

第7章 调试和错误处理

前面已经介绍了使用ASP所需要的基本技能，本章要讨论的另外一个问题是当ASP出现错误时怎么办，ASP出错时是什么情况。当精心编排的ASP页面出现问题停止了执行时，用户一般得到的仅是一些用处不大的建议，诸如：点击“刷新”按钮，或者“与站点的Web管理员联系，告诉他们你的页面不能正常工作了”等等。

本章除了提供有用的信息源之外，还想提供一个帮助区域。我们将详细介绍在脚本和页面中错误如何出现，可能产生的错误类型，以及什么造成了这些错误。更重要的是，要讨论如何尽可能避免错误的发生，如不能避免又如何妥善处理。

因此，本章将要探讨页面调试技术，也就是如何花费不多的精力和时间就能找到错误并解决问题。

本章包括以下内容：

- 能够出现的错误类型。
- 如何防止各种错误的产生。
- 如果不能防止错误发生，如何妥善处理这些错误。
- 如何发现和处理脚本错误及其他类型的错误。
- 如何使用定制的错误页面得到错误信息。
- 如何记录发生的错误以监视我们的网站。
- 创建一个定制错误网页和一个错误日志文件。
- 提供相关的在线帮助。

本章不涉及如何处理使用ActiveX Data Objects(ADO)访问数据源时出现的各种特殊类型的错误。像许多组件一样，ADO提供了自己的错误处理系统，第8章将深入讨论这一点。本章将从讨论能出现的各种错误类型开始，使我们能够认识这些错误并采取相应的措施。

据说，在非洲最黑暗的雨林深处，有这样一群程序员，他们的程序代码从来没有出现过错误。但是，很遗憾他们从没有享受过调试一段不能正常工作的应用程序的乐趣。调试程序代码是一个真正充满快乐的工作，所以我们要面对这个问题，在调试程序的过程中检验我们的观察力和横向思维能力。大多数“真实世界”的程序员能够体验这些乐趣是一件好事。

当然，有些人会说，调试程序与其说是判断，不如说是碰运气。花费了许多时间去调试一段有错误的程序，在某种程度上可以说确实是依赖运气。但是，如果第一步从合适的地方开始查看，可能会更快地解决问题。

但这不是程序调试应采取的办法。从理论上讲，当某段程序运行失败时，应该以逻辑或顺序方式跟踪错误。作为一个聪明和有经验的程序员，这才是调试时常用的方法，只有业余人员才随意改变程序中变量的值，到处添加Response.Write语句进行调试。

然而，为了能够在逻辑上跟踪程序中的错误，必须了解有关错误如何出现方面的基础知识，更重要的是知道错误出现在哪里，以便很快就能找到相应的地方。本章讨论的内容是有关程序中能够出现的不同种类的错误，错误的不同表现，以及如何记录和排除这些错误。同

样重要的是，还将介绍如何避免这些错误的发生。

本章将从介绍可能出现的不同种类的错误开始，如果认为你的代码不会出现任何错误，可以直接跳到下一章。

7.1 错误的种类

7.1.1 语法或“编译”错误

当我们第一次运行新编写的程序代码时，通常看到的第一种错误类型是“syntax error”。这就是所说的，程序代码上的语法错误。这就像在写作中使用了错误的语法，使读者不能了解其中的含义。而解释器（诸如脚本引擎）和编译器对语法要求得更加严格和准确。

语法错误通常也是最早出现和需要排除的。大多数情况下，解释器和编译器会指出行号和所在行中的字符位置，以及在相应的位置上缺少的内容。下面举一个简单的例子，如下所示的这样一段程序：

```
<%  
Response.Write "The repayments for your loan are $" & curPayment _  
    & " per " & strInterval & ", due on the " & strDay & " of each "  
    & strInterval & "."  
%>
```

我们希望得到下面的结果：

The repayments for your loan are \$124.50 per month, due on the 12th of each month.

实际上得到的结果如图7-1所示。

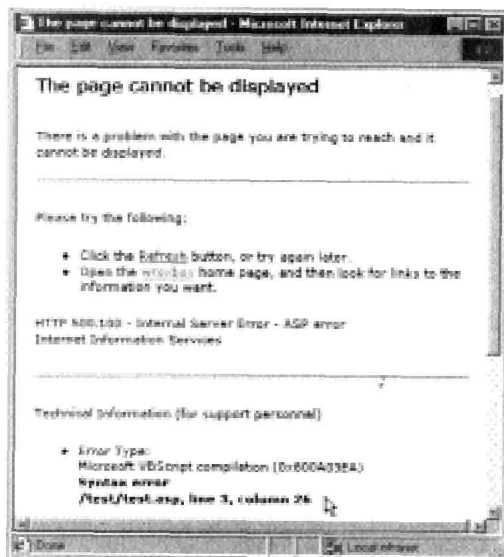


图7-1 程序执行结果1

文件中第3行是Response.Write 语句的第2行。报告错误信息时，VBScript解释器忽略一行中的引导空格和制表符。所以在数完 26个字符之后，可以找到语法错误的地方，这里明显缺少了一个双引号。加上双引号后再运行这个页面，我们可以得到如图7-2所示的画面。

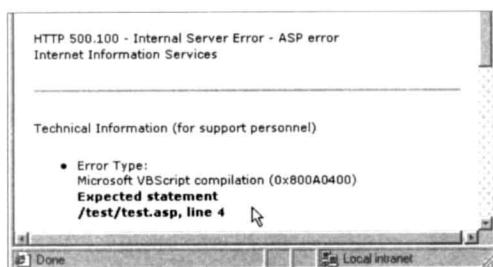


图7-2 程序执行结果2

这次又是另外一个简单错误。实际上错误出现在第 3 行而不是第 4 行。我们漏掉了第三行末尾的续行符'_'。程序代码应该是：

```
<%  
Response.Write "The repayments for your loan are $" & curPayment _  
    & " per " & strInterval & ", due on the " & strDay & " of each " _  
    & strInterval & "."  
%>
```

1. 错误出现在什么地方

需要注意的是脚本解释器仅指出所发现错误的地方，但实际上那儿并不一定是错误真正出现的地方。在上例中，前面三行的语法正确的，并产生相应的输出结果，而恰恰是第 4 行引起问题，因为这一行是以一种非法字符开头的，脚本解释器没有意识到这一行是上一行的一部分。

这样的错误是普遍存在的，因为通常我们主要考虑的是要输出的文本内容，而不是双引号、连字符(在VBScript中为"&")、续行符等的正确顺序。

对于关键字、内部函数名拼写错误或函数的非法参数列表而引起的语法错误，通常比较容易发现，因为错误信息提示可能就指出了错误的实际位置。例如：下面这段代码是想把明天的日期写入页面。

```
Response.Write DateAdd(Now(), "d", 1)
```

实际得到结果的如图 7-3 所示。

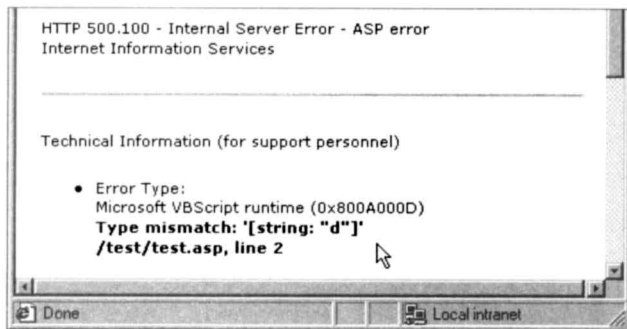


图7-3 程序执行结果3

这是因为DateAdd函数的语法应该是：

```
DateAdd (interval_string, interval_number, start_date)
```

所以应该改写为如下的代码：

```
Response.Write DateAdd("d", 1, Now())
```

脚本解释器检测到了我们为第二个参数提供的是一个字符型数据，而 DataAdd 函数需要的是整型数据类型。

代码结构和脚本结构

语法错误的另一个原因是：当制作网页时使用嵌套的或复杂的脚本结构，如 If Then...Else ...End If 或者 Do While...Loop。这有时会造成难以找到的语法错误。

例如下面这段程序：

```
<%  
If Len(Request.Form("cmdSet")) Then  
    strCounterName = Request.Form("lstSet")  
    strNewValue = Request.Form("txtSet")  
    If IsNumeric(strNewValue) Then  
        intNewValue = CInt(strNewValue)  
        objCounters.Set strCounterName, intNewValue  
        Response.Write "Set counter " & strCounterName & " to " & strNewValue  
    Else  
        Response.Write strNewValue & " is not a valid number"  
        If Len(Request.Form("cmdRemove")) Then  
            strCounterName = Request.Form("lstRemove")  
            objCounters.Remove strCounterName  
            Response.Write "Removed counter " & strCounterName  
        End If  
    End If  
End If  
>%
```

产生的错误如图 7-4 所示。

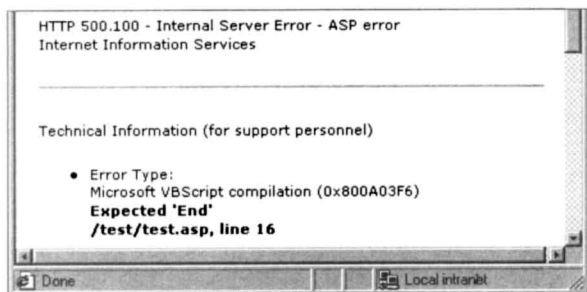


图7-4 程序执行结果4

为什么提示在网页程序中需要一个 End 语句呢？看一下程序就可以发现，丢失了一个 End If，而不是 End，在程序的最末尾应该还有另一个 End If。

```
...  
    Response.Write "Removed counter " & strCounterName  
End If  
End If  
End If  
>%
```

在这种情况下，根据代码的缩排格式可以很容易地找到相应的错误。特别当错误信息指出错误的大致位置时，很快就可以找到错误位置。然而，这段代码很短，如果在分界符 <%...%> 中另外还有 40 行代码，那么错误行号仍然可能指向最后一行 (line 56)；并且如果在新

的代码中的其他脚本结构搞乱了嵌套的结构，错误可能会指向另一个位置。

2. 关于JScript

如果你不是一位 JavaScript 高手，并且确实想试验一些语法错误，那么就从 VBScript 切换到 JScript。JScript 比 VBScript 对程序编写的要求更严格，并且对关键字和变量名大小写敏感，看下面的程序段。

```
<%  
var datToday = new Date();  
Response.Write(datToday.GetMonth());  
%>
```

运行这段程序会产生 "Object doesn't support this property or method" (对象不支持这种属性或方法) 错误，如图 7-5 所示。

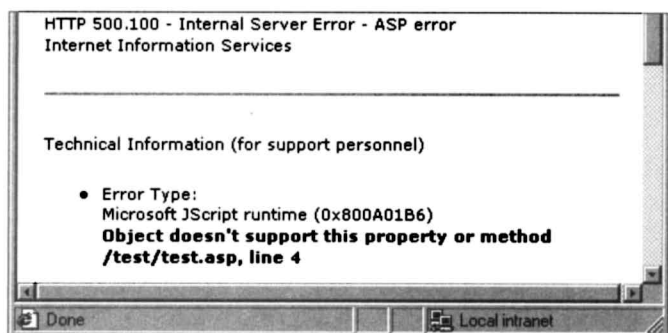


图7-5 程序执行结果5

原因很简单，返回目前月份数的 JScript 函数是 getMonth，而不是 GetMonth。下面这段程序就可以正常运行。

```
<%  
var datToday = new Date();  
Response.Write(datToday.getMonth());  
%>
```

