学习thinkphp5文档；

1。查询数据循环输出

控制器代码：

$data=DB::table('think\_user')->select();

//分配数据

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

视图代码：

<table border="1" width="800px" align="center">

<tr>

<td>ID</td>

<td>NAME</td>

<td>PHONE</td>

<td>ZS</td>

<td>PWD</td>

</tr>

{volist name="data" id="value"}

<tr>

<td>{$value.id}</td>

<td>{$value.name}</td>

<td>{$value.phone}</td>

<td>{$value.zs}</td>

<td>{$value.pwd}</td>

</tr>

{/volist}

</table>

2.调用控制器；

（1.调用前台其他控制器）

$model=new \app\index\controller\User;

echo $model->user();

echo "<hr/>";

$model=new user;

echo $model->user();

echo "<hr/>";

$model=controller('user');

echo $model->user();

1. 调用当前控制器

**echo** $this->index();  
**echo** "<hr/>";  
**echo self**::*diaoyongs*();  
**echo** "<br/>";  
**echo** Index::*index*();  
**echo** "<hr/>";  
**echo** action('diaoyongs');

1. 调用后台控制器

$model=**new** \app\admin\controller\Admin;  
**echo** $model->admin();  
**echo** "<hr/>";  
//调用后台模块的admin  
**echo** action('admin/Admin/admin');

1. 通过系统获取配置
   1. 获取系统配置，use 应付config(),可以直接使用

**<?php  
namespace** app\index\controller;  
//use app\index\controller\user;  
**use** think\config;  
**use** app\admin\controller\Admin;  
**class** Index  
{  
 **public function** index(){  
  
 **echo** config('name');  
 **echo** config('age');  
 **echo** config('kouhao');  
 //通过系统类读取配置  
 //如果配置存在 直接输出 不存在返回NUll  
 **echo** \think\Config::*get*('name');  
 **echo** \think\config::*get*('abc');  
 **echo** dump(config::*get*('kouhao'));  
 }  
  
}

* 1. 获取配置数组

dump(config('teacher')) ;  
dump(config('teacher.name'));  
  
dump(config::*get*('teacher'));

3.调整应用模式状态

'app\_status' => 'home',

创建一个home.php

就是家里的配置数据库

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: admin  
 \* Date: 2018/4/16  
 \* Time: 13:25  
 \*/***return** [  
 "database"=>[  
 'type' => 'mysql',  
 // 服务器地址  
 'hostname' => '127.0.0.1',  
 // 数据库名  
 'database' => 'test',  
 // 用户名  
 'username' => 'root',  
 // 密码  
 'password' => ''  
 ]  
  
  
  
];

还可以创建一个office，公司数据库配置，代码如上

1. 动态配置config

dump(config('name'));  
config('name','PHP开发');  
 dump(config('name'));  
  
 \think\config::*set*('name','WEB前端');  
 dump(config('name'));  
//头文件必须要引入**use** think\config;

config::*set*('name','小程序开发');  
 dump(config('name'));

1. 环境配置，在think根目录下新建.Env类

dump(\think\Env::*get*('name'));

1. 路由
   1. 绑定模块

define('BIND\_MODULE','index');

1. 设置最基本的路由，要引入use think/Route;在Route中设置；

Route::*rule*('test','index/Index/index');

（3）这是路由的类型

Route::*rule*('/','index/Index/index');  
Route::*rule*('test','index/Index/index');  
//带参路由  
//注册一个带参的路由  
//http://localhost/cms/public/index.php/course/1  
Route::*rule*('course/:id','index/Index/course');  
//注册带多个参数的路由，任何一个没有写都会失败  
Route::*rule*('time/:year/:month','index/Index/shijian');

1. 可选参数，在路由设置参数加中括号怕[ ],

访问地址：http://localhost/cms/public/index.php/time/2018

Route::*rule*('time/:year/[:month]','index/Index/shijian');

1. 动态路由，如果匹配到上面的路由，优先访问以上路由

访问地址：http://localhost/cms/public/index.php/a/b

Route::*rule*(':a/:b','index/Index/dongtai');

