

## 第3章 ASP应用程序与会话

在前面的章节中介绍了 ASP提供的访问一个客户请求和产生响应的方法，本章将讨论 ASP 的另两个对象。就是 Application和Session对象。这两个对象不是直接地与请求和响应的管理有关，而是更多地与 ASP网页运行环境的管理相关。

与建立 Web站点或 Web应用程序有关的共同问题之一，是使用 HTTP协议时没有状态。状态提供了与一个指定用户有关的变量值、对象和其他资源，并且应用程序中的任意例程都能使用它；以一种像 VB或C++这样的程序设计语言编写一般的基于客户的应用程序时，使用状态可以完成一些相应的工作。然而，Web并不提供这种能力。在本章中，读者将看到为什么和如何避免这个问题。

本章还涉及到一些术语和技术问题。它迄今为止，本书中已经简单地讨论了“Web应用程序”，但没有真正确切地理解或准确定义它们到底是什么。本书也涉及到了“用户会话”，也没有相应的比较完全的描述。前面有意地省略这方面的内容，因为它们与 ASP的应用程序和会话密切相关。下面将介绍 ASP的Application和Session对象。

本章研究的主要内容：

- Web应用程序是什么，以及它们如何与 ASP Application对象相联系。
- 用户会话是什么，以及它们如何与 ASP Session对象相联系。
- ASP如何自动地创建和管理应用程序和会话。
- Application和Session对象提供的功能。
- 如何把 Application和Session对象放入 ASP网页中。

首先研究整个内容的核心问题：状态。

### 3.1 Web上的状态管理

许多开发人员把应用程序传送到 Web之前从来没有考虑状态的概念。正如前面说过的，Web是一个无状态的环境。因此应该探讨一下状态是什么，了解能够避免产生问题的方法。

#### 3.1.1 状态的准确定义

在单用户程序中，创建一个可执行的应用程序时，例如使用 VB建立的一个 .exe文件，可以声明一个全局(或Public)变量，然后在代码中任何地方可对其进行访问。在应用程序运行的所有时刻，该变量的值一直是有效，并且是可访问的。

对于一个传统的客户机/服务器解决方案，例如一个基于客户机的应用程序对一个基于服务器的数据库引擎进行访问的系统，每个客户端建立了一个与服务器和数据库应用程序的连接。这种连接通常是通过验证用户的方法来建立的。

验证过程是典型的识别用户身份的过程，通过一个用户名和口令组合来证明是否为用户。

一旦通过验证，在客户端和基于服务器的应用程序之间就建立了连接，该连接在用户使用该应用程序的所有时间内一直保持有效。当用户注册到本地 Windows 2000服务器上时，这一切便会发生。无论何时，管理员使用“Active Directory Users and Computers”实用程序(单击“Start”菜单的“Administrative Tools”选项中的“Directory Management”项)都可以观察到活动的用户连接。这个过程在许多系统中都相同，例如 Microsoft SQL Server。

这种永久的连接意味着：当用户发送指令或请求到服务器上时，服务器会很容易地识别每个用户。同样服务器的响应或任何其他用户的信息也能直接返回用户。要进一步指出的是服务器可以比较容易地存储与每个客户相关的值和信息，并在需要的时候提供给相应的客户。当然，服务器应用程序能够拥有主全局变量，以便于用户在需要的时候进行访问。

这种识别每个客户端的请求并在内存中保存相关用户的值的能力构成状态。可以认为状态代表应用程序的值、环境以及用户的内部变量，并贯穿于应用程序和用户连接的整个过程。

### 3.1.2 状态的重要性

如果打算创建与用户进行交互的基于 Web 站点的应用程序，而不是仅显示独立页面的 Web 网站，必须能够为每个用户提供独立的状态。这可能只是记住他们的名字，也可能要为每个用户存储对象引用或不同的记录集。如果不能这样做，ASP 网页就不能做更多的事情，因为该页面执行完成时，页面中的变量和其他相关资料都破坏了。当用户请求下一个页面时，这个页面提供的所有信息将全部失去。

因此，需要找到一种方法，保存每个访问者的状态。能够存储对所有用户而言的全局值是非常重要的。例如，一个 Web 风格的访问或页面点击计数器，它不为每个用户提供自己的计数器，用户们通常想要看到访问者的总数，而不仅仅是他们自己访问的次数。访问者的数目需要与应用程序级状态一起存储，而不是与用户级状态一起存储。

这不是一个刚出现的问题，自从商用站点占据了 Web，就已经存在，甚至更早些。所以已有许多在 Web 上存储状态的传统的解决方案。Web 站点管理员想要了解访问者以前是否曾访问过他们的网站，如果访问过，访问过多少次？是否经常访问？还定期访问其他什么网站等。这样可以更好地制定其广告目标。所有这些都要求一种方法来存储有关用户在访问时所产生的网页请求或每次访问间的信息。

### 3.1.3 在 Web 上创建状态

在页面请求和站点访问之间提供状态常用的方法是通过 cookie。我们在前面的章节中已经看到，如何在客户端的计算机中存放相应的值，这些值与每个页面请求一起发送给对此 cookie 有效的域。通过用 ASP 检查和更新 cookie，在某种程度上能够保持一个状态。可以使用所包含的信息来识别用户，然后把用户连接到一个已存储相应值的集合。

例如，可以检测一个用户请求是否包含一个站点指定的 cookie。如果不包含，则为该用户分配一个某种类型的标识，指明一个数量，并存储在带有一个长有效期的 cookie 中。以后该用户对这个站点的每一次访问，都能够检测到 cookie 并更新所包含的信息。同时可以收集有关访问的次数和持续时间的数据，并存储在服务器上，以备将来使用。

但是，如果用户转移到另一个计算机，或删除了 cookie，或者他们的浏览器拒绝接收发送给他们的 cookie，会发生什么事情呢？在这种情况下，不能维持状态，因为下一次不能识别他

们现在，Web上有许多 cookie，大多数人会接受它们，而不加理会。如果打开浏览器中的“Warn before accepting cookies”选项，接着漫游几个大的站点，你就会明白其中的含意。

### 1. 匿名访问者与授权的访问者

如果认为 cookie 是一个有点草率的解决方案，可以使用更直接的方法。许多站点采用的一种方法是，在访问者点击一个站点时，或者点击一个要求验证身份的页面时，弹出一个进行登录的对话框。访问者首先必须进行注册，获得一个某种类型的用户名 / 口令的组合，才能允许访问相应的站点或页面。

为了证实访问者是一个已知的并且合法的用户，在访问者的计算机上放置的一个 cookie，它或者保存注册的详细数据，或者是一把表明已验证过身份的“钥匙 (key)”。同时，访问者的详细数据永久地保存在服务器上，准备再次访问时使用。如果访问者的浏览器中有了这样一个 cookie，他就可以自由地访问该网站，因为已经验证过了。

如果 cookie 没有有效期限 (Expires)，cookie 的值在关闭浏览器时自动消失，在下一次访问时必须重新注册和再次验证。当然，如果拒绝接收 cookie 或删除了 cookie，就只能再次得到登录对话框。这样的话，如果不被识别，就不能访问该站点。

通过强制用户就像登录到自己的网络一样登录到 Web 服务器，Windows 2000 整体安全性能为 IIS 提供更强和更安全的验证功能。但是，这只能与 Internet Explorer 3.0 和之上版本的浏览器一起工作。IIS 也可以使用 BASIC 验证允许非 Microsoft 浏览器注册 Web 服务器。

### 2. 不再有匿名访问者

在 IIS Web 服务器上使用 ASP 时，除非用户离开该站点到另一个网站或者关闭了浏览器，否则能在当前会话中跟踪用户。在本章的后面，将看到如何使用这个功能来标识一个访问者、存储用户的本地信息和提供状态。下面与已经讨论过的解决方案相比较，讨论其工作方式。

ASP 和 IIS 共同提出了一个用户会话的概念，通过 ASP Session 对象进行交互。在每个访问者第一次访问服务器上的一个 ASP 网页时，为他创建一个新的并且独立的会话对象，分配给该会话一个会话标识号，并把包含会话标识符的特殊加密版本的一个 cookie 发送给客户。

cookie 的路径 (参看前面的章节有关 cookie 属性的描述) 设置为运行在服务器上的 ASP 应用程序的根路径。这很可能是缺省的 Web 网站的根目录 (即 “/”)，但也可能会是另外一个值 (稍后会看到)。在 cookie 中没有提供 Expires 值，所以当浏览器关闭时，cookie 值也就消失。

每当这个用户访问这个 ASP 网页，ASP 都会查找这个 cookie。命名为 ASPSESSIONIDxxxxxxx，其中每个 x 是一个字母字符。从第 2 章图 2-7 所示的 ServerVariables 集合，能够在 HTTP 报头中看到它。这里高亮地显示 ASP cookie，如图 3-1 所示。

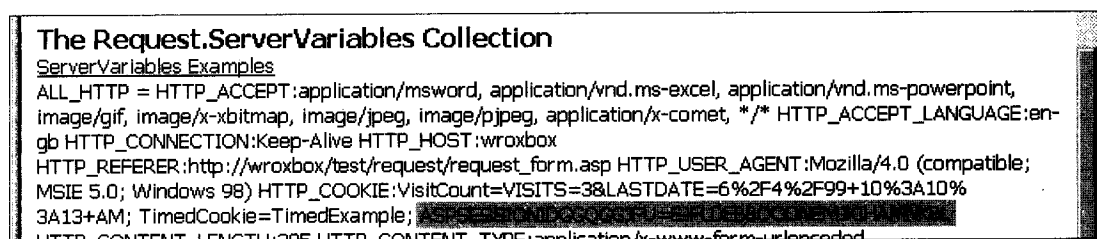


图3-1 显示的 cookie 值

但是，这个cookie不会出现在Request.Cookies或Response.Cookies集合中，ASP把它隐藏起来，但仍保存在浏览器上。对于每个ASP网页的请求，ASP都要查看该值。这个cookie包含的值，指明了这个用户的会话。因此，相应的Session对象(该对象在内存中已被处理，并且一直包含所有在前一页面请求过程中进行操作的值)的内容可以移交给ASP网页中的脚本。