当然，如果重试这段程序，可能得不到同样的错误消息。我们第一次运行这段程序时，得到如图 7-6 所示的错误。

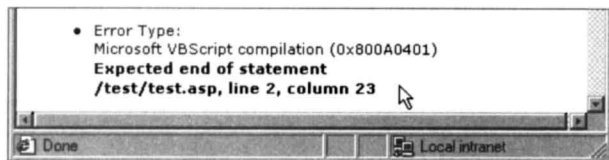


图7-6 程序执行结果6

第2行有什么错误？如果使用 JScript 解释器，没有错误出现。错误消息说明，这是一个 VBScript 语法错误。用 VBScript 解释器分析 JScript 程序，所以会得到奇怪的错误消息。

记住正在使用的语言

之所以出现上述错误是因为在页面的代码前面记了加 @LANGUAGE 指令。缺省是 VBScript (如果在注册表或在 Internet Services Manager 中没有改变它)，所以 VBScript 引擎用于处理前面不带 @LANGUAGE 指令的程序。即使一直使用专为自己的服务器设置的缺省语言，

始终使用 @LANGUAGE 指令是避免产生上述错误的好方法。这样，如果把网页移到另一个缺省语言不同的服务器上，也会得到预期的结果。

这里讲述的内容不可能覆盖所有可能遇到的语法错误，人们往往想知道为什么会出现错误，而错误信息提示并不总是像人们希望的那样准确。理想的方式应该是 ASP 给我们提供一个简洁的错误显示页面，有对错误的全面精确的描述，甚至询问我们是否想自动处理错误。

事实上应用程序 Microsoft Script Debugger 正试图为我们提供类似的功能，本章后面要对其进行讨论，也要概括避免出现语法错误的一些要点。现在，我们继续研究经常在网页中出现的第二类错误。

7.1.2 语义或“运行期”错误

语法错误的发现和处理是令人烦恼的，但在编程中会遇到一些真正“令人兴奋”的另一类型的错误——，语义错误 (semantic error) 或称“运行期”错误 (runtime error)。这类错误仅当运行一个脚本代码或其他程序时才会发现。换句话说，完整有效的代码已经通过解释器或编译器的解释或编译，在执行时产生了错误。术语“运行期错误”通常是指语义错误的结果，也就是说这类错误存在于代码中的语义中，当代码运行时它们才变成可见的。

这种区别来自于这种事实：程序编译器或解释器在处理程序代码之前必须建立一种内部代码的描述，涉及多种结构开头和结尾的匹配，以便标明每种结构包含什么内容，然后分析每个句子，以便知道如何执行这个句子。例如，如果在程序代码中有一个 IF Then . . . Else...End If 结构，解释器或编译器做的第一步工作就是分析哪些语句在“Then”的部分，哪些在“Else”部分。这一步的目的是，在对结构中的 IF 条件进行测试之后，可以决定该到哪个分支去执行。

编译器 (诸如在编程语言像 Visual Basic 和 C++ 中见到的那种) 和解释器 (诸如用于像 VBScript 和 JScript 那样的脚本语言的解释器) 之间真正区别在于：编译器不试图运行程序代码，而是在对源程序进行两次预处理后，形成二进制指令或符号代码，并形成一個 .exe 文件或 .dll 文件。解释器不创建含有代码的文件，而是在运行时逐行执行。

1. 使运行停止的错误

如果程序中含有一个语义错误，通常在运行时可得到提示。如果幸运的话，当错误发生时，程序会停止，这样可以容易地找出错误所在。例如，下面这段程序定义了一个有六个元素的数组。

```
<%  
Dim arrValues(5) 'to hold six elements, indexed from 0 to 5  
arrValues(6) = "Whoops, got an error"  
%>
```

如果试图读或设置下标为 6 的元素值，可以得到一个运行期错误，如图 7-7 所示。

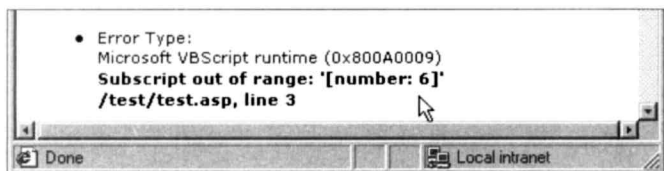


图7-7 程序执行结果6

注意这里的错误类型是“runtime”(相当于语义)错误,而不是语法错误。错误信息显示错误所在行数和错误的描述,有助于我们比较容易地找到相应的错误。但这只是一个简单的例子,在更复杂的程序代码中,这种错误可能出现在一些遍历一些值并把它们加到一个数组中的程序中。如下所示。

```
<%
Dim arrValues(5)                'to hold six elements
For intLoop = 0 To intListCount  'the number of items in some list
    arrValues(intLoop) = Request.Form("SelectedItems")(intListCount)
Next
%>
```

这种情况下,很可能是得到了过多的列表条目,或者是数组的索引不够,根据代码的要求,可以判断是那种错误,并且能够通过增加数组大小来解决这个错误。

```
<%
Dim arrValues(10)               'to hold eleven elements
For intLoop = 0 To intListCount 'the number of items in some list
    arrValues(intLoop) = Request.Form("SelectedItems")(intListCount)
Next
%>
```

或者相应地设置循环的参数来解决处理这个错误。

```
<%
Dim arrValues(5)                'to hold six elements
intArrayMax = intListCount
If intArrayMax > 5 Then intArrayMax = 5
For intLoop = 0 To intArrayMax  'only add the first six items
    arrValues(intLoop) = Request.Form("SelectedItems")(intListCount)
Next
%>
```

许多其他运行期错误能够使网页运行停止,诸如一些组件或对象的实例化失败,原因是ProgID错误,或者是因为组件没有正确安装。在这些情况下,结果总是给出“ActiveX Cannot Create Object”错误提示信息,后面跟着调用Server.CreateObject方法的行号。

2. 产生错误结果的错误

上面提到,如果遇到一个使程序代码停止的运行期错误,我们可能是幸运的。但是另一种情况是程序能很好地执行,好像什么也没有发生,最后产生一个错误的结果。这是最难发现和解决的错误,因为意识不到哪里出错了。例如,假设有一个网页,这个网页把用户的生日作为日期型的值,并且单独显示日期元素(可以把它们作为三个条目加到一个数据库中)。

```
<%
'get the value from the Request and display it
datBirthdate = Request.Form("BirthDate")
Response.Write "The value you entered is: " & datBirthdate & "<P>"

'get the individual date elements
intDay = Day(datBirthdate)
intMonth = Month(datBirthdate)
intYear = Year(datBirthdate)

'and display them
Response.Write "Day: " & CStr(intDay) & "<BR>"
Response.Write "Month: " & CStr(intMonth) & "<BR>"
Response.Write "Year: " & CStr(intYear) & "<BR>"
%>
```

图7-8是结果，是用美国日期风格月/日/年显示的，好像一切都没有问题。

然而如果输入一个非法日期，或者让输入文本框空着，便得到一个运行期错误，如图 7-9所示。

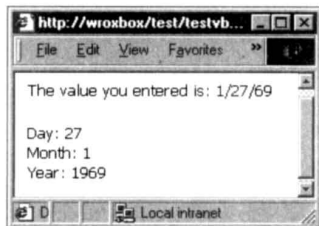


图7-8 显示生日的屏幕

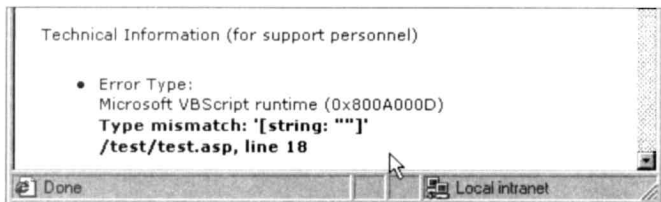


图7-9 错误提示屏幕

(1) 如果不是一位JScript专家

在寻找错误时，这不是一个大问题，因为我们能够迅速发现为什么会出现错误。事实上网页停止运行有助于我们跟踪错误。然而意外的错误可能会发生。例如，用 JScript重写程序代码，由于不是一位JScript专家，里面出现一些细小错误。

```
<%  
// get the value from the Request and display it  
var datBirthdate = new Date(Request.Form("BirthDate"));  
Response.Write("The value you entered is: " + datBirthdate + "<P>");  
  
// get the individual date elements  
intDay = datBirthdate.getDay();  
intMonth = datBirthdate.getMonth();  
intYear = datBirthdate.getYear();  
  
// and display them  
Response.Write("Day: " + intDay.toString() + "<BR>");  
Response.Write("Month: " + intMonth.toString() + "<BR>");  
Response.Write("Year: " + intYear.toString() + "<BR>");  
%>
```

图7-10即是运行结果，尽管程序没有停止运行并给出运行期错误，还是马上看出其中有些问题，月份不可能是0。

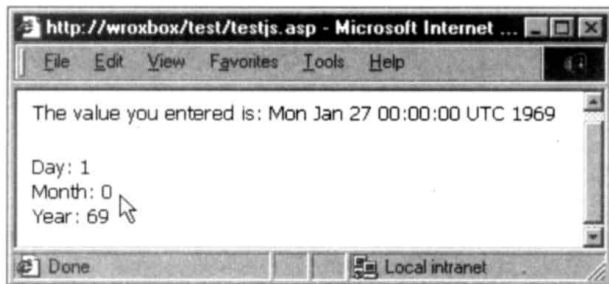


图7-10 显示生日的屏幕

问题出现的原因在于 JScript的getMonth函数返回的结果为0~11范围内的数，因此需要再加1，才能得到正确的结果。

```
intMonth = datBirthdate.getMonth() + 1;
```


(2) 衍生错误

即使不把初始值赋给网页去和结果比较,上面这种错误也可能是相当明显的。然而,如果面对的是一个数据库系统,并且没有看到显示出不正确的结果,可能不知道为什么程序不能正确地更新数据库。更糟糕的是,如果简单地把数值做为整型数据存入数据库,可能直到有人试图对这个数据查询时才能发现这个错误。

现在,发现大约有十二分之一的成员出生在 0 月份可能会使人吃惊,并会引起一些问题。记住,不仅仅是那些 1 月份出生的人员存在数据库中的信息不正确,而且每个成员都是这样。如果有许多应用程序都能增加和修改这个数据库中的记录,跟踪这个错误可能是艰苦的工作,特别是,不能去查找错误出现在那个程序行,而是首先要找出错误出现在哪个应用程序中。

(3) 掌握日期的用法

在上面的程序中出现的日期型数据的错误不是非常明显,不论使用者输入什么样的日期,程序代码只能给出 0~6 之中的值,原因在于编码中的设定,特别是从 VBScript 转换到 JScript 时。在 JScript 中, getDay 函数返回的是周中的某一天,而不是月中的某一天,这等于 VBScript 中的 Weekday 函数, getDay 函数的返回值是 0(代表星期日)到 6(代表星期六)。

注意 VBScript 的 Weekday 函数返回 1(代表星期日)到 7(代表星期六)。

因此,在 Jscript 中由 getDate 函数获得某月的日期的正确代码是:

```
...  
// get the individual date elements  
intDay = datBirthdate.getDate();  
intMonth = datBirthdate.getMonth() + 1;  
intYear = datBirthdate.getYear();  
...
```

