

第5章 脚本运行期库对象

前面章节已经介绍了 ASP 如何使用在服务器上定义的对象实例，并充分利用所提供的方法和属性扩展 ASP 的性能。有一系列的对象可供使用，包括脚本对象和标准 IIS/ASP 安装的组件，以及自己创建的或者从其他供应商处购买的对象。也可以在互联网从各种网站免费下载对象，并在自己的页面上使用。

这一章将讨论由 ASP 脚本环境提供的一般称为“脚本运行期库” (Scripting Runtime Library) 的对象。这些对象通过正在使用的脚本引擎提供给代码，与 ASP 脚本程序一起完成多种实用任务。

还有一种组件是“活动服务器组件” (Active Server Component)，通过单独的 ActiveX DLL 文件或者其他文件来实现。后面章节将讨论相关内容。

当然，需要研究如何在页面上使用这些对象。在前一章中，我们已经了解了服务器如何提供一个方法来实例化对象，本章将深入讨论这个内容。

本章将介绍以下内容：

- 脚本引擎以脚本对象方式提供了什么。
- 如何创建脚本对象及其他组件实例。
- 脚本对象的成员和属性概要。
- 如何在代码中使用脚本对象。

下面开始研究脚本对象的定义。

5.1 脚本对象的定义

前面章节研究了 ASP 对象模型。

对象模型是用来理解系统的各个部分相互关系的一种基本手段。

ASP 对象模型提供了一种结构，用来作为一个整体操纵 HTTP 请求、响应及 ASP 环境中的不同元素。例如，我们已经看到，如何通过查看 ASP 请求对象的 cookie 集合，得到来自浏览器的任何 cookie 值。

我们使用的脚本语言也有对象模型。然而，脚本语言提供的这一对象模型，不同于由 ASP DLL 直接提供的对象模型，脚本对象是由 Microsoft 脚本运行期库 (sccrun.dll) 提供的，安装缺省的 Active Scripting 脚本引擎时，也安装了 Microsoft 脚本运行期库。

5.1.1 不同类型的对象和组件

不要对“对象”和“组件”这两个名词感到困惑，在一定范围内它们都可以作为 ASP 的一部分，同样可以通过 COM 对其进行访问。从概念上可以将它们分为四类：

- ASP 内置对象，如 ObjectContext、Request、Response、Application、Session、Server 和 ASPError。本书的第 2 章到第 4 章已经研究了这些内容。

- 脚本对象。通过脚本运行期库使用，如 Dictionary、FileSystem和TextStream。这是本章要讨论的对象。
- 可安装的组件。由Microsoft在IIS 5.0和ASP 3.0标准安装时提供。这将在下一章讨论。
- 其他组件。从其他独立厂商购买的、在网站上发现的或者自己创建的组件。还有一些其他的由Windows服务或产品提供的组件，如Windows Scripting Host。在本书的附录中提供了相应的列表，本书专门有一部分章节讲述如何构建自己的组件。

5.1.2 VBScript和JScript脚本对象

作为脚本运行期库的一部分，Microsoft提供三个主要的对象：

- Dictionary对象提供一个极为有用的存储对象，它用来存储值，通过对象的名字而不是其索引进行访问和引用。例如，对于存储从ASP Request对象中检索到的名称/值对，这是非常合适的。
- FileSystemObject对象提供了对服务器底层文件系统的访问（在客户端上使用IE5.0，与名为“Hypertext Application(HTA)”的特殊类型的页面协同使用）。可用FileSystemObject对象遍历计算机的本地及网络的驱动器、文件夹和文件。
- TextStream对象提供对存储在磁盘上文件的访问，用于同FileSystemObject对象协同使用。TextStream对象能够读出或写入文本（顺序的）文件，并仅能通过FileSystemObject对象进行实例化，所以人们常常认为TextStream对象是FileSystemObject对象的子对象。

FileSystemObject对象是其他一系列用来与文件系统交互的对象和集合的“父代”。该对象提供了对象的三个集合：Drives、Folders和Files集合，每个集合分别是相应的Drive、Folder和File对象的集合。它们用来进行磁盘上的驱动器、文件夹（目录）和文件的遍历和定位。对象间的关系如图5-1所示。

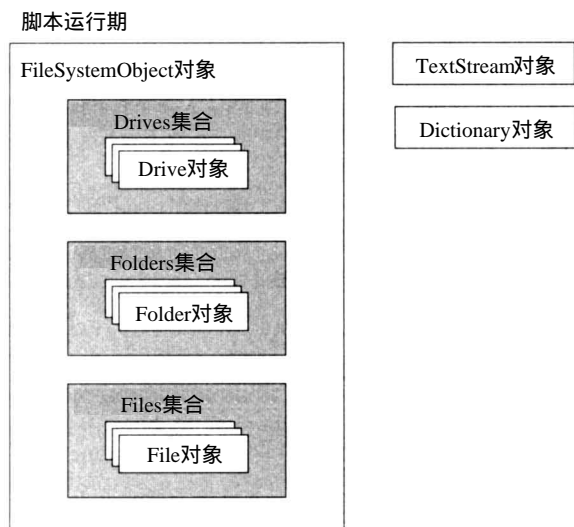


图5-1 脚本运行期库中对象间关系

下面，将依次介绍这些对象和集合，以及如何使用它们。然而，首先要理解对象实例与组件的创建或实例化方式之间的差异。这是下一节的主要内容。

5.2 创建对象和组件实例

创建脚本运行期库对象的实例与创建任何其他对象和组件的实例化方式完全相同。可使用ASP Server对象提供的CreateObject方法(确保对象创建在当前页面的环境内), 或者使用一个<OBJECT>元素。我们将研究这两种方法, 究竟采用那种方法依赖于页面的需要。

5.2.1 使用Server.CreateObject方法

正如在研究Server对象的时候看到的, 组件或其他对象实例可根据它们的 ProgID来创建:

```
<%
Dim objThis
Set objThis = Server.CreateObject("ADODB.Connection")
%>
```

ProgID字符串“正式的”格式是“供应商.组件.版本”, 供应商的名字和版本是可选的。通常ProgID只包含前两部分(如上例)。少数供应商在ProgID中设置版本编号, 这将避免向后兼容的新版本使用同样的ProgID, 这要求改变ASP页面才能使用新版本。

5.2.2 使用<OBJECT>元素

可以使用标准的HTML<OBJECT>元素通过增加RUNAT参数并指定其值为“SERVER”来在服务器上创建一个组件实例。另外, 通常是提供对象的 ProgID字符串而不是数字的ClassID:

```
<OBJECT ID="objThis" RUNAT="SERVER"
  PROGID="This.Object">
  <PARAM NAME="param1" VALUE="value1">
  <PARAM NAME="param2" VALUE="value2">
</OBJECT>
```

如果上面脚本的对象有相应的属性可在脚本中使用, 在 <OBJECT>元素内可通过<PARAM>元素进行设置, 就像通常在HTML页面中所做的一样。在APS中使用<OBJECT>元素时不要求CODEBASE属性, 当其不可用时, 服务器不会试图下载以及安装对象或对象。

1. 指定一个ClassID

另外, 可以指定想要创建的对象或组件的 ClassID。在不知道目标机安装了什么其他组件的情况下, 这是非常有用的。例如在客户端上的浏览器的页面上实例化组件时。

在理论上, 组件的ProgID(文本“供应商.组件”)不应该相互冲突, 应该是唯一的。然而, 这不是无懈可击的。有可能美国北方的一个供应商与希腊小岛上的一个供应商同名。但是, 使用ClassID识别访问时, 因为ClassID是唯一的, 同名情况就不会发生。

如果决定使用对象或组件的ClassID, 应将其放入CLASSID属性中, 而不是PROGID属性。如:

```
<OBJECT ID="objThis" RUNAT="SERVER"
  CLASSID="clsid:892D6DA7-E0F9-11D2-B2E9-00105A42AF30">
  <PARAM NAME="param1" VALUE="value1">
  <PARAM NAME="param2" VALUE="value2">
</OBJECT>
```

但在自己的服务器上实例化对象时, 应该知道对象和组件的安装方式。这样在 ASP代码

中创建对象实例时,可以安全地使用 ProgID。这就是 ClassID 很少在 ASP 页面内使用的原因。然而,因为 ProgID 用于查找 CLSID,如果愿意也可以用组件或对象的 ClassID 代替 ProgID。

2. 设置对象实例的作用域

缺省情况下,所有在 ASP 页面中创建的对象与组件实例(无论用 Server.CreateObject 方法或 <OBJECT> 元素)都有页面内的作用域(page scope)。这意味着,对象与组件只有该页在 ASP 上运行时才存在,当页面完成并且把结果发送到客户端以后就自动地取消了。

然而,如果在 global.asa 文件(它存在于站点或虚拟应用程序的根目录)中放置 <OBJECT> 声明,可以将对象或组件的作用域指定为应用程序或会话作用域。

(1) 在应用程序层作用域创建对象

通过设置 SCOPE 属性为“APPLICATION”,创建应用程序层作用域对象:

```
<OBJECT ID="objThis" RUNAT="SERVER" PROGID="This.Object"  
  SCOPE="APPLICATION">  
</OBJECT>
```

应用程序开始时就创建了对象实例,即一旦用户从虚拟应用程序的目录请求一个页面,就创建对象实例。对于缺省 Web 站点,这可以是站点上的任一目录。直到应用程序结束(最后的用户会话结束)前,对象实例一直存在,并且可以被虚拟应用程序或站点目录内任一页面内的任意用户引用和使用。

(2) 在会话层作用域创建对象

如果想创建由单个用户使用的对象实例,其作用域为他访问的所有页面,可创建会话层作用域对象。这通过将 SCOPE 属性设置为“SESSION”来实现:

```
<OBJECT ID="objThis" RUNAT="SERVER" PROGID="This.Object"  
  SCOPE="SESSION">  
</OBJECT>
```

对象一旦被引用就被创建,引用是由用户从虚拟应用程序或站点载入的页面内的程序代码完成的(在 global.asa 文件中有 <OBJECT> 声明)。当用户会话生命周期结束并被取消时,它引用的对象实例也就取消了。

(3) 关于作用域和状态

使对象实例的作用域为全局的或者为用户会话全局环境看起来是一个好主意,但在实际使用时有些问题需要考虑,其中之一是在用户的许多请求之间能够有效地保护对象的状态。换句话说,可以设定对象的一些属性,它们对使用的所有页面是共用的。因为不必每次都创建新的实例并设置其属性,所以这看起来是个较好的办法。

事实上,微软建议一般情况下不要这样做,这一思想是传统程序设计思想的残余。在 Web 上,要面对的最大问题是服务器以及 Web 应用程序及所提供的动态网页如何应付数以百万计的网站访问者。将组件实例驻留在内存中等待一个特定用户的页面请求,对可能有几百个用户同时浏览的网站来说,这样做不能有效地使用资源。

Window 2000 提供新的 COM+ 运行期特性,它能够处理组件的创建、缓存和使用,采用一种吞吐量最大化但所占服务器资源最小化的方式。对象实例存储在哪里和存储多久的问题,最好由操作系统自己完成,而不是由程序员决定。

也就是说,在页面内需要的地方创建对象实例,当页面终止时让其消失。COM+ 整理这些碎片,自动处理后台的一些复杂工作。如果要了解有关这方面的内容,第 14 章比较详细地

