第2章　ASP的内嵌对象

本章要点：理解ASP的内嵌对象并会使用它们。

通过上面的介绍，我们对ASP已经有了一个基本的了解，如果你想编写ASP应用程序首先应该掌握一种脚本语言，如VBScript，并且熟练掌握ASP的各种内嵌对象和ActiveX组件。这些对象以及组件都可以用来拓展ASP应用程序的功能，实际上，只要掌握了内嵌对象和ActiveX组件，你就可以编写功能强大的ASP应用程序。

一个对象是典型的具有方法、属性或者集合的东西，其中对象的方法决定了我们可以用这个对象做什么事情。对象的属性可以读取，它描述对象状态或者设置对象状态。对象的集合包含了很多和对象有关系的键和值的配对。

举一个日常的例子，本书是一个对象，这个对象包含的方法决定了你可以怎样处理它（即使用它），比如说，去读它，把它送人作为礼物，用它当作敲门的工具，只要你愿意，甚至可以把它撕得粉碎。对象的属性即这本书的页数，作者是谁。对象的集合包含了许多键和值的配对，对书而言，每一页的页码就是键，那么值就是对应于页码的这页的内容。

ASP的内嵌对象和ActiveX组件十分类似，不过，当你使用ASP时，两者之间还是存在着明显的差异。首先，一个组件可能包含不只一个对象，其次，在你使用组件之前，将要明确地创建一个实例。

下面是对每一种内嵌对象的快速浏览：

- Application对象：Application对象用来存储一个应用中所有用户共享的信息。例如，可以利用Application对象在站点的不同用户间传递信息。
- Request对象：Request对象可以用来访问所有从浏览器到服务器间的信息，因此，可以利用Request对象来接受用户在HTML页的窗体中的信息。
- Response对象：Response对象用来将信息发送回浏览器。可以利用Response对象将脚本语言结果输出到浏览器上。
- Server对象：Server对象提供你运用许多服务器端的应用函数。例如，你可以利用Server对象来控制脚本语言在超过时限前的运行时间。也可以利用Server对象来创建其他对象的实例。
- Session对象：Session对象用来存储一些普通用户在滞留期间的信息，可以用Session对象来存储一个用户在访问你的站点时滞留时间。
- ObjectContext对象：ObjectContext对象可以用来控制ASP的执行。这种执行过程由Microsoft Transaction Server(MTS)来进行管理。

内嵌对象不同于正常的对象。在利用内嵌对象的脚本时，不需要首先创建一个它的实例。在整个网站应用中内嵌对象的所有方法、集合以及属性都是自动可访问的。

2.1 Response和Request 对象

Response 对象可以输出信息到客户端。包括直接发送信息给浏览器、重定向浏览器到另一个 URL 或设置 cookie 的值。下面例子演示了该对象的使用方法：

```
<html>
<head><title>Buffer Example</title></head>
<body>
<%
FOR i=1 to 500
  Response.Write(i&"<BR>")
NEXT
%>
</body>
</html>
```

在ASP中引用对象的方法的语法是“对象名.方法名”，如上面脚本中用的Response.Write(i&“<BR>”)语句，其含义为引用 Response对象的Write方法在屏幕上显示从 1~500，每一句命令执行后结果都立即显示，你可以实时地看到执行结果。

Response对象的Write方法将指定的字符串或表达式的结果写到当前的 HTTP 输出。

方法就是嵌入到对象定义中的程序代码，它定义对象怎样去处理信息。使用嵌入的方法，对象便知道如何去执行任务而不用提供额外的指令。

在某些场合下，你也可以利用缓冲区来延缓执行过程，当你利用缓冲区时，直到整个 ASP执行结束才会将结果输出到浏览器上。下面是利用 Response对象的Buffer属性修改后的例子：

```
<% Response.Buffer=True %>
<HTML>
<HEAD><TITLE>Buffer Example</TITLE></HEAD>
<body>
<%
FOR i=1 to 500
  Response.Write(i&"<BR>")
NEXT
%>
</body>
</html>
```

在ASP中设置对象属性的值的方法是“对象名 .属性名 = ”，如上面脚本中的第一行用 <% Response.Buffer=True %>语句将Response对象的Buffer的属性设为 True。这也是和前面那个脚本仅有的一个区别。那么这页执行的时候，这个主页的所有内容会同时显示在浏览器上，这个主页会存在缓存区中直到脚本执行结束。

任何对Buffer属性进行修改的语句都必须在 <HTML>语句和 ASP脚本输出语句的前面，如果在＜HTML＞语句或者脚本输出后面修改 Buffer属性，那么你的程序就会出错。

利用缓存程序可以根据某种条件来显示不同的主页，下面是随机显示的一个例子。

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

```
<HTML>
<HEAD><Title>第一页</title></head>
<Body>
这是第一页。
</body>
</html>
<%
Randomize
注释：初始化随机数生成器。
IF Int(2*rnd)=1 THEN Response.End
Response.Clear
%>
<HTML>
<HEAD><TITLE>第二页</title></head>
<body>
这是第二页。
</body>
</html>
```

在这个例子中，使用了Response对象的两个新的方法：End方法和Clear方法。End方法会立即停止ASP的执行和相应结果。你在执行End方法时不需要考虑是否进行了缓存输出。在这个例子中，End方法被用来防止在显示第一页时第二页也显示出来。

Clear方法是用来在不将缓存中的内容输出的前提下清空当前页的缓存，仅当使用了缓存输出的时候你才可以利用Clear方法，在这里面，Clear方法是用来防止显示第二个页面时第一个页面也显示出来。它把第一个页面从B缓存中清空了。

Response对象中在缓存输出中还用到了另一个方法是Flush，该方法可以将缓存中的内容立即显示出来。该方法有一点和Clear方法一样，它在脚本前面没有将Buffer属性设置为True时会出错。和End方法不同的是，该方法调用后，该ASP可继续执行。

一般情况下，你并不需要缓存输出一个ASP，在大的HTML主页或者运行较长的脚本中利用缓存时，用户的浏览器可能长时间没有反应，这通常会使得用户感到迷惑。

如果希望根据条件显示不同主页内容，完全可以简单地利用VB脚本来做判断，例如，下面这个例子实现了上面例子的同样功能但没有使用缓存。

```
<%
Randomize
'初始化随机数生成器。
IF INT(2*RND)=1 THEN
%>
<HTML>
<HEAD><TITLE>第一页</TITLE></HEAD>
<BODY>
这是第一页
</BODY>
</HTML>
<% ELSE %>
<HTML>
```

```
<HEAD><TITLE>第二页</TITLE></HEAD>
<BODY>
这是第二页
</body>
</html>
<% END IF %>
```

只有一种场合下使用缓存是必须的，那就是在主页内容已经确定后，你却需要改变主页的Header。这时除了将Buffer属性设为True外别无选择。

浏览器的请求和服务端的响应都包含头信息，头信息提供了有关请求和响应的附加信息，也包括了浏览器生成请求和服务端提供响应的过程信息。

ASP包含了相当多的集合和方法来维护 Headers。一部分方法对应于特定的任务，例如提供了站点内容分级以及站点限期时间等功能；其他的方法和集合用于一般的 Headers维护。以下部分详细介绍如何使用这些集合方法来影响 Headers。

当浏览器在服务端接收到一个主页时，这个请求就包括了相当数量的 Headers，你可以利用Request对象的ServerVariable集合来接收这些Headers，ServerVariable集合既包括了Headers也包括了服务端的一些其他信息。

集合存储了相互有关系的字符串、数字、对象和其他值。集合与数组非常相近，但它在存储或取出项目时会自动扩展与搜索。另外，集合被修改后，项目的位置将会移动。可以通过集合中项目的名称、索引访问项目，也可以遍历访问项目。

下面的例子将ServerVariable集合的所有内容名称显示出来。

```
<HTML>
<HEAD><Title>服务端变量</Title></HEAD>
<BODY>
<%
FOR Each name IN Request.ServerVariables
Response.Write("<p><b>"&name&"</b>:")
Response.Write(Request.ServerVariables (name))
NEXT
%>
</BODY>
</HTML>
```

显示结果为：

```
ALL_HTTP:HTTP_ACCEPTimage/gif,image/x-
xbitmap,image/jpeg,image/pjpeg,*/*HTTP_ACCEPT_LANGUAGE:en
 HTTP_CONNECTION:Keep_Alive
 HTTP_HOST:platoHTTP_USER_AGENT:Mozilla/4.01[en](WinNT;I)HTTP_COOKIE:USERID
=HTTP_ACCEPT_CHARSETiso-8859-1*,utf-8
ALL_RAW:Accept:image/gif,image/x-xbitmap,image/jpeg,image/pjpeg,*/* Accept-
Language: en
Connection: Keep-alive Host:plato User-Agent:Mozilla/4.01[en](WinNT;I)
 Cookie:USERID=Accept-Charset:iso-8859-1,*.utf-8
APPL_MD_PATH:/LM/W3SVC/1/ROOT/test/
APPL_PHYSICAL_PATH:/D:\Inetpub\wwwroot
```

```
AUTH_PASSWORD:
AUTH_TYPE:
AUTH_USER:
CERT_COOKIE:
CERT_FLAGS:
CERT_ISSUER:
CERT_KEYSIZE:
CERT_SECRETKEYSIZE:
CERT_SERIALNUMBER:
CERT_ISSUER:
CERT_SUBJECT:
...(显示结果略)
```

可以看出，ServerVariables集合包含了很多类型的信息，下面解释常用的信息类型：

ALL_HTTP：客户端发送的所有 HTTP 头文件。

ALL_RAW：检索未处理表格中所有的头名称。 ALL_RAW 和 ALL_HTTP 不同，ALL_HTTP 在头文件名前面放置 HTTP_ prefix，并且标题名称总是大写的。使用 ALL_RAW 时，头名称和值只在客户端发送时才出现。

