第 39 章 ThinkPHP--为什么要使用框架

学习要点：

1.框架和库

2.个人用框架的优势

3.公司用框架的优势

4.框架和开源系统

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

在正式学习 ThinkPHP 框架之前，我们首先要探讨一个问题——为什么要使用框架。它的好处是什么？有哪些优势？对个人或公司运营有哪些帮助？

一．框架和库

库：英文是（Library）可以理解为仓库、图书馆、存书室。在程序里面理解，就是各种代码的封装集合包，提供原生内置并不存在的功能和方法。

框架：英文是（Framework）可以理解为骨架、架构、书架、书框。在程序里面理解，就各种条条框框像个有规则的书架一样，让你按照它的代码规范、编码模式（比如很多框架是基于 MVC 模式的）去开发应用程序。当然，框架内可能也包含了库的功能，也提供了各种封装功能的集合包。

那么使用框架开发有哪些优势呢？

二．个人用框架的优势

在个人项目开发中，使用框架比采用原生代码开发优势较为明显：

1.效率极高：因为框架已经帮你疏通了管道、构建了厂房、安好了水电、装全了设备、排好了书架，你要做的就是按照书架的布局或规格，整齐的把书放上，就完了。如果你不用框架建设，你还需要自己疏通管道、构建厂房、安水电、装设备、排书架，效率大大提高。

2.安全性高：框架的开发难度较高，一般都是身经百战的架构高手们经过多年、多版本研发、更新迭代孕育而成。经过大量市场和用户的轮番测试和考验，保证了不同情况和状况的稳定性。而如果你是低手或者新手，且自己架构，很有可能会遇到厂房崩塌、漏水漏电等不安全的情况发生。

3.稳定性高：基本同上，因为各种部件进行有有条不紊的架构，导致在不同程度的运行中保证流畅。而自行架构的，在某些极端情况下可能会遇到管道堵塞的问题发生。

PS：使用框架有这么多明显的优势，那对于个人开发者有必要学习那些原生的基础知识吗？答案是：必须要学习。首先，最基础的没有学好，在学习和使用框架的时候会到处碰壁。其次，基础知识越扎实，在使用框架的深度和灵活度上要远高于不扎实的人，这是个人竞争力的体现。

PS：一般对于职业生涯规划，我总结出可能会出现三种情况：1.努力学习基础知识，然后学习一种或多种框架在工作中高效的使用，最终积累的各种资源，成立自己的工作室或公司承办业务；2.努力学习基础知识，然后学习一种或多种框架在工作中高效使用，然后又返璞归真深入研究原生，开发多款工具或开源框架，实现大牛级别；3.改行。

三．公司用框架的优势

在公司团队项目开发中，使用主流框架比原生或自行开发框架优势较为明显：

1.团队协作力强：开源框架提供了统一的规范和编码模式，且模块与模块之间相对独立。对于程序员来说，只要根据这种规范来编码，团队之间的协调性会非常容易。而且主流的开源框架在招募人才方面较为容易，上来就能使用，降低的培训成本。

2.降低培训成本：如果使用原生代码，当另一个团队人员接受或参与，几乎看不到懂之前的技术员人代码，最终可能要重构，成本巨大。如果使用公司研发的框架，那么也需要一定时间的培训才能上手，成本也是巨大的。而使用主流框架，大部分技术人员在来公司之前就已经掌握。

3.去技术人员化：一个公司把核心代码掌握在某个技术人员手里，这是极其危险的事。当这个项目的原生或者自行开发的框架是某一个技术人员设计的，当他离职后，代码将是一堆垃圾，没人能看懂。如果使用主流框架，那么公司的核心是项目和业务本身，降低技术人员的价值。任何一个技术人员离职，都不会影响项目的研发进度，只要懂主流框架的技术人员，上手即可继续。

PS：这里要声明一下，不同级别的公司在原生和自行开发框架、主流开源框架的认识上是有一定分歧的。技术型的大公司，不缺线，不缺技术人员，开发自己的框架主要解决自己庞大的业务需求，其次是捍卫自己技术型公司的地位。而缺钱、缺人才的中小型公司，什么都要自己研发，自己研发框架，自己研发前端库，然后用框架和库开发自己的论坛，内容管理系统，商城系统。不用 discuz、dede、ecshop 之类的开源系统。最终导致烧完 100 万资金，还欠发两个月工资，老板失踪，网站也没上线运营就倒闭了。

四．框架和开源系统

所谓开源系统，就是开放源代码的系统。这些系统，包括比如 discuz 论坛系统、phpcms 内容管理系统、shopex 商城系统。这些系统，都是定制某种领域功能的、已经开发好的系统。原则上，这些系统已经不需要技术人员来编码了，可以直接上线运营了。因为这些是定制的系统，所以系统会专门根据此领域的情况进行特定的优化。那么也就是说，这些开源系统在安全性、稳定性和效率上要高于用框架开发的同类产品。所以，很多公司大量招聘开源系统的二次开发人才。

那么到底是学框架开发，还是学开源系统的二次开发呢？我的建议是，先学基础（比如 PHP1,2,3 季度），再学框架（PHP 第 4 季）。之后如果对开源系统的二次开发感兴趣，其实有了这么多基础，直接看 API 就非常容易了，都不需要专门做视频指引。

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--安装与配置

学习要点：

1.获取 ThinkPHP
2.入口文件
3.自动生成
4.访问控制器

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

ThinkPHP 是一款免费开源的框架，基于 MVC 设计模式和面向对象开发。

一．获取 ThinkPHP

我们直接登录 ThinkPHP 的官网下载频道：http://www.thinkphp.cn/down.html，选择最新的 ThinkPHP3.2.1 正式完整版即可。

由于最新版本采用了命名空间等新特性，所以框架对于 PHP 版本的最低要求是 5.3+。其他需求一般都支持，我们直接使用 wamp 即可满足所有要求。

解压 ThinkPHP3.2.1，打开它或导入到项目中去，展开六个文件及文件夹：

Application	--应用程序目录，当程序开发时自动生成，默认为空；
Public	--公共资源文件目录，存放一些公用的文件，默认为空；
ThinkPHP	--框架目录，框架的核心架构程序包；
README.md	--说明文件，可删；
.htaccess	--配置文件，一般用于配置伪静态；
Index.php	--入口文件，所有程序都通过这里访问。

对于 ThinkPHP 框架目录内，也含有大量的目录及文件：

Common	--核心公共函数目录
Conf	--核心配置目录
Lang	--核心语言包目录
Library	--框架类库目录
|--Think	--核心 Think 类库包目录
|--Behavior	--行为类库目录
|--Org	--Org 类库包目录
|--Vendor	--第三方类库目录
|--...	--更多类库目录
Mode	--框架应用模式目录
Tpl	--系统模版目录
LICENSE.txt	--框架授权协议文件
logo.png	--框架 LOGO 文件
README.txt	--框架 README 文件

index.php --框架入口文件

二．入口文件

ThinkPHP 采用单一入口模式对项目进行部署和访问的，所以我们需要通过 index.php 进行一些部署工作，保证其正确访问。

1.将完整版压缩包解压的内容，拷贝到指定的服务器文件夹内，比如 demo39;

2.打开 ThinkPHP 提供的 index.php 文件，我们发现如下代码：

```
//检测PHP环境
if(version_compare(PHP_VERSION,'5.3.0','<'))  die('require PHP > 5.3.0 !');

//开启调试模式 建议开发阶段开启 部署阶段注释或者设为false
define('APP_DEBUG',True);

//定义应用目录
define('APP_PATH','./Application/');

//引入ThinkPHP入口文件
require './ThinkPHP/ThinkPHP.php';
```

3.如果想按照自己的意愿设置应用目录名称，可以修改成这样：

```
//修改应用目录
define('APP_PATH','./Weibo/');
//修改框架目录
require './Think/ThinkPHP.php';
```

当第一次运行了这个配置后的应用程序，将会在根目录生成一个 Weibo 文件夹，应用程序的所有文件将全部存放在这里。

三．自动生成

当第一次访问应用入口文件的时候，会自动生成 Weibo 这个应用程序目录。里面包含了各种目录，说明如下：

```
Common          --应用公共模块
    |--Common    --应用公共函数目录
    |--Conf      --应用公共配置文件目录
Home            --默认生成的 Home 模块
    |--Conf      --模块配置文件目录
    |--Common    --模块函数公共目录
    |--Controller --模块控制器目录
    |--Model     --模块模型目录
    |--View      --模块视图文件目录
Runtime         --运行时目录
    |--Cache     --模版缓存目录
    |--Data      --数据目录
```

北风网 IBEIFENG.COM
Ycku.com 瓢城Web俱乐部，做最酷的教学

```
|--Logs         --日志目录
|--Temp         --缓存目录
```

index.php 只有一个入口，就是应用程序入口。如果有多个站，或者后台，那就需要另一个入口。

创建一个 admin.php，应用目录改成对应的即可。

```
//修改应用目录
define('APP_PATH','./Admin/');
```

在自动生成的目录中，为了防止访问到应用程序的目录结构，会创建个index.html文件。当然，你也可以自行设置。

```
//设定目录生成的文件
define('DIR_SECURE_FILENAME', 'default.html');

//设置目录页面内容
define('DIR_SECURE_CONTENT', '目录禁止');
```

一般来说，第一次生成应用程序，应该加上静态主页防止目录结构暴露。但如果你的环境非常安全，可以关闭生成静态主页。

```
//禁止目录主页生成
define('BUILD_DIR_SECURE', false);
```

四．访问控制器

控制器路径在：Weibo/Home/Controller 下，有一个默认的控制器 IndexController.class.php 文件。

控制器类的命名方式：控制器名（驼峰式，首字母大写）+Controller

控制器文件的命名方式：类名+class.php

创建一个控制器需要三个部分：1.设置命名空间；2.导入命名空间；3.控制器类

```
//设置命名空间
namespace Home\Controller;                  //设置命名空间，就是当前目录

//导入命名空间
use Think\Controller;                       //继承父类用到Controller类

//控制器类
class IndexController extends Controller {
    public function index() {
      //...
    }
}
```

除了首页直接访问：http://localhost/demo39/，如果想用完整形式则是：http://localhost/demo39/index.php/Home/Index/index。

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

在这里的完整URL中，index.php是单一入口文件，Home是主模块，Index是控制器名，index是控制器里的一个方法。注意：这里大小写区分，因为在Linux是区分大小写的。

如果创建一个test()方法，那么URL就是：

http://localhost/demo39/index.php/Home/Index/test

如果想创建一个User模块，那么可以创建一个User控制器。

```
namespace Home\Controller;
use Think\Controller;

class UserController extends Controller {
    public function index() {
        echo 'user';
    }
}
```

URL访问路径为：http://localhost/demo39/index.php/Home/User/index

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--模块化和 URL 模式

学习要点：

1.模块化设计

2.URL 模式

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们主要探讨两个问题。一个是 ThinkPHP3.2.1 的模块化设计，可以支持多模块的应用创建；第二个就是 URL 模式，提供了多种 URL 显示方式。

一．模块化设计

在上一节，我们直接复制 index.php 创建了 admin.php 文件。这样会自动生成两个目录，分别为 Weibo 和 Admin，前者代表前台，后者代表后台。那么现在采用 ThinkPHP 模块化架构思想，可以把 Weibo 内的 Home 目录当作前台、Weibo/Admin 目录当作后台。

操作方法：把 Home 目录复制一份到同级目录，改名为 Admin。把 Admin 中的 Controller 内的 IndexController.class.php 的命名空间改为如下：

```
//修改Admin模块的命名空间
namespace Admin\Controller;
```

最终访问的地址就是：http://localhost/demo39/index.php/Admin

有一些模块我们希望是被用户禁止访问的，比如 Common 和 Runtime 模块。当然，框架已经在默认就禁止访问了。

当强行访问 Common 模块的时候，会提示：“无法加载模块:Common”的错误信息。

```
//禁止访问模块
'MODULE_DENY_LIST'      =>  array('Common','Runtime'),
```

PS：当你去掉数组里的'Common'，那么会提示：“无法加载控制器:Index”的错误信息。说明这个模块已经可以访问了。

```
//禁止访问模块，添加一个Admin模块
'MODULE_DENY_LIST'      =>  array('Common','Runtime','Admin'),
```

有禁止访问模块的设置，就有允许访问模块的设置。当设置了此选项，就务必把所有允许访问的模块都添加上，否则会变成拒绝访问。

```
//允许访问的模块，设置了，就必须写全，漏写的将无法访问
'MODULE_ALLOW_LIST'    =>    array('Home','Admin'),
```

如果有多个访问模块，那么在默认 URL 访问的时候，应该有一个首选访问。默认是 Home，想设置 Admin 为默认，可以这么设置：

```
//设置默认起始模块
'DEFAULT_MODULE'       =>    'Admin',
```

应用项目如果只允许单个模块的话，可以设置拒绝多个模块，这样创建更多的模块将失效。

```
//单模块设置
'MULTI_MODULE'          =>  false,
```

有时，你会觉得 index.php/Admin 这样很麻烦。你可能想直接 admin.php 就代表后台就方便很多，那么可以使用多入口设置。

这里的多入口和上一节多个应用项目不同，而是通过 admin.php 访问 Weibo 目录下的 Admin 模块。将 index.php 复制出来改成 admin.php，然后添加如下代码：

```
//默认指向Admin模块
$_GET['m'] = 'Admin';
//默认指向Index控制器
$_GET['c'] = 'Index';
```

二．URL 模式

ThinkPHP的URL模式有四种，默认是PATHINFO模式，其他三种分别为：普通模式、REWRITE和兼容模式。

```
http://localhost/demo39/index.php/模块/控制器/操作
```

```
//用户密码传参
class UserController extends Controller {
    public function test($user, $pass) {
        echo 'user:'.$user.'<br />pass:'.$pass;
    }
}
```

```
//PATHINFO模式
http://localhost/demo39/index.php/Home/User/test/user/Lee/pass/123
```

在这条 URL 上，Home 表示模块，User 表示控制器，test 表示方法，user/Lee 表示第一个键值对，pass/123 表示第二个键值对。

PATHINFO 模式下默认的分隔符是/，我们可以设置为你想要的，比如：_

```
//设置键值对分隔符
'URL_PATHINFO_DEPR'=>'_',
```

设置了分隔符的 URL：

```
http://localhost/demo39/index.php/Home_User_test_user_Lee_pass_123
```

```
//普通模式
http://localhost/demo39/index.php?m=Home&c=User&a=test&user=Lee&pass=123
```

在这条 URL 上，我们发现采用的就是传统的 GET 模式，m 表示模块，c 表示控制器，a 表示方法，后面的表示键值对。

普通模式的 m、c、a 可以自行设置为你习惯的键名称：

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
    //修改键名称
    'VAR_MODULE'            =>  'mm',
    'VAR_CONTROLLER'        =>  'cc',
    'VAR_ACTION'            =>  'aa',
```

http://localhost/demo39/index.php?mm=Home&cc=User&aa=test&user=Lee&pass=123

```
    //REWRITE模式（重写模式）
```

httpd.conf 配置文件中加载了 mod_rewrite.so 模块

AllowOverride None 将 None 改为 All

把下面的内容保存为.htaccess 文件放到应用入口文件的同级目录下

这样，ThinkPHP 自带的.htaccess 文件就起作用了，可以过滤掉 index.php 这个字符串。

```
    <IfModule mod_rewrite.c>
      Options +FollowSymlinks
      RewriteEngine On

      RewriteCond %{REQUEST_FILENAME} !-d
      RewriteCond %{REQUEST_FILENAME} !-f
      RewriteRule ^(.*)$ index.php/$1 [QSA,PT,L]
    </IfModule>

    //去除了index.php
    http://localhost/demo39/Home/User/test/user/Lee/pass/123
```

PS：兼容模式一般用于不支持 PATHINFO 的特殊环境，基本上用不到。

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--模型初步

学习要点：

1.创建数据库

2.实例化模型

3.字段定义

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们重点了解一下 ThinkPHP 模型操作部分。模型是 MVC 中的 M，可以理解为操作数据库部分操作。

一. 创建数据库

在使用模型操作之前，我们首先创建一个数据库：thinkphp。创建一个用户表：user。添加一些数据即可。

ThinkPHP 内置了抽象数据库访问层，把不同的数据库操作封装起来。我们只需要使用公共的 Db 类进行操作，无须针对不同的数据库写不同的代码和底层实现。Db 类会自动调用相应的数据库驱动来处理。

目前支持的数据库包括 Mysql(包含 mysql 和 mysqli)、SqlServer、PgSQL、Sqlite、Oracle、Ibase、Mongo、PDO 等。

支持的数据库设置

DB_TYPE 设置	支持的数据库类型
mysql 或 mysqli	mysql
pgsql	pgsql
sqlite	sqlite
mssql 或 sqlsrv	sqlserver
oracle	oracle
ibase	ibase
mongo	mongo
PDO	PDO 支持的所有数据库

```
//全局配置定义
'DB_TYPE'=>'mysql',          //数据库类型
'DB_HOST'=>'localhost',      //服务器地址
'DB_NAME'=>'thinkphp',       //数据库名
'DB_USER'=>'root',           //用户名
'DB_PWD'=>'123456',          //密码
'DB_PORT'=>3306,             //端口
'DB_PREFIX'=>'think_',       //数据库表前缀
```

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

PS：数据库全局配置信息除了 PDO，均可以采用上面的设置。

```
//PDO专用定义
'DB_TYPE'=>'pdo',          //数据库类型
'DB_USER'=>'root',         //用户名
'DB_PWD'=>'123456',        //密码
'DB_PREFIX'=>'think_',     //数据库表前缀
'DB_DSN'=>'mysql:host=localhost;dbname=thinkphp;charset=UTF8',
```

二．实例化模型

连接上数据库后，我们需要从数据库里操作数据，那么就需要实例化模型类。在 ThinkPHP 中，提供了 Model 基类处理，也可以使用 M()方法。

```
//实例化Model类，传一个数据表名
$user = new Model('User');
//显示变量结构
var_dump($user);
```

Model 基类可以传递三个参数：

```
Model(['模型名'],['数据表前缀'],['数据库连接信息']);
//实例化Model类，改变表前缀
$user = new Model('User','tp_');
//实例化Model类，定义数据库链接信息
$user =
new Model('User','think_','mysql://root:123456@localhost/thinkphp');

//打印出所有数据
var_dump($user->select());
```

使用 Model 基类还需要导入命名空间，而使用 M()方法，则不需要。

```
//实例化Model类
$user = M('User');
```

除了使用 Model 基类和 M()方法，还有一种对应数据表的模型定义，比如：UserModel。这种模型类并非必须定义的，只有当存在独立的业务逻辑或者属性的时候才需要。

```
//User模型类
namespace Home\Model;
use Think\Model;
class UserModel extends Model {}
```

创建了 UserModel 模型类后，控制器那头就可以直接声明。

```
//User模型类
$user = new UserModel();
var_dump($user->select());
```

为什么 UserModel 模型类没有指定任何表即可直接访问呢？因为这种模型类基本是直接操作数据表的，所以在命名规范上和数据表名是对应的。

模型类与数据表对应规范

模型名	对应的数据表(假设前缀是 think_)
UserModel	think_user
UserTypeModel	think_user_type

虽然使用模型类和数据表对应较为方便，但当有时我们需要更换表名、前缀、附加数据库名等，就需要一些字段定义的操作。为了更加方便的了解数据表的变化，我们使用一下页面 Trace 工具，可以时时的查询 SQL 的变化。

```
//页面Trace，调试辅助工具
'SHOW_PAGE_TRACE' =>true,
```

数据表定义

字段属性	说明
tablePrefix	定义模型对应数据表的前缀
tableName	不包含表前缀的数据表名称
trueTableName	包含表前缀的数据表名称
dbName	定义模型当前对应的数据库名称

```
//重新定义表前缀
class UserModel extends Model {
    protected $tablePrefix = 'abc_';
}

//重新定义表名
class UserModel extends Model {
    protected $tableName = 'abc';
}

//重新定义完整的带前缀的表名
class UserModel extends Model {
    protected $trueTableName = 'tp_abc';
}

//附加数据库名
class UserModel extends Model {
    protected $dbName = 'tp';
}
```

如果你仅仅使用 CURD 等数据库基本操作，我们建议使用基于 Model 基类的 M()方法。使用 M()方法由于不需要加载具体的模型类（比如 UserModel 类），所以性能会更高。

当然，如果有必要使用具体的模型类时，ThinkPHP 还提供了 D()方法来直接是实例化模型类，并且还可以免去引入命名空间等操作。

```
//实例化UserModel类
$user = D('User');
```

PS：使用 D()方法比直接使用模型类更加的智能，如果在\Home\Model\UserModel 找不到该模型类，那么就会去公共模块下找\Common\Model\UserModel 去找。如果还找不到，就会直接实例化基类 Model()类，也就是等同于使用 M()方法。

D()方法可以直接调用当前模块的模型类，那么如果跨模块调用的话，那怎么处理呢？比如 Admin 后台模块，可以使用目录声明。

```
//跨模块实例化
$user = D('Admin/User');
```

有时，你可能想使用原生的 SQL 语句进行操作数据库。那么可以采用实例化空模型基类或者空 M()方法。

```
//空 M()方法
$user = M();                              //或者new Model();空基类
var_dump($user->query("SELECT * FROM think_user WHERE user='蜡笔小新'"));
```

三．字段定义

每个模型类操作着每个对应的数据表，在大多数情况下，系统会自动获取当前数据表的字段信息。而当模型类第一次实例化时，系统会自动缓存字段，并且永久缓存，除非删除了运行时缓存或者设置不缓存。

如果调试模式下，则不会生成字段缓存文件，每次都是从数据表里重新获取。生成缓存的目的显而易见，就是为了快速响应。ThinkPHP 默认是开启字段缓存，因为在实际运行中，不会更改字段结构。

字段缓存文件保存在 Runtime/Data/_fields/目录里，当你在开发阶段，字段和表会经常变动，所以要关闭缓存。关闭缓存的方法为：

```
// 关闭字段缓存
'DB_FIELDS_CACHE'=>false                  //开启了调试模式，自动关闭
```

PS：如果开启缓存状态，新增了字段，那么可能新字段无法刷新出来，必须删除/Data/_fields 文件夹，重新获取字段。

```
//查看字段结构
var_dump($user->getDbFields());
```

你也可以使用手动定义数据表字段的方式取代字段缓存方式，这种方式可以提高性能，避免 IO 开销。

```php
//手动定义数据表字段，_pk表示主键
class UserModel extends Model {
    protected $fields = array('id', 'user', '_pk'=>'id');
}

//type 定义每个字段的类型，可以永远字段验证
class UserModel extends Model {
    protected $fields = array('id', 'user', '_pk'=>'id',
            'type'=>array('id'=>'smallint','user'=>'varchar'));
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--模型初步

学习要点：

1.创建数据库

2.实例化模型

3.字段定义

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们重点了解一下 ThinkPHP 模型操作部分。模型是 MVC 中的 M，可以理解为操作数据库部分操作。

一. 创建数据库

在使用模型操作之前，我们首先创建一个数据库：thinkphp。创建一个用户表：user。添加一些数据即可。

ThinkPHP 内置了抽象数据库访问层，把不同的数据库操作封装起来。我们只需要使用公共的 Db 类进行操作，无须针对不同的数据库写不同的代码和底层实现。Db 类会自动调用相应的数据库驱动来处理。

目前支持的数据库包括 Mysql(包含 mysql 和 mysqli)、SqlServer、PgSQL、Sqlite、Oracle、Ibase、Mongo、PDO 等。

支持的数据库设置

DB_TYPE 设置	支持的数据库类型
mysql 或 mysqli	mysql
pgsql	pgsql
sqlite	sqlite
mssql 或 sqlsrv	sqlserver
oracle	oracle
ibase	ibase
mongo	mongo
PDO	PDO 支持的所有数据库

```
//全局配置定义
'DB_TYPE'=>'mysql',          //数据库类型
'DB_HOST'=>'localhost',      //服务器地址
'DB_NAME'=>'thinkphp',       //数据库名
'DB_USER'=>'root',           //用户名
'DB_PWD'=>'123456',          //密码
'DB_PORT'=>3306,             //端口
'DB_PREFIX'=>'think_',       //数据库表前缀
```

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

PS：数据库全局配置信息除了 PDO，均可以采用上面的设置。

```
//PDO专用定义
'DB_TYPE'=>'pdo',          //数据库类型
'DB_USER'=>'root',         //用户名
'DB_PWD'=>'123456',        //密码
'DB_PREFIX'=>'think_',     //数据库表前缀
'DB_DSN'=>'mysql:host=localhost;dbname=thinkphp;charset=UTF8',
```

二．实例化模型

连接上数据库后，我们需要从数据库里操作数据，那么就需要实例化模型类。在 ThinkPHP 中，提供了 Model 基类处理，也可以使用 M()方法。

```
//实例化Model类，传一个数据表名
$user = new Model('User');
//显示变量结构
var_dump($user);
```

Model 基类可以传递三个参数：

```
Model(['模型名'],['数据表前缀'],['数据库连接信息']);
//实例化Model类，改变表前缀
$user = new Model('User','tp_');
//实例化Model类，定义数据库链接信息
$user =
new Model('User','think_','mysql://root:123456@localhost/thinkphp');

//打印出所有数据
var_dump($user->select());
```

使用 Model 基类还需要导入命名空间，而使用 M()方法，则不需要。

```
//实例化Model类
$user = M('User');
```

除了使用 Model 基类和 M()方法，还有一种对应数据表的模型定义，比如：UserModel。这种模型类并非必须定义的，只有当存在独立的业务逻辑或者属性的时候才需要。

```
//User模型类
namespace Home\Model;
use Think\Model;
class UserModel extends Model {}
```

创建了 UserModel 模型类后，控制器那头就可以直接声明。

```
//User模型类
$user = new UserModel();
var_dump($user->select());
```

为什么 UserModel 模型类没有指定任何表即可直接访问呢？因为这种模型类基本是直接操作数据表的，所以在命名规范上和数据表名是对应的。

模型类与数据表对应规范

模型名	对应的数据表(假设前缀是 think_)
UserModel	think_user
UserTypeModel	think_user_type

虽然使用模型类和数据表对应较为方便，但当有时我们需要更换表名、前缀、附加数据库名等，就需要一些字段定义的操作。为了更加方便的了解数据表的变化，我们使用一下页面 Trace 工具，可以时时的查询 SQL 的变化。

```
//页面Trace，调试辅助工具
'SHOW_PAGE_TRACE' =>true,
```

数据表定义

字段属性	说明
tablePrefix	定义模型对应数据表的前缀
tableName	不包含表前缀的数据表名称
trueTableName	包含表前缀的数据表名称
dbName	定义模型当前对应的数据库名称

```
//重新定义表前缀
class UserModel extends Model {
    protected $tablePrefix = 'abc_';
}

//重新定义表名
class UserModel extends Model {
    protected $tableName = 'abc';
}

//重新定义完整的带前缀的表名
class UserModel extends Model {
    protected $trueTableName = 'tp_abc';
}

//附加数据库名
class UserModel extends Model {
    protected $dbName = 'tp';
}
```

如果你仅仅使用 CURD 等数据库基本操作，我们建议使用基于 Model 基类的 M()方法。使用 M()方法由于不需要加载具体的模型类（比如 UserModel 类），所以性能会更高。

当然，如果有必要使用具体的模型类时，ThinkPHP 还提供了 D()方法来直接是实例化模型类，并且还可以免去引入命名空间等操作。

```
//实例化UserModel类
$user = D('User');
```

PS：使用 D()方法比直接使用模型类更加的智能，如果在\Home\Model\UserModel 找不到该模型类，那么就会去公共模块下找\Common\Model\UserModel 去找。如果还找不到，就会直接实例化基类 Model()类，也就是等同于使用 M()方法。

D()方法可以直接调用当前模块的模型类，那么如果跨模块调用的话，那怎么处理呢？比如 Admin 后台模块，可以使用目录声明。

```
//跨模块实例化
$user = D('Admin/User');
```