研究了组件的创建。

当然,在某种情况下,我们可能要求一个对象具有应用程序层和会话层的作用域,尤其是在页面请求间保存状态时。在后面讨论 Dictionary对象时,将有一个这方面的实例。

5.2.3 Server.CreateObject与<OBJECT>的区别

Server.CreateObject方法立即创建一个对象实例。在大多数情况下这也是我们所希望的。而<OBJECT>元素只有首次引用一个对象时才创建指定的对象实例。因此如果在代码中停止使用该对象,则不创建该对象实例。

如果代码只在某种情况下使用这个对象(可能依赖于请求参数的值),这也许是有用的。因为如果不需要这个对象,则可以节省服务器的资源。

然而,如果肯定需要创建某一对象,可使用 Server.CreateObject方法完成。用<OBJECT>元素创建对象有助于防止在代码中取消对对象的调用时,忘记取消程序中的 Server.CreateObject行,当然这是一个粗心的程序设计。

最后需要记住的是,如果对象是使用 Server.CreateObject方法创建的,就可以从会话或应用程序中去掉对象,但使用<OBJECT>元素创建的,则不行。

5.2.4 组件线程模型

在页面内使用对象或组件时,应该考虑的另一个问题是该对象涉及到的响应多个请求的行为方式。事实上在 ASP里,这是所需要理解的最复杂的题目之一。一个组件的线程模型,结合其作用域,影响该组件和应用程序的性能和效率,也影响将它实例化的 ASP页面。

线程就是由处理器执行的系统对象,用于完成由组件代码定义的任务。每一个线程都可以被认为是单个二进制指令集。在像 Windows这样的多线程环境中,多个线程可同时运行。

实际上有五个线程模型(包括在 Windows 2000里引入的 Neutral-threading模型):

- Single-threaded(单线程):某一时刻只能有一个进程使用某组件。
- Apartment-threaded(单元线程):若干进程都可以使用某组件,但只有一个在指定的线程上。
- Neutral-threaded(中立线程):若干进程都能使用某组件,并且可以使用指定的一组线程中的任何一个。
- Multiple-threaded或Free-threaded(多线程或自由线程):若干进程都能使用某组件,并且这些进程可以运行在不同的线程上。
- Both-threaded(双线程):对象既可以是单元线程的又可以作为自由线程的。

在这里不解释线程模型的技术细节,本书后面有相应的内容。

单元线程的组件(例如使用 Visual Basic创建的或作为 XML脚本的组件)可在页面层作用域内很好地运行,在会话层作用域内也是可以接受的。事实上,在页面层,由于较低的数据处理开销,也能很好地运行双线程的组件。

Windows 2000中的中立线程的模型甚至提供了更好的性能,尽管到目前为止只有很少的这样的组件和与之相适应的开发工具。

如果需要会话层组件，使用可用的双线程的组件。并且如果需要应用程序层作用域，可一直使用双线程的组件。

然而，微软建议避免使用会话层作用域的组件，甚至不使用应用程序层作用域的组件，除非这些组件是绝对需要的。使组件的活动时间超过作用域为页面级的组件所要求的时间，对于由COM+提供代理特性的对象是没有益处的。

5.2.5 引用对象类型库

在早先的ASP版本中，在脚本中使用对象或组件时，组件内定义的公共常数（如果有的话）在ASP里将不再有效。这意味着我们需要自己声明它们(或等价物)并指定相应的值。

例如，当在早期版本的ASP中使用ActiveX数据库对象(ADO)组件时(将在第8章进行详细的研究)，不得不用记录集的Open方法加入预定义常数声明。例如：

```
Const adOpenKeyset = &H0001
Const adLockPessimistic = &H0003
Const adCmdTable = &H0002
...
rs.Open "Contact", "DSN=GlobalExampleData;UID=examples;Password=;", _
    adOpenKeyset, adLockPessimistic, adCmdTable
...
```

另一种方法是使用#include指令在页面插入一个名为adovbs.inc的文件。该文件由IIS/ASP提供，包含ADO所需的所有预定义常数。更新代码时，必须确认使用的是最新版本，并检查它对于所有的页面请求都可用。

对于IIS 5.0，有一个更好的方法，通过在HTML注释元素内使用METADATA指令，可以给组件或对象的类型库增加引用(IIS 4.0不支持这一功能)。

```
<!-- METADATA TYPE="TypeLib"
      FILE="path_and_name_of_file"
      UUID="type_library_uuid"
      VERSION="major_version_number.minor_version_number"
      LCID="locale_id"
-->
```

其中：

- *path_and_name_of_file* 是某一类型库文件(.tlb)或ActiveX DLL的绝对物理路径，必须提供这一参数或者是*type_library_uuid*参数。
- *type_library_uuid*是该类型库的唯一标识符，必须提供这一参数或者是 *path_and_name_of_file*参数。
- *major_version_number.minor_version_number* (可选)定义了所需组件的版本。如果没有该版本则使用最近的版本。
- *locale_id* (可选)是区域标志符。如果在该区域没有发现类型库，计算机将使用缺省的(安装时定义的)区域。

因此，使用这一技术，通过使用下面的代码，能使内置的 ADO预定义常数在ASP页面可用：

```
<!-- METADATA TYPE="TypeLib"
      FILE="c:\Program Files\Common Files\System\ado\msado15.dll"
-->
```

文件名msado15.dll还可用于更高版本(2.50以后)的ADO组件。

如果ASP不能装载类型库，就返回一个错误并停止该页的执行。可能的错误提示如表 5-1 所示。

表5-1 错误提示代码及说明

错 误	说 明
ASP 0222	无效的类型库说明
ASP 0223	未找到类型库
ASP 0224	类型库不能加载
ASP 0225	类型库不能打包(即ASP不能从指定的类型库中创建类型库包装对象)

5.2.6 在客户端上创建对象实例

在ASP中讨论在服务器上实例化对象和组件的技术时，值得强调的是在浏览器中运行客户端页面而完成同样工作的方式。如果你使用 ASP创建包含客户端脚本程序的页面，或者使用<OBJECT>元素创建客户端组件实例，将会发现这是非常有用的。在大多数情况下，脚本运行期对象可在客户端上实例化和使用，效果与服务端上的 ASP相同。

1. VBScript CreateObject 方法

在客户端使用CreateObject时，在浏览器的环境内创建组件或对象实例，它们与浏览器运行在相同的内存空间里(即进程内)，除非实现的对象是带有.exe扩展文件名的可执行文件。

通常指定对象的 CIassID，而不是使用 ProgID字符串，这样就不可能与其他安装在客户端的对象发生冲突。

```
<SCRIPT LANGUAGE="VBScript">
Dim objThis
Set objThis = CreateObject("clsid:892D6DA7-E0F9-11D2-B2E9-00105A42AF30")
...
</SCRIPT>
```

当然也可以使用ProgID ,并且使用通用的对象或组件(特别是标准安装提供的对象或组件)，那么得到错误的组件的风险是很小的：

```
<SCRIPT LANGUAGE="VBScript">
Dim objThis
Set objThis = CreateObject("Scripting.Dictionary")
...
</SCRIPT>
```

2. JScript ActiveXObject方法

为了在客户端上实例化JScript的对象和组件，必须使用 ActiveXObject方法和new操作符：

```
<SCRIPT LANGUAGE="JScript">
var objMyData = new ActiveXObject('clsid:892D6DA7-E0F9-11D2-B2E9-00105A42AF30');
</SCRIPT>
```

或：

```
<SCRIPT LANGUAGE="JScript">
var objMyData = new ActiveXObject('this.object');
</SCRIPT>
```

3. <OBJECT>元素技术

也可使用<OBJECT>元素创建客户端对象或组件的实例。应省略 RUNAT属性或者将其设

定为“CLIENT”。然而，这个属性在客户端上是被忽略的，因此设置这个属性的唯一目的就是，在ASP页面使用<OBJECT>元素实例化服务器端的组件实例时防止混淆。

```
<OBJECT ID="objThis" RUNAT="CLIENT"
  CLASSID="clsid:892D6DA7-E0F9-11D2-B2E9-00105A42AF30"
  CODEBASE="http://yourserver.com/components/mycomponent.cab">
  <PARAM NAME="param1" VALUE="value1">
  <PARAM NAME="param2" VALUE="value2">
</OBJECT>
```

注意，这里出现的CODEBASE属性，表示允许下载并安装来自URL的组件(如果该组件没有安装)。IE 3.0以上的版本有此功能。

对于使用<OBJECT>元素的方法、可使用的属性、在客户端使用时的值，可查看网站 <http://msdn.microsoft.com/workshop/author/dhtml/reference/objects/OBJECT.asp>，或者Windows 2000 Platform SDK文档中的<OBJECT>tags”，或者看看《IE5 Dynamic HTML Programmer' Reference》一书，ISBN 1-861001-74-6，Wrox 出版社。

5.3 Scripting.Dictionary对象

许多Microsoft的编程语言，如Visual Basic、VBScript和JScript，都提供集合(collection)。可以把集合想象为数组，可以使用其中内建的函数完成存储和操纵数据等基本任务。无须担心数据是在哪些行列，而是使用唯一的键进行访问。

VBScript和JScript都提供类似的对象，通称Scripting.Dictionary对象或Dictionary对象。它类似于二维数组，把键和相关条目的数据存放在一起。然而真正的面向对象的方式，不应直接访问数据条目，必须使用Dictionary对象支持的方法和属性来实现。

本章提供了一些示例页面，允许试验脚本运行期对象的方法和属性。这些实例在下载的文件中的Chapter05子目录里。

5.3.1 创建和使用Dictionary对象

创建一个Dictionary对象的示例如下：

```
' In VBScript:
Dim objMyData
Set objMyData = Server.CreateObject("Scripting.Dictionary")

// In JScript:
var objMyData = Server.CreateObject('Scripting.Dictionary');

<!-- Server-side with an OBJECT element -->
<OBJECT RUNAT="SERVER" SCOPE="PAGE" ID="objMyData"
  PROGID="Scripting.Dictionary">
</OBJECT>
```

Dictionary对象还可用于客户端的IE中。

1. Dictionary对象的成员概要

表5-2和表5-3列出了Dictionary对象的属性和方法及相应的说明。

当增加一个键/条目时，如果该键已存在；或者删除一个键/条目时，该关键字/条目对不存在；或改变已包含数据的Dictionary对象的CompareMode，都将产生错误。

表5-2 Dictionary对象的属性和说明

属 性	说 明
CompareMode	(仅用于VBScript)设定或返回键的字符串比较模式
Count	只读。返回 Dictionary里的键/条目对的数量
Item(key)	设定或返回指定的键的条目值
Key(key)	设定键值

表5-3 Dictionary对象的方法和说明

方 法	说 明
Add(key, item)	增加键/条目对到 Dictionary
Exists(key)	如果指定的键存在，返回 True，否则返回 False
Items()	返回一个包含 Dictionary对象中所有条目的数组
Keys()	返回一个包含 Dictionary对象中所有键的数组
Remove(key)	删除一个指定的键/条目对
RemoveAll()	删除全部键/条目对

