1.分界标示符

PHP分界标示符确定PHP脚本的开始和结束位置，在PHP语言中有四种不同风格的分界标示符可以让PHP代码嵌入到HTML代码中，分别如下：

a.PHP标准分界符：<?php 和 ?>

b.PHP简写分界符：<? 和 ?>

c.JavaScripte 和VBScript风格分界符：<script language=’php’> 和</script>

d.ASP风格分界符: <% 和 %>

2.标示符：由字符，数字，下划线组成，首字母必须是字符或下划线

变量以$开头。

3.PHP是区分大小写的。

4.语句以分号结束

5.注释：

/\* \*/

//

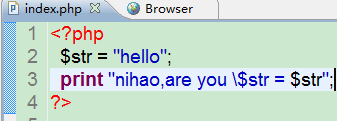
#

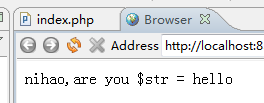
共上面三种

6.数据类型：integer，double，string，array，object(对象类型)，Boolean，resource(资源)，NULL

7.字符串的定义：用双引号或者单引号，但有区别

双引号里面的变量和转义字符都会被解析，而单引号里面的就都当做是字符串来处理。如果在双引号中要想不解析变量，就可以用’\’加在前面。





<?php

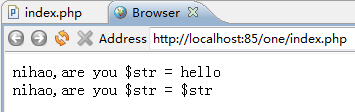
$str = "hello";

**print** "nihao,are you \$str = $str";

**print** "<br/>";

**print** 'nihao,are you $str = $str';

?>



8.数组初始化：

数组定义：

$arr = **array**();

$arr[0]='nihoa';

$arr[1]=1;

正常数组初始化：

方式一：

$arr\_family=**array**('LinNa','Jhon','Peter');

方式二：

$arr\_family[] = 'LinNa';

$arr\_family[] = 'Jhon';

$arr\_family[] = 'Peter';

哈希表形式数组初始化：

方式一：

$arr\_family['mother'] = 'LinNa';

$arr\_family['father'] = 'Jhon';

$arr\_family['mother'] = 'Peter';

方式二：

$arr\_family=**array**('mother'=>'LinNa','father'=>'Jhon','son'=>'Peter');

9.对象：

类：

Class 类名

{

Public $属性名；

Public function 函数名()

{

$变量名;

}

}

<?php

**class** foo{

**public** $C\_foo = 'This is a class';

**public function** do\_foo()

{

**echo** 'Doing foo';

}

}

$newfoo = **new** foo();

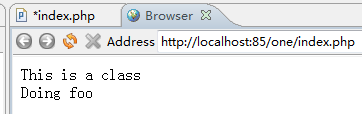
**print** $newfoo->C\_foo;

**print** "<br/>";

$newfoo->do\_foo();

?>

效果如下：



10.NULL

空，可以使null，也可以使NULL,不区分大小写

NULL表示没有值的变量。

11.数据类型转换

一般情况下，数据类型会自动转换，同时我们也可以进行强制类型转换，强制类型转换的方法是把数据类型用括号括起来放在要转换的数据类型前面即可。

(int)/(integer) 转换为整形值

(real)，(double)，(float)转换为双精度

(string)转换为字符串

(array)转换为数组

(object)转换为对象。

12.变量不需要定义就可以使用。

在函数外部定义的变量时全局变量，在函数内部**定义或使用的变量**是局部变量。

**在函数内部不能访问函数外部的变量**

要想访问，就必须在函数内部，复制一个全局变量，global 全局变量名，这样用的变量才是全局变量。

<?php

$a =1;

$b =2;

**function** Sum()

{

$b=$a+$b;

**print** "Internal \$b is $b";

}

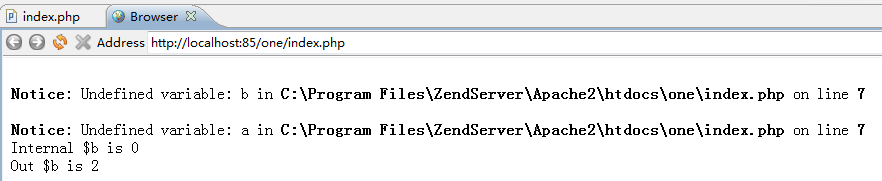
Sum();

**print** "<br>";

**print** "Out \$b is $b";

?>

效果如下：



从上面可以看到，函数里面是没法调用全局变量的，要想调用，需要申明。

<?php

$a =1;

$b =2;

**function** Sum()

{

**global** $a;

**global** $b;

$b= $a+ $b;

**print** "Internal \$b is $b";

}

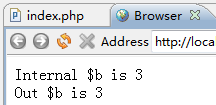
Sum();

**print** "<br>";

**print** "Out \$b is $b";

?>

效果如下：



我们看见，要想在函数里面调用全局变量，必须先用global来复制一遍这个变量，才能使用。

13.静态变量：用static修饰的变量

1.对于全局变量，只要定义了，就会以键值的形式存到$GLOBAL这个数组中，所以，如果我们想在函数中使用全局变量，也可以$GLOBAL[$全局变量名]的形式来引用。

$GLOBALS：包含指向当前程序中全局范围内有效的变量，它是一个数组，该数组的索引（或键名）就是全局变量的名称。

2.PHP还可以获得外部来的变量，外部来的有四种类型：

$\_GET：通过HTTP的GET方法提交至脚本的表单变量。

$\_POST：通过HTTP的POST方法提交至脚本的表单变量。

$\_FILE：通过HTTP的POST文件上传提交至脚本的变量。

$\_COOKIE：通过HTTP的Cookies方法提交至脚本的变量。

NewHtml.php程序如下：

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"*>

<title>Insert title here</title>

</head>

<body>

<p>This page uses frames. The current browser you are using does not support frames.</p>

<form action=*"index.php"* method=*"post"*>

User:<input type=*"text"* name=*"name"*><br>

Password:<input type=*"password"* name=*"password"*><br>

Sex:<select name=*"sex"*>

<option value=*"female"*>female</option>

<option value=*"male"*>male</option>

</select>

<input type=*"submit"* value=*"Submit"*>

</form>

</body>

</html>

index.php程序如下：

<?php

$name = $\_POST['name'];

$password=$\_POST['password'];

$sex=$\_POST['sex'];

**echo** "name is $name";

**echo** "<br>";

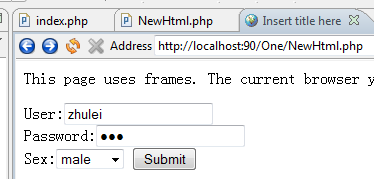
**echo** "passowrd is $password";

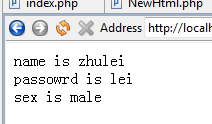
**echo** "<br>";

**echo** "sex is $sex";

?>

效果如下：





其他方式也是一样的。

3.检测是否为那个数据类型：

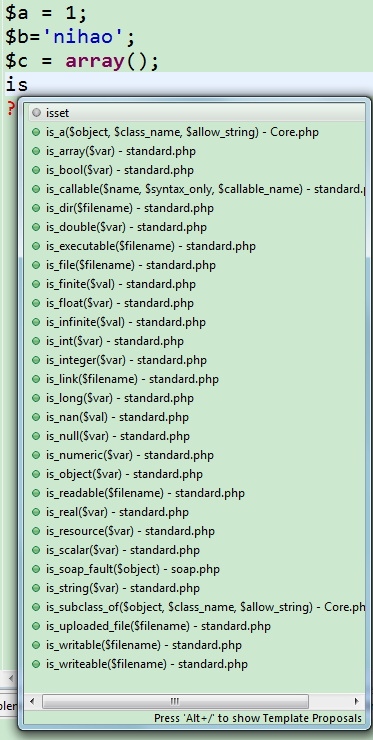
在PHP中，可以通过以下函数对变量的类型做判断。

函数is\_integer判断变量是否为整数。

函数is\_string判断变量是否为字符串。

函数is\_double判断变量是否为浮点数。

函数is\_array判断一个变量是否为数组。



我们会发现有很多很多的检测，用到可以直接拿来用。

4.获得数据类型：

在PHP中，可以使用预定义函数gettype取得一个变量的类型，它接受一个变量作为参数，返回这个变量的类型。

<?php

$a = 1;

$b='nihao';

$c = **array**();

**echo** "\$a is " . gettype($a);

**echo** "<br>";

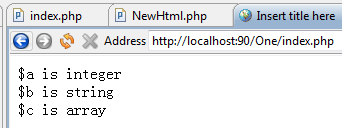
**echo** "\$b is " . gettype($b);

**echo** "<br>";

**echo** "\$c is " . gettype($c);

?>

效果如下：



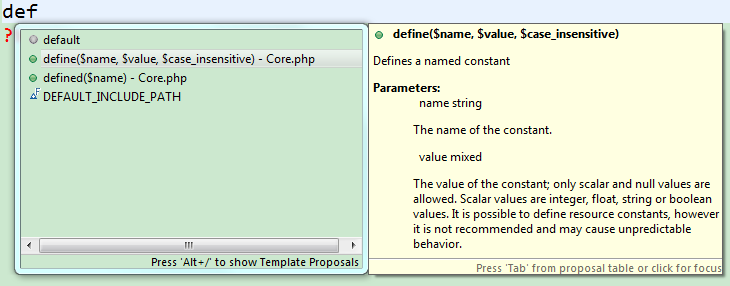
使用预定义函数settype设置一个变量的类型，该函数接受两个参数，第一个参数是变量名，第二个参数是要设置的变量的数据类型。

使用预定义函数isset判断一个变量是否已经定义，它接受一个变量作为参数值，返回值如果为TRUE，说明该变量定义过，否则，说明该变量没有被定义。

使用unset语句删除一个变量。从PHP4开始unset不再有返回值，因此，严格意义上讲，它并不是一个函数，而是一个PHP的语言结构。可以用unset一次删除多个PHP变量。

**unset**($a);

在PHP中通过define()函数定义一个常量。合法的常量名只能以字母和下划线开始，后面可以跟着任意字母、数字或下划线。常量一旦定义就不能再修改或者取消定义。



5.如果是多选框，虽然名字相同，但可能选择好多个，怎么处理呢？

可以再命名名字的时候，加[].

但是单选框不一样，因为单选框只有一个值，可以不需要加[]

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"*>

<title>Insert title here</title>

</head>

<body>

<p>This page uses frames. The current browser you are using does not support frames.</p>

<form action=*"index.php"* method=*"post"*>

User:<input type=*"text"* name=*"name"*><br>

Password:<input type=*"password"* name=*"password"*><br>

Sex:<select name=*"sex"*>

<option value=*"female"*>female</option>

<option value=*"male"*>male</option>

</select><br>

Gender:<input type=*"radio"* name=*"gender"* value=*"male"*>male

<input type=*"radio"* name=*"gender"* value=*"female"*>female<br>

Hobby:<input type=*"checkbox"* value=*"read"* name=*"hobby[]"*> read

<input type=*"checkbox"* value=*"basketball"* name=*"hobby[]"*> basketball

<input type=*"checkbox"* value=*"travel"* name=*"hobby[]"*> travel

<input type=*"checkbox"* value=*"sing"* name=*"hobby[]"*> sing<br>

<input type=*"submit"* value=*"Submit"*>

</form>

</body>

</html>

index.php:

<?php

$name = $\_POST['name'];

$password = $\_POST['password'];

$gender = $\_POST['gender'];

$hobby = $\_POST['hobby'];

$like="";

**foreach** ($hobby **as** $one)

{

$like .="、".$one;

}

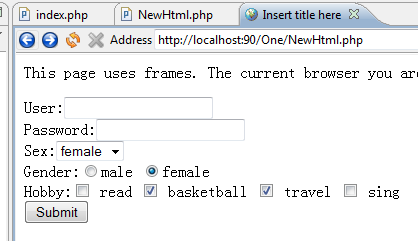
**echo** "gender is $gender";

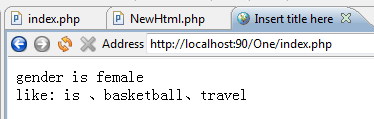
**echo** "<br>";

**echo** "like: is $like";

?>

效果如下：





6. 使用print\_r函数显示数组元素

有时，为了在程序中查看一个数组的结构（即索引和值的对应关系），需要将数组结构显示到页面上。这可以通过PHP的print\_r函数实现。

<?php

$ms\_office = **array**(

0=>'word',

3=>'excel',

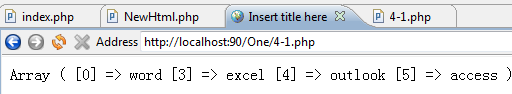
'outlook',

'access');

print\_r($ms\_office);

?>

效果如下：



<?php

$ms\_office = **array**(

'word',

'excel',

'outlook',

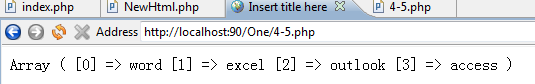
'access'

);

print\_r($ms\_office);

?>

效果如下：



7.实现数组分割

使用函数array\_chunk可以对一个数组进行分割，该函数语法如下所示。

array array\_chunk(array $input\_array, int $size, bool $preserve\_key);

该函数接受3个参数，$input\_array是将要分割的数组（即原数组），$size表示原数组被分割后，每个数组中元素的个数，$preverse\_key是可选参数，它为TRUE时，分割后的每个数组的索引使用原数组的索引名；它为FALSE时，分割后的每个数组的索引都将使用从0开始的整数索引。该函数的返回值是一个数组，这个数组的索引从0开始，该数组的每个元素仍然是一个数组，即原数组被分割后的一个数组。

<?php

$ms\_office = **array**(

'wd'=>'word',

'ec'=>'excel',

'ol'=>'outlook',

'ac'=>'access',

'vs'=>'visio'

);

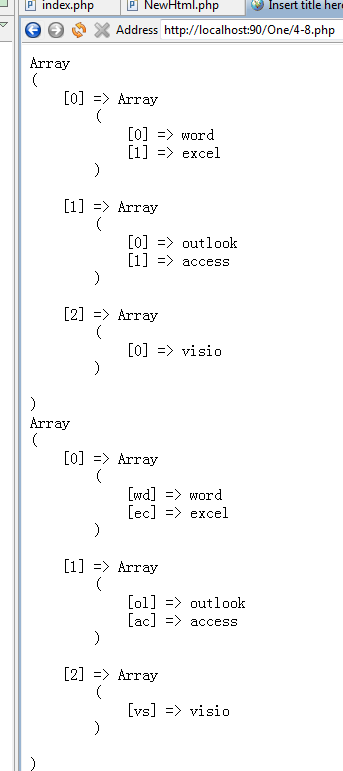
**echo** '<pre>';//作用就是让输出按格式来，不是直接乱七八糟输出

print\_r(array\_chunk($ms\_office,2));

print\_r(array\_chunk($ms\_office,2,**TRUE**));

?>

效果如下：



8. 实现数组合并

使用函数array\_merge可以合并一个或多个数组，该函数的语法如下所示。

array array\_merge($array…);

该函数接受一个或多个数组作为参数，PHP5中，它只接受array类型的参数。该函数返回一个合并后的数组。

<?php

$arr1 = **array**('Earth','Venus');

$arr2 = **array**(4=>'Mars',5=>'Jupiter',6=>'Saturn');

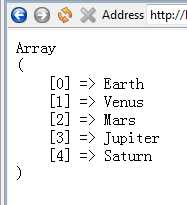
$planet = array\_merge($arr1,$arr2);

**echo** '<pre>';

print\_r($planet);

?>

效果如下：



9. 处理数组元素和键值位置

PHP的每个数组内部均会有一个指针，指向当前元素。这里所谓的指针，可以把它理解成程序即将要处理的元素所在的位置。处理数组元素和键值（索引）的位置，就是获取数组某个位置上的元素，或获取该元素后，将该指针移动到其他位置。PHP中，处理数组元素和键值位置的函数主要有以下几个。

current：该函数返回数组的当前元素，不移动当前指针位置。

next：该函数返回当前元素的下一个元素的值，并将指针向后移动一位，下一个元素不存在时，返回FALSE。

prev：该函数返回当前元素的上一个元素的值，并将指针向前移动一位，上一个元素不存在时，返回FALSE。

end：该函数返回数组的最后一个元素的值。

<?php

$planet = **array**(

'Earth',

'Venus',

'Mars',

'Jupiter',

'Saturn'

);

$pos = current($planet); //此时$pos=Earth

**echo** 'pos1='.$pos;

**echo** '<br/>';

**echo** '<br/>';

$pos = next($planet); //此时$pos=Venus

**echo** 'pos2='.$pos;

**echo** '<br/>';

**echo** '<br/>';

$pos = current($planet); //此时$pos=Venus

**echo** 'pos3='.$pos;

**echo** '<br/>';

**echo** '<br/>';

$pos = prev($planet); //此时$pos=Earth

**echo** 'pos4='.$pos;

**echo** '<br/>';

**echo** '<br/>';

$pos = end($planet); //此时$pos=Saturn

**echo** 'pos5='.$pos;

**echo** '<br/>';

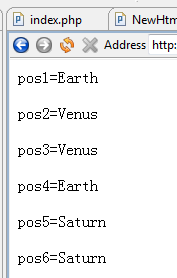
**echo** '<br/>';

$pos = current($planet); //此时$pos=Saturn

**echo** 'pos6='.$pos;

?>

效果如下：



10.数组建新索引排序：

10.1使用函数sort对数组元素按顺向排序，即按字母由前向后或按数字由小到大排序。该函数的语法如下所示。

bool sort(array &$arr [,int $sort\_flag]);

该函数接受一个数组作为输入参数，如果处理成功，该函数返回TRUE，否则，该函数返回FALSE。该函数将会删除所要排序的数组原有的索引值，从而为该数组使用新的索引值。

10.2使用函数rsort对数组元素做逆向排序，即按字母从后到前或按数字由大到小排序。该函数的语法如下所示。

bool rsort(array &$arr [,int $sort\_flag]);

该函数接受一个数组作为输入参数，如果处理成功，函数返回TRUE，否则返回FALSE。该函数将会删除所要排序的数组原有的索引值，从而为该数组使用新的索引值。

11.数组保持原索引的排序：

asort：该函数对数组元素进行排序，并且保持数组的原索引关系不变。该函数仍然对数组元素按顺向排序。如果处理成功，该函数返回TRUE，否则返回FALSE。

ksort：该函数对数组元素按索引名顺序排序，并且保持数组的原索引关系不变，如果处理成功该函数返回TRUE，否则返回FALSE。该函数主要用于关联数组。

<?php

$planet1 = **array**(

'Earth',

'Venus',

'Mars',

'Jupiter',

'Saturn'

);

$planet2 = **array**(

'X'=>'Earth',

'Y'=>'Venus',

'Z'=>'Mars',

'A'=>'Jupiter',

'B'=>'Saturn'

);

asort($planet1);

ksort($planet2);

