## 传智播客TP教程笔记

使用框架的好处：简便，快捷，高效，易于维护

尚未了解的框架：symfony

zend framework:PHP语言官方的框架，功能强大，但是臃肿，启动慢

yii：重量级，美国华人（薛强）开发的框架，外企用的比较多！！！

symfony：重量级框架

laravel：轻量级框架

CI：轻量级框架，（火焰代码）

cakephp：

Think PHP :国人开发，有健全的文档，国内使用广泛！！！

MVC：是一种设计模式，强制将用户的输入、逻辑、输出相分离，将整个项目分为三部分，也就是模型，视图，控制器

在编程中所经历的编程阶段：

第一阶段：混合编程阶段

优点：效率高

缺点：不易于维护，不利于分工

第二阶段：模板引擎阶段，如smarty

优点：将前端的输出和后端的逻辑代码相分离

缺点：效率比第一种低

第三阶段：MVC设计阶段

优点：MVC的定义+易于维护

缺点：效率比前面两种都低

效率不是问题，可以通过硬件来提高，钱能解决的问题都不是问题！！！

TP框架介绍

1.简述：诞生于2006年初，2007年上线,代码中包含了丰富的中文注释！！！

2.下载：http://www.thinkphp.cn

软件版本修饰词

alpha版本：内侧版本

beta八本：公测版本，面向用户，由用户去找bug

RC版本：候选版本，功能不会有太大变化，主要是用于排错

R：发行版本，稳定版

学习版本：3.2.3

ThinkPHP源码文件夹介绍

common：系统函数库目录 functions.php

conf：系统配置文件目录

lang：语言包

Library：tp中的核心目录

Mode：

Tpl：系统模板目录，包含系统所使用的模板

ThinkPHP.php：项目接口文件，需要被项目入口文件引入

ThinkPHP

├─Application

├─Public

└─ThinkPHP

├─Common convention.php debug.php

├─Conf

├─Lang

├─Library

│ ├─Behavior 行为文件目录

│ ├─Org 功能扩展目录，tp自身开发的

│ ├─Think Library的核心目录 MVC的父类 功能类（image类，page类，upload类，verify类）

│ │ ├─Cache

│ │ │ └─Driver

│ │ ├─Db

│ │ │ └─Driver

│ │ ├─Log

│ │ │ └─Driver

│ │ ├─Storage

│ │ │ └─Driver

│ │ └─Template

│ │ ├─Driver

│ │ └─TagLib

│ └─Vendor 第三方扩展目录

├─Mode

└─Tpl

部署：

配置Apache虚拟主机

在vhosts.conf文件中：

<VirtualHost \*:80>

#站点管理员的邮箱

ServerAdmin webmaster@pxy.com

#站点根目录

DocumentRoot "D:\www\thinkphp\_3.2.3\_full"

#站点域名

ServerName www.1006.com

#站点的别名

ServerAlias 1006.com

#错误日志的存储位置，logs在Apache目录下

ErrorLog "logs/1006-error.log"

#正常访问日志的位置 common是日志的记录规则名称

CustomLog "logs/1006-access.log" common

DirectoryIndex index.php

#针对目录的详细配置

<Directory />

#+Indexes表示允许显示站点的目录结构

Options +Indexes +FollowSymLinks +ExecCGI

AllowOverride All

Order allow,deny

#允许所有访问

Allow from all

Require all granted

</Directory>

</VirtualHost>

重启Apache

直接运行drivers选择

修改hosts文件

可以把两个域名写在同一行

127.0.0.1 1006.com www.1006.com

使用ping域名，看是否指向本地ip

浏览器运行生成的文件目录APP\_PATH : Application

├─Common 应用级别的函数库文件和配置文件目录

│ ├─Common

│ └─Conf

├─Home

│ ├─Common 分组级别函数库文件目录

│ ├─Conf 分组配置文件目录

│ ├─Controlle 一下为MVC目录

│ ├─Model

│ └─View

└─Runtime 临时文件目录

├─Cache

│ └─Home

├─Data

├─Logs

│ └─Home

└─Temp

说明：只有在首次运行index.php 入口文件才会产生相应的目录结构，再次运行时不会产生的！！！

定义应用目录，生成的目录的名字取决于这里定义的常量 APP\_PATH 的值

define('APP\_PATH','./Application/');

