一 . 入门简介：

1. Thinkphp的目录如下：
2. ├─index.php     项目入口文件
3. ├─Common 项目公共文件目录
4. ├─Conf 项目配置目录
5. ├─Lang 项目语言目录
6. ├─Lib 项目类库目录
7. │  ├─Action Action类库目录
8. │  ├─Behavior 行为类库目录
9. │  ├─Model 模型类库目录
10. │  └─Widget Widget类库目录
11. ├─Runtime 项目运行时目录
12. │  ├─Cache 模板缓存目录
13. │  ├─Data 数据缓存目录
14. │  ├─Logs 日志文件目录
15. │  └─Temp 临时缓存目录
16. └─Tpl 项目模板目录

项目目录，只要在任意目录，建一个文件夹，例如test，然后在example中找一个工程下的index.php ,copy进来，修改其中的

// 加载框架入口文件

require( "../../ThinkPHP/ThinkPHP.php");

然后通过浏览器访问该工程， 则目录会自动生成。可以部署多个项目共用thinkphp文件。

1. 配置文件 ：项目目录的Conf/config.php
2. 调试模式：define('APP\_DEBUG',TRUE);
3. 控制器：**模块名+Action.class.php** （模块名采用驼峰法并且首字母大写），类名和文件名一致，可用protected或者private，控制访问。默认系统会访问Index/index; 注意：模块名访问时要大写。
4. 视图：ThinkPHP内置了一个编译型模板引擎，也支持原生的PHP模板，并且还提供了包括Smarty在内的模板引擎驱动。和Smarty不同，ThinkPHP在渲染模板的时候如果不指定模板，则会采用系统默认的定位规则，其定义规范是 **Tpl/模块名/操作名.html**，所以，Index模块的index操作的默认模板文件位于项目目录下面的Tpl/Index/index.html。示例： $this->display();显示模板内容
5. url ，四种模式
6. **普通模式**：<http://localhost/app/?m=module&a=action&var=value>
7. **PATHINFO模式**：是系统的默认URL模式，提供了最好的SEO支持: <http://localhost/app/index.php/module/action/var/value/;可定制>

'URL\_PATHINFO\_DEPR'=>'-', // 更改PATHINFO参数分隔符。

1. **REWRITE模式**：是在PATHINFO模式的基础上添加了重写规则的支持，可以去掉URL地址里面的入口文件index.php，在.htaccess中添加内容：

<IfModule mod\_rewrite.c>

RewriteEngine on

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteCond %{REQUEST\_FILENAME} !-f

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

</IfModule>

示例：

<http://localhost/app/module/action/var/value/>

1. **兼容模式**：是用于不支持PATHINFO的特殊环境，URL地址是：
2. http://localhost/app/?s=/module/action/var/value/

复制代码

兼容模式配合Web服务器重写规则的定义，可以达到和REWRITE模式一样的URL效果。

二 . 数据库操作

1. 数据库配置：

'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'    => '', // 密码

'DB\_PORT'   => 3306, // 端口

'DB\_PREFIX' => 'think\_', // 数据库表前缀

或者 ：'DB\_DSN' => 'mysql://root@localhost:3306/thinkphp'

格式 ：**数据库类型://用户名:密码@数据库地址:数据库端口/数据库名（此优先级高）**

1. Curd操作：add、save、select和delete方法表示模型的CURD操作.

$Form   =   M('Form');

$data =   $Form->find($id); // 读

$title = $Form->where('id=3')->getField('title');//读某一字段

? find 和where的区别

$Form->save(); //更新用save方法

1. 写数据库流程：

D函数实例化模型类，一般需要对应一个数据模型类，而且create方法会自动把表单提交的数据进行自动验证和自动完成（如果有定义的话），如果自动验证失败，就可以通过模型的getError方法获取验证提示信息，如果验证通过，就表示数据对象已经成功创建，但目前只是保存在内存中，直到我们调用add方法写入数据到数据库。这样就完成了一个完整的Create操作，所以可以看到ThinkPHP在创建数据的过程中使用了两步：  
第一步，create方法创建数据对象，  
第二步，使用add方法把当前的数据对象写入数据库。  
当然，你完全可以跨过第一步，直接进行第二步，但是这样的预处理有几个优势：  
1、无论表单有多复杂，create方法都可以用一行代码轻松创建数据对象；  
2、在写入数据之前，可以对数据进行验证和补充；  
其实create方法还有很多的功能操作，目的只有一个，确保写入数据库的数据安全和有效。  
我们来验证下表单提交的效果，当我们不输入标题就直接提交表单的话，系统会给出标题必须这样的提示信息。

示例：如果数据内部写入，可以这么写，

$Form   =   D('Form');

$data['title']  =   'ThinkPHP';

$Form->add($data);

对象方式：

$Form   =   D('Form');