APPL_MD_PATH：检索 ISAPI DLL 的 (WAM) Application 的元数据库路径。

APPL_PHYSICAL_PATH：检索与元数据库路径相应的物理路径。IIS 通过将 APPL_MD_PATH 转换为物理（目录）路径以返回值。

AUTH_PASSWORD：该值输入到客户端的鉴定对话中。只有使用基本鉴定时，该变量才可用。

AUTH_TYPE：这是用户访问受保护的脚本时，服务器用于检验用户的验证方法。

AUTH_USER：未被鉴定的用户名。

CERT_COOKIE：客户端验证的唯一 ID，以字符串方式返回。可作为整个客户端验证的签字。

CERT_FLAGS：如有客户端验证，则 bit0 为 1。如果客户端验证的验证人无效（不在服务器承认的 CA 列表中），bit1 被设置为 1。

CERT_ISSUER：用户验证中的颁布者字段（O=MS，OU=IAS，CN=user name，C=USA）。

CERT_KEYSIZE：安全套接字层连接关键字的位数，如 128。

CERT_SECRETKEYSIZE：服务器验证私人关键字的位数。如 1024。

CERT_SERIALNUMBER：用户验证的序列号字段。

CERT_SERVER_ISSUER：服务器验证的颁发者字段。

CERT_SERVER_SUBJECT：服务器验证的主字段。

CERT_SUBJECT：客户端验证的主字段。

CONTENT_LENGTH：客户端发出内容的长度。

CONTENT_TYPE：内容的数据类型。同附加信息的查询一起使用，如 HTTP 查询 GET、POST 和 PUT。

GATEWAY_INTERFACE：服务器使用的 CGI 规格的修订，格式为 CGI/revision。

HTTP_<HeaderName>：HeaderName 存储在头文件中的值。未列入该表的头文件必须以 HTTP_ 作为前缀，以使 ServerVariables 集合检索其值。

注意，服务器将 HeaderName 中的下划线（_）解释为实际头名称中的破折号。例如，如果指定 HTTP_MY_HEADER，服务器将搜索以 MY_HEADER 为名发送的头文件。

HTTPS：如果请求穿过安全通道（SSL），则返回 ON。如果请求来自非安全通道，则返回 OFF。

HTTPS_KEYSIZE：安全套接字层连接关键字的位数，如 128。

HTTPS_SECRETKEYSIZE：服务器验证私人关键字的位数，如 1024。

HTTPS_SERVER_ISSUER：服务器验证的颁发者字段。

HTTPS_SERVER_SUBJECT：服务器验证的主字段。

INSTANCE_ID：文本格式 IIS实例的 ID。如果实例 ID 为 1，则以字符形式出现。使用该变量可以检索请求所属的（元数据库中） Web 服务器实例的 ID。

INSTANCE_META_PATH：响应请求的 IIS 实例的元数据库路径。

LOCAL_ADDR：返回接受请求的服务器地址。如果在绑定多个 IP 地址的多宿主机器上查找请求所使用的地址时，这条变量非常重要。

LOGON_USER：用户登录 Windows NT 的账号。

PATH_INFO：客户端提供的额外路径信息。可以使用这些虚拟路径和 PATH_INFO 服务器变量访问脚本。如果该信息来自 URL，在到达 CGI 脚本前就已经由服务器解码了。

PATH_TRANSLATED：PATH_INFO 转换后的版本，该变量获取路径并进行必要的由虚拟路径至物理路径的映射。

QUERY_STRING：查询 HTTP 请求中问号（?）后的信息。

REMOTE_ADDR：发出请求的远程主机的 IP 地址。

REMOTE_HOST：发出请求的主机名称。如果服务器无此信息，它将设置为空的 MOTE_ADDR 变量。

REMOTE_USER：用户发送的未映射的用户名字符串。该名称是用户实际发送的名称，与服务器上验证过滤器修改过后的名称相对应。

REQUEST_METHOD：该方法用于提出请求。相当于用于 HTTP 的 GET、HEAD、POST 等等。

SCRIPT_NAME：执行脚本的虚拟路径。用于自引用的 URL。

SERVER_NAME：出现在自引用 URL 中的服务器主机名、DNS 化名或 IP 地址。

SERVER_PORT：发送请求的端口号。

SERVER_PORT_SECURE：包含 0 或 1 的字符串。如果安全端口处理了请求，则为 1，否则为 0。

SERVER_PROTOCOL：请求信息协议的名称和修订。格式为 protocol/revision 。

SERVER_SOFTWARE：应答请求并运行网关的服务器软件的名称和版本。格式为 name/version 。

URL：提供 URL 的基本部分。

你可以根据需要利用ServerVariables集合只是去接受特定的头变量，例如下面这个例子，只有从origin.asp进入这个例子，程序才被允许运行：

```
<HTML>
<HEAD><TITLE>服务端变量</TITLE></HEAD>
<BODY>
<%
WhereForm=Request.ServerVariables("HTTP_REFERER")
IF WhereForm="http://www.myserver.com/example/origin.asp" THEN
%>
欢迎进入ASP教程示例
<%
ELSE
%>
您未被授权访问该页！
<%
END IF
%>
</BODY>
</HTML>
```

在这个例子中，HTTP_REFERER 头变量检查用户连接的来源，只有从 http://www.myserver.com/example/origin.asp连接过来的才被允许访问，这样进行授权控制就非常容易。

Cookies集合是Response对象和Request对象的一项经常用到的集合，什么是 cookie呢？实际上，在 HTTP 协议下，cookie仅仅是一个文本文件，cookie是服务器或脚本可以维护用户信息的一种方式。包含用户的有关信息（如身份识别号码、密码、用户在 Web 站点上购物的方式或用户访问该站点的次数）。无论何时用户连接到服务器， Web 站点都可以访问 cookie 信息。

Response对象的Cookies集合用于创建与修改cookie的值，如果指定的cookie不存在，则创建它，如果存在，则修改它。

语法：

```
Response.Cookies(cookie)[(key)|.attribute] = value
```

参数：

cookies 指定cookie的名称

key 为可选参数。如果指定了 key，则 cookie 就是一个字典，而 key 将被设置为 value。

value 指定分配给 key 或 attribute 的值。

attribute指定 cookie 自身的有关信息。attribute 参数可以是下列之一：

名 称	说 明
Domain	若被指定，则 cookie 将被发送到对该域的请求中去
Expires	cookie 的过期日期。为了在会话结束后将 cookie 存储在客户端磁盘上，必须设置该日期。若此项属性的设置未超过当前日期，则在任务结束后 cookie 将到期
HasKeys	指定 cookie 是否包含关键字
Path	若被指定，则 cookie 将只发送到对该路径的请求中。如果未设置该属性，则使用应用程序的路径
Secure	指定 cookie 是否安全

现在，我们就可以用Response对象设置cookie了。

创建一个带有关键字的 cookie，如下列脚本所示：

```
<%
Response.Cookies("mycookie")("type1") = "First"
Response.Cookies("mycookie")("type2") = "Secondly"
%>
```

则此头文件将被发送出去，如下所示：

```
Set-Cookie:MYCOOKIE=TYPE1=sugar&TYPE2=ginger+snap
```

在用户机器的Cookies目录下建立的文本文件如下：

```
myCookie
type1=First&type2=Secondly
```

如果在指定 myCookie 时不指定关键字，将破坏 type1 和 type2。如下面示例所示：

```
<% Response.Cookies("myCookie") = "Third" %>
```

在前面的示例中，关键字 type1 和 type2 被破坏且其值也被删除。myCookie 只有 Third 的值。

在用户机器的Cookies目录下建立的文本文件如下：

```
mycookie
Third
```

而Request对象的Cookies集合读取cookie的值。

语法：

```
Request.Cookies(cookie)[(key)|.attribute]
```

参数：

cookie 指定要检索其值的 cookie。

key 为可选参数，用于从 cookie 字典中检索子关键字的值。

attribute 指定 cookie 自身的有关信息。

可以通过包含一个 key 值来访问 cookie 字典的子关键字。如果访问 cookie 字典时未指定 key，则所有关键字都会作为单个查询字符串返回。例如，如果 MyCookie 有两个关键字，First 和 Second，而在调用 Request.Cookies时并未指定其中任何一个关键字，那么将返回下列字符串：

```
First=firstkeyvalue&Second=secondkeyvalue
```

如果客户端浏览器发送了两个同名的 cookie，那么 Request.Cookie 将返回其中路径结构较深的一个。例如，如果有两个同名的 cookie，但其中一个的路径属性为 /www/ 而另一个为 /www/home/，客户端浏览器同时将两个 cookie 都发送到 /www/home/ 目录中，那么 Request.Cookie 将只返回第二个 cookie。

要确定某个 cookie 是不是 cookie 字典（cookie 有否有关键字），可使用下列脚本：

```
<%= Request.Cookies("myCookie").HasKeys %>
```

如果 myCookie 是一个 cookie 字典，则前面的赋值为 TRUE。否则，为 FALSE。

可以通过循环遍历 Cookies 集合中的所有 cookie 或 cookie 中的所有关键字。但是，通过关

键字在没有关键字的 cookie 上遍历将不产生任何输出。使用 HasKeys 语法先检查一下 cookie 是否有关键字，可以避免这种情况的发生。

Request.Cookies语句必须写在 ASP 文件中的第一个 < HTML> 标记前，这是因为 cookie 是作为 HTTP 传输的头信息的一部分发送给客户的，如果当 HTTP 头信息已经传输给客户后再使用 Response.Cookies，将出现以下错误 :“ HTTP 头信息已经写入到客户浏览器。任何 HTTP 头信息的修改必须在写在页内容之前。”

下面的例子说明如何设置与读取 cookie(见图2-1与图2-2)。

```
<%@  LANGUAGE=VBScript %>
<%
Username=Request.Cookies("UserInfo")
IF Username="" THEN
  Username = Request.Form("Username")
  Response.Cookies("UserInfo") = Username
  Response.Cookies("UserInfo").Expires = "Dec 1, 2000"
  Response.Cookies("Userinfo").Path = "/"
  IF Username <> "" THEN
Response.Write "欢迎你初次光临"
  END IF
ELSE
  Response.Write "欢迎你再次光临"
END IF
%>

<html>
<body>
<p align="center">  </p>
<p align="center">  </p>
<p align="center">  </p>
<Form method="POST" action="register.asp" name=form1 >
  <div align="center">
<table width="45%">
  <tr>
    <td><font size="2">姓名</font></td>
    <td>
      <Input Type="text" name=Username size="15" maxlength="15">
    </td>
  </tr>
    <td>
      <div align="center">
        <Input Type=submit value="提交">
      </div>
    </td>
</table>
  </div>
</Form>
```

```
<p align="center"></p>
</body>
</html>
```

