第 4 章　Response 与 Request 对象

本章学习目标

本章介绍 ASP 的两个重要内置对象：Request 对象和 Response 对象。通过本章的学习，读者应该掌握以下内容：

- ASP 内置对象的基本功能
- Response 对象的使用
- Request 对象的使用

4.1　ASP 内置对象

在 ASP 程序设计过程中，除了使用前面所述的 VBScript 脚本编写代码外，还可以使用 ASP 提供的对象和组件。在 ASP 中提供了 Request、Response、Server、Session、Application 和 ObjectContext 共 6 个内置对象，这些对象在使用时并不需要经过任何声明或建立的过程，其功能如表 4-1 所示。

表 4-1　ASP 内置对象及其功能

对象名称	对象功能
Request 对象	取得用户通过 HTTP 请求传递过来的信息，包括使用 POST 或 GET 方式传递的数据和客户端传递的 Cookie 等
Response 对象	用于向客户端发送指定的信息，包括向浏览器输出数据、将浏览器重定向到另一个 URL 或设置 Cookie 值等
Server 对象	用于访问服务器上的系统方法和属性，这些属性和方法主要是为应用程序提供服务的
Session 对象	用于存储某个特定用户的信息，这个信息会一直伴随该用户，直到该用户离开网站、明确释放 Session 或 Session 超时
Application 对象	用于存储供多个用户使用的数据，Application 中的数据可以被网站的所有用户访问，提供了一种在多个用户间进行数据交换的方式
ObjectContext 对象	用于处理与事务相关的问题，与 ASP 的其他对象有所不同，ObjectContext 对象没有属性和集合，只有方法和事件

注意：上述 6 个 ASP 内置对象都是在服务器端运行的，应该放在服务器脚本中。

在这些对象中，最基本、最常用的是 Request 和 Response 对象，它们实现了客户端浏览器与 Web 服务器端交互的功能。本章将详细介绍这两个对象的使用。

4.2 Response 对象

Response 对象用于动态响应客户端请求，并将动态生成的响应结果以 HTML 超文本的格式输出到客户端浏览器中。使用 Response 对象可以完成动态创建 Web 页面、改变 HTTP 标题头、自动将客户端重定向（Redirect）到一个指定的页面中等功能。另外，如果要向客户端写入 Cookies 时，Response 对象也是一种很好的工具。

Response 的使用语法格式为：

```
Response.collection|property|method
```

其中：collection 表示 Response 对象的集合；property 表示 Response 对象的属性；method 表示 Response 对象的方法。3 个参数只能选择其中的一个。

4.2.1 Response 对象的属性

Response 对象所具有的属性如表 4-2 所示。

表 4-2 Response 对象的属性

属性	功能说明
Buffer	表明页面的输出是否被缓冲
CacheControl	指定是否允许代理服务器缓存页面
Charset	将字符集的名称添加到 Response 对象的 content-type 标题的后面
ContentType	指定响应的 HTTP 内容类型
Expires	指定在浏览器中缓存的页面的超时时间间隔
ExpiresAbsolute	指定浏览器上缓存页面超时的具体日期和时间
IsClientConnected	表明客户端是否与服务器保持连接状态
Pics	将 PICS 标记的值添加到响应标题的 PICS 标记字段中
Status	用于传递 Web 服务器 HTTP 响应的状态

Response 对象的常用属性如下：

1. Buffer 属性

Buffer 属性用于指定是否使用缓冲页向客户端浏览器输出。如果使用了缓冲页，则只有当前 ASP 页面的所有服务器脚本处理完毕或者明显地调用了 Response 对象的 Flush 或 End 方法后，服务器才将响应发送给客户端；反之，如果不使用缓冲页，则服务器在处理脚本的同时就将输出发送给客户端。

Buffer 属性是 Response 对象最常用的属性之一，其语法格式如下：

```
Response.Buffer =Flag
```

其中：Flag 为布尔值。当 Flag 为 False 时，表示不使用缓冲页输出；当 Flag 为 True 时，表示使用缓冲页输出。

对于一个 ASP 页面来说，如果服务器处理的时间较短，用户对于 Buffer 的取值为 True 或 False 不会有太明显的感觉；反之，如果服务器处理的时间较长，用户就能明显地感觉到 Buffer

取值的不同。如下例所示。

例 4-1：

```
<%@ LANGUAGE = "VBScript" %>
<% Response.Buffer=True%>
<HTML><HEAD>
<TITLE>使用缓冲示例</TITLE>
</HEAD><BODY>
<%    firsttime=Timer
timestep=0
i=0
Do While timestep<1
    response.write "the number is:" &i & "       "
    if i mod 4=0 then response.write "<BR>"
    i=i+1
    timestep=Timer-firsttime
Loop
response.write "<BR> the time is:" & timestep
%>
<%@ LANGUAGE = "VBScript" %>
<% Response.Buffer=False%>
<HTML><HEAD>
<TITLE>不使用缓冲示例</TITLE>
</HEAD><BODY>
<%    firsttime=Timer
timestep=0
i=0
Do While timestep<1
    response.write "the number is:" &i & "       "
    if i mod 4=0 then response.write "<BR>"
    i=i+1
    timestep=Timer-firsttime
Loop
response.write "<BR> the time is:" & timestep
%>
```

上例中分别将 Buffer 属性值设置为 True（文件名为 bufferT.asp）和 False（文件名为 bufferF.asp），在规定的时间（1s）内向客户端浏览器输出了指定的字符串，其程序的运行结果如图 4-1 所示。

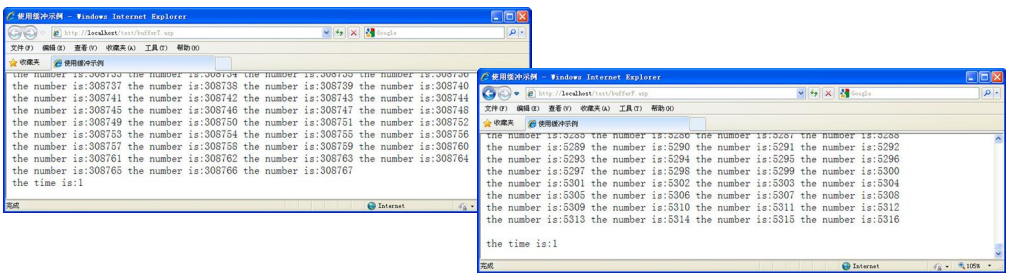

图 4-1　Buffer 属性值不同的执行结果

从例 4-1 中可以看出，在 1s 的时间内，如果将 Buffer 属性值设为 True，可以执行 308767 次循环体，而将 Buffer 属性值设为 False 时，仅可以执行 5316 次同样的循环体。也就是说，使用缓冲页面输出时，ASP 页面在 Web 服务器端的执行速度会更快。

从使用的角度看，在没有缓冲输出的时候，向客户端浏览器输出的内容会立即下载到浏览器，无法再修改。而有了缓冲输出，还可以根据实际情况利用 Response 对象的 Clear 方法中途清除缓冲区中的数据。另一方面，如果使用了缓冲，且 ASP 的运行时间较长，将造成用户长时间处于没有任何结果的等待中，会使用户失去耐心。因此，对于运行时间较长的程序，应该设置为没有缓冲输出。

