# Php

1. Php基础：
2. 连字符.
3. 单引号解释 逻辑计算 变量
4. 双引号解释 字符串
5. %余数只能拿到整数 11.23%10 结果为1小数部分会舍去
6. +=的意思为 $a=$a+1 => $a+=1 -+相同意思
7. Intval(小数) 小数取整
8. var\_export($b) var\_export函数打印布尔值true or false
9. 三目： 条件==条件 ?成了返回 : 不成立返回
10. $stdin=fopen("php://stdin","r"); 打开命令行输入

$input=fgets($stdin); //获取命令行输入

10.trim() //去掉不可见字符

11. 循环

do{

循环体

}while(条件)

While(条件){

循环体

}

for(;循环条件;){

循环体

break //打断循环

continue //跳过当次循环

}

12.判断

if (条件) {

条件成立执行

}else if (条件) {

条件成立执行

}else {

条件不成立执行

}

**switch** (比较值1) { //用于分支较多的判断

**case**较的值2:

条件成立执行 ;

**break**;

**case**较的值3:

条件成立执行 ;

**break**;

**default**:

以上条件都不成立执行;

**break**;

}

12. 数组/对象

$a=array("sad","dsad","dsad","aa"=>"aa","ok"); 定义数组

array\_push($a, "123sd "); 添加元素；

$a[0]='苹果'； 通过索引添加元素

unset($i[1]); 删除一个项

print\_r($a) 打印数组

foreach ($a as $item=>$value){ //便利数组第一种方法

echo “$item:$value ” //键名加键值

}

for ( $a=0;$a<sizeof($i);$a++){ //第二种方法 sizeof($i) || count($i)数组长度

echo $i[$a];

}

$object=**'{"asd":"asd","asd":"dsadas"}'**; //对象的定义  
  
*var\_export*($object); //输出对象

13.函数的基本特性：

function name(){

$GLOBALS['a']=1; //相当于直接调用外部变量，允许函数内部变量与外部变量同名

global $a //相当于调用变量的地址，不能出现内部外部变量同名

}

function Test()   
{   
static $count = 0; //函数中的静态成员在反复调用的时候值不会重置  
$count++;   
echo $count;   
if ($count < 10) {   
Test();   
}   
$count--;   
}

14.随机数：

*md5*(*uniqid*(*rand*(),**true**)) // 使用rand随机函数生成一个字符作为唯一码的表头，再把得到的字符md5加密，more entrope属性为true，返回的唯一码位数更多；

15.字符的处理：

1. 使用符号分割字符返回一个数组：explode（’a’,$b）// a参数为使用什么分割，b参数为要分割的字符
2. 字符长度：*mb\_strlen*($c,**"utf8"**); 多字节字符长度 中文
3. Strlen() //单字节字符长度；
4. trim() //去掉不可见字符
5. sprint() //不懂
6. *strtolower() //将大写转换为小写*
7. *strtoupper() // 将小写转换为大写*
8. *explode*(**"."**,$c) //按符号分割字符
9. *join/implode*($c) //将数组字符，组合成一个字符,2个效果一样

*---------------------正则-------------------------*

匹配函数： preg\_match\_all(a,b,c) b参：需要进行匹配的字符 a参:规则

php正则的规则/ values/ 必须以/开始/结束

c参：匹配的结果，是一个数组 //如果不用all，只会拿到第一个匹配结果 \_all拿的到信息为第0个为有的匹配信息，下面是分组信息

替换函数：*preg\_replace*($v,**"a"**,$c);

参数依次是：匹配规则，替换成什么，匹配的字符

*preg\_replace*($v,**"<$1>"**,$c)

$1表示第一分组，该效果会在第一分组2侧加上符号

16.常量的定义: const NAME= ’values’ //命名规范:大写 老版本define(“NAME”,values’ )

例：**class** B{  
 **static** *$i*;  
  
 **const *NAME***=**'世界你好！'**;·   
  
 **function** f(){  
 **echo self**::*$i*;  
  
 **echo static**::***NAME***; //使用static而不使用self的原因是，用static修饰常量时，常量的父级在被调用的时候确定，

而self修饰常量，常量的父级在系统读取的时候就确定了  
 }  
}  
  
**class** C **extends** B{  
  
 **const *NAME*** = **'世界你好我被继承了'**;  
  
}  
  
$i=**new** B();  
  
