目录

[一、 简介 3](#_Toc401754129)

[1、 什么是ThinkPHP 3](#_Toc401754130)

[2、 什么是框架 3](#_Toc401754131)

[3、 什么是MVC 3](#_Toc401754132)

[4、 获取并布署Thinkphp 4](#_Toc401754133)

[5、 快速入门 4](#_Toc401754134)

[二、 控制器 9](#_Toc401754135)

[1、 什么是控制器 9](#_Toc401754136)

[2、 定义控制器 9](#_Toc401754137)

[3、 URL调度模式 10](#_Toc401754138)

[4、 空操作 12](#_Toc401754139)

[5、 空模块 13](#_Toc401754140)

[6、 项目分组 14](#_Toc401754141)

[7、 URL大小写 15](#_Toc401754142)

[8、 页面跳转 15](#_Toc401754143)

[9、 页面重定向 16](#_Toc401754144)

[三、 视图 17](#_Toc401754145)

[1、 什么是视图 17](#_Toc401754146)

[2、 定义模板 17](#_Toc401754147)

[3、 分隔符 17](#_Toc401754148)

[4、 模板赋值 18](#_Toc401754149)

[5、 模板输出 19](#_Toc401754150)

[6、 模板替换 19](#_Toc401754151)

[7、 获取内容 20](#_Toc401754152)

[8、 模板注释 21](#_Toc401754153)

[9、 变量输出 21](#_Toc401754154)

[10、 系统变量 22](#_Toc401754155)

[11、 使用函数 23](#_Toc401754156)

[12、 默认值 23](#_Toc401754157)

[13、 运算符 23](#_Toc401754158)

[14、 内置标签 24](#_Toc401754159)

[15、 包含文件 24](#_Toc401754160)

[16、 导入文件 26](#_Toc401754161)

[17、 volist 26](#_Toc401754162)

[18、 foreach 28](#_Toc401754163)

[19、 if、elseif、else 28](#_Toc401754164)

[20、 使用php代码 29](#_Toc401754165)

[四、 模型 30](#_Toc401754166)

[1、 什么是模型 30](#_Toc401754167)

[2、 定义模型 30](#_Toc401754168)

[3、 配置数据库信息 31](#_Toc401754169)

[4、 实例化 31](#_Toc401754170)

[5、 CURD操作 32](#_Toc401754171)

[6、 定义字段 33](#_Toc401754172)

[7、 创建数据 35](#_Toc401754173)

[8、 字段映射 37](#_Toc401754174)

[9、 表名操作 38](#_Toc401754175)

[10、 事务处理 39](#_Toc401754176)

[11、 AR模式 40](#_Toc401754177)

[12、 联贯操作 43](#_Toc401754178)

[13、 统计查询 43](#_Toc401754179)

[五、 实用项 44](#_Toc401754180)

[1、 设置trace信息 44](#_Toc401754181)

[2、 session 44](#_Toc401754182)

[3、 cookie 45](#_Toc401754183)

[4、 文件载入 46](#_Toc401754184)

[5、 验证码 48](#_Toc401754185)

[6、 数据分页 50](#_Toc401754186)

[7、 文件上传 51](#_Toc401754187)

[8、 多表连接 52](#_Toc401754188)

[9、 Ajax应用 54](#_Toc401754189)

[10、 关联模型 56](#_Toc401754190)

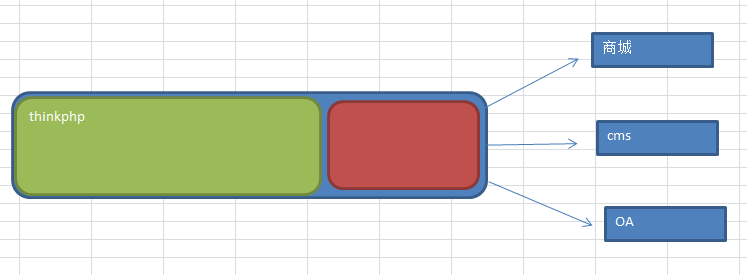
1. 简介
2. 什么是ThinkPHP

它是一个框架

它是一个基于MVC的框架

1. 什么是框架

框架，即framework。其实就是某种应用的半成品，就是一组组件，供你选用完成你自己的系统。



使用框架的好处：

* 代码重用
* 合理分工
* 解决多人协作开发问题

常见的php框架：

* Zend Framework
* Yii
* Symfony
* Codelgniter
* ThinkPHP
* ……

1. 什么是MVC

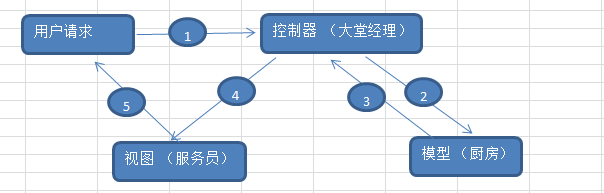
MVC是施乐帕克研究中心 (Xerox PARC)在八十年代为编程语言Smalltalk－80发明的一种软件设计模式，至今已被广泛使用。

MVC是一个设计模式，它强制性的使应用程序的输入、处理和输出分开。使用MVC应用程序被分成三个核心部件：模型、视图、控制器。

M：Model 模型

V：View 视图

C：Control 控制器



1. 获取并布署Thinkphp

* <http://thinkphp.cn>
* 下载thinkphp3.0
* 直接部署到项目中
* php5.2

thinkphp php5.0

thinkphp3.0 php5.2

开发thinkphp项目流程：

* 创建项目
* 创建入口文件
* 配置项目
* 创建控制器
* 创建模型
* 创建模板
* 运行调试，调优
* 部署到生产环境

1. 快速入门
2. 解压thinkphp



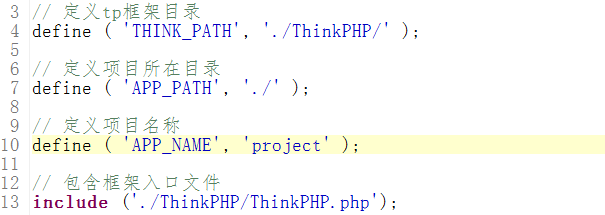
Examples：案例

ThinkPHP：tp框架的源代码

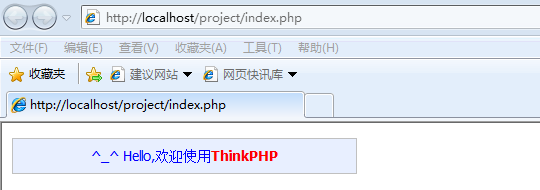
1. 创建项目，并将tp放入到项目中



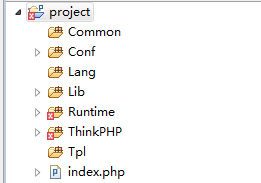
编写项目入口文件（index.php）信息



运行项目，看到以下界面



观察项目目录，会自动产生一些项目文件



我们可以在任何位置为任何目录创建项目，只需要改变相关目录常量的定义即可。

1. 如果你的目录结构与上题中的相同，相关常量的定义可以省略





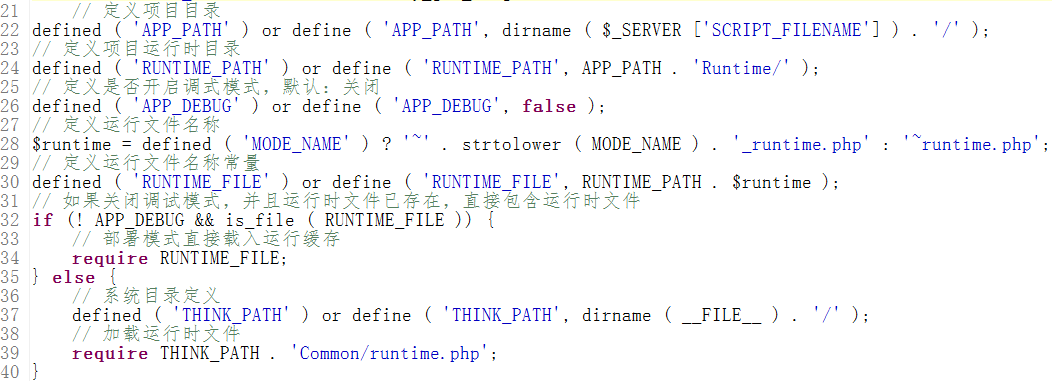
1. 创建出来的项目目录

* Common 项目公共文件目录
* Conf 项目配置目录
* Lang 项目语言包目录
* Lib 项目类库目录
* Tpl 项目模板目录
* Runtime 项目运行时目录

包含常用信息的定义、常用函数的定义、数据文件的缓存

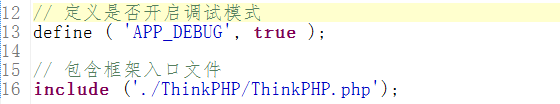
* ~runtime.php ：常用函数和常量的定义

1. 框架入口文件执行过程



可以看出，如果是第一次执行，或者开启了调试模式，程序会自动加载框架内的runtime文件，否则，会加载项目的runtime文件

1. 关于调试模式



如果开启调试模式，不会产生项目的runtime文件

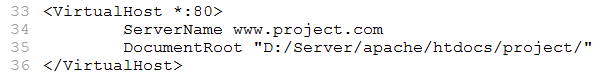
runtime.php文件保存了项目中的一些常量定义、常用函数的定义，但是一个项目在开发的过程中，数据表结构、路径有可能会经常变量，如果有runtime.php，那么可能不能即时看到最新变化。

