

## 第2章 请求和响应的处理

上一章介绍了安装设置的一些基本问题和 ASP 的使用，以及其所提供的内置对象。本章将进一步研究两个最常用的对象。在浏览器 (或其他用户代理) 和 Web 服务器之间，请求与响应中发生的信息交流可以通过 ASP 中的两个内置对象来进行访问和管理，这两个对象称为 Request 和 Response 对象。

在 ASP 页中所要进行的工作几乎都要访问这两个对象，使用这两个对象的方式将影响页面的效率及可靠性。当然，它们的主要用途是访问用户发回到服务器的值，即从 HTML 页的 <FORM> 段获得或附在 URL 后面作为查询字符串，并创建合适的输出返回给用户，且它们可以共享很多相同的因素。例如两个对象都可以使用存储在客户端计算机上的 cookie。

因此，我们把本章分成两个独立的部分 (每个对象一部分)，并且首先从客户端与服务器之间的信息交流入手，然后再研究每一个对象。这将更有助于理解它们之间的关系及其重要性。

因此，本章研究的内容是：

- 客户端与服务器如何交流以传递 Web 页或其他资源。
- Request 和 Response 对象的细节，以及它们之间的共同点。
- 如何通过一个窗体和查询字符串访问相应的值。
- 如何读入或创建 cookie 并存放在客户端的计算机上。
- 服务器的变量是什么？如何访问和修改 HTTP 报头。
- 说明其他相关条目的变化，如客户的证书使用。

首先看一下客户浏览器 (或“用户代理”) 和 ASP 的 Web 服务器之间交流的情况。

### 2.1 客户端和服务器的交流

当浏览器或其他的用户代理访问 Web 站点请求页面时，在客户服务器和 Web 服务器之间就产生了一个对话，我们将仔细研究这个问题，因为理解这个问题有助于掌握使用 ASP 的 Request 和 Response 对象的基本知识，进而才能进一步掌握 ASP 的更多知识。

为节省篇幅，在本章及本书后面的内容中使用“浏览器”(browser) 一词。但需要记住的是，能够访问 Web 页面的应用程序绝不仅只有浏览器，有许多特殊的应用程序从站点上下载网页，如为那些视力有缺陷用户设计的特殊的客户端程序或者是用通常的浏览器有其他困难的人。最显而易见的例子是搜索引擎用来访问 Web 上的站点的 robot。全面考虑这些因素，包括普通的 Web 浏览器，准确的词汇应该是用户代理 (user agent)。

#### 页面请求的对话

当一个浏览器向 Web 站点提出页面请求时，显然必须告诉服务器，其请求的是哪一个页面。首先要做的是通过域名与服务器建立连接，然后提供所请求页面的全路径和名称。为什么要全路径和名称？Web 是一个无国界的环境，所以必须创造一个会话标识每个客户 (将在下

一章介绍 ASP 如何做到这点)。

这就意味着每当服务器完成向客户发送页面后，服务器就彻底忘记了这个客户。因此，当客户请求下一个页面时，与一个新的访问者是完全相同的。服务器无法记住这个客户，相应的，也就无法判断它们上次请求的页面是哪一个。因此，不能使用相对路径来提供一个页面，即使页面包含一个相对的连接也不行，例如：

```
<A HREF="Download.asp">Next Page</A>
```

浏览器将自动建立完整的新页面的 URL，方法是使用当前页的域和路径；或使用页面 <HEAD> 段中的 <BASE> 元素，告诉浏览器一个页面中所有链接的基 URL 是什么。例如：

```
<BASE HREF="http://www.wrox.com/Store">
```

当把鼠标指向一个页面的链接时，可在浏览器的状态栏中看到如图 2-1 所示的情况。当前页面的路径和当前域名或基域名已经与请求的页面名结合在一起了。

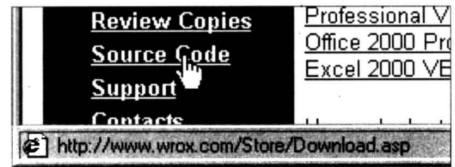


图2-1 状态栏中显示的当前页面路径

### 1. 客户请求的细节

所请求页面的全路径和名称的组合是浏览器请求页面时发往服务器的唯一信息。浏览器的请求也能包含浏览器宿主的信息和客户端运行的操作系统。实际的信息内容将随着浏览器的不同而有相应的变化，只有很少一部分能够由其他的应用程序如搜索引擎 robot 提供。为了更清楚地了解该信息，下面是从 IE 5.0 发出的一个对页面 <http://www.wrox.com/Store/Download.asp> 的请求信息：

```
7/8/99 10:27:16 Sent GET /Store/Download.asp HTTP/1.1
Accept: application/msword, application/vnd.ms-excel, application/vnd.ms-
powerpoint, image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-
comet, */*
Accept-Language: en-us
Encoding: gzip, deflate
Referer: http://www.wrox.com/main_menu.asp
Cookie: VisitCount=2&LASTDATE=6%2F4%2F99+10%3A10%3A13+AM
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98)
Host: 212.250.238.67
Connection: Keep-Alive
```

从中可以看出，这些信息中有关于用户代理和用户连接的细节（如缺省的语言），也有能够接受的文件或应用程序的类型列表，这些都是 MIME 类型的，在后面将会见到更多。浏览器能够接受几种图像文件及多种 Microsoft Office 文件类型。“标准”的文件类型，如 text/html 和 text/text 没有列在其中。文件列表中 \*/\* 表示可向浏览器发回任何类型的文件，由浏览器解释或通过一个插件 (plug-in) 应用程序来进行解释。

cookie：条目包含的 cookie 存放在客户端的计算机上，并仅对该域有效。假如请求是点击链接的结果，而不是在浏览器的地址栏直接输入 URL，则 Referer：条目被显示，它包含了链接页面的完整的 URL。

Host：条目包含客户计算机的 IP 地址或名称。然而，这还不足以准确辨别客户机。因为它们通过 ISP 连接时，IP 地址是动态分配的，或者通过一个代理服务器连接时，IP 地址是代理机的而不是实际的客户机的。

### 2. 服务器响应的细节信息

为响应上述的请求，并对匿名的浏览器（即用户不必提供用户名和访问口令）提供请求的页

面，下面的内容是从服务器发往客户机的：

```
7/8/99 10:27:16 Received HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Connection: Keep-Alive
Date: Thu, 8 Jul 1999 10:27:16 GMT
Content-Type: text/html
Accept-Ranges: bytes
Content-Length: 2946
Last-Modified: Thu, 8 Jul 1999 10:27:16 GMT
Cookie: VisitCount=3&LASTDATE=7%2F8%2F99+10%3A27%3A16+AM
<HTML>
... rest of page ...
</HTML>
```

可以看出服务器向客户端说明自己所用的软件及版本，第一行表明所使用的是HTTP协议，及返回码的状态。信息“200 OK”表示请求被接受并得到了满足。后面的信息是被返回的页面的细节，包括MIME类型(Content-Length:)、大小(字节)、最近更改的时间，和返回客户端存储的cookie。响应中的其他信息是页面内容的信息流。

在某些情况下，服务器响应一个请求后，不能返回一个页面，也许因为页面不存在，或者客户没有相应权限来访问它。我们将在本书后续章节讨论安全问题。现在，对于请求页面不存在的情况(例如用户在浏览器的地址栏输入了错误的URL)，返回的信息开头为：

```
7/8/99 14:27:16 Received HTTP/1.1 404 Not Found
Server: Microsoft-IIS/5.0
...
```

这里，状态码和信息表明客户请求的页面无法找到。浏览器可使用这个信息向用户显示相应的信息(这种情况在IE 5.0中不显示服务器的响应信息，而显示相应“帮助性”的错误提示页面)，也可显示服务器创建的缺省页面(依赖于服务器的设置)。

至此，我们已经看到了客户机与服务器交互作用中的一些细节，下面介绍 ASP对象Request和Response如何把这些转换成相应的值。

## 2.2 Request和Response对象

在ASP中能够应用客户请求和服务器响应的细节是通过ASP内置的Request和Response对象来实现的。

- Request对象：为脚本提供了当客户端请求一个页面或者传递一个窗体时，客户端提供的全部信息。这包括能指明浏览器和用户的HTTP变量，在这个域名下存放在浏览器中的cookie，任何作为查询字符串而附于URL后面的字符串或页面的<FORM>段中的HTML控件的值。也提供使用Secure Socket Layer (SSL)或其他加密通信协议的授权访问，及有助于对连接进行管理的属性。
- Response对象：用来访问服务器端所创建的并发回到客户端的响应信息。为脚本提供HTTP变量，指明服务器和服务器的功能和关于发回浏览器的内容的信息以及任何将为此域而存放在浏览器里新的cookie。它也提供了一系列的方法用来创建输出，例如无处不在的Response.Write方法。

在这一节中，首先概述Request和Response对象的成员，然后继续探讨如何使用它们完成创建任务，以及如何使用每个对象的成员的细节。

### 2.2.1 Request对象成员的概述

本节将给出Request对象的所有成员的主要说明。

1. Request对象的集合

Request对象提供了5个集合，可以用来访问客户端对 Web服务器请求的各类信息，这些集合如表2-1所示。

表2-1 Request对象的集合及说明

集合名称	说 明
ClientCertificate	当客户端访问一个页面或其他资源时，用来向服务器表明身份的客户证书的所 有字段或条目的数值集合，每个成员均是只读
Cookies	根据用户的请求，用户系统发出的所有 cookie的值的集合，这些Cookie仅对相 应的域有效，每个成员均为只读
Form	METHOD的属性值为POST时，所有作为请求提交的 <FORM>段中的HTML控 件单元的值集合，每个成员均为只读
QueryString	依附于用户请求的URL后面的名称 / 数值对或者作为请求提交的且 METHOD 属性值为 GET(或者省略其属性)的，或<FORM>中所有HTML控件单元的值，每 个成员均为只读
ServerVariables	随同客户端请求发出的HTTP 报头值，以及 Web服务器的几种环境变量的值的 集合，每个成员均为只读

2. Request对象的属性

Request对象唯一的属性及说明如表 2-2所示，它提供关于用户请求的字节数量的信息，它很少用于ASP页，我们通常关注指定值而不是整个请求字符串。

表2-2 Request对象的属性及说明

属 性	说 明
TotalBytes	只读，返回由客户端发出的请求的整个字节数量

3. Request对象的方法

Request对象唯一的方法及说明如表 2-3所示，它允许访问从一个 <FORM>段中传递给服务器的用户请求部分的完整内容。

表2-3 Request对象的方法及说明

方 法	说 明
BinaryRead(count)	当数据作为POST请求的一部分发往服务器时，从客户请求中获得 count字节的数 据，返回一个 Variant数组(或者SafeArray)。如果ASP代码已经引用了Request.Form 集合，这个方法就不能用。同样，如果用了 BinaryRead方法，就不能访问 Request.Form集合

2.2.2 Response对象成员概述

这一节概要介绍所有的Response对象成员，及每个成员的主要说明。

1. Response对象的集合

Response对象只有一个集合，如表 2-4所示，该集合设置希望放置在客户系统上的 cookie的值，它直接等同于Request.Cookies集合。

表2-4 Response对象的集合及说明

集合名称	说 明
Cookies	在当前响应中，发回客户端的所有 cookie的值，这个集合为只写

2. Response对象的属性

Response对象也提供一系列的属性，可以读取(多数情况下)和修改，使响应能够适应请求。这些由服务器设置，我们不需要设置它们。需要注意的是，当设置某些属性时，使用的语法可能与通常所使用的有一定的差异。这些属性如表 2-5所示。

表2-5 Response对象的属性及说明

属 性	说 明
Buffer=True False	读/写，布尔型，表明由一个ASP页所创建的输出是否一直存放在 IIS缓冲区，直到当前页面的所有服务器脚本处理完毕或 Flush、End方法被调用。在任何输出(包括HTTP报头信息)送往IIS之前这个属性必须设置。因此在 .asp文件中，这个设置应该在<%@LANGUAGE=...%>语句后面的第一行。ASP 3.0缺省设置缓冲为开(True)，而在早期版本中缺省为关(False)
CacheControl "setting"	读/写，字符型，设置这个属性为“ Public ”允许代理服务器缓存页面，如为“ Private ”则禁止代理服务器缓存的发生。
Charset="value"	读/写，字符型，在由服务器为每个响应创建的 HTTP Content-Type 报头中附上所用的字符集名称(例如：ISO-LATIN-7)
Content Type ="MIME-type"	读/写，字符型，指明响应的 HTTP内容类型，标准的MIME类型(例如“ text/xml ”或者“ Image/gif ”)。假如省略，表示使用MIME类型“ text/html ”，内容类型告诉浏览器所期望内容的类型
Expires minutes	读/写，数值型，指明页面有效的以分钟计算的时间长度，假如用户请求其有效期满之前的相同页面，将直接读取显示缓冲中的内容，这个有效期间过后，页面将不再保留在私有(用户)或公用(代理服务器)缓冲中
Expires Absolute #date[time]#	读/写，日期 / 时间型，指明当一个页面过期和不再有效时的绝对日期和时间
IsClientConnected	只读，布尔型，返回客户是否仍然连接和下载页面的状态标志。在当前的页面已执行完毕之前，假如一个客户转移到另一个页面，这个标志可用来中止处理(使用Response.End方法)
PICS("PICS-Label-string"	只写，字符型，创建一个PICS 报头并将之加到响应中的HTTP 报头中，PICS 报头定义页面内容中的词汇等级，如暴力、性、不良语言等
Status="Code message"	读/写，字符型，指明发回客户的响应的 HTTP 报头中表明错误或页面处理是否成功的状态值和信息。例如“ 200 OK ”和“ 404 Not Found ”