$i->f(); //此时的常量使用static调用，他的父级是B类

16. 接口和继承：

1.**extends //继承父子关系，只能有一个继承**

**abstract //抽象修饰符**

**例：**

**abstract class** kk{ //拥有抽象方法的类必须是抽象类  
 **public abstract function** k();//抽象方法不能有实例： 空白实例也不行如：{}  
}

2. interface implements //**接口允许多继承**

例：

interface by{ //使用interface定义接口  
 public function f(); //内置方法不允许实例  
 }  
  
 class A implements by { 使用implements关键字实现接口  
 function f() { //实现接口方法  
 *// TODO: Implement f() method.* }  
 }

17.js语法：允许在实例类外部定调用并且赋值m本生没有的字段：

**class** B{  
  
}  
  
$a=**new** B();  
$a->**i**=**'dsa'**; //在外部定义变量

$a->f=’daa’; //支持数组便利对于$a;支持unset($a->i)删除  
  
**class** C{  
 **function** f (){  
 **global** $a;  
 **echo** $a->**i**;  
 }  
  
  
}  
  
$i=**new** C();  
  
$i->f();

18.导入外部php文件include\_once(路径);

*spl\_autoload\_register*(); //自动导入外部类，但是外部类与文件名一致，并且首字母大写，和java一致

19.引用类型：

$a=array(123,123,123);

$b=$a //在这里是拷贝复制，而不是引用，所以改变$a，$b不会变,引用值类型

$b=&$a //使用&符号，引用地址类型$b就等于$b

20．数据类型

标量类型：boolean , integer,float,string, float //浮点类型小数

复合类型：array(数组) object(对象)

特殊类型：resource 资源类型*get\_resource\_type*() 返回资源类型；

Null类型

伪类型：mixed混合类型：可以接受多种类型，但不是全部类型；

Number数字类型：接受integer和float类型；

Callback 回调类型：

21．输出函数总结：

echo //只能输出字符

print\_r() //只能输出数组

var\_dump() //可以输出任何类型 并且会显示类型的数据

*var\_export*  //可以输出任何类型

22.作用域：

局部/全局/超全局

超全局：$\_SERVER 他是一个数组

[“REMOTE\_ADDR”] 客户端ip

[“PATH\_INFO”] http路径php后面/的路径

[“QUERY\_STRING”] ?后面的值，也就是用户穿的的参数

[“REQUEST\_URL”] 完整路径

其他超全局变量：

$\_REQUEST;  
$\_FILES;  
$\_POST;  
$\_SESSION;  
$\_GET;  
$\_COOKIE;

23.运算符:

a=b || c 当第一个条件为假时返回c //也可以用| 区别是第一个正确后c将不会读取，而|会读取第二个 。下同

a=b && c 当第一个条件为真时返回true

!!强制转换为boolean

<<3 乘以2的3次方

>>3 除以2的3次方

Heredoc <<< name //name为定界符，可以为任意字符

内容不可以任意多个引号也可以。默认定界符有双引号，

‘name’ 修改为单引号后不识别变量

Name

23.函数：

**function** f($a,$b){  
 *print\_r*(*func\_get\_args*()); //获取所有传入的参数  
 }  
  
f(123,123,123,123123);

1.函数的动态调用：

例1：

**function** f($a,$b){  
 *print\_r*(**"asdasda"**);  
}  
  
 $i=**'f'**; //只要与函数名同名，并且以字符串的形式储存，就可以传递函数  
$i(123,123);

例2：$a=”md5”;

$a(“asdad”); //传递了mad5方法

2.参数类型的限定：

例：**function** f(Closure $name){ //规定传入的参数只能是变量名，不能是值   
}

3.闭包函数：

**function** f(){  
 $a=0;  
  
 **return function** ()**use** ($a){ //使用use()规定调用的变量；一定返回一个函数  
 **echo** $a;  
 };  
}

4.反射：检测维护自身的一种机制

可以查看类上有哪些方法，字段等等

**class** B{  
 **public $a**=**"ok"**;  
 **function** f(){  
 **echo "f"**;  
 }  
}  
$r= **new** ReflectionClass(**'B'**); //new 一个反射对象  
*print\_r*($r->getProperties()); //查看所有字段

