

第4章 字符串操作与正则表达式

在本章中，我们将讨论如何使用PHP的字符串函数来格式化和操作文本。我们还将介绍使用字符串函数或正则表达式来搜索（或替换）单词、短语或字符串中的其他模式。

在许多情况下，这些函数都是非常有用的。通常，你会希望整理或重新格式化将要存入到数据库中的用户输入信息。当需要创建搜索引擎（或其他东西）应用程序时，搜索函数简直棒极了。

在本章中，我们将主要介绍以下内容：

- 字符串的格式化
- 字符串的连接和分割
- 字符串的比较
- 使用字符串函数匹配和替换子字符串
- 使用正则表达式

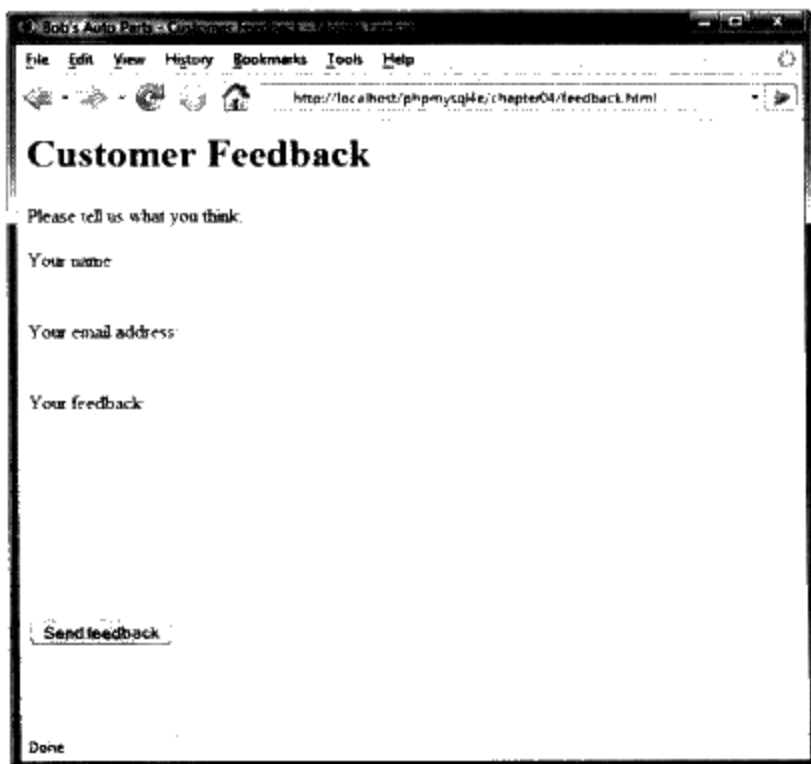
4.1 创建一个示例应用程序：智能表单邮件

在本章中，我们将介绍在一个智能表单邮件应用程序上下文中使用字符串和正则表达式函数。我们将把这些脚本添加到前面几章所介绍的Bob汽车配件站点上。

这一次，我们将为Bob的顾客建立一个直观而又实用的顾客意见反馈表单，在这个表单中，顾客可以输入他们的投诉和表扬，如图4-1所示。但是，与其他网站的类似表单相比，我们的应用程序将有很大的改善。我们不是将表单全部内容都发送到一个统一的电子邮件地址，例如feedback@example.com，而是尝试加入一些智能处理功能，例如在顾客的输入信息中查找一些关键词和短语，然后再将邮件发送到Bob公司适当的雇员那里。

例如，如果电子邮件中包含了单词“advertising（广告）”，那么这个邮件就可能将被反馈送到公司的市场部门。如果邮件是来自Bob的最大客户，那么就要把它直接发送到Bob那里。

我们将从程序清单4-1所示的简单脚本开始，然后再不断地添加新的代码。



The image shows a web browser window with the title "Bob's Auto Parts". The address bar displays "http://localhost/phpmySQL42/chapter04/feedback.html". The main content area is titled "Customer Feedback" and contains the text "Please tell us what you think:". Below this are three input fields: "Your name", "Your email address", and "Your feedback". At the bottom of the form is a button labeled "Send feedback". The browser's status bar at the bottom shows "Done".

图4-1 反馈表单要求顾客填写姓名、邮件地址和意见

程序清单4-1 processfeedback.php——邮件表单内容的基本脚本

```

<?php

//create short variable names
$name=$_POST['name'];
$email=$_POST['email'];
$feedback=$_POST['feedback'];

//set up some static information
$toaddress = "feedback@example.com";

$subject = "Feedback from web site";

$mailcontent = "Customer name: ".$name."\n".
               "Customer email: ".$email."\n".
               "Customer comments:\n".$feedback."\n";

$fromaddress = "From: webserver@example.com";

//invoke mail() function to send mail
mail($toaddress, $subject, $mailcontent, $fromaddress);

?>
<html>
<head>
<title>Bob's Auto Parts - Feedback Submitted</title>
</head>
<body>
<h1>Feedback submitted</h1>
<p>Your feedback has been sent.</p>
</body>
</html>

```

通常，应该使用一些函数（例如isset()）来检查用户是否已经填写了所有要求的表单域。为了简化代码，我们在该脚本和其他例子中都将省略这个函数。

在这个脚本中，将看到我们将表单中各个域的内容连接在一起，然后使用PHP的mail()函数将它们发送到feedback@example.com。这是一个样例电子邮件地址。如果想对本章的代码进行测试，可以在这里替换为你自己的电子邮件地址。我们还没有使用过mail()函数，所以我们将介绍这个函数是如何工作的。

顾名思义，这个函数是用来发送电子邮件的。mail()函数的原型如下所示：

```

bool mail(string to, string subject, string message,
          string ! additional_headers [, string additional_parameters]);

```

该函数的前三个参数是必需的，分别代表发送邮件的目的地址、主题行和消息内容。第四个参数可以用来发送任何额外的、有效的邮件头。有效的邮件头在RFC822文档中有说明，如

果想了解其详细信息，可以通过在线方式查看（RFC，是征求意见文件的缩写）。

它是许多互联网标准的来源——我们将在第19章中详细介绍它们）。在这里，我们通过第四个参数给邮件加了一个“From:”地址。也可以用它添加“Reply-To:”和“Cc:”域等。如果需要附加多个邮件头，只要用换行符（\n\r）在字符串中将它们分开，如下所示：

```
$additional_headers= 'From: webserver@example.com\r\n "
                    . "Reply-To: bob@example.com";
```