3. Response对象的方法

最后，Response对象提供一系列的方法，如表 2-6所示，允许直接处理为返给客户端而创建的页面内容。

表2-6 Response对象的方法及说明

方 法	说 明
AddHeader("name","content")	通过使用name和Content值，创建一个定制的HTTP报头，并增加到响应之中。不能替换现有的相同名称的报头。一旦已经增加了一个报头就不能被删除。这个方法必须在任何页面内容(即text和HTML)被发往客户端前使用
AppendToLog("string")	当使用“ W3C Extended Log File Format ”文件格式时，对于用户请求的 Web 服务器的日志文件增加一个条目。至少要求在包含页面的站点的“ Extended Properties ”页中选择“ URI Stem ”
BinaryWrite(SafeArray)	在当前的HTTP输出流中写入 Variant类型的SafeArray，而不过任何字符转换。对于写入非字符串的信息，例如定制的应用程序请求的二进制数据或组成图像文件的二进制字节，是非常有用的



(续)

方 法	说 明
Clear()	当Response .Buffer为True时, 从IIS响应缓冲中删除现存的缓冲页面内容。但不删除HTTP响应的报头, 可用来放弃部分完成的页面
End()	让ASP结束处理页面的脚本, 并返回当前已创建的内容, 然后放弃页面的任何进一步处理
Flush()	发送IIS缓冲中所有当前缓冲页给客户端。当 Response.buffer为True时, 可以用来发送较大页面的部分内容给个别的用户
Redirect("url")	通过在响应中发送一个 “ 302 Object Moved ” HTTP报头, 指示浏览器根据字符串url下载相应地址的页面
Write("string")	在当前的HTTP响应信息流和IIS缓冲区写入指定的字符, 使之成为返回页面的一部分

在本书中源代码的例子里, 读者会发现一系列的页面, 演示如何使用这些由Request和Response对象提供的属性、方法和集合。在从 Wrox Web站点下载的Chapter02目录下, 有本章其余部分的示例页面。

## 2.3 使用Form和QueryString集合

当用户填写页面<FORM>内容时所提供的全部值, 或在浏览器地址栏输入在 URL后的值, 通过Form和QueryString集合为ASP脚本所用。这是在ASP代码中访问值的一种简单方法。

### 2.3.1 访问ASP集合的一般技术

大多数ASP集合与在VB中见到的普通集合相差不多。实际上, 它们是值的数组, 但能通过使用一个文本字符串键(对大小写不敏感)以及一个整型索引进行访问。因此, 假如客户端Web页面包含的<FORM>如下:

```
<FORM ACTION="show_request.asp" METHOD="POST">
  FirstName: <INPUT TYPE="TEXT" NAME="FirstName">
  LastName: <INPUT TYPE="TEXT" NAME="LastName">
  <INPUT TYPE="SUBMIT" VALUE="Send">
</FORM>
```

可通过访问ASP的Form集合来访问其控件内的值:

```
strFirstName = Request.Form("FirstName")
strLastName = Request.Form("LastName")
```

也可使用窗体中控件的整型索引, 索引的范围从在 HTML中第一个定义的控制开始, 然后根据定义的顺序排序:

```
strFirstName = Request.Form(1)
strLastName = Request.Form(2)
```

然而, 后面的这种以整型为索引的技术不推荐使用, 因为一旦 HTML中的控件发生了变化, 或者插入一个新的控件, 则 ASP代码将得到错误的值。进一步而言, 对于阅读代码的人来讲, 极容易混淆。

#### 1. 访问集合的全部值

可以通过引用集合把整个Form上的一系列值变成单个的字符变量, 且不用提供键或索引。

```
strAllFormContent = Request.Form
```

假如文本框包含值Priscilla和Descartes, 则Request.Form语句将返回下列字符:

```
FirstName=Priscilla&LastName=Descartes
```

注意，提供的值是以名称 / 值对的形式出现的（即控件名称 = 控件值），并且每一对名称 / 值相互之间是用符号“&”相分隔的。假如打算把窗体中的内容传递单独的希望得到值的标准格式的可执行应用程序或 DLL，这个技术是很有用的。然而，一般说来，都是通过以窗体中控件名称为文本键来访问集合中的内容。

## 2. 遍历一个ASP集合

有两种方式遍历一个 ASP 集合中的所有成员，方式与普通 VB 集合的基本相同。每个集合提供一个 Count 属性，返回的是集合中条目数量。可通过使用一个整型索引使用 count 属性来遍历集合。

```
For intLoop = 1 To Request.Form.Count
    Response.Write Request.Form(intLoop) & "<BR>"
Next
```

假如先前的窗体包含 Priscilla 和 Descartes 值的两个文本框，将得到如下结果：

```
Priscilla
Descartes
```

然而，更好的方法是使用 For Each...Next 结构。

```
For Each objItem In Request.Form
    Response.Write objItem & " = " & Request.Form(objItem) & "<BR>"
Next
```

这带来的好处是既可以访问控件的名称又可访问其值。上述代码将得到如下结果：

```
FirstName = Priscilla
LastName = Descartes
```

注意，一些浏览器返回到 ASP 的 <FORM> 值可能与页面上显示的顺序不尽相同。

## 3. 集合成员的多值性

在某些情况下，ASP 集合中的各个成员可能不只一个值，这种情况发生在 HTML 定义中有几个控件有相同 Name 属性时。例如：

```
<FORM ACTION="show_request.asp" METHOD="POST">
  <INPUT TYPE="TEXT" NAME="OtherHobby">
  <INPUT TYPE="TEXT" NAME="OtherHobby">
  <INPUT TYPE="TEXT" NAME="OtherHobby">
  <INPUT TYPE="SUBMIT" VALUE="Send">
</FORM>
```

在 Form 集合中，将为键“OtherHobby”创建一个条目。然而，它将包括从三个文本框中得到的值。假如在提交时，用户留下了一个或多个为空，则返回的值为空字符串。假如用户在第一和第三个文本框分别输入 Gardening 和 Mountaineering，第二个文本框为空，在我们的 ASP 代码中访问 Request.Form(“OtherHobby”)，将返回字符串：

```
Gardening, , Mountaineering
```

为了能够在这种情况下，访问单个值，可以用复杂一些的代码：

```
For Each objItem In Request.Form
    If Request.Form(objItem).Count > 1 Then 'More than one value in this item
        Response.Write objItem & " :<BR>"
```

```

For intLoop = 1 To Request.Form(objItem).Count
    Response.Write "Subkey " & intLoop & " value = " & _
        & Request.Form(objItem)(intLoop) & "<BR>"
Next

Else

    Response.Write objItem & " = " & Request.Form(objItem) & "<BR>"

End If

Next

```

对于前面的包含三个OtherHobby控件的窗体实例，这将返回：

```

OtherHobby:
Subkey 1 value = Gardening
Subkey 2 value =
Subkey 3 value = Mountaineering

```

然而，由于很少给多个文本框相同的名字，因此这种技术很少用到。

#### (1) HTML中的单选或选项按钮控件

在HTML中，需要给几个控件相同的Name属性的情况是单选(或选项)按钮，例如：

```

<FORM ACTION="show_request.asp" METHOD="POST">
  I live in:
  <INPUT TYPE="RADIO" NAME="Country" VALUE="AM"> America<BR>
  <INPUT TYPE="RADIO" NAME="Country" VALUE="EU"> Europe<BR>
  <INPUT TYPE="RADIO" NAME="Country" VALUE="AS"> Asia<BR>
  <INPUT TYPE="RADIO" NAME="Country" VALUE="*"> Elsewhere<P>
  <INPUT TYPE="SUBMIT" VALUE="Send">
</FORM>

```

因为用户只能选择多项中的一个(这就是给它们相同的名字的原因)，将仅得到一个返回值，浏览器只能发送所选择控件的值。因此，假如这个窗体的用户已经选择了“Europe”，将得到这个条目，通过遍历Form集得到其值：

```
Country = EU
```

由于为每个控件提供了不同的VALUE属性，反映了每个条目所对应的国家或地区的名称。假如省略了VALUE属性，浏览器将返回的值为“on”，因此将得到：

```
Country = on
```

这是不经常用到的，因此一般对使用相同名称的单选控件使用VALUE属性。

#### (2) HTML复选框控件

当一个窗体中HTML源码包含一个复选框控件时，一般都给定唯一的名称，例如：

```

<FORM ACTION="show_request.asp" METHOD="POST">
  I enjoy:
  <INPUT TYPE="CHECKBOX" NAME="Reading" CHECKED> Reading &nbsp;
  <INPUT TYPE="CHECKBOX" NAME="Eating"> Eating &nbsp;
  <INPUT TYPE="CHECKBOX" NAME="Sleeping"> Sleeping
  <INPUT TYPE="SUBMIT" VALUE="Send">
</FORM>

```

在这种情况下，提交窗体时，假如仅是第一和第三个复选框被选中（加标记），遍历Form集合时，会得到下列值：

```
Reading = on
Sleeping = on
```

然而，假如为每个复选框提供一个值，把这个值发往服务器代替字符串“on”。例如窗体如下：

```
<FORM ACTION="show_request.asp" METHOD="POST">
```



```
I enjoy:
<INPUT TYPE="CHECKBOX" VALUE="Hobby025" NAME="Hobby" CHECKED> Swimming &nbsp;
<INPUT TYPE="CHECKBOX" VALUE="Hobby003" NAME="Hobby" CHECKED> Reading &nbsp;
<INPUT TYPE="CHECKBOX" VALUE="Hobby068" NAME="Hobby"> Eating &nbsp;
<INPUT TYPE="CHECKBOX" VALUE="Hobby010" NAME="Hobby"> Sleeping
<INPUT TYPE="SUBMIT" VALUE="Send">
</FORM>
```

如果除第三个复选框外，全部提交，在 Request.Form 集合会产生下列结果：

```
Hobby = Hobby025, Hobby003, Hobby010
```

假如编写更复杂一些集合遍历代码，如先前所述（单独显示每个子键），就得到这样结果：

```
Hobby:
Subkey 1 value = Hobby025
Subkey 2 value = Hobby003
Subkey 3 value = Hobby010
```

需要注意的是两种情况，没有选中的控件根本不返回任何值。在第一种情况的结果里，没有欺骗性的逗号，第二种情况也没有空值。这与上述的使用文本框的相当的测试的结果不一样。使用文本框时，每个文本框都返回一个值，即使是一个空字符串。这是浏览器造成这样的结果。因此在 ASP 代码中访问集合时，要注意这个问题。

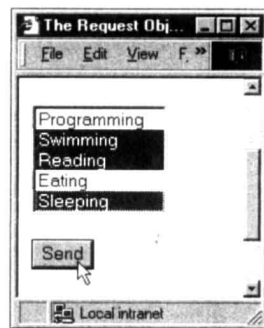
上述情况一个棘手的副作用是使用复选框时，复选框值的索引与在原始的 HTML 中控件的位置没有任何联系，在上述的例子中第四个复选框的子键数为 3，因为当窗体提交时，第二个控件没有选中。

### (3) HTML 列表控件

HTML 中的 <SELECT> 标记用来产生标准的下拉列表框，其值以一种有趣的混合方式表示。下列的窗体创建了包含 5 个值可供用户选择，由于包含了 MULTIPLE 属性，因此可以通过选择时按下 Shift 或 Ctrl 键，选择不仅一个的条目。

```
<FORM ACTION="show_request.asp" METHOD="POST">
<SELECT NAME="Hobby" SIZE="5" MULTIPLE>
<OPTION VALUE="Hobby001">Programming
<OPTION VALUE="Hobby025">Swimming
<OPTION VALUE="Hobby003">Reading
<OPTION VALUE="Hobby068">Eating
<OPTION VALUE="Hobby010">Sleeping
</SELECT><P>
<INPUT TYPE="SUBMIT" VALUE="Send">
</FORM>
```