有时，你可能想使用原生的 SQL 语句进行操作数据库。那么可以采用实例化空模型基类或者空 M()方法。

```
//空 M()方法
$user = M();                              //或者new Model();空基类
var_dump($user->query("SELECT * FROM think_user WHERE user='蜡笔小新'"));
```

三．字段定义

每个模型类操作着每个对应的数据表，在大多数情况下，系统会自动获取当前数据表的字段信息。而当模型类第一次实例化时，系统会自动缓存字段，并且永久缓存，除非删除了运行时缓存或者设置不缓存。

如果调试模式下，则不会生成字段缓存文件，每次都是从数据表里重新获取。生成缓存的目的显而易见，就是为了快速响应。ThinkPHP 默认是开启字段缓存，因为在实际运行中，不会更改字段结构。

字段缓存文件保存在 Runtime/Data/_fields/目录里，当你在开发阶段，字段和表会经常变动，所以要关闭缓存。关闭缓存的方法为：

```
// 关闭字段缓存
'DB_FIELDS_CACHE'=>false                  //开启了调试模式，自动关闭
```

PS：如果开启缓存状态，新增了字段，那么可能新字段无法刷新出来，必须删除/Data/_fields 文件夹，重新获取字段。

```
//查看字段结构
var_dump($user->getDbFields());
```

你也可以使用手动定义数据表字段的方式取代字段缓存方式，这种方式可以提高性能，避免 IO 开销。

```php
//手动定义数据表字段，_pk表示主键
class UserModel extends Model {
    protected $fields = array('id', 'user', '_pk'=>'id');
}

//type 定义每个字段的类型，可以永远字段验证
class UserModel extends Model {
    protected $fields = array('id', 'user', '_pk'=>'id',
            'type'=>array('id'=>'smallint','user'=>'varchar'));
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--SQL 查询语句

学习要点：

1.查询方式
2.表达式查询
3.快捷查询
4.区间查询
5.组合查询
6.统计查询
7.动态查询
8.SQL 查询

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将学习 ThinkPHP 中对于 SQL 查询语句，包含了基本的查询方式、表达式查询、快捷查询、区间查询、组合查询、统计查询、SQL 查询、动态查询和子查询。

一. 查询方式

ThinkPHP 提供了三种基本的查询方式：字符串条件查询、索引数组条件查询和对象条件查询。在大多数情况下，推荐使用索引数组和对象方式作为查询条件，因为会更加安全

1.使用字符串作为条件查询

```
//字符串作为条件查询
$user = M('User');
var_dump($user->where('id=1 AND user="蜡笔小新"')->select());
//最终生成的 SQL 语句
SELECT * FROM `think_user` WHERE ( id=1 AND user="蜡笔小新" )
```

PS:where 查询方法里面只要包含条件即可，多个条件加上 AND 等连接符即可。我们会在 SQL 连贯操作详细学习。

2.使用索引数组作为查询条件

```
//索引数组作为条件查询
$user = M('User');
$condition['id'] = 1;
$condition['user'] = '蜡笔小新';
var_dump($user->where($condition)->select());

//最终生成的 SQL 语句
SELECT * FROM `think_user` WHERE ( `id` = 1 ) AND ( `user` = '蜡笔小新' )
```

PS：索引数组查询的默认逻辑关系是 AND，如果想改变为 OR，可以使用_logic 定义查询逻辑。

基于上面的代码增加如下一行：

```
$condition['_logic'] = 'OR';                    //将默认 AND 改成 OR
```

3.使用对象方式来查询

```
//对象作为条件查询
$user = M('User');
$condition = new \stdClass();
$condition->id = 1;
$condition->user = '蜡笔小新';
var_dump($user->where($condition)->select());

//最终生成的 SQL 语句
SELECT * FROM `think_user` WHERE ( `id` = 1 ) AND ( `user` = '蜡笔小新' )
```

PS：stdClass 类是 PHP 内置的类，可以理解为一个空类，在这里可以理解为把条件的字段作为成员保存到 stdClass 类里。而这里的'\'是将命名空间设置为根目录，否则会导致当前目录找不到此类。使用对象和数组查询，效果是一样的，可以互换。在大多数情况下，ThinkPHP 推荐使用数组形式更加高效。

二．表达式查询

对于那些要实现模糊判断的查询，比如大于、等于、小于之类的SQL查询，可以使用表达式查询方式。

查询表达式格式：$map['字段名'] = array('表达式','查询条件');

表达式查询表

表达式	含义
EQ	等于(=)
NEQ	不等于(<>)
GT	大于(>)
EGT	大于等于(>=)
LT	小于(<)
ELT	小于等于(<=)
[NOT]LIKE	模糊查询
[NOT] BETWEEN	(不在)区间查询
[NOT] IN	(不在)IN查询
EXP	表达式查询，支持SQL语法

PS：表达式不区分大小写。

```
//EQ：等于(=)
$map['id'] = array('eq', 1);                    //where 为 id=1

//NEQ：不等于(<>)
$map['id'] = array('neq', 1);                   //where 为 id<>1

//GT：大于(>)
$map['id'] = array('gt', 1);                    //where 为 id>1

//EGT：大于等于(>=)
$map['id'] = array('egt', 1);                   //where 为 id>=1

//LT：小于(<)
$map['id'] = array('lt', 1);                    //where 为 id<1

//ELT：小于等于(<=)
$map['id'] = array('elt', 1);                   //where 为 id<=1

//[NOT]LIKE：模糊查询
$map['user'] = array('like', '%小%');           //where 为 like %小%

//[NOT]LIKE：模糊查询
$map['user'] = array('notlike', '%小%');        //where 为 not like %小%

//[NOT]LIKE：模糊查询的数组方式
$map['user'] = array('like', array('%小%', '%蜡%'), 'AND');
//生成的 SQL
SELECT * FROM `think_user` WHERE ( (`user` LIKE '%小%' AND `user`
                                                    LIKE '%蜡%') )

//[NOT] BETWEEN：区间查询
$map['id'] = array('between','1,3');
//where 为`id` BETWEEN '1' AND '2'
//同上等效
$map['id'] = array('between',array('1','3'));

//[NOT] BETWEEN：区间查询
$map['id'] = array('not between','1,3');
//where 为`id` NOT BETWEEN '1' AND '2'
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
//[NOT] IN：区间查询
$map['id'] = array('in','1,2,4');
//where 为`id` IN ('1','2','4')

//[NOT] IN：区间查询
$map['id'] = array('not in','1,2,4');
//where 为`id` NOT IN ('1','2','4')

//EXP：自定义
$map['id'] = array('exp','in (1,2,4)');
//where 为`id` NOT IN ('1','2','4')
PS：使用 exp 自定义在第二个参数直接写 where 语句即可

//EXP：自定义增加 OR 语句
$map['id'] = array('exp', '=1');
$map['user'] = array('exp', '="蜡笔小新"');
$map['_logic'] = 'OR';
//WHERE 为( (`id` =1) ) OR ( (`user` ="蜡笔小新") )
```

三．**快捷查询**

快捷查询方式是一种多字段查询的简化写法，在多个字段之间用'|'隔开表示 OR，用'&'隔开表示 AND。

1.不同字段相同查询条件

```
//使用相同查询条件
$user = M('User');
$map['user|eemail'] = 'a';                    //'|'换成'&'变成AND
var_dump($user->where($map)->select());
```

2.不同字段不同查询条件

```
//使用不同查询条件
$user = M('User');
$map['id&user'] = array(1,'蜡笔小新','_multi'=>true);
var_dump($user->where($map)->select());
```

PS：设置'_multi'为 true，是为了让 id 对应 1，让 user 对应'蜡笔小新'，否则就会出现 id 对应了 1 还要对应'蜡笔小新'的情况。而且，这设置要在放在数组最后。

```
//支持使用表达式结合快捷查询
$user = M('User');
$map['id&user'] = array(array('gt', 0),'蜡笔小新','_multi'=>true);
var_dump($user->where($map)->select());
```

四．区间查询

ThinkPHP 支持对某个字段的区间查询。

```
//区间查询
$user = M('User');
$map['id'] = array(array('gt', 1), array('lt', 4));
var_dump($user->where($map)->select());

//第三个参数设置逻辑OR
$user = M('User');
$map['id'] = array(array('gt', 1), array('lt', 4), 'OR');
var_dump($user->where($map)->select());
```

五．组合查询

组合查询是基于索引数组查询方式的一个扩展性查询，添加了字符串查询(_string)、复合查询(_complex)、请求字符串查询(_query)，由于采用的是索引数组，重复的会被覆盖。

```
//字符串查询(_string)
$user = M('User');
$map['id'] = array('eq', 1);
$map['_string'] ='user="蜡笔小新" AND email="xiaoxin@163.com"';
var_dump($user->where($map)->select());

//请求字符串查询(_query)
$user = M('User');
$map['id'] = array('eq', 1);
$map['_query'] ='user=蜡笔小新&email=xiaoxin@163.com&_logic=OR';
var_dump($user->where($map)->select());
```

PS：这种方式是 URL 方式，不需要加引号。

```
//复合查询(_complex)
$user = M('User');
$where['user'] = array('like', '%小%');
$where['id'] = 1;
$where['_logic'] = 'OR';
$map['_complex'] = $where;
$map['id'] = 3;
$map['_logic'] = 'OR';
var_dump($user->where($map)->select());
```

PS：复合查询可以构建更加复杂的查询，这里 id=1 或者 id=3 可以构建实现。

北风网 IBEIFENG.COM
Ycku.com 瀛城Web俱乐部，做最酷的教学

六. 统计查询

ThinkPHP 提供了一些数据统计查询的方法。

```
//数据总条数
$user = M('User');
var_dump($user->count());

//字段总条数，遇到NULL不统计
$user = M('User');
var_dump($user->count('email'));

//最大值
$user = M('User');
var_dump($user->max('id'));

//最小值
$user = M('User');
var_dump($user->min('id'));

//平均值
$user = M('User');
var_dump($user->avg('id'));

//求总和
$user = M('User');
var_dump($user->sum('id'));
```

七. 动态查询

借助 PHP5 语言的特性，ThinkPHP 实现了动态查询。

1.getBy 动态查询

```
//查找email=xiaoin@163.com的数据
$user = M('User');
var_dump($user->getByemail('xiaoxin@163.com'));
```

2.getFieldBy 动态查询

```
//通过user得到相对应id值
$user = M('User');
var_dump($user->getFieldByUser('路飞', 'id'));
```

八. SQL 查询

ThinkPHP 支持原生 SQL 查询。

1.query 读取

```
//查询结果集，如果采用分布式读写分离，则始终在读服务器执行
$user = M('User');
```

北风网 IBEIFENG.COM
Ycku.com 瓢城Web俱乐部，做最酷的教学

```
    var_dump($user->query('SELECT * FROM think_user'));

    2.execute写入
    //更新和写入，如果采用分布式读写分离，则始终在写服务器执行
    $user = M('User');
    var_dump($user->execute('UPDATE think_user set user="蜡笔大新" WHERE
id=1'));
```

PS：由于子查询用了不少连贯操作，我们会在连贯操作讲解。

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--SQL 查询语句

学习要点：

1.查询方式
2.表达式查询
3.快捷查询
4.区间查询
5.组合查询
6.统计查询
7.动态查询
8.SQL 查询

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将学习 ThinkPHP 中对于 SQL 查询语句，包含了基本的查询方式、表达式查询、快捷查询、区间查询、组合查询、统计查询、SQL 查询、动态查询和子查询。

一. 查询方式

ThinkPHP 提供了三种基本的查询方式：字符串条件查询、索引数组条件查询和对象条件查询。在大多数情况下，推荐使用索引数组和对象方式作为查询条件，因为会更加安全

1.使用字符串作为条件查询

```
//字符串作为条件查询
$user = M('User');
var_dump($user->where('id=1 AND user="蜡笔小新"')->select());
//最终生成的 SQL 语句
SELECT * FROM `think_user` WHERE ( id=1 AND user="蜡笔小新" )
```

PS:where 查询方法里面只要包含条件即可，多个条件加上 AND 等连接符即可。我们会在 SQL 连贯操作详细学习。

2.使用索引数组作为查询条件

```
//索引数组作为条件查询
$user = M('User');
$condition['id'] = 1;
$condition['user'] = '蜡笔小新';
var_dump($user->where($condition)->select());

//最终生成的 SQL 语句
SELECT * FROM `think_user` WHERE ( `id` = 1 ) AND ( `user` = '蜡笔小新' )
```

PS：索引数组查询的默认逻辑关系是 AND，如果想改变为 OR，可以使用_logic 定义查询逻辑。

基于上面的代码增加如下一行：

```
$condition['_logic'] = 'OR';                    //将默认 AND 改成 OR
```

3.使用对象方式来查询

```
//对象作为条件查询
$user = M('User');
$condition = new \stdClass();
$condition->id = 1;
$condition->user = '蜡笔小新';
var_dump($user->where($condition)->select());

//最终生成的 SQL 语句
SELECT * FROM `think_user` WHERE ( `id` = 1 ) AND ( `user` = '蜡笔小新' )
```

PS：stdClass 类是 PHP 内置的类，可以理解为一个空类，在这里可以理解为把条件的字段作为成员保存到 stdClass 类里。而这里的'\'是将命名空间设置为根目录，否则会导致当前目录找不到此类。使用对象和数组查询，效果是一样的，可以互换。在大多数情况下，ThinkPHP 推荐使用数组形式更加高效。

二．表达式查询

对于那些要实现模糊判断的查询，比如大于、等于、小于之类的SQL查询，可以使用表达式查询方式。

查询表达式格式：$map['字段名'] = array('表达式','查询条件');

表达式查询表

表达式	含义
EQ	等于(=)
NEQ	不等于(<>)
GT	大于(>)
EGT	大于等于(>=)
LT	小于(<)
ELT	小于等于(<=)
[NOT]LIKE	模糊查询
[NOT] BETWEEN	(不在)区间查询
[NOT] IN	(不在)IN查询
EXP	表达式查询，支持SQL语法

PS：表达式不区分大小写。

```
//EQ：等于(=)
$map['id'] = array('eq', 1);                    //where 为 id=1

//NEQ：不等于(<>)
$map['id'] = array('neq', 1);                   //where 为 id<>1

//GT：大于(>)
$map['id'] = array('gt', 1);                    //where 为 id>1

//EGT：大于等于(>=)
$map['id'] = array('egt', 1);                   //where 为 id>=1

//LT：小于(<)
$map['id'] = array('lt', 1);                    //where 为 id<1

//ELT：小于等于(<=)
$map['id'] = array('elt', 1);                   //where 为 id<=1

//[NOT]LIKE：模糊查询
$map['user'] = array('like', '%小%');           //where 为 like %小%

//[NOT]LIKE：模糊查询
$map['user'] = array('notlike', '%小%');        //where 为 not like %小%

//[NOT]LIKE：模糊查询的数组方式
$map['user'] = array('like', array('%小%', '%蜡%'), 'AND');
//生成的 SQL
SELECT * FROM `think_user` WHERE ( (`user` LIKE '%小%' AND `user`
                                                    LIKE '%蜡%') )

//[NOT] BETWEEN：区间查询
$map['id'] = array('between','1,3');
//where 为`id` BETWEEN '1' AND '2'
//同上等效
$map['id'] = array('between',array('1','3'));

//[NOT] BETWEEN：区间查询
$map['id'] = array('not between','1,3');
//where 为`id` NOT BETWEEN '1' AND '2'
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
//[NOT] IN：区间查询
$map['id'] = array('in','1,2,4');
//where 为`id` IN ('1','2','4')

//[NOT] IN：区间查询
$map['id'] = array('not in','1,2,4');
//where 为`id` NOT IN ('1','2','4')

//EXP：自定义
$map['id'] = array('exp','in (1,2,4)');
//where 为`id` NOT IN ('1','2','4')
PS：使用 exp 自定义在第二个参数直接写 where 语句即可

//EXP：自定义增加 OR 语句
$map['id'] = array('exp', '=1');
$map['user'] = array('exp', '="蜡笔小新"');
$map['_logic'] = 'OR';
//WHERE 为( (`id` =1) ) OR ( (`user` ="蜡笔小新") )
```

三．**快捷查询**

快捷查询方式是一种多字段查询的简化写法，在多个字段之间用'|'隔开表示 OR，用'&'隔开表示 AND。

1.不同字段相同查询条件

```
//使用相同查询条件
$user = M('User');
$map['user|eemail'] = 'a';                    //'|'换成'&'变成AND
var_dump($user->where($map)->select());
```

2.不同字段不同查询条件

```
//使用不同查询条件
$user = M('User');
$map['id&user'] = array(1,'蜡笔小新','_multi'=>true);
var_dump($user->where($map)->select());
```

PS：设置'_multi'为 true，是为了让 id 对应 1，让 user 对应'蜡笔小新'，否则就会出现 id 对应了 1 还要对应'蜡笔小新'的情况。而且，这设置要在放在数组最后。

```
//支持使用表达式结合快捷查询
$user = M('User');
$map['id&user'] = array(array('gt', 0),'蜡笔小新','_multi'=>true);
var_dump($user->where($map)->select());
```

四．区间查询

ThinkPHP 支持对某个字段的区间查询。

```
//区间查询
$user = M('User');
$map['id'] = array(array('gt', 1), array('lt', 4));
var_dump($user->where($map)->select());

//第三个参数设置逻辑OR
$user = M('User');
$map['id'] = array(array('gt', 1), array('lt', 4), 'OR');
var_dump($user->where($map)->select());
```

五．组合查询

组合查询是基于索引数组查询方式的一个扩展性查询，添加了字符串查询(_string)、复合查询(_complex)、请求字符串查询(_query)，由于采用的是索引数组，重复的会被覆盖。

```
//字符串查询(_string)
$user = M('User');
$map['id'] = array('eq', 1);
$map['_string'] ='user="蜡笔小新" AND email="xiaoxin@163.com"';
var_dump($user->where($map)->select());

//请求字符串查询(_query)
$user = M('User');
$map['id'] = array('eq', 1);
$map['_query'] ='user=蜡笔小新&email=xiaoxin@163.com&_logic=OR';
var_dump($user->where($map)->select());
```

PS：这种方式是 URL 方式，不需要加引号。

```
//复合查询(_complex)
$user = M('User');
$where['user'] = array('like', '%小%');
$where['id'] = 1;
$where['_logic'] = 'OR';
$map['_complex'] = $where;
$map['id'] = 3;
$map['_logic'] = 'OR';
var_dump($user->where($map)->select());
```

PS：复合查询可以构建更加复杂的查询，这里 id=1 或者 id=3 可以构建实现。

北风网 IBEIFENG.COM
Ycku.com 瀛城Web俱乐部，做最酷的教学

六. 统计查询

ThinkPHP 提供了一些数据统计查询的方法。

```
//数据总条数
$user = M('User');
var_dump($user->count());

//字段总条数，遇到NULL不统计
$user = M('User');
var_dump($user->count('email'));

//最大值
$user = M('User');
var_dump($user->max('id'));

//最小值
$user = M('User');
var_dump($user->min('id'));

//平均值
$user = M('User');
var_dump($user->avg('id'));

//求总和
$user = M('User');
var_dump($user->sum('id'));
```

七. 动态查询

借助 PHP5 语言的特性，ThinkPHP 实现了动态查询。

1.getBy 动态查询

```
//查找email=xiaoin@163.com的数据
$user = M('User');
var_dump($user->getByemail('xiaoxin@163.com'));
```

2.getFieldBy 动态查询

```
//通过user得到相对应id值
$user = M('User');
var_dump($user->getFieldByUser('路飞', 'id'));
```

八. SQL 查询

ThinkPHP 支持原生 SQL 查询。

1.query 读取

```
//查询结果集，如果采用分布式读写分离，则始终在读服务器执行
$user = M('User');
```

北风网 IBEIFENG.COM
Ycku.com 瓢城Web俱乐部，做最酷的教学

```
    var_dump($user->query('SELECT * FROM think_user'));

    2.execute写入
    //更新和写入，如果采用分布式读写分离，则始终在写服务器执行
    $user = M('User');
    var_dump($user->execute('UPDATE think_user set user="蜡笔大新" WHERE
id=1'));
```

PS：由于子查询用了不少连贯操作，我们会在连贯操作讲解。

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--SQL 连贯操作

学习要点：

1.连贯入门

2.连贯方法

3.命名范围

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将学习 ThinkPHP 模型基础类提供的连贯操作，通过连贯操作可以有效的提供数据存取的代码清晰度和开发效率，并且支持所有的 CURD 操作。

一. 连贯入门

连贯操作使用起来非常简单，比如查找到 id 为 1,2,3,4 中按照创建时间的倒序的前两位。

```
//连贯操作入门
$user = M('User');
var_dump($user->where('id in (1,2,3,4)')->order('date DESC')->limit(2)->select());
```

PS：这里的 where、order 和 limit 方法都是连贯操作方法，所以它们都能返回$user 本身，可以互换位置。而 select 方法不是连贯方法，需要放在最后，用以显示数据集。

```
//数组操作
$user = M('User');
var_dump($user->select(array('where'=>'id in (1,2,3,4)', 'limit'=>'2', 'order'=>'date DESC')));

//CURD处理，CURD会在专门章节讲解
$user = M('User');
var_dump($user->where('id=1')->find());
var_dump($user->where('id=7')->delete());
```

系统支持的连贯方法

连贯操作	作用	支持的参数类型
where*	用于查询或者更新条件的定义	字符串、数组和对象
table	用于定义要操作的数据表名称	字符串和数组
alias	用于给当前数据表定义别名	字符串
data	用于新增或者更新数据之前的数据对象赋值	数组和对象

field	用于定义要查询的字段（支持字段排除）	字符串和数组
order	用于对结果排序	字符串和数组
limit	用于限制查询结果数量	字符串和数字
page	用于查询分页（内部会转换成 limit）	字符串和数字
group	用于对查询的 group 支持	字符串
having	用于对查询的 having 支持	字符串
join*	用于对查询的 join 支持	字符串和数组
union*	用于对查询的 union 支持	字符串、数组和对象
distinct	用于查询的 distinct 支持	布尔值
lock	用于数据库的锁机制	布尔值
cache	用于查询缓存	支持多个参数
relation	用于关联查询（需要关联模型支持）	字符串
result	用于返回数据转换	字符串
validate	用于数据自动验证	数组
auto	用于数据自动完成	数组
filter	用于数据过滤	字符串
scope*	用于命名范围	字符串、数组
bind*	用于数据绑定操作	数组或多个参数
token	用于令牌验证	布尔值
comment	用于 SQL 注释	字符串

PS：带*号的可以多次调用。本节课没有讲到的连贯方法，会在其他章节探讨。

二. 连贯方法

1.where

where 方法支持字符串条件、数组条件（推荐用法）和多次调用。

```
//字符串方式
$user = M('User');
var_dump($user->where('id=1')->select());

//索引数组方式
$user = M('User');
$map['id'] = 1;                          //使用表达式array('eq', 1);
var_dump($user->where($map)->select());

//多次调用方式
```

```
$user = M('User');
$map['id'] = array('eq', 1);
var_dump($user->where($map)->where('user="蜡笔小新"')->select());
```

2.order

order 用于对结果集排序。

```
//倒序
$user = M('User');
$map['id'] = array('eq', 1);
var_dump($user->order('id desc')->select());    //正序默认或 ASC

//第二排序
var_dump($user->order('id desc,email desc')->select());
```

PS：先按 id 倒序，再按 email 倒序

```
//数组形式防止字段和mysql关键字冲突
$user = M('User');
$map['id'] = array('eq', 1);
var_dump($user->order(array('id'=>'DESC'))->select());
```

3.feild

feild 方法可以返回或操作字段，可以用于查询和写入操作。

```
//只显示id和user两个字段
$user = M('User');
var_dump($user->field('id, user')->select());

//使用SQL函数和别名
$user = M('User');
var_dump($user->field('SUM(id) as count, user')->select());

//使用数组参数结合SQL函数
$user = M('User');
var_dump($user->field(array('id','LEFT(user,3)'=>'left_user'))->sele
ct());

//获取所有字段
$user = M('User');
var_dump($user->field()->select());          //可以传入*号，或者省略方法

//用于写入
$user = M('User');
$user->field('user,email')->create();        //CURD 将在专门的章节学习
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

3.limit

limit 方法主要用于指定查询和操作的数量。

```
//限制结果集数量
$user = M('User');
var_dump($user->limit(2)->select());

//分页查询
$user = M('User');
var_dump($user->limit(0,2)->select());              //2,2、,4,2
```

4.page

page 方法完全用于分页查询。

```
//page分页
$user = M('User');
var_dump($user->page(1,2)->select());               //2,2、3,2
```

5.table

table 方法用于数据表操作，主要是切换数据表或多表操作。

```
//切换数据表
$user = M('User');
var_dump($user->table('think_info')->select());

//获取简化表名
$user = M('User');
var_dump($user->table('__USER__')->select());       //__INFO__尚可

//多表查询
$user = M('User');
var_dump($user->field('a.id,b.id')->table('__USER__ a,__INFO__
b')->select());

//多表查询，使用数组形式避免关键字冲突
$user = M('User');
var_dump($user->field('a.id,b.id')->table(array('think_user'=>'a',
'think_info'=>'b'))->select());
```

6.alias

alias 用于设置数据表别名

```
//设置别名
$user = M('User');
var_dump($user->alias('a')->select());
```

7.group

group 方法通常用于对结合函数统计的结果集分组。

```
//分组统计
$user = M('User');
var_dump($user->field('user,max(id)')->group('id')->select());
```

PS：group 会在 mysql 部分单独探讨。

8.having

having 方法一般用于配合 group 方法完成从分组的结果中再筛选数据。

```
//分组统计结合having
$user = M('User');
var_dump($user->field('user,max(id)')->group('id')->having('id>2')->
select());
```

PS：having 会在 mysql 部分单独探讨。

9.comment

comment 方法用于对 SQL 语句进行注释

```
//SQL注释
$user = M('User');
var_dump($user->comment('所有用户')->select());
```

10.join

join 方法用于多表的连接查询。

```
//JOIN多表关联，默认是INNER JOIN
$user = M('User');
var_dump($user->join('think_user ON think_info.id =
think_user.id')->select());                    //__USER__和__INFO__代替

//RIGHT、LEFT、FULL
var_dump($user->join('think_user ON think_info.id =
think_user.id','RIGHT')->select());
```

PS：join 会在 mysql 部分单独探讨。

11.union

union 方法用于合并多个 SELECT 的结果集

```
//合并多个SELECT结果集
$user = M('User');
var_dump($user->union("SELECT * FROM think_info")->select());
```

PS：union 会在 mysql 部分单独探讨。

12.distinct

distinct 方法用于返回唯一不同的值

```
//返回不重复的列
$user = M('User');
var_dump($user->distinct(true)->field('user')->select());
```

13.cache

cache 用于查询缓存操作

```
//查询缓存，第二次读取缓存内容
$user = M('User');
var_dump($user->cache(true)->select());
```

PS：第一次查询数据库，第二次查询相同的内容直接调用缓存，不用再查询数据库。更多关于 cache 和缓存的使用方法，我们将在缓存那节详细探讨。

三. 命名范围

命名范围其实就是将 SQL 语句封装在模型定义类里，而不在控制器里。这样的分层操作有利于代码的可读性，避免开发人员在写 CURD 操作时出现问题。架构人员只要在命名范围内合理的规划即可，类似于架构师架构了接口，让开发人员面向接口开发一样。

要使用命名范围，第一步要定义属性：

```
class UserModel extends Model {
    protected $_scope = array(                  //属性名必须是_scope
        'sql1'=>array(
            'where'=>array('id'=>1),
        ),
        'sql2'=>array(
            'order'=>'date DESC',
            'limit'=>2,
        ),
        'default'=>array(
            'where'=>array('id'=>2),
        ),
    );
}
```

