**ThinkPHP5**

# 数据库的操作:

*//原生的代码  
 //插入数据* $result = Db::*execute*(**'insert into think\_data (name,status) values("周洁",1)'**);  
 $result = Db::*execute*(**'insert into think\_data (name,status) values("小明",1)'**);  
 $result = Db::*execute*(**'insert into think\_data (name,status) values("小红",1)'**);  
 *//修改数据* $result = Db::*execute*(**'update think\_data set name="张学友",status = 0 where id =1'**);  
 *//删除数据* $result = Db::*execute*(**'delete from think\_data where id =1'**);  
 *//查询的方法* $result = Db::*query*(**'select** *\** **from think\_data'**);  
 dump($result);

*//构造器  
 //插入记录* $result = Db::*table*(**'think\_data'**)->insert([**'name'**=>**'小黄'**,**'status'**=>1]);  
 *//更新记录* $result = Db::*table*(**'think\_data'**)->where(**'id'**,2)->update([**'name'**=>**"刘德华"**,**'status'**=>0]);  
 *//查询条件* $result = Db::*table*(**'think\_data'**)->where(**'id'**,2)->select();dump($result);  
 *//删除条件* $result = Db::*table*(**'think\_data'**)->where(**'id'**,2)->delete();

***助手函数db*** $db = db(**'data'**);  
 $db->insert([**'name'**=>**"林明"**,**'status'**=>1]);  
 $db->where(**'id'**,5)->update([**'name'**=>**'郭富城'**,**'status'**=>2]);  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 ***链式操作*** $list=Db::*name*(**'data'**)->where(**'status'**,1)->field(**'id'**,**'name'**)->or der(**'id'**,**'desc'**)->limit(3)->select();  
 dump($list);  
  
 $result = Db::*name*(**'data'**)->where(**'status'**,1)->find();  
 $result = Db::*name*(**'data'**)->where(**'status'**,1)->select();  
 ***模糊查询*** $result = Db::*name*(**'data'**)->where(**'name'**,**'like'**,**'%黄%'**)->select();  
 ***区间查询*** $result = Db::*name*(**'data'**)->where(**'id'**,**'BETWEEN'**,[3,6])->select();  
  
 ***一次性插入多条数据*** $data = [  
 [**'name'**=>**'林志凌'**,**'status'**=>1],  
 [**'name'**=>**'林志颖'**,**'status'**=>2],  
 ];  
 $result = Db::*name*(**'data'**)->insertAll($data);  
  
 ***更改字段值*** $result = Db::*name*(**'data'**)->where(**'id'**,3)->setfield(**'name'**,**'thinkphp'**);  
  
  
 ***更改自增和自减字段值 setInc***/***setDec*** $result = Db::*name*(**'data'**)->where(**'id'**,3)->setInc(**'status'**);*//+1* $result = Db::*name*(**'data'**)->where(**'id'**,4)->setInc(**'status'**,3);*//+3* $result = Db::*name*(**'data'**)->where(**'id'**,5)->setDec(**'status'**);*//-1* $result = Db::*name*(**'data'**)->where(**'id'**,9)->setDec(**'status'**,2);*//-2* dump($result);

# 模型定义:

模型操作之增加

**use** app\index\model\User **as** UserModel;  
**class** User  
{  
 新增一条数据的方法  
 **public function** add()  
 {  
 ***对象*** $user = **new** UserModel();  
 $user->**name** = **'刘云'**;  
 $user->**email** = **'liuyun@qq.com'**;  
 $user->**brithday** = *strtotime*(**'1989-7-11'**);  
 **if**($user->save()) {  
 **return '用户新增成功'**;  
 } **else** {  
 **return '用户新增失败'**;  
 }  
  
 ***数组*** $user[**'name'**] = **'看云'**;  
 $user[**'email'**] = **'kanyun@qq.com'**;  
 $user[**'brithday'**] = *strtotime*(**'1999-6-12'**);  
 **if**($result = UserModel::*create*($user)) {  
 **return '用户新增成功'**;  
 } **else** {  
 **return '用户新增失败'**;  
 }  
  