图 2-2 所示为该页面，显示的是选中了三个条目。



这种特殊的情况返回的是在 Form 集合中单个条目，它包含选择的值（单个的 <OPTION> 标记中指定的 VALUE 属性），用逗号分隔：

图 2-2 HTML 列表控件屏幕

```
Hobby = Hobby025, Hobby003, Hobby010
```

假如使用更加复杂一些的集合遍历代码（单独显示每个子键），将得到：

```
Hobby:
Subkey 1 value = Hobby025
Subkey 2 value = Hobby003
Subkey 3 value = Hobby010
```

这与上述的相同名称的复选框的情况相同。事实上可以认为一个 SELECT 列表是一列复选框的列表供选择（不是选中）相应的条目。

然而，列表框也有指定的值，假如在 <OPTION> 标记中设置 VALUE 属性，将得到的是选择的选项的文本内容，Request.Form 集将包含这样一个项目：

```
Hobby = Swimming, Reading, Sleeping
```

并且, 同样, 复杂一些的集合遍历代码将返回如下结果:

```
Hobby:
Subkey 1 value = Swimming
Subkey 2 value = Reading
Subkey 3 value = Sleeping
```

当然, 假如单个项目被选择, 且在<OPTION>中提供了VALUE属性, 得到结果包含的仅是:

```
Hobby=Hobby025
```

如果没有提供VALUE属性, 得到:

```
Hobby=Swimming
```

这允许既可以缺省(即无VALUE)显示选项文本, 也可做相应的改变。后一种情况在某些情况下是极为有用的, 如要显示(一个说明的字符串)和传递一个完全不同的内容(如用一个短码代表一个说明性的字符串)。

#### (4) HTML提交和图像控件

复选框和单选框是布尔型控件的例子, 选中或选择返回的为“on”, 不像文本框和大多数其他的HTML控件, 浏览器不包含没有选中或没有选择的控件的值。

还有另外一种常用的布尔型控件, 称为HTML按钮。如<INPUT TYPE = “SUBMIT”>、<INPUT TYPE = “RESET”>、<INPUT TYPE= “IMAGE”>、<INPUT TYPE = “BUTTON”>和<BUTTON>...</BUTTON>类型。

BUTTON类型的控件不返回任何值, 因其对窗体没有直接的影响。即使使用用来调用窗体的Submit方法, 浏览器在任何请求中将不包含BUTTON类型控件的值。同样, 一个<INPUTTYPE = “RESET”>按钮的值也决不会发往服务器。

然而, 输入按钮控件SUBMIT和IMAGE类型实际提交窗体给服务器, 其VALUE属性包含窗体的其他控件的值(只要在HTML定义中包含一个NAME属性)。例如, 这个窗体可能是向导类型Web应用程序的一部分, 允许用户一步步进行或取消进程:

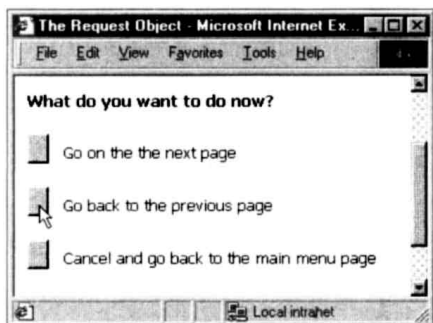
```
<FORM ACTION="show_request.asp" METHOD="POST">
  <INPUT TYPE="SUBMIT" NAME="btnSubmit" VALUE="Next">
  <INPUT TYPE="SUBMIT" NAME="btnSubmit" VALUE="Previous">
  <INPUT TYPE="SUBMIT" NAME="btnSubmit" VALUE="Cancel">
</FORM>
```

在一个窗体中, 可以包括多个SUBMIT按钮。在这种情况下, 应该给每一个按钮唯一的VALUE属性, 如上所示。当一个窗体被提交时, 遍历Request.Form集合的值, 将产生一个值, 这个值依赖于按下哪个按钮用于提交这个窗体。假如用户按下的“Previous”按钮, 将得到:

```
btnSubmit = Previous
```

因此, 可查询Request.Form集合来决定下一个显示的页面, 例如:

```
Select Case Request.Form("btnSubmit")
  Case "Next"
    Response.Redirect "page_3.asp"
  Case "Previous"
    Response.Redirect "page_1.asp"
  Case "Cancel"
    Response.Redirect "main_menu.asp"
End Select
```



此屏幕上的界面由下列代码产生：

在ASP页面中，接收到该数据后，可以检查按钮名称提供的值来判断按下的是哪个按钮。

[illegible]

访问一个ASP集合来下载一个值是费时的需计算资源的过程，因为这个操作包含了一系列对相关集合的搜索，这比访问一个局部变量要慢得多。因此，如果打算在页面中多次使用集合中的一个值，应该考虑将其存贮成为一个局部变量，例如：

```
strTitle = Request.Form("Title")
strFirstName = Request.Form("FirstName")
strLastName = Request.Form("LastName")

If Len(strTitle) Then strTitle = strTitle & " "

If strFirstName = "" Then
    strFullName = strTitle & " " & strLastName
ElseIf Len(strFirstName) = 1 Then
    strFullName = strTitle & strFirstName & ". " & strLastName
Else
```

```
strFullName = strTitle & strFirstName & " " & strLastName  
End If
```

#### (6) 搜索所有的Request集合

在某些情况下,可能知道一个值的键名将出现在 Request集合中,但不能准确地知道是哪个集合。例如,假如有几个页面(或一页面的不同段)发送一个值给同一个ASP脚本,它可能在Form或者QueryString集合中出现。

本章后面部分将研究Form和QueryString集合的差异。

要看下一个值为什么可能出现在不同的集合中,考虑一下这种情况:使用了 <A>超级链接元素请求一个页面。在这种情况下,增加一个值到请求的唯一方法是把它加到 URL上。然而,同样的值可能已出现在另一个页面的 <FORM>中,或同一页面不同部分:

```
...  
<FORM ACTION="process_page.asp" METHOD="POST">  
  <INPUT TYPE="SUBMIT" NAME="page" VALUE="Next">  
  <INPUT TYPE="SUBMIT" NAME="page" VALUE="Previous">  
  <INPUT TYPE="SUBMIT" NAME="page" VALUE="Help">  
</FORM>  
...  
...  
For help go to the <A HREF="process_page.asp?page=Help">Help Page</A>  
...
```

在这种情况下,按下窗体上的 Help按钮,将发送 Request.Form集合中一对名称/值“page=Help”。然而,按下<A>超级链接也可能发送名称/值“page=Help”,但是这次却是在QueryString集合里。为访问这个值,可使用ASP Request对象的一个特殊功能:

```
strPage = Request("page")
```

这将按序搜索全部的集合——QueryString、Form、Cookies、ClientCertificate、ServerVariables,直到发现第一个匹配值的名称。这样做比直接访问适当的集合效率低,并且是不安全的,除非能绝对保证这个值不会出现在另外一个集合中。

例如,可能希望搜集满足客户请求的Web服务器的名称,这通过出现在每个查询中的Request.ServerVariables集合中寻找“SRVER\_NAME”来实现。然而,假如任一其他的集合也包含名为“server\_name”的值(记住键名不区分大小写),当使用Request(“server\_name”)时,得到的是错误的结果。使用Request.ServerVariables(“server\_name”)句法,我们将很难进行错误追踪。

总而言之,使用“搜索全部集合”技术要格外小心,且只在没有其他技术能够提供你需要的结果时使用。

#### (7) 访问其他的集合

本章的这一节里,已经集中讨论了Form集合,这可能是使用得最多的一个。然而,所有这些技术同样适用于其他的对象。包括那些由Request对象提供的(即Form、QueryString、Cookies、ServerVariables和ClientCertificate)集合,及由Response对象提供的cookies(及将在下两章遇到的其他对象提供的集合)。

我们将简短了解一个值如何进入一个QueryString集合,及其优点和不足。然而,同时这两个Cookies集合有额外的功能,可以使使用cookie更加方便,下面讨论这个内容。

### 2.3.2 访问和更新Cookies集合

Cookies的值比ASP其他集合(例如Form和ServerVariables)的值要复杂得多。cookie是一小

块由浏览器存贮在客户端系统上的文本，且随同每次请求发往它们应用于的域中的服务器。

ASP使得应用cookie较为容易，可以从Request对象的Cookies集合中获得所有随同请求发出去的cookie的值，并可创建或修改cookie，通过Response对象的Cookies集合发回给用户。

cookie包含可用两种方式构造的信息，单值 cookie提供其值给代码是通过一个一般的类ASP集合。然而，集合的每个成员可能本身也是一个集合，包含这种信息的 cookie通常称为多值(multiple-Value)cookie。

创建一个单值的cookie较为简单，如下所示：

```
Response.Cookies("item-name") = "item-value"
```

创建一个多值的cookie，可以使用如下命令：

```
Response.Cookies("item-name")("sub-item-name") = "sub-item-value"
```

设置cookie应用的域及路径及其有效期，我们使用：

```
Response.Cookies("item-name").domain = "domain-url"
Response.Cookies("item-name").path = "virtual-path"
Response.Cookies("item-name").expires = #date#
```

通常，客户只在对创建cookie的目录中的页面提出请求时，才将cookie随请示发往服务器。通过指定path属性，可以指定站点中何处这个cookie是合法的，并且这个cookie将随请求发送。如果cookie随对整个站点的页面请求发送，设置path为“/”。

假如Expires属性没有设置，关闭当前的浏览器实例时，cookie将自动被消除。

注意，我们在向浏览器发送任何输出时，已经创建了 cookie，因为，这些cookie是页面HTTP报头的一部分。

在ASP 3.0中，缓冲的缺省状态是打开的，且没有输出被发送，除非使用Response.Flush指定做这个工作或者页面已到末端。这意味着创建cookie的代码可以在页面上的任何位置，直到任何输出“刷新”(flush)到客户端前，它都可以被执行。

要读现有的cookie，使用Request.Cookies集合。可以单独访问其中项目，方法类似于创建它们时使用的方法。

```
strSingleValue = Request.Cookies("item-name")
strSubItemValue = Request.Cookies("item-name")("sub-item-name")
```

注意Request.Cookies集合(和所有其他Request集合一样)是只读的。Response.Cookies集合是只写的，事实上可以访问这个集合中一系列cookie的名称，而不是它们的值。

遍历Cookies集合

为了使用Cookies集合更加方便，可使用名称为Haskeys的附加属性。假如访问的cookie本身也是个集合，即它是一个多值的cookie，这将返回True。使用Haskeys属性，可以遍历完整的Request.Cookies集合，从而获得所有cookie的列表及它们的值。

```
For Each objItem In Request.Cookies
    If Request.Cookies(objItem).HasKeys Then
        'Use another For Each to iterate all subkeys
        For Each objItemKey in Request.Cookies(objItem)
            Response.Write objItem & "(" & objItemKey & ") = " & _
                & Request.Cookies(objItem)(objItemKey) & "<BR>"
        Next
    Else
        'Print out the cookie string as normal
        Response.Write objItem & " = " & Request.Cookies(objItem) & "<BR>"
    End If
Next
```

```
End If
Next
```

这非常类似于前面的从 Request.Form 集合中提取多个值的复杂代码。但是这里可以使用 Haskeys 属性来判别每个条目是否为一个集合。而在 Form 例子里, 必须查询 Request.Form(item\_name).Count 属性, 这是因为 Form 集合(和所有的除 cookie 外的其他集合)成员不可能是真正的集合。ASP 只是做了“幕后”的工作, 得到了每个多条目集合的值。

### 2.3.3 Form 和 QueryString 的差异

了解了访问各种 ASP 集合的技术以后, 需要解决另一个问题是: Form 和 QueryString 集合之间的差异是什么? 假如准备使用 ASP, 毫无疑问应该清楚这种差异, 但需要参考 HTTP 工作方式重新认识, 理解它们。

通过 HTTP 从 Web 服务器请求页面或其他资源, 有两个通用的方法。可使用 GET 方法直接获得资源, 也可使用 POST 把值传给相应资源。GET 方法是缺省的, 可以看一下本章前面的一个 HTTP 请求的实例:

```
7/8/99 10:27:16 Sent GET /Store/Download.asp HTTP/1.1
```

假如把一个或多个成对的名称/值附在请求页面的 URL 后, 就变成请求的查询字符串, 且在 QueryString 集合中提供给 ASP 页面。单击 Web 页面、email 消息或其它文档的超链接, 或在浏览器的地址栏中输入地址并按回车, 或单击浏览器中的 Links 或 Favorites 按钮, 所有这些都要使用 GET 方法。