**echo** '使用函数asort对数组元素排序：';

**echo** '<br/>';

**foreach**($planet1 **as** $key => $value)

{

**echo** 'planet1['.$key.']='.$value;

**echo** '<br/>';

**echo** '<br/>';

}

**echo** '<br/>';

**echo** '使用函数ksort对数组元素排序：';

**echo** '<br/>';

**foreach**($planet2 **as** $key => $value)

{

**echo** 'planet2['.$key.']='.$value;

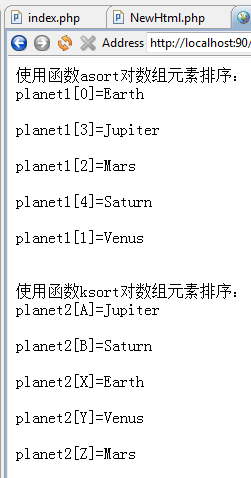
**echo** '<br/>';

**echo** '<br/>';

}

?>

效果如下：



12. 数组的随机排序

PHP提供一个随机排序数组元素的函数shuffle()，该函数的语法如下所示。

bool shuffle (array $array)

该函数为参数数组$array中的元素赋予新的键名，这将删除原有的键名而不仅仅是重新排序。如果执行成功该函数返回TRUE。

13. 数组的反向排序

本节最后，介绍一个可以将数组元素按原顺序反向排序的函数，该函数是array\_reverse()。其语法如下所示。

array array\_reverse (array $array [, bool $preserve\_keys])

该函数返回一个与原数组单元顺序相反的数组，原数组以参数$array传入该函数，该函数的第2个参数是可选的，如果第2个参数$preserve\_keys设置为TRUE，则原数组的键名将会被保留。

14. 重置一个数组

PHP中，重置一个数组指的是，将数组中的内部指针移动到该数组的第一个元素所在的单元。使用函数reset可以完成对数组的重置。该函数的语法如下所示。

mixed reset(array &$arr);

该函数接受一个数组类型的变量，将函数内部指针指向数组第一个单元，并将第一个单元的元素返回。如果数组为空，该函数返回FALSE。

<?php

$planet = **array**(

'Earth',

'Venus',

'Mars',

'Jupiter',

'Saturn'

);

**echo** '当前元素是：'.current($planet);

**echo** '<br/>';

next($planet);

next($planet);

**echo** '<br/>';

**echo** '调用两次next函数之后，当前元素是：'.current($planet);

**echo** '<br/>';

reset($planet);

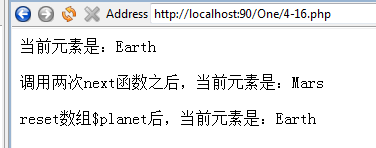
**echo** '<br/>';

**echo** 'reset数组$planet后，当前元素是：'.current($planet);

**echo** '<br/>';

?>

效果如下：



15. 用数组对变量赋值

PHP中，可以将数组的元素作为值赋给变量，使用语言结构list()可以完成这个功能。严格意义上讲，list()和array()一样，不是函数。

在下述程序中，通过list，将数组$planet的第一个元素’Earth’赋给了变量$no1，将第二个元素赋给了变量$no2，将第3个元素赋给了变量$no3。

<?php

$planet = **array**(

'Earth',

'Venus',

'Mars',

'Jupiter',

'Saturn'

);

**list**($no1,$no2,$no3) = $planet; //将数组中的前3个元素分别赋给list结构中对应的3个变量

**echo** 'no1='.$no1;

**echo** '<br/>';

**echo** '<br/>';

**echo** 'no2='.$no2;

**echo** '<br/>';

**echo** '<br/>';

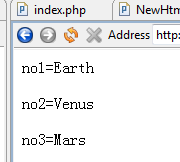
**echo** 'no3='.$no3;

**echo** '<br/>';

**echo** '<br/>';

?>

效果如下：



16. 快速创建数组

函数range()可以快速创建制定元素范围的数组。其语法如下所示。

array range(mixed $low, mixed $high [, number $step])

函数range()返回一个数组，其中元素为参数从$low到$high的序列，包括它们本身。例如range(1,9)相当于array(1,2,3,4,5,6,7,8,9)，如果$low>$high，则序列将从$high到$low。该函数的第3个参数$step是可选的，如果给出了step的值，它将被作为元素之间的跨度值。step应该为正值。如果未指定，step则默认为1。

<?php

**echo** '<pre>';

$arr1 = range(5,10);

print\_r($arr1);

$arr2 = range('a','f');

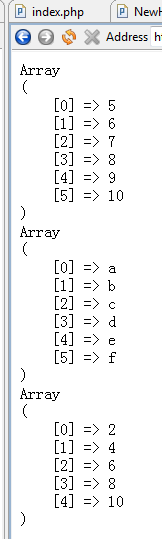
print\_r($arr2);

$arr3 = range(2,10,2);

print\_r($arr3);

?>

效果如下;



17. 压入和弹出数组元素

这里所说的压入是指，向数组的末尾增加一个元素。而弹出是指，将数组中的最后一个元素取出。PHP提供两个函数分别完成这两个功能，这两个函数是array\_push()和array\_poo()。需要注意的是，使用本函数后会重置数组指针，相当于使用了函数reset()。

18. 改变数组字符索引名的大小写

PHP提供了一个函数，可以将以字符串作为数组索引（键名）的大小写全部更换，这个函数是array\_change\_key\_case()，该函数的语法如下所示。

array array\_change\_key\_case(array $input [, int $case])

函数array\_change\_key\_case()将参数$input数组中的所有索引改为全小写或大写。可选参数$case用来指定大小写转换方式，它为两个常量值：CASE\_UPPER和CASE\_LOWER。没有指定该参数时，该函数使用的默认值是CASE\_LOWER。

<?php

$olympic = **array**(

'Barcelona'=>1992,

'AtLanTa'=>1996,

'sydney'=>2000,

'AthEns'=>2004,

'BEIJING'=>2008

);

**echo** '原数组：';

**echo** '<pre>';

print\_r($olympic);

**echo** '</pre>';

$nol = array\_change\_key\_case($olympic, CASE\_UPPER);

**echo** '<br/>';

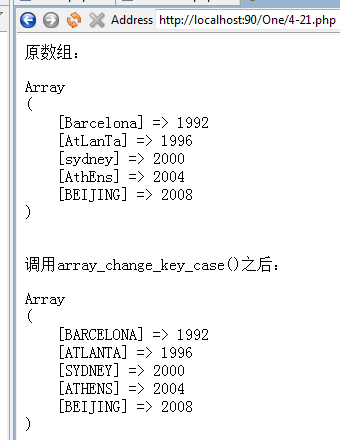
**echo** '调用array\_change\_key\_case()之后：';

**echo** '<pre>';

print\_r($nol);

?>

效果如下：



19. 交换数组索引和元素

PHP中的数组可以通过函数实现索引和元素的交换，这对程序实现某些操作带来极大的便利，这个函数就是array\_flip();

<?php

$olympic = **array**(

'Barcelona'=>1992,

'Atlanta'=>1996,

'Sydney'=>2000,

'Athens'=>2004,

'Beijing'=>2008

);

**echo** '原数组：';

**echo** '<pre>';

print\_r($olympic);

**echo** '</pre>';

$nol = array\_flip($olympic);

**echo** '<br/>';

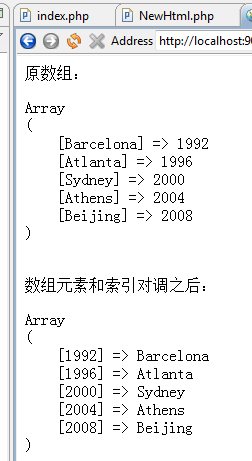
**echo** '数组元素和索引对调之后：';

**echo** '<pre>';

print\_r($nol);

?>

效果如下：



20. 快速填充数组元素

PHP提供函数快速填充数组，将数组中的某些或所有元素都置成通一个值，这个函数是array\_fill()，其语法如下所示。

array array\_fill(int $start, int $num, mixed $value)

函数array\_fill()有3个参数：其中参数$start指定要填充元素的开始索引，参数$num表示将$num个数组元素以参数$value填充。

<?php

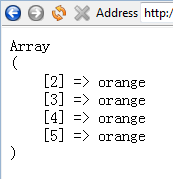
$arr = array\_fill(2, 4, 'orange');

**echo** '<pre>';

print\_r($arr);

?>

效果如下：



21. 统计数组元素出现次数

函数cout()用来计算整个数组元素的个数，而函数array\_count\_values()用来统计数组中各元素出现的次数，即数组中各元素的个数。函数array\_count\_values()的语法如下所示。

array array\_count\_values(array $arr)

该函数返回一个数组，这个数组把参数数组$arr中的元素作为索引，元素出现的次数作为对应索引的值。

函数array\_count\_values()统计出数组$arr中，各个元素出现的次数，而且从这个执行结果可以看出，函数array\_count\_values()统计元素时，是区分元素大小写的，比如Say和say会分别被统计。

<?php

$say = **array**(

'Say',

'you',

'say',

'me',

'Say',

'it',

'together'

);

**echo** '原数组：';

**echo** '<pre>';

print\_r($say);

**echo** '</pre>';

$say\_tmp = array\_count\_values($say);

**echo** '<br/>';

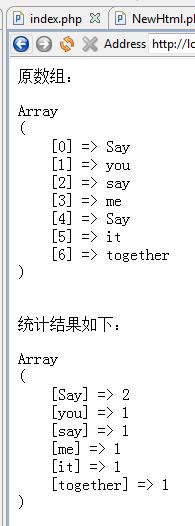
**echo** '统计结果如下：';

**echo** '<pre>';

print\_r($say\_tmp);

?>

效果如下：



22. 检查数组索引是否存在

PHP提供函数array\_key\_exists()检查给定的索引是否存在于数组中，该函数语法如下所示。

bool array\_key\_exists(mixed $key, array $search)

该函数查找由参数$key指定的索引，是否存在于由参数$sarch指定的数组中，如果存在，该函数返回TRUE，否则或者出错时，该函数返回FALSE。参数$key可以是任何能作为数组索引的值。

23. 取得数组中的所有索引

PHP提供函数array\_keys()用来获取数组中所有的索引名，该函数语法如下所示。

array array\_keys(array $arr [, mixed $search\_value [, bool $strict]])

该函数会将参数数组$arr中，所有的索引名返回到一个数组中。参数search\_value是可选的，如果指定该参数，则函数array\_keys只返回该元素值对应的索引名，可选参数$strict用来进行全等比较。

<?php

$arr1 = **array**(

0=>100,

"gold"=>"money"

);

$arr2 = **array**(

'Sunday',

'Saturday',

'Monday',

'Sunday',

'Sunday'

);

**echo** '<pre>';

**echo** '数组arr1的全部索引是：';

print\_r(array\_keys($arr1));

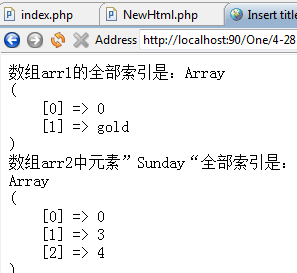
**echo** '数组arr2中元素”Sunday“全部索引是：';

**echo** '<br/>';

print\_r(array\_keys($arr2,'Sunday'));

?>

效果如下：



24. 用回调函数处理数组

PHP支持回调函数操作给定数组的元素上，这也是PHP数组函数的强大功能之一。函数array\_map()就可以完成这样的工作，它将回调函数作用到给定数组的元素上。其语法如下所示。

array array\_map(callback\_func, array $arr1 [, array ...])

函数array\_map()返回一个数组，该数组包含了参数$arr1中的所有元素经过函数callback\_func处理过之后的元素。callback接受参数的数目应该和传递给array\_map()函数的数组数目一致。

<?php

**function** cube($n)

{

$cb = $n\*$n\*$n;

**return** $cb;

}

$a = **array**(1, 2, 3, 4, 5);

$b = array\_map("cube", $a);

**echo** '计算原数组各元素的立方，结果如下：';

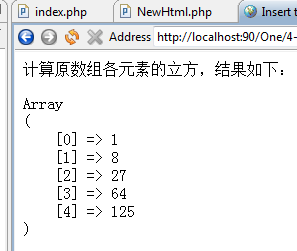
**echo** '<br/>';

**echo** '<pre>';

print\_r($b);

?>

效果如下：



25. in\_array：该函数可以判断某个值是否存在于数组中。如果在数组中找到该值，函数返回TRUE，否则返回FALSE。

key：该函数可以获取数组中当前单元的索引值。

<?php

$planet = **array**(

'Earth',

'Venus',

'Mars',

'Jupiter',

'Saturn'

);

$temp = 'mars';

**if**(in\_array('Mars',$planet))

{

**echo** 'Mars 存在于数组$planet中';

**echo** '<br/>';

**echo** '<br/>';

}

**if**(in\_array($temp,$planet))

{

**echo** $temp.'存在于数组$planet中';

**echo** '<br/>';

**echo** '<br/>';

}

**else**

{

**echo** $temp.'不存在于数组$planet中';

**echo** '<br/>';

**echo** '<br/>';

}

?>

效果如下：



<?php

$planet = **array**(

'Eth'=>'Earth',

'Vns'=>'Venus',

'Mrs'=>'Mars',

'Jpt'=>'Jupiter',

'Stn'=>'Saturn'

);

end($planet);

**echo** '当前元素的索引为：'.key($planet);

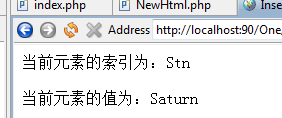
**echo** '<br/>';

**echo** '<br/>';

**echo** '当前元素的值为：'.end($planet);

?>

效果如下：



14.获取HTML的变量

HTML中的设计：

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<form action=*"index.php"* method=*"post"*>

name:<input type=*"text"* name=*"User[name]"*><br>

sex:<input type=*"text"* name=*"User[sex]"*><br>

Email:<input type=*"text"* name=*"User[Email]"*><br>

<input type=*"submit"* value=*"Submit"*>

</form>

</body>

</html>

Index.php文件：

<?php

$user =$\_POST['User'];

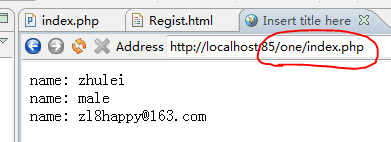
**echo** "name: $user[name]<br>\n";

**echo** "name: $user[sex]<br>\n";

**echo** "name: $user[Email]<br>\n";

?>

效果：



15.运算符：+ - \* / %

16.连接字符串： 用 . 把两个字符串连接起来，如果两个字符串中有一个数字，将自动转换为字符串，但数字两边要加空格，如果两个都是数字，如果不加空格，就会当做小数点来处理，如果当空格，就会当连接字符串处理。

**echo** "gege". 3 . 1 . "nihao". 3.1;

?>

效果如下：



17.赋值运算：

%= 将左边的值对右边取余数

.= 将右边的字符串加到左边

18.关系运算符：

== 等于 $a==$b $a 和$b的值相等

=== 恒等于 $a===$b $a和$b的数据类型和值都相等。

$a = 4;

$b = '4';

**if**($a==$b)

{

**echo** '$a==$b';

}

**else**

{

**echo** '$a!=$b';

}

**echo** "<br>";

**if**($a===$b)

{

**echo** '$a===$b';

}

**else**

{

**echo** '$a!==$b';

}



19.语句：

If

If else if

Switch()

{

Case “”:

Xxxx

Case “”:

xxxxx

}

For

While()

{}

Do

{}while();

Break;

Continue;

20.函数

函数的参数传递有两种：一种是值传递，一种是引用传递

值传递是在传递的过程中，又创建了一个副本，导致不会改变原来的数据

引用传递是会改变原来数据的。

值传递函数：function add($a,$b)

引用传递函数：funciton add(&$a,&$b)

<?php

$a=1;

$b=2;

**function** addvalue($arg1,$arg2)

{

$arg1=$arg1+$arg2;

**print** "internal \$a is $arg1";

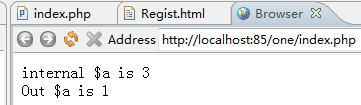
}

addvalue($a, $b);

**print** "<br>Out \$a is $a";

?>

效果如下：



<?php

$a=1;

$b=2;

**function** addvalue(&$arg1,&$arg2)

{

$arg1=$arg1+$arg2;

**print** "internal \$a is $arg1";

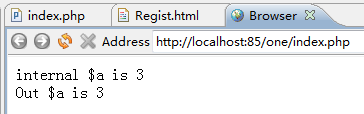
}

addvalue(&$a, &$b);

**print** "<br>Out \$a is $a";

?>

效果如下：



21.系统函数：date()

格式：

Date(string formate[,int timestamp])

Formate 如下：

| **格式字串可以识别以下 *format* 参数的字符串** | | |
| --- | --- | --- |
| ***format* 字符** | **说明** | **返回值例子** |
| ***日*** | --- | --- |
| *d* | 月份中的第几天，有前导零的 2 位数字 | *01* 到 *31* |
| *D* | 星期中的第几天，文本表示，3 个字母 | *Mon* 到 *Sun* |
| *j* | 月份中的第几天，没有前导零 | *1* 到 *31* |
| *l*（“L”的小写字母） | 星期几，完整的文本格式 | *Sunday* 到 *Saturday* |
| *N* | ISO-8601 格式数字表示的星期中的第几天（PHP 5.1.0 新加） | *1*（表示星期一）到 *7*（表示星期天） |
| *S* | 每月天数后面的英文后缀，2 个字符 | *st*，*nd*，*rd* 或者 *th*。可以和 *j* 一起用 |
| *w* | 星期中的第几天，数字表示 | *0*（表示星期天）到 *6*（表示星期六） |
| *z* | 年份中的第几天 | *0* 到 *366* |
| ***星期*** | --- | --- |
| *W* | ISO-8601 格式年份中的第几周，每周从星期一开始（PHP 4.1.0 新加的） | 例如：*42*（当年的第 42 周） |
| ***月*** | --- | --- |
| *F* | 月份，完整的文本格式，例如 January 或者 March | *January* 到 *December* |
| *m* | 数字表示的月份，有前导零 | *01* 到 *12* |
| *M* | 三个字母缩写表示的月份 | *Jan* 到 *Dec* |
| *n* | 数字表示的月份，没有前导零 | *1* 到 *12* |
| *t* | 给定月份所应有的天数 | *28* 到 *31* |
| ***年*** | --- | --- |
| *L* | 是否为闰年 | 如果是闰年为 *1*，否则为 *0* |
| *o* | ISO-8601 格式年份数字。这和 *Y* 的值相同，只除了如果 ISO 的星期数（*W*）属于前一年或下一年，则用那一年。（PHP 5.1.0 新加） | Examples: *1999* or *2003* |
| *Y* | 4 位数字完整表示的年份 | 例如：*1999* 或 *2003* |
| *y* | 2 位数字表示的年份 | 例如：*99* 或 *03* |
| ***时间*** | --- | --- |
| *a* | 小写的上午和下午值 | *am* 或 *pm* |
| *A* | 大写的上午和下午值 | *AM* 或 *PM* |
| *B* | Swatch Internet 标准时 | *000* 到 *999* |
| *g* | 小时，12 小时格式，没有前导零 | *1* 到 *12* |
| *G* | 小时，24 小时格式，没有前导零 | *0* 到 *23* |
| *h* | 小时，12 小时格式，有前导零 | *01* 到 *12* |
| *H* | 小时，24 小时格式，有前导零 | *00* 到 *23* |
| *i* | 有前导零的分钟数 | *00* 到 *59*> |
| *s* | 秒数，有前导零 | *00* 到 *59*> |
| ***时区*** | --- | --- |
| *e* | 时区标识（PHP 5.1.0 新加） | 例如：*UTC*，*GMT*，*Atlantic/Azores* |
| *I* | 是否为夏令时 | 如果是夏令时为 *1*，否则为 *0* |
| *O* | 与格林威治时间相差的小时数 | 例如：*+0200* |
| *P* | 与格林威治时间（GMT）的差别，小时和分钟之间有冒号分隔（PHP 5.1.3 新加） | 例如：*+02:00* |
| *T* | 本机所在的时区 | 例如：*EST*，*MDT*（【译者注】在 Windows 下为完整文本格式，例如“Eastern Standard Time”，中文版会显示“中国标准时间”）。 |
| *Z* | 时差偏移量的秒数。UTC 西边的时区偏移量总是负的，UTC 东边的时区偏移量总是正的。 | *-43200* 到 *43200* |
| ***完整的日期／时间*** | --- | --- |
| *c* | ISO 8601 格式的日期（PHP 5 新加） | 2004-02-12T15:19:21+00:00 |
| *r* | RFC 822 格式的日期 | 例如：*Thu, 21 Dec 2000 16:01:07 +0200* |
| *U* | 从 Unix 纪元（January 1 1970 00:00:00 GMT）开始至今的秒数 | 参见 [time()](http://cn.php.net/manual/zh/function.time.php) |