所以，我们建议：

开发时，开启调试模式

布署时，关闭调试模式

1. 配置虚拟主机





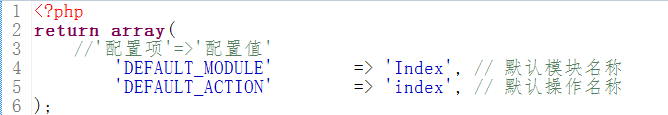


1. 为什么可以看到上面这个界面

入口文件/模块名/动作名

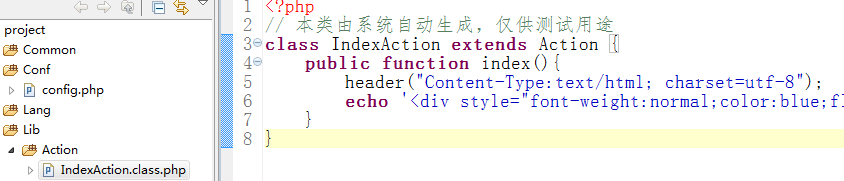
它执行了默认的模块下的默认的方法

将tp默认的配置复制我们的配置文件中：



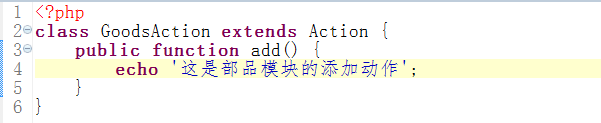
它执行了Index模块的index方法。

打开index控制器：



1. 创建商品模块的添加方法

在Lib下创建控制器及方法：



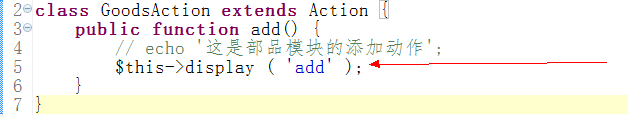
访问时：



1. 为add方法添加模板

在tpl目录下创建Goods文件夹，表示所有Goods模块下的模板所在路径





运行：



1. 控制器
2. 什么是控制器

控制器接受用户的输入并调用模型和视图去完成用户的需求。

1. 定义控制器

Lib/Action

class UserAction extends Action{

public function add(){

$this->display();

}

}





1. URL调度模式

url调度模式就是指在tp中访问页面url的方式

在tp中，调度模式共有四种，可以通过配置选项：URL\_MDOEL来设置



0：普通模式

1：pathinfo模式

2：rewrite模式

3：兼容模式

1. 普通模式



语法：

http://项目/入口文件?g=分组名&m=模块名&a=动作名

g：分组名

m：模块名

a：动作名



1. pathinfo模式

这种也是tp推荐使用的模式，也是默认的



语法：

http://项目/入口文件/分组名/模块名/动作名



1. rewrite模式

语法：

http://项目/分组名/模块名/动作名

* httpd.conf配置文件中加载了mod\_rewrite.so模块
* AllowOverride None 将None改为 All
* 确保URL\_MODEL设置为2
* 把下面的内容保存为.htaccess文件放到入口文件的同级目录下
* <IfModule mod\_rewrite.c>

RewriteEngine on

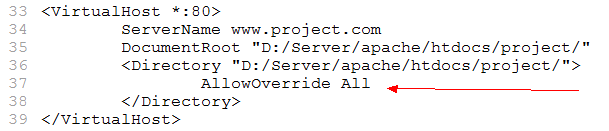
RewriteCond %{REQUEST\_FILENAME} !-d

RewriteCond %{REQUEST\_FILENAME} !-f

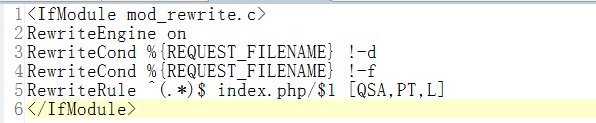
RewriteRule ^(.\*)$ index.php/$1 [QSA,PT,L]

* </IfModule>











1. 兼容模式

有些web服务器可能不支持pathinfo模式，但我们还想使用，可以考虑使用这个模式

语法：

http://项目/入口文件?s=/分组名/模块名/动作名



访问：



* 在访问的url中，我们可以自己指定分隔符

URL\_PATHINFO\_DEPR



<http://www.sohu.com/index.php/Admin,Goods,add>



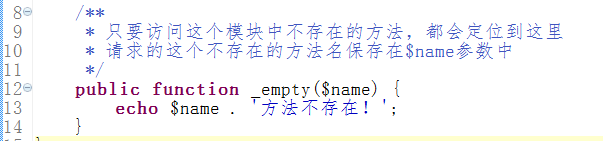
1. 空操作

空操作是指系统在找不到指定的操作方法的时候，会定位到空操作（\_empty）方法来执行，利用这个机制，我们可以实现错误页面和一些URL的优化。

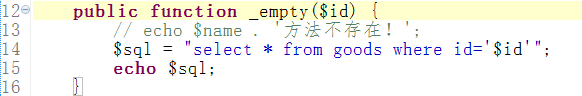
语法：

* \_empty($name)

GoodsAction.class.php



实现url的优化：



运行时：



1. 空模块

空模块的概念是指当系统找不到指定的模块名称的时候，系统会尝试定位空模块(EmptyAction)，利用这个机制我们可以用来定制错误页面和进行URL的优化。

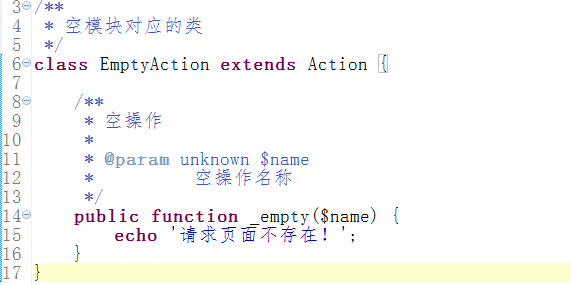
语法：

class EmptyAction extends Action{}

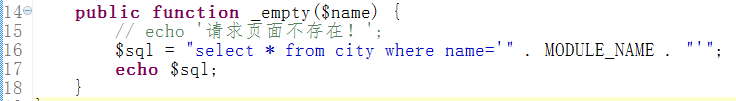
MODULE\_NAME

订制错误页面：

EmptyAction.class.php



实现URL优化：





注意：空模块中的常量 MODULE\_NAME表示请求的当前的不存在的模块名

1. 项目分组

前台

后台

博客

会员

商品模块

前台商品模块

商品信息的浏览

后台商品模块

商品信息的维护

在项目，如果需要分组，修改config.php配置文件，添加以下两个选项即可：

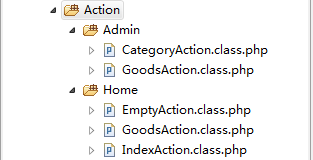
config.php

* APP\_GROUP\_LIST ：分组列表
* DEFAULT\_GROUP ：默认分组

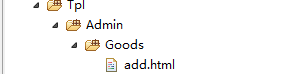


配置完分组信息后：要对控制器目录与模板目录进行分组

控制器分组：



模板分组：



1. URL大小写

如果想访问时，url不区分大小写，可以修改配置文件：

config.php

‘URL\_CASE\_INSENSITIVE’=>true



访问时：



1. 页面跳转

* success ：作成功后调用的方法
* error ：操作失败后调用的方法

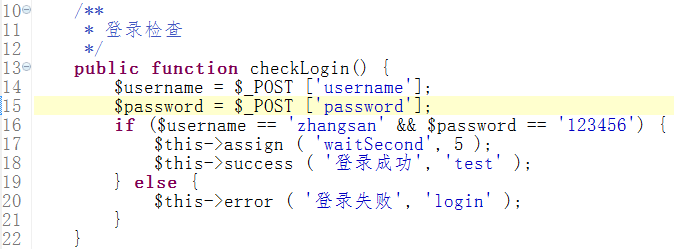
在配置文件中，有两个选项用于控制以上两个方法所使用的模板

config.php

* TMPL\_ACTION\_SUCCESS
* TMPL\_ACTION\_ERROR



MemberAction.class.php



1. 页面重定向

* redirect

$this->redirect( “goods/show”, array(‘id’=>2), 5, “页面跳转中”);

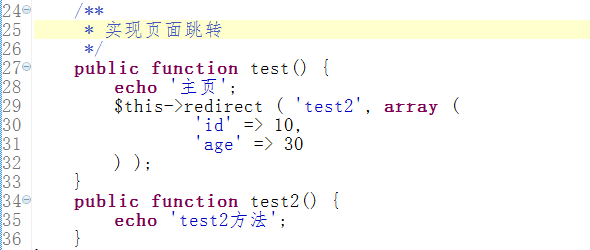
参数1：要跳转的url

参数2：传递的参数

参数3：停留秒数

参数4：提示信息

MemberAction.class.php



1. 视图
2. 什么是视图

视图就是用户看到的并与之交互的界面

在tp中，视图包含两部分：

1. 视图类
2. 模板
3. 定义模板

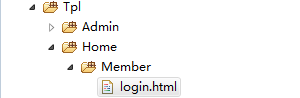
默认的模板文件定义规则：

模板目录/[分组名/]模块名/操作名+模板后缀

在配置文件中，下面这个选项用户有设置模板后缀

TMPL\_TEMPLATE\_SUFFIX





1. 分隔符

* 'TMPL\_L\_DELIM'=>'<{',
* 'TMPL\_R\_DELIM'=>'}>',

{$name}



1. 模板赋值

assign($name,$value);

