# PHP基础部分

变量概念和命名规范：

变量名，变量值，存储地址

变量详解：相当于内存中有一张变量表，存放变量名，这个变量名指向内存的另外一个存放变量值的空间。

所以声明变量时：首先，变量表多出一个$a,然后开辟一块内存空间，值是 58 ， $a 指向58所在的内存地址

输出变量就是先从变量表找到这个变量，通过这个变量再找到他的值

改变变量的值，仅仅是将原来的值换成新的值

注意：变量名不能以数字开头，这一点在很多命名规范中都是相通的！！！

echo 多个变量可以用逗号隔开，一次输出多个变量和值

变量严格区分大小写

使用echo输出布尔值是分别对应 1 ， 0

动态变量：即是变量名也是一个变量

也就是$$str;

$str = 'hello';

$hello = 'welcome to maoming!';

echo $$str; //welcome to maoming;

变量类型判断：就是判断变量$var是不是这个类型

is\_int($var)

is\_bool($var)

is\_null($var)

is\_float($var)

is\_string($var)

isset($var) :检测变量是否存在

还可以直接通过gettype($var)获取变量类型

变量的销毁：unset($var);

变量的赋值方式：引用赋值 传值赋值

传值赋值：

$a = 3;

$b = 5;

$a = $b;

以上语句表示：把$b的值读出来 然后赋给 $a,

在内存中，$a $b 都有各自的值空间，值都为5

引用传递，也就是传地址

$a = 3;

$b = 5;

$a = &$b;这里表示有两个变量$a $b共同指向$b的值

unset($b)并不影响$a,对象就是默认这样按照地址传递的！！！

把$b的地址读出来，赋给$a,此时$a重新指向5所在的空间，一改全改

0.3-0.2==0.1为什么返回false？

因为计算机在存储浮点数的时候存在精度损失，不能很精确，所以并不是简单的数学运算！！！

## 常量概念及定义方法

常量：就是不变的值，而且是一旦定义不能改变，也不能重复定义

习惯上常量全部大写

变量是可以随意赋值的，如果想定义一个不希望被改变的值，比如圆周率，就应该用常量，常量是全局不变的，全局有效，所谓全局就是在函数内外都是有效的，在哪里都可以访问的到！！！

存在即是合理，你想得到的，都是已经有解决方法的！！

define('常量名','常量值')；

echo 常量名;

注意：常量没有$符号

PHP中变量的作用域和js中变量的作用域不一样

PHP中函数体就是一个作用域，他是不会访问到外面的变量的，如果试图访问一个外部的变量，会提示undefined

js中，会一直往上找，如果实在找不到就会报错 not define

注意：数组，对象，资源句柄不可以赋给常量

动态常量：使用constant($var)函数，把变量值当做常量名，读出常量的值

这一点与动态变量有点不一样

常量的应用：网站根目录；配置文件中一些特别重要的配置项，比如DEBUG

算术运算符：

PHP是动态语言，变量值和类型在运行过程中是可以改变的。有时还会自动转型

一个数对2取模，结果为1，奇数，结果为0 ， 偶数

注意：负数取模，结果的符号只取决于被除数的符号

=== 变量的类型和值都相等才返回true

== 只需要变量值相等就返回true

类型转换：布尔值和数的比较，数转成布尔，只有0 0.0为false

字符串与数比较，只是取字符串开头的数字部分

布尔转字符串，true '1' false '0'

字符串转布尔 '' '0'转为false，其他为true

赋值运算符的运算结果就是 = 右边的值

凡是运算必有返回值，比较运算返回布尔值

$a = $b = $c = 10;

运算优先级：在有逻辑运算和比较运算时，先运算哪个呢？可以用()把先运行的部分括起来，也就是把比较运算括起来

注意坑：

$a = 3;

$b = 5;

if(($a = 8) || ($b = 9)) {

$a = $a + 1;

$b = $b + 1;

}

echo $a,'~',$b,'<br />';

逻辑运算的短路特性.