运行这段程序便可得到想要的结果,如图 7-11 所示。

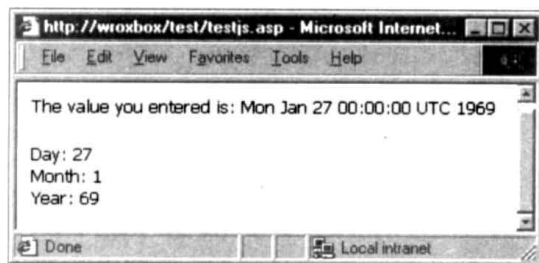


图7-11 显示正确生日的屏幕

7.2 各种运行期错误

本章前面部分展示了一些问题,包括错误如何出现、如何寻找错误和如何处理错误等等。现在更重要的是要掌握能够发生的不同种类的错误,并且如何区分这些错误。需要记住的是,如果知道了到哪里去找和寻找什么,调试则是比较容易的。在本章最后,将介绍错误确实出现时如何捕获错误,并且要尽可能早地阻止错误的发生。

在学习这些内容之前,首先要深入了解一下在某个阶段肯定会遇到的不同类型的运行期和语义错误,主要讨论以下内容:

- 逻辑错误。

- 脚本运行期错误。
- ASP和SSI运行期错误。
- 客户端脚本错误。

7.2.1 逻辑错误

逻辑错误在脚本中通常难于跟踪,因为这些错误常常是产生错误的结果而不中止网页运行。通常只有一些值出现超出边界的情况,如在前面数组实例中看到的那样,错误才显现出来。

然而,在错误和调试环境中,一种算法并不像数学课上所学的那样复杂。从计算的角度看,算法只是指一段能完成某个任务(通常返回某个结果)的程序。

1. 数值超界(数据溢出)

典型的逻辑错误一般涉及到数值,或者是涉及数据溢出等。例如,如果有名为 image1.gif、image2.gif等的一系列图像,编写以下一段程序随机挑选一幅图象用以显示:

```
<%  
'create a random number between 1 and 5  
intRandom = CInt(Rnd() * 5) + 1  
>%  
<IMG SRC="<% = "image" & CStr(intRandom) & ".gif" %>">
```

在网页中创建 元素用以指定随机选中的图象,例如:

```
<IMG SRC="image3.gif">
```

然而,如果碰巧这段程序产生的结果是 image6.gif文件。在这种情况下,如果本来仅希望得到在1~5中的一个结果,网页会是一个破碎的图像符号。原因是 VBScript中的CInt函数将值取整到最近的整数值。为了舍去小数部分,需要使用 Int或者Fix函数代替CInt。

2. 运算符的优先级

其他类型的逻辑错误有按指令计算而出现的错误,例如想用除法时采用了乘法会产生错误的结果。而由于程序中数学运算符的运行顺序或优先级,会引起一些更难发现的错误,例如,下面这段程序可能会产生不正确的结果。

```
intResult = intValue1 * intValue2 + intValue3
```

因为乘法比加法有较高的运算优先级,所以先进行计算。但是如果想把第一个数和后两个数的和相乘,必须用括号来改变这种缺省的运算优先权。

```
intResult = intValue1 * (intValue2 + intValue3)
```

在VBScript 5.0文档中的VBScript Basics|VBScript Operators中,给出了所有脚本运算符的优先级表。对于JScript,在JScript Tutorial|JScript Basics|JScript Operators下也可找到相应的优先级表。然而需要记住的最基本原则是:乘、除法优先于加、减法。

3. 管理和格式化字符串数据

从计算意义上考虑,具有计算功能的任何结构或函数都可看作一种算法。例如,可以从数据库中取值构成一个字符串,代表顾客的名字。这里不涉及如何从数据库中提取数据(本书的后面部分进行讨论)。下面程序的功能是字符串连接。

```
strTitle = {get from database}  
strFirstName = {get from database}  
strMiddleInitial = {get from database}
```

```
strLastName = {get from database}
strOther = {get from database}

strPrint = strTitle & ". " & strFirstName & " " & strMiddleInitial _
          & ". " & strLastName & " " & strOther
```

运行这段程序可以得到如下结果：

```
Ms. Janet C. Clarke MBNA.BSc.MechEng.
```

但不是每个人都和 "Janet" 一样，有一个中间名字。并且许多人可能没有头衔，所以可能仅仅得到：

```
. Alex . Homer
```

这当然不是一个能引起脚本不能运行或者产生运行期错误的致命错误。然而，对于用户来说，提供这样的脚本是不可接受的。最好程序能在输出字符串之前检查名字的每一部分。

```
...
strPrint = ""
If Len(strTitle) Then strPrint = strPrint & strTitle & ". "
If Len(strFirstName) Then strPrint = strPrint & strFirstName & " "
If Len(strMiddleInitial) Then strPrint = strPrint & strMiddleInitial & ". "
If Len(strLastName) Then strPrint = strPrint & strLastName
If Len(strOther) Then strPrint = " " & strOther
```

上面这段程序保证了空格和小数点仅加在名字中有值的地方。如果仅给 strOther 字符串赋值，而对其他都不赋值的话，将在开始处得到一个空格。然而出现这种情况的可能性非常小。如果有姓的话，通过仅添加 " other " 部分可以防止这种错误的发生。

```
...
strPrint = ""
If Len(strTitle) Then strPrint = strPrint & strTitle & ". "
If Len(strFirstName) Then strPrint = strPrint & strFirstName & " "
If Len(strMiddleInitial) Then strPrint = strPrint & strMiddleInitial & ". "
If Len(strLastName) Then
    strPrint = strPrint & strLastName
    If Len(strOther) Then strPrint = " " & strOther
End If
```

最坏的情况是结果为一个空字符串，可以检查这种可能性并中止打印。

```
...
If Len(strPrint) = 0 Then
    Response.Clear
    Response.End
End If
```

7.2.2 脚本运行期错误

使用一个不存在的函数，或者破坏了脚本语言使用的规则，会出现脚本运行期错误。许多错误是语法错误(本章前面讨论过的)，但是许多错误是由于所赋的值和函数参数的要求不一致引起的。

例如，用一个窗体收集来自用户的日期，并存入数据库中，或者用其他方式进行处理。为了确定日期是有效的，在把数据插入数据库之前使用 CDate 函数：

```
<%
strData = Request.Form("TheDate")
datDate = CDate(strDate)
...

```

如果用户在填表时出现了差错，程序便会产生一个脚本错误，如图 7-12 所示。

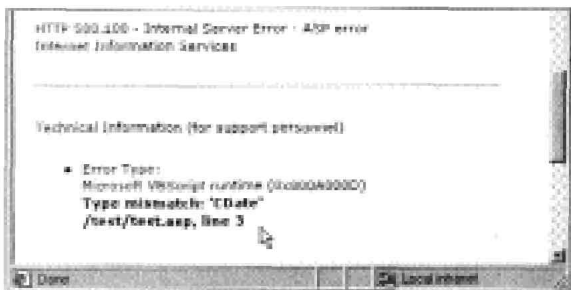


图7-12 出错信息的屏幕

查看错误信息，可以发现错误是由执行程序代码的脚本引擎产生的。错误号用十六进制显示出来，它是由 VBScript 错误号和十六进制数 0X800A0000 相加得到的（见第 4 章），上例中 VBScript 错误号是十六进制 0xD，或者十进制数的 13。

大多数微软技术（包括 ASP）返回的错误号是由 8 位十六进制数组成的。第一位字符总是 8，表明这个状态信息是服务器错误信息。后面跟着 2 位 0，然后是服务代码。对 VBScript 和 JScript 错误，服务代码总是“A”，最后 4 位字符是用十六进制数表示的错误号。

如果查看一下 VBScript 文档，你会发现 13 号错误是“Type Mismatch”错误。当然，我们从 ASP 错误页中显示的错误描述中已经知道了这一点。然而，在本章后面我们将要看到，在错误处理技术中，得到错误号是非常有用的。

注意，在错误信息显示窗口中，显示的是服务器对错误的反馈信息。HTTP 状态代码为 500.100，属于“Internal Server Error”。在第 4 章，讨论 ASP 定制错误网页的工作方式时，我们发现这种错误常常因为载入了错误网页。本章后面，将会看到在网页中如何处理这些错误。

7.2.3 ASP 和 SSI 的运行期错误

脚本错误是由正在使用的脚本引擎发现的，然而 ASP DLL 和 SSI DLL 也能发现脚本错误，尽管它们与使用的脚本引擎无关。典型的 SSI 例子是在 #include 指令中给文件一个错误的名字或路径。错误由 SSI DLL 或 ASP 发现的，而不由脚本引擎发现。可看到此时错误类型是“Active Server Pages”，ASP 内部错误代码是“ASP 0126”，如图 7-13 所示，然而在这种情况下，错误号是 4005，指出了这是一种为 SSI DLL (ssinc.dll) 定义的特殊错误错误。

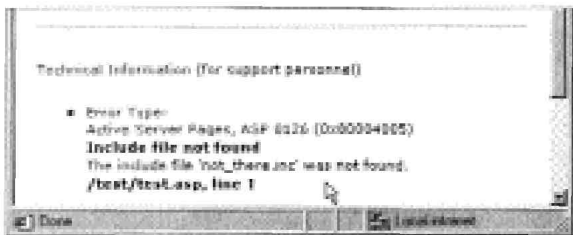


图7-13 出错信息的屏幕

ASP错误代码总览

对于在ASP DLL中造成失败的错误，表 7-1是返回的错误代码。当这类错误发生时，你可以在ASPError对象的ASPCode属性中找到这些错误代码。

表7-1 ASP 错误代码

错误代码	错误消息和扩展信息
ASP0100	Out of memory(内存溢出)
ASP0101	Unexpected error(函数返回exception_name)
ASP0102	Expecting string input(期待字符串输入)
ASP0103	Expecting numeric input(期待数字量输入)
ASP0104	Operating not allowed(操作不允许)
ASP0105	Index out of range(数组下标溢出)
ASP0106	Type Mismatch(数据类型不匹配)
ASP0107	Stack Overflow(处理的数据量超过了允许的范围)
ASP0115	Unexpected error(出现在外部对象中的可捕获的错误 exception_name，脚本不能继续运行)
ASP0177	Server.CreateObject Failed(无效的ProgID)
ASP0190	Unexpected error(当释放外部对象时，出现的可捕获的错误)
ASP0191	Unexpected error(在外部对象的 OnStartPage方法中出现的可捕获的错误)
ASP0192	Unexpected error(在外部对象的 OnEndPage 方法中出现的可捕获的错误)
ASP0193	OnStartPage Failed(在外部对象的 OnStartPage方法中出现错误)
ASP0194	OnEndPage Failed(在外部对象的 OnEndPage 方法中出现错误)
ASP0240	Script Engine Exception(脚本引擎从 object_name抛出异常exception_name)
ASP0241	CreateObject Exception(object_name的CreateObject方法所导致的异常exception_name)
ASP0242	Query OnStartPage Interface Exception(查询对象 object_name的OnStartPage或OnEndPage方法所导致的异常exception_name)

ASP 错误通常仅当组件有问题或服务器本身有问题时才出现。最常见是使用 Sever.CreateObject时的ASP 0177错误和严重的ASP 0115错误。ASP 0115错误通常表示组件程序代码中发生的错误，而ASP 0177错误通常是由不能正确安装组件引起的或者由我们指定的 Prog ID字符串的错误引起的。

7.2.4 客户端脚本错误

到目前为止，我们已了解了来自 ASP的错误。然而ASP也经常用于创建包含客户端脚本的网页。如果包含客户端代码的 < SCRIPT > 元素没有被设置成RUNAT = "SERVER"属性，ASP将不考虑服务器，而把网页信息不加改变地传送到客户端。