*print\_r*($r->getMethods()); //查看函数名

$a=$r->newInstance(); *//使用$r实例化B;*$bi=$r->getProperty(**"a"**); *//获取字段名***echo** $bi->getValue($a); *//使用字段名去获取值*

24.魔术方法： //用于对未定义的属性的访问

**class** A{  
 **function** \_\_get($name)  
 {  
 **echo "你在访问未知的字段：**$name**"**;  
 }  
  
 **function** \_\_call($name, $arguments)  
 {  
 **echo "你在访问未知的方法：**$name**,他的属性是"**;  
 *print\_r*($arguments);  
 }

}  
  
 $a=**new** A();  
$a->**pp**;  
$a->xx();

25.数组传参：

**function** f($i=0,$j=0){  
 *print\_r*(*func\_get\_args*());  
}  
  
$a=**"f"**;  
  
  
*call\_user\_func\_array*($a,**array**(213,123)); 使用数组传参//传入2个

参数

*call\_user\_func*($a,**array**(213,123))； 传入一个数组类型参数，与正常调用无差异

26.数组式访问：

27.函数判断：

**if** (!*function\_exists*(**'f'**)){ //判断一个函数是否存在  
 **function** (){  
   
 }  
}

28.常用函数：

1. 文件配置

配置文件后缀：.ini

例：[user] 分组一  
 ur=2  
 [mysql] 分组二  
 id=1

配置文件导入：*parse\_ini\_file*(**"",true**) //数组类型,true分组类型

——————————文件操作——————————————

1. *copy*(**'a'**,**'b'**); a:规定要复制的文件 b:输出的路径带名字

例：*copy*(**'aa.ini'**,**'js/aa.ini'**);

1. *unlink*(**"aa.php"**); 删除文价，参数文件名，成功返回true,失false
2. file\_exists(“path“); 判断文件或目录是否存在，成功返回true,失false
3. is\_dir(“path”); 判断文件名是否是一个目录, 成功返回true,失false;
4. is\_file(“name”); 判断文件是否是一个文件成功返回true,失false
5. *file\_put\_contents*(**"aa.txt"**,**"世界你好"**) 文件写入
6. *file\_get\_contents*(**"aa.txt"**) 读取文件
7. *filesize*(**"aa.txt"**); 获取文件大小
8. *basename*(**"C:**\U**sers**\m**ac**\P**hpstormProjects**\u**ntitled2**\a**a.txt"**); 返回文件名称和后缀
9. *dirname*(**"C:**\U**sers**\m**ac**\P**hpstormProjects**\u**ntitled2**\a**a.txt"**); 返回去掉文件后的路径

*---------------文件流-----------------*

1. $f= *fopen*(**"aa.txt"**,**"b"**); //打开一个文件

b参数为权限：

r：只读方式

r+:读写方式

w：写入方式 //w和r的区别是如果文件不存在，w将建立一个

w+：读写方式

a :写入模式 //a的指针在最后，w和r在最前面

a+：读写模式

1. Fclose() 关闭一个打开文件
2. Fflush() 将缓冲内容输入到文件； //不懂
3. Fgets() 读取文件；
4. Fgetss() 与fgets()不同的是他会尝试去点文件种的任何html和php标记
5. *fwrite*($f,**"values"**); 给文件写入内容；第一参：一个写入流，第二个参写入内容；
6. fputs() 写入2进制文件
7. feof()。
8. fseek() 把文件指针从当前位置移动到新的位置，新位置从文件开头以数字计算
9. ftruncare();
10. json\_encode($i) 数组转对象格式字符
11. json\_decode($i,true) 对象格式字符转对象，加true转数组
12. U

---------------时间------------------------------------------

# Php\_webserver

1. form表单传值

<form method="get" action="text.php" name="aa"> // 设置请求方式get  
 <input name="name">  
 <button type="submit">确定</button>   
</form>