 }  
  
 批量新增数据  
 **public function** addList()  
 {  
 $user = **new** UserModel() ;  
 $list = [  
 [**'name'** =>**'张三'**,**'email'**=>**'zhangsan@qq.com'**,**'brithday'**=>*strtotime*(**'1998-02-11'**)],  
 [**'name'**=>**'李四'**,**'email'**=>**'lisi@qq.com'**,**'brithday'**=>*strtotime*(**'1989-02-11'**)],  
 ];  
 **if**($user->saveAll($list)) {  
 **return '用户批量新增成功'**;  
 } **else** {  
 **return '用户新增失败'**;  
 }  
 }

模型操作之更新

更新数据  
查找并更新  
**public function** update()  
{  
 $user = UserModel::*get*(1);  
 $user->**name** = **'安迪'**;  
 $user->**email** = **'andi@qq.com'**;  
 **if**($user->save()) {  
 **return '更新数据成功'**;  
 } **else** {  
 **return '更新数据失败'**;  
 }  
  
  
 ***直接更新数据*** $user = UserModel::*get*(1);  
 $user->save([**'name'**=>**'刘号'**,**'email'**=>**'liutao@qq.com'**],[**'id'**=>1]);  
  
 ***批量更新数据*** $user = **new** UserModel();  
 $list = [  
 [**'id'**=>2,**'name'**=>**'小明'**,**'email'**=>**'xiaoming@qq.com'**],  
 [**'id'**=>3,**'name'**=>**'小红'**,**'email'**=>**'xiaohong@qq.com'**]  
 ];  
 $user->saveAll($list);  
  
 ***通过数据库类更新数据*** $user = **new** UserModel();  
 $user->update([**'id'**=>1,**'name'**=>**'thinkphp'**]);  
  
 ***静态方法*** UserModel::*update*([**'id'**=>1,**'name'**=>**'haonan'**]);

}

模型操作之查询

*//查询数据* **public function** select() {  
 *//单个数据查询* $user = UserModel::*get*(1);  
 **echo** $user->**name**.**'<br />'**;  
 **echo** $user->**email**.**'<br />'**;  
  
 $user = **new** UserModel();  
 $result = $user->where(**'name'**,**'小明'**)->find();  
 **echo** $result->**id**;  
  
 *//获取多个数据查询  
 //数组查询* $list = UserModel::*all*([1,2,3]);  
 **foreach**($list **as** $key=>$value) {  
 **echo** $value->**name**.**'<br />'**;  
 }  
  
 $user = **new** UserModel();  
 $result = $user->where(**'status'**,1)->limit(2)->order(**'id'**,**'desc'**)->select();  
 **foreach**($result **as** $key=>$value) {  
 **echo** $value[**'name'**].**'<br />'**;  
 **echo** $value[**'email'**].**'<br />'**;  
 }  
  
 *//聚合函数的调用* $user = **new** UserModel();  
 **echo** $user->Count(**'id'**); *//id之和* **echo** $user->Max(**'score'**);  
}

模型操作之删除

*/删除数据* **public function** delete()  
 {  
  
 /$user = UserModel::*get*(1);  
 **if**($user->delete()) {  
 **return '删除数据成功'**;  
 } **else** {  
 **return '删除数据失败'**;  
 }  
 *//根据主键来删除  
 //删除单条数据* **if**(UserModel::*destroy*(2)) {  
 **return '数据删除成功'**;  
 } **else** {  
 **return '数据删除失败'**;  
 }  
 *//删除多条数据* **if**(UserModel::*destroy*([3,4])) {  
 **return '删除多条数据成功'**;  
 } **else** {  
 **return '删除多条数据失败'**;  
 }  
  
 *//根据条件删除数据* **if**(UserModel::*destroy*([**'status'**=>2])) {  
 **return '删除成功'**;  
 } **else** {  
 **return '删除失败'**;  
 }  
   