1. 完全匹配路由，可加$,完全匹配

访问地址：http://localhost/cms/public/index.php/test1

Route::*rule*('test1$','Index/index/test1');

1. 路由额外带参

访问地址http://localhost/cms/public/index.php/test2

Route::*rule*('test2','Index/index/test2?id=10$name=zhangsan');

1. 路由请求类型
   1. Tp5中的请求类型：
      1. 1.get/post/put/delete
   2. 设置路由请求方式
      1. 路由默认支持所有请求类型
      2. Get请求和POST请求

Route::*rule*('type','Index/index/type','get');  
Route::*get*('type','Index/index/type');  
//支持POST请求  
Route::*post*('type','Index/index/type','post');  
Route::*post*('type','Index/index/type')

同时支持get和post,在get和post中加竖线 |

Route::*rule*('type','Index/index/type','get|post');

支持所有运用any或者\*

Route::*any*('type','Index/index/type','get|post');  
Route::*rule*('type','Index/index/type','\*');

Put和delete请求

Route::*rule*('type','Index/index/type','put');  
Route::*put*('type','Index/index/type');  
  
  
Route::*rule*('type','Index/index/type','delete');  
Route::*delete*('type','Index/index/type');

模拟请求

<input type="hidden" name="\_method" value="put">

1. 批量注册
   1. 普通批量注册

Route::*rule*([  
 "test"=>"index/index/test1",  
 "course/:id"=>"index/index/course"  
],'','get');

* 1. Get批量注册

Route::*get*([  
 "test"=>"index/index/test1",  
]);

* 1. 配置文件批量注册，推荐

**return**[  
 "test"=>"index/index/test1",  
 "course/:id"=>"index/index/course"  
];

1. 变量规则
   1. 匹配数字，只能传递数字‘id’=>’\d+’;

Route::*rule*('course/:id','index/index/course','get',[],['id'=>'\d+']);

1. 变量参数（请求类型get,设置后缀，’ext’=>’html’,访问路径时一定要加html）

Route::*rule*('course/:id','index/index/course','get',['method'=>'get'],['id'=>'\

资源路由

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

快捷路由

注册快接路由：

Route::*controller*('blog','Index/blog');

方法内容：

**public function** geta(){  
 **echo** "我是geta";  
}

访问路径

<http://localhost/cms/public/index.php/blog/a>

生成url地址

dump(Url::*build*('index/index/index'));  
dump(Url::*build*('index/index/course/200'));  
dump(Url('index/index/course/200'));

渲染输出

需要use think\View;

$view=**new** View();  
 **return** $view->fetch();

controller渲染输出，需要继承controller

**return** $this->fetch();

直接return,输出

**return** view();

控制器初始化方法：

运行这个类中的任意一个方法时，都会加载这个初始化方法，必须先继承系统控制器，后台权限把控，公共代码集成在初始化方法

**public function** \_initialize(){  
 **echo** "我是初始化方法";  
}

前置方法，在运行某个方法时先运行前置方法

**namespace** app\index\controller;  
**use** think\Controller;  
  
**class** UserInfo **extends** Controller  
{  
  
  
 **protected** $beforeActionList=[  
 'one',  
 //不想让index使用这个two方法  
 'two'=>['except'=>"index"],  
 //只想让index使用three方法  
 'three'=>['only'=>'index'],  
 ];  
  
 **public function** one(){  
 **echo** "one<br/>";  
 }  
  
 **public function** two(){  
 **echo** "two<br/>";  
 }  
 **public function** three(){  
 **echo** "three<br/>";  
 }  
 **public function** index(){  
 **return** "你好这个是UserInfo下的index方法";  
 }  
  
}

跳转：

成功跳转：

$this->success('跳转成功',url('Index/index'));

登陆失败：

$this->error('登陆失败',url('Login/index'));

重定向：

参数一：跳转地址

参数二：携带参数

参数三：301，暂时性转移，302永久性转移

参数四：隐式传输数据

$this->redirect('index/index',['id'=>100,'name'=>'adc']);