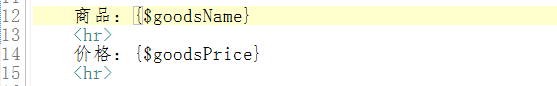
$name：模板变量名

$value：要赋值的模板变量的值

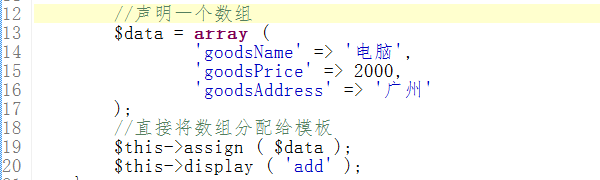
GoodsAction.class.php



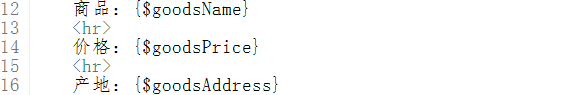
模板：



在tp视图中，有这样一个功能，可以向模板直接分配一个数组变量，然后数组中的元素名就是模板变量名，元素的值就是模板变量的值



模板：



运行：



1. 模板输出

$this->display();

display();

$this->display(); 调用与当前操作同名的模板

display(‘操作名')

$this->display(‘add’); 调用当前模块下的add.html模板

display(‘模块名:操作名')

$this->display(“Category:show”); 调用Category模块下的show.html模板

display(‘show’,‘utf-8', ‘text/xml');

参数1：模板

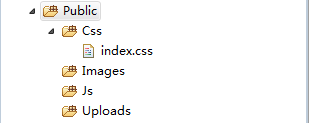
参数2：编码

参数3：输出的内容类型 Content-Type:text/xml

1. 模板替换

在TP的模板中，有一些模板字符串常量，当执行时，会自动被解析成为一些路径：

* \_\_PUBLIC\_\_：当前网站的公共目录
* \_\_APP\_\_： 当前项目的URL地址
* \_\_GROUP\_\_：当前分组的URL地址
* \_\_URL\_\_： 当前模块的URL地址
* \_\_ACTION\_\_：当前操作的URL地址



引用css代码：



运行：

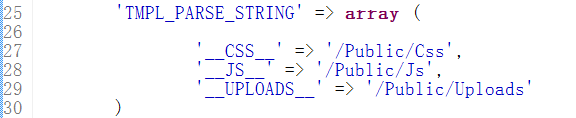




同时，在配置文件中，可以加入选项：

* TMPL\_PARSE\_STRING

修改config.php



Goods/add.html





运行：



1. 获取内容

* $this->fetch();

fetch

读取 替换 返回

display

读取 替换 输出



1. 模板注释

语法：

* {/\* 注释内容 \*/ } 多行注释
* {// 注释内容 } 单行注释



html注释会通过http响应给客户端

thinkphp注释不会响应给客户端（它是服务器端解释）

1. 变量输出

语法：

{$name}

在tp可，向模板中赋值变量共有三种：

普通变量

{$name}

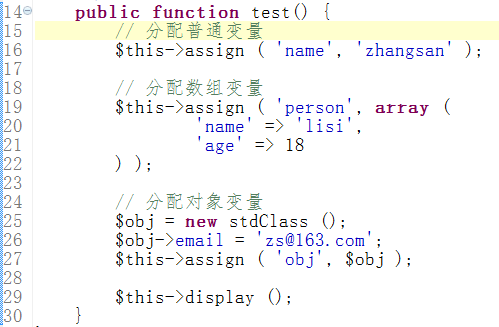
数组变量

{$product[‘name’]}、{$product.name}

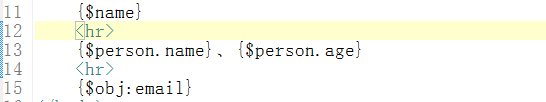
对象变量

{$obj:name}、{$obj:say()}

CategoryAction.class.php



模板：



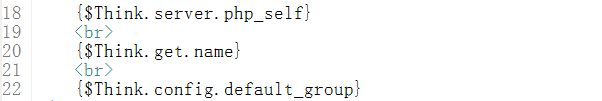
运行：



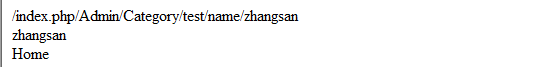
1. 系统变量

在tp的模板中，有以下内置的系统变量

* $Think.server ：$\_SERVER
* $Think.get ：$\_GET
* $Think.post ：$\_POST
* $Think.request ：$\_REQUEST
* $Think.cookie ：$\_COOKIE
* $Think.session ：$\_SESSION
* $Think.config ：获取tp配置文件中的信息



运行：



1. 使用函数

语法：

{$name|fn1|fn2=arg1,arg2,### }



运行：



1. 默认值

当一个变量的值为空时，会执行默认值

语法：

{$变量|default="默认值"}



模板：



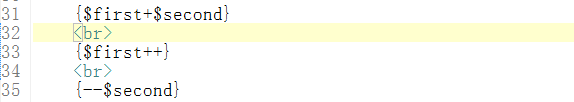
运行：



1. 运算符

* + {$a+$b}
* - {$a-b}
* \* {$a\*$b}
* / {$a/$b}
* % {$a%$b}
* ++ {$a++} 或  {++$a}
* -- {$a--}  或 {--$a}

在tp中的模板里，可以直接使用运算符对变量进行数学运算



运行：



1. 内置标签

{$name}

标签语法：

<tag></tag>

<tag />

tp中的内置标签共分两种：

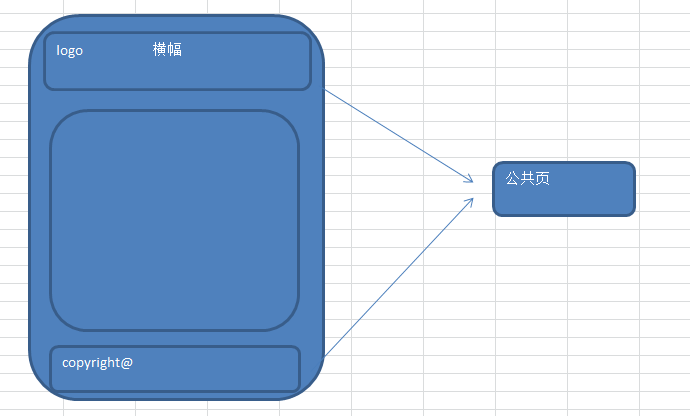
开放标签 <tag></tag>

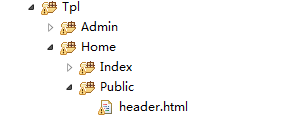
闭合标签 <tag />

1. 包含文件

语法：

<include file="模板文件名" />





index.html：



在include标签中，可以使用这种语法：

<include file=’模块名:操作名’ />



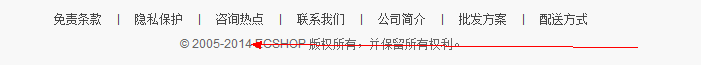
运行：



include标签在引用模板时，也可以传递参数过去：



运行：



1. 导入文件

导入文件是指对资源文件的导入，指css、js文件

语法：

<import type=‘type’ file=‘file’ />

file（必须）：资源文件

type（可选）：资源文件类型，取值：js和css 默认为js

起始路径是网站的Public目录

采用命名空间方式

以前的引入方式：



现在的引入方式：



1. volist

volist标签用于解析数组

语法：

<volist name="list" id="vo">

{$vo.id}

{$vo.name}

</volist>

name：要解析的数组变量名

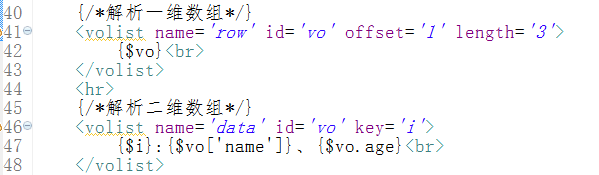
id：将解析到的变量放在指定的变量中

offset： 要输出数据的offset

length： 输出数据的长度,需要指定offset

key： 循环索引 默认值：i 从1开始

模板：



控制器：



运行结果



1. foreach

foreach语句用于遍历数组

语法：

<foreach name="list" item="vo" >

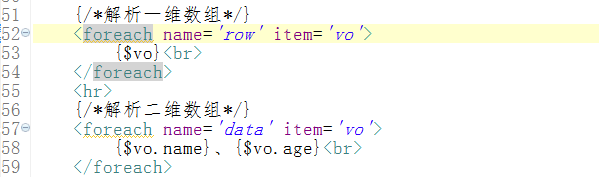
{$vo.id}

{$vo.name}

</foreach>

name：要遍历的数组名

item：每次遍历到的数组元素



运行：



foreach与volist相比，没有其余的属性

1. if、elseif、else

对于简单的逻辑判断语句，建议使用if标签

语法：

<if condition="$name eq 1">

value1

<elseif condition="$name eq 2"/>

value2

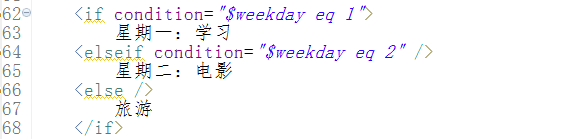
<else />

value3

</if>

* eq或者 equal：等于
* neq 或者notequal：不等于
* gt：大于
* egt：大于等于
* lt：小于
* elt：小于等于
* heq：恒等于
* nheq：不恒等于