2. 对Dictionary中增加和删除条目

一旦得到一个新的(空的) Dictionary，可以对其添加条目，从中获取条目以及删除条目：

```
' In VBScript:
objMyData.Add "MyKey", "MyItem"
objMyData.Add "YourKey", "YourItem"
blnIsThere = objMyData.Exists("MyKey")
strItem = objMyData.Item("YourKey")
objMyData.Remove("MyKey")
objMyData.RemoveAll

'Add value MyItem with key MyKey
'Add value YourItem with key YourKey
'Returns True because the item exists
'Retrieve value of YourKey
'Retrieve and remove YourKey
'Remove all the items
```

在JScript中，等价的代码为：

```
// In JScript:
objMyData.Add('MyKey', 'MyItem');
objMyData.Add('YourKey', 'YourItem');
var blnIsThere = objMyData.Exists('MyKey');
var strItem = objMyData.Item('YourKey');
var strItem = objMyData.Remove('MyKey');
objMyData.RemoveAll();

// Add value MyItem with key MyKey
// Add value YourItem with key YourKey
// true because the item exists
// Retrieve value of YourKey
// Retrieve and remove YourKey
// Remove all the items
```

3. 修改键或条目的值

可以通过修改键的值，或通过修改与特定的键关联的条目的数据，来改变存储在 Dictionary内的数据。下面的代码改变键为 MyKey的条目中的数据。

```
objMyData.Item("MyKey") = "NewValue"      ' In VBScript
objMyData.Item('MyKey') = 'NewValue';    // In JScript
```

如果指定的键在 Dictionary未找到，将在 Dictionary中创建一个以 MyKey为键，以 NewValue为其条目值的新的键/条目对。有意思的是，如果使用一个不存在的键来检索条目，不仅得到一个空的字符串(这是可以想到的)，而且还在 Dictionary里添加一个新的键/条目对，键即是指定的键，但条目的数据为空。

可以使用 Key属性仅改变键的值而不改变与之对应的条目的数据。将一个已存在的键 MyKey改变为MyNewKey，可以用：

```
objMyData.Key("MyKey") = "MyNewKey"      ' In VBScript
```

```
objMyData.Key('MyKey') = 'MyNewKey';    // In JScript
```

如果指定的键未找到，则产生运行期错误。

4. 设置比较模式

Dictionary的CompareMode属性仅适用于VBScript，不能在JScript中使用。当比较字符串键时，允许指定比较的方式。两个允许的值为 BinaryCompare(0)和TextCompare(1)。BinaryCompare(0)为二进制数对照(即区分大小写)；TextCompare(1)为文本对照(即不区分大小写)。

5. 遍历Dictionary

研究Dictionary时，有两个方法和一个属性需要特别注意，它们允许我们遍历存储在Dictionary里的所有键/条目对。Items方法用一个一维数组的形式返回 Dictionary里所有的条目数据，而keys方法用一个一维数组返回所有已存在的键值。可以使用 Count属性得到键或条目的数量。

例如，可以使用下列代码得到名称为 bjMyData的Dictionary中所有的键和条目值。注意，虽然Count属性保存了在Dictionary里的键/条目数量，但VBScript和JScript的数组总是从下标0开始的。因此，数组下标应从0到Count - 1。

```
' In VBScript:
arrKeys = objMyData.Keys           'Get all the keys into an array
arrItems = objMyData.Items         'Get all the items into an array

For intLoop = 0 To objMyData.Count - 1 'Iterate through the array
    strThisKey = arrKeys (intLoop)    'This is the key value
    strThisItem = arrItems (intLoop)  'This is the item (data) value
Next

// In JScript:
// Get VB-style arrays using the Keys() and Items() methods
var arrKeys = new VBArray(objMyData.Keys()).toArray();
var arrItems = new VBArray(objMyData.Items()).toArray();

for (intLoop = 0; intLoop < objMyData.Count; intLoop++) {
    // Iterate through the arrays
    strThisKey = arrKeys[intLoop];    // This is the key value
    strThisItem = arrItems[intLoop];  // This is the item (data) value
}
```

在VBScript里也可以使用For Each . . . Next 语句完成同样的功能：

```
' Iterate the dictionary as a collection in VBScript
For Each objItem in arrItems
    Response.Write objItem & " = " & arrItems(objItem) & "<BR>"
Next
```

5.3.2 Dictionary对象示例

本书提供了一系列示例文件可用来试验脚本运行时间库的各种属性。

本章代码的缺省页面提供了一系列可使用的 VBScript示例链接。有些示例对JScript同样有效。这些示例存放在Chapter05目录下相应的子目录里，显示的界面如图5-2所示。

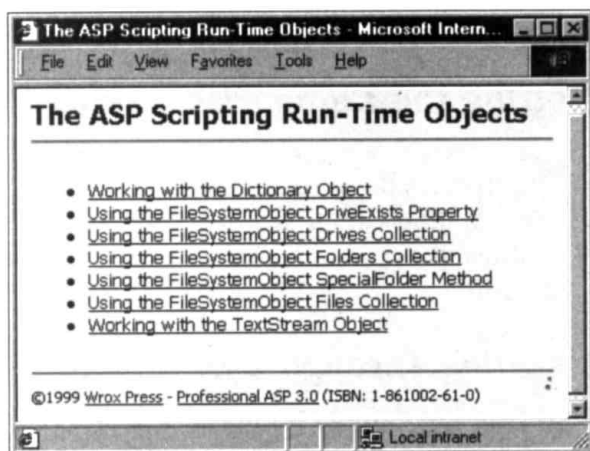


图5-2 ASP脚本运行期对象示例页面

要查看 Dictionary 对象的运行，在菜单页面点击第一个链接，打开名叫 show_dictionary.asp 的页面。这个页面显示了我们提供的 Dictionary 对象的内容，允许试验其属性和方法。屏幕如图 5-3 所示。

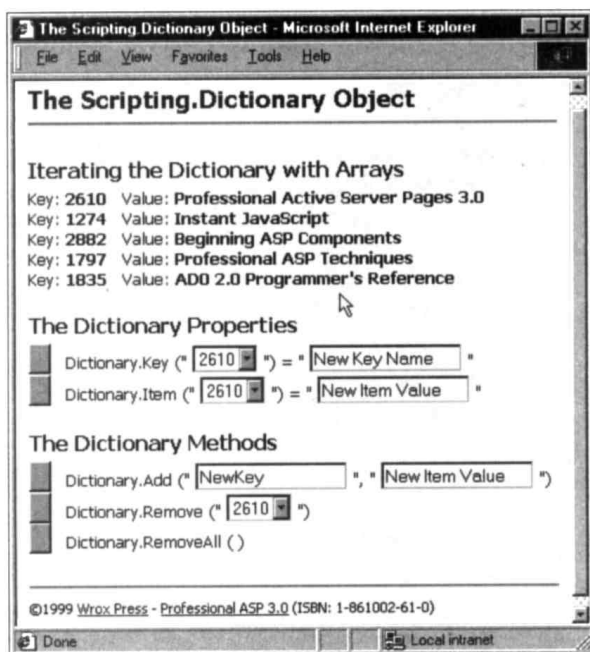


图5-3 Dictionary对象的属性和方法

1. Dictionary的global.asa文件

随 Dictionary 对象示例页面提供的文件之一是 global.asa。它创建并预先填充了一个会话层作用域的 Dictionary 对象，因此其内容在页面请求之间不会丢失。一般说来（考虑到可扩展性），这不是一个理想的做法。在这个例子里，可以看到 Dictionary 的属性和方法的效果。

如果在自己的服务器上下载并安装示例，必须创建一个基于此 global.asa 文件的虚拟应用

程序。或者将其内容添加到缺省站点的根文件夹中的 global.asa 文件里。在第3章讲述了如何用向导创建虚拟应用程序。然而对于本示例，创建一个虚拟应用程序最简单的方法是在 Chapter05 示例文件夹内右击 dictionary 子文件夹，在 Properties 对话框的 Home Directory 选项卡里，点击 Create 按钮，如图 5-4 所示。

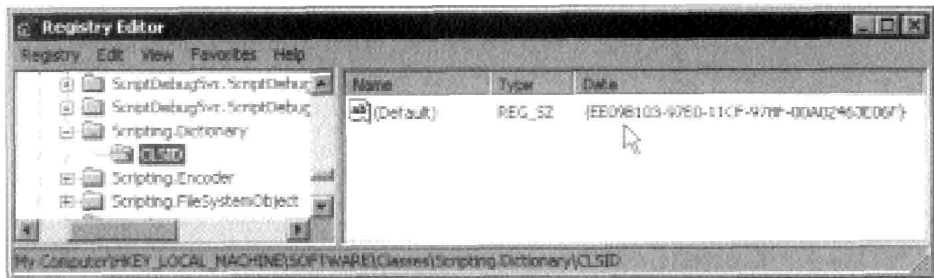


图5-4 创建虚拟应用程序

在这个 global.asa 文件里，代码使用 <OBJECT> 元素创建一个会话层作用域的 Scripting.Dictionary 对象实例。然后在 Session_onStart 事件处理程序里将一系列值用 Add 方法放入 Dictionary 中，并将对 Dictionary 对象的引用指定给 ASP 会话变量 MyDictionary：

```
<OBJECT ID="objBookList" RUNAT="SERVER" SCOPE="SESSION"
    PROGID="Scripting.Dictionary">
</OBJECT>

<SCRIPT LANGUAGE="VBScript" RUNAT="SERVER">

Sub Session_onStart()
    objBookList.Add "2610", "Professional Active Server Pages 3.0"
    objBookList.Add "1274", "Instant JavaScript"
    objBookList.Add "2882", "Beginning ASP Components"
    objBookList.Add "1797", "Professional ASP Techniques"
    objBookList.Add "1835", "ADO 2.0 Programmer's Reference"
    Set Session("MyDictionary") = objBookList
End Sub

</SCRIPT>
```

2. Dictionary 示例页面

在 “Scripting.Dictionary Object” 主页面里，首要的任务是得到一个会话层作用域的 Dictionary 对象实例的引用。注意，这个引用是一个对象变量，因此必须在 VBScript 里使用 Set 关键字。

然后，检查一下是否得到了一个对象（这是个好习惯），如果没有正确地建立包含 global.asa 文件的虚拟应用程序，检查一下问题出在哪里。你将看到我们自己的消息代替了 ASP 的错误消息（但是注意，对于这一操作必须关闭缺省的错误处理）。

```
<%

On Error Resume Next    'turn off default error handling

' Retrieve Dictionary object from user's session
Set objMyData = Session("MyDictionary")

If IsObject(objMyData) Then ' Found Dictionary object in Session

...

