第 2 章　数据的存储与检索

我们已经了解了如何访问和操作输入到HTML表单的数据，现在可以开始了解如何保存这些信息以备后期使用。在大多数情况下，包括上一章介绍的例子，我们都需要将数据存储起来并且以后再使用。在本章的例子中，我们需要将客户的订货单保存起来以便后期送货使用。

在这一章中，我们将介绍如何将上一章所介绍的客户订单例子保存到一个文件，然后再将其读出来。我们还将介绍为什么这种方法并不是一个很好的解决方案；当具有大量订单时，应该使用一个数据库管理系统，例如MySQL。

在本章中，我们将主要介绍以下内容：

- 保存数据以便后期使用
- 打开文件
- 创建并写入文件
- 关闭文件
- 读文件
- 给文件加锁
- 删除文件
- 其他有用的文件操作函数
- 更好的方式：数据库管理系统

2.1　保存数据以便后期使用

存储数据有两种基本方法：保存到普通文件（flat file），或者保存到数据库中。

普通文件可以具有多种格式，但是，通常所指的普通文件是简单的文本文件。在这个例子中，我们将顾客的订单写入到一个文本文件中，一个订单占据一行的位置。

这样保存起来非常简单，但是也存在一定的局限性，我们将在本章的后续内容中介绍其局限性。如果要处理相当数量的信息，很可能会想到使用数据库来代替它。但是，普通文件有其自己的用途，在某些情况下需要了解如何使用它们。

文件的读写操作与大多数编程语言的文件读写操作是类似的。如果读者曾经编写过C语言或者是UNIX Shell脚本，就会非常熟悉这些操作。

2.2　存储和检索Bob的订单

在本章中，我们将使用上一章所介绍的订单的改进版本。我们将从这个表单开始，编写一些PHP代码来处理订单数据。

提示　本章所使用的HTML和PHP脚本可以在随书附带文件的`chapter02`目录下找到。

我们已经对表单进行了修改，添加可以获得用户送货地址的快捷方法。在图2-1中，可以看到这个修改过的表单。

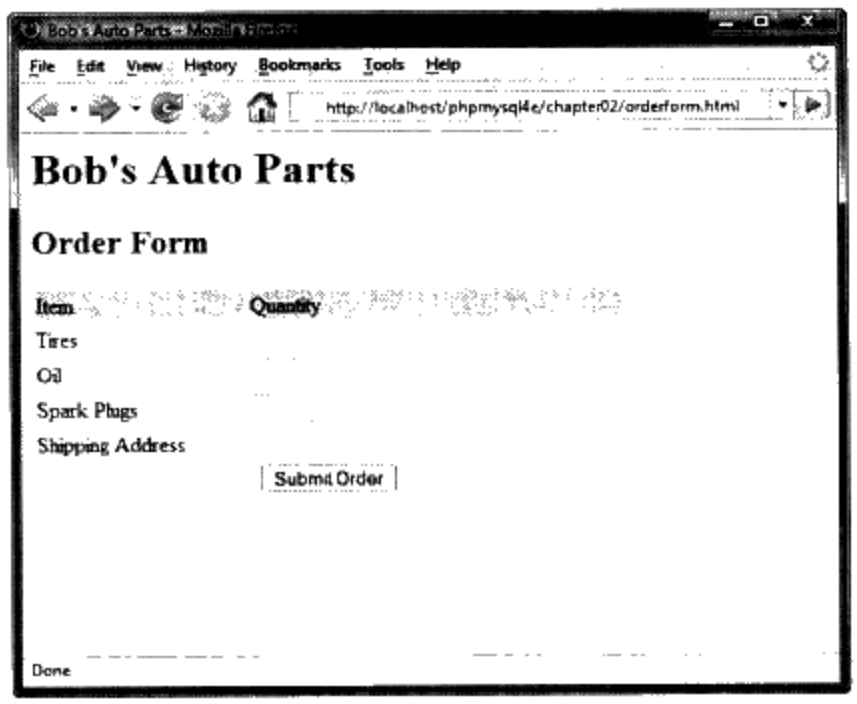

图2-1　订单的改进版本可以获得客户的送货地址

送货地址表单域称为address。这又给我们增加了一个变量，根据表单提交的METHOD不同，我们可以通过$_REQUEST['address']或$_POST['address']或$_GET['address']访问这个变量（请参阅第1章的详细介绍）。

在本章中，我们将所有订单都写入同一个文件。然后，我们将为Bob的员工构建一个Web界面，以便查看已经收到的订单。

2.3　文件处理

将数据写入一个文件，有以下3步操作。

1) 打开这个文件。如果文件不存在，需要先创建它。

2) 将数据写入这个文件。

3) 关闭这个文件。

同样，从一个文件中读出数据，也有以下3步操作。

1) 打开这个文件。如果这个文件不能打开（例如，文件不存在），就应该意识到这一点并且正确地退出。

2) 从文件中读出数据。

3) 关闭这个文件。

当希望从一个文件中读出数据时，可以选择一次从文件读取多少数据。我们将详细介绍一些常见的选择。现在，我们从打开文件开始。

2.4　打开文件

要在PHP中打开一个文件，可以使用fopen()函数。当打开一个文件的时候，还需要指定

如何使用它，也就是文件模式。

2.4.1　选择文件模式

服务器上的操作系统必须知道要对打开的文件进行什么操作。操作系统需要了解在打开这个文件后，这个文件是否还允许其他脚本再打开，它还需要了解使用者（或脚本属主）是否具有在这种方式下使用该文件的权限。从本质上说，文件模式可以告诉操作系统一种机制，这种机制可以决定如何处理来自其他人或脚本的访问请求，以及一种用来检查你是否有权访问这个特定文件的方法。