命名范围支持的属性有：where、field、order、table、limit、page、having、group、lock、distinct、cache。

```
//调用命名范围
$user = D('User');
var_dump($user->scope('sql2')->select());

//支持调用多个scope方法
$user = D('User');
```

```
    var_dump($user->scope('sql1')->scope('sql2')->select());

    //default默认
    $user = D('User');
    var_dump($user->scope()->select());          //传递 default 也行

    PS：如果传递不存在的命名范围，则忽略。

    //对命名范围的SQL进行调整
    $user = D('User');
    var_dump($user->scope('sql2', array('limit'=>4))->select());

    //直接覆盖命名范围
    $user = D('User');
    var_dump($user->scope(array('where'=>1,'order'=>'date
DESC','limit'=>2))->select());

    //直接用命名范围名调用
    $user = D('User');
    var_dump($user->sql2()->select());
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--SQL 连贯操作

学习要点：

1.连贯入门

2.连贯方法

3.命名范围

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将学习 ThinkPHP 模型基础类提供的连贯操作，通过连贯操作可以有效的提供数据存取的代码清晰度和开发效率，并且支持所有的 CURD 操作。

一. 连贯入门

连贯操作使用起来非常简单，比如查找到 id 为 1,2,3,4 中按照创建时间的倒序的前两位。

```
//连贯操作入门
$user = M('User');
var_dump($user->where('id in (1,2,3,4)')->order('date DESC')->limit(2)->select());
```

PS：这里的 where、order 和 limit 方法都是连贯操作方法，所以它们都能返回$user 本身，可以互换位置。而 select 方法不是连贯方法，需要放在最后，用以显示数据集。

```
//数组操作
$user = M('User');
var_dump($user->select(array('where'=>'id in (1,2,3,4)', 'limit'=>'2', 'order'=>'date DESC')));

//CURD处理，CURD会在专门章节讲解
$user = M('User');
var_dump($user->where('id=1')->find());
var_dump($user->where('id=7')->delete());
```

系统支持的连贯方法

连贯操作	作用	支持的参数类型
where*	用于查询或者更新条件的定义	字符串、数组和对象
table	用于定义要操作的数据表名称	字符串和数组
alias	用于给当前数据表定义别名	字符串
data	用于新增或者更新数据之前的数据对象赋值	数组和对象

field	用于定义要查询的字段（支持字段排除）	字符串和数组
order	用于对结果排序	字符串和数组
limit	用于限制查询结果数量	字符串和数字
page	用于查询分页（内部会转换成 limit）	字符串和数字
group	用于对查询的 group 支持	字符串
having	用于对查询的 having 支持	字符串
join*	用于对查询的 join 支持	字符串和数组
union*	用于对查询的 union 支持	字符串、数组和对象
distinct	用于查询的 distinct 支持	布尔值
lock	用于数据库的锁机制	布尔值
cache	用于查询缓存	支持多个参数
relation	用于关联查询（需要关联模型支持）	字符串
result	用于返回数据转换	字符串
validate	用于数据自动验证	数组
auto	用于数据自动完成	数组
filter	用于数据过滤	字符串
scope*	用于命名范围	字符串、数组
bind*	用于数据绑定操作	数组或多个参数
token	用于令牌验证	布尔值
comment	用于 SQL 注释	字符串

PS：带*号的可以多次调用。本节课没有讲到的连贯方法，会在其他章节探讨。

二. 连贯方法

1.where

where 方法支持字符串条件、数组条件（推荐用法）和多次调用。

```
//字符串方式
$user = M('User');
var_dump($user->where('id=1')->select());

//索引数组方式
$user = M('User');
$map['id'] = 1;                          //使用表达式array('eq', 1);
var_dump($user->where($map)->select());

//多次调用方式
```

```
$user = M('User');
$map['id'] = array('eq', 1);
var_dump($user->where($map)->where('user="蜡笔小新"')->select());
```

2.order

order 用于对结果集排序。

```
//倒序
$user = M('User');
$map['id'] = array('eq', 1);
var_dump($user->order('id desc')->select());    //正序默认或 ASC

//第二排序
var_dump($user->order('id desc,email desc')->select());
```

PS：先按 id 倒序，再按 email 倒序

```
//数组形式防止字段和mysql关键字冲突
$user = M('User');
$map['id'] = array('eq', 1);
var_dump($user->order(array('id'=>'DESC'))->select());
```

3.feild

feild 方法可以返回或操作字段，可以用于查询和写入操作。

```
//只显示id和user两个字段
$user = M('User');
var_dump($user->field('id, user')->select());

//使用SQL函数和别名
$user = M('User');
var_dump($user->field('SUM(id) as count, user')->select());

//使用数组参数结合SQL函数
$user = M('User');
var_dump($user->field(array('id','LEFT(user,3)'=>'left_user'))->sele
ct());

//获取所有字段
$user = M('User');
var_dump($user->field()->select());          //可以传入*号，或者省略方法

//用于写入
$user = M('User');
$user->field('user,email')->create();        //CURD 将在专门的章节学习
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

3.limit

limit 方法主要用于指定查询和操作的数量。

```
//限制结果集数量
$user = M('User');
var_dump($user->limit(2)->select());

//分页查询
$user = M('User');
var_dump($user->limit(0,2)->select());              //2,2、,4,2
```

4.page

page 方法完全用于分页查询。

```
//page分页
$user = M('User');
var_dump($user->page(1,2)->select());               //2,2、3,2
```

5.table

table 方法用于数据表操作，主要是切换数据表或多表操作。

```
//切换数据表
$user = M('User');
var_dump($user->table('think_info')->select());

//获取简化表名
$user = M('User');
var_dump($user->table('__USER__')->select());       //__INFO__尚可

//多表查询
$user = M('User');
var_dump($user->field('a.id,b.id')->table('__USER__ a,__INFO__
b')->select());

//多表查询，使用数组形式避免关键字冲突
$user = M('User');
var_dump($user->field('a.id,b.id')->table(array('think_user'=>'a',
'think_info'=>'b'))->select());
```

6.alias

alias 用于设置数据表别名

```
//设置别名
$user = M('User');
var_dump($user->alias('a')->select());
```

7.group

group 方法通常用于对结合函数统计的结果集分组。

```
//分组统计
$user = M('User');
var_dump($user->field('user,max(id)')->group('id')->select());
```

PS：group 会在 mysql 部分单独探讨。

8.having

having 方法一般用于配合 group 方法完成从分组的结果中再筛选数据。

```
//分组统计结合having
$user = M('User');
var_dump($user->field('user,max(id)')->group('id')->having('id>2')->
select());
```

PS：having 会在 mysql 部分单独探讨。

9.comment

comment 方法用于对 SQL 语句进行注释

```
//SQL注释
$user = M('User');
var_dump($user->comment('所有用户')->select());
```

10.join

join 方法用于多表的连接查询。

```
//JOIN多表关联，默认是INNER JOIN
$user = M('User');
var_dump($user->join('think_user ON think_info.id =
think_user.id')->select());                    //__USER__和__INFO__代替

//RIGHT、LEFT、FULL
var_dump($user->join('think_user ON think_info.id =
think_user.id','RIGHT')->select());
```

PS：join 会在 mysql 部分单独探讨。

11.union

union 方法用于合并多个 SELECT 的结果集

```
//合并多个SELECT结果集
$user = M('User');
var_dump($user->union("SELECT * FROM think_info")->select());
```

PS：union 会在 mysql 部分单独探讨。

12.distinct

distinct 方法用于返回唯一不同的值

```
//返回不重复的列
$user = M('User');
var_dump($user->distinct(true)->field('user')->select());
```

13.cache

cache 用于查询缓存操作

```
//查询缓存，第二次读取缓存内容
$user = M('User');
var_dump($user->cache(true)->select());
```

PS：第一次查询数据库，第二次查询相同的内容直接调用缓存，不用再查询数据库。更多关于 cache 和缓存的使用方法，我们将在缓存那节详细探讨。

三. 命名范围

命名范围其实就是将 SQL 语句封装在模型定义类里，而不在控制器里。这样的分层操作有利于代码的可读性，避免开发人员在写 CURD 操作时出现问题。架构人员只要在命名范围内合理的规划即可，类似于架构师架构了接口，让开发人员面向接口开发一样。

要使用命名范围，第一步要定义属性：

```
class UserModel extends Model {
    protected $_scope = array(                  //属性名必须是_scope
        'sql1'=>array(
            'where'=>array('id'=>1),
        ),
        'sql2'=>array(
            'order'=>'date DESC',
            'limit'=>2,
        ),
        'default'=>array(
            'where'=>array('id'=>2),
        ),
    );
}
```

命名范围支持的属性有：where、field、order、table、limit、page、having、group、lock、distinct、cache。

```
//调用命名范围
$user = D('User');
var_dump($user->scope('sql2')->select());

//支持调用多个scope方法
$user = D('User');
```

```
    var_dump($user->scope('sql1')->scope('sql2')->select());

    //default默认
    $user = D('User');
    var_dump($user->scope()->select());          //传递 default 也行

    PS：如果传递不存在的命名范围，则忽略。

    //对命名范围的SQL进行调整
    $user = D('User');
    var_dump($user->scope('sql2', array('limit'=>4))->select());

    //直接覆盖命名范围
    $user = D('User');
    var_dump($user->scope(array('where'=>1,'order'=>'date
DESC','limit'=>2))->select());

    //直接用命名范围名调用
    $user = D('User');
    var_dump($user->sql2()->select());
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--CURD 操作

学习要点：

1.数据创建
2.数据写入
3.数据读取
4.数据更新
5.数据删除

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将学习 ThinkPHP 模型中的 CURD 操作，也就是增删改查。通过 CURD，我们可以方便快速的对数据库进行操作。这里我们用到了静态 html 文件，Zend Studio 安装 Aptana 插件，安装配置方法：http://mrthink.net/zend-studio-aptana-tips/。

一．数据创建

在数据库添加等操作之前，我们首先需要对数据进行创建。何为数据创建，就是接受提交过来的数据，比如表单提交的 POST（默认）数据。接受到数据后，还可以对数据进行有效的验证、完成、生成等工作。

```
//根据表单提交的POST数据，创建数据对象
$user = M('User');
var_dump($user->create());
```

PS：这里 create()方法就是数据创建，数据的结果就是提交的 POST 数据的键值对。特别注意的是：提交过来的字段和数据表字段是对应的，否则无法解析。

```
//通过数组手工获取数据，覆盖提交的
$user = M('User');
$data['user'] = $_POST['user'];
$data['email'] = $_POST['email'];
$data['date'] = date('Y-m-d H:i:s');     //和数据表对应，否则无效
var_dump($user->create($data));

//通过对象手工获取数据，覆盖提交的
$user = M('User');
$data = new \stdClass();
$data->user = $_POST['user'];
$data->email = $_POST['email'];
$data->date = date('Y-m-d H:i:s');
var_dump($user->create($data));
```

```
//默认是$_POST，传递$_GET修改
$user = M('User');
var_dump($user->create($_GET));
```

create()方法可以传递第二个参数，将要操作的模式，有两种：Model::MODEL_INSERT 和 Model::MODEL_UPDATE，即新增和修改。当没有指定的时候，系统会根据数据源是否包含主键来自动判断，如果包含主键，则就是修改操作。

```
//设置将要新增操作
$user = M('User');
var_dump($user->create($_POST, Model::MODEL_INSERT));
```

create()方法的内部工作分为 9 步：

1.获取数据源（默认是 POST）；
2.验证数据合法性（非数据或对象会过滤），失败则返回 false；
3.检查字段映射；
4.判断数据状态（新增还是修改）；
5.数据自动验证，失败则返回 false；
6.表单令牌验证，失败则返回 false；
7.表单数据赋值（过滤非法字段和字符串处理）；
8.数据自动完成；
9.生成数据对象（保存在内存）。

create()方法可以配合连贯操作配合数据创建，支持的连贯操作有：

1.field，用于定义合法的字段；
2.validate，用于数据自动验证；
3.auto，用于数据自动完成；
4.token，用于令牌验证。

```
//限制可操作的字段
$user = M('User');
var_dump($user->field('user')->create());

//在模型类里限制字段
class UserModel extends Model {
    protected $insertFields = 'user';
    protected $updateFields = 'user';
}
```

二. 数据写入

数据写入使用的是 add()方法。

```
//新增一条数据
$user = M('User');
$data['user'] = '李炎恢';
```

```php
$data['email'] = 'yc60.com@gmail.com';
$data['date'] = date('Y-m-d H:i:s');
$user->add($data);

//结合create()方法
$user = M('User');
$data = $user->create();
$data['date'] = date('Y-m-d H:i:s');
$user->add($data);
```

add()方法支持的连贯操作有：
1.table，定义数据表名称；
2.data，指定要写入的数据对象；
3.field，定义要写入的字段；
4.relation，关联查询；
5.validate，数据自动验证；
6.auto，数据自动完成；
7.filter，数据过滤；
8.scope*，命名范围；
9.bind，数据绑定操作；
10.token，令牌验证；
11.comment，SQL 注释；

```php
//使用data连贯方法
$user = M('User');
$data = $user->create();
$data['date'] = date('Y-m-d H:i:s');
$user->data($data)->add();

//data连贯方法 支持字符串、数组、对象
$user = M('User');
$data = 'user=星矢&mail=xinshi@qq.com&date='.date('Y-m-d H:i:s');
$user->data($data)->add();
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--CURD 操作

学习要点：

1.数据创建
2.数据写入
3.数据读取
4.数据更新
5.数据删除
6.ActiveRecord 模式
7.字段映射

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将学习 ThinkPHP 模型中的 CURD 操作，也就是增删改查。通过 CURD，我们可以方便快速的对数据库进行操作。这里我们用到了静态 html 文件，Zend Studio 安装 Aptana 插件，安装配置方法：http://mrthink.net/zend-studio-aptana-tips/。

一．数据创建

在数据库添加等操作之前，我们首先需要对数据进行创建。何为数据创建，就是接受提交过来的数据，比如表单提交的 POST（默认）数据。接受到数据后，还可以对数据进行有效的验证、完成、生成等工作。

```
//根据表单提交的POST数据，创建数据对象
$user = M('User');
var_dump($user->create());
```

PS：这里 create()方法就是数据创建，数据的结果就是提交的 POST 数据的键值对。特别注意的是：提交过来的字段和数据表字段是对应的，否则无法解析。

```
//通过数组手工获取数据，覆盖提交的
$user = M('User');
$data['user'] = $_POST['user'];
$data['email'] = $_POST['email'];
$data['date'] = date('Y-m-d H:i:s');     //和数据表对应，否则无效
var_dump($user->create($data));

//通过对象手工获取数据，覆盖提交的
$user = M('User');
$data = new \stdClass();
$data->user = $_POST['user'];
$data->email = $_POST['email'];
$data->date = date('Y-m-d H:i:s');
```

```
var_dump($user->create($data));

//默认是$_POST，传递$_GET修改
$user = M('User');
var_dump($user->create($_GET));
```

create()方法可以传递第二个参数，将要操作的模式，有两种：Model::MODEL_INSERT 和 Model::MODEL_UPDATE，即新增和修改。当没有指定的时候，系统会根据数据源是否包含主键来自动判断，如果包含主键，则就是修改操作。

```
//设置将要新增操作
$user = M('User');
var_dump($user->create($_POST, Model::MODEL_INSERT));
```

create()方法的内部工作分为 9 步：

1.获取数据源（默认是 POST）；
2.验证数据合法性（非数据或对象会过滤），失败则返回 false；
3.检查字段映射；
4.判断数据状态（新增还是修改）；
5.数据自动验证，失败则返回 false；
6.表单令牌验证，失败则返回 false；
7.表单数据赋值（过滤非法字段和字符串处理）；
8.数据自动完成；
9.生成数据对象（保存在内存）。

create()方法可以配合连贯操作配合数据创建，支持的连贯操作有：

1.field，用于定义合法的字段；
2.validate，用于数据自动验证；
3.auto，用于数据自动完成；
4.token，用于令牌验证。

```
//限制可操作的字段
$user = M('User');
var_dump($user->field('user')->create());

//在模型类里限制字段
class UserModel extends Model {
    protected $insertFields = 'user';
    protected $updateFields = 'user';
}
```

二．数据写入

数据写入使用的是 add()方法。

```
//新增一条数据
```

```php
$user = M('User');
$data['user'] = '李炎恢';
$data['email'] = 'yc60.com@gmail.com';
$data['date'] = date('Y-m-d H:i:s');
$user->add($data);

//结合create()方法
$user = M('User');
$data = $user->create();
$data['date'] = date('Y-m-d H:i:s');
$user->add($data);
```

add()方法支持的连贯操作有：

1.table，定义数据表名称；
2.data，指定要写入的数据对象；
3.field，定义要写入的字段；
4.relation，关联查询；
5.validate，数据自动验证；
6.auto，数据自动完成；
7.filter，数据过滤；
8.scope*，命名范围；
9.bind，数据绑定操作；
10.token，令牌验证；
11.comment，SQL 注释；

```php
//使用data连贯方法
$user = M('User');
$data = $user->create();
$data['date'] = date('Y-m-d H:i:s');
$user->data($data)->add();

//data连贯方法 支持字符串、数组、对象
$user = M('User');
$data = 'user=星矢&mail=xinshi@qq.com&date='.date('Y-m-d H:i:s');
$user->data($data)->add();
```

三．数据读取

在之前的课程中，我们已经大量使用了数据读取的功能，比如 select()方法。结合各种连贯方法可以实现数据读取的不同要求，支持连贯的方法有：

1.where，查询或更新条件；
2.table，要操作的数据表名称；
3.alias，数据表别名；
4.field，查询字段；

5.order，结果排序；
6.group，查询分组；
7.having，分组再查询；
8.join，多表链接查询；
9.union，合并 SELECT；
10.distinct，取唯一值；
11.lock，锁；
12.cache，缓存；
13.relation，关联查询；
14.result，数据转换；
15.scope，命名范围；
16.bind，数据绑定操作；
17.comment，SQL 注释。

```
//显示默认第一条数据
$user = M('User');
var_dump($user->find());                    //可以传递数字参数，AR 模式

//显示默认所有数据
$user = M('User');
var_dump($user->select());                  //可以传递数组形式的 SQL

//获取第一条user字段的值
$user = M('User');
var_dump($user->getField('user'));

//获取所有user字段的值
$user = M('User');
var_dump($user->getField('user',true));

//传递多个字段，获取所有
$user = M('User');
var_dump($user->getField('user,email'));

//id冒号分隔
$user = M('User');
var_dump($user->getField('id,user,email',':'));

//限制2条数据
$user = M('User');
var_dump($user->getField('id,user,email',2));
```

四．数据更新

北风网 IBEIFENG.COM
Ycku.com 瀛城Web俱乐部，做最酷的教学

数据更新使用的方法是 save()方法，主要是对数据的修改操作。

```
//修改第一条数据
$user = M('User');
$data['user'] = '蜡笔大新';
$data['email'] = 'daxin@qq.com';
$map['id'] = 1;
$user->where($map)->save($data);                    //成功后返回 1，否则 0

//默认主键为条件
$user = M('User');
$data['id'] = 1;
$data['user'] = '蜡笔小新';
$data['email'] = 'xiaoxin@163.com';
$user->save($data);
```

数据更新的 save()方法支持的连贯方法有：

1.where，查询或更新条件；
2.table，要操作的数据表名称；
3.alias，数据表别名；
4.field，查询字段；
5.order，结果排序；
6.lock，锁；
7.relation，关联查询；
8.scope，命名范围；
9.bind，数据绑定操作；
10.comment，SQL 注释。

```
//结合create()
$user = M('User');
$user->create();                                    //POST必须包含主键
$user->save();

//修改某一个值
$user = M('User');
$map['id'] = 1;
$user->where($map)->setField('user', '蜡笔大新');

//统计累计，累加累减
$user = M('User');
$map['id'] = 1;
$user->where($map)->setInc('count',1);  //累加，setDec 累减
```

五．数据删除

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

数据删除使用的方法是 delete()方法。

```
//直接删除主键(id=17)
$user = M('User');
$user->delete(17);

//根据ID来删除
$user = M('User');
$map['id'] = 16;
$user->where($map)->delete();

//批量删除多个
$user = M('User');
$user->delete('1,3,5');

//删除count为0且按时间倒序的前五个
$user = M('User');
$map['count'] = 0;
$user->where($map)->order(array('date'=>'DESC'))->limit(5)->
                                                    delete();

//删除所有数据，谨慎
$user = M('User');
echo $user->where('1')->delete();
```

delete()方法支持的连贯操作有：

1.where，查询或更新条件；
2.table，要操作的数据表名称；
3.alias，数据表别名；
4.order，结果排序；
5.lock，锁；
6.relation，关联查询；
7.scope，命名范围；
8.bind，数据绑定操作；
9.comment，SQL 注释。

六. ActiveReocrd 模式

这种模式最大的特别就是简化了 CURD 的操作，并且采用对象化的操作方式，便于使用和理解。

```
//添加一条数据
$user = M('User');
$user->user = '火影忍者';
$user->email = 'huoyin@qq.com';
$user->date = date('Y-m-d H:i:s');
```

```php
$user->add();

//结合create
$user = M('User');
$user->create();
$user->date = date('Y-m-d H:i:s');
$user->add();

//找到主键为4的值
$user = M('User');
var_dump($user->find(4));

//查找user=蜡笔小新的记录
$user = M('User');
var_dump($user->getByUser('蜡笔小新'));

//输出user
echo $user->user;

//通过主键查询多个
$user = M('User');
var_dump($user->select('1,2,3'));

//修改一条数据
$user = M('User');
$user->find(1);
$user->user = '蜡笔老新';
$user->save();

//删除当前找到的数据
$user = M('User');
$user->find(11);
$user->delete();

//删除主键为10的数据
$user = M('User');
$user->delete(10);

//删除主键为10,11的数据
$user = M('User');
$user->delete('10,11');
```

七. 字段映射

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

字段映射可以将表单里的 name 名称对应到数据表里的字段，这样防止系统自动屏蔽掉不对应的 POST 值。

```
//字段映射
protected $_map = array(
    'xingming'=>'user',
    'youxiang'=>'email',
);

//字段映射获取
$user = D('User');
var_dump($user->create());
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--自动验证

学习要点：

1.验证规则

2.静态定义

3.动态验证

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 模型层提供的一种数据验证方法，可以在使用 create 创建数据对象的时候进行自动验证。

一. 验证规则

数据验证可以对表单中的字段进行非法的验证操作。一般提供了两种验证方式：静态定义（$_validate 属性）和动态验证（validate()方法）。

```
//验证规则
array(
    array(验证字段1,验证规则,错误提示,[验证条件,附加规则,验证时间]),
    array(验证字段2,验证规则,错误提示,[验证条件,附加规则,验证时间]),
    ......
);
```

PS：验证字段、验证规则和错误提示这三项是必选的，大多数也是用这三项；而验证条件、附加规则和验证时间是可选的。

验证字段：一般来说是表单的字段名，不一定必须和数据表匹配的，因为有一些比如密码确认等辅助字段的存在。

验证规则：系统内置了常用的规则，require(字段必填)、email(邮箱格式)、url(url 格式)、currency(货币)、number(正整数)、integer(整数)、double(浮点数)、zip(邮政编码)、english(英文)。这些规则默认采用的附加规则是 regex，正则表达式验证，只不过是设定好的。

错误信息：验证失败后的提示。

验证条件：共三种：

1.self::EXISTS_VALIDATE 或 0，表示存在字段就验证（默认）；

2.self::MUST_VALIDATE 或 1，表示必须验证；

3.self::VALUE_VALIDATE 或 2，表示值不为空的时候验证。

附加规则：配合验证规则使用，包括一下规则：

规则	说明
regex	正则验证，定义的验证规则是一个正则表达式（默认）
function	函数验证，定义的验证规则是一个函数名

callback	方法验证，定义的验证规则是当前模型类的一个方法
confirm	验证表单中的两个字段是否相同，定义的验证规则是一个字段名
equal	验证是否等于某个值，该值由前面的验证规则定义
notequal	验证是否不等于某个值，该值由前面的验证规则定义（3.1.2版本新增）
in	验证是否在某个范围内，定义的验证规则可以是一个数组或者逗号分割的字符串
notin	验证是否不在某个范围内，定义的验证规则可以是一个数组或者逗号分割的字符串（3.1.2版本新增）
length	验证长度，定义的验证规则可以是一个数字（表示固定长度）或者数字范围（例如3,12 表示长度从3到12的范围）
between	验证范围，定义的验证规则表示范围，可以使用字符串或者数组，例如1,31或者 array(1,31)
notbetween	验证不在某个范围，定义的验证规则表示范围，可以使用字符串或者数组（3.1.2版本新增）
expire	验证是否在有效期，定义的验证规则表示时间范围，可以到时间，例如可以使用 2012-1-15,2013-1-15 表示当前提交有效期在2012-1-15到2013-1-15之间，也可以使用时间戳定义
ip_allow	验证 IP 是否允许，定义的验证规则表示允许的 IP 地址列表，用逗号分隔，例如201.12.2.5,201.12.2.6
ip_deny	验证 IP 是否禁止，定义的验证规则表示禁止的 ip 地址列表，用逗号分隔，例如201.12.2.5,201.12.2.6
unique	验证是否唯一，系统会根据字段目前的值查询数据库来判断是否存在相同的值，当表单数据中包含主键字段时 unique 不可用于判断主键字段本身

验证时间：主要新增修改等验证。

1.self::MODEL_INSERT 或 1 新增数据时验证；

2.self::MODEL_UPDATE 或 2 编辑数据时验证；

3.self::MODEL_BOTH 或 3 全部情况下验证(默认)。

二. 静态定义

在模型类里预先定义好该模型的自动验证规则，就是静态定义。

```
//模型类定义验证规则
class UserModel extends Model {
    protected $_validate = array(
        //默认情况下用系统内置
        array('user', 'require', '用户不得为空！'),
        //验证数据唯一性
```

```
        array('user', '', '用户名称已存在！',2,'unique',1),
        //验证密码不得为空
        array('pass', 'require', '密码不得为空！'),
        //密码不得小于六位不得大于20位
        array('pass', '6,20', '密码不得小于6位，不得大于20位', 3,
                                                        'length'),
    );
}
```

为了测试方便，我们可以直接通过模拟提交 POST：

```
//控制器create()方法自动调用验证
$user = D('User');
$data['user'] = '蜡笔小新';
$data['pass'] = '123';
if ($user->create($data)) {
    echo '所有数据验证成功！';
} else {
    //输出错误信息
    var_dump($user->getError());
}
```

ThinkPHP 提供了九种自动验证内置方案，具体如下：

```
//内置验证require，不得为空的用法
array('user', 'require', '用户不得为空！'),

//内置验证email，合法的邮箱格式
array('user', 'email', '邮箱格式不合法！'),

//内置验证url，验证网址是否合法
array('user', 'url', 'URL 路径不合法！'),

//内置验证currency，验证是否为货币
array('user', 'currency', '货币格式不正确！'),

//内置验证zip，验证是否为六位整数邮政编码
array('user', 'zip', '邮政编码格式不正确！'),

//内置验证number，验证是否为正整数
array('user', number, '正整数格式不正确！'),

//内置验证integer，验证是否为整数，正负均可
array('user', 'integer', '整数格式不正确！'),
```

北风网 IBEIFENG.COM
Ycku.com 瓢城Web俱乐部，做最酷的教学

```
//内置验证double，验证是否为浮点数，正负均可
array('user', 'double', '整数格式不正确！'),

//内置验证english，验证是纯英文
array('user', 'english', '不是纯英文！'),
```

ThinkPHP 还提供了附加规则，来提升自动验证的扩展性：

```
//附加规则regex，验证3-6位纯数字
array('user', '/^\d{3,6}$/', '不是 3-6 位纯正数字', 0, 'regex'),

//附加规则equal，验证是否和指定值相等
array('user', '李炎恢', '值不对等', 0, 'equal'),

//附加规则notequal，验证是否与指定值不等
array('user', '李炎恢', '值不能相等', 0, 'notequal'),

//附加规则confirm，验证两条字段是否相同
array('user', 'name', '两个用户名对比不同！',0,'confirm'),

//附加规则in，某个范围，可以是数组或逗号分割的字符串
array('user', array(1,2,3), '不在指定范围', 0, 'in'),
array('user', '张三,李四,王五', '不在指定范围', 0, 'in'),