由于 Buffer 属性的设置直接影响输出结果，使用时要注意以下两点：

（1）在 IIS 5.0 及以后的版本中，Buffer 属性的默认值为 True；在以前的版本中，Buffer 属性的默认值为 False。用户也可以在“Internet 信息服务”控制台中进行相应的设置（请参阅第 1 章在 IIS 5.1 中“应用程序配置”方面的内容）。

（2）设置 Buffer 属性的语句应放在<%@ LANGUAGE = …%>命令后面的第 1 行。如果在 HTML 或脚本输出之后更改 Buffer 属性值，将会出现错误。

2. Expires 属性

Expires 属性用来设置 Web 页面保留在客户端浏览器缓冲区的时间长度。如果用户在某个页过期之前又回到此页，客户端浏览器就会显示缓冲区中的版本，否则就要重新到 Web 服务器上去读取该页面。语法格式如下：

```
Response.Expires=分钟数
```

如果分钟数设为 0，就不会在客户端浏览器保存数据，这样用户每次访问该网页时，都必须重新请求并下载该页面。这对于需要实时传送信息的页面来说是比较合适的；此外，当用户通过 ASP 的登录页面进入 Web 站点后，将 Expires 属性的值设为 0，使其页面立即过期，可以保证当用户重新进入该页面时，必须与 Web 站点重新建立联系，有利于 Web 站点的安全。

注意：该属性必须放在<HTML>标记之前，否则会出错。如果在一个 Web 页面中多次设置了该属性，则使用最短的时间。

3. ExpiresAbsolute 属性

ExpiresAbsolute 属性明确指定缓存于浏览器中 Web 页面的到期日期和时间，在指定的日期和时间未到期之前返回该页面时，就显示缓存的内容。语法格式如下：

```
Response.ExpiresAbsolute[= [日期] [时间]]
```

其中，日期指定页面的到期日期，取值应符合 RFC-1123 规定的日期格式，如果未指定日期，则该页面在脚本运行当天的指定时间到期；时间指定页面的到期时间，如果未指定时间，该页面在当天午夜到期。

例如，指定页面在 2007 年 11 月 20 日的 21 点 25 分 30 秒到期，可以设置为：

```
<% Response.ExpiresAbsolute =#Nov 20,2007 21:25:30# %>
```

4. IsClientConnected 属性

IsClientConnected 属性是只读属性，用于判断客户端是否与服务器保持连接状态。语法格式如下：

```
布尔值＝Response.IsClientConnected()
```

当用户提出请求时，可能请求执行的程序会运行很长的时间。如果在这段时间内，用户

已经离开了该网站，那么被请求的程序就没有必要再执行下去了。可以利用 IsClientConnected 属性来判断客户端是否依然与服务器处于连接状态，来决定下一步执行的动作。典型的代码如下：

```
<%
If not Response.IsClientConnected Then
……'失去连接的处理代码
Response.End
End if
%>
```

5. Status 属性

Status 属性用来设置 Web 服务器要响应给客户端浏览器状态行的值。语法格式如下：

```
Response.Status = "状态描述字符串"
```

在 HTTP 协议中定义了“状态描述字符串”。该字符串由一个 3 位整数和一串说明文字组成，客户端可以根据这些代码和说明信息得到服务器端的执行情况。常用的有：

- 400：错误请求。
- 410：超越权限的请求。
- 404：无法找到。
- 406：无法接受。
- 412：初始化主页时间过长。
- 500：服务器内部错误。
- 502：网关错误。

注意：必须把该属性放在<HTML>标记之前，否则会出错。

4.2.2　Response 对象的方法

Response 对象可以使用的方法如表 4-3 所示。

表 4-3　Response 对象的方法

方法	功能说明
AddHeader	设置 HTML 标题
AppendToLog	在 Web 服务器的日志文件中记录日志
BinaryWrite	按照字节格式向客户端浏览器输出数据，不进行任何字符集的转换
Clear	清除服务器中缓存的 HTML 信息
End	停止处理.asp 文件并返回当前的结果
Flush	立即发送缓冲的输出
Redirect	重定向当前页面，尝试连接另外一个 URL
Write	直接向客户端浏览器输出数据

Response 对象的常用方法如下：

1. Write 方法

Write 方法是 Response 对象最常用的方法，该方法可以向浏览器输出动态信息。语法格式如下：

Response.Write Variant

其中，Variant 可以是 VBScript 中支持的任何数据类型的数据，包括字符型数据、数值型数据以及变量的值、HTML 标记等都可以用 Response.Write 方式输出到客户端浏览器。在使用 Response 对象的 Write 方法时要注意以下几点：

（1）Write 方法在输出数据时将所有数据都作为字符型数据处理，如果同时输出不同类型的数据，需要在数据间使用字符串连接运算符“&”。如下例所示：

例 4-2：

```
<%
'显示字符串
   Response.Write "This is a string. "&"<BR>"
'显示数字
   Response.Write 123&"<BR>"
'数字与字符混合显示
   dim a
   a=123
   Response.Write "This is a string. "&a
%>
```

程序的运行结果如图 4-2 所示。

图 4-2　字符及数字的显示

注意：实际上 Response 对象的 Write 方法并不区分数据类型，而是将所有数据一律都作为字符型数据输出，因此需要在不同的数据间使用字符串连接运算符“&”。

（2）直接向客户端浏览器输出 HTML 标记时，浏览器就会解释该 HTML 标记，并按指定的格式显示给用户。如果在 HTML 标记中包含“"”时，可以把“"”写成“'”。如下例所示。

例 4-3：

```
<HTML><BODY>
<%
'向客户端输出一个表格
Response.Write "<TABLE align='left' border='1' width='100%' >"
Response.Write "<TR><TD width='20%'>4.1ASP 内置对象</TD>"
Response.Write "<TD width='20%'>4.2Response 对象</TD>"
Response.Write "<TD width='20%'>4.3Request 对象</TD>"
Response.Write "<TD width='20%'>4.4 综合实例</TD>"
Response.Write "<TD width='20%'>4.5 思考与练习</TD>"
Response.Write "</TR>"
Response.Write "</TABLE>"
%>
```

```
<BR><BR><HR>
<%'向客户端输出一个项目，并加上超链接
Response.Write "<A href='chap4.htm'>" %>
第 4 章　Request 和 Response 对象<%="</A>" %>
</BODY></HTML>
```

运行结果如图 4-3 所示。

图 4-3　输出表格和超链接

实际上，例 4-3 中的显示结果完全可以用第 2 章介绍的 HTML 标记实现，这里只是演示如何利用 Response.Write 方法编写脚本命令。通常情况下，只有变量或一些需要改变的数据才使用 Response.Write 方法输出。

说明：如果在“<%”和“%>”之间只有一行 Response.Write 语句，可以使用“=”代替 Response.Write，如上例中的<%="</A>" %>脚本。

（3）在 ASP 程序中，由于“%>”和“"”两个字符具有特殊的含义，输出的数据中不能包括字符“%>”或“"”。如果确实需要输出这两个字符，可用转义序列“%\>”或使用“"""”字符来代替，如下例所示。

例 4-4：

```
<%
'显示字符串
   Response.Write "This is a first string. "&"<BR>"
'显示带引号的字符串
   Response.Write """&" This is a second string. "&"""&"<BR>"
'显示“%”
   Response.Write """&" This is a third string%\>. "&"""&"<BR>"
%>
```