目录安全文件:每个目录下都有空白的index.html

作用：防止用户看到网站的目录结构

由于Apache配置中会默认访问index.html

文件的生成：

首次运行生成一系列目录，是怎样生成的呢？

答:是根据系统流程生成的

---------------------------------

默认访问：

IndexController.class.php->index()

可以在核心配置文件convention.php 中看到

'DEFAULT\_MODULE' => 'Home', // 默认模块

'DEFAULT\_CONTROLLER' => 'Index', // 默认控制器名称

'DEFAULT\_ACTION' => 'index', // 默认操作名称 '

分组也叫模块，平台

---------------------------------

TP中的控制器

命名规则：AaaController.class.php

比如：GoodsController.class.php

控制器的书写：

１.声明控制器的命名空间

２.引入父类控制器

３.声明控制器，继承父类

TP中的命名空间必须要与目录挂钩

出现无法加载模块？

两种情况：

1.命名空间写错；

2.URL配置错或写错

---------------------------------

路由形式：

0.普通形式路由get形式：http://www.1006.com/index.php?m=Home&c=User&a=test

缺点：不安全，不好看

1.pathinfo形式：http://www.1006.com/index.php/Home/User/test

为什么呢？

2.rewrite路由形式（不推荐使用，因为只有Apache支持！！！）

就是URL中缺少入口文件index.php

需要设置：

一、开启重写模块150行

二、虚拟主机：设置AllowOverride all

三、将.htaccess复制到入口文件的同级目录！！！

一定要把原来的重写规则改成下面才能正常访问

RewriteEngine On

RewriteRule ^(.\*)$ index.php

3.兼容模式

路由形式，在配置文件中：

'URL\_MODEL' => 1, // URL访问模式,可选参数0、1、2、3,代表以下四种模式：

// 0 (普通模式); 1 (PATHINFO 模式); 2 (REWRITE 模式); 3 (兼容模式) 默认为PATHINFO 模式

注意：convenient.php中对路由的设置并不影响地址栏URL中输入，只是给tp自己用的！！！

分组：根据功能的使用对象来区分代码

比如Home和Admin，前台和后台两个文件夹

如何创建分组：直接复制新建的Home目录

步骤：参考Home的目录结构创建新的目录

注意：如果出现内部错误，很可能是命名空间写错了！！！

---------------------------------

URL组装

方法：如A,B,C,D,E,F,G 方法都定义在系统的函数库里，functions.php 中

U语法格式：U('url路径',参数数组,伪静态后缀,true显示域名)

返回一个拼接好的URL地址！！！

class TestController extends Controller

{

public function test()

{

echo U('test');

}

}

可以拼接从 出入口文件到（当前控制器/指定方法）的详细路径/index.php/Admin/Test/test.html

这个.html是伪静态后缀

echo U('/Think/Demo/hello');

特别注意这个URL的书写，

/hello : 表示当前控制器的hello方法，而且这个方法是自己组装的，并不一定要存在

/Demo/hello : 表示Demo控制器下的hello方法

/THink/Demo/hello : 表示Think模块(分组)下的Demo控制器下的hello方法

参数部分可以是字符串或数组

参数数组的写法：['id'=>1,'name'=>'zmh']

---------------------------------

跳转方法

1.成功跳转：$this->success()

/\*\*

\* 操作成功跳转的快捷方法

\* @access protected 方法的权限说明

\* @param string $message 提示信息

\* @param string $jumpUrl 页面跳转地址

\* @param mixed $ajax 是否为Ajax方式 当数字时指定跳转时间

\* @return void 方法的返回值说明

\*/

public function index()

{

$this->success('操作成功',U('/Home/Index/index'),5);

}

2.失败跳转：$this->error()

与成功跳转没什么区别，仅仅是图片不一样！！！

---------------------------------

视图：负责信息的输出和展示

创建位置：分组目录下View中指定的控制器同名目录中

视图的显示：$this->display();

