W3School VBS & ASP教程

来源：www.w3school.com.cn
整理：飞龙
日期：2014.9.30

VBScript 简介

学习之前，您需要具备的基础知识：

在继续学习之前，您应该对下面的知识有基本的了解：

- WWW, HTML 以及网站建设的基础知识

如果您希望首先学习以上的项目，请访问我们的首页。

什么是VBScript?

- VBScript 是一种脚本语言
- 脚本语言是一种轻量级的编程语言
- VBScript 是微软的编程语言 Visual Basic 的轻量级的版本

它如何工作？

当VBScript 被插入一个 HTML 文档后，因特网浏览器会读取这个文档，并对 VBScript 进行解释。VBScript 可能会立即执行，也可能在之后的事件发生时执行。

VBScript How To ...

实例

使用 *VBScript* 写文本

如何在页面上写文本。

```
<html>
<body>

<script type="text/vbscript">
document.write("Hello from VBScript!")
</script>

</body>
</html>
```

使用 ***HTML*** 标签格式化文本

如何协同使用 HTML 标签和 VBScript。

```
<html>
<body>

<script type="text/vbscript">
document.write("<h1>Hello World!</h1>")
document.write("<h2>Hello World!</h2>")
</script>

</body>
</html>
```

如何在 **HTML** 文档中放置 **VBScript**

```
<html>
<head>
</head>
<body>

<script type="text/vbscript">
document.write("Hello from VBScript!")
</script>

</body>
</html>
```

上面的代码会生成以下输出：

```
Hello from VBScript!
```

如需在 HTML 文档中插入脚本，请使用 <script> 标签。使用 type 属性来定义脚本语言。

```
<script type="text/vbscript">
```

然后输入 VBScript：在页面上写文本的命令是 document.write：

```
document.write("Hello from VBScript!")
```

脚本在此结束：

```
</script>
```

如何应对老式的浏览器

不支持脚本的老式浏览器会把脚本作为网页的内容显示出来。为了避免出现这样的情况，我们可以使用

HTML 的注释标签：

```
<script type="text/vbscript">
<!--
  在此输入语句
-->
</script>
```

VBScript Where To ...

实例

head 部分中的函数

可以把脚本放置在 head 部分。经常，我们会把所有的函数放置于 head 部分。这么做的目的是为了确保函数在调用前已经被载入。

```
<html>
<head>
<script type="text/vbscript">
alert("Hello")
</script>
</head>

<body>
<p>
通常，我们在 head 部分放置函数。理由是，可以确保函数在调用前已经加载。
</p>

</body>
</html>
```

body 部分中的脚本

本例会执行被放置于 body 部分的一段脚本。在 body 中的脚本会在页面载入时执行。

```
<html>
<body>

<script type="text/vbscript">
document.write("在页面加载时，会执行 body 部分的脚本。")
</script>

</body>
</html>
```

在何处放置 **VBScript**

当页面载入浏览器时，页面中的脚本会立即被执行。我们并不希望这种情况发生。有时我们希望当页面

载入时执行脚本，而有时我们则希望当用户触发某个事件时执行这些脚本。

在 head 部分的脚本：当脚本被调用时，它们会被执行，或者某个事件被触发时，脚本也有可能会执行。当我们把脚本放置于 head 部分时，就可以确保在用户使用之前它们已经被载入了：

```
<html>
<head>
<script type="text/vbscript">
  some statements
</script>
</head>
```

在 body 部分的脚本：当页面的 body 部分被载入时，脚本就会被执行。当我们把脚本放置于 body 部分，它会生成页面的内容：

```
<html>
<head>
</head>
<body>
<script type="text/vbscript">
  some statements
</script>
</body>
```

位于 body 和 head 部分的脚本：您可以在文档中放置如何数量的脚本，因此您可以同时在 body 和 head 部分放置脚本：

```
<html>
<head>
<script type="text/vbscript">
  some statements
</script>
</head>
<body>
<script type="text/vbscript">
  some statements
</script>
</body>
```

VBScript 变量

实例

创建变量

变量用于存储信息。本例演示如何创建一个变量，并为它赋值。

```
<html>
<body>

<script type="text/vbscript">
dim name
name="John Adams"
document.write(name)
</script>

</body>
</html>
```

在一段文本中插入变量值

本例为您演示如何在一段文本中插入变量值。

```
<html>
<body>

<script type="text/vbscript">
dim name
name="John Adams"
document.write("My name is: " & name)
</script>

</body>
</html>
```

创建数组

数组用来存储一系列相关的数据项。本例演示如何创建一个存储名字的数组。（我们使用 "for loop" 来演示如何输出名字。）

```
<html>
<body>

<script type="text/vbscript">

dim fname(5)
fname(0) = "George"
fname(1) = "John"
fname(2) = "Thomas"
fname(3) = "James"
fname(4) = "Adrew"
fname(5) = "Martin"

for i=0 to 5
  document.write(fname(i) & "<br />")
next

</script>
```

```
</body>
</html>
```

什么是变量？

变量是可存储信息的“容器”。在脚本中，变量的值是可以改变的。您可以通过引用某个变量的名称，来查看或修改它的值。在 VBScript 中，所有的变量都与类型相关，可存储不同类型的数据。

变量名称的规则：

- 必须以字母开头
- 不能包含点号 (.)
- 不能超过 255 个字符

变量声明

您可以使用 Dim、Public 或 Private 语句来声明变量，比如这样：

```
dim name
name=some value
```

现在，你创建了一个变量。变量名是 "name"。

您也可以通过使用其名称来创建变量。比如这样：

```
name=some value
```

这样，您同样创建了一个名为 "name" 的变量。

不过，后面这样的做法不是一种好习惯，这是因为您可能会在脚本中拼错变量名，那样可能会在脚本运行时引起奇怪的结果。比如，当您把 "name" 变量错拼为 "nime" 时，脚本会自动创建一个名为 "nime" 的变量。为了防止脚本这样做，您可以使用 Option Explicit 语句。如果您使用这个语句，就必须使用 dim、public 或 private 语句来声明所有的变量。把 Option Explicit 语句放置于脚本的顶端，这像这样：

```
option explicit
dim name
name=some value
```

为变量赋值

您可以像这样为某个变量赋值：

```
name="George"
i=300
```

变量名在表达式的左侧，需要赋的值在表达式的右侧。现在，变量 "name" 的值是 "George"。

变量的生存期

变量的生存期指的是它可以存在的时长。

当您在一个子程序中声明变量后，变量只能在此程序内进行访问。当退出此程序时，变量也会失效。这样的变量称为本地变量。您可以在不同的子程序中使用名称相同的本地变量，因为每个变量只能在声明它的程序内得到识别。

如果您在子程序以外声明了一个变量，在您的页面上的所有子程序都可以访问访问它。这类变量的生存期始于它们被声明，止于页面被关闭。

数组变量

有时，您需要向一个单一的变量赋于多个值。那么您可以创建一个可包含一系列值的变量。这种变量被称为数组。数组变量的声明使用变量名后跟一个括号()。在下面的例子中，创建了一个包含三个元素的数组：

```
dim names(2)
```

括号中的数字是 2。数组的下标以 0 开始，因为此数组包含三个元素。这是容量固定的数组。您可以为数组的每个元素分配数据：

```
names(0)="George"
names(1)="John"
names(2)="Thomas"
```

同样地，通过使用特定数组元素的下标号，我们也可以取回任何元素的值。比如：

```
father=names(0)
```

您可以在一个数组中使用多达 60 个维数。声明多维数组的方法是在括号中用逗号来分隔数字。比如，我们声明了一个包含 5 行 7 列的 2 维数组：

```
dim table(4, 6)
```

VBScript 程序

实例

子程序

这个子程序不会返回值。

```
<html>

<head>
<script type="text/vbscript">
```

```
sub mySub()
  msgbox("这是一段子程序。")
end sub
</script>
</head>

<body>
<script type="text/vbscript">
call mySub()
</script>
<p>子程序不返回结果。</p>
</body>
</html>
```

函数程序

假如你希望返回某个值时，可以使用函数程序。

```
<html>

<head>
<script type="text/vbscript">
function myFunction()
 myFunction = "蓝色"
end function
</script>
</head>

<body>
<script type="text/vbscript">
document.write("我喜欢的颜色是：" & myFunction())
</script>
<p>函数程序可返回结果。</p>
</body>

</html>
```

VBScript 程序

我们可使用两种程序：子程序和函数程序。

子程序：

- 是一系列的语句，被封装在 Sub 和 End Sub 语句内。
- 可执行某些操作，但不会返回值。
- 可带有通过程序调用来向子程序传递参数。
- 如果没有，必须带有空的圆括号

```
Sub mysub()
 some statements
```

```
End Sub
```

```
Sub mysub(argument1,argument2)
 some statements
End Sub
```

函数程序：

- 是一系列的语句，被封装在 Function 和 End Function 语句内。
- 可执行某些操作并返回值。
- 可带有通过程序调用来向其传递参数。
- 如果没有，必须带有空的圆括号
- 通过向函数程序名赋值的方式，可使其返回值。

```
Function myfunction()
 some statements
 myfunction=some value
End Function
```

```
Function myfunction(argument1,argument2)
 some statements
 myfunction=some value
End Function
```

调用子程序或函数程序

可以这样调用某个函数：

```
name = findname()
```

此函数名为 "findname"，函数会返回一个值，这个值会存储于变量 "name" 中。

或者可以这样做：

```
msgbox "Your name is " & findname()
```

我们通过调用了名为 "findname" 的函数，这个函数返回的值会显示在消息框中。

可以这样调用子程序：

```
Call MyProc(argument)
```

或者，也可以省略 Call 语句：

```
MyProc argument
```

VBScript 条件语句

实例

If...then..else 语句

本例演示如何编写 if...then..else 语句。

```
<html>

<head>
<script type="text/vbscript">
function greeting()
i=hour(time)
if i < 10 then
 document.write("Good morning!")
else
 document.write("Have a nice day!")
end if
end function
</script>
</head>

<body onload="greeting()">
</body>

</html>
```

If...then..elseif 语句

本例演示如何编写 if...then...elseif... 语句。

```
<html>

<head>
<script type="text/vbscript">
function greeting()
i=hour(time)
If i = 10 then
        document.write("Just started...!")
elseif i = 11 then
        document.write("Hungry!")
elseif i = 12 then
        document.write("Ah, lunch-time!")
elseif i = 16 then
        document.write("Time to go home!")
else
        document.write("Unknown")
end if
end function
```

```
</script>
</head>

<body onload="greeting()">
</body>

</html>
```

Select case 语句

本例演示如何编写 select case 语句。

```
<html>

<body>
<script type="text/vbscript">
d=weekday(date)

select case d
  case 1
    document.write("Sleepy Sunday")
  case 2
    document.write("Monday again!")
  case 3
    document.write("Just Tuesday!")
  case 4
    document.write("Wednesday!")
  case 5
    document.write("Thursday...")
  case 6
    document.write("Finally Friday!")
  case else
    document.write("Super Saturday!!!!")
end select
</script>

<p>本例演示 "select case" 语句。<br />
您会得到基于日期的不同问候。<br />
请注意，Sunday=1, Monday=2, Tuesday=3, 以此类推。</p>

</body>
</html>
```

条件语句

经常地，当我们编写代码时，我们需要根据不同的判断执行不同操作。我们可以使用条件语句完成这个工作。

在 **VBScript** 我们可以使用三种条件语句：

if 语句

假如你希望在条件为 true 时执行一系列的代码，可以使用这个语句。

if...then...else 语句

假如你希望执行两套代码其中之一，可以使用这个语句。

if...then...elseif 语句

假如你希望选择多套代码之一来执行，可以使用这个语句。

select case 语句

假如你希望选择多套代码之一来执行，可以使用这个语句。

If....Then.....Else

在下面的情况中，您可以使用 If...Then...Else 语句：

- 在条件为 true 时，执行某段代码
- 选择两段代码之一来执行时

如果需要在条件为 true 时只执行一行语句，可以把代码写为一行：

```
if i=10 Then msgbox "Hello"
```

在上面的代码中，没有 .else.. 语句。我们仅仅让代码在条件为 true 时执行一项操作（当 i 为 10 时）。

假如我们需要在条件为 true 时执行不止一条语句，那么就必须在一行写一条语句，然后使用关键词 "End If" 来结束这个语句：

```
if i=10 Then
   msgbox "Hello"
   i = i+1
end If
```

在上面的代码中，同样没有 .else.. 语句。我们仅仅让代码在条件为 true 时执行了多项操作。

假如我们希望在条件为 true 时执行某条语句，并当条件不为 true 时执行另一条语句，就必须添加关键词 "Else"：

```
if i=10 then
   msgbox "Hello"
else
   msgbox "Goodbye"
end If
```

当条件为 true 时会执行第一段代码，当条件不成立时执行第二段代码（当 i 不等于 10 时）。

If....Then.....Elseif

假如你希望选择多套代码之一来执行，可以使用if...then...elseif语句：

```
if payment="Cash" then
   msgbox "You are going to pay cash!"
 elseif payment="Visa" then
   msgbox "You are going to pay with visa."
 elseif payment="AmEx" then
   msgbox "You are going to pay with American Express."
 else
   msgbox "Unknown method of payment."
end If
```

Select Case

假如你希望选择多套代码之一来执行，可以使用 SELECT 语句：

```
select case payment
 case "Cash"
   msgbox "You are going to pay cash"
 case "Visa"
   msgbox "You are going to pay with visa"
 case "AmEx"
   msgbox "You are going to pay with American Express"
 case Else
   msgbox "Unknown method of payment"
end select
```

以上代码的工作原理：首先，我们需要一个简单的表达式（常常是一个变量），并且这个表达式会被做一次求值运算。然后，表达式的值会与每个 case 中的值作比较，如果匹配，被匹配的 case 所对应的代码会被执行。

VBScript 循环语句

实例

For..next 循环

本例演示如何编写一个简单的 For....Next 循环。

```
<html>
<body>

<script type="text/vbscript">
for i = 0 to 5
 document.write("数字是：" & i & "<br />")
next
</script>
```

```
</body>
</html>
```

循环输出***HTML***标题

本例演示如何循环生成 6 个 HTML 标题。

```
<html>
<body>

<script type="text/vbscript">
for i=1 to 6
 document.write("<h" & i & ">这是标题 " & i & "</h" & i & ">")
next
</script>

</body>
</html>
```

For..each 循环

本例演示如何编写一个简单的 For.....Each 循环。

```
<html>
<body>

<script type="text/vbscript">
dim names(2)
names(0) = "George"
names(1) = "John"
names(2) = "Thomas"

for each x in names
  document.write(x & "<br />")
next
</script>

</body>
</html>
```

Do...While 循环

本例演示如何编写简单的 Do...While 循环。

```
<html>
<body>

<script type="text/vbscript">
i=0
do while i < 10
  document.write(i & "<br />")
```

```
  i=i+1
loop
</script>

</body>
</html>
```

Looping 语句

经常地，当编写代码时，我们希望将一段代码执行若干次。我们可以在代码中使用循环语句来完成这项工作。

在 **VBScript** 中，我们可以使用四种循环语句：

For...Next 语句

运行一段语句指定的次数

For Each...Next 语句

针对集合中的每个项目或者数组中的每个元素来运行某段语句。

Do...Loop 语句

运行循环，当条件为 true 或者直到条件为 true 时。

While...Wend 语句

不要使用这个语句 - 请使用 Do...Loop 语句代替它。

For...Next 循环

如果您已经确定需要重复执行代码的次数，那么您可以使用 For...Next 语句来运行这段代码。

我们可以使用一个计数器变量，这个变量会随着每次循环递增或递减，例如这样：

```
For i=1 to 10
  some code
Next
```

For 语句规定计数变量以及它的开始值和结束值。

Next 语句会以 1 作为步进值来递增变量i。

Step 关键词

通过使用 Step 关键词，我们可以规定计数变量递增或递减的步进值。

在下面的例子中，计数变量i每次循环的递增步进值为 2。

```
For i=2 To 10 Step 2
  some code
```

```
Next
```

如果要递减计数变量，就必须负的步进值。并且需要规定小于开始值的结束值。

在下面的例子中，计数变量i每次循环的递减步进值为 2。

```
For i=10 To 2 Step -2
  some code
Next
```

退出 **For...Next**

如需退出 For...Next 语句，可以使用 Exit 关键词。

ASP 简介

ASP 文件可包含文本、 **HTML** 标签和脚本。**ASP** 文件中的脚本可在服务器上执行。

在学习之前，应具备的知识：

在继续学习之前，您需要对以下知识有基本的了解：

- HTML / XHTML
- 脚本语言，比如 JavaScript 或者 VBScript

如果希望学习上面的项目，请在我们的 首页 访问这些教程。

ASP 是什么？

- ASP 指 Active Server Pages （动态服务器页面）
- ASP 是一项微软公司的技术
- ASP 是在 IIS 中运行的程序
- IIS 指 Internet Information Services （Internet 信息服务）
- IIS 是 Windows 2000 及 Windows 2003 的免费组件
- IIS 同时也是 Windows NT 4.0 的可选组件
- 此可选组件可通过因特网下载
- PWS 的体积更小 - 不过拥有 IIS 的完整功能
- PWS 可在 Windows 95/98 的安装 CD 中找到

ASP 兼容性

- 运行 IIS，需要 Windows NT 4.0 或更高的版本。
- 运行 PWS ，需要 Windows 95 或者更高的版本。
- ChiliASP 是一种在非 Windows 操作系统上运行 ASP 的技术
- InstantASP 是另一种在非 Windows 操作系统上运行 ASP 的技术

ASP 文件是什么？

- ASP 文件和 HTML 文件类似
- ASP 文件可包含文本、HTML、XML 和脚本
- ASP 文件中的脚本可在服务器上执行。
- ASP 文件的扩展名是 ".asp"

ASP 和 HTML 有何不同？

- 当浏览器请求某个 HTML 文件时，服务器会返回这个文件
- 而当浏览器请求某个 ASP 文件时，IIS 将这个请求传递至 ASP 引擎。ASP 引擎会逐行地读取这个文件，并执行文件中的脚本。最后，ASP 文件将以纯 HTML 的形式返回到浏览器。

ASP 能为你做什么？

- 动态地编辑、改变或者添加页面的任何内容
- 对由用户从 HTML 表单提交的查询或者数据作出响应
- 访问数据或者数据库，并向浏览器返回结果
- 为不同的用户定制网页，提高这些页面的可用性
- 用 ASP 替代 CGI 和 Perl 的优势在于它的简易性和速度
- 由于 ASP 代码无法从来浏览器端察看，ASP 确保了站点的安全性。
- 优秀的 ASP 编程可将网络负载降至最低

