河北师大软件学院 *@Software College*

前端开发与HTML5
程序设计基础

王岩

2.13 MVC与ThinkPHP框架（三）

模型

模型定义

模型定义

- ThinkPHP内置model对象，提供数据库操作的各项方法
  - 该对象的定义位置在ThinkPHP/Library/Think/Model.class.php
- 自定义Model对象必须继承自该对象
- 查看model对象定义文件

模型定义

- M（）函数实例化
  - 参数为要操作的表名
  - 实例化基础模型（model类）
    - 特点：简单高效，无需自定义模型；无法使用自定义业务逻辑，只能完成基本CURD操作。
- D（）函数实例化
  - 参数为要实例化的自定义模型类名
  - 实例化已定义的模型类
  - 特点：可在模型对象中封装自定义的业务逻辑

CURD操作

- CURD是一个数据库技术中的缩写词，一般的项目开发的基本功能都是CURD。是数据库处理数据的原子操作。
  - 创建（Create）
  - 更新（Update）
  - 读取（Read）
  - 删除（Delete）

创建（Create）

- 内置model对象的create（）方法自动从表单获取数据并组织成数据对象
- 内置model对象的add（）方法完成数据插入操作
- 示例：
  - 创建数据库表
  - 创建表单页面
  - 创建控制器
  - 完善控制器代码，完成插入操作

create( )

- create（）方法用于组织要插入数据库或者进行修改的数据，组织之后的数据称为数据对象
- 参数（可选）：
  - 保存有数据的关联数组
  - 如不设置参数，则方法自动获取$_POST数组中的数据
  - 注意：数组键名必须对应数据表各项字段名称，否则数据无法正确插入。
- 使用此方法使从表单中获取数据并保存到数据库的操作更加便捷

add( )

- add（）方法完成数据的插入操作
- 参数（可选）：
  - 保存有数据的二维数组
    - 如果不设置参数，则方法使用当前数据对象中的参数
- 返回值：
  - 失败返回false
  - 插入成功并且表中具有自增字段则返回新增的主键值

读取数据（Read）

- Model对象
  - select（）方法，查询所有数据
    - 参数：值为数字或字符串，表示主键值。为空时表示查询所有数据
    - 返回值：成功则返回包含数据的二维数组
  - find（）方法，查询一条数据
    - 参数：值为数字或字符串，表示主键值。为空时表示不限条件
    - 返回值：成功则返回包含数据的数组
- 示例
  - 创建并实现控制器代码，完成数据库读取
  - 创建视图页面，显示数据

更新数据（Update）

- Model对象
  - save（）方法，进行修改操作
  - 根据数据数组中的主键字段值进行匹配修改
- 参数：
  - 保存有数据的二维数组
  - 如果不设置参数，则方法使用当前数据对象中的参数
- 示例：
  - 创建显示控制器、视图，实现原有数据展示
  - 创建修改控制器，获取更新之后的数据，并创建数据对象保存在数据库中

删除数据（Delete）

- Model对象
  - delete（）方法，删除数据
- 参数：
  - 值为数字或字符串，表示主键值。为空时表示查询所有数据
  - 如果执行成功返回删除的个数，失败则返回false

查询条件

设定条件

- 指定where条件
- 指定排序（order by）
- 指定数据条数（limit）
- 指定查询字段
- 分组查询（group by）
- 联合查询（join）
- …

where()方法

- 内置model对象提供where方法，用于在查询之前设置条件，设置之后可用于select、find、save、delete方法。
- 使用字符串作为参数：
- 例：

```
$User = M('user');
$User->where('type=1 and status=1');
$User->select();
```

- 生成的sql：select * from user where type=1 and status=1
- 字符串参数的缺点：安全性不高

where()方法

- 使用数组作为查询条件
- 设置表现条件的关联数组，并传递给where方法
- 当有多个条件关系时，默认逻辑关系是与，即AND
- 例：

```
$User = M('User');
$condition['name'] = 'testUser';
$condition['type'] = 1;
$User->where($condition);
$User->select();
```

- 生成的SQL：select * from user where name=‘testUser’ and type=1
- 推荐使用数组的方式设置条件，ThinkPHP在转换为SQL时，会对数组的条件进行安全过滤。

where()方法

- 使用数组作为查询条件，并且使用逻辑或关系
- 在条件数组中增加一项'_logic'，用于设置逻辑关系

```php
$User = M('User');
$condition['name'] = 'testUser';
$condition['type'] = 1;
$condition['_logic'] = 'OR';
$User->where($condition);
$User->select();
```

