第18章

排 除 故 障

费了很大的劲编写出来的程序，却只生成了令人困惑、意料不到的错误，显示了一串不能理解的错误信息，没有比这更头痛的了。

许多开发人员都经历过这种头痛。所以本章将讨论一些读者可能遇到的问题，提出几个排除故障的策略。

18.1 安装故障的排除

假定要访问 PHP、MySQL 或 Apache，但却遇到了许多问题。可能它们由于某种原因不能很好地合作，并显示出奇怪的错误。也可能它们没有按照安装指令来工作。

在许多情况下，常见的错误或故障都在 AMP 中每个组件的网站上讨论过了。网站还为用户使用的系统提供了详细的指导，读者最好仔细阅读它们，确保完全按照指令来安装。

如果配置 PHP 时遇到服务器找不到某个库的错误，建议检查如下内容：

- 验证在配置命令中指定了正确的路径
- 确保在机器上安装了该库及其所有从属文件
- 确保在修改 php.ini 或 httpd.cof 文件后，重启动了 Apache 网络服务器(修改.htaccess 文件不需要重启动)

18.2 解析错误

读者肯定常常见到下面的信息：

```
Parse error: parse error, expecting "," or ";" in /foo/public_html/forum/
index.php on line 25
```

这是该死的解析错误。这很常见，甚至即使是有经验的编程人员也常常遇到它们。即使使用最好的彩色编码 PHP 文本编辑器来帮助检查语法，也肯定会漏掉一两个解析错误。这些错误非常令人沮丧，但它们通常是最容易改正的错误，因为它们常常是由于语法错误引起的，而不是逻辑错误。可以通过检查是否遗漏了分号、逗号，或引号的位置有错误来

改正这些错误。

18.2.1 清理第 16 行，但其实第 94 行有错

PHP 显示解析错误时，会包含行号，这提供了解决问题的第一个线索。但有时行号也会误导我们。实际上，有时有错误的行会出现在服务器指定的行的前面几行。

例如，考虑某个遗漏的分号，由于没有分号告诉服务器语句已结束，服务器就会把后续的行串在一起。服务器在数行代码后才发现问题，于是就在错误的行上发出一个解析错误。遗漏了引号或括号，也会出现相同的情况。例如下面的代码(已经添加了行号，以演示上述情况)：

```
1 <?php
2 $greeting1="aloha";
3 $greeting2="bon jour";
4 $greeting3="hola"
5 $greeting4="good morning";
6 ?>
```

运行这个测试代码，得到的错误如下：

```
Parse error: parse error, unexpected T_VARIABLE in C:\Program files\Apache
Software Foundation\Apache2.2\HTDOCS\error.php on line 5
```

为了便于演示，把上述脚本命名为 error.php，可以看出，服务器指出第 5 行有错，但实际上错误发生在第 4 行。因为我们在第 4 行末遗漏了分号，所以第 5 行就被看作第 4 行的续行，因此，PHP 被搞糊涂了。

18.2.2 基本常识

有时最简单的答案就是正确的答案。确保完成了如下所有的工作：

- 每个语句都以分号结尾
- 所有的左引号、括号和花括号都有对应的右引号、括号和花括号匹配。
- 所有的单引号和双引号都正确嵌套和/或转义。

如果在编写代码时，定期检查语法，就会大大减少解析错误。可以使用熟悉的 PHP 编辑器，在编写代码时，给程序显示彩色代码。在拼错了函数名或者忘记关闭引号时，语法的突出显示就会使我们更容易识别这些错误。我们用一个表格来比较了各种文本编辑器，其中一些支持对语法的突出显示，这个表格参见附录 E。

18.3 空变量

我们建立了一个很大的页面，从用户处收集了 50 个字段的信息，其中没有解析错误。然后在线填充表单，并点击提交按钮。下一个页面也正常加载，但唯一的问题是似乎没有把变量传送到新表单上。

这其实很常见。第一个可能的原因是希望传送值，但忘记在表单上使用 method="post"。因为表单默认使用的是 get 方法。

如何解决这个问题？检查一下第二个页面的地址。这些变量在查询字符串中吗？如果在，就肯定在使用 GET 方法，此时就需要返回，把方法改为 POST。问题解决了。

18.3.1　一致而有效的变量名

首先应根据命名规则，确保变量名是合适而有效的，如第 2 章所述。应该确保所有变量名都不以数字开头，也不应使用预定义的变量作为变量名，例如$php_errormsg。预定义变量的完整列表可以在 PHP 手册 www.php.net/manual/en/reserved.variables.php 中找到。

另外应检查引用变量时名字的大小写，因为变量名是区分大小写的。数据库和表名也是如此。一定要确保正确、一致地引用它们。如果修改了某个变量名，就应修改该变量名的所有实例。