运行结果：



1. 使用php代码

在tp的模板中，可以使用以下标记编写php代码：

1）<php>echo “hello”;</php>

2）<?php echo “think”; ?>

在配置文件中，有一个选项，模板中是否可以使用php

TMPL\_DENY\_PHP ：





在配置文件中，将以上选项设置为true，表示可以禁上上面的第二种写法，但是第一种写法还是支持的。

1. 模型
2. 什么是模型

模型表示企业数据和业务规则

1. 定义模型

模型文件：

Lib/Model

表名(首字母大写)+Model.class.php

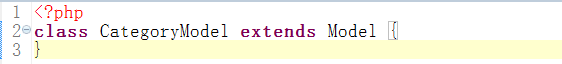
class 表名(首字母大写)Model extends Model{

}

数据库有哪些数据表，就有哪些模型，每个数据表都会对应一个模型

创建分类模型：

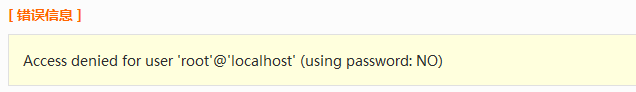




我们接下来，尝试对模型进行实例化：



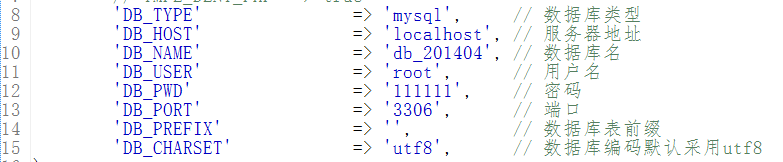
运行：



在tp中，实例化模型对象之前，必须要配置数据库相关信息，也就是说，创建模型对时，tp会自动创建与数据库服务器的连接

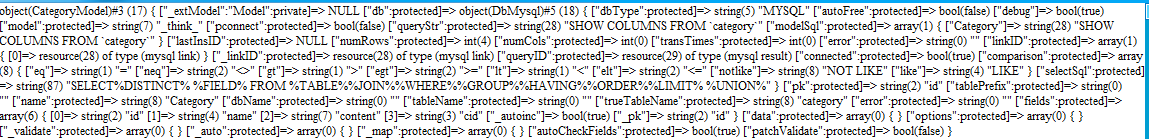
1. 配置数据库信息

打开tp框架的配置文件，将其中的数据库配置信息复制到我们的项目配置文件中：



* DB\_TYPE ：数据库类型
* DB\_HOST ：服务器地址
* DB\_USER ：用户名
* DB\_PWD ：密码
* DB\_NAME ：数据库名称
* DB\_PREFIX ：表前缀
* DB\_PORT ：端口
* DB\_CHARSET ：数据编码

重新运行上题：



说明配置信息正确！

1. 实例化

在tp中，模型共分两种：

自定义模型

实例化自定义模型

快速实例化自定义模型

基础模型（Model）

实例化基础模型

快速实例化基础模型

1. 实例化用户自定义模型





1. 快速实例化用户自定义模型





1. 实例化基础模型





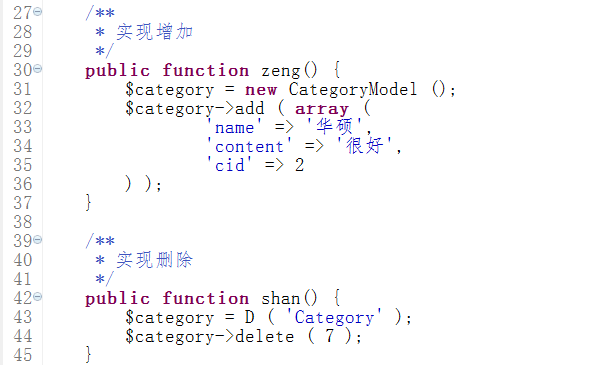
1. 快速实例化基础模型

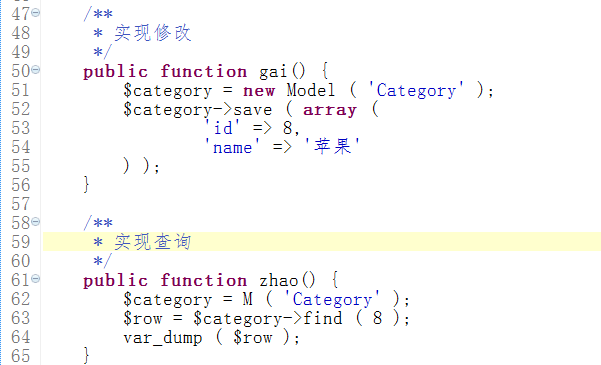




1. CURD操作

* 创建操作（Create）
* 读取数据（Read）
* 更新数据（Update）
* 删除数据（Delete）



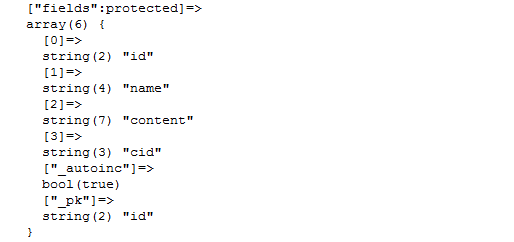


1. 定义字段

1）在tp里，一共有四种实例化方式，不管是哪种实例化形式，当实例化对象时，这时,tp会自动创建与数据库的联接，并解析模型对应的数据表

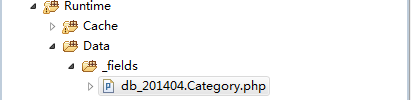






可以看出，无论是子类模型还是基础模型都会自动实现对表结构的解析

2）字段缓存

* 字段缓存保存在Runtime/Data/\_fields/ 目录下面  
  
* 只要是不开启调试模式都会生成字段缓存（字段缓存可以单独设置关闭）。



* 可以通过设置DB\_FIELDS\_CACHE 参数来关闭字段自动缓存

开发时，由于数据表结构可能不确定，建议关闭字段缓存功能

运行时，可以看出，每次都对对应的数据表结构进行解析

* 可以在模型类里面手动定义数据表字段的名称，可以避免IO加载的效率开销，在模型类里面添加fields属性即可

1. 定义字段的语法

* protected $fields=array(

‘id’,

‘username’,

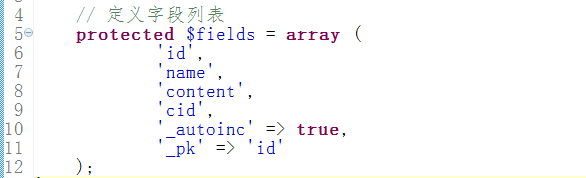
‘password’,

‘\_pk’=>‘id’,

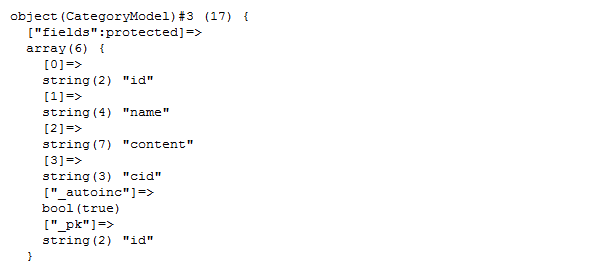
‘\_autoinc’=>true

);

CategoryModel.class.php



实例化对象后：



可以看出，定义字段列表后，对象创建时，不会解析表结构，而直接使用我们定义的字段列表

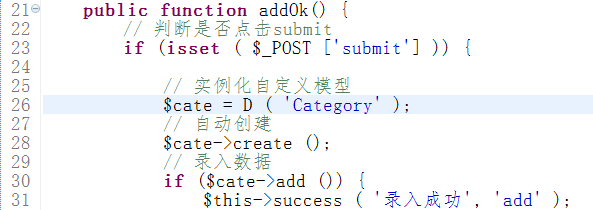
注意：字段列表的定义是在用户自定义类中，所以当我们实例化基础模型类时，是不会使用定义的列表，而是再次解析表结构。

1. 创建数据

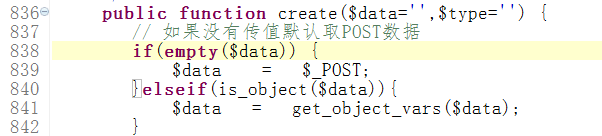
ThinkPHP可以帮助你快速地创建数据对象，最典型的应用就是自动根据表单数据创建数据对象，这个优势在一个数据表的字段非常之多的情况下尤其明显。

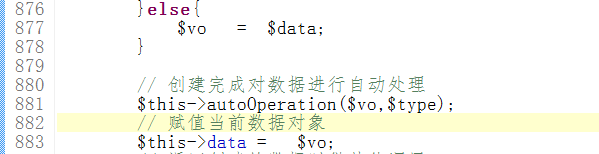
语法：

$model->create();



分析create方法执行过程：



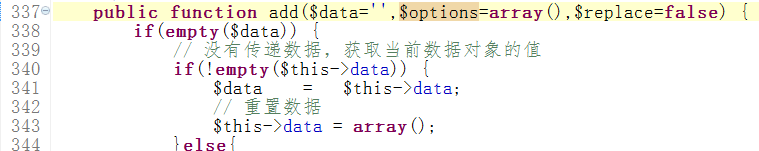


create方法的参数data是默认参数

首先判断data参数是否为空，如果为空，将$\_POST超全局变量数组中的数据赋值给参数data

然后，再将$data的值进行其它设置，再将数据赋值给$vo，最后将vo变量的值赋值给当前模型对象的data属性中，其中，data属性是一个数组

分析add方法执行：



add方法的参数是默认参数，我们调用完create方法后，直接调用add方法并没有传参数，所以参数data为空

下面继续判断，如果当前对象的data属性数组不为空，所以会将当前对象的data属性数组中的值赋值给参数data,便于下面代码的录入

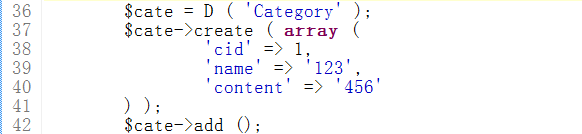
通过以上代码分析：

create方法的本质就是从一个数组中创建数据

假如表单以get形式提交数据，我们还想使用自动创建功能，怎么写？

$model->create($\_GET);