可选的第五个参数可以向任何经过配置用来发送电子邮件的程序传递参数。

为了使用mail()函数，必须将PHP设置为指向邮件发送程序。如果以上脚本不能在当前的表单中正常工作，安装可能是问题所在。请参阅附录A的详细介绍。

在贯穿本章的内容中，将使用PHP的字符串处理函数和正则表达式函数来改进这个基本的脚本。

4.2 字符串的格式化

通常，在使用用户输入的字符串（通常来自HTML表单界面）之前，必须对它们进行整理。在接下来的内容中，将介绍一些可用的函数。

4.2.1 字符串的整理：chop()、ltrim()和trim()

整理字符串的第一步是清理字符串中多余的空格。虽然这一步操作不是必需的，但如果要将字符串存入一个文件或数据库中，或者将它和别的字符串进行比较，这就是非常有用的。

为了实现该功能，PHP提供了3个非常有用的函数。在脚本的开始处，当我们给表单输入变量定义简短变量名称时，可以使用trim()函数来整理用户输入的数据，如下所示：

```
$name = trim($_POST['name']);
$email = trim($_POST['email']);
$feedback = trim($_POST['feedback']);
```

trim()函数可以除去字符串开始位置和结束位置的空格，并将结果字符串返回。默认情况下，除去的字符是换行符和回车符（\n和\r）、水平和垂直制表符（\t和\x0B）、字符串结束符（\0）和空格。除了这个默认的过滤字符列表外，也可以在该函数的第二个参数中提供要过滤的特殊字符。根据特定用途，可能会希望使用ltrim()函数或rtrim()函数。

这两个函数的功能都类似于trim()函数，它们都以需要处理的字符串作为输入参数，然后返回经过格式化的字符串。这三个函数的不同之处在于trim()将除去整个字符串前后的空格，而ltrim()只从字符串的开始处（左边）除去空格，rtrim()只从字符串的结束处（右边）除去空格。

4.2.2 格式化字符串以便显示

PHP具有一系列可供使用的函数来重新格式化字符串，这些函数的工作方式是各不相同的。

1. 使用HTML格式化：nl2br()函数

nl2br()函数将字符串作为输入参数，用XHTML中的
标记代替字符串中的换行

符。这对于将一个长字符串显示在浏览器中是非常有用的。例如，我们使用这个函数来格式化订单中的顾客反馈并将它返回到浏览器中：

```
<p>Your feedback (shown below) has been sent.</p>
<p><?php echo nl2br($mailcontent); ?> </p>
```

请记住，HTML将忽略纯空格，所以如果不使用nl2br()函数来过滤这个输出结果，那么它看上去就是单独的一行（除非浏览器窗口进行了强制的换行）。举例说明如图4-2所示。

2. 为打印输出而格式化字符串

到目前为止，我们已经用echo语言结构将字符串输出到浏览器。PHP也支持print()结构，它实现的功能与echo相同，但具有返回值（true或false，表示成功或失败）。

这两种方法都会打印一个字符串。使用函数printf()和sprintf()，还可以实现一些更复杂的格式。它们的工作方式基本相同，只是printf()函数是将一个格式化的字符串输出到浏览器中，而sprintf()函数是返回一个格式化了了的字符串。

如果你以前曾经使用过C语言，会发现这些函数从概念的角度和C语言中的一样，但是，其语法与C语言的函数并不是完全一致的。如果没有使用过C语言，你也会慢慢习惯并会发现它们非常有用并且功能强大。

这些函数的原型如下所示：

```
string sprintf ( string format [, mixed args...] )
void printf ( string format [, mixed args...] )
```

传递给这两个函数的第一个参数都是字符串格式，它们使用格式代码而不是变量来描述输出字符串的基本形状。其他参数是用来替换格式字符串的变量。

例如，在使用echo时，我们把要用的变量直接打印至该行中，如下所示：

```
echo "Total amount of order is $total. ";
```

要使用printf()函数得到相同的结果，应该使用如下语句：

```
printf ("Total amount of order is %s.", $total);
```

格式化字符串中的%s是转换说明。它的意思是“用一个字符串来代替”。在这个例子中，它会被已解释成字符串的\$total代替。如果保存在\$total变量中的值是12.4，这两种方法都将它打印为12.4。

printf()函数的优点在于，可以使用更有用的转换说明来指定\$total为一个浮点数，

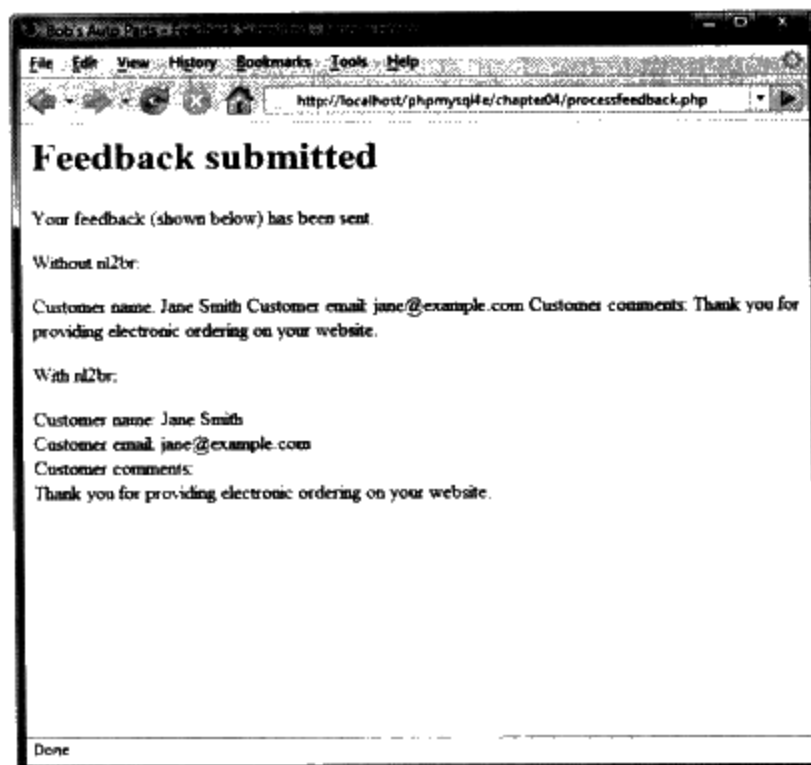


图4-2 使用PHP的nl2br()函数改进HTML中的长字符串显示

它的小数点后面应该有两位小数，如下所示：

```
printf ("Total amount of order is %.2f ", $total);
```

经过这行代码的格式化处理，存储在\$total中的12.4将打印为12.40。

可以在格式化字符串中使用多个转换说明。如果有*n*个转换说明，在格式化字符串后面就应该带有*n*个参数。每个转换说明都将按给出的顺序被一个重新格式化过的参数代替。

如下所示：

```
printf ('Total amount of order is %.2f (with shipping %.2f) ',
        $total, $total_shipping);
```