//附加规则notin，某个范围，可以是数组或逗号分割的字符串
array('user', array(1,2,3), '不得在指定范围', 0, 'notin'),
array('user', '张三,李四,王五', '不得在指定范围', 0, 'notin'),

//附加规则length，验证长度或数字范围
array('user', '3', '不得小于 3 位', 0, 'length'),
array('user', '3,5', '不得小于 3 位，不得大于 5 位', 0, 'length'),

//附加规则between，验证某个范围，数字或逗号字符串
array('user', array(3,5), '必须是 3-5 之间的数字', 0, 'between'),
array('user', '3,5', '必须是 3-5 之间的数字', 0, 'between'),

//附加规则notbetween，验证某个范围，数字或逗号字符串
array('user', array(3,5), '必须不是 3-5 之间的数字', 0, 'notbetween'),
array('user', '3,5', '必须不是 3-5 之间的数字', 0, 'notbetween'),

//附加规则expire，设置有效期范围，必须是表单提交有效，可以是时间戳
array('user', '2014-1-10,2015-10-10', '时间已过期', 0, 'expire'),

//附加规则ip_deny，IP禁止列表
```

北风网 IBEIFENG.COM
Ycku.com 瓢城Web俱乐部，做最酷的教学

```
    array('user', '127.0.0.1', '当钱 IP 被禁止', 0, 'ip_deny'),

    //附加规则ip_allow，IP允许列表
    array('user', '127.0.0.1', '当前 IP 没有被允许', 0, 'ip_allow'),

    //附加规则callback，回调验证
    array('user', 'checkLength', '用户名必须在 3-5 位', 0, 'callback', 3,
array(3,5)),

    //回调方法
    protected function checkLength($str,$min,$max) {
        preg_match_all("/./", $str, $matches);
        $len = count($matches[0]);
        if ($len < $min || $len > $max) {
            return false;
        } else {
            return true;
        }
    }

    //附加规则function，函数验证
    array('user', 'checkLength', '用户名必须在 3-5 位', 0, 'function', 3,
array(3,5)),

    //在 Common 文件夹下的 Common 文件夹建立 function.php 文件，会自动加载
    function checkLength($str,$min,$max) {
        preg_match_all("/./", $str, $matches);
        $len = count($matches[0]);
        if ($len < $min || $len > $max) {
            return false;
        } else {
            return true;
        }
    }
```

如果有多个字段都包含错误，默认只显示一个错误。如果想显示全部错误，可以设置属性：

```
    //批量验证
    protected $patchValidate = true;
```

如果是直接 POST 过来的，直接使用 create()方法即可。

```
    //控制器create()方法自动调用验证
    $user = D('User');
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
if ($user->create()) {
    echo '所有数据验证成功！';
} else {
    //输出错误信息
    var_dump($user->getError());
}
```

PS：由于使用的 UserModel 模型类，所以必须是 D()方法实例化。

如果想把错误信息返回给ajax处理，可以是同ajaxReturn()方法返回JSON数据。

```php
//返回JSON格式
$this->ajaxReturn($user->getError());

//1指定新增数据验证，2表示修改，
if ($user->create($_POST,1)) {}                    //一般会自动判断
```

三．动态验证

动态验证就是把验证的规则放在控制器端，这样，在操作的时候比较灵活，缺点就是比较混乱。

```php
//动态验证
$rule = array(
        array('user', 'require', '用户名不得为空'),
);
$user = M('User');
$data['user'] = '';
if ($user->validate($rule)->create($data)) {
    echo '验证所有字段成功！';
} else {
    var_dump($user->geterror());
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--自动验证

学习要点：

1.验证规则

2.静态定义

3.动态验证

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 模型层提供的一种数据验证方法，可以在使用 create 创建数据对象的时候进行自动验证。

一. 验证规则

数据验证可以对表单中的字段进行非法的验证操作。一般提供了两种验证方式：静态定义（$_validate 属性）和动态验证（validate()方法）。

```
//验证规则
array(
    array(验证字段1,验证规则,错误提示,[验证条件,附加规则,验证时间]),
    array(验证字段2,验证规则,错误提示,[验证条件,附加规则,验证时间]),
    ......
);
```

PS：验证字段、验证规则和错误提示这三项是必选的，大多数也是用这三项；而验证条件、附加规则和验证时间是可选的。

验证字段：一般来说是表单的字段名，不一定必须和数据表匹配的，因为有一些比如密码确认等辅助字段的存在。

验证规则：系统内置了常用的规则，require(字段必填)、email(邮箱格式)、url(url 格式)、currency(货币)、number(正整数)、integer(整数)、double(浮点数)、zip(邮政编码)、english(英文)。这些规则默认采用的附加规则是 regex，正则表达式验证，只不过是设定好的。

错误信息：验证失败后的提示。

验证条件：共三种：

1.self::EXISTS_VALIDATE 或 0，表示存在字段就验证（默认）；

2.self::MUST_VALIDATE 或 1，表示必须验证；

3.self::VALUE_VALIDATE 或 2，表示值不为空的时候验证。

附加规则：配合验证规则使用，包括一下规则：

规则	说明
regex	正则验证，定义的验证规则是一个正则表达式（默认）
function	函数验证，定义的验证规则是一个函数名

callback	方法验证，定义的验证规则是当前模型类的一个方法
confirm	验证表单中的两个字段是否相同，定义的验证规则是一个字段名
equal	验证是否等于某个值，该值由前面的验证规则定义
notequal	验证是否不等于某个值，该值由前面的验证规则定义（3.1.2版本新增）
in	验证是否在某个范围内，定义的验证规则可以是一个数组或者逗号分割的字符串
notin	验证是否不在某个范围内，定义的验证规则可以是一个数组或者逗号分割的字符串（3.1.2版本新增）
length	验证长度，定义的验证规则可以是一个数字（表示固定长度）或者数字范围（例如3,12 表示长度从3到12的范围）
between	验证范围，定义的验证规则表示范围，可以使用字符串或者数组，例如1,31或者 array(1,31)
notbetween	验证不在某个范围，定义的验证规则表示范围，可以使用字符串或者数组（3.1.2版本新增）
expire	验证是否在有效期，定义的验证规则表示时间范围，可以到时间，例如可以使用 2012-1-15,2013-1-15 表示当前提交有效期在2012-1-15到2013-1-15之间，也可以使用时间戳定义
ip_allow	验证 IP 是否允许，定义的验证规则表示允许的 IP 地址列表，用逗号分隔，例如201.12.2.5,201.12.2.6
ip_deny	验证 IP 是否禁止，定义的验证规则表示禁止的 ip 地址列表，用逗号分隔，例如201.12.2.5,201.12.2.6
unique	验证是否唯一，系统会根据字段目前的值查询数据库来判断是否存在相同的值，当表单数据中包含主键字段时 unique 不可用于判断主键字段本身

验证时间：主要新增修改等验证。

1.self::MODEL_INSERT 或 1 新增数据时验证；

2.self::MODEL_UPDATE 或 2 编辑数据时验证；

3.self::MODEL_BOTH 或 3 全部情况下验证(默认)。

二. 静态定义

在模型类里预先定义好该模型的自动验证规则，就是静态定义。

```
//模型类定义验证规则
class UserModel extends Model {
    protected $_validate = array(
        //默认情况下用系统内置
        array('user', 'require', '用户不得为空！'),
        //验证数据唯一性
```

```
        array('user', '', '用户名称已存在！',2,'unique',1),
        //验证密码不得为空
        array('pass', 'require', '密码不得为空！'),
        //密码不得小于六位不得大于20位
        array('pass', '6,20', '密码不得小于6位，不得大于20位', 3,
                                                        'length'),
    );
}
```

为了测试方便，我们可以直接通过模拟提交 POST：

```
//控制器create()方法自动调用验证
$user = D('User');
$data['user'] = '蜡笔小新';
$data['pass'] = '123';
if ($user->create($data)) {
    echo '所有数据验证成功！';
} else {
    //输出错误信息
    var_dump($user->getError());
}
```

ThinkPHP 提供了九种自动验证内置方案，具体如下：

```
//内置验证require，不得为空的用法
array('user', 'require', '用户不得为空！'),

//内置验证email，合法的邮箱格式
array('user', 'email', '邮箱格式不合法！'),

//内置验证url，验证网址是否合法
array('user', 'url', 'URL 路径不合法！'),

//内置验证currency，验证是否为货币
array('user', 'currency', '货币格式不正确！'),

//内置验证zip，验证是否为六位整数邮政编码
array('user', 'zip', '邮政编码格式不正确！'),

//内置验证number，验证是否为正整数
array('user', number, '正整数格式不正确！'),

//内置验证integer，验证是否为整数，正负均可
array('user', 'integer', '整数格式不正确！'),
```

北风网 IBEIFENG.COM
Ycku.com 瓢城Web俱乐部，做最酷的教学

```
//内置验证double，验证是否为浮点数，正负均可
array('user', 'double', '整数格式不正确！'),

//内置验证english，验证是纯英文
array('user', 'english', '不是纯英文！'),
```

ThinkPHP 还提供了附加规则，来提升自动验证的扩展性：

```
//附加规则regex，验证3-6位纯数字
array('user', '/^\d{3,6}$/', '不是 3-6 位纯正数字', 0, 'regex'),

//附加规则equal，验证是否和指定值相等
array('user', '李炎恢', '值不对等', 0, 'equal'),

//附加规则notequal，验证是否与指定值不等
array('user', '李炎恢', '值不能相等', 0, 'notequal'),

//附加规则confirm，验证两条字段是否相同
array('user', 'name', '两个用户名对比不同！',0,'confirm'),

//附加规则in，某个范围，可以是数组或逗号分割的字符串
array('user', array(1,2,3), '不在指定范围', 0, 'in'),
array('user', '张三,李四,王五', '不在指定范围', 0, 'in'),

//附加规则notin，某个范围，可以是数组或逗号分割的字符串
array('user', array(1,2,3), '不得在指定范围', 0, 'notin'),
array('user', '张三,李四,王五', '不得在指定范围', 0, 'notin'),

//附加规则length，验证长度或数字范围
array('user', '3', '不得小于 3 位', 0, 'length'),
array('user', '3,5', '不得小于 3 位，不得大于 5 位', 0, 'length'),

//附加规则between，验证某个范围，数字或逗号字符串
array('user', array(3,5), '必须是 3-5 之间的数字', 0, 'between'),
array('user', '3,5', '必须是 3-5 之间的数字', 0, 'between'),

//附加规则notbetween，验证某个范围，数字或逗号字符串
array('user', array(3,5), '必须不是 3-5 之间的数字', 0, 'notbetween'),
array('user', '3,5', '必须不是 3-5 之间的数字', 0, 'notbetween'),

//附加规则expire，设置有效期范围，必须是表单提交有效，可以是时间戳
array('user', '2014-1-10,2015-10-10', '时间已过期', 0, 'expire'),

//附加规则ip_deny，IP禁止列表
```

北风网 IBEIFENG.COM
Ycku.com 瓢城Web俱乐部，做最酷的教学

```
    array('user', '127.0.0.1', '当前 IP 被禁止', 0, 'ip_deny'),

    //附加规则ip_allow，IP允许列表
    array('user', '127.0.0.1', '当前 IP 没有被允许', 0, 'ip_allow'),

    //附加规则callback，回调验证
    array('user', 'checkLength', '用户名必须在 3-5 位', 0, 'callback', 3,
array(3,5)),

    //回调方法
    protected function checkLength($str,$min,$max) {
        preg_match_all("/./u", $str, $matches);
        $len = count($matches[0]);
        if ($len < $min || $len > $max) {
            return false;
        } else {
            return true;
        }
    }

    //附加规则function，函数验证
    array('user', 'checkLength', '用户名必须在 3-5 位', 0, 'function', 3,
array(3,5)),

    //在 Common 文件夹下的 Common 文件夹建立 function.php 文件，会自动加载
    function checkLength($str,$min,$max) {
        preg_match_all("/./u", $str, $matches);
        $len = count($matches[0]);
        if ($len < $min || $len > $max) {
            return false;
        } else {
            return true;
        }
    }
```

如果有多个字段都包含错误，默认只显示一个错误。如果想显示全部错误，可以设置属性：

```
    //批量验证
    protected $patchValidate = true;
```

如果是直接 POST 过来的，直接使用 create()方法即可。

```
    //控制器create()方法自动调用验证
    $user = D('User');
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
if ($user->create()) {
    echo '所有数据验证成功！';
} else {
    //输出错误信息
    var_dump($user->getError());
}
```

PS：由于使用的 UserModel 模型类，所以必须是 D()方法实例化。

如果想把错误信息返回给ajax处理，可以是同ajaxReturn()方法返回JSON数据。

```php
//返回JSON格式
$this->ajaxReturn($user->getError());

//1指定新增数据验证，2表示修改，
if ($user->create($_POST,1)) {}                    //一般会自动判断
```

三．动态验证

动态验证就是把验证的规则放在控制器端，这样，在操作的时候比较灵活，缺点就是比较混乱。

```php
//动态验证
$rule = array(
        array('user', 'require', '用户名不得为空'),
);
$user = M('User');
$data['user'] = '';
if ($user->validate($rule)->create($data)) {
    echo '验证所有字段成功！';
} else {
    var_dump($user->geterror());
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--自动完成

学习要点：

1.完成规则

2.静态定义

3.动态完成

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 模型层提供的数据处理方法，主要用于数据的自动处理和过滤，使用 create()方法创建数据是会自动完成。

一．完成规则

自动完成一般通过默认字段写入、安全字段过滤以及业务逻辑的自动处理等。有两种方式实现自动完成的规则：1.静态方式：在模型类里通过$_auto 属性定义处理规则；2 动态方式：使用模型类的 auto 方法动态创建自动处理规则。

```
//完成规则
array(
    array(完成字段1,完成规则,[完成条件,附加规则]),
    array(完成字段2,完成规则,[完成条件,附加规则]),
    ......
);
```

完成字段：必填，需要的字段名；

完成规则：必填，配合附加规则完成；

完成条件：可选，具体如下：

1.self::MODEL_INSERT 或 1，新增数据的时候处理（默认）；

2.self::MODEL_UPDATE 或 2，更新数据的时候处理；

3.self::MODEL_BOTH 或 3，所有情况均处理。

附加规则：可选，配合完成规则使用，包括一下规则：

规则	说明
function	函数完成，定义的验证规则是一个函数名
callback	方法完成，定义的验证规则是当前模型类的一个方法
field	用其他字段填充，表示填充的内容是一个其他字段的值
string	字符串（默认）
ignore	为空则忽略（3.1.2新增）

二. 静态定义

在模型类里预先定义好该模型的自动完成规则，就是静态定义。

```
class UserModel extends Model {
    //自动完成
    protected $_auto = array(
        //自动设置count字段为1
        array('count', '1'),
        //给密码加密，加密类型为sha1，sha1函数PHP内置
        array('user', 'sha1', 3, 'function'),
    );
}
```

为了测试方便，我们可以直接通过模拟提交 POST：

```
//控制器create()方法自动调用验证
$user = D('User');
$data['user'] = '蜡笔小新';
if ($user->create($data)) {
    $user->add();
}

//string,自动设置count字段为1
array('count', '1'),

//function，给密码加密，加密类型为sha1，sha1函数PHP内置
array('user', 'sha1', 3, 'function'),

//把email字段的值填充到user字段冲去
array('user', 'email', 3, 'field'),

//callback，给用户名加前缀
array('user', 'updateUser', 3, 'callback', '_'),

//回调函数
protected function updateUser($str, $prefix) {
    return $prefix.$str;
}

//ignore，用于修改时密码留空时，忽略修改
array('pass', '', 2, 'ignore'),
```

三. 动态完成

动态完成就是把完成的规则放在控制器端，这样，在操作的时候比较灵活，缺点就是比较混乱。

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
//动态完成
$rules = array(
    array('user', 'sha1', 3, 'function'),
);
$user = M('User');
$data['user'] = '蜡笔小新';
if ($user->auto($rules)->create($data)) {
    $user->add();
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--视图

学习要点：

1.模版定义
2.赋值和渲染
3.模版地址
4.获取内容

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 视图，视图是 Web 的可见内容，一般是 HTML 结合 PHP 获取的数据提供给用户使用的部分，属于 MVC 中的 V。

一. 模版定义

模版在使用之前需要一定的设置，才能方便开发者使用。每个模块的模版文件是独立的，为了对模版文件更加有效的管理，ThinkPHP 对模版文件机型目录划分，默认的模版文件定义规则是：

视图目录/[模版主题/]控制器名/操作名+模版后缀

第一步：在 User 控制器模块执行一条语句：

```
//渲染模版输出
$this->display();
```

什么都没有定义的情况下，会自动报错，错误信息提示如下：

模板不存在:./Weibo/Home/View/User/index.html

通过这个错误提示，我们只要在 View 目录下创建 User 目录，并在 User 目录下创建 index.html 文件，这时模版渲染模版输出成功。

默认情况下视图目录是 View，如果你想改变成别的目录名可以设置：

```
//修改模版的视图目录
'DEFAULT_V_LAYER' =>'Template',
```

默认情况下的模版文件后缀是.html，如果你想改变成别的后缀可以设置：

```
//修改模版中的文件的后缀
'TMPL_TEMPLATE_SUFFIX'=>'.tpl'
```

如果感觉每一个模块，都要创建相应的目录太过于麻烦，可以设置：

```
//用下划线代替目录层次
'TMPL_FILE_DEPR'=>'_',                    //User_index.tpl
```

如果不想将模版存在在当前 Weibo 目录下，而设置在外部：

```
//设置外部的模版目录
'VIEW_PATH'=>'./Public/',
```

如果一个系统要考虑多套界面皮肤的话，要考虑到默认皮肤以及可选皮肤：

```
//设置默认主题目录
'DEFAULT_THEME'=>'default',

//切换主题
$this->theme('blue')->display();
```

二．赋值和渲染

如果要在模版中输出变量，必须在控制器中把变量传递给模版。ThinkPHP 提供了 assign 方法对模版变量赋值，无论何种变量类型都统一使用 assign 赋值。

```
//给模版传递一个变量
$this->assign('user', '蜡笔小新');
//模版中调用变量
{$user}
```

PS：这个方法必须在 display()方法之前使用，保存变量正确传递。

渲染模版输出使用的是 display 方法，有三个可选参数：
display([模版文件][,字符编码][,输出类型]);

如果不传递任何参数，它会按照默认的目录定位模版的位置：
当前模块/默认视图目录/当前控制器/当前操作.html

```
//不传参数
$this->display();
```

./Weibo/Home/View/default/User/index.tpl

```
//修改默认模版
$this->display('add');
```

./Weibo/Home/View/default/User/add.tpl

```
//修改默认模版，目录加模版
$this->display('Bbb/add');
```

./Weibo/Home/View/default/Bbb/add.tpl

```
//修改默认模版，模块加目录加模版
$this->display('Admin@Bbb/add');
```

./Weibo/Admin/View/default/Bbb/add.tpl

```
//修改默认模版，主题加目录加模版
$this->theme('blue')->display('Bbb/add');
```

./Weibo/Home/View/blue/Bbb/add.tpl

```
//修改默认模版，自定义模版
$this->display('./Template/Public/add.tpl');
```

./Template/Public/add.tpl　　//Template 和 Weibo 同级

```
//修改默认模版，指定编码和文件类型，一般不用填写，默认即可
$this->display('add', 'utf-8', 'text/xml');
```

三．模版地址

ThinkPHP 封装了一个 T 函数，专门用于生成模版文件。格式如下：

```
T([资源://][模块@][主题/][控制器/]操作,[视图分层]);
//输出当前模版地址
echo T();
```

当然可以结合上面讲的所有定义方法来自定义模版：

```
echo T('Public/add')
echo T('Admin@index');
echo T('Admin@Public/add', 'Template');

//直接使用T函数输出
$this->display(T());
```

四．获取内容

如果需要获取模版的内容，可以使用 fetch()方法，这个方法的使用和 display()方法一致。

```
//获取模版里的内容
$content = $this->fetch();
var_dump($content);

//通过内容再渲染输出
$this->show($content);
```

PS：使用 fetch()方法获取内容，主要是为了可以处理和过滤更加复杂的内容。然后处理后再由 show()方法输出。

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--模版基础

学习要点：

1.变量输出
2.系统变量
3.使用函数
4.使用运算符
5.包含文件
6.模版注释
7.模版继承
8.模版布局
9.模版替换

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 模版，ThinkPHP 内置了一个基于 XML 的性能卓越的模版引擎 ThinkTemplate，使用了动态编译和缓存技术，支持自定义标签库。

一. 变量输出

在模版中输出变量是非常容易的，使用 assign()方法，以键值对的方式传递变量和值。

```
//给模版传递一个变量
$user = '蜡笔小新';
$this->assign('user', $user);
```

在模版中可以这么调用：

```
{$user}
```

模版编译后的结果是：

```
<?php echo ($user); ?>                    //可以在 Runtime 可以查看
```

模版标签{和$之间不能用任何空格，否则无法解析。如果你想更换两个{}可以设置：

```
//修改默认标签
'TMPL_L_DELIM'=>'<{',
'TMPL_R_DELIM'=>'}>',
```

调用方式就必须改变：

```
<{$user}>
```

如果传递一个数组，我们直接传递过去后，通过两种方式调用：

```
//给模版传递一个数组
$data['user'] = '蜡笔小新';
```

```
$data['email'] = 'bnbbs@163.com';
$this->assign('data', $data);
```

调用方式为：

```
User:{$data.user} Email:{$data.email}
User:{$data['user']} Email:{$data['email']}
```

如果传递一个对象，我们直接传递过去后，通过两种方式调用：

```
//给模版传递一个对象
$data = new \stdClass();
$data->user = '蜡笔小新';
$data->email = 'bnbbs@163.com';
$this->assign('data', $data);
```

调用方式为：

```
User:{$data->user} Email:{$data->email}
User:{$data:user} Email:{$data:email}
```

二. 系统变量

在模版中，不但可以输出 PHP 的系统变量，还可以输出 ThinkPHP 的系统变量。

```
//输出PHP系统变量
{$Think.server.script_name}          //$_SERVER['SCRIPT_NAME']
{$Think.session.admin}               //$_SESSION['admin']
{$Think.get.user}                    //$_GET['user']
{$Think.post.user}                   //$_POST['user']
{$Think.request.user}                //$_REQUEST['user']
{$Think.cookie.name}                 //$_COOKIE['name']

//ThinkPHP的系统变量
{$Think.const.APP_PATH}              //目录
{$Think.config.url_model}            //URL模式
{$Think.lang.var_error}              //语言变量
```

三. 使用函数

如果有时，我们需要在模版中使用 PHP 函数的话，可以按照下面的格式使用：

```
{$user|md5}
```

如果有多个参数要传递，可以参考如下设置：

```
{$date|date="Y-m-d H:i:s",###}
```

PS：表示 date 函数传入两个参数，每个参数用逗号分割，这里第一个参数是 Y-m-d H:i:s，第二个参数是前面要输出的 date 变量，因为该变量是第二个参数，因此需要用### 标识变量位置，编译后的结果是：

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
<?php echo (date($date,"Y-m-d H:i:s")); ?>
```

前面输出变量，在后面定义，则不需要###

```
{$user|mb_substr=0,3,'UTF-8'}
```

多个函数用"|"隔开即可

```
{$user|mb_substr=0,3,'UTF-8'|md5|sha1}
```

PS：如果你觉得以上写法需要在脑海里二次翻译，太过于麻烦，那么可以用以下的格式写法：

```
{:md5(mb_substr($user,0,3,'UTF-8'))}
```

PS：我个人认为把这些数据处理，都尽可能不要模版端，除非是必须在模版端才能处理的。因为模版就是为了 HTML 和 JS，然后就是提供数据，不要包含各种过滤和业务逻辑。

当传递过来的变量如果没有值的时候，模版提供了默认值输出功能。

```
{$user|default='什么都没有！'}
```

四．使用运算符

我们可以在模版中使用运算符，包括对“+”、“-”、“*”、“/”、“%”、“--”和“++”的支持。

```
{$data['num']+10}                    //中括号
{$data['num']+getNum()}              //函数
```

PS：在使用运算符的时候，不再支持语法和常规的函数用法：

```
{$data.num+10}                       //错误的
{$data['num']+getNum}                //错误
```

模版还支持三元运算符：

```
{$user ? '有值' : '无值'}
```

五．包含文件

在一个系统中，可以包含通用的头文件和脚文件：header 和 footer。由于每个页面的头脚都是相同的，所以需要独立分离出来，再用包含文件引入他们。

我们可以在 View 目录下 default 主题目录下创建一个 public 目录，这个目录专门存放公共调用模版文件。

```
<include file='Public/header' />
<include file='Public/footer' />
```

也可以调用绝对路径模式：

```
<include file='./Weibo/Home/View/default/Public/header.tpl' />
```

同时调用多个模版文件：

```
<include file='Public/header,Public/footer' />
```

六．模版注释

模版支持注释功能，提供个模版制作人员参考。

```
{//这是注释}
{/*这也是注释*/}
{/*这是
    多行注释*/}
```

七．模版继承

模版继承是一项灵活的模版布局方式，它类似于子类继承父类，然后子类还可以进行适当的修改以满足当前页面的需要。

```
//基础模版需要子模版修改的部分可以用<block>包含，并且设置name值
<block name="main">主要内容</block>

//子模版只要按同样的方法，修改<block>内部的值即可完成修改
<block name="main">修改了</block>

//创建Public下base.tpl模版基页
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<block name="title"><title>{$user}</title></block>
</head>
<body>
    <include file="Public/header" />
    <block name="main">主要内容</block>
    <include file="Public/footer" />
</body>
</html>

//使用<extend>导入模版基页，name为路径，和include导入方法一致
<extend name="Public/base" />
<block name="main">修改了</block>
```

八．模版布局

ThinkPHP 的模版引擎内置了布局模版功能支持，可以方便实现模版布局以及布局嵌套功能。有三种布局方式：

1.全局配置方式

```
//开启模版布局功能，并指定基础页
'LAYOUT_ON'=>true,
'LAYOUT_NAME'=>'Public/layout',                    //layout.tpl 文件
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
//基础页，{__CONTENT__}将被子页面的内容替换
<include file="Public/header" />
{__CONTENT__}
<include file="Public/footer" />

//替换变量可以变更为{__REPLACE__}
'TMPL_LAYOUT_ITEM' =>'{__REPLACE__}',

//子模版不需要载入模版基页，可以在开头加上{__NOLAYOUT__}
{__NOLAYOUT__}
```

2.模版标签方式

标签方式，并不需要在系统做任何配置，和模版继承类似，直接引入即可。

```
//子模版引入模版基页
<layout name="Public/layout" />
//替换变量的方法
<layout name="Public/layout" replace="{__REPLACE__}" />
```

3.layout 控制布局

这个方法是在控制器里操作的。

```
//开启布局，并引入默认地址基页
public function index() {
    layout(true);
}

//引入指定基页
public function index() {
    layout('Public/layout');
    //layout(false);                                //关闭
}
```

九．模版替换

在模版渲染之前，系统还会对读取的模版内容进行一些特殊字符串替换操作，也就实现了模版输出的替换和过滤。

__ROOT__： 会替换成当前网站的地址（不含域名）

__APP__： 会替换成当前应用的 URL 地址 （不含域名）

__MODULE__：会替换成当前模块的 URL 地址 （不含域名）

__CONTROLLER__（或者__URL__ 兼容考虑）： 会替换成当前控制器的 URL 地址（不含域名）

__ACTION__：会替换成当前操作的 URL 地址 （不含域名）

__SELF__： 会替换成当前的页面 URL

__PUBLIC__：会被替换成当前网站的公共目录 通常是 /Public/

```
'TMPL_PARSE_STRING'  =>array(
       '__PUBLIC__' => '/Common', // 更改默认的/Public 替换规则
       '__UPLOAD__' => '/Uploads', // 增加新的上传路径替换规则
)
```

PS：__PUBLIC__可以改成--PUBLIC--同样的也可以。

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--模版基础

学习要点：

1.变量输出
2.系统变量
3.使用函数
4.使用运算符
5.包含文件
6.模版注释
7.模版继承
8.模版布局
9.模版替换

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 模版，ThinkPHP 内置了一个基于 XML 的性能卓越的模版引擎 ThinkTemplate，使用了动态编译和缓存技术，支持自定义标签库。

