**事务，多条sql语句，要么都成功，要么都失败**

DECLARE t\_error INTEGER DEFAULT 0;

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET t\_error=1;

//开始一个事务

START TRANSACTION;

//这里放sql语句，看存储过程参数.txt

IF t\_error = 1 THEN

ROLLBACK;

ELSE

COMMIT;

END IF;

END

**存储过程.txt**

BEGIN

DECLARE t\_error INTEGER DEFAULT 0;

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET t\_error=1;

set @i=0;

SET @cnt = 1+(LENGTH(checkedproduct) - LENGTH(REPLACE(checkedproduct,',','')));

set @i=1;

START TRANSACTION;

insert into t\_order0(ordercode,userid,address,ispay) values(varordercode,varuserid,varaddress,'0');

set @orderid=(select orderid from t\_order0 where ordercode=varordercode limit 1);

WHILE @i <=@cnt DO

SET @result = REPLACE(SUBSTRING(SUBSTRING\_INDEX(checkedproduct , ',', @i),

LENGTH(SUBSTRING\_INDEX(checkedproduct , ',', @i -1)) + 1),

',', '');

SET @i = @i + 1;

SET @result1 = REPLACE(SUBSTRING(SUBSTRING\_INDEX(checkedproduct , ',', @i),

LENGTH(SUBSTRING\_INDEX(checkedproduct , ',', @i -1)) + 1),

',', '');

SET @i = @i + 1;

insert into t\_orderdetail(orderid, productcode, price, purchasenumber, productimage) values(@orderid, @result, (select price from t\_product where productcode= @result limit 1), @result1, (select productimage from t\_product where productcode= @result0 limit 1));

update t\_product set stocknumber=stocknumber-@result1 where productcode=@result;

END WHILE;

update t\_order0 set orderprice = (select sum(purchasenumber\*price) from t\_orderdetail where orderid=@orderid) where ordercode=varordercode;

IF t\_error = 1 THEN

ROLLBACK;

ELSE

COMMIT;

END IF;

END

**Foreach.php**

<?php

/\*

foreach foreach 语句用于循环遍历数组。

每进行一次循环，当前数组元素的值就会被赋值给 value 变量

（数组指针会逐一地移动） - 以此类推。

\*/

?>

<?php

$arr = array("one", "two", "three");

foreach ($arr as $value)

{

  //echo "Value: " . $value . "<br>";

  echo "Value:  $value" . "<br>";

  //echo 'Value: $value' . "<br>";

}

?>

<?php

/\*

从PHP 5开始，你可以通过在变量前面使用&操作符来修改数组的元素，

这样的话分配的是引用，而不是值。

\*/

$arr = array(1, 2, 3, 4);

foreach ($arr as &$value) {

  print $value. "<br>";

  $value = $value \* 2;

}

//reset($arr);

unset($value);

foreach ($arr as $value) {

  print $value. "<br>";

}

// $arr is now array(2, 4, 6, 8)

unset($value);  //break the reference with the last element

?>

<?php

$language=array("ASP","PHP","JSP");

//list($key,$value) 和each()一起使用是将数组当前指针所指向单元的键/值对分别赋值给变量

while((list($key,$value)=each($language))){

 echo $key."=>".$value;

 echo "<br>";

}

?>

<?php

$arr = array("Apple", "Banana", "Orange");

//reset() 函数把数组的内部指针指向第一个元素，并返回这个元素的值。 若失败，则返回 FALSE。

reset($arr);

//each()会将作用的数组的当前单元的健/值对返回，并且将数组指针向下移动一个位

$i=0;

while (list(, $value) = each($arr)) {

    echo "Fruit: $value<br />\n";

}

foreach ($arr as $value) {

  $i++;

  if ($i >2) break;

    echo "Fruit: " . $value. "<br />\n";

    echo "Fruit: $value <br />\n";