空操作：

作用：主要解决一些人恶意输入，返回首页

**public function** \_empty(){  
 $this->redirect('index/index');  
}

空控制器

只要解决用户恶意输入控制器，

网站上线的时候，每一个控制都需要写一个空操作，

不论前台后台，都要写一个空控制器。

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: admin  
 \* Date: 2018/4/18  
 \* Time: 10:53  
 \*/***namespace** app\index\controller;  
**use** think\Controller;  
  
**class** Error **extends** Controller  
{  
 **public function** index(){  
 $this->redirect('index/index');  
 }  
 **public function** \_empty(){  
 $this->redirect('index/index');  
 }  
  
}

资源控制器

使用命令行：

切换到thinkphp的根目

加载页面：

**namespace** app\index\controller;  
**use** think\Controller;  
**use** think\View;  
**class** Goods **extends** Controller  
{  
 **public function** index(){  
 **return** "我是前台Goods控制器嗲的方法";  
 }  
 **public function** jiazai(){  
 **return** View();  
 }  
 **public function** jiazai1(){  
 $View=**new** View();  
 **return** $View->fetch();  
 }  
 **public function** jiazai2(){  
 **return** $this->fetch();  
 }  
  
}

在创建加载页面，要同时创建视图，

获取请求信息：

**public function** index1(){  
 ///$request=request();  
 //使用Request类  
 $request=Request::*instance*();  
  
 dump($request);  
}

第二种要 use think/Request;

第三种系统控制器获取请求，也要导入use think/Controller

**public function** index1( Request $request){  
 ///$request=request();  
 //使用Request类  
// $request=Request::instance();  
//  
// dump($request);  
 //使用系统控制类获取请求信息  
 dump($request);  
  
}

获取请求的URL信息

**public function** getUrl(Request $request){  
 **echo** 'domain: ' . $request->domain() . '<br/>';  
// 获取当前入口文件  
 **echo** 'file: ' . $request->baseFile() . '<br/>';  
// 获取当前URL地址 不含域名  
 **echo** 'url: ' . $request->url() . '<br/>';  
// 获取包含域名的完整URL地址  
 **echo** 'url with domain: ' . $request->url(**true**) . '<br/>';  
// 获取当前URL地址 不含QUERY\_STRING  
 **echo** 'url without query: ' . $request->baseUrl() . '<br/>';  
// 获取URL访问的ROOT地址  
 **echo** 'root:' . $request->root() . '<br/>';  
// 获取URL访问的ROOT地址  
 **echo** 'root with domain: ' . $request->root(**true**) . '<br/>';  
// 获取URL地址中的PATH\_INFO信息  
 **echo** 'pathinfo: ' . $request->pathinfo() . '<br/>';  
// 获取URL地址中的PATH\_INFO信息 不含后缀  
 **echo** 'pathinfo: ' . $request->path() . '<br/>';  
// 获取URL地址中的后缀信息  
 **echo** 'ext: ' . $request->ext() . '<br/>';  
}

获取请求模块/控制器/方法

1. 控制器
2. 模块
3. 方法

**public function** getinfo(Request $request){  
 //当前模块  
 dump($request->module());  
 //当前控制器  
 dump($request->controller());  
 //当前方法  
 dump($request->action());  
}

获取请求相关类型：

//请求类型

dump($request->method());

//请求资源类型  
dump($request->type());

//请求资源地址  
dump($request->ip());

//是否AJAX请求  
dump($request->isAjax());

获取所有地址栏参数

dump($request->param());

获取特定地址栏参数

dump($request->only('name'));

剔除部分地址栏参数

dump($request->except('id'));

判断get中的id参数是否存在：

**public function** getData(Request$request){  
 dump($request->has('id','get'));  
}

一下方法作用同上：

dump(input('?get.id'));

读取get中的id的参数

**public function** getData(Request$request){  
 dump($request->has('id','get'));  
 dump(input('?get.id'));  
 dump(input('get.id'));  
}

