



ThinkPHP Framework 1.6RC1 QuickStart

ThinkPHP 1.6RC1 快速入门

编写：ThinkPHP 文档组

最后更新：2009-05-01

目录

1	版权信息	3
2	什么是 ThinkPHP	4
3	新版特性	4
4	环境要求	5
5	获取 ThinkPHP	6
6	构建项目	7
7	创建数据表和项目配置	11
8	控制器和 URL 访问	14
9	数据操作	20
9.1	增加记录	20
9.2	查询记录	28
9.3	更新记录	34
9.4	删除记录	34
10	使用模板	35

1 版权信息

发布本资料须遵守开放出版许可协议 1.0 或者更新版本。

未经版权所有者明确授权，禁止发行本文档及其被实质上修改的版本。

未经版权所有者事先授权，禁止将此作品及其衍生作品以标准（纸质）书籍形式发行。

如果有兴趣再发行或再版本手册的全部或部分内容，不论修改过与否，或者有任何问题，请联系版权所有者 liu21st@gmail.com。

对 ThinkPHP 有任何疑问或者建议，请进入官方论坛 [<http://bbs.thinkphp.cn>] 发布相关讨论。并在此感谢 ThinkPHP 团队的所有成员和所有关注和支持 ThinkPHP 的朋友。

有关 ThinkPHP 项目及本文档的最新资料，请及时访问 ThinkPHP 项目主站 <http://thinkphp.cn> 。

2 什么是 ThinkPHP

ThinkPHP 是一个免费开源的，快速、简单的面向对象的轻量级 PHP 开发框架，遵循 Apache2 开源协议发布，是为了简化企业级应用开发和敏捷 WEB 应用开发而诞生的。借鉴了国外很多优秀的框架和模式，使用面向对象的开发结构和 MVC 模式，融合了 Struts 的 Action 思想和 JSP 的 TagLib（标签库）、RoR 的 ORM 映射和 ActiveRecord 模式，封装了 CURD 和一些常用操作，单一入口模式等，在模版引擎、缓存机制、认证机制和扩展性方面均有独特的表现。

使用 ThinkPHP，你可以更方便和快捷的开发和部署应用。当然不仅仅是企业级应用，任何 PHP 应用开发都可以从 ThinkPHP 的简单、兼容和快速的特性中受益。简洁、快速和实用是 ThinkPHP 发展秉承的宗旨，为此 ThinkPHP 会不断吸收和融入更好的技术以保证其新鲜和活力，提供 WEB 应用开发的最佳实践！

ThinkPHP 遵循 Apache2 开源许可协议发布，意味着你可以免费使用 ThinkPHP，甚至允许把你的 ThinkPHP 应用采用商业闭源发布。

3 新版特性

新版 ThinkPHP 在易用性、扩展性和性能方面做了很大的改进，并且增加了下面的特性：

- ✧ 简单易用的 MVC 模式
- ✧ 简单易用的模型和强大的扩展接口
- ✧ 内置 XML 模板引擎，支持标签库
- ✧ 高级模型、视图模型和关联模型支持
- ✧ CURD 和操作高度自动化支持

- ✧ 插件和行为的支持
- ✧ 丰富的查询语言支持
- ✧ 目录结构自动创建
- ✧ 分布式数据库支持
- ✧ 多数据库连接和切换支持
- ✧ ActiveRecord 模式和丰富的 ROR 特性
- ✧ 灵活简单的项目配置
- ✧ 模型自动验证和处理
- ✧ 静态页面生成和多元化缓存机制
- ✧ 丰富的数据库及 PDO 支持
- ✧ SEO 和 URL 路由支持
- ✧ AJAX 支持
- ✧ 易扩展的系统基类库
- ✧ 自动编码转换
- ✧ 组件和插件支持
- ✧ 基于角色的权限控制体系
- ✧ 详尽的开发指南和全中文注释

4 环境要求

ThinkPHP 可以支持 WIN/Unix 服务器环境 ,正式版需要 PHP5.0 以上版本支持 ,支持 Mysql、PgSQL、Sqlite 以及 PDO 等多种数据库 ,ThinkPHP 框架本身没有什么特别模块要求 ,具体的应用系统运行环境要求视开发所涉及的模块。

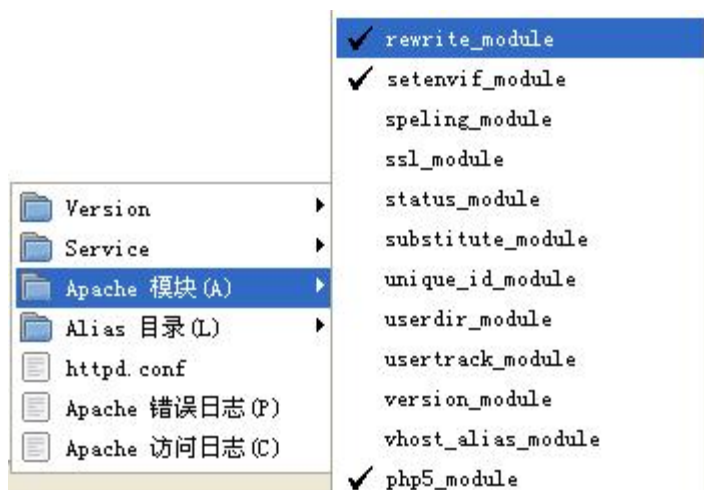
附加知识 搭建环境

如果您还没有搭建好服务器使用，下面的内容可以作为参考。

要想快速构建开发环境，可以下载 WAMPServer2 套件作为本地开发测试环境。WAMPServer2 用于架设 Apache+Php+Mysql 空间,官方下载地址是 <http://www.wampserver.com/en/download.php>。

WAMPServer2 默认 WEB 目录是 C:\wamp\www 通过 <http://localhost/>或<http://127.0.0.1/>进行访问。

由于本教程涉及到 rewrite 模块，在 WAMPServer2 安装成功后会在右下角的系统托盘中出现一个半圆形的图标，左键单击 -> Apache ->Apache 模块(A) -> 使 rewrite_module 处于选中状态，然后重启 Apache 就可以开启使用 rewrite 功能。



* 其它设置请参考相关 WAMPServer2 使用资料。

5 获取 ThinkPHP

获取 ThinkPHP 的方式很多，官方网站（<http://thinkphp.cn>）是最好的下载和文档获取来源。

官方首页提供了 ThinkPHP 的核心包和完整包下载，

◇ 核心包仅仅包含 ThinkPHP 框架本身

✧ 完整包除了核心框架外，还包括示例、手册和开发指南

你还还可以通过 SVN 获取最新的更新版本。

SVN 地址：

完整版本 <http://thinkphp.googlecode.com/svn/trunk>

核心版本 <http://thinkphp.googlecode.com/svn/trunk/ThinkPHP>

更多的 ThinkPHP 相关资源：

Google 项目地址：<http://code.google.com/p/thinkphp/>

SF 项目地址：<http://sourceforge.net/projects/thinkphp>

6 构建项目

ThinkPHP 具有项目目录自动创建功能，因此构建项目应用程序非常简单，您只需要定义好项目的入口文件，在第一次访问入口文件的时候，系统就会自动根据您在入口文件中所定义的目录路径，迅速为您创建好项目的相关目录结构。