在这里，第一个转换说明将使用变量\$total，而第二个转换说明将使用变量\$total_shipping。

每一个转换说明都遵循同样的格式，如下所示：

```
%[ padding_character ][ - ][ width ][ .precision ] type
```

所有转换说明都以%开始。如果想打印一个“%”符号，必须使用“%%”。

参数padding_character是可选的。它将被用来填充变量直至所指定的宽度。该参数的作用就像使用计算器那样在数字前面加零。默认的填充字符是一个空格，如果指定了一个空格或0，就不需要使用“.”作为前缀。对于任何其他填充字符，必须指定“.”作为前缀。

字符“-”是可选的。它指明该域中的数据应该左对齐，而不是默认的右对齐。

参数width告诉printf()函数在这里为将被替换的变量留下多少空间（按字符计算）。

参数precision表示必须是以一个小数点开始。它指明了小数点后面要显示的位数。

转换说明的最后一部分是一个类型码。其支持的所有类型码如表4-1所示。

表4-1 转换说明的类型码

类 型	意 义
b	解释为整数并作为二进制数输出
c	解释为整数并作为字符输出
d	解释为整数并作为小数输出
e	解释为双精度并作为浮点数输出
o	解释为整数并作为八进制数输出
s	解释为字符串并作为字符串输出
u	解释为整数并作为非指定小数输出
x	解释为整数并作为带有小写字母a~f的十六进制数输出
X	解释为整数并作为带有大写字母A~F的十六进制数输出

当在类型转换代码中使用printf()函数时，你可以使用带序号的参数方式，这就意味着参数的顺序并不一定要与转换说明中的顺序相同。例如：

```
printf ("Total amount of order is %2\$.2f (with shipping %1\$.2f)",
        $total_shipping, $total);
```

只要直接在“%”符号后添加参数的位置，并且以\$符号为结束——在这个例子中，“2\\$”意味着“用列表中的第二个参数替换”。这个方法也可以在重复参数中使用。

这些函数还有两种可替换的版本，分别是`vprintf()`和`vsprintf()`。这些变体函数接收两个参数：格式字符串和参数数组，而不是可变数量的参数。

3. 改变字符串中的字母大小写

可以重新格式化字符串中的字母大小写。对于我们的示例应用程序而言，这个特性并不是非常有用的，但是我们可以来看一些简单的例子。

如果电子邮件中的主题行字符串是以`$subject`开始，可以通过几个函数来改变它的大小写。这些函数的功能概要如表4-2所示。该表的第一列显示了函数名，第二列描述了它的功能，第三列显示了如何在字符串`$subject`中使用它，最后一列显示了该函数的返回值。

表4-2 字符串大小写函数和它们的效果

函 数	描 述	使 用	返 回 值
<code>strtoupper()</code>	将字符串转换为大写	<code>\$subject</code> <code>strtoupper(\$subject)</code>	Feedback from web site FEEDBACK FROM WEB SITE
<code>strtolower()</code>	将字符串转换成小写	<code>strtolower(\$subject)</code>	feedback from web site
<code>ucfirst()</code>	如果字符串的第一个字符是字母，就将该字符转换为大写	<code>ucfirst(\$subject)</code>	Feedback from web site
<code>ucwords()</code>	将字符串每个单词的第一个字母转换为大写	<code>ucwords(\$subject)</code>	Feedback From Web Site

4.2.3 格式化字符串以便存储：`addslashes()`和`stripslashes()`

除了使用字符串函数来重新格式化一个可见的字符串之外，也可以使用其中的一些函数来重新格式化字符串，以便将其存入数据库。虽然在本书的第二篇之前，我们还没有介绍在数据库中执行真正的写操作，但现在，我们将介绍如何为了数据库存储而对字符串进行格式化。

对于字符串来说，某些字符肯定是有效的，但是当将数据插入到数据库中的时候可能会引起一些问题，因为数据库会将这些字符解释成控制符。这些有问题的字符就是引号（单引和双引）、反斜杠（`\`）和`NULL`字符。

我们需要找到一种标记或是转义它们的办法，以便使像MySQL这样的数据库能够理解我们表示的是有实际意义的特殊文本字符，而不是控制序列。为了将这些字符进行转义处理，可以在它们前面加一个反斜杠。例如，“（双引号）就变成`\`”（反斜杠双引号），`\`（反斜杠）就变成`\\`（反斜杠反斜杠）。（这个规则对所有特殊字符都通用，所以，如果在字符串中存在`\\`字符，就需要用`\\\\`进行替换。）

PHP提供了两个专门用于转义字符串的函数。在将任何字符串写到数据库之前，如果你的PHP的默认配置还没有启用该功能，你应该使用`addslashes()`将它们重新格式化，例如：

```
$feedback = addslashes(trim($_POST['feedback']));
```