重要事项：由于 ASP 在服务器上运行，浏览器无需支持客户端脚本就可以显示 ASP 文件！

在自己的 PC 上运行 ASP

您可以在自己的 **PC** 上运行 **ASP** 。

把自己的 Windows PC 作为 Web 服务器

如果您安装了 IIS 或者 PWS，就可以把自己的 PC 配置为一台 web 服务器。

IIS 或 PWS 可以把您的计算机转变为 web 服务器。

微软的 IIS 和 PWS 是免费的 web 服务器组件。

IIS - Internet Information Server

IIS 是一个基于因特网的服务的集合，由微软开发，在 Windows 平台上使用。

Windows 2000、XP、Vista 以及 Windows 7 均提供 IIS。Windows NT 也可用 IIS。

IIS 很容易安装，是开发和测试 web 应用程序的理想工具。

PWS - Personal Web Server

PWS 用于更老的 Windows 系统，比如 Windows 95、98 以及 NT。

PWS 很容易安装，可用于开发和测试包含 ASP 的 web 应用程序。

我们不推荐使用 PWS，除非是用于培训。它已经过时，并存在安全问题。

Windows Web 服务器版本

Windows 版本	是否提供或支持
Windows 7（所有版本）	提供 IIS 7.5
Windows Server 2008	提供 IIS 7
Windows Vista Business, Enterprise 以及 Ultimate	提供 IIS 7
Windows Vista Home Premium	提供 IIS 7
Windows Vista Home Edition	不支持 PWS 或 IIS
Windows Server 2003	提供 IIS 6
Windows XP Professional	提供 IIS 5.1
Windows XP Home Edition	不支持 IIS 或 PWS
Windows 2000 Professional	提供 IIS 5.0
Windows NT Professional	提供 IIS 3，并支持 IIS 4
Windows NT Workstation	支持 PWS 和 IIS 3
Windows ME	不支持 PWS 或 IIS
Windows 98	提供 PWS
Windows 95	支持 PWS

如何在 **Windows 7** 及 **Windows Vista** 上安装 **IIS**

请根据以下四个步骤来安装 IIS：

1. 从开始菜单打开控制面板
2. 双击“程序和功能”
3. 点击“打开或关闭 Windows 功能”
4. 选择 “Internet 信息服务”的复选框，然后点击确定

在您安装完成 IIS 之后，请确保安装所有补丁包（运行 Windows 更新）。

如何在 **Windows Server 2003** 上安装 **IIS** 并运行 **ASP**

1. 当你启动 Windows Server 2003 后，会看到服务器管理向导
2. 如果向导没有显示，可以打开管理工具，然后选择“配置您的服务器向导”
3. 出现提示后，点击下一步
4. 随后会出现一个“预备步骤”的提示，点击下一步，随后系统自动搜索已在本机安装了的系统服务组件
5. 在随后出现的服务器角色窗口中选择应用程序服务器，然后点击下一步
6. 选中启用*ASP.NET*
7. 随后向导会提示这个组件的大概安装过程，请点击下一步
8. 现在，向导会请求 *Server 2003 CD*。请插入CD后继续运行向导。
9. 最后，向导会提示“此服务器目前是一台应用程序服务器”。
10. 点击完成后，你会发现“应用程序服务器”已在管理你的服务器角色窗口中出现
11. 点击“管理此应用程序服务器”会打在应用程序服务器管理控制台*(MMC)*
12. 展开*Internet* 信息服务 *(IIS)*管理器，然后展开你的服务器，然后是站点文件夹。
13. 你会看到默认的网站，并且它的状态应该是运行中。
14. 在Internet 信息服务 (IIS)管理器中点击“*Web*服务扩展”，你会看到 Active Server Pages 是被禁止的。
15. 选中*Active Server Pages*，然后点击允许按钮，这样 ASP 就被激活了！

如何在 **Windows 2000** 上安装 **IIS** 并运行**ASP**

1. 开始按钮 - 设置 - 控制面板
2. 在控制面板中选择添加/删除程序
3. 在添加/删除程序中选择添加/删除*Windows*组件
4. 在向导窗口中选中 *Internet* 信息服务，然后点击确定
5. *Inetpub* 文件夹会在硬盘中被创建
6. 打开 Inetpub 文件夹，找到名为 *wwwroot* 的文件夹
7. 在 wwwroot 下创建一个新文件夹，比如 "MyWeb"
8. 使用文本编辑器编写几行 ASP 代码，将这个文件取名为 "test1.asp" 保存在 "MyWeb" 文件夹。
9. 确保你的服务器正在运行 - 安装程序会在系统托盘创建一个IIS的图标。点击这个图标，然后在出现的新窗口中按开始按钮。
10. 打开浏览器，在地址栏键入 "http://localhost/MyWeb/test1.asp"，就可以看到你的第一个 ASP 页面了。

如何在 **Windows XP Professional** 上安装 **IIS** 并运行 **ASP**

1. 在 CD-Rom 驱动器中插入 Windows XP Professional CD-Rom
2. 开始菜单，设置，控制面板
3. 在控制面板选择添加/删除程序
4. 在添加/删除程序中选择添加/删除 *Windows* 组件
5. 在向导窗口中选中 *Internet Information Services*，然后点击确定
6. *Inetpub* 文件夹会在硬盘中创建
7. 打开 Inetpub 文件夹，找到名为 *wwwroot* 的文件夹
8. 在 wwwroot下创建一个新文件夹，比如 "MyWeb"
9. 使用文本编辑器编写几行 ASP 代码，将这个文件取名为 "test1.asp" 保存在 "MyWeb" 文件夹。

10. 确保你的服务器正在运行，使用下面的方法确认它的运行状态：进入控制面板，然后是管理工具，然后双击“*IIS* 管理器”图标。
11. 打开浏览器，在地址栏键入 "http://localhost/MyWeb/test1.asp"，就可以看到你的第一个 ASP 页面了。

提示：Windows XP Home Edition 无法运行 ASP。

如何在老版本的 **Windows** 操作系统中运行 **ASP**

如何在 **Windows 95** 上安装 **PWS** 并运行 **ASP**

Windows 95 不包含 PWS！！

要想在 Windows 95 上运行 ASP，就必须从微软的站点下载 “Windows NT 4.0 Option Pack”。

下载"Windows NT 4.0 Option Pack"

如何在 **Windows NT** 上安装 **PWS** 并运行 **ASP**

Windows NT 同样不包含 PWS！！

要想在 Windows NT 上运行 ASP，就必须从微软的站点下载 “Windows NT 4.0 Option Pack”。

下载"Windows NT 4.0 Option Pack"

如何在 **Windows 98** 上安装 **PWS** 并运行 **ASP**

1. 打开 Windows 98 CD 上的 Add-ons 文件夹，找到PWS文件夹并运行其中的 setup.exe 文件。
2. 安装程序会在硬盘创建一个 Inetpub 文件夹。打开这个文件夹，找到 wwwroot 文件夹。
3. 然后在 wwwroot 文件夹下面创建一个新的文件夹，比如 "MyWeb"。
4. 使用文本编辑器编写几行 ASP 代码，将这个文件取名为 "test1.asp" 保存在 "MyWeb" 文件夹。
5. 确保你的服务器正在运行 - 安装程序会在系统托盘创建一个PWS的图标。点击这个图标，然后在出现的新窗口中按开始按钮。
6. 打开浏览器，在地址栏键入 "http://localhost/MyWeb/test1.asp"，就可以看到你的第一个 ASP 页面了。

如何在 **Windows ME** 上安装 **PWS** 并运行 **ASP**

Windows ME 同样不包含 PWS！！

阅读微软站点的信息

来自 Bill James 的安装方法。

ASP 语法

在浏览器中通过查看源代码的方式是无法看到 **ASP** 源代码的，你只能看到由 **ASP** 文件输出的结果，而那些只是纯粹的 **HTML** 而已。这是因为，在结果被送回浏览器前，脚本已经在服务器上执

行了。

在我们的 **ASP** 教程中，每个例子都提供隐藏的 **ASP** 代码。这样会使您更容易理解它们的工作原理。

实例

用 *ASP* 写文本

如果使用 ASP 生成文本。

```
<html>
<body>

<%
response.write("Hello World!")
%>

</body>
</html>
```

向文本添加 *HTML*

如果同时生成 HTML 标签和纯文本。

```
<html>
<body>
<%
response.write("<h2>您可以使用 HTML 标签来格式化文本</h2>")
%>

<%
response.write("<p style='color:#0000ff'>这段文本的样式是通过 style 属性添加的。</p>")
%>
</body>
</html>
```

基本的 **ASP** 语法规则

通常情况下，ASP 文件包含 HTML 标签，类似 HTML 文件。不过，ASP 文件也能够包含服务器端脚本，这些脚本被分隔符 **<%** 和 **%>** 包围起来。

服务器脚本在服务器上执行，可包含合法的表达式、语句、或者运算符。

向浏览器写输出

response.write 命令用来向浏览器写输出。下面的例子向浏览器传送了一段文本："Hello World"。

```
<html>
<body>
<%
response.write("Hello World!")
%>
</body>
</html>
```

还有一种 response.write 命令的简写方法。下面的例子和上面的例子是等效的：

```
<html>
<body>
<%="Hello World!"%>
</body>
</html>
```

在 **ASP** 中使用 **VBScript**

你可以在 ASP 中使用若干种脚本语言。不过，默认的脚本语言是 VBScript：

```
<html>
<body>
<%
response.write("Hello World!")
%>
</body>
</html>
```

上面的例子向文档的 body 部分写入了文本 "Hello World!"。

提示：如果您需要了解更多有关 VBScript 的知识，请学习我们的 VBScript 教程。

在 **ASP** 中使用 **JavaScript**

如果需要使用 JavaScript 作为某个特定页面的默认脚本语言，就必须在页面的顶端插入一行语言设定：

```
<%@ language="javascript"%>
<html>
<body>
<%
Response.Write("Hello World!")
%>
</body>
</html>
```

注意：与 VBScript 不同 - JavaScript 对大小写敏感。所以你需要根据 JavaScript 的需要使用不同的大小写字母编写 ASP 代码。

提示：如果您需要了解更多有关 JavaScript 的知识，请学习我们的 JavaScript 教程。

其他的脚本语言

ASP 与 VBScript 和 JScript 的配合是原生性的（JScript 是微软的 JavaScript 实现）。如果你需要使用其他语言编写脚本，比如 PERL、REXX 或者 Python，那就必须安装相应的脚本引擎。

重要事项：因为脚本在服务器端执行，所以显示 ASP 文件的浏览器根本无需支持脚本。

ASP 变量

变量用于存储信息。

假如在子程序之外声明变量，那么这个变量可被 **ASP** 文件中的任何脚本改变。假如在子程序中声明变量，那么当子程序每次执行时，它才会被创建和撤销。

实例：

声明变量

变量用于存储信息。本例演示如何声明变量，为变量赋值，并在程序中使用这个变量

```
<html>
<body>

<%
dim name
name="Donald Duck"
response.write("My name is: " & name)
%>

</body>
</html>
```

声明数组

数组用于存储一系列相关的数据项目。本例演示如何声明一个存储名字的数组。

```
<html>
<body>

<%
Dim fname(5),i
fname(0) = "George"
fname(1) = "John"
fname(2) = "Thomas"
fname(3) = "James"
fname(4) = "Adrew"
```

```
fname(5) = "Martin"

For i = 0 to 5
      response.write(fname(i) & "<br />")
Next
%>

</body>
</html>
```

循环生成 *HTML* 标题

如何循环生成 6 个不同的 HTML 标题。

```
<html>
<body>

<%
dim i
for i=1 to 6
   response.write("<h" & i & ">Header " & i & "</h" & i & ">")
next
%>

</body>
</html>
```

使用 *Vbscript* 制作基于时间的问候语

本例将根据服务器时间向用户显示不同的消息。

```
<html>
<body>
<%
dim h
h=hour(now())

response.write("<p>" & now())
response.write(" (Beijing Time) </p>")
If h<12 then
   response.write("Good Morning!")
else
   response.write("Good day!")
end if
%>
</body>
</html>
```

使用 *JavaScript* 制作基于时间的问候语

本例同上，只是语法不同而已。

```
<%@ language="javascript" %>
<html>
<body>
<%
var d=new Date()
var h=d.getHours()

Response.Write("<p>")
Response.Write(d + " (Beijing Time)")
Response.Write("</p>")
if (h<12)
   {
   Response.Write("Good Morning!")
   }
else
   {
   Response.Write("Good day!")
   }
%>
</body>
</html>
```

变量的生存期

在子程序外声明的变量可被 ASP 文件中的任何脚本访问和修改。

在子程序中声明的变量只有当子程序每次执行时才会被创建和撤销。子程序外的脚本无法访问和修改该变量。

如需声明供多个 ASP 文件使用的变量，请将变量声明为 session 变量或者 application 变量。

Session 变量

Session 变量用于存储单一用户的信息，并且对一个应用程序中的所有页面均有效。存储于 session 中的典型数据是姓名、id 或参数。

Application 变量

Application 变量同样对一个应用程序中的所有页面均有效。Application 变量用于存储一个特定的应用程序中所有用户的信息。

ASP 子程序

在 **ASP** 中，你可通过 **VBScript** 和其他方式调用子程序。

实例：

调用使用 ***VBScript*** 的子程序

如何从 ASP 调用以 VBScript 编写的子程序。

```
<html>

<head>
<%
sub vbproc(num1,num2)
response.write(num1*num2)
end sub
%>
</head>

<body>
<p>您可以像这样调用一个程序：</p>
<p>结果：<%call vbproc(3,4)%></p>

<p>或者，像这样：</p>
<p>结果：<%vbproc 3,4%></p>
</body>

</html>
```

调用使用 ***JavaScript*** 的子程序

如何从 ASP 调用以 JavaScript 编写的子程序。

```
<%@ language="javascript" %>
<html>
<head>
<%
function jsproc(num1,num2)
{
Response.Write(num1*num2)
}
%>
</head>

<body>
<p>
结果：<%jsproc(3,4)%>
</p>
</body>

</html>
```

调用使用 *VBScript* 和 *JavaScript* 的子程序

如何在一个 ASP 文件中调用以 VBScript 和 JavaScript 编写的子程序。

```
<html>
<head>
<%
sub vbproc(num1,num2)
Response.Write(num1*num2)
end sub
%>
<script  language="javascript" runat="server">
function jsproc(num1,num2)
{
Response.Write(num1*num2)
}
</script>
</head>

<body>
<p>结果: <%call vbproc(3,4)%></p>
<p>结果: <%call jsproc(3,4)%></p>
</body>

</html>
```

子程序

ASP 源代码可包含子程序和函数：

```
<html>
<head>
<%
sub vbproc(num1,num2)
response.write(num1*num2)
end sub
%>
</head>

<body>
<p>Result: <%call vbproc(3,4)%></p>
</body>

</html>
```

将 <%@ language="*language*" %> 这一行写到 <html> 标签的上面，就可以使用另外一种脚本语言来编写子程序或者函数：

```
<%@ language="javascript" %>
```

```
<html>
<head>
<%
function jsproc(num1,num2)
{
Response.Write(num1*num2)
}
%>
</head>

<body>
<p>Result: <%jsproc(3,4)%></p>
</body>

</html>
```

VBScript 与 JavaScript 之间的差异

当从一个用 VBScript 编写的 ASP 文件中调用 VBScript 或者 JavaScript 子程序时，可以使用关键词 "call"，后面跟着子程序名称。假如子程序需要参数，当使用关键词 "call" 时必须使用括号包围参数。假如省略 "call"，参数则不必由括号包围。假如子程序没有参数，那么括号则是可选项。

当从一个用 JavaScript 编写的 ASP 文件中调用 VBScript 或者 JavaScript 子程序时，必须在子程序名后使用括号。

ASP 表单和用户输入

Request.QueryString 和 **Request.Form** 命令可用于从表单取回信息，比如用户的输入。

实例：

使用 ***method="get"*** 的表单

如何使用 Request.QueryString 命令与用户进行交互。

```
<html>
<body>
<form action="/example/aspe/demo_aspe_reqquery.asp" method="get">
您的姓名：<input type="text" name="fname" size="20" />
<input type="submit" value="提交" />
</form>
<%
dim fname
fname=Request.QueryString("fname")
If fname<>"" Then
      Response.Write("你好！ " & fname & "！<br />")
      Response.Write("今天过得怎么样？")
End If
```

```
%>
</body>
</html>
```

使用 ***method="post"*** 的表单

如何使用 Request.Form 命令与用户进行交互。

```
<html>
<body>
<form action="/example/aspe/demo_aspe_simpleform.asp" method="post">
您的姓名：<input type="text" name="fname" size="20" />
<input type="submit" value="提交" />
</form>
<%
dim fname
fname=Request.Form("fname")
If fname<>"" Then
      Response.Write("您好！ " & fname & "！<br />")
      Response.Write("今天过得怎么样？")
End If
%>
</body>
</html>
```

使用单选按钮的表单

如何使用 Request.Form 通过单选按钮与用户进行交互。

```
<html>
<%
dim cars
cars=Request.Form("cars")
%>
<body>
<form action="/example/aspe/demo_aspe_radiob.asp" method="post">
<p>请选择您喜欢的汽车：</p>

<input type="radio" name="cars"
<%if cars="Volvo" then Response.Write("checked")%>
value="Volvo">Volvo</input>
<br />
<input type="radio" name="cars"
<%if cars="Saab" then Response.Write("checked")%>
value="Saab">Saab</input>
<br />
<input type="radio" name="cars"
<%if cars="BMW" then Response.Write("checked")%>
value="BMW">BMW</input>
<br /><br />
<input type="submit" value="提交" />
```

```
</form>
<%
if cars<>"" then
   Response.Write("<p>您喜欢的汽车是" & cars & "</p>")
end if
%>
</body>
</html>
```

用户输入

Request 对象可用于从表单取回用户信息。

HTML 表单实例

```
<form method="get" action="simpleform.asp">
<p>First Name: <input type="text" name="fname" /></p>
<p>Last Name: <input type="text" name="lname" /></p>
<input type="submit" value="Submit" />
</form>
```

用户输入的信息可通过两种方式取回：Request.QueryString 或 Request.Form。

Request.QueryString

Request.QueryString 命令用于搜集使用 method="get" 的表单中的值。使用 GET 方法从表单传送的信息对所有的用户都是可见的（出现在浏览器的地址栏），并且对所发送信息的量也有限制。

HTML 表单实例

```
<form method="get" action="simpleform.asp">
<p>First Name: <input type="text" name="fname" /></p>
<p>Last Name: <input type="text" name="lname" /></p>
<input type="submit" value="Submit" />
</form>
```

如果用户在上面的表单实例中输入 "Bill" 和 "Gates"，发送至服务器的 URL 会类似这样：

```
http://www.w3school.com.cn/simpleform.asp?fname=Bill&lname=Gates
```

假设 ASP 文件 "simpleform.asp" 包含下面的代码：

```
<body>
Welcome
<%
response.write(request.querystring("fname"))
response.write(" " & request.querystring("lname"))
%>
```

```
</body>
```

浏览器将显示如下：

```
Welcome Bill Gates
```

Request.Form

Request.Form 命令用于搜集使用 "post" 方法的表单中的值。使用 POST 方法从表单传送的信息对用户是不可见的，并且对所发送信息的量也没有限制。

HTML 表单实例

```
<form method="post" action="simpleform.asp">
<p>First Name: <input type="text" name="fname" /></p>
<p>Last Name: <input type="text" name="lname" /></p>
<input type="submit" value="Submit" />
</form>
```

如果用户在上面的表单实例中输入 "Bill" 和 "Gates"，发送至服务器的 URL 会类似这样：

```
http://www.w3school.com.cn/simpleform.asp
```

假设 ASP 文件 "simpleform.asp" 包含下面的代码：

```
<body>
Welcome
<%
response.write(request.form("fname"))
response.write(" " & request.form("lname"))
%>
</body>
```