/\*\*

\* 模板显示 调用内置的模板引擎显示方法，

\* @access protected

\* @param string $templateFile 指定要调用的模板文件

\* 默认为空 由系统自动定位模板文件

\* @param string $charset 输出编码

\* @param string $contentType 输出类型

\* @param string $content 输出内容

\* @param string $prefix 模板缓存前缀

\* @return void

\*/

public function index()

{

//调用当前控制器的index方法,如果调用当前控制器的当前方法，display的参数可以省略！！！

//$this->display('index');

//这里指向的实际上是view下面的文件，而不是控制器和方法！！！

//跨方法调用

//$this->display('test2');

//跨控制器所在文件夹调用

$this->display('Index/index');

}

$this->show() 它的内部也是调用的display方法，只是没有调用模板！！！

/\*\*

\* 输出内容文本可以包括Html 并支持内容解析

\* @access protected

\* @param string $content 输出内容

\* @param string $charset 模板输出字符集

\* @param string $contentType 输出类型

\* @param string $prefix 模板缓存前缀

\* @return mixed

\*/

---------------------------------

问题：如何实现跨分组调用？

-------------------------------

变量分配：

将控制器中变量所存储的数据分配到模板中来显示

用法：$this->assign()

参数：

/\*\*

\* 模板变量赋值

\* @access protected

\* @param mixed $name 要显示的模板变量

\* @param mixed $value 变量的值

\* @return Action

\*/

public function test2()

{

$this->assign('name','朱明浩');

$this->display();

}

模板中：{$name}

分配数组：

$data = ['title'=>'我是标题','content'=>'内容'];

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

$this->display();

标题：{$data['title']}

<br>

内容：{$data['content']}

模板中的边界限定符可以在convention.php 中设置

'TMPL\_L\_DELIM' => '{', // 模板引擎普通标签开始标记

'TMPL\_R\_DELIM' => '}', // 模板引擎普通标签结束标记

---------------------------------

模板\*\*常量\*\*替换机制

情景：在引入图片，js，css时往往要写复杂的路径

这时可以用常量，将长路径简化

在TP中，有以下常用模板常量

只能在模板中用！！！

\_\_MODULE\_\_:域名后一直到分组的路由 /index.php/Admin

\_\_CONTROLLER\_\_:域名后一直到控制器的路由 /index.php/Admin/Test

\_\_ACTION\_\_:域名后一直到方法的路由

\_\_PUBLIC\_\_:站点根目录下的public目录

\_\_SELF\_\_:域名开始一直到最后，包括参数！！！

\_\_MODULE\_\_ <br>

\_\_CONTROLLER\_\_ <br>

\_\_ACTION\_\_ <br>

\_\_PUBLIC\_\_ <br>

\_\_SELF\_\_ <br>

<!--

/index.php/Admin

/index.php/Admin/Test

/index.php/Admin/Test/test3

/Public

/admin/test/test3

-->

原理：TP中模板常量是通过模板内容替换机制实现的，可以查看ContentReplace.Behavior.class.php

可以在配置文件中定义一个自定义的模板常量，但是一定不要修改convention.php中的配置项，只需要修改分组的config.php

'TMPL\_PARSE\_STRING' => [

'\_\_ADMIN\_\_'=>\_\_ROOT\_\_.'/Public/Admin'

]

排错技巧：当代码肯定没错时，检查文件名！！！

为什么我是找不到文件路径呢？

原来是文件夹创建错了，站点根目录为index.php所在目录，Public就放在这个目录下

fetch方法：获取模板 有返回值的！！！

$tpl = $this->fetch();

fetch与display的差异：

display方法：替换模板常量->获取内容->输出内容

fetch方法：替换模板常量->获取内容 但不直接输出

TP中视图模板的注释

行注释：{//} {#}

块注释：{/\*\*/}

注释里面不能有变量

与普通HTML之间的区别：普通的HTML属于客户端的注释，会在浏览器的源代码输出，而PHP的源代码注释属于服务端的注释，不会再浏览器的源代码输出！！！

TP中模板中数组的输出支持[]和.的形式，一维二维都适用！！！

变量分配2

对象变量：可以将对象实例化后的变量分配到模板中，通过对象的属性来取得相关数据

public function test4()

{

//由于TP会到当前命名空间去找，所以，不用引入也行！！！

$stu = new Student();

print\_r($stu);

}

在入口文件index.php 添加

header('Content-Type:text/html;charset=utf-8'); 这个-不是必须的！！！

可以完美显示中文！！！

对象的分配:

$this->assign('student',$student);

$this-display();

对象在模板中取出：

箭头形式：{$stu->name}

冒号模式：{$stu:name}

注意：不允许使用 . 的形式 ！！！

出错：Call to undefined function Admin\Controller\assign()

原来是没有使用箭头 $this->assign()!!!