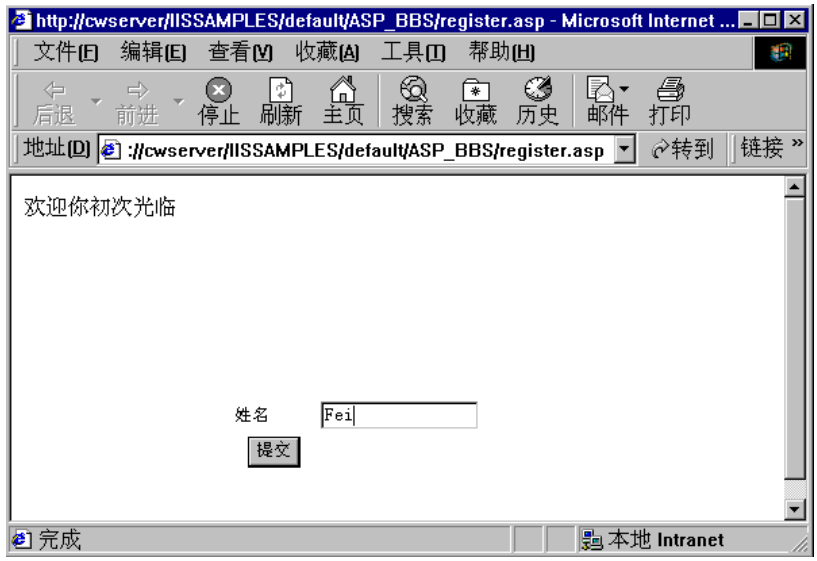

图2-1 使用Cookies集合，初次提交后的结果

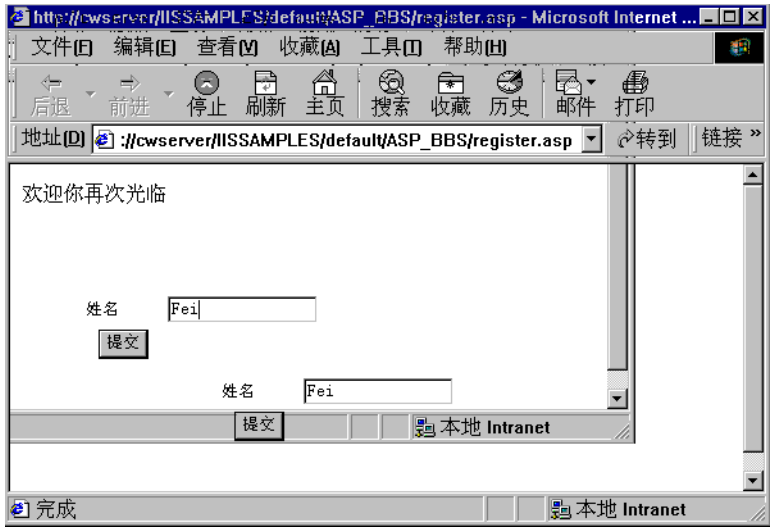

图2-2 使用Cookies集合，再次提交后的结果

在记事本中输入以上代码，保存为 Register.asp，**用**HTTP**的方式访问**Register.asp。
现在让我们来看看上面的代码是如何工作的：

```
Username=Request.Cookies("UserInfo")
```

用Request对象读取cookie的值，如果存在，赋给变量 Username。并显示“欢迎你再次光临”。

```
IF Username="" THEN
  如果这个Cookie不存在，则创建它，并显示"欢迎你初次光临"。
  Username = Request.Form("Username")
  Response.Cookies("UserInfo") = Username
  Response.Cookies("UserInfo").Expires = "Dec 1, 2000"
  Response.Cookies("Userinfo").Path = "/"
  IF Username <> "" THEN
     Response.Write欢迎你初次光临"
  END IF
ELSE
     Response.Write欢迎你再次光临"
END IF
```

尽管上面的这个例子很简单，但可以从中扩展许多富有创造力的应用程序。你可以在表单中加入许多功能，以便定制化 Web站点。你还可以让访问者定制网站的色彩、字体，以及其他Web元素。有可能的话，你可以询问访问者的生日，当访问者在那一天来访时，你就可以显示”生日快乐”的信息给他。

在上面的例子中，我们看到了对 Request对象的新的引用：

```
Username = Request.Form("Username")
```

当需要在多个 ASP主页中传递消息的时候， Request的Form方法是一个你可以选择的方法。它通过使用 POST 方法的表格检索传送到 HTTP 请求正文中的表格元素的值。它的具体语法是：

```
Request.Form(element)[(index)|.Count]
```

参数：

element 指定集合要检索的表格元素的名称。

index 可选参数，使用该参数可以访问某参数中多个值中的一个。它可以是 1 到 Request.Form (parameter).Count 之间的任意整数。

Count 集合中元素的个数

Form 集合按请求正文中参数的名称来索引。 Request.Form(element) 的值是请求正文中所有 element 值的数组。通过调用 Request.Form(element).Count 来确定参数中值的个数。如果参数未关联多个值，则计数为 1。如果找不到参数，计数为 0。要引用有多个值的表格元素中的单个值，必须指定 index 值。index 参数可以是从 1 到Request.Form(element).Count 中的任意数字。如果引用多个表格参数中的一个，而未指定 index 值，返回的数据将是以逗号分隔的字符串。

可以使用重述符来显示表格请求中的所有数据值。在下例中，用户通过指定的几个值填写表格（结果参见图2-3与图2-4），代码如下：

```
<Html>
<Head><Title>爱好</Title></Head>

<Body>
<H2>个人爱好情况</H2>
```

```
<P>
请提供以下信息，然后单击"提交":
<Form Method="POST" Action="form.asp">
<P>
<Input Type="checkbox" name="title" value="足球"> 足球 <Input Type="checkbox"
name="title" value="乒乓球 "> 乒乓球 <Input Type="checkbox" name="title" value="
网球 "> 网球 <Input Type="checkbox" name="title" value="篮球"> 蓝球 <Input
Type="checkbox" name="title" value="羽毛球 "> 羽毛球
<P><Input Type="checkbox" name="title" value="排球"> 排球 <Input Type="checkbox"
name="title" value="棒球 "> 棒球 <Input Type="checkbox" name="title" value="冰球"
"> 冰球 <Input Type="checkbox" name="title" value="壁球"> 壁球
<P><Input Type=SUBMIT value="提交"><Input Type=RESET value="重置">
</Form>
</Body>

</Html>
```

在记事本中输入以上代码，保存为form.htm。

```
<%@ LANGUAGE = "VBScript" %>
<%
FOR Each item In Request.Form("title")
  Response.Write item & "<BR>"
NEXT
%>
```

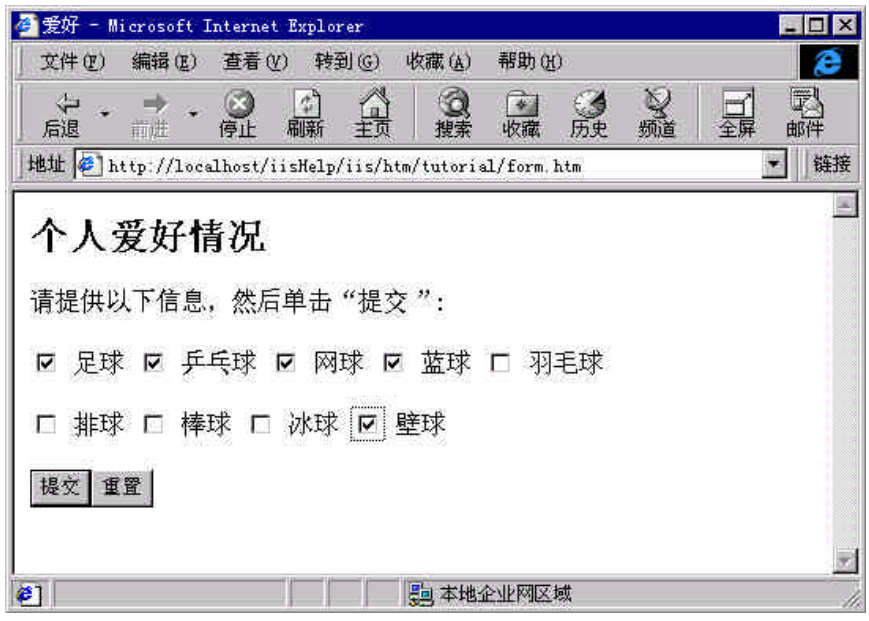

图2-3 供选择的表单

在记事本中输入以上代码，保存为 form.asp。把两个文件拷贝到同一个目录，用 HTTP的方式打开form.htm浏览，观察单击“提交”按钮后的浏览器输出。

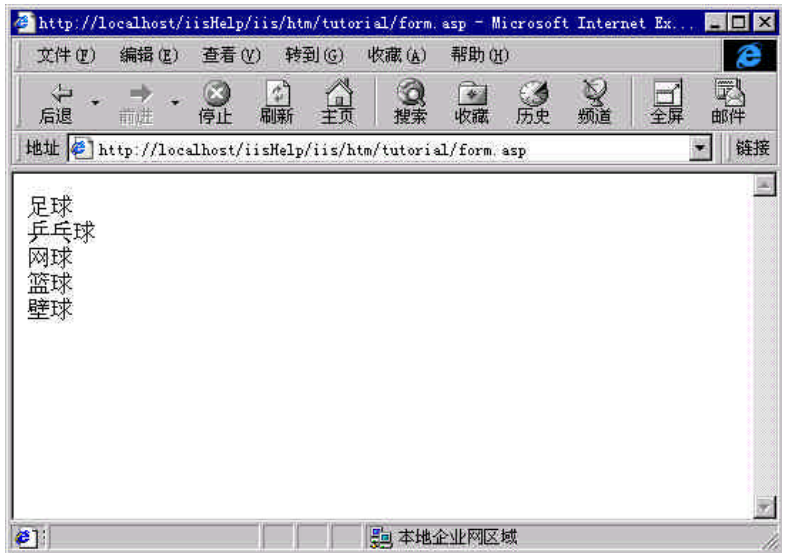

图2-4 用Request得到Form集合的内容

相信看到这里的读者应当十分熟悉HTML，显示结果应该能想象到了（如果不是的话，请先看本书的附录关于HTML的内容或者其他有关制作主页的入门书籍）。在这里，主页的介绍就不再多写了。请注意这里面的Form Method="POST"，也就是说，文本框内容在“提交”按钮点击确认后会传送到服务端，由于Form的内容将作为HTTP请求的一部分，那么就是说可以用ASP的Request对象来处理了。针对这种目的，ASP的Request对象特别指定了一个Form集合来进行相关处理。Form集合就可以包含所有添加HTML Form的信息。