和许多其他字符串函数一样，`addslashes()`函数需要一个字符串作为输入参数，经过该函数处理，将返回一个重新格式化后的字符串。

图4-3所示的是对一个字符串应用这些函数后的结果。

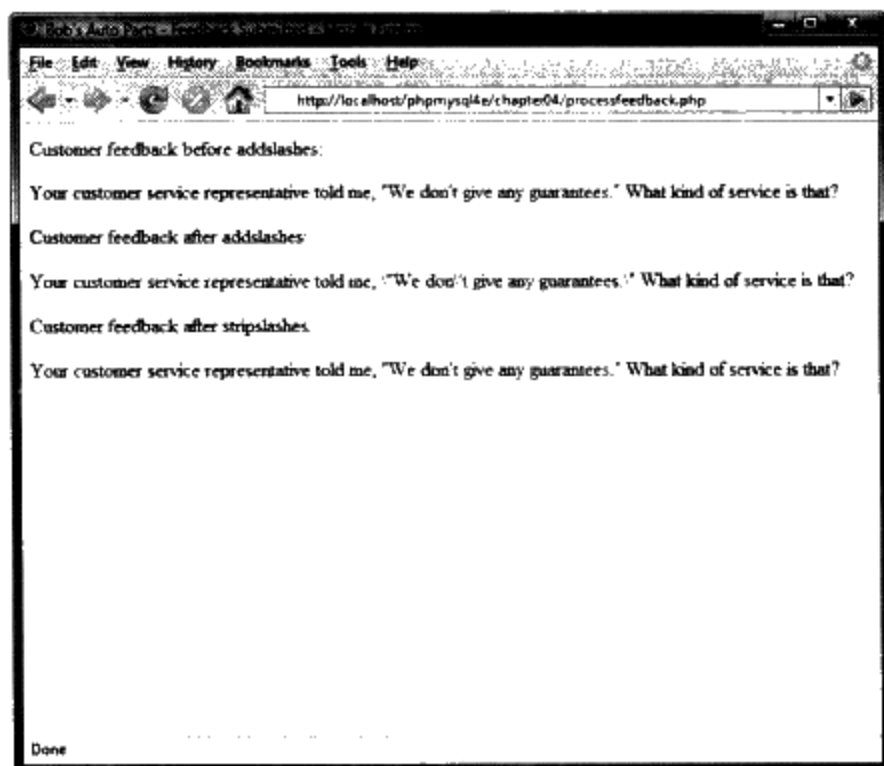


图4-3 调用`addslashes()`后，所有引号将被加上反斜杠，而`Stripslashes()`会移除这些反斜杠

也可以在服务器上尝试执行这些函数，将获得与图4-4类似的输出结果。

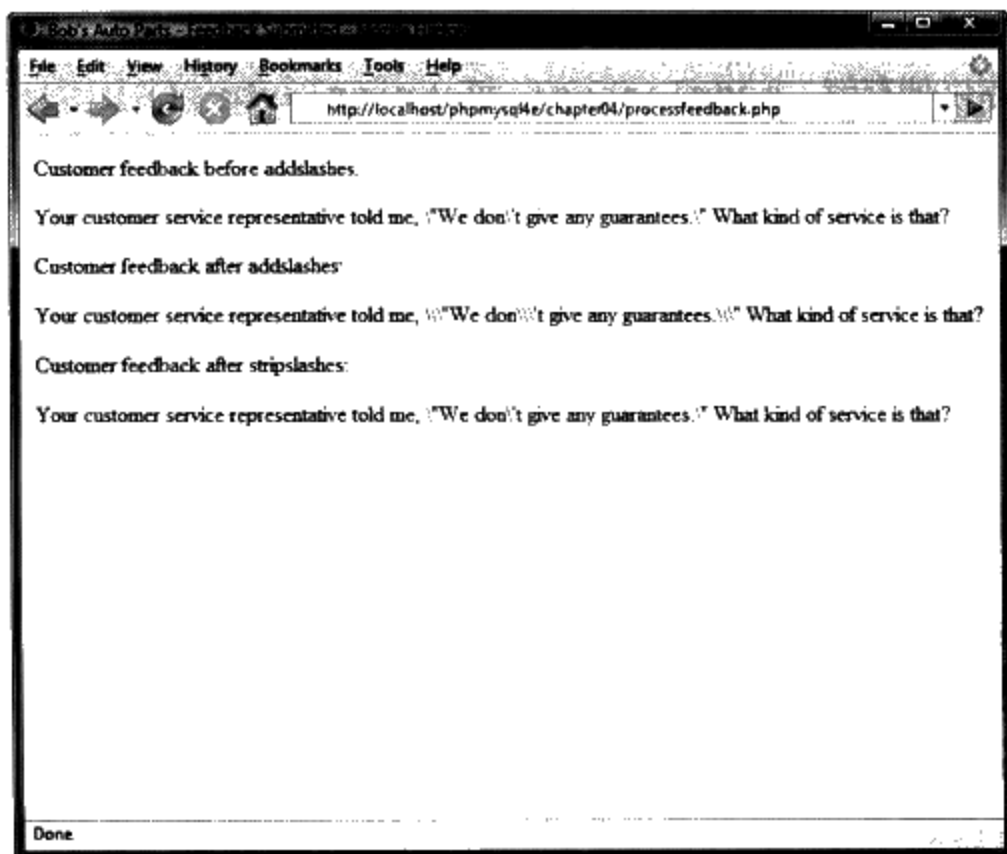


图4-4 所有存在问题的字符都被转义了两次，这就意味着魔术引号特征被启用了

如果你看到了上图中的结果，这表明你的PHP配置将自动添加或去除反斜杠。这个功能是由`magic_quotes_gpc`配置指令控制的。如今，在PHP新版本的默认安装情况下，该指令是启用的。`gpc`表示GET、POST和cookie，是第一个字母的组合。这就意味着，来自这些方法或

方式的变量将被自动包括在引号内。使用`get_magic_quotes_gpc()`函数，可以检查系统上的这个指令是否已经启用，如果来自这些方法的变量被自动引用在引号中，该函数将返回`true`。如果系统上该指令为启用的，在显示用户数据之前，必须调用`stripslashes()`函数；否则，这些反斜杠会被显示出来。

使用魔术引号将允许你编写具有更好可移植性的代码。可以在第23章中了解关于该特性的更多内容。

4.3 用字符串函数连接和分割字符串

通常，我们想查看字符串的各个部分。例如，查看句子中的单词（例如，拼写检查），或者要将一个域名或电子邮件地址分割成一个个的组件部分。PHP提供了几个字符串函数（和一个正则表达式函数）来实现此功能。

在我们的例子中，Bob想让顾客的反馈信息直接从`bigcustomer.com`提交到他那里，所以，可以将客户输入的电子邮件地址分割为几个部分，以便判断是否为大客户。

4.3.1 使用函数`explode()`、`implode()`和`join()`

为了实现这个功能，我们将使用的第一个函数是`explode()`，它的函数原型如下所示：

```
array explode(string separator, string input [, int limit]);
```

这个函数带有一个输入字符串作为参数，并根据一个指定的分隔符字符串将字符串本身分割为小块，将分割后的小块返回到一个数组中。可以通过可选的参数`limit`来限制分成字符串小块的数量。

要在我们的脚本中通过顾客的电子邮件地址获得域名，可以使用如下所示的代码：

```
$email_array = explode( '@', $email);
```

在这里，调用函数`explode()`将顾客的电子邮件地址分割成两部分：用户名称，它保存于`$email_array[0]`中，而域名则保存在`$email_array[1]`中。现在，我们已经可以测试域名来判断顾客的来源，然后将他们的反馈送到合适的人那里：

```
if ($email_array[1] == "bigcustomer.com") {
    $toaddress = "bob@example.com ";
} else {
    $toaddress = "feedback@example.com ";
}
```

然而请注意，如果域名是大写的或者大小写混合的，这个函数就无法正常使用。你可以通过将域名转换成全是大写或小写的方法避免这个问题，然后再按如下所示方法进行检查是否匹配：

```
if (strtolower($email_array[1]) == "bigcustomer.com") {
    $toaddress = "bob@example.com ";
} else {
    $toaddress = "feedback@example.com ";
}
```


使用implode()或join()函数来实现与函数explode()相反的效果，这两个函数的效果是一致的。例如：

```
$new_email = implode( '@', $email_array);
```