程序的运行结果如图 4-4 所示。

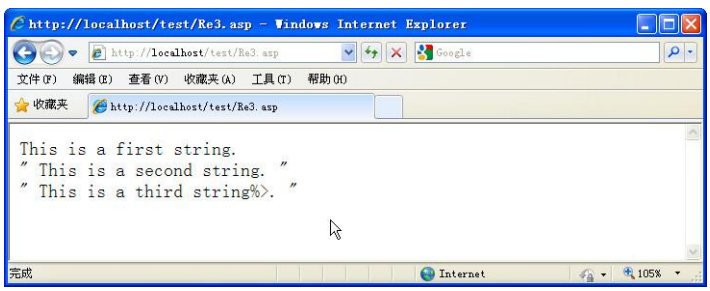

图 4-4　输出“%>”和“"”

2. Redirect 方法

Redirect 方法可以将客户端的浏览器重定向到一个新的网页，语法格式如下：

```
Response.Redirect URL
```

其中：URL 是指浏览器重定向到页面的统一资源定位符。

使用 Redirect 方法的好处是可以把复杂的网页分解成多个小网页，然后根据不同的情况将用户的请求重定向到不同的网页。

例 4-5：

```
<%
 Response.Buffer=True
 %>
<HTML>
<BODY>
<%'获取系统当前时间
CurrentH = Hour(Now())
'判断是否为上午
If CurrentH >= 8 and CurrentH <= 18 Then
    Response.Redirect "working.htm"
Else
    Response.Redirect "stop.htm"
End If
%>
</BODY>
</HTML>
```

working.htm 文件的代码如下：

```
<HTML>
<BODY>
欢迎，现在是工作时间！
</BODY>
</HTML>
```

stop.htm 文件的代码如下：

```
<HTML>
<BODY>
对不起，现在休息，请上班时间访问！
</BODY>
</HTML>
```

例 4-5 在不同时间的访问结果如图 4-5 所示。

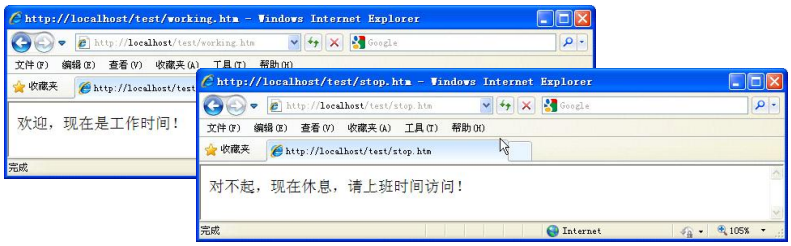

图 4-5　Response.Redirect 示例

注意：由于 Redirect 方法将引导用户浏览器打开一个新的网页，因此在使用该方法之前不能有任何数据被输出到客户浏览器。也就是说，Response.Redirect 应放在程序的任何输出语句之前。或者设置 Response.Buffer=True，以启用缓冲处理，将输出存放到缓冲区。例 4-5 中，如果将第 1 行改为 Response.Buffer=False，将会产生如图 4-6 所示的错误。

图 4-6　Response.Redirect 错误示例

3. Clear 方法

Clear 方法删除服务器缓冲区中的所有 HTML 输出，但只删除响应正文而不删除响应标题。语法格式如下：

```
Response.Clear
```

注意：如果未将 Response.Buffer 设置为 True，该方法将导致运行时错误。

由于使用 Response 对象的 Clear 方法会清空服务器缓冲区的所有数据，因此在使用该方法时应该慎重。可以用该方法处理错误情况。

4. End 方法

End 方法使 Web 服务器停止处理脚本并返回当前结果，文件中剩余的内容将不被处理。如果 Response.Buffer 已设置为 True，则调用 Response.End 将缓冲输出。语法格式如下：

```
Response.End
```

使用 Response 对象的 End 方法可以强制结束 ASP 程序的执行，如例 4-6 所示。

例 4-6：

```
<%
Response.Write "this is the first string"
Response.End
Response.Write "this is the second string"
%>
```

例 4-6 的运行结果如图 4-7 所示。

图 4-7　Response.End 示例

5. Flush 方法

Response 对象的 Flush 方法可以立即发送缓冲区中的数据。如果未将 Response 对象的 Buffer 设置为 True，则该方法将导致运行时的错误。语法格式如下：

```
Response.Flush
```

如前所述，当 Response 对象的 Buffer 设置为 True 时，只有在当前的页面执行结束时，服务器才会向浏览器输出数据。如果需要根据实际情况当某个条件成立时，就将已经完成的页面发送到客户端时，可以使用本方法，具体示例如下：

```
<%
    if Response.Buffer then
      if flag=true then '当输出条件成立时
        Response.Flush'立即输出
        Response.Clear
        Response.End
    else
      ……'其他处理代码
      end if
    end if
%>
```

6. BinaryWrite 方法

Response. BinaryWrite 方法可以不经任何字符转换就将指定的信息输出。该方法主要用于输出非字符串信息（如客户端应用程序所需的二进制数据等）。语法格式如下：

```
Response.BinaryWrite 二进制数据
```

4.2.3 Response 对象的数据集合

Response 对象只有 Cookies 一个数据集合。

1. Cookie 概述

Netscape 首先在它的浏览器中引入了 Cookie，从那以后，WWW 协会就支持了 Cookie 标准。目前大部分浏览器都兼容 Cookie 的使用。

Cookie 实际上是一个字符串或一个标志，当一个包含 Cookie 的页面被用户浏览器读取时，一个 Cookie 就会被存入到用户计算机的本地硬盘中，当需要时该网站就可以从用户的本地硬盘中读取这些 Cookie。

注意：Cookie 被存储在用户本地计算机上，而非 Web 服务器上。

所有的 Cookie 都被存放在客户计算机的硬盘上，存储的位置与使用的操作系统有关：在 Windows 98 中存放在 Windows\Cookies 文件夹下，在 Windows 2000/XP 中存放在 Documents and Settings\用户名\Cookies 文件夹下。Cookie 文件的命名规则为：用户名@网站名.txt，如 zjf@google[1].txt，有时也使用 IP 地址来描述网站，如 zjf@127.0.0[2].txt。这些文件是纯文本文件，可以使用任何文本编辑器打开它们。

目前有些 Cookie 是临时的，还有一些是持续的。例如，当 Cookie 被 ASP 用来跟踪用户进程直到用户离开网站时，Cookie 就是临时的。如果 Cookie 被保持在 Cookie 文件中直到用户返回时又进行调用，这时的 Cookie 就是持续的。

由于 Cookie 能够读、写用户本地硬盘中的数据，对于 Cookie 的使用一直有很大的争议，

很多用户担心个人隐私被泄露或对本地计算机的安全构成威胁。从目前的使用情况来看，Cookie 只能向用户本地硬盘的固定目录中写入文本文件，而不是可执行文件，它们对计算机不会构成危害。当然，用户也可以在本地的浏览器中进行相应的设置，以决定 Cookie 的使用情况。以 IE8.0 为例，设置 Cookie 的方法是：依次选择“工具”→“Internet 选项”→“隐私”，在该弹出的对话框中可以设置 Cookie，如图 4-8 所示。