用时间date会出现错误解释：

实际上，从 PHP 5.1.0 ，当对使用date()等函数时，如果timezone设置不正确，在每一次调用时间函数时,都会产生E\_NOTICE 或者 E\_WARNING 信息。而又在php5.1.0中，date.timezone这个选项，默认情况下是关闭的，无论用什么php命令都是格林威治标准时间，但是PHP5.3中好像如果没有设置也会强行抛出了这个错误的,解决此问题，只要本地化一下，就行了。

以下是三种方法(任选一种都行)：

一、在页头使用date\_default\_timezone\_set()设置 date\_default\_timezone\_set('PRC'); //东八时区 echo date('Y-m-d H:i:s');

二、在页头使用 ini\_set('date.timezone','Asia/Shanghai');

三、修改php.ini。打开php5.ini查找date.timezone 去掉前面的分号修改成为：date.timezone =PRC

重启http服务（如apache2或iis等）即可。

XXX可以任意正确的值。对于我们国内来说：可以为以下值：Asia/Chongqing ，Asia/Shanghai ，Asia/Urumqi （依次为重庆，上海，乌鲁木齐）港台地区可用：Asia/Macao ，Asia/Hong\_Kong ，Asia/Taipei （依次为澳门，香港，台北），还有新加坡：Asia/Singapore，当然PRC也行。

<?php

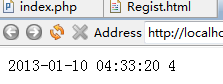
date\_default\_timezone\_set('PRC');

$a = date("Y-m-d h:i:s w");

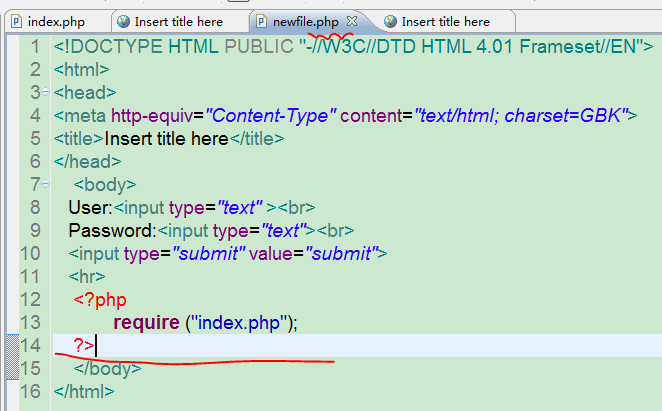
**echo** $a;

?>

效果如下：



22.PHP文件的HTML中，可以添加php文件

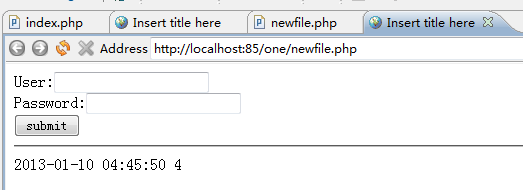


注意上面不是.html 而是.php

在.php里面可以写html 代码

上面我们是通过require来保护php文件，还可以通过include,效果一样。

效果如下：



上面的<hr>是代表添加一横线的意思。

23.数组

下面有数组的定义，赋值，以及获取数组的长度：

<?php

$arr = **array**();

$arr[0]=1;

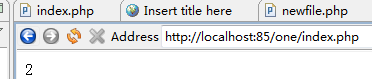
$arr[1]=2;

$count= count($arr);

**echo** $count;

?>

效果如下：



我们可以通过count()函数或者sizeof()函数来获取数组的大小。

24.数组的遍历

数组的遍历可以通过for，也可以通过foreach

Foreach的写法如下：

<?php

$arr = range(1, 10);

**foreach** ( $arr **as** $variable)

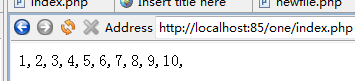
**echo** $variable.",";

?>

Range的作用取得从最小到最大的递增数，默认步长为1，也可以在第三个参数中设置步长。

注意上面的foreach的写法，数组写在前面，变量写在后面

效果如下：



25.非数字索引数组，类哈希数组

按键值方式存储的数组

创建数组：

<?php

$arr = **array**('PHP'=>50,'Java'=>60,'C++'=>'nihao');

$newarr['xiaoming']=30;

$newarr['zhulei']='male';

$newarr[10]='duoduo';

**foreach** ($arr **as** $mykey=>$myvalue)

{

**echo** "\$mykey is $mykey and \$myvalue is $myvalue"."<br>";

}

**foreach** ($newarr **as** $mykey=>$myvalue)

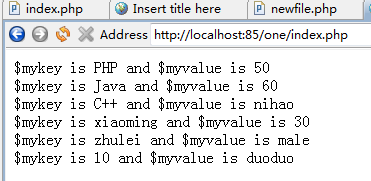
{

**echo** "\$mykey is $mykey and \$myvalue is $myvalue"."<br>";

}

?>

效果如下：



如果要想添加一个数据，直接输就行了,比如添加一个$newarr[‘zhagnyang’]=10;

如果zhangyan已经存在，就会被覆盖。

删除元素：

Unset(数组名[‘键名’])；

<?php

$newarr['xiaoming']=30;

$newarr['zhulei']='male';

$newarr[10]='duoduo';

**foreach** ($newarr **as** $mykey=>$myvalue)

{

**echo** "\$mykey is $mykey and \$myvalue is $myvalue"."<br>";

}

**echo** "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

**echo** "<br>";

$newarr['zhangyang']=10;

**foreach** ($newarr **as** $mykey=>$myvalue)

{

**echo** "\$mykey is $mykey and \$myvalue is $myvalue"."<br>";

}

**echo** "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

**echo** "<br>";

**unset**($newarr['xiaoming']);

**foreach** ($newarr **as** $mykey=>$myvalue)

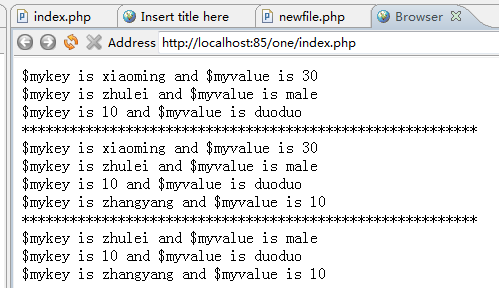
{

**echo** "\$mykey is $mykey and \$myvalue is $myvalue"."<br>";

}

?>

效果如下：



检测键是否存在：

Isset(数组名[‘键名’])

<?php

$newarr['xiaoming']=30;

$newarr['zhulei']='male';

$newarr[10]='duoduo';

**if**(**isset**($newarr['zhulei']))

{

**echo** "yes, key zhulei is exist!";

}

**else**

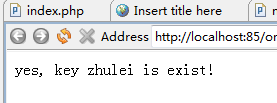
{

**echo** "no, key zhulei is not exist!";

}

?>

效果如下：



检测值是否存在

In\_array(‘值’,数组名)

<?php

$newarr['xiaoming']=30;

$newarr['zhulei']='male';

$newarr[10]='duoduo';

**if**(in\_array(20, $newarr))

{

**echo** "yes, value 20 is exist!";

}

**else**

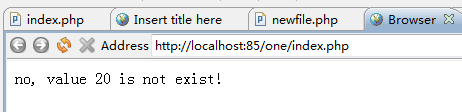
{

**echo** "no, value 20 is not exist!";

}

?>

效果如下：



Php 中的数组操作符：



26.二维数组

count() 函数计算数组中的单元数目或对象中的属性个数。  
对于数组，返回其元素的个数，对于其他值，返回 1。如果参数是变量而变量没有定义，则返回 0。如果 mode 被设置为 COUNT\_RECURSIVE（或 1），则会递归底计算多维数组中的数组的元素个数。  
语法  
count(array,mode)  
参数 描述  
array 必需。规定要计数的数组或对象。  
mode   
可选。规定函数的模式。可能的值：  
0 - 默认。不检测多维数组（数组中的数组）。  
1 - 检测多维数组。

创建二维数组：

<?php

$book = **array**(

**array**('Php','C++','Java'),

**array**(1,2,3),

**array**('zhulei','xiaoming','guozhi'),

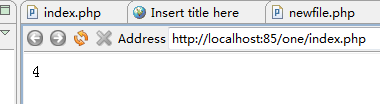
**array**('nihao','ok','tt')

);

**echo** count($book,0);

?>

效果如下：



二维数组取值：

<?php

$book = **array**(

**array**('Php','C++','Java'),

**array**(1,2,3),

**array**('zhulei','xiaoming','guozhi'),

**array**('nihao','ok','tt')

);

**for** ($i =0;$i<4;$i++)

{

**for**($j =0;$j<3;$j++)

{

**echo** $book[$i][$j].",";

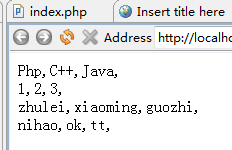
}

**echo** "<br>";

}

?>

效果如下：



我们可以通过count($book)来获取二维数组的第一维的长度

我们可以通过这样的方法来获取第二维数组的长度：

<?php

$book = **array**(

**array**('Php'),

**array**(1,2),

**array**('zhulei','xiaoming','guozhi'),

**array**('nihao','ok')

);

**foreach** ($book **as** $one)

{

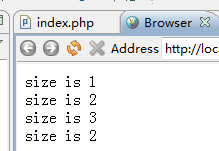
**echo** "size is ".sizeof($one);

**echo** "<br>";

}

?>

效果如下：



对于键值数组也是一样的。

27.数组排序

一维数组：对于数字索引数组，可以用sort()来进行排序，而对于非数字索引，可以使用asort()函数。

<?php

$book = **array**(1,4,3,9,7,5);

sort($book);

**foreach** ($book **as** $one)

{

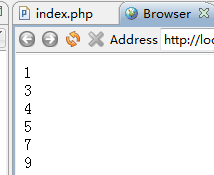
**echo** $one;

**echo** "<br>";

}

?>

效果如下：



Assort()是对键值数组中按值进行排序的，一般值都是数字，这样排序比较有意义；

28.数组的合并

Array\_merge(arr,arr1),把arr1追加到arr中

数组相交：

Array\_intersect(arr,arr1)求arr和arr1的交集。

29.统计数组元素的个数：count(),sizeof()和array\_count\_values()

Count()和sizeof()类似。

而array\_count\_values()会返回一个数组，统计各个值出现的次数。

<?php

$book = **array**(1,4,3,9,7,5,1,2,5,7,3,7,5);

$newback = array\_count\_values($book);

**foreach** ($newback **as** $key=>$value)

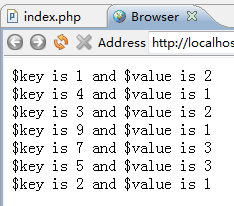
{

**echo** "\$key is $key and \$value is $value";

**echo** "<br>";

}

?>



反转数组：

如果是数字索引数组，就是普通的反转，如果是键值数组，就把键值反转。

Array\_reverse();

**第5章 PHP对字符串的处理**

PHP对字符串的处理的完美支持，是PHP的强大功能之一。PHP提供了数十个用来处理字符串的内置函数，使用这些函数，可以在PHP程序中很方便地完成对字符串的各种操作。本章为读者介绍PHP开发中一些比较常用的字符串处理函数。

5.1 对字符串做分割和合并

1.把字符串切割，explode

2.把数组中间加东西组合成字符串：implode

array explode(string $separator, string $str [,int $limit])

函数explode使用字符串$separator（该函数的第1个参数）作为边界点，来分割字符串$str（该函数的第2个参数）。该函数的返回值是一个数组，字符串被分割后，分割的部分会存入该数组。该函数最后一个参数$limit是可选参数。如果设置了该参数，则返回的数组中最多有limit个元素，最后一个元素包含字符串$str的所有剩余部分。

<?php

$str = "tea,milk,coffee,juice";

$arr = explode(',',$str);

**echo** "<pre>";

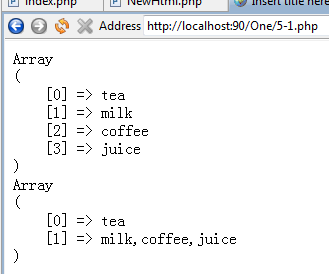
print\_r($arr);

$arr = explode(',',$str,2);

print\_r($arr);

?>

效果如下：



**语法**

implode(separator,array)

|  |  |
| --- | --- |
| **参数** | **描述** |
| separator | 可选。规定数组元素之间放置的内容。默认是 ""（空字符串）。 |
| array | 必需。要结合为字符串的数组。 |

<?php

$drink\_arr = **array**(

"tea",

"milk",

"coffee",

"juice"

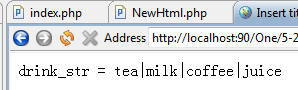
);

$drink\_str = implode('|',$drink\_arr);

**echo** "drink\_str = ".$drink\_str;

?>

组合效果：



5.2 对字符串做比较和替换

字符串比较通常是指，判断两个字符串是否一样，用更专业的说法，就是比较两个字符串的大小。PHP通过字符的ASCII码来决定字符的大小，从而决定字符串的大小。例如，字母a的ASCII码是097，字母b的ASCII码是098，因此’b’>’a’。字母表中，靠后字母的ASCII码大，所有的小写字母比大写字母的ASCII码大。

5.2.1 比较两个字符串大小

PHP提供函数strcmp对两个字符串比较大小。该函数语法如下所示。

int strcmp(string $str1, string $str2);

函数strcmp接受两个参数，它们是待比较的字符串。该函数返回值是整数，如果字符串$str1和字符串$str2相等，则函数返回0，如果字符串$str1小于字符串$str2，则函数返回-1，如果字符串$str1大于字符串$str2，则函数返回值大于0。

5.2.2 字符串的替换

在实际开发中，有时需要将字符串中的某些字符串替换成其他字符串，这可以通过PHP的内置函数str\_replace来完成。该函数的语法如下所示。

string str\_replace(string $search, string $replace, string $str);

该函数将字符串$str中的$search部分全部替换成字符串$replace，并且返回替换后的字符串。

5.3 输出打印字符串

向Web页面输出字符串是PHP程序最常用的操作之一。事实上，在本节之前的几乎示例程序中都有输出字符串的代码。在PHP输出字符串的方式有如下两种。

（1）使用print。它的用法如下所示。

int print(string $str);

print将字符串$str输出，并且总是返回1。严格意义上讲，print不是函数，它和echo一样，只是PHP的语言结构，因此，可以不使用括号将其参数括住。

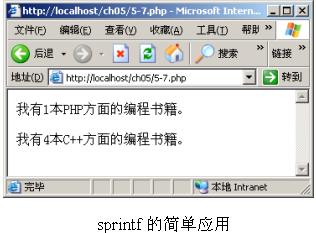
（2）使用echo。使用echo输出字符串，在此之前已经大量使用，这里不再举例说明。

5.4 格式化字符串

将字符串转换成某种特定的格式，叫做字符串的格式化。比如一本书的价格是39元，一般都会格式化成“39.00”的形式。对字符串进行格式化，是程序开发中经常遇到处理操作之一。PHP语言提供函数sprintf，将字符串格式化，该函数的语法如下所示。

string sprintf(string $format, mixed $args);

参数$format是要转换的格式，该参数通常以百分号%开始，到转换字符为止。例如，%1.2f就是一个转换格式，其中f就是转换字符，它表示将字符串格式化成浮点数。这种用法，对于有C语言基础的读者来说，一定不会陌生。该函数返回格式化后的字符串。



5.5 获取字符串中的一个子串

获取字符串中的一个子串是指，获取这个字符串的某一部分。在PHP中，可以使用函数substr来获取字符串的某一部分。该函数的语法如下所示。

string substr(string $str, int $start, int $length);

函数substr将字符串$str中，从位置$start的字符开始，截取长度为$length的字符串，然后返回该字符串。需要注意的是，字符串的第1个字符的位置是0，第2个字符的位置是1，以此类推。

5.6 删除字符串末尾部分的空白字符

PHP程序有时需要去掉字符串中的空白字符，比如用户从网页提交数据时，在文本框输入内容后，多敲入了一些空格，而PHP处理程序很可能要将这些数据存入数据库，这时，不希望将这些没有实际意义的空白字符存入数据库，因此要将这些空白字符处理掉。

PHP提供了一些函数，分别可以处理字符串末尾、开头的空白字符，也提供了同时处理字符串两端空白字符的函数。本节先介绍处理字符串末尾部分空白字符的内置函数。在PHP中，可以使用函数rtrim删除字符串末尾部分的空白字符。该函数的语法如下所示。

string rtrim(string $str);

5.7 删除字符串开始部分的空白字符

和函数rtrim对应的是函数ltrim，该函数将字符串开始部分的空白字符删除掉。从这个函数的命名可以看出，ltrim中的字母l代表的是left，一个字符串的left就是它的开始部分，所以，ltrim表示去掉字符串开始部分的空白字符。同理，rtrim中的首字母r表示right。这样，初学者就容易记住这两个函数及其用法了。函数ltrim的语法如下所示。

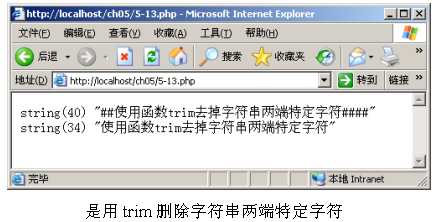
string ltrim(string $str);

该函数将字符串$str开始部分的空白字符删除，返回开始部分没有空白的字符串。

5.8 删除字符串两头的空白字符

如果希望删除字符串两头的空白字符，可以使用函数trim，该函数的语法如下所示。

string trim(string $str);