```


4. 使用Dictionary的属性和方法

在“Scripting.Dictionary Object”页面，点击用来检查并改变条目的Key属性的按钮，如图5-5所示。

把窗体再次提交给页面。该页面包含一个脚本段，检查被点击的按钮的值。它通过在Request.Form集合里查找按钮的名字来断定单击的是哪个按钮。如果发现一个对应于cmdChangeKey的值，则从列表中或文本框中得到相应的值并用来改变Key属性：

```
...
'Look for a command sent from the FORM section buttons
If Len(Request.Form("cmdChangeKey")) Then
    strKeyName = Request.Form("lstChangeKey")
    strNewKey = Request.Form("txtChangeKey")
    objMyData.Key(strKeyName) = strNewKey
End If
...
```

'Existing key from list box
'New key value from text box
'Set Key property of this item

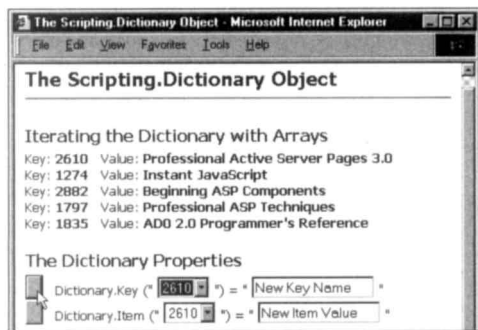


图5-5 使用Dictionary的Key属性

页面重新载入后，在Dictionary的内容列表里能看到相应的结果，如图5-6所示。

页面的其余代码用来设定一个条目的Item属性，或者执行Dictionary对象的方法。下面是这些操作的代码，每段代码与演示Key属性的代码非常类似。每次都把结果显示在Dictionary的内容列表中：

```
...
If Len(Request.Form("cmdChangeItem")) Then
    strKeyName = Request.Form("lstChangeItem")
    strNewValue = Request.Form("txtChangeItem")
    objMyData.Item(strKeyName) = strNewValue
End If

If Len(Request.Form("cmdAdd")) Then
    strKeyName = Request.Form("txtAddKey")
    strItemValue = Request.Form("txtAddItem")
    objMyData.Add strKeyName, strItemValue
End If

If Len(Request.Form("cmdRemove")) Then
    strKeyName = Request.Form("lstRemove")
    objMyData.Remove strKeyName
End If

If Len(Request.Form("cmdRemoveAll")) Then
    objMyData.RemoveAll
End If
...
```

'Existing key from list box
'New item value from text box
'Set the Item property

'New key value from text box
'New item value from text box
'Execute the Add method

'Existing key from list box
'Execute the Remove method

'Execute the RemoveAll method

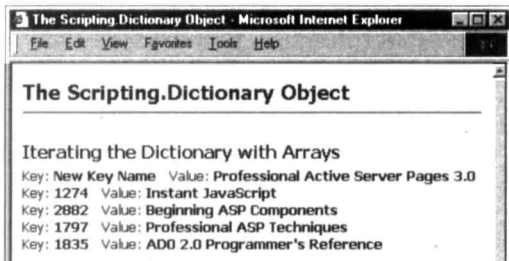


图5-6 页面重载后的结果

例如，如果现在点击Add方法的按钮，在Dictionary的内容列表里将增加一个新的条目，如图5-7所示。

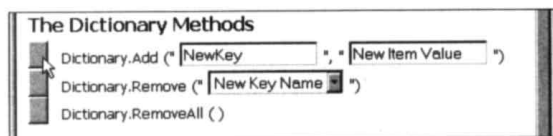


图5-7 增加一个新方法

结果如图5-8所示。

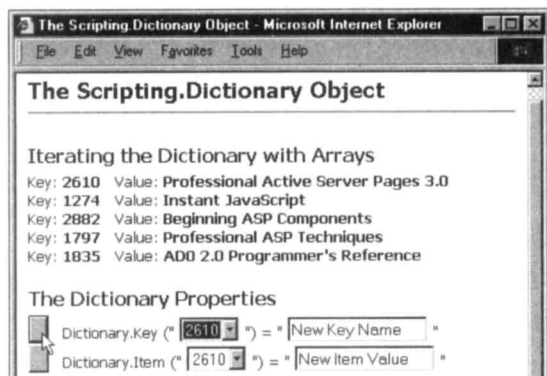


图5-8 Add方法的结果

可以在这个页面中试验 Dictionary对象的属性和方法，你将会发现什么因素及在什么环境下能引起Dictionary对象错误。例如，尝试用与已经存在的一个条目相同的键值增加一个条目，看看会出现什么结果。

5.4 Scripting.FileSystemObject对象

FileSystemObject对象提供对计算机文件系统的访问，它允许我们在代码内操作文本文件、文件夹及驱动器。它是脚本运行期库提供的对象之一，对于服务器 ASP页面内的VBScript和JScript都有效。如果页面的扩展名为.hta(表示它们是HTA的一部分)，它也可用在客户端的IE 5中。本节仅讨论在服务器上的ASP脚本如何使用FileSystemObject对象。

超级文本应用程序(HTA)由指定的“受信任的”页面组成，在页面的<HEAD>段里包含<HTA:APPLICATION>元素。例如：

```
<HTA:APPLICATION ID="objMyApp" APPLICATIONNAME="myApp">
```

这些页面可以使用客户端脚本引擎中的一些不常用特性，这些特性中有FileSystemObject对象和TextStream对象。关于超级文本应用程序的更多信息，请访问 Microsoft Workshop网站。

可以使用下面的程序创建一个FileSystemObject对象实例：

```
' In VBScript:
Dim objMyFSO
Set objMyFSO = Server.CreateObject("Scripting.FileSystemObject")

// In JScript:
var objMyFSO = Server.CreateObject('Scripting.FileSystemObject');
```

```
<!-- Server-side with an OBJECT element -->
<OBJECT RUNAT="SERVER" SCOPE="PAGE" ID="objFSO"
  PROGID="Scripting.FileSystemObject">
</OBJECT>
```

在ASP页面里，增加一个对于 FileSystemObject 类型库的引用是非常有用的。这允许使用它直接定义的内置常数，不用像过去那样必须用数字等效表达式来代替。整个脚本运行期库的类型库可以增加到任何 ASP 页面中，代码如下：

```
<!-- METADATA TYPE="typelib" FILE="C:\WinNT\System32\scrrun.dll" -->
```

如果你是在另一个目录下安装 Windows，必须编辑 FILE 的属性值。

5.4.1 FileSystemObject对象成员概要

FileSystemObject 对象提供一个属性和一系列方法，可用它们来操纵 FileSystemObject 对象实现的一些从属对象。这里提供了全部的内容概要，然后介绍每一个从属对象。

1. FileSystemObject 的属性

FileSystemObject 对象只有一个属性，它用于得到当前机器上的所有有效驱动器的列表，如表 5-4 所示。

表5-4 FileSystemObject对象的属性及说明

属 性	说 明
Drives	返回本地计算机可用的驱动器列表。包括从这台机器映射的网络驱动器

2. FileSystemObject 的方法

FileSystemObject 对象提供了使用从属对象的一系列方法，从属对象包括 Drive、Folder 和 File 等对象。它也实现了用于 TextStream 对象的两个方法：CreateTextFile 和 OpenTextFile。根据所使用的对象的类型，将方法划分为三类。

(1) 与驱动器有关的方法

与驱动器有关的方法如表 5-5 所示。

表5-5 与驱动器有关的方法及说明

方 法	说 明
DriveExists (<i>drivespec</i>)	如果在 <i>drivespec</i> 中指定的驱动器存在，则返回 True，否则返回 False。 <i>drivespec</i> 参数可以是一个驱动器字母，或者是文件、文件夹的完整绝对路径
GetDrive (<i>drivespec</i>)	返回 <i>drivespec</i> 指定的驱动器所对应的 Drive 对象。 <i>drivespec</i> 可以包含冒号、路径分隔符或者是网络共享名，即："c:"、"c:"、"c:\"及"\\machine\sharename"
GetDriveName (<i>drivespec</i>)	用字符串返回 <i>drivespec</i> 指定的驱动器的名称。 <i>drivespec</i> 参数必须是文件或文件夹的绝对路径，或者仅仅是驱动器字母，例如："c:"或"c"

(2) 与文件夹有关的方法

与文件夹有关的方法如表 5-6 所示。

表5-6 与文件夹有关的方法及说明

方 法	说 明
BuildPath(<i>path</i> , <i>name</i>)	在已有的路径 <i>path</i> 上增添名字为 <i>name</i> 的文件或文件夹，如果需要，则增添路径分隔符 "\

(续)

方 法	说 明
CopyFolder (source, destination, overwrite)	从指定的源文件夹 <i>source</i> (可以包含通配符)中复制一个或多个文件夹到指定的目标文件夹 <i>destination</i> , 包含了源文件夹中的所有文件。如果 <i>source</i> 包含通配符或 <i>destination</i> 末尾是路径分隔符 ('\\') , 那么认为 <i>destination</i> 是要放置源文件夹的拷贝的文件夹。否则的话 , 认为 <i>destination</i> 是要创建的新文件夹的路径名。如果 <i>destination</i> 文件夹已经存在且 <i>overwrite</i> 参数设置为 False , 将产生错误 , 缺省的 <i>overwrite</i> 参数是 True
CreateFolder (foldername)	创建一个路径名为 <i>foldername</i> 的文件夹。如果 <i>foldername</i> 已经存在将产生错误
DeleteFolder (folderspec, force)	删除由 <i>folderspec</i> 指定的一个或多个文件夹 (可以在路径的最后部分包含通配符)及文件夹中的所有内容。如果可选的 <i>force</i> 参数设置为 true , 那么即使文件夹包含的文件具有只读属性 , 也将删除该文件夹。缺省的 <i>force</i> 参数是 False
FolderExists (folderspec)	如果 <i>folderspec</i> 指定的文件夹存在则返回 True , 否则返回 False。 <i>Folderspec</i> 参数可以包含文件夹的绝对或相对路径 , 或者仅仅是当前文件夹中看到的文件夹名
GetAbsolutePathName (pathspec)	返回明确指定文件夹的路径 , 其中要考虑到当前文件夹的路径。例如 , 如果当前文件夹是 "c:\docs\sales\" , 而 <i>pathspec</i> 是 "jan" , 返回的字符是 "c:\docs\sales\jan"。通配符、 ".." 和 "\\ " 路径操作符都是可接受的
GetFolder (folderspec)	返回 <i>folderspec</i> 指定的文件夹对应的 Folder 对象。 <i>Folderspec</i> 可以是文件夹的相对的或绝对的路径
GetParentFolderName (pathspec)	返回 <i>pathspec</i> 文件或文件夹的上一级文件夹。不检验该文件夹是否存在
GetSpecialfolder (folderspec)	返回一个与特定的 Windows 文件夹相对应的 Folder 对象。参数 <i>folderspec</i> 的允许值是 WindowsFolder(0)、SystemFolder(1)和 TemporaryFolder (2)
MoveFolder(source, destination)	将 <i>source</i> 指定的一个或多个文件夹移动到 <i>destination</i> 指定的文件夹。在 <i>source</i> 里可以包含通配符 , 但在 <i>destination</i> 中不行。如果 <i>source</i> 包含通配符或 <i>destination</i> 末尾是路径分隔符 ('\\') , 则认为 <i>destination</i> 是要放置源文件夹的文件夹 , 否则认为它是一个新文件夹的完整路径和名字。如果目的文件夹 <i>destination</i> 已经存在则产生错误

(3) 与文件有关的方法。

与文件有关的方法如表 5-7 所示。

表5-7 与文件有关的方法及说明

方 法	说 明
CopyFile(source, destination, overwrite)	将 <i>source</i> (可以包含通配符)指定的一个或多个文件复制到指定的目标文件夹 <i>destination</i> 。如果 <i>source</i> 包含通配符或 <i>destination</i> 末尾是路径分隔符 ('\\') , 那么认为 <i>destination</i> 一文件夹。否则认为 <i>destination</i> 为一新文件的完全路径和名称。如果目标文件夹已经存在且 <i>overwrite</i> 参数设置为 False , 将产生错误。缺省的 <i>overwrite</i> 参数是 True