也就是说：下面代码是成立的



问题：

我们创建的表单中的元素名称必须要与数据表字段相同，但会在客户端源文件显露数据表结构。如果不同，这个字段的值就会丢失，如何解决这个问题？

1. 字段映射

ThinkPHP的字段映射功能可以让你在表单中隐藏真正的数据表字段，而不用担心放弃自动创建表单对象的功能

语法：

protected $\_map = array(

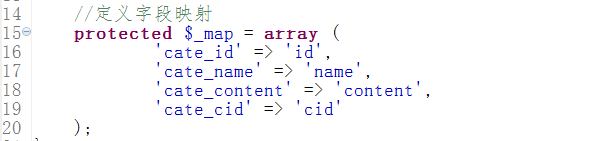
‘name'   => ‘username',

‘pwd’ => ‘userpwd’

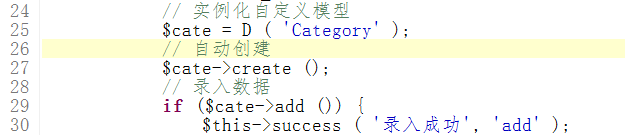
);

定义数组，数组元素是假名，数组的值是真名

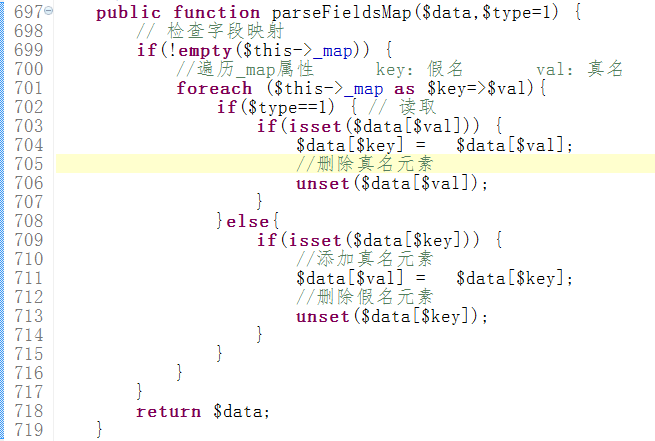
CategoryModel.class.php



CategoryAction.class.php



分析字段映射原理：



经过分析，可以看出，type参数的取值为：0或1

如果是0：会将数组中的假名元素全部移除，保留真名元素

如果是1：会将数组中的真名元素全部移除，保留假名元素

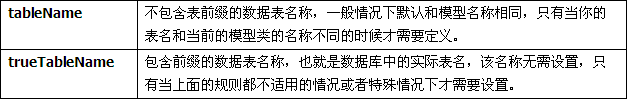
1. 表名操作

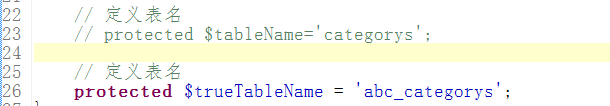
1）在一个数据库中，如果部署了多个项目，那么我们可以使用表前缀解决问题

‘DB\_PREFIX' =>‘think\_’



2）在ThinkPHP的模型里面，有两个关于数据表名称的属性定义：





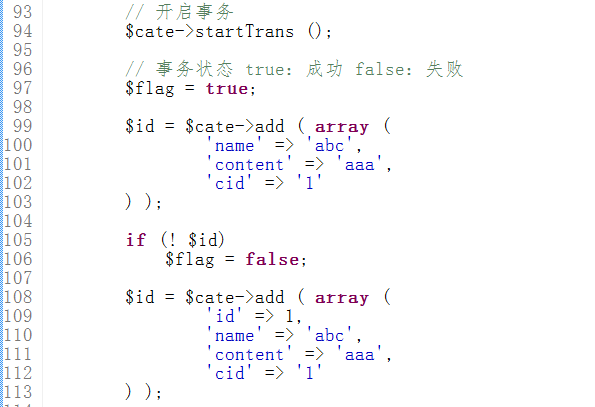
当你的表表前缀符合要求，但表名与模型类名不一致，这时使用tableName

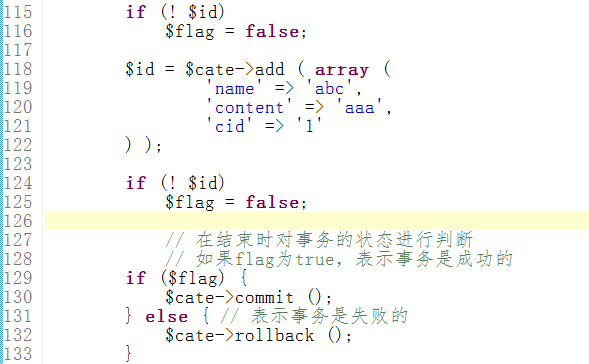
当你的表表前缀不符合要求，或没有表前缀，这时使用trueTableName

1. 事务处理

语法：

* $User->startTrans(); //开启事务
* $User->commit() ; //提交事务
* $User->rollback() ; //回滚事务





1. AR模式

AR：ActiveRecord

将数据表映射到类

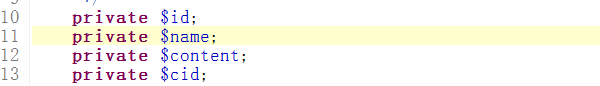
将数据表字段映射到属性

将数据映射到对象

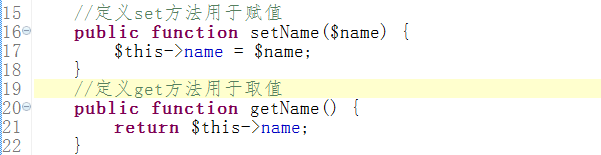
1. AR模式



我们写程序，强调封装性，将公有属性修改为私有成员属性：



改为私有之后，之前程序不能使用，不能在类外直接使用私有成员属性。

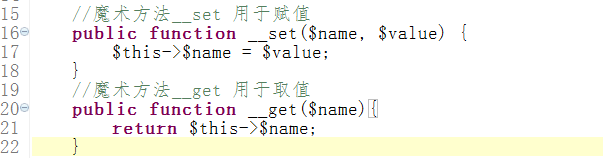


但这样一来，属性很多的情况下，需要定义很多方法，过于麻烦：

\_\_set($name,$value)

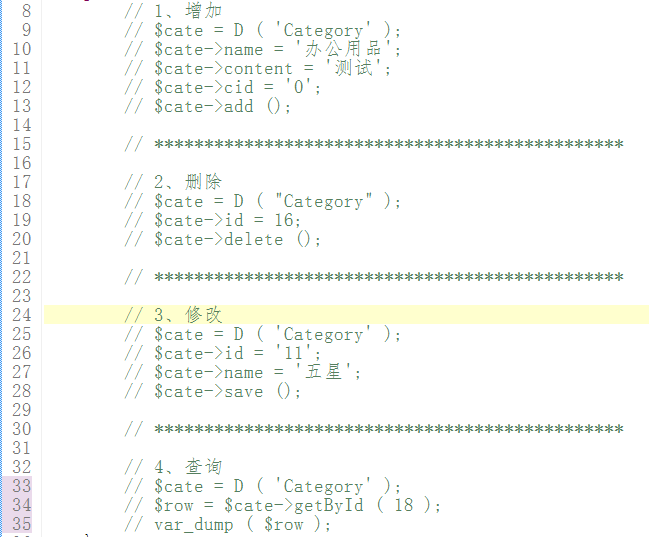
\_\_get($name)

当对一个不存在或不可访问的属性进行操作时，会自动执行\_\_set或\_\_get



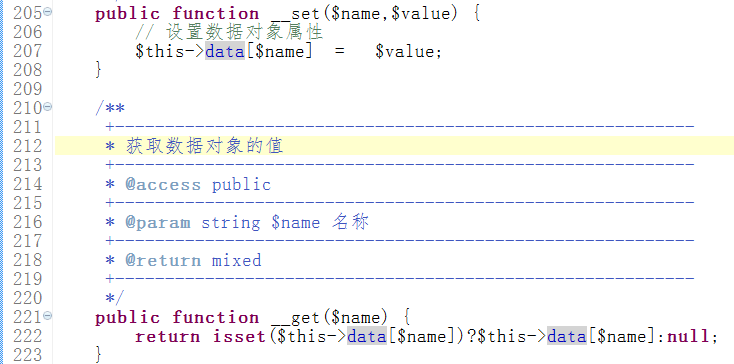
1. TP中的AR模式

CategoryModel.class.php



我们使用的对象直接调用了如：id、name、content、cid这些属性，但子类模型与父类模型并没有定义这样的属性，如何理解？

1. 分析：

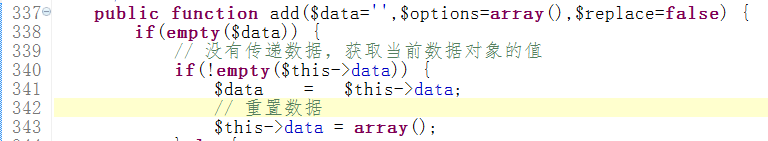


分析以上源码：

当我们用AR模式为模型对象赋值或取值时，其实是因为子类模型继承父类Model，父类中定义魔术方法\_\_set 与\_\_get

\_\_set在工作时，其实是向当前对象的data属性数组中添加一个新的元素，元素名以属性名命名

\_\_get在工作时，其实是从当前对象的data属性数组中获取同名的元素



add方法首先判断参数data是否为空，如果为空，继续判断当前对象的data属性数组是否不为空，然后将当前对象的data属性赋值给data参数变量 $data=$this->data