这里以 Myapp 为项目名称进行示例，先创建好目录结构后，再逐一为您介绍各目录功能用途。

具体操作步骤：

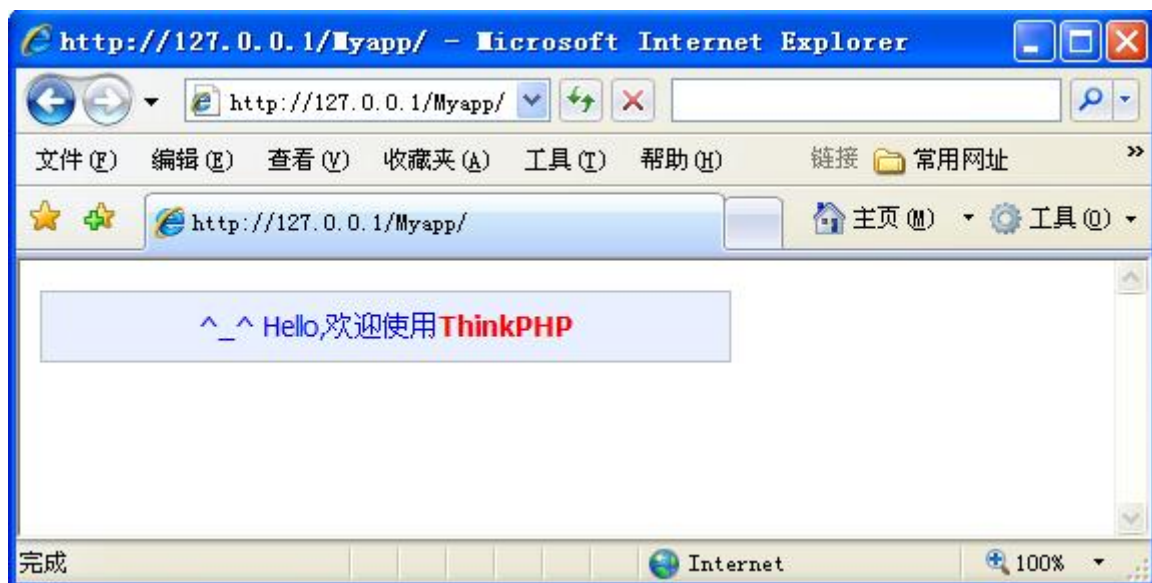
1. 在 www 根目录下新建一个文件夹，起名为 Myapp。
2. 下载 ThinkPHP 核心文件，将其复制到 Myapp 目录下。
3. 在 Myapp 目录下，新建一个入口文件，起名为 index.php，代码如下：

```
<?php
// 定义项目名称
define ('APP_NAME','Myapp');
```

```
// 加载入口文件
require ('./ThinkPHP/ThinkPHP.php');
// 实例化这个项目
$app = new App();
// 执行初始化
$app->run();
?>
```

4. 打开浏览器，访问该项目的入口文件：<http://127.0.0.1/Myapp/index.php>

这时可以看到项目构建成功后的提示画面，并且在 Myapp 目录下，已为您构建好了项目目录。



操作步骤说明：

在步骤 1 中，我们在根目录下创建 Myapp 文件夹，其实这个文件夹命名什么名称是没有关系的，仅是为了与服务器中的其它应用程序系统（例如 bbs、blog 这些）作为区分的命名，对于同一服务器多个应用程序时比较好管理，这点应很容易理解。

步骤 2 中，我们将框架核心文件放置到 Myapp 目录下，同样是为了让服务器目录比较清晰条理。其实 ThinkPHP 框架除了模板目录和网站入口文件必须放到 WEB 目录下之外，其它所有框架的文件和目录都可以放置在任意位置不受限制，您需要做的仅仅是在入口文件（例如示例的 index.php）中正确引入 ThinkPHP.php 公共文件。

在步骤 3 中的入口文件(index.php), 可以根据您所需要的实际情况而决定文件存放位置, 在任何通过 web 能访问的路径中就行(例如根目录), 通常只需要定义项目名称和引入文件然后执行初始化就可以。如果入口文件是在项目目录下, 项目名称也可以不用定义。

注意：如果是 linux 环境下面部署,需要给项目入口文件里面指定的路径设置可写权限。这些目录仅仅针对项目目录, 系统目录无需设置任何可写权限, 因为每个项目的模版缓存和数据缓存, 以及日志文件都是独立的。

在步骤 4 中, 首次访问入口文件 index.php 时, 在项目的 Temp 目录下面会生成两个编译缓存文件: ~runtime.php 和~app.php。这两个文件是 ThinkPHP 的为了减少运行时的文件加载开销而生成的编译缓存文件。其中~runtime.php 是核心编译缓存, 把框架运行所需要的核心类库缓存到一个文件里面, ~app.php 是项目编译缓存文件, 其中包含了项目的公共文件、项目配置等。如果公共文件发生修改, 就必须删除该文件重新生成。

注意: ThinkPHP 框架的所有文件都是采用 UTF-8 编码保存, 但是这不影响你的项目中使用其他编码开发和浏览。请注意确保文件保存的时候去掉 UTF-8 的 BOM 头信息, 防止因产生隐藏的输出而导致程序运行不正常。

ThinkPHP 框架目录结构



ThinkPHP 的系统基本目录非常简单, 同时命名也很清晰明确:

ThinkPHP.php : ThinkPHP 公共文件。

Common 目录：系统公共文件目录

Lang 目录：系统语言包目录

Lib 目录：系统基类库目录

Tpl 目录：框架系统模版目录

Vendor 目录：第三方类库目录

另外系统基类库目录 Lib 文件夹下，还有相关的子目录：

Think 文件夹：系统运行库目录（必须）

ORG 文件夹：扩展类库目录（非必须）

具体内容可以打开目录查看结构和文件组成，ThinkPHP 使用全中文的详尽注释，当应用熟悉后，了解每个类库文件的功能和相关方法会让您开发过程事半功倍。

项目目录结构

再来看看系统自动生成的项目目录结构，如下图所示：



Cache 目录：项目自动生成的模版缓存

Common 目录：项目中应用的公共文件，通常包含 common.php 公共函数文件。

Conf 目录：项目各项配置文件。

Data 目录：项目的数据缓存目录。

Lang 目录：项目的语言包文件。

Lib 目录：应用类库目录，在这文件夹内通常包含两个子目录：Action 和 Model，Action 目录放置命名控制器文件，Model 目录放置模块类文件。除此之外，还可以放置其他的项目类库。

Logs 目录：项目的日志文件。

Temp 目录：数据缓存目录，存放项目缓存文件。

Tpl 目录：项目模板文件，按风格创建对应的子目录，系统自动生成的 default 文件夹，用以保存项目默认风格的模板文件。

注意：在 Linux 环境下，项目目录的 Cache（模版缓存目录）、Data（数据缓存目录）、Temp（编译缓存目录）、Logs（日志文件目录）必须设置为可写，另外，如果设置了 Public 项目公用文件目录和在 Public 下面设置了 Uploads 目录作为公共上传目录，也都需要设置可写权限。通常都是设置目录属性为 777。

一定要注意在 Linux 环境下面的文件大小写问题，否则会导致文件加载错误。

7 创建数据表和项目配置

构建好开发环境和项目应用程序后，接下来我们通过 phpMyAdmin 或其他 MySQL 管理工具创建所需的数据库，然后再对项目进行相应配置。

如何与数据库中的表进行连接？

- 1 创建数据库
- 2 在 config.php 配置文件中设置数据库访问的连接信息
- 3 在 项目/Lib/Model 目录下，建立模型文件,就可以直接对数据表进行增删改查等操作了。