 $result = UserModel::*where*(**'id'**,**'>'**,1)->delete();  
 **if**($result) {  
 **return '删除成功！'**;  
 } **else** {  
 **return '删除失败！'**;  
 }  
 }

视图之模板赋值和模板渲染

*//模板的赋值  
 //assign方法  
 //赋值给模板变量* $name = **'小明'**;  
 $email = **'xiaoming@qq.com'**;  
 $this->assign([  
 **'name'** =>$name,  
 **'email'**=>$email  
 ]);  
 **return** $this->fetch(**'user/user'**);  
  
*//传入参数的方法* **return** $this->fetch(**'index'**,[  
 **"name"** => **"thinkphp"**,  
 **"email"** => **"thinkphp@qq.com"** ]);\*/  
  
*//助手函数view* **return** view(**''**,[  
 **'name'**=>**'thinkphp3'**,  
 **'email'**=>**'thinkphp@qq.com'** ]);

模板系统变量输出和函数使用

系统变量：

**代码放在view/index/index.html**

{$Think.server.server\_name} <**br** /> //输出域名  
{$Think.APP\_PA} <**br** /> //输出常量  
{$name|strtoupper|substr=0,3} <**br** /> //将字母转换为大写字母|截取0~3个字母  
{$time|date="y-m-d","###"}

**Index.php**

$name = **'hello world!'**;  
$time = *time*();  
$this->assign(**'name'**,$name);  
$this->assign(**'time'**,$time);  
 **return** $this->fetch();

模板之模板的布局

原样输出

{literal}

{$name}

{/literal}

模板之模板的继承和包含文件

**模板的继承**:

**Index.html:**

{extend name="base" /}

**在view/base.html:**

{block name="head"}  
<**div style="width**:100%;**height**:100**px**;**background-color**:**yellow**;**"**>这是基础模板的头部信息</**div**>  
{/block}<**br** />  
{block name="footer"}  
<**div style="width**:100%;**height**:100**px**;**background-color**:**blue**;**"**>这是基础模板的底部信息</**div**>  
{/block}

把所有的代码都放在了index.html:

{extend name="base" /}  
{block name="head"}{\_\_block\_\_}  
<**div style="width**:100%;**height**:100**px**;**background-color**:**yellow**;**"**>这是子模板的头部信息</**div**>  
{/block}<**br** />  
{block name="footer"}  
<**div style="width**:100%;**height**:100**px**;**background-color**:**blue**;**"**>这是子模板的底部信息</**div**>  
{/block}  
这是子模板区块外面的信息，是不会展示出来的

**包含文件：**

在index.php

**public function list**()  
{  
 **return** $this->fetch();  
}

在index/list.html:

{include file='../application/index/view/index/lang.html'}  
这个页面文件是用来包含文件

在index/lang.html:

<**div style="width**:100%;**height**:100**px**;**background-color**:**yellow**;**"**>这是包含文件的使用</**div**>

模板之内置标签

内置标签：

循环输出标签：

Volist标签：

在index.php:

**public function list**()  
{  
 $list = UserModel::*all*();  
 $this->assign(**'list'**,$list);  
 **return** $this->fetch();  
}

在list.html:

{volist name="list" id="vo" offset="0" length="3"}  
 {$vo.id}====>{$vo.name}{br /}  
{/volist}<**br** />  
{foreach $list as $value}  
 {$value.id}:{$value.name}<**br** />  
{/foreach}  
{for start="1" end="10"}  
 {$i}  
{/for}<**br** />

路由

**在index.php**

**public function list**()  
{  
 **return '这是我自己定义的路由规则'**;  
}  
**public function** read()  
{  
 **return '这是定义了简化的方法'**;  
}  
**public function** test($id)  
{  
 dump($id);  
}