读取get中的所有

读取get中的id的参数

Dump(‘get.’);

**public function** getData(Request$request){  
   
 dump(input('get.'));

//方法不同功能一样

dump($request->get());

}

读取类型：

方法同上，把get改成post,delete,put.....即可

设置过滤，先获取Request值，然后在进行过滤

Request::instance()->filter(['strip\_tags','htmlspecialchars']),

密码进行过滤和加密，直接在后面.md5就可以了

$request->get('strip\_tags','htmlspecialchars,md5');

过滤，只想获取自己想要的数据

//只要name  
dump($request->only('name'));  
//剔除name  
dump($request->except('name'));

变量的修饰符

在后面加

| **修饰符** | **作用** |
| --- | --- |
| s | 强制转换为字符串类型 |
| d | 强制转换为整型类型 |
| b | 强制转换为布尔类型 |
| a | 强制转换为数组类型 |
| f | 强制转换为浮点类型 |

**public function** xiushi(Request $request){  
 dump(input('get.id/d'));//x强制转换整形  
 dump(input('get.name/s'));//强制转换成字符串  
 dump($request->get('id/d'));  
}

修改参数，

访问页面：http://localhost/cms/public/index.php/index/xiugai?id=10

**public function** xiugai(Request $request){  
 dump($request->get('id/d'));  
 dump($request->get(['id/d'=>20]));  
  
}

请求类型判断：分别是get.Post.ajax.phone

**public function** type1(Request $request){  
 //是否get请求  
 dump($request::*instance*()->isGet());  
 //是否post请求  
 dump($request::*instance*()->post());  
 //是否是AJAx请求  
 dump($request::*instance*()->isAjax());  
 //是否是手机访问  
 dump($request::*instance*()->isMobile());  
}

模拟PUT请求

<form action="{:url('type1')}" method="post">  
 <p>  
 User<input type="text" name="name" >  
 <input type="hidden" name="\_method" value="PUT">  
 </p>  
 <p>  
 pass<input type="password" name="pwd" >  
 </p>  
 <p>  
 <input type="submit" value="提交" >  
 </p>  
  
  
</form>

在模拟PUT请求时一定要PSOT方法，还要设置隐藏域

获取请求类型：是get/post/put/delete

dump($request->method());

模拟ajax的放文件路径http://localhost/cms/public/index.php/index/type1?\_ajax=1

重点是要带参数：**?\_ajax=1**

**伪静态**

**作用**

Url伪静态通常是为了SEO的优化（主要）

为了保证网站的优化

使用

在config.php中修改：

'url\_html\_suffix' => 'html',

获取当前伪静态的后缀

dump($request->ext());

绑定参数

**public function** bangd( $id,$name){  
// dump(input('id'));  
// dump(input('name'));  
 dump($id);  
 dump($name);  
}

就是band($id,$name);

绑定的名字要和地址栏的地址一对一

绑定的参数可以设置默值

**public function** bangd( $id="123",$name="admin"){  
// dump(input('id'));  
// dump(input('name'));  
 dump($id);  
 dump($name);  
}

连接数据库，配置数据库文件：

Database.php

// 数据库类型  
'type' => 'mysql',  
// 服务器地址  
'hostname' => '127.0.0.1',  
// 数据库名  
'database' => 'test',  
// 用户名  
'username' => 'root',  
// 密码  
'password' => '',  
// 端口  
'hostport' => '',

查询数据

先导入use think\Db;

$DB=**new** Db();  
$data=$DB::*table*("think\_user")->select();  
dump($data);

第二种查询方式，发送查询语句

**public function** data(){  
 $DB=**new** Db();  
// $data=$DB::table("think\_user")->select();  
  
 $data=$DB::*query*("select \* from think\_user");  
 dump($data);  
}

数组查询：

**public function** data1(){  
 **echo** "我是使用方法配置连接数据库";  
 $Db=Db::*connect*([  
 'type'=>'mysql',  
 'hostname'=>'127.0.0.1',  
 'database'=>'test',  
 'username'=>'root',  
 'password'=>'',  
 'hostport'=>'3306',  
 ]);  
 $data=$Db->table('think\_user')->select();  
 dump($data);  
}