以上代码是从\$email_array中取出数组元素，然后用第一个传入的参数字符将它们连接在一起。这个函数的调用同explode()十分相似，但效果却相反。

4.3.2 使用strtok()函数

与函数explode()每次都将一个字符串全部分割成若干小块不同，strtok()函数一次只从字符串中取出一些片段（称为令牌）。对于一次从字符串中取出一个单词的处理来说，strtok()函数比explode()函数的效果更好。

strtok()函数的原型如下所示：

```
string strtok(string input, string separator);
```

分隔符可以是一个字符，也可以是一个字符串，但是，需要注意的是，输入的字符串会根据分隔符字符串中的每个字符来进行分割，而不是根据整个分隔字符串来分隔（就像explode()函数一样）。

函数strtok()的调用并不像它的函数原型中那样简单。为了从字符串中得到第一个令牌，可以调用strtok()函数，并带有两个输入参数：一个是要进行令牌化处理的字符串，还有一个就是分隔符。为了从字符串中得到令牌序列，可以只用一个参数——分隔符。该函数会保持它自己的内部指针在字符串中的位置。如果想重置指针，可以重新将该字符串传给这个函数。

strtok()函数的典型应用如下所示：

```
$token = strtok($feedback, " ");
echo $token. "<br /> ";
while ($token != "") {
    $token = strtok( " ");
    echo $token. "<br /> ";
}
```

通常，使用像empty()这样的函数来检查顾客是否在表单中真正输入了反馈信息，是一个非常不错的主意。为了简洁起见，我们将不对这些检查进行详细介绍。

以上代码将顾客反馈中的每个令牌打印在每一行上，并一直循环到不再有令牌。在这个过程中，空字符串将被自动跳过。

4.3.3 使用substr()函数

函数substr()允许我们访问一个字符串给定起点和终点的子字符串。这个函数并不适用于我们的例子中，但是，当需要得到某个固定格式字符串中的一部分时，它会非常有用。

substr()函数具有如下所示的函数原型：

```
string substr(string string, int start[, int length] );
```

这个函数将返回字符串的子字符串副本。

让我们来看几个使用如下测试字符串的例子：

```
$test = 'Your customer service is excellent' ;
```

如果只用一个正数作为子字符串起点来调用这个函数，将得到从起点到字符串结束的整个字符串。例如：

```
substr($test, 1);
```

以上代码将返回“our customer service is excellent”。请注意，字符串的起点和数组一样是从零开始的。

如果只用一个负数作为子字符串起点来调用它，将得到一个原字符串尾部的一个子字符串，字符个数等于给定负数的绝对值，例如：

```
substr($test, -9);
```

将返回“excellent”。

*length*参数可以用于指定返回字符的个数（如果它是正数），或是字符串序列的尾部（如果它是负数）。例如：

```
substr($test, 0, 4);
```

将返回字符串的头4个字符，即“Your”。下面的代码：

```
echo substr($test, 5, -13);
```

将返回从第4个到倒数第13个字符，即“customer service”。第一个字符的位置为0。因此位置5就是第6个字符。

4.4 字符串的比较

到目前为止，我们已经用过“==”号来比较两个字符串是否相等。使用PHP可以进行一些更复杂的比较。这些比较分为两类：部分匹配和其他情况。在这里，我们首先将讨论一下其他情况，然后再讨论在进一步开发Smart例子（智能表单邮件）中要用到的部分匹配。

4.4.1 字符串的排序：strcmp()、strcasecmp()和strnatcmp()

strcmp()、strcasecmp()和strnatcmp()函数可用于字符串的排序。当进行数据排序的时候，这些函数是非常有用的。

strcmp()的函数原型如下所示：

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

该函数需要两个进行比较的参数字符串。如果这两个字符串相等，该函数就返回0，如果按字典顺序str1在str2后面（大于str2）就返回一个正数，如果str1小于str2就返回一个负数。这个函数是区分大小写的。

函数strcasecmp()除了不区分大小写之外，其他和strcmp()一样。

函数strnatcmp()及与之对应的不区分大小写的strnatcasecmp()将按“自然排序”比较字符串，所谓自然排序是按人们习惯的顺序进行排序。例如，strcmp()会认为2大于12，

因为按字典顺序2要大于12，而`strnatcmp()`则是相反。关于自然排序可以在<http://www.naturalordersort.org/>网站上进一步了解。

4.4.2 使用`strlen()`函数测试字符串的长度

可以使用函数`strlen()`来检查字符串的长度。如果传给它一个字符串，这个函数将返回字符串的长度。例如，如下所示的代码将返回5：

```
echo strlen('hello'); .
```

这个函数可以用来验证输入的数据。考虑一下我们表单中的电子邮件地址，它存储在变量`$email`中。检验一个保存在`$email`变量中的电子邮件地址的基本方法就是检查它的长度。根据推理，如果一个国家的代码没有二级域名，只有一个字母的服务器名称和一个字母的电子邮件地址，那么它的最小长度是6个字符——例如a@a.to。因此，如果一个地址没有达到这个长度就会报错，如下所示：

```
if (strlen($email) < 6){  
    echo 'That email address is not valid ;  
    exit; // force execution of PHP script  
}
```

很明显，这是一个验证信息是否有效的非常简单的方法。我们将在下一节中介绍一种更好的方法。

4.5 使用字符串函数匹配和替换子字符串

通常，我们需要检查一个更长的字符串中是否含有一个特定的子字符串。这种部分匹配通常比测试字符串的完全等价更有用处。

在智能表单例子中，我们希望根据反馈信息中的一些关键词来将它们发送到适当的部门。

例如，如果希望将关于Bob商店的信件发到销售经理那里，就需要知道消息中是否出现了单词“shop”（或它的派生词）。

在了解了前面所介绍的函数后，就可以使用函数`explode()`和`strtok()`在消息中检索每个单词，然后通过运算符“==”或函数`strcmp()`对它们进行比较。

然而，还可以调用一个字符串函数或正则表达式匹配函数来完成相同的操作。这些函数可以用于在一个字符串中搜索一个模式。稍后，我们将依次介绍这些函数。

4.5.1 在字符串中查找字符串：`strstr()`、`strchr()`、`strrchr()`和`stristr()`

为了在一个字符串中查找另一个字符串，可以使用函数`strstr()`、`strchr()`、`strrchr()`和`stristr()`中的任意一个。

函数`strstr()`是最常见的，它可以用于在一个较长的字符串中查找匹配的字符串或字符。请注意，在PHP中，函数`strchr()`和`strstr()`完全一样，虽然其函数名的意思是在一个字符串中查找一个字符，类似于C语言中的同样函数。在PHP中，这两个函数都可用于在字符串中查找一个字符串，包括查找只包含一个字符的字符串。