浏览器将显示如下：

```
Welcome Bill Gates
```

表单验证

只要有可能，就应该对用户输入的数据进行验证（通过客户端的脚本）。浏览器端的验证速度更快，并可以减少服务器的负载。

如果用户数据会输入到数据库中，那么你应该考虑使用服务器端的验证。有一种在服务器端验证表单的好的方式，就是将（验证过的）表单传回表单页面，而不是转至不同的页面。用户随后就可以在同一个页面中得到错误的信息。这样做的话，用户就更容易发现错误了。

ASP Cookie

cookie 常用来对用户进行识别。

实例

Welcome cookie

如何创建欢迎 cookie。

```
<%
dim numvisits
response.cookies("NumVisits").Expires=date+365
numvisits=request.cookies("NumVisits")

if numvisits="" then
   response.cookies("NumVisits")=1
   response.write("欢迎！这是您第一次访问本页面。")
else
   response.cookies("NumVisits")=numvisits+1
   response.write("之前，您已经访问过本页面 ")
   response.write(numvisits & " 次。")
end if
%>
<html>
<body>
</body>
</html>
```

什么是 **Cookie?**

cookie 常用来对用户进行识别。cookie 是一种服务器留在用户电脑中的小文件。每当同一台电脑通过浏览器请求页面时，这台电脑也会发送 cookie。通过 ASP，您能够创建并取回 cookie 的值。

如何创建 **cookie?**

"Response.Cookies" 命令用于创建 cookie。

注意：Response.Cookies 命令必须位于 <html> 标签之前。

在下面的例子中，我们会创建一个名为 "firstname" 的 cookie，并向其赋值 "Alex"：

```
<%
Response.Cookies("firstname")="Alex"
%>
```

向 cookie 分配属性也是可以的，比如设置 cookie 的失效时间：

```
<%
```

```
Response.Cookies("firstname")="Alex"
Response.Cookies("firstname").Expires=#May 10,2020#
%>
```

如何取回 **cookie** 的值？

"Request.Cookies" 命令用于取回 cookie 的值。

在下面的例子中，我们取回了名为 "firstname" 的 cookie 的值，并把值显示到了页面上：

```
<%
fname=Request.Cookies("firstname")
response.write("Firstname=" & fname)
%>
```

输出：

```
Firstname=Alex
```

带有键的 **cookie**

如果一个 cookie 包含多个值的一个集合，我们就可以说 cookie 拥有键（Keys）。

在下面的例子中，我们会创建一个名为 "user" 的 cookie 集。"user" cookie 拥有包含用户信息的键：

```
<%
Response.Cookies("user")("firstname")="John"
Response.Cookies("user")("lastname")="Adams"
Response.Cookies("user")("country")="UK"
Response.Cookies("user")("age")="25"
%>
```

读取所有的 **cookie**

请阅读下面的代码：

```
<%
Response.Cookies("firstname")="Alex"
Response.Cookies("user")("firstname")="John"
Response.Cookies("user")("lastname")="Adams"
Response.Cookies("user")("country")="UK"
Response.Cookies("user")("age")="25"
%>
```

假设您的服务器将所有的这些 cookie 传给了某个用户。

现在，我们需要读取这些 cookie。下面的例子向您展示如何做到这一点（请注意，下面的代码会使用 HasKeys 检查 cookie 是否拥有键）：

```
<html>
<body>

<%
dim x,y

 for each x in Request.Cookies
  response.write("<p>")
  if Request.Cookies(x).HasKeys then
    for each y in Request.Cookies(x)
      response.write(x & ":" & y & "=" & Request.Cookies(x)(y))
      response.write("<br />")
    next
  else
    Response.Write(x & "=" & Request.Cookies(x) & "<br />")
  end if
  response.write "</p>"
next
%>

</body>
</html>
```

输出：

```
firstname=Alex

user:firstname=John
user:lastname=Adams
user:country=UK
user:age=25
```

如何应对不支持 **cookie** 的浏览器？

如果您的应用程序需要和不支持 cookie 的浏览器打交道，那么您不得不使用其他的办法在您的应用程序中的页面之间传递信息。这里有两种办法：

1. 向 **URL** 添加参数

您可以向 URL 添加参数：

```
<a href="welcome.asp?fname=John&lname=Adams">
Go to Welcome Page
</a>
```

然后在类似于下面这个 "welcome.asp" 文件中取回这些值：

```
<%
fname=Request.querystring("fname")
```

```
lname=Request.querystring("lname")
response.write("<p>Hello " & fname & " " & lname & "!</p>")
response.write("<p>Welcome to my Web site!</p>")
%>
```

2. 使用表单

您还可以使用表单。当用户点击提交按钮时，表单会把用户输入的数据提交给 "welcome.asp" ：

```
<form method="post" action="welcome.asp">
First Name:  <input type="text" name="fname" value="">
Last Name: <input type="text" name="lname" value="">
<input type="submit" value="Submit">
</form>
```

然后在 "welcome.asp" 文件中取回这些值，就像这样：

```
<%
fname=Request.form("fname")
lname=Request.form("lname")
response.write("<p>Hello " & fname & " " & lname & "!</p>")
response.write("<p>Welcome to my Web site!</p>")
%>
```

ASP Session 对象

Session 对象用于存储用户的信息。存储于 **session** 对象中的变量持有单一用户的信息，并且对于一个应用程序中的所有页面都是可用的。

Session 对象

当您操作某个应用程序时，您打开它，做些改变，然后将它关闭。这很像一次对话（Session）。计算机知道您是谁。它清楚您在何时打开和关闭应用程序。但是在因特网上有一个问题：由于 HTTP 地址无法存留状态，web 服务器并不知道您是谁以及您做了什么。

ASP 通过为每位用户创建一个唯一的 cookie 的方式解决了这个问题。cookie 被传送至客户端，它含有可识别用户的信息。这种接口被称作 Session 对象。

Session 对象用于存储关于用户的信息，或者为一个用户的 session 更改设置。存储于 session 对象中的变量存有单一用户的信息，并且对于应用程序中的所有页面都是可用的。存储于 session 对象中的信息通常是 name、id 以及参数。服务器会为每个新的用户创建一个新的 Session，并在 session 到期时撤销掉这个 Session 对象。

Session 何时开始？

Session 开始于：

- 当某个新用户请求了一个 ASP 文件，并且 Global.asa 文件引用了 Session_OnStart 子程序时；
- 当某个值存储在 Session 变量中时；
- 当某个用户请求了一个 ASP 文件，并且 Global.asa 使用 <object> 标签通过 session 的 scope 来例示某个对象时；

Session 何时结束?

假如用户没有在规定的时间内在应用程序中请求或者刷新页面，session 就会结束。默认值为 20 分钟。

如果您希望将超时的时间间隔设置得更长或更短，可以设置 *Timeout* 属性。

下面的例子设置了 5 分钟的超时时间间隔：

```
<%
Session.Timeout=5
%>
```

要立即结束 session，可使用 *Abandon* 方法：

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

注意：使用 session 时主要的问题是它们该在何时结束。我们不会知道用户最近的请求是否是最后的请求。因此我们不清楚该让 session“ 存活”多久。为某个空闲的 session 等待太久会耗尽服务器的资源。然而假如 session 被过早地删除，那么用户就不得不一遍又一遍地重新开始，这是因为服务器已经删除了所有的信息。寻找合适的超时间隔时间是很困难的。

提示：如果您正在使用 session 变量，请不要在其中存储大量的数据。

存储和取回 session 变量

Session 对象最大的优点是可在其中存储变量，以供后续的网页读取，其应用范围是很广的。

下面的例子把 "Donald Duck" 赋值给名为 username 的 session 变量，并把 "50" 赋值给名为 age 的 session 变量：

```
<%
Session("username")="Donald Duck"
Session("age")=50
%>
```

一旦值被存入 session 变量，它就能被 ASP 应用程序中的任何页面使用：

```
Welcome <%Response.Write(Session("username"))%>
```

上面这行程序返回的结果是: "Welcome Donald Duck"。

也可以在 session 对象中保存用户参数，然后通过访问这些参数来决定向用户返回什么页面。

下面的例子规定，假如用户使用低显示器分辨率，则返回纯文本版本的页面：

```
<%If Session("screenres")="low" Then%>
  This is the text version of the page
<%Else%>
  This is the multimedia version of the page
<%End If%>
```

移除 **session** 变量

contents 集合包含所有的 session 变量。

可通过 remove 方法来移除 session 变量。

在下面的例子中，假如 session 变量 "age" 的值小于 18，则移除 session 变量 "sale"：

```
<%
If Session.Contents("age")<18 then
  Session.Contents.Remove("sale")
End If
%>
```

如需移除 session 中的所有变量，请使用 RemoveAll 方法：

```
<%
Session.Contents.RemoveAll()
%>
```

遍历 **contents** 集合

contents 集合包含所有的 session 变量。可通过遍历 contents 集合，来查看其中存储的变量：

```
<%
Session("username")="Donald Duck"
Session("age")=50

dim i
For Each i in Session.Contents
  Response.Write(i & "<br />")
Next
%>
```

结果：

```
username
age
```

如果需要了解 contents 集合中的项目数量，可使用 count 属性：

```
<%
dim i
dim j
j=Session.Contents.Count
Response.Write("Session variables: " & j)
For i=1 to j
  Response.Write(Session.Contents(i) & "<br />")
Next
%>
```

结果：

```
Session variables: 2
Donald Duck
50
```

遍历 **StaticObjects** 集合

可通过循环 StaticObjects 集合，来查看存储在 session 对象中所有对象的值：

```
<%
dim i
For Each i in Session.StaticObjects
  Response.Write(i & "<br />")
Next
%>
```

ASP Application 对象

在一起协同工作以完成某项任务的一组 **ASP** 文件称作应用程序（**application**）。**ASP** 中的 **Application** 对象用于将这些文件捆绑在一起。

Application 对象

web 上的一个应用程序可以是一组 ASP 文件。这些 ASP 文件一起协同工作来完成某项任务。ASP 中的 Application 对象用来把这些文件捆绑在一起。

Application 对象用于存储和访问来自任何页面的变量，类似于 session 对象。不同之处在于，所有的用户分享一个 Application 对象，而 session 对象和用户的关系是一一对应的。

Application 对象存有会被应用程序中的许多页面使用的信息（比如数据库连接信息）。这意味着可以从任何的页面访问这些信息。同时也意味着你可在一个地点改变这些信息，然后这些改变会自动反映在所

有的页面上。

存储和取回 **Application** 变量

Application 变量可被应用程序中的任何页面访问和改变。

可以像这样在 "Global.asa" 中创建 Application 变量：

```
<script language="vbscript" runat="server">

Sub Application_OnStart
application("vartime")=""
application("users")=1
End Sub

</script>
```

在上面的例子中，我们创建了两个 Application 变量："vartime" 和 "users"。

可以像这样访问 Application 变量的值：

```
There are
<%
Response.Write(Application("users"))
%>
active connections.
```

遍历 **Contents** 集合

Contents 集合包含着所有的 application 变量。我们可以通过对 contents 集合进行遍历，来查看其中存储的变量：

```
<%
dim i
For Each i in Application.Contents
  Response.Write(i & "<br />")
Next
%>
```

如果你不清楚 contents 集中的项目数量，可使用 count 属性：

```
<%
dim i
dim j
j=Application.Contents.Count
For i=1 to j
  Response.Write(Application.Contents(i) & "<br />")
Next
%>
```

遍历 **StaticObjects** 集合

可通过循环 StaticObjects 集合，来查看所有存储于 Application 对象中的对象的值：

```
<%
dim i
For Each i in Application.StaticObjects
  Response.Write(i & "<br />")
Next
%>
```

锁定和解锁

我们可以使用 "Lock" 方法来锁定应用程序。当应用程序锁定后，用户们就无法改变 Application 变量了（除了正在访问 Application 变量的用户）。我们也可使用 "Unlock" 方法来对应用程序进行解锁。这个方法会移除对 Application 变量的锁定：

```
<%
Application.Lock
  'do some application object operations
Application.Unlock
%>
```

ASP 文件引用

#include 指令用于在多重页面上创建需重复使用的函数、页眉、页脚或者其他元素等。

#include 指令

通过使用 #include 指令，我们可以在服务器执行 ASP 文件之前，把另一个ASP文件插入这个文件中。#include 命令用于在多个页面上创建需要重复使用的函数、页眉、页脚或者其他元素等。

如何使用 **#include** 指令

这里有一个名为 "mypage.asp" 的文件:

```
<html>
<body>
<h2>Words of Wisdom:</h2>
<p><!--#include file="wisdom.inc"--></p>
<h2>The time is:</h2>
<p><!--#include file="time.inc"--></p>
</body>
</html>
```

这是 "wisdom.inc" 文件:

```
"One should never increase, beyond what is necessary,
the number of entities required to explain anything."
```

这是 "time.inc" 文件:

```
<%
Response.Write(Time)
%>
```

在浏览器中查看的源代码应该类似这样：

```
<html>
<body>
<h2>Words of Wisdom:</h2>
<p>"One should never increase, beyond what is necessary,
the number of entities required to explain anything."</p>
<h2>The time is:</h2>
<p>11:33:42 AM</p>
</body>
</html>
```

Including 文件的语法：

如需在 ASP 中引用文件，请把 #include 命令置于注释标签之中：

```
<!--#include virtual="somefilename"-->
```

或者：

```
<!--#include file ="somefilename"-->
```

关键词 **Virtual**

关键词 virtual 指示路径以虚拟目录开始。

如果 "header.inc" 文件位于虚拟目录 /html 中，下面这行代码会插入文件 "header.inc" 中的内容：

```
<!-- #include virtual ="/html/header.inc" -->
```

关键词 **File**

关键词 File 指示一个相对的路径。相对路径起始于含有引用文件的目录。

假设文件位于 html 文件夹的子文件夹 headers 中，下面这段代码可引用 "header.inc" 文件的内容：

```
<!-- #include file ="headers\header.inc" -->
```

注意：被引用文件的路径是相对于引用文件的。假如包含 #include 声明的文件不在 html 目录中，这个声明就不会起效。

您也可以使用关键词 file 和语法 (..\) 来引用上级目录中的文件。

提示和注释

在上面的一节中，我们使用 ".inc" 来作为被引用文件的后缀。注意：假如用户尝试直接浏览 INC 文件，这个文件中内容就会暴露。假如被引用的文件中的内容涉及机密，那么最好还是使用 "asp" 作为后缀。ASP 文件中的源代码被编译后是不可见的。被引用的文件也可引用其他文件，同时一个 ASP 文件可以对同一个文件引用多次。

重要事项：在脚本执行前，被引用的文件就会被处理和插入。

下面的代码无法执行，这是由于 ASP 会在为变量赋值之前执行 #include 命令：

```
<%
fname="header.inc"
%>
<!--#include file="<%=fname%>"-->
```

不能在脚本分隔符之间包含文件引用：

```
<%
For i = 1 To n
  <!--#include file="count.inc"-->
Next
%>
```

但是这段脚本可以工作：

```
<% For i = 1 to n %>
<!--#include file="count.inc" -->
<% Next %>
```

ASP Global.asa 文件

Global.asa 文件是一个可选的文件，它可包含可被 **ASP** 应用程序中每个页面访问的对象、变量以及方法的声明。

Global.asa 文件

Global.asa 文件是一个可选的文件，它可包含可被 ASP 应用程序中每个页面访问的对象、变量以及方法的声明。所有合法的浏览器脚本都能在 Global.asa 中使用。

Global.asa 文件可包含下列内容：

- Application 事件
- Session 事件
- <object> 声明
- TypeLibrary 声明
- #include 指令

注释：Global.asa 文件须存放于 ASP 应用程序的根目录中，且每个应用程序只能有一个 Global.asa 文件。

Global.asa 中的事件

在 Global.asa 中，我们可以告知 application 和 session 对象在启动和结束时做什么事情。完成这项任务的代码被放置在事件操作器中。Global.asa 文件能包含四种类型的事件：

Application_OnStart - 此事件会在首位用户从 ASP 应用程序调用第一个页面时发生。此事件会在 web 服务器重起或者 Global.asa 文件被编辑之后发生。"Session_OnStart" 事件会在此事件发生之后立即发生。

Session_OnStart - 此事件会在每当新用户请求他或她的在 ASP 应用程序中的首个页面时发生。

Session_OnEnd - 此事件会在每当用户结束 session 时发生。在规定的时间（默认的事件为 20 分钟）内如果没有页面被请求，session 就会结束。

Application_OnEnd - 此事件会在最后一位用户结束其 session 之后发生。典型的情况是，此事件会在 Web 服务器停止时发生。此子程序用于在应用程序停止后清除设置，比如删除记录或者向文本文件写信息。

Global.asa 文件可能类似这样：

```
<script language="vbscript" runat="server">

sub Application_OnStart
  'some code
end sub

sub Application_OnEnd
  'some code
end sub

sub Session_OnStart
  'some code
end sub

sub Session_OnEnd
  'some code
end sub

</script>
```

注释：由于无法使用 ASP 的脚本分隔符 (<% 和 %>) 在 Global.asa 文件中插入脚本，我们需使用 HTML 的 <script> 元素。

<object> 声明

可通过使用 <object> 标签在 Global.asa 文件中创建带有 session 或者 application 作用域的对象。

注释：<object> 标签应位于 <script> 标签之外。

语法：

```
<object runat="server" scope="scope" id="id"
{progid="progID"|classid="classID"}>
....
</object>
```

参数	描述
scope	设置对象的作用域（作用范围）（Session 或者 Application）。
id	为对象指定一个唯一的 id。
ProgID	与 ClassID 关联的 id。ProgID 的格式是：[Vendor.]Component[.Version]。 ProgID 或 ClassID 必需被指定。
ClassID	为 COM 类对象指定唯一的 id。 ProgID 或 ClassID 必需被指定。

实例

第一个实例创建了一个名为 "MyAd" 且使用 ProgID 参数的 session 作用域对象：

```
<object runat="server" scope="session" id="MyAd" progid="MSWC.AdRotator">
</object>
```

第二个实例创建了名为 "MyConnection" 且使用 ClassID 参数的

```
<object runat="server" scope="application" id="MyConnection"
classid="Clsid:8AD3067A-B3FC-11CF-A560-00A0C9081C21">
</object>
```

在此 Global.asa 文件中声明的这些对象可被应用程序中的任何脚本使用。

GLOBAL.ASA:

```
<object runat="server" scope="session" id="MyAd" progid="MSWC.AdRotator">
</object>
```

您可以从 ASP 应用程序中的任意页面引用此 "MyAd" 对象：

某个 .ASP 文件:

```
<%=MyAd.GetAdvertisement("/banners/adrot.txt")%>
```

TypeLibrary 声明

TypeLibrary （类型库）是一个容器，其中装有对应于 COM 对象的 DLL 文件。通过在 Global.asa 中包含对 TypeLibrary 的调用，可以访问 COM 对象的常量，同时 ASP 代码也能更好地报告错误。假如您的站点的应用程序依赖于已在类型库中声明过数据类型的 COM 对象，您可以在 Global.asa 中对类型库进行声明。

语法：

```
<!--METADATA TYPE="TypeLib"
file="filename"
uuid="typelibraryuuid"
version="versionnumber"
lcid="localeid"
-->
```

参数	描述
file	规定指向类型库的绝对路径。参数 file 或者 uuid，两者缺一不可。
uuid	规定了针对类型库的唯一的标识符。参数 file 或者 uuid，两者缺一不可。
version	可选。用于选择版本。假如没有找到指定的版本，将使用最接近的版本。
lcid	可选。用于类型库的地区标识符。