因此，如果打开了一个 ASP网页，并且显示的是一个浏览器错误对话框，就不应该在服务器端寻找ASP程序代码的错误。浏览器看不到 ASP程序代码，所以不能识别任何错误，如果有一个对话框出现在客户端，那么在客户端代码中必定有一个错误。

1. 语法错误

如果在网页中的客户端程序代码有语法错误的话，当脚本下载到客户端，浏览器便会出现相应的错误。尽管网页中内容仍可正常载入(除非由这些客户端脚本代码动态装入)，但网页停止执行。用户将看到一个包含错误细节的对话框，或者是一个指示网页包含错误的状态条消息。

现代浏览器趋向于隐藏网页脚本错误的细节，而仅在状态条上显示一个小的错误图标。

在IE 4.0和IE 5.0中, 正常的错误对话框可以通过 Internet Options对话框的 Advanced页进行设置来激活, 如图7-14所示。

处理脚本程序代码中的客户端错误和在服务器端相似, 并且通常会更容易些, 因为经常可以直接从服务器目录中通过双击来下载网页。一般不需要通过 Web服务器和HTTP获得网页来观察浏览器中的结果, 其中的唯一不同是一些服务器交互由客户端脚本来完成, 如使用 RDS的数据绑定或者动态装入。

2. 运行期或语义错误

在客户端脚本中, 通常可能会遇到语法错误, 也会经常遇到运行期或语义错误。事实上, 在客户端, 这种现象是很普遍的。因为在客户端不能像服务器端那样对脚本的环境进行控制, 不能肯定用户在他们的机器上正运行什么, 实际上在服务器上仅能从一些组件如Browser Capabilities中得到大概情况。

所以, 使用客户端对象或特殊版本的脚本语言和属性的脚本程序很可能不能正常工作。尽管如此, 处理客户端错误和处理服务器端错误是差不多的。

3. 在服务器上创建的客户端程序代码

在错误发生时, 作为“客户端对话框对应于 ASP错误页面”规则(关于出错的地方)的一个特别的例外是, 使用 ASP程序代码在服务器上动态地创建客户端程序代码。例如, 可能想在ASP中进行求值运算, 然后把数据传给运行在客户端的脚本代码, 可能最容易的方法是把数据作为一个变量插入脚本代码中:

```
<%  
'get the name of our server from the ServerVariables collection  
strServerNameInASP = Request.ServerVariables("SERVER_NAME")  
%>  
  
<SCRIPT LANGUAGE="JScript" RUNAT="CLIENT">  
<!-- hide code from older browsers  
var strServerName = "<% = strServerNameInASP %>";  
...  
alert('Server name is: ' + strServerName);  
...  
// stop hiding code -->  
</SCRIPT>
```

在客户端, 在ASP处理这个页面之后, 将得到的是:

```
<SCRIPT LANGUAGE="JScript" RUNAT="CLIENT">  
<!-- hide code from older browsers  
var strServerName = "WROXBOX";  
...  
alert('Server name is: + strServerName);  
...
```

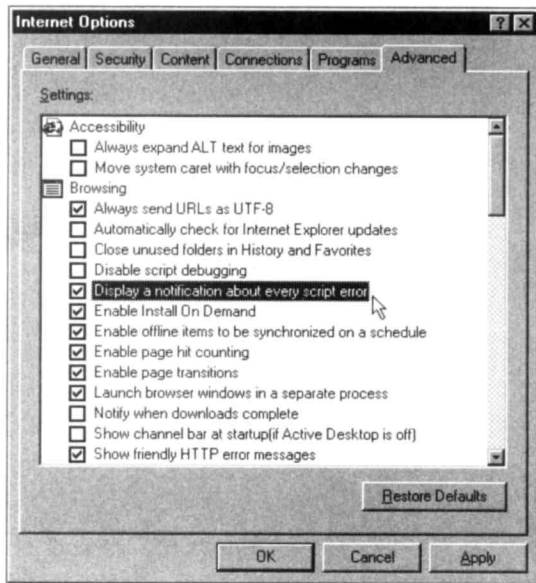


图7-14 Advanced页面设置屏幕


```
// stop hiding code -->
</SCRIPT>
```

可以忽略RUNAT="CLIENT"属性,但是加上这一项可以使得在查看运行代码的ASP网页时更加清楚。

这样,如果在某个位置想把服务器端数据库中的数据加入到一个客户端数组中,可以采用下面的程序实现:

```
<SCRIPT LANGUAGE="JScript" RUNAT="CLIENT">
<!-- hide code from older browsers
var arrBooks = new Array(10) // highest available index will be 9

<% 'start of ASP processing
intIndex = 0
Do While { not at the end of some recordset }
    strTitle = { get title from database record }
    Response.Write 'arrBooks[' & CInt(intIndex) & '] = ' ' _
        & strTitle & ';' & vbCrLf
    intIndex = intIndex + 1
    { move to next record in database }
Loop
%>
...
do something here on the client with the array of book titles
...
// stop hiding code -->
</SCRIPT>
```

这段服务器端ASP程序代码产生的客户端代码,在客户端运行时创建书名标题数组。同时产生的客户端脚本错误出现在浏览器的错误对话框中。错误的原因是以 arrBooks命名的数组是由JavaScript代码运行在客户端时创建的,仅能接受 9个书名;而服务器端代码能很可能产生多于9个的书名,具体多少由源数据库中的记录数来决定。这相当于如上客户端代码:

```
<SCRIPT LANGUAGE="JScript" RUNAT="CLIENT">
<!-- hide code from older browsers
var arrBooks = new Array(10) // highest available index will be 9
arrBooks[0] = 'Instant JavaScript';
arrBooks[1] = 'Professional ASP 3.0 Programming';
arrBooks[2] = 'ADO 2.5 Programmers Reference';
...
etc
...
arrBooks[9] = 'ASP Techniques for Webmasters';
arrBooks[10] = 'ASP Programmers Reference'; // <- client-side error occurs here
arrBooks[11] = 'ADSI CDO Programming';
arrBooks[12] = 'Professional MTS and MSMQ Programming';
...
do something here on the client with the array of book titles
...
// stop hiding code -->
</SCRIPT>
```

这个页面只有经过修正之后才能正常工作,可以通过增加数组大小,也可以通过控制来自数据库的记录数使其正常工作。

7.3 防止错误

上面已经看到了能够出现的一些不同类型的错误,并且有了一些查找错误的感觉。下面

将考虑如何避免把错误引入程序中，尽管不能保证所编的程序没有错误，但是这里概括的许多技术有助于减少错误数目。

良好的编程习惯

在编程中避免出现错误是和良好的编程习惯相关的，这里有许多工作我们要做，以减少把错误带进网页的可能性。可能有些人因采用某个技术而走向极端，甚至一定程度上在某个特殊问题上因书生气十足而引入了更多的错误。当然编程人员也不可能采用了这里列出的所有技术。

要考虑的主要内容是：

- 代码的格式化和缩进编排。
- 变量显式表明。
- 变量转换为合适的数据类型。
- 使用有意义的变量命名约定。
- 封装脚本。
- 注意潜在的错误情况。
- 程序测试。

1. 代码的格式化和缩进编排

许多VBScript编程员懒于格式化编排其书写的程序。尽管这并不阻碍程序运行，但这使得查找何处产生了错误变得困难。例如，在前面我们看到的程序中，丢失了一个 End If，由于嵌套结构的缩进，错误在哪里是相当明显的：

```
objCounters.Remove strCounterName
Response.Write "Removed counter " & strCounterName
<--- missing 'End If' should be here
End If
End If
%>
```

如果程序看起来像下面所示的那样，寻找错误将不是一件易事：

```
<% if Len( Request.Form("cmdSet")) then
strCounterName=Request.Form("lstSet" )
StrNewValue=Request.Form ("txtSet")
if isnumeric (strnewvalue) then
intNewValue =cint(strNewValue)
objCounters.Set strCounterName, intNewValue
Response.write "Set counter " & strCounterName & " to "& strNewValue

else
Response.write strNewValue & " is not a valid number"
If Len ( Request.Form ("cmdRemove")) then
strCounterName = Request.Form("lstRemove")

objCounters.Remove strCounterName
Response.write "Removed counter "& strCounterName
end if
End IF
%>
```

2. 显式表明变量

VBScript支持Option Explicit语句。在一个脚本页面的开头插入Option Explicit语句时，可

以避免使用没有用 Dim 命令(或用于动态数组的 ReDim)定义的变量。似乎不需要这么做,因为脚本语言允许通过给一个变量赋值来创建一个需要的变量。然而用 Option Explicit 进行定义有助于避免错误,特别是那些难以发现的引起脚本产生不正确结果的逻辑错误。

例如,编写如下程序:

```
<%  
'get values for calculation  
strSalesTotal = Request.Form("SalesTotal")  
curSalesTotal = CCur(strSalesTotal)  
sngCommissionPercent = 2.5  
  
'calculate commission payment  
sngCommission = curSalesTotal * (sngComissionPercent / 100)  
%>
```

运行这段程序不会产生错误(当然,除非用户给销售合计值赋了非法的值)。然而这段程序总是会产生0的结果,因为在程序的最后一行中 SngCommissionPercent 变量名拼写错了。脚本解释器将产生一个新的变量名(叫作 SngComissionPercent),由于没有赋值,在数学计算时返回值总为0。

为了防止这种错误,仅需在程序开头增加 Option Explicit 语句。

```
<%  
Option Explicit  
Dim strSalesTotal  
Dim curSalesTotal  
Dim sngCommissionPercent  
  
'get values for calculation  
strSalesTotal = Request.Form("SalesTotal")  
curSalesTotal = CCur(strSalesTotal)  
sngCommissionPercent = 2.5  
  
'calculate commission payment  
sngCommission = curSalesTotal * (sngComissionPercent / 100)  
%>
```

这时,当脚本引擎试图解释程序时将识别出一个语法错误,并且能够指出此变量没有声明,如图 7-15 所示。

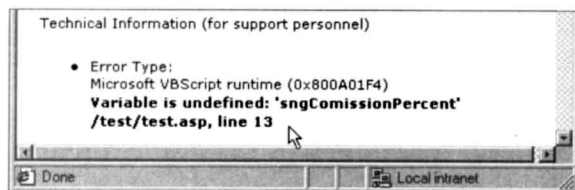


图 7-15 显示的错误信息

在 JScript 中引用一个没有声明的变量将返回一个 "Undefined" 信息,并且在试图使用变量之前,能够检测到这种情况。

3. 变量转换为合适的数据类型

回头看看前面的程序,可能发现用 CCur 函数把用户提供的数据转换成了货币型数据类型。在 VBScript 中,有一系列类似这样的数据类型变换函数,在第 3 章中有详细的描述。

```
blnBoolean = CBool(varVariant) 'converts to a Variant of subtype Boolean
bytByte = CByte(varVariant) 'converts to a Variant of subtype Byte
curCurrency = CCur(varVariant) 'converts to a Variant of subtype Currency
datDate = CDate(varVariant) 'converts to a Variant of subtype Date
dblDouble = CDb1(varVariant) 'converts to a Variant of subtype Double
intInteger = CInt(varVariant) 'converts to a Variant of subtype Integer
lngLong = CLng(varVariant) 'converts to a Variant of subtype Long
sngSingle = CSng(varVariant) 'converts to a Variant of subtype Single
strString = CStr(varVariant) 'converts to a Variant of subtype String
```