一旦从逻辑上能分析出结果后,后面的表达式将不再运行

异或运算 xor：两个条件必须 一真一假 才为真

三元运算符：比较常用的用法是：把运算结果返回给一个变量 $res = $name != ''? $name : '游客';

详解递增递减运算符：

$a++;

这是一个自身递增运算，有运算就必有结果，其结果为$a本身，然后再把$a的值+1

$a = 3;

$b = $a++;这句相当于：$b = $a ; $a = $a + 1;

此时$b = 3, $a = 4;

同理：a--

但是++$a;表示先把变量 +1,再把变量返回

也就是 $a= $a + 1; return $a;

这两个递增运算符的共同点是，最终结果都是自身的值 +1,只是过程有点不一样，加号在前就先加，再进行返回，加号在后就是先返回再自加！！！

主要区别就是 自加 和 返回 的顺序不一样

switch分支结构：

switch(){

case 值1:

xxx;

break;

case 值2:

xxx;

break;

case 值3:

xxx;

break;

case 值4:

xxx;

break;

default:

xxx;

}

switch case ,只能判断某变量 是否 与指定的值相等.

比如 值与 1,2,3..7比较,

但是,不能用来判断 范围.

而if /else if 可用来判断范围,甚至更复杂的判断表达式也可以.

而 case,却只能提供一个准备的值,和变量做 ==的比较

break 与 抓罪犯,

// break有破坏的意思,直接破坏整个循环语句,不再往下执行了.

for($i = 1;$i<10;$i++) {

if($i == 6) {

break;

} else {

echo $i,'<br />';

}

}

// continue是继续的意思,是说你继续下一次循环单元吧,这次的可以跳过了.

for($i = 1;$i<10;$i++) {

if($i == 5) {

continue;

}

echo $i,'<br />';

}

位运算详解：

在计算机中，数以二进制形式保存，有原码，反码，补码

计算机中的最小存储单位是字节，一个字节=8位

任何一个数都是以字节形式保存的

如果是一个字节的数，8位二进制，可以表示十进制范围的数-128-127，为什么呢?

答：最高位是符号位，正数为0，负数为1，他们不参与运算

01111111 正数 127

10000000 负数 -128 0-128=-128

特别说明：负数在计算机中以补码形式存放，转为十进制的方法是：补码中除了符号位的 绝对值 - 128 = 负数的十进制值

11111111 负数 127-128=-1

00000000 0

反过来，知道一个负数，求它的补码 负数 + 128 然后转为二进制，填充入1的后面

十进制转为二进制：与2相除直到商为0，得出系列余数，反过来写余数即可

如果无符号类型：一个字节，可以表示0-255 2^8-1=255

正数的原码，反码，补码都是自身

负数的原码，反码，补码，需要转换

反码：负数的符号位不变，各位取反

补码:反码+1，逢二进一

按位与 & 按位或 | 按位异或 ^

既然是两个数的位运算，先转成二进制，然后按照相应的逻辑运算规则进行运算，1代表true，0代表false

移位运算

<<:左移，相当于乘2

>>：右移，相当于除2

整体把二进制数移多少位，左移，右边补零，整体扩大；右移，左边补零，整体缩小

可以类比十进制的小数点左移，右移

进制详解

首先，所有的进制都是有可能的

十进制转成n进制：十进制数 除以 n 得出余数，从后往前写余数，即可

八进制转二进制：由于2^3=8,所以可以用三位二进制表示一个八进制的数

相对应的，十六进制转二进制，可以用四位二进制的数表示

反过来，由低到高，取三位算值，当做八进制的一个位数值；

取四位算值，作为十六进制的一个位数值

函数部分：

函数名不区分大小写

在调用的过程中,

// 实参,形参,从左到右,严格的逐一对应.[对号入座]

// 一一对应上,然后实参值,挨个儿赋给形参.

如果参数有默认值的话,要把该参数,写到最后面

如果实参多于形参

实参对号给形参赋值时, 后面多出来的实参,将会被舍弃掉. 没有其他副作用.

如果实参少于形参,没对应上

warning警告,并尽量往下执行.

函数可以设置默认值.

有默认值参数一般在没有默认值的形参的后面.

如果某个有默认值的形参,对应位置传了实参

那么,实参的值 将要覆盖 默认值.

函数与变量的作用域

PHP中,

函数运行时, 有其自己,独立的变量空间.

函数体内的变量,与外界的变量,无关.

即使是变量的名字相同.

函数的执行权与交互

超全局变量

PHP 中的许多预定义变量都是“超全局的”，这意味着它们在一个脚本的全部作用域中都可用。

1.$GLOBALS $GLOBALS — 引用全局作用域中可用的全部变量 $GLOBALS [ "foo" ]

2.$\_SERVER $\_SERVER 是一个包含了诸如头信息(header)、路径(path)、以及脚本位置(script locations)等等信息的数组。

3.$\_GET 通过 URL 参数传递给当前脚本的变量的数组

4.$\_POST 通过 HTTP POST 方法传递给当前脚本的变量的数组。

5.$\_FILES

6.$\_COOKIE

7.$\_SESSION 当前脚本可用 SESSION 变量的数组。

8.$\_REQUEST 默认情况下包含了 $\_GET ， $\_POST 和 $\_COOKIE 的 数组 。

9.$\_ENV

字符串：

1:单双引号定义字符串有何区别?