系统变量：

在TP中，有以下系统变量，也就是超全局变量在模板中的使用

$Think.server 等价于$\_SERVER

$Think.get 等价于$\_GET

$Think.post 等价于$\_POST

$Think.request 等价于$\_REQUEST

$Think.cookie 等价于$\_COOKIE

$Think.session 等价于$\_SESSION

$Think.config 获取TP中所有配置文件的总和，如果指定了则获取指定的配置

在模板中的用法：{$Think.xxx.aaa}

比如：{$Think.get.id}

服务器信息：{$Think.server.path} <br>

id:{$Think.get.id}<br>

request:{$Think.request.pid}<br>

cookie:{$Think.cookie.PHPSESSID}<br>

$Think.config:{$Think.config.DEFAULT\_MODULE}<br>

-------------------------------

视图中使用函数：

比如在模板中直接将时间戳格式化

语法格式：{$变量|函数名1=参数1|函数名2=参数1,参数2...}

|：变量修饰符

函数名1：表示需要使用的第一个函数

函数名2：表示需要使用的第二个函数

参数1，参数2是第二个函数的参数

### 表示变量自身

注意：当一个函数只有一个参数，并且这个参数就是本身，可以不写###

函数名必须是PHP内置的函数或者函数库文件中定义好的函数

函数名不能有括号()

案例：时间戳的格式化，经常使用!!!

public function test6()

{

$time = time();

$this->assign('time',$time);

$this->display();

}

当前时间：{$time|date='Y-m-d H:i:s',###}

案例：截取字符串前5个，并转为大写

截取字符串并转为大写：{$str|substr=###,0,5|strtoupper=###}

这里的###可以省略！

先用哪个对结果没有影响

默认值：当某个变量不存在（没有定义）或者为空的时候（空字符串），默认字符就是变量的值

使用场景：论坛的个性签名，如果用户没写，会有默认的“这个家伙很懒，什么都没留下”

语法：{$变量名|default=默认值}

其实这个default就是TP自己封装的一个函数

在PHP中也可以通过三元运算符来实现默认值：$sign = $sign ? : '这个家伙很懒！';

模板中的运算符：

+ - \* / % ++ -- 都是支持的

文件包含：

实际开发中一般会把网站的公共部分，头部和尾部单独存放到一个文件中，在后期需要的时候直接引入，方便维护

在TP中可以使用include标签俩引入一些公共部分的代码文件

语法：<include file=''/>

案例：使用include实现页面首位引入效果

<body>

<include file='./Application/Admin/View/Test/header.html'/>

<div>我有很多小花</div>

<include file='Application/Admin/View/Test/footer.html'/>

</body>

路径：可以是相对入口文件的路径

说明：上述的路径很容易写错，往往我们写成<include file="View目录下的目录名/模板文件名" />不需要文件格式

<body>

<include file='Test/header'/>

<div>我有很多小花</div>

<include file='Test/footer'/>

</body>

include还可以用来传递参数给引入的文件

<include file='' 参数名='参数值'/>

//这个引号必须要

在目标/源文件引用 [参数名]

如果不存在会原样输出！！！

循环遍历：

在TP中，有两种在模板中遍历数组的标签

1.

<volist name='需要遍历的模板变量' id='当前遍历到的元素' >

循环体

</volist>

2.