当打开一个文件的时候，有以下3种选择。

1) 打开文件为了只读、只写或者读和写。

2) 如果要写一个文件，你可能希望覆盖所有已有的文件内容，或者仅仅将新数据追加到文件末尾。如果该文件已经存在，也可以终止程序的执行而不是覆盖该文件。

3) 如果希望在一个区分了二进制方式和纯文本方式的系统上写一个文件，还必须指定采用的方式。

函数`fopen()`支持以上3种方式的组合。

2.4.2　使用`fopen()`打开文件

假设要将一个顾客订单写入Bob的订单文件中，可以使用如下所示的语句打开这个文件：

```
$fp = fopen( "$DOCUMENT_ROOT/../orders/orders.txt ", 'w');
```

调用`fopen()`的时候，需要传递2个、3个或4个参数。通常使用2个参数，正如以上代码所示。

第一个参数是要打开的文件。正如以上代码所示，可以指定该文件的路径——在这里，`orders.txt`文件保存在`orders`目录中。我们已经使用了PHP内置变量`$_SERVER['DOCUMENT_ROOT']`。由于整个表单变量名称太长了，我们可以指定一个简短的名称。

这个变量指向了Web服务器文档树的根。我们使用“..”表示文档根目录的父目录。出于安全考虑的原因，这个目录位于整个文档树的外部。在这个例子中，除了通过我们所提供的接口之外，我们不希望还有其他Web接口能够访问它。这个路径称为**相对路径**，因为它描述了一个相对于文档根目录的文件系统位置。

由于我们为表单变量定义了一个简短名称，我们需要在脚本的开始处加上如下代码：

```
$DOCUMENT_ROOT = $_SERVER[ 'DOCUMENT_ROOT '];
```

将冗长风格变量内容复制给简短风格的变量名称。

就像有不同的方法可以访问表单数据一样，也可以使用不同的方法访问预定义的服务器变量。根据服务器设置不同（请参阅第1章获得详细介绍），可以通过如下3种方式得到文档根目录：

- `$_SERVER['DOCUMENT_ROOT']`
- `$DOCUMENT_ROOT`
- `$HTTP_SERVER_VARS['DOCUMENT_ROOT']`

对于表单数据，第一个风格是首选的。

我们还可以指定文件的绝对路径。这个路径是从根目录开始的（在UNIX系统中，根目录是/，而在Windows系统中通常都是c:\）。在UNIX服务器中，根目录就是/home/book/orders。这样做的问题在于，特别是如果将网站安装在别人的服务器上，这个绝对路径可能会改变。我们曾经有过沉重的教训，如果系统管理员没有发出任何通知就决定修改目录结构后，我们就不得不手工更改包含在大量脚本中的绝对路径。

如果没有指定路径，这个文件就将在脚本自身所在的相同目录中查找或者创建。如果通过某种CGI封装程序来运行PHP的，这可能又会有所不同，具体需要根据服务器的设置而定。

在UNIX环境下，目录中的间隔符是正斜线（/）。如果你使用的是Windows平台，可以使用正斜线或者反斜线。如果使用反斜线，就必须使用转义（escape，标注为一个特殊字符）字符，这样fopen()函数才能正确理解这些字符。要转义一个字符，只需简单地在其前面添加一个反斜线。如下代码所示：

```
$ip = fopen( "$DOCUMENT_ROOT\\..\\orders\\orders.txt ", 'w');
```

在PHP代码中，只有少数人会使用反斜线，因为这意味着代码只能在Windows上运行。

如果使用了正斜线，代码不需要任何修改就可以在Windows和UNIX机器上运行。

fopen()函数的第二个参数是文件模式，它是一个字符串，指定了将对文件进行的操作。在这个例子中，我们将“w”传给了fopen()——这就意味着要以写的方式打开这个文件。表2-1给出了所有的文件模式及其意义。

表2-1　fopen()函数的文件模式总结

模　式	模式名称	意　义
r	只读	读模式——打开文件，从文件头开始读
r+	只读	读写模式——打开文件，从文件头开始读写
w	只写	写模式——打开文件，从文件头开始读。如果该文件已经存在，将删除所有文件已有内容。如果该文件不存在，函数将创建这个文件
w+	只写	写模式——打开文件，从文件头开始读写。如果该文件已经存在，将删除所有文件已有内容。如果该文件不存在，函数将创建这个文件
x	谨慎写	写模式打开文件，从文件头开始写。如果文件已经存在，该文件将不会被打开，fopen()函数将返回false，而且PHP将产生一个警告
x+	谨慎写	读/写模式打开文件，从文件头开始写。如果文件已经存在，该文件将不会被打开，fopen()函数将返回false，而且PHP将产生一个警告
a	追加	追加模式——打开文件，如果该文件已有内容，将从文件末尾开始追加（写），如果该文件不存在，函数将创建这个文件
a+	追加	追加模式——打开文件，如果该文件已有内容，将从文件末尾开始追加（写）或者读，如果该文件不存在，函数将创建这个文件
b	二进制	二进制模式——用于与其他模式进行连接。如果文件系统能够区分二进制文件和文本文件，你可能会使用它。Windows系统可以区分，而UNIX则不区分。推荐一直使用这个选项，以便获得最大程度的可移植性。二进制模式是默认的模式
t	文本	用于与其他模式的结合。这个模式只是Windows系统下一个选项。它不是推荐选项，除非你曾经在代码中使用了b选项

在我们的例子中所使用的文件模式取决于需要如何使用这个系统。我们已经使用了“w”，

这表示只可以将一个订单写入文件中。每当一个新订单被写入文件，它将覆盖以前的订单。这样做可能没有什么意义，所以最好使用追加模式（以及推荐的二进制模式）：

```
$fp = fopen( "$DOCUMENT_ROOT/../orders/orders.txt", 'ab');
```

fopen()函数的第3个参数是可选的。如果要在include_path（在PHP的配置中设置。请参阅附录A）中搜索一个文件，就可以使用它。如果希望进行如此操作，可以将这个参数设置为1。如果希望PHP搜索include_path，就不需要提供目录名称或路径：

```
$fp = fopen( 'orders.txt', 'ab', true);
```

第4个参数也是可选的。fopen()函数允许文件名称以协议名称开始（例如，http://）并且在一个远程的位置打开文件。对于这个额外的参数，它还支持一些其他的协议。我们将在本章的下一节详细介绍该函数的使用。