字符串连接，我的这个例子没有密码，请注意

//字符集配置连接：数据库类型://用户名:密码@数据库地址:数据库端口/数据库名#字符集

$Db=Db::*connect*('mysql://root:@127.0.0.1:3306/test#utf8');  
  
  
$data=$Db->table('think\_user')->select();  
dump($data);

用model连接数据库：

**namespace** app\index\model;  
**use** think\Model;  
**class** User **extends** Model  
{  
 //使用数组连接数据库  
 **protected** $connection=[  
 // 数据库类型  
 'type' => 'mysql',  
 // 服务器地址  
 'hostname' => '127.0.0.1',  
 // 数据库名  
 'database' => 'test',  
 // 用户名  
 'username' => 'root',  
 // 密码  
 'password' => '',  
 // 端口  
 'hostport' => '3306',  
 'prefix' => 'think\_',  
 ];  
  
}

创建一个user.php,user就是表名，请注意，如果有前缀，就在数据库连接配置里加’prefix’=>’前缀：think\_’

然后引用：

**public function** data2(){  
 **echo** "使用模型连接数据库";  
 $user=**new** \app\index\model\User();  
 dump($user::*all*());  
}

数据库基本操作：

1. 增
   1. 下面三种增加数据方法，注意第二种的后面数据加中括号[]，还有第三种方式：后面的数据名称可以随便写，后面赋值要对应就好也要加中括号：

$data=Db::*execute*("insert into think\_user value(null,'xiangzong',18523555617,'你好我来自拉萨','12345645' )");  
dump($data);  
$data=Db::*execute*("insert into think\_user value(null,?,?,?,? )",['xianghao',18523555619,'wolaizihainan','123654']);  
dump($data);  
$data=Db::*execute*("insert into think\_user value(null,:name,:phone,:zs,:pwd )",['name'=>'xiangzong','phone'=>'18536523652','zs'=>'shahsa','pwd'=>'qq12345678']);  
dump($data);

1. 删

**public function** delete(){  
 $data=Db::*execute*("delete from think\_user where id =9");  
 dump($data);  
 $data=Db::*execute*("delete from think\_user where id =?",[4]);  
 dump($data);  
 $data=Db::*execute*("delete from think\_user where id >:id",['id'=>5]);  
 dump($data);  
}

1. 改

**public function** update(){  
 $data=Db::*execute*("update think\_user set name ='liling' where id=1");  
 dump($data);  
 $data=Db::*execute*("update think\_user set name =? where id=?",['wangxiao',1]);  
 dump($data);  
 $data=Db::*execute*("update think\_user set name =:name where id=:id",['name'=>'xijingping','id'=>1]);  
 dump($data);  
}

1. 查

$data=Db::*query*("select\* from think\_user");  
dump($data);  
$data=Db::*query*("select \* from think\_user where id>=? and id<=?",[2,10]);  
dump($data);

1. 获取
   1. 获取最后执行的sql语句

**echo** Db::*getLastSql*();

查询操作：

Table方法查询数据

**public function** index(){  
 //查询所有数据  
 $data=Db::*table*("think\_user")->select();  
 dump($data);  
 //查询一条数据  
 $data=Db::*table*("think\_user")->find();  
 dump($data);  
 //获取最后执行的sql;  
 **echo** Db::*getLastSql*();  
}

Name方法查询数据，

Name方法查询数据时，如果配置了数据，方法会自动添加表前缀，但是table不会自动添加表前缀

//查询所有数据,  
$data=Db::*name*("think\_user")->select();  
dump($data);  
//查询一条数据  
$data=Db::*name*("think\_user")->find();  
dump($data);  
//获取最后执行的sql;  
**echo** Db::*getLastSql*();

助手函数查询

//助手函数查询  
$data=db("think\_user")->select();  
dump($data);  
$data=db("think\_user")->find();  
dump($data);