具体操作步骤：（以 MySQL 为例，如果是其他数据库请修改后再执行）

1 创建一个数据库，名为 myapp

2 运行下面的 sql 增加一个 think_message 表。

```
CREATE TABLE `think_message` (  
  `id` smallint(4) unsigned NOT NULL AUTO_INCREMENT COMMENT '自增编号',  
  `title` varchar(255) NOT NULL COMMENT '标题',  
  `content` varchar(255) NOT NULL COMMENT '内容',  
  `create_time` int(10) unsigned NOT NULL COMMENT '创建时间',  
  `email` varchar(50) NOT NULL COMMENT '邮箱地址',  
  `status` tinyint(1) unsigned NOT NULL DEFAULT '1',  
  PRIMARY KEY (`id`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC COMMENT='留言表'  
AUTO_INCREMENT=1 ;
```

3 在 Myapp/Conf 目录下面创建一个 config.php 项目配置文件，代码如下：

```
<?php  
return array(  
    // 定义数据库连接信息  
    'DB_TYPE'=> 'mysql', // 指定数据库是 mysql  
    'DB_HOST'=> 'localhost',  
    'DB_NAME'=>'myapp', // 数据库名  
    'DB_USER'=>'root',  
    'DB_PWD'=>'123456', //您的数据库连接密码  
    'DB_PORT'=>'3306',  
    'DB_PREFIX'=>'think_', //数据表前缀（与数据库 myapp 中的表 think_message 对应）  
);  
?>
```

注意：请确保配置文件里面的设置和你本地的数据库连接信息一致。

4 在 Myapp/Lib/Model/目录下，新建一个文件 MessageModel.class.php，代码如下：

```
class MessageModel extends Model{ }
```

通过以上步骤 我们现在已将数据库表和模型类建立了联接关系。在项目的 Action 类和 Model 类中，
已可以直接对数据库进行相关操作。

基础知识 ThinkPHP 配置设置

ThinkPHP 提供了灵活的配置功能，采用最有效率的 PHP 返回数组方式定义，支持惯例配置、项目配置、调试配置和模块配置等，并且会自动生成配置缓存文件，无需重复解析的开销。

对于简单的应用,例如不涉及到数据库操作时,ThinkPHP 可以无需定义任何配置文件,因为 ThinkPHP 框架有自身的惯例配置。(ThinkPHP/Common/convention.php) 。

默认生成的项目并没有包含项目配置文件，因此需要配置时，要在项目的 conf 目录下创建一个配置文件 config.php,其配置格式为：

```
return array(  
    '配置名称 1' => 配置值,  
    '配置名称 2' => 配置值,  
);
```

并且支持二维数组的形式：

```
return array(  
    'DEBUG_MODE' => true,  
    'USER_CONFIG' => array(  
        'USER_AUTH' => true,  
        'USER_TYPE' => 2,  
    ),  
);
```

提示 1：所有的配置定义都是不区分大小写的（因为无论大小写定义都会转换成小写），但是习惯上保持大写定义的原则。

提示 2：每个项目配置文件除了定义 ThinkPHP 所需要的配置参数之外，您也可以在里面添加项目需要的一些配置参数，用于自己的应用。例如上例中的 USER_CONFIG。

提示 3：同一配置设定，项目中的配置参数会覆盖 ThinkPHP 系统默认的惯例配置，对于应用项目的配

置文件，往往只需要配置和惯例配置不同的或者新增的配置参数。

提示 4：项目配置文件默认会进行自动加载并缓存。并且一旦修改项目配置文件的话会自动更新配置缓存文件。

基础知识 ThinkPHP 的表和模型的命名规则

ThinkPHP 要求数据库的表名和模型类 (model) 的命名要遵循一定的规范，首先数据库的表名采用小写形式，模型类文件名的命名规则是：**不含前缀的数据表表名(采用驼峰命名法)+ Model**

在创建 message 表时，我们使用的名称是 think_message，think_就是数据表前缀。在配置文件中我们已使用 '`DB_PREFIX`'=>'think_' 作出了指定。(如果你的规则和系统的约定不符合，那么需要设置 Model 类的 tableName 属性)

模型类的类名定义也同样是 **不含前缀的数据表表名+Model**，并且该类要继承所需要的 Model 类。

ThinkPHP1.6 的模型类按功能作了细分，目前共有：

- ✧ Model 基本模型类
- ✧ AdvModel 高级模型类
- ✧ RelationModel 关联模型类
- ✧ ViewModel 视图模型类

具体请根据实际应用情况需要来进行继承。

8 控制器和 URL 访问

如何定义一个应用模块？

在项目目录/Lib/Action 文件夹下，建立一个 模块名+Action.class.php 文件，并继承 Action 类则可。

如何定义一个操作？

在应用模块类的代码中，添加一个 function 方法则可。

如何访问具体操作？

<http://服务器地址/项目地址/模块名称/操作名称>

如何通过 URL 传递变量

<http://服务器地址/项目地址/模块名称/操作名称/变量名 1/变量值 1/变量名 2/变量值 2/>

基础知识 URL 访问

在进行本节操作步骤前，需要简单了解 ThinkPHP 控制器和 URL 访问基本知识。

ThinkPHP 的控制器指的是模块类，位于项目的 Lib\Action 目录下面。每个 Action 文件均代表着该项目的的一个应用模块，而 Action 文件中的每一个方法(function)，则代表着此应用模块其中一个功能操作。

操作又分为有输出到模板的操作和只具执行不需要输出的操作。对于有输出的操作，我们可以通过模板将内容进行输出展示，即用户可见到的 web 页面。

打开 Myapp/Lib/Action 目录，我们可以发现在目录下存在了一个 IndexAction.class.php 文件，这个是 ThinkPHP 初次启动时自动创建的默认应用模块类。

在 ThinkPHP 框架的规范里面 类名和文件名是一致的，并且类文件都是以.class.php 为后缀。所以，IndexAction 类就表示该项目的 Index 应用模块。而类中的 index 方法，就是 Index 应用模块中的一个操作。

通常一个项目会由许多个应用模块组成，在开发时将项目的功能按类别来区分，并用一个意义明确的类名(例如 Index 代表着首页模块，Message 代表着留言模块等等，模块类命名只要清晰并且意义明确就行)，通过在 项目目录/Lib/Action 中建立一个个以 类名（首字母大写）+Action.class.php 的类文件，然后根据该模块的需要，在类中添加不同的功能操作并书写逻辑结构（例如用户注册，用户登录，数据内容列表展示，数据添加，数据修改，数据删除等等），这样就构成了整个项目应用程序。

ThinkPHP 框架的应用，采取单一入口文件来执行，所有的模块和操作都通过 URL 的参数来访问和执行。ThinkPHP 支持的 URL 模式包括普通模式、PATHINFO 模式和 REWRITE 模式以及兼容模式，并且都提供路由支持。默认为 PATHINFO 模式，提供最好的用户体验和搜索引擎友好支持。

PATHINFO 模式下的 URL 基本结构是这样的

<http://localhost/appName/index.php/moduleName/actionName/id/1/>

你可以理解为

<http://服务器地址/入口文件位置/应用模块名称/具体操作名称/GET 变量名称/GET 变量值>

通常情况下，我们的入口文件 index.php 都是放在根目录中，这时项目名称就会被入口文件 index.php 所替代，因此，假如要访问 Index 模块下的 index 操作。地址应该是：

<http://127.0.0.1/index.php/Index/index>

我们实例中的入口文件并不是在服务器的根目录，而是在 myapp 目录下，那么入口文件位置就是 Myapp/index.php，此时上面的 URL 访问就相应地换成：