当然，如前所述，如果客户浏览器不接收或不支持这些cookie，这个处理将失败。在这种情况下，不能创建ASP会话，对这个访问者的状态也不进行自动维护。

## 3.2 Web应用程序的定义

前面的章节中已经多次使用过Web应用程序(Web Application)术语，所指的既不是一个真正意义上的Web网站，又不是一个传统的应用程序。换句话说，而是认为它是一些Web网页和用来完成某些任务的其他资源的一个集合。它隐含这样一层意思：有一个预定义的路线贯穿于网页之中，用户可做出选择或提供信息使任务能够完成。

例如，一个在线商店，你为了购买货物，进行反复的观察和选择，浏览一系列网页，收集所需要的信息，支付相应的费用，最后发出定单。也可能是一个“软件升级向导”，指导用户完成下载和安装新软件的过程，或者可能是一个基于Intranet的报价单或销售报告的生成工具。

所有这些不同于“标准”的Web网站，一般的Web站点使用一系列菜单或导航栏以预先未定义的路径漫游该站点。但是一个Web应用程序远不只是受控制的导航器。自由地漫游于一个Web网站时，可以进行无状态的和匿名的访问，但Web应用程序一般不接受。

### 3.2.1 ASP应用程序的定义

上述内容可以认为是术语“Web应用程序”的一个合理的一般定义，但遗憾的是，在谈论有关“ASP应用”时，仅这些还不够。回答什么是“Web应用”可以是主观的，而回答什么是“ASP应用”则需要从技术上的解释。在ASP中术语“应用程序”有自己特定的含义，在讨论如何实现之前，弄懂这个概念至关重要。

ASP应用程序与两个主要的内容有关：

- 全局范围的规定，具有一个全局可访问的变量存储区域。
- 通过COM+与IIS的集成，可更好地管理组件。

下面讨论这些内容。第二个内容涉及到其他ASP对象，其覆盖范围相当广泛。在下一章研究ASP Server对象时，将对这部分内容进行详细地讨论。

#### 1. 提供Web应用程序的全局范围

ASP提供一个Application对象，基本上与前面讨论的Session对象相当。但是，这是在应用程序层而不是在用户层。换句话说，该对象是全局的，不是对单独用户的，而是对应用程序的所有用户，其作用域不限制为单独用户的访问。这与在一个正常的可执行应用程序中的全局(或Public)变量相同。Application对象可用于在全局环境中存储变量和信息(即状态)，该应用程序内的任何ASP网页中运行的脚本都可访问这些值，而不管是哪个访问者发出的请求。

但是，这没有回答主要问题：什么是一个ASP应用程序？为此，需要研究ASP内部的一些情况。

当用户请求一个ASP网页时，IIS通过实例化asp.dll(用来实现ASP)创建一个环境(如第1章



所述)。将该页面解释为服务器端脚本，相应的脚本引擎的实例用来执行该脚本。

实例化的 asp.dll 初始事件启动一个 ASP 应用程序，创建一个 Application 对象。然后，为用户启动一个会话，并创建单独的 Session 对象。当更多的会话启动时，这个 Application 对象保留在作用域中(即已经实例化和可用)。一旦最后保持活动的会话结束，该应用程序就结束，并取消相应的 Application 对象。因此，当站点上还有活动会话时，将会有有一个单独的 Application 对象提供给所有用户使用。

### (1) 缺省的ASP应用

Windows 2000 在安装 IIS 和 ASP 时，创建了一个缺省的 Web 站点。它被配置成一个 ASP 应用程序，涉及到在 Properties 对话框中针对站点根目录文件夹(缺省为 C:\inetpub\WWWroot)的一些设置。图 3-2 所示是缺省 Web 站点的 Properties 对话框的屏幕。

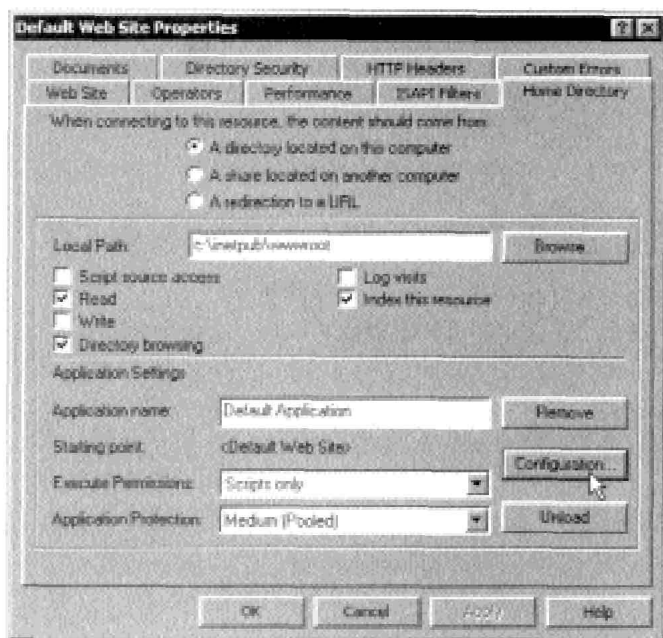


图 3-2 缺省 Web 网站的 Properties 对话框

涉及到 ASP 应用程序的文件之一是 global.asa。这个文件用于定制应用程序行为的方式。放置在应用程序的根目录下，可用于该目录下的所有子目录。因此，如果它放置在整个 Web 站点的根目录下，则定义整个网站作为缺省的 ASP 应用程序的一部分。

在本章后面有关应用程序和会话事件的部分中，将看到这个文件及其使用方法。

### (2) ASP 虚拟应用程序

如同在设置过程中创建缺省的应用程序一样，可以在该 Web 网站的任何子目录中创建属于自己的 ASP 虚拟应用程序。这个应用程序包含作为“应用程序目录”而定义的目录中所有的子目录。并且，这个目录和子文件夹也都是缺省应用程序的一部分，共享由缺省的 Application 对象创建的全局空间。

事实上，在缺省的应用程序中存储的所有变量在子目录中的应用程序中也都是可用的。然而，如果该子目录应用程序中的一个 ASP 网页把一个值写入 Application 对象，而 Application 对象与缺省(根)应用程序中已存在的一个值有相同的名字，那么，原先的值在子目录应用程序

中就不能再用。但是在其他的应用程序或 ASP网页中，将保留原有的值，因为根目录的应用程序不能访问子目录应用中的值。

从一个子程序或函数的变量的角度考虑这个问题。如果定义一个变量 `intMyValue` 为 `Public` 或全局的变量，可以从任何的子程序或函数内部访问该变量。但是，如果又声明一个具有相同名字的局部变量，并在该子程序或函数内对该变量进行引用，则得到此变量的局部值。不能再访问原先的值。当子程序或函数结束，局部值被撤消，全局变量原有的值仍然保留着：

```
Public intMyValue = 42

Function DoSomething()

    Response.Write intMyValue    'Gives 42 from global variable

    Dim intMyValue
    intMyValue = 17
    Response.Write intMyValue    'Gives 17 from new local variable, but
                                'the global value of MyValue is still 42

End Function
```

### (3) 创建自己的ASP虚拟应用程序

为了建立一个新的虚拟应用程序，使用 Internet Services Manager 应用程序或具有相同功能的 HTML Web Manager 网页。在 Internet Services Manager 中，在要创建的虚拟应用程序的目录上单击右键，并选择 `New`，接着选择 `Virtual Directory`，屏幕如图 3-3 所示。

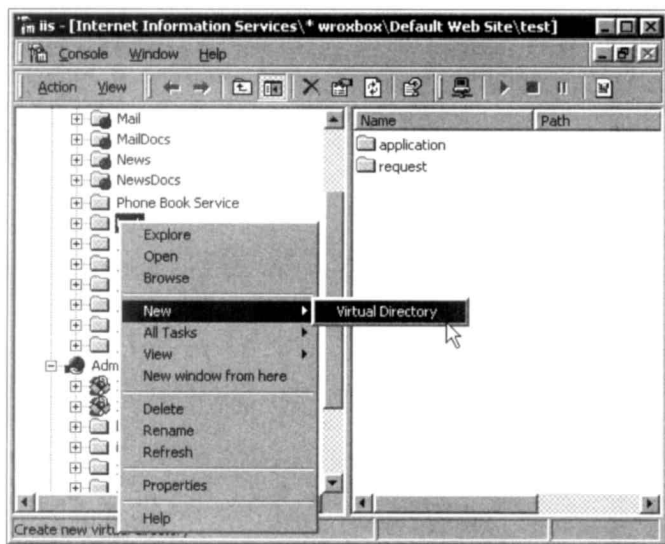


图3-3 创建ASP虚拟应用程序的屏幕

这个操作启动 New Virtual Directory Wizard，该向导的开始屏幕提供有关向导的操作信息。单击 `Next`，在第二页中键入新的虚拟应用程序的名字（或别名）。这个名字与在 Internet Services Manager 中选择的目录的路径联合起来，将成为该应用程序的 URL。屏幕如图 3-4 所示。

为了把一个现有的目录转换为与该目录具有相同名字的一个应用程序，选择包含想要转换的目录的目录，并在向导的 Virtual Directory Alias 页中使用该目录名。例如把已有的 `test` 目录转换为一个虚拟应用程序，应该在 Internet Services Manager 中选择 `Default Web Site` 条目，并提供一个别名 “`test`”。