因此, 从这些动作中传递值给 ASP 的唯一方法是通过 QueryString 集合, 把值附在 URL 后。

出现在 Request.QueryString 集合中并被访问的值, 与前面看到的 Form 集合实例中的工作方式相同。URL 和查询字符串的结合:

```
http://mysite.com/process_page.asp?FirstName=Priscilla&LastName=Descartes
```

可以采用如下方式访问在 QueryString 集合中提供的值:

```
strFirstName = Request.QueryString("FirstName") 'Returns "Priscilla"
strLastName = Request.QueryString("LastName")   'Returns "Descartes"
strRaw = Request.QueryString 'Returns "FirstName=Priscilla&LastName=Descartes"
```

#### 窗体的 GET 和 POST 方法

在一个页面内使用 <FORM> 段时, 可以设置打开的 FORM 标记的 METHOD 属性值为“GET”或“POST”。缺省值为“GET”。假如使用“GET”或省略其属性, 浏览器将该值绑定在页面所有控件上, 成为一个查询字符串, 且附在被请求页面的 URL 上。

当这个请求到达 Web 服务器时, 其值由 ASP 的 Request.QueryString 集合提供。然而, 假如设置 METHOD 属性为“POST”, 浏览器将值包装进发送到服务器的 HTTP 报头中, 通过 Request.Form 集合提供给 ASP。

通常来说, 可以在所有的 HTML 窗体中使用 POST 方法。然而, 浏览器或服务器的 URL 字符串长度存在一定的限制。因此, 附有长的字符串可能会引起溢出和某些字符串的字符被截掉。同时, 查询字符串出现在浏览器的地址栏和所有的保存的链连和收藏夹中。不仅如此,



还显露了通过 Web 服务器时在 HTTP 请求中不想显示的值，它也可能出现你的服务器和其他路由服务器的日志文件中。在 HTTP 请求报头中的值很少是可见的，并且不出现在日志文件中。

使用 POST 方法需要注意的小问题是，当用户重新下载 <FORM> 时，窗体的值将不再保留，其值为空且必须重新输入。然而，当附在 URL 上时，其值被存储为一个链接，将被保留，因此将出现在所有的 URL 与字符串结合的请求中，这或许是个优点也可能是个缺点，这根据应用而定（一些浏览器在客户端上能够在一定范围内自动保留一个页面上的值）。

另一点是 URL 与查询字符串的结合体不能包含任何空格或其他非法字符，否则的话，Navigator 和一些其他的浏览器将出现问题。非法字符是那些用来分隔 URL 和查询字符串的部分，例如 “/”，“:”，“?”，和 “&”（IE 能够自动将空格转换为正确的格式——加号 “+”。但其他的非法字符不能处理）。ASP 服务器对象提供 URLEncode 方法处理这种变换，第 4 章将讨论相关内容。

### 2.3.4 查看 Request 和 Response 对象内容

到目前，主要讨论了一些理论问题，没有列举特别的实例。因为已经讨论过的内容多数情况下互相之间是密切相关的。然而本书为这一章提供了一系列的实例页面，说明 Request 和 Response 对象的大多数属性。应用所讲过的实例，能够理解这些页面，并可进行相应的修改，用它们作为试验实例。

本章及其他所有章节的代码样例均提供给用户，可以从 Wrox 出版社站点下载。

必须首先在 Web 服务器的 WWWRoot 内的子目录下安装实例，然后使用浏览器访问 Chapter02 子目录，使用：

`http://your_server_name_or_IP/subdirectory_name/Chapter02/Default.asp`

这里 your\_server\_name\_or\_IP/subdirectory\_name 是安装下载文件的本地路径。

#### 1. 查看 Request 对象成员

这提供一个包含选项的菜单用来试验 Request 和 Response 对象，首先选择 Using the Request Object，如图 2-4 所示。

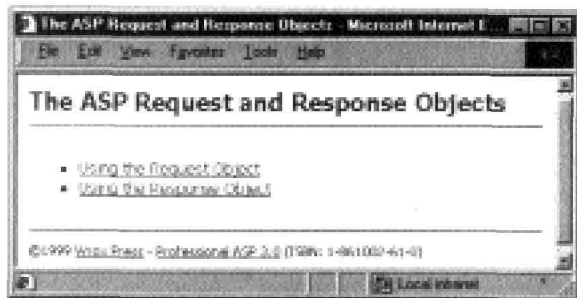


图 2-4 实例屏幕 1

图 2-5 显示一个 HTML 窗体的实例，包含一些预先设置好的值，可以按自己的想法编辑这些值，然后点击 “Submit” 按钮。

这将打开一个页面，如图 2-6 所示，显示集合和 TotalBytes 属性的全部内容。第一屏显示的是 Form、QueryString 和 Cookies 集合。

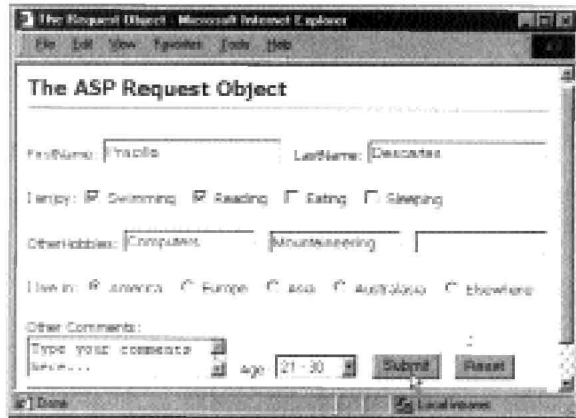


图2-5 实例屏幕2



图2-6 实例屏幕3

注意，假如在窗体页中编辑了 HTML 控件的值，对于 Cookies 集合以及其他的集合，在读者计算机上的页面上可能有不同的值。

可以从“Form Collection”段中看到窗体上的 HTML 控件的值如何在 ASP 的 Request.Form 集合中表示。也可以用原始的 <FORM> 页(名称为 request\_form.asp)来试验和检测，以了解创建窗体的 HTML 和如何与相应值相联系。

这个页面后面是 ClientCertificate 集合。这里是空的，因为服务器不要求客户端提供证书。下面的是 ServerVariables 集合，图 2-7 的屏幕图显示的是集合中包含的有用的值。

在本书的后面附录中可以找到所有 ServerVariables 集合成员的一个列表，及其值的说明。然而，可从前面讨论的在请求页面时从客户端发出的 HTTP 报头中见到这些成员。当请求收到后，Web 服务器也增加它本身的一些值到集合中，正如下面可以看

到的由运行在名为 wroxbox 服务器上的 IIS 5.0 创建的页面那样。



图2-7 实例屏幕4

#### (1) 页面是如何工作的

为创建这个页面，使用了本章前面在对 Form 集合和如何访问其值的讨论中所看到的完全相同的代码。例如，遍历所有的集合 (除 Request.Cookies 外)，使用：

```
For Each objItem In Request.collection_name

    Response.Write objItem & " = " & Request.collection_name(objItem) & "<BR>"

Next
```

遍历 Cookies 集合，可以使用：

```
For Each objItem In Request.Cookies

    If Request.Cookies(objItem).HasKeys Then
        'Use another For...Each to iterate all keys of dictionary
        For Each objItemKey in Request.Cookies(objItem)
            Response.Write objItem & "(" & objItemKey & ") = " & _
                & Request.Cookies(objItem)(objItemKey) & "<BR>"
        Next
    Else
        'Print out the cookie string as normal
        Response.Write objItem & " = " & Request.Cookies(objItem) & "<BR>"
    End If

Next
```

为获得TotalBytes属性，可简单地使用：

```
Request.TotalBytes = <% = Request.TotalBytes %><P>
```

读者应该注意到，在两个集中出现的某些值不是从窗体的HTML控件中直接得到的。QueryString集合包括了两个名为chapter和sample的值，如图2-8所示。



图2-8 实例的集合内容

为在请求中创建这两个值，将一个字符串附在窗体的ACTION属性的URL上，这是可以接受的。工作方式与附在一个<A>元素的HREF属性上相类似。查询字符的值出现在QueryString集合中，且被POST的窗体控件值出现在Form集合中。

```
<FORM ACTION="show_request.asp?chapter=2&sample=The+Request+Object" METHOD="POST">
```

为防止非IE类的浏览器出现错误，必须将查询字符串中的空格用加号“+”来代替，读者将在第4章的Server对象的URLEncode方法中看到更多这种情况。

## (2) 创建客户端的cookie

为了确保至少有些值出现在Request.Cookies集合中，增加一些客户端脚本代码到原始的<FORM>页面request\_form.asp。将创建名称为VisitCount的多值cookie。另一个cookie是由另一个页面创建的，且已经存在于浏览器中。如图2-9所示，读者可以看到另外的cookie。

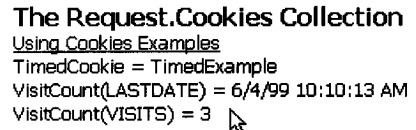


图2-9 实例的cookie内容

这是一段当窗体被装载时设置文档对象的cookies属性的客户端代码：

```
<SCRIPT LANGUAGE="JavaScript">
<!--
document.cookie = 'VisitCount=VISITS=3&LASTDATE=6%2F4%2F99+10%3A10%3A13+AM';
//-->
</SCRIPT>
```

另外，必须将内容进行编码，以便它能被正确地传送到服务器（同样的规则也适用于将查询字符串附到URL上）。在第4章中，讨论Server对象的URLEncode方法时，读者将了解到更多细节。

## 2. 查看Response对象的成员

回到Chapter02实例的最初的Default.asp页面，这次选择“Using the Response Object”链接，这个页面显示的是Response对象的集合和属性的内容，且提供到所有Response对象方法的链接。

图2-10是使用浏览器Netscape Communicator4.61的屏幕，用来证明使用的是纯服务器端和跨平台兼容技术。需要注意的是Cookies集合是为Response对象而建立的，仅显示cookie的名称而不显示其值。浏览该页时，可能得不到cookie或得到与这个页面不同的cookie。

各种Response属性说明了将要用来创建HTTP报头的一些信息。HTTP报头与页面的其他部分(HTML和文本内容)被发往到客户端。这些属性中的一些以及所有的Response对象的方法均有链接，允许读者打开另一个页面来显示其使用情况。我们稍后再回到这些页面。

页面中的属性是通过读取相应的属性并插入到页面中而创建的。由于这些是动态链接，通过<A>元素来选择。

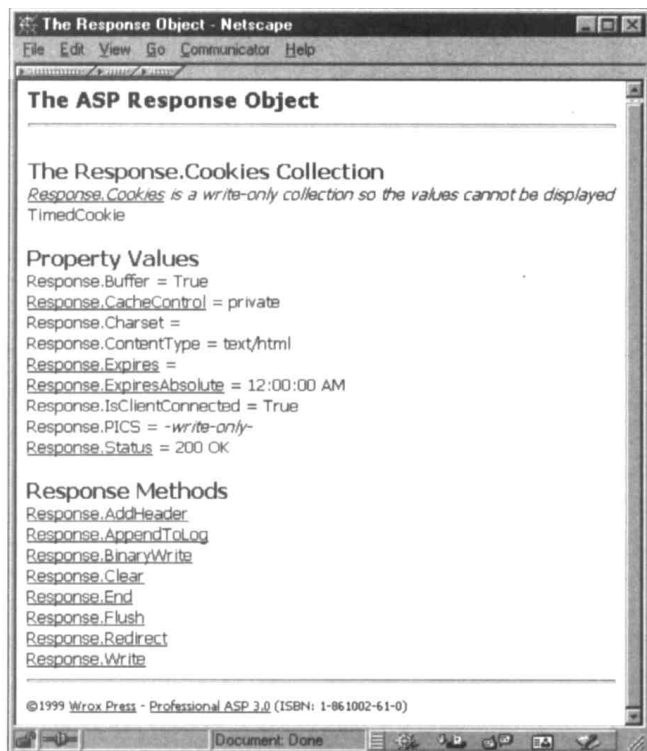


图2-10 实例的Response对象的集合和属性的内容

```
<A HREF="headers/expiretest_form.asp">Response.CacheControl</A>
= <% = Response.CacheControl %><BR>
```