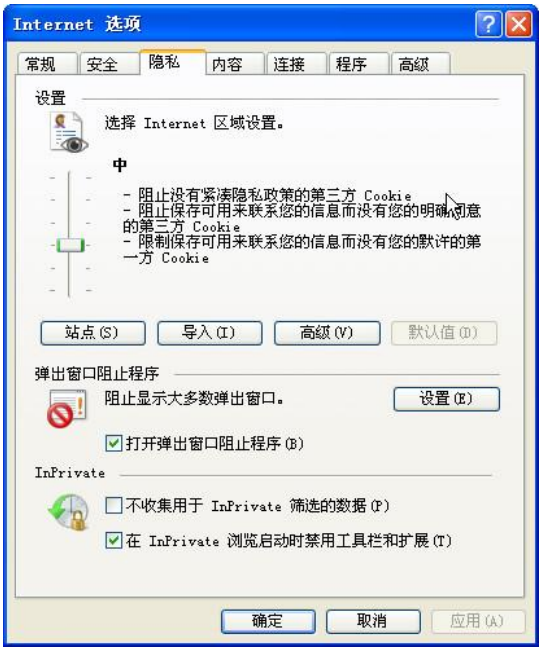

图 4-8　设置 Cookie 对话框

在 IE 8.0 中，将隐私的设置分为：阻止所有 Cookie、高、中上、中、低、接受所有 Cookie 六个级别，用户可以根据实际情况进行选择。

2. 创建 Cookie

使用 Response 对象的 Cookies 数据集合可以在客户端定义 Cookie 变量，语法格式如下：

```
Response.Cookies(Cookie)[(key)|attribute]=Value
```

其中：参数 Cookie 用于指定创建或设置 Cookie 的名称，如果指定的 Cookie 不存在就创建它；如果存在的话就赋予它一个新值。

参数 Value 用来指定分配给 Cookie 的值。

参数 key 为可选参数，如果不指定 key，则创建一个单值 Cookie；如果指定 key，则创建一个 Cookie 字典，而且该 key 将被设置为 Value。

Cookie 可分为字典式和非字典式两类。非字典式 Cookie 是指一条 Cookie 信息只有一个 Cookie 名称和对应的一个值（单值 Cookie），相当于一个变量；字典式 Cookie 是一个集合，有一个名称，集合内部有多个子 Cookie，每个子 Cookie 都有自己的名称和对应的值。

参数 Attribute 指定 Cookie 的属性，有以下几种：

Domain：指定只有某个 Domain（网域）可以存取该 Cookie，只写属性。

Expires：指定 Cookie 的过期日期，只写属性。

Cookie 有两种形式：临时 Cookie 和永久 Cookie。临时 Cookie 只有在浏览器打开时存在，一旦浏览器与 Web 服务器之间的会话结束，就删除所创建的 Cookie。永久 Cookie 将 Cookie 保存在客户的磁盘中，直到由 Expires 指定的日期前一直可用。如果 Expires 设置的日期未超过当前日期，则在会话结束后 Cookie 将到期。

HasKeys：用来判断指定 Cookie 是否包含关键字（即是否为一个 Cookie 字典），只读属性。

Path：指定存取该 Cookie 的路径，默认为 Web 应用程序所在的路径，只写属性。

Secure：指定 Cookie 是否安全，即在数据的传输过程中是否采用加密算法，只写属性。

（1）创建单值的 Cookie。

要创建不带关键字 key 的 Cookie，只要指定参数 Cookie 和 Value 的值就可以了。

```
<%
Response.Cookies("test")="hello"
Response.Cookies("test").Expires=Date()+7
Response.Cookies("test").Domain="127.0.0.1"
Response.Cookies("test").Path="/"
Response.Cookies("test").Secure=False
%>
```

Cookie 是利用 HTTP 的 Header 进行数据传送的，因此应该在 ASP 的任何输出语句之前进行上述操作，也可以使用 Buffer 输出。

这个脚本程序创建了一个名为 test 的不带关键字的 Cookie。它的值是"hello"，同时还指定了相应的属性值。

（2）创建带有关键字的 Cookie 字典。

创建带有关键字的 Cookie 字典时，需要带上 key 参数。例如：

```
<%
Response.Cookies("myCookie")("name")="Tom"
Response.Cookies("myCookie")("password")="Good boy"
%>
```

上面的脚本创建了一个名为 myCookie 的 Cookie 字典，其中含有 name 和 password 两个关键字。需要注意的是，如果想给一个带有关键字的 Cookie 字典指定属性值，一定不要带上关键字，否则会产生语法错误。正确的语法格式如下：

```
<% Response.Cookies("myCookie").Expires= Date ()+7 %>
```

4.3 Request 对象

Request 对象是 ASP 中最常用的对象之一，利用 Request 对象可以在服务器端获得用户端通过 Web 页面提交的信息，如浏览器的种类、表头信息、表单参数及 cookies 等。语法格式如下：

```
Request[.collection|property|method](variable)
```

其中：collection 表示 Request 对象的集合；property 表示 Request 对象的属性；method 表示 Request 对象的方法，collection、property 和 method 三个参数只能选择一个，也可以三个都不选。变量参数（variable）是一些字符串，这些字符串指定要从集合中检索的项目，或作为方法与属性的输入。

4.3.1　Request 对象的属性

Request 对象只提供一个 TotalBytes 属性，这是一个只读的属性，表示从客户端所接收数据字节的长度，其语法格式如下：

字节长度=Request.TotalBytes

下面的程序将示范如何取得从客户端接收的数据字节大小。

例 4-7：

```
<%
Response.Write "从客户端接收的数据字节大小为：" &Request.TotalBytes
%>
```

如果直接通过浏览器运行本程序，由于没有传递任何信息给 Web 服务器端，因此其返回值为 0。

4.3.2　Request 对象的方法

Request 对象只提供一种 BinaryRead 方法，该方法是以二进制方式来读取客户端使用 POST 传送方法所传递的数据。其语法格式如下：

Variant　数组=Request.BinaryRead(Count)

BinaryRead 方法的返回值为通用变量数组（Variant Array），其参数 Count 是一个整型数据，用以表示每次读取数据的字节大小，范围介于 0 到 Request 对象 TotalBytes 方法所取得的字节大小之间。

一般来说，如果使用 BinaryRead 方法取得客户端所传递的数据，就不能使用 Request 对象所提供的各种数据集合（Collections），否则会发生错误；反之，如果已经使用 Request 对象的数据集合取得客户端信息，也不能再使用 BinaryRead 方法，否则同样会发生错误。

4.3.3　Request 对象的数据集合

Request 对象将用户通过 HTTP 请求传送的信息保存在几个集合中，其语法格式如下：

Request[.collection]("variable")

其中：collection 用于指定 Request 对象的数据集合；variable 用于指定变量名或索引值。Request 对象的数据集合如表 4-4 所示。

表 4-4　Request 对象的数据集合

集合	功能说明
ClientCertificate	取得客户端的身份权限数据
Cookies	取得存在于客户端浏览器的 Cookies 数据
Form	取得客户端利用 POST 方式所传递的数据
QueryString	取得客户端利用 HTTP 查询字符串所传递的数据
ServerVariables	取得 Web 服务器端的环境变量信息

利用 Request 对象的数据集合取得数据时，collection 是可以省略的，也就是说只要使用

Request("变量名称")，就可以取得该变量的内容值。这时 ASP 会按照 QueryString、Form、Cookies、ClientCertificate、ServerVariables 的顺序在各个数据集合中搜索该变量，并返回第一个出现的变量值。显然，省略集合名称会影响执行效率，同时为了避免不同集合中同名变量引用的二义性，最好明确地指定集合。