一. 变量输出

在模版中输出变量是非常容易的，使用 assign()方法，以键值对的方式传递变量和值。

```
//给模版传递一个变量
$user = '蜡笔小新';
$this->assign('user', $user);
```

在模版中可以这么调用：

```
{$user}
```

模版编译后的结果是：

```
<?php echo ($user); ?>                    //可以在 Runtime 可以查看
```

模版标签{和$之间不能用任何空格，否则无法解析。如果你想更换两个{}可以设置：

```
//修改默认标签
'TMPL_L_DELIM'=>'<{',
'TMPL_R_DELIM'=>'}>',
```

调用方式就必须改变：

```
<{$user}>
```

如果传递一个数组，我们直接传递过去后，通过两种方式调用：

```
//给模版传递一个数组
$data['user'] = '蜡笔小新';
```

```
$data['email'] = 'bnbbs@163.com';
$this->assign('data', $data);
```

调用方式为：

```
User:{$data.user} Email:{$data.email}
User:{$data['user']} Email:{$data['email']}
```

如果传递一个对象，我们直接传递过去后，通过两种方式调用：

```
//给模版传递一个对象
$data = new \stdClass();
$data->user = '蜡笔小新';
$data->email = 'bnbbs@163.com';
$this->assign('data', $data);
```

调用方式为：

```
User:{$data->user} Email:{$data->email}
User:{$data:user} Email:{$data:email}
```

二. 系统变量

在模版中，不但可以输出 PHP 的系统变量，还可以输出 ThinkPHP 的系统变量。

```
//输出PHP系统变量
{$Think.server.script_name}          //$_SERVER['SCRIPT_NAME']
{$Think.session.admin}               //$_SESSION['admin']
{$Think.get.user}                    //$_GET['user']
{$Think.post.user}                   //$_POST['user']
{$Think.request.user}                //$_REQUEST['user']
{$Think.cookie.name}                 //$_COOKIE['name']

//ThinkPHP的系统变量
{$Think.const.APP_PATH}              //目录
{$Think.config.url_model}            //URL模式
{$Think.lang.var_error}              //语言变量
```

三. 使用函数

如果有时，我们需要在模版中使用 PHP 函数的话，可以按照下面的格式使用：

```
{$user|md5}
```

如果有多个参数要传递，可以参考如下设置：

```
{$date|date="Y-m-d H:i:s",###}
```

PS：表示 date 函数传入两个参数，每个参数用逗号分割，这里第一个参数是 Y-m-d H:i:s，第二个参数是前面要输出的 date 变量，因为该变量是第二个参数，因此需要用### 标识变量位置，编译后的结果是：

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
<?php echo (date($date,"Y-m-d H:i:s")); ?>
```

前面输出变量，在后面定义，则不需要###

```
{$user|mb_substr=0,3,'UTF-8'}
```

多个函数用"|"隔开即可

```
{$user|mb_substr=0,3,'UTF-8'|md5|sha1}
```

PS：如果你觉得以上写法需要在脑海里二次翻译，太过于麻烦，那么可以用以下的格式写法：

```
{:md5(mb_substr($user,0,3,'UTF-8'))}
```

PS：我个人认为把这些数据处理，都尽可能不要模版端，除非是必须在模版端才能处理的。因为模版就是为了 HTML 和 JS，然后就是提供数据，不要包含各种过滤和业务逻辑。

当传递过来的变量如果没有值的时候，模版提供了默认值输出功能。

```
{$user|default='什么都没有！'}
```

四．使用运算符

我们可以在模版中使用运算符，包括对“+”、“-”、“*”、“/”、“%”、“--”和“++”的支持。

```
{$data['num']+10}                    //中括号
{$data['num']+getNum()}              //函数
```

PS：在使用运算符的时候，不再支持语法和常规的函数用法：

```
{$data.num+10}                       //错误的
{$data['num']+getNum}                //错误
```

模版还支持三元运算符：

```
{$user ? '有值' : '无值'}
```

五．包含文件

在一个系统中，可以包含通用的头文件和脚文件：header 和 footer。由于每个页面的头脚都是相同的，所以需要独立分离出来，再用包含文件引入他们。

我们可以在 View 目录下 default 主题目录下创建一个 public 目录，这个目录专门存放公共调用模版文件。

```
<include file='Public/header' />
<include file='Public/footer' />
```

也可以调用绝对路径模式：

```
<include file='./Weibo/Home/View/default/Public/header.tpl' />
```

同时调用多个模版文件：

```
<include file='Public/header,Public/footer' />
```

六．模版注释

模版支持注释功能，提供个模版制作人员参考。

```
{//这是注释}
{/*这也是注释*/}
{/*这是
    多行注释*/}
```

七．模版继承

模版继承是一项灵活的模版布局方式，它类似于子类继承父类，然后子类还可以进行适当的修改以满足当前页面的需要。

```
//基础模版需要子模版修改的部分可以用<block>包含，并且设置name值
<block name="main">主要内容</block>

//子模版只要按同样的方法，修改<block>内部的值即可完成修改
<block name="main">修改了</block>

//创建Public下base.tpl模版基页
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<block name="title"><title>{$user}</title></block>
</head>
<body>
    <include file="Public/header" />
    <block name="main">主要内容</block>
    <include file="Public/footer" />
</body>
</html>

//使用<extend>导入模版基页，name为路径，和include导入方法一致
<extend name="Public/base" />
<block name="main">修改了</block>
```

八．模版布局

ThinkPHP 的模版引擎内置了布局模版功能支持，可以方便实现模版布局以及布局嵌套功能。有三种布局方式：

1.全局配置方式

```
//开启模版布局功能，并指定基础页
'LAYOUT_ON'=>true,
'LAYOUT_NAME'=>'Public/layout',                    //layout.tpl 文件
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
//基础页，{__CONTENT__}将被子页面的内容替换
<include file="Public/header" />
{__CONTENT__}
<include file="Public/footer" />

//替换变量可以变更为{__REPLACE__}
'TMPL_LAYOUT_ITEM' =>'{__REPLACE__}',

//子模版不需要载入模版基页，可以在开头加上{__NOLAYOUT__}
{__NOLAYOUT__}
```

2.模版标签方式

标签方式，并不需要在系统做任何配置，和模版继承类似，直接引入即可。

```
//子模版引入模版基页
<layout name="Public/layout" />
//替换变量的方法
<layout name="Public/layout" replace="{__REPLACE__}" />
```

3.layout 控制布局

这个方法是在控制器里操作的。

```
//开启布局，并引入默认地址基页
public function index() {
    layout(true);
}

//引入指定基页
public function index() {
    layout('Public/layout');
    //layout(false);                                //关闭
}
```

九．模版替换

在模版渲染之前，系统还会对读取的模版内容进行一些特殊字符串替换操作，也就实现了模版输出的替换和过滤。

__ROOT__： 会替换成当前网站的地址（不含域名）

__APP__： 会替换成当前应用的 URL 地址 （不含域名）

__MODULE__：会替换成当前模块的 URL 地址 （不含域名）

__CONTROLLER__（或者__URL__ 兼容考虑）： 会替换成当前控制器的 URL 地址（不含域名）

__ACTION__：会替换成当前操作的 URL 地址 （不含域名）

__SELF__： 会替换成当前的页面 URL

__PUBLIC__：会被替换成当前网站的公共目录 通常是 /Public/

```
'TMPL_PARSE_STRING'  =>array(
       '__PUBLIC__' => '/Common', // 更改默认的/Public 替换规则
       '__UPLOAD__' => '/Uploads', // 增加新的上传路径替换规则
)
```

PS：__PUBLIC__可以改成--PUBLIC--同样的也可以。

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--内置标签

学习要点：

1.判断比较

2.遍历循环

3.其他标签

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 模版中的内置标签，所谓内置标签就是模版引擎提供的一组可以完成控制、循环和判断功能的类似 HTML 语法的标签。

一．判断比较

ThinkPHP 内置了 IF 标签用于在模版中进行条件判断。

```
//IF 语句的完整格式
<if condition="$user eq '蜡笔小新'">
    小新
<elseif condition="$user eq '黑崎一护'" />
    一护
<else />
    错误
</if>
```

PS：condition 属性中支持 eq 等判断表达式，由于会导致模版解析混淆，所以不支持“>”、“<”等表达式。

```
//condition 条件里可以使用 PHP 函数
<if condition="mb_substr($user,0,3,'UTF-8') eq '蜡笔小'">

//condition 条件里使用点语法，智能判断数组或对象
<if condition="$data.user eq '蜡笔小新'">

//condition 条件里使用冒号语法，直接使用对象
<if condition="$data:user eq '蜡笔小新'">

//condition 条件里使用系统变量
<if condition="$Think.get.user eq '蜡笔小新'">
```

PS：由于 if 标签的 condition 属性里面基本上使用的是 php 语法，尽可能使用判断标签和 Switch 标签会更加简洁，原则上来说，能够用 switch 和比较标签解决的尽量不用 if 标签完成。因为 switch 和比较标签可以使用变量调节器和系统变量。如果某些特殊的要求下面，

IF 标签仍然无法满足要求的话，可以使用原生 php 代码或者 PHP 标签来直接书写代码。

ThinkPHP 内置了 Switch 标签用于多条件判断。

```
//name 的值和 IF 语句一样，可以是变量、数组、对象、函数或系统变量
<switch name="$user">
    <case value="蜡笔小新">小新</case>
    <case value="黑崎一护">一护</case>
    <default />错误
</switch>
```

默认情况下，case 会自动添加 break 语句防止穿透，如果你就是想要穿透，那么可以去除 break。当然，绝大部分不需要设置 break。

```
//去除 break
<case value="蜡笔小新" break="0">小新</case>
```

ThinkPHP 提供了一组比较标签用于简单的变量比较，复杂的判断条件可以用 IF 标签替换。

比较标签

标签	含义
eq 或 equal	等于
neq 或 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

```
//控制器变量$user=蜡笔小新，输出小新
<eq name="user" value="蜡笔小新">小新</eq>
```

PS：name 的值和 IF 语句一样，可以是变量、数组、对象、函数或系统变量，其他比较标签使用方法类似，这里不在赘述。比较标签还提供了一个统一标签 compare，比较类型通过 type 属性完成。

```
//compare 统一方法
<compare type="eq" name="user" value="蜡笔小新">小新</compare>
```

ThinkPHP 提供了一组范围判断标签：in、notin、between、notbetween 四个标签，都用于判断变量是否在某个范围中。

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
//如果 id 是 1，2，3 任意一个将输出
<in name="id" value="1,2,3">id 在范围内</in>

//in 标签，支持 else
<in name="id" value="1,2,3">
    id在范围内
<else/>
    id不在范围内
</in>

//notin 标签，正好相反
<notin name="id" value="1,2,3">id 不在范围内</notin>

//between标签，从哪里到哪里的范围
<between name="id" value="1,10">id 在范围内</between>

//notbetween标签，从哪里到哪里的范围
<notbetween name="id" value="1,10">id 不在范围内</notbetween>

//range标签，可以统一in、notin、between和notbetween
<range name="id" value="1,2,3" type="in">id 在范围内</range>

//name值可以是系统变量
<in name="Think.get.id" value="1,2,3">id 在范围内</in>

//value值可以是变量或系统变量
<in name="id" value="$Think.get.range">id 在范围内</in>

ThinkPHP 模版提供了一组用于判断变量的标签。
//判断变量是否已赋值，赋值了就输出，空字符串也算赋值
<present name="user">user 已经赋值</present>

//判断变量是否已赋值，没有创建或赋值为null，都算没有值
<notpresent name="user">user 还没有值</notpresent>

//判断变量是否已赋值，组合
<present name="user">
    user已赋值
<else />
    user未赋值
</present>

//判断变量是否已赋值，系统变量
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
<present name="Think.get.user">$_GET['user']已赋值</present>

//判断变量是否为空，空则输出
<empty name="user">user 为空值</empty>

//判断变量是否为空，不空则输出
<notempty name="user">user不为空值</notempty>

//判断变量是否为空，组合
<empty name="user">
    user为空值
<else />
    user不为空值
</empty>

//判断变量是否为空，系统变量
<empty name="Think.get.user">$_GET['user']为空值</empty>

//判断常量是否定义，定义了输出
<defined name="APP_PATH">APP_PATH 常量已定义</defined>

//判断常量是否定义，没定义输出
<notdefined name="APP_PATH">APP_PATH 常量未定义</notdefined>

//判断常量是否定义，组合
<defined name="APP_PATH">
    APP_PATH常量已定义
<else />
    APP_PATH常量未定义
</defined>
```

二．遍历循环

ThinkPHP 内置标签提供了 Volist 标签、Foreach 标签和 For 标签。

```
//给模版传一个数据库数据
$user = M('User');
$this->assign('data', $user->select());
$this->display();
```

Volist 标签：通常用于查询数据集的结果输出，通过 select()方法返回的是一个二维数组，可以直接使用 volist 标签进行输出。

```
<ul>
    <volist name="data" id="arr">
        <li>{$arr.id} -- {$arr.user} -- {$arr.mail}</li>
```

```
    </volist>
</ul>
```

PS：这里的 name="data"中的 data 是控制器给模版定义的数据变量，要对应；这里的 id="arr"中的 arr 是将要遍历循环的数组名，可以自定义。

```
//offset 从第几条开始，起始值 0，length 共多少条
<volist name="data" id="arr" offset="5" length="10">
    <li>{$arr.id} -- {$arr.user} -- {$arr.mail}</li>
</volist>

//mod 求当前余数，当前 index 除以 2 余 1，输出偶数
<volist name="data" id="vo" mod="2">
    <eq name="mod" value="1">
    <li>{$vo.id} -- {$vo.user} -- {$vo.mail}</li>
    </eq>
</volist>

//没有数据的情况下使用 empty 填充
<volist name="data" id="vo" empty="没有任何数据">
    <li>{$vo.id} -- {$vo.user} -- {$vo.mail}</li>
</volist>

//empty 属性不支持 HTML 直接输入，但可以通过变量输出
$this->assign('empty', '<span style="color:red">没有任何数据</span>');
<volist name="data" id="vo" empty="$empty">

//通过 key 输出循环遍历的变量
<volist name="data" id="vo" key="k">
    <li>{$k} -- {$vo.id} -- {$vo.user} -- {$vo.mail}</li>
</volist>

//没有指定 k，也可以用 i 变量输出
<li>{$i} -- {$vo.id} -- {$vo.user} -- {$vo.mail}</li>

//key 变量，可以直接输出索引值，从 0 开始
<li>{$key} -- {$vo.id} -- {$vo.user} -- {$vo.mail}</li>
```

Foreach 标签和 Volist 标签类似，只不过 Foreach 标签更加简单，没有额外属性。

```
//只支持 key 属性，但{$key}变量支持
<ul>
    <foreach name="data" item="vo">
        <ul>{$vo.id} -- {$vo.user}</ul>
```

```
    </foreach>
</ul>
```

For标签就是简单的循环标签。

```
//从 1 到 99 循环
<for start="1" end="100">
    {$i}<br />
</for>
```

除了start和end必须值，还有三个可选值。comparison是大于还是小于，默认值为lt（<）；name是循环变量名，默认值为i；step是步进值，默认为1。

```
//完整形式
<for start="1" end="100" comparison="lt" step="2" name="k">
    {$i}<br />
</for>
```

三. 其他标签

ThinkPHP 模版中还有一些其他标签。

```
//在模版中定义变量，value 值可以为变量（$user）或系统变量（$Think.get.user）
<assign name="var" value="123" />
{$var}

//在模版中定义常量，value 值可以为变量（$user）或系统变量（$Think.get.user）
<define name="MY_NAME" value="Lee" />
{$Think.const.MY_NAME}
```

import 导入标签可以智能的导入 js 和 css 文件

```
//常规引入方法
<script type="text/javascript" src="__PUBLIC__/js/index.js"></script>
<link rel="stylesheet" type="text/css"
                                href="__PUBLIC__/css/index.css" />

//使用 import 导入，js 和 css 是目录
<import type="js" file="js.index" />
<import type="css" file="css.index" />

//使用 import 导入，basepath 修改默认路径
<import type="js" file="js.index" basepath="./Common" />
```

load 加载标签可以智能的加载 js 和 css 文件

```
//使用 load 加载
<load href="__PUBLIC__/js/index.js" />
<load href="__PUBLIC__/css/index.css" />
```

北风网 IBEIFENG.COM
Ycku.com 慕城Web俱乐部，做最酷的教学

系统还提供了专用 js 和 css 标签，专门用于加载

```
//使用专用标签
<js href="__PUBLIC__/js/index.js" />
<css href="__PUBLIC__/css/index.css" />
```

PHP 代码可以和标签在模版文件中混合使用，可以在模版文件里面书写任意的 PHP 语句代码。

```
//使用 php 标签
<php>
    $a = 1;
    $b = 2;
    echo $a + $b;
</php>

//使用 php 语法
<?php
    $a = 1;
    $b = 2;
    echo $a + $b;
?>
```

如果想原样输出标签极其内容，可以使用 literal 标签

```
//原样输出
<literal>
    <php>echo 123;</php>
</literal>
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--内置标签

学习要点：

1.判断比较

2.遍历循环

3.其他标签

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 模版中的内置标签，所谓内置标签就是模版引擎提供的一组可以完成控制、循环和判断功能的类似 HTML 语法的标签。

一．判断比较

ThinkPHP 内置了 IF 标签用于在模版中进行条件判断。

```
//IF 语句的完整格式
<if condition="$user eq '蜡笔小新'">
    小新
<elseif condition="$user eq '黑崎一护'" />
    一护
<else />
    错误
</if>
```

PS：condition 属性中支持 eq 等判断表达式，由于会导致模版解析混淆，所以不支持“>”、“<”等表达式。

```
//condition 条件里可以使用 PHP 函数
<if condition="mb_substr($user,0,3,'UTF-8') eq '蜡笔小'">

//condition 条件里使用点语法，智能判断数组或对象
<if condition="$data.user eq '蜡笔小新'">

//condition 条件里使用冒号语法，直接使用对象
<if condition="$data:user eq '蜡笔小新'">

//condition 条件里使用系统变量
<if condition="$Think.get.user eq '蜡笔小新'">
```

PS：由于 if 标签的 condition 属性里面基本上使用的是 php 语法，尽可能使用判断标签和 Switch 标签会更加简洁，原则上来说，能够用 switch 和比较标签解决的尽量不用 if 标签完成。因为 switch 和比较标签可以使用变量调节器和系统变量。如果某些特殊的要求下面，

IF 标签仍然无法满足要求的话，可以使用原生 php 代码或者 PHP 标签来直接书写代码。

ThinkPHP 内置了 Switch 标签用于多条件判断。

```
//name 的值和 IF 语句一样，可以是变量、数组、对象、函数或系统变量
<switch name="$user">
    <case value="蜡笔小新">小新</case>
    <case value="黑崎一护">一护</case>
    <default />错误
</switch>
```

默认情况下，case 会自动添加 break 语句防止穿透，如果你就是想要穿透，那么可以去除 break。当然，绝大部分不需要设置 break。

```
//去除 break
<case value="蜡笔小新" break="0">小新</case>
```

ThinkPHP 提供了一组比较标签用于简单的变量比较，复杂的判断条件可以用 IF 标签替换。

比较标签

标签	含义
eq 或 equal	等于
neq 或 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

```
//控制器变量$user=蜡笔小新，输出小新
<eq name="user" value="蜡笔小新">小新</eq>
```

PS：name 的值和 IF 语句一样，可以是变量、数组、对象、函数或系统变量，其他比较标签使用方法类似，这里不在赘述。比较标签还提供了一个统一标签 compare，比较类型通过 type 属性完成。

```
//compare 统一方法
<compare type="eq" name="user" value="蜡笔小新">小新</compare>
```

ThinkPHP 提供了一组范围判断标签：in、notin、between、notbetween 四个标签，都用于判断变量是否在某个范围中。

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
//如果 id 是 1，2，3 任意一个将输出
<in name="id" value="1,2,3">id 在范围内</in>

//in 标签，支持 else
<in name="id" value="1,2,3">
    id在范围内
<else/>
    id不在范围内
</in>

//notin 标签，正好相反
<notin name="id" value="1,2,3">id 不在范围内</notin>

//between标签，从哪里到哪里的范围
<between name="id" value="1,10">id 在范围内</between>

//notbetween标签，从哪里到哪里的范围
<notbetween name="id" value="1,10">id 不在范围内</notbetween>

//range标签，可以统一in、notin、between和notbetween
<range name="id" value="1,2,3" type="in">id 在范围内</range>

//name值可以是系统变量
<in name="Think.get.id" value="1,2,3">id 在范围内</in>

//value值可以是变量或系统变量
<in name="id" value="$Think.get.range">id 在范围内</in>

ThinkPHP 模版提供了一组用于判断变量的标签。
//判断变量是否已赋值，赋值了就输出，空字符串也算赋值
<present name="user">user 已经赋值</present>

//判断变量是否已赋值，没有创建或赋值为null，都算没有值
<notpresent name="user">user 还没有值</notpresent>

//判断变量是否已赋值，组合
<present name="user">
    user已赋值
<else />
    user未赋值
</present>

//判断变量是否已赋值，系统变量
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
<present name="Think.get.user">$_GET['user']已赋值</present>

//判断变量是否为空，空则输出
<empty name="user">user 为空值</empty>

//判断变量是否为空，不空则输出
<notempty name="user">user不为空值</notempty>

//判断变量是否为空，组合
<empty name="user">
    user为空值
<else />
    user不为空值
</empty>

//判断变量是否为空，系统变量
<empty name="Think.get.user">$_GET['user']为空值</empty>

//判断常量是否定义，定义了输出
<defined name="APP_PATH">APP_PATH 常量已定义</defined>

//判断常量是否定义，没定义输出
<notdefined name="APP_PATH">APP_PATH 常量未定义</notdefined>

//判断常量是否定义，组合
<defined name="APP_PATH">
    APP_PATH常量已定义
<else />
    APP_PATH常量未定义
</defined>
```

二．遍历循环

ThinkPHP 内置标签提供了 Volist 标签、Foreach 标签和 For 标签。

```
//给模版传一个数据库数据
$user = M('User');
$this->assign('data', $user->select());
$this->display();
```

Volist 标签：通常用于查询数据集的结果输出，通过 select()方法返回的是一个二维数组，可以直接使用 volist 标签进行输出。

```
<ul>
    <volist name="data" id="arr">
        <li>{$arr.id} -- {$arr.user} -- {$arr.mail}</li>
```

```
    </volist>
</ul>
```

PS：这里的 name="data"中的 data 是控制器给模版定义的数据变量，要对应；这里的 id="arr"中的 arr 是将要遍历循环的数组名，可以自定义。

```
//offset 从第几条开始，起始值 0，length 共多少条
<volist name="data" id="arr" offset="5" length="10">
    <li>{$arr.id} -- {$arr.user} -- {$arr.mail}</li>
</volist>

//mod 求当前余数，当前 index 除以 2 余 1，输出偶数
<volist name="data" id="vo" mod="2">
    <eq name="mod" value="1">
    <li>{$vo.id} -- {$vo.user} -- {$vo.mail}</li>
    </eq>
</volist>

//没有数据的情况下使用 empty 填充
<volist name="data" id="vo" empty="没有任何数据">
    <li>{$vo.id} -- {$vo.user} -- {$vo.mail}</li>
</volist>

//empty 属性不支持 HTML 直接输入，但可以通过变量输出
$this->assign('empty', '<span style="color:red">没有任何数据</span>');
<volist name="data" id="vo" empty="$empty">

//通过 key 输出循环遍历的变量
<volist name="data" id="vo" key="k">
    <li>{$k} -- {$vo.id} -- {$vo.user} -- {$vo.mail}</li>
</volist>

//没有指定 k，也可以用 i 变量输出
<li>{$i} -- {$vo.id} -- {$vo.user} -- {$vo.mail}</li>

//key 变量，可以直接输出索引值，从 0 开始
<li>{$key} -- {$vo.id} -- {$vo.user} -- {$vo.mail}</li>
```

Foreach 标签和 Volist 标签类似，只不过 Foreach 标签更加简单，没有额外属性。

```
//只支持 key 属性，但{$key}变量支持
<ul>
    <foreach name="data" item="vo">
        <ul>{$vo.id} -- {$vo.user}</ul>
```

```
    </foreach>
</ul>
```

For标签就是简单的循环标签。

```
//从 1 到 99 循环
<for start="1" end="100">
    {$i}<br />
</for>
```

除了start和end必须值，还有三个可选值。comparison是大于还是小于，默认值为lt（<）；name是循环变量名，默认值为i；step是步进值，默认为1。

```
//完整形式
<for start="1" end="100" comparison="lt" step="2" name="k">
    {$i}<br />
</for>
```

三. 其他标签

ThinkPHP 模版中还有一些其他标签。

```
//在模版中定义变量，value 值可以为变量（$user）或系统变量（$Think.get.user）
<assign name="var" value="123" />
{$var}

//在模版中定义常量，value 值可以为变量（$user）或系统变量（$Think.get.user）
<define name="MY_NAME" value="Lee" />
{$Think.const.MY_NAME}
```

import 导入标签可以智能的导入 js 和 css 文件

```
//常规引入方法
<script type="text/javascript" src="__PUBLIC__/js/index.js"></script>
<link rel="stylesheet" type="text/css"
                                href="__PUBLIC__/css/index.css" />

//使用 import 导入，js 和 css 是目录
<import type="js" file="js.index" />
<import type="css" file="css.index" />

//使用 import 导入，basepath 修改默认路径
<import type="js" file="js.index" basepath="./Common" />
```

load 加载标签可以智能的加载 js 和 css 文件

```
//使用 load 加载
<load href="__PUBLIC__/js/index.js" />
<load href="__PUBLIC__/css/index.css" />
```

北风网 IBEIFENG.COM
Ycku.com 慕城Web俱乐部，做最酷的教学

系统还提供了专用 js 和 css 标签，专门用于加载

```
//使用专用标签
<js href="__PUBLIC__/js/index.js" />
<css href="__PUBLIC__/css/index.css" />
```

PHP 代码可以和标签在模版文件中混合使用，可以在模版文件里面书写任意的 PHP 语句代码。

```
//使用 php 标签
<php>
    $a = 1;
    $b = 2;
    echo $a + $b;
</php>

//使用 php 语法
<?php
    $a = 1;
    $b = 2;
    echo $a + $b;
?>
```

如果想原样输出标签极其内容，可以使用 literal 标签

```
//原样输出
<literal>
    <php>echo 123;</php>
</literal>
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--标签库

学习要点：

1.内置标签库

2.扩展标签库

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 模版中的标签库，我们使用的很多内置标签就是通过标签库声明而来的，当然还可以自定义扩展标签。

一. 内置标签

前两节课，我们重点探讨了内置标签。但我们却不知道模版中的这些内置标签是从哪里构建出来的呢？它们是在 Think/Library/Think/Template/TagLib 下。在完整版中提供了两个类文件：Cx.class.php（内置标签核心类）、Html.class.php（html 扩展标签）。

观察 Cx.class.php 源码，我们发现其实就是内置标签的核心类。比如：

```
/**
 * volist标签解析 循环输出数据集
 * 格式：
 * <volist name="userList" id="user" empty="" >
 * {user.username}
 * {user.email}
 * </volist>
 * @access public
 * @param array $tag 标签属性
 * @param string $content  标签内容
 * @return string|void
 */
public function _volist($tag,$content) {
    //核心代码省略
}
```

在 ThinkPHP 中，Cx.class.php 是自动加载的，所以并不需要各种配置或引入而可以直接编写标签代码即可运行。但如果是 Html.class.php 这种扩展性标签库，则需要通过其他方式才可以运行：

```
/**
 * select标签解析
 * 格式： <html:select options="name" selected="value" />
 * @access public
 * @param array $tag 标签属性
 * @return string|void
```

```
     */
    public function _select($tag) {
        //核心代码省略
    }
```

如果要使用 Html.class.php 的扩展标签，那么首先要导入这个标签：