如果fopen()成功地打开一个文件，该函数将返回一个指向这个文件的文件指针。在这个例子中，文件指针保存在$fp中。当读者的确希望能够读写这个文件时，将使用这个变量来访问文件。

2.4.3 通过FTP或HTTP打开文件

除了打开一个本地文件进行读写操作之外，也可以使用fopen()函数通过FTP、HTTP或其他协议来打开文件。在php.ini文件中，可以通过关闭allow_url_fopen指令来禁用这个功能。如果在使用该函数打开一个远程文件时遇到问题，请检查php.ini文件。

如果使用的文件名是以ftp://开始的，fopen()函数将建立一个连接到指定服务器的被动模式，并返回一个指向文件开始的指针。

如果使用的文件名是以http://开始的，fopen()函数将建立一个到指定服务器的HTTP连接，并返回一个指向HTTP响应的指针。当使用PHP早期版本的HTTP模式时，必须在目录名称后添加结束斜线，如下所示：

http://www.example.com/

而不是：

http://www.example.com

当使用后一种地址形式（不带斜线）时，Web服务器通常会使用HTTP重定向到第一个地址（带斜线的地址）。

请记住，URL中的域名不区分大小写，但是路径和文件名可能会区分大小写。

2.4.4 解决打开文件时可能遇到的问题

当打开文件时，可能经常遇到的错误是试图打开一个没有权限进行读写操作的文件（这种错误通常只会在类似于UNIX的操作系统见到，但是偶尔也会在Windows平台上遇到）。PHP将会给出一个类似于图2-2所示的警告。

如果遇到这样的问题，必须确认运行该脚本的用户是否有权访问要使用的文件。根据服务器设置的不同，该脚本可能是作为Web服务器用户或者脚本所在目录的拥有者来运行的。

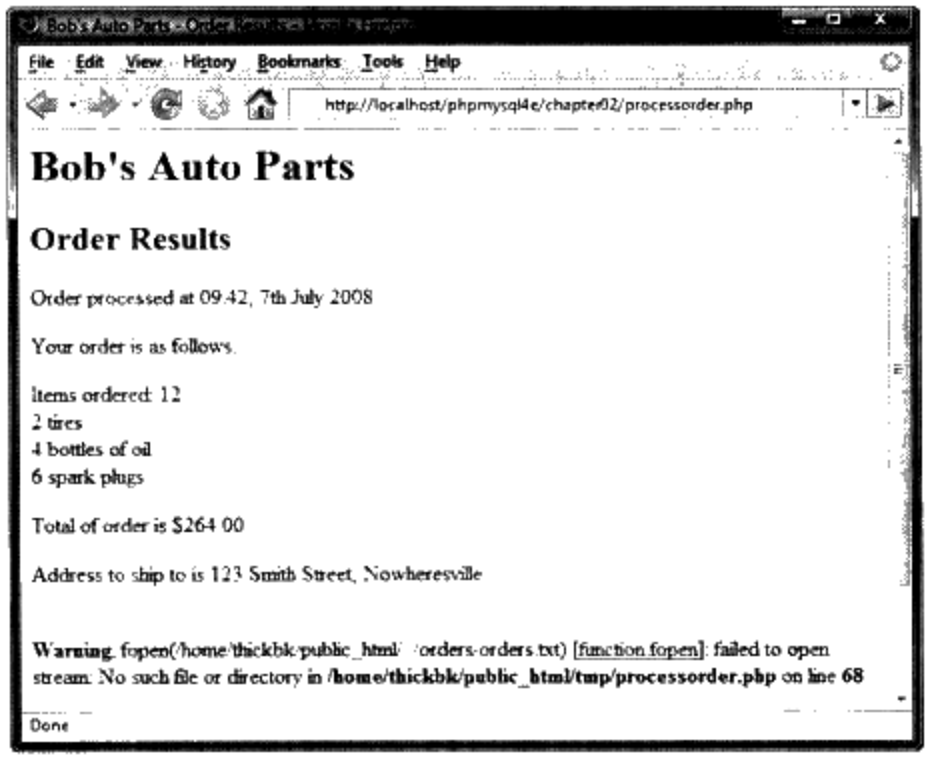

图2-2　当一个文件不能打开时，PHP将给出特定警告

在大多数系统中，该脚本将作为Web服务器用户来运行。如果脚本是在UNIX系统的~/public_html/chapter2/目录下，输入如下所示的命令，可以创建一个全局可写的目录来存储订单：

```
mkdir ~/orders
chmod 777 ~/orders
```

请记住，任何人都可以写的目录和文件是非常危险的。不应该具有可以从Web上直接可写的目录。正是由于这个原因，orders目录是两个子目录，都在public_html目录之上。在第15章中详细介绍安全问题。

设置了不正确的访问权限可能是造成打开文件时出现错误的常见原因。如果文件不能打开，你需要知道这一点，这样就不会再去读写数据。

如果fopen()函数调用失败，函数将返回false。可以以一种对于用户友好的方式来处理这个错误，可以通过抑制PHP的错误信息并且根据自己的方式给出错误信息：

```
@ $fp = fopen( "$DOCUMENT_ROOT/../orders/orders.txt", 'ab');
if (!$fp){
  echo "<p><strong> Your order could not be processed at this time.'
        .Please try again later.</strong></p></body></html>";
  exit;
}
```

fopen()函数调用前面的@符号可以告诉PHP抑制所有由该函数调用所产生的错误。通常，在出现错误的时候，这是一个不错的方法。但是，在这种情况下，要在其他地方处理它。

以上代码也可以写成：

```
$fp = @fopen( "$DOCUMENT_ROOT/../orders/orders.txt", 'a');
```

但是，这样使用错误抑制操作符并不是非常直观，而且只会使得代码调试更困难。这里所介绍的方法只是处理错误的简单方法。在第7章中，我们将详细介绍错误处理的好方法。

if语句可以用来测试变量$fp，查看fopen()函数是否返回了一个有效的文件指针。如果没有，它就会打印出一个错误信息并且终止脚本的执行。由于页面在这一步结束执行，请注意，我们在这里关闭了HTML标记，从而可以生成有效的HTML。