在Request对象中，还有一个集合实现类似的功能，它就是QueryString。我们经常可以看见类似这样的网络地址：“http://www.stockt.com/query.asp?stockCode=0660”。

问号(?)后面的值就是HTTP查询字符串，而QueryString集合检索HTTP查询字符串中变量的值。通过发送表格数据或由用户在其浏览器的地址框中键入查询都可以生成类似的查询字符串。它和Form集合的区别是用QueryString集合检索HTTP查询字符串中变量的值时，变量和它的值是可见的。也就是说，任何一个路过的人都可以看见由QueryString集合接收的任何变量的值，这就意味着如果用它来传递用户密码是很危险的。

如果传递的数据量比较大，用Query字段也不好办。对于不同的浏览器，这种信息传递的大小是有不同的限制的，例如IE 4.0无法处理超过2000个字符的Query字段。如果URL连接带Query字段超过这个长度，浏览器就无法正确处理。有时无法确认字段长度具体有多少，因为实际上对于浏览器限制的长度是指浏览器地址栏中所能显示并发送的最大长度，也就是URL地址和Query字段的总长度。

再有，很多浏览器所限定的长度还远远达不到IE 4.0的2000个字符的限制，尤其是早期的版本，因此，一般说来，如果传递数据量比较大，就不要使用这种方式了。

所以，利用Query字段传递的信息都应当是简洁的和非保密的，如果有大量数据需要传递，那么可以利用HTTP的Form中的hidden类型，详细介绍请参考有关HTML的介绍。在这里值得解

释的是，HTTP协议在传递Form时比传递Query字段有效得多。QueryString集合的具体语法是：

```
Request.QueryString(variable)[(index)|.Count]
```

参数：

variable：在 HTTP 查询字符串中指定要检索的变量名。

index：这是一个可选参数，可以用来检索 variable 的多个值中的某一个值。可以是从 1 到 Request.QueryString(variable).Count 之间的任何整数。

QueryString集合可以让你用名称检索环境变量QUERY_STRING的值，QUERY_STRING 包含查询HTTP请求中问号（?）后的信息。Request.QueryString (参数) 的值是出现在 QUERY_STRING 中所有参数值的数组。通过调用 Request.QueryString(parameter).Count 可以确定参数有多少个值。如果变量未关联多个数据集，则计数为 1。如果找不到变量，计数为 0。

可以使用循环在查询字符串中遍历所有的数据值。例如，如果发送以下的请求：

```
http://Names.asp?NAME=Fred&NAME=Sally
```

而且 Names.asp 包含下面的脚本：

```
<%
FOR Each item In Request.QueryString("NAME")
  Response.Write item & "<BR>"
NEXT
%>
```

Names.asp将显示如下：

```
Fred
Sally
```

上述脚本也可以用 Count 来写。

```
<%
FOR I = 1 To Request.QueryString("NAME").Count
  Response.Write Request.QueryString("NAME")(I) & "<BR>"
NEXT
%>
```

还有，上面关于Form集合的例子Form.asp可以改写成：

```
<%@ LANGUAGE = "VBScript" %>
<%
FOR Each item In Request.QueryString("title")
  Response.Write item & "<BR>"
NEXT
%>
```

可以得到相同的结果。

对象Response的另一个常用的方法是Redirect 方法。在很多场合下，你需要引导用户到另一个主页上，例如，用户注册单没有填写完全或填写错误就进行了提交，那么提交程序会自动将用户返回到注册页等等。Redirect 方法使浏览器立即重定向到程序指定的 URL。这也是一个我们经常用到方法，这样，程序员就可以根据客户的不同响应，为不同的客户指定不同的页面，或根据不同的情况指定不同的页面。一旦使用了 Redirect 方法，任何在页中显式设置的响应正文内容都将被忽略。然而，此方法不向客户端发送该页设置的其他 HTTP 头信息，而是产生一个将重定向 URL 作为链接包含的自动响应正文。Redirect 方法发送下列显式头文件，其中 URL

是传递给该方法的值。如：

```
<%
IF Request.Form("UserName")="" THEN Response.Redirect "register.asp"
%>
<HTML>
<HEAD><TITLE>注册提交结果</TITLE></HEAD>
<BODY>
谢谢您、<%=Request.Form("UserName")%>注册教程示例。
</BODY>
</HTML>
```

假设一个用户没有填写姓名就提交了注册表，那么第一个脚本的判断就会将用户重新引导回注册页面。

必须在浏览器显示任何文本前使用 Response.Redirect方法，最好是在<HTML>标记之前进行使用。

可以利用 Response.Redirect方法来指向任何合法的网址，该网址可以是 HTML主页或ASP页面等，可以处于当前服务器，也可以处于局域网甚至广域网上的任何一台自带操作系统和 HTTP Server的服务器。

不过，现在这种操作并不顺利，老版本的浏览器通常无法处理这个方法，更糟的是，新的浏览器（诸如NetScape4.0）也无法自动响应，而是会显示如下信息：

```
Object Moved
This Object may be found here
用鼠标单击here后，转到相应页面。
```

这显然是站点制作人所不愿意看到的，用户也会很不习惯。所以我们可以把要转到的页面包含进当前文件。

利用服务端的INCLUDE命令可以很容易地在ASP中包含其他文件。这种服务端的 INCLUDE命令不需要在脚本中实现，它完全可以作为 HTML代码的一部分，例如：

```
<HTML>
<HEAD><TITLE>欢迎</TITLE></HEAD>
<BODY>
<!--#INCLUDE VIRTUAL="mybanner.inc" -->
欢迎进入文件包容示例程序
</BODY>
</HTML>
```

在这个例子中。文件 mybanner.inc将被插入到这个 ASP文件<BODY>标记下面，当这个 ASP文件执行时，在 mybanner.inc中的HTML代码以及脚本也将在相应位置执行或出现（其实和子程序差不多，只不过没有参数传递）。这种文件包含有两种路径，虚拟路径或者物理路径。下面是包含物理路径的示例：

```
<HTML>
<HEAD><TITLE>欢迎</TITLE></HEAD>
<BODY>
<!--#INCLUDE FILE="mybanner.inc" -->
```

```
欢迎进入文件包容示例程序
</BODY>
</HTML>
```

如果你使用FILE命令来指定物理路径，那么文件必须在当前目录或者子目录下面。本例中，文件在当前目录下，这样限制就比较多，因此，一般你应当使用虚拟路径命令。

这种文件对于任何扩展名都是有效的，一般来说，习惯上用 .inc扩展名，但是还可以使用.asp、.htm、.html或者任何其他扩展名。

当改变了包含的文件时，并不一定马上就看到效果，那是因为 IIS的缓存作用，IIS在对正常文件变动的响应比包含文件变动的响应要快。在这种场合下，你有两种方法处理。一种是在Internet Service Manager中重新启动一下Server。另一种是将包含其他文件的母文件也做一下改动，例如添加一个无意义的空格，那么 IIS就会意识到这种变动并且作出正确响应，不过，如果包含变动文件的母文件较多，显然这就成了耗时耗力的工作。

有两种情况需要包含其他文件，一种是有一些十分通用的常用程序段或者脚本，不需要每个ASP文件都写一遍，那么完全可以在每个需要的 ASP文件中进行包含就行了。

另外一种是进行判断后根据条件重定向的场合，例如上面的例子，也完全可以利用INCLUDE语句实现。请看下面这段例子：

```
<%
IF Request.Form("Username")="" THEN
%>
<!--#INCLUDE VIRTUAL="regester.asp">
<%
Response.End
END IF
%>
<HTML>
<HEAD><TITLE>注册结果</TITLE></HEAD>
<BODY>
谢谢您、<%=Request.Form("UserName")%>注册网络教程示例。
</BODY>
</HTML>
```

这个例子说明它可以和Response.Redirect方法具有同样的作用，当用户没有填写姓名时，会被引导回注册主页，不过，由于这个语句的执行完全在服务器端完成，因此基本上不存在浏览器的兼容性问题。

再次提请注意Response.End的调用，这是防止注册表调用后欢迎页跟着显示出来。还有很重要的一点是，在 IIS中对INCLUDE语句的执行是优先于脚本执行的，也就是说，如果把INCLUDE语句放在脚本外面并等待脚本结果来传递参数，那么将是行不通的。例如下面这个例子：

```
<%
IF Request.Form("Username")="" THEN
  Myinlcude="register.asp"
ELSE
```

```
  Myinclude="Homepage.asp"
END IF
%>
<!--#INCLUDE VIRTUAL="<%=MyInclude%>" -->
```

这个脚本是错误的，因为在 IIS中先执行 INCLUDE语句，后执行 VB脚本。这样，文件"<%=MyInclude%>"自然找不到。

2.1.1 Response对象的属性

1. ContentType属性

ContentType属性指定响应的 HTTP 内容类型。如果未指定 ContentType，其默认值为text/HTML。

2. Charset属性

Charset 属性将字符集名称附加到 Response 对象中 content-type 头信息的后面。对于不包含 Response.Charset 属性的 ASP 页，content-type 头信息将为 :content-type:text/html。 我们可以在 .asp 文件中指定 content-type 头信息，如：

```
< % Response.Charset="gb2312") %>
```

将产生以下结果：

```
content-type:text/html; charset=gb2312
```

无论字符串表示的字符集是否有效，该功能都会将其插入 content-type 头信息中。且如果某个页包含多个含有 Response.Charset 的标记，则每个 Response.Charset 都将替代前一个 CharsetName。这样，字符集将被设置为该页中 Response.Charset 的最后一个实例所指定的值。