1．Form 数据集合

Form 数据集合是 Request 对象中最常使用的数据集合。当使用 POST 方法将 HTML 表单提交给服务器时，表单中的各个元素被存储在 Form 中。利用 Form 数据集合可以取得客户端表单上的各项对象内容值，包括单行文本（Text）、文本块（TextArea）、复选框（CheckBox）、单选按钮（Radio）或下拉式列表框（Select）等，其语法格式如下：

```
表单对象内容=Request.Form("表单对象名称")
```

或

```
表单对象内容=Request.Form("索引值")
```

其中，表单对象名称是要检索的表单元素的名称；索引值是表单元素在表单集合中的顺序号。

（1）取得Form数据集合中元素的值。

例 4-8：

第 1 步：建立一个 HTML 的表单输入程序，其存储文件名称为 Input1.htm，要求输入姓名、性别及电子邮件信箱等信息，其完整的 HTML 内容如下：

```
<HTML>
<BODY><FORM method="POST" action="Output1.asp">
<P>姓名：<INPUT type="text"name="Name" size="10"></P>
<P>性别：<SELECT name="Sex" size="1">
        <option value="帅哥">帅哥</option>
        <option value="美女">美女</option>
    </SELECT>
</P>
<P>电子邮件信箱：<INPUT type="text" name="E-mail" size="30"</P>
<P><INPUT type="submit" value="确定">
    <INPUT type="reset" value="取消">
</P>
</FORM>
</BODY></HTML>
```

第 2 步：建立一个处理表单的 ASP 程序，其存储文件名称为 Output1.asp。这个 ASP 程序将利用 Request 对象的 Form 数据集合取得用户在表单中填写的内容，其程序代码如下：

```
<HTML><BODY>
<P>您的姓名是：<%Response.Write Request.Form("Name")%>。</P>
<P>您是一位<%Response.Write Request.Form("Sex")%>！</P>
<P>您的 E-mail 地址是：<%Response.Write Request.Form(3)%>。</P>
</BODY></HTML>
```

第 3 步：运行示例。在浏览器地址栏输入请求的 Input1.htm 页面。系统首先会显示一个 HTML 输入界面，如图 4-9 所示。

输入姓名、性别及电子邮件地址数据后单击“确定”按钮，此时系统会将所输入的数据

提交给 Output1.asp 处理。这里 Output1.asp 只是显示所输入的数据，其结果如图 4-10 所示。

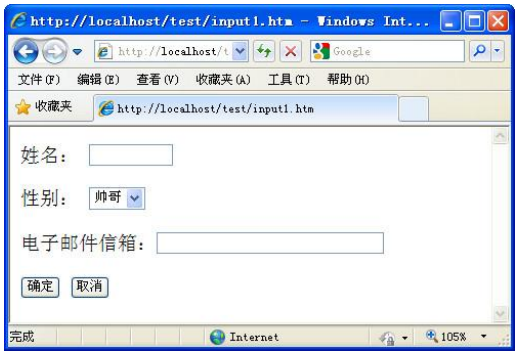

图 4-9　HTML 输入界面

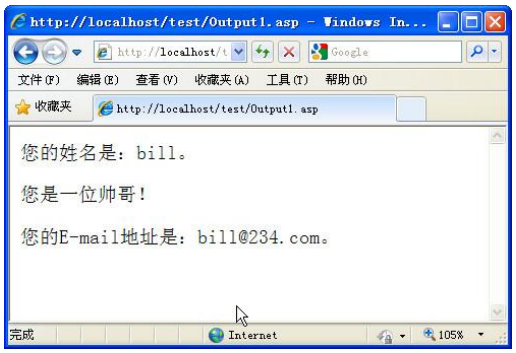

图 4-10　Request 对象的 Form 数据集合

说明：在 Output1.asp 中，除了可以直接使用表单对象名称取得该对象的内容（如 Request.Form("Name")）外，还可以利用变量的索引值（如 Request.Form(3)）取得表单对象的内容，当然也可以用循环的方法取得所有表单对象的内容值。将 Output1.asp 改写为下面的程序代码：

```
<%
For each item in Request.Form
    Response.Write item & ":"
    Response.Write Request.Form(item)
Next
%>
```

或

```
<%
For i=1 to Request.Form.count
    Response.Write Request.Form(i) & "<BR>"
Next
%>
```

前面所介绍的方法都只能取得具有某一名称的表单对象的值，如果多个对象具有相同的名称（如 CheckBox），取得该对象的值可以采用以下方法：

（1）首先应该取得具有相同名称对象的总数。利用该对象的 Count 属性可以获得，其使用语法格式如下：

```
名称相同的对象总数=Request.Form("表单对象名称").Count
```

或

```
名称相同的对象总数=Request.Form(索引值).Count
```

（2）在取得表单对象内容的语法后再加上一个索引值，就可以取得相同名称的对象内容值。其语法格式如下：

```
名称相同的对象内容值=Request.Form("表单对象名称")(索引值)
```

或

```
名称相同的对象内容值=Request.Form(索引值).(索引值)
```

例 4-9：

第 1 步：HTML 的表单输入程序，其存储文件名称为 Input2.htm，其完整的 HTML 程序

内容如下：

```
<HTML>
<BODY>
<FORM method="POST" action="Output2.asp">
<P >谢谢光临。请选择您的兴趣：<HR></P>
<INPUT name="interest" type=checkbox value="计算机">计算机
<BR><INPUT name="interest" type=checkbox value="羽毛球">羽毛球
<BR><INPUT name="interest" type=checkbox value="电影">电影
<BR><INPUT name="interest" type=checkbox value="登山">登山
<BR><INPUT name="interest" type=checkbox value="唱歌">唱歌
<BR><INPUT name="interest" type=checkbox value="唱歌">跳舞
<BR><INPUT name="interest" type=checkbox value="唱歌">游泳
<P><INPUT type="submit" value="确定">
    <INPUT type="reset" value="取消"></P>
</FORM>
</BODY>
</HTML>
```

第 2 步：建立一个处理表单的 ASP 程序，其存储文件名称为 Output2.asp，这个 ASP 程序的主要目的是利用 Request 对象的 Form 数据集合取得用户在 CheckBox 中的选择，完整的程序如下：

```
<HTML>
<BODY>
<%dim count
count=Request.Form("interest").count%>
根据我们的调查，您的兴趣有<%Response.Write count%>种，其中包括：
<%dim i
Response.Write   "<HR><BR>"
For i=1 to count
    Response.Write   Request.Form("interest")(i)&"<BR>"
Next%>
</BODY>
</HTML>
```