$Form->title  =   'ThinkPHP';

$Form->add();

//保存

$Form->save($data);// 也可以用对象方式保存

$Form->save()

$Form->where('id=5')->save($data); / 根据条件保存修改的数据

或

$Form->where('id=5')->save(); // 根据条件保存修改的数据

对于统计字段，系统还提供了更加方便的setInc和setDec方法。

1. $User->where('id=5')->setInc('score',3); // 用户的积分加3
2. $User->where('id=5')->setInc('score'); // 用户的积分加1
3. $User->where('id=5')->setDec('score',5); // 用户的积分减5
4. $User->where('id=5')->setDec('score'); // 用户的积分减1

删除数据：

$Form->delete(5); // 单个

$User->delete('1,2,5'); // 多个

$User->where('status=0')->delete(); //条件删除 删除所有状态为0的用户数据

1. 字母函数：
2. M(tableName) 针对数据表,有基本的curd操作，
3. D(tableName),和M相同，需要对应的model类
4. 查询

$User = M("User"); // 实例化User对象

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

使用数组作为查询条件

$condition['name'] = 'thinkphp';

$condition['account'] = 'thinkphp';

$condition['\_logic'] = 'OR'; //逻辑判断

$User->where($condition)->select();

以上是简单的相等表达式，还有更多查询方法。

|  |  |
| --- | --- |
| **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（<=） |
| 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 |

使用方法：

$map['id']  = array('eq',100);

$map['id']  = array('neq',100);

$map['id']  = array('gt',100);

$map['name'] = array('like','thinkphp%');

如果配置了DB\_LIKE\_FIELDS参数的话，某些字段也会自动进行模糊查询。例如设置了：

'DB\_LIKE\_FIELDS'=>'title|content'

$map['title'] = 'thinkphp';

查询条件就会变成 title like '%thinkphp%'

支持数组方式：

$map['a'] =array('like',array('%thinkphp%','%tp'),'OR');

$map['b'] =array('notlike',array('%thinkphp%','%tp'),'AND');

$map['id']  = array('between','1,8');

等效于

$map['id']  = array('between',array('1','8'));

$map['id']  = array('not in','1,5,8');

等效于

$map['id']  = array('not in',array('1','5','8'));

**EXP**：表达式，支持更复杂的查询情况

$map['id']  = array('in','1,3,8');可以改成

$map['id']  = array('exp',' IN (1,3,8) ');

exp查询的条件不会被当成字符串， 所以后面的查询条件可以使用任何SQL支持的语法，包括使用函数和字段名称。查询表达式不仅可用于查询条件，也可以用于数据更新，

$data['name'] = 'ThinkPHP';

$data['score'] = array('exp','score+1');// 用户的积分加1

$User->where('id=5')->save($data); // 根据条件保存修改的数据

快捷查询：

1 . $map['name|title'] = 'thinkphp';

$User->where($map)->select();

等效于： name= 'thinkphp' OR title = 'thinkphp'

2 . $map['status&title'] =array('1','thinkphp','\_multi'=>true); 相当于

status= 1 AND title = 'thinkphp'

'\_multi'=>true必须加在数组的最后，表示当前是多条件匹配

3 . $map['status&score&title'] =array('1',array('gt','0'),'thinkphp','\_multi'=>true);

status= 1 AND score >0 AND  title = 'thinkphp'

注意：快捷查询方式中“|”和“&”不能同时使用。

区间查询：

$map['id'] = array(array('gt',1),array('lt',10)) ;

=

(`id` > 1) AND (`id` < 10)

$map['id'] = array(array('gt',3),array('lt',10), 'or') ; =

(`id` != 6) AND (`id` > 3)

最后一个可以是AND、 OR或者 XOR运算符，如果不写，默认是AND运算。

区间查询的条件可以支持普通查询的所有表达式，也就是说类似LIKE、GT和EXP这样的表达式都可以支持。另外区间查询还可以支持更多的条件，只要是针对一个字段的条件都可以写到一起

$map['name']  = array(array('like','%a%'), array('like','%b%'), array('like','%c%'), 'ThinkPHP','or');

=

(`name` LIKE '%a%') OR (`name` LIKE '%b%') OR (`name` LIKE '%c%') OR (`name` = 'ThinkPHP')

组合查询

三 . 视图模板标签

1. volist

$this->data = $Data->select();

1. 模板：<volist name="data" id="vo">
2. {$vo.id}--{$vo.data}<br/>
3. </volist>

相当于读取表中对应字段 id 、 data

注意事项

1. ThinkPHP在没有找到对应操作方法的情况下，会检查是否存在对应的模板文件，由于我们有对应的add模板文件，所以控制器就直接渲染该模板文件输出了。所以说对于没有任何实际逻辑的操作方法，我们只需要直接定义对应的模板文件就行了。