block>

<pre *id*="footer">

there is nothing here.</br>

本次查询共使用0.162348 秒,IPV6已禁用,内存占用34.95MB

</pre>

<block *name*="js"></block>*<!--用来写js-->*

</body>

</html>

剩下的可能变动的地方使用<block name=”名称”></block>代替

### 4.对分类的添加模板使用继承

在文本开头添加 <extend *name*="Public:base"/>

估计这个路径是从View下开始写的

套用了继承,其余的代码就被精简成只要填写block块就行了.同样的,使用name=”block块中填写的名称”就能对应的上

<extend *name*="Public:base"/>

<block *name*="title">comment</block>

<block *name*="description">留言展示与评论发布</block>

<block *name*="body">

<volist *name*="cate" *id*="vo">*<!-- 该volist标签可以被TP框架检测并 功能为自动把cate当作vo遍历 -->*

{$vo.id}<b>{$vo.name}</b>{$vo.comment}</br> *<!-- vo就是cate数组的一个元素 -->*

</volist>

<div *class*="form-group" >

<div *class*="" *style*="padding: 10px 10px 10px;">

<textarea *id*="commontTextarea" *class* ="form-control" *type*="text" *name*="comment" *placeholder*="有什么想说的吗"></textarea>

<input *id*="nameInput" *class* ="form-control" *type*="textarea" *name*="name" *placeholder*="昵称"/>

<button *onclick*="**postComment**()" *class* ="form-control" *type*="submit" *value*="提交" >发送</button>

</div>

</div>

</block>

<block *name*="js">

<script>*//发送评论*

function **postComment**(){

var val1 = document.**getElementById**("commontTextarea").value;

var val2 = document.**getElementById**("nameInput").value;

$.**post**(window.location.href, {'name':val1, 'comment':val2});

location.**reload**();

}

</script>

</block>

### 4.7评论的删除

Html中,使用js发送post删除请求

$.**post**(window.location.href, {'id':id});

绑定到button上

<button *onclick*="**delet**('{$vo.id}')">删除</button><

会转到Controller类中

*//实现删除*

public function **admin**(){

if(IS\_GET){

*//获取格式化之后的分类xinxi*

$model=**D**('Category');

$data=$model->**getCateTree**();*//获取格式化分类信息*

*//将信息复制给模板*

$this->**assign**('cate',$data);

$this->**display**();

}else{

$id= **intval**(**I**('post.id'));

if($id<=0){

$this->**error**('参数错误啦~~'.$id);

}

$model=**D**('Category');*//实例化模型对象*

*//调用模型中的删除方法实现删除*

$res=$model->**del**($id);

if($res===false){

$this->**error**('删除失败');

}

$this->**success**('删除成功');

}

}

其中调用了Model中的del函数并传递了参数$id

Model中删除也很简单

*//删除评论*

public function **del**($id){

return $this->**where**('id='.$id)->**delete**();

}

### 4.8评论的修改

在controller中添加代码

}else if(**I**('post.operation'=='edit')){

*//更改数据*

$model=**D**('Category');

$data=$model->**create**();

if(!$data){

$this->**error**($model->**getError**());*//打开数据库失败*

}

$res=$model->**update**($data);

if($res===false){

$this->**error**('修改失败');

}

$this->**success**('修改成功');

}

其中调用了模型(model)中的**update**

*//修改评论*

public function **update**($data){

*// dump($data);exit();*

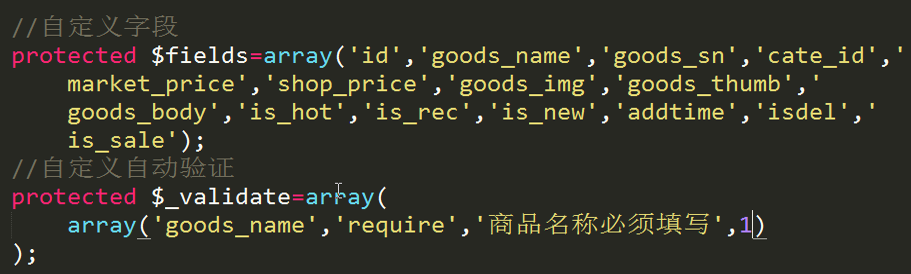
$this->**save**($data); //嗯,就这么简单…

}

# 5 数据库的一系列自动化骚操作

(具体自己看文档 <http://document.thinkphp.cn/manual_3_2.html#insert_data>)

新建一个模型文件TestModel.class.php



用特定字段fields按顺序新建数据表模型

\_validate数组

Array(‘出错的数据段’,’require’,’返回的报错信息’,1代表必须填写),

# 6 上传图片

## 5.1实现图片上传及缩略图制作

### 1.手动创建Uploads文件夹

jxshop\wwwroot\Uploads

### 2.在模型类中调用钩子函数\_before\_insert处理上传的文件

有几个钩子函数见<http://www.thinkphp.cn/topic/52249.html>

### 3.(上面的操作并没有用到)直接上传,向数据库中写入记录信息

具体的写入<https://www.kancloud.cn/manual/thinkphp/1764> 有说明

<?php

namespace **Admin\Controller**;

use Think\**Controller**;

*//分类控制器*

class **UploadController** extends **CommonController**{

public function **img**(){

if(IS\_GET){

$this->**display**();

}else if(IS\_POST){

*// $this->display();*

*//实现图片上传*

$upload=new \Think\**Upload**(); *//实例化 \Think\Upload 类*

$info = $upload->**uploadOne**($\_FILES['img']);

if(!$info) {*// 上传错误提示错误信息*

$this->**error**($upload->**getError**());

}

$upload->saveName = 'com\_create\_guid';

$path=$info['savepath'].$info['savename'];*//保存的本地路径 从upload文件夹下开始写*

$info['name']==''?$info['name']='nano.jpg':'';

*// 写入数据库*

$Upload=**D**('upload');

$Upload->**create**();

$Upload->url=$path;

$Upload->used='y';

$Upload->imgName=$info['name'];

$Upload->**add**();

**echo**('UploadSuccess');

}

}

}