说明：此例中的 Count 属性表示 CheckBox 被选择的项目数量，并不是 CheckBox 所有项目的总数。

第 3 步：运行此例。通过浏览器运行输入程序 Input3.htm，系统首先会显示一个输入界面，如图 4-11 所示。

选择兴趣后单击“确定”按钮，此时系统会将所输入的数据提交给 Output2.asp 处理，显示结果如图 4-12 所示。

（2）自响应页面。对于简单的页面，也可以将请求与响应放在同一页面中实现，以提高效率，这种页面称为“自响应页面”。如下所示（文件名为 io.asp）。

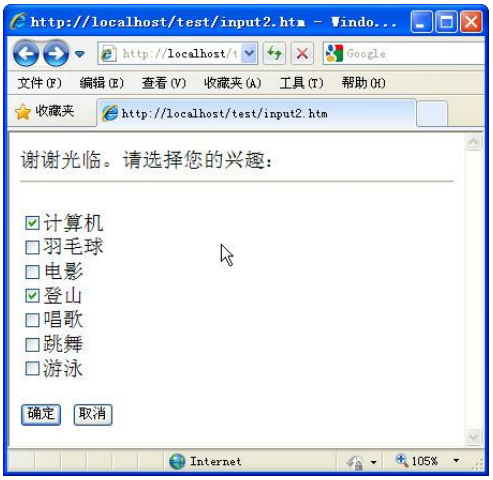

图 4-11　输入界面

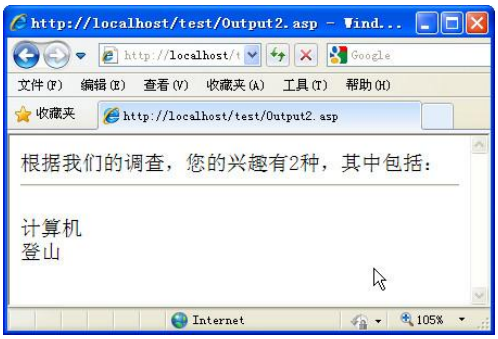

图 4-12　显示 CheckBox 中选择的结果

例 4-10：

```
<%
If Request.Form("ok")<>"确定" then
%>
<HTML>
<BODY>
<FORM method="POST" action="io.asp">
<h4 align="center">欢迎访问，请输入您的用户名和密码</h4>
<P align="center">用户名：<INPUT type="text" name="Username" size="10"></P>
<P align="center">密  码：<INPUT type="password" name="Userpass" size="10"></P>
<P align="center"><INPUT name="ok" type="submit" value="确定">
    <INPUT type="reset" value="取消">
</FORM>
</BODY>
</HTML>
<%
Else
    name=Request.Form("Username")
    pass=Request.Form("Userpass")
    If name<>"" then
        Response.Write "<P>欢迎访问!</P>"
        Response.Write "<P>您输入的用户名是：" & name & "</P>"
        Response.Write "<P>您输入的密码是：" & pass & "</P>"
    Else
        Response.Write "<P>对不起,请重新输入！"
        Response.Write "<a href='io.asp'>上一页</a>"
    End If
End If
%>
```

程序刚开始运行时的页面显示如图 4-13 所示。

图 4-13　io.asp 最初执行的显示效果

根据用户不同的输入信息，程序会有不同的执行结果。图 4-14 显示了用户名为空和非空两种状态下程序的执行结果。

图 4-14　io.asp 执行结果

2. QueryString 数据集合

QueryString 数据集合用于取得通过 HTTP 查询字符串传递的数据，查询字符串附加在 URL 的后面，其语法格式如下：

URL 地址? QueryString

在 URL 地址和参数 QueryString 间使用“?”字符分隔，当传递多个 QueryString 时，用“&”符号作为参数间的分隔符。例如：

http://www.example.com/login.asp ? username=admin & password=123

在访问 www.example.com/login.asp 文件的同时向该文件传递了 username（值为 admin）和 password（值为 123）两个 QueryString 参数。

利用 QueryString 数据集合取得客户端传送数据的语法格式如下：

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

其中，variable 指定了 QueryString 中参数的名称；index 是可选参数，用于指定 QueryString 中参数的索引值。

QueryString 数据集合常用的方法有以下几种：

（1）利用超级链接标记传递参数。在程序中可以直接利用 HTML 的<A> 标记传递参数，如下例所示。

例 4-11：

第 1 步：建立用户传递参数的 ASP 程序，其文件存储名称为 Input3.asp，程序如下：

```
<HTML>
<BODY>
<P><A HREF=http://localhost/Output3.asp?Select=1&Sex=男>显示的字符串 1</A></P>
<P><A HREF="Output3.asp?Select=2&sex=女">显示的字符串 2</A></P>
<%
Response.Write "<A HREF='Output3.asp?Select=3&Sex=男'>显示的字符串 3</A>"
%>
</BODY>
</HTML>
```

第 2 步：建立处理参数的 ASP 程序 Output3.asp，程序将利用 Request 对象的 QueryString 数据集合来取得参数的值。程序如下：

```
<HTML><BODY>
<%
'取得客户信息
QueryN=Request.QueryString("Select")
QueryS=Request.QueryString("Sex")
'根据选择的不同做出不同的处理
SELECT CASE QueryN
CASE "1"
   Response.Write "<BR>"&"您的选择是　"&QueryN&"　显示的字符串 1"
   Response.Write "<BR>"&"您的性别是　"&QueryS
CASE "2"
   Response.Write "<BR>"&"您的选择是　"&QueryN&"　显示的字符串 2"
   Response.Write "<BR>"&"您的性别是　"&QueryS
CASE "3"
   Response.Write "<BR>"&"您的选择是　"&QueryN&"　显示的字符串 3"
   Response.Write "<BR>"&"您的性别是　"&QueryS
END SELECT
%>
</BODY></HTML>
```

第 3 步：通过浏览器运行输入程序 Input3.asp，系统会出现一个超文本链接界面，如图 4-15 所示。

单击指定的字符串，系统就会将字符串所对应的参数传给输出程序 Output3.asp，运行结果如图 4-16 所示。

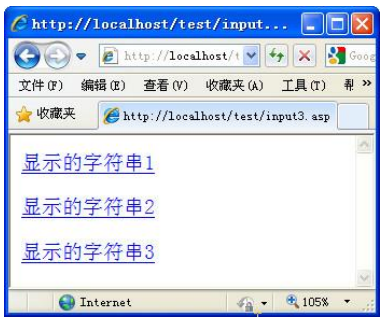

图 4-15　超文本链接界面

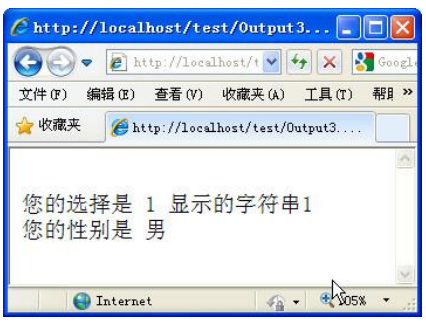

图 4-16　QueryString 数据集合运行结果

请读者仔细观察浏览器地址栏中显示的内容，如果直接在地址栏中输入相应的 URL 也可以达到同样的效果。

与 Form 数据集合一样，在 QueryString 数据集合中可以利用参数的名称，也可以用索引值来取得参数的值，对于具有多个取值参数的处理方法也与 Form 数据集合类似。

（2）取得在表单中通过 GET 方式提交的数据。在表单中通过 method="GET"可以指定表单使用 GET 方式提交数据。

例 4-12：

第 1 步：建立一个 HTML 的表单输入程序，其存储文件名称为 Input4.htm，要求输入用户姓名、用户密码等信息，其完整的 HTML 内容如下：

```
<HTML><BODY>
<FORM method="GET" action="Output4.asp">
<P>姓名：<INPUT type="text" name="Name" size="10"></P>
<P>密码：<INPUT type="Password" name="Password" size="10"></P>
<P><INPUT type="submit" value="确定">
    <INPUT type="reset" value="取消">
</P>
</FORM></BODY>
</HTML>
```