<foreach name='需要遍历的模板变量' item='当前遍历到的元素'>

循环体

</foreach>

注意：name没有$符号

区别：volist还支持mod，key，length...,可以理解成foreach（也有key）是volist的简化版本

<foreach name="list" item="vo" key="k" > {$k}|{$vo.id}:{$vo.name}</foreach>

可以看出，属性里面不加$,但是在{}中必须加$来引用以上用引号包起来的’变量‘！！！

建议使用volist，更利于扩展！！！

<body>

volist循环输出四大名著：<br>

<volist name='arr1' id='value'>

{$key}---{$value} <br>

</volist>

<hr>

foreach循环输出四大名著：<br>

<foreach name='arr1' item='value'>

{$key}---{$value} <br>

</foreach>

<hr>

volist输出四大名著所有角色：<br>

<volist name='arr2' id='value1'>

<volist name='value1' id='value2' >

{$key}---{$value2}<br>

</volist>

</volist>

<hr>

foreach输出四大名著所有角色：<br>

<foreach name='arr2' item='value1'>

<foreach name='value1' item='value2' >

{$key}---{$value}<br>

</foreach>

</foreach>

</body>

如果不嵌套，可以使用{$value1[0]}依次输出！！！

为什么输出不了下标了呢？

无解，可能就是全角半角的问题！！！

if标签

语法结构

<if condition='条件表达式'>

输出结果1

<elseif condition='条件表达式2'/>

输出结果2

<elseif condition='条件表达式3'/>

输出结果3

<else/>

最后输出结果

案例：注意条件部分，不用{},细节：elseif中间没有空格！！！

</if>

public function test10()

{//date()第二个参数默认为time()

$date = date('N',time());

$this->assign('date',$date);

$this->display();

}

<if condition="$date==3">//注意

星期三

<else/>

不是星期三

</if>

模板中的php标签

TP支持两种风格：

1.<?php PHP代码段?>

2.<php>PHP代码段</php>

不建议在HTML中混编PHP标签！！！

在配置项中禁用PHP标签TMPL\_DENY\_PHP

模型：M

数据库连接配置

/\* 数据库设置 \*/

'DB\_TYPE' => '', // 数据库类型

'DB\_HOST' => '', // 服务器地址

'DB\_NAME' => '', // 数据库名

'DB\_USER' => '', // 用户名

'DB\_PWD' => '', // 密码

'DB\_PORT' => '', // 端口

'DB\_PREFIX' => '', // 数据库表前缀

'DB\_PARAMS' => array(), // 数据库连接参数

'DB\_DEBUG' => TRUE, // 数据库调试模式 开启后可以记录SQL日志

'DB\_FIELDS\_CACHE' => true, // 启用字段缓存

'DB\_CHARSET' => 'utf8', // 数据库编码默认采用utf8

一般把数据库的配置项放到应用层级的配置文件，因为前台和后台用的都是一个数据库

数据库设计：

create database db\_oa;

use db\_oa;

create table sp\_dept(

id int not null auto\_increment,

name varchar(50) not null,

pid int not null default 0,

sort int not null default 50,

remark varchar(255),

primary key(id)

)engine=myisam default charset=utf8;

什么是模型？

作用：负责与数据表的数据的交互操作，curd

模型类的命名规则：

除去表前缀的数据表名称，采用驼峰法命名，并且首字母大写，然后加上模型层的名称（默认定义是Model）

文件名：CategoryModel.class.php

namespace Home\Model;//指明该文件的目录位置

use Think\Model;//只有use才能继承！！！

class CategoryModel extends Model

{

//protected $tableName = 'categories';

}

tablePrefix 定义模型对应数据表的前缀，如果未定义则获取配置文件中的DB\_PREFIX参数

tableName 不包含表前缀的数据表名称，一般情况下默认和模型名称相同，只有当你的表名和当前的模型类的名称不同的时候才需要定义。

trueTableName 包含前缀的数据表名称，也就是数据库中的实际表名，该名称无需设置，只有当上面的规则都不适用的情况或者特殊情况下才需要设置。

dbName 定义模型当前对应的数据库名称，只有当你当前的模型类对应的数据库名称和配置文件不同的时候才需要定义。

模型实例化1

<?php

//这里的命名空间最前面不用\!!!

namespace Admin\Model;

use Think\Model;

/\*\*

\* 数据库模型

\*/

class DeptModel extends Model

{

}

?>

控制器中：

public function shilihua()

{

//注意这里的命名空间，最前面需要\ , 是完整的MOdel名

$dept = new \Admin\Model\DeptModel();

dump($dept);