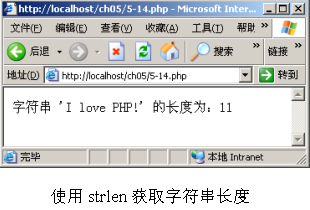


5.9 获取字符串的长度

字符串的长度一般是指，组成字符串的字符的数量。PHP提供函数strlen来获取一个字符串的长度。该函数的语法如下所示。

int strlen(string $str);

该函数返回字符串$str的长度。



5.10 对字符串做大小写转换

PHP提供两个函数分别完成对字符串中字母的大小写转换，它们是：

strtoupper($str)，将字符串$str中的字母全部转换为大写字母。

strtolower($str)，将字符串$str中的字母全部转换为小写字母。



5.11 处理含有HTML标记的字符串

使用PHP进行Web页面开发，经常会遇到处理HTML标记的问题。PHP提供的函数htmlentities完成对字符串中HTML标记的处理，该函数可将字符串中有关字符转换成HTML实体，即HTML字符编码。该函数的语法如下所示。

string htmlentities(string $str);

函数htmlentities将字符串$str中的HTML标记（如<、>等）转换成相关编码的形式。

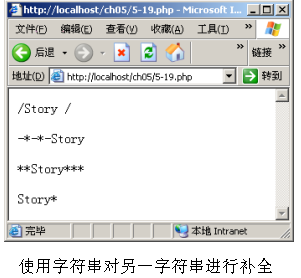


5.12 多次生成一个字符串

函数str\_repeat()可以用来重复生成一个字符串，该函数的语法如下所示。

string str\_repeat (string $input, int $num)

函数str\_repeat()有两个参数，第1个参数$input是要重复生成的字符串，第2个参数$num是要重复的次数。参数$num的取值必须大于等于0，如果$num设置为0，该函数会返回一个空字符串。



5.13 将字符串分析到变量

这小节所说的分析字符串，是指分析URL中的查询字符串。例如一个URL是：http://www.somesite.com/index.php?i=000&key=1a2b3c&cnt=10，那么该URL的查询字符串就是i=000&key=1a2b3c&cnt=10。PHP提供函数parse\_str()可以分析类似的字符，并将分析结果赋值给变量。函数parse\_str()的语法如下所示。

void parse\_str(string $str [, array $arr])

本函数可将URL的查询字符串参数$str解析，返回的变量名及值就和查询字符串中的名称及值相对应。该函数第2个参数一个指定的数组，是可选的。如果指定这个数组参数，那么函数parse\_str()将分析字符串得到变量作为数组索引、值作为数组元素存入该数组。

<?php

$arr = **array**();

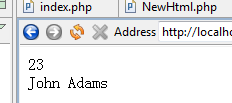
parse\_str("id=23&name=John%20Adams",$arr);

**echo** $arr['id']."<br />";

**echo** $arr['name'];

?>

效果如下：



<?php

$str = "first=php&second[]=string+functions&second[]=useing";

**echo** '原字符串：';

**echo** '<br/>';

**echo** $str;

parse\_str($str);

**echo** '<br/>';

**echo** '<br/>';

**echo** '分析结果：';

**echo** '<br/>';

**echo** 'first = '.$first;

**echo** '<br/>';

**echo** 'second[0] = '.$second[0];

**echo** '<br/>';

**echo** 'second[1] = '.$second[1];

parse\_str($str,$input);

**echo** '<br/>';

**echo** '<br/>';

**echo** '指定输出数组参数的结果：';

**echo** '<br/>';

**echo** "input['first'] = ".$input['first'];

**echo** '<br/>';

**echo** "input['second'][0] = ".$input['second'][0];

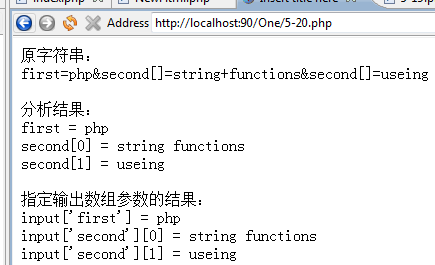
**echo** '<br/>';

**echo** "input['second'][1] = ".$input['second'][1];

**echo** '<br/>';

?>

效果如下：



5.14 转换字符串到数组

PHP提供的函数str\_split()，可将字符串转换成一个数组，即将数组分割成几个部分，每个部分的值作为数组的元素。该函数的语法如下所示。

array str\_split(string $str [, int $split\_length])

该函数将字符串$str做分割，将分割后的字符串各部分存入数组，并将数组返回。可选参数$split\_length用来指定分割字符串长度。如果没有指定该参数，函数将把字符串$str按一个字符的长度分割。

<?php

$str = 'How are you?';

**echo** '原字符串：';

**echo** '<br/>';

**echo** $str;

**echo** '<br/>';

**echo** '<br/>';

$arr1 = str\_split($str);

$arr2 = str\_split($str, 3);

**echo** '<pre>';

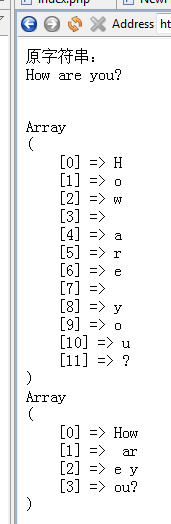
print\_r($arr1);

print\_r($arr2);

**echo** '</pre>';

?>

效果如下：



5.15 计算字符串的Hash（哈希）

所谓的哈希（Hash）也叫散列，它就是指把任意长度的输入（又叫做预映射），通过散列算法，变换成固定长度的输出，该输出就是散列值。

Hash主要用于信息安全领域的加密算法，它把一些不同长度的信息转化成杂乱的128位的编码，这个编码,叫做Hash值。也可以说，Hash就是找到一种数据内容和数据存放地址之间的映射关系。常见的Hash算法有MD5，SHA1等。

PHP中提供了对应的函数，完成对字符串做MD5哈希计算的函数，也提供函数完成SHA1哈希计算的函数，这两个函数分别是md5()和sha1()，下面分别加以简要说明。函数md5()的语法如下所示。

string md5(string $str )

该函数用来计算参数字符串$str的MD5哈希，另外一个计算哈希的函数sha1()的语法如下所示。

string sha1(string $str )

<?php

$str1 = 'This is a secret';

$str2 = 'root';

**echo** '原字符串：';

**echo** '<br/>';

**echo** 'str1='.$str1;

**echo** '<br/>';

**echo** 'str2='.$str2;

**echo** '<br/>';

**echo** '<br/>';

**echo** '使用md5加密：';

**echo** '<br/>';

**echo** "md5($str1)=".md5($str1);

**echo** '<br/>';

**echo** "md5($str2)=".md5($str2);

**echo** '<br/>';

**echo** '<br/>';

**echo** '使用sha1加密：';

**echo** '<br/>';

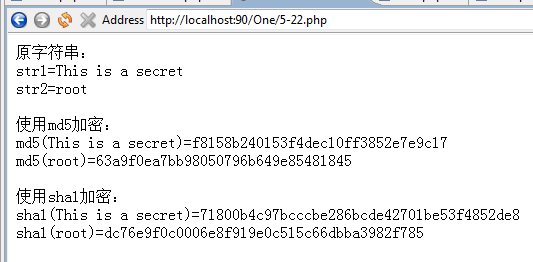
**echo** "sha1($str1)=".sha1($str1);

**echo** '<br/>';

**echo** "sha1($str2)=".sha1($str2);

?>

效果如下：



5.16 小结

本章通过介绍一些主要的字符串处理函数，来学习在PHP程序中如何完成对字符串的操作。这些函数都是比较基本的，读者务必掌握。本章讲到的函数包括：将字符串分割后存入数组的函数explode()。将数组中的元素合并成字符串的函数implode()。比较两个字符串大小的函数strcmp()。字符串替换函数str\_replace()。输出字符串的语言结构echo和print。字符串格式化函数sprintf()。

截取字符串的函数substr()。删除字符串开头和末尾空白字符的函数ltrim()、rtrim()、trim()。获取字符串长度的函数strlen()。对字符串做大小写转换的函数strtolower()和strtoupper()。处理含有HTML标记的函数htmlentities()、htmlspecialchars()和html\_entity\_decode()。重复生成字符串的函数str\_repeat()。分析字符串到变量的函数parse\_str()。转换字符串到数组的函数str\_split()。计算字符串Hash的函数md5()。和sha1()。

30.格式化字符串

Printf()函数和sprintf()函数实现的功能与echo相同，但是有返回值(成功返回true,失败返回false).

不过printf()函数和sprintf()函数还可以实现一些更复杂的格式输出，这两个函数 功能基本相同，pringf()函数是将一个格式化的字符串输出到浏览器，而sprintf()函数式返回一个格式化了的字符串。

Void printf(string format[,mixed args””])

String sprint(string formate[,mexed args””])

Printf是和c语言中一样的。

<?php

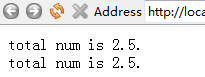
$total = 2.5;

**echo** "total num is $total."."<br>";

printf("total num is %s.",$total);

?>

效果如下：



31.改变字符串中字母的大小写

Strtoupper():将字符串转换为大小

Strtolower():将字符串转换为小写

Uefirst():如果字符串的第一个字符是字母，就转换为大写

Ucwords():将字符串中的每一个单词的第一个字母转换为大写。

32.获取和转换ASCII

String chr(int ascii):把ASCII编码转换为字符串

Int ord(char char):把字符转换为ASCII编码

32.PHP中存在的系统变量有很多，比如：

<?php

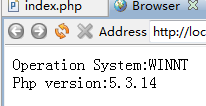
**echo** "Operation System:". PHP\_OS;

**echo** "<br>";

**echo** "Php version:" . PHP\_VERSION . "\n";

?>

效果：



33.文件包含：

33.1使用include

Include(string filename)

Include 进来的文件里面的变量和包含它的文件使用的是同一个变量

<?php

$a =100;

**echo** "\$a: $a \n";

**include** ('index.php');

**echo** "\$a: $a \n";

**echo** "\$b: $b \n" ;

?>

Index.php：

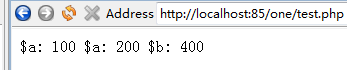
<?php

$a =200;

$b =400;

?>

效果如下：



对于被包含的文件(如index.php),我们可以使用return 语句来终止它的执行，也可以使用return语句来返回一个值.

<?php

$a =100;

**echo include** 'index.php';

?>

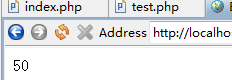
Index.php:

<?php

**return** $a/2;

?>

效果如下：



Include\_once语句和include的区别在于：如果该文件中的代码已经被包含了，则不会再次被包含，就是只能include一个文件一次。

33.2 使用require

Require的作用和include是一样的，唯一不同在于include是执行是包含，而require是先全部包含，不管会不会执行都包含，然后再执行。

如果require发现找不到文件，就会报错，而include发现找不到文件，只会警告，程序继续往下走。

34.系统函数

Rand:int rand([int min],[int max])：min和max都是可选值，返回一个在min和max自己的随机整数。

Round:float round(float val[, int precision]),对浮点数val取四舍五入，小数后的位数为precision指定，可选，默认为0.

Floor:floor(float value):舍去法取整。

Ceil:ceil(float value):进一法取整

35.全局数组：

$\_SERVER[]：包含一些诸如头信息，路径和脚本等信息的数组。

元素：

**1.  
$\_SERVER['HTTP\_ACCEPT\_LANGUAGE']//浏览器语言   
显示:zh-cn  
  
2.  
$\_SERVER['REMOTE\_ADDR'] //当前用户 IP 。   
显示:127.0.0.1  
  
3.  
$\_SERVER['REMOTE\_HOST'] //当前用户主机名   
显示:  
  
4  
$\_SERVER['REQUEST\_URI'] //URL  
显示:/test.php  
  
5  
$\_SERVER['REMOTE\_PORT'] //端口。   
显示:3864  
  
6  
$\_SERVER['SERVER\_NAME'] //服务器主机的名称。   
显示:127.0.0.1  
  
7  
$\_SERVER['PHP\_SELF']//正在执行脚本的文件名   
显示:/test.php  
  
8  
$\_SERVER['argv'] //传递给该脚本的参数。  
显示:ARRAY  
  
9   
$\_SERVER['argc'] //传递给程序的命令行参数的个数。   
显示:0  
  
10  
$\_SERVER['GATEWAY\_INTERFACE']//CGI 规范的版本。  
显示: CGI/1.1  
  
11  
$\_SERVER['SERVER\_SOFTWARE'] //服务器标识的字串   
显示:Apache/2.0.52 (Win32) PHP/5.2.1  
  
12  
$\_SERVER['SERVER\_PROTOCOL'] //请求页面时通信协议的名称和版本   
显示:HTTP/1.1  
  
13  
$\_SERVER['REQUEST\_METHOD']//访问页面时的请求方法   
显示:GET   
  
14  
$\_SERVER['QUERY\_STRING'] //查询(query)的字符串。  
显示:  
  
15   
$\_SERVER['DOCUMENT\_ROOT'] //当前运行脚本所在的文档根目录   
显示:D:/Program Files/Apache/www  
  
16  
$\_SERVER['HTTP\_ACCEPT'] //当前请求的 Accept: 头部的内容。  
显示:\*/\*  
  
17   
$\_SERVER['HTTP\_ACCEPT\_CHARSET'] //当前请求的 Accept-Charset: 头部的内容。   
显示:  
  
18  
$\_SERVER['HTTP\_ACCEPT\_ENCODING'] //当前请求的 Accept-Encoding: 头部的内容   
显示:gzip,deflate  
  
19  
$\_SERVER['HTTP\_CONNECTION'] //当前请求的 Connection: 头部的内容。例如：“Keep-Alive”。  
显示:Keep-Alive  
  
20   
$\_SERVER['HTTP\_HOST'] //当前请求的 Host: 头部的内容。   
显示:127.0.0.1:8080  
  
21  
$\_SERVER['HTTP\_REFERER'] //链接到当前页面的前一页面的 URL 地址。   
显示:http://127.0.0.1:8080/  
  
22  
$\_SERVER['HTTP\_USER\_AGENT'] //当前请求的 User\_Agent: 头部的内容。   
显示:Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.0.3705; .NET CLR 1.1.4322; .NET CLR 2.0.50727)  
  
23  
$\_SERVER['HTTPS']//如果通过https访问,则被设为一个非空的值(on)，否则返回off   
显示:  
  
24  
$\_SERVER['SCRIPT\_FILENAME'] #当前执行脚本的绝对路径名。   
显示:D:/Program Files/Apache/www/test.php  
  
25  
$\_SERVER['SERVER\_ADMIN'] #管理员信息   
显示:w@w.com  
  
26  
$\_SERVER['SERVER\_PORT'] #服务器所使用的端口   
显示:8080  
  
27  
$\_SERVER['SERVER\_SIGNATURE'] #包含服务器版本和虚拟主机名的字符串。   
显示:Apache/2.0.52 (Win32) PHP/5.2.1 Server at 10.145.40.150 Port 8080  
  
28  
$\_SERVER['PATH\_TRANSLATED'] #当前脚本所在文件系统（不是文档根目录）的基本路径。  
显示:  
  
29   
$\_SERVER['SCRIPT\_NAME'] #包含当前脚本的路径。这在页面需要指向自己时非常有用。  
显示 :/test.php  
  
30  
$\_SERVER['PHP\_AUTH\_USER'] #当 PHP 运行在 Apache 模块方式下，并且正在使用 HTTP 认证功能，这个变量便是用户输入的用户名。  
显示:  
  
31  
$\_SERVER['PHP\_AUTH\_PW'] #当 PHP 运行在 Apache 模块方式下，并且正在使用 HTTP 认证功能，这个变量便是用户输入的密码。   
显示:  
  
32  
$\_SERVER['AUTH\_TYPE'] #当 PHP 运行在 Apache 模块方式下，并且正在使用 HTTP 认证功能，这个变量便是认证的类型**

$\_ENV[]:包含环境变量的超全局数组

$\_GET[]:是一个包含所有以HTTP GET方式传递的变量的超全局数组

$\_POST[]:是一个包含所有以HTTP POST方式传递的变量的一个超全局数组。

$\_FILES[]:是一个通过HTTP POST方式传递的已经上传文件项目组成的数组。

$\_COOKIE[]:是一个通过HTTP Cookies传递的变量所组成的一个超全局数组

$\_SESSION[]:是一个当前脚本中session变量的一个超全局数组

$\_REQUEST[]是一个包含$\_GET[],$\_POST[],和$\_COOKIE[]数组中全部内容的超全局数组

36.print和echo

Print只支持一个参数，而echo支持多个参数

Print输出字符串总是返回值1，而echo输出字符串将不返回值。

正则表达式：

|  |  |
| --- | --- |
| 元字符 | 用途 |
| . | 所有单个字符 |
| | | 分隔多个模式 |
| - | 指定一个字符范围 |
| ^ | 出现在行头或在字符串开头 |
| $ | 匹配行尾货字符串尾 |
| \d | 匹配所有的一位数字 |
| \D | \d的相反，匹配所有的非数字字符 |
| \i | 不区分大小写 |
| [A-Z] | 匹配所有A到Z的任意一个 |
| [^A-Z] | 匹配所有非 A到Z的任意 |
| X? | 出现0次或1次 |
| X\* | 零个或多个 |
| X+ | 一个或多个 |
| X{n} | 匹配n个 |
| X{n,m} | 匹配n到m个 |
| \b | 代表单词的开头或结尾 |

正则表达式返回正则表达式匹配的结果。

进行正则表达式匹配函数：

1.Int preg\_match(string 正则表达式, string 需要匹配的字符串[,array 用于存储匹配结果的数组(可选)][,PREG\_OFFET\_CAPTURE标记是否也存储偏移量])。

返回的结果是返回匹配成功与否，如果成功，就返回1，否则就返回0.