注意：在表单中指定了使用 GET 方式。

第 2 步：建立一个处理表单的 ASP 程序，其存储文件名称为 Output4.asp，在这个 ASP 程序中利用 QueryString 数据集合取得客户端传送的数据，完整的程序如下：

```
<HTML><BODY>
<P>您的姓名是：<%Response.Write Request.QueryString("Name")%>。</P>
<P>您的密码是：<%Response.Write Request.QueryString("Password")%>！</P>
</BODY></HTML>
```

第 3 步：运行此例，显示结果与例 4-10 类似，不再赘述。

HTTP 查询字符串在 Web 页面间传递参数时是非常有用的，但由于它是通过 HTTP 的附加参数来传递的，不同的浏览器对附加参数的长度有限制。因此，对于传输数据量比较大时，应该使用表单传递。此外，利用 HTTP 查询字符串传递参数时，通过浏览器的地址栏可以方便地得到传递的参数，保密性不够好，因此，不能用其传递涉及网站安全的信息。

3．Cookies 数据集合

一般来说，当用户第一次进入网站时，可以利用 Response 对象的 Cookies 数据集合将数据存储在用户的计算机中。当用户再次进入该网站时，就可以利用 Request 对象的 Cookies 数据集合取得相关信息。其语法格式如下：

```
Request.cookies(Cookie)[(key)|.attribute]
```

其中，参数 Cookie 用来指定被检索或读取的 Cookie 的名称；参数 Key 为一个可选项，用于指定 Cookie 字典中子 Cookie 的名称；参数 attribute 是 Cookies 数据集合的属性，只有一个取值 HasKeys，用来表示 Cookie 是否带有关键字（即是否为一个 Cookie 字典），只读属性。

（1）读取单值的 Cookie。对于一般不带关键字的 Cookie，可以采用指定 Cookie 名称的方式来检索 Cookie 的值。例如：

```
<%=Request.Cookie("test")%>
```

此外，也可以采用指定序号的方式来检索 Cookie 的值。假设建立了两个 Cookie，分别是 C1 和 C2，那么下面 4 条语句中前两条与后两条的含义是等价的。

```
Response. write(Request.Cookies("C1"))
Response. write(Request.Cookies("C2"))
```

或

```
Response. write(Request.Cookies(1))
Response. write(Request.Cookies(2))
```

（2）读取 Cookie 字典。对于 Cookie 字典的检索和读取，可以通过使用关键字来进行，也可以使用序号来进行。例如，要检索前面建立的 myCookie 字典的值，可以用下面的脚本：

```
Request.Cookies("myCookie")("Name")
Request.Cookies("myCookie")("password")
```

或

```
Request.Cookies("myCookie")(1)
Request.Cookies("myCookie")(2)
```

上面语句中，前两条语句和后两条语句是等价的。它们都可以显示 myCookie 这个 Cookie 字典中的关键字的取值。如果在访问 myCookie 时不指定关键字，将返回所有关键字和对应的值。例如：

```
Request.Cookies("myCookie")
```

将得到以下结果：

```
Name=对应的值& password=对应的值
```

因为所有的 Cookie 都保存在 Cookies 集合中，所以可以通过循环的方式遍历 Cookie 集合，以检索所有 Cookie 或关键字的值。下面的例子利用了 HasKeys 属性来遍历所有 Cookie，并将其值输出：

```
<%
For Each strKey In Request.Cookies
'单值 Cookie
  If Not Request.Cookies(strKey).HasKeys Then
     Response.Write    strKey &"=" & Request.Cookies(strKey)&"<BR>"
'Cookie 字典
  Else
     For Each strSubKey In Request.Cookies(strKey)
        Response.Write "->" & strKey & "(" & strSubKey & ") = " & _
        Request.Cookies(strKey)(strSubKey) & "<BR>"
     Next
  End If
Next
%>
```

Cookie 的使用较为广泛，下面的例子演示了如何利用 Cookie 实现用户自动登录的功能，这是一个包含 Cookies 读、写的综合实例。

例 4-13：

登录页面（loginc.asp）的代码如下：

```
<%
name=Request.Cookies("username")
```

```
pass=Request.Cookies("userpass")
%>
<HTML><BODY>
<FORM method="POST" action="cookiew.asp">
<h4 align="center">欢迎访问，请输入您的用户名和密码</h4>
<P align="center">用户名：<INPUT type="text" name="Username" size="10" value=<%=name%>></P>
<P align="center">密  码：<INPUT type="password" name="Userpass" size="10" value=
<%=pass%>></P>
<P align="center"><INPUT type="submit" value="确定">
    <INPUT type="reset" value="取消">
</FORM></BODY></HTML>
```

如果用户是第一次访问登录页面，由于 Cookie 中还没有具体的值，因此用户名和密码两项的值为空，如图 4-17 所示。

当用户输入了用户名和密码并单击“确定”按钮后，提交给 cookiew.asp 文件处理，该文件实现 cookie 的写入功能，代码如下：

```
<%
username=Request.form("username")
userpass=Request.form("userpass")
Response.Cookies("username")=username
Response.Cookies("username").Expires=Date()+15
Response.Cookies("userpass")=userpass
Response.Cookies("userpass").Expires=Date()+15
%>
```

此后，由于 cookie 中有了具体的值，当用户再次登录时，就会自动显示以前用过的用户名和密码，如图 4-18 所示。

图 4-17　初次登录界面

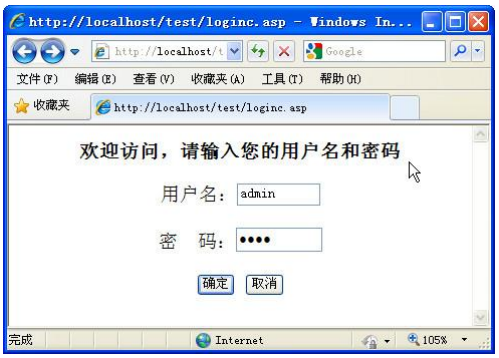

图 4-18　自动登录

4. ServerVariables 数据集合

使用 ServerVariables 数据集合可以获得服务器端环境变量的取值，这些环境变量存储着与 Web 服务器相关的一些信息和用户发送请求时浏览器通过 HTTP 报头传送的一些信息。其语法格式如下：

```
Request.ServerVariables (server environment variable)
```

其中，server environment variable 指定了某个环境变量的名称，常用的值如表 4-5 所示。

表 4-5　ServerVariables 环境变量

变量	说明
HTTP_USER_AGENT	发出请求的浏览器名称
REMOTE_ADDR	发出请求的远端主机的 IP 地址
REMOTE_HOST	发出请求的主机名称。
REQUEST_METHOD	发出 Request 请求的方法（对于 HTTP，可以是 GET、POST、HEAD 或其他方法）
SCRIPT_NAME	获取当前脚本的路径
SERVER_NAME	服务器的名称、DNS 别名或 IP 地址及指定的 URL 地址
SERVER_PORT	数据请求的端口号
SERVER_PROTOCOL	请求信息的协议名称及版本
SERVER_SOFTWARE	服务器运行的软件名称及版本

下面的例子演示了如何拒绝某个客户机的访问。

例 4-14：

```
<%
Dim strip
strip=Request.ServerVariables("REMOTE_ADDR")
If strip="127.0.0.1" then
    Response.Write "谢谢您的访问！"
Else
    Response.Write "对不起，拒绝访问！"
End If
%>
```