**在route.php**

**use** think\Route;  
Route::*get*(**'/'**,**function**(){  
 **return 'Hello,world!'**;  
});  
  
*//Route::rule('new/:id','index/Index/list');*Route::*get*(**'new/:id'**,**'index/Index/read'**);  
Route::*post*(**'new/:id'**,**'index/Index/test'**);

资源路由：

Route::*resource*(**'blog'**,**'index/blog'**);

Blog.php:

**public function** index()  
{  
 dump(**'index'**);  
}  
**public function** update($id)  
{  
 dump(**'update $id'**);  
}  
**public function** delete($id)  
{  
 dump(**'delete $id'**);  
}

快捷路由：

Route::*controller*(**'detail'**,**'index/detail'**);

访问url 时：直接 [www.tp5.com/detail/info](http://www.tp5.com/detail/info) 发送是get方法，就会直接访问getinfo

**class** Detail  
{  
 **public function** getInfo()  
 {  
 dump(**"getInfo"**);  
 }  
 **public function** postInfo()  
 {  
 dump(**"postInfo"**);  
 }  
 **public function** putInfo()  
 {  
 dump(**"putInfo"**);  
 }  
 **public function** deleteInfo()  
 {  
 dump(**"deleteInfo"**);  
 }  
}

# 文件上传

单文件的上传：

**class** Upload **extends** \think\Controller  
{  
 *//文件上传表单* **public function** index()  
 {  
 **return** $this->fetch();  
 }  
 *//文件上传提交* **public function** up(Request $request)  
 {  
 *//获取表单上传文件* $file = $request->file(**'file'**);  
 *//上传文件验证* $result = $this->validate([**'file'**=>$file],[**'file'**=>**'require|image'**],  
 [**'file.require'**=>**'请选择上传文件'**,  
 **'file.image'**=>**'非法图像文件'**]);  
 **if**(**true** !==$result) {  
 $this->error($result);  
 }  
 *//移动到框架应用根目录/public/upload/目录下* $info = $file->validate([**'ext'**=>**'gif,jpg,png'**])->  
 *//move(ROOT\_PATH . 'public' . DS . 'uploads','test');//以自己定义的名字保存* move(***ROOT\_PATH*** . **'public'** . ***DS*** . **'uploads'**,**''**);*//以原文件的名字保存* **if**($info) {  
 $this->success(**'文件上传成功：'**.$info->getRealPath());  
 } **else** {  
 *//上传失败获取错误信息* $this->error($file->getError());  
 }  
 *//如果要获取同文件名而不希望被覆盖则用：* $file->move(***ROOT\_PATH*** . **'public'**.***DS*** .**'uploads'**,**true**,**false**);  
 $info = $file->rule(**'md5'**)->move(***ROOT\_PATH*** . **'public'** . ***DS*** . **'uploads'**);  
 **if**($info) {  
 **echo** $info->getSaveName();  
 } **else** {  
 $this->error($file->getError());  
 }  
 }  
}

多文件的上传：

缩略图:

**<?php****namespace** app\index\controller;  
**use** think\Db;  
**use** think\Request;  
**use** think\Image;  
  
**class** Upload **extends** \think\Controller  
{  
 **public function** index(){  
 **return** $this->fetch();  
 }  
 *// 图片上传处理* **public function** picture(Request $request)  
 {  
*// 获取表单上传文件* $file = $request->file(**'image'**);  
 **if** (**true** !== $this->validate([**'image'** => $file], [**'image'** => **'require|image'**]))  
 {  
 $this->error(**'请选择图像文件'**);  
 } **else** {  
*// 读取图片* $image = Image::*open*($file);  
*// 图片处理* $image->thumb(150, 150, Image::***THUMB\_CENTER***);  
*// 保存图片（以当前时间戳）* $saveName = $request->time() . **'.png'**;  
 $image->save(***ROOT\_PATH*** . **'public/uploads/'** . $saveName);  
 $this->success(**'图片处理完毕...'**, **'/uploads/'** . $saveName, 1);  
 }  
 }  
}