(续)

方 法	说 明
CreateTextFile(filename, overwrite, unicode)	用指定的文件名filename在磁盘上创建一个新的文本文件,并返回与其对应的 TextStream对象,如果可选的 overwrite参数设置为True,则覆盖同一路径下已有的同名文件。缺省的 overwrite参数是False。如果可选的 unicode参数设置为True,则该文件的内容将存储为Unicode文本,缺省的 unicode参数是False
DeleteFile(filespec, force)	删除由filespec指定的一个或多个文件(可以在路径的最后部分包含通配符)。如果可选的 force参数设置为true,那么也删除具有只读属性的文件。缺省的 force参数是False
FileExists (filespec)	如果filespec指定的文件存在则返回True,否则返回False。filespec参数可以包含文件的绝对路径或相对路径,或者是当前文件夹中的文件名
GetBaseName (filespec)	返回filespec指定的文件的名称,即包含文件路径但去掉了文件的扩展名
GetExtensionName (filespec)	返回filespec指定的文件的扩展名
GetFile (filespec)	返回filespec指定的文件所对应的 File对象。可以指定文件的相对或绝对路径
GetFileName (pathspec)	返回pathspec指定的文件的路径或文件名,如果没有文件名就返回最后的文件夹名。不检查该文件或文件夹是否存在
GetTempName ()	返回一个随机产生的文件名,用于完成运算所需的临时文件或文件夹
MoveFile (source, destination)	将source指定的一个或多个源文件移动到 destination指定的目的文件夹。在 source里可以包含通配符,但 destination不行。如果 source包含通配符或 destination末尾是路径分隔符('\'),那么认为 destination是一文件夹。否则,认为 destination是一新文件夹的完整路径和名称。如果目的文件夹已经存在则产生错误
OpenTextFile (filename, iomode, create, format)	创建一个名叫做filename的文件,或打开一个现有的名为filename的文件,并且返回一个与其相关的 TextStream对象。filename参数可以包含绝对或相对路径。iomode参数指定了所要求的访问类型。允许的数值是 ForReading(1)(缺省)、ForWriting(2)、ForAppending(8)。当写入或追加到一个不存在的文件时,如果 create参数设置为true,就将创建一个新文件。缺省的 create参数是False。format参数说明对文件读或写的数据格式。允许数值是:TristateFalse(0)(缺省),按照 ASCII格式打开;TristateTrue(-1),按照Unicode格式打开;TristateUseDefault(-2),用系统缺省格式打开

Unicode 文件使用两个字节标识每个字符,取消了 ASCII字符最多256个的限制。

5.4.2 使用驱动器

下面是使用 FileSystemObject对象的简单例子,它使用 DriveExists方法得到现有的驱动器字母的列表:

```
' In VBScript:
Set objMyFSO = Server.CreateObject("Scripting.FileSystemObject")

Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
For intCode = 65 To 90  'ANSI codes for 'A' to 'Z'
    strLetter = Chr(intCode)
    If objFSO.DriveExists(strLetter) Then
```

```
Response.Write "Found drive " & strLetter & "<br>"
End If
Next
```

或用JScript：

```
// In JScript:
var objFSO = Server.CreateObject('Scripting.FileSystemObject');
for (var intCode = 65; intCode <= 90; intCode++) {
    strLetter = String.fromCharCode(intCode);
    if (objFSO.DriveExists(strLetter))
        Response.Write('Found drive ' + strLetter + '<br>');
}
```

这两个程序段的运行结果是相同的，如图 5-9所示。这一页面为 driveexists_vb.asp，由本书的示例文件提供。

1. Drive对象

正如已经看到的， FileSystemObject对象包含一个属性——Drives，它返回一个包括本地计算机上所有可用驱动器的集合。

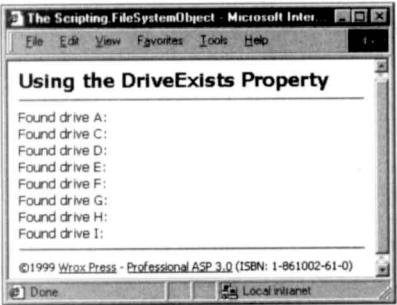


图5-9 驱动器列表

Drives集合里的每个条目是一个 Drive对象。 Drive对象的属性如表 5-8所示。

表5-8 Drive对象的属性及说明

属 性	说 明
AvailableSpace	考虑了帐户定额和/或其他限制，返回驱动器上对于该用户可用的空间的大小
DriveLetter	返回驱动器的字母
DriveType	返回驱动器的类型。返回值可以是Unknown(0)、Removable(1)、Fixed(2)、Netwok(3)、CDRom(4)和RamDisk(5)。然而需要注意的是当前版本的 srrun.dll不支持预定义常数Netwok，必须使用十进制数3来代替
FileSystem	返回驱动器文件系统的类型。返回值包括“ FAT”、“ NTFS ”和“ CDFS ”
FreeSpace	返回驱动器上可用剩余空间的总量
IsReady	返回一个布尔值表明驱动器是否已准备好
Path	返回一个由驱动器字母和冒号组成的驱动器路径，即“ C: ”
RootFolder	返回代表的驱动器根目录文件夹的 Folder对象
SerialNumber	返回一个用于识别磁盘卷的十进制的序列号
ShareName	如果是一个网络驱动器，返回该驱动器的网络共享名
TotalSize	返回驱动器的总容量(以字节为单位)
VolumeName	设定或返回本地驱动器卷名

因此，通过使用 Drives集合里的 Drive对象，可以在服务器上产生一个驱动器列表，与通过检查每个可能的驱动器字母来判别驱动器是否存在的方法相比，效率更高。我们也可以得到关于该驱动器的信息。在 VBScript里，代码如下：

```
' In VBScript:
' Create a FileSystemObject instance
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
' Create a Drives collection
Set colDrives = objFSO.Drives

' Iterate through the Drives collection
```

```

For Each objDrive in colDrives
    Response.Write "DriveLetter: " & objDrive.DriveLetter & "<BR>"
    Response.Write "DriveType: " & objDrive.DriveType & "<BR>"

    If objDrive.DriveType = 3 Then
        If objDrive.IsReady Then
            Response.Write "Remote drive with ShareName: " & _
                objDrive.ShareName & "<BR>"
        Else
            Response.Write "Remote drive - IsReady property returned False<BR>"
        End If
    Else
        Response.Write "FileSystem: " & objDrive.FileSystem & "<BR>"
        Response.Write "SerialNumber: " & objDrive.SerialNumber & "<BR>"
        Response.Write "Local drive with VolumeName: " & _
            objDrive.VolumeName & "<BR>"
        Response.Write "AvailableSpace: " & objDrive.AvailableSpace & " bytes<BR>"
        Response.Write "FreeSpace: " & objDrive.FreeSpace & " bytes<BR>"
        Response.Write "TotalSize: " & objDrive.TotalSize & " bytes<P>"
    End If
Next

```

注意，不能用预定义常数 Network 比较驱动器的 DriveType 属性，因为（至少在 scrrun.dll 的当前版本中）在类型库中省略了 Network 常数，因此不再作为公用的常数使用。

在 JScript 中，该程序是：

```

// In JScript:
// Create a FileSystemObject instance
var objFSO = Server.CreateObject('Scripting.FileSystemObject');
// Create a Drives collection
var colDrives = new Enumerator(objFSO.Drives);
// Iterate through the Drives collection
for (; !colDrives.atEnd(); colDrives.moveNext()) {
    objDrive = colDrives.item();
    Response.Write('DriveLetter: ' + objDrive.DriveLetter + '<BR>');
    Response.Write('DriveType: ' + objDrive.DriveType + '<BR>');
    if (objDrive.DriveType == 3)
        if (objDrive.IsReady)
            Response.Write('Remote drive with ShareName: ' +
                objDrive.ShareName + '<BR>')
        else
            Response.Write('Remote drive - IsReady property returned False<BR>');
    else
        Response.Write('Local drive with VolumeName: ' +
            objDrive.VolumeName + '<BR>');
    Response.Write('FileSystem: ' + objDrive.FileSystem + '<BR>');
    Response.Write('SerialNumber: ' + objDrive.SerialNumber + '<BR>');
    Response.Write('AvailableSpace: ' + objDrive.AvailableSpace + ' bytes<BR>');
    Response.Write('FreeSpace: ' + objDrive.FreeSpace + ' bytes<BR>');
    Response.Write('TotalSize: ' + objDrive.TotalSize + ' bytes<P>');
}

```

在系统上运行这段程序以前有一点要注意。如果在 A 驱动器里没有磁盘，或 CD-ROM 驱动器里没有光盘，将得到一个错误提示：“Disk Not Ready”。除了 DriveLetter 属性和 DriveType 属性外，在使用其他属性和方法前，通过检查每个驱动器的 IsReady 属性，可以保护该页面。

当在服务器上运行以上 VBScript代码时，运行结果如图 5-10所示。这一页面为 drivescollection_vb.asp，来自本书提供的示例文件。

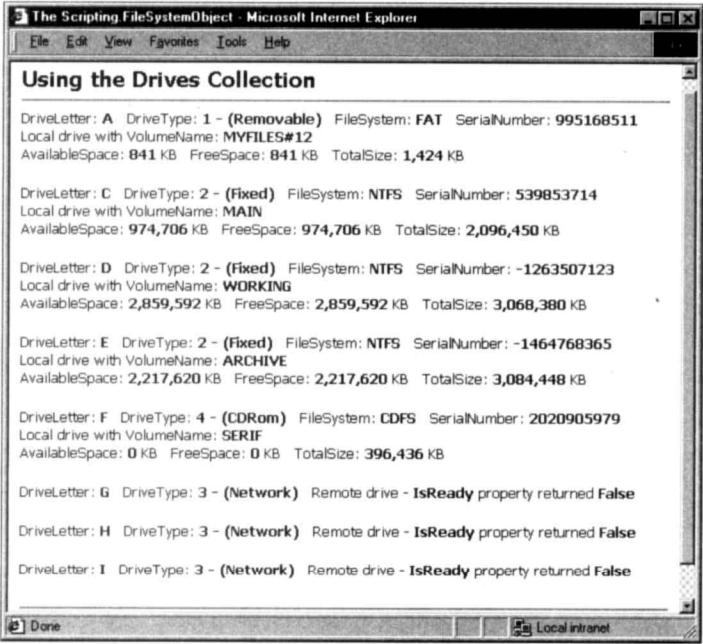


图5-10 驱动器详细列表

2. 文件系统定位

FileSystemObject的几个方法可用于得到其他对象的引用，因此可以在服务器的文件系统 和任何网络驱动器中定位。事实上，在 ASP代码里使用的所有对象或组件中，除了 ActiveX