下面在增删改查时直接到data变量中取值即可。

1. 联贯操作
2. 语法：

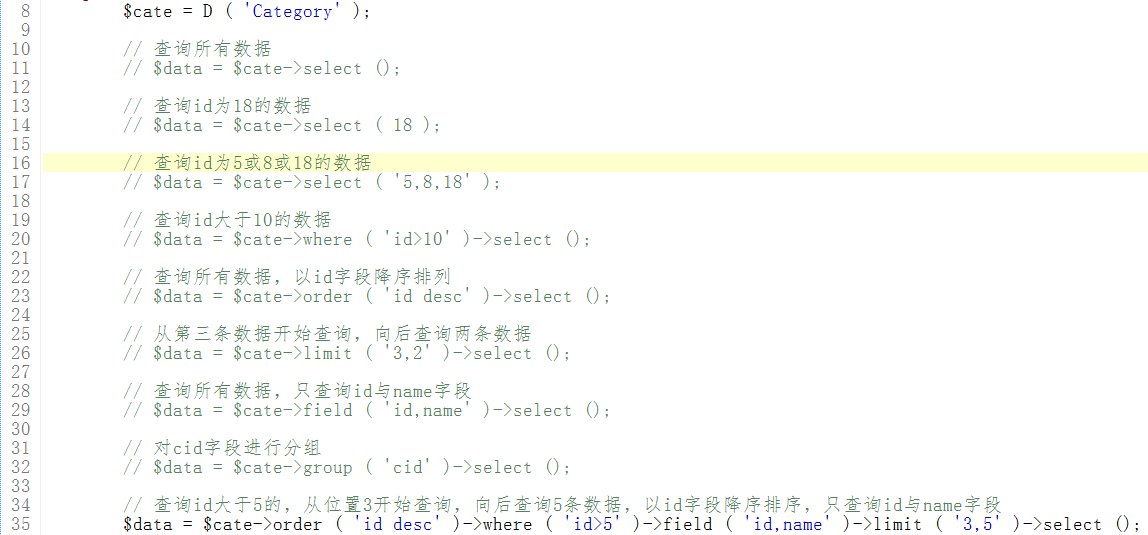
$model->方法1()->方法2()->方法n()->select();

支持联贯操作的方法主要有：

* where ：指定查询条件
* order ：排序
* limit ：查询指定的条数
* field ：指定查询的字段名称
* group ：分组查询

以上方法的最后必须连接select()查询方法，前面的方法顺序没有要求

1. 案例



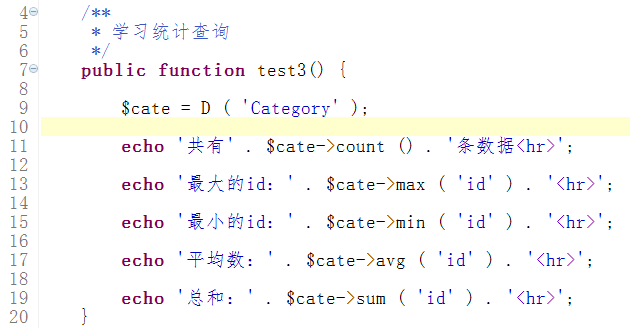
1. 统计查询

1）语法：

$model->统计查询方法();

* count ：统计行数
* max ：最大数
* min ：最小数
* avg ：平均数
* sum ：求和

1. 代码示例

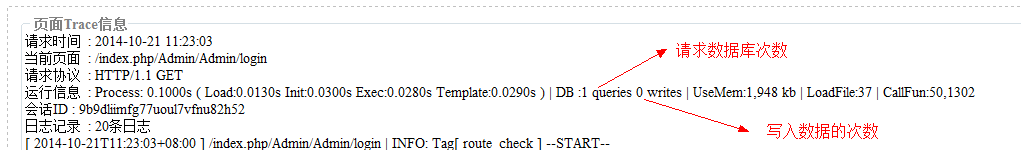


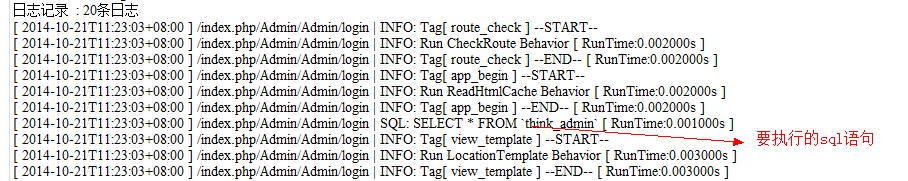
1. 实用项
2. 设置trace信息

在配置文件中，添加以下选项：



true：开启trace功能





1. session

使用session之前，必须开启session机制

在tp中，session机制默认是开启的

在config.php文件中：



tp中的session相关方法：

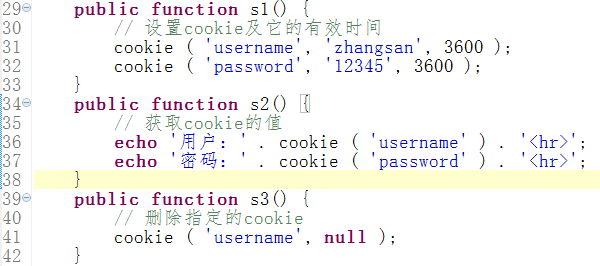
* session('name','value');  创建session并赋值
* $value = session('name'); 获取session的值
* session('name',null); 清空指定的session的值
* session(null); 清空所有session的值
* session('?name'); 判断指定的session是否设置

代码示例：



1. cookie

* cookie('name','value');  创建cookie
* cookie('name','value',3600); 创建cookie并设置它的有效时间
* $value = cookie('name'); 获取指定的cookie的值
* cookie('name',null); 删除指定的cookie



1. 文件载入
2. 载入函数文件

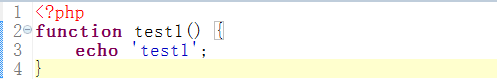
（1）Common/common.php

在项目的Common目录创建，创建common.php文件

Common目录



comm.php



控制器：



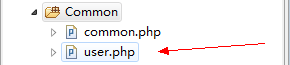
必须创建的文件名：common.php，如果是其它，不会自动加载，只有文件名为：common.php时，tp才会自动加载

common.php文件中创建的函数会被编译到~runtime.php文件中，下次执行时直接执行runtime.php文件中的函数，所以如果函数发生改变，不会直接看到最新效果。

（2）config.php

LOAD\_EXT\_FILE=“user”

在Common目录下创建php文件，保存自己的函数，文件名任意指定



但这个文件不会被自动加载，所以需要在配置文件config.php中，加入选项：

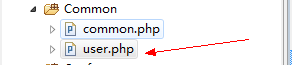


可以看出，这样引用的文件的函数不会被编译到runtime.php文件中，如果修改函数内容，可以即时看到最新效果。

（3）Common/user.php

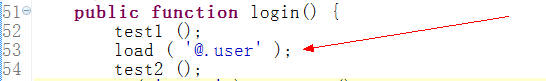
以上两种做法，都是tp对文件实行自动加载，但是在很多时候，某一个页面是用不到这么多的方法，比较浪费资源，我们需要让程序在需要的时候再去加载函数文件。

在Common下创建user.php



在控制器方法内，使用load函数对用户自定义的函数文件进行动态加载

load(‘@.user’);



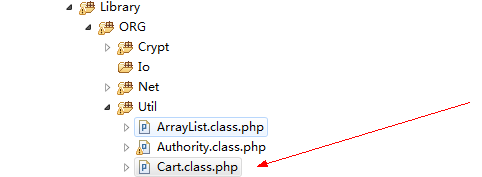
观察runtime.php文件，发现test2方法没有写入进去，说明每次执行时都会在源文件中去执行

1. 载入类文件

如：验证码类、分页类、画图类

1. import(‘ORG.Util.User’);

将类文件放在ORG/Util下：



在控制器中，如果想使用这个类，可以使用import方法导入这个类文件



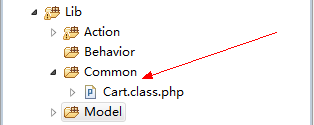
import是tp提供的方法，默认要对于thinkphp框架中的extend/Library文件夹，引用文件时采用命名空间方式：

文件夹.文件夹.文件

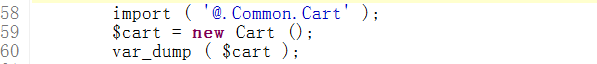
（2）import(‘@.Common.User’); Lib/Common/User.class.php

以上做法将文件放在了TP框架内，如果只想将文件放在项目中：

在Lib目录下，创建Common文件夹，将用户自定义类文件放在其中：

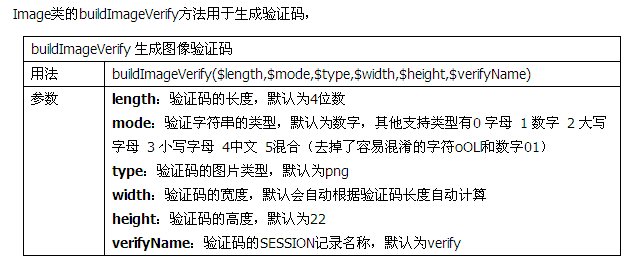


在控制器中，使引用这个文件时：



1. 验证码
2. 生成数字、英文验证码

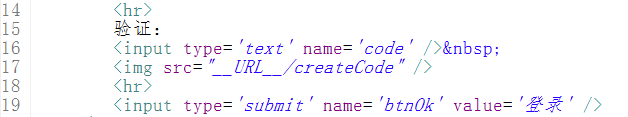
* import(‘ORG.Util.Image’);
* Image::buildImageVerify();