错误值

服务器会返回以下的错误消息之一：

错误	代码	描述
ASP	0222	Invalid type library specification
ASP	0223	Type library not found
ASP	0224	Type library cannot be loaded
ASP	0225	Type library cannot be wrapped

注释：METADATA 标签可位于 Global.asa 文件中的任何位置（在 <script> 标签的内外均可）。不过，我们还是推荐将 METADATA 标签放置于 Global.asa 文件的顶部。

限定

关于可以在 Global.asa 文件中引用的内容的限定：

你无法显示 Global.asa 文件中的文本。此文件无法显示信息。

你只能在 Application_OnStart 和 Application_OnEnd 子例程中使用 Server 和 Application 对象。在 Session_OnEnd 子例程中，你可以使用 Server、Application 和 Session 对象。在 Session_OnStart 子例程中，你可使用任何内建的对象。

如何使用子例程

Global.asa 常用于对变量进行初始化。

下面的例子展示如何检测访问者首次到达站点的确切时间。时间存储在名为 "started" 的 Session 对象中，并且 "started" 变量的值可被应用程序中的任何 ASP 页面访问：

```
<script language="vbscript" runat="server">
sub Session_OnStart
Session("started")=now()
end sub
</script>
```

Global.asa 也可用于控制页面访问。

下面的例子展示如何把每位新的访问者重定向到另一个页面，在这个例子中会定向到 "newpage.asp" 这个页面：

```
<script language="vbscript" runat="server">
sub Session_OnStart
Response.Redirect("newpage.asp")
end sub
</script>
```

我们还可以在 Global.asa 中包含函数。

在下面的例子中，当 Web 服务器启动时，Application_OnStart 子例程也会启动。随后，Application_OnStart 子例程会调用另一个名为 "getcustomers" 的子例程。"getcustomers" 子例程会打开一个数据库，然后从 "customers" 表中取回一个记录集。此记录集会赋值给一个数组，在不查询数据库的情况下，任何 ASP 页面都能够访问这个数组：

```
<script language="vbscript" runat="server">

sub Application_OnStart
getcustomers
```

```
end sub

sub getcustomers
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
set rs=conn.execute("select name from customers")
Application("customers")=rs.GetRows
rs.Close
conn.Close
end sub

</script>
```

Global.asa 实例

在这个例子中，我们要创建一个可计算当前访客的 Global.asa 文件。

Application_OnStart 设置当服务器启动时，Application 变量 "visitors" 的值为 0。

每当有新用户访问时，Session_OnStart 子例程就会给变量 "visitors" 加 1。

每当 Session_OnEnd 子例程被触发时，此子例程就会从变量 "visitors" 减 1。

Global.asa 文件：

```
<script language="vbscript" runat="server">

Sub Application_OnStart
Application("visitors")=0
End Sub

Sub Session_OnStart
Application.Lock
Application("visitors")=Application("visitors")+1
Application.UnLock
End Sub

Sub Session_OnEnd
Application.Lock
Application("visitors")=Application("visitors")-1
Application.UnLock
End Sub

</script>
```

此 ASP 文件会显示当前用户的数目：

```
<html>
<head>
```

```
</head>
<body>
<p>There are <%response.write(Application("visitors"))%> online now!</p>
</body>
</html>
```

ASP 使用 CDOSYS 发送电子邮件

CDOSYS 是 **ASP** 中的内置组件。此组建用于通过 **ASP** 来发送电子邮件。

使用 CDOSYS 发送电子邮件

CDO (Collaboration Data Objects) 是一项微软的技术，设计目的是用来简化通信程序的创建。

CDOSYS 是 ASP 中的内置组件。我们会向您展示如何使用该组件来发送电子邮件。

CDONTs 怎么样？

微软已经在 Windows 2000、Windows XP 以及 Windows 2003 中淘汰了 CDONTs。如果您还在应用程序中使用 CDONTs，就需要更新代码，并使用新的 CDO 技术。

使用 CDOSYS 的实例

发送电子邮件：

```
<%
Set myMail=CreateObject("CDO.Message")
myMail.Subject="Sending email with CDO"
myMail.From="mymail@mydomain.com"
myMail.To="someone@somedomain.com"
myMail.TextBody="This is a message."
myMail.Send
set myMail=nothing
%>
```

发送带有 Bcc 和 CC 字段的文本邮件:

```
<%
Set myMail=CreateObject("CDO.Message")
myMail.Subject="Sending email with CDO"
myMail.From="mymail@mydomain.com"
myMail.To="someone@somedomain.com"
myMail.Bcc="someoneelse@somedomain.com"
myMail.Cc="someoneelse2@somedomain.com"
myMail.TextBody="This is a message."
myMail.Send
set myMail=nothing
%>
```

发送 HTML 邮件:

```
<%
Set myMail=CreateObject("CDO.Message")
myMail.Subject="Sending email with CDO"
myMail.From="mymail@mydomain.com"
myMail.To="someone@somedomain.com"
myMail.HTMLBody = "<h1>This is a message.</h1>"
myMail.Send
set myMail=nothing
%>
```

发送一封发送来自网站的网页的 HTML 邮件：

```
<%
Set myMail=CreateObject("CDO.Message")
myMail.Subject="Sending email with CDO"
myMail.From="mymail@mydomain.com"
myMail.To="someone@somedomain.com"
myMail.CreateMHTMLBody "http://www.w3school.com.cn/asp/"
myMail.Send
set myMail=nothing
%>
```

发送一封发送来自电脑中文件的网页的 HTML 邮件：

```
<%
Set myMail=CreateObject("CDO.Message")
myMail.Subject="Sending email with CDO"
myMail.From="mymail@mydomain.com"
myMail.To="someone@somedomain.com"
myMail.CreateMHTMLBody "file://c:/mydocuments/test.htm"
myMail.Send
set myMail=nothing
%>
```

发送一封带有附件的电子邮件：

```
<%
Set myMail=CreateObject("CDO.Message")
myMail.Subject="Sending email with CDO"
myMail.From="mymail@mydomain.com"
myMail.To="someone@somedomain.com"
myMail.TextBody="This is a message."
myMail.AddAttachment "c:\mydocuments\test.txt"
myMail.Send
set myMail=nothing
%>
```

使用远程服务器发送一封文本邮件：

```
<%
Set myMail=CreateObject("CDO.Message")
myMail.Subject="Sending email with CDO"
myMail.From="mymail@mydomain.com"
myMail.To="someone@somedomain.com"
myMail.TextBody="This is a message."
myMail.Configuration.Fields.Item _
("http://schemas.microsoft.com/cdo/configuration/sendusing")=2
'远程 SMTP 服务器的 IP 或名称
myMail.Configuration.Fields.Item _
("http://schemas.microsoft.com/cdo/configuration/smtpserver") _
="smtp.server.com"
'服务器端口
myMail.Configuration.Fields.Item _
("http://schemas.microsoft.com/cdo/configuration/smtpserverport") _
=25
myMail.Configuration.Fields.Update
myMail.Send
set myMail=nothing
%>
```

ASP Response 对象

ASP Response 对象用于从服务器向用户发送输出的结果。

实例

使用 *ASP* 写文本

本例演示如何使用 ASP 来写文本。

```
<html>
<body>

<%
response.write("Hello World!")
%>

</body>
</html>
```

在 *ASP* 中使用 *HTML* 标签格式化文本

本例演示如何使用 ASP 将文本和 HTML 标签结合起来。

```
<html>
<body>
<%
response.write("<h2>您可以使用 HTML 标签来格式化文本</h2>")
%>

<%
response.write("<p style='color:#0000ff'>这段文本的样式是通过 style 属性添加的。</p>")
%>
</body>
</html>
```

将用户重定向至不同的 *URL*

本例演示如何将用户重定向至另一个的 URL。

```
<%
if Request.Form("select")<>"" then
       Response.Redirect(Request.Form("select"))
end if
%>

<html>
<body>

<form action="/example/aspe/demo_aspe_redirect.asp" method="post">

<input type="radio" name="select"
value="/example/aspe/demo_aspe_server.asp">
服务器实例<br />

<input type="radio" name="select"
value="/example/aspe/demo_aspe_text.asp">
文本实例<br /><br />
<input type="submit" value="跳转！">

</form>

</body>
</html>
```

显示随机的链接

本例演示一个超级链接，当您每次载入页面时，它将显示两个链接中的其中一个。

```

<html>
<body>

<%
randomize()
```

```
r=rnd()
if r>0.5 then
  response.write("<a href='http://www.w3school.com.cn'>W3School.com.cn!</a>")
else
  response.write("<a href='http://www.news.cn'>news.cn!</a>")
end if
%>

<p>
本例演示一个链接，每当您加载本页时，就会显示两个链接之一：W3School.com.cn! 或 news.cn! 各占百
</p>

</body>
</html>
```

控制缓存

本例演示如何控制缓存。

```
<%
Response.Buffer=true
%>
<html>
<body>
<p>
当您的 response 缓存清空时，这段文本就会发送到浏览器。
</p>
<%
Response.Flush
%>
</body>
</html>
```

清空缓存

本例演示如何清空缓存。

```
<%
Response.Buffer=true
%>
<html>
<body>
<p>这是我希望发送给用户的文本。</p>
<p>不，我改变主意了。我希望清除这些文本。</p>
<%
Response.Clear
%>
</body>
```

```
</html>
```

在处理过程中终止脚本并返回结果

本例演示如何在处理过程中中断脚本的运行。

```
<html>
<body>
<p>我正在写文本。这些文本不会被<br />
<%
Response.End
%>
完全发送。这时候已经不能输出任何文本了！</p>
</body>
</html>
```

设置在页面失效前把页面在浏览器中缓存多少分钟

本例演示如何规定页面在失效前在浏览器中的缓存时间。

```
<%Response.Expires=-1%>
<html>
<body>
<p>每当被访问，本页都会被刷新！</p>
</body>
</html>
```

设置页面缓存在浏览器中的失效日期或时间

本例演示如何规定页面在浏览器中的缓存时间日期或时间

```
<%
Response.ExpiresAbsolute=#May 05,2001 05:30:30#
%>
<html>
<body>
<p>本页面的缓存会在该日期失效：05, 2001 05:30:30！</p>
</body>
</html>
```

检查用户是否仍然与服务器相连

本例演示如何检查用户是否已与服务器断开。

```
<html>
<body>

<%
If Response.IsClientConnected=true then
```

```
Response.Write("用户仍然保持连接。")
else
Response.Write("用户未连接。")
end if
%>

</body>
</html>
```

设置内容类型

本例演示如何规定内容的类型。

```
<%
Response.ContentType="text/html"
%>
<html>
<body>

<p>This is some text.</p>

</body>
</html>
```

设置字符集

本例演示如何规定字符集的名称。

```
<%
Response.Charset="ISO8859-1"
%>
<html>
<body>

<p>This is some text</p>

</body>
</html>
```

Response 对象

ASP Response 对象用于从服务器向用户发送输出的结果。它的集合、属性和方法如下：

集合

集合	描述
Cookies	设置 cookie 的值。假如不存在，就创建 cookie ，然后设置指定的值。

属性

属性	描述
Buffer	规定是否缓存页面的输出。
CacheControl	设置代理服务器是否可以缓存由 ASP 产生的输出。
Charset	将字符集的名称追加到 Response 对象中的 content-type 报头。
ContentType	设置 Response 对象的 HTTP 内容类型。
Expires	设置页面在失效前的浏览器缓存时间（分钟）。
ExpiresAbsolute	设置浏览器上页面缓存失效的日期和时间。
IsClientConnected	指示客户端是否已从服务器断开。
Pics	向 response 报头的 PICS 标志追加值。
Status	规定由服务器返回的状态行的值。

方法

方法	描述
AddHeader	向 HTTP 响应添加新的 HTTP 报头和值。
AppendToLog	向服务器记录项目（server log entry）的末端添加字符串。
BinaryWrite	在没有任何字符转换的情况下直接向输出写数据。
Clear	清除已缓存的 HTML 输出。
End	停止处理脚本，并返回当前的结果。
Flush	立即发送已缓存的 HTML 输出。
Redirect	把用户重定向到另一个 URL。
Write	向输出写指定的字符串。

ASP Request 对象

ASP Request 对象用于从用户那里取得信息。

实例

QueryString 集合 实例

当用户点击链接时发送查询信息

本例演示如何在链接中向页面发送一些额外的查询信息，并在目标页面中取回这些信息（在本例中是同一页面）。

```
<html>
<body>

<a href="/example/aspe/demo_aspe_simplequerystring.asp?color=green">Example</a>

<%
Response.Write(Request.QueryString)
%>

</body>
</html>
```

对 *QueryString* 集合的简单应用

本例演示 *QueryString* 集合如何从表单取回值。此表单使用 GET 方法，这意外着所发送的信息对用户来说是可见的（在地址中）。GET 方法还会限制所发送信息的数量。

```
<html>
<body>

<form action="/example/aspe/demo_aspe_simplereqquery.asp" method="get">
First name: <input type="text" name="fname"><br />
Last name: <input type="text" name="lname"><br />
<input type="submit" value="Submit">
</form>

<%
Response.Write(Request.QueryString)
%>

</body>
</html>
```

如何使用从表单传来的信息

本例演示如何使用从表单取回的值。我们会使用 *QueryString* 集合。此表单使用 GET 方法。

```
<html>
<body>
<form action="/example/aspe/demo_aspe_reqquery.asp" method="get">
您的姓名：<input type="text" name="fname" size="20" />
<input type="submit" value="提交" />
</form>
<%
dim fname
fname=Request.QueryString("fname")
```

```
If fname<>"" Then
      Response.Write("你好! " & fname & "! <br />")
      Response.Write("今天过得怎么样？")
End If
%>
</body>
</html>
```

来自表单的更多信息

本例演示假如输入字段包含若干相同的名称的话，*QueryString* 会包含什么内容。它将展示如何把这些相同的名称分隔开来。它也会展示如何使用 *count* 关键词来对 "name" 属性进行计数。此表单使用 GET 方法。

```
<html>
<body>

<%
If Request.QueryString<>"" Then
      If Request.QueryString("name")<>", " Then
           name1=Request.QueryString("name")(1)
           name2=Request.QueryString("name")(2)
      end if
end if
%>

<form action="/example/aspe/demo_aspe_reqquery2.asp" method="get">
First name:
<input type="text" name="name" value="<%=name1%>" />
<br />
Last name:
<input type="text" name="name" value="<%=name2%>" />
<br />
<input type="submit" value="Submit" />
</form>
<hr>
<%
If Request.QueryString<>"" Then
      Response.Write("<p>")
      Response.Write("The information received from the form was:")
      Response.Write("</p><p>")
      Response.Write("name=" & Request.QueryString("name"))
      Response.Write("</p><p>")
      Response.Write("The name property's count is: ")
      Response.Write(Request.QueryString("name").Count)
      Response.Write("</p><p>")
      Response.Write("First name=" & name1)
      Response.Write("</p><p>")
      Response.Write("Last name=" & name2)
      Response.Write("</p>")
```

```
end if
%>
</body>
</html>
```

Form 集合 实例

一个 ***Form*** 集合的简单应用

本例演示 *Form* 集合如何从表单取回值。此表单使用 POST 方法，这意味着发送的信息对用户来说是不可见的，并且对所发送信息的量没有限制（可发送大量的信息）。

```
<html>
<body>

<form action="/example/aspe/demo_aspe_simpleform1.asp" method="post">
First name:
<input type="text" name="fname" value="Donald" />
<br />
Last name:
<input type="text" name="lname" value="Duck" />
<br />
<input type="submit" value="Submit" />
</form>

<%
Response.Write(Request.Form)
%>

</body>
</html>
```

如何使用来自表单的信息

本例演示如何使用从表单取回的信息。我们使用了 *Form* 集合。表单使用了 POST 方法。

```

<html>
<body>
<form action="/example/aspe/demo_aspe_simpleform.asp" method="post">
您的姓名：<input type="text" name="fname" size="20" />
<input type="submit" value="提交" />
</form>
<%
dim fname
fname=Request.Form("fname")
If fname<>"" Then
      Response.Write("您好！ " & fname & "！ <br />")
      Response.Write("今天过得怎么样？ ")
End If
%>
```

```
</body>
</html>
```

来自表单的更多信息

本例演示假如若干的输入域使用了相同的名称，*Form* 集合会包含什么信息。它将展示如何把这些相同的名称分割开来。它也会展示如何使用 *count* 关键词来对 "name" 属性进行计数。此表单使用 POST 方法。

```
<html>
<body>

<form action="/example/aspe/demo_aspe_form2.asp" method="post">
First name:
<input type="text" name="name" value="Donald" />
<br />
Last name:
<input type="text" name="name" value="Duck" />
<br />
<input type="submit" value="Submit" />
</form>
<hr />

<p>来自上面的表单的信息：</p>
<%
If Request.Form("name")<>"" Then
      Response.Write("<p>")
      Response.Write("name=" & Request.Form("name"))
      Response.Write("</p><p>")
      Response.Write("name 属性的数目：")
      Response.Write(Request.Form("name").Count)
      Response.Write("</p><p>")
      Response.Write("First name=" & Request.Form("name")(1))
      Response.Write("</p><p>")
      Response.Write("Last name=" & Request.Form("name")(2))
      Response.Write("</p>")
End if
%>

</body>
</html>
```

带有单选按钮的表单

本例演示如何使用 *Form* 集合通过单选按钮与用户进行交互。此表单使用 POST 方法。

```

<html>
<%
dim cars
cars=Request.Form("cars")
```

```
%>
<body>
<form action="/example/aspe/demo_aspe_radiob.asp" method="post">
<p>请选择您喜欢的汽车：</p>

<input type="radio" name="cars"
<%if cars="Volvo" then Response.Write("checked")%>
value="Volvo">Volvo</input>
<br />
<input type="radio" name="cars"
<%if cars="Saab" then Response.Write("checked")%>
value="Saab">Saab</input>
<br />
<input type="radio" name="cars"
<%if cars="BMW" then Response.Write("checked")%>
value="BMW">BMW</input>
<br /><br />
<input type="submit" value="提交" />
</form>
<%
if cars<>"" then
   Response.Write("<p>您喜欢的汽车是" & cars & "</p>")
end if
%>
</body>
</html>
```

带有复选按钮的表单

本例演示如何使用 *Form* 集合通过复选按钮与用户进行交互。此表单使用 POST 方法。

```
<html>
<body>
<%
fruits=Request.Form("fruits")
%>

<form action="/example/aspe/demo_aspe_checkboxes.asp" method="post">
<p>您喜欢哪些水果：</p>
<input type="checkbox" name="fruits" value="Apples"
<%if instr(fruits,"Apple") then Response.Write("checked")%>>
Apple
<br />
<input type="checkbox" name="fruits" value="Oranges"
<%if instr(fruits,"Oranges") then Response.Write("checked")%>>
Orange
<br />
<input type="checkbox" name="fruits" value="Bananas"
<%if instr(fruits,"Banana") then Response.Write("checked")%>>
Banana
```

```
<br />
<input type="submit" value="提交">
</form>
<%
if fruits<>"" then%>
   <p>您喜欢：<%Response.Write(fruits)%></p>
<%end if
%>

</body>
</html>
```