3. Expires 属性

Expires属性指定了在浏览器上缓冲存储的页距过期还有多少时间。如果用户在某个页过期之前又回到此页，就会显示缓冲区中的版本。如设置 response.expires=0，则可使缓存的页面立即过期。这是一个较实用的属性，当客户通过 ASP 的登陆页面进入 Web 站点后，应该利用该属性使登陆页面立即过期，以确保安全。

4. ExpiresAbsolute 属性

与 Expires 属性不同，ExpiresAbsolute 属性指定缓存于浏览器中页面的确切到期日期和时间。在未到期之前，若用户返回到该页，该缓存中的页面就显示。如果未指定时间，该主页在当天午夜到期。如果未指定日期，则该主页在脚本运行当天的指定时间到期。如下示例指定页面在 2000 年 7 月 31 日上午 9:00 分 30 秒到期。

```
<% Response.ExpiresAbsolute=#Jul 31,2000 9:00:30# %>
```

2.1.2 Request对象的属性

1. TotalBytes 属性

得到客户端在请求正文中发送的总字节数。该属性为只读。

语法：

```
Counter = Request.TotalBytes
```

参数：

2. Counter属性

指定一个变量来接收客户端在请求中发送的总字节数。

示例：以下脚本设置一个等于请求对象中包括的总字节数的变量。

```
<%
Dim bytecount
bytecount = Request.TotalBytes
%>
```

2.1.3 Request对象的方法

Request对象包括BinaryRead 方法，该方法获取作为 POST 请求的一部分而从客户端传送到服务器的数据。此方法获取来自客户端的数据并将其储存在 SafeArray 中。SafeArray 是一个数组，其中包含维数和边界信息。

语法：

```
variant = Request.BinaryRead(count)
```

参数：

variant：包含由该方法返回的无符号数的数组。该参数的类型为 VT_ARRAY | VT_UI1。

count：在执行前，指定要从客户端读取的字节数。此方法返回后， count 将包含从客户机成功读取的字节数。实际读取的字节总数将小于或等于 Request.TotalBytes。

BinaryRead 方法用于读取作为 POST 请求的一部分从客户端发出的未加工数据。此方法用于在底层访问数据。与此相反， Request.Form 集合用于查看在公告请求中发送的表格数据。一旦调用了 BinaryRead，则引用 Request.Form 集合中的任何变量都将导致错误发生。反之，一旦引用了 Request.Form 集合中的一个变量，则调用 BinaryWrite 也将导致错误发生。请记住，如果在访问 Request 集合中的变量时未指定该变量属于哪一个子集，将搜索 Request.Form 集合并强制使用上述规则。

下列示例使用 BinaryRead 方法将请求的内容放入一个安全的数组中。

```
<%
Dim binread
Dim bytecount
bytecount = Request.TotalBytes
binread = Request.BinaryRead(bytecount)
%>
```

2.2 Application 对象

微软希望用户将ASP作为一种常规的编程语言。当你创建了一个 ASP页的时候，你就创建了一个类似子程序这样的东西。当你创建 了一组ASP页，那么你就是创建了一个 Application 对象。

因此，一个Application对象就是在硬盘上的一组主页以及ASP文件，当一个ASP 加入了一个application对象，那么它就拥有了作为单独主页所无法拥有的属性。下面是 ASP的Application对象的一些特性：

- 数据可以在Application对象内部共享，因此一个Application对象可以覆盖多个用户。
- 一个Application对象包含事件可以触发某些Applicatin对象脚本。
- 一个对象的例子可以被整个Application对象共享。
- 个别的Application对象可以用Internet Service Manager来设置而获得不同属性。
- 单独的Application对象可以隔离出来在他们自己的内存中运行，这就是说，如果一个人的Application遭到破坏，不会影响其他人。
- 可以停止一个Application对象（将其所有组件从内存中驱除）而不会影响到其他应用程序。

一个网站可以有不止一个Application对象。典型情况下，你可以针对个别任务的一些ASP文件创建个别的Application对象。例如，可以一个Application对象来适用于全部公用用户，而再创建另外一个Application对象只适用于网络管理员。

Application 对象使给定应用程序的所有用户之间共享信息，并且在服务器运行期间持久地保存数据。因为多个用户可以共享一个 Application 对象，所以必须要有 Lock 和 Unlock 方法以确保多个用户无法同时改变某一属性。

语法：

```
Application.method
```

2.2.1 集合

1. Contents集合

Contents 是由所有通过脚本命令添加到应用程序的项目组成的集合。可以使用 Contents 集合获取给定的应用程序作用域的项目的列表或设置给定应用程序的属性。

语法：

```
Application.Contents(Key)
```

参数：

Key：指定要获取的项目的名称。

Application.Contents 集合包含了用 Server.CreateObject 创建的对象和通过 Application对象声明建立的数值变量。例如，在下面的脚本中， MyVar 和 MyObj 将成为 Application.Contents 集合的成员：

```
<%
   Application("MyVar") = "Hello"
   Set Application("MyObj") = Server.CreateObject("MyComponent")
%>
```

Application.Contents 集合支持 FOR...EACH 和 FOR...NEXT 循环。下列两个脚本说明遍历 Application.Contents 集合的每种方法。

```
<%
FOR EACH Key in Application.Contents
  Response.Write ("Key")
NEXT Key
%>

<%
FOR i = 1 to Application.Contents.Count
  Response.Write ("Key")
NEXT Key
%>
```

2. StaticObjects 集合

StaticObjects 集合包含所有的在 Application 对象范围中使用<OBJECT> 标记创立的对象。可以使用该集合确定某对象的指定属性的值或遍历集合及检索所有静态对象的所有属性。

语法：

```
Application.StaticObjects(Key)
```

参数：

Key：指定要检索的项目的值。

使用循环控制结构可以遍历 StaticObjects 集合中的关键字。请看下面的示例。

```
<%
DIM ObjProp
FOR Each ObjProp in Application.StaticObjects
  Response.Write(ObjProperty & " : " & Application.StaticObjects(ObjProp) &
"<BR>")
NEXT ObjProp
%>
```

2.2.2 方法

1. Lock方法

Lock 方法可以阻止其他客户修改存储在 Application 对象中的变量，以确保在同一时刻仅有一个客户可修改和存取 Application 变量。如果用户没有明确调用 Unlock 方法，则服务器将在 .asp 文件结束或超时后即解除对 Application 对象的锁定。

语法：

```
Application.Lock
```

2. Unlock方法

Unlock 方法可以使其他客户端在使用 Lock 方法锁住 Application 对象后，修改存储在该对象中的变量。如果未显式调用该方法， Web 服务器将在 .asp 文件结束或超时后解锁 Application 对象。

语法：

```
Application.Unlock
```

示例：

```
<%
Application("NumVisits")=0
Application.Lock
Application("NumVisits") = Application("NumVisits") + 1
Application.Unlock
%>

这个应用程序网页已被访问了
<%= Application("NumVisits") %>次!
```

在前面的示例中，Lock **方法保护变量** NumVisits **在同一时刻不被多个客户所访问。** Unlock **方法解除对象的锁定，使得下一个客户端能够增加** NumVisits**。如果应用程序未被锁定，则两个客户就可以同时增加变量** NumVisits **的值。**

这样，我们用 Application**对象轻易的建立了一个简单的计数器。**

2.2.3 事件

1. Application_OnEnd**事件**

Application_OnEnd **事件在应用程序退出时于** Session_OnEnd **事件之后发生，只有** Application **和** Server **内嵌对象可用。**

语法：

```
<SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server> Sub Application_OnEnd. . . End
</SCRIPT>
```

参数：

```
ScriptLanguage
```

指定用于编写事件脚本的脚本编写语言。可以是任何一种支持脚本的语言，例如 VBScript **或** JScript**。**

2. Application_OnStart**事件**

Application_OnStart **事件在首次创建新的会话（即** Session_OnStart **事件）之前发生。只有** Application **和** Server **内嵌对象是可用的。在** Application_OnStart **事件脚本中引用** Session**、**Request **或** Response **对象将导致错误。**

语法：

```
<SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server> Sub Application_OnStart. . . End
Sub
</SCRIPT>
```

参数：

```
ScriptLanguage
```

指定用于编写事件脚本的脚本语言。它可以是任何支持脚本编写的语言，例如 VBScript **或** JScript**。**

一个Application**对象的**OnStart**事件肯定发生在**Session_Start**事件之前。不过，**Application**对**

象不会像Session对象那样在一个新用户请求后就触发， Application对象只触发一次，即第一个用户的第一次请求。

Application_OnEnd事件只有在服务终止或者该 Application对象卸载时才会触发，例如，在Internet Service Manager中关闭了网络服务，那么Application_OnEnd事件就会触发，如果针对单独目的使用Application，这个事件可以通过Application在利用Unload按钮卸载时进行触发。一个Application_OnEnd事件肯定发生在Session_OnEnd事件之后。

Application_OnStart和Application_OnEnd事件都是触发唯一一个脚本程序。而且这些事件都在一个文件中，即Global.asa文件。

应该注意的是，由于 Application对象是多用户共享的，因此它与 Session对象有着本质的区别。同时Application对象不会因为某一个甚至全部用户离开就消失，一旦建立了 Application对象，那么它就会一直存在到网站关闭或者这个 Application对象被卸载。这经常可能是几周或者几个月的时间。

由于Application对象创建后不会自己消亡，因此，就要特别小心的使用，它会占用内存，要斟酌使用以免降低服务器对其他工作的响应速度。 Application对象终止的方法有三种：服务被终止、Global.asa被改变或者该Application对象被卸载。

2.3 Session 对象

Session的发明是填补HTTP协议的局限，HTTP协议工作过程是，用户发出请求，服务器端作出响应，这种用户端和服务器端之间的联系都是离散的，非连续的。在 HTTP协议中没有什么能够允许服务器端来跟踪用户请求。在服务器端完成响应用户请求后，服务器端不能持续与该浏览器保持连接。从网站的观点上看，每一个新的请求都是单独存在的，因此，当用户在多个主页间转换时，你就根本无法知道他的身份。