如果不能完成变换，也就是说变量内容对新数据类型来说是无效的，便会出现一个运行期错误。然而，如果对数值类型进行变换，我们希望这个数值是有效的，并且能在程序中使用。因此能够检测一个无效的值对于防止错误的出现是一件“幸事”。

如果想把输入空格作0对待，并且把任何其他无效的输入作为用户错误对待，前面程序变为：

```
strSalesTotal = Request.Form("SalesTotal")
If Len(strSalesTotal) = 0 Then
    'no value entered, so assume zero
    curSalesTotal = 0
ElseIf Not IsNumeric(strSalesTotal) Then
    'not a valid number, so report an error and stop
    Response.Write "The value you entered is not a valid number."
    Response.Flush
    Response.End
Else
    'OK to convert the string value and use it
    curSalesTotal = CCur(strSalesTotal)
End If
```

在JScript中，所有的变量都是对象，并且有 `typeof()` 方法。可以使用 `typeof()` 来确定存在变量中的数据是什么类型，见第3章中的详细论述。

也可以对 `"null"` (VBScript中为 `Null`) 进行测试保证在程序使用各种变量之前它们已经赋了值。一个特例是从数据库中获得数据时，数据库中的字段内容经常是 `Null`，表示没有数据。

4. 变量命名和编码约定

阅读过本章和前面几章后，读者可以看出我们对变量名使用三个字母的前缀，用以指明它所代表的数据类型。尽管在这两种脚本语言中用 ASP 提供的所有变量都是 `Variant` (或JScript中的等价物) 类型的，但用变量名来区分出存储在其中的数据的类型仍是非常有用的，有助于防止编写程序时出错。

有许多不同的变量命名约定，经常使用的见表 7-2。

表7-2 变量类型及前缀

变量类型	前 缀
布尔型(Boolean)	bln
字节型(Byte)	byt
日期/时间型(Date/Time)	dat或dtm
集合型(Collection)	col
双精度型(Double)	dbl
整型(Integer)	int
长整型(Long)	lng
对象型(Object)	obj
单精度型(Single)	sng
字符型(string)	str

对一个包含函数和子程序的网页，指出某个变量是否已经声明或存在于任何函数和子程序之外是非常有用的。若已经声明，则该变量对网页来说是全局变量。对全局变量加上“g”前缀，所以一全局字符串变量可被命名为 gstrMystring；类似地，以“a”为前缀的变量是数组或数组元素。

程序注释

许多编程人员感觉到对程序增加注释不仅增加了不必要的开发时间，而且也减缓了网页的运行速度，因为脚本解释器每次必须先读整个程序，然后再跳过这些注释。尽管这种观点有一定的道理，但是一个月后再回过头来想读懂没有注释的程序，是非常困难的。

至少应该对常用函数和子程序进行注释以便你和其他人能重新使用这些程序。特别是，使用新的 Server.Execute 方法更加容易（详细情况参阅第4章）。下面是微软提供的一个例程的注释格式。

```
*****
' Purpose: what the routine is designed to achieve
' Inputs: a list of all the parameters to the routine
'   parameter1: description, data-type, etc.
'   parameter2: description, data-type, etc.
' Returns: what data type is returned, and what it contains
' Comments: other comments about the routine, update history, etc.
*****
```

5. 封装脚本语言以便代码重用

刚刚看到了如何注释子程序和函数以便易于重新使用。面向对象编程的原理是建立在程序代码重用的基础上的，并且 SSI 的 #include 和新的 Server.Execute 方法使调用存储在程序库中的函数更容易。

例如，如果有一系列函数用于计算税收和商品的的应付费用。可把包含这段程序的页面插入其他页面中：

```
<!-- #include VIRTUAL="/library/code/online_sales/tax_and_delivery.inc" -->
```

包含文件必须含有脚本定界符，或者用 <SCRIPT RUNAT="SERVER">..</SCRIPT> 或者用 <%...%>，每一个子程序和函数应该采用其要求的数值做参数，并且用函数值或更新的参数返回结果。不能使用全局变量，况且不同网页之间的全局变量也是不可用的。但在主网页中的程序能安全地调用所需的函数和子程序。

另外可使用 Server.Execute（或者 Server.Transfer）把执行转到另一个网页。如果有一段 ASP 代码用来为客户创建在线采购一览表，这种方法是很有用的。它包含 HTML 用来创建标题、表格，用代码进行计算并用 Request 集合的内容填写相应值（记住不能使用 Server.Execute 或 Server.Transfer 把脚本变量传到另一个网页）。另外，运行的网页能够支持函数、类定义（在 VBScript 中），或者其他设计为可重新使用的内容。

6. 注意潜在的错误情况

编程时不管如何仔细，比如在使用和对变量类型转换之前对变量值进行测试，但总还是有一些情况不能避免错误的出现。明显的例子是：当使用 FileSystemObject 对象的方法设法访问一个用户指定的文件时，不能确定这个文件是否已移动、删除或者标记成只读型，所有这些操作都可能使程序不能工作。

其他类似的情况可能是，当访问数据库或其他数据存储时，对用户帐户而言，有时要求

某一层权限。在这种情况下，可能因为需要的访问不能实现，使程序不能工作。

可以通过采取预防性措施编写程序，来测试类似的潜在错误。例如，可以使用 Tools 组件或者 FileSystemObject 对象的 FileExists 方法来查看，是否一个文件在访问之前就已存在了；或者使用 Permission Checker 组件来查看当前用户帐号是否有访问需要的文件或资源的权限；也可以通过使用 FileSystemObject 获得一个文件的属性设置，以便在删除或重写之前查看文件是否是只读的。

如果不考虑可能发生的错误并防止错误发生的话，这些情况和许多类似的情况都可能是潜在的运行期错误源。

7. 最后的测试

很明显，测试是对错误的最好防范方法。错误能通过应用程序影响到其他操作，如果不及时发现能引起不可估量的损失。用各种数值（如用户提交的，或者访问一个数据库得到的数据）对网页进行测试。同时，更重要的是，如果采用我们不希望的值会发生什么。程序能避免这些情况产生其他错误或者避免扰乱正在测试的子程序吗？

好的测试技术应该包括一系列值，如期望的值、边界条件值和超出边界的值。用期望传送到网页的值进行测试是应该经常做的工作，同样超出边界的值通常比较容易阻止。例如，可以限制可接受的数值范围为 - 100~ + 100，程序如下：

```
If (intValue < -100) Or (intValue > 100) Then
    'not a valid value, so report an error and stop
    Response.Write "Values must be between -100 and +100 inclusive."
    Response.Flush
    Response.End
End If
```

然而要记住查看边界条件，上面的程序能处理 -100和 + 100吗？想把数据限制在 -100~ + 100中吗？取0时会发生什么？上面的程序会显示“Divide By Zero”错误而最终停止执行吗？

7.4 处理错误

即使采用了防御性编程技术之后，错误仍能进入到网页，这可能是由于测试并不充分，或者是因为所依靠的一些其他资源或服务没有正确工作。为了防止页面出现问题，在程序中要能够进行定制错误处理。

7.4.1 ASP 缺省错误处理器

前面已经看到过，ASP和IIS能找出网页中的大多数错误，并且能自动生成错误信息页，这些错误几乎总是 500.100 类型的，并且 IIS 用 Server.Transfer 方法装载以 500-100.asp 命名的缺省错误页，然后传送给客户。第 4 章介绍了这一工作过程，以及如何与定制错误网页接口。

然而，运行期脚本错误不总是由 IIS 发现的，当一个运行期错误发生时，脚本引擎会查看一下目前执行点或语句的环境。如果正在执行一个子程序或函数，缺省的脚本引擎错误处理器通过终止子程序的运行并返回调用子程序的地方来指出错误。

在这里，程序会查看是否实现了其他的错误处理器，如果没有的话，又会重复这个过程，然后返回到调用子程序的地方。当子程序返回到网页的主程序（在任何其他子程序或函数外面）时，程序又会查看是否实现了任何其他错误处理器。在这个过程中，只有确实没有发现其

他的错误处理器，程序才给 ASP 提示错误，指示 IIS 把执行转到缺省的错误页面。

7.4.2 VBScript 错误处理

在 VBScript 中，可以使脚本解释器不处理其找到的任何错误，并且使用 On Error Resume Next 语句继续运行下个语句。一旦这个语句已被处理，脚本引擎将继续运行后面的程序，而不理会已经发现的任何错误。然而，这种过程仅适用于顺序执行语句的环境，换句话说，不适用于嵌套的函数或子程序。

1. 使用 on Error Resume Next 语句

一个错误在子程序中出现时，如果没有运行 On Error Resume Next 语句，那么错误将被交给调用它的环境，这个过程一直重复到找到运行 On Error Resume Next 语句的环境继续运行，或者找到缺省的脚本错误处理器，把错误交给 ASP 并且 IIS 显示缺省错误网页。这个过程如图 7-16 所示。

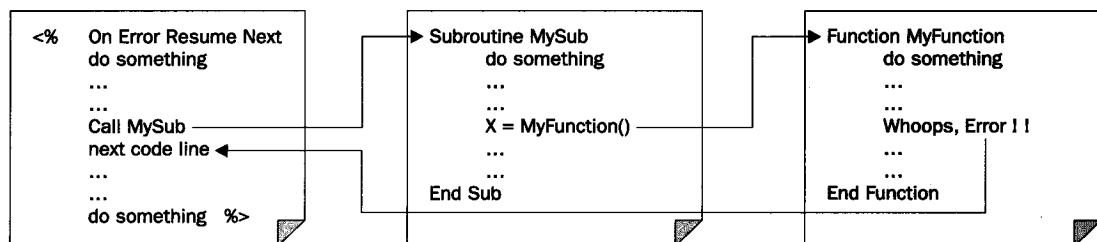


图7-16 错误处理过程

这种错误调用链意味着可以创建防止使程序停止运行的运行期错误的函数和子程序。如果在子程序的开头放置一个 On Error Resume Next 语句，任何运行期错误会中止这个子程序的运行，但是调用该子程序的程序将继续运行而不会引起网页的停止。

例如，如果需要向一个文件中写入字符串，可以通过一个独立的函数对文件进行访问文件，防止错误中断整个程序的运行：

```
'creates a file named strFileName, overwriting any existing one with that name
'and writes strContent into it then closes the file
'returns True if it succeeds, or False on any error
Function WriteNewFile(strFileName, strContent)
    On Error Resume Next 'turn off the default error handler
    WriteNewFile = False 'default return value of function
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    If Err.Number = 0 Then Set objFile = objFSO.CreateTextFile(strFileName, True)
    If Err.Number = 0 Then objFile.WriteLine strContent
    If Err.Number = 0 Then objFile.Close
    If Err.Number = 0 Then WriteNewFile = True
End Function
```

注意上面的程序在试图处理每个程序语句之前，先检查 VBScript 的 Err 对象的 Number 属性。如果这个值为 0 (还没有出现错误)，那么就能够继续对文件的写入和创建过程。然而如果错误确实发生了，脚本引擎将设置 Err 对象的属性的值，并且继续处理下一行。