<?php  
 $a=$\_GET["name"]; //按请请求方式获取数据  
 echo $a;  
?>

获取请求数据的方式：

$\_GET[**"name"**] 支持gei请求

$c=$\_GET[**'k'**] 支持post请求

$\_REQUEST[**'name'**] 支持所有请求

$\_FILES[**'file'**] 支持文件上传

1. 路径传值：

例：<**a href="text.php?name=ssdasdsa&cc=2g"**>11</**a**>

路径后面加?传值 格式为 名字=值

如果需要传2个则使用&符号连接

1. 隐藏域传值：

<**input type="hidden" value="我是看不见的" name="k"**>

Input的类型为：hidden

1. 共享session

*session\_start*(); //打开session共享  
$\_SESSION[**'a'**]=22; //名为a的session赋值

*session\_destroy*(); // 清空所有的session；

1. Php中的跳转：

延迟跳转：*header*(**'refresh:2,url=tt.php'**);2秒后跳转；

1. 数据库操作：

$pdo=newPDO('mysql:dbname=usermgr;charset=UTF8','root','root'); //建立连接

//使用dataname属性定义访问的数据库名称  
$mmm= $pdo->prepare('set character set utf8');  
$mmm->execute(); //设置编码  
$mmm= $pdo->prepare('select *\** from user where id= :name ||id=lg'); //sql语句  
$mmm->execute(**array**(**'name'**=>**$a**,**'lg'**=>**$b**)); //执行数据库操作//使用:代替$，execute中使用数组传递sql参数  
$data=$mmm-> fetch(PDO::*FETCH\_NAMED*); //获取返回结果  
例：将返回结果放入视图模块  
**<?php foreach** ($data **as** $datum=>$value) { **?> //** <**tr**>  
 <**td**>  
 **<?php echo** $data[$datum][**"id"**] **?>** </**td**>  
 <**td**>  
 **<?php echo** $data[$datum][**"name"**] **?>** </**td**>  
 <**td**>  
 **<?php echo** $data[$datum][**"role\_id"**] **?>** </**td**>  
 </**tr**>  
**<?php** } **?>**

1. 文件上传（图片）：

例：

$files=$\_FILES[**'file'**]; *//按inp名称获取文件，*$files是一个数组，内含文件信息$filesPash=$files[**'tmp\_name'**]; *//获取临时文件路径*$name= *md5*(*uniqid*(*rand*(),**true**)); *//生成随机名字*$hz=$files[**'name'**]; *//原文件名称及后缀*$hz=，*explode*(**'.'**,$hz); *//分离名字和后缀*$hz=$hz[1]; *//得到后最  
move\_uploaded\_file*($filesPash,**'img/'**.$name.**'.'**.$hz);  
 *//写入文件 fliename 参数为临时文件路径，destination参数为写入路径包括写入后的文件名字及后缀*

# Php\_面向对象

1. 类的应用：

例： //在php里.符号用于连接，所以调用方法使用->符号

**class** A{  
  
 **public $a**=5;  
  
 **function** f(){  
  
 **echo** $this->**a**; //$this代表对象本身体；  
 }  
}  
  
$a=**new** A();  
$a->f();

1. 访问权限:

Public :表示全局类内部，外部，子类等都可以访问；

Private: 表示私有的，只能内部访问；

Protected: 表示受保护的只有本类，父类，子类可以访问

1. 构造方法：

**public function** \_\_construct() //使用关键字\_\_construct区别构造方法，构造方法特性：在实例化对象时该方法会自动调用

**class** A{  
 **public function** \_\_construct()  
 {  
 **echo "我是A"**;  
 }  
  
  
}  
  
**class** B **extends** A{  
 **public function** \_\_construct() //重写自己的构造方法  
 {  
 **parent**::*\_\_construct*(); // 使用关键字parent调用父级构造方法；  
 **echo '我是B'**;  
 }  
  
  
}  
  
**new** B();

3.