```
//导入扩展标签
<taglib name="html" />

//在使用标签的时候，前面需要加上html:，表示这个是html标签库
<html:select options="name" selected="value" />
```

PS：Cx.class.php 因为是内置标签，所以不需要加 Cx:，但 Html.class.php 则需要加上 html:，否则会无法解析标签。

```
//将Cx和Html都设置为内置标签
'TAGLIB_BUILD_IN' =>'cx,html',

//设置成内置标签，则不需要 html:了
<select options="name" selected="value" />
```

PS：设置内置标签有一定的危险性，因为可能会出现标签的重复导致冲突。

如果没有设置为内置标签的话，那么都需要通过 tablib 进行导入，我们可以设置标签预加载。这样，就不需要每次都进行导入操作了。

```
//标签预加载
'TAGLIB_PRE_LOAD'=>'html',
```

二．扩展标签库

有时感觉内置的标签不能满足项目日益繁杂的项目需求，这是可能想自己扩展一些标签来使用，那么自己怎么扩展标签呢？

第一步：在 Think/Library/Think/Template/TagLib 下创建一个 Test.class.php

```
//自定义扩展标签
namespace Think\Template\TagLib;
use Think\Template\TagLib;
defined('THINK_PATH') or exit();
/**
 * Test测试标签
 */
class Test extends TagLib{
    // 标签定义
    protected $tags   =  array(
        'mytest'=>array('attr'=>'color,border','close'=>1)
    );
```

```php
    public function _mytest($tag,$content) {
        $color = '';
        $border = '';
        if (isset($tag['color'])) {
            $color = 'color:'.$tag['color'];
        }
        if (isset($tag['border'])) {
            $border = 'border:'.$tag['border'].'px solid #ccc';
        }
        $css = $color.';'.$border;
        return '<div style="'.$css.'">'.$content.'</div>';
    }
}

//导入 test 扩展标签
<taglib name="test" />

//使用扩展标签
<Test:mytest color="blue" border="5">测试标签</Test:mytest>
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--路由功能

学习要点：

1.路由定义
2.规则路由
3.正则路由
4.静态路由
5.闭包支持

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的路由功能。路由功能，就是配置 URL，让 URL 更加的简洁和优雅。路由模式只对 PATHINFO 模式和兼容模式起作用。

一. 路由定义

在使用路由功能之前，我们需要启用路由。启用路由前提是你的 URL 支持 PATHINFO 模式或者兼容模式，而 ThinkPHP 默认的就是 PATHINF 模式。只要配置如下启用：

```
//启用路由功能
'URL_ROUTER_ON'=>true,

//配置路由规则
'URL_ROUTE_RULES'=>array(
    //每条键值对，对应一个路由规则
),
```

PS：由于 ThinkPHP3.2 或以上版本针对模块设置的，所以模块名本身（比如 Home 和 Admin）是无法路由的。当然这点也无需路由，因为可以通过设置默认模块进行去除。

```
//设置模块以及默认模块
'MODULE_ALLOW_LIST'    =>    array('Home','Admin'),
'DEFAULT_MODULE'       =>    'Home',  // 默认模块，可以省去模块名输入
```

去掉 Home 之后的 URL：http://localhost/demo39/User/index/

路由规则定义的格式为：'路由表达式'=>'路由地址和参数'，路由表达式包括规则路由和正则路由两种。

规则表达式例如：blog/:id

正则表达式例如：/^blog\/(\d+)$/

二. 规则路由

规则路由是 ThinkPHP 设计的一种简单实用的路由规则。

```
//配置路由规则
```

```
'URL_ROUTE_RULES'=>array(
    //静态地址路由
    'u'=>'User/index',
    这里只要输入：http://localhost/demo39/u
    对应的原URL：http://localhost/demo39/User/index

    //静态地址和动态地址结合路由
    'u/:id'=>'User/index',
    这里只要输入：http://localhost/demo39/u/5
    对应的原URL：http://localhost/demo39/User/index/id/5
    这里的5其实就是传递的id值，通过$_GET['id']获取。

    //静态地址和动态地址结合路由
    'u/:type/:attr/:list'=>'User/index',
    这里只要输入：http://localhost/demo39/u/5/6/7
    对应的原URL：
    http://localhost/demo39/User/index/type/5/attr/6/list/7
    这里的567分别对应了动态type、attr、list

    //全动态地址
    ':u/:id'=>'User/index',
    这里只要输入：http://localhost/demo39/u/5
    对应的原URL：http://localhost/demo39/User/index/id/5
    这里的u和5都是可以随便输入的

    //数字约束
    'u/:id\d'=>'User/index',
    给id后面加上一个正则的\d表示只能传递数字

    //支持函数
    'u/:id\d|md5'=>'User/index',
    给id进行md5加密

    //可选传参
    'u/[:id\d]'=>'User/index',
    这样的话，参数处在可选状态

    //$表示后面不允许再有东西
    'u/:id$'=>'User/index',
),
```

三．正则路由

正则路由也就是采用正则表达式定义路由的一种方式，很多复杂的路由设置无法通过规

则路由设置，则可以使用正则路由。

```
//正则路由
'/^u\/(\d{2})$/'=>'User/index?id=:1',
```

后面用普通的 URL 模式对应规则，不能使用'User/index/id/:1',

这里的:1 表示对应第一个正则圆括号。

```
//正则路由也支持函数
'/^u\/(\d{2})$/'=>'User/index?id=:1|md5',
```

四．静态路由

静态路由其实就是规则路由的简化版（又称 URL 映射），它就是撇除了动态路由的功能。

```
//配置静态路由
'URL_MAP_RULES'=>array(
    'u/i'=>'User/index',
),
```

PS：由于性能问题，ThinkPHP 把 MAP 替换为 ROUTER，专门处理静态路由。

五．闭包支持

有时，我们并非需要执行控制器的操作方法。可以用闭包定义一些特殊的需求的路由。

```
//配置闭包定义，不执行控制器方法
'URL_ROUTE_RULES'=>array(
    //规则路由
    'u/:id'=>function ($id) {
        echo 'id:'.$id;
    },
    //正则路由
    '/^u\/(\d{2})$/'=>function ($id) {
        echo 'id:'.$id;
    },
),
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--URL 操作

学习要点：

1.URL 大小写
2.URL 伪静态
3.URL 生成
4.模版中的 U

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的 URL 操作。主要涉及到 URL 路径大小写、伪静态、生成以及模版中的 U()方法。

一. URL 大小写

系统默认的规范是根据 URL 里面的模块名、控制器名来定位到具体的控制器类。比如：

http://localhost/demo39/index.php/Home/User/index/id/5

PS：在 windows 平台，URL 中的大小写会自动忽略，但作为开发人员，保持大小写区分是一个良好的习惯。而 Linux 平台，如果大小写错误，会导致无法定位到指定的模块、控制器。

```
//URL可以不区分大小写
'URL_CASE_INSENSITIVE' =>true
```

PS：如果开启了不区分大小写，那么对于 UserTypeController.class.php 这样的控制器就需要如下访问方式：

http://localhost/demo39/index.php/home/user_type/index/

如果区分大小写，则是这样：

http://localhost/demo39/index.php/Home/UserType/index/

二. URL 伪静态

URL 伪静态同城是为了满足更好的 SEO 效果，ThinkPHP 支持伪静态设置。

```
//默认情况下，伪静态后缀为.html
```

http://localhost/demo39/User/index.html

http://localhost/demo39/User/index/id/5.html

```
//设置伪静态后缀，默认为html
'URL_HTML_SUFFIX'=>'shtml',
```

http://localhost/demo39/User/index.shtml

```
//如果设置为空，那么就任意后缀
'URL_HTML_SUFFIX'=>'',
```

```
//如果设置为空，可以用__EXT__常量获取当前使用的后缀
{$Think.const.__EXT__}

//设置可以伪静态的后缀
'URL_HTML_SUFFIX'=>'html|shtml|xml',

//禁止访问的后缀
'URL_DENY_SUFFIX' => 'html|pdf|ico|png|gif|jpg',
```

三. URL 生成

为了配合所使用的 URL 模式，我们需要能够动态的根据当前的 URL 设置生成对应的 URL 地址。为此，ThinkPHP 内置了一个 U()方法，用于 URL 的动态生成。

定义格式：U('地址表达式',['参数'],['伪静态后缀'],['显示域名'])

```
//默认得到当前URL
echo U();                    ///demo39/User/index.html
```

地址表达式的格式如下：

[模块/控制器/操作#锚点@域名]?参数 1=值 1&参数 2=值 2...

```
//控制器+方法
echo U('User/add');
///demo39/index.php/User/add.html
//控制器+方法+?参数1=值1
echo U('User/add?id=5');
///demo39/index.php/User/add/id/5.html
//模块+控制器+方法+?参数1=值1
echo U('Admin/User/add?id=5');
///demo39/index.php/Admin/User/add/id/5.html
```

U()方法第二个参数可以分离参数和值的操作，支持字符串和数组的写法。

```
//使用数组参数1=值1,参数2=值2
echo U('User/add',array('id'=>5,'type'=>'a'));
///demo39/index.php/User/add/id/5/type/a.html

//使用字符串参数1=值1,参数2=值2
echo U('User/add','id=5&type=a');
///demo39/index.php/User/add/id/5/type/a.html
```

PS：不允许使用 PATHINFO 模式的路径来编写路径，比如：

```
echo U('User/add/id/1/type/a');
///demo39/index.php/1/type/a.html，这个生成的 URL 是错误的
```

U()方法第三个参数可以指定伪静态后缀，比如：

```
//指定伪静态后缀
echo U('User/add',array('id'=>5),'xml');
///demo39/index.php/User/add/id/5.xml

//设置'URL_MODEL'=>0,普通模式后URL生成将如下：
///demo39/index.php?m=&c=User&a=add&id=5

//设置'URL_MODEL'=>1,PATHINFO模式后URL生成将如下：
///demo39/index.php/User/add/id/5.html

//设置'URL_MODEL'=>2,REWRITE模式后URL生成将如下：
///demo39/User/add/id/5.html

//设置'URL_MODEL'=>3,兼容模式后URL生成将如下：
///demo39/index.php?s=/User/add/id/5.html
```

PS：在这里我们使用 REWRITE 重写模式，它是 PATHINFO 的升级版，地址最短。

我们也可以生成路由地址，使用路由的规则生成的地址：

```
//规则路由
'u/:id\d'=>'User/index',
//生成规则路由地址
echo U('/u/5');
///demo39/u/5.html

//生成正则路由地址
'/^u_(\d+)$/'=>'User/index?id=:1',
echo U('/u_5');
///demo39/u_5.html

//域名支持
echo U('User/add@www.ycuk.com?id=5');
//http://www.ycuk.com/demo39/index.php/User/add/id/5.html

//锚点支持
echo U('User/add#comment?id=5');
///demo39/User/add/id/5.html#comment
```

四．模版中的 U

我们之前使用 U()方法，都是在控制器里实现的。但实际用途中，大量的超链接是在模版中设置的，而在控制器把 U()当变量传递过去，有点过于繁杂。所以，我们可以在模版使

用使用U()方法。

```
//在模版中使用U()方法
{:U('User/add', array('id'=>5))}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--控制器

学习要点：

1.控制器操作
2.Action 参数绑定
3.跳转与重定向
4.其他操作

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的控制器，虽然在这之前，一直使用控制器，但还是有许多关于控制器的知识需要了解。

一. 控制器操作

我们首先复习一下基本的控制器定义和方法执行方式。

第一步：控制器默认创建在当前模块下的 Controller 目录下；

第二步：按照指定的命名规则：控制器名(首字母大写)+Controller.class.php；

第三步：控制器里的方法必须是 public 公共的；

```
//控制器
class UserController extends Controller {
    public function index() {
        //index()方法在URL访问时可以忽略
    }
}
```

URL 访问：http://localhost/demo39/User/index/

有时方法名可能和关键字冲突，那么定义起来就比较麻烦，控制器提供了一个配置定义：

```
//配置控制器方法后缀
'ACTION_SUFFIX'=>'Action',

//每个方法后面加上Action，对于URL访问不需要加Action
class UserController extends Controller {
    public function indexAction() {
        echo 'index';
    }
    public function testAction() {
        echo 'test';
    }
}
```

默认情况下通过 URL 访问的页面是控制器模块下的方法，即：Controller 目录下的

UserController.class.php 类。它是对外公开可访问的，我们可以称为访问控制器。那么很多时候，由于项目业务的高度繁杂，我们可能想法很多业务分离到另外的层，比如事件控制器层。那么可以这么操作：

第一步：创建事件控制器目录：Event；

第二步：创建 UserEvent.class.php 类；

```
//是否继承控制器基类或者其他基类，取决于你的业务
class UserEvent extends Controller {
    public function test() {
        echo '当触发测试事件，执行这个方法，这方法内部调用。';
    }
}

//访问控制器内部调用事件控制器方法
class UserController extends Controller {
    public function testAction() {
        $userEvent = new UserEvent();
        $userEvent->test();
        echo 'test';
    }
}
```

PS：这里调用别的控制器还可以使用快捷调用方法 A()。

```
$userEvent = A('User', 'Event');
$userEvent = A('Admin/User', 'Event');              //增加模块
```

默认情况下，控制器只支持一级控制器。当项目较为繁杂的时候，想创建统一模块下多个控制器时，可以配置二级控制器。

第一步：配置允许二级控制器；

```
//允许二级控制器
'CONTROLLER_LEVEL'=>2,
```

第二步：在 Controller 目录下创建 User 目录，在 User 目录下创建一个控制器；

```
//二级控制器写法
namespace Home\Controller\User;
use Think\Controller;

class MemberController extends Controller {
    public function index() {
        echo '多级控制器';
    }
}
```

ThinkPHP 控制器还提供了两个特殊的方法，前置操作方法_before_index()和后置操作方法_after_index()。这两个方法，在控制器调用 index()方法的时候，一个在之前执

行，一个在之后执行。

```
//前置执行
public function _before_index() {
    echo '前置操作方法';
}

//后置执行
public function _after_index() {
    echo '后置操作方法';
}
```

二．Action 参数绑定

参数绑定是通过直接绑定 URL 地址中的变量作为操作方法的参数，可以简化方法的定义甚至路由的解析。

```
//启用Action参数绑定，默认为true，可以不写
'URL_PARAMS_BIND'=>true,
```

默认的参数绑定是按变量名来绑定的，直接通过 URL 传递参数。

```
//变量名为$id
class UserController extends Controller {
    public function index($id) {
        echo 'id:'.$id;
    }
}
URL：http://localhost/demo39/User/index/id/5
```

PS：当你 URL 没有传递参数(id/5)的时候，那么页面会报错：参数错误或者未定义：id。当然，如果取消绑定，则不会报错'URL_PARAMS_BIND'=>false,。但是，为了程序的严谨性，一般需要不要取消参数绑定。

如果你个那个参数设定一个默认值，当没有传递参数的时候，会直接启用默认值：

```
//给变量$id 设置默认值
public function index($id=1)
```

还有一种是按顺序传递参数，这个方法可以省略掉键值对中的键，也就是说 id 不需要传递，只需要传递 5 这个值即可。但必须按照多个参数的顺序。

```
//定义按顺序传参绑定
'URL_PARAMS_BIND_TYPE'=>1,

//传递两个参数
class UserController extends Controller {
    public function index($id, $type) {
        echo 'id:'.$id.',type:'.$type;
```

```
    }
}
URL: http://localhost/demo39/User/index/5/a
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--控制器

学习要点：

1.控制器操作
2.Action 参数绑定
3.跳转与重定向
4.输入变量
5.请求类型
6.空操作
7.空控制器
8.操作绑定到类

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的控制器，虽然在这之前，一直使用控制器，但还是有许多关于控制器的知识需要了解。

一. 控制器操作

我们首先复习一下基本的控制器定义和方法执行方式。

第一步：控制器默认创建在当前模块下的 Controller 目录下；

第二步：按照指定的命名规则：控制器名(首字母大写)+Controller.class.php；

第三步：控制器里的方法必须是 public 公共的；

```
//控制器
class UserController extends Controller {
    public function index() {
        //index()方法在URL访问时可以忽略
    }
}
```

URL 访问：http://localhost/demo39/User/index/

有时方法名可能和关键字冲突，那么定义起来就比较麻烦，控制器提供了一个配置定义：

```
//配置控制器方法后缀
'ACTION_SUFFIX'=>'Action',

//每个方法后面加上Action，对于URL访问不需要加Action
class UserController extends Controller {
    public function indexAction() {
        echo 'index';
    }
    public function testAction() {
        echo 'test';
```

```
    }
}
```

默认情况下通过 URL 访问的页面是控制器模块下的方法，即：Controller 目录下的 UserController.class.php 类。它是对外公开可访问的，我们可以称为访问控制器。那么很多时候，由于项目业务的高度繁杂，我们可能想法很多业务分离到另外的层，比如事件控制器层。那么可以这么操作：

第一步：创建事件控制器目录：Event；

第二步：创建 UserEvent.class.php 类；

```
//是否继承控制器基类或者其他基类，取决于你的业务
class UserEvent extends Controller {
    public function test() {
        echo '当触发测试事件，执行这个方法，这方法内部调用。';
    }
}

//访问控制器内部调用事件控制器方法
class UserController extends Controller {
    public function testAction() {
        $userEvent = new UserEvent();
        $userEvent->test();
        echo 'test';
    }
}
```

PS：这里调用别的控制器还可以使用快捷调用方法 A()。

```
$userEvent = A('User', 'Event');
$userEvent = A('Admin/User', 'Event');              //增加模块
```

默认情况下，控制器只支持一级控制器。当项目较为繁杂的时候，想创建统一模块下多个控制器时，可以配置二级控制器。

第一步：配置允许二级控制器；

```
//允许二级控制器
'CONTROLLER_LEVEL'=>2,
```

第二步：在 Controller 目录下创建 User 目录，在 User 目录下创建一个控制器；

```
//二级控制器写法
namespace Home\Controller\User;
use Think\Controller;

class MemberController extends Controller {
    public function index() {
        echo '多级控制器';
    }
```

```
}
```

ThinkPHP 控制器还提供了两个特殊的方法，前置操作方法_before_index()和后置操作方法_after_index()。这两个方法，在控制器 index()方法的时候，一个在之前执行，一个在之后执行。

```
//前置执行
public function _before_index() {
    echo '前置操作方法';
}

//后置执行
public function _after_index() {
    echo '后置操作方法';
}
```

二．Action 参数绑定

参数绑定是通过直接绑定 URL 地址中的变量作为操作方法的参数，可以简化方法的定义甚至路由的解析。

```
//启用Action参数绑定，默认为true，可以不写
'URL_PARAMS_BIND'=>true,
```

默认的参数绑定是按变量名来绑定的，直接通过 URL 传递参数。

```
//变量名为$id
class UserController extends Controller {
    public function index($id) {
        echo 'id:'.$id;
    }
}
URL：http://localhost/demo39/User/index/id/5
```

PS：当你 URL 没有传递参数(id/5)的时候，那么页面会报错：参数错误或者未定义：id。当然，如果取消绑定，则不会报错'URL_PARAMS_BIND'=>false,。但是，为了程序的严谨性，一般需要不要取消参数绑定。

如果你个那个参数设定一个默认值，当没有传递参数的时候，会直接启用默认值：

```
//给变量$id 设置默认值
public function index($id=1)
```

还有一种是按顺序传递参数，这个方法可以省略掉键值对中的键，也就是说 id 不需要传递，只需要传递 5 这个值即可。但必须按照多个参数的顺序。

```
//定义按顺序传参绑定
'URL_PARAMS_BIND_TYPE'=>1,
```

```
//传递两个参数
class UserController extends Controller {
    public function index($id, $type) {
        echo 'id:'.$id.',type:'.$type;
    }
}
URL：http://localhost/demo39/User/index/5/a
```

三．跳转和重定向

ThinkPHP 在操作数据库时，需要跳转和重定向页面。ThinkPHP 提供了一组方法来解决了这个问题。

```
//成功和失败的跳转
class UserController extends Controller {
    public function index() {
        $flag = true;
        if ($flag) {
            //会跳转到：http://localhost/demo39/User/all
            $this->success('新增成功！', '../User/all');
        } else {
            //会跳转到本页的上一页
            $this->error('新增失败！');
        }
    }
}
```

PS：success()方法和 error()方法，第一个参数是提示信息、第二个跳转的地址、第三个跳转时间。默认情况下 success()方法是 1 秒，error()方法是 3 秒。

```
//设置5秒
$this->success('新增成功！', 'list', 5);
```

success()方法和 error()方法，跳转的时候就对应了相应的模版，默认如下：

```
//默认错误跳转对应的模板文件
'TMPL_ACTION_ERROR' =>THINK_PATH .'Tpl/dispatch_jump.tpl',
//默认成功跳转对应的模板文件
'TMPL_ACTION_SUCCESS'=>THINK_PATH.'Tpl/dispatch_jump.tpl',

//自定义成功和错误提示模版页面
'TMPL_ACTION_SUCCESS'=>'Public/success',
'TMPL_ACTION_ERROR'=>'Public/error',
```

对于自定义的提示模版，ThinkPHP 提供了一组变量可供使用。

变量	含义
$msgTitle	操作标题

$message	页面提示信息
$status	操作状态 1 表示成功 0 表示失败
$waitSecond	跳转等待时间，单位为秒
$jumpUrl	跳转页面地址

```
//模版中使用提示变量
操作标题：{$msgTitle}<br />
页面提示信息：{$message}<br />
操作状态：{$status}<br />
跳转等待时间：{$waitSecond}<br />
页面跳转地址：{$jumpUrl}
```

PS：如果对于数据库操作，采用了 Ajax 方式，那么 success()和 error()会自动 ajaxReturn()方法返回数据提供调用。

ThinkPHP 还单独提供了重定向方法 redirect()，参数和 U()方法一样。这个方法使用的是 URL 规则。

```
$this->redirect('User/test', array('id'=>5), 5, '页面跳转中...');
```

如果只是想纯粹的 URL 跳转，不去使用 URL 规则，那么直接使用 redirect()函数。

```
redirect('http://www.baidu.com', 5, '页面跳转中...');
```

四．输入变量

我们在获取系统变量或者用户提交的数据时，这些变量数据错综复杂，一不小心就容易引擎安全隐患，所以，ThinkPHP 提供了轻松和安全获取变量的解决方案。

```
//常规获取变量方式
$id = $_GET['id'];                  //获取get变量
$name = $_POST['name'];             //获取post变量
$value = $_SESSION['var'];          //获取session变量
$name = $_COOKIE['name'];           //获取cookie变量
$file = $_SERVER['PHP_SELF'];       //获取 server 变量
```

PS：在 ThinkPHP 中，我们不再建议直接使用传统方式获取，因为没有统一的安全处理机制，后期调整也会比较麻烦。所以，ThinkPHP 提供 I()方法进行变量的获取和过滤。

I()方法的格式：I('变量类型.变量名',['默认值'],['过滤方法'])

变量类型	含义
get	获取 GET 参数
post	获取 POST 参数
param	自动判断请求类型获取 GET、POST 或者 PUT 参数

request	获取 REQUEST 参数
put	获取 PUT 参数
session	获取$_SESSION 参数
cookie	获取$_COOKIE 参数
server	获取$_SERVER 参数
globals	获取$GLOBALS 参数

```
//获取$_GET['id'];
echo I('get.id');
//$_GET['id']没有值，则默认1
echo I('get.id', 1);
//过滤$_GET['id']
echo I('get.id', '', 'md5');
//获取$_GET;
print_r(I('get.'));
```

PS：其他几个系统变量均采用以上方式，这里不在赘述。

ThinkPHP 还提供了一个自动识别各种系统变量类型的接受方法：param。

```
//param变量自动判断变量类型
echo I('param.id');
//param可以省略
echo I('id');
```

如果是 PATHINFO 模式的 URL，param 也可以接收。

```
//param获取URL所有参数
print_r(I('param.0'));
```

PS：取消和开启 URL 的'URL_PARAMS_BIND_TYPE'=>1 配置定义，在访问这个 URL 的时候结果有所不同：http://localhost/demo39/User/index/id/5/type/a

关闭 URL 传参顺序：Array ([id] => 5 [type] => a)

开启 URL 传参顺序：Array ([0] => id [1] => 5 [2] => type [3] => a)

```
//param获取指定参数值
echo I('param.id');
echo I('param.type');
```

默认情况下，I()的默认过滤是：htmlspecialchars，过滤掉 HTML。

```
//过滤HTML，如果去掉了，则传递包含HTML的字符串，将不过滤
'DEFAULT_FILTER'=>'htmlspecialchars',
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

当然，如果不过滤，也可以在程序中过滤。

```
//设置过滤函数，会忽略配置文件里的设置
echo I('get.id', '', 'htmlspecialchars');
```

如果系统设置了默认的过滤，本身某个函数又不想过滤，那么可以这么处理：

```
//设置屏蔽系统默认过滤
echo I('get.id','', false);          //第三参数为空字符串均可
```

五．请求类型

ThinkPHP 提供了一组常量来判断当前请求是否是 GET、POST 等。通过判断请求处理不同的业务逻辑。

常量	含义
IS_GET	判断是否 GET 提交请求
IS_POST	判断是否 POST 提交请求
IS_PUT	判断是否 PUT 提交请求
IS_DELTE	判断是否 DELETE 提交请求
IS_AJAX	判断是否 AJAX 提交请求

```
//判断是否GET请求
if (IS_GET) {
    echo '是GET请求';
} else {
    echo '不是GET请求';
}
```

六．空操作

空操作是指系统在找不到请求的操作方法时，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。

```
//如果没有相关方法则执行
public function _empty($name) {
    echo '找不到方法：'.$name;
}
```

七．空控制器

所谓空控制器，就是请求不到指定控制器时，调用一个专门的空控制器。利用这个机制，我们可以实现错误页面和一些 URL 的优化。

```
//如果没有相关控制器则执行
class EmptyController extends Controller {
    public function index() {
        echo '找不到控制器：'.CONTROLLER_NAME;
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
    }
}
```

八．操作绑定到类

ThinkPHP 提供了把每个操作方法定位到一个类的功能，即每个把层次分的更加细腻。

```
//设置操作绑定到类
'ACTION_BIND_CLASS'=>True,
```

然后，在 Controller 目录下建立 User 目录，在 User 目录建立 index.class.php，再建立 test.class.php。

```
//index.class.php
namespace Home\Controller\User;
use Think\Controller;

class index extends Controller {
    public function run() {
        echo 'User模块下的index类';
    }
}

//test.class.php
namespace Home\Controller\User;
use Think\Controller;

class test extends Controller {
    public function run() {
        echo 'User模块下的test类';
    }
}
```

浏览器分别访问：

```
http://localhost/demo39/User/
http://localhost/demo39/User/test/
```

```
//前置后置方法
public function _before_run() {
    echo 'before_'.ACTION_NAME;
}
public function _after_run() {
    echo 'after_'.ACTION_NAME;
}
```

空方法，在目录里建立一个_empty.class.php

```php
//空方法
class _empty extends Controller {
    public function run() {
        echo '找不到'.ACTION_NAME.'方法';
    }
}
```

空控制器，可以创建一个目录_empty，然后建立 index.class.php

```php
//空控制器
namespace Home\Controller\_empty;
use Think\Controller;

class index extends Controller {
    public function run() {
        echo '找不到'.CONTROLLER_NAME.'控制器的'.ACTION_NAME.'方法';
    }
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--控制器

学习要点：

1.控制器操作
2.Action 参数绑定
3.跳转与重定向
4.输入变量
5.请求类型
6.空操作
7.空控制器
8.操作绑定到类

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的控制器，虽然在这之前，一直使用控制器，但还是有许多关于控制器的知识需要了解。

一. 控制器操作

我们首先复习一下基本的控制器定义和方法执行方式。

第一步：控制器默认创建在当前模块下的 Controller 目录下；

第二步：按照指定的命名规则：控制器名(首字母大写)+Controller.class.php；

第三步：控制器里的方法必须是 public 公共的；