}

方法二：

使用TP提供的快速方法D和M方法

D方法是实例化我们自己指定的模型，如果没有指定，则实例化Model模型

M方法是直接实例化父类模型Model，如果指定了表名，则实例化父类模型时关联指定表，一般用于原生的sql语句

面试题：

实例化方法中M、D有什么不同？

如何选择使用哪种实例化方法呢？

答：如果需要使用的方法，父类已经封装好了，可以直接实例化父类M方法，如果父类中的方法不能满足我们的开发需要，需要自己定义，可以使用D方法实例化模型

所以，我们一般实例化父类

CURD操作：

增加操作：$model->add(一维关联数组);而且键必须和数据表中的字段名一致，如果不匹配，会被TP过滤

add()的返回值是新增主键的id

//模型实例化并增加数据

public function zengjia()

{

//$dept = new \Admin\Model\DeptModel();

$dept = M('Dept');

$data1 = [

'name'=>'组织部',

'pid'=> 0,

'sort'=>'2',

'remark'=>'这是组织部门'

];

$data2 = [

['name'=>'宣传部',

'pid'=> 0,

'sort'=>'3',

'remark'=>'这是宣传部门'

],

['name'=>'公关部',

'pid'=> 0,

'sort'=>'4',

'remark'=>'这是公关部门'

]

];

// $res = $dept->add($data1);

$res = $dept->addAll($data2);

dump($res);

}

如何增加多条记录？

1.循环使用add()

2.使用addAll(二维数组)方法，

要求：外层数组必须是下标为0开始的索引数组，内层数组必须是关联数组

特别注意：内层数组必须是一一对应！！！

修改操作：

在TP中使用$model->save(一维关联数组)方法，

条件：这里的一维数组必须要有主键信息！！！

public function xiugai()

{

$dept = M('Dept');

$data = [

'id' =>1,

'name'=>'财务部',

'sort'=>'2',

'remark'=>'今天发工资！！'

];

$res = $dept->save($data);

dump($res);

}

注意：对于更新或者修改操作的返回值，操作失败返回false，这个false又有两种情况

1.没有执行 tp中属于这种

2.更新或者修改失败

成功则返回影响的行数！！！

查询操作：

select方法语法：

$model->select();//查询全部的信息，如果有where则表示查询符合条件的信息

$model->select(id);//查询指定id

$model->select('id1,id2,id3');//查询指定id集合的信息等价于mysql中的 in[]

find语法：只能查一条记录

$model->find();//查询第一条记录

$model->find(id);//查询指定id记录

返回值：select方法始终返回二维数组

array(2) {

[0] => array(5) {

["id"] => string(1) "1"

["name"] => string(9) "财务部"

["pid"] => string(1) "0"

["sort"] => string(1) "2"

["remark"] => string(21) "今天发工资！！"

}

[1] => array(5) {

["id"] => string(2) "10"

["name"] => string(9) "公关部"

["pid"] => string(1) "0"

["sort"] => string(1) "3"

["remark"] => string(18) "这是公关部门"

}

}

find返回一维数组

array(5) {

["id"] => string(1) "2"

["name"] => string(9) "组织部"

["pid"] => string(1) "0"

["sort"] => string(1) "2"

["remark"] => string(18) "这是组织部门"

}

删除操作：

在TP中，可以使用$model->delte()

$model->delete(id);

$model->delete('id1,id2,id3');

返回值是受影响的行数！！！

物理删除：真的删除

逻辑删除：假删除，本质是修改操作，在数据表中定义一个状态字段，0或者1，查询时读取状态为1的，当点击删除时，触发修改状体为0，所以就不会显示在页面上。

-------------------------------

开发实用项：

用于调错：比如浏览器自带的审核，在TP中，可以使用一系列调试方法

1.跟踪信息：

作用：展示系统执行 的相关状况

开启：SHOW\_PAGE\_TRACE=>true

需要用户在自己的配置文件中开启

也就是TP中还有一些配置项中在系统的配置文件是没有而实际上又是支持的！！！

2.两种模式：开发模式，生产模式

2.1开发/调试模式:开发阶段使用的模式，错误信息详细，暴露全部的错误

2.2生产模式：项目上线后所使用的模式，会屏蔽很多错误，错误信息比较模糊

在入口文件中开启：APP\_DEBUG=>true/false

3.sql调试：

$model->getLastSql()