当使用这种方法时，将得到如图2-3所示的输出。

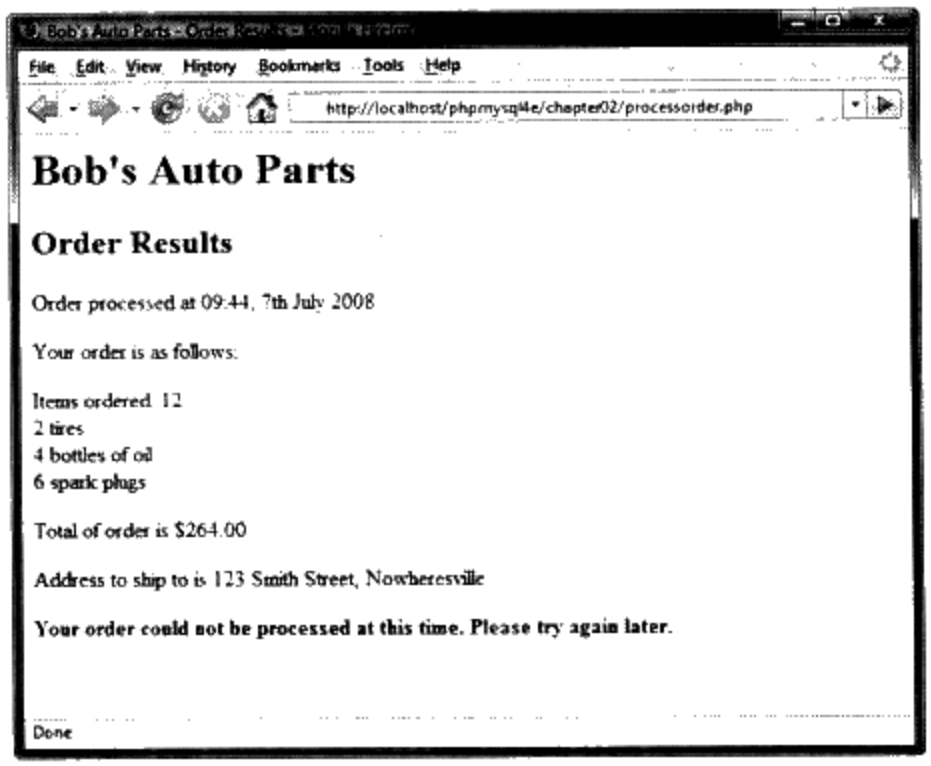

图2-3 用自己的错误信息替代PHP的错误信息可以使用户觉得更加友好

2.5 写文件

在PHP中写文件相对比较简单。可以使用fwrite()（file write，文件写）或者fputs()（file put string），fputs()是fwrite()的别名函数。我们可以使用如下方式调用fwrite()：

```
fwrite($fp, $outputstring);
```

这个函数告诉PHP将保存在$outputstring中的字符串写入到$fp指向的文件中。

fwrite()函数的一个替换函数是file_put_contents()，其函数原型如下所示：

```
int file_put_contents ( string filename,
                        string data
                        [, int flags
                        [, resource context]])
```

这个函数可以在不需要调用fopen()（或fclose()）函数打开要写的文件以前，将包含在data中的字符串数据写入到*filename*所指定的文件中。这个函数是PHP 5新引入的，与之匹配的函数是file_get_contents()，稍后我们将介绍这两个函数。当使用FTP或HTTP向远程文件写入数据时，最常用的是可选参数*flags*和*context*。（我们将在第20章中详细介绍这些函数）。

2.5.1 fwrite()的参数

实际上，函数fwrite()具有3个参数，但是第3个参数是可选的。fwrite()的原型如下所示：

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

第3个参数length是写入的最大字符数。如果给出了这个参数，fwrite()将向handle指向的文件写入字符串，一直写到字符串的末尾，或者已经写入了length字节，满足这两个条件之一就停止写入。

可以通过PHP的内置strlen()函数获得字符串的长度，如下所示：

```
fwrite($fp, $outputstring, strlen($outputstring));
```

当使用二进制模式执行写操作的时候，你可能会希望使用第3个参数，因为它可以帮助你避免一些跨平台的兼容性问题。

2.5.2 文件格式

当创建一个如我们例子中使用到的数据文件时，保存数据的格式将完全由你决定。(然而，如果打算在另一个应用程序中使用这个数据文件，你可能就不得不遵循那个应用程序的规则。)

下面，让我们构造一个表示数据文件中一条记录的字符串。可以使用如下所示语句：

```
$outputstring = $date. "\t ".$tireqty. "tires \t ".$oilqty. "oil\t"
                .$sparkqty. "spark plugs\t\$ ".$totalamount
                .'\t '. $address. '\n ";
```

在这个简单的例子中，我们将每一个订单记录保存在文件的一行中。我们选择每行记录一个订单这种格式是因为这样可以使用换行字符作为简单的记录间隔符。由于换行字符并不是可见的，因此我们使用控制序列"\n"来表示。

在本书的所有例子中，我们每次按照相同的顺序写入数据字段，并且使用制表符来分隔每一个字段。需要再次提到的是，由于制表符是不可见的，因此可以使用控制序列"\t"来表示。可以选择任何便于以后读取的、有意义的定界符。

分隔字符或定界字符一定不能出现在输入中，或者我们对输入进行处理，将分隔符删除或者进行转义处理。在第4章中详细介绍输入的处理。

现在，我们假设没有人会在订单中输入制表符。对于一个用户来说，在一行HTML的输入域中输入一个制表符或者换行字符是比较困难的，但是这并不是没有可能的。

使用特殊的域分隔符便于在读取数据的时候将数据分隔成不同的变量。在第3章和第4章中，我们将详细讨论这一点。从现在开始，我们将每一个订单当作一个字符串进行处理。

处理了一些订单后，该文件的内容将类似于程序清单2-1所示。

程序清单2-1　orders.txt——订单文件可能包含内容的示例

```
20:30, 31st March 2008 4 tires 1 oil 6 spark plugs $434.00 22 Short St,
Smalltown
20:44, 31st March 2008 1 tires 0 oil 0 spark plugs $100.00 33 Main Rd,
```

```
Newtown
20:43, 31st March 2008 0 tires 1 oil 4 spark plugs $26.00 127 Acacia St,
Springfield
```