`strstr()`的函数原型如下所示:

```
string strstr(string haystack, string needle);
```

你必须向函数传递一个要被搜索的子字符串参数和一个目标关键字字符串参数。如果找到了目标关键字的一个精确匹配,函数会从目标关键字前面返回被搜索的字符串,否则返回值为`false`。如果存在不止一个目标关键字,返回的字符串从出现第一个目标关键字的位置开始。

例如,在智能表单应用程序中,可以按如下方式决定将邮件送到哪里:

```
$toaddress = 'feedback@example.com'; // the default value

// Change the $toaddress if the criteria are met
if (strstr($feedback, 'shop'))
    $toaddress = 'retail@example.com';
else if (strstr($feedback, 'delivery'))
    $toaddress = 'fulfillment@example.com';
else if (strstr($feedback, 'bill'))
    $toaddress = 'accounts@example.com';
```

首先,这段代码将检查反馈信息中特定的关键字,然后将邮件发送给适当的人。例如,如果客户的反馈信息是“I still haven’t received delivery of my last order”,以上代码就将找到字符串“delivery”,这样该反馈信息就将被送到fulfillment@example.com。

函数`strstr()`有两个变体。第一个变体是`stristr()`,它几乎和`strstr()`一样,其区别在于不区分字符大小写。对于我们的智能表单应用程序来说,这个函数非常有用,因为用户可以输入“delivery”、“Delivery”或“DELIVERY”以及其他大小写混合的情况。

第二个变体是`strrchr()`,它也几乎和`strstr()`一样,但会从最后出现目标关键字的位置的前面返回被搜索字符串。

4.5.2 查找子字符串的位置: `strpos()`、`strrpos()`

函数`strpos()`和`strrpos()`的操作和`strstr()`类似,但它不是返回一个子字符串,而返回目标关键字子字符串在被搜索字符串中的位置。更有趣的是,现在的PHP手册建议使用`strpos()`函数替代`strstr()`函数来查看一个子字符串在一个字符串中出现的位置,因为前者的运行速度更快。

函数`strpos()`的原型如下所示:

```
int strpos(string haystack , string needle, int [ offset ] );
```

返回的整数代表被搜索字符串中第一次出现目标关键字子字符串的位置。通常,第一个字符是位置0。

例如,如下代码将会在浏览器中显示数值4:

```
$test = "Hello world ";
echo strpos($test, 'o');
```

在这个例子中,我们只是用一个字符作为目标关键字参数,实际上目标关键字参数可以是任意长度的字符串。

该函数的可选参数`offset`是用来指定被搜索字符串的开始搜索位置。例如：

```
echo strpos($test, 'o', 5);
```

以上代码会在浏览器中显示数值7，因为PHP是从位置5开始搜索字符“o”的，所以就看不到位置4的那个字符。

函数`strrpos()`也几乎是一样的，但返回的是被搜索字符串中最后一次出现目标关键字子字符串的位置。

在任何情况下，如果目标关键字不在字符串中，`strpos()`或`strrpos()`都将返回`false`。因此，这就可能带来新的问题，因为`false`在一个如PHP这样的弱类型语言中等于0，也就是说字符串的第一个字符。

可以使用运算符“`===`”来测试返回值，从而避免这个问题：

```
$result = strpos($test, 'H');
if ($result === false) {
    echo "Not found ";
} else {
    echo "Found at position ".$result;
}
```

4.5.3 替换子字符串：`str_replace()`、`substr_replace()`

查找替换功能在字符串中非常有用。可以使用查找替换从而通过PHP生成个性化文档—例如，用人名来替换`<name>`，用他们的地址来替换`<address>`。也可以使用这项功能来删改特定的术语，例如在一个论坛应用程序中，或是在智能表单应用程序中。需要再次提到的是，可以用字符串函数或者正则表达式函数来实现此功能。

进行替换操作最常用的字符串函数是`str_replace()`。它的函数原型如下所示：

```
mixed str_replace(mixed needle, mixed new_needle, mixed haystack[, int & count]);
```

这个函数用“`new_needle`”替换所有`haystack`中的“`needle`”，并且返回`haystack`替换后的结果。可选的第四个参数是`count`，它包含了要执行的替换操作次数。

提示 你可以以数组的方式传递所有参数，该函数可以很好地完成替换。可以传递一个要被替换单词的数组，一个替换单词的数组，以及应用这些规则的目标字符串数组。这个函数将返回替换后的字符串数组。

例如，因为人们使用智能表单来投诉，所以可能会用一些具有“感情色彩”的单词。作为程序员，我们通过使用一个包含了带有“感情色彩”单词的数组`$offcolor`让Bob公司的各部门免于受到辱骂，如下所示的代码就是在`str_replace()`函数中使用数组的例子：

```
$feedback = str_replace($offcolor, '%!@*', $feedback);
```

函数`substr_replace()`则用来在给定位置中查找和替换字符串中特定的子字符串。它的原型如下所示：

```
string substr_replace(string string, string replacement,
                      int start, int [length]);
```

这个函数使用字符串`replacement`替换字符串`string`中的一部分。具体是哪一部分则取决于起始位置值和可选参数`length`的值。

`start`的值代表要替换字符串位置的开始偏移量。如果它为0或是一个正值，就是一个从字符串开始处计算的偏移量；如果它是一个负值，就是从字符串末尾开始的一个偏移量。

例如，如下代码会用“X”替换`$test`中的最后一个字符：

```
$test = substr_replace($test, 'X', -1);
```

参数`length`是可选的，它代表PHP停止替换操作的位置。如果不给定它的值，它会从字符串`start`位置开始一直到字符串结束。

如果`length`为零，替换字符串实际上会插入到字符串中而覆盖原有的字符串。一个正的`length`表示要用新字符串替换掉的字符串长度。一个负的`length`表示从字符串尾部开始到第`length`个字符停止替换。

4.6 正则表达式的介绍

PHP支持两种风格的正则表达式语法：POSIX和Perl。这两种风格的正则表达式是PHP编译时的默认风格。在PHP 5.3版本中，Perl风格不能被禁用。然而，这里我们将介绍更简单的POSIX风格，但如果你已经是一位Perl程序员，或者希望了解更多关于PCRE的内容，可以阅读在线手册：<http://www.php.net/pcre>。

提示 POSIX正则表达式更容易掌握，但示它们不是二进制安全的。

到目前为止，我们进行的所有模式匹配都使用了字符串函数。我们只限于进行精确匹配，或精确的子字符串匹配。如果希望完成一些更复杂的模式匹配，应该用正则表达式。正则表达式在开始时候很难掌握，但却是非常有用的。

4.6.1 基础知识

正则表达式是一种描述一段文本模式的方法。到目前为止，我们前面所用到过的精确（文字）匹配也是一种正则表达式。例如，前面我们曾搜索过正则表达式的术语，像“shop”和“delivery”。