}

?>

<?php

/\*

在老版本的PHP中list()是和each()一起用来遍历数组的，

但是在现在流行PHP5中已经被foreach($array as $key=>$value)给代替，

所以list()可以说已经没有什么作用。

但是你试图将数组的前面几个元素的值赋给list()括号中所列的变量时还是有点用的

\*/

?>

<?php

//Program List：更多foreach的使用例子

$arr = array("Apple", "Banana", "Orange");

reset($arr);

while (list($key, $value) = each($arr)) {

    echo "Key: $key; Value: $value<br />\n";

}

reset($arr);

foreach ($arr as $key => $value) {

    echo "Key: $key; Value: $value<br />\n";

}

?>

<?php

/\* foreach example 1: value only \*/

$a = array(1, 2, 3, 17);

foreach ($a as $v) {

    echo "Current value of \$a: $v.\n <br>";

}

/\* foreach example 2: value (with its manual access notation printed for illustration) \*/

$a = array(1, 2, 3, 17);

$i = 0; /\* for illustrative purposes only \*/

foreach ($a as $v) {

    echo "\$a[$i] => $v.\n  <br>";

    $i++;

}

/\* foreach example 3: key and value \*/

$a = array(

    "one" => 1,

    "two" => 2,

    "three" => 3,

    "seventeen" => 17

);

foreach ($a as $k => $v) {

    echo "\$a[$k] => $v.\n";

}

/\* foreach example 4: multi-dimensional arrays \*/

$a = array();

$a[0][0] = "a";

$a[0][1] = "b";

$a[1][0] = "y";

$a[1][1] = "z";

foreach ($a as $v1) {

    foreach ($v1 as $v2) {

        echo "$v2\n";

    }

}

/\* foreach example 5: dynamic arrays \*/

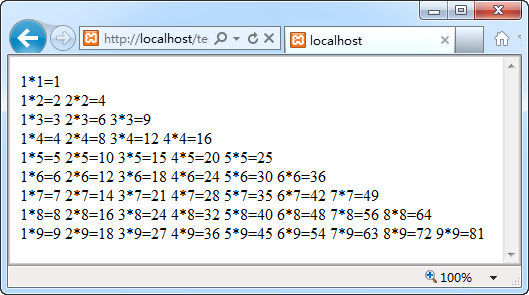
foreach (array(1, 2, 3, 4, 5) as $v) {

    echo "$v\n";

}

?>

**九九乘法表**

****

<?php

for ($i=1;$i<=9;$i++){

    for ($j=1;$j<=9;$j++){

        if ($j>$i) {

            break;

        }

        $k=$i\*$j;

        echo $i."\*".$j."=".$k." ";

        if ($j==$i) {

            echo "<br>";

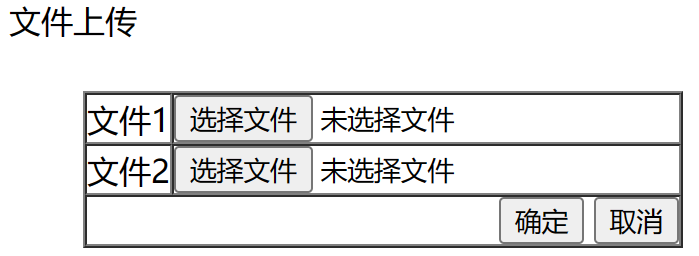
        }

    }

}

?>

Fileupload.html 文件上传



<body>

<table id="divContainer" style="HEIGHT: 100%; WIDTH: 380" border="0">

<!--

<form> 标签的 enctype 属性规定了在提交表单时要使用哪种内容类型。

在表单需要二进制数据时，比如文件内容，请使用 "multipart/form-data"。

<input> 标签的 type="file" 属性规定了应该把输入作为文件来处理。

举例来说，当在浏览器中预览时，会看到输入框旁边有一个浏览按钮。

-->

<form method="post" name="frmUpload" enctype="multipart/form-data" action="upload\_file.php">