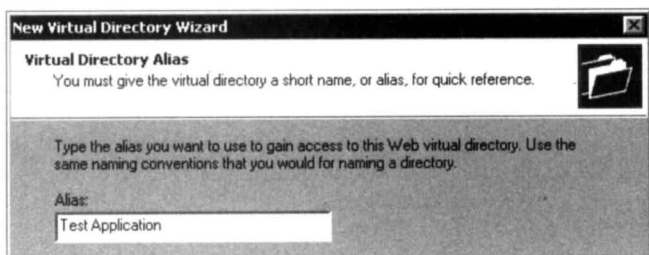


图3-4 New Virtual Directory Wizard的屏幕

再单击Next，指定包含该应用程序的内容(页面)的路径。单击Browse选择一个已有目录。这个目录是新的虚拟应用指向的目录。屏幕如图3-5所示。

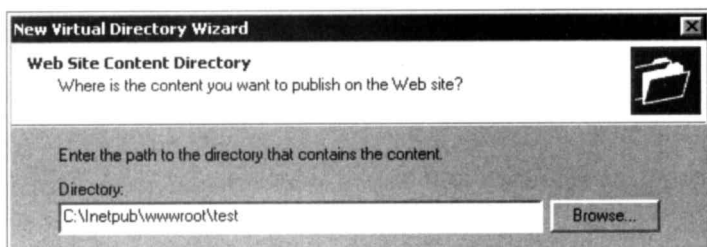


图3-5 指定路径时的屏幕

单击Next打开Access Permissions页，选择给予这个应用程序的所有用户的权限。缺省值是Read和Run Scripts，对大多数用户而言是适合的。屏幕如图3-6所示。

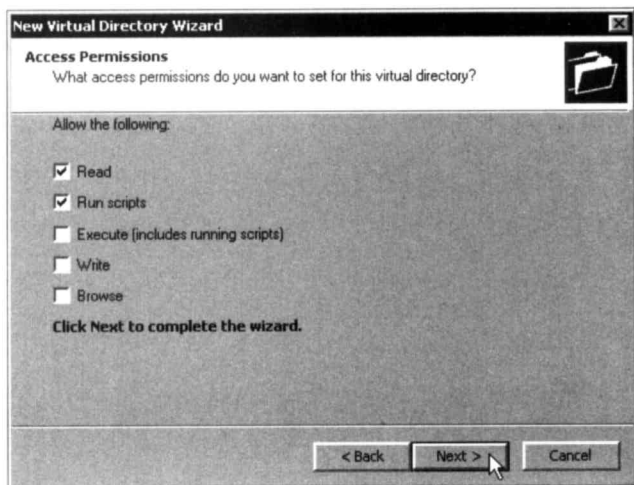


图3-6 设置用户许可权限时的屏幕

如果想编写用户可直接执行的、定制的编译的CGI应用程序，只选择“Execute”：例如，用户通过在请求的URL中指定相应名字的方法执行的一个.exe文件，像“http://mysite.com/.../Test Application/ create\_user.exe?user=JJones”。

单击Next，向导创建该虚拟应用程序。在图3-7所示的屏幕中，可在左边的列表栏中看到带有一个包含一些填充物的打开的小盒子图标。

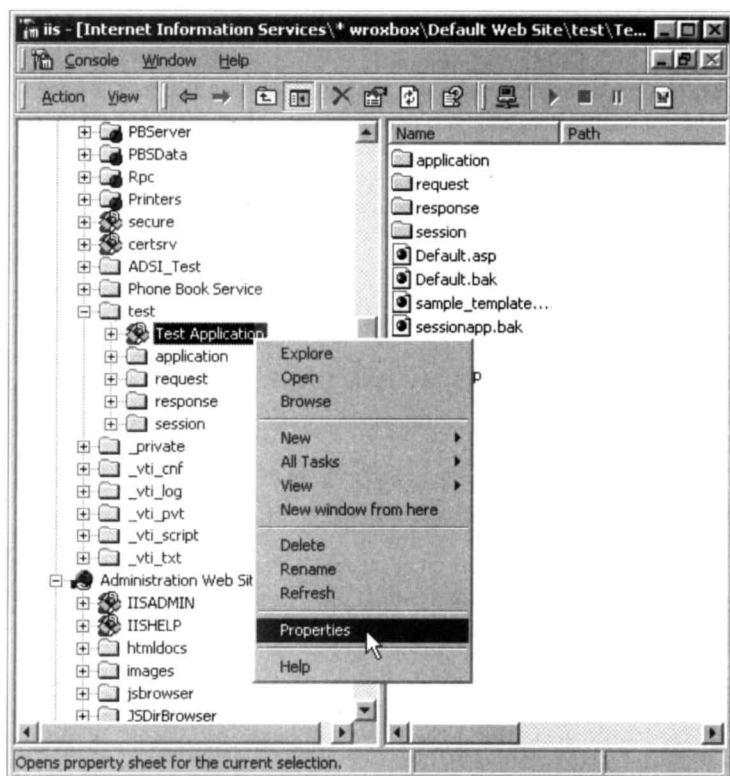


图3-7 虚拟应用程序创建完成后的屏幕

如果现在用右键单击新的应用程序并选择 Properties，可以看到向导已经选择的设置。在这里可根据需要修改访问权限、“Local Path”和“Application Settings”。同时会看到一个 Remove按钮，可以用来删除该虚拟应用程序，如图 3-8所示。

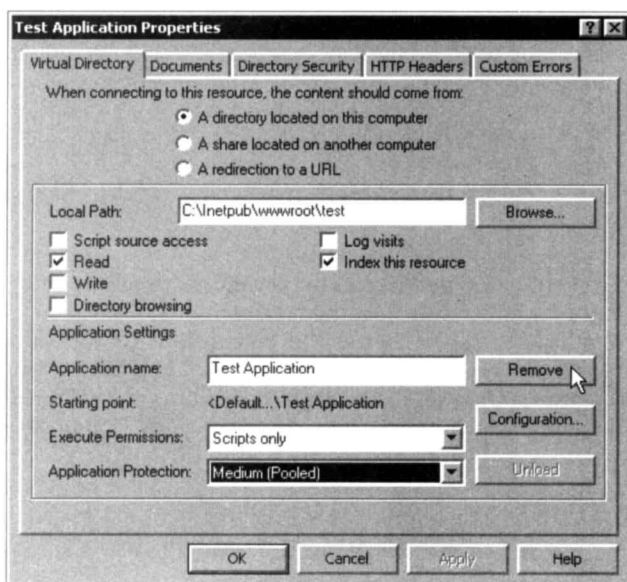


图3-8 虚拟应用程序属性设置的屏幕



#### (4) 删除虚拟应用程序

单击“Remove”按钮不会真正地删除 Internet Services Manager 中的该条目。而是把现有的虚拟应用程序转换为一个虚拟目录。这个目录有一个带有蓝色球的“文件夹”图标，该图标表示这并不是 Web 网站中一个真正的目录，而是对磁盘上另一个文件夹的一个重定向。对它进行的访问方式与创建它的虚拟应用程序相同（即使用相同的 URL），但是不作为一个应用程序。换句话说，它不支持自己的 Application 对象，而是继承缺省 Web 网站的 Application 对象，或者是在该目录的父目录中的另一个应用程序的 Application 对象。

为了删除一个虚拟应用程序，可简单地在 Internet Services Manager 中的对象上单击右键，在弹出的菜单中选择 Delete。

#### (5) 应用程序存储的内容

ASP Application 对象提供的全局存储空间可以用来存储下列内容：

- 简单变量，例如字符串和数字（存储为 Variant，类似于 ASP 脚本变量）。
- Variant 类型数组，一维或多维。
- 对一个 COM 对象实例的变量引用（类似于 Variant）。

#### (6) Variant 的定义

Variant 是在 VBScript 脚本引擎中为 ASP（和 Internet Explorer）提供的唯一变量类型，与在 VB 和 VBA 中定义的数据类型 Variant 相类似。与大多数常用的基本数据类型（例如字符串或整数）相比，尽管它不能使存储单元最高效地存储变量，需要进行额外的处理，但 Variant 提供许多用处。

从内部来讲，Variant 数据类型存储数值作为独立的子类型。它能自动地处理隐含的类型转换，允许使用如下的代码：

```
strString = "30"
intInteger = 12

Response.Write strString & intInteger 'Writes 3012 in the resulting page
Response.Write strString + intInteger 'Writes 42 in the resulting page
```

隐含的数据类型转换允许忽略 HTML 文本的值和列表控件是字符串的情况，并且如果需要的话可以把它们当成数字（假设字符串确实包含一个有效的数字）。也可以随意地做一些其他的事情，诸如使用 VBScript 的 Len 方法检查 Request.Form 集合中并不存在的一个值的长度。Len 函数试图把从 Request.Form 的调用的值转换成一个字符串。如果该值丢失，返回空（Empty），转换为带有 " " 的一个字符串，因此长度是零。

#### (7) VBScript 数据类型及转换

也可以进行显式的数据类型转换。在 VBScript 中，VarType 方法返回一个整数，表明 Variant 当前保存的子类型：

```
Select Case VarType(varMyValue)

Case 0 : Response.Write "Empty (uninitialized)"
Case 1 : Response.Write "Null (no valid data)"
Case 2 : Response.Write "Integer"
Case 3 : Response.Write "Long integer"
Case 4 : Response.Write "Single-precision floating-point number"
Case 5 : Response.Write "Double-precision floating-point number"
Case 6 : Response.Write "Currency"
Case 7 : Response.Write "Date"
Case 8 : Response.Write "String"
Case 9 : Response.Write "Automation Object reference"
Case 10 : Response.Write "Error"
Case 11 : Response.Write "Boolean"
```

```
Case 12 : Response.Write "Variant (used only with items in arrays of Variants)"
Case 13 : Response.Write "Data-access object"
Case 17 : Response.Write "Byte"
Case 8192 : Response.Write "Variant Array"
```

```
End Select
```