**function** \_\_destruct() //程序执行完毕，释放内存空间时，制动调用的方法；  
{  
 **echo '销毁'**;  
}

1. 静态成员:static

静态成员的调用方式： 使用::代替.号

**class** B{  
 **static** *$i*;  
 **function** f(){  
 **echo** B::*$i*;  
 }  
}  
**echo** B::*$i*;

在静态方法中调用静态字段：

**class** B{  
 **static** *$i*;  
  
 **function** f(){  
 **echo self**::*$i*; //使用self代替$this  
 }  
}  
  
**echo** B::*$i*;

# THINKPHP

1. 框架配置
2. 配置模块：Conf -----------------------------数据库的配置
3. 控制器: Lib\Action // Lib\Model 建立自己的验证模块
4. 视图 : Tpl
5. 命名规范：控制器方法名必须与模板名相同
6. 公共文件：Common //配置：LOAD\_EXT\_FILE='扩展文件名' 不带后缀任何地方可以访问该文件下的函数
7. 常用api：
8. this->show(**"世界你好！"**); 直接在控制器输出视图
9. this->assign('name',$value) 对模板变量赋值，可以直接在页面调用的变量。
10. $this->display();输出模板文件 默认模板文件与方法同名
11. M()实例化系统内置的model类
12. D()实例化数据库表与m不同D必须自定义建模型类，定义模型类可以定制自定义验证规则：

**在模型类中定义校验规则：**

**protected $\_validate =**array(

**array(验证字段1,验证规则,错误提示),**

**array(验证字段2,验证规则,错误提示)**

**rray('name', 'unique', '商品类型名称已经存在！',3,'unique'), 唯一验证  
*//array(验证字段，验证规则，错误提示，验证条件，验证时间，附加规则)***

);

常见验证规则：

|  |  |
| --- | --- |
| **regex** | **正则验证，定义的验证规则是一个正则表达式（默认）** |
| require、 | 字段必须 |
| email | 邮箱、 |
| url URL | 地址 |
| currency | 货币、 |
| number | 数字 |
| **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 |
| **notequal** | 验证是否不等于某个值，该值由前面的验证规则定义（**3.1.2版本新增**） |
| **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 |
| **notin** | 验证是否不在某个范围内，定义的验证规则可以是一个数组或者逗号分割的字符串（**3.1.2版本新增**） |
| **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)** |
| **notbetween** | 验证不在某个范围，定义的验证规则表示范围，可以使用字符串或者数组（**3.1.2版本新增**） |
| **expire** | **验证是否在有效期，定义的验证规则表示时间范围，可以到时间，例如可以使用 2012-1-15,2013-1-15 表示当前提交有效期在2012-1-15到2013-1-15之间，也可以使用时间戳定义** |
| **ip\_allow** | 验证IP是否允许，定义的验证规则表示允许的IP地址列表，用逗号分隔，例如201.12.2.5,201.12.2.6 |
| **ip\_deny** | 验证IP是否禁止，定义的验证规则表示禁止的ip地址列表，用逗号分隔，例如201.12.2.5,201.12.2.6 |
| **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。** |

## 批量验证

系统支持数据的批量验证功能，只需要在模型类里面设置patchValidate属性为true（ 默认为false），

设置批处理验证后，getError() 方法返回的错误信息是一个数组，返回格式是：

array("字段名1"=>"错误提示1","字段名2"=>"错误提示2"... )

1. $User = D("User"); // 实例化User对象
2. if (!$User->create()){
3. // 如果创建失败 表示验证没有通过 输出错误提示信息
4. $this->ajaxReturn($User->getError());
5. }else{
6. // 验证通过 可以进行其他数据操作
7. }

Model类中的自动填充：

例：

**protected $\_auto** = **array**(  
 **array**(**'create\_time'**,**'time'**,1,**'function'**),  
);

规则如下：

array(填充字段,填充内容,[填充条件,附加规则])

|  |  |  |
| --- | --- | --- |
| 填充字段 | 必须 | 就是需要进行处理的表单字段，这个字段不一定是数据库字段，也可以是表单的一些辅助字段，例如确认密码和验证码等等。 |
| 填充规则 | 必须 | 配合附加规则完成 |
| 填充时间 | 可选 | 包括： Model:: MODEL\_INSERT或者1 新增数据的时候处理（默认） Model:: MODEL\_UPDATE或者2更新数据的时候处理 Model:: MODEL\_BOTH或者3所有情况都进行处理 |
| 附加规则 | 可选 | 包括： 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 ignore ：为空则忽略，3.1.2版本开始支持，如果该字段的值恒等于空字符串则忽略该字段的值 string ：字符串（默认方式） |