可以使用 Session 对象存储特定用户会话所需的信息。这样，当用户在应用程序的 Web 页之间跳转时，存储在 Session 对象中的变量将不会丢失，而是在整个用户会话中一直存在下去。

当用户请求来自应用程序的 Web 页时，如果该用户还没有会话，则 Web 服务器将自动创建一个 Session 对象。当会话过期或被放弃后，服务器将终止该会话。

当用户第一次请求给定的应用程序中的 .asp 文件时， ASP 将生成一个 SessionID。SessionID 是由一个复杂算法生成的号码，它唯一标识每个用户会话。在新会话开始时，服务器将 Session ID 作为一个 cookie 存储在用户的 Web 浏览器中。

在将 SessionID cookie 存储于用户的浏览器之后，即使用户请求了另一个 .asp 文件，或请求了运行在另一个应用程序中的 .asp 文件，ASP 仍会重用该 cookie 跟踪会话。与此相似，如果用户故意放弃会话或让会话超时，然后再请求另一个 .asp 文件，那么 ASP 将以同一个 cookie 开始新的会话。只有当服务器管理员重新启动服务器或用户重新启动 Web 浏览器时，此时存储在内存中的 SessionID 设置将被清除,用户将会获得新的 SessionID cookie。

通过重用 SessionID cookie，ASP 将发送给用户浏览器的 cookie 数量降为最低。另外，如果您决定您的 ASP 应用程序不需要会话管理，就可以不让 ASP 跟踪会话和向用户发送 SessionID 。

Session 对象最常见的一个用法就是存储用户的首选项。例如，如果用户指明不喜欢查看图

形，另外其还经常被用在鉴别客户身份的程序中。要注意的是，会话状态仅在支持 cookie 的浏览器中保留，如果客户关闭了 Cookies 选项，Session 也就不能发挥作用了。

ASP的Sessions非常好用，能够利用 Session对象来对Session全面控制，如果需要在一个用户Session中存储信息，只需要简单地直接调用 Session对象就可以了，下面是个例子：

```
<%
Session("Myname")=Response.form("Username")
Session("Mycompany")=Response.form("Usercompany")
%>
```

当ASP执行时，浏览器上什么都不会显示，但脚本第一行是给 Myname赋值为表单”Username”的值，第二行给Mycompany赋值为表单”Usercompany”的值。

当一个同样的用户进入另一个主页，例如，下面这个 ASP页：

```
<HTML>
<HEAD><TITLE>另一页</TITLE></HEAD>
<%=Session("Myname")%>
<%=Session("Mycompany")%>
</Body>
</html>
```

当该用户进入这页，他在表单 Username和表单Usercompany中的赋值就显示出来了，注意这一页没有赋值操作，Myname和Mycompany变量的值是前面那页赋值的。

你无法用普通的脚本变量来进行这种处理，因为一般的变量只在一个单独主页内有效，而Session变量在用户离开网站前一直存在生效。

应注意的是Session对象是与特定用户相联系的。针对某一个用户赋值的 Session对象是和其他用户的Session对象完全独立的，不会相互影响。换句话说，这里面针对每一个用户保存的信息是每一个用户自己独享的，不会产生共享情况。

很明显，对于不同的用户，Session对象的Myname变量和Mycompany变量各自是不同的，当每个人在网站的不同主页间浏览时，这种针对个人的变量会一直保留，这样作为身份认证是十分有效的。

2.3.1 集合

1. Contents集合

Session.contents 集合包括所有未使用 <OBJECT> 标记而为该会话建立的项目。此集合可用于确定指定会话项的值或遍历集合并检索出会话中所有项的列表。

语法：

```
Session.Contents( Key )
```

参数：

Key：要获取的属性的名称。

几乎所有Session对象存储的内容都在Contents集合中。例如，下面两个语句是等效的：

```
<% Session("MyVar")=些数据"  %>
```

```
<% Session.Contents("MyVar")=些数据"  %>
```

可以使用一个循环控制结构通过 Contents 集合的关键字来循环。下面的示例演示这一过程：

```
<%
Dim Sessitem
FOR EACH sessitem in Session.Contents
  Response.Write(sessitem & " : " & Session.Contents(sessitem) & "<BR>")
NEXT
%>
```

2. StaticObjects集合

StaticObjects 集合包含 Session 对象范围中用 <OBJECT> 标记创建的所有对象。该集合可用于确定对象特定属性的值，或用于遍历集合并获取所有对象的全部属性。

语法：

```
Session.StaticObjects( Key )
```

参数：

Key：要检索的属性。

使用循环控制结构可以遍历 StaticObjects 集合中的关键字。请看下面的示例：

```
<%
DIM objprop
FOR EACH objprop in Session.StaticObjects
  Response.Write(objproperty & " : " & Session.StaticObjects(objprop) & "<BR
NEXT
%>
```

2.3.2 属性

1. SessionID属性

SessionID 属性返回用户的会话标识符。在创建会话时，服务器会为每一个会话生成一个单独的标识符。会话标识符以长整形数据类型返回。在很多情况下， SessionID 可以用于 Web 页面注册统计。

不要用 SessionID 属性为数据库应用程序创建主关键字。这是因为，如果 Web 服务器重新启动，则部分 SessionID 的值可能同服务器终止前产生的值相同。可以使用自动增加的列数据类型来代替，如 Microsoft SQL Server 中的 IDENTITY，或 Microsoft Access 中的 COUNTER 。

2. Timeout属性

服务器怎么知道一个 Session对象结束了呢？换句话说，怎样知道用户是否已经离开了这个站点而去了另一个站点或者已经关掉电脑了呢。如果一个人一直没有提出请求或者刷新主页长达20分钟，那么服务器就默认为用户已经离开了。这种策略就使得服务端可以释放对用户进程进行跟踪时使用的资源。

对于有些网络站点， 20分钟显然有些短，例如，对于高水平选手进行的网络围棋，很多步子是要考虑很长时间的。那么这时候 20分钟如果释放了资源，这个棋手就可能被服务器端

轰出局。

有些网络站点则相反，资源有限而访问量又很大，没有什么需要耗费时间的信息传递，那么白白浪费资源是很可惜的，也会使其他访问者的访问速度受到影响。

不过，对于ASP来说，对这些进行控制都没什么难度， Session对象有这种Timeout属性，完全可以用Session对象限定时间。例如：下面这个脚本将限制时间设为 60分钟：

```
<% Session.Timeout=60 %>
```

2.3.3 方法

Session对象的方法有Abandon方法，该方法可删除所有存储在 Session 对象中的对象，并释放这些对象的资源。如果未明确地调用 Abandon 方法，一旦会话超时，服务器将删除这些对象。

Abandon 方法被调用时，将按序删除当前的 Session 对象，不过在当前页中所有脚本命令都处理完后，对象才会被真正删除。这就是说，在调用 Abandon 时，可以在当前页上访问存储在 Session 对象中的变量，但在随后的 Web 页上不行。

当服务器处理完当前页时，下面示例将释放 Session对象资源：

```
<% Session.Abandon %>
```

当用户的Session时间过期后，如果用户刷新了主页，那么该用户将被认为是新的访问者，所有以前的Session信息会全部失去。也可以利用 Abandon方法来撤消一个 Session。这里再引入一个SessionID属性，该属性将自动为每一个 Sessioin分配不同的编号。

```
<HTM>
<HEAD><TITLE>Abandon Session</TITLE></HEAD>
<BODY>
<BR>这个用户自动编号为<%=SessionID %>
<% Session.Abandon %>
<BR>这个用户自动编号为<%=SessionID %>
</BODY>
</HTML>
```

这个例子的显示结果为：

这个用户自动编号为 1520692

这个用户自动编号为 1520693

要说明的是，对于一个 Session对象来说，无论用户怎样进行主页间切换，只能有一个SessionID，但是由于这里面使用了 Session.Abandon，那么就使得这个主页开辟了一个 Session后随即消除，然后又开辟了一个，对于服务器端来说，是两个不同的 Session对象，其中前面的已经关闭，后面的仍然保持。

2.3.4 事件

1. Session_OnEnd事件

Session_OnEnd 事件在会话被放弃或超时发生。在服务器内嵌对象中，只有 Application、Server 和 Session 对象可用。

语法：

```
<SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server> Sub Session_OnEnd. . . End Sub
</SCRIPT>
```

参数：

ScriptLanguage

指定用于编写事件脚本的脚本编写语言。可以是任一种支持脚本编写的语言，例如 VBScript **或** JScript**。**

2. Session_OnStart**事件**

Session_OnStart **事件在服务器创建新会话时发生。服务器在执行请求的页之前先处理该脚本。**Session_OnStart **事件是设置会话期变量的最佳时机，因为在访问任何页之前都会先设置它们。**

尽管 Session_OnStart **事件在包含** Redirect **或** End **方法调用的情况下** Session **对象仍会保持，然而服务器将停止处理** Global.asa **文件并触发** Session_OnStart **事件的文件中的脚本。**

为了确保用户在打开某个特定的 Web **页时始终启动一个会话，可以在** Session_OnStart **事件中调用** Redirect **方法。当用户进入应用程序时，服务器将为用户创建一个会话并处理** Session_OnStart **事件脚本。可以将脚本包含在该事件中以便检查用户打开的页是不是启动页，如果不是，就指示用户调用** Response.Redirect **方法启动网页。程序如下**：

```
< SCRIPT RUNAT=Server Language=VBScript>
Sub Session_OnStart
startPage = "/MyApp/StartHere.asp"
currentPage = Request.ServerVariables("SCRIPT_NAME")
IF strcomp(currentPage,startPage,1) THEN
Response.Redirect(startPage)
END IF
END SUB
< /SCRIPT>
```

上述程序只能在支持 cookie **的浏览器中运行。因为不支持** cookie **的浏览器不能返回** SessionID cookie**。所以，每当用户请求** Web **页时，服务器都会创建一个新会话。这样，对于每个请求服务器都将处理** Session_OnStart **脚本并将用户重定向到启动页中。**

Session_OnStart **事件和**Session_OnEnd**事件的脚本位于特定的文件** Global.asa**中。这个文件位于网站应用的根目录。它包括了一些通用程序段和网站应用。** Global.asa**文件有如下结构：**