请求和响应：

获取请求变量 系统推荐使用 param 方法统一获取当前请求变量，该方法最大的优势是让你不需要区分当前请求类型而使用 不同的全局变量或者方法，并且可以满足大部分的参数需求，下面举一个例子。  
<?php

namespace app\index\controller;

use think\Request;

class Index

{

public function hello(Request $request)

{ echo '请求参数：';

//用param:

//dump($request->param());

//echo 'name:'.$request->param('name');

//用input:

dump(input());

echo 'name:'.input('name');

}

}

访问下面的URL地址： <http://tp5.com/index/index/hello.html?test=ddd&name=thinkphp>

# 模块

用模型查询多数据库的方法

**use** app\index\model\User;  
**use** think\Loader;

***get方法：***\*$res = User::*get*(1);  
 $res = $res->toArray();  
***Loader方法：***$user = Loader::*model*(**"User"**);  
$res = $user::*get*(4);  
***助手函数model：***$user = model(**"User"**);  
$res = $user::*get*(5);  
$res = $res->toArray();  
dump($res);

模型聚合操作

***根据条件语句查找***$res = User::*where*(**"id"**, **"<"**, 5)  
 -> max(**'num'**);  
 ***获取数据库里有几条数据：***$res = User::*count*();  
***获取某个属性元素的最大值***$res = User::*max*(**'num'**);  
***获取某个属性元素的最小值***$res = User::*min*(**'num'**);  
***获取某个属性的所有元素之和***$res = User::*sum*(**'num'**);  
***获取某个属性的所有元素平均数***$res = User::*avg*(**'num'**);  
dump($res);

模型的获取器

在控制器的代码：

$res = User::*get*(3);  
**echo** $res->**sex**;

可以根据sex里面的属性值到执行switch:

在模块里面的代码;

**public function** getSexAttr($val) {  
 **switch** ($val) {  
 **case '0'**:  
 **return "男"**;  
 **break**;  
 **case '1'**:  
 **return "女"**;  
 **break**;  
 **default**:  
 **return "未知"**;  
 **break**;  
 }  
}

模型的修改器+自动完成

在控制器里

***新增数据***$res = User::*create*([  
 **'username'** => **'think'**,  
 **'sex'** => 0,  
 **'email'** => **'think@qq.com'**,  
 **'password'** => **'think'**,  
 **'num'** => 1  
]);  
***修改数据***$userModel = User::*get*(11);  
$userModel->**sex** = 1;  
$userModel->save();  
dump($userModel);

在model模块里代码：

数据修改或新增时，自动都会执行  
**protected $auto** = [  
 **'time'**];  
在数据新增的时候会执行  
**protected $insert** = [  
 **'time\_insert'**];  
在数据被修改时执行  
**protected $update** = [  
 **'time\_update'**];  
**public function** getSexAttr($val) {  
 **switch** ($val) {  
 **case '0'**:  
 **return "男"**;  
 **break**;  
 **case '1'**:  
 **return "女"**;  
 **break**;  
 **default**:  
 **return "未知"**;  
 **break**;  
 }  
}  
  
可以使用set+字段名+Attr来完成数据修改，在增加时我们可以  
在这里对字段的数据修改  
**public function** setPasswordAttr($val) {  
 **return** *md5*($val);  
}  
**public function** setTimeAttr() {  
 **return** *time*();  
}  
**public function** setTimeInsertAttr() {  
 **return** *time*();  
}  
**public function** setTimeUpdateAttr() {  
 **return** *time*();  
}

时间戳和软删除

在控制器：

新增数据时自动增加时间戳

$res = Test::*create*([  
 **'name'** => **'think'**,  
 **'password'** => *md5*(**'think'**)  
]);

***软删除***,***后加true的话，就直接删除***$res = Test::*destroy*(3,**true**);  
dump($res);

可以***查询软删除的数据***$res = Test::*onlyTrashed*()->select();  
**foreach**($res **as** $val) {  
 dump($val->getData());  
}