<?php

$str = "PHP is the web scripting language web";

$pattern = "/web/";

$count = preg\_match($pattern, $str,$result);

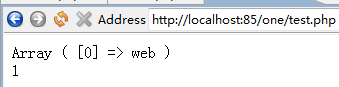
print\_r($result);

**echo** "<br>";

**echo** "$count";

?>

效果：



<?php

$str = "PHP is the web scripting language web";

$pattern = "/web/";

$count = preg\_match($pattern, $str,$result,PREG\_OFFSET\_CAPTURE);

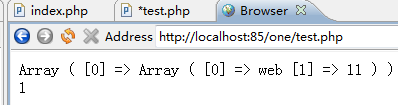
print\_r($result);

**echo** "<br>";

**echo** "$count";

?>

效果如下：



我们可以看到，结果数组多了一条数据，记录匹配成功的偏移量是11

上面那个函数是只能匹配第一个，匹配到就结束了。

2.如果是preg\_match\_all()就可以全部匹配

最后一个可选参数有三个选项：

PREG\_PATTERN\_ORDER(默认)

PREG\_SET\_ORDER

PREG\_OFFSET\_CAPTURE

3.匹配数组：preg\_grep

Array preg\_grep(string $pattern, array $input[,int $flags])

返回值是数组，用键值表示

Flag:如果是PREG\_GREP\_INVERT,则返回的是不匹配的。

<?php

$arr = **array**(".123","12.34","1234","123.4","123.","0.1234");

$pattern ="/^(\d+)?\.\d+$/";

$result = preg\_grep($pattern, $arr);

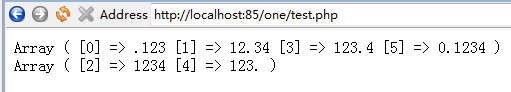
$result1 = preg\_grep($pattern, $arr,PREG\_GREP\_INVERT);

print\_r($result);

**echo** "<br>";

print\_r($result1);

?>



上面是匹配的数据，下面是不匹配的数据

4.正则表达式的搜索以及替换

Mixed preg\_replace(mixed $正则表达式，mixed $准备进行替换的字符串或正则表达式,mixed $需要匹配的字符串[,int 次数])

如果次数缺省或为-1,就表示全部替换。

返回结果为替换后的东西。

<?php

$arr = "PHP is a web language web develop";

$pattern ="/web/";

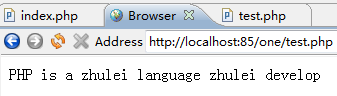
$repstr = "zhulei";

$result = preg\_replace($pattern, $repstr, $arr);

**echo** $result;

?>

效果如下：



5.用正则表达式进行字符串分割

Array preg\_split(string $正则表达式,string 字符串,[int limit],[int flags])

以正则表达式搜出来的东西进行分割，返回数组。

Flags有三个选择：PREG\_SPLIT\_ON\_EMPTY:只返回非空的成分

PREG\_SPLIT\_DELIM\_CAPTURE:定界符模式中的括号表达式也会被捕捉并返回

PREG\_SPLIT\_OFFSET\_CAPTURE:返回的同时加每个的偏移量

<?php

$str = 'PHP hypertext language programming ';

$arr1 = preg\_split("/ /", $str,-1,PREG\_SPLIT\_NO\_EMPTY);

$arr2 = preg\_split("/ /", $str,-1,PREG\_SPLIT\_DELIM\_CAPTURE);

$arr3 = preg\_split("/ /", $str,-1,PREG\_SPLIT\_OFFSET\_CAPTURE);

print\_r($arr1);

**echo** "<br>";

print\_r($arr2);

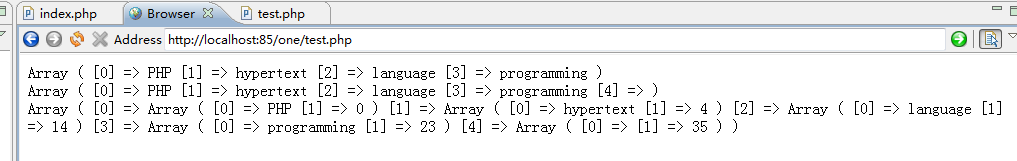
**echo** "<br>";

print\_r($arr3);

**echo** "<br>";

?>

效果如下：



面向对象的PHP

1.类定义

class className

{

//类成员列表

//类方法列表function

}

定义类的属性，首先要定义它的访问权限public/private/protect

2.在执行对象时自动定义一个$this的特殊变量，该变量表示对对象本身的引用，可以通过

$this ->的方式来引用对象的所有属性和方法。

3.构造函数

function \_\_construct()

{

}

析构函数时构造函数的相反，在对象引用结束时，自动会调用析构函数。

4.类的继承：

类的继承关键字为：extends

在PHP中，使用parent表示父类，使用self表示当前类

使用parent 和self进行属性和方法的调用时，需要使用范围解析操作符：::

5.子类和父类的属性和方法调用：

方法的调用：

$this->方法名();如果子类中有该方法则调用的是子类中的方法，若没有则是调用父类中的。

parent::则始终调用的是父类中的方法。

变量的调用：

$this->变量名；如果子类中有该变量则调用的是子类中的，若没有则调用的是父类中的

注意：子类无法访问父类的私有属性，那为什么访问却没有出错呢？其实是在子类中新建了一个变量而已，因为PHP是弱定义的语言。

<?php

**class** Animal

{

**public** $name = "Animal name";

**public** $age =18;

**public function** cry()

{

**echo** "Animal can cry";

**echo** "<br>";

}

}

**class** cat **extends** Animal

{

**public** $name;

**public** $color;

**function** \_\_construct($selfname,$color,$parentname,$age)

{

$this->name = $selfname;

$this->color =$color;

$this->age=$age;

}

**public function** cry()

{

**echo** "cat can cry";

**echo** "<br>";

}

**public function** printout()

{

**self**::cry();

**parent**::cry();

}

}

$cat = **new** cat("cat name","red","animalname",19);

$cat->printout();

**echo** "this.\$name is $cat->name";

**echo** "<br>";

**echo** "this.\$age is $cat->age";

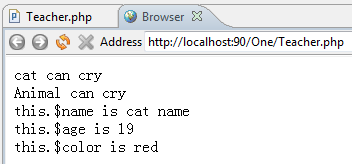
**echo** "<br>";

**echo** "this.\$color is $cat->color";

**echo** "<br>";

?>

效果如下：



6.类的常量

const 常量名 = value;

可以在类中通过self::常量名来调用。

类的常量属于类本身，不能被继承和访问。

7.访问权限

public:可以被继承，也可以通过实例访问

protect:可以被继承，不能通过实例访问

private:不能被继承，也不能通过实例访问

8.静态属性和方法

静态属性和类的常量一样，只属于类本身，通过static来定义。

方法也是一样的。

9.字符串转换函数：\_\_toString()函数

这个函数和构造函数一样，都是系统内置的函数。

如果我们直接用print 或echo 来打印对象实例名(注意是对象实例名)，就会自动调用这个函数。

<?php

**class** Animal

{

**public** $name = "Animal name";

**public** $age =18;

**function** \_\_toString()

{

**echo** "you are printing the class";

**echo** "<br>";

**return** $this->name;

}

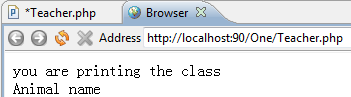
}

$animal = **new** Animal();

**echo** $animal;

?>

效果如下：



10.内置克隆方法\_\_clone()

当我们进行对对象实例进行克隆，克隆出一个新的对象实例出来时，就会自动调用类中的这个克隆函数。

<?php

**class** Animal

{

**public** $name = "Animal name";

**public** $age =18;

**function** \_\_clone()

{

**echo** "you are wanting to clone a new clas object";

**echo** "<br>";

}

}

$animal = **new** Animal();

$coloneNewAnimal = **clone** $animal;

**echo** "name is" . $coloneNewAnimal->name;

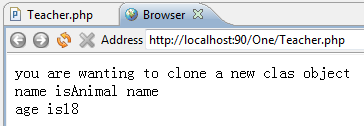
**echo** "<br>";

**echo** "age is" . $coloneNewAnimal->age;

**echo** "<br>";

?>

效果如下：



11.\_\_get()和\_\_set()

我们一般情况下，会把类中的属性设置为private，然后通过get和set函数来取得和设置属性的值，在Java中，需要对每一个属性写get和set的值，这很麻烦。

在PHP中，只要写一次，全部通用。

//\_\_get()方法用来获取私有属性

function \_\_get($property\_name)

{

if(isset($this->$property\_name))

{

return($this->$property\_name);

}else

{

return(NULL);

}

}

//\_\_set()方法用来设置私有属性

function \_\_set($property\_name, $value)

{

$this->$property\_name = $value;

}

用法如下：

<?php

**class** Person

{

//下面是人的成员属性，都是封装的私有成员

**private** $name; //人的名子

**private** $sex; //人的性别

**private** $age; //人的年龄

//\_\_get()方法用来获取私有属性

**function** \_\_get($property\_name)

{

**echo** "在直接获取私有属性值的时候，自动调用了这个\_\_get()方法<br>";

**if**(**isset**($this->$property\_name))

{

**return**($this->$property\_name);

}

**else**

{

**return**(**NULL**);

}

}

//\_\_set()方法用来设置私有属性

**function** \_\_set($property\_name, $value)

{

**echo** "在直接设置私有属性值的时候，自动调用了这个\_\_set()方法为私有属性赋值<br>";

$this->$property\_name = $value;

}

}

$p1=**new** Person();

//直接为私有属性赋值的操作，会自动调用\_\_set()方法进行赋值

$p1->name="张三";

$p1->sex="男";

$p1->age=20;

//直接获取私有属性的值，会自动调用\_\_get()方法，返回成员属性的值

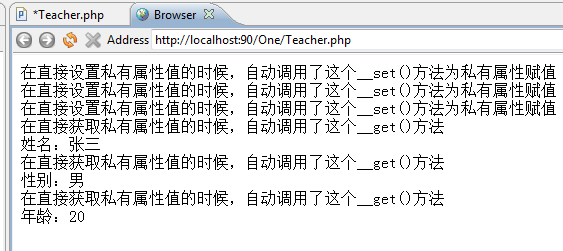
**echo** "姓名：".$p1->name."<br>";

**echo** "性别：".$p1->sex."<br>";

**echo** "年龄：".$p1->age."<br>";

?>

效果如下：



12.final关键字

如果我们希望父类中的函数或者属性不想让子类重写，就可以用final进行修饰。

文件系统与操作

文件夹(目录)操作

1.打开目录

resource opendir(string $path)，返回一个类型为resource的数据

bool is\_dir(string $path):检测路径是否有效

<?php

$path = "C:\\zhulei\\backup";

**if**(is\_dir($path))

{

$dir = opendir($path) or **die**("can not open dir".$path."\n");

**echo** $dir;

}

**else**

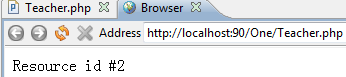
{

**echo** "$path is not a right dir";

}

?>

效果如下：



2.打开目录后，需要释放资源，可以通过closedir来关闭目录

void closedir(resource $dir\_handle);

$dir = opendir($path) or **die**("can not open dir".$path."\n");

**echo** $dir;

closedir($dir);

3.读取目录：readdir

string readdir(resource $dir\_handle)

<?php

$path = "C:\\zhulei";

**if**(is\_dir($path))

{

$dir = opendir($path) or **die**("can not open dir".$path."\n");

**while** ($file = readdir($dir))

{

**echo** $file . "<br>";

}

closedir($dir);

}

**else**

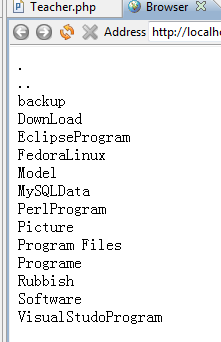
{

**echo** "$path is not a right dir";

}

?>

效果如下：



我们会发现，出来文件夹里面的内容外，还有两个特殊的目录：

. 是表示当前目录，.. 代表当前目录的上级目录

4.创建目录(文件夹)

bool mkdir(string $pathname [,int $mode])

$pathname为要创建的文件夹名，mode是新建目录的权限，使用八进制表示，以0开头，默认为0777，即最大可能的访问权限。对于window系统，model无效。

<?php

$path = "C:\\zhulei\\test";

**if**(is\_dir($path))

{

**echo** "dir already have";

}

**else**

{

**if**(mkdir($path))

{

**echo** "dir have been created successfully!";

}

**else**

{

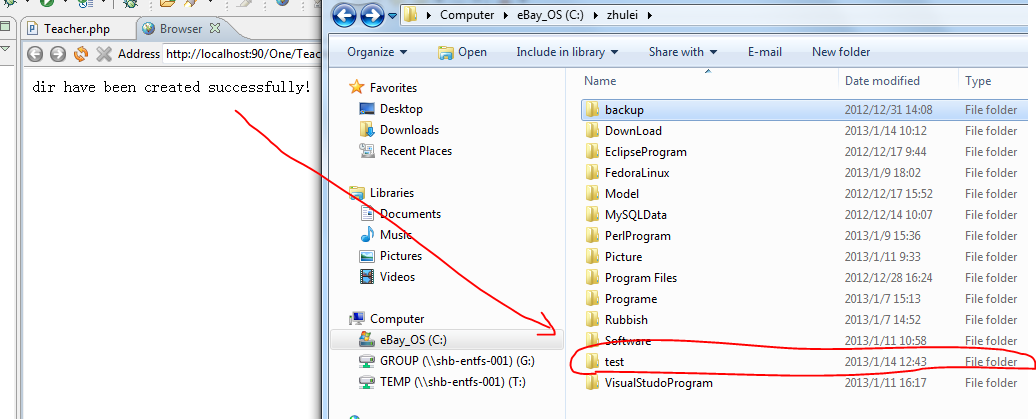
**echo** "create dir " . $path . "error";

}

}

?>

效果如下：



5.删除目录

bool rmdir(string $dirname)

<?php

$path = "C:\\zhulei\\test";

**if**(is\_dir($path))

{

**if**(rmdir($path))

{

**echo** "dir" . $path . "have been delete successfully";

}

**else**

{

**echo** "dir" . $path . "delete error!";

}

}

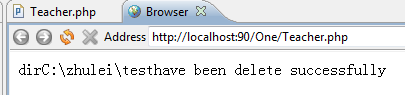
**else**

{

**echo** "dir" . $path . "do not exist!";

}

?>



6.获取当前目录：

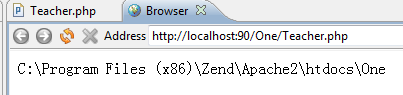
string getcwd();

<?php

**echo** getcwd();

?>

效果：



7.打开文件

resource fopen(string $filename,string mode[, bool $use\_include\_path])

mode为访问类型：

|  |  |
| --- | --- |
| Mode | 说明 |
| r | 只读方式打开，文件指针指向头 |
| r+ | 读写方式打开，文件指针指向头 |
| w | 写入方式打开，如果文件不存在，则创建新文件。文件指针指向头 |
| w+ | 读写方式打开，如果文件不存在，则创建新文件。文件指针指向头 |
| a | 写入方式打开，如果文件不存在，则创建新文件。文件指针指向尾 |
| a+ | 读写方式打开，如果文件不存在，则创建新文件。文件指针指向尾。 |
| x | 创建并以写入方式打开，如果文件已经存在，则fopen()调用失败并返回false。文件指针指向头 |
| x+ | 创建并以读写方式打开，如果文件已经存在，则fopen()调用失败并返回false。文件指针指向头 |

如果打开成功，就返回一个resource对象。

<?php

$filename = "test.txt";

**if**(is\_file($filename))

{

**if**(($fileResource = fopen($filename, 'wb')) === **false**)

{

**echo** "open file " . $filename . "fail";

}

**else**

{

**echo** "open file " . $filename . "successful";

}

}

**else**

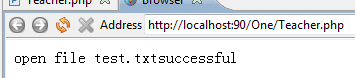
{

**echo** "file" . $filename . "do not exist";

}

?>

效果如下：



8.关闭文件

bool fclose(resource $)

9.读取文件

string fgetc(rescource $handle):读取文件的一个字符

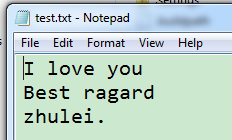
string fgets(rescource $handle[, int $length]):读取文件的一行

$length为返回的字符串的长度

string fread(rescource $handle, int $length)

string file\_get\_contents(string $filename):读取文件filename的内容到一个字符串。

test.txt文件如下：



<?php

$filename = "test.txt";

**if**(is\_file($filename))

{

**if**(($fileResource = fopen($filename, 'rb')) === **false**)//这里要用rb

{

**echo** "open file " . $filename . "fail <br>";

}

**else**

{

**echo** "open file " . $filename . "successful <br>";

**if** (($char = fgetc($fileResource))===**false**)

{

**echo** "read one char fail <br>";

}

**else**

{

**echo** "the char is: " . $char;

}

}

}

**else**

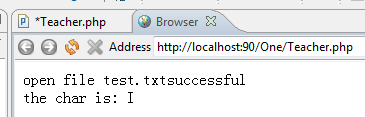
{

**echo** "file" . $filename . "do not exist<br>";

}

?>

效果如下：



**if** (($char = fgets($fileResource,50))===**false**)

{

**echo** "read one char fail <br>";

}

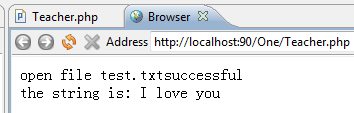
**else**

{

**echo** "the string is: " . $char;

}

效果如下：



**if** (($char = file\_get\_contents($filename))===**false**)

{

**echo** "read one char fail <br>";

}

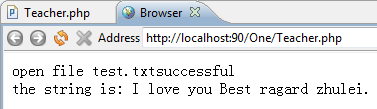
**else**

{

**echo** "the string is: " . $char;

}

效果如下：



10.写入文件

int fwrite(resource $handle, string $string [, int $length])

$string为要写入的字符串，$length可选，为要截取的长度

如果写入成功，返回写入字符串的个数，如果写入失败，返回false.