链接到每个方法仅是通过<A>链接元素，页面中唯一复杂的部分是Response.Cookies集合。通常只能访问cookie，读取Request.Cookies集合中的值。当访问Response.Cookies集合时，必须在发送任何输出到客户端之前结束对它的所有引用。因此在页面的上部，通过遍历集合创建页面的HTML放在一个局部字符串变量中。

```
strCookies = ""
'We can only read the key names and not the values because
'the Response.Cookies collection is 'write only'
For Each objItem In Response.Cookies

    If Response.Cookies(objItem).HasKeys Then
        'Use another For Each to iterate all subkeys
        For Each objItemKey in Response.Cookies(objItem)
            strCookies = strCookies & objItem & "(" & objItemKey & ")<BR>"
        Next
    Else
        'print out the cookie string as normal
        strCookies = strCookies & objItem & "<BR>"
    End If
Next
```

然后在页面的适当点上插入结果。

```
<P><DIV CLASS="subhead">The Response.Cookies Collection</DIV>
<I><A HREF="cookies/setcookies.asp">Response.Cookies</A>
is a write-only collection so the values cannot be displayed</I><BR>
<% = strCookies %>
```



### 2.3.5 ASP中cookie的使用

在前面所看到页面中，一些集合、属性和方法已经链接到其他的页面，用来显示 Request 和 Response 对象的各个特性细节，我们将在本章的其余的部分研究这些内容，我们将学习那些提供给 ASP 代码使用的集合、方法和属性的各种技术。

在本章前面已经看到了如何使用 Request.Cookies 和 Response.Cookies 集合来创建和阅读 cookie，点击上面两个页面中任一个的“Cookies”链接时，这个页面包含一些设置了三个 cookie 的值的 ASP 代码，且在页面上显示被执行的代码，如图 2-11 所示。



图2-11 实例的Response.Cookies的内容屏幕1

点击“Show Cookies”的链接时，cookie 的内容就显示出来了。这是通过遍历 Request.Cookies 集合而得到的，这与在上一页所用到的方式完全相同，如图 2-12 所示。

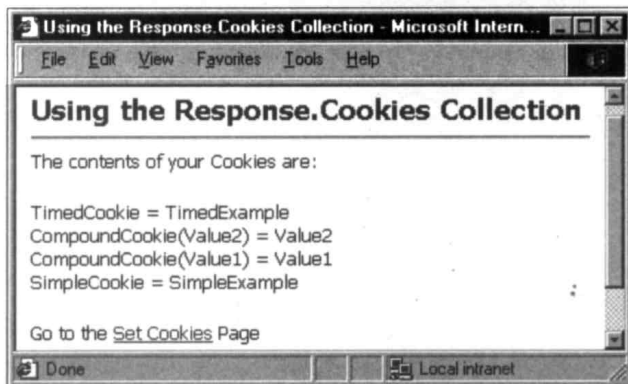


图2-12 实例的Response.Cookies的内容屏幕2

这个屏幕图显示的是运行前面看到的设置 cookie 值的代码的结果。可能会看到其他已经存储在计算机系统里的 cookie。然而，假如现在关闭浏览器然后重新打开浏览器，然后运行显示 cookie 的页面，除了 TimedCookie 外，所有的 cookie 都不见了，这是由于只有这个 TimedCookie 具有有效期的设置，其他的在浏览器关闭时，自动消失了。

#### 1. cookie中存储用户的细节情况

可以使用 cookie 来存储这两类值：当浏览器关闭时我们不想保存的值（例如用户的注册信息）以及在用户访问站点时要保留的值。在每种情况下 cookie 的值对于来自用户浏览器的每个



页面请求的ASP都是可用的。

然而，需要记住的是，cookie只有在对Cookie中的虚拟路径(path)内的页面发出请求时，才会发往服务器。缺省时，假如 path 的值在cookie中没有设置，则其值为创建 cookie的页面的虚拟路径。为使一个cookie发往一个站点的所有页面，需要使用 Path="/"。

这里是个实例，从自定义的 Login页面中，将用户的注册信息存贮在一个 cookie中，由于没有应用有效期，cookie值仅在关闭这个浏览器之前保留：

```
...
Response.Cookies("User")("UID") = "<% = Request("UserName") %>"
Response.Cookies("User")("PWD") = "<% = Request("Password") %>"
Response.Cookies("User").Path = "/adminstuff" 'Only applies to admin pages
...
```

现在，在用户从 adminstuff目录或其子目录请求的每个页面中，都可以找到这个 cookie。假如它不存在，可以将用户重定向到注册页面：

```
...
If (Request.Cookies("User")("UID") <> "alexhomer") _
    Or (Request.Cookies("User")("PWD") <> "secret") Then

    Response.Redirect "login.asp?UserName=" & Request.Cookies("User")("UID")

End If
...
```

由于把cookie中的用户名放在 Response.Redirect的URL查询字符串中，假如在口令输入时出现错误且希望用户不必重新键入用户名，可以在 login.asp页面中使用它：

```
<FORM ACTION="check_user.asp" METHOD="POST">
<INPUT TYPE="TEXT" NAME="UserName"
    VALUE="<% = Request.QueryString("UserName") %>"><P>
<INPUT TYPE="TEXT" NAME="Password"><P>
<INPUT TYPE="SUBMIT" VALUE="Login">
</FORM>
```

## 2. 修改现有的cookie

可以使用 ASP修改现有的 cookie，但不能只多值 cookie中的一个值。当更新一个在 Response.Cookies集中的Cookie时，现有的值将丢失。我们可以用如下代码创建一个 cookie，可以使用：

```
Response.Cookies("VisitCount")("StartDate") = dtmStart
Response.Cookies("VisitCount")("LastDate") = Now
Response.Cookies("VisitCount")("Visits") = CStr(intVisits)
Response.Cookies("VisitCount").Path = "/" 'Apply to entire site
Response.Cookies("VisitCount").Expires = DateAdd("m", 3, Now)
```

假如想要更新Visits和LastDate的值，必须先找到不需改变的所有值，然后重写整个的cookie：

```
datStart = Request.Cookies("VisitCount")("StartDate")
intVisits = Request.Cookies("VisitCount")("Visits")
Response.Cookies("VisitCount")("StartDate") = dtmStart
Response.Cookies("VisitCount")("LastDate") = Now
Response.Cookies("VisitCount")("Visits") = CStr(intVisits)
Response.Cookies("VisitCount").Path = "/"
Response.Cookies("VisitCount").Expires = DateAdd("m", 3, Now + 1)
```

且对于几乎所有的其他 Response方法和属性，应该在写入任何内容(即打开<HYML>标记或任何文本或其他HTML)到响应之前完成这个工作。

## 2.4 使用ServerVariables集合

当讨论Request对象的内容时，要研究的集合之一就是 ServerVariables集合。这个集合包

含了两种值的结合体，一种是随同页面请求从客户端发送到服务器的 HTTP 报头中的值，另外一种是由服务器在接收到请求时本身所提供的值。为显示 Server Variables 集合中值的使用方式，在 Request Object 页面 (Show\_request.asp) 中，点击 “ServerVariables Examples” 链接，打开另外一个页面，如图 2-13 所示。

图 2-14 所示窗口显示的是 Server Variables 集合中一些非常有用的值的一个子集。

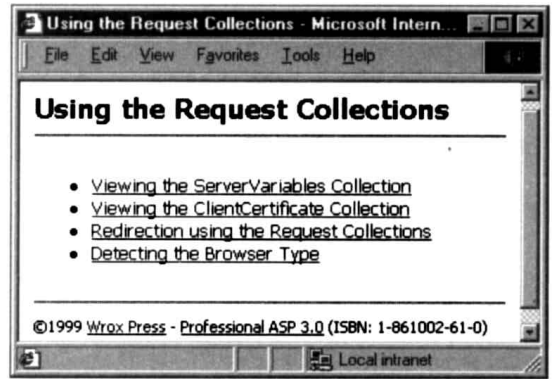


图2-13 实例的Response Collections的内容屏幕1

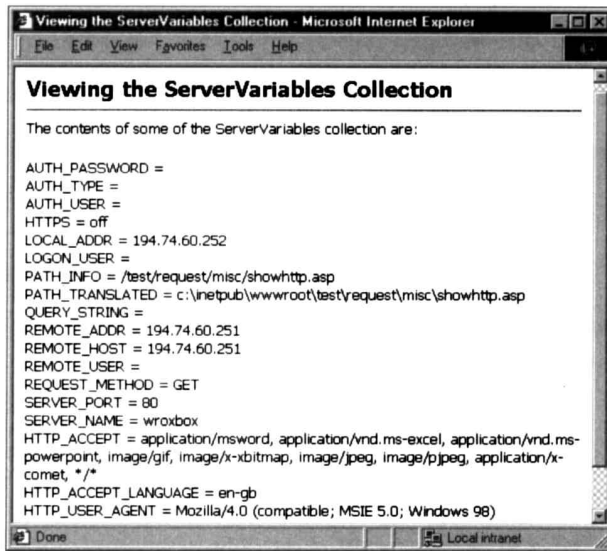


图2-14 实例的Response Collections的内容屏幕2

#### 2.4.1 “自引用”页面

在 ServerVariables 集合中返回的值包含 Web 服务器的详细信息和当前页面的路径信息。在任何地方创建一个页面都可使用这些信息。例如创建一个 “自引用” 页面，此页面能够再次调用自身完成另一项任务，我们可以用以下代码：

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

同样的效果可以用 HTTP 的 “SCRIPT\_NAME” 值获得：

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

使用 <A> 元素打开一个不同页，可以使用：

```
...
<%
strFullPath = Request.ServerVariables("PATH_INFO")
'Strip off the file name
strPathOnly = Left(strFullPath, InStrRev(strFullPath, "/"))
strNextPage = strPathOnly & "pages/next_page.asp"
%>
```

```
%>
...
<A HREF="<% = strNextPage %>">Next Page</A>
...
```

即使原始页面的名称或位置发生变化,这些实例都能正常工作,因为使用了当前页面的路径信息(当然,第二个例子在分离的目标页的名称发生变化时运行会失败)。

换句话说,如要为搜索引擎的子会话自动建立 URL,可以收集 ServerVariable 的一些值:

```
strFullURL = "http://" & Request.ServerVariables("LOCAL_ADDR") _
    & ":" & Request.ServerVariables("SERVER_PORT") _
    & Request.ServerVariables("PATH_INFO")
```

这将创建一个完整的 URL 包括端口号(这种情况下,不是标准值 80)。例如,结果可能是:  
http://194.74.60.254:1768/thispath/thispage.asp

## 2.4.2 检测浏览器的版本

ServerVariables 集合中,另外一个有用的值是用户浏览器的用户代理字符串。在“Detecting the Browser Type”页面(browser type.asp),使用 ServerVariables 集合中的“HTTP\_USER\_AGENT”值来获得用户代理字符串,一些脚本用来解析该信息并寻找生产厂家名称和浏览器版本。

```
<%
strUA = Request.ServerVariables("HTTP_USER_AGENT")
Response.Write "The User Agent string is <B>" & strUA & "</B><P>"

If InStr(strUA, "MSIE") Then

    Response.Write "To upgrade your browser go to " _
        & "<A HREF=" & Chr(34) & "http://www.microsoft.com/ie/" _
        & Chr(34) & ">http://www.microsoft.com/ie/</A></P>"
    intVersion = CInt(Mid(strUA, InStr(strUA, "MSIE") + 5, 1))

    If intVersion >= 4 Then
        Response.Write "You can use Microsoft Dynamic HTML"
    End If

Else

    If InStr(strUA, "Mozilla") Then

        If InStr(strUA, "compatible;") = 0 Then

            Response.Write "Your browser is probably Navigator. You can " _
                & "download the latest version of Navigator from " _
                & "<A HREF=" & Chr(34) & "http://home.netscape.com/download/" _
                & Chr(34) & ">http://home.netscape.com/download/</A></P>"
            intVersion = CInt(Mid(strUA, InStr(strUA, "/") + 1, 1))

            If intVersion >= 4 Then
                Response.Write "You can probably use Netscape Dynamic HTML"
            End If

        Else

            strVersion = Mid(strUA, InStr(strUA, "compatible;") + 12)
            strProduct = Left(strVersion, InStr(strVersion, " "))
            Response.Write "Your browser is Navigator-compatible. You can " _
                & "search for the manufacturer using a search engine, such as " _
                & "<A HREF=" & Chr(34) _
                & "http://www.altavista.digital.com/cgi-bin/query?q=" _
                & strProduct _
                & Chr(34) & ">http://www.altavista.com/</A></P>"

        End If

    End If

End If
```

```
End If
```

```
End If
%>
```

对IE 5.0和Navigator 4.61的搜索结果分别如图2-15和图2-16所示。对于其他厂家的浏览器，可以得到一个链接在 AltaVista Web 站点自动开始搜索厂家的名称。