在admin控制器，创建生成验证码的方法：



在模板中，创建图像标签引入这个动作：



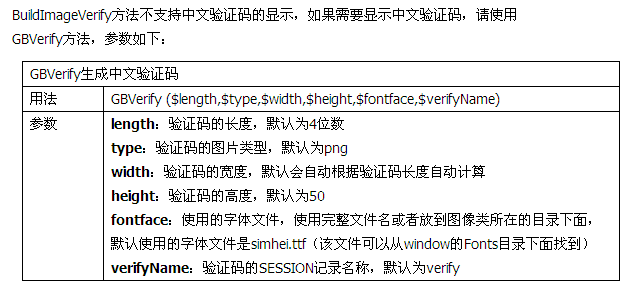
最后，用户提交表单对验证码进行判断



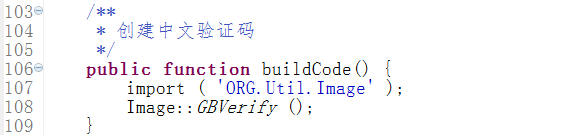
注意：验证码类会对验证进行md5加密。

1. 生成中文验证码

* import(‘ORG.Util.Image’);
* Image::GBVerify();



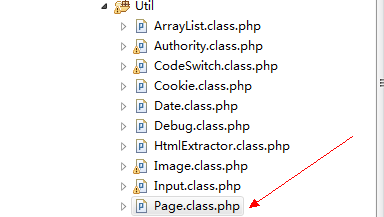
在控制器中创建方法用于生成中文验证码：



将字体库文件放在与Image.class.php文件平级目录下

1. 数据分页

1）在ThinkPHP/extends/Library/ORG/Util/Page.class.php文件，可以实现分页功能

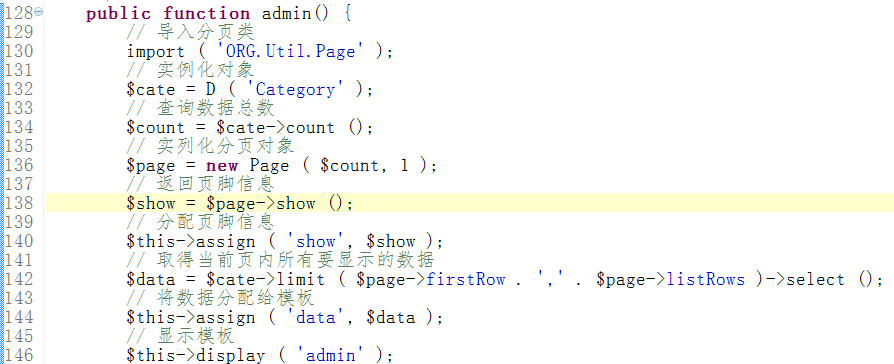


import(‘ORG.Util.Page’);

* $User = M(“User”);
* $count = $User->count(); //查询数据总数
* $Page = new Page($count,25); //实例化page类对象
* $show = $Page->show(); //用于返回数据分页的页脚信息

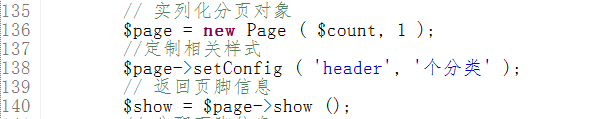
//查询当前页所有数据

* $list = $User->limit($Page->firstRow.’,’.$Page->listRows)->select();
* $this->assign('list',$list); //将当前页数据分配给模板
* $this->assign('page',$show); //将页脚信息分配给模板
* $this->display(); //显示模板



2）数据分页的相关设置

* setConfig
* header
* prev
* next
* first
* last



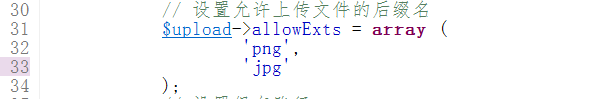


1. 文件上传
2. 实现代码

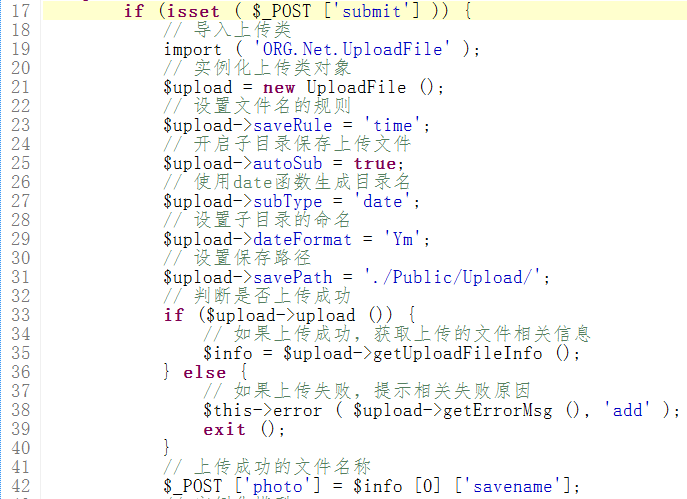
* import ( 'ORG.Net.UploadFile' ); 引用类文件
* $upload = new UploadFile (); 实例化上传类对象
* $upload->savePath = './Public/Uploads/'; 设置文件保存路径
* if (! $upload->upload ()) { 上传文件并判断上传结果
* echo $upload->getErrorMsg (); 输出错误信息
* } else { 上传成功
* $info = $upload->getUploadFileInfo ();
* }

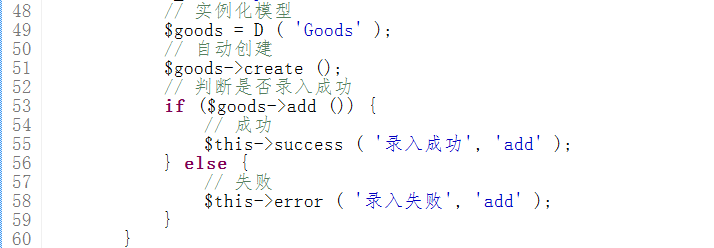
1. 上传类的相关属性

* saveRule ：文件名的保存规则
* autoSub ：是否开启子目录保存文件 true：开启
* subtype ：子目录类型
* dateFormat ：目录使用的时间格式
* allowExts ：允许的文件扩展名数组 jpg bmp png
* allowTypes ：允许的文件类型数组 image/pjpeg



1. 相关代码

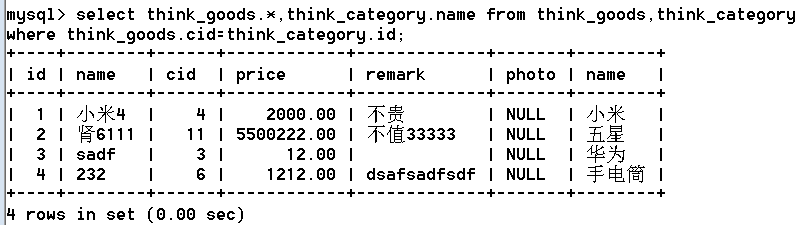




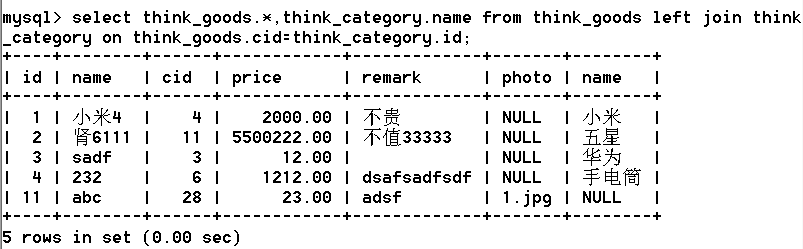
1. 多表连接

查询结果如果不在一个表中，需要使用多表连接

1. sql语句
2. select table1.\*,table2.\* from tabel1,table2 where table1.id=table.cid;



（2）select table1.\*,table2.\* from table1 left(right) join table2 on table1.id=table2.cid;



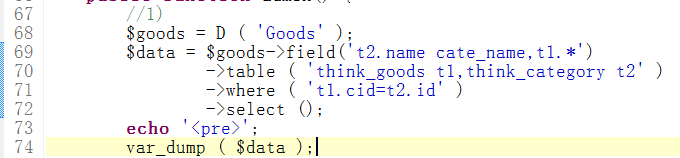
1. tp中的多表连接
2. table

语法：

$model->table(‘table1 t1,table t2’)

->where(‘t1.id=t2.sid’)

->select();

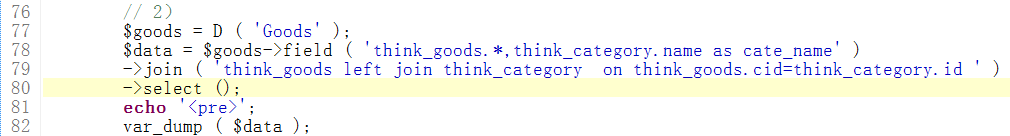


1. join

语法：

$model->join(‘left(right) join t2 on t1.sid=t2.id’)

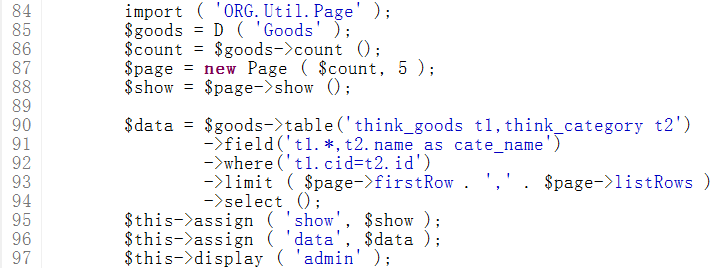
->select();