<?php

date\_default\_timezone\_set('PRC');//加这个用时间不会出警告

$filename = "test.txt";

$string = date("Y-m-j");

**if**(is\_file($filename))

{

**if**(($fileResource = fopen($filename, 'ab')) === **false**)//这里a代表在尾部添加

{

**echo** "open file " . $filename . "fail <br>";

}

**else**

{

**echo** "open file " . $filename . "successful <br>";

**if**(($count = fwrite($fileResource, $string)) === **false**)

{

**echo** "write string fail";

}

**else**

{

**echo** "have successful write char num: " . $count;

}

}

}

**else**

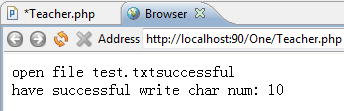
{

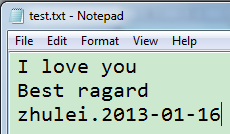
**echo** "file" . $filename . "do not exist<br>";

}

?>

效果如下：





如果上面我们打开的方式是wb，那么就在头部插入了。

11.删除文件

bool unlink (string $filename)

12.复制文件

bool copy(string $source, string $dest)

$source和$dest都包括路径及文件名称。

13.文件的移动和重命名

bool rename(string $oldname, string $newname)

如果$oldname和$newname路径相同，就是重命名，如果路径不同，就是移动

14.文件上传

在PHP系统中，文件的上传是通过HTML表单中的file控件将文件上传至php.ini文件的upload\_tmp\_dir选项所指定的临时目录，然后由PHP的函数move\_uploaded\_file将上传的临时文件移动到指定的位置实现的。

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"*>

<title>HTML Upload File</title>

</head>

<body>

<form action=*"upload.php"* method=*"post"* enctype = *"multipart/form-data"*>

<input type=*"hidden"* name=*"MAX\_FILE\_SIZE"* value=*"30000"*/>

choose the file need be uploaded:<input type=*"file"* id=*"upfile"* name=*"upfile"*/><br>

<input type=*"submit"* value=*"upload"* id = *"submit"*/>

</form>

</body>

</html>

upload.php:

<?php

$dirfile ='C:\\zhulei\\Rubbish\\PHP Program\\PHPUpload\\';

**if**(!**isset**($\_FILES['upfile']))

{//如果没有提交过来，就提示没有文件，从新上传

**echo** "no file,please <a href = 'HTML.php'> upload </a> again!";

}

**else**

{

$fileImg = $\_FILES['upfile'];

**if**(!in\_array($fileImg['type'], **array**('image/jpeg','image/jpg','image/png')))

{//如果文件不是png或jpg 图片，提示必须限定这两种

**echo** "just can upload jpg and png picture,please <a href = 'HTML.php'> upload </a> again!";

}

**else**

{

**if**(!move\_uploaded\_file($fileImg['tmp\_name'], $dirfile.$fileImg['name']))

{//如果从临时文件拷贝到我们的目标路径失败，就提示错误

**echo** "move file error, please <a href = 'HTML.php'> upload </a> again!";

}

**else**

{

**echo** "upload successful!";//如果都ok,就提示上传成功

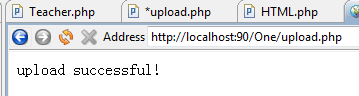
}

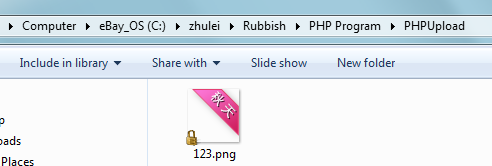
}

}

?>

效果如下：

上传的图片：



表单

1.隐藏框并不用于用户的输入，只是用于存放初始值或用于接收客户端脚本的信息，并且用户在页面看不到隐藏框的存在，在提交表单时，隐藏框中的内容一并提交到服务器中。

下面看一个隐藏框的用途：

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"*>

<title>HTML Upload File</title>

</head>

<body>

<form action=*"formaction.php"* method=*"post"* enctype = *"multipart/form-data"*>

<table>

<tr><td><input type=*"checkbox"* name=*"checkbox1"* value=*"choose one"*/></td><td>choose one</td></tr>

<tr><td><input type=*"checkbox"* name=*"checkbox2"* value=*"choose two"*/></td><td>choose two</td></tr>

<tr><td><input type=*"checkbox"* name=*"checkbox3"* value=*"choose three"*/></td><td>choose three</td></tr>

<tr><td><input type=*"checkbox"* name=*"checkbox4"* value=*"choose four"*/></td><td>choose four</td></tr>

</table>

<input type=*"hidden"* name=*"checkboxNum"* value=*"4"*/><br>//用于存储服务器信息，很好的

<input type=*"submit"* value = *"submit"* name=*"submit"*>

</form>

</body>

</html>

formaction.php代码：

<?php

$checkboxcount= $\_POST['checkboxNum'];

**echo** "you choose is: <br>";

**for**($i = 1; $i <= $checkboxcount; $i++)

{

$checkname = 'checkbox' . $i;

**if**(**isset**($\_POST[$checkname]))

{

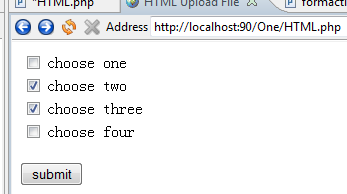
**echo** $\_POST[$checkname] . ",";

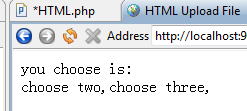
}

}

?>

效果如下：





当然了，复选框还有另一种写法，就是全部复选框同名，名字后面加[]来也行。

2.下拉框和列表框

下拉框中多加一个属性：multiple size = “5” 就是一个列表框，有5个项，否则就是一个下拉框。

<body>

<form action=*"formaction.php"* method=*"post"* enctype = *"multipart/form-data"*>

<select name=*"sex"*>

<option value=*"male"*>male</option>

<option value=*"female"*>female</option>

</select><br>

<br>

<br>

<select name=*"country"* multiple size =*"4"*>

<option value=*"china"*>CHINA</option>

<option value=*"japan"*>JAPAN</option>

<option value=*"USA"*>USA</option>

<option value=*"UK"*>UK</option>

</select><br>

</form>

</body>

效果如下：



3.cookie

当相同的计算机通过浏览器请求一个页面时，原先存储的cookie也会发送到服务器。

4.设置cookie

一般用setcookie函数：

bool setcookie(string $name[,string $value[, int $expire [, string $path [, string $domain [, bool $secure [, bool $httponly]]]]]]);

|  |  |
| --- | --- |
| 参数名称 | 参数意义 |
| name | 设置cookie的名称 |
| value | 设置cookie的值 |
| expire | cookie的实效时间，这是一个UNIX时间截，因此它是采用从UNIX经元开始的总秒数表示 |
| path | 参数path为cookie在服务器上的有效路径，如果设置为 ’/’ ,在整个网站所有页面均可以访问，如果设置为 ‘/foo/’,则只有/foo/目录及其子目录下的页面可访问，默认值为当前目录。 |
| domain | cookie 的有效域名 |
| secure | cookie是否仅允许通过安全的HTTPS协议进行传输，值为true表示只允许通过安全的HTTPS协议进行传输，false为表示允许通过普通的HTTP协议进行传输。 |
| httponly | 表示cookie是否只允许通过HTTP协议范文，这就意味着不能被脚本语言访问，如javascript |

设置cookie:

<?php

$str1 = "1miao";

$str2 = "100miao";

setcookie("onecookie",$str1,time()+1);

setcookie("two",$str2,time()+100);

?>

读取cookie:

<?php

**if**(**isset**($\_COOKIE['one']))

{

**echo** "one cookie is: ".$\_COOKIE['one'] ."<br>";

}

**else**

{

**echo** "cookie name one have expired! <br>";

}

**if**(**isset**($\_COOKIE['two']))

{

**echo** "two cookie is: ".$\_COOKIE['two'];

}

**else**

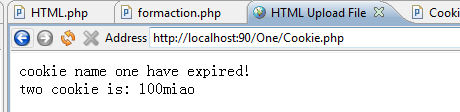
{

**echo** "cookie name two have expired!";

}

?>

效果如下：



删除cookie:

通过从新设定cookie，但不设定value的值，就删除了。还有一种方法就是从新设置，把时间设置为time()-1就删除了。

5.Session

如果需要使用Session,就必须在调用前启动Session,启动Session使用session\_start()函数

Bool session\_start(void)

注意：在调用session\_start()函数前，程序不能有任何输出，否则PHP将抛出警告。

<?php

session\_start();

$\_SESSION['username']='username';

?>

6.Session的检测与注销

Bool Isset($\_SESSION[‘username’]):session的检测

Void unset($\_SESSION[‘username’]):session的注销

注销所有的Session: bool session\_destroy(void);

Login.php:

<?php

session\_start();

**if**(**isset**($\_SESSION['logined']) && $\_SESSION['logined'])

{

header("location:welcome.php");

}

?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"*>

<title>Insert title here</title>

</head>

<body>

<form action=*"check.php"* method=*"post"*>

<table>

<tr><td>username:</td><td><input type=*"text"* name=*"username"* id=*"username"* /></td></tr>

<tr><td>password:</td><td><input type=*"password"* name=*"password"* id=*"password"* /></td></tr>

</table><br>

<input type=*"submit"* value =*"submit"*>

</form>

</body>

</html>

Check.php:

<?php

session\_start();

$username = $\_POST['username'];

$password =$\_POST['password'];

**if**($username == "zhulei" && $password == "123")

{

$\_SESSION['logined'] = **true**;

$\_SESSION['username']=$username;

header("location:welcome.php");

}

**else**

{

**echo** "username or password error, please <a href='login.php'>login</a> again!";

}

?>

Welcome.php:

<?php

**if**(**isset**($\_SESSION['logined']) && $\_SESSION['logined'])

{

**echo** "welcome user: " . $\_SESSION['username'] . " !";

}

**else**

{

**echo** "you don't login in, please <a href = 'login.php'> login</a> !";

}

?>

PHP页面跳转：

**PHP页面跳转一、header()函数**

header()函数是PHP中进行页面跳转的一种十分简单的方法。header()函数的主要功能是将HTTP协议标头(header)输出到浏览器。

header()函数的定义如下：

void header (string string [,bool replace [,int http\_response\_code]])

可选参数replace指明是替换前一条类似标头还是添加一条相同类型的标头，默认为替换。

第二个可选参数http\_response\_code强制将HTTP相应代码设为指定值。 header函数中Location类型的标头是一种特殊的header调用，常用来实现页面跳转。注意：1.location和“:”号间不能有空格，否则不会跳转。

2.在用header前不能有任何的输出。

3.header后的PHP代码还会被执行。例如，将浏览器重定向到lamp兄弟连官方论坛

1. **<** ?php
2. //重定向浏览器
3. header("Location: http://bbs.  
   lampbrother.net");
4. //确保重定向后，后续代码不会被执行
5. exit;
6. **?>**

**PHP页面跳转二、Meta标签**

Meta标签是HTML中负责提供文档元信息的标签，在PHP程序中使用该标签，也可以实现页面跳转。 若定义http-equiv为refresh,则打开该页面时将根据content规定的值在一定时间内跳转到相应页面。

若设置content="秒数;url=网址"，则定义了经过多长时间后页面跳转到指定的网址。例如，使用meta标签实现疫苗后页面自动跳转到LAMP兄弟连官方论坛。

1. **<** **meta** http-equiv="refresh"
2. content="1;url=http://  
   bbs.lampbrother.net"**>**

例如，以下程序meta.php实现在该页面中停留一秒后页面自动跳转到bbs.lampbrother.net。

1. **<** ?php
2. $url = "http://bbs.lampbrother.net"; **?>**
3. **<** **html>**
4. **<** **head>**
5. **<** **meta** http-equiv="refresh" content="1;
6. url=**<** ?php echo $url; **?>**"**>**
7. **<** /head**>**
8. **<** **body>**
9. 页面只停留一秒……
10. **<** /body**>**
11. **<** /html**>**

**PHP页面跳转三、JavaScript**

例如，此代码可以放在程序中的任何合法位置。

1. **<** ?php
2. $url = "http://bbs.lampbrother.net";
3. echo "**<** **script** language='javascript'
4. type='text/javascript'**>**";
5. echo "window.location.href='$url'";
6. echo "**<** /script**>**";
7. **?>**

AJAX

1.Ajax的核心是XMLHttpRequest

2.创建XMLHttpRequest对象

在IE中创建：

Xmlhttp\_request = new ActiveXObject(“Mscml2.XMLHTTP.3.0”);//IE 3.5、IE4.0、IE5.0

Xmlhttp\_request = new ActiveXObject(“Msxml2.XMLHTTP”);//IE6.0

Xmlhttp\_request = new ActiveXObject(“Microsoft.XMLHTTP”);//IE7.0及以上

在非IE中创建

Xmlhttp\_request = new XMLHTTPRequest();

3.AJAX的核心

<script type=*"text/javascript"*>

**var** xmlHttp = **null**;

**try**

{

xmlHttp=**new** XMLHttpRequest(); // Firefox, Opera 8.0+, Safari

}

**catch** (e)

{

**try**

{ // Internet Explorer

xmlHttp=**new** ActiveXObject('Msxml2.XMLHTTP');//IE6

}

**catch** (e)

{

**try**

{

xmlHttp=**new** ActiveXObject('Microsoft.XMLHTTP'); //IE7

}

**catch** (e)

{

alert('Your browser does not support AJAX!');

**return false**;

}

}

}

xmlHttp.onreadystatechange=**function**()

{

**if**(xmlHttp.readyState == 4 && xmlHttp.status == 200)

{

**var** someDiv = document.getElementById('someDiv');

someDiv.innerHTML=xmlHttp.responseText;

}

}

xmlHttp.open('GET','www/index.php?param1='+var1+'m2='+var2,**true**);

xmlHttp.send(**null**);

}

}

</script>

a. Xmlteep\_request.open(send\_method, url, flag)

Send\_method:post或get

<url:要提交的路径>

flag:如果是false，为同步模式，如果为true，为异步模式。

b. xmlhttp\_request.send(null);用于发送数据

要通过a和b的操作才行，可以看上面。

4.处理服务器的响应

我们上面也有，就是定义服务器返回的状态变化时要做什么：

Xmlhttp\_request.onreadystatechange=function(){};

Readstate属性取值：

0：未初始化

1：正在装载

2：装载完毕

3：交互中

4：完成

对于服务器响应的数据，可以使用以下两种方式得到：

a.以文本字符串的方式返回服务器响应

b.以XMLDocument对象方式返回响应

5.使用POST方式提交

如果使用POST方式提交，那么用下面两个不够

xmlHttp.open('POST','www/index.php?param1='+var1+'m2='+var2,**true**);

xmlHttp.send(**null**);

中间还需要加一个头信息。

xmlhttp.setRequestHeader(“Context-Type”,”applicatioin/x-www-form-urlencoded;”);

HTML：

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"*>

<title>HTML Upload File</title>

<script type=*"text/javascript"*>

**var** xmlHttp = **null**;

**try**

{

xmlHttp=**new** XMLHttpRequest(); // Firefox, Opera 8.0+, Safari

}

**catch** (e)

{

**try**

{ // Internet Explorer

xmlHttp=**new** ActiveXObject('Msxml2.XMLHTTP');//IE6

}

**catch** (e)

{

**try**

{

xmlHttp=**new** ActiveXObject('Microsoft.XMLHTTP'); //IE7

}

**catch** (e)

{

alert('Your browser does not support AJAX!');

**return false**;

}

}

}

**function** addComment()

{

url = "formaction.php";

**var** param = "username=" +document.getElementById("username").value + "&email="+document.getElementById("email").value+"&advice="+document.getElementById("advice").value;

xmlHttp.onreadystatechange=**function**()

{

**if**(xmlHttp.readyState == 4 && xmlHttp.status == 200)

{

document.getElementById("username").value="";

document.getElementById("email").value="";

document.getElementById("advice").value="" ;

alert("successful");

}

}

xmlHttp.open('POST',url,**true**);

xmlHttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

xmlHttp.send(param);

}

</script>

</head>

<body>

<form action=*""* name=*"from1"* id=*"form1"*>

<table>

<caption>give advice</caption>

<tr><td>name:<td><td><input type=*"text"* name=*"username"* id=*"username"*></td></tr>

<tr><td>Email:</td><td><input type=*"text"* name=*"email"* id=*"email"*></td></tr>

<tr><td>advice:</td><td><textarea rows=*"5"* cols=*"5"* name=*"advice"* id=*"advice"*></textarea> </td></tr>

</table><br>

<input type=*"submit"* value=*"submit"* onclick ="addComment();"/>

</form>

</body>

</html>

formaction.php文件：

<?php

$name = $\_POST['usename'];

$email = $\_POST['email'];

$advice = $\_POST['advice'];

**echo** "name is: ". $name ."<br>";

**echo** "email is: ". $email ."<br>";

**echo** "advice is: ". $advice ."<br>";

?>

图像处理

1.在PHP中要能够使用GD库进行图像处理，需要在PHP中将GD库激活。

激活办法：

打开php.ini

找到语句：

; extension = php\_gd.dll

把前面的分号去掉，保存，重启服务器。

验证有没有成功：

<?php

**if**(function\_exists("gd\_info"))

{

**echo** "GD dll have";

}

**else**

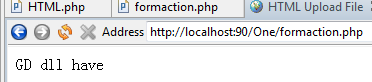
{

**echo** "error to load gd dll";

}

?>

效果如下：



<?php

**if**(function\_exists("gd\_info"))

{

$gd\_info = gd\_info();

**echo** "your GD dll Version: " . $gd\_info['GD Version'] . "<br>";

}

**else**

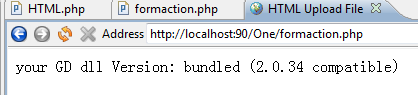
{

**echo** "error to load gd dll";

}

?>

效果如下：



2.使用GD库创建图像，首先需要创建画布，创建一个真彩色的图像可以使用imagecreateturecolor函数，格式如下：

resource imagecreateturecolor(int $x\_size, int $y\_size)

创建真彩图像：

<?php

header("Content-type: image/png");

$im= imagecreatetruecolor(200, 100) or **die**("cannot initialize new GD image stream");

?>

我们还可以根据现有图片文件创建图像:

resource imagecreatefrompng(string $filename);

resource imagecreatefromjpeg(string $filename);

resource imagecreatefromgif(string $filename);

resource imagecreatefrombmp(string $filename);

我们所创建的都是一种图像资源，还需要输出到浏览器才能被浏览器解析，输出函数有：

bool imagepng(resource $image[, string $filename])

上面是以png文件格式输出至浏览器，$image是图像资源，如果给出了$filename，则函数将GD图像以PNG格式输出至该文件，如果没有给粗$filename,则函数以PNG格式输出至浏览器。

<?php

header("Content-type: image/png");

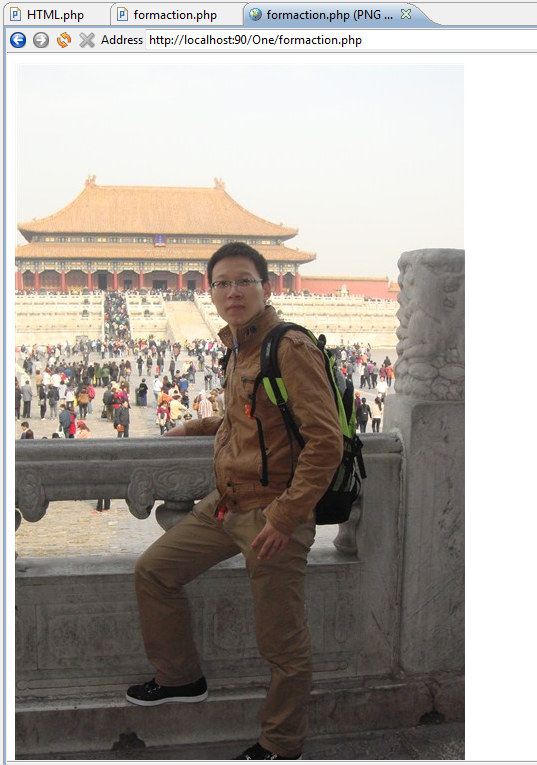
$filepath = "C:\\zhulei\\Picture\\zhulei.jpg";//图像路径

$img= imagecreatefromjpeg($filepath);//创建图像资源

imagepng($img);//输出图像

?>

效果如下：



3.设置图像的颜色：

int imagecolorallocate(resource $image, int $red, int $green, int $blue)

<?php

header("Content-type: image/png");

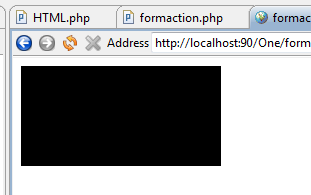
$im= imagecreatetruecolor(200, 100) or **die**("cannot initialize new GD image stream");

imagecolorallocate($im, 255, 255, 255);

imagepng($im);

?>

效果如下：



4.释放图像资源：

bool imagedestroy(resource $image)

5.绘制点线

首先要设置所画线的风格：bool imagesetstyle(resource $image, array $style)

设置完后，就可以用imageline绘制线段了。

bool imageline(resource $image, int $x1, int $x2,int $y1, int $y2,int $color)

x1和y1为线的起点，x2和y2为线的终点

<?php

header("Content-type: image/png");