转义上的区别:

单引号,系统不做复杂的转义,

只转义\',\\, 其他的---一律原样输出.

双引号,则转义的比较多,\",\\,\r,\n,\t,\$等.

对于变量解释的区别:

单引号,不对字符串的$作变量名解析,

而双引号,会试着$理解成变量名,来解析.

2:单双引号定义字符串,哪个速度更快一些?

答:单引号解析的更快,因为不需要进行过多的转义和变量解析

问:什么时间选用单/双引号呢?

答:大段文本,比如新闻标题,文本内容,自我介绍等这样大段话,用',解析快.

但有时候,确实需要在一段字符串夹杂变量,比如拼凑sql语句时，用双引合适.

当需要写一大段文本,好多行,里面又有单引号,又有双引号,导致转义的时候,都比较麻烦.

// 还可以用heredoc, nowdoc两种界定字符串的方式

heredoc的写法

<<<标识符

中间写大段的文本

标识符;

$str = <<<cont

我来写一行,

再来第2行, ' " \,

dsafd abc \t \r\n haha

$age

cont;

heredoc的标识符,一定是独占一行,而且前面没有任何其他字符.

heredoc对文本的处理,和 双引号一样,即\r\n\t,变量,等,能解析出来.

/\*\*\*

nowdoc写法和heredoc一样,唯独在标识符两边用单引号包起来,

这样,大段文本的解析,就和单引号一样的效果.

不转义 \n \r \t,变量等等

\*\*\*/

$str = <<<'cont'

我来写一行,

再来第2行, ' " \,

dsafd abc \t \r\n haha

$age

cont;

用 get\_defined\_constants() 可以获得所有已定义的常量列表。

PHP 支持一个错误控制运算符：@。当将其放置在一个 PHP 表达式之前，该表达式可能产生的任何错误信息都被忽略掉。

+ 运算符把右边的数组元素附加到左边的数组后面，两个数组中都有的键名，则只用左边数组中的，右边的被忽略。

$a = array( "a" => "apple" , "b" => "banana" );

$b = array( "a" => "pear" , "b" => "strawberry" , "c" => "cherry" );

$c = $a + $b ; // Union of $a and $b

echo "Union of \$a and \$b: \n" ;

var\_dump ( $c );

array(3) {

["a"]=>

string(5) "apple"

["b"]=>

string(6) "banana"

["c"]=>

string(6) "cherry"

}

strlen 是计算字符串"字节"长度

mb\_strlen,是根据编码,计算字符串的"字符"个数.

mb\_strlen($str,'utf-8')

offset 的意思是 ---偏移量.

// 就是说,把光标先偏移一段距离,再查找,别从头查找.

echo strpos($str,'fine',17),'<br />';

strpos — 查找字符串首次出现的位置

思路:

先从头查找子串,找到后,则偏移过去子串,继续查找. 直到查不到.

\*/

function subnum($str,$sub) {

$sublen = strlen($sub); // 计算出子串的长度

$strlen = strlen($str); // 计算出父串的长度

if($sublen > $strlen) {

return 0;

} // 如果子串比父串长,没必要找了

for($offset = 0,$num=0;($offset = strpos($str,$sub,$offset))!==false;) {

$num += 1;

$offset += $sublen;

}

return $num;

}

substr\_count — 计算字串出现的次数

数组是什么---就是键值对的集合

分析: 声明数组时,不写键,行不行?

答:行!

这种情况下, 键会从0开始,逐一递增,来自动分配.

$stars[] = '刘德华';

$stars[] = '张学友';

$stars[] = '郭富城';

$stars[] = '黎明'; // 不断的开辟一个新单元

如果指定的键,则以指定为准,

如果没指定,则从0开始,逐一递增.(不够准确的)

如果给出方括号但没有指定键名，则取当前最大整数索引值，新的键名将是该值 + 1。// 上例从5+1开始

如果当前还没有整数索引，则键名将为 0。// 在$yan中得到体现

如果指定的键名已经有值了，该值将被覆盖

$arr = array('中','华','民','国'); // 0 1 2 3

print\_r($arr);

unset($arr[3]);

print\_r($arr);

$arr[] = '族';

print\_r($arr); // 键是:0124,删掉了一个单元,键的增长不受该影响,继续往上增长.