如果在脚本中一直遵循某个命名规则，就很容易保证变量名的一致性。这与第 2 章中讨论的优秀编码规则有关。

18.3.2　打开一个新浏览器

有时，如果在会话中工作，且处于脚本的测试阶段，就有外来的会话设置妨碍我们获得希望的结果，并修改变量值。

只有关闭网络浏览器，再打开一个新的浏览器，才可以清除所有的会话变量(假定没有修改配置文件，如第 2 章所述)。

18.4　“标题已发送”错误

用户可能会遇到如下错误消息：

```
Warning: Cannot modify header information – headers already sent by (output
started at C:\Program files\Apache Software Foundation\Apache2.2\htdocs\
headererror.php: 1) in C:\Program files\Apache Software Foundation\Apache2.2\
htdocs\headererror.php on line 2
```

在使用会话和 cookie 时，这是一个常见错误。如果试图在给服务器发送 HTML 代码后设置它们，就会出现这个错误。服务器必须在给浏览器发送 HTML 输出之前处理会话和 cookie，也就是说，这些代码行必须放在 HTML 代码或 echo 语句的前面。甚至在第一行代码<?php 之前有一个前导空格，也会出现这个错误。

如果需要在代码体中设置会话和 cookie 变量，就需要重新思考逻辑，以包容这个限制。如第 2 章所述，这些变量必须放在代码的开头，才能由 PHP 服务器正确解析。

可以使用输出缓冲区来抑制该错误。输出缓冲区可以在缓冲区中存储所有的 HTML 输出，直到准备把它们发送到浏览器上为止。使用函数 ob_start()可以开始输出缓冲的过程，ob_end_flush()则可以把所有存储的 HTML 输出发送给浏览器，清空缓冲区，结束输出存储

过程。这样就可以欺骗系统，把会话和 cookie 变量存储在代码体中，也允许在代码体中使用 header()函数。例如，下面的代码就通过使用输出缓冲区来抑制错误。在下面的例子中，header.php 文件包含了连接 MySQL 数据库的连接变量和站点上所有页面都有的一些 HTML 代码：

```
<?php
ob_start()
include 'header.php'
//perform a mysql query to determine which page the user is supposed to see;

if($userage<18){
  header('Location: child.php');
} else {
  header('Location: adult.php');
}
ob_end_flush();
?>
```

如果不使用函数 ob_start()和 ob_end_flush()，则在重定向用户时，就会出现“标题已发送”错误。这是由 header.php 文件中的代码造成的。可以看出，是逻辑出现了错误，因为我们把连接变量放在了一个单独的文件中，而没有和 HTML 代码放在一起，但这不是使网站不工作的致命设计错误。因此我们可以欺骗系统。

初学者最好不要这么做，因为学习编写好的代码和遵循规则是最重要的，但比较有经验的编程人员则可以使用这组有用的函数。如果要学习更多的输出缓冲区函数，可以参考附录 C 中的完整列表，或者访问 www.php.net。

18.5 一般调试提示

即使遇到最困难的错误我们也可以通过一定的方法，用无比的耐心来将其改正。毕竟这只是代码，没有什么可怕的。对于一般的调试，下面给出了一些提示，有助于解决许多讨厌的错误。

18.5.1 使用 echo

显然，我们希望知道服务器的想法，看看它是如何考虑问题的。一种方式是在代码中定期显示变量的内容，来验证服务器是否正确解析了代码。

可以在使用变量的过程中通过 echo 逐步显示变量的值，看看服务器如何在代码中处理值。如果要对变量的值执行复杂的数学计算，但输出了错误的答案，这种方法就会很有帮助。我们需要找出问题出在哪里，所以在数学计算的每一步中插入 echo 语句，验证服务器在执行数学计算时是否正确。于是，我们会看到不断变化的变量值。

echo 命令还可以用于 if 语句、foreach 语句、函数等，以确保这些循环正确调用或处理。

下面是一个简单的例子，演示了 echo 如何帮助我们。假定有如下脚本：

```
<?php
$curr_var=0;

while ($curr_var<20){
  $abc=2*$curr_var;
  $curr_var ++;
}
echo $abc;
?>
```

在浏览器上运行这段代码，就得到数字 38。如果希望得到是 40，或者想查看$abc 变量是否正确，就可以通过在处理过程中显示变量的值，来检查程序是如何工作的，于是：

```
<?php
$curr_var=0;

while ($curr_var<20){
  $abc=2*$curr_var;
  $curr_var ++;

  //debig line
  echo $curr_var . '<br/>';
}
echo $abc;
?>
```