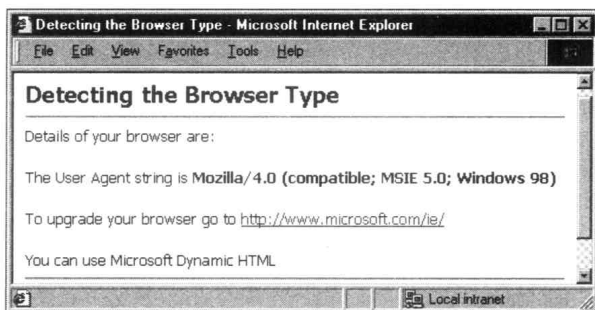


图2-15 IE 5.0中显示的结果

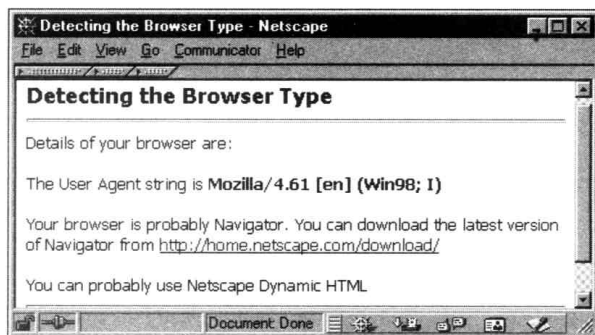


图2-16 Navigator 4.61中显示的结果

注意，Netscape在用户代理字符串中不提供厂家的名称，因而无法绝对保证一个浏览器一定是Navigator。

### 2.4.3 检测浏览器的语言

ServerVariables集合中另外一个有用的值是“HTTP\_ACCEPT\_LANGUAGE”，它包含了一个当浏览器安装时指定的，或硬编码进用户的地区版本的语言代码。语言代码的例子有 en-us(英国、美国)、de-at(德国、澳大利亚)和es-pe(西班牙、秘鲁)。

语言代码可以是一般的且省略方言标识：例如，在我们的站点 Wrox 者，大批浏览者都是将en(英语)作为语言代码。

因此，可以检测语言代码并自动装载一个合适的特定地区或指定特言版本的页面。

```
strLocale = LCase(Left(Request.ServerVariables("HTTP_ACCEPT_LANGUAGE"), 2))

Select Case strLocale
    Case "en": Response.Redirect "http://uk_site.co.uk/"
    Case "de": Response.Redirect "http://de_site.co.de/"
    Case "fr": Response.Redirect "http://fr_site.co.fr/"
    '... etc
    Case Else: Response.Redirect "http://us_site1.com/"
End Select
```

或者根据特定的方言，重定向页面：

```
strLocale = LCase(Request.ServerVariables("HTTP_ACCEPT_LANGUAGE"))

Select Case strLocale

    Case "en-gb": Response.Redirect "http://uk_site.co.uk/"
    Case "en-us": Response.Redirect "http://us_site.com/"
    Case "es-pe": Response.Redirect "http://es_site2.co.pe/"
    '... etc
    Case Else: Response.Redirect "http://us_site1.com/"

End Select
```

#### 2.4.4 其他有用的ServerVariables集合的值

可以访问和使用ServerVariables集合中的任何一成员，控制ASP页面响应一个请求的方式。可以检查一个浏览者访问站点时使用的是否是缺省端口 80或还是另一个。在这个例子里，寻找通过端口443的访问——这个端口提供的是安全套接字层 (Secure Socket Layer, SSI)访问(和其他的协议)，且将它们重定向到一个相应的页面。

```
If Request.ServerVariables("SERVER_PORT") = "443" Then
    Response.Redirect "/securesite/default.asp" 'Secure user
Else
    Response.Redirect "/normalsite/default.asp" 'Non-secure user
End If
```

假如要求浏览者注册且由服务器验证 (而不是允许他们在 Web服务器的IUSR帐号下匿名访问，这个问题将在后面章节中详细讨论)，可以查询用户名称，来判定正在与我们打交道的用户是谁，是否装载页面给该用户。例如，下面的这个代码将只向名为 Administrator的用户显示管理链接。

```
...
<A HREF="dispcnfg.asp">Change Display Configuration</A><BR>
<A HREF="dispcolr.asp">Change Display Colors</A><BR>
<A HREF="keyboard.asp">Change Keyboard Configuration</A><BR>
<%
If Request.ServerVariables("AUTH_USER") _
    = UCase(Request.ServerVariables("SERVER_NAME")) & "\Administrator" Then
%>
    <A HREF="allusers.asp">Administer All Users</A><BR>
    <A HREF="usrlogon.asp">Administer Logon Information</A>
<%
End If
%>
...

```

注意ASP不填写ServerVariables集合直到你访问其中的一个成员。首次访问该集合的一个成员将使IIS得到它的全部，应只在需要时才使用ServerVariables集合。

## 2.5 其他Request和Response技巧

现在，来看一下几个使用 Request和Response对象的有用技巧，包括：

- 连接、缓冲和页面重定向的管理。
- HTTP 报头、缓存与“到期”页面的操作。
- 利用客户证书。
- 创建定制的日志文件消息。

### 2.5.1 连接、缓冲和页面重定向的管理

ASP的一个很有用的特点就是使用户能够从一个 ASP网页转向到另一个网页 (ASP或HTML), 或另一个源文件 (例如一个ZIP文件或文本文件)。这对用户来说是透明的, 实际上是浏览器做这个工作。当使用 Response.Redirect方法来载入一个新的网页时, 实际上是发送回一个特殊的HTTP 报头到客户。此报头为:

```
HTTP/1.1 302 Object Moved
Location /newpath/newpage.asp
```

浏览器读到此报头信息, 并按 Location值的指示载入页面。这在功能上与在 Web页中使用客户端HTML<META>标记相同, 例如:

```
<META HTTP-EQUIV="REFRESH" CONTENT="0;URL=/newpath/newpage.asp">
```

这带来的一个问题是, 服务器与用户之间的代理服务器可能会提供它自己的包含与新页面的链接的消息, 而不是直接载入新页面。而且浏览器根据厂商和版本可能做同样的工作。这就去除了假定的透明, 而且对用户来说一直收到的是错误信息, 则对你的站点的访问变得比较麻烦。

在发送诸如文本或HTML等任何页面内容后, 我们就不能再使用 Redirect方法。然而, 一个看起来能够限制“代理服务器影响”的方法就是, 先确定没有输出 (包括HTTP报头)被发送到客户。在ASP 2.0中, 必须打开缓冲, 然后使用 Clear方法来清空缓冲区:

```
Response.Buffer = True

'Some condition to select the appropriate page:
If Request.ServerVariables("SERVER_PORT") = 1856 Then
    strNewPage = "/newpath/this_page.asp"
Else
    strNewPage = "/newpath/the_other_page.asp"
End If

Response.Clear
Response.Redirect strNewPage
```

在ASP 3.0中, 缓冲缺省为打开, 所以第一行可被忽略, 但它是无害的, 而且能确保我们的网页即使在ASP2.0环境中也仍然能工作。

与其使用这种类型的HTTP报头重定向, 不如使用ASP 3.0的一个新特性, 它允许我们通过Server对象的Transfer方法转换为执行另一个网页, 我们将在第4章进一步研究这个问题。

#### 1. ASP页面缓冲区

正如已看到过的, IIS 5.0中ASP 3.0页面缓冲是缺省打开的, 在早期的版本中是缺省关闭的。微软告诉我们缓冲在IIS 5.0中提供了更有效的网页传送, 这就是缓冲缺省状态被改变的原因。在大部分情况下, 这对我们没有影响。但是, 假如有一个非常大的网页, 或一个用ASP或别的服务器端代码和组件花费一定时间创建的网页, 当其各部分完成时, 我们能够分批刷新它们到客户:

```
...
... Code to create first part of the page
...
Response.Flush
...
... Code to create next part of page
```



```
...  
Response.Flush  
...
```

有时可能希望在页面结束之前的某些点上停止代码的执行,可以通过调用 `End` 方法去刷新所有的当前内容到客户并中止任何进一步的处理过程。

```
...  
If strUserName = "" Then Response.End 'No name so stop execution  
...  
... Code to use UserName value  
...
```

假如正在创建被缓冲的输出,且没有发往客户,则我们可以改变主意,可以使用 `Clear` 方法来删除它:

```
...  
... Code to create first part of the page  
...  
If strUserName = "" Then Response.Clear  
...  
... Code to create a new version of this part of the page  
...
```

这里有两个演示缓冲和重定向的实例网页,可以从“ `Response Object` ”主页面(`show_response.asp`)下载它们。第一个 `Response.Redirect` 例子网页命名为 `redirect.asp`,它在缓冲的页面中写入一些内容,清除缓冲区,并重定向到另一个网页:

```
For intLoop = 1 To 1000000  
    Response.Write "."  
Next  
  
Response.Clear  
Response.Redirect "show_redirect.asp"  
Response.End
```

目标页 `show_response.asp`,做同样的工作,但重定向则是回到“ `Response Object` ”主页。因为这些网页都在缓冲区内,而且所有的输出在重定向之前必须被清除,故在浏览器中没有可见的输出。然而,可以通过观察浏览器的状态条看到发生的每一次重定向,如图 2-17所示。

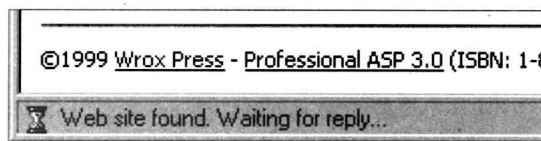


图2-17 浏览器的状态条

在“ `Request Object` ”主页中,点击“ `Response.Flush` ”链接将打开第二个示例网页 `usebuffer.asp`,它简单地遍历一个字符串的每一字符,以一定的延迟将它们刷新到客户,这虽是Web服务器和ASP极低效率的使用方式,但它演示了缓冲的工作方式。

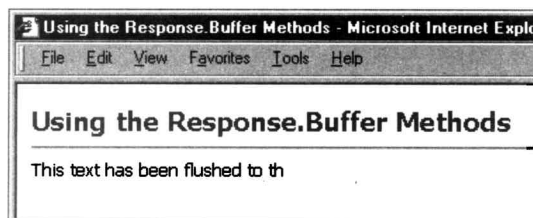


图2-18 usebuffer.asp运行情况

下面是所要求的最小化的 ASP 代码，注意我们分别把每个字符刷新到浏览器，因为这样的话它将被存放在缓冲区中，直至网页完成：

```
strText = "This text has been flushed to the browser using " & _  
          "<B>Response.Flush</B><P>"  
  
For intChar = 1 To Len(strText)  
  
    For intWrite = 1 To 100000  
        Next  
  
    Response.Write Mid(strText, intChar, 1)  
    Response.Flush  
  
Next
```

## 2. Response.IsClientConnected 属性

IsClientConnected 属性在 ASP 2.0 中已经存在了，但却有些不可靠。在其返回一个准确的结果之前必须发送一些输出到客户。这一问题在 ASP 3.0 中已被解决。现在这一属性可被自由使用。

IsClientConnected 是观察用户是否仍连到服务器和正在载入 ASP 创建的网页的有用方式。如果用户断开连接或停止下载，我们就不要再浪费服务器的资源创建网页，因为缓冲区内容将被 IIS 丢弃。所以，对那些需要大量时间计算或资源使用较多的网页来说，值得在每一阶段都检查浏览器是否已离线：

```
...  
... Code to create first part of the page  
...  
If Response.IsClientConnected Then  
    Response.Flush  
Else  
    Response.End  
End If  
...  
... Code to create next part of page  
...
```

## 2.5.2 操作 HTTP 报头

我们已经在几处见到 ASP 如何创建或修改在响应页面请求时被发送到客户的 HTTP 报头。在 Response 对象中有几个属性和方法可帮助我们做到一点。下面是一些报头方法：

- 控制缓存和有效期。
- 创建状态和定制的 HTTP 报头。
- 指定 MIME 类型或内容类型。
- 添加 PICS 标签。

接下来将简要地研究每一个方面。可在“Response Object”主页 (show\_response.asp) 上，单击相关属性名或方法名，来检查我们所说明的属性和方法，如图 2-19 所示。

### 1. 缓存和“到期”ASP 网页

用户的浏览器以及他们和服务器之间的任一代理服务器，都可以缓存 HTML 和用 ASP 创建的网页。当用户随后请求页面时，浏览器就发送一个“最新修改”的请求到服务器（使用一个包含缓存版本的日期的 HTTP\_IF\_MODIFIED\_SINCE 报