由于 ServerVariables 数据集合中的环境变量较多，限于篇幅，本书没有全部列出，读者可查阅相关技术文档或使用 IIS 的帮助（http://localhost/iisHelp/iis/misc/default.asp）获取相关的内容。

4.4　综合实例

在 ASP 程序中，Response 和 Request 对象使用非常频繁，它们是 ASP 的基本对象。只有掌握好这两个对象才能进行 ASP 的程序设计。本节以用户登录为例，加深对这两个对象的认识，熟悉它们的使用方法。

4.4.1　创建登录页面

登录页面要求输入或选择用户名和密码。为了简便起见，定义了两类用户：普通用户和超级用户。登录页面的文件名为Userlogin.asp，其代码如下：

```
<HTML>
<HEAD><TITLE>用户登录</TITLE></HEAD><CENTER>
<FORM action="UserLoginRespond.asp" method="POST">
```

```
<P><FONT size="3"><B>请选择用户名并输入密码</B></FONT></P>
<HR size="1" width='50%'>
<TABLE border=1>
    <TR>
        <TD>用户名:</TD>
        <TD><SELECT name="UserName">
            <option    selected><%=Request.QueryString("UserName")%>
            <option >普通用户
            <option >超级用户
        </TD>
    </TR>
    <TR>
        <TD>密码:</TD>
        <TD><INPUT type="PASSWORD" NAME="UserPassword" size="10" ></TD>
    </TR>
    <TR>
        <TD colspan=2 align="center"><INPUT type="SUBMIT" value="登    录"></TD>
    </TR>
</TABLE></FORM>
<FONT color="red"><%=Request("ErrorMessage")%></FONT>
</CENTER></HTML>
```

Userlogin.asp 页面的显示效果如图 4-19 所示。

图 4-19　登录页面

在Userlogin.asp文件中，使用表单中的下拉列表（UserName）传递用户名，使用类型为“PASSWORD”的单行输入文本标记（UserPassword）传递用户输入的密码。当提交表单后，用户选择的用户名和密码就会传递给UserLoginRespond.asp文件。

注意：在 Userlogin.asp 文件中有以下两个服务器端脚本:

```
<option    selected><%=Request("UserName")%>
<FONT color="red"><%=Request("ErrorMessage")%></FONT>
```

它们的作用是当用户名和密码不正确时重新返回登录页面时显示的内容。

4.4.2　用户验证

在用户验证程序中，要取得登录页面中的用户类型和密码，并检查是否正确。如果正确，

将用户重定向到正确页面，如果没有通过验证，将用户重定向到登录页面，并给出提示信息。

用户验证的文件名为UserLoginRespond.asp，其代码如下：

```
<% Dim strNoName, strBadUserName, strBadPassword,flag
'设置错误信息
strNoName = "请选择用户名并输入密码以登录网站"
strBadUserName = "对不起！输入的用户名错误"
strBadPassword = "对不起！输入的密码错误"
'取得网页表单的值
strUserName = Request.Form("Username")
strUserPassword = Request.Form("Userpassword")
'是否输入用户名和密码
If strUserName = "" or strUserPassword = "" Then
    Response.Redirect "UserLogin.asp?ErrorMessage=" & strNoName & "&UserName=" & strUserName
End If
'检查密码
If strUsername="普通用户" or strUsername="超级用户"  Then
  '密码正确，找到用户
  If strUsername="普通用户" and  strUserPassword="001"  Then
  '进入网站的网页
  Session("UserLevel")=1
  Response.Redirect "main.asp"
  Else If strUsername="超级用户" and  strUserPassword="002" Then
        Session("UserLevel") =2
        Response.Redirect "main.asp"
        Else
        '密码错误
        Response.Redirect "UserLogin.asp?ErrorMessage=" & strBadPassword &_
"&UserName=" & strUserName
        End If
      End if
Else
   '用户错误
  Response.Redirect "UserLogin.asp?ErrorMessage=" & strBadPassword &_
"&UserName=" & strUserName
strUserName
End If  %>
```

在UserLoginRespond.asp文件中，使用Response.Redirect方法将用户重定向到不同的页面，当验证正确时的代码如下：

```
Session("UserLevel") =2
Response.Redirect "main.asp"
```

这里使用Session变量是为了进入其后的页面时能够区分出不同的用户。关于Session的详细内容，请参阅后面的章节。main.asp代码如下：

```
<% If Session("UserLevel")=1 or Session("UserLevel")=2 Then
    Response.Write "欢迎光临本网站！"
End If%>
```

当验证错误时，代码为：

```
Response.Redirect "UserLogin.asp?ErrorMessage="&strBadPassword&"UserName=" & strUserName
```

这段代码将用户重定向到登录页面的同时，向 Userlogin.asp 文件传递了两个 QueryString 参数：ErrorMessage 和 UserName，以使重新回到登录页面时能够显示一些信息。

用户输入不同情况的显示效果如图 4-20 所示。

图 4-20　不同登录情况的显示效果

思考题

1．ASP 提供了哪几个内置对象？简述其各自的功能。

2．如果在 ASP 程序中要向浏览器中输出“%>”和“""”两个字符，如何编写输出语句？

3．在使用 Response 对象的 Redirect 方法时应注意什么？

4．简述 Cookie 的作用及存储位置和存储文件名。

5．当利用 Request 对象的数据集合取得数据时，如果省略数据集合，ASP 会按照什么样的顺序搜索该变量？

6．当表单中的数据通过 GET 方式提交时，需要使用 Request 对象的哪种数据集合获取数据？

上机实验

实验目的：

1．掌握 Response 对象的使用方法。

2．掌握 Request 对象的使用方法。

3．灵活使用 ASP 内置对象的能力。

实验内容：

1．编写一个 ASP 文件，要求在客户端浏览器中以不同的字号显示“Hello World!”

2．编写一个 ASP 文件，要求用 Response.Write 向客户端浏览器输出显示一个完整的表格和指向第一个文件的超链接。

3．自行设计两个 ASP 程序，用于验证 Response.Buffer 的不同取值。

4．利用 Response.Redirect 方法实现：当用户在星期一到星期五访问页面时，显示“工作时间!”；其余时间显示“休息时间!”。

5．编写一个 HTML 文件和一个 ASP 文件，在 HTML 文件中利用表单完成用户调查信息的录入，要求

包含文本框、单选按钮、复选框、下拉列表等，使用 POST 方式将表单中的数据提交给 ASP 文件后，在 ASP 文件中将用户的数据回显。

6．使用 GET 方式提交上述表单内容（注意：提交的数据要少一些），重新完成实验内容 5 的功能。

7．制作两个自响应文件分别完成实验 5、6 的功能。

8．利用 QueryString 向客户端浏览器输出一组超级链接（指向同一文件，但参数不同），要求当用户单击超级链接时，显示该链接对应的参数名称及其取值。

9．参照书上的例题，制作两个完成用户注册功能的页面。注册页面包含用户名、密码和确认密码等输入信息，要求：用户名、密码和确认密码都不能为空且密码和确认密码应相同，否则返回注册页面，并给出提示信息；当输入正确后，给出“注册成功”信息。

10．编写一个 HTML 文件和两个 ASP 文件。在 HTML 文件中完成用户登录信息的输入，在第一个 ASP 文件中获取该用户名并将用户名写入 Cookie，在第二个 ASP 文件中读取 Cookie 并显示该用户名。