```
//控制器
class UserController extends Controller {
    public function index() {
        //index()方法在URL访问时可以忽略
    }
}
```

URL 访问：http://localhost/demo39/User/index/

有时方法名可能和关键字冲突，那么定义起来就比较麻烦，控制器提供了一个配置定义：

```
//配置控制器方法后缀
'ACTION_SUFFIX'=>'Action',

//每个方法后面加上Action，对于URL访问不需要加Action
class UserController extends Controller {
    public function indexAction() {
        echo 'index';
    }
    public function testAction() {
        echo 'test';
```

```
    }
}
```

默认情况下通过 URL 访问的页面是控制器模块下的方法，即：Controller 目录下的 UserController.class.php 类。它是对外公开可访问的，我们可以称为访问控制器。那么很多时候，由于项目业务的高度繁杂，我们可能想法很多业务分离到另外的层，比如事件控制器层。那么可以这么操作：

第一步：创建事件控制器目录：Event；

第二步：创建 UserEvent.class.php 类；

```
//是否继承控制器基类或者其他基类，取决于你的业务
class UserEvent extends Controller {
    public function test() {
        echo '当触发测试事件，执行这个方法，这方法内部调用。';
    }
}

//访问控制器内部调用事件控制器方法
class UserController extends Controller {
    public function testAction() {
        $userEvent = new UserEvent();
        $userEvent->test();
        echo 'test';
    }
}
```

PS：这里调用别的控制器还可以使用快捷调用方法 A()。

```
$userEvent = A('User', 'Event');
$userEvent = A('Admin/User', 'Event');              //增加模块
```

默认情况下，控制器只支持一级控制器。当项目较为繁杂的时候，想创建统一模块下多个控制器时，可以配置二级控制器。

第一步：配置允许二级控制器；

```
//允许二级控制器
'CONTROLLER_LEVEL'=>2,
```

第二步：在 Controller 目录下创建 User 目录，在 User 目录下创建一个控制器；

```
//二级控制器写法
namespace Home\Controller\User;
use Think\Controller;

class MemberController extends Controller {
    public function index() {
        echo '多级控制器';
    }
```

```
}
```

ThinkPHP 控制器还提供了两个特殊的方法，前置操作方法_before_index()和后置操作方法_after_index()。这两个方法，在控制器 index()方法的时候，一个在之前执行，一个在之后执行。

```
//前置执行
public function _before_index() {
    echo '前置操作方法';
}

//后置执行
public function _after_index() {
    echo '后置操作方法';
}
```

二．Action 参数绑定

参数绑定是通过直接绑定 URL 地址中的变量作为操作方法的参数，可以简化方法的定义甚至路由的解析。

```
//启用Action参数绑定，默认为true，可以不写
'URL_PARAMS_BIND'=>true,
```

默认的参数绑定是按变量名来绑定的，直接通过 URL 传递参数。

```
//变量名为$id
class UserController extends Controller {
    public function index($id) {
        echo 'id:'.$id;
    }
}
URL：http://localhost/demo39/User/index/id/5
```

PS：当你 URL 没有传递参数(id/5)的时候，那么页面会报错：参数错误或者未定义：id。当然，如果取消绑定，则不会报错'URL_PARAMS_BIND'=>false,。但是，为了程序的严谨性，一般需要不要取消参数绑定。

如果你个那个参数设定一个默认值，当没有传递参数的时候，会直接启用默认值：

```
//给变量$id 设置默认值
public function index($id=1)
```

还有一种是按顺序传递参数，这个方法可以省略掉键值对中的键，也就是说 id 不需要传递，只需要传递 5 这个值即可。但必须按照多个参数的顺序。

```
//定义按顺序传参绑定
'URL_PARAMS_BIND_TYPE'=>1,
```

```
//传递两个参数
class UserController extends Controller {
    public function index($id, $type) {
        echo 'id:'.$id.',type:'.$type;
    }
}
URL：http://localhost/demo39/User/index/5/a
```

三．跳转和重定向

ThinkPHP 在操作数据库时，需要跳转和重定向页面。ThinkPHP 提供了一组方法来解决了这个问题。

```
//成功和失败的跳转
class UserController extends Controller {
    public function index() {
        $flag = true;
        if ($flag) {
            //会跳转到：http://localhost/demo39/User/all
            $this->success('新增成功！', '../User/all');
        } else {
            //会跳转到本页的上一页
            $this->error('新增失败！');
        }
    }
}
```

PS：success()方法和 error()方法，第一个参数是提示信息、第二个跳转的地址、第三个跳转时间。默认情况下 success()方法是 1 秒，error()方法是 3 秒。

```
//设置5秒
$this->success('新增成功！', 'list', 5);
```

success()方法和 error()方法，跳转的时候就对应了相应的模版，默认如下：

```
//默认错误跳转对应的模板文件
'TMPL_ACTION_ERROR' =>THINK_PATH .'Tpl/dispatch_jump.tpl',
//默认成功跳转对应的模板文件
'TMPL_ACTION_SUCCESS'=>THINK_PATH.'Tpl/dispatch_jump.tpl',

//自定义成功和错误提示模版页面
'TMPL_ACTION_SUCCESS'=>'Public/success',
'TMPL_ACTION_ERROR'=>'Public/error',
```

对于自定义的提示模版，ThinkPHP 提供了一组变量可供使用。

变量	含义
$msgTitle	操作标题

$message	页面提示信息
$status	操作状态 1 表示成功 0 表示失败
$waitSecond	跳转等待时间，单位为秒
$jumpUrl	跳转页面地址

```
//模版中使用提示变量
操作标题：{$msgTitle}<br />
页面提示信息：{$message}<br />
操作状态：{$status}<br />
跳转等待时间：{$waitSecond}<br />
页面跳转地址：{$jumpUrl}
```

PS：如果对于数据库操作，采用了 Ajax 方式，那么 success()和 error()会自动 ajaxReturn()方法返回数据提供调用。

ThinkPHP 还单独提供了重定向方法 redirect()，参数和 U()方法一样。这个方法使用的是 URL 规则。

```
$this->redirect('User/test', array('id'=>5), 5, '页面跳转中...');
```

如果只是想纯粹的 URL 跳转，不去使用 URL 规则，那么直接使用 redirect()函数。

```
redirect('http://www.baidu.com', 5, '页面跳转中...');
```

四．输入变量

我们在获取系统变量或者用户提交的数据时，这些变量数据错综复杂，一不小心就容易引擎安全隐患，所以，ThinkPHP 提供了轻松和安全获取变量的解决方案。

```
//常规获取变量方式
$id = $_GET['id'];                  //获取get变量
$name = $_POST['name'];             //获取post变量
$value = $_SESSION['var'];          //获取session变量
$name = $_COOKIE['name'];           //获取cookie变量
$file = $_SERVER['PHP_SELF'];       //获取 server 变量
```

PS：在 ThinkPHP 中，我们不再建议直接使用传统方式获取，因为没有统一的安全处理机制，后期调整也会比较麻烦。所以，ThinkPHP 提供 I()方法进行变量的获取和过滤。

I()方法的格式：I('变量类型.变量名',['默认值'],['过滤方法'])

变量类型	含义
get	获取 GET 参数
post	获取 POST 参数
param	自动判断请求类型获取 GET、POST 或者 PUT 参数

request	获取 REQUEST 参数
put	获取 PUT 参数
session	获取$_SESSION 参数
cookie	获取$_COOKIE 参数
server	获取$_SERVER 参数
globals	获取$GLOBALS 参数

```
//获取$_GET['id'];
echo I('get.id');
//$_GET['id']没有值，则默认1
echo I('get.id', 1);
//过滤$_GET['id']
echo I('get.id', '', 'md5');
//获取$_GET;
print_r(I('get.'));
```

PS：其他几个系统变量均采用以上方式，这里不在赘述。

ThinkPHP 还提供了一个自动识别各种系统变量类型的接受方法：param。

```
//param变量自动判断变量类型
echo I('param.id');
//param可以省略
echo I('id');
```

如果是 PATHINFO 模式的 URL，param 也可以接收。

```
//param获取URL所有参数
print_r(I('param.0'));
```

PS：取消和开启 URL 的'URL_PARAMS_BIND_TYPE'=>1 配置定义，在访问这个 URL 的时候结果有所不同：http://localhost/demo39/User/index/id/5/type/a

关闭 URL 传参顺序：Array ([id] => 5 [type] => a)

开启 URL 传参顺序：Array ([0] => id [1] => 5 [2] => type [3] => a)

```
//param获取指定参数值
echo I('param.id');
echo I('param.type');
```

默认情况下，I()的默认过滤是：htmlspecialchars，过滤掉 HTML。

```
//过滤HTML，如果去掉了，则传递包含HTML的字符串，将不过滤
'DEFAULT_FILTER'=>'htmlspecialchars',
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

当然，如果不过滤，也可以在程序中过滤。

```
//设置过滤函数，会忽略配置文件里的设置
echo I('get.id', '', 'htmlspecialchars');
```

如果系统设置了默认的过滤，本身某个函数又不想过滤，那么可以这么处理：

```
//设置屏蔽系统默认过滤
echo I('get.id','', false);          //第三参数为空字符串均可
```

五．请求类型

ThinkPHP 提供了一组常量来判断当前请求是否是 GET、POST 等。通过判断请求处理不同的业务逻辑。

常量	含义
IS_GET	判断是否 GET 提交请求
IS_POST	判断是否 POST 提交请求
IS_PUT	判断是否 PUT 提交请求
IS_DELTE	判断是否 DELETE 提交请求
IS_AJAX	判断是否 AJAX 提交请求

```
//判断是否GET请求
if (IS_GET) {
    echo '是GET请求';
} else {
    echo '不是GET请求';
}
```

六．空操作

空操作是指系统在找不到请求的操作方法时，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。

```
//如果没有相关方法则执行
public function _empty($name) {
    echo '找不到方法：'.$name;
}
```

七．空控制器

所谓空控制器，就是请求不到指定控制器时，调用一个专门的空控制器。利用这个机制，我们可以实现错误页面和一些 URL 的优化。

```
//如果没有相关控制器则执行
class EmptyController extends Controller {
    public function index() {
        echo '找不到控制器：'.CONTROLLER_NAME;
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```
    }
}
```

八．操作绑定到类

ThinkPHP 提供了把每个操作方法定位到一个类的功能，即每个把层次分的更加细腻。

```
//设置操作绑定到类
'ACTION_BIND_CLASS'=>True,
```

然后，在 Controller 目录下建立 User 目录，在 User 目录建立 index.class.php，再建立 test.class.php。

```
//index.class.php
namespace Home\Controller\User;
use Think\Controller;

class index extends Controller {
    public function run() {
        echo 'User模块下的index类';
    }
}

//test.class.php
namespace Home\Controller\User;
use Think\Controller;

class test extends Controller {
    public function run() {
        echo 'User模块下的test类';
    }
}
```

浏览器分别访问：

```
http://localhost/demo39/User/
http://localhost/demo39/User/test/
```

```
//前置后置方法
public function _before_run() {
    echo 'before_'.ACTION_NAME;
}
public function _after_run() {
    echo 'after_'.ACTION_NAME;
}
```

空方法，在目录里建立一个_empty.class.php

```php
//空方法
class _empty extends Controller {
    public function run() {
        echo '找不到'.ACTION_NAME.'方法';
    }
}
```

空控制器，可以创建一个目录_empty，然后建立 index.class.php

```php
//空控制器
namespace Home\Controller\_empty;
use Think\Controller;

class index extends Controller {
    public function run() {
        echo '找不到'.CONTROLLER_NAME.'控制器的'.ACTION_NAME.'方法';
    }
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--Session 与 Cookie

学习要点：

1.Session

2.Cookie

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的 Session 与 Cookie 的处理，提供了对这两个核心变量的完善支持。

一. Session

系统提供了 Session 管理和操作的完善支持，只需要通过 session()函数完成，该函数可以完成 Session 的设置、获取、删除和管理操作。

一般来说，在 ThinkPHP 中不需要手动初始化 session，系统会自动初始化。

参数名	说明
id	session_id 值
name	session_name 值
path	session_save_path 值
prefix	session 本地化空间前缀
expire	session.gc_maxlifetime 设置值
domain	session.cookie_domain 设置值
use_cookies	session.use_cookies 设置值
use_trans_sid	session.use_trans_sid 设置值
type	session 处理类型，支持驱动扩展

```
//手工初始化
session(array('name'=>'session_id', 'expire'=>3600));

//配置SESSION初始化
SESSION_OPTIONS=>array(
    'name'=>'session_id',
    'expire'=>3600,
),
```

默认情况下，初始化之后系统会自动启动 session，如果不希望系统自动启动 session 的话，可以如下设置：

```
//设置系统不启用session
'SESSION_AUTO_START' =>false,

//session赋值
session('user', 'Mr.Lee');          //$_SESSION['user'] = 'Mr.Lee';

//获取session值
echo session('user');               //echo $_SESSION['user'];

//session删除
session('user', null);              //unset($_SESSION['user']);

//删除所有session
session(null);                      //$_SESSION=array();

//判断session是否存在
echo session('?user');              //isset($_SESSION['user']);

//暂停session，写入关闭
session('[pause]');                 //session_write_close();

 //启动session，可开启写入
session('[start]');                 //session_start()

//销毁session
session('[destroy]');               //session_destroy();

//重新生成session id
session('[regenerate]');            //session_regenerate_id();
```

PS：如果在初始化 session 设置的时候传入 prefix 参数或者单独设置了 SESSION_PREFIX 参数的话，就可以启用本地化 session 管理支持。

```
//SESSION前缀
'SESSION_PREFIX'=>'think_',
```

这时处理的$_SESSION['user']将变成$_SESSION['think_']['user']，而我们在处理上没有任何区别。

二．Cookie

ThinkPHP 内置了一个 cookie 函数用于支持和简化 Cookie 的相关操作，该函数可以完成 Cookie 的设置、获取、删除操作。

```
//cookie赋值
cookie('user', 'Mr.Lee');           //setcookie('user', 'Mr.Lee');
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
//cookie取值
echo cookie('user');                    //echo $_COOKIE['user'];

//cookie赋值，设置过期时间
cookie('user', 'Mr.Lee', 3600);

//cookie赋值，第三参数数组设置过期时间和前缀
cookie('user', 'Lee', array('expire'=>3600,'prefix'=>'think_'));
cookie('user', 'Mr.Lee', 'expire=3600&prefix=think_');

//cookie取值，带前缀
echo cookie('think_user');
```

PS：除了过期和前缀，还有 path(保存路径)和 domain(域)。当然也支持配置文件的写法：COOKIE_PREFIX、COOKIE_EXPIRE、COOKIE_PATH、COOKIE_DOMAIN。

```php
//cookie赋值，数组
cookie('user', array('Mr.', 'Lee'));

//cookie删除user
cookie('user', null);

//cookie删除在配置文件里指定前缀的所有
cookie(null);

//cookie删除指定前缀的
cookie(null, 'think_');
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--图像处理

学习要点：

1.实例化类库

2.获取图像信息

3.图像处理

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的图像处理功能，即 Image.class.php。支持 GD 库和 Imagick 库，包括对 GIF 图像处理的支持。

一. 实例化类库

```
//实例化图像处理类，默认为GD库
$image = new Image();

 //采用Imageick扩展库，默认没安装
$image = new Image(Image::IMAGE_IMAGICK);
$image = new Image('Imagick');
```

PS：Imageick 扩展库在处理更加复杂的图像上具有很强的优势，但如果只是一些普通的需求，使用 GD 库完全够用。

二. 获取图像信息

我们讲一张图片保存在 Public 公共目录下，命名为：1.jpg。首先，通过 Image 打开图像。

```
//实例化图象处理类库
$image = new Image();
//加载一张预处理的图片
$image->open('./Public/1.jpg');

//上面两句，可以用一句话包含
$image = new Image(Image::IMAGE_GD, './Public/1.jpg');

//获取图片信息
$arr['width'] = $image->width();
$arr['height'] = $image->height();
$arr['type'] = $image->type();
$arr['mime'] = $image->mime();
$arr['size'] = $image->size();
var_dump($arr);
```

三．图像处理

使用 crop()和 save()方法可以实现图像的裁剪功能。

```
//裁剪图片，高400，宽400
$image->crop(400,400)->save('./Public/2.jpg');

//裁剪图片，坐标100,100开始裁剪
$image->crop(400,400,100,100)->save('./Public/2.jpg');

//裁剪图片，生成后的图片压缩成高宽200
$image->crop(400,400,100,100,200,200)->save('./Public/2.jpg');
```

使用 thumb 方法生成缩略图。

```
//生成缩略图
$image->thumb(300,300)->save('./Public/2.jpg');
```

PS：虽然我们设置了宽高 300，但实际上它是 300 x 168，因为生成缩略图采用默认等比例缩放的。如果想用其它缩略图模式，可以使用以下方式。

```
IMAGE_THUMB_SCALE      =   1 ;     //等比例缩放类型
IMAGE_THUMB_FILLED     =   2 ;     //缩放后填充类型
IMAGE_THUMB_CENTER     =   3 ;     //居中裁剪类型
IMAGE_THUMB_NORTHWEST  =   4 ;     //左上角裁剪类型
IMAGE_THUMB_SOUTHEAST  =   5 ;     //右下角裁剪类型
IMAGE_THUMB_FIXED      =   6 ;     //固定尺寸缩放类型
```

```
//生成缩略图，以中心点截取，直接传3也可以
$image->thumb(300,300,Image::IMAGE_THUMB_CENTER)
                                        ->save('./Public/2.jpg');
```

使用 water()方法可以生成一张包含水印的图片。

```
//在图片右下角添加水印并生成
$image->water('./Public/logo.png')->save('./Public/2.jpg');
```

由于默认是右下角水印位置，如果想更改水印位置，可以在第二参数使用如下：

```
IMAGE_WATER_NORTHWEST     =   1 ;     //左上角水印
IMAGE_WATER_NORTH         =   2 ;     //上居中水印
IMAGE_WATER_NORTHEAST     =   3 ;     //右上角水印
IMAGE_WATER_WEST          =   4 ;     //左居中水印
IMAGE_WATER_CENTER        =   5 ;     //居中水印
IMAGE_WATER_EAST          =   6 ;     //右居中水印
IMAGE_WATER_SOUTHWEST     =   7 ;     //左下角水印
IMAGE_WATER_SOUTH         =   8 ;     //下居中水印
IMAGE_WATER_SOUTHEAST     =   9 ;     //右下角水印
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
//更改水印到左上角
$image->water('./Public/logo.png', Image::IMAGE_WATER_NORTHWEST)
                                        ->save('./Public/2.jpg');

//设置水印的透明度，默认为80，1-100之间
$image->water('./Public/logo.png', Image::IMAGE_WATER_NORTHWEST, 30)
                                        ->save('./Public/2.jpg');

//设置文本水印
$image->text('YCKU.COM', './Public/consola.ttf', 20, '#ffffff',
          Image::IMAGE_WATER_SOUTHEAST)->save('./Public/2.jpg');
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--验证码

学习要点：

1.生成验证码

2.验证码参数

3.验证码检测

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的验证码功能，即 Verify.class.php。此验证码可以支持验证码的生成和验证功能。

一. 生成验证码

```
//实例化验证码类
$verify = new Verify();
//生成一个验证码图形
$verify->entry();
```

PS:生成的验证码会将信息保存在session中，可以使用print_r($_SESSION)来查看。如果想生成多个验证码，可以传递数字参数即可。

```
//生成第一个验证码
$verify->entry(1);
//生成第二个验证码
$verify->entry(2);
```

二. 验证码参数

可以对生成的验证码设置相关的参数，以达到不同的显示效果。

参数	描述
expire	验证码的有效期（秒）
useImgBg	是否使用背景图片 默认为 false
fontSize	验证码字体大小（像素） 默认为25
useCurve	是否使用混淆曲线 默认为 true
useNoise	是否添加杂点 默认为 true
imageW	验证码宽度 设置为0为自动计算
imageH	验证码高度 设置为0为自动计算
length	验证码位数
fontttf	指定验证码字体 默认为随机获取
useZh	是否使用中文验证码

bg	验证码背景颜色 rgb 数组设置，例如 array(243, 251, 254)
seKey	验证码的加密密钥
codeSet	验证码字符集合
zhSet	验证码字符集合（中文）

```
//设置验证码参数
$config = array(
    'fontSize'=>30,         //验证码字体大小
    'length'=>3,        //验证码位数
    'useNoise'=>false,//关闭验证码杂点
);
//实例化验证码类，并传入参数
$verify = new Verify($config);
//生成验证码
$verify->entry();
```

上面，采用的配置变量方式来修改验证码参数，也可以使用动态设置方法：

```
//实例化验证码类，并传入参数
$verify = new Verify();
//验证码字体大小
$verify->fontSize = 30;
//验证码位数
$verify->length = 3;
//关闭验证码杂点
$verify->useNoise = false;
//生成验证码
$verify->entry();

//设置验证码字体
$verify->fontttf = '5.ttf';

//设置背景图片功能，随机使用
$verify->useImgBg = true;

//取消曲线混淆
$verify->useCurve = false;

//设置验证码的高
$verify->imageH = 100;
//设置验证码的宽
$verify->imageW = 100;
```

```php
//设置验证码的长度
$verify->length = 3;

//设置验证码加密密钥
$verify->seKey = 'YCKU.COM';

//设置验证码背景颜色
$verify->bg = array(200,200,200);
```

如果想让验证码显示中文字符，那么需要几个步骤：

1.在 Think/zhttfs 文件夹下引入中文字体文件，比如加粗雅黑 msyhbd.ttf；

2.设置字体文件；

```php
//设置验证码字体
$verify->fontttf = 'msyhbd.ttf';
```

3.设置中文验证码

```php
//设置中文验证码
$verify->useZh = true;
```

PS：由于中文字符较为庞大，你也可以限定指定的中文字符。

```php
//限定中文字符
$verify->zhSet = '李炎恢是个好人';

//设置纯数字
$verify->codeSet='0123456789';
```

三. 验证码检测

我们可以使用 Verify.class.php 类中的 check 方法检测验证码是否输入正确。

第一步：在 function.php 里构建一个函数：

```php
//函数$code 是验证码字符，$id 是第几个验证码，如果只有一个，可以忽略
function check_verify($code, $id = ''){
    $verify = new \Think\Verify();
    return $verify->check($code, $id);
}

//通过输入验证，来验证，是否返回 true
public function test($code) {
    echo $code;
    var_dump(check_verify($code));          //($code, 1)
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com
瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--文件上传

学习要点：

1.上传表单

2.上传参数

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的文件上传功能，即 Upload.class.php。上传类可以提供单个文件上传、多个文件上传等上传方式。

一. 上传表单

```
//上传单个文件的表单模式
<form action="__CONTROLLER__/upload"
                    enctype="multipart/form-data" method="post">
    <input type="text" name="name" />
    <input type="file" name="photo" />
    <input type="submit" value="提交" />
</form>

//多文件上传设置，或者改为 photo1，photo2，photo3 即可
<input type="file" name="photo[]" />
<input type="file" name="photo[]" />
<input type="file" name="photo[]" />
```

在控制器里创建一个 upload 方法，实现上传功能：

```
//上传功能
public function upload() {
    $upload = new Upload();             //实例化上传类
    $upload->maxSize = 3145728;         //设置上传大小，字节
    $upload->exts = array('jpg','gif','png','jpeg');    //限定后缀
    $upload->savePath = './';           //在根目录Uploads下
    $info = $upload->upload();          //执行上传方法
    if (!$info) {
        $this->error($upload->getError());      //错误了
    } else {
        $this->success('上传成功！');            //成功了
    }
}
```

二．文件上传

在上传操作之前，我们可以对上传的属性进行一些设置，Upload 类支持的属性：

参数	描述
maxSize	文件上传的最大文件大小（以字节为单位），0为不限大小
rootPath	文件上传保存的根路径
savePath	文件上传的保存路径（相对于根路径）
saveName	上传文件的保存规则，支持数组和字符串方式定义
saveExt	上传文件的保存后缀，不设置的话使用原文件后缀
replace	存在同名文件是否是覆盖，默认为 false
exts	允许上传的文件后缀（留空为不限制），使用数组或者逗号分隔的字符串设置，默认为空
mimes	允许上传的文件类型（留空为不限制），使用数组或者逗号分隔的字符串设置，默认为空
autoSub	自动使用子目录保存上传文件 默认为 true
subName	子目录创建方式，采用数组或者字符串方式定义
hash	是否生成文件的 hash 编码 默认为 true
callback	检测文件是否存在回调，如果存在返回文件信息数组

```php
//实例化配置参数
$config = array(
    'maxSize'=>3145728,
    'exts'=>array('jpg','gif','png','jpeg'),
    'savePath'=>'./',
);
$upload = new Upload($config);

//动态配置参数
$upload = new Upload();
$upload->maxSize = 3145728;
$upload->exts = array('jpg','gif','png','jpeg');
$upload->savePath = './';

//设置文件名
$upload->saveName = '123';
//限制上传文件类型
$upload->mimes = array('image/jpeg');
//设置同名覆盖
$upload->replace = true;
//不设置子目录
$upload->autoSub = false;
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
//设置子目录格式
$upload->subName  = array('date','Ymd');
//不生成hash编码
$upload->hash = false;

//子目录采用函数返回
$upload->subName  = 'get_user_id';
//在 function.php 中创建一个函数
function get_user_id() {
    return rand();
}

//获取上传文件信息
foreach ($info as $file) {
    echo $file['savepath'].$file['savename'];
}
```

$file 数组可以使用的属性

属性	描述
key	附件上传的表单名称
savepath	上传文件的保存路径
name	上传文件的原始名称
savename	上传文件的保存名称
size	上传文件的大小
type	上传文件的 MIME 类型
ext	上传文件的后缀类型
md5	上传文件的 md5哈希验证字符串 仅当 hash 设置开启后有效
sha1	上传文件的 sha1哈希验证字符串 仅当 hash 设置开启后有效

```php
//获取上传文件信息
foreach ($info as $file) {
    echo $file['key'].'<br />';
    echo $file['savepath'].'<br />';
    echo $file['name'].'<br />';
    echo $file['savename'].'<br />';
    echo $file['size'].'<br />';
    echo $file['type'].'<br />';
    echo $file['ext'].'<br />';
    echo $file['md5'].'<br />';
    echo $file['sha1'].'<br />';
}
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--多语言

学习要点：

1.系统信息

2.应用信息

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的多语言设置，多语言就是国际化问题，可以改变网站各种信息语言的设置。

一. 系统信息

ThinkPHP 内置多语言支持，如果涉及到国际化的支持，那么可以定义相关的语言包文件。任何字符串形式的输出，都可以定义语言常量。

要启用多语言功能，需要配置开启多语言行为，在应用配置目录中：

```
//在应用模块 Weibo 下 Common/Conf 建立 tags.php
return array(
    'app_begin'=>array('Behavior\CheckLangBehavior'),
);
```

PS：这个行为主要是检测多语言功能。

```
//开启语言包功能
'LANG_SWITCH_ON' => true,
//自动侦测语言 开启多语言功能后有效
'LANG_AUTO_DETECT' => true,
//允许切换的语言列表 用逗号分隔
'LANG_LIST' => 'zh-cn,en-us',
//默认语言切换变量
'VAR_LANGUAGE' => 'lang',
```

ThinkPHP 系统提供了三个默认语言包，分别是简体中文、英文、繁体中文。而这三种语言包保存在：Think\Lang 下。

我们想显示系统提示信息，可以通过 L()方法来传入多语言变量。

```
//设置语言定义的键值对
L('name', 'Lee');
echo L('name');

//获取无法加载模块的常量值
echo L('_MODULE_NOT_EXIST_');
```

```php
//通过cookie切换语言定义
public function lang() {
    setcookie('think_language','en-us');
}
```

如果配置好多语言，会自动生成一个 think_language 的 cookie，默认的值是 zh-cn，我们在开发网站的时候，一旦遇到错误，显示中文提示的错误信息。

如果想显示其它的的语言包，那么可以采用 URL 的方式来切换：

http://localhost/demo39/User/?lang=en-us

二．应用信息

除了系统开发信息的语言包，我们用的更多的是项目应用中的语言定义切换。比如，一个表单支持中英文双语结构。

除了以上的配置外，实现应用信息中英文切换需要在应用目录进行配置。

第一步：在 Home 目录下建立 Lang 目录；

第二步：分别建立 zh-cn.php 和 en-us.php 两个文件；

第三步：分别在两个语言包编写语言常量；