其他实例

获取用户信息

如何查明访问者的浏览器类型、IP 地址等信息。

```
<html>
<body>
<p>
<b>您正在通过这款浏览器访问我们的站点：</b>
<%Response.Write(Request.ServerVariables("http_user_agent"))%>
</p>
<p>
<b>您的 IP 地址是：</b>
<%Response.Write(Request.ServerVariables("remote_addr"))%>
</p>
<p>
<b>IP 地址的 DNS 查询是：</b>
<%Response.Write(Request.ServerVariables("remote_host"))%>
</p>
<p>
<b>调用该页面所用的方法是：</b>
<%Response.Write(Request.ServerVariables("request_method"))%>
</p>
<p>
<b>服务器的域名：</b>
<%Response.Write(Request.ServerVariables("server_name"))%>
</p>
<p>
<b>服务器的端口：</b>
<%Response.Write(Request.ServerVariables("server_port"))%>
</p>
<p>
<b>服务器的软件：</b>
<%Response.Write(Request.ServerVariables("server_software"))%>
</p>

</body>
```

```
</html>
```

获取服务器变量

本例演示如何使用 *ServerVariables* 集合取得访问者的浏览器类型、IP 地址等信息。

```
<html>
<body>

<p>
所有可能的服务器变量：
</p>
<%
For Each Item in Request.ServerVariables
      Response.Write(Item & "<br />")
Next
%>

</body>
</html>
```

创建 *welcome cookie*

本例演示如何使用 Cookies 集合创建一个欢迎 *cookie*。

```
<%
dim numvisits
response.cookies("NumVisits").Expires=date+365
numvisits=request.cookies("NumVisits")

if numvisits="" then
   response.cookies("NumVisits")=1
   response.write("欢迎！这是您第一次访问本页面。")
else
   response.cookies("NumVisits")=numvisits+1
   response.write("之前，您已经访问过本页面 ")
   response.write(numvisits & " 次。")
end if
%>
<html>
<body>
</body>
</html>
```

探测用户发送的字节总数

本例演示如何使用 *TotalBytes* 属性来取得用户在 Request 对象中发送的字节总数。

```
```

```
<html>
<body>

<form action="/example/aspe/demo_aspe_totalbytes.asp" method="post">
请键入一些字符：
<input type="text" name="txt"><br /><br />
<input type="submit" value="提交">
</form>

<%
If Request.Form("txt")<>"" Then
   Response.Write("您提交了：")
   Response.Write(Request.Form)
   Response.Write("<br /><br />")
   Response.Write("字节总计：")
   Response.Write(Request.Totalbytes)
End If
%>

</body>
</html>
```

Request 对象

当浏览器向服务器请求页面时，这个行为就被称为一个 request（请求）。

ASP Request 对象用于从用户那里获取信息。它的集合、属性和方法描述如下：

集合

集合	描述
ClientCertificate	包含了在客户证书中存储的字段值
Cookies	包含了 HTTP 请求中发送的所有 cookie 值
Form	包含了使用 post 方法由表单发送的所有的表单（输入）值
QueryString	包含了 HTTP 查询字符串中所有的变量值
ServerVariables	包含了所有的服务器变量值

属性

属性	描述
TotalBytes	返回在请求正文中客户端所发送的字节总数

方法

方法	描述
BinaryRead	取回作为 post 请求的一部分而从客户端送往服务器的数据，并把它存放到一个安全的数组之中。

ASP Application 对象

在一起协同工作以完成某项任务的一组 **ASP** 文件称为一个应用程序。而 **ASP** 中的 **Application** 对象的作用是把这些文件捆绑在一起。

Application 对象

Web 上的一个应用程序可以是一组 ASP 文件。这些 ASP 在一起协同工作来完成一项任务。而 ASP 中的 Application 对象的作用是把这些文件捆绑在一起。

Application 对象用于存储和访问来自任意页面的变量，类似 Session 对象。不同之处在于所有的用户分享一个 Application 对象，而 session 对象和用户的关系是一一对应的。

Application 对象掌握的信息会被应用程序中的很多页面使用（比如数据库连接信息）。这就意味我们可以从任意页面访问这些信息。也意味着你可以在在一个页面上改变这些信息，随后这些改变会自动地反映到所有的页面中。

Application 对象的集合、方法和事件的描述如下：

集合

集合	描述
Contents	包含所有通过脚本命令追加到应用程序中的项目。
StaticObjects	包含所有使用 HTML 的 <object> 标签追加到应用程序中的对象。

方法

方法	描述
Contents.Remove	从 Contents 集合中删除一个项目。
Contents.RemoveAll()	从 Contents 集合中删除所有的项目。
Lock	防止其余的用户修改 Application 对象中的变量。
Unlock	使其他的用户可以修改 Application 对象中的变量（在被 Lock 方法锁定之后）。

事件

事件	描述
Application_OnEnd	当所有用户的 session 都结束，并且应用程序结束时，此事件发生。
Application_OnStart	在首个新的 session 被创建之前（这时 Application 对象被首次引用），此事件会发生。

ASP Session 对象

Session 对象用于存储关于某个用户会话（**session**）的信息，或者修改相关的设置。存储在 **session** 对象中的变量掌握着单一用户的信息，同时这些信息对于页面中的所有页面都是可用的。

实例

设置并返回 *LCID*

本例演示 "LCID" 属性。此属性设置并返回一个指示位置或者地区的整数。类似于日期、时间以及货币等内容都要根据位置或者地区来显示。

```
<html>
<body>

<%
response.write("<p>")
response.write("本页的默认 LCID 是：" & Session.LCID & "<br />")
response.write("上面的 LCID 的日期格式是：" & date() & "<br />")
response.write("上面的 LCID 的货币格式是：" & FormatCurrency(350))
response.write("</p>")

Session.LCID=1036

response.write("<p>")
response.write("现在 LCID 变更为：" & Session.LCID & "<br />")
response.write("上面的 LCID 的日期格式是：" & date() & "<br />")
response.write("上面的 LCID 的货币格式是：" & FormatCurrency(350))
response.write("</p>")

Session.LCID = 3079

response.write("<p>")
response.write("现在 LCID 变更为：" & Session.LCID & "<br />")
response.write("上面的 LCID 的日期格式是：" & date() & "<br />")
response.write("上面的 LCID 的货币格式是：" & FormatCurrency(350))
response.write("</p>")

Session.LCID = 2057
```

```
response.write("<p>")
response.write("现在 LCID 变更为：" & Session.LCID & "<br />")
response.write("上面的 LCID 的日期格式是：" & date() & "<br />")
response.write("上面的 LCID 的货币格式是：" & FormatCurrency(350))
response.write("</p>")
%>

</body>
</html>
```

返回 *SessionID*

本例演示 "SessionID" 属性。该属性为每位用户返回一个唯一的 id。这个 id 由服务器生成。

```
<html>
<body>

<%
Response.Write(Session.SessionID)
%>

</body>
</html>
```

session 的超时

本例演示 "Timeout" 属性。这个例子设置并返回 session 的超时时间（分钟）。

```
<html>
<body>

<%
response.write("<p>")
response.write("默认 Timeout 是：" & Session.Timeout & " 分钟。")
response.write("</p>")

Session.Timeout=30

response.write("<p>")
response.write("现在的 Timeout 是 " & Session.Timeout & " 分钟。")
response.write("</p>")
%>

</body>
</html>
```

Session 对象

当您正在操作一个应用程序时，您会启动它，然后做些改变，随后关闭它。这个过程很像一次对话（Session）。计算机知道你是谁。它也知道你在何时启动和关闭这个应用程序。但是在因特网上，问

题出现了：web 服务器不知道你是谁，也不知道你做什么，这是由于 HTTP 地址无法留存状态（信息）。

ASP 通过为每个用户创一个唯一的 cookie 解决了这个问题。cookie 发送到服务器，它包含了可识别用户的信息。这个接口称作 Session 对象。

Session 对象用于存储关于某个用户会话（session）的信息，或者修改相关的设置。存储在 session 对象中的变量掌握着单一用户的信息，同时这些信息对于页面中的所有页面都是可用的。存储于 session 变量中的信息通常是 name、id 以及参数等。服务器会为每位新用户创建一个新的 Session 对象，并在 session 到期后撤销这个对象。

下面是 Session 对象的集合、属性、方法以及事件：

集合

集合	描述
Contents	包含所有通过脚本命令追加到 session 的条目。
StaticObjects	包含了所有使用 HTML 的 <object> 标签追加到 session 的对象。

属性

属性	描述
CodePage	规定显示动态内容时使用的字符集
LCID	设置或返回指定位置或者地区的一个整数。诸如日期、时间以及货币的内容会根据位置或者地区来显示。
SessionID	为每个用户返回一个唯一的 id。此 id 由服务器生成。
Timeout	设置或返回应用程序中的 session 对象的超时时间（分钟）。

方法

方法	描述
Abandon	撤销一个用户的 session。
Contents.Remove	从 Contents 集合删除一个项目。
Contents.RemoveAll()	从 Contents 集合删除全部项目。

事件

事件	描述

Session_OnEnd	当一个会话结束时此事件发生。
Session_OnStart	当一个会话开始时此事件发生。

ASP Server 对象

ASP Server 对象的作用是访问有关服务器的属性和方法。

实例

此文件最后被修改的时间是？

探测文件的最后更新时间。

```
<html>
<body>

<%
Set fs = Server.CreateObject("Scripting.FileSystemObject")
Set rs = fs.GetFile(Server.MapPath("/example/aspe/demo_aspe_lastmodified.asp"))
modified = rs.DateLastModified
%>
本文件的最后修改时间是：<%response.write(modified)
Set rs = Nothing
Set fs = Nothing
%>

</body>
</html>
```

打开并读取某个文本文件

本例会打开文件 "Textfile.txt" 以供读取。

```
<html>
<body>

<%
Set FS = Server.CreateObject("Scripting.FileSystemObject")
Set RS = FS.OpenTextFile(Server.MapPath("/example/aspe") & "\textfile.txt",1)
While not rs.AtEndOfStream
      Response.Write RS.ReadLine
      Response.Write("<br />")
Wend
%>

<p>
<a href="/example/aspe/textfile.txt">查看此文本文件</a>
</p>
```

```
</body>
</html>
```

自制的点击计数器

本例可从一个文件中读取一个数字，并在此数字上累加 1，然后将此数写回这个文件。

```
<%
Set FS=Server.CreateObject("Scripting.FileSystemObject")
Set RS=FS.OpenTextFile(Server.MapPath("/example/aspe/counter.txt"), 1, False)
fcount=RS.ReadLine
RS.Close

fcount=fcount+1

'This code is disabled due to the write access security on our server:
'Set RS=FS.OpenTextFile(Server.MapPath("counter.txt"), 2, False)
'RS.Write fcount
'RS.Close

Set RS=Nothing
Set FS=Nothing

%>
<html>
<body>
<p>
本页已被访问了 <%=fcount%>  次。
</p>
</body>
</html>
```

Server 对象

ASP Server 对象的作用是访问有关服务器的属性和方法。其属性和方法描述如下：

属性

属性	描述
ScriptTimeout	设置或返回在一段脚本终止前它所能运行时间（秒）的最大值。

方法

方法	描述
CreateObject	创建对象的实例（instance）。

Execute	从另一个 ASP 文件中执行一个 ASP 文件。
GetLastError()	返回可描述已发生错误状态的 ASPError 对象。
HTMLEncode	将 HTML 编码应用到某个指定的字符串。
MapPath	将一个指定的地址映射到一个物理地址。
Transfer	把一个 ASP 文件中创建的所有信息传输到另一个 ASP 文件。
URLEncode	把 URL 编码规则应用到指定的字符串。

ASP ASPError 对象

ASPError 对象用于显示在 **ASP** 文件的脚本中发生的任何错误的详细信息。

ASP ASPError 对象

ASP 3.0 提供这个对象，且在 IIS5 及更高版本中可用。

ASPError 对象用于显示在 ASP 文件的脚本中发生的任何错误的详细信息。当 Server.GetLastError 被调用时，ASPError 对象就会被创建，因此只能通过使用 Server.GetLastError 方法来访问错误信息。

ASPError 对象的属性描述如下（所有属性都是可读的）：

注释：下面的属性只能 Server.GetLastError() 方法来访问。

属性

属性	描述
ASPCode	返回由 IIS 生成的错误代码。
ASPDescription	返回有关错误的详细信息。（假如错误和 ASP 相关。）
Category	返回错误来源。(是由 ASP、脚本语言还是对象引起的？)
Column	返回在出错文件中的列位置。
Description	返回关于错误的简短描述。
File	返回出错 ASP 文件的文件名。
Line	返回错误所在的行数。
Number	返回关于错误的标准 COM 错误代码。
Source	返回错误所在行的实际的源代码

ASP FileSystemObject 对象

FileSystemObject 对象用于访问服务器上的文件系统。

实例

指定的文件存在吗？

本例演示如何首先创建 FileSystemObject 对象，然后使用 FileExists 方法来探测某文件是否存在。

```
<html>
<body>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

If (fs.FileExists("c:\windows\cursors\xxx.cur"))=true Then

Response.Write("文件 c:\windows\cursors\xxx.cur 存在。")

Else
      Response.Write("文件 c:\windows\cursors\xxx.cur 不存在。")
End If

set fs=nothing
%>

</body>
</html>
```

指定的文件夹存在吗？

本例演示如何使用 FolderExists 方法探测某文件夹是否存在。

```

<html>
<body>
<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

If fs.FolderExists("c:\temp") = true Then
      Response.Write("文件夹 c:\temp 存在。")
Else
      Response.Write("文件夹 c:\temp 不存在。")
End If

set fs=nothing
%>

</body>
</html>
```

指定的驱动器存在吗？

本例演示如何使用 DriveExists 方法来探测某个驱动器是否存在。

```
<html>
<body>
<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

if fs.driveexists("c:") = true then
      Response.Write("驱动器 c: 存在。")
Else
      Response.Write("驱动器 c: 不存在。")
End If

Response.write("<br />")

if fs.driveexists("g:") = true then
      Response.Write("驱动器 g: 存在。")
Else
      Response.Write("驱动器 g: 不存在。")
End If

set fs=nothing
%>

</body>
</html>
```

取得某个指定驱动器的名称

本例演示如何使用 GetDriveName 方法来取得某个指定的驱动器的名称。

```
<html>
<body>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")
p=fs.GetDriveName("c:\windows\cursors\abc.cur")

Response.Write("驱动器名称是: " & p)

set fs=nothing
%>

</body>
</html>
```

取得某个指定路径的父文件夹的名称

本例演示如何使用 GetParentFolderName 方法来取得某个指定的路径的父文件夹的名称。

```
<html>
<body>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")
p=fs.GetParentFolderName("c:\winnt\cursors\3dgarro.cur")

Response.Write("c:\windows\cursors\abc.cur 的父文件夹名称是：" & p)

set fs=nothing
%>

</body>
</html>
```

取得文件夹扩展名

本例演示如何使用 GetExtensionName 方法来取得指定的路径中的最后一个成分的文件扩展名。

```
<html>
<body>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")
p=fs.GetParentFolderName("c:\winnt\cursors\3dgarro.cur")

Response.Write("c:\windows\cursors\abc.cur 的父文件夹名称是：" & p)

set fs=nothing
%>

</body>
</html>
```

取得文件名

本例演示如何使用 GetFileName 方法来取得指定的路径中的最后一个成分的文件名。

```
<html>
<body>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

Response.Write("这个文件名的最后一个成分是：")
Response.Write(fs.GetFileName("c:\windows\cursors\abc.cur"))
set fs=nothing
%>
```

```
</body>
</html>
```

取得文件或文件夹的基名称

本例演示如何使用 GetBaseName 方法来返回在指定的路径中文件或者文件夹的基名称。

```
<html>
<body>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

Response.Write(fs.GetBaseName("c:\windows\cursors\abc.cur"))
Response.Write("<br />")
Response.Write(fs.GetBaseName("c:\windows\cursors\"))
Response.Write("<br />")
Response.Write(fs.GetBaseName("c:\windows\"))

set fs=nothing
%>

</body>
</html>
```

FileSystemObject 对象

FileSystemObject 对象用于访问服务器上的文件系统。此对象可对文件、文件夹以及目录路径进行操作。也可通过此对象获取文件系统的信息。

下面的代码会创建一个文本文件 (c:\test.txt)，然后向这个文件写一些文本：

```
<%
dim fs,fname
      set fs=Server.CreateObject("Scripting.FileSystemObject")
      set fname=fs.CreateTextFile("c:\test.txt",true)
      fname.WriteLine("Hello World!")
      fname.Close
set fname=nothing
set fs=nothing
%>
```

FileSystemObject 对象的属性和方法描述如下：

属性

属性	描述
Drives	返回本地计算机上所有驱动器对象的集合。

方法

方法	描述
BuildPath	将一个名称追加到已有的路径后
CopyFile	从一个位置向另一个位置拷贝一个或多个文件。
CopyFolder	从一个位置向另一个位置拷贝一个或多个文件夹。
CreateFolder	创建新文件夹。
CreateTextFile	创建文本文件，并返回一个 TextStream 对象。
DeleteFile	删除一个或者多个指定的文件。
DeleteFolder	删除一个或者多个指定的文件夹。
DriveExists	检查指定的驱动器是否存在。
FileExists	检查指定的文件是否存在。
FolderExists	检查某个文件夹是否存在。
GetAbsolutePathName	针对指定的路径返回从驱动器根部起始的完整路径。
GetBaseName	返回指定文件或者文件夹的基名称。
GetDrive	返回指定路径中所对应的驱动器的 Drive 对象。
GetDriveName	返回指定的路径的驱动器名称。
GetExtensionName	返回在指定的路径中最后一个成分的文件扩展名。
GetFile	返回一个针对指定路径的 File 对象。
GetFileName	返回在指定的路径中最后一个成分的文件名。
GetFolder	返回一个针对指定路径的 Folder 对象。
GetParentFolderName	返回在指定的路径中最后一个成分的父文件名称。
GetSpecialFolder	返回某些 Windows 的特殊文件夹的路径。
GetTempName	返回一个随机生成的文件或文件夹。
MoveFile	从一个位置向另一个位置移动一个或多个文件。
MoveFolder	从一个位置向另一个位置移动一个或多个文件夹。
OpenTextFile	打开文件，并返回一个用于访问此文件的 TextStream 对象。

ASP TextStream 对象

TextStream 对象用于访问文本文件的内容。

实例

读文件

本例演示如何使用 FileSystemObject 的 OpenTextFile 方法来创建一个 TextStream 对象。TextStream 对象的 ReadAll 方法会从已打开的文本文件中取得内容。

```
<html>
<body>
<p>这就是文本文件中的文本：</p>
<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

Set f=fs.OpenTextFile(Server.MapPath("/example/aspe/testread.txt"), 1)
Response.Write(f.ReadAll)
f.Close

Set f=Nothing
Set fs=Nothing
%>
</body>
</html>
```

读文本文件中的一个部分

本例演示如何仅仅读取一个文本流文件的部分内容。

```
<html>
<body>
<p>这是从文本文件中读取的前 5 个字符：</p>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

Set f=fs.OpenTextFile(Server.MapPath("testread.txt"), 1)
Response.Write(f.Read(5))
f.Close

Set f=Nothing
Set fs=Nothing
%>

</body>
</html>
```

读文本文件中的一行

本例演示如何从一个文本流文件中读取一行内容。

```
<html>
<body>
<p>这是从文本文件中读取的第一行：</p>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

Set f=fs.OpenTextFile(Server.MapPath("testread.txt"), 1)
Response.Write(f.ReadLine)
f.Close

Set f=Nothing
Set fs=Nothing
%>

</body>
</html>
```