2.6 关闭文件

当使用完文件后，应该将其关闭。应该按照如下所示的方式调用fclose()函数：

```
fclose($fp);
```

如果该文件被成功地关闭，函数将返回一个true值。反之，该函数将返回false。通常，关闭文件的操作并不像打开文件容易出错，所以在这个例子中我们并没有对该操作进行测试。

processorder.php的完整脚本清单如程序清单2-2所示。

程序清单2-2 processorder.php——订单处理脚本的最终版本

```
<?php
  // create short variable names
  $tireqty = $_POST['tireqty'];
  $oilqty = $_POST['oilqty'];
  $sparkqty = $_POST['sparkqty'];
  $address = $_POST['address'];
  $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
  $date = date('H:i, jS F Y');
?>
<html>
<head>
  <title>Bob's Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
<?php

    echo "<p>Order processed at ".date('H:i, jS F Y')."</p>";

    echo "<p>Your order is as follows: </p>";

    $totalqty = 0;

    $totalqty = $tireqty + $oilqty + $sparkqty;
    echo 'Items ordered: '.$totalqty."<br />";


    if ($totalqty == 0) {

      echo "You did not order anything on the previous page!<br />";

    } else {

      if ($tireqty > 0) {
```

```
        echo $tireqty." tires<br />";
    }

    if ($oilqty > 0) {
      echo $oilqty." bottles of oil<br />";
    }

    if ($sparkqty > 0) {
      echo $sparkqty." spark plugs<br />";
    }
}

$totalamount = 0.00;

define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);

$totalamount = $tireqty * TIREPRICE
             + $oilqty * OILPRICE
             + $sparkqty * SPARKPRICE;

$totalamount=number_format($totalamount, 2, '.', '');

echo "<p>Total of order is $".$totalamount."</p>";
echo "<p>Address to ship to is ".$address."</p>";

$outputstring = $date."\t".$tireqty." tires \t".$oilqty." oil\t"
                .$sparkqty." spark plugs\t\$".$totalamount
                ."\t". $address."\n";
// open file for appending
@ $fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'ab');

flock($fp, LOCK_EX);

if (!$fp) {
  echo "<p><strong> Your order could not be processed at this time.
        Please try again later.</strong></p></body></html>";
  exit;
}

fwrite($fp, $outputstring, strlen($outputstring));
flock($fp, LOCK_UN);
fclose($fp);

echo "<p>Order written.</p>";
?>
</body>
```

2.7 读文件

现在，Bob的客户可以通过Web下订单了，但是如果Bob的员工希望查看这些订单，他们就必须自己打开这些文件。

我们可以建立一个Web界面，从而方便Bob的员工读取这些文件。这个界面代码如程序清单2-3所示。

程序清单2-3 vieworders.php——用来查看订单文件的员工界面

```
<?php
  //create short variable name
  $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
?>
<html>
<head>
  <title>Bob's Auto Parts - Customer Orders</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Customer Orders</h2>
<?php

    @$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'rb');
    if (!$fp) {

      echo "<p><strong>No orders pending.
            Please try again later.</strong></p>";
      exit;
    }

    while (!feof($fp)) {
       $order= fgets($fp, 999);
       echo $order."<br />";
    }
?>
</body>
```

这段脚本是按照前面所介绍的步骤进行的：打开文件、读文件、关闭文件。这段脚本在读取程序清单2-1所示数据后的运行结果如图2-4所示。

下面，我们详细介绍这个脚本中用到的函数。

2.7.1 以只读模式打开文件：fopen()

仍然使用fopen()函数打开文件。在这个例子中，以只读模式打开这个文件，所以使用了"rb"文件模式：

```
$fp = fopen( '$DOCUMENT_ROOT/../orders/orders.txt", 'rb');
```

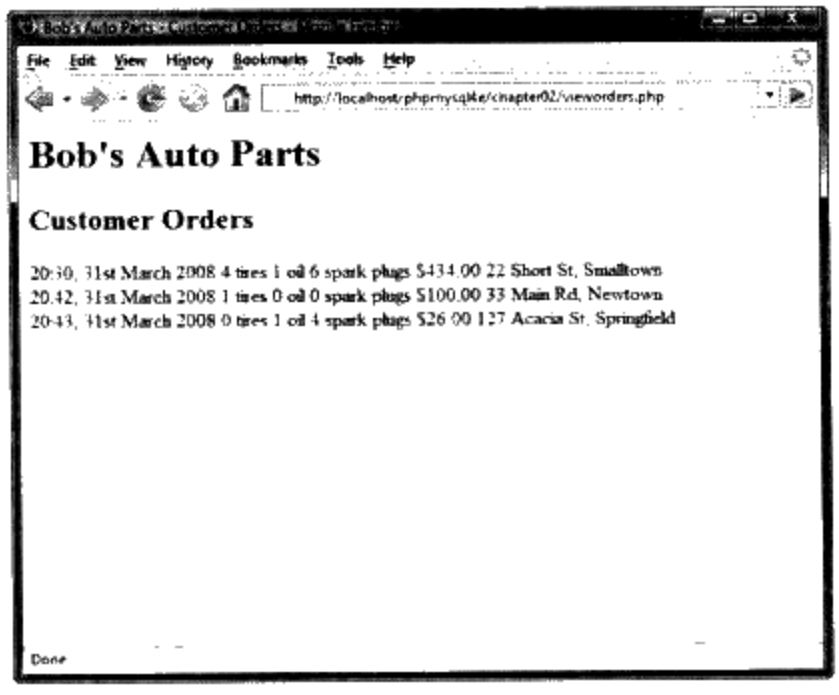

图2-4　vieworders.php在浏览器窗口中显示orders.txt文件当前的订单

2.7.2　知道何时读完文件：feof()