<http://127.0.0.1/Myapp/index.php/Index/index>

PATHINFO 模式对以往的编程方式没有影响，使用 GET 和 POST 方式传值依然有效，系统会对 PATHINFO 方式自动处理。

其它模式的 URL 访问方法会在附加知识中再提及。

理解上述基础知识后，下面我们在实际操作中进行体验和加深了解。（要注意文件编码问题）

具体操作实践：

1 打开 Myapp/Lib/Action/IndexAction.class.php 文件，在类中添加一个 test1 操作，代码如下

```
// URL 访问实践

public function test1 ( ) {
    header("Content-Type:text/html; charset=utf-8");
    echo 'HI,ThinkPHP ! ';
}
```

2 打开浏览器，对 test1 操作进行访问。

地址是：<http://127.0.0.1/Myapp/index.php/Index/test1>

3 在 Myapp/Lib/Action/目录下，新建一个应用模块，文件名为 MessageAction.class.php

4 在 MessageAction.class.php 中添加一个 test2 操作。代码如下：

```
<?php
// 添加应用模块实践

class MessageAction extends Action{
    // URL 访问实践 2

    public function test2 () {
        header("Content-Type:text/html; charset=utf-8");
        echo ' test2 操作访问成功 !';
    }
}
?>
```

5 在浏览器中尝试输入访问地址，对 test2 进行正确访问。（成功可以看到 echo 所输出的内容）

6 在 MessageAction.class.php 中添加一个操作 test3，进行获取 GET 传递的变量参数，代码如下：

```
// URL 访问实践 --- 获取 GET 变量

public function test3 () {
    header("Content-Type:text/html; charset=utf-8");
    dump($_GET['abc']);
}
```

7 在浏览器中输入 <http://127.0.0.1/Myapp/index.php/Message/test3> 进行访问。

8 在浏览器中输入 <http://127.0.0.1/Myapp/index.php/Message/test3/abc/thinkphp> 进行访问

9 在浏览器中输入 <http://127.0.0.1/Myapp/index.php/Message/test3?abc=thinkphp> 进行访问，
并留意浏览器的最终地址是什么。

10 为 MessageAction 类添加一个 ThinkPHP 的初始化操作_initialize()，代码如下

```
// ThinkPHP 特殊的操作方法 - 初始化

public function _initialize() {
    echo '每个操作都会先执行此方法逻辑内容<br /><br />';
}
```

```
}
```

11 分别访问 Message 应用模块的 test2 和 test3 操作，看看输出的内容。

通过上面的实践，我们应该知道了如何添加应用模块和操作，并进行对操作的访问。

Action 控制器定义

应用开发中的一个重要过程就是给不同的模块定义具体的操作。一个应用如果不需要和数据库交互的时候可以不需要定义模型类，但是必须定义 Action 控制器。

Action 控制器的定义非常简单，只要继承 Action 基础类就可以了，例如：

```
Class UserAction extends Action{}
```

URL 模式相关配置设置

ThinkPHP 支持四种 URL 模式，可以通过 config.php 对 Dispatch 方式进行设置

```
'DISPATCH_ON'=> true,    // 该参数设置是否启用 Dispatcher 功能。
```

在开启 Dispatch 功能后，就可以对 URL 模式进行设置了。

```
'URL_MODEL' => 1,
```

系统默认是 PATHINFO 模式，提供最好的用户体验和 SEO 支持，可选设置为(0、1、2、3)

0 普通模式

1 PATHINFO

2 REWRITE

3 兼容模式

兼容模式是普通模式和 PATHINFO 模式的结合，并且可以让应用在需要的时候直接切换到 PATHINFO 模式而不需要更改模板和程序。兼容模式 URL 可以支持任何的运行环境。

PATHINFO 模式还包括普通模式、智能模式和兼容模式三种，通过以下参数配置：

```
'PATH_MODEL'=> 2,
```

系统默认采用智能模式，可选设置为(1、2、3)：

- 1 普通模式 参数没有顺序/m/module/a/action/id/1
- 2 智能模式 自动识别模块和操作/module/action/id/1/ 或者 /module,action,id,1/...

3 兼容模式 通过一个 GET 变量将 PATHINFO 传递给 dispatcher，默认为 s

例如：index.php?s=/module/action/id/1

使用 rewrite

要去除 URL 访问时的 index.php 入口文件，可以使用 **Rewrite** 功能(需要服务器环境支持)。使 URL 类似这样的形式：

<http://127.0.0.1/Index/index>

- 1 确认 httpd.conf 配置文件中加载了 mod_rewrite.so 模块
- 2 AllowOverride None 将 None 改为 All
- 3 打开 Myapp/Conf/config.php 文件，在配置的数组中添加一行：

```
'URL_MODEL'=>2,
```

在入口文件所在的同级目录下，新建一个.htaccess 文件，内容是

```
<IfModule mod_rewrite.c>
    RewriteEngine on
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^(.*)$ index.php?s=/$1 [QSA,PT,L]
</IfModule>
```

如果你的服务器环境支持 rewrite，使用 <http://127.0.0.1/Index/index> 就可以正常访问到 Index 模块的 index 操作了。

URL 参数传递

URL 地址中，后面的参数是显式传递的，而且必须成对出现，例如：

<http://<serverName>/appName/module/action/year/2008/month/09/day/21/>

其中参数之间的分割符号由 `PATH_DEPR` 参数设置，默认为“/”，例如我们设置 `PATH_DEPR` 为^的话，就可以使用下面的 URL 访问：

<http://<serverName>/appName/module^action^id^1/>

注意不要使用“:” 和“&” 符号进行分割，该符号有特殊用途。

略加修改，就可以展示出富有诗意的 URL，呵呵～

9 数据操作

使用 ThinkPHP 开发时，项目中每一个继承了 `Model` 的模型类，都会对应着一个数据表(或者视图)，模型与数据表存在一种映射关系。(例如之前我们所定义的 `MessageModel` 类，对应的就是 `think_message` 表)。

开发过程中，只要简单地定义好与数据表对应的模型类 (`xxxModel.class.php`)，甚至无需增加任何属性和方法就可以进行对数据操作了。

ThinkPHP 提供了灵活和方便的数据操作方法，不仅实现了对数据库操作的四个基本操作 (`CURD`)：创建、更新、读取和删除的实现，还内置了很多实用的数据操作方法。

9.1 增加记录

如何添加记录到数据库表？

- 1 获取 (设置) 要添加的数据内容。
- 2 实例化数据模型，调用 `Model` 类的 `add()` 方法即可添加。

具体操作实践一 添加数据：数据对象赋值

1.1 在 Index 模块中新添一个方法 add1，代码如下：

```
public function add1() {  
    // 使用 D 函数实例化消息模型类  
    $Message = D('Message');  
    // 然后给数据对象赋值  
    $Message->title = 'ThinkPHP';  
    $Message->email = 'ThinkPHP@gmail.com';  
    // 使用 add 方法保存内容  
    $Message->add();  
    // 使用 select 方法进行数据查询  
    $rs = $Message->select();  
    // 使用 dump 函数 输出查询数据查看  
    dump ( $rs );  
}
```

1.2 在浏览器中访问 <http://127.0.0.1/Myapp/index.php/Index/add1>，成功可以看到如下数据：

```
array(1) {  
    [0] => array(6) {  
        ["id"] => string(1) "1"  
        ["title"] => string(8) "ThinkPHP"  
        ["content"] => string(0) ""  
        ["create_time"] => string(1) "0"  
        ["email"] => string(18) "ThinkPHP@gmail.com"  
        ["status"] => string(1) "1"  
    }  
}
```

代码相关知识：

D() 函数

通常要实例化一个类，都是导入这个类文件，然后使用 new 关键字进行实例化：

```
$dao = new MessageModel();
```

但在 ThinkPHP 中，我们可以通过 D 函数更简单地进行实例化模型类。D 函数会自动导入相应类文件，并对该类进行检测，如果之前已实例化过 MessageModel，就不会再进行实例化，如果该 Model