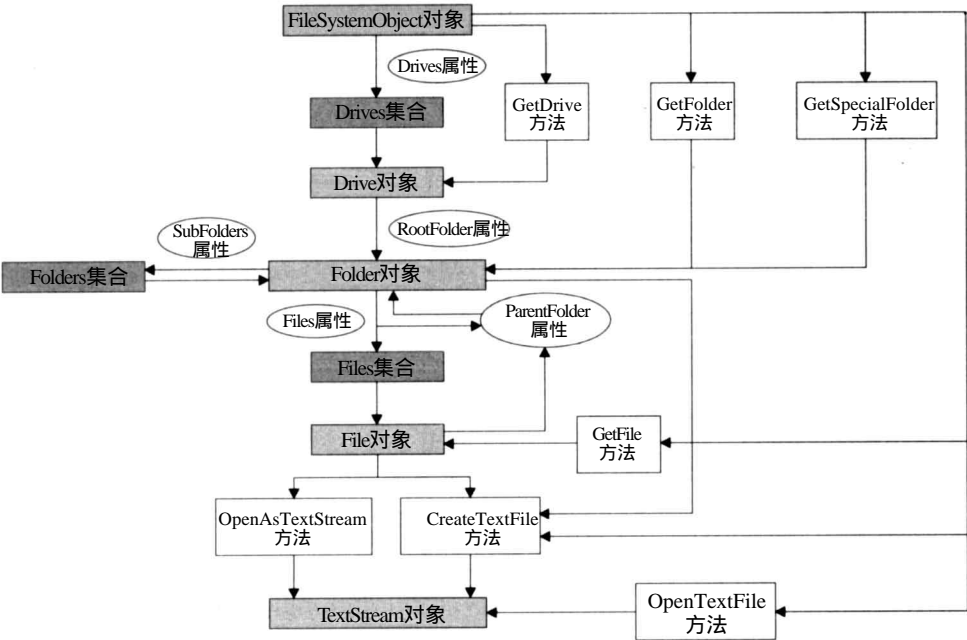


图5-11 文件系统定位关系

Data Objects组件，FileSystemObject对象很可能是最复杂的对象之一。

这种复杂性是由于对如何访问文件系统的不同部分，要求有极高的灵活性。例如，可以从FileSystemObject向下通过使用各种从属对象定位一个文件。其过程是从 Drives集合开始，到一个Drive对象，再到驱动器的根Folder对象，然后到子Folder对象，再到文件夹的Files集合，最后到集合内的File对象。

另外，如果已知要访问的驱动器、文件夹或文件。可以直接对其使用 GetDrive、GetFolder、GetSpecialFolder和GetFile方法。图 5-11有助于理解所有与文件系统定位相关的组件、对象、方法和属性之间的关系。

3. Folder对象

Drive对象的RootFolder属性返回一个Folder对象，通过该对象可访问这个驱动器内的所有的内容。可以使用这个Folder对象的属性和方法遍历驱动器上的目录，并得到该文件夹和其他文件夹的属性。

(1) Folder对象的属性

Folder对象提供一组属性，可用这些属性得到关于当前文件夹的更多信息，也可以改变该文件夹的名称。其属性及说明如表 5-9所示。

表5-9 Folder对象的属性及说明

属 性	说 明
Attributes	返回文件夹的属性。可以是下列值中的一个或其组合 :Normal(0)、ReadOnly(1)、Hidden(2)、System(4)、Volume (名称)(8)、Directory(文件夹)(16)、Archive(32)、Alias(64)和Compressed(128)。例如，一个隐藏的只读文件，Attributes的值为3
DateCreated	返回该文件夹的创建日期和时间
DateLastAccessed	返回最后一次访问该文件夹的日期和时间
DateLastModified	返回最后一次修改该文件夹的日期和时间
Drive	返回该文件夹所在的驱动器的驱动器字母
Files	返回Folder对象包含的Files集合，表示该文件夹内所有的文件
IsRootFolder	返回一个布尔值说明该文件夹是否是当前驱动器的根文件夹
Name	设定或返回文件夹的名字
ParentFolder	返回该文件夹的父文件夹对应的 Folder对象
Path	返回文件夹的绝对路径，使用相应的长文件名
ShortName	返回DOS风格的8.3形式的的文件夹名
ShortPath	返回DOS风格的8.3形式的文件夹的绝对路径
Size	返回包含在该文件夹里所有文件和子文件夹的大小
SubFolders	返回该文件夹内包含的所有子文件夹对应的 Folders集合，包括隐藏文件夹和系统文件夹
Type	如果可能，返回一个文件夹类型的说明字符串（例如，“Recycle Bin”）

(2) Folder对象的方法

Folder对象提供一组可用于复制、删除和移动当前文件夹的方法。这些方法的运行方式与FileSystemObject对象的CopyFolder、 DeleteFolder和MoveFolder方法相同，但这些方法不要 求source参数，因为源文件就是这个文件夹。这些方法及说明如表 5-10所示。

在文件夹之间可以使用当前文件夹的 ParentFolder属性，返回到父目录。当到达一个文件夹时，如果IsRootFolder属性是True，就停下来。离开驱动器的根目录，沿目录树向下，可遍历或访问在Folders集合(由当前文件夹的SubFolders属性返回)内的指定文件夹。


```
Response.Write('ShortName: ' + objFolder.ShortName + '&nbsp; &nbsp;');  
Response.Write('Type: ' + objFolder.Type + '<BR>');  
Response.Write('Path: ' + objFolder.Path + '&nbsp; &nbsp;');  
Response.Write('ShortPath: ' + objFolder.ShortPath + '<BR>');  
Response.Write('Created: ' + objFolder.DateCreated + '&nbsp; &nbsp;');  
Response.Write('LastModified: ' + objFolder.DateLastModified + '<P>');  
}
```

该VBScript程序在服务器上运行时的结果如图 5-12所示。该页面为 folderscollection_vb.asp, 来自本书提供的示例文件。

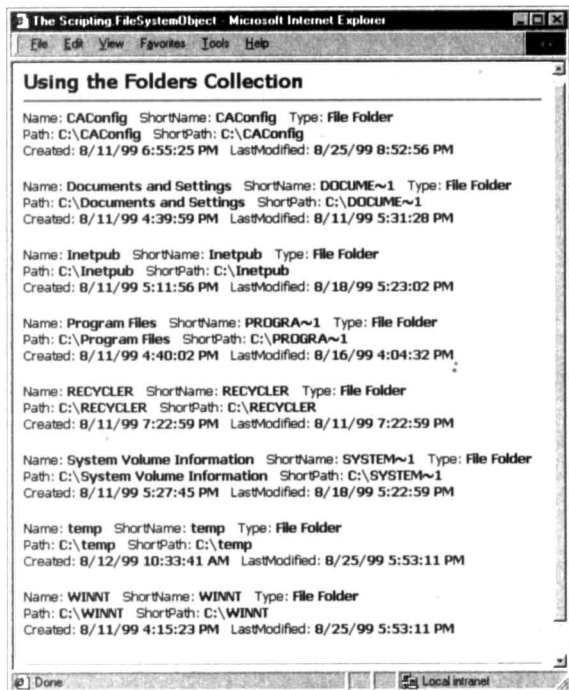


图5-12 Folders集合的内容

(3) 使用特殊文件夹

GetSpecialFolder是FileSystemObject对象的方法之一, 它返回计算机上三个“特殊文件夹”对应的Folder对象:

- WindowsFolder: %Windows% 目录, 缺省为 WinNT (或 Windows, 在非 NT/2000 的计算机上。) 目录
- SystemFolder: %System% 目录, 缺省为 WinNT\System32 (或 Windows\System, 在非 NT/2000 计算机上) 目录。
- TemporaryFolder: %Temp% 目录, 缺省为 WinNT\Temp (或 Windows\Temp, 在非 NT/2000 计算机上) 目录。

为得到对特殊文件夹的引用, 我们提供相应的预定义常数作为 GetSpecialFolder 方法的参数:

```
' In VBScript:  
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
```

```
Set objFolder = objFSO.GetSpecialFolder(WindowsFolder)
Response.Write "GetSpecialFolder(WindowsFolder) returned:<BR>"
Response.Write "Path: " & objFolder.Path & "<BR>"
Response.Write "Type: " & objFolder.Type & "<P>"

Set objFolder = objFSO.GetSpecialFolder(SystemFolder)
Response.Write "GetSpecialFolder(SystemFolder) returned:<BR>"
Response.Write "Path: " & objFolder.Path & "<BR>"
Response.Write "Type: " & objFolder.Type & "<P>"

Set objFolder = objFSO.GetSpecialFolder(TemporaryFolder)
Response.Write "GetSpecialFolder(TemporaryFolder) returned:<BR>"
Response.Write "Path: " & objFolder.Path & "<BR>"
Response.Write "Type: " & objFolder.Type & "<P>"
```

或用JScript：

```
// In JScript:
var objFSO = Server.CreateObject('Scripting.FileSystemObject');

var objFolder = objFSO.GetSpecialFolder(WindowsFolder);
Response.Write('GetSpecialFolder(WindowsFolder) returned - &nbsp;&nbsp;&nbsp;');
Response.Write('Path: ' + objFolder.Path + '&nbsp;&nbsp;&nbsp; &nbsp;&nbsp;&nbsp;');
Response.Write('Type: ' + objFolder.Type + '<BR>');

var objFolder = objFSO.GetSpecialFolder(SystemFolder);
Response.Write('GetSpecialFolder(SystemFolder) returned - &nbsp;&nbsp;&nbsp;');
Response.Write('Path: ' + objFolder.Path + '&nbsp;&nbsp;&nbsp; &nbsp;&nbsp;&nbsp;');
Response.Write('Type: ' + objFolder.Type + '<BR>');

var objFolder = objFSO.GetSpecialFolder(TemporaryFolder);
Response.Write('GetSpecialFolder(TemporaryFolder) returned - &nbsp;&nbsp;&nbsp;');
Response.Write('Path: ' + objFolder.Path + '&nbsp;&nbsp;&nbsp; &nbsp;&nbsp;&nbsp;');
Response.Write('Type: ' + objFolder.Type + '<BR>');
```

该VBScript程序在服务器上运行时的结果如图 5-13所示。该页面名为 specialfolder_vb.asp，来自本书提供的示例文件。

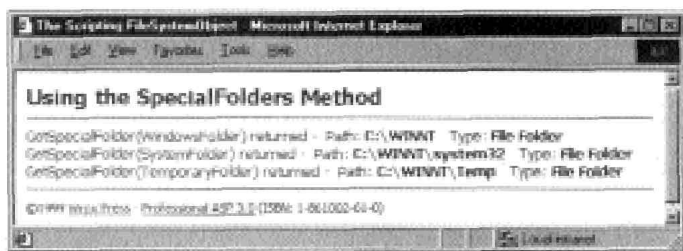


图5-13 GetSpecialFolder方法的使用结果

4. File对象

File对象提供了对文件的属性的访问，通过它的方法能够对文件进行操作。每个 Folder对象提供了一个Files集合，包含文件夹中文件对应的File对象。还可以直接地从 FileSystemObject对象中通过使用 GetFile方法得到一个File对象引用。