现在就会看到数字 1～20 和最初的答案 38。很容易看出，尽管$curr_var 达到了 20，但答案只处理了 19 次，所以会得到 38。此时，应把 while 语句改为：

```
while ($curr_var<=20){
```

现在当$curr_var=20 时，while 语句也会处理，所以结果为 40。可以使用注释来提醒自己，在解决了问题后删除调试行，以避免在页面激活后给浏览器输出不想要的结果。

数组和对象引用虽然是变量，但显示它们时有点区别。例如，如果使用 echo 输出一个数组，屏幕上就只会显示 Array()。所以要查看数组的内容，应使用 print_r()而不是 echo。print_r()会输出数组的每个成员，甚至多维数组也是如此。对于对象引用，print_r()会显示对象的所有成员。可以使用的另一个类似函数是 var_dump()。

18.5.2　分而治之

处理大问题的另一个好方法是把问题分解为基本的步骤，通过测试每个步骤，来确保每一步都得到正确的结果。在复杂语句块的开头，一个小错误可能导致滚雪球效应，最终完全改变结果。所以逐个步骤地检查，就可以找出这些小错误，最终得到希望的结果。有时我们可以注释掉一段代码，看看脚本没有这段代码的运行情况，或者隔离出某个有问题的代码段来看看脚本的运行情况。

18.5.3 测试、测试、再测试

许多编码人员都会在自己的系统上测试代码，只要这些代码可以在他们的设置下正常工作，他们就认为代码没有错误。我们应尽可能使用每个不同的环境来测试代码，即不同的浏览器、不同的配置、不同的计算机系统等，以达到全面测试代码的目的。如果知道相应的方法，甚至可以通过攻击自己的系统，来查找出某个对自己不太友好的人可能利用的安全漏洞。

18.5.4 用 Xdebug 调试

PHP 有非常多的内省函数，可帮助我们调试系统，如 var_dump()和 print_r()。另外，还有 Xdebug，这是 PHP 的一个强大扩展，它添加了额外的调试和配置功能。Xdebug 可以查看堆栈跟踪(PHP 执行各种函数，到达某个点的路径)和代码覆盖信息，甚至可以与理解 DBGp 协议的调试客户机一起交互式调试脚本。

如前所述，Xdebug 是一个扩展，默认不能用作 PHP 的一部分。必须正确安装和配置 Xdebug，才能使用它的功能。Xdebug 网站 www.xdebug.org 上有 Xdebug 的更多信息。

18.6 寻求帮助

PHP、Apache 和 MySQL 的用户社区非常有活力。在其中可以在线获得许多资源，并在遇到问题时获得帮助。前面曾经多次提到其中的一些资源，这里再次提及它们。

18.6.1 www.wrox.com

本书在一个附加的网站中提供了在线帮助，如果遇到问题，建议访问本书的姊妹网站 www.wrox.com。

18.6.2 PHPBuilder.com

网上有许多 PHP 帮助网站，但这里推荐 PHPBuilder.com。在这个网站上，有许多文章、归档文件、有用的代码片段，最重要的是有一个发展成熟的、非常有帮助的在线社区，其中的编码人员来自世界各地，有不同的能力，用户可以很快地获得帮助。其他地方也有活跃的、友好的社区，用户可以在他们的论坛上发贴子。

本书的一位作者也常常光顾 PHPBuilder.com，我们都是常规的撰写人，其中一些是协调员。

18.6.3 源网站

我们曾经多次提到这个建议，但与其他建议一样，无论怎样强调这个建议都不过分。只要有问题，答案就可能在源网站上找到。这些网站提供了非常全面的手册，包括软件的

几乎所有信息。

下面再复习一下这些网站：

- PHP：www.php.net(提示：如果要查找函数的帮助，例如 echo，就可以在浏览器中输入 www.php.net/echo，这样就会进入 echo 页面)

PHP 还提供了 Microsoft Windows Help(CHM)格式的手册，这对 Windows 用户非常有用。可以从 www.php.net 上下载该手册，并在本地机器上安装它。

- Apache：httpd.apache.org
- MySQL：www.mysql.com

18.6.4　搜索和求助

如果在脚本中遇到问题，但并不是第一次遇到该问题，就可以使用搜索引擎，查找 Internet 上的文章、论坛上的贴子、教程或讨论该问题的其他信息。这是一种非常快速、方便的方式，不需要推倒重来。

18.6.5　IRC 通道

如果需要问题的即时帮助，IRC 资源就是很好的解决方案。有许多 PHP IRC 通道：例如，quakenet 网络上的#php 和#phphelp 就是其中的两个。

18.7　小结

错误总是会发生的，但在遇到错误时，一定不要气馁，而应耐心地检查代码，找出错误。希望读者在遇到困难时，本章给出的提示能有助于读者调试程序，从而改正其中的错误。