$im= imagecreatetruecolor(200, 100) or **die**("cannot initialize new GD image stream");

$bgcolor = imagecolorallocate($im, 255, 255, 255);

$ftcolor = imagecolorallocate($im, 255, 0, 0);

$style = **array**($ftcolor,$bgcolor);

imagesetstyle($im, $style);

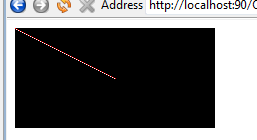
imageline($im, 0, 0, 100, 50, IMG\_COLOR\_STYLED);//画线

imagepng($im);//输出图像

imagedestroy($im);//销毁图像资源

?>

效果如下：



6.绘制几何图像：

bool imagefilledpolygon(resource $image, array $points, int $num\_points,int $color)

points为一个按顺序包含有多边形各个顶点的x和y坐标的数组。

num\_points为顶点的总数，必须大于3

color为填充的颜色。

<?php

header("Content-type: image/png");

$im= imagecreatetruecolor(300, 300) or **die**("cannot initialize new GD image stream");

$bgcolor = imagecolorallocate($im, 255, 255, 255);

$ftcolor = imagecolorallocate($im, 255, 0, 0);

$pointArr = **array**(40,50, 20,240, 60,60, 240,20, 50,40, 10,10);

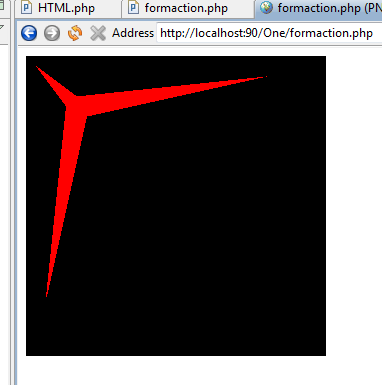
imagefilledpolygon($im, $pointArr, 6, $ftcolor);

imagepng($im);

imagedestroy($im);

?>

效果如下：



7.输出文字

bool imagestring (resource $image, int $font, int $x,int $y,string $s, int $col)

如果$font为1,2,3,4或5,则使用内置字体。

<?php

header("Content-type: image/png");

$im= imagecreatetruecolor(150, 150) or **die**("cannot initialize new GD image stream");

$bg = imagecolorallocate($im, 255, 0, 0);

$textcolor = imagecolorallocate($im, 0, 0, 255);

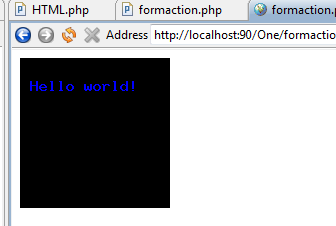
imagestring($im, 5, 10, 20, "Hello world!", $textcolor);

imagepng($im);

imagedestroy($im);

?>

效果如下：



8.制作验证码图片：

<?php

header("Content-type: image/png");

session\_start();

$authnum = '';

$str = 'abcdefghigklmnopqrstuvwxyz1234567890';

$strLength = strlen($str);

**for**($i=0;$i<=4;$i++)

{

$num = rand(0, $strLength -1);

$authnum .= $str[$num];

}

$\_SESSION['authnum'] = $authnum;

srand((double)microtime()\*1000000);

$im = imagecreate(200, 100);

$gary = imagecolorallocate($im, 200, 200, 100);

$white = imagecolorallocate($im, 255, 255, 255);

imagefill($im, 10, 5, $gary);

$li = imagecolorallocate($im, 150, 150, 150);

**for**($i =0;$i<3;$i++)//绘制三条干扰线

{

imageline($im, rand(0, 20), rand(0, 100), rand(180, 200), rand(0, 100), $li);

//起点的x尽量靠左边，y任意，终点尽量靠右边，y任意

}

**for**($i=0;$i<strlen($\_SESSION['authnum']);$i++)//绘制四个字符

{

$strcolor= imagecolorallocate($im, mt\_rand(0, 100), mt\_rand(50, 150), mt\_rand(100, 200));

$fontsize = mt\_rand(10, 20);

$x = mt\_rand(1,20)+200\*$i/4;

$y = mt\_rand(1, 50);

imagestring($im, $fontsize, $x, $y, $\_SESSION['authnum'][$i], $strcolor);

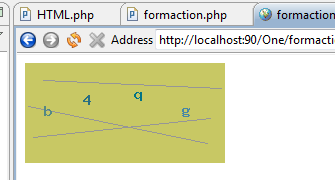
}

imagepng($im);

imagedestroy($im);

?>

效果如下：

完整程序，直接可以使用。

完整的HTML验证：

HTML文件：

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"*>

<title>HTML Upload File</title>

<script type=*"text/javascript"*>

**function** RefreshImage(id)

{

document.getElementById(id).src ='formaction.php?'+Math.random(1);

//这里必须加入随机数，否则地址相同不会重新加载！

}

</script>

<body>

<form action=*"check.php"* method=*"post"*>

<table>

<tr><td>

<img id=*"checkcode"* name=*"checkcode"* src=*"formaction.php"* onclick="RefreshImage('checkcode');">//这里也可以直接点击图片变化

</td><td><input type=*"button"* value = *"can't see very clear, change!"* onclick="RefreshImage('checkcode');" /> </td></tr>

<tr><td>checkcode:</td><td><input type=*"text"* id=*"checkcodetext"* name=*"checkcodetext"*></td></tr>

</table><br>

<input type=*"submit"* value = *"submit"*>

</form>

</body>

</html>

check.php:

<?php

session\_start();

$chkcode = strtolower($\_POST['checkcodetext']);

**echo** "session code: " .$\_SESSION['authnum'] ."<br>";

**echo** "your input code: " .$chkcode ."<br>";

**if**($chkcode == $\_SESSION['authnum'])

{

**echo** "right, you can go <a href = 'HTML.php'>back<a> to test again!";

}

**else**

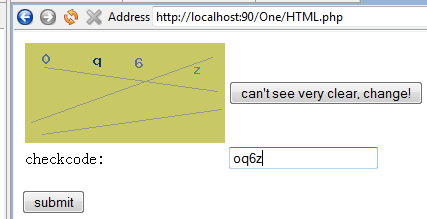
{

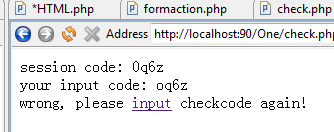
**echo** "wrong, please <a href = 'HTML.php'>input<a> checkcode again!";

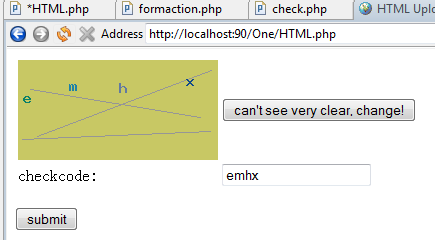
}

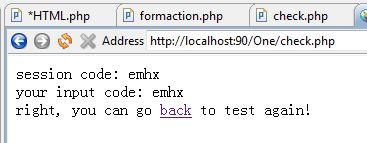
?>

效果如下：









9.生成缩略图：

HTML.php:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=GBK"*>

<title>HTML Upload File</title>

<body>

<form action=*"formaction.php"* method=*"post"* enctype=*"multipart/form-data"*>

<input type =*"hidden"* name=*"MAX\_FILE\_SIZE"* value=*"20000"*/>

choose the picture:<input type=*"file"* id=*"file"* name=*"file"* />

<br>

<input type=*"submit"* value =*"submit"*>

</form>

</body>

</html>

formaction.php:

<?php

$imgfile = $\_FILES['file'];//获取Image对象

$src\_file = $imgfile['name'];//获取image的完整路径

$imginfo = getimagesize("C:\\zhulei\\Picture\\" .$src\_file);//获取文件信息

$srcimagewidth = $imginfo[0];//获取图片宽度

$srcimageheight = $imginfo[1];//获取图片高度

$radio = max($srcimagewidth/100,$srcimageheight/80);//取压缩比例

$newimagewidth = $srcimagewidth/$radio;//压缩宽度

$newimageheight = $srcimageheight/$radio;//压缩高度

$dest\_file\_path = "C:\\" . $imgfile['name'];//设置保存缩略图的位置

$quantity = "80";//设置压缩质量

**if**($imginfo[2] == 2)//如果是jpeg图

{//由于发现$imgfile['name']只是得到了名字，没有路径，没办法，只能加进去看行不行。

$srcimagefile = imagecreatefromjpeg("C:\\zhulei\\Picture\\" .$src\_file);//创建原来的图片

}

**elseif** ($imginfo[2] == 3)//如果是png图

{

$srcimagefile = imagecreatefrompng($src\_file);//创建原来的图片

}

**else**//否则，就当做gif图来处理

{

$srcimagefile = imagecreatefromgif($src\_file);

}

$newimagefile = imagecreatetruecolor($newimagewidth, $newimageheight);//设置真彩画布的宽度和高度

imagecopyresampled($newimagefile, $srcimagefile, 0, 0, 0, 0, $newimagewidth, $newimageheight, $srcimagewidth, $srcimageheight);

//复制原来图到画布中

imagejpeg($newimagefile,$dest\_file\_path,$quantity);//输出缩略图，由于有路径，所以不输出到浏览器，而是输出到文件

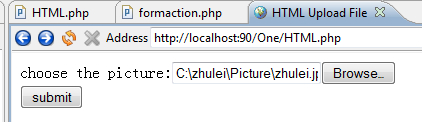
imagedestroy($srcimagefile);//销毁原图

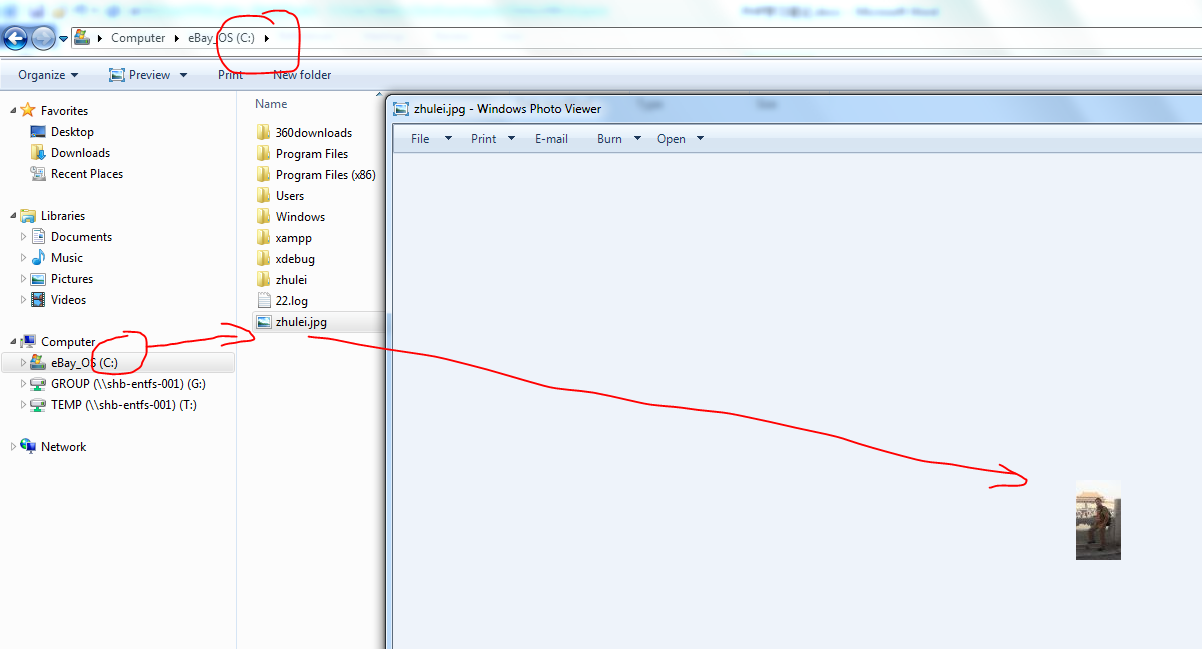
imagedestroy($newimagefile);//销毁缩略图

**echo** "OK";

?>

效果如下：





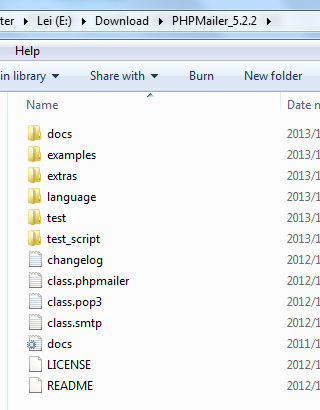
PHP发送邮件：

1.发送邮件我们用phpmailer.php这个来就会非常方便

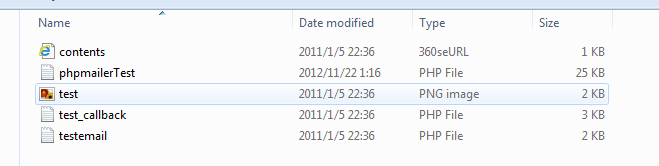
首先我们要下载这个东西：



打开文件夹：

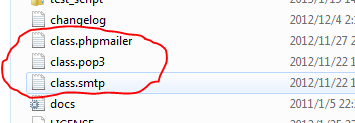


查看test文件夹：



我们可以直接用testemail.php这个，我们只需要修改一下，就可以使用了。

当然还需要把



这三个类拷贝进去

Testemail.php修改后的程序如下：

<?php

/\*\*

\* Simple example script using PHPMailer with exceptions enabled

\* **@package** phpmailer

\* **@version** $Id$

\*/

date\_default\_timezone\_set('UTC');//自己加的一句代码，如果没有，会提示时间的警告

**require** '../class.phpmailer.php';//这里路径要进行修改

**try** {

$mail = **new** PHPMailer(**true**); //New instance, with exceptions enabled

$body = file\_get\_contents('contents.html');

$body = preg\_replace('/\\\\/','', $body); //Strip backslashes

$mail->IsSMTP(); // tell the class to use SMTP

$mail->SMTPAuth = **true**; // enable SMTP authentication

$mail->Port = 25; // set the SMTP server port

$mail->Host = "mail.yourdomain.com"; // SMTP server

$mail->Username = "name@domain.com"; // SMTP server username

$mail->Password = "password"; // SMTP server password

#$mail->IsSendmail(); //这句话要注释掉

$mail->AddReplyTo("name@domain.com","First Last");

$mail->From = "name@domain.com";

$mail->FromName = "First Last";

$to = "someone@example...com";

$mail->AddAddress($to);

$mail->Subject = "First PHPMailer Message";

$mail->AltBody = "To view the message, please use an HTML compatible email viewer!"; // optional, comment out and test

$mail->WordWrap = 80; // set word wrap

$mail->MsgHTML($body);

$mail->IsHTML(**true**); // send as HTML

$mail->Send();

**echo** 'Message has been sent.';

} **catch** (phpmailerException $e) {

**echo** $e->errorMessage();

}

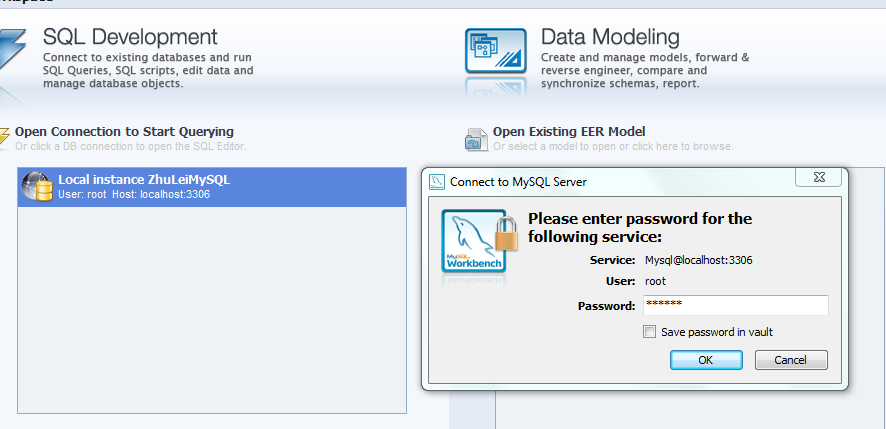
?>

经过我亲自测试，不需要做任何配置，只需要修改为自己邮箱的域名，用户名，密码就可以发送成功。

数据库操作

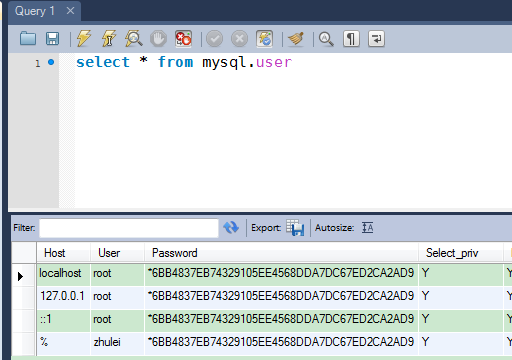
Mysql:

登陆：



以root登陆之后：

查看所有的用户：



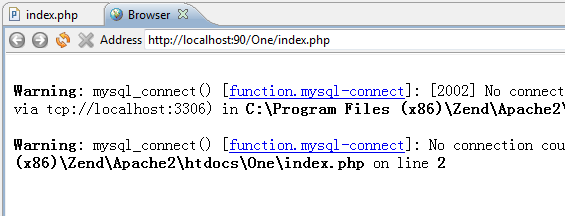
我们在操作数据库的时候，如果出现错误，会有警告，但这回泄露信息：

<?php

mysql\_connect("localhost","root","123456");

?>

效果如下：



所以我们可以这样做：

在前面加@,然后后面加 or die(“error”);

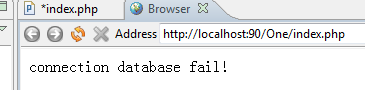
这样就只会提示error，但具体信息不会透露。

<?php

@mysql\_connect("localhost","root","123456") or **die**("connection database fail!");

?>

效果如下：



1.连接关闭mysql数据库：

<?php

mysql\_connect("localhost","zhulei","xxxxxx");

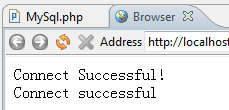
**echo** "Connect Successful!<br>";

mysql\_close();

**echo** "Connect successful";

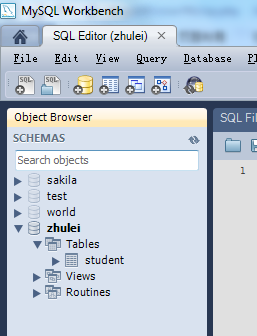
?>

效果如下：



2.选择数据库

我们连接数据成功后，需要选择到底使用哪个数据库：

我们发现，里面有4个数据库，saklia,test,world,zhulei.

其中，我们发现数据库zhulei下面有一个表名字叫做student.

假设我们选择zhulei这个数据库

mysql\_select\_db("zhulei");

**echo** "select database Successful!<br>";

3.执行sql指令

Resource mysql\_query(string $query[, resource $link\_identifier])

$link\_identifier为打开的数据库连接，如果没有这个参数，就默认使用一个已经打开的数据库连接。

这个函数只对select、show、explain和describe语句返回一个资源标示符用于存储SQL语句执行结果，如果执行不成功，返回false.