(1) File对象的属性

File对象有一系列的属性，类似于 Folder对象的属性，如表 5-11所示。

(2) File对象的方法

表5-11 File对象的属性及说明

属 性	说 明
Attributes	返回文件的属性。可以是下列值中的一个或其组合 :Normal(0)、ReadOnly(1)、Hidden(2)、System(4)、volume(名称)(8)、Directory (文件夹) (16)、Archive(32)、Alias(64)和 compressed (128)
DateCreated	返回该文件的创建日期和时间
DateLastAccessed	返回最后一次访问该文件的日期和时间
DateLastModified	返回最后一次修改该文件的日期和时间
Drive	返回该文件所在的驱动器的 Drive对象
Name	设定或返回文件的名字
ParentFolder	返回该文件的父文件夹的 Folder对象。
Path	返回文件的绝对路径，可使用长文件名。
ShortName	返回DOS风格的8.3形式的文件名
ShortPath	返回DOS风格的8.3形式的文件绝对路径
Size	返回该文件的大小(字节)
Type	如果可能，返回一个文件类型的说明字符串 (例如：“Text Document ”表示.txt文件)

同样类似于 Folder对象，File对象的方法允许复制、删除以及移动文件。它也有一个使用文本流打开文件的方法。File对象的方法及说明如表 5-12所示。

表5-12 File对象的方法及说明

方 法	说 明
Copy(destination, overwrite)	将这个文件复制到 destination指定的文件夹。如果 destination的末尾是路径分隔符(“\”), 那么认为 destination是放置拷贝文件的文件夹。否则认为 destination是要创建的新文件的路径和名字。如果目标文件夹已经存在且 overwrite参数设置为 False，将产生错误，缺省的 overwrite参数是 True
Delete (force)	删除这个文件。如果可选择的 force参数设置为 True，文件即使具有只读属性也会被删除。缺省的 force是 False
Move ()	将文件移动到 destination指定的文件夹。如果 destination的末尾是路径分隔符(“\”), 那么认为 destination是一文件夹。否则认为 destination是一个新的文件的路径和名字。如果目标文件夹已经存在，则出错
CreateTextFile (filename, overwrite, unicode)	用指定的文件名创建一个新的文本文件，并且返回一个相应的 TextStream 对象。如果可选的 overwrite参数设置为 True，将覆盖已有的同名文件。缺省的 overwrite参数是 False.。如果可选的 unicode参数设置为 True，文件的内容将存储为 unicode文本。缺省的 unicode是 False
OpenAsTextStream (iomode, format)	打开指定文件并且返回一个 TextStream对象，用于文件的读、写或追加。 iomode 参数指定了要求的访问类型，允许值是 ForReading(1)(缺省值)、ForWriting(2)、ForAppending(8)。format 参数说明了读、写文件的数据格式。允许值是 TristateFalse(0)(缺省)，说明用 ASCII数据格式;TristateTrue(-1)说明用 Unicode数据格式;TristateUseDefault (-2)说明使用系统缺省的格式

因此给定一个File对象后，可以使用ParentFolder属性得到包含该文件的Folder对象的引用，用来在文件系统中导航。甚至可以用 Drive属性获得相应的 Drive对象的引用，并得到各种 Folder对象以及所包含的 File对象。

另外，给定一个Folder对象以及对应的Files集合后，可以通过遍历该集合检查这一文件夹中的每个文件。还可以使用File对象的各种方法以一定方式处理该文件，如复制、移动或删除。下面的代码给出了C驱动器的第一个文件夹的文件列表：

两个程序的结果是相同的，如图 5-14所示。该页面为 filescollection_vb.asp，来自本书提供的示例文件。

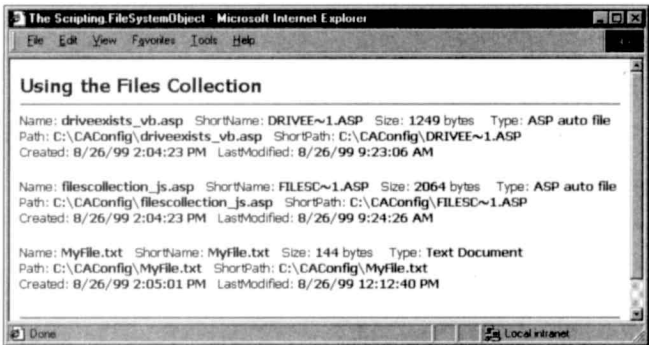


图5-14 File集合的内容

在C驱动器上的第一个文件夹可能是 CAConfig，缺省情况下该文件夹是空的。这种情况下，可先向该文件夹复制一些文件，完成实验以后再删除这些文件。

5.5 Scripting.TextStream对象

FileSystemObject、Folder和File对象的一些方法都与通过 TextStream对象创建、读取或写入文本文件有关。

虽然TextStream对象定义为FileSystemObject对象的一个独立的附属对象，但我们不得使用FileSystemObject对象或其附属对象来创建一个 TextStream对象并访问磁盘文件的内容。

5.5.1 创建TextStream对象的方法

有三个常用方法用于创建或打开一个文本文件并返回 TextStram对象，如表 5-13所示。

表5-13 创建TextStream对象的方法及说明

方 法	说 明
CreateTextFile (filename, overwrite, unicode)	在磁盘上用指定的文件名filename创建一个新文本文件，并返回一个与该文件对应的TextStream对象。如果可选的 overwrite参数设置为 True，将覆盖具有同样路径的同名文件。缺省的 overwrite是False。如果可选的unicode参数设置为 False，该文件的内容将存储为 unicode格式。缺省的 unicode是False
OpenTextFile (filename, iomode, create, format)	打开或创建(如果不存在)一个名为filename的文件，并且返回与该文件对应的TextStream对象。filename参数可以包含绝对或相对路径。 iomode参数说明需要的访问类型。容许值是 ForReading(1)(缺省)、ForWriting(2)、ForAppending(8)。写入或追加到一个不存在的文件时，如果 create参数设置为 True，将创建一个新文件。缺省的 create是False。 format参数说明了读或写文件时的数据格式。容许值是 TristateFalse(0)(缺省)，说明用 ASCII数据格式;TristateTrue (-1)说明用 Unicode数据格式;TristateUseDefault (-2)说明使用系统缺省的格式
OpenAsTextStream (iomode, format)	打开一个指定的文件并且返回一个TextStream对象，可用于对该文件的读、写或追加。 iomode参数说明了需要的访问类型。容许值是 ForReading(1)(缺省)、ForWriting(2)、ForAppending(8)。 format参数说明了读写文件的数据格式。容许值是 TristateFalse(0)(缺省)，说明用 ASCII数据格式;TristateTrue (-1)说明用 Unicode数据格式;TristateUseDefault (-2)说明使用系统缺省的格式

上面列出的方法在 FileSystemObject、Folder和File对象中的实现有所不同。如表 5-14所示。

表5-14 三个对象中包含的方法

方 法	FileSystemObject对象	Folder对象	File对象
CreateTextFile	有	有	有
OpenTextFile	有	无	无
OpenAsTextStream	无	无	有

因此，可以使用这些方法创建一个新的文本文件，或者打开一个已存在的文件。则可得与该文件相应的一个 TextStream对象，可以使用 TextStream对象的属性和方法操作文件。

1. 创建新的文本文件

可以用 CreateTextFile方法创建新的文本文件，或覆盖一个已存在的文件。返回的 TextStream对象可用来读写文件。

首先创建一个 FileSystemObject对象，用来创建 TextStream对象。下面这个例子是用 VBScript创建一个“普通的”（即非Unicode）名为MyFile.txt的文件，并覆盖已存在的同名文件：

```
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
Set objTStream = objFSO.CreateTextFile("C:\TextFiles\MyFile.txt", True, False)
```

这同样可用JScript实现：

```
var objFSO = Server.CreateObject('Scripting.FileSystemObject');
var objTStream = objFSO.CreateTextFile('C:\TextFiles\MyFile.txt', true, false);
```

一旦创建了该文件，就可以使用 objTStream(它是对一个TextStream对象的引用)对文件进行操作。

2. 打开已存在的文本文件

OpenTextFile方法用于打开一个已有的文本文件。它返回一个 TextStream对象，可用这个对象对文件读或追加数据。

同样，首先创建一个 FileSystemObjet对象，然后用其创建一个 TextStream对象。下面的 VBScript程序例子打开一个名为MyFile.txt的文件，准备读出其内容：

```
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
Set objTStream = objFSO.OpenTextFile("C:\TextFiles\MyFile.txt", ForReading)
```

用JScript：

```
var objFSO = Server.CreateObject('Scripting.FileSystemObject');
var objTStream = objFSO.OpenTextFile('C:\TextFiles\MyFile.txt');
```

为了写入一个文件或创建一个不存在的文件，可以用以下代码：

```
! In VBScript:
Set objTStream = objFSO.OpenTextFile("C:\TextFiles\MyFile.txt", ForWriting, True)

// In JScript:
var objTStream = objFSO.OpenTextFile('C:\TextFiles\MyFile.txt', ForWriting, true);
```

如果要打开一个已有的 Unicode文件，准备对其追加数据，但是不创建不存在的文件，可以用：

```
' In VBScript:
Set objTStream = objFSO.OpenTextFile("C:\TextFiles\MyFile.txt", ForAppending, _
    False, TristateTrue)

// In JScript:
var objTStream = objFSO.OpenTextFile('C:\TextFiles\MyFile.txt', ForAppending,
    false, true);
```

3. 作为一个TextStream对象打开一个File对象

可用File对象的OpenAsTextStream方法打开与该对象相应的文件，并且返回一个能对该文件进行读、写和追加的 TextStream对象。所以，给定一个 File对象（这种情况下不是 FileSystemObject对象）——objFileObject，可作为一个“普通的”（非Unicode）TextStream对象打开它，以供追加文件内容：

```
' In VBScript:
Set objTStream = objFileObject.OpenAsTextStream(ForAppending, False)

// In JScript:
var objTStream = objFileObject.OpenTextFile(ForAppending, false);
```

注意，使用这种方法不需要文件名，因为程序的执行通过引用 File对象进行，并且没有 create参数，因为该文件必须已存在。如果想从一个新的空的文件开始，可以用：

```
' In VBScript:
Set objTStream = objFileObject.OpenAsTextStream(ForWriting)

// In JScript:
var objTStream = objFileObject.OpenTextFile(ForWriting);
```

如果想读取该文件：

```
' In VBScript:
Set objTStream = objFileObject.OpenAsTextStream(ForReading)

// In JScript:
var objTStream = objFileObject.OpenTextFile(ForReading);
```

5.5.2 TextStream对象成员概要

表5-15和表5-16是TextStream对象的全部属性和方法的列表。下面将简短介绍各个重要的成员的细节。

1. TextStream对象的属性

TextStream的属性提供有关文件内文件指针当前位置的信息，如表5-15所示。注意，所有的属性是只读的。

表5-15 TextStream对象的属性及说明

属 性	说 明
AtEndOfLine	如果文件位置指针在文件中一行的末尾则返回 True
AtEndOfStream	如果文件位置指针在文件的末尾则返回 True
Column	从1开始返回文件中当前字符的列号
Line	从1开始返回文件中当前行的行号

AtEndOfLine和AtEndOfStream属性仅对以iomode参数为ForReading的方式打开的文件可用，否则将会出错。

2. TextStream对象的方法

TextStream对象的方法如表5-16所示。

表5-16 TextStream对象的方法及说明

属 性	说 明
Close ()	关闭一个打开的文件
Read(numchars)	从文件中读出 numchars个字符
ReadAll ()	作为单个字符串读出整个文件
ReadLine ()	作为一个字符串从文件中读出一行(直到回车符和换行)
Skip (numchars)	当从文件读出时忽略 numchars个字符
SkipLine ()	当从文件读出时忽略下一行
Write (string)	向文件写入字符串 string
WriteLine (string)	向文件写入字符串 string (可选)和换行符
WriteBlankLines (n)	向文件写入 n个换行符