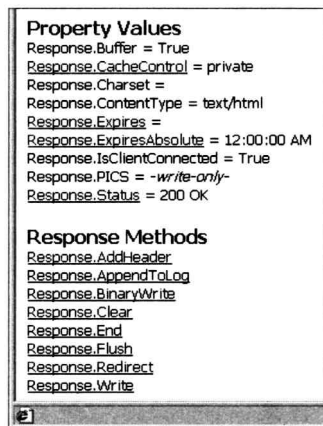


图 2-19 检查属性和方法

头), 询问网页是否已被修改。

若没有被修改, 服务器应用状态码和消息 “ 304 Not Modified ” 来响应, 浏览器将使用缓存的内容而不会通过网络下载一个副本。若已经存在已修改的版本, 它就会与 “ 200 OK ” 状态码和消息一道被发送出去。

#### (1) Response.CacheControl属性

其他的一些因素也会影响这一处理过程。然而, 任一被网页使用的网络路由内的代理服务(一般位于客户机端), 能被通过设置 Response.CacheControl 属性为 Private 来放弃缓存网页。在 ASP 3.0 中对 ASP 网页这是缺省的, 不用设置。但在网页为个别访问者特别定制时尤其有用。这可以阻止别的在同一网络上的用户进入同一网页。当 CacheControl 的属性值被设为 Public 时, 允许代理服务器缓存网页。注意, 一些代理服务器可能表现得不尽相同, 或忽视或越过这个报头。

在 IE 4 中, 在代理服务器缓存可用时, 有可能得到一个虚假的 “ This page has expired ” 消息。我们已提供了一个网页 (expiretest\_form.asp), 可以通过自己的代理服务器在网络上做试验, 来检查这一属性的影响。可以通过在 “ Response Object ” 主页中单击 “ Response.CacheControl ” 链接来显示这个网页, 如图 2-20 所示。

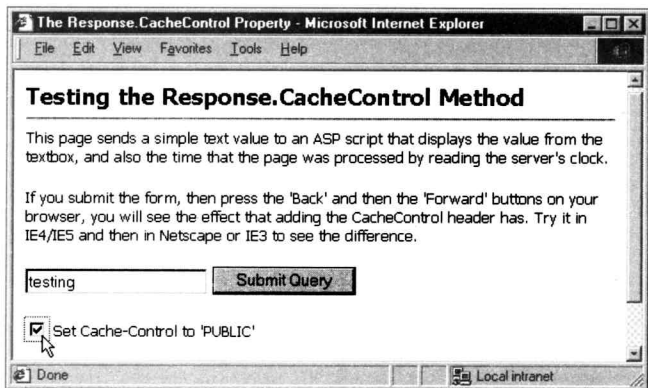


图2-20 显示检查结果 1

这一页面提交到 expiretest\_result.asp 网页时, 能够设置 Response.CacheControl 属性, 然后在网页中插入值和脚本被执行的时间:

```
<%  
If Request.Form("public") = "on" Then 'Cache-Control check box was ticked  
    Response.CacheControl = "Public"  
Else  
    Response.CacheControl = "Private"  
End If  
%>  
  
<HTML>  
...  
Cache-Control is: <B><% = Response.CacheControl %></B><P>  
Value in text box is: <B><% Response.Write Request.Form("textbox") %>  
</B><P>Script was executed at:<B>  
<%  
Response.Write Right("0" & Hour(Now),2) & ":" & Right("0" & Minute(Now),2) _  
                & ":" & Right("0" & Second(Now),2)  
%></B>
```

通过单击浏览器上的 “ Back ” 和 “ Forward ”, 能看到代码是自动重执行还是使用缓存的

副本，如图 2-21 所示。结果随浏览器的不同而变化。

## (2) Response.Expires和Response.ExpiresAbsolute属性

控制缓存的网页存放时间的两个属性为

Response对象的Expires和ExpiresAbsolute属性。Response.Expires定义了网页在从缓存区被丢弃前应保持有效的时间长度，以创建以来的分钟数形式表示。ExpiresAbsolute属性为到期时间设置了一个绝对的日期和时间。

我们提供一个命名为 addheaders\_form.asp 的例子网页，用于演示如何使用这些属性。在

“Response Object” 主页中单击对这两种属性的链接，如图 2-22 所示。

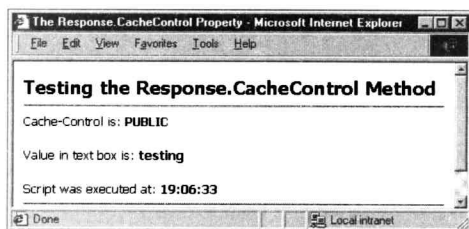


图 2-21 显示检查结果 2

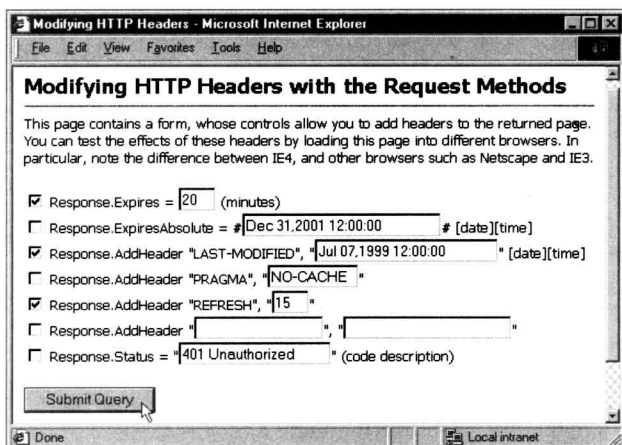


图 2-22 addheaders\_form.asp 显示的属性

在这一页面中，可加入自己定制的 HTTP 报头，并可设置一些影响响应的 HTTP 报头的多种属性。在“Submit”按钮上单击时，页面 show\_headers.asp 在返回的数据流中添加所选的报头，然后显示用来完成此操作的代码，显示相应的执行时间，可用来检查页面是被缓存还是被再次执行，如图 2-23 所示。

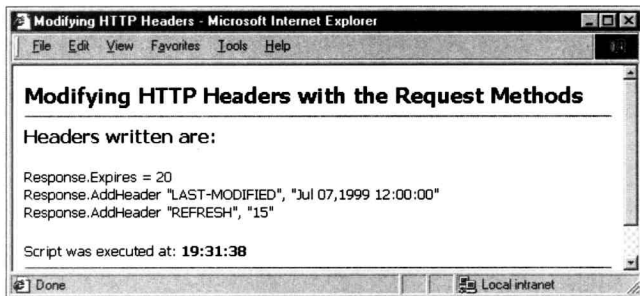


图 2-23 addheaders\_form.asp 显示的报头

show\_headers.asp 网页中的代码创建和添加 HTTP 报头，程序如下：

```
<%  
' Write HTTP headers before any other output  
If Request.Form("expires") = "on" Then _
```

```

Response.Expires = Request.Form("expires_value")

If Request.Form("expiresabs") = "on" Then _
    Response.ExpiresAbsolute = Request.Form("expiresabs_value")

If Request.Form("lastmod") = "on" Then _
    Response.AddHeader "LAST-MODIFIED", CStr(Request.Form("lastmod_value"))

If Request.Form("pragma") = "on" Then _
    Response.AddHeader "PRAGMA", CStr(Request.Form("pragma_value"))

If Request.Form("refresh") = "on" Then _
    Response.AddHeader "REFRESH", CStr(Request.Form("refresh_value"))

If Request.Form("addheader") = "on" And Len(Request.Form("addheader_name")) Then _
    Response.AddHeader CStr(Request.Form("addheader_name")), _
        CStr(Request.Form("addheader_value"))

If Request.Form("status") = "on" Then _
    Response.Status = Request.Form("status_value")
%>

<HTML>
...
...Show code and execution time
...

```

其余的部分仅仅是显示已被执行的代码和执行时间。读者会注意到包含在网页中的定制的报头“PRAGMA”(至今我们还没讨论过)。一些(先前的)的代理服务器使用它作为网页是否应被缓存的指示。缺省是网页被缓冲,除非接受到HTTP报头“PRAGMA=NO-CACHE”。

## 2. 创建状态码和定制的HTTP报头

可使用先前在实例网页中所看到的Response对象的AddHeader方法来创建自己的状态码或自己喜欢的定制的报头。这一方法需要两个参数: HTTP报头名称或一个包含其值或分配给它的值的字符串。作为一个例子,下面的代码在页面中添加REFRESH报头:

```
Response.AddHeader "REFRESH", "60;URL=newpath/newpage.asp"
```

这等同于客户端<META>元素:

```
<META HTTP-EQUIV="REFRESH" CONTENT="60;URL=newpath/newpage.asp">
```

换句话说,也可配合Status属性使用AddHeader方法使浏览器载入一个新的页面:

```
Response.Status = "302 Object Moved"
Response.AddHeader "Location", "newpath/newpage.asp"
```

这等同于使用Response.Redirect方法:

```
Response.Redirect "newpath/newpage.asp"
```

Response.Status属性可被用来发送一些所需要的状态消息,例如添加如下几行:

```
Response.Status = "401 Unauthorized"
Response.AddHeader "WWW-Authenticate", "BASIC"
```

强制浏览器显示一个用户名/口令对话框,然后使用BASIC验证把它们发送回服务器(将在本书后续部分看到验证方法)。

## 3. MIME类型和内容类型

当我们想向浏览器发送一个动态创建的字符串,而且它们自己提供给浏览器时没有直接指明内容类型,而是提供表示是否是磁盘文件的扩展名时,Response.ContentType是非常有用的。除非特别指定,所有ASP创建的网页缺省都为“text/type”。内容类型的标识符是MIME类



型(MIME代表Multi-purpose Internet Multimedia Extension 或Multi-purpose Internet Mail Extension, 通常依据上下文来定)。

例如, 若发送到客户的数据流是通过从数据库读二进制值创建的图片, 就需要在发送任何内容之前添加合适的CONTENT-TYPE报头:

```
Response.ContentType "image/jpeg"
```

假如从一个数据库创建一个XML文件, 使用MIME类型“text/xml”; 并且如果正在创建一个文本文件可以在文件编辑器中显示或作为一个磁盘文件在客户上被存储起来, 使用“text/text”。

#### 4. 添加PICS卷标

Response.Pics属性仅仅是添加一个PICS(Platform for Internet Content system)卷标到页面上, 方式与通常用<META>标记所用的方式相同:

```
QUOT = Chr(34)
```

```
strPicsLabel = "(PICS-1.0 " & QUOT & "http://www.rsac.org/ratingsv01.html" _  
                & QUOT & " l gen true comment " & QUOT _  
                & "RSACi North America Server" & QUOT & " for " & QUOT _  
                & "http://yoursite.com" & QUOT & " on " & QUOT _  
                & "1999.08.01T03:04-0500" & QUOT & " r (n 0 s 0 v 2 1 3))"
```

```
Response.Pics(strPicsLabel)
```

这段代码添加了如下的PICS卷标:

```
(PICS-1.0 "http://www.rsac.org/ratingsv01.html" l gen true comment "RSACi  
North America Server" for "http://yoursite.com" on "1999.08.01T03:04-0500"  
r (n 0 s 0 v 2 1 3))
```

要得到关于 PICS的更多的信息, 或了解更多的定义页面内容的方式, 请检索 <http://www.rsac.org> 站点。

在Internet Service Manager中定义报头

在第1章, 已经说明了如何在 Internet Service Manager(MMC插件)应用程序中设置每个

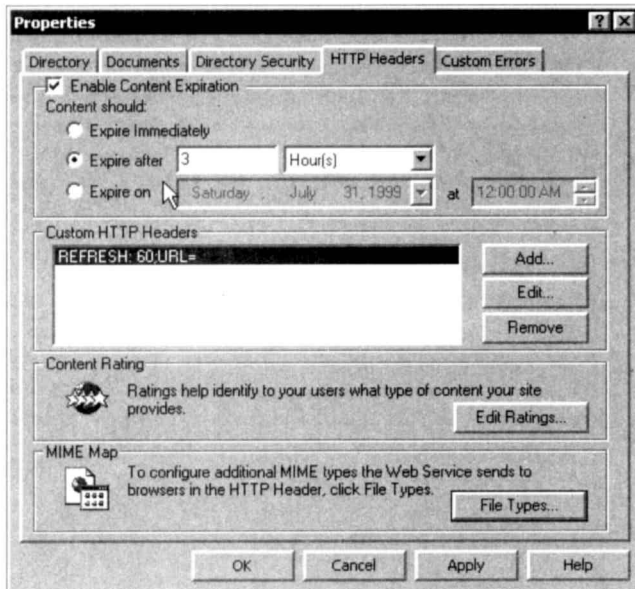


图2-24 HTTP Headers选项卡屏幕



Web网站和IIS 5.0中目录的属性，这就定义了使用此站点或目录资源发送到客户机的所有请求的HTTP报头，也就提供了使用每个网页中的ASP脚本代码设置这些属性的替代方法。