还有一些函数，如IsArray、IsDate、IsEmpty、IsNull、IsNumeric和IsObject，对特定的子类型返回一个Boolean结果。一旦知道了数据的子类型，如果包含的数据合适的话，就可以将之转换成不同的Variant子类型。这对代码的透明性和检查变量是否包含合法值都是有用的，非法的转换将导致运行期错误，转换如下所示：

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
dtmDate = CDate(varVariant)	'Converts to a Variant of subtype Date
dblDouble = CDbl(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

#### (8) JScript数据类型及转换

在JScript中，其他一些变量类型类似于VBScript，但是没有代表对象的Variant。所有值都是对象，类型是下列六种数据类型之一：

- undefined：只有单个值?undefined?，用于表明请求中的变量没有声明和创建，或者若是隐含创建但还未分配任何值。类似于VBScript的Empty。
- Null：变量不包含一个有效的值。类似于VBScript的Nothing。
- Boolean。
- String。
- Number。
- Object。

JScript提供了一个typeof函数，返回表示数据的类型的字符串，例如：

```
strString = '30';
intInteger = 12;

Response.Write(typeof(strString)); // Writes 'string' in the resulting page
Response.Write(typeof(intInteger)); // Writes 'number' in the resulting page
```

每种数据类型都有toString方法和valueOf方法，toString方法将变量的值转换为一个字符串返回，valueOf方法把变量的值作为它的原有数据类型返回。

#### (9) JScript的级联和加法问题

JScript在许多方面不同于VBScript，在一定程度上是因为没有“&”级联运算符可用。当使用加法运算符时，它先检查变量的数据类型以决定要做什么。如果两个变量都是数值，结果是数值的和。如果一个或两个是字符串，结果是字符串的级联：

```
strString = '30';
intInteger = 12;

Response.Write(intInteger + intInteger); // Writes 24 in the resulting page
Response.Write(strString + intInteger); // Writes 3012 in the resulting page
Response.Write(intInteger + strString); // Writes 1230 in the resulting page
```

如果进行加法的值超过两个，要看执行的顺序。下面的代码行进行相应的示范：

```
intInteger = 12;
Response.Write(intInteger + intInteger + "<P>"); // Result is '24<P>'
Response.Write("<P>" + intInteger + intInteger + "</P>"); // Result is '<P>1212</P>'
```

在第一种情况下，两个数相加，结果被转换为一个字符串并与 "<P>"字符串进行级联。在第二种情况下，第一个运算符是一个字符串和一个数的级联，所以数被转换为一个字符串。而后，所有的运算符都是级联运算符。为了避免这个问题，可以使用括号强制第一个运算符是两个数值的加法：

```
Response.Write("<P>" + (intInteger + intInteger) + "</P>");  
// result is '<P>24</P>'
```

2. Web应用程序中的组件管理

在ASP中使用虚拟应用程序的第二个主要方面是：具有较好的对在脚本代码内实例化和执行的组件进行管理的能力。先不讨论有关它怎样进行工作以及为什么会如此有用的详细情况，在学习ASP组件的部分时再讨论相关内容。

这里非常简要地介绍在 ASP网页(此ASP网页在该应用程序中)中使用组件时，如何对一个虚拟应用程序在Properties对话框(在Internet Services Manager中)进行设置。

在一个虚拟应用程序的 Properties对话框的Home Directory页的底部，有两个组合框，为Execute Permissions和Application Protection，如图3-9所示。

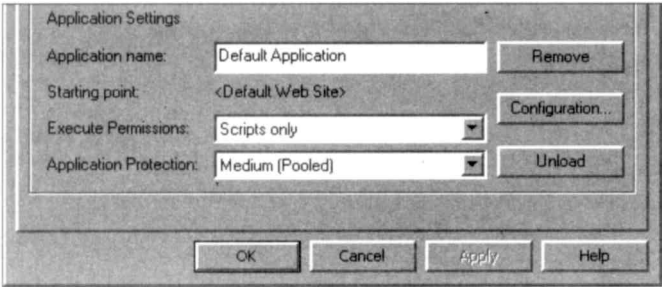


图3-9 Execute Permissions和Application Protection组合框

应用的保护和执行设置

因为在本章中，不讨论在一个 Web网页内如何创建组件的实例，因此这里先列出这些选项，在创建应用程序时可能要对此进行设置。Execute Permissions选项如表3-1所示。

表3-1 Execute Permissions 的选项及说明

选 项	说 明
None	在这个虚拟应用程序中不能运行脚本或可执行文件。实际上，提供了禁止一个应用程序的快速和简单的方法
ScriptsOnly	只允许脚本文件(例如ASP、IDC或其他的)在这个虚拟应用程序中运行，不能运行可执行文件
Scriptsand Executables	允许任何的脚本和可执行文件在这个虚拟应用程序内运行

Execute Permissions 选项控制可在该虚拟应用程序中执行的类型，而 Application Protection ” 选项影响可执行文件和组件运行的方式。在第 1章中已经讨论过可用的选项，但是在这里再重复一次，Application Protection选项如表3-2所示。

Microsoft推荐一个配置是：关键性应用程序运行在自己的进程中，即 High(Isolated)；其他的应用程序运行在共享的进程中，即 Medium(Pooled)。

现在，已经讨论了 ASP应用程序涉及到的相关内容，现在详细地讨论 ASP Session对象方

面的内容。

表3-2 Application Protection的选项及说明

选 项	说 明
Low(IIS Process)	带有这种设置的ASP虚拟应用程序的所有可执行文件和组件运行在 Web服务器的可执行文件 (Inetinfo.exe)的进程(即内存空间)中。因此, 如果可执行文件或组件之一失败的话, Web服务器处于危险状态。这提供了最快和以最少的资源执行的选项
Medium(Pooled)	(缺省)带有这种设置的ASP虚拟应用程序的所有应用程序的可执行文件和组件是运行在DLLHost.exe的单个共享实例的进程(即内存空间)中。这就防止了Web服务器可执行文件 (Inetinfo.exe)受可执行文件或组件失败的影响。然而, 一个失败的可执行文件或组件可能引起 DLLHost.exe进程失败, 以及所有其他驻留其中的可执行文件和组件失败
High(Isolated)	带有这种设置的ASP虚拟应用程序的所有应用程序的可执行部分和组件是运行在DLLHost.exe的单个共享实例的进程(即内存空间)中, 但是每个ASP应用程序都有自己的DLLHost.exe实例, 该实例对该应用程序是独占的。这就防止了 Web服务器可执行文件 (Inetinfo.exe)受可执行文件或组件失败的影响, 并防止虚拟应用的单个共享实例受另一个虚拟应用程序的一个可执行文件或组件失败的影响。Microsoft建议最多有十个这样的虚拟应用程序驻留在一个 Web服务器上

### 3.2.2 ASP会话的定义

ASP会话引入了一个Web应用程序中粒度的下一层。ASP的Application对象可用来存储对于“正在运行此应用程序”的所有用户都是全局的和可访问的状态(即简单变量、对象、数组等)。换句话说, 用于响应这个应用程序内所有访问者的请求的全部 ASP代码能够对这些值进行访问(假设已经建立了一个有效的会话, 稍后将看到)。

但是在多数情况下这还不够。需要具备存储指定给每个用户的值的能力, 而不必通过给这些值分配名字指明其隶属于哪个用户。例如, 下列值很可能弄乱应用程序的全局存储空间:

```
MikeJones003PrefFGColor = "darkblue"
MikeJones003PrefBGColor = "white"
MikeJones003PrefLinkColor = "green"
PriscillaDelores001PrefFGColor = "red"
PriscillaDelores001PrefBGColor = "darkgrey"
...
etc.
```

从载入系统资源和要求一些代码访问每个用户的相应会话的角度来看, 还有另外的不足之处。只要有访问者, 该应用程序就一直存在, 这意味着应用程序的全局存储空间需要不断增大, 除非在用户离开该网站时采取步骤删除这些值。

在Web应用程序中提供用户层作用域

除了使用全局变量的存储以外, 应该为每个访问者分配他们自己的私有变量存储空间, 使其对指定访问者载入的所有页面都可用的。这种情况下, 可以对每个变量使用相同的名字, 使ASP代码非常简单地得以实现。这些相同的代码对每个用户将透明地进行工作, 因为访问的只是访问者拥有的私有存储区域:

```
PrefFGColor = "darkblue"
PrefBGColor = "white"
PrefLinkColor = "green"
```

这就是Session对象产生的地方。



### (1) 会话存储的内容

会话存储的内容对指定的访问者是全局的，而对其他访问者来讲是私有的，这使得 ASP 的会话非常有用。可以用来存储在 ASP Application 对象中存储的相同类型的数据，即：

- 简单变量，例如字符串和数值 (像所有的 ASP 脚本变量一样存储为 Variant)。
- Variant 数组，一维或多维。
- 对一个 COM 对象的实例的变量引用 (如同 Variant)。