类不存在，就会抛出异常方便调试。并且 D 函数支持跨项目调用 Model 类。

访问当前项目的 Model，只要直接使用 D('Model 名称'); ----- 例如上例的 D('Message');

如果要跨项目访问，可以使用 D('Model 名称','项目名称');在多项目应用时，调用非常方便。

dump() 函数

dump 函数是 ThinkPHP 内置的对浏览器友好的变量输出函数，在开发过程中使用该函数可以快速地查看结果输出，当只有一个 \$var 参数时，会直接打印这个 \$var。如上例中的：dump(\$rs);

另外，dump 函数也可以设置参数，设置方式是：dump(\$var, \$echo=true, \$label=null, \$strict=true)

\$var 要输出的变量

\$echo 是否直接打印

\$label 变量标签

\$strict 是否以正规数组的形式

例如：dump(\$rs,1,'Message 查询结果',0); 有兴趣可以换成这种代码对查询结果 \$rs 进行输出测试

具体操作实践二 添加数据：数组赋值

1 在 Index 模块中再添加一个方法 add2，代码如下：

```
public function add2(){
    // 使用 D 函数实例化一个模型类
    $dao = D('Message');
    // 然后给数据对象赋值
    $data['title'] = '敏捷式开发框架 ThinkPHP!';
    $data['email'] = 'TP@thinkphp.cn';
    $data['create_time'] = time();
    $data['content'] = '使用 ThinkPHP 进行数据操作';
    // 使用 add 方法保存内容
    // 不需要传入主键的值就可以新建数据
    $dao->add($data);
    // 使用 select 方法进行数据查询
    $rs = $dao->select();
```

```
// 使用 dump 函数 输出查询数据查看 注意编码
header("Content-Type:text/html;charset=utf-8");
dump ( $rs );
}
```

2 在浏览器中访问 <http://127.0.0.1/Myapp/index.php/Idnex/add2> ,成功后同样可以看到刚加入的数据。

代码相关知识：

add() 方法

添加数据可以通过 add()方法进行实现，如果表的主键是自动增长类型，不需要传入主键的值就可以新建数据，并且如果插入数据成功的话，add 方法的返回值就是最新插入的主键值，可以直接获取。

例如

```
$insertId = $dao->add($data);
```

select() 方法

对数据进行查询使用 select()方法，当该方法没有传入任何参数时，会获取全部数据。

例如上例中的 \$dao->select()。

对于数据查询方法所涉及内容及各种方法，在下面的章节中会详细再学习。

具体操作实践三 添加数据：利用表单创建数据并添加记录

一般来说，在表中添加数据都是通过表单来建立,下面我们使用 Message 模块进行实践体验。

1 打开 Myapp/Lib/Action/MessageAction.class.php 文件，将原来的操作方法代码删去，并新添一个方法 add(),代码很简单，只要渲染模板进行输出则可，如下所示：

```
public function add() {

    $this->assign('title','添加数据');

    $this->display();

}
```

2 对应 Message 应用模块 ,我们在 Myapp/Tpl/default/目录下新建一个文件夹 ,起名为 Message ,

并在 Myapp/Tpl/default/Message 目录下新添一个模板文件 add.html，关键代码如下：

```
<form action="__URL__/insert" method="post" >
```

```
标题 : <input name="title" type="text" />
邮箱 : <input name="email" type="text" />
内容 : <textarea name="content" rows="6" cols="50" ></textarea>
<input type="submit" value="提交" />
</form>
```

add.html 详细代码请参看附件中的 Myapp/code 章节示例代码/第 5 节/add.html

3 在 MessageAction.class.php 中添加一个表单处理的方法 insert，代码如下：

```
// 处理表单数据的方法
function insert() {
    header("Content-Type:text/html; charset=utf-8");
    $dao = D('Message');
    // 调用 Model 类的 create 方法创建通过表单提交的数据对象
    $vo = $dao->create();
    // 创建失败提示
    if(false === $vo) {
        exit($dao->getError().' [ <a href="javascript:history.back()">返 回</a> ]');
    }
    // 当主键是自动增长类型时，方法的返回值就是最新插入的主键值，可以直接获取
    $rs = $dao->add();
    // 添加数据失败提示
    if (!$rs) {
        exit($dao->getError().' [ <a href="javascript:history.back()">返 回</a> ]');
    }
    // 成功提示并返回，使用了函数 redirect($url,$time=0,$msg=")
    redirect(__APP__.'Message/add',1,'数据添加成功！');
}
```

4 接着，通过访问 <http://127.0.0.1/Myapp/Message/add> 进行表单数据提交测试。

小知识：特殊字符串替换

上面示例中的 action 提交到的处理地址是__URL__/insert，其中__URL__是 TP 内置的特殊变量，在模板中可以自动解析成相应的地址，在上例中，TP 模板引擎会自动将这句替换为 /index.php/Message/insert，常见模板使用的特殊变量还有：

__ROOT__	网站根目录地址
__APP__	当前项目（入口文件）地址
__URL__	当前模块地址
__ACTION__	当前操作地址
__SELF__	当前 URL 地址
__PUBLIC__	公共文件夹 Public 地址

如果要添加或修改模板地址替换，可通过配置 'TMPL_PARSE_STRING' 参数进行设置。

create() 方法

create 方法用来创建数据对象，当参数为空时，默认是通过表单提交的数据。调用该方法后，就可以直接使用 add()方法进行添加数据，而不用再设置插入数据的内容。

getError() 方法

用来捕获数据库操作过程中的出错提示。

小知识：ThinkPHP 内置调试方式

为了方便使用 ThinkPHP 在开发过程中快速解决遇到的问题，框架具有很多特性可以方便进行调试处理。

1 页面 Trace

页面 Trace 功能提供了很好的 SQL 调试方法，而且可以很直观的在当前页面查看到 SQL 语句信息，我们只需要设置参数：`'SHOW_PAGE_TRACE' => TRUE`，这样，我们可以在页面看到类似下面的信息：

SQL 记录：1 条 SQL

[09-04-08 05:38:13] RunTime:0.000238s SQL = SELECT * FROM think_message

2 开启 SQL 日志

页面 Trace 信息虽然直观，但是只能显示当前页面的 SQL 执行信息，如果你使用了 Ajax 操作，那么后台的执行 SQL 可能就不会显示，我们只能通过查看 SQL 日志了。可以在配置参数里面开启 SQL 日志记录：

```
'SQL_DEBUG_LOG' => TRUE,
```

我们可以在 Logs 目录下面的 sql 日志文件里面看到：

```
[ 09-04-08 05:50:12 ]
```

```
RunTime:0.000238s SQL = SELECT * FROM think_message WHERE id in (1,2,5)
```

这样的记录信息，可以分析具体执行的 SQL 语句是否有问题，以及执行时间，便于优化。

3 使用 **getLastSql** 方法

如果你没有开启 SQL 日志，也没有使用页面 Trace 信息显示功能，那么依然可以调试可能出现错误的 SQL 语句，我们可以在查询方法的后面使用 `getLastSql` 方法来查看最后一次执行的 SQL 语句。便于分析错误可能的原因。

```
$dao->find(8);  
echo $dao->getLastSql()
```