获取当前模型中最后一条成功执行的sql语句。

注意：有一个别名$model->\_sql();

SHOW COLUMNS FROM `sp\_dept`相当于 desc sp\_dept

性能测试：

TP提供了一个性能测试的快速方法G方法

用法：G('开始标记')

需要统计性能的代码段

G('结束标记')

G('开始标记','结束标记',数字/字符m)

第三个参数：数字则表示统计代码的执行事件，数字表示精确的时间位数（s），m则表示统计内存的开销，注意，这里需要服务器的支持！！！

案例：使用G方法来统计某段代码的执行时间开销

public function test2()

{

G('start');

for($i=0;$i<1000000;$i++){

echo $i;

}

echo "<br>";

G('end');

echo G('start','end',4);

}

AR模式：对象关系映射

类Model 模型类$model AR类属性 记录

对应关系： 表 字段 操作结果

模型类是类的实例结果，一个模型类就对应数据库的一张表，模型的属性映射到表的字段，模型的每一个操作映射到表的记录

AR模式在TP中的CURD操作

1.增加操作：

//AR模式增加操作

public function test12()

{

//第一个映射，类映射到表

$model = M('Dept');

// 第二个映射，属性到字段

$model->name = '技术部';

$model->pid = 0;

$model->sort = 10;

$model->remark = '技术部最重要';

// 第三个映射，实例映射到记录

$res = $model->add();//没有参数

dump($res);

}

这是如何实现的呢？

在Model类中使用了魔术方法\_\_set($name,$value)动态添加数据

追踪Model里的add方法，我们发现，当add方法没有传参时，会判断$this->data里面是否有数据，如果有，就把这个数据赋给 $data,$data 为一个数组，然后拼接sql语句，插入数据到数据表

2.更新操作

注意：一定要指定id

$model->id = 2;

$model->remark = '财务部最新公告';

$model->save();

/ AR模式更新操作

public function test13()

{

$model = M('Dept');

$model->id = 2;

$model->remark = '组织部最新公告';

$res = $model->save();

dump($res);

}

3.查询操作：

在TP的AR模式中没有查询操作，所以还是按照以前的select和find方法

4.删除操作：

注意：必须指定主键信息，比如id

// AR模式删除操作

public function test14()

{

$model = M('Dept');

//$model->id = '1,2,3;

$model->id = 2;

$res = $model->delete();

dump($res);

}

总结：AR模型操作的add，save，delete方法都是没有参数的，因为在$model 中已经指定了

注意：如果在更新或者删除之前有使用find方法执行过查询，可以不指定id，也可以进行操作，TP会自动获取上面查询所用到的id！！！所以主键一定要写上，避免误操作！！！

辅助方法：

就像原生的group、where、order、limit...

在TP中，可以使用group、where、order、limit这些方法

where：限制的条件

limit：限制输出的条数

field：限制输出的字段 比如select id,name from tab;

order:按照指定的字段进行指定的排序

group：按照指定字段进行指定分组

where方法：

$model->where(条件表达式);//TP中支持字符串和数组形式

$model->CURD操作

编程判断技巧：首先判断最坏的情况！！！

where和having的区别：

where是从物理表中直接判断，having是对结果集进行过滤！！！

limit方法：典型应用分页

原生的：limit $start $size

TP中，$model->limit(n);下标从0开始

$model->limit($start,$size);

$Model->where('status=1')->order('id desc')->limit(5)->select();

field方法指定字段

在查询操作中field方法是使用最频繁的。

$Model->field('id,title,content')->select();

可以给某个字段设置别名，例如：

$Model->field('id,nickname as name')->select();

可以在field方法中直接使用函数，例如：

$Model->field('id,SUM(score)')->select();

order方法：按照指定字段、规则排序

$model->order('字段 desc/asc');

group方法：

作用：在分组查询中使用

$model->group('字段名');

// group方法

public function test18()

{

$model = M('Dept');

$model->field('name,count(\*)');

$model->group('name');

//与顺序无关

$res = $model->select();

dump($res);