在这个例子中，使用了while循环来读取文件内容，直到文件末尾。在这个while循环语句中，使用feof()函数作为文件结束的测试条件：

```
while (!feof($fp))
```

函数feof()的唯一参数是文件指针。如果该文件指针指向了文件末尾，它将返回true。虽然这个函数名称看上去有点古怪，但是如果知道feof表示File End Of File，就会很容易记住它。

在这个例子（通常是在读文件的时候）中，我们持续进行读文件操作，直至遇到EOF。

2.7.3　每次读取一行数据：fgets()、fgetss()和fgetcsv()

在这个例子中，使用fgets()函数来读取文件内容：

```
$order = fgets($fp, 999);
```

这个函数可以从文件中每次读取一行内容。这样，它将不断地读入数据，直至读到一个换行字符（\n）、或者文件结束符EOF，或者是从文件中读取了998B。可以读取的最大长度为指定的长度减去1B。

也可以使用许多不同的函数来读文件。当需要按块方式处理一些纯文本文件时，fgets()函数将会非常有用。

fgets()函数的一个非常有趣的变体是fgetss()函数，其函数原型如下所示：

```
string fgetss(resource fp, int length, string [allowable_tags]);
```

fgetss()函数与fgets()非常相似，但是它可以过滤字符串中包含的PHP和HTML标记。如果要过滤任何特殊的标记，可以将它们包含在allowable_tags字符串中。当读取由别人所编写的文件或者包含用户输入的文件时，出于操作安全的考虑，可以使用fgetss()函数。

允许无限制的HTML代码出现在文件中可能会破坏你精心设计好的格式。允许无限制的PHP代码出现在文件可能会让恶意用户以几乎自由的方式控制服务器。

fgetcsv()函数是fgets()的另一个变体。它具有如下所示的函数原型：

```
array fgetcsv ( resource fp, int length [, string delimiter
                [, string enclosure]])
```

当在文件中使用了定界符时，例如我们在前面所介绍的制表符或者在电子制表软件和其他应用程序中使用的逗号，可以使用fgetcsv()函数将文件分成多行。如果希望重新构建订单中的变量，而不是将整个订单作为一行文本，使用fgetcsv()函数可以很容易实现。可以像调用fgets()一样调用它，但是必须向这个函数传递一个用于分隔表单域的定界符。

例如：

```
$order = fgetcsv($fp, 100, "\t");
```

以上代码将从文件中读取一行，并且在有制表符（\t）的地方将文件内容分行。该函数结果将返回一个数组（在以上代码就是$order）。我们将在第3章详细介绍数组。

参数*length*应该比要读的文件中最长数据行的字符数大。

*enclosure*参数用来指定每行中封闭每一个域的字符。如果没有指定任何字符，在默认情况下，这个字符就是"（双引号）。

2.7.4 读取整个文件：readfile()、fpassthru()和file()

除了可以每次读取文件一行外，还可以一次读取整个文件。PHP提供了4种不同的方式来读取整个文件。

第一种方式是readfile()。你几乎可以使用如下一行语句来代替前面所编写的所有脚本：

```
readfile("$DOCUMENT_ROOT/../orders/orders.txt");
```

调用readfile()函数将打开这个文件，并且将文件内容输出到标准输出（浏览器）中，然后再关闭这个文件。readfile()的函数原型如下所示：

```
int readfile(string filename, [int use_include_path [, resource context]] );
```

第二个可选参数指定了PHP是否应该在include_path中查找文件，这一点与fopen()函数一样。可选的context参数只有在文件被远程打开（例如通过HTTP）时才使用；我们将在第20章详细介绍这种用法。这个函数的返回值是从文件中读出的字节总数。

第二种方式是fpassthru()。要使用这个函数，必须先使用fopen()打开文件。然后将文件指针作为参数传递给fpassthru()。这样就可以把文件指针所指向的文件内容发送到标准输出。然后再将这个文件关闭。

可以使用如下代码替代前面的脚本：

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", "rb");
fpassthru($fp);
```

如果读操作成功，fpassthru()函数将返回true，否则返回false。

第三种读取整个文件的函数是file()。除了可以将文件内容回显到标准输出外，它和

readfile()是一样的，它是把结果发送到一个数组中。我们将在第3章介绍数组时详细介绍。作为参考，可以按如下方式调用它：

```
$tilearray = file($DOCUMENT_ROOT/../orders/orders.txt ");
```

这行代码可以将整个文件读入到一个名为$filearray的数组中。文件中的每一行都将作为一个元素保存在这个数组中。请注意，在PHP的早期版本中，该函数对二进制文件并不是安全的。

第四种选择是使用file_get_contents()函数。这个函数与readfile()相同，但是该函数将以字符串的形式返回文件内容，而不是将文件内容回显到浏览器中。

2.7.5 读取一个字符：fgetc()

文件处理的另一个方法是从一个文件中一次读取一个字符。可以使用fgetc()函数来实现。它具有一个文件指针参数，这也是该函数的唯一参数，而且它将返回文件的下一个字符。我们可以使用具有fgetc()函数的循环来代替原来脚本中的while循环，如下所示：

```
while (!feof($fp)){
  $char = fgetc($fp);
  if (!feof($fp))
    echo ($char== "\n "? '<br /> ": $char);
  }
}
```

这段代码使用fgetc()函数从文件中一次读取一个字符，并且将该字符保存在$char中，直到文件结束。然后再用HTML的换行符（
）替换文本中的行结束符（\n）。

这样做仅仅是为了整理输出格式。如果输出文件的记录之间带有\n，那么整个文件将显示在一行中（试一下就会知道）。Web浏览器并不会表示空格，例如新行。因此你必须用HTML的换行符（
）替换文本中的行结束符（\n）。我们可以使用三元运算符来完成此操作。

使用fgetc()函数的一个缺点就是它返回文件结束符EOF，而fgets()则不会。读取出字符后还需要判断feof()，因为我们并不希望将文件结束符EOF回显到浏览器中。

如果不是为了某些原因需要对文件逐个字符进行处理，这种逐个字符读取的方法现实意义并不大。

2.7.6 读取任意长度：fread()

读取一个文件的最后一种方法是使用fread()函数从文件中读取任意长度的字节。这个函数的原型如下所示：