只要不引起错误而能正常运行，函数的返回值将设置为“True”。否则函数将返回“False”。在编程中可以在对其进行测试以后，再使用该函数和采取其他行动。

下面是一个简单的例子，我们希望对任务的每一部分采用一个独立的函数，以便能更精

确地辨别出错误产生在何处。这样，调试时也更容易阅读代码。在页面的主程序中，可以调用三个单独的函数。

```
If CreateNewFile(strFileName) Then                                'create the new file
    Response.Write "New file successfully created<BR>"
    If WriteContent(strContent) Then                               'write the content
        Response.Write "Content written to file<BR>"
    Else
        Response.Write "ERROR: Failed to write to the file<BR>"
    End If
    If CloseFile(strFileName) Then                                 'close the file
        Response.Write "File closed<BR>"
    Else
        Response.Write "ERROR: Failed to close the file<BR>"
    End If
Else
    Response.Write "ERROR: Failed to create the new file<BR>"
End Function
```

2. 使用On Error Goto 0

在ASP 2.0(尽管没有文档记录)和ASP3.0中，也能使用On Error Goto 0语句恢复缺省的错误处理行为。在运行这个语句后，发生的运行期错误将导致缺省错误处理，在环境链中检查每个嵌套的程序，直到主页面代码。如果没有其他的环境关闭缺省错误处理，网页的执行将停止并显示IIS缺省错误网页。

3. VBScript Err对象

在前面的例子中，关闭缺省错误处理时，通过检查 VBScript Err 对象的Number属性，查看错误是否已经出现。Err 对象存储了关于运行期错误的信息，表 7-3和表7-4给出了VBScript Err 对象提供的方法和属性。

表7-3 VBScript Err 对象的方法

方 法	说 明
Clear	清除当前所有的Err对象设置
Raise	产生一个运行期错误

表7-4 VBScript Err 对象的属性

属 性	说 明
Description	设置或返回一个描述错误的字符串
Number	(缺省)设置或返回指定一个错误的值
Source	设置或返回产生错误的对象的名称

使用这些属性可以检查发生了哪种错误。例如，可以根据错误号采取不同的措施，也可以用Source和Description的属性值为用户提供错误信息，或者传送到一个文件中。

也可以使用Err 对象生成一个错误。为什么要做这些呢？因为有时想把一个定制的错误消息传送给用户。可以把Err对象的属性设置成所希望的任何值。然后调用 Raise方法来产生这种错误，这样做会停止程序的运行，并且把错误沿调用链向回传递。

下面的例子显示了在服务器磁盘上读取一个文本文件时，如何处理错误。注意如何使用常数vbObjectError，以确定所选择的错误号不会和一个已存在的错误号混淆。通过把任意选

择的错误号加到此常数中，就能够保证和预定义的错误号不混淆。

```
Function ReadThisFile(strFileName) 'returns the content as a string
    On Error Resume Next
    ReadThisFile = "" 'default return value of function
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    Set objFile = objFSO.OpenTextFile(strFileName, ForReading)
    Select Case Err.Number
        Case 0 'OK, take no action
        Case 50, 53 'standard file or path not found errors
            'create custom error values and raise error back up the call chain
            intErrNumber = vbObjectError + 1073 'custom error number
            strErrDescription = "The file has been deleted or moved."
            strErrSource = " ReadThisFile function"
            Err.Raise intErrNumber, strErrSource, strErrDescription
            Exit Function
        Case Else 'some other error
            'raise the standard error back up the call chain
            Err.Raise Err.Number, Err.Source, Err.Description
            Exit Function
    End Select
    ReadThisFile = objFile.ReadAll 'we opened it OK, so return the content
    objFile.Close
End Function
```

调用这个函数的代码可以使用 On Error Resume Next 语句，并且能捕获这个函数产生的错误。

```
On Error Resume Next
strContent = ReadThisFile("myfile.txt")
If Err.Number = 0 Then
    Response.Write "File content is:<BR>" & strContent
Else
    Response.Write "Error in " & Err.Source & "<BR>" & Err.Description
End If
```

7.4.3 JScript 错误处理

在 5.0 版之前，JScript 的错误处理能力并不出色，然而在 5.0 版中情况改变了，JScript 采用了一套和 Java 以及 C++ 非常类似的错误处理系统。它掌握起来尽管不像 VBScript 技术那样容易，但人们认为在错误处理上，JScript 走在前头。

在第 1 章中，在讨论这两个主要脚本语言的新特点的时候，已详细讨论了 JScript 错误处理，这里不再重复。如果阅读第 1 章时跳过了这部分，可以回到那里看看。

7.4.4 使用 IIS 错误页面

与 ASP 错误处理过程相关的内容是为 IIS 提供可定制的错误页面。事实上，在 IIS 4.0 中也有这个特点。但新的 ASP 内置对象 ASPError，更易于使用且提供更加强大的功能。

在第 4 章，当我们研究 Server.Execute 和 Server.Transfer 方法时，已经讲述了如何建立定制的错误页面。我们也讨论和使用了 ASPError 对象，但这种方式受到了一定的限制。在这一部分，将介绍如何将定制的错误网页和 ASPError 对象结合起来建立一个更好的处理 ASP 错误的方法。

我们可以使用 VBScript 检查 ASPError 对象的内容，从而创建一个定制的错误页面。构建

一个包含错误内容全面信息的字符串，且写入到服务器磁盘上的日志文件中。然而网页的设计仅使访问者看到网页不可用这样一条信息是不行的，应该使访问者能够选择是重新载入上一个网页还是回到主页，使他们没有意识已经发生了错误。

尽管我们采用VBScript创建这个网页，但其使用的一些特性对JScript来说也是适用的，这两种脚本语言的相互转换也是比较容易的。

可以从<http://www.wrox.com>站点下载本章及本书其他章节的示例文件。

1. 设置定制的错误页面

在能使用定制的错误页面之前，必须在Internet Services Manager进行相应的设置(设置方式见第4章)。把示例文件装入计算机的wwwroot目录中，打开Chapter07子目录的Properties对话框，在Custom Errors选项卡中，滚动列表并选中HTTP错误“500:100”条目，点击Edit Properties按钮，并键入定制的错误页面Custom_error.asp的URL，如图7-17所示。

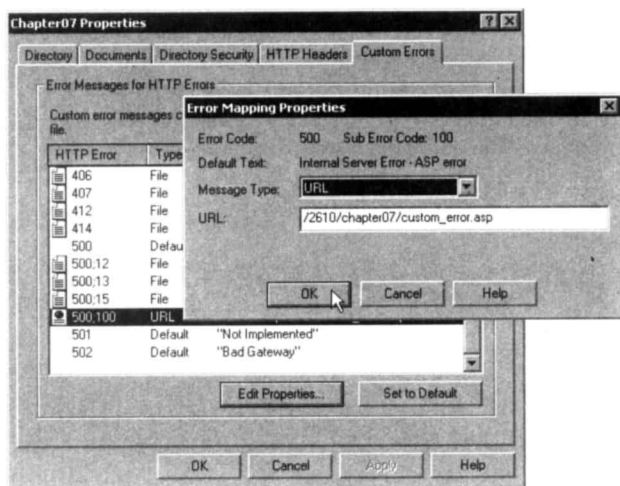


图7-17 Custom Errors选项卡

现在Chapter07子目录中的页面出现一个ASP错误时，就会打开定制的错误页面。

2. 使用定制的错误页面

在浏览器中打开chapter07目录并选择到“Using a Custom Error Page”的链接，这个页面显示了一系列用于产生各种类型的错误的按钮，点击标有“Load a Page with a Syntax error”的按钮，如图7-18所示。

这将载入了一个名为syntax_error.asp的简单页面。然而看不到这个页面，因为这个页面包含了一个语法错误。ASP终止这个页面的编译/执行，并把执行转到定制错误页面，这个页面展示了错误的细节和两个按钮，这两个按钮用以返回上个页面(主菜单)或返回Web站点的缺省主页，如图7-19所示。

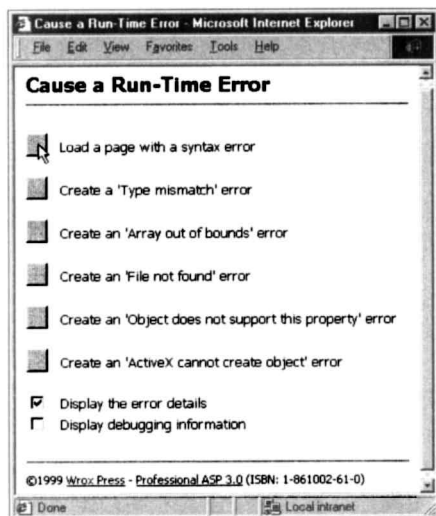


图7-18 演示定制错误页面的屏幕1



图7-19 演示定制错误页面的屏幕2

这个页面也把错误报告追加到服务器磁盘 C:\temp 文件夹中名为 custom_error.log 的日志文件中，可以在文本编辑器中打开并查看它，图 7-20 所示的日志文件已经记录了几个错误。

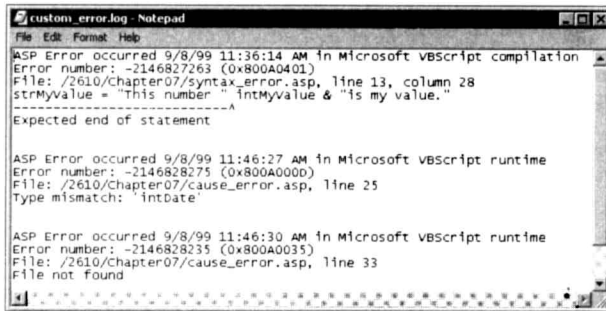


图7-20 日志文件

如果在页面中得到了一个信息，指明日志文件不能写入信息，可能是因为 IUSR_machinename(IUSR_计算机名)帐号没有访问 C:\temp 目录的权限。当测试这个页面时，应该给予 IUSR_machine name 帐号对这个目录的全部控制权，或者改变 custom_error.asp 页面的程序代码以指向一个 IUSR 有全部控制权的文件夹

错误消息出现在页面中的唯一原因，是因为在 cause_error.asp 页面中我们选择了相应的复选框。如果关闭该选项并再次点击按钮，便看不到错误的详细情况，然而错误信息仍然记录在服务器磁盘上的 custom_error.log 错误日志文件中。

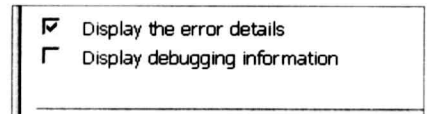


图7-21 cause_error.asp 页面的选择框

“Display debugging information” 复选框给定制错误页面(而不是日志文件)提供了更多的信息，有助于调试那些使用 ASP 内置对象集合值的页面，如图 7-21 所示。

在本章下面部分，将再讨论这一个问题，同时也可以了解“Cause an Error”页面上的其他按钮所提供的其他种类的错误信息。注意有一些按钮能够比其他的按钮能够提供更多信息。特别是只有最后一个按钮给出 ASP 错误代码的值(这里是 ASP 0177)。

(1) “Cause An Error” 页面的功能

与先前讨论的示例页面一样，引起错误的页面使用同样的技术，用 < Form > 把值提交给

同一个页面。然后ASP程序查看窗口上点击的是那个 SUBMIT按钮，然后运行代码的相应部分。同时查看是否页面上两个复选框是否选中，如果是这样，程序首先设置一个或两个会话级的变量以指明这一点。

```
<%
'see if we are going to display error and debug information
'set session variables to retrieve in the custom error page
If Len(Request.Form("chkShowError")) Then
    Session("ShowError") = "Yes"
Else
    Session("ShowError") = ""
End If
If Len(Request.Form("chkShowDebug")) Then
    Session("ShowDebug") = "Yes"
Else
    Session("ShowDebug") = ""
End If
...