Model类中的关联模型：  **类继承于RelationModel**

**protected $\_link** = **array**(  
 **array**(  
 **'mapping\_type'**=>***BELONGS\_TO***, //一对多关系  
 **'class\_name'**=>**'role'**, //数据库表名  
 **'mapping\_name'**=>**'roles'**) //映射到数组中的字段  
 );

$form->relation(**true**)->select(); // relation(**true**)手动执行关联，代替关联，

数据库的字段映射：给数据库表从新取名字

Class UserModel extends Model{

     protected $\_map = array(

         'name' =>'username', // 把表单中name映射到数据表的username字段

         'mail'  =>'email', // 把表单中的mail映射到数据表的email字段

     );

}

1. 数据库操作：
2. $Form->create() 获取整个表单数据
3. 获取数据所有：select()

获取单一数据：$Form->where('id=3')->getField('title');

1. 添加数据：add()
2. 删除数据：delete() $Form->delete(5);主键的作为参数，$User->where('status=0')->delete();  其他键值作为参数
3. 修改数据：save()

其他方法一： 对象赋值 //不依靠获取表单

$Form = M("Form");

// 要修改的数据对象属性赋值

$data['id'] = 5;

$data['title'] = 'ThinkPHP';

$data['content'] = 'ThinkPHP3.1版本发布';

$Form->save($data); // 根据条件保存修改的数据

其他方法二：对象赋值 //不依靠获取表单

$Form = M("Form");

// 要修改的数据对象属性赋值

$data['title'] = 'ThinkPHP';

$data['content'] = 'ThinkPHP3.1版本发布';

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

其他方法三：对象赋值 //不依靠获取表单

$Form = M("Form");

// 要修改的数据对象属性赋值

$Form->title = 'ThinkPHP';

$Form->content = 'ThinkPHP3.1版本发布';

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

其他方法四：修改单一属性 //不依靠获取表单

$Form = M("Form");

// 更改title值

$Form->where('id=5')->setField('title','ThinkPHP');

其他方法：直接进行数据的增加和减少

$User = M("User"); // 实例化User对象

    $User->where('id=5')->setInc('score',3); // 用户的积分加3

    $User->where('id=5')->setInc('score'); // 用户的积分加1

    $User->where('id=5')->setDec('score',5); // 用户的积分减5

    $User->where('id=5')->setDec('score'); // 用户的积分减1

1. 复杂逻辑的快速查询：3.0以上版本：
2. **getBy动态查询**  
   该查询方式针对数据表的字段进行查询。例如，User对象拥有id,name,email,address 等属性，那么我们就可以使用下面的查询方法来直接根据某个属性来查询符合条件的记录。

例：$user = $User->getByName('liu21st');

1. **getFieldBy动态查询**  
   针对某个字段查询并返回某个字段的值，例如

$userId = $User->getFieldByName('liu21st','id');

表示根据用户的name获取用户的id值。

-------------------------------------其他杂项--------------------------------------------

1. **echo** MODULE\_NAME 当前模块名的获取
2. ACTION\_NAME 当前模方法名
3. CONTROLLER\_NAME 模板名
4. getLastSql() 获取最后一次执行的sql，返回sql语句
5. 路径传值 参数绑定 方法可以直接接受参数，但是参数名必须和路径传参名相同
6. $this->buildHtml() 生成包含静态网页，用于放置常用又不变化的数据库数据
7. U方法：用于连接：模板中使用：**{:U('Form/add?id=2')}**
8. Success()的定制在ThinkPHP/Tpl/dispatch\_jump.tpl里面定制自己的样式（成功消息的提示）
9. Error()错误消息提示
10. Redirect() 页面跳转
11. ajaxReturn(data,format) ajax回传数据
12. **function** \_empty($a) 类似魔术方法，访问没有的方法时，调用该方法，$a方法名
13. 配置URL\_CASE\_INSENSITIVE=true 忽略访问方法的大小写
14. 路由例：

'URL\_ROUTER\_ON' => true, *//开启路由*'URL\_ROUTE\_RULES' => array( *//定义路由规则*'url' => array('Form/add'), //相当于给路径映射一个名字，通过其他路径访问，防止内部信息暴露