在PHP中，匹配正则表达式更有点像`strstr()`匹配，而不像相等比较，因为是在一个字符串的某个位置（如果不指明则可能在字符串中的任何位置）匹配另一个字符串。例如，字符串“shop”匹配正则表达式“shop”。它也可以匹配正则表达式“h”、“ho”，等。

除了精确匹配字符外，还可以用特殊字符来指定表达式的元意（meta-meaning）。例如，使用特殊字符，可以指定一个在字符串开始或末尾肯定存在的模式，该模式的某部分可能被重复，或模式中的字符属于特定的某一类型。此外，还可以按特殊字符的出现来匹配。接下来，我们将逐个讨论这些变化。

4.6.2 字符集和类

使用字符集可以马上给出比精确匹配功能还要强大的正则表达式。字符集可以用于匹配属

于特定类型的任何字符；事实上它们是一种通配符。

首先，可以用字符作为一个通配符来代替除换行符（`\n`）之外的任一个字符。例如，正则表达式：

```
.at
```

可以与“cat”、“sat”和“mat”等进行匹配。通常，这种通配符匹配用于操作系统中的文件名匹配。

但是，使用正则表达式，可以更具体地指明希望匹配的字符类型，而且可以指明字符所属的一个集合。在前面的例子中，正则表达式匹配“cat”和“mat”，但也可以匹配“#at”。如果要限定它是a到z之间的字符，就可以像下面这样指明：

```
[a-z]at
```

任何包含在方括号（`[]`）中的内容都是一个字符类——一个被匹配字符所属的字符集合。请注意，方括号中的表达式只匹配一个字符。

我们可以列出一个集合，例如：

```
{aeiou}
```

可以用来表示元音字母。

也可以描述一个范围，正如前面用连字符那样，也可以是一个范围集：

```
[a-zA-Z]
```

这个范围集代表任何的大小写字母。

此外，还可以用集合来指明字符不属于某个集。例如：

```
[^a-z]
```

可以用来匹配任何不在a和z之间的字符。当把脱字符号（`^`）包括在方括号里面时，表示否。当该符号用在方括号的外面，则表示另外一个意思，我们稍后将详细介绍。

除了列出了集合和范围，许多预定义字符类也可以在正则表达式中使用，如表4-3所示。

表4-3 用于POSIX风格的正则表达式的字符类

类	匹 配	类	匹 配
<code>[[:alnum:]]</code>	文字数字字符	<code>[[:punct:]]</code>	标点符号
<code>[[:alpha:]]</code>	字母字符	<code>[[:blank:]]</code>	制表符和空格
<code>[[:lower:]]</code>	小写字母	<code>[[:space:]]</code>	空白字符
<code>[[:upper:]]</code>	大写字母	<code>[[:cntrl:]]</code>	控制符
<code>[[:digit:]]</code>	小数	<code>[[:print:]]</code>	所有可打印的字符
<code>[[:xdigit:]]</code>	十六进制数字	<code>[[:graph:]]</code>	除空格外所有可打印的字符

4.6.3 重复

通常，读者会希望指明某个字符串或字符类将不止一次地出现。可以在正则表达式中使用两个特殊字符代替。符号“*”表示这个模式可以被重复0次或更多次，符号“+”则表示这个

模式可以被重复1次或更多次。这两个符号应该放在要作用的表达式的后面。

例如：

```
[[:alnum:]]+
```

表示“至少有一个字母字符”。

4.6.4 子表达式

通常，将一个表达式分隔为几个子表达式是非常有用的，例如，可以表示“至少这些字符串中的一个需要精确匹配”。可以使用圆括号来实现，与在数学表达式中的方法一样。

例如：

```
(very)*large
```

可以匹配“large”、“very large”、“very very large”等。

4.6.5 子表达式计数

可以用在花括号（{}）中的数字表达式来指定内容允许重复的次数。可以指定一个确切的重复次数（{3}表示重复3次），或者一个重复次数的范围（{2, 4}表示重复2~4次），或是一个开底域的重复范围（{2, }表示至少要重复两次）。

例如：

```
(very){1, 3}
```

表示匹配“very”、“very very”和“very very very”。

4.6.6 定位到字符串的开始或末尾

[a-z]模式将匹配任何包含了小写字母字符的字符串。无论该字符串只有一个字符，或者在整个更长的字符串中只包含一个匹配的字符，都没有关系。

也可以确定一个特定的子表达式是否出现在开始、末尾或在两个位置都出现。当要确定字符串中只有要找的单词而没有其他单词出现时，它将相当有用。

脱字符号、(^)用于正则表达式的开始，表示子字符串必须出现在被搜索字符串的开始处，字符“\$”用于正则表达式的末尾，表示子字符串必须出现在字符串的末尾。

例如，以下是在字符串开始处匹配bob：

```
^bob
```

这个模式将匹配com出现在字符串末尾处的字符串：

```
com$
```

最后，这个模式将匹配只包含a到z之间一个字符的字符串：

```
^[a-z]$
```


4.6.7 分支

可以使用正则表达式中的一条竖线来表示一个选择。例如，如果要匹配com、edu或net，就可以使用如下所示的表达式：

```
com|edu|net
```

4.6.8 匹配特殊字符

如果要匹配本节前面提到过的特殊字符，例如，.、{或者\$，就必须在它们前面加一个反斜杠（\）。如果要匹配一个反斜杠，则必须用两个反斜杠（\\）来表示。

在PHP中，必须将正则表达式模式包括在一个单引号字符串中。使用双引号引用的正则表达式将带来一些不必要的复杂性。PHP还使用反斜杠来转义特殊字符——例如反斜杠。

如果希望在模式中匹配一个反斜杠，必须使用两个反斜杠来表示它是一个反斜杠字符，而不是一个转义字符。

同样，由于相同的原因，如果希望在一个双引号引用的PHP字符串中使用反斜杠字符，必须使用两个反斜杠。这可能会有些混淆，这样要求的结果将是表示一个包含了反斜杠字符的正则表达式的一个PHP字符串需要4个反斜杠。PHP解释器将这4个反斜杠解释成2个。然后，由正则表达式解释器解析为一个。

\$符号也是双引号引用的PHP字符串和正则表达式的特殊字符。要使一个\$字符能够在模式中匹配，必须使用“\\\$”。因为这个字符串被引用在双引号中，PHP解释器将其解析为\\$，而正则表达式解释器将其解析成一个\$字符。