PHP语言非常灵活,

在C语言中,js中,java中,数组的索引只能是数字

而在PHP中,key/index可以是字符串,也可以是数字.

如果索引是纯数字,这种数组称为"索引数组",

否则,称为关联数组.

// 优化这道题,细节能体现一个人的水平

for($i=0,$len = count($arr);$i<$len;$i++) {

echo $arr[$i],'<br />';

}

foreach($arr as $k=>$v) { // 这是说:循环数组的每个单元,把单元的键赋给$k,值赋给$v

echo $k,'~',$v,'<br />';

}

在数组内部,有一个指针,指针指着某一个单元.

每循环一个单元,指针往尾部移动.

// current,key,

// next,prev,reset end

current "当前"的意思,该函数是 取出数组内部 指针指向的单元 的值

key,该函数是 取出数组内部 指针指向的单元的 "键"

next "下一个", 把数组内部的指针往尾部移动一个单元

end ,该函数把指针指向数组的最后一个单元

// prev,该函数把指针往头部移动一个单元

不定数组的指针跑哪儿了,用reset,可以重置,使数组的指针归0,指向开头处

// 但结合游标操作之后,就可以处理

for(;current($arr);next($arr)) {

echo key($arr),'~',current($arr),'<br />';

}

// 这里只是为了练习游标操作,用for来处理 ,在实际开发中,用foreach

走2步,退1步

reset($arr); // 想想这句话为什么要用?

for($flag = true;current($arr);) {

echo current($arr),'<br />';

if($flag) {

next($arr);

next($arr);

$flag = false;

} else {

prev($arr);

$flag = true;

}

}

foreach在运行前,会自动reset数组,因此不会受上一次的影响.

问:foreach循环后,会不会帮我们把数组reset指针初始化?

答;不会.

each的作用是把指针指向的当前单元的信息组成一个数组

数组有4个单元,具体如下:

array(

1=>'单元值',

'value'=>'单元值',

0=>'单元的键',

'key'=>'单元的键'

)

each不仅把当前指针指向的单元的信息组合成一个数组返回

// 并且顺手把指针往尾部移动一次，如果 取不出来单元,返回false

$arr = array(-3,-2,-0,1,2,3);

while($ceil = each($arr)) { // 如果能取出单元,则返回1,value,0,key4单元的数组,不为假

// 如果指针已经超出界限,返回false,自然while结束

echo $ceil[0],'~',$ceil[1],'<br />';

}

each经常和list一起用

list是重合赋值语句,可以一次性为多个变量赋值

把数组的哪个单元 对应 赋给哪个变量???

具体对应如下:

在list中,按","隔开的顺序,从左到右,分别对应右侧数组(设为$arr)

$arr[0],$arr[1],$arr[2],.... $arr[N]

如果需要对应右侧的$arr[x] 不存在怎么办?

答:依然是先对应,不存在,当成NULL来处理.

$a <===> $arr[0]

$b <===> $arr[1]

$c <===> $arr[2]

映射工作完成!!!!!

开始赋值

赋值的过程,和映射的顺序是反的,是从右往左赋的值

$c = $arr[2];

$b = $arr[1];

$a = $arr[0];

用each + list 自己写一个模拟foreach的函数

function for\_each($arr) {

while(list($k,$v) = each($arr)) {

echo $k,$v,'<br />';

}

}

数组的下标应该怎么写????

1: 对于数字的索引值,不必加单双引号

因为加了单双引号,比如"1",则需要把"1"转成整型1,来寻找该单元.

影响了效率

为什么引用$arr[dsafdsafdsaf] 这个单元?

答:因为name没有加单双引号,理解成常量,并且按常量值来做下标.

如果没有该常量,再次把name当成字符串的下标来处理.

不加单双引号,流程上,先理解成常量,如无,再当成字符串,效率必然低.

在字符串学习的时候,单引号的解析速度远比双引号快.

因此,优先加 单引号.

总结:

数字类型的索引,不必加引号

字符串类型索引必加,而且优先加单引号

count 函数

作用:计算数组的单元个数[或对象的属性个数]

注:count函数的参数,可以是数组,也可以是"非数组"

返回值分3种情况讨论

1: 数组/对象, 返回数组的单元个数/对象的属性个数

2: NULL ,返回0

3: 非数组,也非NULL,通通返回1

count($arr,true)

第2个参数为true时,代表递归的来计算数组的单元个数

array\_key\_exists

数组 键 存在, 用来判断数组有没有某一个键

用isset和array\_key\_exists来判断

有什么区别呢?