```
string fread(resource fp, int length);
```

使用该函数时，它或者是读满了*length*参数所指定的字节数，或者就是读到了文件末尾或网络数据包的结束。

2.8 使用其他有用的文件函数

在PHP中，还有许多我们经常使用的有用的文件函数。

2.8.1 查看文件是否存在：file_exists()

如果希望在不打开文件的前提下，检查一个文件是否存在，可以使用file_exists()函数，如下所示：

```
if (file_exists("$DOCUMENT_ROOT/../orders/orders.txt")) {
    echo 'There are orders waiting to be processed.';
} else {
    echo 'There are currently no orders.';
}
```

2.8.2 确定文件大小：filesize()

可以使用filesize()函数来查看一个文件的大小。如下所示：

```
echo filesize( "$DOCUMENT_ROOT/../orders/orders.txt ");
```

它以字节为单位返回一个文件的大小，结合fread()函数，可以使用它们一次读取整个文件（或者文件的某部分）。可以用如下代码来替换以前的代码：

```
$fp = fopen('$DOCUMENT_ROOT/../orders/orders.txt', 'rb');
echo nl2br(fread( $fp, filesize('$DOCUMENT_ROOT/../orders/orders.txt')));
fclose( $fp );
```

nl2br()函数将输出的\n字符转换成HTML的换行符（
）。

2.8.3 删除一个文件：unlink()

在处理完订单后，可能希望删除这个订单文件，可以使用unlink()函数。（PHP中没有名为delete的函数。）例如：

```
unlink("$DOCUMENT_ROOT/../orders/orders.txt");
```

如果无法删除这个文件，该函数将返回false。通常，如果对该文件的访问权限不够或者该文件不存在，该函数将返回false。

2.8.4 在文件中定位：rewind()、fseek()和ftell()

可以使用rewind()、fseek()和ftell()对文件指针进行操作，或者确定发现它在文件中的位置。

rewind()函数可以将文件指针复位到文件的开始。ftell()函数可以以字节为单位报告文件指针当前在文件中的位置。例如，我们可以在初始脚本的结束处添加如下几行代码（在fclose()命令之前）：

```
echo 'Final position of the file pointer is'.(ftell($fp));
echo '<br />';
rewind($fp);
echo 'After rewind, the position is'.(ftell($fp));
echo '<br />';
```

该脚本在浏览器中的输出结果类似于图2-5所示。

你也可以使用fseek()函数将文件指针指向文件的某个位置。其函数原型如下所示：

```
int fseek ( resource fp, int offset
, int whence])
```

调用fseek()函数可以将文件指针*fp*从*whence*位置移动*offset*个字节。*whence*是一个可选参数，其默认值*SEEK_SET*表示文件的开始处。该参数的其他可能值为*SEEK_CUR*（文件指针的当前位置）和*SEEK_END*（文件的结束）。

rewind()函数等价于调用一个具有零偏移量的fseek()函数。例如，可以使用fseek()函数找到文件中间的记录，或者完成一个二进制查找。通常，如果所涉及的数据文件具有一定的复杂程度，在必须完成这些操作时，使用数据库可以使得这些工作变得更加简单。

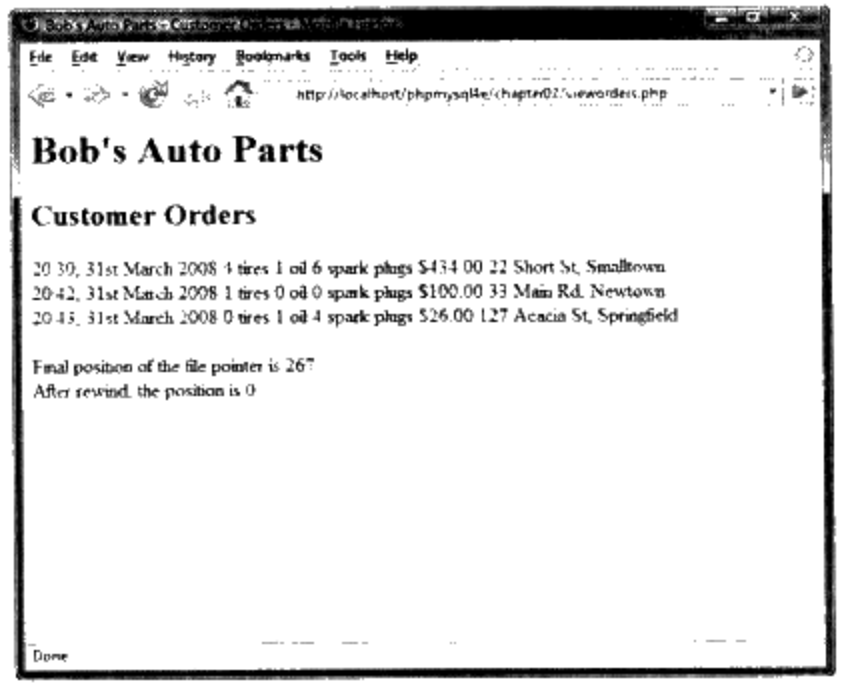

图2-5　在读取这些订单后，文件指针指向了文件的结尾，总共267字节的偏移量。调用rewind()函数将文件指针重置为0，位于文件的开始处

2.9　文件锁定

假设遇到这种情况，两个客户试图同时订购同一件商品（这种情况并不少见，尤其是当网站上遇到某种程度的网络堵塞的时候。）如果一个客户调用fopen()函数打开一个文件并且开始写这个文件，而此时其他客户也调用了fopen()函数打开这个文件并且要写这个文件，将会出现什么情况呢？文件的最终内容是什么？会是第一个订单后面就是第二个订单吗？还是恰好相反呢？订单是第一个客户的还是第二个客户的？或者将变成一些没用的东西，就像两个订单交错在一起？这些问题的答案取决于操作系统，但是，通常都是不可知的。

为了避免这样的问题，可以使用文件锁定的方法。在PHP中，文件锁定是通过flock()函数来实现的。当一个文件被打开并且在进行读写操作之前，应该调用这个函数。

flock()函数原型如下所示：