```

由于使用了 Server.Transfer，当错误发生时，正在运行的网页的整个 ASP环境由 IIS 传给定制错误页面。然而，脚本变量的值并没有传给定制错误页面，所以必须使用 Session 变量，或者把值添加到 Request.Form 或 Request.QueryString 集合以便把值传送给定制错误页面。

设置了 Session 变量之后，程序继续查看点击了哪个按钮。每个类型的错误（除了第一类型外），都是由运行相应的 ASP 代码产生的，第一类型的错误需要调用另一个页面。

```
...
'look for a command sent from the FORM section buttons
If Len(Request.Form("cmdSyntax")) Then
    Response.Clear
    Response.Redirect "syntax_error.asp"
End If
If Len(Request.Form("cmdParamType")) Then
    intDate = "error"
    intDay = Day(intDate)
End If
If Len(Request.Form("cmdArray")) Then
    Dim arrThis(3)
    arrThis(4) = "Causes an error"
End If
If Len(Request.Form("cmdFile")) Then
    Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
    Set objTStream = objFSO.OpenTextFile("does_not_exist.txt")
End If
If Len(Request.Form("cmdPageCount")) Then
    Set objPageCount = Server.CreateObject("MSWC.PageCounter")
    objPageCount.WrongProperty = 10
End If
If Len(Request.Form("cmdObject")) Then
    Set objThis = Server.CreateObject("Doesnot.Exist")
End If
%>

```

(2) 定制错误页面的工作

知道了如何创建错误后，让我们来看看定制的错误页面。在前面的章节里已经知道了构建网页需要的理论，这里再概要地描述一下其工作过程。第一步是关闭缺省的错误处理器以便页面程序不被另一个错误中断。第二步通过创建一个新的 ASPError 对象收集原始错误信息。

进行这个工作时要格式化一些值，并把它们转换成合适的数据类型。

```
<%
'prevent any other errors from stopping execution
On Error Resume Next

'get a reference to the ASPError object
Set objASPError = Server.GetLastError()

'get the property values
strErrNumber = CStr(objASPError.Number) 'normal error code
strASPCode = objASPError.ASPCode 'ASP error code (if available)
If Len(strASPCode) Then
    strASPCode = "'" & strASPCode & "'"
Else
    strASPCode = ""
End If
strErrDescription = objASPError.Description
strASPDescription = objASPError.ASPDescription
strCategory = objASPError.Category 'type or source of error
strFileName = objASPError.File 'file path and name
strLineNum = objASPError.Line 'line number in file
strColNum = objASPError.Column 'column number in line
If IsNumeric(strColNum) Then 'if available convert to integer
    lngColNum = CLng(strColNum)
Else
    lngColNum = 0
End If
strSourceCode = objASPError.Source 'source code of line
...

```

现在构建一个错误报告字符串，这段程序看起来复杂，但实际上仅是一系列 If Then 语句的嵌套，用以产生良好的报告格式，没有任何空的段落。如果错误是语法错误，来自 ASPError 对象的 Source 属性的源代码可在 strSourceCode 变量中得到，可以使用这个变量及 lngColNum 的值(从 ASPError 对象的 Column 属性中得到)增加一个标记用来指明在源程序中的什么地方发现了错误。

```
...
'create the error message string
strDetail = "ASP Error " & strASPCode & " occurred " & Now
If Len(strCategory) Then
    strDetail = strDetail & " in " & strCategory
End If
strDetail = strDetail & vbCrLf & "Error number: " & strErrNumber _
    & " (0x" & Hex(strErrNumber) & ")" & vbCrLf

If Len(strFileName) Then
    strDetail = strDetail & "File: " & strFileName
    If strLineNum > 0 Then
        strDetail = strDetail & ", line " & strLineNum
        If lngColNum > 0 Then
            strDetail = strDetail & ", column " & lngColNum
            If Len(strSourceCode) Then
                'got the source line so put a ^ marker in the string
                strDetail = strDetail & vbCrLf & strSourceCode & vbCrLf _
                    & String(lngColNum - 1, "-") & "^"
            End If
        End If
    End If
    strDetail = strDetail & vbCrLf
End If

```

```

End If
strDetail = strDetail & strErrDescription & vbCrLf
If Len(strASPDescription) Then
    strDetail = strDetail & "ASP reports: " & strASPDescription & vbCrLf
End If
...

```

(3) 记录错误

用名为 strDetail 的字符串变量创建了错误报告后，可以在第 5 章中做的那样，采用 FileSystemObject 对象把它追加到日志文件中。如果成功，布尔型 “ failed flag ” 变量将被设置成 False。

```

...
'now log error to a file. Edit the path to suit your machine.
'you need to give the IUSR_machinename permission to write and modify
'the file or directory used for the log file:
strErrorLog = "c:\temp\custom_error.log"
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
Set objTStream = objFSO.OpenTextFile(strErrorLog, 8, True) '8 = ForAppending
If Err.Number = 0 Then objTStream.WriteLine strDetail & vbCrLf
If Err.Number = 0 Then
    objTStream.Close
    blnFailedToLog = False
Else
    blnFailedToLog = True
End If
%>

```

(4) 跳转到另一个页面

现在准备在网页中创建一些输出。在此之前，需要检查错误细节以确定下一步要做什么。例如，可用 ASPError 对象的 Number 或其他属性检查错误类型。在这里，可认为 “ Type Mismatch ” 错误不是代码中有错误，可能是由于用户在文本框中输入错误数据产生的。所以不显示这个网页的剩余部分，而是跳转到另一个网页

```

If objASPError.Number = -2146828275 Then '0x800A000D - type mismatch
    Response.Clear
    Response.Redirect "/" 'go to the Home page
End If

```

是否决定这样做依赖于你自己的情况以及你打算发现、记录或显示的错误类型。需要注意的是，因为我们不想把目前的网页环境传送到新的网页上，所以选择使用 Response.Redirect 语句而不用 Server.Transfer 语句。

(5) 显示错误信息

最后，显示错误报告和其他信息以及返回到上一个网页或主页的按钮。

```

<%
'see if the logging to file failed - if so display a message
If blnFailedToLog Then
    Response.Write "WARNING: Cannot log error to file " & strErrorLog & "<P>"
End If

'see if we are displaying the error information
If Session("ShowError") = "Yes" Then
'use HTMLEncode in case source code contains HTML characters
%>
    <PRE><% = Server.HTMLEncode(strDetail) %></PRE>
<%

```


放的位置，必须给文件设置相应的路径。

```
<% Server.Execute "/" & path_to_file & "/debug_Request.asp" %>
```

这是一种很好的方法，保证我们希望在 Request、Session 和 Application 集合中找到的任何值确实存在，并且包含了合适的值。在本书的示例文件 Chapterc07 子目录中提供了一个相应的文件，取名为 debug_Request.asp。它基本上是由于第 2 章的 show_request.asp 网页和用于第 3 章的 show_application.asp 和 show_session.asp 网页的一个组合，但删除了部分 HTML 程序代码。它只是简单地遍历了集合并把值放到当前页面中。

可以通过运行“Custom Error Page”实例来查看这个页面。这个实例在本章前而看到过，打开时请选中“Display debugging information”复选框，或者直接在 chapter07 目录中的主菜单网页中打开。

2. 显示中间值

在网页中查看运行情况的第二个方法是显示网页运行时变量的值。当大概知道了错误来自何处，哪个变量在起作用时，这种传统技术还是不能废弃的。但由于 IIS 5.0 网页缓冲方式的改变，使得使用这项技术比较困难。

在 ASP 和 IIS 的先前版本中，缺省时关闭页面缓冲，并且几乎没有人想到将缓冲打开（使用 Response.Buffer = True 打开），除非想使用 Response.Redirect 完成网页的再定向（参看第 2 章）。响应多个请求时，由于缓冲减小了网页间切换的次数，从而提高了 IIS 的效率。

然而，当出现一个使运行停止的运行期错误时，IIS 自动调用 Response.Clear 方法，再调用 Server.Execute 来装入定制错误网页，因此写进网页的任何输出都丢失了。解决方法是暂时增加下面的程序行：

```
<%Response.Buffer = False %>
```

此程序行放在页面顶部 <@LANGUAGE . . . > 指令后面，任何由 Response.Write 语句生成的调试输出将出现在定制错误网页的顶部。记住在完成网页调试之后将它去掉。

强行使程序运行通过一个错误点，然后显示可疑的变量值，这种方法有时也是有用的。只需在网页开始处附近增加 On Error Resume Next 语句，然后就能访问 Err 对象（在 VBScript 中），并显示错误号、错误源和描述。

3. 检查组件属性值

如果使用的组件具有在 ASP 脚本代码中设置的属性，在完成设置之后，并且调用组件方法之前和之后，能通过显示所有属性（或仅是可疑的属性）来跟踪错误。当一个方法运行时，可能发现属性值意外地被组件改变了，这或许是故意的，或者是因为组件中的缺陷。没有亲自检查实际代码，不要做任何假设。

7.5.2 Microsoft Script Debugger

当开发更复杂的处理实际任务的应用程序时，经常需要一个更加强大的工具来进行调试。Microsoft Script Debugger（微软脚本调试器）是一种允许调试运行在客户机和服务器上的脚本的调试工具。它能用于调试任何启用 ActiveX 的脚本语言（包括 VBScript 和 JScript）编写的程序，也能够用来调试对 Java applet、Java Bean 和 ActiveX 组件的调用。

在研究这个工具之前，先简要说明一些问题。如前所述，ASP 应用程序由两种脚本组成，一种是客户端脚本，一种是服务器端脚本。客户端脚本通常由 VBScript 或 JScript 脚本语句组成，

当其到达客户端时出现在 HTML 页面中并在此执行,可能是在载入文档时或是在对一些事件的响应中。服务器端脚本通常也由 VBScript 或 JScript 语句组成。当浏览器请求网页时,服务器端脚本由 IIS 执行。在下面的讨论中,将讨论服务器端脚本调试的方法。然而所讨论的许多技术也可用于客户端脚本调试。

1. 服务器端的调试

为了调试服务器端脚本,在运行 IIS 的计算机上运行脚本调试器,然而在使用脚本调试器之前,必须启用调试。为了使性能最优化,基于 ASP 的应用程序在缺省情况下关闭了调试功能。

注意,不要对生产性的应用程序(即处于活动状态的并被他人使用的公用网站)打开调试功能。这样会减慢整个应用程序的运行,并且错误能使网页出现不确定的停止运行情况。

调试仅能为虚拟应用程序和整个 Web 网站进行设置,为了启用调试,打开应用程序或站点的 Properties 对话框,在 Home Directory 选项卡中,点击 Configuration 按钮,在 Application Configuration 对话框的 App Debugging 选项卡中,选择 Enable ASP server-side script debugging”,如图 7-22 所示。下面准备调试我们的应用程序。