1:

isset 不是函数,是语法结构

而array\_key\_exists是函数

速度上,isset省去了一次函数调用,速度要快些.

2:isset来判断是根据值来判断的,

如果某个单元的值 为 NULL,依然是返回false

因为 isset是靠值来判断的.

如果对于键/值的都不固定,不确定是否有NULL存在,

请用 array\_key\_exists来判断

if(array\_key\_exists('abc',$arr)) {

echo '有abc键';

} else {

echo '$arr["abc"]不存在<br />';

}

3: isset是用来判断变量是否设置的,不限于数组

如果数组是动态生成,不好判断单元值及键值时,用array\_key\_exists保险

array\_flip

交换键值, 即值做键,键做值

$arr = array(0=>'a',1=>'b',2=>'c');

print\_r(array\_flip($arr));

// Array ( [a] => 0 [b] => 1 [c] => 2 )

print\_r($arr);

// 原数组没变,只是做为参数传给函数而已.

能否,有一个函数,判断某个值存不存在呢?

// 答:有 in\_array

if(in\_array('小刚',$arr)) {

echo '有小刚<br />';

} else {

echo '没有小刚<br />';

}

PHP操作mysql数据库部分

$conn = mysql\_connect('localhost','root','111111');

mysql默认占用的3306,因为,我们在mysql\_connect中并没有指定端口,

而PHP会用3306来进行连接

如果,你的mysql服务器端口偏是3307,如何指定

答:mysql\_connect('localhost:3307','root','111111');

问: 资源型变量是什么东西?

答: 资源就是一个管道

以PHP连接mysql为例

PHP<------->mysql服务器

PHP和mysql之间打通了连接,有一个通道

PHP要操作mysql,得沿着通道走.

只要是合法的sql语句,都可以通过通道发送给mysql服务器查询

具体用: mysql\_query()函数

地址栏来一个 id=83 or 1

根据$sql拼接的结果

$sql = 'delete from stu where id=83 or 1'

1恒为真,因为所有的行,都满足条件,都被删了.

这是一个非常简单的sql注入的例子.

怎么避免?

答:大原则,永远不要信任从客户处传来的数据

$sname = addslashes($\_POST['sname']);

$yy = addslashes($\_POST['yy']);

$id = $\_POST['id'] + 0;

之所以能注入,是因为利用表单值,故意传一些sql的语句的关键字,如单引号,#等.

我们转义,则把单引号转成\',这样,\'就和abcd等普通字符一样,构不成对sql语句的威胁

mysql\_fetch\_array 取出的结果 既有数字索引,也有字段索引

是 前2者结果的 并集

Array

(

[0] => 46

[id] => 46

[1] => rxyzqu

[sname] => rxyzqu

[2] => rxyzqu

[yy] => rxyzqu

)

mysql\_fetch\_object 返回的是对象

stdClass Object

(

[id] => 33

[sname] => stu3333

[yy] => yy333333

)

查询所有结果并放在一个数组中，可以马上关闭资源

$sql = 'select \* from stu';

$rs = mysql\_query($sql,$conn);

$list = array();

while($row = mysql\_fetch\_assoc($rs)) {

$list[] = $row;

}

$stu = mysql\_fetch\_assoc($rs); // 为什么不用while取?

if(empty($stu)) {

echo '没有这个用户';

exit;

}

判断用户又没有把id传过来，+0 表示转为整型

$id = isset($\_GET['id']) ? $\_GET['id'] + 0 :0;

严重注意:

对于增/删/改 来说,返回值 都是true/false2种情况

其中:insert,返回true,数据肯定是插入成功了.

对于update/delete ,即修改/删除来说

返回为真,代表:这个sql的语法没有问题,执行成功了.

是从语法角度/执行角度来说的.

但是,未必这个语句能影响到数据,比如你的条件不对

所以,要想真正判断有没有删/改数据,得以实际影响的行数为准

那么,在执行delete/update后,如果获取受影响行数?

答:用函数 mysql\_affected\_rows

在执行delete/update后立即调用此函数,

可得到delete/update语句影响的行数.

mysql\_num\_rows , 返回 select 语句的结果行数

在运行select 之后,立即调用mysql\_num\_rows 可以得到这次select的行数

$num = mysql\_num\_rows($rs);

当mysql\_query执行失败时,必有原因

可以用mysql\_errno()打印出错误号

和mysql\_error() 打印出错误信息

怎么来理解 include/require

就相当于, 把conn.php的代码,复制粘贴,代替 include那一行.