读取文本文件的所有行

本例演示如何从文本流文件中读取所有的行。

```
<html>
<body>
<p>这是从文本文件中读取的所有行：</p>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")
Set f=fs.OpenTextFile(Server.MapPath("testread.txt"), 1)

do while f.AtEndOfStream = false
Response.Write(f.ReadLine)
Response.Write("<br />")
loop

f.Close
Set f=Nothing
Set fs=Nothing
%>

</body>
</html>
```

略过文本文件的一部分

本例演示如何在读取文本流文件时跳过指定的字符数。

```

<html>
<body>
<p>文本文件中的前 4 个字符被略掉了：</p>
```

```
<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

Set f=fs.OpenTextFile(Server.MapPath("testread.txt"), 1)
f.Skip(4)
Response.Write(f.ReadAll)
f.Close

Set f=Nothing
Set fs=Nothing
%>

</body>
</html>
```

略过文本文件的一行

本例演示如何在读取文本流文件时跳过一行。

```
<html>
<body>
<p>文本文件中的第一行被略掉了：</p>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

Set f=fs.OpenTextFile(Server.MapPath("testread.txt"), 1)
f.SkipLine
Response.Write(f.ReadAll)
f.Close

Set f=Nothing
Set fs=Nothing
%>

</body>
</html>
```

返回行数

本例演示如何返回在文本流文件中的当前行号。

```
<html>
<body>
<p>这是文本文件中的所有行（带有行号）：</p>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")
```

```
Set f=fs.OpenTextFile(Server.MapPath("testread.txt"), 1)

do while f.AtEndOfStream = false
Response.Write("Line:" & f.Line & " ")
Response.Write(f.ReadLine)
Response.Write("<br />")
loop

f.Close
Set f=Nothing
Set fs=Nothing
%>

</body>
</html>
```

取得列数

本例演示如何取得在文件中当前字符的列号。

```
<html>
<body>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

Set f=fs.OpenTextFile(Server.MapPath("testread.txt"), 1)
Response.Write(f.Read(2))
Response.Write("<p>指针目前位于文本文件中的位置 " & f.Column & "。</p>")
f.Close

Set f=Nothing
Set fs=Nothing
%>

</body>
</html>
```

TextStream 对象

TextStream 对象用于访问文本文件的内容。

下面的代码会创建一个文本文件 (c:\test.txt)，然后向此文件写一些文本（变量 f 是 TextStream 对象的一个实例）：

```
<%
dim fs, f
set fs=Server.CreateObject("Scripting.FileSystemObject")
set f=fs.CreateTextFile("c:\test.txt",true)
```

```
f.WriteLine("Hello World!")
f.Close
set f=nothing
set fs=nothing
%>
```

如需创建TextStream对象的一个实例，我们可以使用 FileSystemObject 对象的 CreateTextFile 方法或者 OpenTextFile 方法，也可以使用 File 对象的 OpenAsTextStream 方法。

TextStream 对象的属性和方法描述如下：

属性

属性	描述
AtEndOfLine	在 TextStream 文件中，如果文件指针正好位于行尾标记的前面，那么该属性值返回 True；否则返回 False。
AtEndOfStream	如果文件指针在 TextStream 文件末尾，则该属性值返回 True；否则返回 False。
Column	返回 TextStream 文件中当前字符位置的列号。
Line	返回 TextStream 文件中的当前行号。

方法

方法	描述
Close	关闭一个打开的 TextStream 文件。
Read	从一个 TextStream 文件中读取指定数量的字符并返回结果（得到的字符串）。
ReadAll	读取整个 TextStream 文件并返回结果。
ReadLine	从一个 TextStream 文件读取一整行（到换行符但不包括换行符）并返回结果。
Skip	当读一个 TextStream 文件时跳过指定数量的字符。
SkipLine	当读一个 TextStream 文件时跳过下一行。
Write	写一段指定的文本（字符串）到一个 TextStream 文件。
WriteLine	写入一段指定的文本（字符串）和换行符到一个 TextStream 文件中。
WriteBlankLines	写入指定数量的换行符到一个 TextStream 文件中。

ASP Drive 对象

Drive 对象用于返回关于本地磁盘驱动器或者网络共享驱动器的信息。

实例

取得指定驱动器的可用空间数

本例演示如何首先创建一个 FileSystemObject 对象，然后使用 AvailableSpace 属性来获得指定驱动器的可用空间。

```
<html>
<body>

<%
Set fs=Server.CreateObject("Scripting.FileSystemObject")

Set f=fs.OpenTextFile(Server.MapPath("testread.txt"), 1)
Response.Write(f.Read(2))
Response.Write("<p>指针目前位于文本文件中的位置 " & f.Column & "。</p>")
f.Close

Set f=Nothing
Set fs=Nothing
%>

</body>
</html>
```

取得指定驱动器的剩余空间容量

本例演示如何使用 FreeSpace 空间属性来取得指定驱动器的剩余空间。

```
<html>
<body>

<%
Dim fs, d, n
Set fs=Server.CreateObject("Scripting.FileSystemObject")
Set d=fs.GetDrive("c:")
n = "驱动器：" & d
n = n & "<br />以字节计的剩余空间：" & d.FreeSpace
Response.Write(n)
set d=nothing
set fs=nothing
%>

</body>
</html>
```

取得指定驱动器的总容量

本例演示如何使用 TotalSize 属性来获得指定驱动器的总容量。

```
<html>
<body>

<%
Dim fs,d,n
Set fs=Server.CreateObject("Scripting.FileSystemObject")
Set d=fs.GetDrive("c:")
n = "驱动器：" & d
n = n & "<br />以字节计的总容量：" & d.TotalSize
Response.Write(n)
set d=nothing
set fs=nothing
%>

</body>
</html>
```

取得指定驱动器的驱动器字母

本例演示如何使用 DriveLetter 属性来获得指定驱动器的驱动器字母。

```
<html>
<body>

<%
dim fs, d, n
set fs=Server.CreateObject("Scripting.FileSystemObject")
set d=fs.GetDrive("c:")
Response.Write("驱动器字母是：" & d.driveletter)
set d=nothing
set fs=nothing
%>

</body>
</html>
```

取得指定驱动器的驱动器类型

本例演示如何使用 DriveType 属性来获得指定驱动器的驱动器类型。

```
<html>
<body>

<%
dim fs, d, n
set fs=Server.CreateObject("Scripting.FileSystemObject")
set d=fs.GetDrive("c:")
Response.Write("驱动器类型是：" & d.DriveType)
set d=nothing
set fs=nothing
```

```
%>

</body>
</html>
```

取得指定驱动器的文件系统信息

本例演示如何使用 FileSystem 来取得指定驱动器的文件系统类型。

```
<html>
<body>

<%
dim fs, d, n
set fs=Server.CreateObject("Scripting.FileSystemObject")
set d=fs.GetDrive("c:")
Response.Write("文件系统是: " & d.FileSystem)
set d=nothing
set fs=nothing
%>

</body>
</html>
```

驱动器是否已就绪？

本例演示如何使用 IsReady 属性来检查指定的驱动器是否已就绪。

```
<html>
<body>

<%
dim fs,d,n
set fs=Server.CreateObject("Scripting.FileSystemObject")
set d=fs.GetDrive("c:")
n = "此 " & d.DriveLetter
if d.IsReady=true then
    n = n & " 驱动器已就绪。"
else
    n = n & " 驱动器未就绪。"
end if
Response.Write(n)
set d=nothing
set fs=nothing
%>

</body>
</html>
```

取得指定驱动器的路径

本例演示如何使用 Path 属性来取得指定驱动器的路径。

```
<html>
<body>

<%
dim fs,d
set fs=Server.CreateObject("Scripting.FileSystemObject")
set d=fs.GetDrive("c:")
Response.Write("路径是：" & d.Path)
set d=nothing
set fs=nothing
%>

</body>
</html>
```

取得指定驱动器的根文件夹

本例演示如何使用 RootFolder 属性来取得指定驱动器的根文件夹。

```
<html>
<body>

<%
dim fs,d
set fs=Server.CreateObject("Scripting.FileSystemObject")
set d=fs.GetDrive("c:")
Response.Write("根文件是：" & d.RootFolder)
set d=nothing
set fs=nothing
%>

</body>
</html>
```

取得指定驱动器的序列号

本例演示如何使用 Serialnumber 属性来取得指定驱动器的序列号。

```
<html>
<body>

<%
dim fs,d
set fs=Server.CreateObject("Scripting.FileSystemObject")
set d=fs.GetDrive("c:")
Response.Write("序列号：" & d.SerialNumber)
set d=nothing
set fs=nothing
%>
```

```
</body>
</html>
```

Drive 对象

Drive 对象用于返回关于本地磁盘驱动器或者网络共享驱动器的信息。Drive 对象可以返回有关驱动器的文件系统、剩余容量、序列号、卷标名等信息。

注释：无法通过 Drive 对象返回有关驱动器内容的信息。要达到这个目的，请使用 Folder 对象。

如需操作 Drive 对象的相关属性，我们需要创建通过 FileSystemObject 对象来创建 Drive 对象的实例。首先，创建一个 FileSystemObject 对象，然后通过 FileSystemObject 对象的 GetDrive 方法或者 Drives 属性来例示 Drive 对象。

下面的例子使用 FileSystemObject 对象的 GetDrive 方法来例示 Drive 对象，并使用 TotalSize 属性来返回指定驱动器 (c:) 的容量总数（字节）：

```
<%
Dim fs,d
Set fs=Server.CreateObject("Scripting.FileSystemObject")
Set d=fs.GetDrive("c:")
Response.Write("Drive " & d & ":")
Response.Write("Total size in bytes: " & d.TotalSize)
set d=nothing
set fs=nothing
%>
```

输出：

Drive c: Total size in bytes: 5893563398

Drive 对象的属性

属性	描述
AvailableSpace	向用户返回在指定的驱动器或网络共享驱动器上的可用空间容量。
DriveLetter	返回识别本地驱动器或网络共享驱动器的大写字母。
DriveType	返回指定驱动器的类型。
FileSystem	返回指定驱动器所使用的文件系统类型。
FreeSpace	向用户返回在指定的驱动器或网络共享驱动器上的剩余空间容量。
IsReady	如果指定驱动器已就绪，则返回 true。否则返回 false。
Path	返回其后有一个冒号的大写字母，用来指示指定驱动器的路径名。

RootFolder	返回一个文件夹对象，该文件夹代表指定驱动器的根文件夹。
SerialNumber	返回指定驱动器的序列号。
ShareName	返回指定驱动器的网络共享名。
TotalSize	返回指定的驱动器或网络共享驱动器的总容量
VolumeName	设置或者返回指定驱动器的卷标名

ASP File 对象

File 对象用于返回关于指定文件的信息。

实例

文件何时被创建？

本例演示如何首先创建 FileSystemObject 对象，然后使用 File 对象的 DateCreated 属性来取得指定文件被创建的日期和时间。

```
<html>
<body>

<%
dim fs, f
set fs=Server.CreateObject("Scripting.FileSystemObject")
set f=fs.GetFile(Server.MapPath("testread.txt"))
Response.Write("文件 testread.txt 的创建时间是：" & f.DateCreated)
set f=nothing
set fs=nothing
%>

</body>
</html>
```

此文件何时被修改？

本例演示如何使用 DateLastModified 属性来取得指定文件被修改的日期和时间。

```
<html>
<body>

<%
dim fs, f
set fs=Server.CreateObject("Scripting.FileSystemObject")
set f=fs.GetFile(Server.MapPath("testread.txt"))
Response.Write("文件 testread.txt 的最后修改时间是：" & f.DateLastModified)
set f=nothing
set fs=nothing
```

```
%>

</body>
</html>
```

此文件何时被访问过？

此例演示如何使用 DateLastAccessed 属性来取得指定文件最后被访问的日期和时间。

```
<html>
<body>

<%
dim fs, f
set fs=Server.CreateObject("Scripting.FileSystemObject")
set f=fs.GetFile(Server.MapPath("testread.txt"))
Response.Write("文件 testread.txt 的最后访问时间是：" & f.DateLastAccessed)
set f=nothing
set fs=nothing
%>

</body>
</html>
```

返回指定文件的属性

本例演示如何使用 Attributes 来返回指定文件的属性。

```

<html>
<body>

<%
dim fs,f
set fs=Server.CreateObject("Scripting.FileSystemObject")
set f=fs.GetFile(Server.MapPath("testread.txt"))
Response.Write("文件 testread.txt 的属性是：" & f.Attributes)
set f=nothing
set fs=nothing
%>

</body>
</html>
```

File 对象

File 对象用于返回有关指定文件的信息。

如需操作 File 对象的相关属性和方法，我们需要通过 FileSystemObject 来创建 File 对象的实例。首先，创建一个 FileSystemObject 对象，然后通过 FileSystemObject 对象的 GetFile 方法，或者通过

Folder 对象的 Files 属性来例示此 File 对象。

下面的代码使用 FileSystemObject 对象的 GetFile 方法来例示这个 File 对象，并使用 DateCreated 属性来返回指定文件被创建的日期：

```
<%
Dim fs,f
Set fs=Server.CreateObject("Scripting.FileSystemObject")
Set f=fs.GetFile("c:\test.txt")
Response.Write("File created: " & f.DateCreated)
set f=nothing
set fs=nothing
%>
```

输出：

```
File created: 8/8/2008 10:01:19 AM
```

File 对象的属性和方法

属性

属性	描述
Attributes	设置或返回指定文件的属性。
DateCreated	返回指定文件创建的日期和时间。
DateLastAccessed	返回指定文件最后被访问的日期和时间。
DateLastModified	返回指定文件最后被修改的日期和时间。
Drive	返回指定文件或文件夹所在的驱动器的驱动器字母。
Name	设置或返回指定文件的名称。
ParentFolder	返回指定文件或文件夹的父文件夹对象。
Path	返回指定文件的路径。
ShortName	返回指定文件的短名称（8.3 命名约定）。
ShortPath	返回指定文件的短路径（8.3 命名约定）。
Size	返回指定文件的尺寸（字节）。
Type	返回指定文件的类型。

方法

方法	描述
Copy	把指定文件从一个位置拷贝到另一个位置。
Delete	删除指定文件。
Move	把指定文件从一个位置移动到另一个位置。
OpenAsTextStream	打开指定文件，并返回一个 TextStream 对象以便访问此文件。

ASP Folder 对象

Folder 对象用来返回有关指定文件夹的信息。

Folder 对象

Folder 对象用于返回有关指定文件夹的信息。

如需操作 Folder 对象，我们需要通过 FileSystemObject 对象来创建 Folder 对象的实例。首先，创建一个 FileSystemObject 对象，然后通过 FileSystemObject 对象的 GetFolder 方法来例示这个 Folder 对象。

下面的代码使用 FileSystemObject 对象的 GetFolder 方法来例示这个 Folder 对象，并使用 DateCreated 属性来返回指定文件的创建日期：

```
<%
Dim fs,fo
Set fs=Server.CreateObject("Scripting.FileSystemObject")
Set fo=fs.GetFolder("c:\test")
Response.Write("Folder created: " & fo.DateCreated)
set fo=nothing
set fs=nothing
%>
```

输出：

```
Folder created: 10/22/2001 10:01:19 AM
```

Folder 对象的集合、属性以及方法

集合

集合	描述
Files	返回指定文件夹中所有文件夹的集合。
SubFolders	返回指定文件夹中所有子文件夹的集合。

属性

属性	描述
Attributes	设置或返回指定文件夹的属性。
DateCreated	返回指定文件夹被创建的日期和时间。
DateLastAccessed	返回指定文件夹最后被访问的日期和时间。
DateLastModified	返回指定文件夹最后被修改的日期和时间。
Drive	返回指定文件夹所在的驱动器的驱动器字母。
IsRootFolder	假如文件夹是根文件夹，则返回 ture，否则返回 false。
Name	设置或返回指定文件夹的名称。
ParentFolder	返回指定文件夹的父文件夹。
Path	返回指定文件的路径。
ShortName	返回指定文件夹的短名称。（8.3 命名约定）
ShortPath	返回指定文件夹的短路径。（8.3 命名约定）
Size	返回指定文件夹的大小。
Type	返回指定文件夹的类型。

方法

方法	描述
Copy	把指定的文件夹从一个位置拷贝到另一个位置。
Delete	删除指定文件夹。
Move	把指定的文件夹从一个位置移动到另一个位置。
CreateTextFile	在指定的文件夹创建一个新的文本文件，并返回一个 TextStream 对象以访问这个文件。

ASP Dictionary 对象

Dictionary 对象用于在结对的名称/值中存储信息（等同于键和项目）。

实例

指定的键存在吗？

本例演示如何受首先创建一个 Dictionary 对象，然后使用 Exists 方法来检查指定的键是否存在。

```
<html>
<body>

<%
dim d
set d=Server.CreateObject("Scripting.Dictionary")
d.Add "c", "China"
d.Add "i", "Italy"
if d.Exists("c")= true then
    Response.Write("键存在。")
else
    Response.Write("键不存在。")
end if
set d=nothing
%>

</body>
</html>
```

返回一个所有项目的数组

本例演示如何使用 Items 方法来返回所有项目的一个数组。

```
<html>
<body>

<%
dim d,a,i,s
set d=Server.CreateObject("Scripting.Dictionary")
d.Add "c", "China"
d.Add "i", "Italy"

Response.Write("<p>项目的值是：</p>")
a=d.Items
for i = 0 To d.Count -1
    s = s & a(i) & "<br />"
next
Response.Write(s)

set d=nothing
%>

</body>
</html>
```

返回一个所有键的数组

本例演示如何使用 Keys 方法来返回所有键的一个数组。

```
<html>
<body>
```

```
<%
dim d,a,i,s
set d=Server.CreateObject("Scripting.Dictionary")
d.Add "c", "China"
d.Add "i", "Italy"
Response.Write("<p>键的值是：</p>")
a=d.Keys
for i = 0 To d.Count -1
    s = s & a(i) & "<br />"
next
Response.Write(s)
set d=nothing
%>

</body>
</html>
```

返回某个项目的值

本例演示如何使用 Item 属性来返回一个项目的值。

```
<html>
<body>

<%
dim d
set d=Server.CreateObject("Scripting.Dictionary")
d.Add "c", "China"
d.Add "i", "Italy"
Response.Write("项目 i 的值是：" & d.item("i"))
set d=nothing
%>

</body>
</html>
```

设置一个键

本例演示如何使用 Key 属性来在 Dictionary 对象中设置一个键。

```
<html>
<body>

<%
dim d
set d=Server.CreateObject("Scripting.Dictionary")
d.Add "c", "China"
d.Add "i", "Italy"
d.Key("i") = "it"
Response.Write("键 i 已设置为 it，其值是：" & d.Item("it"))
```

```
set d=nothing
%>

</body>
</html>
```

返回键/项目对的数目

本例演示如何使用 Count 属性来返回键/项目对的数目。

```
<html>
<body>

<%
dim d, a, s, i
set d=Server.CreateObject("Scripting.Dictionary")
d.Add "c", "China"
d.Add "i", "Italy"
Response.Write("key/item 对的数目是: " & d.Count)
set d=nothing
%>

</body>
</html>
```

Dictionary 对象

Dictionary 对象用于在结对的名称/值中存储信息（(等同于键和项目）。Dictionary 对象看似比数组更为简单，然而，Dictionary 对象却是更令人满意的处理关联数据的解决方案。

比较 **Dictionary** 和数组：

- 键用于识别 Dictionary 对象中的项目
- 无需调用 ReDim 来改变 Dictionary 对象的尺寸
- 当从 Dictionary 删除一个项目时，其余的项目会自动上移
- Dictionary 不是多维，而数组是
- Dictionary 与数组相比，有更多的内建对象
- Dictionary 在频繁地访问随机元素时，比数组工作得更好
- Dictionary 在根据它们的内容定位项目时，比数组工作得更好

下面的例子创建了一个 Dictionary 对象，并向对象添加了一些键/项目对，然后取回了键 bl 的值：

```
<%
Dim d
Set d=Server.CreateObject("Scripting.Dictionary")
d.Add "re","Red"
d.Add "gr","Green"
d.Add "bl","Blue"
d.Add "pi","Pink"
```

```
Response.Write("The value of key bl is: " & d.Item("bl"))
%>
```

输出：

```
The value of key bl is: Blue
```

Dictionary 对象的属性和方法描述如下：

属性

属性	描述
CompareMode	设置或返回用于在 Dictionary 对象中比较键的比较模式。
Count	返回 Dictionary 对象中键/项目对的数目。
Item	设置或返回 Dictionary 对象中一个项目的值。
Key	为 Dictionary 对象中已有的键值设置新的键值。

方法

方法	描述
Add	向 Dictionary 对象添加新的键/项目对。
Exists	返回一个逻辑值，这个值可指示指定的键是否存在于 Dictionary 对象中。
Items	返回 Dictionary 对象中所有项目的一个数组。
Keys	返回 Dictionary 对象中所有键的一个数组。
Remove	从 Dictionary 对象中删除指定的键/项目对。
RemoveAll	删除 Dictionary 对象中所有的键/项目对。

ADO 简介

ADO 用于从网页访问数据库。

从 ASP 页面访问数据库

从 ASP 文件内部访问数据库的通常途径是：

1. 创建至数据库的 ADO 连接（ADO connection）
2. 打开数据库连接
3. 创建 ADO 记录集（ADO recordset）

4. 打开记录集（recordset）
5. 从数据集中提取你所需要的数据
6. 关闭数据集
7. 关闭连接

什么是 ADO?