}

以上的order和group方法在Model中是不存在的，在Model中是利用了PHP的一个魔术方法\_\_call($method,$args);当调用一个不存在的方法，此方法自动执行！！！

Model中：

/\*\*

\* 利用\_\_call方法实现一些特殊的Model方法

\* @access public

\* @param string $method 方法名称

\* @param array $args 调用参数

\* @return mixed

\*/

public function \_\_call($method,$args)

连贯操作：

将辅助方法全部写在一行上的写法。

$User->where('type=1 AND status=1')->select();

$Model->table('db\_name.think\_user')->where('status>1')->select();

select方法前的都是辅助方法,辅助方法的顺序随意！！！

只要模型在前，CURD在后即可！！！

之所以可以支持连贯操作，是因为所有的辅助方法最终返回的都是Model这个对象！！！return $this

PHP\_OS为PHP保留字段

注意：我们大多数问题都能够在手册上查找到答案！！！

THINKPHP 中的统计查询

count() 查询总的记录数

max() 查询字段的最大值

min() 最小值

avg() 查询字段平均值

sum() 查询某个字段的总和

也上方法都是有模型封装的，也是特殊的，有\_\_CALL自动生成的

// count() max() min()方法的使用

public function test19()

{

$model = M('Dept');

// $res = $model->count();//返回字符串

// $res = $model->max('id');//返回字符串

// $res = $model->min('id');

// $res = $model->avg('sort');

$res = $model->sum('sort');

dump($res);

}

执行流程所要加载的文件

D:\www\thinkphp\_3.2.3\_full\index.php ( 1.01 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\ThinkPHP.php ( 4.71 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Think.class.php ( 12.32 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Storage.class.php ( 1.38 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Storage\Driver\File.class.php ( 3.56 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Mode\common.php ( 2.82 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Common\functions.php ( 52.60 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Hook.class.php ( 4.02 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\App.class.php ( 12.44 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Dispatcher.class.php ( 15.15 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Route.class.php ( 13.38 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Controller.class.php ( 10.95 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\View.class.php ( 7.96 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Behavior\BuildLiteBehavior.class.php ( 3.69 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Behavior\ParseTemplateBehavior.class.php ( 3.89 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Behavior\ContentReplaceBehavior.class.php ( 1.93 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Conf\convention.php ( 11.18 KB )

D:\www\thinkphp\_3.2.3\_full\Application\Common\Conf\config.php ( 0.50 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Lang\zh-cn.php ( 2.57 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Conf\debug.php ( 1.51 KB )

D:\www\thinkphp\_3.2.3\_full\Application\Admin\Conf\config.php ( 0.11 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Behavior\ReadHtmlCacheBehavior.class.php ( 5.62 KB )

D:\www\thinkphp\_3.2.3\_full\Application\Admin\Controller\TestController.class.php ( 4.31 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Model.class.php ( 67.27 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Db.class.php ( 5.70 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Db\Driver\Mysql.class.php ( 8.73 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Think\Db\Driver.class.php ( 41.60 KB )

D:\www\thinkphp\_3.2.3\_full\ThinkPHP\Library\Behavior\ShowPageTraceBehavior.class.php ( 5.27 KB )

fetSql的使用：

可以放在$model与CURD语句之间，所有的sql语句都会被打印出来，并且不会真正执行，可以很方便看到TP拼接好的sql语句，利于排错，之前的getLastSql只能打印上一次成功执行的sql语句！！！

public function test20()

{

$model = M('Dept');

$data = $model->field('name,count(\*)')->group('name')->fetchSql(true)->select();

dump($data);

}

执行结果

string(52) "SELECT `name`,count(\*) FROM `sp\_dept` GROUP BY name "

第二天案例：

需要特别注意：模板常量只能在第一级模板中使用,所以如果需要使用模板常量，必须放在View文件夹下，并且有相应的控制器文件夹方法模板文件！！！

在模板中也可以使用U方法来指定URL地址

注意： 需要在外面加上 {:U('home')}

常见的左侧菜单列表

dd>(div>img+span)+(ul>li\*2>a)

图标加文字功能菜单效果

ul>li\*5>(span>img)

所有的样式都用css来设置，HTML只管内容！！！