### (2) 会话带来的问题

会话提供了一个存储每个用户特定的值的方法。然而有几个意想不到的问题要注意：

- 记住一些浏览器和 Web 服务器对 URL、路径和文件名的大小写形式是敏感的 (例如 Navigator 和基于 UNIX/Linux 的服务器)。如果把一个超级链接放置在网页的一个 URL 上，并且它们不是同样的书写形式，则在浏览器中被认为是不相同的。同样，如果路径和文件名的书写形式不是相同的，则浏览器认为是不同的路径或文件。对于在服务器上的定位资源来讲，这并不重要，因为 IIS 对书写形式不敏感，可以接受大写形式和小写形式的任何组合，并返回书写形式不同的具有相同字符的文件。然而，如果一个 cookie 已经指定了一个路径，而且与在超级链接中所指定的路径在书写形式上不同，浏览器可能不会把它以及相应目录的页面一起返回给服务器。这有可能找不到依赖于这个 cookie 的一个用户会话，并且 Session 对象将不会在作用域中 (即其中的任何变量都是不可用的)。因此，在所有的目录和网页名字中，坚持都采用小写形式或者是比较明显的混合字母形式，是一个好办法。
- 在 IIS 和 ASP 的早期版本中，对于嵌套的应用程序还有一些小的 “Bug”，有时，当用户离开嵌套的 ASP 虚拟应用程序并返回到缺省的 ASP 应用程序层时，与嵌套应用程序内部已经定义的局部变量相同名字的任何全局变量不能重新显现。还有，当会话使用 Session.Abandon 方法 (稍后将会看到) 终止时，global.asp 文件中的代码将执行失败。在 ASP 3.0 中已经解决了这些问题。
- 记住会话依赖于 cookie。如果访问者已经禁止使用 cookie 或者浏览器不支持 cookie，将不能启动一个会话，并且不能访问 Session 对象。

### (3) 禁止会话

虽然状态提供了有益环境，但让用户门户大开。如果不需要保留状态，可以禁止会话以节约计算机的处理时间。例如，在一个不需要跟踪访问者或不需要为访问者保留全局值的 Web 网站上，可以防止会话启动，通过在 Internet Services Manager 中设置属性，或为不需要状态的独立网页增加代码 (稍后将看到)。

为了禁止整个 Web 网站的所有会话，可编辑缺省 Web 网站应用程序的属性。为了禁止一个指定应用程序的会话，可编辑相应虚拟应用程序的属性。打开相应的应用程序的 Properties 对话框，在 Home Directory 页上单击 Configuration 按钮，如图 3-10 所示。

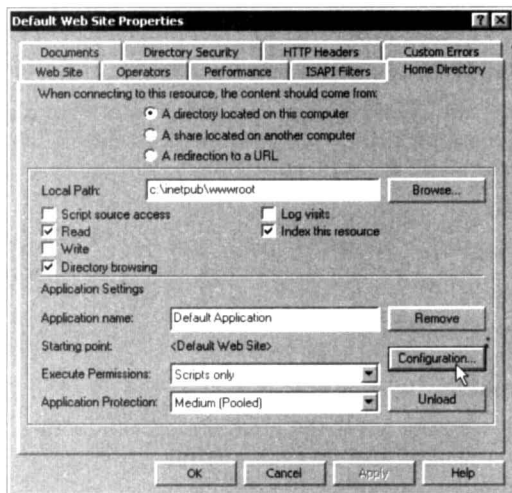


图3-10 Home Directory页

在出现的 Configuration 对话框中, 打开 App Option。这里可以允许或禁止整个应用程序的会话(在本例中是整个缺省 Web 站点), 也可以修改缺省的会话 Timeout 值。在图 3-11 中可以看到 Timeout 设置为 20min。在 ASP 的早期的版本中这是缺省值, 可根据要求设置相应的值(在 ASP 3.0 中, 缺省值是 10min)。

如果要禁止一个指定网页的会话, 同时允许它们在同一个应用程序的其他网页中创建和使用, 可为该页面增加一条 ASP 处理指令。它跟在指定缺省语言的语句后(如果没指定一个缺省语言, 可单独使用该指令):

```
<%@LANGUAGE="VBScript" ENABLESESSIONSTATE="False"%>
```

到此为止, 大致介绍了 ASP 应用程序和 Session 对象的一些情况, 下面详细地进行讨论。

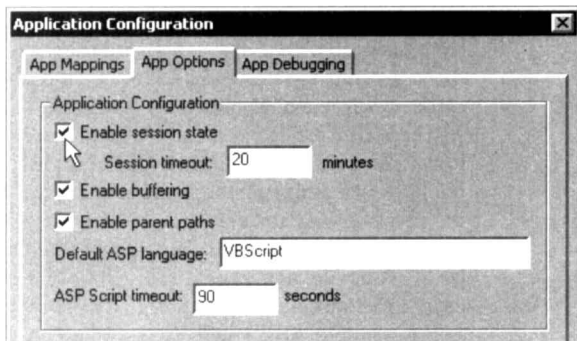


图3-11 App Options页

### 3.3 ASP的Application对象和Session对象

本章已经讨论了两个 ASP 对象: Application 对象和 Session 对象, 因此能够访问 Application 对象和 Session 对象提供的集合、方法、属性和事件。本节将从程序设计的角度对这两个对象进行研究。

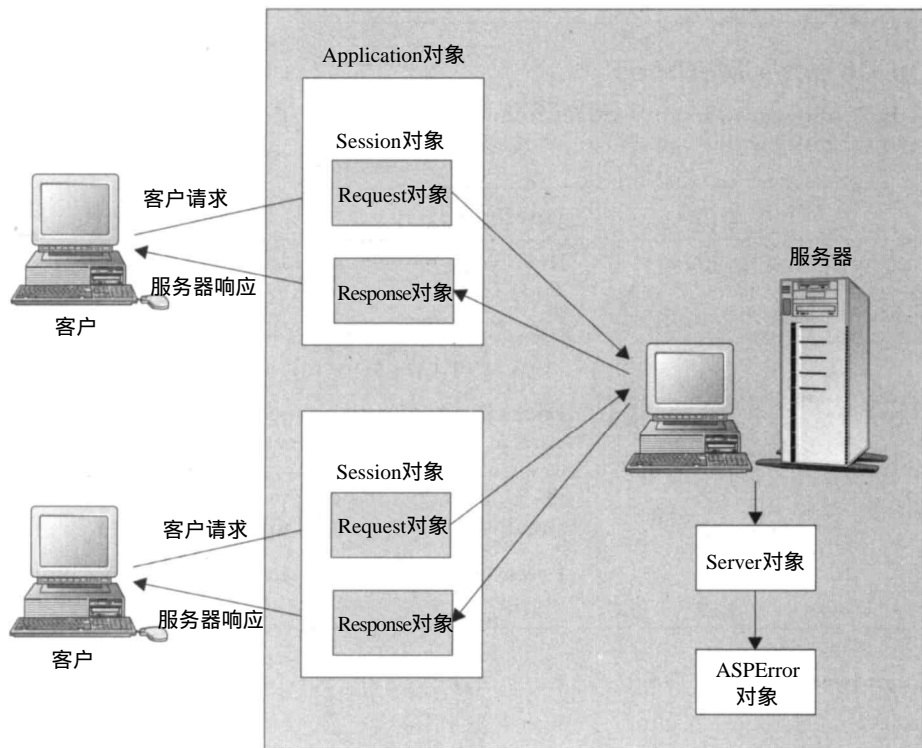


图3-12 ASP对象关系

- 当载入 ASP DLL 并响应对一个 ASP 网页的第一个请求时，创建 Application 对象。该对象提供一个存储场所，用来存储对于所有访问者打开的所有网页都可用的变量和对象。
- 当访问者首次从站点请求一个 ASP 页面时，为他创建一个 Session 对象，并保持有效直到缺省的超时周期 (或由脚本确定的超时周期)。该对象提供一个存储场所，用来存储仅仅对该访问者在会话的活动期间打开的网页可用的变量和对象。

图3-12(即图1-20)给出了用户的请求和服务器的响应在会话中的分布情况。所有的会话都在ASP应用程序中。

3.3.1 ASP的 Application对象成员概述

本节说明 Application 对象的集合、方法和事件 (Application 对象没有属性)。在下一节对 Session 对象(具有属性)进行同样的说明。然后将继续探讨使用这些对象所完成的任务，更详细地说明每个对象的各成员如何进行工作。

1. Application对象的集合

Application 对象提供了两个集合，可以用来访问存储于全局应用程序空间中的变量和对象。集合及说明如表 3-3 所示。

表3-3 Application 对象的集合及说明

集 合	说 明
Contents	没有使用 <OBJECT> 元素定义的存储于 Application 对象中的所有变量 (及它们的值) 的一个集合。包括 Variant 数组和 Variant 类型对象实例的引用
StaticObjects	使用 <OBJECT> 元素定义的存储于 Application 对象中的所有变量 (及它们的值) 的一个集合

2. Application对象的方法

Application 对象的方法允许删除全局应用程序空间中的值，控制在该空间内对变量的并发访问。方法及说明如表 3-4 所示。

表3-4 Application对象的方法及说明

方 法	说 明
Contents.Remove("variable_name")	从 Application.Content 集合中删除一个名为 variable_name 的变量
Contents.RemoveAll()	从 Application.Content 集合中删除所有变量
Lock()	锁定 Application 对象，使得只有当前的 ASP 页面内容能够进行访问。用于确保通过允许两个用户同时地读取和修改该值的方法而进行的并发操作不会破坏内容
Unlock()	解除对在 Application 对象上的 ASP 网页的锁定

注意，在运行期间不能从 Application.StaticObjects 集合中删除变量。

3. Application对象的事件

Application 对象提供了在它启动和结束时触发的两个事件，如表 3-5 所示。

表3-5 Application对象的事件及说明

事 件	说 明
OnStart	当 ASP 启动时触发，在用户请求的网页执行之前和任何用户创建 Session 对象之前。用于初始化变量、创建对象或运行其他代码
OnEnd	当 ASP 应用程序结束时触发。在最后一个用户会话已经结束并且该会话的 OnEnd 事件中的所有代码已经执行之后发生。其结束时，应用程序中存在的所有变量被取消