4 使用调试配置文件

单独定义调试配置文件，可以给你的项目设置最合理的参数。而且你还可以给调试模式增加单独的数据库连接。

我们可以在项目的 Conf 目录下面增加调试配置，内容如下：

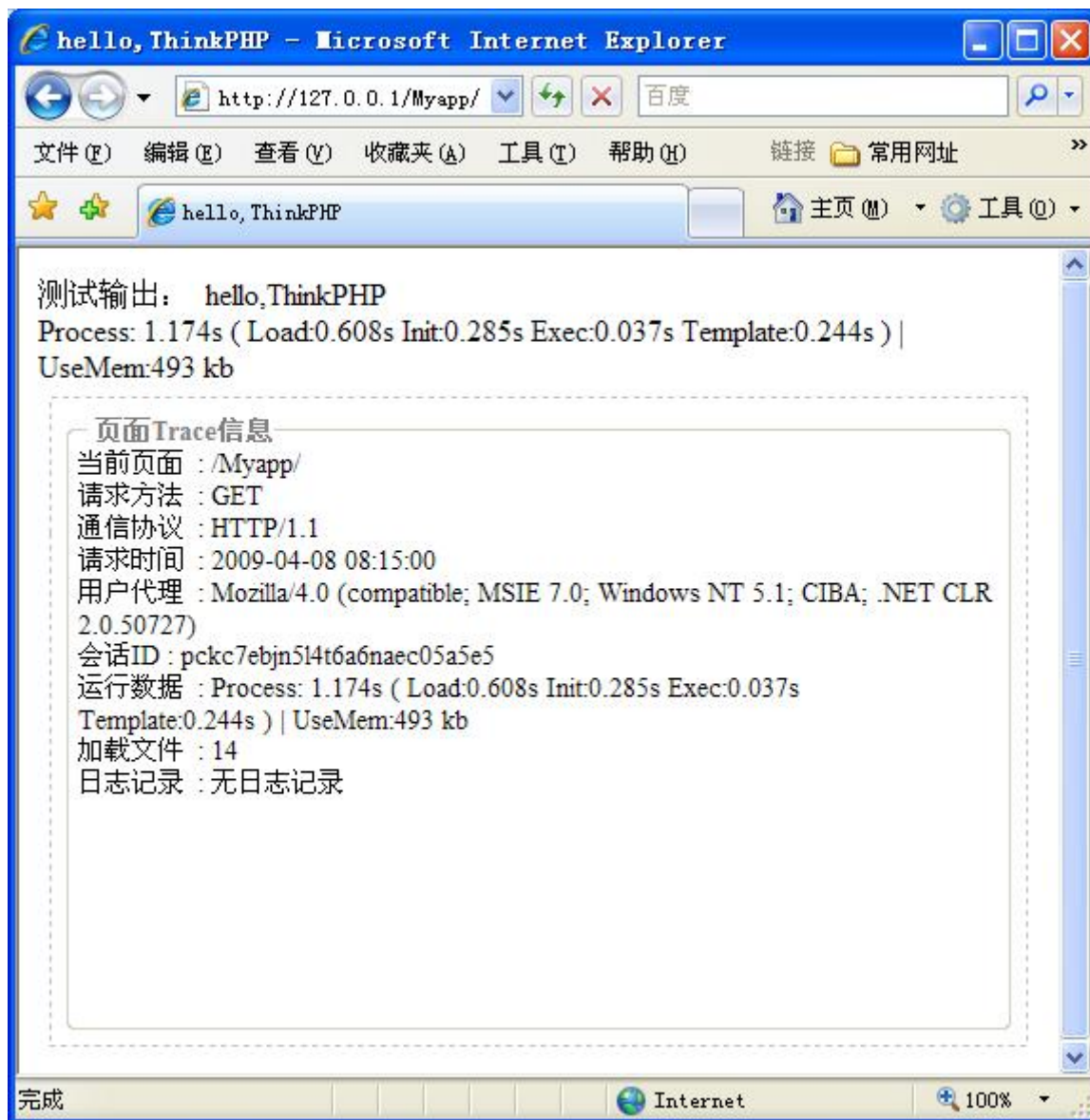
```
'SHOW_RUN_TIME' => TRUE, // 开启运行时间显示  
'SHOW_ADV_TIME' => TRUE, // 显示详细运行时间信息  
'SHOW_DB_TIMES' => TRUE, // 显示数据库查询和写入次数  
'SHOW_RUN_TIME' => TRUE, // 开启运行时间显示
```

5 开启调试模式

只要在配置中设置 `'DEBUG_MODE'=>true`，就可以开启调试模式。

系统默认是调用 `ThinkPHP/Common/debug.php` 的配置设置。要自行定义时，只要在项目 `Myapp/Conf/` 目录下，添加 `debug.php` 文件，然后根据需要进行设置。

开启调试模式之后再进行页面访问，在页面后面就会自动附上相关调试信息，包括了页面执行时间、数据库操作次数、内存使用情况当前加载的类库数量，以及页面执行的 SQL 语句和错误记录等等。这些信息是能够给开发调试带来很大的帮助，而且页面 Trace 信息是可以定制显示的。



当项目开发完成后，需要正式部署到服务器了，建议关闭调试模式，将 '`DEBUG_MODE`' => `true`，改成 '`DEBUG_MODE`' => `false`，或直接删去或注释掉这行代码就可以了。这样又会恢复到原来干净的界面。

9.2 查询记录

ThinkPHP 有一个非常明确的约定，就是单个数据查询和多个数据查询的方法是分开的，或者你会认为有时候自己也不知道要查询的数据是单个还是多个，但是有一点是明确的，你需要的是返回一个数据还是希望返回的是一个数据集。因为对两种类型的返回数据的操作方式是截然不同的，无论何种方式的返回，我们都可以直接在模型对象里面操作，当然也一样可以作为数据传递给你需要的变量。

如何查询记录？

- 1 设置查询条件（如果是查找所有记录，这一步可省略）
- 2 使用最直观易懂的操作方式：[要进行数据操作的模型](#)->[对查询设定（连贯操作）](#)->[进行查询](#)

例如：`$User->where('name=thinkphp')->find();`

基础知识 连贯操作

所谓连贯操作，是 ThinkPHP 使用了魔术方法来实现的一些特殊的 Model 类方法，可以对数据对象连续调用数个方法(参数)来设定特定的条件，然后执行最终所指定的操作，这类特定方法的命名都很直观易记，主要方法有：

field	//要查询的字段名
table	//数据表名称 不指定时，默认为当前数据对象所对应的表
where	//查询条件
order	// 排序
limit	//结果限制
having	//having 支持
group	//group 支持
distinct	//distinct 筛选支持

连贯操作应用代码示例：

```
$User->where('status=1')->table('`other_db`.`think_user`')->order('create_time')->limit(10)->select();
```

数据查询具体操作实践

为了方便学习，我们新建立一个应用模块 Search 专门用来体验 ThinkPHP 的各种查询方式。

先在 Myapp/Lib/Action 目录下，新建一个 PHP 文件，命名为：SearchAction.class.php。并定义类：

```
<?php
// 数据查询体验
class SearchAction extends Action{
    function _initialize( ) {
        // 为了方便使用 dump 函数查看结果输出，先在初始化方法里统一编码
        header("Content-Type:text/html; charset=utf-8");
    }
} //类定义 end
?>
```

1 查询单条记录

当数据表的主键为 id 并且是自动增长时，可以直接使用 find(id 编号)进行查询。

1.1 在 Search 应用模块中新添一个方法，代码如下：

```
// 使用 find 方法查询 1
public function find1(){
    // 实例化模型
    $dao = D('Message');
    // 查询 id 为 3 的记录
    $rs = $dao->find(3);
    // 输出结果
    dump ( $rs );
    // 输出 sql 语句进行对比
    dump( $dao->getLastSql( ) );
}
```