在Web站点或目录上右击鼠标并选择“Properties”，在其对话框的“HTTP Headers”选项卡中，可设置页面内容有效期的相对时间或绝对日期，定义定制的报头，创建 PICS 内容等级标签，也可以通过 MIME 类型映射来定义内容类型，如图2-24所示。

在图2-24中，可以看到已创建了自定义的 REFRESH HTTP 报头，应用于从此目录载入的所有网页。即每一分钟自动地重载（刷新）一次（对于显示棒球比赛的最新比分是非常理想的，但对服务器而言负担太重了）。Custom HTTP Headers 栏的 Edit 对话框如图2-25所示。

要在“MIME Map”框中添加自定义的内容类型映射，只需在“Properties”主对话框中单击“File Types”按钮把它们添加到清单中即可，如图2-26所示。

当使用HTTP报头开始试验时，你很快会发现不是所有的浏览器表现都相同，许多浏览器以不同的方式响应不同的HTTP报头，使得可靠地建立一个普遍适用的原则有时极为困难。

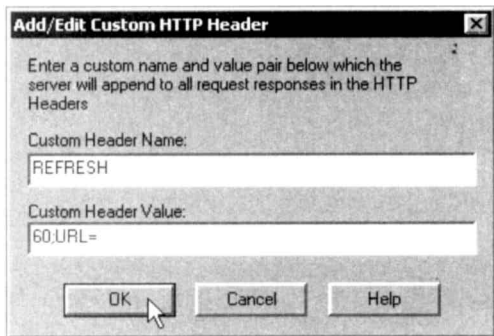


图2-25 Custom HTTP Headers 栏的 Edit 对话框

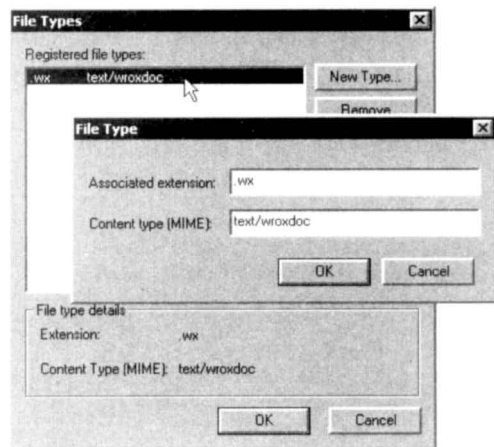


图2-26 添加内容类型映射的屏幕

### 2.5.3 使用客户证书

假如设立了一个安全的 Web 网站或部分内容具有安全机制的网站，可安装一个数字服务器证书，通过允许访问者使用证书中的加密的细节，来验证服务器。每一次对该站点或目录的页面请求，服务器都将发送证书的一个副本，浏览器可检查这个副本以确定正在和谁交谈。

同样，也可设置服务器，要求用户在进入网站时提供一个有效的数字证书。他们可从很多来源获得此证书，例如 Verisign (<http://www.verisign.com>) 或 Thawte Consulting (<http://www.thawte.com>)。读者将在第25章看到这一处理过程的细节。

这些情况都使用了 Request 对象的 ClientCertificate 集合的值，本章的实例代码中，已包含了一个显示用户如何使用此集合值的一些方法的页面。

在 Request Object 主页面 (show\_request.asp) 中单击 ClientCertificateExample，则打开 Using the Request Collection 菜单。在此页面中，选择 Viewing the ClientCertificate Collection 链接，打开显示随同页面请求发往服务器的客户证书的内容的页面。

这一网页被命名为 showcert.asp，而且其所做的一切就是遍历 ClientCertificate 集合显示其包含的所有值。可使用以前经常使用的简单代码来完成它，唯一的不同之处就是建立一个

HTML表以容纳结果，并将其截为每60个字符一组。

```
<TABLE CELLPADDING=0 CELLSPACING=0>
<%
For Each keyItem In Request.ClientCertificate()
    strItemValue = Request.ClientCertificate(keyItem)
    If Len(strItemValue) > 90 Then strItemValue = Left(strItemValue, 60) & "... etc."
    Response.Write "<TR><TD>" & keyItem & " = " & strItemValue & "</TD></TR>"
Next
%>
</TABLE>
```

运行结果如图2-27所示。



图2-27 运行结果屏幕

然而，要在自己系统上得到结果，必须建立一个安全的网站或目录。

使用客户证书重定向

一旦要求所有访问网站或部分网站的浏览者给出的其客户证书，就可以使用其包含的信息来制作我们为此用户创建的网页。例如，可使用他们的证书的 Organization 条目来自动使他们重定向到该网站的指定部分，使别的访问者重定向到别的地方：

```
If Request.ClientCertificate("SubjectO") = "Wrox Press Inc" Then
    Response.Redirect "/wrox_staff/default.asp" 'Wrox staff site
Else
    Response.Redirect "/public/Default.asp" 'Normal public site
End If
```

相应地，可使用 Country 条目来使访问者重定向到一个相应的网站：

```
Select Case Request.ClientCertificate("SubjectC")
    Case "UK": Response.Redirect "https://uk_site.co.uk/"
    Case "DE": Response.Redirect "https://de_site.co.de/"
```

```
Case "FR": Response.Redirect "https://fr_site.co.fr/"
'... etc.
Case Else: Response.Redirect "https://us_site.com/"
End Select
```

## 2.5.4 读写二进制数据

有两个方法提供了对从浏览器发送到服务器的 HTTP 数据流和从服务器返回到浏览器的数据流的二进制数据访问。Request.BinaryRead 方法可得到指定要读取的字节数的参数，并返回变体类型的数组，其中包含从请求的 POST 段中得到的字节（例如在 ASP 的 Form 集合中数据）。下面的程序读数据的头 64 个字节：

```
varContent = Request.BinaryRead(64)
```

假如使用了 BinaryRead 方法，以后就不能访问 ASP 的 Request.Form 集合。同样，一旦我们采用任何方式引用了 Request.Form 集合，就不能使用 BinaryRead 方法。

把二进制数据写进 ASP 创建的响应流中也是可能的，可采用 BinaryWrite 方法。需要给其提供想写到客户的字节的变体类型数组：

```
Response.BinaryWrite(varContent)
```

这些方法都很少使用，除非从一个数据库创建非 HTML 源才用到这些方法。使用的一个实例就是从数据库读取组成图像的字节，并使用 BinaryWrite 方法把它发送到客户。

## 2.5.5 创建定制的日志消息

假如设置了服务器，以 W3C Extended Log File Format 格式将请求记录到一个文本文件，可使用 Response.AppendToLog 方法在日志文件条目的结尾处添加一条消息字符串。若想为特定的网页存储一些值或消息，或在脚本中出现了特定的情况时，这种方式是非常有用的。

例如，通过的 Intranet 的 “stationary order” 应用程序，可以记录超过特定的条目数目的雇员的部门号码：

```
...
If intItemCount > 25 Then
    Response.AppendToLog "Large order from '" & strDept & "' department."
End If
...
```

### 设置扩展的日志

要使用 AppendToLog 方法，必须激活 W3C Extended Log File Format 日志设置。该设置方法是，进入 Properties 对话框中的 Web Site 选项卡，选中 Enable Logging 复选框，选择 W3C Extended Log File Format 并单击 Properties 按钮，如图 2-28 所示。

在出现的 Extended Logging Properties 对话框中，可选择想包括进日志文件的条目。确保选中 URI Stem，否则 AppendToLog 方法将失败，如图 2-29 所示。

我们提供了一个试图在日志文件中写入一个条目的简单实例页面，可从 Request Object 主页 (show\_request.asp) 中的 AppendToLog 方法链接处打开它。这一页面所做的全部工作就是创建一个包含当前日期和时间的简单字符串，然后执行 AppendToLog 方法：

```
strToAppend = "Page executed on " & Now
Response.AppendToLog strToAppend
```

结果如图 2-30 所示。

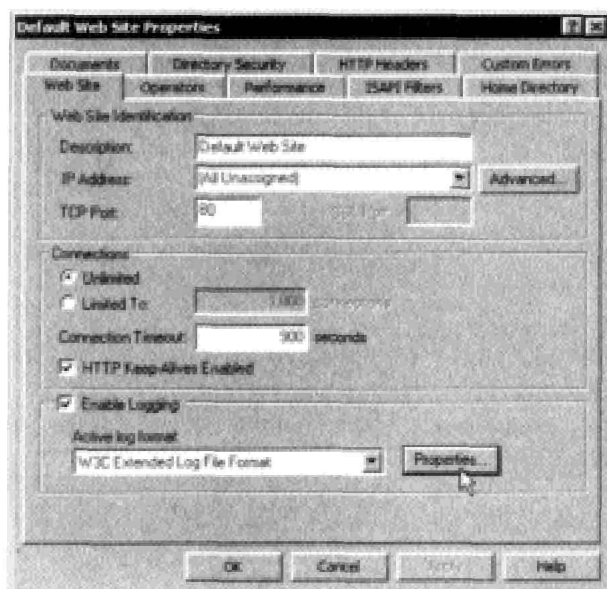


图2-28 设置Web站点日志的屏幕

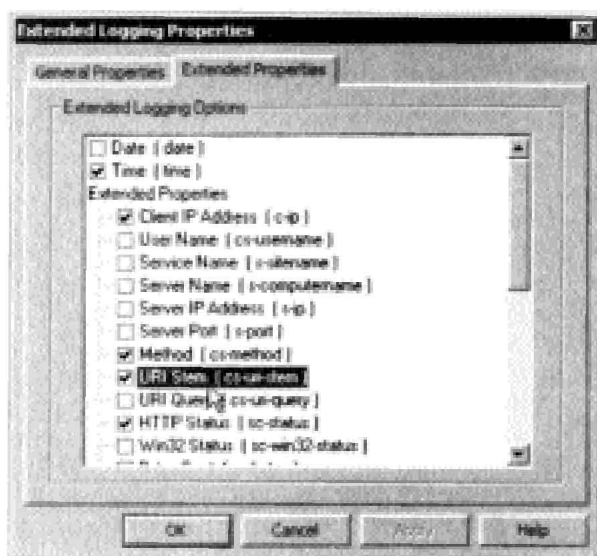


图2-29 选择要放进日志文件的条目

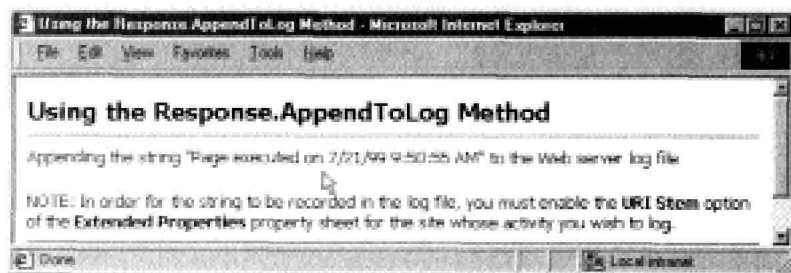


图2-30 执行AppendToLog方法后的屏幕

## 2.6 小结

本章已经开始了对 ASP 3.0 的研究，而且我们也看到了 ASP 3.0 如何与 Internet Information Server 5.0 共同工作，以提供一个易用的、高效的创建动态 Web 网页和 Web 应用程序的方法。当然，仍有一些地方需要去研究，本章仅仅是学习了 ASP 内置的两个最基本的对象。

这两个最基本的对象是 Request 和 Response 对象，允许我们访问和使用作为客户机 / 服务器会话一部分的值，无论用户何时从 Web 网站请求和载入一个网页或资源，这种会话就会进行，意味着 Request 对象能够提供对用户请求的全部内容的访问，同时 Response 对象允许创建和修改服务器发回的响应。

这些对象能够通过集合和属性揭示会话的各个部分，并提供了多个能用来检索和修改各段的方法。假如把它们当作分解用户请求和使用相应内容创建响应的工具，这有助于理解究竟发生了什么。这也将有助于理解各种方法如何影响客户、服务器和正在创建的网页。

本章我们讨论了下列内容：

- 客户和服务端如何相互作用来传递动态页面。
- Request 和 Response 对象的细节以及它们的共同之处。
- 如何访问窗体和查询字符串的值。
- 如何读取和创建保存在客户上的 cookie。
- 什么是服务器变量，如何访问和修改 HTTP 报头。
- 其他一些项目的细节，例如客户机证书。

这些只是一些使网站能够发出动态网页的基本知识，仍然有许多未知的内容需要进一步了解，随着贯穿本书的这些对象的使用，读者将越来越清楚其中的内容，同时将继续探索 ASP 的扩展能力及其精华。下一章将讨论用户会话和应用程序以及相应的两个 ASP 对象。