- 生成的SQL：select * from user where name='testUser' OR type=1

where()方法

- 细化字段条件
- 格式：
- $condition[‘字段名’] = array(‘表达式’，‘条件值’)

```php
//查询条件: id=100
$condition['id'] = array("eq", 100);
//查询条件: id<>100
$condition['id'] = array('neq', 100);
//查询条件: id>100
$condition['id'] = array('gt', 100);
//查询条件: id<100
$condition['id'] = array('lt', 100);
```

where()方法

- 细化字段条件
  - 表达式种类

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

其他条件设置

- 跟where类似，model对象中提供多种方法用于设置使用条件：

方法名	作用
order	对查询结果排序
limit	限制查询的结果范围
field	指定要查询的字段
join	在查询中加入join支持
group	在查询中加入分组支持
…	…

连贯操作

- 设置条件的方法在调用之后，可以直接调用下一个方法，称为连贯操作。

```
$User = M('User');
$condition['name'] = 'testUser';
$condition['type'] = 1;
$User->where($condition)->order("id")->limit(0,10)->select();
```

- 设置条件的方法必须要在查询方法前调用，查询方法有：select、find、getField、save、delete
- 连贯操作原理：每个设置条件的方法的返回值都是当前模型对象

order()方法

order 用于对操作结果排序	
用法	order($order)
参数	order（必须）：排序的字段名，支持字符串和数组，支持多个字段排序
返回值	当前模型实例
备注	如果不调用order方法，按照数据库的默认规则

```php
$User = M('User');
$User->order("id desc")->select();
$User->order("status desc,id asc")->select();
$User->order(array('status'=>'desc', 'id'=>'asc'))->select();
```

limit()方法

limit 用于定义要查询的结果限制（支持所有的数据库类型）	
用法	limit($limit)
参数	limit（必须）：限制数量，支持字符串
返回值	当前模型实例
备注	如果不调用limit方法，则表示没有限制

```php
$User = M('User');
//查询前十条数据
$User->limit(10)->select();
$User->limit(0,10)->select();
//查询第5-15条数据
$User->limit("4, 10")->select();
```

field()方法

field 用于定义要查询的字段	
用法	field($field,$except=false)
参数	field（必须）：字段名，支持字符串和数组，支持指定字段别名；如果为true则表示显式或者数据表的所有字段。 except（可选）：是否排除，默认为false，如果为true表示定义的字段为数据表中排除field参数定义之外的所有字段。
返回值	当前模型实例
备注	如果不调用field方法，则默认返回所有字段，和field（'*'）等效

```
$User = M('User');
$User->field('id,nickname as name')->select();
$User->field(array('id','nickname'=>'name'))->select();
```

group()方法

group 用于数据库的group查询支持	
用法	group($group)
参数	group（必须）：group的字段名，支持字符串
返回值	当前模型实例
备注	无

```php
$User = M('User');
$User->group('user_id')->select();
```

join()方法

join 用于数据库的join查询支持	
用法	join($join)
参数	join（必须）：join操作，支持字符串和数组
返回值	当前模型实例
备注	join方法支持多次调用

```php
$User = M('User');
//支持调用多次，默认使用左连接
$User->join(' work ON artist.id = work.artist_id')
->join('card ON artist.card_id = card.id')->select();
//使用其他连接方式
$Model->join('RIGHT JOIN work ON artist.id =
 work.artist_id')->select
```

其他方法

方法名称	作用
union	用于对查询的union支持
having	用于对查询的having支持
distinct	用于对查询的distinct支持
lock	用于数据库的锁机制
cache	用于查询缓存
table	用于定义要操作的数据表名称
data	用于新增或者更新数据之前的数据对象赋值

统计查询

方法	说明
Count	统计数量，参数是要统计的字段名（可选）
Max	获取最大值，参数是要统计的字段名（必须）
Min	获取最小值，参数是要统计的字段名（必须）
Avg	获取平均值，参数是要统计的字段名（必须）
Sum	获取总分，参数是要统计的字段名（必须）