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
SUB Application_OnStart
END SUB
</SCRIPT>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
SUB Application_OnEnd
END SUB
</SCRIPT>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
SUB Session_OnStart
```

```
END SUB
</SCRIPT>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
SUB Session_OnEnd
END SUB
</SCRIPT>
```

Global.asa包括4个脚本。它们分别是，根据Session_OnStart事件触发、根据Session_OnEnd事件触发、根据Application_OnStart事件触发、根据Application_OnEnd事件触发。

请注意Global.asa使用了微软的HTML拓展<SCRIPT>标记语法来限制脚本，这也就是说，你必须用<SCRIPT>标记来引用这两个事件而不能用<%和%>符号引用。例子中Global.asa使用的是VBScript，但是你也可以使用其他脚本语言。

在Global.asa中不能有任何输出语句，无论是HTML的语法还是Response.Write()方法都是不行的，Global.asa是任何情况下也不能进行显示的。

Session对象其实是利用Cookies进行信息处理的（参见前面有关Cookies的介绍），当用户首先进行了请求后，服务器端就在用户浏览器上创建了一个Cookies，当这个Session结束时，其实就意味着这个Cookies过期了。所以它的兼容性就受到了限制，一些老的浏览器显然是不行的，新的浏览器也提供了屏蔽Cookies的选项。

2.4 Sever 对象

Server对象提供对服务器上的方法和属性的访问。其中大多数方法和属性是作为实用程序的功能服务的。

语法：

Server.属性|方法

2.4.1 属性

Server对象包含ScriptTimeout属性。

ScriptTimeout属性指定脚本在结束前最大可运行多长时间。当处理服务器组件时，超时限制将不再生效。

语法：

```
Server.ScriptTimeout = NumSeconds
```

参数：

```
NumSeconds
```

指定脚本在被服务器结束前最大可运行的秒数。默认值为90秒。

当你的脚本生成了一个十分巨大的主页时，你肯定不希望主页显示到一半就超时了。那么你可以利用Server对象的ScriptTimeout属性定制你希望的限制时间。

如下所示：

```
<% Server.ScriptTimeOut=150 %>
```

2.4.2 方法

1. CreateObject 方法

Server.CreateObject用于创建已经注册到服务器上的 ActiveX 组件实例。这是一个非常重要的特性，因为通过使用 ActiveX 组件能够使你轻松地扩展 ActiveX 的能力，譬如数据库连接、文件访问、广告显示和其他 VBScript 不能提供或不能简单地依靠单独使用 ActiveX 所能完成的功能。正是因为这些组件才使得 ASP 具有了强大的生命力。

语法：

```
Server.CreateObject("Component Name")
```

默认情况下，由 Server.CreateObject 方法创建的对象具有页作用域。这就是说，在当前 ASP 页处理完成之后，服务器将自动破坏这些对象。如果要创建有会话或应用程序作用域的对象，可以使用 < OBJECT> 标记并设置 Session 或 Application对象的 Scope属性，也可以在对话及应用程序变量中存储该对象：

```
< % Set Session("ad") = Server.CreateObject("MSWC.AdRotator")%>
```

这里需要注意的是，不能创建与内嵌对象同名的对象实例，否则，如下列脚本将返回错误信息：

```
< % Set Response = Server.CreateObject("Response") %>
```

2. HTMLEncode方法

HTMLEncode 方法允许你对特定的字符串进行 HTML 编码，虽然 HTML 可以显示大部分你写入 ASP 文件中的文本，但是当你需要实际包含 HTML 标记中所使用的字符时，就会遇到问题。这是因为，当浏览器读到这样的字符串时，会试图进行解释。

语法：

```
Server.HTMLEncode( string )
```

参数：

string：指定要编码的字符串。

例如你想说明
在HTML文件中将产生换行，你会这样写：

```
<HTML>
<BODY>
<FONT SIZE=3>在HTML中，符号<br>将进行换行操作</FONT>
</BODY>
</HTML>
```

浏览器显示为：

在HTML中，符号

将进行换行操作

上面语句运行的结果并设有显示我们想象的结果。为了避免此类问题，我们就需要使用 Server 对象的 HTMLEncode 方法，采用对应的不由浏览器解释的 HTML 编码替代 HTML 标记字符。所以，用下面的代码才能显示正确的 HTMLEncode 字符串，从而在浏览器中按你的需要输出文本。

```
<%
Response.Write Server.HTMLEncode("在HTML中，符号<br>将进行换行操作")
%>
```

上面语句浏览器显示为：在HTML中，符号
将进行换行操作

3. URLEncode方法

就像 HTMLEncode 方法使客户可以将字符串翻译成可接受的 HTML 格式一样，Server 对象的 URLEncode 方法可以根据 URL 规则对字符串进行正确编码，当字符串数据以 URL 的形式传递到服务器时，在字符串中不允许出现空格，也不允许出现特殊字符。为此，如果你希望在发送字符串之前进行 URL 编码，可以使用 Server.URLEncode 方法。

语法：

```
Server.URLEncode( string )
```

参数：

String：指定要编码的字符串。

例如：

```
<%Response.Write(Server.URLEncode("http://www.microsoft.com")) %>
```

产生如下输出：

```
http%3A%2F%2Fwww%2Emicrosoft%2Ecom
```

在利用QueryString在不同主页间传递信息时，如果信息带有空格或特殊字符，那么必须进行Encode操作，因为如果不这样做，你很可能使得接受信息的那边接受到一些你所不期望的奇怪字串。例如：

```
<a href="test.asp?Message=<%=Server.URLEncode("This Query String has been UR
encoded.")%>">点击这里</a>
```

注意不要对QueryString的名称以及等号进行 Encode操作，你只需要将其值进行 Encode操作就可以了。

进行了Encode操作后，效果如下：

```
Message=This+Query+String+has+been+URL+encoded%2E
```

你并不需要考虑对上面的字符串再进行解码， ASP会自动进行这样的处理。例如，假设test.asp中有这样的脚本：

```
<%=Request.QueryString("message")%>
```

这时，显示结果为：

```
This Query String has been URL encoded.
```

4. MapPath 方法

MapPath 方法将指定的相对或虚拟路径映射到服务器上相应的物理目录上。

语法：

```
Server.MapPath(Path)
```

参数：

Path：指定要映射物理目录的相对或虚拟路径。若 Path 以一个正斜杠 (/) 或反斜杠 (\) 开始，

则 MapPath 方法返回路径时将 Path 视为完整的虚拟路径。若 Path 不是以斜杠开始，则 MapPath 方法返回同 .asp 文件中已有的路径相对的路径。这里需要注意的是 MapPath 方法不检查返回的路径是否正确或在服务器上是否存在。

对于下列示例，文件 data.txt 和包含下列脚本的 test.asp 文件都位于目录 C:\Inetpub\Wwwroot\asp 下。C:\Inetpub\Wwwroot 目录被设置为服务器的宿主目录。下列示例使用服务器变量 PATH_INFO 映射当前文件的物理路径。以下脚本：

```
< %= Server.mappath(Request.ServerVariables("PATH_INFO"))%>
```

输出

```
c:\inetpub\wwwroot\asp\test.asp
```

由于下列示例中的路径参数不是以斜杠字符开始的，所以它们被相对映射到当前目录，此处是目录 C:\Inetpub\Wwwroot\asp。以下脚本：

```
< %= Server.mappath("data.txt")%>
< %= Server.mappath("asp/data.txt")%>
```

输出

```
c:\inetpub\wwwroot\asp\data.txt
c:\inetpub\wwwroot\asp\asp\data.txt
```

2.5 应用示例

本节介绍一个聊天室的源程序（见图 2-5与图2-6）。

图2-5 聊天室登陆界面

首先修改global.asa文件，在Session_OnStart事件中设置如果一个用户 10分钟没有发言，则释放这个用户的资源。在Session_OnEnd中设置如果一个用户离开，则释放他的用户名及 IP地址。

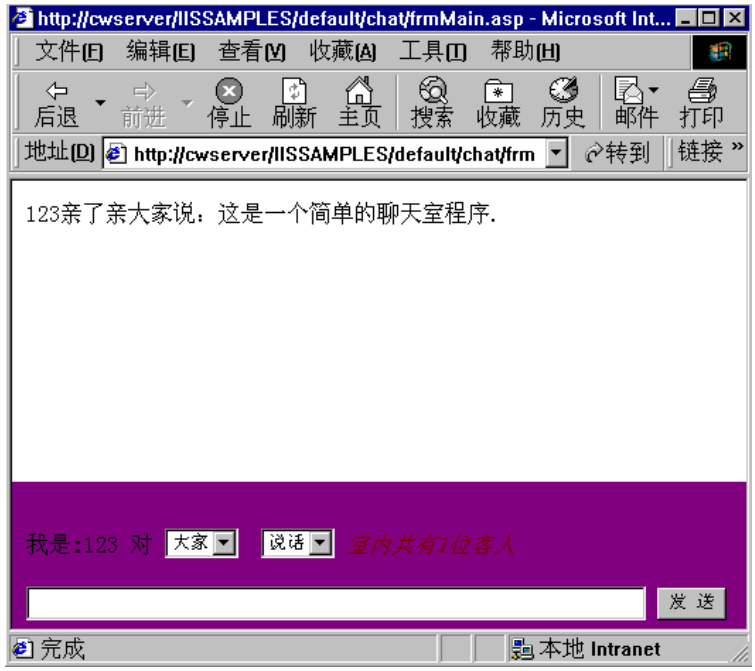

图2-6 聊天室主界面

用Application_OnStard**事件初始化所有的用户名称与用户** IP**地址。用**Application_OnEnd**事件清除所有的用户名称与用户** IP**地址。程序如下：**

```
------ global.asa ------
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
SUB Session_OnStart
Session.timeout=10
END SUB
SUB Session_OnEnd
FOR i=1 to 40
   IF Application("user" & i)=session("username") THEN
    Application("ip" & i)=""
    Application("user" & i)=""
   END IF
NEXT
END SUB

SUB Application_OnStard
  FOR i=1 to 40
    Application("ip" & i)=""
    Application("user" & i)=""
  NEXT
END SUB