4.6.9 特殊字符一览

所有特殊字符的摘要如表4-4和表4-5所示。表4-4显示了方括号外特殊字符的意义，表4-5显示了当它们用在方括号里面的意义。

表4-4 在POSIX正则表达式中，用于方括号外面特殊字符的摘要

字 符	意 义	字 符	意 义
\	转义字符)	子模式的结束
^	在字符串开始匹配	*	重复0次或更多次
\$	在字符串末尾匹配	+	重复一次或更多次
.	匹配除换行符（\n）之外的字符	{	最小/最大量记号的开始
	选择分支的开始（读为或）	}	最小/最大量记号的结束
(子模式的开始	?	标记一个子模式为可选的

表4-5 POSIX正则表达式中，用于方括号里面特殊字符的摘要

字 符	意 义
\	转义字符
^	非，仅用在开始位置
-	用于指明字符范围

4.6.10 在智能表单中应用

在智能表单应用程序中，正则表达式至少有两种用途。第一种用途是在顾客的反馈中查找特定的名词。使用正则表达式，可以做得更智能一些。使用一个字符串函数，如果希望匹配“shop”、“customer service”或“retail”，就必须做3次不同的搜索。如果使用一个正则表达式，就可以同时匹配所有3个，如下所示：

```
shop|customer service|retail
```

第二个用途是验证程序中用户的电子邮件地址，这需要通过用正则表达式来对电子邮件地址的标准格式进行编码。这个格式中包含一些数字或标点符号，接着是符号“@”，然后是包括文字或数字和字符组成的字符串，后面接着是一个“.”（点号），后面包括文字或数字以连字符组成的字符串，可能还有更多的点号，直到字符串结束，它的编码如下所示：

```
^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9_\-]+\.[a-zA-Z0-9_\-\.]+$
```

子表达式`^[a-zA-Z0-9_\-\.]+`表示“至少由一个字母、数字、下画线、连字符、点号或者这些字符组合为开始的字符串”。请注意，当在一个字符类的开始或末尾处使用点号时，点号将失去其特殊通配符的意义，只能成为一个点号字符。

符号“@”匹配字符“@”。

而子表达式`[a-zA-Z0-9_\-\.]+`与包含文字数字字符和连字符的主机名匹配。请注意，我们去除了连字符，因为它是方括号内的特殊字符。

字符组合“`\.`”匹配“`.`”字符。我们在字符类外部使用点号，因此必须对其转义，使其能够匹配一个点号字符。

子表达式`[a-zA-Z0-9_\-\.]+$`匹配域名的剩下部分，它包含字母、数字和连字符，如果需要还可包含更多的点号直到字符串的末尾。

不难发现，有时一个无效的电子邮件地址也会符合这个正则表达式。找到所有无效电子邮件几乎是不可能的，但是经过分析，情形将会有所改善。可以按许多不同的方式精化这个表达式。例如，可以列出所有有效的顶级域（TLD）。当对某些对象进行限制的时候，请千万小心，因为可能排斥1%的有效数据的校验函数比允许出现10%的无效数据的校验函数还要麻烦。

以上我们了解了正则表达式，下面我们将介绍一下使用正则表达式的PHP函数。

4.7 用正则表达式查找子字符串

查找子字符串是正则表达式的主要应用。在PHP中，可以使用的并且用于匹配POSIX风格正则表达式的两个函数是`ereg()`和`eregi()`。`ereg()`函数原型如下所示：

```
int ereg(string pattern, string search, array [matches]);
```

该函数搜索字符串`search`，在`pattern`中寻找与正则表达式相匹配的字符串。如果发现了与`pattern`的子表达式相匹配的字符串，这些字符串将会存储在数组`matches`中，每个数组元素对应一个子表达式。

函数`eregi()`除了不区分大小写外，其他功能与`ereg()`一样。

可以使用如下所示的代码对智能表单例子进行修改：

```
if (!ereg( '[a-zA-Z0-9_\\-\\.]+@[a-zA-Z0-9\\-\\.]+[a-zA-Z0-9_\\-\\.]+$', $email)) {
    echo "<p>That is not a valid email address.</p>".
        "<p>Please return to the previous page and try again.</p>";
    exit;
}
$toaddress = "feedback@example.com"; // the default value
if (ereg("shop|customer service|retail", $feedback))
    $toaddress = "retail@example.com";
} else if (ereg("deliver|fulfill", $feedback)) {
    $toaddress = "fulfillment@example.com ";
} else if (ereg("bill|account", $feedback)) {
    $toaddress = "accounts@example.com ";
}
if (ereg("bigcustomer\\.com", $email)) {
    $toaddress = "bob@example.com ";
}
```

4.8 用正则表达式替换子字符串

与前面使用的`str_replace()`函数一样，也可以使用正则表达式来查找和替换子字符串。在正则表达式中，可以使用的两个函数是`ereg_replace()`和`eregi_replace()`，其原型如下所示：

```
string ereg_replace(string pattern, string replacement, string search);
```

该函数在字符串`search`中查找正则表达式`pattern`的字符串，并且用字符串`replacement`来替换。

函数`eregi_replace()`除了不区分大小写外，其他与`ereg_replace()`相同。

4.9 使用正则表达式分割字符串

另一个实用的正则表达式函数是`split()`，它的原型如下所示：

```
array split(string pattern, string search[, int max]);
```

这个函数将字符串`search`分割成符合正则表达式模式的子字符串，然后将子字符串返回到一个数组中。整数`max`指定进入数组中的元素个数。

该函数对分割电子邮件地址、域名或日期是非常有用的。例如：

```
$address = 'username@example.com';
$arr = split ("\\.|@ ", $address);
while (list($key, $value) = each ($arr)) {
    echo "<br /> ".$value;
}
```

以上代码将主机名分割为5个部分并将它们分别输出到一行。

```
username
@
```

```
example  
.  
com
```

提示 一般而言，对于同样的功能，正则表达式函数运行效率要低于字符串函数。如果应用程序足够简单，那么就用字符串表达式。但是，对于可以通过单个正则表达式执行的任务来说，如果使用多个字符串函数，则是不对的。

4.10 进一步学习

PHP有许多字符串函数。在本章中，我们已经介绍了最有用的部分函数，但是如果你有特殊需求（例如，将字符转换成Cyrillic），请查阅PHP的联机手册，以确认PHP是否具有所需要的功能。

大量关于正则表达式的资料可以使用。如果使用的是UNIX操作系统，可以从关于regex的man页面开始。在devshed.com和phpbuilder.com站点提供了许多关于正则表达式的文章。

Zend的网站有一个比我们在此处开发的更复杂、功能更强大的电子邮件验证函数，该函数叫做MailVal()，可以在<http://www.zend.com/codex.php?ozid=88&single=1>找到。

关于正则表达式就谈到这里了——你所看到并运行的例子越多，用起来也就会越有把握。

4.11 下一章

在下一章中，我们将讨论如何在PHP中实现代码重用，从而节省编程时间和精力以及减少代码冗余的方法。