### 3.3.2 ASP的Session对象成员概述

本节概述Session对象的所有成员。

#### 1. Session对象的集合

Session对象提供了两个集合，可以用来访问存储于用户的局部会话空间中的变量和对象。这些集合及说明如表3-6所示。

表3-6 Session 对象的集合及说明

集 合	说 明
Contents	存储于这个特定 Session对象中的所有变量和其值的一个集合，并且这些变量和值没有使用<OBJECT>元素进行定义。包括 Variant数组和 Variant类型对象实例的引用
StaticObjects	通过使用<OBJECT>元素定义的、存储于这个 Session对象中的所有变量的一个集合

#### 2. Session对象的特性

Session对象提供了四个属性。这些属性及说明如表 3-7所示。

表3-7 Session 对象的属性及说明

属 性	说 明
CodePage	读/写。整型。定义用于在浏览器中显示页内容的代码页 (Code Page)。代码页是字符集的数字值，不同的语言和场所可能使用不同的代码页。例如， ANSI代码页 1252用于美国英语和大多数欧洲语言。代码页 932用于日文字
LCID	读/写。整型。定义发送给浏览器的页面地区标识 (LCID)。LCID是唯一地标识地区的一个国际标准缩写，例如， 2057定义当前地区的货币符号是 ' £ '。LCID也可用于 FormatCurrency等语句中，只要其中有一个可选的 LCID参数。LCID也可在ASP处理指令 <%...%>中设置，并优先于会话的 LCID属性中的设置。本章后面提供一个 ASP处理指令的列表
SessionID	只读。长整型。返回这个会话的会话标识符，创建会话时，该标识符由服务器产生。只在父 Application对象的生存期内是唯一的，因此当一个新的应用程序启动时可重新使用
Timeout	读/写。整型。为这个会话定义以分钟为单位的超时周期。如果用户在超时周期内没有进行刷新或请求一个网页，该会话结束。在各网页中根据需要可以修改。缺省值是 10min，在使用率高的站点上该时间应更短

#### 3. Session对象的方法

Session对象允许从用户级的会话空间删除指定值，并根据需要终止会话。 Session对象的方法及说明如表3-8所示。

表3-8 Session对象的方法及说明

方 法	说 明
Contents.Remove("variable_name")	从Session.Content集合中删除一个名为 variable_name的变量
Contents.RemoveAll()	从Session.Content集合中删除所有变量
Abandon()	当网页的执行完成时，结束当前用户会话并撤消当前 Session对象。但即使在调用该方法以后，仍可访问该页中的当前会话的变量。当用户请求下一个页面时将启动一个新的会话，并建立一个新的 Session对象(如果存在的话)

注意，在运行期间不能从 Session.StaticObjects集合中删除变量。



4. Session对象的事件

Session对象提供了在启动和结束时触发的两个事件，如表 3-9所示。

表3-9 Session 对象的事件及说明

事 件	说 明
OnStart	当ASP用户会话启动时触发，在用户请求的网页执行之前。用于初始化变量、创建对象或运行其他代码
OnEnd	当ASP用户会话结束时触发。从用户对应用程序的最后一个页面请求开始，如果已经超出预定的会话超时周期则触发该事件。当会话结束时，取消该会话中的所有变量。在代码中使用 Abandon方法结束ASP用户会话时，也触发该事件

3.3.3 使用Application 和Session的事件

ASP的Application和Session对象体现了其他 ASP内置对象所没有的特征——事件。然而，正像在前面的对象成员表中看到的那样，这些都是与 ASP会话和应用程序的工作相联系的事件。

1. Application 和Session的事件处理器

每当一个应用程序或会话启动或结束时， ASP触发一个事件。可以通过在一个特殊的文件中编写普通的脚本代码来检测和应答这些事件，这个文件名为 global.asa，位于一个应用程序的根目录中(对于缺省的Web网站是\InerPub\WWWRoot目录，或是作为一个实际应用程序定义的一个文件夹)。这个文件可以包含一个或多个 HTML的<OBJECT>元素，用于创建将在该应用程序或用户会话内使用的组件实例。

在第4章中将详细地介绍如何创建组件实例。下面的代码是 global.asa文件的一个例子。我们只关注<OBJECT>元素以及以Set关键字开始的那些代码行：

```
<!-- Declare instance of the ASPCounter component
with application-level scope -->
<OBJECT ID="ASPCounter" RUNAT="Server" SCOPE="Application"
PROGID="MSWC.Counters">
</OBJECT>

<!-- Declare instance of the ASPContentLink component
with session-level scope -->
<OBJECT ID="ASPContentLink" RUNAT="Server" SCOPE="Session"
PROGID="MSWC.NextLink">
</OBJECT>

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

Sub Application_onStart()

'Create an instance of an ADO Recordset with application-level scope
Set Application("ADOConnection") _
    = Server.CreateObject("ADODB.Connection")
Dim varArray(3) 'Create a Variant array and fill it
varArray(0) = "This is a"
varArray(1) = "Variant array"
varArray(2) = "stored in the"
varArray(3) = "Application object"
Application("Variant_Array") = varArray 'Store it in the Application
Application("Start_Time") = CStr(Now) 'Store the date/time as a string
Application("Visit_Count") = 0 'Set counter variable to zero

End Sub

Sub Application_onEnd()
Set Application("ADOConnection") = Nothing
End Sub
```

```

Sub Session_onStart()

'Create an instance of the AdRotator component with session-level scope
Set Session("ASPADRotator") = Server.CreateObject("MSWC.AdRotator")
Dim varArray(3)                                'Create a Variant array and fill it
varArray(0) = "This is a"
varArray(1) = "Variant array"
varArray(2) = "stored in the"
varArray(3) = "Session object"
Session("Variant_Array") = varArray            'Store it in the Session
Session("Start_Time") = CStr(Now)              'Store the date/time as a string

'We can access the contents of the Request and Response in a Session_onStart
'event handler for the page that initiated the session. This is the *only*
'place that the ASP page context is available like this.
'as an example, we can get the IP address of the user:
Session("Your_IP_Address") = Request.ServerVariables("REMOTE_ADDR")

Application.Lock                                'Prevent concurrent updates
intVisits = Application("Visit_Count") + 1      'Increment counter variable
Application("Visit_Count") = intVisits          'Store back in Application
Application.Unlock                              'Release lock on Application

End Sub

Sub Session_onEnd()
Set Session("ASPADRotator") = Nothing
End Sub

</SCRIPT>

```

因为这个global.asa文件用于本章中的示例页面，所以将需要将该文件放到 Web 网站的根目录中，或者放到已配置为一个虚拟应用程序的目录中，并且在该目录中包含有其他示例文件。

### 读取和存储值

注意上面的例子怎样读取和存储 Application和Session的变量，与在 Request和Response对象的集合中所采取的方式相同。设置这些变量的值：

```

Application("variable_name") = variable_value
Application("variable_name") = variant_array_variable_name
Set Application("variable_name") = object_reference

```

获取这些变量的值：

```

variable_value = Application("variable_name")
variant_array_variable = Application("variable_name")
Set object_reference = Application("variable_name")

```

当然，对于Session对象可采取同样的方法。

可以看到，当从一个Session事件处理器访问时，怎样“锁定”(Lock)和“解锁”(Unlock)该Application对象；当从一个ASP网页访问时，需要进行相同的处理。用 Application事件内的代码访问 Application对象中的值时，不要求这么做。这是因为在任何应用程序中只有一个 Application对象的实例，并且其事件处理器的代码只在没有活动的用户会话时运行。

也可看到一个基本的用户会话计数器是如何实现的。这里使用一个应用程序级的变量 Visit\_count，当新的会话启动时它就自动增加。

一般也不限制简单地把值保存到 Application或Session对象中。例如，Web开发者的Web站点在http://webdev.wrox.co.uk上，有相应的一个global.asa文件，当一个新的会话启动时该文件就在服务器上的数据库中写入相应的条目，数据细节从 Request.ServerVariables集合中获取。这提供了一个基本的方法统计访问者的数量，并收集访问者的一些基本信息。

2. 创建Variant数组

在Session和Application对象中创建和使用一个 Variant数组来存储值的方法目前尚未讨论，在这里作为一个非常有用的技术进行讨论。正如已经看到的那样，一个 Variant数据类型可以包含一个数组，而不仅仅是一个值。

一个数组只是在内存的一个连续区域中以指定的次序存储二进制值的一个长行。要安排 Variant，需要指向首项，并给出有关大小和结构的信息，脚本引擎可以做余下的事情。

可在一个 Variant变量中创建一维、二维或多维数组，然后把该数组分配给一个应用程序层或用户会话层的变量，并保证整个数组可在相应的地方使用。下面代码演示了一个简单的一维数组的使用技术：

```
Dim varArray(3)
varArray(0) = "This is a"
varArray(1) = "Variant array"
varArray(2) = "stored in the"
varArray(3) = "Session object"
Session("Variant_Array") = varArray
```

3. 应用程序和会话在何时启动和结束

在介绍ASP应用程序和会话如何进行工作时提到过这个内容。以最基本的术语概述如下：

- 当第一个用户请求应用程序作用域内（即Web网站的缺省根目录内），或者在该网站的一个子目录内的一个用户定义的虚拟应用程序的一个 ASP网页时，启动该应用程序。在任何用户会话启动之前发生。
- 当任意用户第一次请求在缺省应用程序或一个虚拟应用程序内的一个 ASP网页时，启动一个会话(如果还没一个活动的会话)。
- 当用户在会话指定的超时周期内没有下载一个 ASP网页时，会话结束。超时时间可以在脚本代码中使用 Session.Timeout属性进行设置，可在 Properties对话框中对各个应用程序单独设置，或者通过 Active Directory的IIS：部分修改IIS元数据库中的缺省值进行设置。调用Session.Abandon方法的一个网页完成执行以后，会话也会结束。
- 在一个应用程序中的最后一个活动会话结束以后，该应用程序立即结束。