- ADO 是一项微软公司的技术
- ADO 指 ActiveX Data Objects
- ADO 是一个微软的 Active-X 组件
- ADO 会随着微软 IIS 自动安装
- ADO 是用以访问数据库中数据的编程接口

下一步学习什么内容？

假如您希望学习更多关于 ADO 的知识，请阅读我们的 ADO 教程。

ASP AdRotator 组件

实例

简单的 *AdRotator* 实例

本例展示：每当用户访问网站或者刷新一次页面，如何使用 AdRotator 组件来显示一幅不同的广告图像。

```
<html>
<body>

<%
set adrotator=Server.CreateObject("MSWC.AdRotator")
adrotator.Border="2"
Response.Write(adrotator.GetAdvertisement("/example/aspe/advertisements.txt"))
%>

<p>
<b>注释：</b>由于图像是随机变化的，
同时由于本页可供选择的图片很少，
因此经常会出现两次显示同一广告的情况。
</p>

<p>
<a href="/example/aspe/advertisements.txt">
查看 advertisements.txt
</a>
</p>

</body>
```

```
</html>
```

AdRotator - 图片链接

本例展示：每当用户访问网站或者刷新一次页面，如何使用 AdRotator 组件来显示一幅不同的广告图像。此外，图片本身就是链接。

```
<%
url=Request.QueryString("url")
If url<>"" then Response.Redirect(url)
%>
<html>
<body>

<%
set adrotator=Server.CreateObject("MSWC.AdRotator")
adrotator.TargetFrame="target='_blank'"
response.write(adrotator.GetAdvertisement("/example/aspe/advertisements2.txt"))
%>

<p>
<b>注释：</b>由于图像是随机变化的，
同时由于本页可供选择的图片很少，
因此经常会出现两次显示同一广告的情况。
</p>

<p>
<a href="/example/aspe/advertisements2.txt">
查看 advertisements2.txt
</a>
</p>
</body>
</html>
```

ASP AdRotator 组件

每当用户进入网站或刷新页面时，ASP AdRotator 组件就会创建一个 AdRotator 对象来显示一幅不同的图片。

语法：

```
<%
set adrotator=server.createobject("MSWC.AdRotator")
adrotator.GetAdvertisement("textfile.txt")
%>
```

实例

假设我们有一个文件名为 "banners.asp"。它类似于这样：

```
<html%>
<body%>
<%
set adrotator=Server.CreateObject("MSWC.AdRotator")
response.write(adrotator.GetAdvertisement("ads.txt"))
%>
</body%>
</html%>
```

文件 "ads.txt" 类似这样：

```
*
w3school.gif
http://www.w3school.com.cn/
Visit W3School
80
microsoft.gif
http://www.microsoft.com/
Visit Microsoft
20
```

"ads.txt" 文件中星号下面的代码定义了如何显示这些图像，链接地址，图像的替换文本，在每百次点击中的显示几率。我们可以看到，W3School 图片的显示几率是 80%，而 Microsoft 图片的显示几率是 20%。

注释：为了使这些链接在用户点击时可以正常工作，我们需要对文件 "ads.txt" 进行一点点小小的修改：

```
REDIRECT banners.asp
*
w3school.gif
http://www.w3school.com.cn/
Visit W3School
80
microsoft.gif
http://www.microsoft.com/
Visit Microsoft
20
```

转向页面会接收到名为 url 的变量的查询字符串，其中含有供转向的 URL。

注释：如需规定图像的高度、宽度和边框，我们可以在 REDIRECT 下面插入这些代码：

```
REDIRECT banners.asp
WIDTH 468
HEIGHT 60
BORDER 0
*
w3school.gif
...
```

```
...
```

最后要做的是把这些代码加入文件"banners.asp"中：

```
<%
url=Request.QueryString("url")
If url<>"" then Response.Redirect(url)
%>
<html>
<body>
<%
set adrotator=Server.CreateObject("MSWC.AdRotator")
response.write(adrotator.GetAdvertisement("textfile.txt"))
%>
</body>
</html>
```

好了，这就是全部的内容！

AdRotator 组件的属性

Border 属性

规定围绕广告的边框的尺寸。

```
<%
set adrot=Server.CreateObject("MSWC.AdRotator")
adrot.Border="2"
Response.Write(adrot.GetAdvertisement("ads.txt"))
%>
```

Clickable 属性

规定广告本身是否是超级链接。

```
<%
set adrot=Server.CreateObject("MSWC.AdRotator")
adrot.Clickable=false
Response.Write(adrot.GetAdvertisement("ads.txt"))
%>
```

TargetFrame 属性

显示广告的框架名称。

```
<%
set adrot=Server.CreateObject("MSWC.AdRotator")
adrot.TargetFrame="target='_blank'"
Response.Write(adrot.GetAdvertisement("ads.txt"))
```

```
%>
```

AdRotator 组件的方法

GetAdvertisement 方法

返回在页面中显示广告的 HTML。

```
<%
set adrot=Server.CreateObject("MSWC.AdRotator")
Response.Write(adrot.GetAdvertisement("ads.txt"))
%>
```

ASP Browser Capabilities 组件

实例

Browser Capabilities 组件

本例演示如何测定每一个访问网站的浏览器的类型、性能以及版本号。

```
<html>
<body>

<%
Set MyBrow=Server.CreateObject("MSWC.BrowserType")
%>

<table border="1" width="65%">
   <tr>
     <td width="52%">客户机操作系统</td>
     <td width="48%"><%=MyBrow.platform%></td>
   </tr>
   <tr>
     <td >Web 浏览器</td>
     <td ><%=MyBrow.browser%></td>
   </tr>
   <tr>
     <td>浏览器版本</td>
     <td><%=MyBrow.version%></td>
   </tr>
   <tr>
     <td>框架支持</td>
     <td><%=MyBrow.frames%></td>
   </tr>
   <tr>
     <td>表格支持</td>
     <td><%=MyBrow.tables%></td>
```

```
  </tr>
  <tr>
    <td>音频支持</td>
    <td><%=MyBrow.backgroundsounds%></td>
  </tr>
  <tr>
    <td>Cookies 支持</td>
    <td><%=MyBrow.cookies%></td>
  </tr>
  <tr>
    <td>VBScript 支持</td>
    <td><%=MyBrow.vbscript%></td>
  </tr>
  <tr>
    <td>JavaScript 支持</td>
    <td><%=MyBrow.javascript%></td>
  </tr>
</table>

</body>
</html>
```

ASP Browser Capabilities 组件

ASP Browser Capabilities 组件会创建一个 BrowserType 对象，这个对象可测定访问者浏览器的类型、性能以及版本号。

当浏览器连接到服务器时，就会向服务器发送一个 HTTP User Agent 报头。这个报头包含着有关浏览器的信息（比如浏览器类型和版本号）。BrowserType 对象会把报头中的信息与服务器上名为 "Browscap.ini" 的文件中的信息作比较。

如果标题中被发送的浏览器类型和版本号和 "Browsercap.ini" 文件中信息可以匹配，那么我们就可以使用 BrowserType 对象列出这个匹配的浏览器的相关属性。如果上述情况不匹配，这个对象会把每个属性设置为 "UNKNOWN"。

语法

```
<%
Set MyBrow=Server.CreateObject("MSWC.BrowserType")
%>
```

下面的例子对在 ASP 文件中创建一个 BrowserType 对象，并显示一个展示当前浏览器性能的表格：

```
<html>
<body>

<%
Set MyBrow=Server.CreateObject("MSWC.BrowserType")
%>
```

```
<table border="1" width="100%">
<tr>
<th>Client OS</th>
<th><%=MyBrow.platform%></th>
</tr><tr>
<td >Web Browser</td>
<td ><%=MyBrow.browser%></td>
</tr><tr>
<td>Browser version</td>
<td><%=MyBrow.version%></td>
</tr><tr>
<td>Frame support?</td>
<td><%=MyBrow.frames%></td>
</tr><tr>
<td>Table support?</td>
<td><%=MyBrow.tables%></td>
</tr><tr>
<td>Sound support?</td>
<td><%=MyBrow.backgroundsounds%></td>
</tr><tr>
<td>Cookies support?</td>
<td><%=MyBrow.cookies%></td>
</tr><tr>
<td>VBScript support?</td>
<td><%=MyBrow.vbscript%></td>
</tr><tr>
<td>JavaScript support?</td>
<td><%=MyBrow.javascript%></td>
</tr>
</table>

</body>
</html>
```

输出：

Client OS	WinNT
Web Browser	IE
Browser version	5.0
Frame support?	True
Table support?	True
Sound support?	True
Cookies support?	True
VBScript support?	True

JavaScript support?	True

Browscap.ini文件

"Browsercap.ini" 文件用于声明属性，并设置各浏览器的默认值。

本节内容不是关于如何 Browscap.ini 文件的教程，我们只提供一些关于 "Browsercap.ini" 的基础知识和概念。

"Browsercap.ini" 文件可包含下面的信息：

```
[;comments]
[HTTPUserAgentHeader]
[parent=browserDefinition]
[property1=value1]
[propertyN=valueN]
[Default Browser Capability Settings]
[defaultProperty1=defaultValue1]
[defaultPropertyN=defaultValueN]
```

参数	描述
comments	可选项。任何起始于分号的代码行都被 BrowserType 对象忽略
HTTPUserAgentHeader	可选项。规定与在 propertyN 中设定的 browser-property 值声明相关的 HTTP User Agent 报头。允许使用通配符。
browserDefinition	可选项。规定作为父浏览器使用的某个浏览器的 HTTP User Agent header-string。当前浏览器的定义会继承在父浏览器的定义中所有声明过的属性值。
propertyN	可选项。规定浏览器的属性。下面的表格列出了某些可能的属性： • ActiveXControls - 是否支持ActiveX控件? • Backgroundsounds - 是否支持背景声音? • Cdf - 是否支持针对网络广播（Webcasting）的频道定义格式（Channel Definition Format）? • Tables - 是否支持表格? • Cookies - 是否支持cookies? • Frames - 是否支持框架? • Javaapplets - 是否支持Java applets? • Javascript - 是否支持JScript? • Vbscript - 是否支持VBScript? • Browser - 定义浏览器的名称 • Beta - 浏览器是否为beta软件? • Platform - 规定浏览器运行的平台 • Version - 规定浏览器的版本号。

valueN	可选项。规定 propertyN 的值。可为字符串、整数（前缀为#）或者逻辑值。
defaultPropertyN	可选项。规定浏览器属性的名称，假如已定义的 HTTPUserAgentHeader 值中没有值能与浏览器发送的 HTTP 用户代理报头相匹配，则为这个属性分配一个默认的值。
defaultValueN	Optional. 规定 defaultPropertyN 的值。可为字符串、整数（前缀为 #）或者逻辑值。

"Browsercap.ini"文件会类似这样：

```
;IE 5.0
[IE 5.0]
browser=IE
Version=5.0
majorver=#5
minorver=#0
frames=TRUE
tables=TRUE
cookies=TRUE
backgroundsounds=TRUE
vbscript=TRUE
javascript=TRUE
javaapplets=TRUE
ActiveXControls=TRUE
beta=False;DEFAULT BROWSER
[*]
browser=Default
frames=FALSE
tables=TRUE
cookies=FALSE
backgroundsounds=FALSE
vbscript=FALSE
javascript=FALSE
```

ASP Content Linking 组件

实例

Content Linking 组件

本例会构建一个内容列表。

```
<html>
<body>

<p>下面的例子构建了一个内容列表：</p>
```

```
<%
dim c
dim i
set nl=server.createobject("MSWC.Nextlink")
c = nl.GetListCount("/example/aspe/links.txt")
i = 1
%>
<ul>
<%do while (i <= c) %>
<li><a href="<%=nl.GetNthURL("/example/aspe/links.txt", i)%>">
<%=nl.GetNthDescription("/example/aspe/links.txt", i)%></a>
<%
i = (i + 1)
loop
%>
</ul>

<p>文本文件包含页面 URL 的列表和链接描述。
每行文本针对一个页面。
请注意，URL 和描述必须由 TAB 字符分隔。
</p>

<p>
<a href="/example/aspe/links.txt">查看 links.txt</a>
</p>

</body>
</html>
```

Content Linking 组件 *2*

本例使用 Content Linking 组件在一个文本文件所列的页面间进行导航。

```
<html>
<body>

<h1>这是页面 1</h1>

<%
Set nl=Server.CreateObject("MSWC.NextLink")
If (nl.GetListIndex("/example/aspe/links2.txt")>1) Then
%>
<a href="<%Response.Write(nl.GetPreviousURL("/example/aspe/links2.txt"))%>">
上一页
</a>
<%End If%>

<a href="<%Response.Write(nl.GetNextURL("/example/aspe/links2.txt"))%>">
下一页
</a>
```

```
<p>本例使用 Content Linking 组件
对文本文件中的 URL 进行导航。</p>

<p><a href="/example/aspe/links2.txt">查看 links2.txt</a></p>
</body>
</html>
```

ASP Content Linking 组件

ASP Content Linking 组件用于创建快捷便利的导航系统。

Content Linking 组件会返回一个 Nextlink 对象，这个对象用于容纳需要导航网页的一个列表。

语法

```
<%
Set nl=Server.CreateObject( "MSWC.NextLink" )
%>
```

首先，我们会创建文本文件 - "links.txt"。此文件包含需要导航的页面的相关信息。页面的排列顺序应该与它们的显示顺序相同，并包含对每个文件的描述（使用制表符来分隔文件名和描述信息）。

注释：如果你希望向列表添加文件信息，或者改变在列表中的页面顺序，那么你需要做的所有事情仅仅是修改这个文本文件而已！然后导航系统会自动地更新！

"links.txt":

```
asp_intro.asp ASP 简介
asp_syntax.asp ASP 语法
asp_variables.asp ASP 变量
asp_procedures.asp ASP 程序
```

请在上面列出的页面中放置这行代码：<!-- #include file="nlcode.inc"-->。这行代码会在 "links.txt" 中列出每个页面上引用下面这段代码，这样导航就可以工作了。

"nlcode.inc":

```
<%
'Use the Content Linking Component
'to navigate between the pages listed
'in links.txt

dim nl
Set nl=Server.CreateObject("MSWC.NextLink")
if (nl.GetListIndex("links.txt")>1) then
  Response.Write("<a href='" & nl.GetPreviousURL("links.txt"))
  Response.Write("'>Previous Page</a>")
end if
Response.Write("<a href='" & nl.GetNextURL("links.txt"))
```

```
Response.Write("'>Next Page</a>")
%>
```

ASP Content Linking 组件的方法

GetListCount 方法

返回内容链接列表文件中所列项目的数目：

```
<%
dim nl,c
Set nl=Server.CreateObject("MSWC.NextLink")
c=nl.GetListCount("links.txt")
Response.Write("There are ")
Response.Write(c)
Response.Write(" items in the list")
%>
```

输出：

There are 4 items in the list

GetListIndex 方法

返回在内容链接列表文件中当前文件的索引号。第一个条目的索引号是 1。假如当前页面不在列表文件中，则返回 0。

例子

```
<%
dim nl,c
Set nl=Server.CreateObject("MSWC.NextLink")
c=nl.GetListIndex("links.txt")
Response.Write("Item number ")
Response.Write(c)
%>
```

输出：

Item number 3

GetNextDescription 方法

返回在内容链接列表文件中所列的下一个条目的文本描述。假如在列表文件中没有找到当前文件，则列表中最后一个页面的文本描述。

例子

```
<%
```

```
dim nl,c
Set nl=Server.CreateObject("MSWC.NextLink")
c=nl.GetNextDescription("links.txt")
Response.Write("Next ")
Response.Write("description is: ")
Response.Write(c)
%>
```

输出：Next description is: ASP Variables

GetNextURL 方法

返回在内容链接列表文件中所列的下一个条目的 URL。假如在列表文件中没有找到当前文件，则列表中最后一个页面的 URL。

例子

```
<%
dim nl,c
Set nl=Server.CreateObject("MSWC.NextLink")
c=nl.GetNextURL("links.txt")
Response.Write("Next ")
Response.Write("URL is: ")
Response.Write(c)
%>
```

输出：Next URL is: asp_variables.asp

GetNthDescription 方法

返在内容链接列表文件中所列的第 N 个页面的描述信息。

例子

```
<%
dim nl,c
Set nl=Server.CreateObject("MSWC.NextLink")
c=nl.GetNthDescription("links.txt",3)
Response.Write("Third ")
Response.Write("description is: ")
Response.Write(c)
%>
```

输出：Third description is: ASP Variables

GetNthURL 方法

返在内容链接列表文件中所列的第 N 个页面的 URL。

例子

```
<%
dim nl,c
Set nl=Server.CreateObject("MSWC.NextLink")
c=nl.GetNthURL("links.txt",3)
Response.Write("Third ")
Response.Write("URL is: ")
Response.Write(c)
%>
```

输出：Third URL is: asp_variables.asp

GetPreviousDescription 方法

返回在内容链接列表文件中所列前一个条目的文本描述。假如在列表文件中没有找到当前文件，则列表中第一个页面的文本描述。

例子

```
<%
dim nl,c
Set nl=Server.CreateObject("MSWC.NextLink")
c=nl.GetPreviousDescription("links.txt")
Response.Write("Previous ")
Response.Write("description is: ")
Response.Write(c)
%>
```