如何进行条件匹配

//SELECT \* FROM `think\_user` WHERE `id` > 3  
$data=Db::*table*("think\_user")->where("id",">",3)->select();  
**echo** Db::*getLastSql*();  
//SELECT \* FROM `think\_user` WHERE ( `id` > 3 AND `id` < 19 )  
$data=Db::*table*("think\_user")->where("id",">",3)->where("id","<",19)->select();  
**echo** Db::*getLastSql*();  
 //模糊查询匹配name包含aa的数据  
//SELECT \* FROM `think\_user` WHERE `name` LIKE '%aa%'  
$data=Db::*table*("think\_user")->where("name","like","%aa%")->select();  
**echo** Db::*getLastSql*();  
dump($data);  
  
//匹配密码是1，用户名是amdind的用户  
//SELECT \* FROM `think\_user` WHERE `name` LIKE '%aa%'  
$data=Db::*table*("think\_user")->where("name","admin")->where("pwd","1")->select();  
**echo** Db::*getLastSql*();  
dump($data);

Where or查询：

//SELECT \* FROM `think\_user` WHERE `id` <= 10 OR `id` >= 20  
$data=Db::*table*("think\_user")->where("id","<=",10)->whereOr("id",">=",20)->select();  
 //SELECT \* FROM `think\_user` WHERE `name` LIKE '%a%' OR `name` LIKE '%b%'  
$data=Db::*table*("think\_user")->where("name","like","%a%")->whereOr("name","like","%b%")->select();  
  
  
dump($data);  
**echo** Db::*getLastsql*();

截取数据：

就是从第二个到第六个数据截取出来，如果只写3，就从第一个开始，截取三个。

$data=Db::*table*("think\_user")->limit(2,6)->select();  
  
dump($data);  
**echo** Db::*getLastsql*();

实现排序，

//根据id进行排序desc是倒叙  
$data=Db::*table*("think\_user")->order("id","desc")->select();  
dump($data);

只查询：name和id列

$data=Db::*table*("think\_user")->field('id,name')->select();  
  
dump($data);  
  
**echo** Db::*getLastsql*();

分页查询，page,

//用于分页查询，page(第几页，一页多少条数据)  
$data=Db::*table*("think\_user")->page(2,10)->select();

多表查询：

$data=Db::*query*("select shangping.\*,fl.name tname from fl,shangping where fl.id=shangping.fenlei");  
  
// SELECT `shangping`.\*,fl.name tname FROM `shangping` INNER JOIN `fl` ON `fl`.`id`=`shangping`.`fenlei`  
 $data=Db::*table*("shangping")->field("shangping.\*,fl.name tname")->join("fl","fl.id=shangping.fenlei")->select();  
 //右连接，以右表为主  
 $data=Db::*table*("shangping")->field("shangping.\*,fl.name tname")->join("fl","fl.id=shangping.fenlei","right")->select();  
 //右连接，以左表为主  
 $data=Db::*table*("shangping")->field("shangping.\*,fl.name tname")->join("fl","fl.id=shangping.fenlei","left")->select();  
  
 dump($data);  
  
 **echo** Db::*getLastsql*();

设置别名

//设置别名 ->alias()  
 $data=Db::*table*("shangping")->alias("s")->field("s.\*,f.name tname")->join("fl f","f.id=s.fenlei")->select();  
 dump($data);  
  
 **echo** Db::*getLastsql*();

Union 集合

Bind 绑定参数

不要使用原生的sql尽量用THINKPHP的，有一定的防护措施。为了防止sql注入

Db::table('think\_user')->where('id',':id')->where('name',':name')->bind(['id'=>[10,\PDO::PARAM\_INT],'name'=>'thinkphp'])

->select();

统计数据（聚合函数）

**echo** "最大值";  
$data=Db::*table*("think\_user")->max("id");  
dump($data);  
**echo** "最小值";  
$data=Db::*table*("think\_user")->min("id");  
dump($data);  
**echo** "平均值";  
$data=Db::*table*("think\_user")->avg("id");  
dump($data);  
**echo** "统计数量";  
$data=Db::*table*("think\_user")->count("id");  
dump($data);  
**echo** "获取总分";  
$data=Db::*table*("think\_user")->sum("id");  
  
dump($data);  
  