路径传值：

例：'url:id\d' => array('Form/add'),

'url:name' => array('Form/add'),

访问时必须有id或者name，同时支持2参数绑定（\d用于区别数字）

路由支持重定向：

'url:name' =>’http:www.baidu.com’, 支持句对路径重定向

--------------------------------前端杂项----------------------------------------

1. {$a| default="默认值"} 给变量设置默认值
2. *date*(**'Y-m-d h:i:s'**,*time*());获取当前时间
3. {$a|str\_replace='s',###} 函数的使用
4. {:date('y-m-d h:i:s')} :代表有返回值，~是没有返回值的函数

Tp的页头和页尾的导入：<!DOCTYPE **html**>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
 导航  
 <hr/>  
 <block name="body"> //将需要变化的内容写成块  
 变化的内容  
 </block>  
 <hr/>  
 脚注  
</body>  
</html>

<**extend name="../public/my.html"**/>模块中导入模板，但是模块中不再有html等标签

<**block name="body"**> //名字必须相同  
 内容 //重写内容  
</**block**>

1. Tp3.1的配置：*define*(**'APP\_DEBUG'**,**TRUE**); *// 开启调试模式 /必须开启***require '../ThinkPHP/ThinkPHP.php'**;路劲
2. 数据库:

**放置sql注入:使用%s或者%d 代替变量**

例**$a->query("select *\** from user where id=%s or id=%d",$i,$d);  
*var\_export*($this->a);**

1. **失误事物：**

$User->startTrans();//  在User模型中启动事务

 // 进行相关的业务逻辑操作

$Info = M("Info"); // 实例化Info对象

$Info->save($User); // 保存用户信息

 if (操作成功){

     $User->commit();    // 提交事务，此处修改数据库

 }else{

   $User->rollback();  // 事务回滚，失误回滚，事物开始后的数据库修改全部取消

 }

# Tp3.2架构

1. 控制器：Controller
2. 模型：Model
3. 视图：view
4. Tp3.2实现多模块管理

Common是共有模块 用于配置共有数据库，方法之类的模块

Home用于客户模块

Admin 管理模块 ---属于手动建立的模块

由于模块化的管理出现了3级目录：

入口文件的如下配置：*define*(**'BIND\_MODULE'**,**'Home'**);

可以跳过Home目录直接访问下面的模型方法

1. 验证码：

**public function** Verify(){  
 $Verify = **new** \Think\Verify();  
 $Verify->entry();  
}

<**img src="{:u('Home/Index/Verify',array())}" onclick="this**.**src**=**'{:u(\'Home/Index/Verify\',array())}'"**>

1. 数据库配置：

// 添加数据库配置信息

'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

1. 内置分页功能：

$menu\_m=M(**'menu'**);

$menu\_all= $menu\_m->count(); //查询总条数

$page=**new** Page( $menu\_all, 5); //传入总页数，每页多少条

$menu\_data= $menu\_m->limit($page->**firstRow** . **','** . $page->**listRows**)->select(); //查询语句

$this->assign ( **"page"**, $page->show ( **'Admin'** ) ); //将生成的翻页数据映射到视图 在视图调用变量即可

$this->assign ( **"data"**, $menu\_data );   
  
$this->display();

对于条件查询的翻页状态保持：

Publick function \_initialize(){

$vs=array(‘name’,’id’,) //要保持的字段

Foreach(I(‘post.’) as $k=>$v){ 第一次查询时提前在设置下次提交的字段

If(is\_array($k,$vs)){

$\_GRT[$k]=$v;

}

}

Parent::\_initialize();由于重写了构造方法说以要重行执行父类的构造方法

}

由于是构造函数所以每次访问都会执行这个方法，所以状态会一直保持下去

1. 连贯操作：

Where[‘id’]=array(‘in’,array(1,2,3)) 单字段多条件的，用于修改多条数据

1. 在视图调用系统标量：

{$Think.server.script\_name} // 输出$\_SERVER['SCRIPT\_NAME']变量

{$Think.session.user\_id} // 输出$\_SESSION['user\_id']变量

{$Think.get.pageNumber} // 输出$\_GET['pageNumber']变量

{$Think.cookie.name} // 输出$\_COOKIE['name']变量