3. 写文本文件

一旦使用 CreateTextFile、 OpenTextFile或 OpenAsTextStream方法以及 ForWriting或 ForAppending参数，创建一个对应于某个文件的 TextStream对象，可以用下面的 VBScript程序写文件和关闭文件：

```
' In VBScript:
objTStream.WriteLine "At last I can create files with VBScript!"
objTStream.WriteLine
objTStream.WriteLine "Here are three blank lines:"
objTStream.WriteBlankLines 3
objTStream.Write "... and this is "
objTStream.WriteLine "the last line."
objTStream.Close
```

或者用JScript:

```
// In JScript:
objTStream.WriteLine('At last I can create files with JScript!');
objTStream.WriteLine();
objTStream.WriteLine('Here are three blank lines:');
objTStream.WriteBlankLines(3);
objTStream.Write('... and this is ');
objTStream.WriteLine('the last line.');
```

4. 读文本文件

一旦使用 CreateTextFile、 OpenTextFile或 OpenAsTextStream方法以及 ForReading参数，创建一个对应于某个文件的 TextStream对象，可以用下面的 VBScript程序读文件和关闭文件：

```
' In VBScript:
' Read one line at a time until the end of the file is reached
Do While Not objTStream.AtEndOfStream
    'Get the line number
    intLineNum = objTStream.Line
    'Format it as a 4-character string with leading zeros
    strLineNum = Right("000" & CStr(intLineNum), 4)
```

```
'Get the text of the line from the file
strLineText = objTStream.ReadLine
Response.Write strLineNum & ": " & strLineText & "<BR>"
Loop
objTStream.Close

或用 JScript :

// In JScript:
// Read one line at a time until the end of the file is reached
while (! objTStream.AtEndOfStream) {
    // Get the line number
    intLineNum = objTStream.Line;
    // Format and convert to a string
    strLineNum = '000' + intLineNum.toString();
    strLineNum = substr(strLineNum, strLineNum.length - 4, 4)
    // Get the text of the line from the file
    strLineText = objTStream.ReadLine();
    Response.Write(strLineNum + ': ' + strLineText + '<BR>');
}
objTStream.Close();
```

5.5.3 TextStream对象举例

为了了解使用TextStream对象操作文本文件的几种方式，本书提供了一个 VBScript示例页面，该页使用了大量的上述的代码。从示例的 Chapter05主菜单页，选择链接“Working With the TextStream Object”打开show_textstream.asp页面。

该页显示了存储在磁盘上名为 MyFile.txt 的文件的文本内容。在<TEXTAREA>控件里显示的文本内容允许进行编辑，并且下面有三个按钮。三个按钮的作用分别是用<TEXTAREA>控件的内容更新(即取代)最初的文本，在已有文件内容的后面添加文本，或用初始的缺省内容重写文件，如图5-15所示。

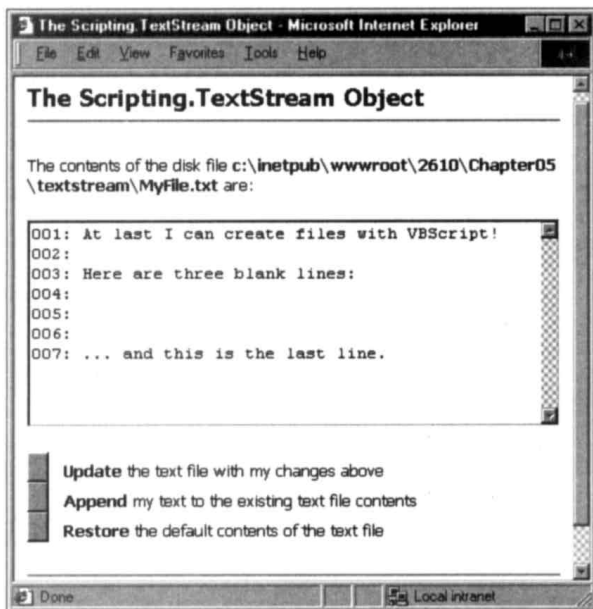


图5-15 TextStream对象示例页面

1. 读取已存在的文本文件的内容

每次载入该页面，将打开文本文件并将读取的内容置入 <TEXTAREA>控件。注意我们如何使用 Server.MapPath 方法得到文件 MyFile.txt 的绝对物理路径，该文件与示例页面在同一目录下。下面的程序创建 FileSystemObject 实例：

```
<%
strTextFile = Server.MapPath('MyFile.txt') 'Get absolute the path of MyFile.txt

'Create an instance of a FileSystemObject object
Set objFSO = Server.CreateObject('Scripting.FileSystemObject')
...

```

与书中本部分的大多数其他示例一样，该页包含一个 <FORM>段以保存该页面的 HTML 控件。ACTION 是当前页面，因此窗体的内容送回到同一页面。

每次载入该页面，<TEXTAREA>控件用文本文件的当前内容填充：

```
...
<FORM ACTION="<% = Request.ServerVariables('SCRIPT_NAME') %>" METHOD="POST">

The contents of the disk file <B><% = strTextFile %></B> are:<P>
<TEXTAREA NAME="txtContent" ROWS="10" COLS="50" WRAP="PHYSICAL">
<%
'open the text file as a TextStream object
Set objTStream = objFSO.OpenTextFile(strTextFile, ForReading)
'read one line at a time until the end of the file is reached
Do While Not objTStream.AtEndOfStream
  'get the line number
  intLineNum = objTStream.Line
  'format and convert to a string
  strLineNum = Right("00" & CStr(intLineNum), 3)
  'get the text of the line from the file
  strLineText = objTStream.ReadLine
  Response.Write strLineNum & ": " & strLineText & vbCrLf
Loop
objTStream.Close
%>
</TEXTAREA><P>

```

由上面程序可知道如何打开文本文件进行读取，遍历整个文件每次读取一行（而不是作为一个字符串读出整个文件）。这是因为要添加自己的行号，行号不属于该文件的文本。对从该文件读出的每行（读之前），检索并且格式化 Line 属性并创建一个三位数字的行号。然后把行编号和文本行放置页面的 <TEXTAREA>控件内。

2. 更新文本文件的内容

当点击页面的 Update 按钮时（一般是在编辑了 <TEXTAREA>控件里的文本以后），将把 <TEXTAREA>控件里的内容重新写入到该文本文件内。为此，该页有相应的一些 ASP 代码，在创建 HTML 控件以前检验 Request.Form 集合，查看点击的是那一个按钮（如果有的话），然后就重新载入该页。

如果点击 Update 按钮，搜集 <TEXTAREA>控件的内容作为一个字符串，分离这个字符串使之成为独立文本行的数组，并且打开文本文件准备重写其内容，然后遍历刚刚创建的数组，按行号循环写入该行的内容：

```
...
'Look for a command sent from the FORM section buttons

```

```

If Len(Request.Form("cmdUpdate")) Then
    'Get contents of TEXTAREA control
    strNewText = Request.Form("txtContent")
    'Split it into an array of lines at each carriage return
    arrLines = Split(strNewText, vbCrLf)
    'Open the text file for writing, which replaces all existing content
    Set objTStream = objFSO.OpenTextFile(strTextFile, ForWriting)
    For intLine = 0 To UBound(arrLines)
        strThisLine = arrLines(intLine)
        'Write out each line in turn as long as it's got a line number
        If Len(strThisLine) > 4 Then objTStream.WriteLine Mid(strThisLine, 6)
    Next
    objTStream.Close
End If
...

```

HTML<TEXTAREA>控件可在返回的 Value 中增加额外字符,这依赖于原始 HTML 页内的内容格式和 WRAP 属性的设置。特别是应在 ASP 脚本结束定界符“%>”后立即写上</TEXTAREA>标记,以防止增加一个额外的回车符号。即使用:

```
%></TEXTAREA><P>
```

而不使用:

```
%>
</TEXTAREA><P>
```

3. 向文本文件追加内容

当点击 Append 按钮时,可对已有的文件追加内容,与修改该文件内容类似,如图 5-16 所示。区别是打开该文件是为了追加而不是为了改写文件。调用 OpenTextFile 方法时可增加额外参数,防止在指定的文件不存在时创建新文件。

```

...
If Len(Request.Form("cmdAppend")) Then
    strNewText = Request.Form("txtContent")
    arrLines = Split(strNewText, vbCrLf)
    Set objTStream = objFSO.OpenTextFile(strTextFile, ForAppending, False)
    For intLine = 0 To UBound(arrLines)
        strThisLine = arrLines(intLine)
        If Len(strThisLine) > 4 Then objTStream.WriteLine Mid(strThisLine, 6)
    Next
    objTStream.Close
End If...
...

```



图5-16 向文本文件追加内容时的示例页面

4. 重写缺省内容

最后, Restore按钮用来将初始缺省内容简单地重写回文本文件。代码与用 TextStream的方法写一个文本文件类似:

```
...
If Len(Request.Form("cmdDefault")) Then
    'write out default contents to file
    Set objTStream = objFSO.CreateTextFile(strTextFile, True, False)
    objTStream.WriteLine "At last I can create files with VBScript!"
    objTStream.WriteLine
    objTStream.WriteLine "Here are three blank lines:"
    objTStream.WriteLineBlankLines 3
    objTStream.Write "... and this is "
    objTStream.WriteLine "the last line."
    objTStream.Close
End If
```

5.6 小结

本章讲述了在ASP页面中使用对象和组件的强大能力。首先讨论对象和组件的一般特性,以及它们的类型。然后集中论述了如何在ASP(及客户端)脚本代码序内创建对象实例。

在页面上使用的许多对象可能都是“外部”组件,这些组件安装在服务器上,独立于ASP。本章所讨论的对象,当ASP使用一种缺省的脚本语言(如VBScript或JScript)时,总是可用的。其实现是通过scrrun.dll文件里的脚本运行期库完成的。

这些对象是指Dictionary对象、FileSystemObject对象和TextStream对象。

Dictionary对象为我们提供了存储值的一种有效方式,可根据名字进行索引和访问,而不是根据一个数字索引进行访问。这是存储名字/值对这样的数据的理想方式。

FileSystemObject对象和TextStream对象相互之间联系密切,可使用它们访问服务器或网络(映射)的磁盘驱动器的目录。FileSystemObject对象提供对驱动器、文件夹(目录)和文件的存取,并提供了用于对于获得更多的信息或移动、复制、删除它们的属性及方法。

可以创建对应于系统内的任何文件的TextStream对象,并通过该对象对文件进行读取和写入。对于读写过程它作为文本文件操作,甚至可以处理Unicode格式的文件。对文件系统的导航和读写能力的结合允许对服务器文件系统进行极其复杂的控制。还可以在客户端脚本代码中使用对象(有一定限制)。

本章讨论的主要内容为:

- 脚本引擎以什么方式提供脚本对象。
- 如何创建脚本对象及其他组件的实例。
- 脚本对象的成员和特性概要。
- 如何在代码中使用脚本对象。

下一章将把注意力从脚本运行期库对象转移到可以在ASP页面中使用的其他组件,包括由IIS/ASP提供的可安装组件和在其他任何地方得到的组件。