原生查询

- query()

query 执行SQL查询操作	
用法	query($sql,$parse=false)
参数	sql（必须）：要查询的SQL语句 parse（可选）：是否需要解析SQL
返回值	如果数据非法或者查询错误则返回false 否则返回查询结果数据集（同select方法）

```
$Model = new Model() // 实例化一个model对象 没有对应任何数据表
$Model->query("select * from think_user where status=1");
```

原生查询

- execute()

execute用于更新和写入数据的sql操作	
用法	execute($sql,$parse=false)
参数	sql（必须）：要执行的SQL语句 parse（可选）：是否需要解析SQL
返回值	如果数据非法或者查询错误则返回false 否则返回影响的记录数

```
$Model = new Model() // 实例化一个model对象 没有对应任何数据表
$Model->execute("update think_user set name='thinkPHP'
 where status=1");
```

数据自动校验

数据自动校验

- 自动验证是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自动进行数据验证。
- 使用模型的数据自动校验可方便的验证要进行数据操作的数据，保证数据的合法性
- 数据自动校验简单上手：
  - 建立表单页面视图文件
  - 建立模型对象，在对象中设置$_validate属性值

```php
$_validate = array(
  array("name", "require", "名称必填")
);
```

  - 建立控制器类
  - 测试表单数据的提交

验证规则写法

```
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]),
    ......
);
```

验证规则参数

- 验证规则参数：
  - 验证字段（必须）：进行校验的表单项名称
  - 验证规则（必须）：
    - 内置：require、email、url、number
    - 配合附加规则使用
- 提示信息（必须）：在验证不通过时的提示信息
- 验证条件（可选）：
  - 0 存在字段就验证（默认）
  - 1 必须验证
  - 2 值不为空的时候验证

内置验证规则

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
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
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)
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。

验证规则参数

- 验证时间（可选），取值：
  - 1：新增数据时候验证
  - 2：编辑数据时候验证
  - 3：新增/编辑数据时均验证

自动校验示例

- 建立验证规则：

```php
class UserModel extends Model{
  protected $_validate = array(
    //默认情况下验证字段必须有值
    array('verify','require','验证码必须！'),
    // 在新增的时候验证name字段是否唯一
    array('name','','帐号名称已经存在！',0,'unique',1),
    // 当值不为空的时候判断是否在一个范围内
    array('value',array(1,2,3),'值的范围不正确！',2,'in'),
    // 验证确认密码是否和密码一致
    array('repassword','password','确认密码不正确',0,'confirm'),
    // 自定义函数验证密码格式
    array('password','checkPwd','密码格式不正确',0,'function'),
  );
}
```

自动校验示例

- 进行校验

```
$User = D("User"); // 实例化User对象
 if (!$User->create()){
    // 如果创建失败 表示验证没有通过 输出错误提示信息
    exit($User->getError());
}else{
    // 验证通过 可以进行其他数据操作
}
```

数据自动完成

数据自动完成

- 自动完成是ThinkPHP提供用来完成数据自动处理和过滤的方法，使用create方法创建数据对象的时候会自动完成数据处理。

- 自动完成通常用来完成默认字段写入，安全字段过滤以及业务逻辑的自动处理等，和自动验证的定义方式类似

- 定义规则：

- 设置自定义对象的$_auto属性

- 规则格式：

```
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]),
    ......
 );
```

自动完成规则

- 规则字段：
  - 完成字段：（必须）需要进行处理的数据表实际字段名称。
  - 完成规则：（必须）需要处理的规则，配合附加规则完成。
- 完成条件：（可选）包括：
  - 1 新增数据的时候处理（默认）
  - 2更新数据的时候处理
  - 3所有情况都进行处理

自动完成规则

- 附加规则：

function	使用函数，表示填充的内容是一个函数名
callback	回调方法 ，表示填充的内容是一个当前模型的方法
field	用其它字段填充，表示填充的内容是一个其他字段的值
string	字符串（默认方式）
ignore	为空则忽略（3.1.2新增）

数据自动完成示例

```php
class UserModel extends Model{
    protected $_auto = array (
        // 新增的时候把status字段设置为1
        array('status','1'),
        // 对password字段在新增和编辑的时候使md5函数处理
        array('password','md5',3,'function') ,
        // 对name字段在新增和编辑的时候回调getName方法
        array('name','getName',3,'callback'),
        // 对update_time字段在更新的时候写入当前时间戳
        array('update_time','time',2,'function'),
    );
}
```

谢谢！