4. ASP处理指令

正如在第1章中所看到的，可以把一条处理指令增加到一个 ASP网页。处理指令可以根据需要包含一个以上的条目。可以在语句中使用的关键字及其说明如表 3-10所示。

表3-10 ASP指令关键字及说明

指令关键字	说 明
LANGUAGE="language_name"	设置该网页的缺省的脚本语言，如：<% @ LANGUAGE = "VBScript" %>
ENABLESESSIONSTATE="True" "Fasle"	当设置为“ True ”时，防止一个会话的 cookie发送到浏览器，因此将不会创建新的 Session对象，任何现有会话的内容将不再可用
CODEPAGE="code_page"	设置该网页的代码页，如<% @CODEPAGE="1252"%>
LCID="locale_identifier"	设置该网页的位置标识符，如<% @LCID="2057"%>
TRANSACTION="transaction_type"	指明该网页文件将在一个事务环境下运行。有效值是：“ Required ”：如果已有可用的事务，脚本将在其中运行；如果没有可用的事务，启动一个新的事务。

(续)

指令关键字

说 明

“Requires\_New”：脚本将初始化一个新的事务。

“Supported”：如果已有可用的事务，脚本将在其中运行；而且不启动一个新的事务。

“Not\_Supported”：脚本将不运行于任何已有的事务中，并且不初始化一个新的事务。

在第18章中详细介绍事务。

在一个网页上只能允许有一条处理指令，并且应该放在第一行。在处理指令中可以包含不止一个这样的条目，但必须用空格进行分隔，等号两端不能有空格，例如：

```
<%@LANGUAGE="VBScript" CODEPAGE="1252" LCID="2057"%>
```

### 3.3.4 活动中的ASP Application对象

我们提供一些简单的网页，这些网页示范了使用过程中的ASP Application和Session对象。为了能够正常使用，必须把它们放到服务器上的一个虚拟应用程序内，并且把所提供的global.asa文件放到该应用程序的根目录中。最简单的办法是把global.asa文件放到缺省Web网站的根目录(缺省情况下是C:/InetPub/WWWRoot)中。

对任何已有的global.asa文件重命名是一个好方法，可以在以后对该文件进行恢复。

本书的所有例子文件都可以从我们的Web网站上得到，在例子的Chapter03子目录中还有本章的所有其余示例网页。

在Chapter03子目录中，Default.asp网页是一个简单的菜单，该菜单允许运行Application和Session示例网页，运行屏幕如图3-13所示。

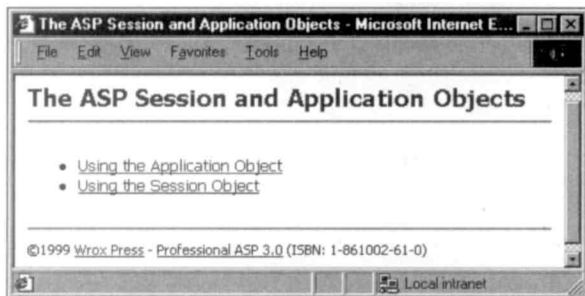


图3-13 Default.asp运行屏幕

#### 1. 显示Application集合的内容

单击第一个链接，打开名为show\_application.asp的Application对象示例页面。它显示了虚拟应用程序当前的Application对象的内容，如图3-14所示。

注意到ASPCounter对象是StaticObjects集合的一个成员(通过<OBJECT>元素进行定义)，但是其余部分(由Server.CreateObject实例化)是Contents集合的成员。

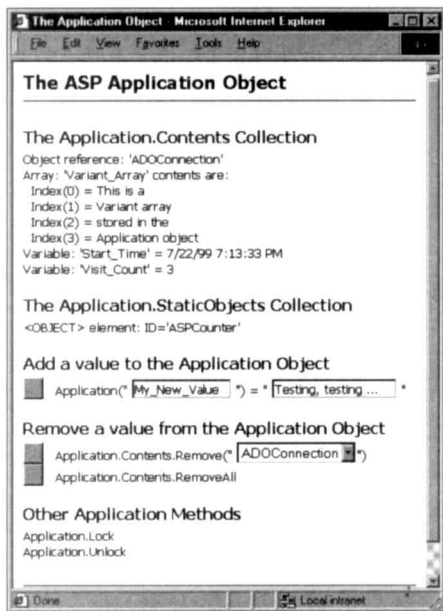


图3-14 Application对象的内容屏幕



可以看到使用global.asa例子网页放到这些集合中的值，这在前面已经看到：

```
<!-- Declare instance of the ASPCounter component with
application-level scope -->
<OBJECT ID="ASPCounter" RUNAT="Server" SCOPE="Application"
PROGID="MSWC.Counters">
</OBJECT>
...
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

Sub Application_onStart()

'Create an instance of an ADO Connection with application-level scope
Set Application("ADOConnection") = Server.CreateObject("ADODB.Connection")
Dim varArray(3) 'Create a Variant array and fill it
varArray(0) = "This is a"
varArray(1) = "Variant array"
varArray(2) = "stored in the"
varArray(3) = "Application object"
Application("Variant_Array") = varArray 'Store it in the Application
Application("Start_Time") = CStr(Now) 'Store the date/time as a string
Application("Visit_Count") = 0 'Set counter variable to zero

End Sub
...
</SCRIPT>
```

#### (1) 遍历Contents集合的代码

为了遍历Contents集合，可使用一个For Each . . . Next结构。集合中的每一项可以是一个简单的Variant类型变量、一个Variant数组或者一个对象的引用。因为需要对每种类型的值进行不同的处理，所以就不得不对每一个进行检查来判别其类型。

在VBScript中可使用VarType函数完成这个工作。这里使用IsObject和IsArray函数代替：

```
For Each objItem in Application.Contents
If IsObject(Application.Contents(objItem)) Then
Response.Write "Object reference: " & objItem & "<BR>"
ElseIf IsArray(Application.Contents(objItem)) Then
Response.Write "Array: " & objItem & " contents are:<BR>"
varArray = Application.Contents(objItem)

'Note: the following only works with a one-dimensional array
For intLoop = 0 To UBound(varArray)
Response.Write "&nbsp; Index(" & intLoop & ") = " & _
varArray(intLoop) & "<BR>"
Next

Else
Response.Write "Variable: " & objItem & " = " & _
& Application.Contents(objItem) & "<BR>"

End If
Next
```

注意程序如何从Application对象检索该数组。将其分配给一个局部 (Variant) 变量，使用下面的语句：

```
varArray = Application.Contents(objItem)
```

使用UBound函数可以查找出数组的大小(元素的数量)，这个值可以作为遍历的终止条件：

```
For intLoop = 0 To UBound(varArray)
```

```
For intLoop = 0 To UBound(varArray)
    intNumberOfDimensions = UBound(varArray, 1)
    For intDimension = 0 To intNumberOfDimensions
        Response.Write "   Index(" & intLoop & ") = " _
            & varArray(intLoop, intDimension)
    Next
    Response.Write "<BR>"
Next
```

StaticObjects集合包含了所有在global.asa中使用<OBJECT>元素声明的对象引用。因为每个条目都是一个对象变量，可用简单些的代码对这个数组进行遍历。我们将输出对象的名字（在ID属性中原有的定义）：

```

For Each objItem in Application.StaticObjects
    If IsObject(Application.StaticObjects(objItem)) Then
        Response.Write "<!--OBJECT--> element: ID=" & objItem & "<!--BR-->"
    End if
Next

```

增加值到 Contents集合的方法，与在 global.asa网页的脚本代码中使用过的方法相同。示例网页允许把一个新的 Variant值增加到Application对象中，并已有建议的名字和值(可根据需要进行编辑)，如图3-15所示。

单击按钮，重新载入这个网页，把值增加到Application.Contents集合中，并且在列表中显示，如图3-16所示。

所有的按钮和其他 HTML 控件放置在示例网页中的一个窗体上。ACTION 设置了当前网页的路径，提交该窗体时，重新装入。METHOD 属性为“POST”，所以控件中的值出现在 Request.Form 集合中。在以前的章节中采用过这两种技术：

该窗体上的按钮都是普通的 HTML INPUT 控件，具有相同的标题 (三个空格) 但名字不同。例如，创建第一个按钮 (把值增加到 Application 对象中) 的代码是：

重新载入该网页时，检查 Request.Form 集合，判定单击的是哪个 SUBMIT 按钮，并进行相



应的处理。如果是增加一个值到 Application对象的按钮(该按钮在HTML的<INPUT>元素中被命名为cmdAdd),使用下面的程序段:

```
If Len(Request.Form("cmdAdd")) Then
    strVarName = Request.Form("txtVarName")
    strVarValue = Request.Form("txtVarValue")
    Application.Lock
    Application(strVarName) = strVarValue
    Application.Unlock
End If
```

注意程序如何使用 Application.Lock和Application.Unlock方法,确保这些值不会因两个用户并发地访问而产生混乱。如果只是对一个特定的值进行设置,一般不可能发生这种情况。但一直使用Lock和Unlock方法是明智的。

### 3. 从Contents集合中删除值

在例子网页的底部有两个按钮,如图 3-17所示。这两个按钮允许从 Application.Contents集合中删除值。第一个按钮从集合中删除单个的指定值,下拉列表框显示的是 Contents集合值的名字列表(记住,不能从 StaticObjects集合中删除值,因为它们是静态的)。