注意：left(right) join 前面可以使用表名，也可以不使用表名

如果加上表名：表示当前模型对应的表的别名

1. 加入程序



1. Ajax应用

* Action类提供了ajaxReturn方法用于AJAX调用后返回数据给客户端
* 通过配置DEFAULT\_AJAX\_RETURN进行设置，默认配置采用JSON格式返回数据

关于ajaxReturn方法：

* $this->ajaxReturn($result,“ok！",1);

data 返回数据

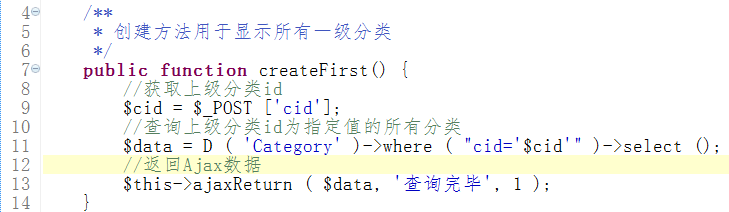
info 提示信息

status 操作状态

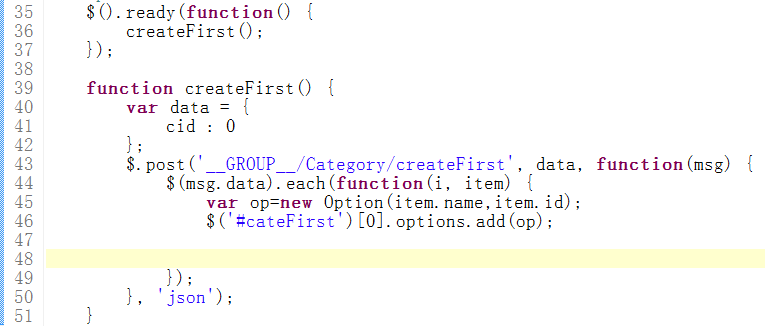
1. 配置选项：config.php



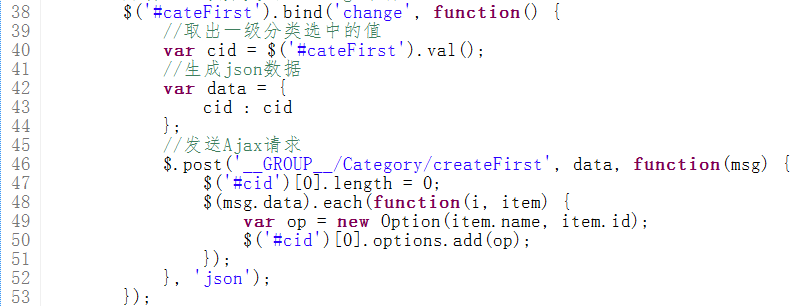
1. 在控制器创建数组，返回对应的数据



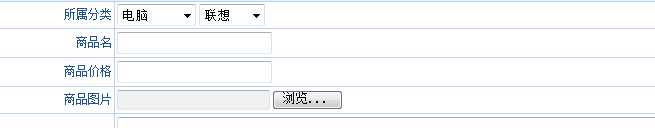
1. 创建前台页面，用于发送Ajax请求



创建二级分类：



运行结果：



1. 关联模型
2. 模型种类

用户自定义模型

基础模型

关联模型

1. 表与表之间的关系

一对一

一对多

多对多

HAS\_ONE 一对一

用户表 用户资料表

HAS\_MANY 一对多

分类表 商品表

MANY\_TO\_MANY 多对多

用户表 权限表

BELONGS\_TO 当前数据的父对象在另一个表中

商品表 分类表

文章表 用户表

1. 定义关联模型语法

如果想使用关联模型，那么用户自定义模型必须继承RelationModel

class UserModel extends RelationModel{

protected $\_link = array(

'关联1'  =>  array(

'关联属性1' =>'定义',

'关联属性N' =>'定义',

   ),

    ...

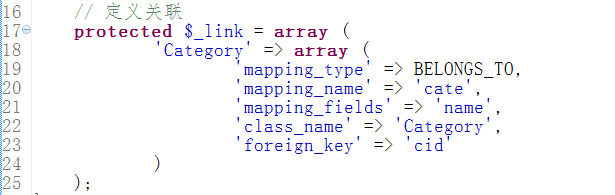
);

}

1. 支持的关联属性

* mapping\_type ：表关系
* mapping\_name ：关联映射名称（这里写什么，将就从哪个数组中去取数据）
* mapping\_fields ：关联的字段名称
* class\_name ：要关联的模型名称（如果不写，默认与关联名称表示模型连接）
* foreign\_key ：关联外键字段名称
* parent\_key ：关联外键字段名称（只适应用于自连接）
* condition ：关联条件

GoodsModel.class.php

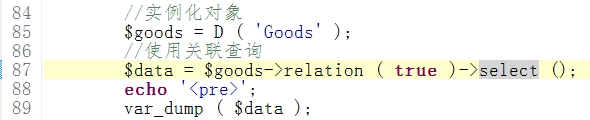


1. 使用关联模型

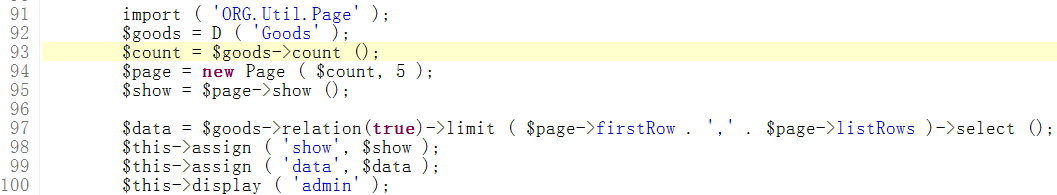
* relation(true)

$model->relation(true)->select();

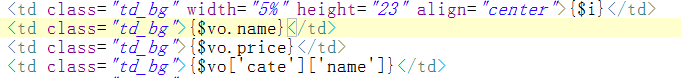
GoodsAction.class.php



更改商品分页查询



模板：

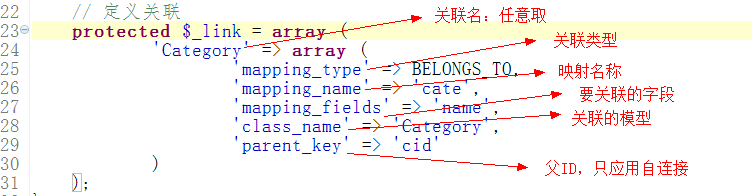


{$vo.cate.name} 相当于 {$vo[‘cate’][‘name’]}

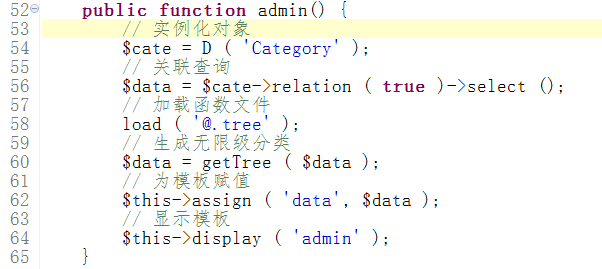
1. 关联查询

在分类管理中，查询每个分类上级分类名称

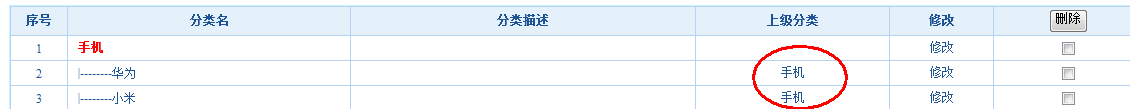
CategoryModel.class.php



CategoryAction.class.php



运行：

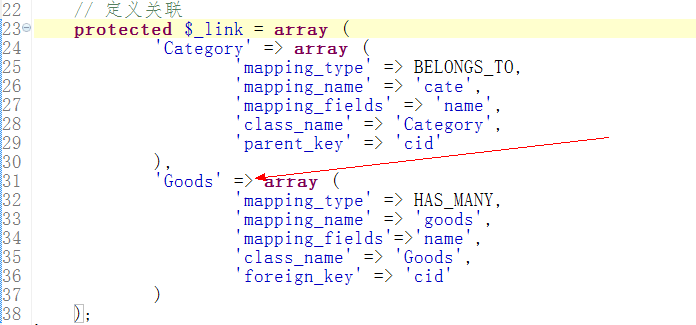


1. 关联删除

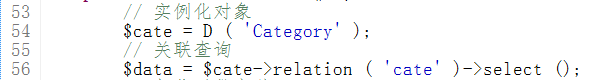
希望在删除分类时，删除对应的商品

在分类模型添加关联商品的定义

CategoryModel.class.php

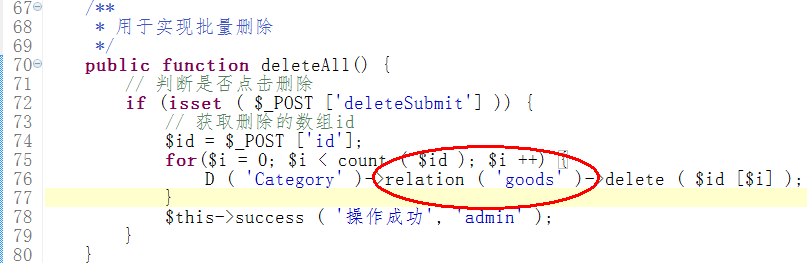


注意：之前的查询使用的是 $model->relation(true)->select()，这样一来，会将所有关联的数据一起查询出来，需要进行一改变：



只需要将参数改为需要关联的映射名即可。

CategoryAction.class.php



以上代码在删除分类时就可以直接删除对应的商品

1. 聊天室
2. 聊天室



1. 数据库