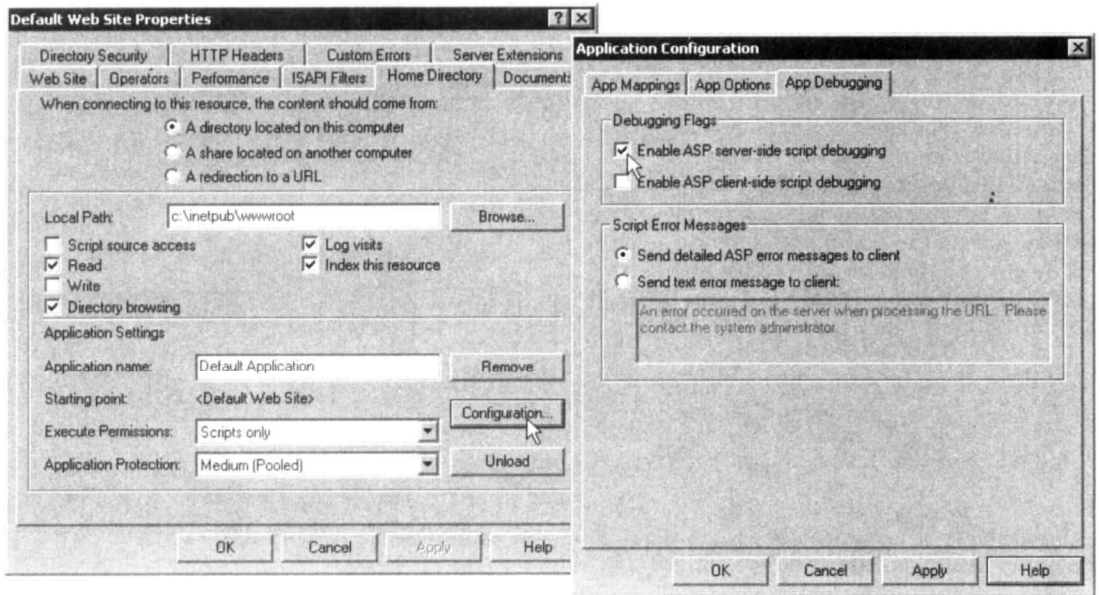


图 7-22 启用调试的屏幕

注意 Application Configuration 对话框包含一个复选框,能够启用客户端脚本调试。这一点在 IIS 5.0 中没有实现,在文档中仅标记为“reserved for future use”。如果通常的 500-100.asp 定制错误页面不可用,Script Error Messages 部分中包含将文本。

(1) 处理服务器脚本

不像客户端脚本,基于 ASP 的应用程序脚本不是事件驱动的。当客户端要求一个来自服务器的网页时,服务器读取网页内容,并处理所有的服务器脚本(即在<%....%>和<SCRIPT RUNAT="SERVER"></SCRIPT>段中的所有内容),也包括在 HTML 文本中的“行内”脚本段

内容,例如:

```
The valne of the result is : <% = strResult %>
```

处理流程显示在图 7-23 所示的框图中。

当 IIS 载入网页时将处理 ASP 页面中的所有脚本,在任何输出送给客户端之前,ASP 及脚本引擎能够捕获语法和运行期错误(除非你关闭缓冲或调用 Response.Flush 方法)。

(2) 脚本调试器提供的帮助

启用脚本调试时,如果出现错误,在服务器屏幕上可以看到一个描述 ASP 代码错误的对话框,点击 OK,然后调入当前 ASP 网页的一个只读拷贝,打开的脚本调试器,错误出现的行由箭头指示,如图 7-24 所示。

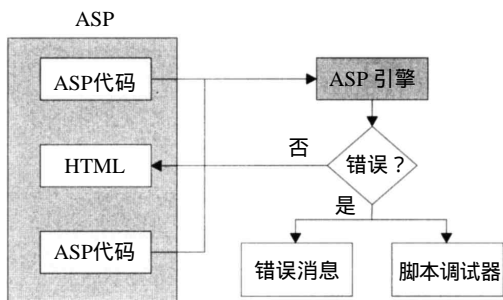


图 7-23 服务器脚本运行流程图

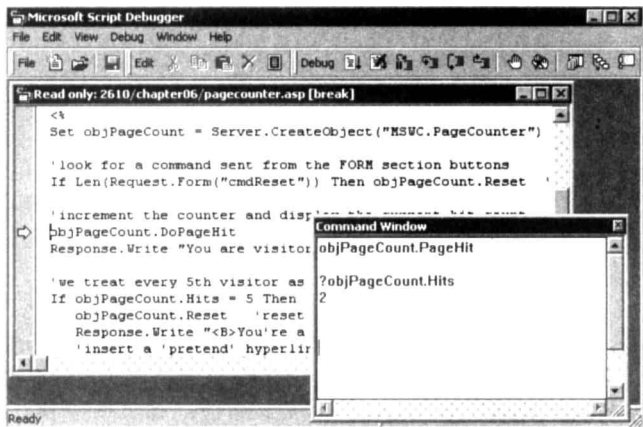


图 7-24 脚本调试器

这里,错误的产生是由于出现了 Page Counter 对象方法的名字错误,应是 DoPageHit 而不是 PageHit。同时,脚本调试器找到了错误并且终止了页面的运行,工具条上的按钮用于程序的继续运行、单步程序运行或者终止页面的处理。

工具条最右边的按钮打开脚本调试器中的 Immediate 窗口,可以用它和页面进行交互,并且很可能找到出错的地方。例如,可以查询或者设置变量值或组件属性,可以执行内部函数和子程序、自定义函数和子程序以及已经创建的对象方法等。在图中,调用了 Page Counter 组件的 PageHit 方法,然后查询 Hits 属性以得到正在运行的脚本中该处的值。

为了了解为什么在“公共”网站上不应使用脚本调试器,可以从客户机上打开一个包含服务器端错误的页面。在这种情况下,错误信息对话框出现在服务器上,脚本调试器也在服务器上打开。在客户机上,直到运行在服务器上的脚本调试器关闭,才开始载入该页面。

(3) 启动和使用调试器

启用脚本调试后,虚拟应用程序的网页中出现错误时,脚本调试器自动启动。还可以人

工启动脚本调试器，在 Windows 2000 的 Start 菜单 (Programs|Accessories|Microsof Script Debugger) 中完成。相应地，也可在想打开脚本调试器的地方把一个 Stop 语句插入 ASP 程序中，当运行至 Stop 语句时，IIS 会终止 ASP 程序的执行，启动脚本调试器，显示当前页面并指出含有 Stop 语句的当前行。

脚本调试器能完成下列工作：

- 查看正在或已经运行的文档的列表，并从中选择一个进行查看或编辑。
- 在打开的网页中设置一个新断点，页面在该点停止运行以便进行调试。
- 单步调试，一步运行一条语句，可选地执行子程序和函数。
- 查看调用栈 (Call Stack)，显示程序中在该点被调用的嵌套子程序或函数。

2. 脚本调试器的技巧和窍门

下面是使用 Microsoft Script Debugger 时，有助于找到脚本中错误的一些窍门。

- 如果调试服务器端脚本，为 ASP 应用程序启用脚本调试器。否则，错误信息将作为文本传送给客户端的浏览器，并且不能对服务器端脚本使用脚本调试器。
- 调试完成后，关闭调试功能，否则会降低服务器性能，并且错误的页面会停留在客户端。
- 对于一个或更多的 ASP 应用程序，如果启用脚本调试器，将传送给它所有的服务器错误，包括那些远程客户访问网页时出现的错误。因此，除非能在自己的服务器上调试，否则不要启用脚本调试器。
- 如果在一个没有安装在服务器上的浏览器中工作，并且在网页中显示错误，则错误在服务器端脚本中。如果一个错误信息出现在对话框中，则错误在客户端脚本中。
- 如果在 .asp 文件中有一个语法或运行期错误，并且已经对这个 ASP 应用程序启用调试功能，客户端浏览器将不显示语法错误（除非浏览器运行在服务器上），仅显示超时或不能打开网页。
- 在表明服务器端脚本中是否有错误的消息中，显示的行号指的是包含这个错误的 .asp 文件的相应行。
- 如果在由 .asp 文件创建的客户端脚本中有错误，行号并不指向 .asp 文件的错误行，而是指向错误出现的 .asp 文件的 HTML 输出行。为了查看这行，应在客户浏览器中查看 HTML 文件的源程序。
- VBScript 和 JScript 错误代码在附录 D 中。

7.5.3 获得 ASP 的帮助和支持

如果遇到一个不能处理的错误，或者看来像 ASP 中的一个 “bug” 的事情，最好能够寻求帮助以解决这个问题。关于 ASP 在 Web 上有许多有用的信息源，第 1 章后面我们列举了许多。但是，对一些特殊问题，确实需要一些更直接的帮助。

在计算机上安装的 ASP 和 IIS 文档是一个好起点，并且能通过浏览器的 URL 为 <http://yourservername/iishelp/> 进行访问。运行 Windows 2000 的附加组件设置或主设置程序（依赖于安装的 Windows 2000 版本）时，应保证安装了全部的文档。

也可以从 “Microsoft for Windows 2000” 得到的完整平台 SDK，其中包含了大量的关于 Windows 和 Windows 2000 中 Internet 服务的附加信息。它包含完整的 VBScript 和 JScript 参考。

它可以微软获得，并提供给 MSDN 成员。脚本参考文档可单独获得，也可以从 “ <http://www.microsoft.com/scripting/> ” 网站下载。也可从这里获得脚本调试器。

微软开发者网络(The Microsoft Developer's Network, MSDN) Web 站点也提供了许多支持和帮助，即使这部分信息有时难以找到。还可以从 Workshop 网站(<http://msdn.microsoft.com/workshop/>)开始，这个网站在左边导航栏中有很好的索引和一系列标题。

另外，位于 <http://msdn.microsoft.com/Library/default.htm> 的主 MSDN 库包含有文章、基础知识、FAQ 和其他用于 IIS 和 ASP 的支持材料，左边窗口使用一个 Java 扩展列表控件，使用户很容易进行查找。

如果需要特别的帮助，或者需要向其他开发者提出一些问题，在 msnews.microsoft.com 网站上有一些有用的新闻组。还可以订阅 microsoft.public.inetserver.iis、microsoft.public.inetserver.activeserverpages 和 microsoft.public.inetserver.iis.misc。一些 ASP Web 网站也提供有关 ASP 方面的讨论话题、论坛或聊天室。

7.6 小结

本章讨论的内容是大多数编程者最不喜爱的工作。即使最简单的脚本，也不可能就能第一次正确地工作。随着 ASP 提供越来越多的特性，在脚本中出现错误的机会也增加了。了解如何发现并处理错误的相关基本知识，是非常必要的。

通过分析可能出现的不同种类的错误，弄清楚缺省的 ASP 和脚本引擎错误处理系统捕获错误的机理，有助于我们更容易地跟踪错误。随着 ASP 的成功使用，ASP 脚本和运行环境更复杂，产生错误的可能性也提高了。

在研究错误是什么和来自哪里的同时，也介绍了一些预防性编程的方法以便能尽早地发现问题，防止把错误和无效数据传给其他的应用程序。编程时出现的错误越少，越容易发现和解决它们。

然而，好的编程习惯并不能阻止某些种类的错误发生，例如那些由外部资源和服务造成的错误。这意味着提供自己的定制错误处理代码，在出现错误时知道如何跟踪错误并进行妥善处理，是非常重要的。

最后，以对 Microsoft Script Debugger 的阐述结束了本章，它是有助于找到和解决网页中的运行期错误的一个有用工具。它可以暂停程序执行和进行单步执行，同时还能够观察程序在进行什么，甚至能够和脚本进行交互。

因此，本章主要覆盖如下内容：

- 能够出现的错误的类型。
- 如何防止各种错误出现。
- 如果不能防止错误发生，如何妥善处理错误。
- 如何发现和处理脚本错误及其他类型的错误。
- 如何使用定制错误页面获得错误信息。
- 如何记录发生的错误以监视我们的网站。

本书的下面章节将回过头来再次讨论服务器组件。我们将深入讨论 ActiveX Data Objects(ADO) 组件，并熟悉如何使用 ADO 访问数据库和各种类型的数据存储。