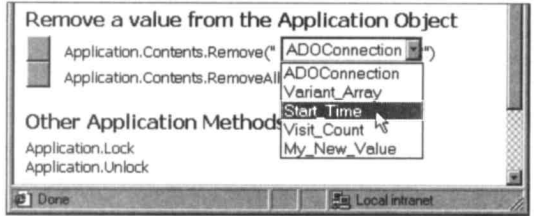


图3-17 显示在网页底部的两个按钮

通过遍历 Contents集合(如前面我们所做的)执行ASP网页时,创建该列表。但是,我们仅收集每项的名字并把它们放到 <SELECT>列表元素内的<OPTION>元素中:

```
...
<SELECT NAME="lstRemove" SIZE="1">
<%
For Each objItem in Application.Contents
    Response.Write "<OPTION>" & objItem & "</OPTION>"
Next
%>
</SELECT>
...
```

该ASP代码执行以后,在浏览器中看到的结果是:

```
<SELECT NAME="lstRemove" SIZE="1">
<OPTION>ADOConnection</OPTION>
<OPTION>Variant_Array</OPTION>
<OPTION>Start_Time</OPTION>
<OPTION>Visit_Count</OPTION>
<OPTION>My_New_Value</OPTION>
</SELECT>
```

#### (1) 删除单个的值

当单击按钮删除单个值时,该窗体再次提交给相同的网页,但是这一次将查找名为 cmdRemoveThis的SUBMIT按钮。如果存在(即单击了这个按钮),则使用列表框的值,调用 Application.Contents集合的Remove方法:

```
If Len(Request.Form("cmdRemoveThis")) Then
    strToRemove = Request.Form("lstRemove")
    Application.Lock
    Application.Contents.Remove(strToRemove)
    Application.Unlock
End If
```

注意这是 Contents集合的一个方法，而不是 Application对象的。语法是 Application.Contents.Remove，而不是 Application.Remove。

从Contents集合中删除Start\_Time值的结果如图3-18所示。

#### (2) 删除所有的值

如果单击三个SUBMIT类型按钮中的最后一个(如图3-18所示)，该网页中的代码将检测到单击的按钮为cmdRemoveAll，将执行Application.Contents集合的RemoveAll方法：

```
If Len(Request.Form("cmdRemoveAll")) Then  
    Application.Lock  
    Application.Contents.RemoveAll  
    Application.Unlock  
End If
```

再次提醒，这是 Contents集合的一个方法，而不是 Application。语法是 Application.Contents.RemoveAll，而不是 Application.RemoveAll。

图3-19所示的是从 Contents集合中删除所有值的结果(记住在运行时间不能从 StaticObjects集合删除项)。

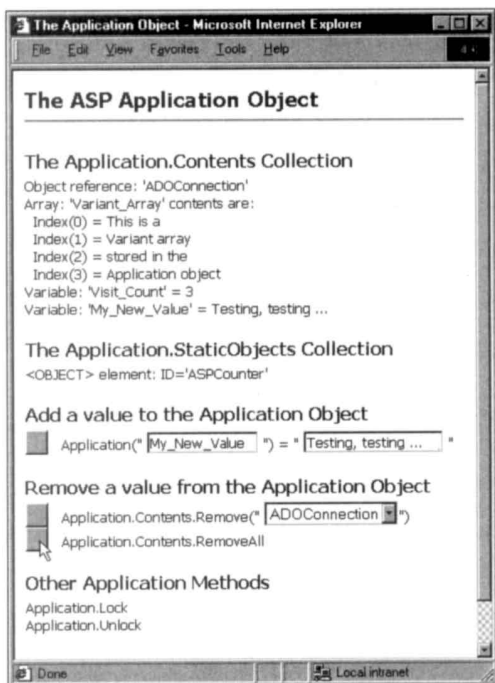


图3-18 删除Start\_Time值后的屏幕



图3-19 删除Content集合中所有值的屏幕

### 3.3.5 活动中的ASP Session对象

示例网页的第二个示例页面 show\_session.asp，示范了如何使用 Session对象。可在 Chapter03子目录中的开始菜单页面(Default.asp)中打开它。



### 1. 显示和更新Session集合

Session对象示例页面看起来与刚刚使用过的 Application对象示例页面相似。它遍历Session对象的Contents和StaticObjects集合，显示其名字和(可能的话)相应的值。如果把这些值与Application对象页面进行比较，将会看到不同之处。

这里还能够看到客户端 IP地址的一些其他值。这是当会话启动时 global.asa中的代码从Request.ServerVariables集合中得到的。这个页面还显示四个会话属性的值，如图 3-20所示。



图3-20 Session对象显示属性的屏幕

下面是例子中使用的 global.asa文件的相关段落，它把缺省值增加到图 3-20所示的屏幕上所看到的会话中：

```
<!-- declare instance of the ASPContentLink component with
session-level scope -->
<OBJECT ID="ASPContentLink" RUNAT="Server" SCOPE="Session"
PROGID="MSWC.NextLink">
</OBJECT>

<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
...
...
Sub Session_onStart()
```

End Sub

## 2. 终止一个用户会话

```
If Len(Request.Form("cmdAbandon")) Then
    Response.Clear
    Response.Redirect "abandon.asp"
    Response.End
End If
```

&lt;% Session.Abandon %&gt;

```
<FORM ACTION="<% = Request.ServerVariables("HTTP_REFERER") %>" METHOD="POST">  
    <P><DIV CLASS="subhead">Your Session Has Been Terminated</DIV>  
  
A new <B>Session</B> will be started when you load another<br>  
ASP page. It will contain any values that are defined in<br>  
the <B>global.asa</B> file for this application.<P>  
  
<INPUT TYPE="SUBMIT" NAME="cmdOK" VALUE="%&nbsp;%&nbsp;%&nbsp%;">
```

```
&nbsp;Return to the previous page<P>  
</FORM>
```

结果如图3-21所示。这时，当前的用户会话已经被终止（放弃），并且该用户不能引用原先的Session集合或属性的内容。

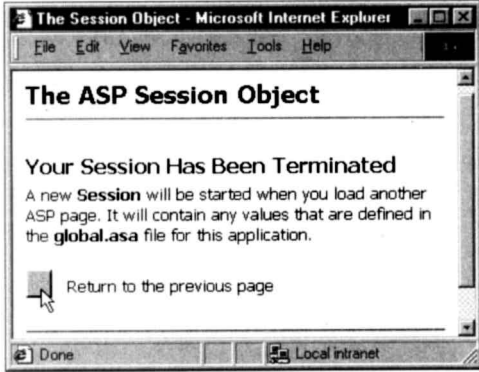


图3-21 终止一个用户会话后的屏幕

然而，记住这一切是在调用 Abandon方法的网页已经执行完毕后发生的。调用 Abandon方法的网页执行时，甚至在该方法的调用已经完成以后，仍能够从 Session对象中获得用户的会话内容。当此网页结束时，会话才结束。

当然，当返回到显示会话内容的 Session对象示例网页时，将启动一个新的 ASP会话。它将有一个不同的 Start\_Time值和通过执行 global.asa中的代码创建的其他缺省的会话值，如图 3-22所示。

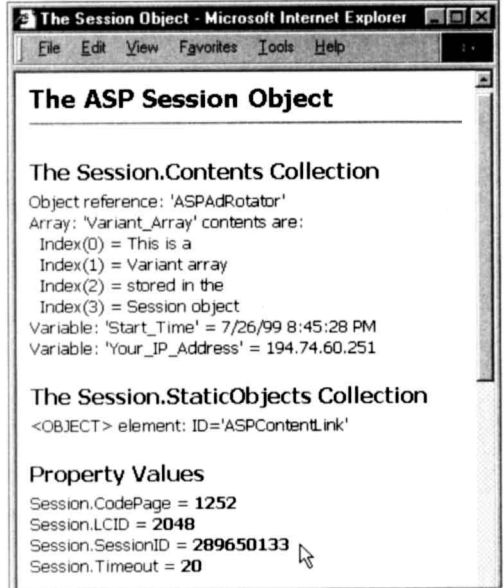


图3-22 重新建立一个用户会话后的屏幕

注意，Session.SessionID属性值没有改变。ASP试图重新分配相同的会话 ID，因此不能依靠该值来判定一个新的会话已经启动。

### 3.4 小结

本章介绍了两个ASP内置对象：Application和Session对象。这些对象引入了ASP的应用程序和用户会话的概念（两者都是特定的术语，并且不是通常谈话时的意义）。ASP应用程序允许分配专门的属性给页面集，定义了IIS和ASP如何管理这些网页及所使用的其他组件。

然而，使用ASP应用程序和会话的主要原因是需要自动地获得状态。换句话说，存储信息和变量引用的能力，要么对用户装载的所有网页是全局和可用的（即在一个应用程序中），要么仅仅对一个指定的用户的所有网页是可行的（在一个会话中）。这使建立Web应用程序变得非常简单（即应用程序在Web上工作，但能够像传统的编译的应用程序一样能完成指定的任务）。

本章通过一些示例页面，详细介绍如何使用ASP的Application和Session对象。这些页面示范了这两个对象可用的所有技术。

本章的内容为：

- Web应用程序是什么，以及它们如何与 ASP Application对象相联系。
- 用户会话是什么，以及它们如何与 ASP Session对象相联系。
- ASP如何自动地创建和管理应用程序和会话。
- Application和Session对象提供的功能。
- 如何把Application和Session对象放入ASP网页中。

下一章研究另一个ASP内置对象：Server对象，你将会发现ASP 3.0的许多令人兴奋的新特性。