<tr height="35"><td align="left" valign="bottom">文件上传</td></tr>

<tr>

<td align="center" valign="top">

    <table class="Transex" border="1" cellspacing="0" cellpadding="0">

        <tr>

<td nowrap><label class="FieldLabel">文件1</label></td>

<!--nowrap 属性规定表格单元格中的内容不换行-->

        <td><input type="file" class="TxtInput" tabindex="1" name="afile1"></td>

        </tr>

……

        <tr>

<td nowrap><label class="FieldLabel"> 文件2</label></td>

<td><input type="file" class="TxtInput" tabindex="1" name="afile2"></td>

  </tr>

        <tr>

        <td colspan="2" align="right">

        <!--<button name="btnOk" id="btnOk" >确定</button>-->

        <input type="submit" name="submit" value="确定" />

        <!--<button name="btnCancel" id="btnCancel" onclick="window.close();">取消</button> -->

        <button type="button" name="btnCancel" id="btnCancel" onclick="window.close();"> 取消</button>

        </td>

        <input type="hidden" name="ifupload" value=1>&nbsp;&nbsp;

        </tr>

        </table>

        </td>

  </tr>

</form>

</table>

</body>

Uploadfile.php 上传并修改存储路径

<?php

header("Cache-Control: no-cache, post-check=0, pre-check=0");

header("Content-type:text/html;charset=gb2312");

//filename:upldfile.php

if($\_POST["ifupload"]=="1") {

/\*

addslashes() 函数在指定的预定义字符前添加反斜杠。

这些预定义字符是：

单引号 (')

双引号 (")

反斜杠 (\)

NULL

\*/

//在程序中，有时我们会看到这样的路径写法，"D:\\Driver\\Lan" 也就是两个反斜杠来分隔路径。

$path=AddSlashes(dirname(\_\_FILE\_\_))  . "\\\\upload\\\\";

//echo AddSlashes(dirname(\_\_FILE\_\_)); die();

echo $path;//die();

for($i=1;$i<=2;$i++) {

$files="afile$i";

    echo $files. " Kb<br />";

//用户只能上传pdf文件，文件大小必须大于 20 kb：

//if (($\_FILES[$files]["type"] == "application/pdf")

//&& ($\_FILES[$files]["size"] > 20000))

//校验部分，没要求可以不写

    if ($\_FILES[$files]["error"] > 0)

    {

    echo "Error: " . $\_FILES[$files]["error"] . "<br />";

    }

    else

    {

    echo "Upload: " . $\_FILES[$files]["name"] . "<br />";

        echo "Type: " . $\_FILES[$files]["type"] . "<br />";

        echo "Size: " . ($\_FILES[$files]["size"] / 1024) . " Kb<br />";

        echo "Stored in: " . $\_FILES[$files]["tmp\_name"] . " Kb<br />";

     }

/\*

通过使用 PHP 的全局数组 $\_FILES，你可以从客户计算机向远程服务器上传文件。

第一个参数是表单的 input name，

第二个下标可以是 "name", "type", "size", "tmp\_name" 或 "error"。就像这样：

$\_FILES["file"]["name"] - 被上传文件的名称

$\_FILES["file"]["type"] - 被上传文件的类型

$\_FILES["file"]["size"] - 被上传文件的大小，以字节计

$\_FILES["file"]["tmp\_name"] - 存储在服务器的文件的临时副本的名称

$\_FILES["file"]["error"] - 由文件上传导致的错误代码

\*/

//下面这一行可以不写

echo $\_FILES[$files]['tmp\_name'];

//is\_uploaded\_file() 函数判断指定的文件是否是通过 HTTP POST 上传的

    if (is\_uploaded\_file($\_FILES[$files]['tmp\_name'])) {

        $filename = $\_FILES[$files]['name'];

        $localfile = $path . $filename;

        echo $localfile;

        //move\_uploaded\_file() 函数将上传的文件移动到新位置。

        move\_uploaded\_file($\_FILES[$files]['tmp\_name'], $localfile);

    }

//}

}

echo "<br><b>You have uploaded files successfully</b><br>";

exit;

}

?>