在模块：

**use** traits\model\SoftDelete; *//引入软删除的文件***class** Test **extends** Model  
{  
 *//开启自动增加时间戳* **protected $autoWriteTimestamp** = **true**;  
}

模板布局

把前端每个页面相同的头部与底部分离出来，直接在HTML里面引用，引用的代码如下：

首先需要定义一个布局模板文件，放到 application/index/view/layout.html 内容如下：

{include file="user/header" /}

{\_\_CONTENT\_\_}

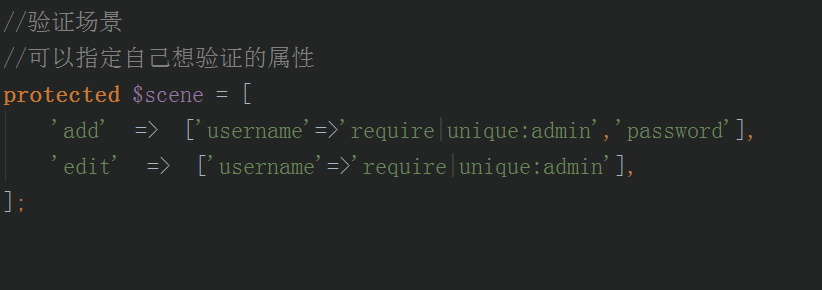
{include file="user/footer" /}

# 验证与验证场景

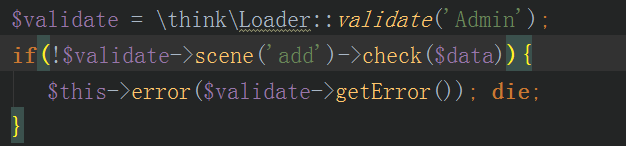
一般会在添加数据的时候用到：

在validate文件中





在控制器中代码：



# 分页

分页输出 可以很简单的输出用户的分页数据，控制器 index 方法修改为： // 获取用户数据列表

public function index() {

// 分页输出列表 每页显示3条数据

$list = UserModel::paginate(3);

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

return $this->fetch();

}

# 微信小程序登录授权代码

**<?php  
namespace** app\index\controller;  
  
**use** think\Controller;  
**use** think\Db;  
  
*header*(**'Access-Control-Allow-Origin: \*'**);  
  
**class** Login **extends** Controller  
{  
 **public function** index()  
 {  
 $code = $\_GET[**"code"**];  
*//配置appid* $appid = $\_GET[**"appid"**];  
*//配置appscret* $secret = $\_GET[**"appSecret"**];  
  
*//api接口* $api = **"https://api.weixin.qq.com/sns/jscode2session?appid=**{$appid}**&secret=**{$secret}**&js\_code=**{$code}**&grant\_type=authorization\_code"**;  
*//获取GET请求* **function** httpGet($url){  
 $curl = *curl\_init*();  
 *curl\_setopt*($curl, ***CURLOPT\_RETURNTRANSFER***, **true**);  
 *curl\_setopt*($curl, ***CURLOPT\_TIMEOUT***, 500);  
 *curl\_setopt*($curl, ***CURLOPT\_SSL\_VERIFYPEER***, **true**);  
 *curl\_setopt*($curl, ***CURLOPT\_SSL\_VERIFYHOST***, 2);  
 *curl\_setopt*($curl, ***CURLOPT\_URL***, $url);  
 $res = *curl\_exec*($curl);  
 *curl\_close*($curl);  
 **return** $res;  
 }  
*//发送* $str = httpGet($api);  
 **echo** $str;  
 $arr = *json\_decode*($str, **true**);  
*// echo $arr;* $openid = $arr[**'openid'**];  
 $re = Db::*table*(**'dy\_user'**)->where(**'openid'**,$openid)->select();  
 **if**($re){  
   
 } **else** {  
 $insert = Db::*table*(**'dy\_user'**)  
 ->insert([  
 **'openid'** => $openid,  
 ]);  
 }  
  
 }  
  
}