**echo** Db::*getLastsql*();

数据插入（一条插入，多条插入，插入一条并获取最后插入的id）

//数字中的字段名 必须要和数据库中的字段名一致  
  
 //出入一条数据  
// $data=[  
// 'name'=>"zhangsan",  
// 'phone'=>'18523555617',  
// 'zs'=>'no',  
// 'pwd'=>'123456'  
// ];  
  
// $code=Db::table("think\_user")->insert($data);  
//多条数据插入  
// $data=[  
// [  
// 'name'=>"zhangsan",  
// 'phone'=>'18523555617',  
// 'zs'=>'no',  
// 'pwd'=>'123456'  
// ],  
// [  
// 'name'=>"lisi",  
// 'phone'=>'1888888888',  
// 'zs'=>'no',  
// 'pwd'=>'1233333333456'  
// ]  
// ];  
// $code=Db::table("think\_user")->insertAll($data);  
//  
// dump($code);  
  
 //插入并输出最后插入的id  
 $data=[  
 'name'=>"mr.li",  
 'phone'=>'18523555617',  
 'zs'=>'no',  
 'pwd'=>'123456'  
 ];  
 **echo** Db::*table*("think\_user")->insertGetId($data);

助手函数插入

和上面的区别在于：一个是Db::table(“表名”)和Db(“表名”)

$data=[  
 'name'=>"zhangsan",  
 'phone'=>'18523555617',  
 'zs'=>'no',  
 'pwd'=>'123456'  
 ];  
  
 $code=Db("think\_user")->insert($data);  
//多条数据插入  
 $data=[  
 [  
 'name'=>"zhangsan",  
 'phone'=>'18523555617',  
 'zs'=>'no',  
 'pwd'=>'123456'  
 ],  
 [  
 'name'=>"lisi",  
 'phone'=>'1888888888',  
 'zs'=>'no',  
 'pwd'=>'1233333333456'  
 ]  
 ];  
 $code=Db("think\_user")->insertAll($data);  
  
 dump($code);  
  
// 插入并输出最后插入的id  
 $data=[  
 'name'=>"mr.li",  
 'phone'=>'18523555617',  
 'zs'=>'no',  
 'pwd'=>'123456'  
 ];  
 **echo** Db("think\_user")->insertGetId($data);

更新数据（修改数据，自增，自减）

// $code=Db::table("think\_user")->where("id",'1')->update(["name"=>"zuizuibang"]);  
 //自增：  
// $code=Db::table("think\_user")->where("id",1)->setInc("pwd");  
 //自减  
// $code=Db::table("think\_user")->where("id",1)->setDec("pwd");  
 //自减10个数  
 $code=Db::*table*("think\_user")->where("id",1)->setDec("pwd",10);  
 dump($code);

助手函数：同上：

//助手函数  
 // $code=("think\_user")->where("id",'1')->update(["name"=>"zuizuibang"]);  
 //自增：  
// $code=("think\_user")->where("id",1)->setInc("pwd");  
 //自减  
// $code=("think\_user")->where("id",1)->setDec("pwd");  
 //自减10个数  
 $code=("think\_user")->where("id",1)->setDec("pwd",10);  
 dump($code);

删除数据

**public function** delete(){  
// $code=Db::table("think\_user")->where("id","1")->delete();  
  
 //删除多条数据  
// $code=Db::table("think\_user")->where("id in(1072,1073)")->delete();  
 $code=Db::*table*("think\_user")->where("id in(1072,1073)")->delete();  
 //删除id为1074，1074  
 $code=Db::*table*("think\_user")->delete([1074,1074]);  
  
  
 //删除区间数据  
 $code=Db::*table*("think\_user")->where("id>1075 and id <1077")->delete();  
  
   
 **echo** Db::*getLastSql*();  
 dump($code);  
}

助手函数只需要去掉{：：table}