```
bool flock (resource fp, int operation [, int & wouldblock ])
```

还必须将一个指向被打开文件的指针和一个表示所需锁定类型的常数作为参数传递给这个函数。如果文件锁定成功，其返回值为true，否则为false。如果获得文件锁将导致当前的进程被阻塞（也就是，不得不等待），可选的第3个参数将包含值true。

operation参数的可能值如表2-2所示。在PHP 4.0.1中，该参数的可能值已经发生了变化，因此表2-2给出了两个值集合。

如果打算使用flock()函数，必须将其添加到所有使用文件的脚本中；否则，就没有任何意义。

表2-2 flock()的操作值

操 作 值	意 义
LOCK_SH（以前为1）	读操作锁定。这意味着文件可以共享，其他人可以读该文件
LOCK_EX（以前为2）	写操作锁定。这是互斥的。该文件不能被共享
LOCK_UN（以前为3）	释放已有的锁定
LOCK_NB（以前为4）	防止在请求加锁时发生阻塞

请注意，flock()函数无法在NFS或其他网络文件系统中使用。它还无法在其他更早不支持文件锁定的文件系统中使用，例如FAT。在某些操作系统中，它是在进程级别上实现的，因此，如果你在多线程服务器API中使用，该函数也无法正确使用。

要在这个例子中使用flock()函数，我们可以对processorder.php脚本进行如下所示的修改：

```
$fp = fopen( "$DOCUMENT_ROOT/../orders/orders.txt", 'ab');
flock($fp, LOCK_EX); // lock the file for writing
fwrite($fp, $outputstring);
flock($fp, LOCK_UN); // release write lock
fclose($fp);
```

你还应该在vieworders.php脚本中添加如下所示的文件锁：

```
$fp = fopen( "$DOCUMENT_ROOT /../orders/orders.txt",  r );
flock($fp, LOCK_SH); // lock file for reading
// read from the file
flock($fp, LOCK_UN); // release read lock
fclose($fp);
```

现在，我们的代码更加健壮，但是还不完美。如果有两个脚本同时申请对一个文件加锁，情况又会如何呢？这将导致竞争条件的问题，这两个进程将竞争加锁，但是无法确定哪一个进程将会成功，这样就会导致更多的问题。使用数据库管理系统（DBMS），我们可以很好地解决这个问题。

2.10 更好的方式：数据库管理系统

到目前为止，在我们的例子中所使用的文件都是普通文件。在本书的第二篇中，我们将介绍如何使用MySQL，它是一个关系数据库管理系统（RDBMS）。你可能会问，“我为什么要使用它？”

2.10.1 使用普通文件的几个问题

使用普通文件，你可能会遇到如下这些问题：

- 当文件变大时，使用普通文件将会变得非常慢。
- 在一个普通文件中查找特定的一个或者一组记录将会非常困难。如果记录是按顺序保存，你可以使用某种二分法并结合按定长记录来搜索一个关键字域。如果你希望查找模式信息（例如，需要查找所有生活在Smalltown的客户），就不得不读入每一个记录并且进行

逐个检查。

- 处理并发访问可能会遇到问题。你已经了解了如何锁定文件，但是锁定可能导致我们前面介绍的竞争条件。它也可以导致一个瓶颈。如果一个站点具有太多的访问量，大量的用户就可能在能够创建订单之前必须等待该文件解锁。如果该等待时间太长，人们可能会到其他地方购买。
- 到目前为止，我们所看到的文件处理都是顺序的文件处理——也就是我们从文件开始处一直读到文件的结束。如果我们希望在文件中间插入记录或者删除记录（随机访问），这可能会比较困难——你将必须将整个文件读入到内存中，在内存中修改它，然后再将整个文件写回去。如果这是一个很大的数据文件，这可能会带来巨大的开销。
- 除了使用文件访问权限作为限制外，还没有一个简单的方法可以区分不同级别的数据访问。

2.10.2 RDBMS是如何解决这些问题的

关系型数据库管理系统（RDBMS）可以解决以上所有问题：

- RDBMS提供了比普通文件更快的数据访问。我们在本书中所使用的数据库系统MySQL在许多方面都拥有比任何RDBMS更快的速度。
- RDBMS可以很容易地查找并检索满足特定条件的数据集合。
- RDBMS具有内置的处理并发访问的机制。作为一位编程人员，你不必担心这一点。
- RDBMS可以随机访问数据。
- RDBMS具有内置的权限系统。MySQL在这一方面具有特别的优势。

使用RDBMS的主要原因是RDBMS实现了数据存储系统所必需的所有（或者至少是大多数）功能。当然，也可以编写你自己的PHP函数库，但是为什么不利用已有的功能呢？

在本书的第二篇将介绍关系数据库的基本工作原理，以及如何安装并使用MySQL来创建支持后台数据库的Web站点。

如果要创建一个简单的系统而又觉得不需要一个功能全面的数据库，但是又希望避免锁定和其他与使用普通文件相关的问题，你可能会考虑使用PHP的SQLite扩展。这个扩展对普通文件提供了一个基本的SQL接口。在本书中，我们的重点是使用MySQL，但是如果希望获得更多关于SQLite的信息，可以在http://sqlite.org/和http://www.php.net/sqlite找到。

2.11 进一步学习

关于与文件系统进行交互的更多信息，可以参阅第19章。在该部分中，我们将详细介绍如何修改文件权限、属主和名称；如何使用目录以及如何与文件系统环境进行交互。

如需阅读PHP在线手册中关于文件系统的介绍，可以在http://www.php.net/filesystem中找到。

2.12 下一章

在下一章中，我们将介绍什么是数组及如何在PHP脚本中使用它们来处理数据。