注意: include/require 包含进来的文件必须要加<?php ?>

因为在包含时,首先理解文件内容是普通字符串,

碰到<?php ?> 标签时,才去解释

在win下,路径可以用正斜线,也可以反斜线

而在linux下,则只能用正斜线 /

因此: 用正斜线 /

include 是包含的意思

在找不到包含文件时,会报warning, 然后试着往下运行.

require 是必须的意思

在找不到包含文件时,会报fatal error, 然后不再往下运行.

加once后,则在包含时,系统会进行判断

如果已经包含,则不会再次包含.

加\_once虽然系统帮你考虑了只加载一次,但系统的判断导致效率降低.

因此,更应该在开发之初,就把目录结构整理好.

尽量不要用\_once的情况.

下面这个用法,是利用 include /require返回被包含页面的值

被包含页面A中 return value

包含页面B中 $v = include('A.php');

则 $v 被赋值为 value

回到$\_POST上来,要循环对POST做转义

$\_POST = array('age'=>28,'name'=>'abc"');

foreach($\_POST as $k=>$v) {

if(is\_string($v)) {

$\_POST[$k] = addslashes($v);

}

}

array\_walk\_recursive ,这是递归处理数组单元的函数

// 函数本身只有一个功能------递归的把数组每个单元走一遍

// 至于你怎么处理这个数组,自己写一个函数,

// 然后 array\_walk\_recursive会带着你写的函数把数组单元走一遍.

如下,array\_walk\_recursive带着addslashes在数组各单元走一遍,

// 自然,各单元都被转义了.

function abc(&$v,$k) {

$v = addslashes($v);

}

array\_walk\_recursive(&$\_GET,'abc');

array\_walk\_recursive — 对数组中的每个成员递归地应用用户函数

在PHP5.3中魔术引号已经被废弃，在PHP5.4中已经被移除

/\*

1: php.ini里修改error\_reporting 选项

2: 可以在php页面里,用error\_reporting()函数来修改

\*/

/\*

错误级别用2进制的值来表示的

1111 1111 1111 111

从左到右,每位上的1,代表一种错误级别

\*/

/\*

fatal error致命错误: 0000 0000 0000 001 开启

warning 警告错误 : 0000 0000 0000 010 开启

NOTICE 警告 : 0000 0000 0001 000 开启

\*/

系统为我们把各个级别的值,用系统常量代替了.

E\_ERROR 1

E\_WARNING 2

E\_NOTICE 8

报除了notice之个的所有错误,即除了NOTICE,其他都报.

error\_reporting(E\_ALL & ~E\_NOTICE);

时间戳: 是指1970-01-01 00:00:00 --> 某个时刻所经历的秒数

问:时间戳虽利于计算机来计算,但人眼看起来,还是不直观啊?

答:有专门的函数,用来把时间戳格式化你想要的格式,date函数

strtotime,则是把字符串形式的时间转化成时间戳

$birth = '1985-05-17';

echo strtotime($birth),'<br />';

// 从当前时间,退回1天的时间戳

echo strtotime('-1 day'),'<br />';

// 退回到一周前的时间戳

echo strtotime('-1 week'),'<br />';

贴吧项目：

建表部分

数据库的建模部分很重要，要想做到项目经理必须要懂得数据库建模！！！

mysql\_connect()代码没有重用，有可能发布恶意的sql注入代码

mysql\_query();mysql\_fetch\_assoc();一般封装成一个函数

==========建库建表语句=====

create database tieba charset utf8;

use tieba;

create table thread (

tid int not null auto\_increment primary key,

username varchar(20) not null default '',

title varchar(30) not null default '',

content text,

pubtime int unsigned not null default 0

)engine myisam charset utf8;

create table reply (

rid int not null auto\_increment primary key,

tid int not null default 0,

username varchar(20) not null default '',

content text,

reptime int unsigned not null default 0

)engine myisam charset utf8;

新版本已经不允许在函数调用时，采用引用的方式，如 getFormMethod(&$method)，

mysql\_insert\_id — 取得上一步 INSERT 操作产生的 ID

mysql\_insert\_id() 返回给定的 link\_identifier 中上一步 INSERT 查询中产生的 AUTO\_INCREMENT 的 ID 号。如果没有指定 link\_identifier，则使用上一个打开的连接。