还有一个执行sql语句的函数：

Resource mysql\_db\_query(string $database,string $query[, resource $link\_identifier])

Database为要执行SQL语句的数据库。

4.分析返回的数据集

4.1获取影响的行数

Int mysql\_num\_rows(resource $result)

<?php

mysql\_connect("localhost","zhulei","123456");

**echo** "Connect Successful!<br>";

mysql\_select\_db("zhulei");

**echo** "select database Successful!<br>";

$result = mysql\_query("select \* from student");

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

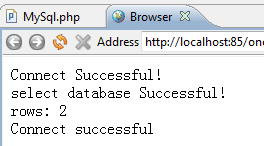
**echo** "rows: " . $num . "<br>";

mysql\_close();

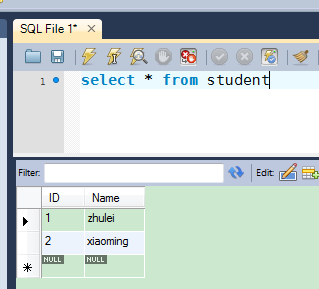
**echo** "Connect successful";

?>

效果如下：



数据库如下：



表明效果很好，没有错误。

如果是由INSERT、UPDATE和DELETE语句所执行的，需要通过mysql\_affected\_rows()函数才能获取所影响的记录数。

Int mysql\_affected\_rows(resource $link\_identifier);

4.2获取结果集中的数据

Mixed mysql\_result(resource $result ,int row[, mixed $field])

Row为指定要返回的结果集中的某一行的行号

Filed为要返回的列名或列的偏移量，若未设置改参数，默认返回第一列的值，则函数返回结果集中一个单元格的数据。

<?php

mysql\_connect("localhost","zhulei","123456");

**echo** "Connect Successful!<br>";

mysql\_select\_db("zhulei");

**echo** "select database Successful!<br>";

$result = mysql\_query("select \* from student");

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

**echo** "rows: " . $num . "<br>";

**echo** "<table border = 1><tr><td>num</td><td>name</td></tr>";

**for**($i =0;$i<$num;$i++)

{

**echo** "<tr><td>" . mysql\_result($result, $i,0) . "</td><td>" . mysql\_result($result, $i,1) . "</td></tr>";

}

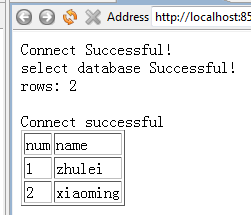
**echo** "<br>";

mysql\_close();

**echo** "Connect successful";

?>

效果如下：



4.3逐行获取结果集中的记录

Array mysql\_fetch\_row(resource $result)

返回一个数组，可以通过数组序号进行字段的读取。

比如

$arr = mysql\_fetch\_row($result);

则$arr[0],$arr[1]…..

都是可以用的。

如果我们想要不但可以通过序号，也可以通过字段名来获取，就应该用另一个函数：

Mysql\_fetch\_array(resource $result[, int $result\_type])

$result\_type有三种取值：

MYSQL\_ASSOC:以数据库中的列名作为数组键名，其结果与函数mysql\_fetch\_assor()返回结果一样。

MYSQL\_NUM:返回数组和mysql\_fetch\_row()返回结果一样。

MYSQL\_BOTH:返回数组可用序号或者列名索引都可以。

<?php

mysql\_connect("localhost","zhulei","123456");

**echo** "Connect Successful!<br>";

mysql\_select\_db("zhulei");

**echo** "select database Successful!<br>";

$result = mysql\_query("select \* from student");

**echo** "<table border = 1><tr><td>num</td><td>name</td></tr>";

**while**($line = mysql\_fetch\_array($result,MYSQL\_BOTH))

{

**echo** "<tr><td>" . $line[0] . "</td><td>" . $line['Name'] . "</td></tr>";

}

**echo** "</table>";

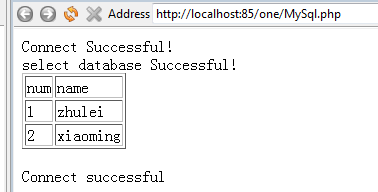
**echo** "<br>";

mysql\_close();

**echo** "Connect successful";

?>

效果如下：



注意：MySQL在delete和update时会发生错误，所以我们要再前面写这么一句话：

SET SQL\_SAFE\_UPDATES = 0;

4.4添加数据

$sql = "insert into worker values(2,'liufang','female',23,'singing')";

mysql\_query($sql);

这就是执行sql语句，没什么问题。

4.4更新数据

$updatesql="update worker set name= 'zhulei' where id =2";

mysql\_query($updatesql);

4.5删除数据

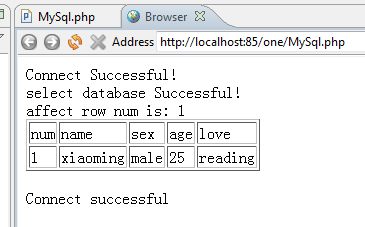
$deletesql="delete from worker where id =2";

mysql\_query($deletesql);

$affectrownum=mysql\_affected\_rows();

**echo** "affect row num is: " .$affectrownum ."<br>";

效果如下：



5.获取数据库信息

5.1获取数据库的信息

Resource mysql\_list\_dbs([resource $link\_identifier]);

<?php

mysql\_connect("localhost","zhulei","123456");

**echo** "Connect Successful!<br>";

$dbs = mysql\_list\_dbs();

**echo** "have databases: <br><br>";

**while** ($dbrow = mysql\_fetch\_array($dbs))

{

**echo** "$dbrow[0] <br>";

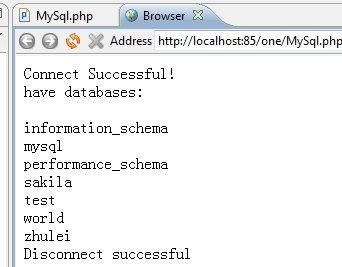
}

mysql\_close();

**echo** "Disconnect successful";

?>

效果如下：



5.2获取数据表的信息

Resource mysql\_list\_tables(string $database[, resource $link])

<?php

mysql\_connect("localhost","zhulei","123456");

**echo** "Connect Successful!<br>";

$tables = mysql\_list\_tables("zhulei");

**if**($tables)

{

**echo** "zhulei database have tables:: <br><br>";

**while** ($tablesrow = mysql\_fetch\_array($tables))

{

**echo** "$tablesrow[0] <br>";

}

}

**else**

{

**echo** "get tables information error!";

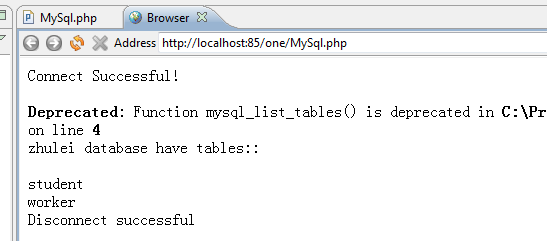
}

mysql\_close();

**echo** "Disconnect successful";

?>

效果如下：



我们发现，提示说mysql\_list\_tablse()这个函数已经过时了。

我们可以使用另一个替换的方法：

<?php

mysql\_connect("localhost","zhulei","123456");

**echo** "Connect Successful!<br>";

$rs = mysql\_query("show tables from zhulei");

**if**($rs)

{

**echo** "zhulei database have tables: <br>";

**while** ($row = mysql\_fetch\_array($rs))

{

**echo** $row[0] ."<br>";

}

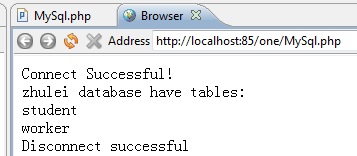
}

mysql\_close();

**echo** "Disconnect successful";

?>

效果如下：



6.获取数据表中列的信息

6.1获取列的数目

Mysql\_num\_fields(resource $result);

6.2获取行的数目

Mysql\_num\_rows(resource $result);

6.3获取列的名称：

String mysql\_field\_name(resource $result, int $field\_index)

$field\_index 为偏移量。

6.4获取列的长度

Int mysql\_field\_len(resource $result, int $field\_index)

6.5获取列的类型

String mysql\_field\_type(resource $result, int $field\_index)

<?php

mysql\_connect("localhost","zhulei","123456");

**echo** "Connect Successful!<br>";

mysql\_select\_db("zhulei");

$result = mysql\_query("select \* from worker");

$fieldnum = mysql\_num\_fields($result);

$rownum = mysql\_num\_rows($result);

**echo** "have field num is: " . $fieldnum ."<br>";

**echo** "have rows num is: " . $rownum ."<br>";

**echo** "have field name is:<br>";

**for**($i=0;$i<$fieldnum;$i++)

{

**echo** "name: " . mysql\_field\_name($result, $i) . " length:" . mysql\_field\_len($result, $i) . " type:" . mysql\_field\_type($result, $i). "<br>";

}

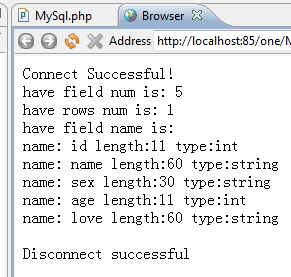
**echo** "<br>";

mysql\_close();

**echo** "Disconnect successful";

?>

效果如下：



连接操作Microsoft SQL Server数据库

1.连接数据库

要使用SQL Server数据库，首先需要确认Apache服务器安装有SQL Server数据库的mssql扩展，找到php.ini,打开，找到 ;extension=php\_mssql.dll(如果找不到这一行，就是因为是安装版的php,而不是解压版的php,我们只需要自己加这一行就行了)

去掉前面的;,然后重启服务器.

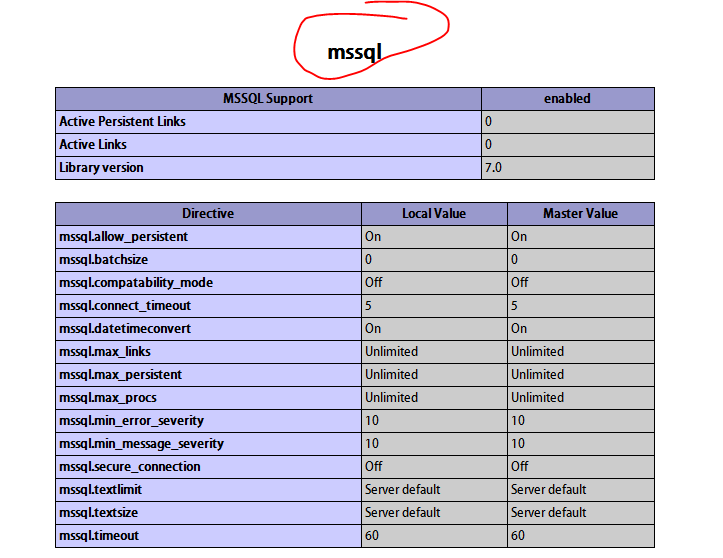
重启后，执行

<?php

Phpinfo();

?>

如果出现下面的信息，就表面可以了：



1.连接SQL Server数据库

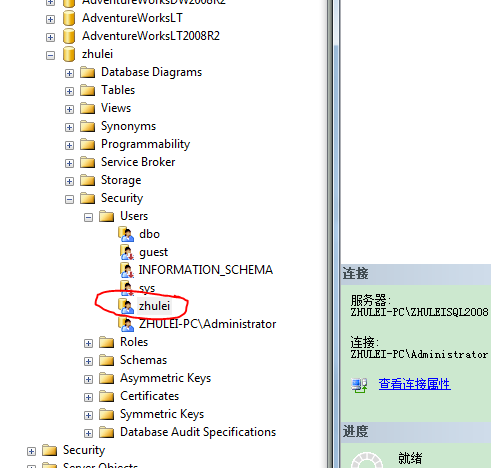
Resource Mssql\_connect([string $servername[,string $username[, $password[, bool $new\_link]]]])

$servername为要连接的SQL Server数据库的地址。

$new\_link 用于标识是否每一次使用该函数总是打开新的连接。

函数返回一个连接数据库的资源对象。





<?php

$conn = mssql\_connect("ZHULEI-PC\ZHULEISQL2008","zhulei","123456");

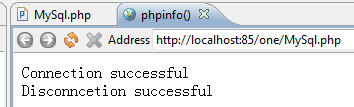
**echo** "Connection successful <br>";

mssql\_close();

**echo** "Disconncetion successful";

?>

效果如下：



<?php

$conn = mssql\_connect("ZHULEI-PC\ZHULEISQL2008","zhulei","123456");

**echo** "Connection successful <br>";

mssql\_select\_db("zhulei");

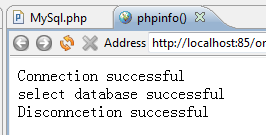
**echo** "select database successful<br>";

mssql\_close();

**echo** "Disconncetion successful";

?>

效果如下：



2.执行sql

和mysql 一样的，只是把mysql 换成mssql

PHP与XML

1.xml文件

<?xml version=”1.0” encoding=”UTF-8 ?>

以上面这句话开头

2.PHP所有对XML的操作都是通过操作SimpleXML对象进行的。

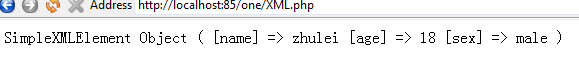
<?php

$simpleXML = simplexml\_load\_file("testxml.xml");

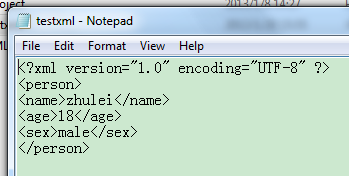
print\_r($simpleXML);

?>

效果如下：



Testxml.xml文件内容如下：



<?php

$simpleXML = <<<Eof

<?xml version="1.0" encoding="UTF-8" ?>

<person>

<name>zhulei</name>

<age>18</age>

<sex>male</sex>

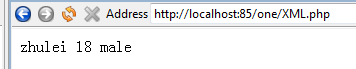
</person>

Eof;

print\_r($simpleXML);

?>

效果如下：



3.读取XML中的数据

<?php

$simpleXML = simplexml\_load\_file("testxml.xml");

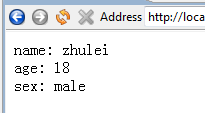
**echo** "name: " . $simpleXML->name . "<br>";

**echo** "age: " .$simpleXML->age . "<br>";

**echo** "sex: " . $simpleXML->sex . "<br>";

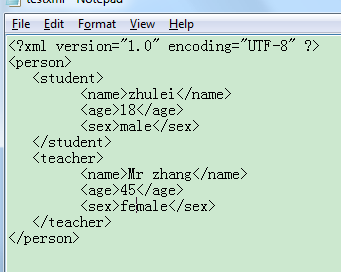
?>

效果如下：



4.读取多段数据：

Testxml.xml数据如下：



<?php

$simpleXML = simplexml\_load\_file("testxml.xml");

$student = $simpleXML->student;

**echo** "student:<br>";

**echo** "name: " . $student->name . "<br>";

**echo** "age: " .$student->age . "<br>";

**echo** "sex: " . $student->sex . "<br><br>";

$teacher = $simpleXML->teacher;

**echo** "teacher:<br>";

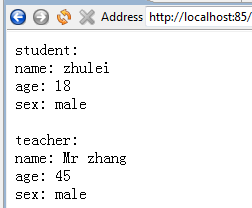
**echo** "name: " . $teacher->name . "<br>";

**echo** "age: " .$teacher->age . "<br>";

**echo** "sex: " . $student->sex . "<br><br>";

?>

效果如下：



如果数据有相同的标签，我们可以用数组来处理：

<?php

$simpleXML = simplexml\_load\_file("testxml.xml");

$student = $simpleXML->student;

**echo** "student:<br>";

**echo** "name: " . $student[0]->name . "<br>";

**echo** "age: " .$student[0]->age . "<br>";

**echo** "sex: " . $student[0]->sex . "<br><br>";

**echo** "name: " . $student[1]->name . "<br>";

**echo** "age: " .$student[1]->age . "<br>";

**echo** "sex: " . $student[1]->sex . "<br><br>";

$teacher = $simpleXML->teacher;

**echo** "teacher:<br>";

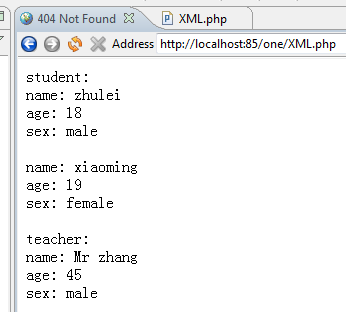
**echo** "name: " . $teacher->name . "<br>";

**echo** "age: " .$teacher->age . "<br>";

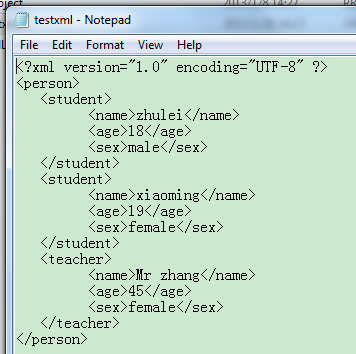
**echo** "sex: " . $student->sex . "<br><br>";

?>

效果如下;



Testxml.xml 文件数据如下：



5.修改XML数据

如果我们只修改数据，不修改源文件，那么我们只需要把获得节点数据后面加 =value 就行。

如果我们需要修改源文件，就需要保存。

<?php

$simpleXML = simplexml\_load\_file("testxml.xml");

$student = $simpleXML->student;

$student[0]->name = "feifei";

$student[1]->sex = "I don't know!";

$teacher = $simpleXML->teacher;

$teacher->age =50;

**if**($simpleXML->asXML("testxml.xml"))//回存

{

**echo** "save xml file successful";

}

**else**

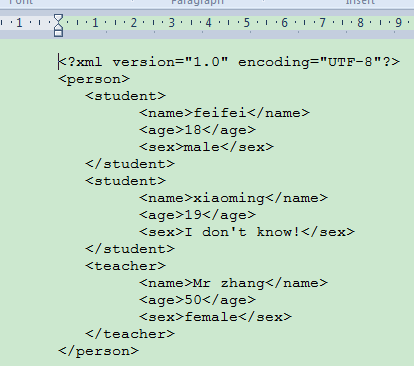
{

**echo** "Error";

}

?>

修改后效果：



6.动态创建XML

<?php

$dom = **new** DOMDocument("1.0", "UTF-8");

$Person = $dom->createElement("Person");

$dom->appendChild($Person);//将person以根节点添加到dom对象中

$Student = $dom->createElement("Student");

$Person->appendChild($Student );

$name = $dom->createElement("name","zhulei");

$id = $dom->createElement("age","18");

$sex = $dom->createElement("sex","male");

$Student->appendChild($name);

$Student->appendChild($id);

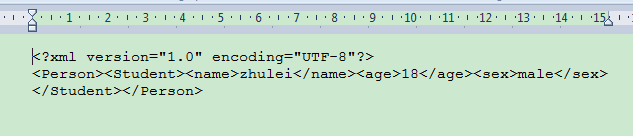
$Student->appendChild($sex);

$dom->save("NewXml.xml");

**echo** "create xml file OK";

?>

效果如下：



MVC模式

1.MVC把交互系统分解成为模型(Model)、视图(View)和控制器(Control)3个部件。