1.2 在浏览器中访问该操作：<http://127.0.0.1/Myapp/index.php/Search/find1>，结果如下图所示：



可以看到很方便地获得了 id 为 3 的单条记录。可以对比 sql 去理解 find 方法

2 查询符合特定条件的单条数据

2.1 新添一个方法 find2，代码如下：

```
// 使用 find 方法查询 2
public function find2() {
    // 实例化模型
    $dao = D('Message');
    // 设置查询条件
    $where['status'] = 0;
    $where['email'] = 'thinkphp@thinkphp.cn';
    // 需要查询的字段
    $field = 'email,title';
    // 进行查询
    $rs = $dao->where($where)->field($field)->find();
    // 输出结果
    dump ( $rs );
    // 输出 sql 语句
    dump( $dao->getLastSql() );
}
```

2.2 访问 find2 操作 <http://127.0.0.1/Myapp/index.php/Search/find2>，结果如下：



可以看到这样进行查询的 sql 就是：

```
"SELECT email,title FROM `think_message` WHERE ( `status` = 0 ) AND ( `email` = 'thinkphp@thinkphp.cn' ) LIMIT 1 "
```

注意：使用 `find()` 方法进行查询，无论符合条件的结果有多少个，都只会返回单条记录。

在连贯操作中的参数（where,field,order 等）顺序是不分先后的，如本例，如果使用

```
$rs = $dao->field($field)->where($where)->find();
```

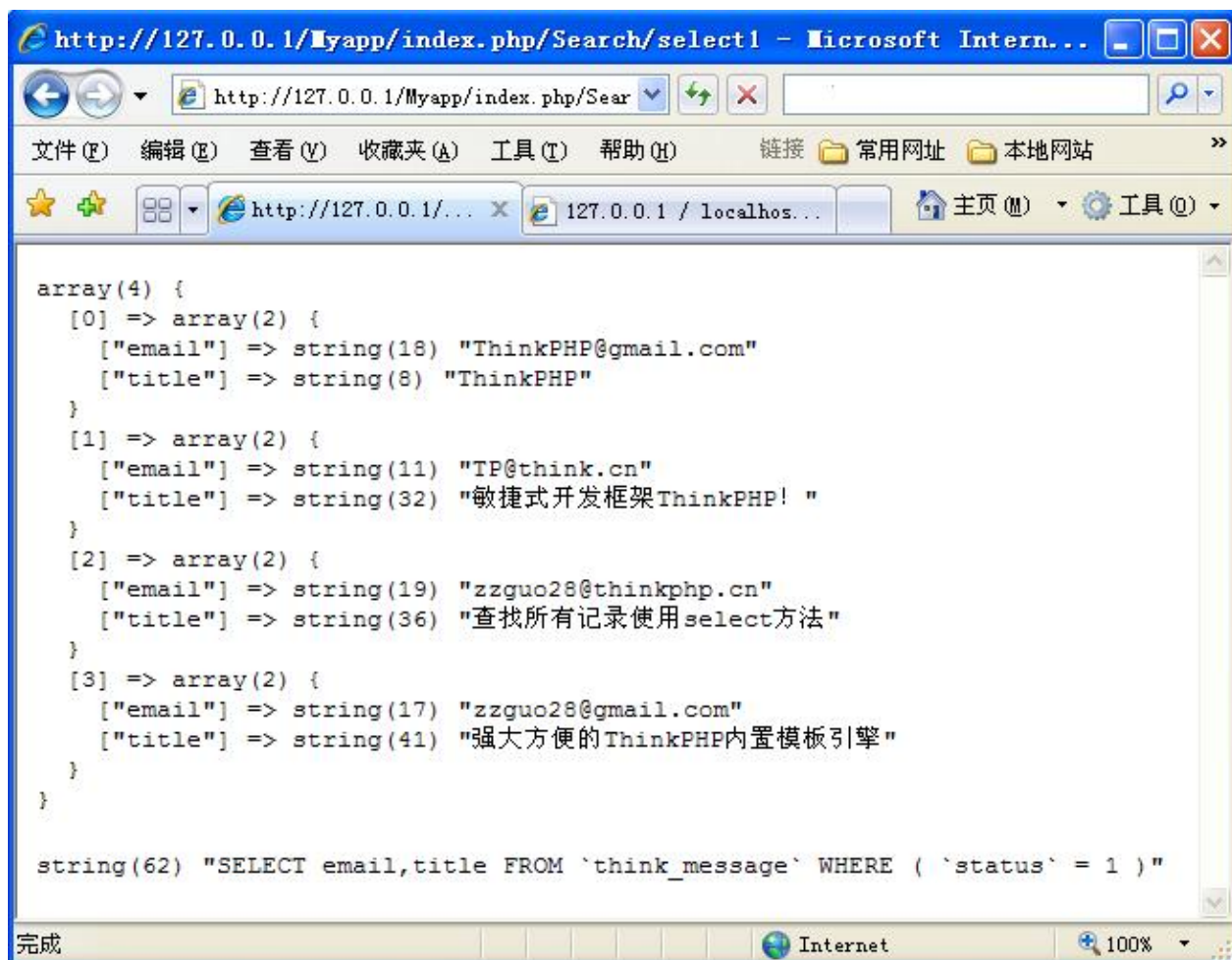
所得的操作结果完全一样，所以不必去记住顺序。

3 获取所有记录

3.1 添加一个方法 `select1`，代码如下：

```
// 获取所有记录集
public function select1() {
    // 实例化模型
    $dao = D('Message');
    // 设置查询条件
    $where['status'] = 1;
    // 需要查询的字段
    $field = 'email,title';
    // 进行查询
    $rs = $dao->where($where)->field($field)->select();
    // 输出结果
    dump ( $rs );
    // 输出 sql 语句进行
    dump( $dao->getLastSql() );
}
```


3.2 访问 select1 操作：<http://127.0.0.1/Myapp/index.php/Search/select1>，结果如下：



```
array(4) {
  [0] => array(2) {
    ["email"] => string(18) "ThinkPHP@gmail.com"
    ["title"] => string(8) "ThinkPHP"
  }
  [1] => array(2) {
    ["email"] => string(11) "TP@think.cn"
    ["title"] => string(32) "敏捷式开发框架ThinkPHP! "
  }
  [2] => array(2) {
    ["email"] => string(19) "zzguo28@thinkphp.cn"
    ["title"] => string(36) "查找所有记录使用select方法"
  }
  [3] => array(2) {
    ["email"] => string(17) "zzguo28@gmail.com"
    ["title"] => string(41) "强大方便的ThinkPHP内置模板引擎"
  }
}

string(62) "SELECT email,title FROM `think_message` WHERE ( `status` = 1 )"
```

同 find 方法一样，连贯操作中，除了 select 方法必须放到最后，参数条件顺序不限。

使用 select 方法进行查询，返回的是一个数据记录集。

select 方法也支持 join,like 等更复杂的查询，具体应用可以参看相关手册。

select 方法还有一个别名 findall，使用方式是一样的。

```
$rs = $dao->where($where)->field($field)->findall();
```

附加知识 1 动态查询

getby 字段名('字段值')

getby 方法可以根据指定字段获取一条数据，例如

```
$data = $dao->getbyemail('thinkphp@thinkphp.cn');
```

使用的 sql 语句是：

```
SELECT * FROM `think_message` WHERE email='thinkphp@thinkphp.cn' LIMIT 1
```

附加知识 2 获取一条记录的某个字段的值

`getField($field,$condition='')`

例如：

```
$data = $dao->where('id=1')->getField('email');
```

使用的 sql 语句是

```
SELECT `email` FROM `think_message` WHERE id =1 LIMIT 1
```

附加知识 3 获取数据集的个别字段值

`getFields($field,$condition="",$sepa=' ')`

```
$data = $dao->getFields('email');
```