```php
//中文语言包
return array(
    //表单部分
    'form_user'=>'用户',
    'form_pass'=>'密码',
    'form_email'=>'邮箱',
    'form_submit'=>'提交',
);

//英文语言包
return array(
    //表单部分
    'form_user'=>'user',
    'form_pass'=>'pass',
    'form_email'=>'email',
    'form_submit'=>'submit',
);

//模版获取语言定义
<p>{$Think.lang.form_user}：<input type="text" name="user" /></p>
<p>{$Think.lang.form_pass}：<input type="password" name="pass" /></p>
<p>{$Think.lang.form_email}：<input type="text" name="email" /></p>
<p><input type="submit" value="{$Think.lang.form_submit}" /></p>
```

PS：这里还可以使用{:L('form_user')}来获取语言定义。

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com
瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--auth 权限控制

学习要点：

1.权限介绍

2.简单登录

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的 auth 权限控制，本课程简单的了解一下 Auth 权限类的简单使用过程。

一. 权限介绍

所谓权限控制，大部分是在管理后台上使用。比如超级管理员登录，会得到所有操作的控制权；认证专员，只能给会员做认证操作；审核专员，只能操作内容的审核、删除、加精等操作，以此类推。那么 ThinkPHP 提供了一个内置的 Auth.class.php 类来实现权限控制，这个类提供了三个表：think_auth_rule(认证规则表)、think_auth_group(用户组表)、think_auth_group_access(用户和组对应关系表)。当然还要结合自己创建的用户表进行对应即可。

二. 简单登录

第一步：在 Admin 模块下创建一个 IndexController.class.php（默认就有了），创建 index 方法，输出一句话即可。

```
namespace Admin\Controller;
use Think\Controller;

class IndexController extends Controller {
    public function index() {
        echo '后台首页！';
    }
}
```

这是通过 URL 访问：http://localhost/demo39/Admin/Index/index，即可访问。

第二步：在 Weibo 根目录下的 Common 公共模块下创建 Controller 文件夹，并在里面创建一个 AuthController.class.php 类，这个类用于权限控制。

```
namespace Common\Controller;
use Think\Controller;
use Think\Auth;

class AuthController extends Controller {
    protected function _initialize() {
        $auth = new Auth();
```

```
            if(!$auth->check()) {
                $this->error('没有权限');
            }
        }
    }
```

这里使用的方法固定为：_initialize()。由于 AuthController.class.php 继承了 Controller 类，所以，第一步的 Index 类改为继承它。

```
    namespace Admin\Controller;
    use Common\Controller\AuthController;

    class IndexController extends AuthController {
        public function index() {
            echo '后台首页！';
        }
    }
```

此时，我们再访问后台首页的时候，已经没有权限了。主要是$auth->check()验证无法通过的原因。

第三步：创建一个 LoginController.class.php，模版为 index.tpl。

```
    <form method="post" action="{:U('index')}">
        <p>用户名：<input type="text" name="user" /></p>
        <p><input type="submit" value="登录" /></p>
    </form>

    class LoginController extends Controller {
        public function index() {
            if (IS_POST) {
                $login = array();
                switch (I('user', null, false)) {
                    case 'admin' :
                        $login['uid'] = 1;
                        $login['user'] = 'admin';
                        break;
                    case 'test' :
                        $login['uid'] = 2;
                        $login['user'] = 'test';
                        break;
                    case 'guest' :
                        $login['uid'] = 3;
                        $login['user'] = 'guest';
                        break;
                    default:
                        $this->error('登录的用户不存在！');
```

```
            }
            if (count($login)) {
                session('auth', $login);
                $this->success('登录成功！', U('Index/index'));
            }
        } else {
            $this->display();
        }
    }

    public function logout() {
        session('[destroy]');
        $this->success('退出成功！', U('Login/index'));
    }
}
```

这里登录的 LoginController.class.php 类只要继承 Controller 即可，否则无法运行。因为继承 AuthController 类的是需要权限控制的类。

第四步：完善 AuthController 类的权限验证过程。

```
class AuthController extends Controller {
    protected function _initialize() {
        $sess_auth = session('auth');
        if (!$sess_auth) {
            $this->error('非法访问！正在跳转登录页面！',
                                                U('Login/index'));
        }
        if ($sess_auth['uid'] == 1) {
            return true;
        }
        $auth = new Auth();
        if(!$auth->check(MODULE_NAME.'/'.CONTROLLER_NAME.'/'
                                .ACTION_NAME, $sess_auth['uid'])){
            $this->error('没有权限', U('Login/logout'));
        }
    }
}
```

表操作

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com
瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--关联模型

学习要点：

1.模型简介

2.关联操作

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的关联模型，关联模型可以非常容易的操作多种关联表的增、删、改、查。

一．模型简介

关联模型，一共有三种模式。一对一：ONE_TO_ONE，包括 HAS_ONE 和 BELONGS_TO；一对多：ONE_TO_MANY，包括 HAS_MANY 和 BELONGS_TO；多对多：MANY_TO_MANY。

用表关系来理解这三种模式：

一对一：用户表和身份证表，一个用户只能对应一个身份证，而一个身份证只能对应一个用户。这就是一对一。

一对多：用户表和留言表：一个用户可以发表 N 条留言，而每条留言只能由某一个用户发表。这就是一对多。就算有多个用户发表了相同的留言内容，但 ID 却不一样，所以，不能理解为多对多。

多对多：用户表和角色表：一个用户可以是认证专员，同时也是审核专员。那么角色表中的审核专员也可以是蜡笔小新的角色，也可以是路飞的角色。这就是多对多。

二．关联操作

用户表和身份证表关联，HAS_ONE（一对一），表示一个用户只有一个身份证。

```
//User 控制器部分
namespace Home\Controller;
use Think\Controller;
use Home\Model\UserModel;

class UserController extends Controller {
    public function index() {
        $user = D('User');
        $arr = $user->relation(true)->select();
        var_dump($arr);
    }
}

//User模型部分
namespace Home\Model;
use Think\Model;
```

```php
use Think\Model\RelationModel;

class UserModel extends RelationModel {
    protected $_link = array(
        'Card'=>array(
            'mapping_type'=>self::HAS_ONE,
            'class_name'=>'Card',
            'mapping_name'=>'card',
            'foreign_key'=>'uid',
            'mapping_fields'=>'code',
            'as_fields'=>'code',
            'condition'=>'id=1',
        ),
    );
}
```

HAS_ONE 支持以下属性

属性	
class_name	关联的模型类名，如果不写，会自动定位相关数据表。
mapping_name	关联映射名称，用于获取数据的数组名。
mapping_type	设置关联模式，比如一对一 self::HAS_ONE。
foreign_key	关联外键的名称，会自动对应当前数据表的 id。
mapping_fields	关联要查询的字段，默认是查询所有。
as_fields	关联的字段映射成同级的字段。
condition	关联条件，额外查询使用。

用户表和身份证表关联，BELONGS_TO（一对一），表示一个身份证从属于一个用户。

```php
//Card控制器部分
namespace Home\Controller;
use Think\Controller;
use Home\Model\CardModel;

class CardController extends Controller {
    public function index() {
        $card = D('Card');
        $arr = $card->relation(true)->select();
        var_dump($arr);
    }
}
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
//Card模型部分
namespace Home\Model;
use Think\Model;
use Think\Model\RelationModel;

class CardModel extends RelationModel {

    protected $_link = array(
        'User'=>array(
            'mapping_type'=>self::BELONGS_TO,
            'foreign_key'=>'uid',
            'mapping_fields'=>'user',
            'as_fields'=>'user',
        ),
    );
}
```

用户表和留言表：HAS_MANY（一对多）表示一个用户发表 N 条留言。

```php
namespace Home\Controller;
use Think\Controller;
use Home\Model\UserModel;

//User控制器部分
class UserController extends Controller {
    public function index() {
        $user = D('User');
        $arr = $user->relation(true)->select();
        print_r($arr);
    }
}

namespace Home\Model;
use Think\Model;
use Think\Model\RelationModel;

//User模型部分
class UserModel extends RelationModel {
    protected $_link = array(
        'Content'=>array(
            'mapping_type'=>self::HAS_MANY,
            'mapping_name'=>'contents',
            'class_name'=>'Content',
            'foreign_key'=>'uid',
```

```
            'mapping_fields'=>'content',
            'mapping_limit'=>'0,2',
            'mapping_order'=>'id DESC',
        ),
    );
}
```

PS：如果使用 HAS_MANY，还支持 mapping_limit 和 mapping_order 功能。

用户表和留言表：BELONGS_TO（多对一）表示 N 条留言从属于某个用户。

```
namespace Home\Controller;
use Think\Controller;
use Home\Model\ContentModel;

class ContentController extends Controller {
    public function index() {
        $content = D('Content');
        $arr = $content->relation(true)->select();
        var_dump($arr);
    }
}

namespace Home\Model;
use Think\Model\RelationModel;

class ContentModel extends RelationModel {
    protected $_link = array(
        'User'=>array(
            'mapping_type'=>self::BELONGS_TO,
            'class_name'=>'User',
            'foreign_key'=>'uid',
            'mapping_name'=>'user',
            'mapping_fields'=>'user',
            'as_fields'=>'user',
        ),
    );
}
```

PS：多对一的使用和一对一是一样的。

用户表、角色表和中间关联表，MANY_TO_MANY 实现多对多关联。

```
namespace Home\Model;
use Think\Model;
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
use Think\Model\RelationModel;

//User模型部分
class UserModel extends RelationModel {
    protected $_link = array(
        'Role'=>array(
            'mapping_type'=>self::MANY_TO_MANY,
            'foreign_key'=>'uid',
            'relation_foreign_key'=>'gid',
            'relation_table'=>'think_group',
        ),
    );
}
```

PS：这里的用户表和角色表并没有关联字段，而是采用中间表来关联他们，再通过ThinkPHP 的关联模式，进行多对多匹配。

表与表之间进行管理后，还可以进行写入、更新和删除。

```php
//新增一个用户，同时增加一个身份证
public function add() {
    $user = D('User');
    $data['user'] = '测试用户';
    $data['mail'] = 'test@163.com';
    $data['Card'] = array(
        'code' => '3209xx',
    );
    $user->relation(true)->add($data);
}

//删除一个用户，同时删除关联的身份证
$user = D('User');
$user->relation(true)->delete(68);

//更新一个用户，同时更新对应的身份证
public function save() {
    $user = D('User');
    $data['user'] = '测试用户4';
    $data['mail'] = 'test4@163.com';
    $data['Card'] = array(
        'code'=>'3209zz',
    );
    $user->relation(true)->where(array('id'=>67))->save($data);
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com
瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--关联模型

学习要点：

1.模型简介

2.关联操作

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的关联模型，关联模型可以非常容易的操作多种关联表的增、删、改、查。

一．模型简介

关联模型，一共有三种模式。一对一：ONE_TO_ONE，包括 HAS_ONE 和 BELONGS_TO；一对多：ONE_TO_MANY，包括 HAS_MANY 和 BELONGS_TO；多对多：MANY_TO_MANY。

用表关系来理解这三种模式：

一对一：用户表和身份证表，一个用户只能对应一个身份证，而一个身份证只能对应一个用户。这就是一对一。

一对多：用户表和留言表：一个用户可以发表 N 条留言，而每条留言只能由某一个用户发表。这就是一对多。就算有多个用户发表了相同的留言内容，但 ID 却不一样，所以，不能理解为多对多。

多对多：用户表和角色表：一个用户可以是认证专员，同时也是审核专员。那么角色表中的审核专员也可以是蜡笔小新的角色，也可以是路飞的角色。这就是多对多。

二．关联操作

用户表和身份证表关联，HAS_ONE（一对一），表示一个用户只有一个身份证。

```
//User 控制器部分
namespace Home\Controller;
use Think\Controller;
use Home\Model\UserModel;

class UserController extends Controller {
    public function index() {
        $user = D('User');
        $arr = $user->relation(true)->select();
        var_dump($arr);
    }
}

//User模型部分
namespace Home\Model;
use Think\Model;
```

```php
use Think\Model\RelationModel;

class UserModel extends RelationModel {
    protected $_link = array(
        'Card'=>array(
            'mapping_type'=>self::HAS_ONE,
            'class_name'=>'Card',
            'mapping_name'=>'card',
            'foreign_key'=>'uid',
            'mapping_fields'=>'code',
            'as_fields'=>'code',
            'condition'=>'id=1',
        ),
    );
}
```

HAS_ONE 支持以下属性

属性	
class_name	关联的模型类名，如果不写，会自动定位相关数据表。
mapping_name	关联映射名称，用于获取数据的数组名。
mapping_type	设置关联模式，比如一对一 self::HAS_ONE。
foreign_key	关联外键的名称，会自动对应当前数据表的 id。
mapping_fields	关联要查询的字段，默认是查询所有。
as_fields	关联的字段映射成同级的字段。
condition	关联条件，额外查询使用。

用户表和身份证表关联，BELONGS_TO（一对一），表示一个身份证从属于一个用户。

```php
//Card控制器部分
namespace Home\Controller;
use Think\Controller;
use Home\Model\CardModel;

class CardController extends Controller {
    public function index() {
        $card = D('Card');
        $arr = $card->relation(true)->select();
        var_dump($arr);
    }
}
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
//Card模型部分
namespace Home\Model;
use Think\Model;
use Think\Model\RelationModel;

class CardModel extends RelationModel {

    protected $_link = array(
        'User'=>array(
            'mapping_type'=>self::BELONGS_TO,
            'foreign_key'=>'uid',
            'mapping_fields'=>'user',
            'as_fields'=>'user',
        ),
    );
}
```

用户表和留言表：HAS_MANY（一对多）表示一个用户发表 N 条留言。

```php
namespace Home\Controller;
use Think\Controller;
use Home\Model\UserModel;

//User控制器部分
class UserController extends Controller {
    public function index() {
        $user = D('User');
        $arr = $user->relation(true)->select();
        print_r($arr);
    }
}

namespace Home\Model;
use Think\Model;
use Think\Model\RelationModel;

//User模型部分
class UserModel extends RelationModel {
    protected $_link = array(
        'Content'=>array(
            'mapping_type'=>self::HAS_MANY,
            'mapping_name'=>'contents',
            'class_name'=>'Content',
            'foreign_key'=>'uid',
```

```
            'mapping_fields'=>'content',
            'mapping_limit'=>'0,2',
            'mapping_order'=>'id DESC',
        ),
    );
}
```

PS：如果使用 HAS_MANY，还支持 mapping_limit 和 mapping_order 功能。

用户表和留言表：BELONGS_TO（多对一）表示 N 条留言从属于某个用户。

```
namespace Home\Controller;
use Think\Controller;
use Home\Model\ContentModel;

class ContentController extends Controller {
    public function index() {
        $content = D('Content');
        $arr = $content->relation(true)->select();
        var_dump($arr);
    }
}

namespace Home\Model;
use Think\Model\RelationModel;

class ContentModel extends RelationModel {
    protected $_link = array(
        'User'=>array(
            'mapping_type'=>self::BELONGS_TO,
            'class_name'=>'User',
            'foreign_key'=>'uid',
            'mapping_name'=>'user',
            'mapping_fields'=>'user',
            'as_fields'=>'user',
        ),
    );
}
```

PS：多对一的使用和一对一是一样的。

用户表、角色表和中间关联表，MANY_TO_MANY 实现多对多关联。

```
namespace Home\Model;
use Think\Model;
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
use Think\Model\RelationModel;

//User模型部分
class UserModel extends RelationModel {
    protected $_link = array(
        'Role'=>array(
            'mapping_type'=>self::MANY_TO_MANY,
            'foreign_key'=>'uid',
            'relation_foreign_key'=>'gid',
            'relation_table'=>'think_group',
        ),
    );
}
```

PS：这里的用户表和角色表并没有关联字段，而是采用中间表来关联他们，再通过ThinkPHP 的关联模式，进行多对多匹配。

表与表之间进行管理后，还可以进行写入、更新和删除。

```php
//新增一个用户，同时增加一个身份证
public function add() {
    $user = D('User');
    $data['user'] = '测试用户';
    $data['mail'] = 'test@163.com';
    $data['Card'] = array(
        'code' => '3209xx',
    );
    $user->relation(true)->add($data);
}

//删除一个用户，同时删除关联的身份证
$user = D('User');
$user->relation(true)->delete(68);

//更新一个用户，同时更新对应的身份证
public function save() {
    $user = D('User');
    $data['user'] = '测试用户4';
    $data['mail'] = 'test4@163.com';
    $data['Card'] = array(
        'code'=>'3209zz',
    );
    $user->relation(true)->where(array('id'=>67))->save($data);
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com
瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--视图模型和分页

学习要点：

1.视图模型

2.数据分页

主讲教师：李炎恢

本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的视图模型和数据分页。

一. 视图模型

关联模型可以解决一对一、一对多和多对多等关联操作。它们通过生成多条 SQL 语句然后进行数组拼装得到最终想要的结果。对于一对一这种 HAS_ONE 和 BELONGS_TO 的多表关联查询，可以使用视图模型。它采用的是联合查询（JOIN），非常适合此类查询需求。

```
//User控制器部分
class UserController extends Controller {
    public function index() {
        $user = D('UserView');
        var_dump($user->select());
    }
}

//User模型部分
class UserViewModel extends ViewModel {
    public $viewFields = array(
        'User'=>array('id','user','email'),
        'Card'=>array('code','_on'=>'User.id=Card.uid'),
        //'Content'=>array('content', '_on'=>'User.id=Content.uid'),
    );
}
```

PS：使用关联查询，显示 User 表且包含 Card 表关联的数据，未关联的则忽略。如果想把未关联的查询出来，可以使用 LEFT 左查询。

```
//LEFT JOIN
'User'=>array('id','user','email','_type'=>'LEFT'),

//使用group分组去除重复数据
var_dump($user->group('id')->select());

//使用COUNT得到关联的条数
'User'=>array('id','user', 'email',
```

```
                    '_type'=>'LEFT','COUNT(Content.id)'=>'abc'),
```

二. 数据分页

ThinkPHP 封装了数据分页功能，只要进行一些设置即可完成分页。

```
//User控制器部分
class UserController extends Controller {
    public function index() {
        $user = D('User');
        $count = $user->count();
        $page = new Page($count, 5);
        $show = $page->show();
        $list = $user->order('date')->
                limit($page->firstRow.','.$page->listRows)->select();
        $this->assign('list', $list);
        $this->assign('page', $show);
        $this->display();
    }
}

//模版部分
<table border="1">
    <tr><th>id</th><th>user</th><th>email</th></tr>
    <foreach name="list" item="obj">
    <tr><td>{$obj.id}</td><td>{$obj.user}</td><td>{$obj.email}</td>
                                                        </tr>
    </foreach>
</table>

<p>{$page}</p>


//设置分页参数
$page->setConfig('first','首页');
$page->setConfig('prev','上一页');
$page->setConfig('next','下一页');
$page->setConfig('last','末页');
$page->setConfig('theme', ' 共 %TOTAL_ROW% %HEADER% 共%TOTAL_PAGE%页
%FIRST% %UP_PAGE% %LINK_PAGE% %DOWN_PAGE% %END%');

//设置分页变量名
'VAR_PAGE'=>'page',
```

另外还有一个分页方法，就是使用 page()方法

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

```php
//User控制器部分
class UserController extends Controller {
    public function index() {
        $user = D('User');
        $list = $user->page($_GET['page'].',2')->select();
        $count = $user->count();
        $page = new Page($count, 2);
        $show = $page->show();
        $this->assign('list', $list);
        $this->assign('page', $show);
        $this->display();
    }
}
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--缓存处理

学习要点：

1.数据缓存
2.快速缓存
3.查询缓存
4.静态缓存

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的缓存处理，提供了数据缓存、快速缓存、查询缓存、和静态缓存。

一. 数据缓存

在 ThinkPHP 中，有一个专门处理缓存的类：Cache.class.php。而我们使用的时候，并不需要直接声明它，直接使用一个大 S()方法即可。

系统目前已经支持的缓存类型包括：Apachenote、Apc、Db、Eaccelerator、File、Memcache、Redis、Shmop、Sqlite、Wincache 和 Xcache。如果不初始化，会默认使用 File 文件缓存。

初始化的方法为：

```
S(array(
    'type'=>'memcache',
    'host'=>'127.0.0.1',
    'port'=>'11211',
    'prefix'=>'think',
    'expire'=>60
));
```

PS：如果不初始化的化，将使用默认配置使用。

type 的默认设置为：C('DATA_CACHE_TYPE');　　//File
prefix 的默认设置为：C('DATA_CACHE_PREFIX');　　//无
expire 的默认设置为：C('DATA_CACHE_TIME');　　//0
其他参数，如果没设置，将没有值。

```
//设置缓存
S('list', $list, 30);

//读取缓存是否存在，不存在则设置
if (!$list = S('list')) {
    S('list', $list, 30);
}
```

```
//删除缓存
S('list', null);
```

二．快速缓存

如果不想设置有效期存储，可以使用大 F()方法的快速缓存。

```
//使用F()方法快速缓存
public function index() {
    if (!$list = F('list')) {
        $user = D('User');
        $list = $user->select();
        F('list', $list, TEMP_PATH);
    }
    $this->assign('list', $list);
    $this->display();
}

//删除缓存
F('list', null);
```

三．查询缓存

对于及时性要求不高的数据查询，我们可以使用查询缓存功能来提高性能。

```
//查询SQL后缓存
$list = $user->cache(true)->select();

//指定缓存的Key会更加高效
$list = $user->cache('cache_user')->select();

//可以设定过期时间和其他缓存方式
$list = $user->cache('cache_user', 10, 'xcache')->select();
```

PS：如果没有安装 xcache 缓存，则会报错。而过期时间，必须关闭调试模式方可有效。

```
//获取缓存
var_dump(S('cache_user'));
//删除缓存
S('cache_user', null);
```

除了查询缓存之外，对于 SQL 查询语句本身也有解析缓存，就是把 SQL 语句给缓存下来，下次执行就不需要再解析了。当然，如果使用了查询缓存本身，由于已经将内容缓存，所以解析缓存基本上用的较少。

```
//SQL解析缓存
'DB_SQL_BUILD_CACHE' => true,
```

四．**静态缓存**

所谓静态缓存，就是第一次访问 PHP 时生成一个纯静态文件。当第二次访问时，就直接访问这个静态文件。特别适合类似于 CMS 系统这种页面状态变化较少的程序。

```
//开启静态缓存
'HTML_CACHE_ON'=>true,
//全局缓存时间60秒
'HTML_CACHE_TIME'=>60,
//缓存的后缀
'HTML_FILE_SUFFIX'=>'.html',
//缓存规则
'HTML_CACHE_RULES'=>array(
    //控制器:方法          //模块_控制器_方法_ID，局部60秒
    'User:index'=>array('{:module}_{:controller}_{:action}_{id}',
                                                        60),
),
```

除了上面的缓存规则，还有其他的缓存规格选择：

```
//方法名
'index'=>array('{id}', 60),

//控制器:          //User目录下的 方法_ID
'User:'=>array('User/{:action}_{id}', 60),

//控制器名:方法名
'User:index'=>array('{id}', 60),

//将当前地址加密作为文件名
'*'=>array('{$_SERVER.REQUEST_URI|md5}'),
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学

第 39 章 ThinkPHP--调试方法

学习要点：

1.调试模式
2.异常处理
3.模型调试
4.性能调试
5.变量调试
6.日志记录
7.Track

主讲教师：李炎恢
本课程由 北风网 和 瓢城 Web 俱乐部 提供

本节课，我们将要学习一下 ThinkPHP 的调试方法，包括各种调试配置和方法。

一. 调试模式

ThinkPHP 专门为开发过程而设置了调试模式，调试模式开启后，特别方便我们进行排错和调整。但由于它执行效率会稍低，所以在正式部署项目的时候，关闭调试模式。

```
// 入口文件处，开启调试模式 建议开发阶段开启 部署阶段注释或者设为false
define('APP_DEBUG',true);
```

调试模式在开发中的优势在于：

1.开启日志记录，任何错误和调试信息都会详细记录；
2.关闭模版缓存，模版修改可以及时生效；
3.记录 SQL 日志，方便分析 SQL；
4.关闭字段缓存，数据表字段修改不受缓存影响；
5.严格检查文件大小写，帮助提前发现 Linux 部署问题；
6.通过页面 Trace 功能更好的调试和发现问题。

二. 异常处理

和 PHP 默认的异常处理不同，ThinkPHP 抛出的不单纯的错误信息，而是一个人性化的错误页面。当然，我们也可以自行调用这种错误信息。

```
//手动抛出异常
E('失败！');
```

三. 模型调试

在模型操作中，为了更好的查明错误，经常需要查看下最近使用的 SQL 语句。

```
//获取最近的SQL语句
echo $user->getLastSql();
```

```
//查看SQL的错误信息
$list = $user->query("SELECT * FROM bbb");
echo $user->getDbError();
```

四．性能调试

我们可以使用大 G()方法，来测试某段代码的内存开销和运行时间。

```
//包裹要测试的代码
G('begin');
    $user = D('User');
    $list = $user->select();
    $this->assign('list', $list);
    $this->display();
G('end');

//计算运行时间，单位秒
echo G('begin','end').'s';
echo '<br />';
//计算消耗内存，单位kb
echo G('begin','end','m').'kb';
```

PS：如果你的环境不支持内存消耗测试，它会自动按时间运算。

五．变量调试

在开发中，我们经常需要对变量进行调试。一般会采用 PHP 自带的 var_dump()和 print_r 方法，ThinkPHP 内置了一个对浏览器友好的 dump()方法，用于输出变量信息。

```
//参数1必选，其他可选
dump($list, true, null, false);
```

PS：参数 1 为变量，参数 2 为是否打印，参数 3 标识，参数 4 是否采用 print_r。

六．日志记录

日志处理是系统自动进行的，在开启日志记录的情况下，会记录下运行的日志级别的所有日志信息。

一般情况下，SQL 日志级别必须在调试模式下开启有效，否则不会记录。默认情况下，一般在调试模式下记录日志，如果要在部署模式下开启日志记录，则需要进行配置。

```
//部署模式下开启日志记录
'LOG_RECORD' =>true,
//运行的日志记录级别
'LOG_LEVEL'  =>'EMERG,ALERT,CRIT,ERR',
```

ThinkPHP 对系统的日志按照级别来分类，包括：

1.**EMERG** 严重错误，导致系统崩溃无法使用
2.**ALERT** 警戒性错误， 必须被立即修改的错误
3.**CRIT** 临界值错误， 超过临界值的错误
4.**ERR** 一般性错误
5.**WARN** 警告性错误， 需要发出警告的错误
6.**NOTICE** 通知，程序可以运行但是还不够完美的错误
7.**INFO** 信息，程序输出信息
8.**DEBUG** 调试，用于调试信息
9.**SQL** SQL 语句，该级别只在调试模式开启时有效

```
//手动写入，第二参数为级别，第三为强制
Log::record('日志','WARN',true);

//不受配置文件影响
Log::write('日志','WARN');
```

七. Track

ThinkPHP 提供了一个非常方便的调试工具：页面 Track。这个工具必须在调试模式下有效。

```
//页面Trace
'SHOW_PAGE_TRACE' =>true,

//设置要显示的调试模块
'TRACE_PAGE_TABS'=>array(
    'base'=>'基本',
    'file'=>'文件',
    'think'=>'流程',
    'error'=>'错误',
    'sql'=>'SQL',
    'debug'=>'调试'
),

//调试方法
trace('调试方法', '提示', 'user');
```

北风网 IBEIFENG.COM
Ycku.com 飘城Web俱乐部，做最酷的教学

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(ycku.com)联合提供：

本次主讲老师：李炎恢

谢谢大家，再见！

北风网 IBEIFENG.COM

Ycku.com 瓢城Web俱乐部，做最酷的教学