SUB Application_OnEnd
```

```
  FOR i=1 to 40
    Application("ip" & i)=""
    Application("user" & i)=""
  NEXT
END SUB
</SCRIPT>
```

index.htm显示聊天室的初始界面（见图 2-5），并让用户输入聊天用的名字。函数 islength用于检测名字的长度，看是否符合标准。函数 isalpha用于检测名字中是否包含有非法字符。程序如下：

```
------ index.htm ------

<html>

<head>
<title>Chat</title>
</head>
<SCRIPT LANGUAGE=VBScript>
FUNCTION islength(val)
  val=cstr(val)
  namelen=0
IF len(val)>2 THEN
 FOR i=1 to len (val)
     IF asc(mid(val,i,1))<0 THEN
        namelen=namelen+2
     ELSE
        namelen=namelen+1
     END IF
 NEXT
 IF namelen<11 THEN
   islength=true
 ELSE
   msgbox 提示：昵称不能超过10个宽度"
   islength=false
 END IF
  ELSE
 msgbox 提示：昵称不能少于3个宽度"
 islength=false
  END IF
END FUNCTION

FUNCTION isalpha(val)
  IF islength(val) THEN
 IF instr(val,"<")>0 or instr(val,">")>0 _
    or instr(val,chr(34))>0 or instr(val,chr(39))>0 THEN
    msgbox 提示：您用了不合法字符"
    val.select
    isalpha=false
```

```
    EXIT FUNCTION
 END IF
 isalpha=true
  ELSE
 isalpha=false
  END IF
END FUNCTION

FUNCTION check()
  IF not islength(document.login.name.value) THEN
     EXIT FUNCTION
  ELSEIF not isalpha(document.login.name.value) THEN
     EXIT FUNCTION
  END IF
  document.login.submit ()
END FUNCTION
</SCRIPT>

<body>
<blockquote>
  <blockquote>
    <blockquote>
  <p align="center"><big><big><font color="#0000A0" 宋体ce="
"><strong><big><big> 
  激情时刻</big></big></strong></font></big><font 宋体" color="#800000"><em>
情侣天堂</em></font></big><img src="001.gif" WIDTH="128" HEIGHT="152"></p>
    </blockquote>
  </blockquote>
</blockquote>

<Form method="POST" name=login action="frmMain.asp">
  <div align="center"><center><p>
  </center></div><div align="center"><center><table border="0" bgcolor="#000
width="408" height="88">
<tr>
  <td bgcolor="#000050" width="117" align="left" height="22"><font
color="#FFFFFF"><strong></strong></font></td>
  <td bgcolor="#000050" width="182" align="left" height="22"></td>
  <td bgcolor="#000050" width="97" align="left" height="22"></td>
</tr>
<tr>
  <td bgcolor="#000050" width="117" align="left" height="40"><div
align="right"><p><font color="#FFFFFF"><st 昵称g></strong></font></td>
  <td bgcolor="#000050" width="182" align="left" height="40"><input type="te
name="name" size="16"></td>
  <td bgcolor="#000050" width="97" align="left" height="40"><input type="but
value="下一步" onClick="check()" name="button"></td>
</tr>
```

```
<tr>
  <td width="389" colspan="3" align="left" height="18"><div
align="center"><center><p></td>
</tr>
  </table>
  </center></div><div align="center"><center></p>
  </center></div>
</Form>
</body>
</html>
```

frmmain.asp用request对象的form集合获得用户输入的名字，用 ServerVariables集合获得用户的IP地址，并保存入应用程序级变量的 Application对象。程序如下：

```
------ frmmain.asp ------

<%@ LANGUAGE="VBSCRIPT" %>
<%
DIM Cantin
DIM Twoman
name=request.form("name")
Cantin=false
Twoman=false
Session("username")=name
FOR i=1 to 40
   IF Application("user" & i)=Session("username") and _
  Application("ip" & i)=Request.ServerVariables("REMOTE_ADDR") THEN
  Application("user" & i)=Session("username")
  Application("ip" & i)=Request.ServerVariables("REMOTE_ADDR")
  EXIT FOR
END IF
IF Application("ip" & i)=request.ServerVariables("REMOTE_ADDR") and _
  Application("user" & i)<>Session("username") THEN
  Cantin=true
  loginedname=Application("user" & i)
  EXIT FOR
END IF
IF Application("user" & i)=Session("username") and _
  Application("ip" & i)<>Request.ServerVariables("REMOTE_ADDR") THEN
  Cantin=true
  Twoman=true
  EXIT FOR
   END IF
   IF Application("user" & i)="" THEN
  Application("user" & i)=Session("username")
  Application("ip" & i)=Request.ServerVariables("REMOTE_ADDR")
  EXIT FOR
END IF
NEXT
```

```
%>
<html>

<%IF Cantin<>true THEN%>

<frameset framespacing="0" border="false" frameborder="0" rows="33,16">
  <frame name="main" src="list.asp" scrolling="no">
  <frame name="footer" scrolling="no" src="input.asp" target="main">
  <noframes>
  <body topmargin="0" leftmargin="0">
<%
ELSE
  IF loginedname<>"" THEN%> </h2>
  <h5>你已经用昵称<font color="red"><%=loginedname%></进入！><br>
  一个IP地址只能有一人进入， <br>
  请使用<font color="red"><%=loginedname%></进入！><br>
  要更换聊天代号， <br>
<%END IF
  IF twoman=true THEN%> </h5>
  <h4>来自另一个IP的客人正在使用该代号。<br>
  <br>
  请另选其它呢称进入</h4>
<%
END IF
END IF
%>
</body>
</noframes>
</frameset>
</html>
```

由于聊天室的所有客户都要能够共享信息，所以主要用具有应用程序级变量的对象Application，这是建立 Chat 程序的关键，所有的谈话数据都存放在一个应用程序级变量中，以便让所有的客户读取。 Input.asp用request 对象的form集合获取客户所输入的谈话，并保存在变量 Talkstr 中，然后将 Talkstr 的值存入应用程序级变量 output中。程序如下：

```
------ input.asp ------

<%@ LANGUAGE="VBSCRIPT" %>
<%
DIM Talkstr
Talkstr=Request.form("input")
IF instr(Talkstr,"script")>0 THEN Talkstr=""
IF instr(Talkstr,chr(39))>0 THEN
Talkstr=replace(Talkstr,chr(39),"\"&chr(39))
END IF
IF instr(Talkstr,chr(39))>0 THEN
Talkstr=replace(Talkstr,chr(39),"\"&chr(39))
```

```
END IF
IF Talkstr<> "" THEN
  action=Request.form("action")
  towho=Request.form("towho")
  name=Request.form("name")
  SELECT CASE action
CASE 0
    Talkstr=name &对" & towho &说：" & Talkstr
CASE 1
    Talkstr=name &对" & towho &嚷道：" & Talkstr
CASE 2
    Talkstr=name &亲了亲" & towho &说：" & Talkstr
CASE 3
    Talkstr=name &对" & towho &瞪圆了眼睛：" & Talkstr
CASE 4
    Talkstr=name &对" & towho &挥舞着拳头：" & Talkstr
CASE 5
    Talkstr=name &向" & towho &吐了一下舌头：" & Talkstr
CASE 6
    Talkstr=name &看着" & towho &张大了嘴：" & Talkstr
  END SELECT
  IF Application("output10")="" THEN
FOR i=1 to 10
  IF Application("output" & i)="" THEN
    Application.lock
    Application("output" & i)=Talkstr
    Application.unlock
    Axit for
  EXIT FOR
    END IF
   NEXT
  ELSE
   FOR k=1 to 9
  Application.lock
  Application("output" & k)=Application("output" & (k+1) )
  Application.unlock
NEXT
Application.lock
Application("output10")=Talkstr
Application.unlock
  END IF
END IF
%>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
```

```
<title>chat</title>
<base target="main">
<meta name="Microsoft Border" content="none">
</head>

<body bgcolor="#8A2BE2">
<%IF session("username")="" THEN%>
<h2>操作超时，请从新登录 <%else%> </h2>
<Form method="POST" name="inputForm" action="input.asp" target="footer">
<Input Type="hidden" name="name" value="<%=Session("username")%>">
<div align="center"><center>
<table border="0" cellspacing="0" cellpadding="0" width="902" height="70">
<tr>
<td width="904" colspan="2" valign="bottom" height="46">
<font color="#000000"><br>
我是<b>:</b><%=Session("username")%></font><b>  </b>
<font color="#00000对&</font>
<Select name="towho" size="1">
<option value=大家">大家</option>
<%FOR i=1 to 40 %>
<%
IF Application("user" & i)<>"" THEN
number=number+1
%>
<option <%
IF towho=Application("user" & i) THEN%> selected <% END IF%>
value="<%=Application("user" & i)%>">
<%response.write(Application("user" & i))%> </option>
<%
END IF
NEXT
%>
</Select> 
<Select NAME="ACTION" size="1">
    <option VALUE="0说话 </option>
    <option VALUE="1叫嚷 </option>
    <option VALUE="2亲亲 </option>
    <option VALUE="3暴怒 </option>
    <option VALUE="4抗议 </option>
    <option VALUE="5默意 </option>
    <option VALUE="6惊讶 </option>
</Select>
<em><font color="#80000室内共有<%=number%>位客人</font></em><p>
<Input Type="text" name="input" size="53" style="font-size: 12pt">
<Input Type="submit" valu发=送" name="B1"></td>
</tr>
<tr align="center">
<td width="642" height="5"><div align="right"></div></td>
```

```
<td width="255" height="5"><div align="center"></div></td>
</tr>
</table>
</center></div>
</Form>
<%end if%>
</body>
</html>
```

list.asp用response对象的Write方法输出Application对象中所保存的说话内容。程序如下：

```
------ list.asp ------

</head><%@ LANGUAGE="VBSCRIPT" %>
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<meta http-equiv="refresh" content="2">
<title>chat room</title>
<body>
<%
FOR i=0 to 10
  IF Application("output" & i)<>"" THEN
Response.Write Application("output" & i) & "<BR>"
  END IF
NEXT
%>
</body>
</html>
```

以上，就是一个简单的聊天室的全部代码，请用记事本或其他编辑器以给定的文件名依次建立好以上文件，用HTTP的方式打开index.htm浏览器，就可以看到效果。