使用的 sql 语句是：

```
SELECT `email` FROM `think_message`
```

得出的结果是一个数组，可以指定需要的字段(\$field)，条件(\$condition)和指定分隔符号(\$sepa)

例如：

```
$data = $dao->getFields('id,create_time,email,content','||');
```

使用的 sql 语句是：

```
SELECT id,create_time,email,content FROM `think_message`
```

附加知识 4 使用原生 SQL 进行查询

对于查询可以使用 query 方法直接对 sql 语句执行查询

例如

```
$model = new Model();
```

```
$model->query('select name,email from think_user where status=1');
```

以上各种方法建议在操作中自行去实践体验。

9.3 更新记录

更新记录使用 save 方法：

```
$User->find(1); // 查找主键为 1 的数据
$User->email = 'TOPThink@thinkphp.cn'; // 修改数据对象
$User->save(); // 保存当前数据对象

// 还可以使用下面的方式更新
$User->score = array('exp','(score+1)'); // 对用户的积分加 1
$User->save();
```

如果不是使用数据对象的方式来保存，可以传入要保存的数据和条件

```
$data['id'] = 1;
$data['name'] = 'TopThink';
$User->save($data);
```

除了 save 方法外，你还可以使用 setField 方法来更新特定字段的值，例如：

```
$User->where('id=1')->setField('name','TopThink');
```

使用原生 sql

通过 execute 方法执行 sql 语句，

```
$User->execute("UPDATE __TABLE__ SET name='thinkPHP' WHERE status=1");
```

9.4 删除记录

删除记录使用 delete 方法

删除单条记录

```
$User->find(2);
$User->delete(); // 删除查找到的记录
```

删除多条记录

```
$where['id'] = array('in','1,2,3,4,6,8');  
$User->where($where)->delete();
```

使用原生 sql

```
$User->execute("DELETE FROM __TABLE__ WHERE name='thinkPHP'");
```

10 使用模板

通常一个应用模块中，会有若干操作(function)需要有与用户交互的页面，这就需要用到模板输出，ThinkPHP 本身已内置了一套具有 ThinkPHP 特色的，很强大易扩展且应用非常方便的模板引擎。

如何使用模板？

仅需一步，在 项目/Tpl/风格目录/下，建立一个 html 文件与操作名称相对应就可以了

在应用模块中，如果某个操作需要进行页面输出显示，只要相应地在 Myapp/Tpl/default/目录下建立一个文件夹，文件夹以应用模块的名称来命名，然后在这个文件夹下，建立一个以这个操作 (function) 相同名称的 html 文件，就可以在这个方法中使用 Action 类的 display() 方法直接调用该模板显示。

除此之外，ThinkPHP 模板引擎当然也支持跨风格跨模块调用不同操作的模板使用，相关知识请参阅本章节附加资料。

具体操作实践

- 1 在 Myapp/Tpl/default/下建立一个目录，根据应用模块的名称，我们将这个文件夹命名为 Index
- 2 在 Myapp/Tpl/default/Index/目录下建立一个 html 文件，根据操作名称，我们命名该文件为 index.html

3 打开 Myapp/Lib/Action/IndexAction.class.php 文件，将 index 操作的代码修改为

```
<?php
class IndexAction extends Action{
    public function index(){
        $value = 'hello,ThinkPHP';
        $this->assign('title',$value);
        $this->display( );
    }
}
?>
```

4 打开 Myapp/Tpl/default/Index/index.html 文件,代码为

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>{$title}</title>
</head>
<body>
测试输出： {$title}
</body>
</html>
```

5 在浏览器中输入 <http://127.0.0.1/Myapp/>，我们可以看到，在浏览器的标题栏和页面中，index 方法所定义的模板变量\$title 已成功地输出。

附加知识 1 模板变量赋值

在 Action 类中赋值模板变量使用 assign 方法，无论何种变量类型都统一使用 assign 赋值。如下：

```
$this->assign('title',$value);
```

模板变量赋值后就需要调用模板文件来输出相关的变量，模板调用通过 display 方法来实现

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

在模板中，不同的模板变量类型，使用不同的变量符号，其中标签可以在配置文件中进行定义，默认的变量输出使用一对大括号{ }标签包围，例如本例的{\$title}。

系统还支持一些常用的变量输出快捷标签，包括：

```
{:function(...)} //执行方法并输出返回值
{~function} //执行方法不输出
{@var} //输出 Session 变量
{#var} //输出 Cookie 变量
{&var} //输出配置参数
{%var} //输出语言变量
{.var} //输出 GET 变量
{^var} //输出 POST 变量
{*var} //输出常量
```

如果需要输出二维数组，例如要输出\$_SESSION['var1']['var2']的值快捷输出可以使用：

[{@var1.var2}](#)的方式

同理，{#var1.var2} 可以输出 \$_COOKIE['var1']['var2']的值

如果要同时输出多个模板变量，可以使用下面的方式：

```
$array = array();
$array['name'] = 'thinkphp';
$array['email'] = 'liu21st@gmail.com';
$array['phone'] = '12335678';
$this->assign($array);
```

这样，就可以在模板文件中同时输出 name、email 和 phone 三个变量。

附加知识 2 模板调用

模板变量赋值后就需要调用模板文件来输出相关的变量，模板调用通过 display 方法来实现，下面给出了 display 方法的基本使用。

假设当前操作是 Index 模块下面的 index 操作

1 直接调用 index 操作模板

```
$this->display();  
  
// 对应 Myapp/Tpl/default/Index/index.html
```

2 调用 Index 模块的 test1 操作模板

```
$this->display('test1');  
  
//对应 Myapp/Tpl/default/Index/test1.html
```

2 调用 Message 模块的 test2 操作模板()

```
$this->display('Message:test2');  
  
//对应 Myapp/Tpl/default/Message/test2.html
```

3 调用 XP 主题的 Message 模块的 test2 操作模板

```
$this->display('Xp@Message:test2');  
  
//对应 Myapp/Tpl/Xp/Message/test2.html
```

4 直接指定模板文件的全名

```
$this->display('../Message/test3.html');
```

```
//对应 Myapp/Tpl/default/Message/test3.html
```

附加知识 3 模板变量循环输出

在模板中对数据进行循环输出使用 `iterate` 标签。

模版变量赋值

```
$Message = D('Message');  
$list = $Message->select();  
$this->assign('list',$list);
```

模版定义

```
<iterate name="list" id="vo">  
{ $vo.name }  
</iterate>
```

注意 `name` 和 `id` 表示的含义，`name` 是已赋值的模板变量，`id` 是在这个循环中所使用的变量。

`id=' vo'` 中的 `vo` 即 `foreach($list as $val)` 中的 `$val`。

// 输出 `list` 的第 5 ~ 15 条记录

```
<iterate name="list" id="vo" offset="5" length='10'>  
{ $vo.name }  
</iterate>
```

// 输出偶数记录

```
<iterate name="list" id="vo" mod="2" >  
<eq name="mod" value="1">  
{ $vo.name }  
</eq>  
</iterate>
```

// 输出 `key`

```
<iterate name="list" id="vo" key="k" >
```

```
{ $k }. { $vo.name }  
</iterate>
```

iterate 还有其它别名是 volist, sublist

子循环输出

```
<volist name="list" id="vo">  
<sublist name="vo['sub']" id="sub">  
    { $sub.name }  
</sublist>  
</volist>
```

总结：到此为止，我们已经了解了如何使用 ThinkPHP 进行简单的开发，当然，ThinkPHP 的殿堂还有更多更好的功能在等着你，再次希望您的 ThinkPHP 学习之旅顺利、工作顺利！