输出：Previous description is: ASP Variables

GetPreviousURL 方法

返回在内容链接列表文件中所列前一个条目的 URL。假如在列表文件中没有找到当前文件，则列表中第一个页面的URL。

例子

```
<%
dim nl,c
Set nl=Server.CreateObject("MSWC.NextLink")
c=nl.GetPreviousURL("links.txt")
Response.Write("Previous ")
Response.Write("URL is: ")
Response.Write(c)
%>
```

输出：Previous URL is: asp_variables.asp

ASP Content Rotator (ASP 3.0)

实例

Content Rotator 组件

每当用户访问或者刷新页面时，该组件就会显示不同的 HTML 内容字符串。

```
<html>
<body>

<p><b>注释：</b>如果服务器未安装 ASP 3.0，则本例无法运行。</p>

<p>
<a href="/example/aspe/textads.asp">查看 textads.txt</a>
</p>

<%
set cr=server.createobject("MSWC.ContentRotator")
response.write(cr.ChooseContent("/example/aspe/textads.txt"))
%>

<p>
<b>注释：</b>由于文本文件中的内容字符串是随机改变的，
同时本页只有四条内容可供选择，
因此，有时页面会连续显示两次相同的内容。
</p>

</body>
</html>
```

ASP Content Rotator 组件

ASP Content Rotator 组件会创建一个 ContentRotator 对象，每当用户访问或者刷新某个页面时，该对象就会显示一段不同的 HTML 内容字符串。一个名为内容目录文件（Content Schedule File）的文本文件包含着有关内容字符串的信息。

内容字符串可包含 HTML 标签，这样你就可以显示 HTML 可呈现的任何内容：文本、图像、颜色或者超级链接。

语法

```
<%
Set cr=Server.CreateObject( "MSWC.ContentRotator" )
%>
```

每当某用户查看网页时，下面这个例子就会显示不同的内容。首先在站点根目录的子文件夹 text 中创建一个名为 "textads.txt" 的文件。

"textads.txt":

```
%% #1
This is a great day!!

%% #2
<h1>Smile</h1>

%% #3
<img src="smiley.gif">

%% #4
Here's a <a href="http://www.w3school.com.cn">link</a>
```

注意：在每个内容字符串起始位置的#号码。这个号码是一个可选的参数，用来 HTML 内容字符串的相对权重。在本例中，Content Rotator 有十分之一的几率显示第一个内容字符串，有十分之二的几率显示第二个内容字符串，有十分之三的几率显示第三个字符串，而第四个字符串为十分之四的几率。

然后，创建一个 ASP 文件，并插入下面的代码：

```
<html>
<body>

<%
set cr=server.createobject("MSWC.ContentRotator")
response.write(cr.ChooseContent("text/textads.txt"))
%>

</body>
</html>
```

ASP Content Rotator 组件的方法

方法	描述	实例
ChooseContent	获取并显示某个内容字符串	<% dim cr Set cr=Server.CreateObject("MSWC.ContentRotator") response.write(cr.ChooseContent("text/textads.txt")) %> 输出：
	取	

GetAllContent	回并显示文本文件中所有的内容字符串	<% dim cr Set cr=Server.CreateObject("MSWC.ContentRotator") response.write(cr.GetAllContent("text/textads.txt")) %> 输出：

AJAX 简介

AJAX 旨在不重载整个页面的情况下对网页的某些部分进行更新。

AJAX 是什么？

AJAX = Asynchronous JavaScript and XML（异步 JavaScript 和 XML）。

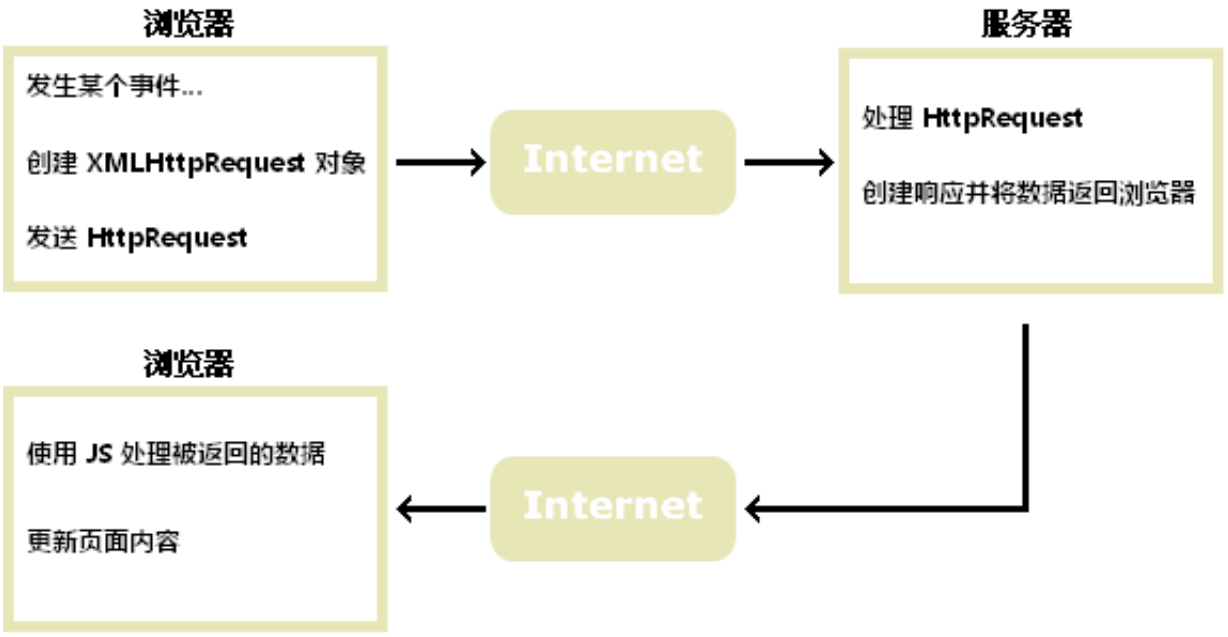
AJAX 是一种创建快速动态网页的技术。

通过在后台与服务器交换少量数据，AJAX 允许网页进行异步更新。这意味着，在不重新加载整个网页的情况下，对网页某些部分进行更新。

典型的网页（不使用 AJAX），如果内容发生变化，就必须重载整个页面。

使用 AJAX 的典型案例包括：谷歌地图、腾讯微博、优酷视频等等。

AJAX 如何工作



AJAX 基于因特网标准

AJAX 基于因特网标准，并使用以下技术组合：

- XMLHttpRequest 对象（与服务器异步交互数据）
- JavaScript/DOM （显示/取回信息）
- CSS （设置数据的样式）
- XML （常用作数据传输的格式）

提示：AJAX 应用程序是独立于浏览器和平台的！

谷歌搜索建议（Google Suggest）

随着谷歌搜索建议功能在 2005 的发布，AJAX 开始流行起来。

谷歌搜索建议使用 AJAX 创造出动态性极强的 web 界面：当您在谷歌的搜索框中键入内容时，JavaScript 会把字符发送到服务器，服务器则会返回建议列表。

今天就开始使用 AJAX

在我们的 ASP 教程中，我们将演示 AJAX 如何更新网页的某个部分，在不重载整个页面的情况下。我们将使用 ASP 来编写服务器端的脚步。

如果您希望学习更多有关 AJAX 的知识，请访问我们的 AJAX 教程。

ASP - AJAX 与 ASP

AJAX 用于创建动态性更强的应用程序。

AJAX ASP 实例

下面的例子将演示当用户在输入框中键入字符时，网页如何与服务器进行通信：

实例

```
<html>
<head>
<script type="text/javascript">
function showHint(str)
{
var xmlhttp;
if (str.length==0)
  {
  document.getElementById("txtHint").innerHTML="";
  return;
  }
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","/ajax/gethint.asp?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>

<h3>请在下面的输入框中键入字母（A - Z）：</h3>
<form action="">
姓氏：<input type="text" id="txt1" onkeyup="showHint(this.value)" />
</form>
<p>建议：<span id="txtHint"></span></p>

</body>
</html>
```

实例解释 - HTML 页面

当用户在上面的输入框中键入字符时，会执行 "showHint()" 函数。该函数由 "onkeyup" 事件触发：

```
<!DOCTYPE html>
<html>
<head>
<script>
function showHint(str)
{
if (str.length==0)
  {
  document.getElementById("txtHint").innerHTML="";
  return;
  }
if (window.XMLHttpRequest)
  {// 针对 IE7+, Firefox, Chrome, Opera, Safari 的代码
  xmlhttp=new XMLHttpRequest();
  }
else
  {// 针对 IE6, IE5 的代码
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","gethint.asp?q="+str,true);
xmlhttp.send();
}
</script>
</head
<body>

<p><b>请在输入框中输入英文字符：</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)" size="20">
</form>
<p>Suggestions: <span id="txtHint"></span></p>

</body>
</html>
```

源代码解释：

如果输入框是空的（str.length==0），该函数会清空占位符 txtHint 的内容，并推出该函数。

如果输入框不是空的，那么 showHint() 会执行以下步骤：

- 创建 XMLHttpRequest 对象
- 创建在服务器响应就绪时执行的函数

- 向服务器上的文件发送请求
- 请注意添加到 URL 末端的参数（q）（包含输入框的内容）

ASP 文件

上面这段 JavaScript 调用的服务器页面是名为 "gethint.asp" 的 ASP 文件。

"gethint.asp" 中的源代码会检查姓名数组，然后向浏览器返回对应的姓名：

```
<%
response.expires=-1
dim a(30)
'Fill up array with names
a(1)="Anna"
a(2)="Brittany"
a(3)="Cinderella"
a(4)="Diana"
a(5)="Eva"
a(6)="Fiona"
a(7)="Gunda"
a(8)="Hege"
a(9)="Inga"
a(10)="Johanna"
a(11)="Kitty"
a(12)="Linda"
a(13)="Nina"
a(14)="Ophelia"
a(15)="Petunia"
a(16)="Amanda"
a(17)="Raquel"
a(18)="Cindy"
a(19)="Doris"
a(20)="Eve"
a(21)="Evita"
a(22)="Sunniva"
a(23)="Tove"
a(24)="Unni"
a(25)="Violet"
a(26)="Liza"
a(27)="Elizabeth"
a(28)="Ellen"
a(29)="Wenche"
a(30)="Vicky"

'从 URL 获得参数 q
q=ucase(request.querystring("q"))

'如果长度 q>0，则从数组中查找所有提示
if len(q)>0 then
  hint=""
```

```
  for i=1 to 30
    if q=ucase(mid(a(i),1,len(q))) then
      if hint="" then
        hint=a(i)
      else
        hint=hint & " , " & a(i)
      end if
    end if
  next
end if

'如果未找到提示，则输出 "no suggestion"
'or output the correct values
if hint="" then
  response.write("no suggestion")
else
  response.write(hint)
end if
%>
```

源代码解释：

如果 JavaScript 发送了任何文本（即 strlen($q) 大于 0），则会发生：

- 查找匹配来自 JavaScript 的字符的姓名
- 如果未找到匹配，则将响应字符串设置为 "no suggestion"
- 如果找到一个或多个匹配姓名，则用所有姓名设置响应字符串
- 把响应发送到占位符 "txtHint"

AJAX 数据库实例

AJAX 可用来与数据库进行相互的通信。

AJAX 数据库实例

下面的例子演示网页如何通过 AJAX 从数据库中读取信息：

```
<html>
<head>
<script type="text/javascript">
function showCustomer(str)
{
var xmlhttp;
if (str=="")
  {
  document.getElementById("txtHint").innerHTML="";
  return;
  }
if (window.XMLHttpRequest)
```

```
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","/ajax/getcustomer.asp?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>

<form action="" style="margin-top:15px;">
<label>请选择一位客户：
<select name="customers" onchange="showCustomer(this.value)" style="font-family:Verdana
<option value="APPLE">Apple Computer, Inc.</option>
<option value="BAIDU ">BAIDU, Inc</option>
<option value="Canon">Canon USA, Inc.</option>
<option value="Google">Google, Inc.</option>
<option value="Nokia">Nokia Corporation</option>
<option value="SONY">Sony Corporation of America</option>
</select>
</label>
</form>
<br />
<div id="txtHint">客户信息将在此处列出 ...</div>

</body>
</html>
```

实例解释 - **HTML** 页面

当用户在上面的下拉列表中选择某位客户时，会执行名为 "showCustomer()" 的函数。该函数由 "onchange" 事件触发：

```
<!DOCTYPE html>
<html>
<head>
<script>
function showCustomer(str)
```

```
{
if (str=="")
  {
  document.getElementById("txtHint").innerHTML="";
  return;
  }
if (window.XMLHttpRequest)
  {// 针对 IE7+, Firefox, Chrome, Opera, Safari 的代码
  xmlhttp=new XMLHttpRequest();
  }
else
  {// 针对 IE6, IE5 的代码
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","getcustomer.asp?q="+str,true);
xmlhttp.send();
}
</script>
</head
<body>

<form>
<select name="customers" onchange="showCustomer(this.value)">
<option value="">Select a customer:</option>
<option value="ALFKI">Alfreds Futterkiste</option>
<option value="NORTS ">North/South</option>
<option value="WOLZA">Wolski Zajazd</option>
</select>
</form>
<br>
<div id="txtHint">客户信息将在此处列出...</div>

</body>
</html>
```

源代码解释：

如果没有选择客户（str.length 等于 0），那么该函数会清空 txtHint 占位符，然后退出该函数。

如果已选择一位客户，则 showCustomer() 函数会执行以下步骤：

- 创建 XMLHttpRequest 对象
- 创建在服务器响应就绪时执行的函数
- 向服务器上的文件发送请求

- 请注意添加到 URL 末端的参数（q）（包含下拉列表的内容）

ASP 文件

上面这段 JavaScript 调用的服务器页面是名为 "getcustomer.asp" 的 ASP 文件。

"getcustomer.asp" 中的源代码会运行一次针对数据库的查询，然后在 HTML 表格中返回结果：

```
<%
response.expires=-1
sql="SELECT * FROM CUSTOMERS WHERE CUSTOMERID="
sql=sql & "'" & request.querystring("q") & "'"

set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs=Server.CreateObject("ADODB.recordset")
rs.Open sql,conn

response.write("<table>")
do until rs.EOF
  for each x in rs.Fields
    response.write("<tr><td><b>" & x.name & "</b></td>")
    response.write("<td>" & x.value & "</td></tr>")
  next
  rs.MoveNext
loop
response.write("</table>")
%>
```

ADO 简介

ADO 被用于从网页访问数据库。

您应当具备的基础知识

在继续学习之前，您需要对下面的知识有基本的了解：

- WWW、HTML 以及对网站构建的基本了解
- ASP（动态服务器页面）
- SQL（结构化查询语言）

如果您希望首先学习这些项目，请在我们的 首页 访问这些教程。

什么是 **ADO**？

- ADO 是一项微软的技术
- ADO 指 ActiveX 数据对象（*A*ctiveX *D*ata *O*bjects）

- ADO 是一个微软的 Active-X 组件
- ADO 会随微软的 IIS 被自动安装
- ADO 是一个访问数据库中数据的编程接口

从 **ASP** 页面访问数据库

从一个 ASP 页面内部访问数据库的通常的方法是：

1. 创建一个到数据库的 ADO 连接
2. 打开数据库连接
3. 创建 ADO 记录集
4. 从记录集提取您需要的数据
5. 关闭记录集
6. 关闭连接

ADO 数据库连接

在从某个网页访问数据之前，必须先建立一个数据库连接。

创建一个 **DSN-less** 数据库连接

连接到某一个数据库的最简单的方法是使用一个 DSN-less 连接。DSN-less 连接可被用于您的站点上的任何微软 Access 数据库。

假设您拥有一个名为 "northwind.mdb" 的数据库位于 "c:/webdata/" 的 web 目录中，您可以使用下面的 ASP 代码连接到此数据库：

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
%>
```

注意，在上面的例子中，您必须规定微软的 Access 数据库驱动程序（Provider），以及此数据库在计算机上的物理路径。

创建一个 **ODBC** 数据库连接

假设您拥有一个名为 "northwind" 的 ODBC 数据库，您可以使用下面的 ASP 代码连接到此数据库：

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Open "northwind"
%>
```

通过一个 ODBC 连接，您可以连接到您的网络中任何计算机上的任何数据库，只要 ODBC 连接是可用的。

到 **MS Access** 数据库的 **ODBC** 连接

下面为您讲解如何创建到一个 MS Access 数据库的连接：

1. 打开控制面板中的 *ODBC* 图标
2. 选择系统 *ODBC* 选项卡
3. 点击 ODBC 选项卡中的添加按钮
4. 选择 Driver to Microsoft Access，然后点击完成按钮
5. 在下一个窗口中点击“选择”按钮来定位数据库
6. 为此数据库赋予一个数据源名称（*D*ata *S*ource *N*ame，DSN）
7. 点击"确定"

注意：此配置必须在您的网站所在的计算机上完成。假如您正在自己的计算机上运行PWS或者IIS，此架构是可以运行的，但是假如您的网站位于一台远程的服务器，您就必须拥有此服务器的物理访问权限，或者请您的 web 主机提供商为您做这些事情。

ADO 连接对象（**ADO Connection Object**）

ADO 连接对象用来创建到某个数据源的开放连接。通过此连接，您可以对此数据库进行访问和操作。

查看此连接对象的所有方法和属性。

ADO Recordset（记录集）

如需读取数据库的数据，那么其中的数据必须首先被载入一个记录集中。

创建一个 **ADO** 表记录集（**ADO Table Recordset**）

在 ADO 数据库连接创建之后，如上一章所述，接下来就可以建立一个 ADO 记录集了。

假设我们有一个名为 "Northwind" 的数据库，我们可以通过下面的代码访问数据库中的 "Customers" 表：

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

set rs=Server.CreateObject("ADODB.recordset")
rs.Open "Customers", conn
%>
```

创建一个 **ADO SQL** 记录集 （**ADO SQL Recordset**）

我们也可使用 SQL 访问 "Customers" 表中的数据：

```
<%
```

```
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

set rs=Server.CreateObject("ADODB.recordset")
rs.Open "Select * from Customers", conn
%>
```

从记录集中提取数据

在记录集被打开后，我们可以从记录集中提取数据。

假设我们用一个名为 "Northwind" 的数据库，我们可以通过下面的代码访问数据库中 "Customers" 表：

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

set rs=Server.CreateObject("ADODB.recordset")
rs.Open "Select * from Customers", conn

for each x in rs.fields
   response.write(x.name)
   response.write(" = ")
   response.write(x.value)
next
%>
```

ADO 记录集对象（ADO Recordset Object）

ADO Recordset 对象可被用来容纳来自数据库表的记录集。

查看 ADO Recordset 对象的所有方法和属性。

ADO 显示

显示来自记录集中的数据的最常用的方法，就是把数据显示在 **HTML** 表格中。

实例

显示记录

如何首先创建一个数据库连接，然后创建一个记录集，然后把其中的数据显示在HTML中。

```
<html>
<body>

<%
```

```
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
rs.Open "Select * from Customers", conn

do until rs.EOF
    for each x in rs.Fields
       Response.Write(x.name)
       Response.Write(" = ")
       Response.Write(x.value & "<br />")
    next
    Response.Write("<br />")
    rs.MoveNext
loop

rs.close
conn.close
%>

</body>
</html>
```

在 *HTML* 表格中显示记录

如何把数据表中的数据显示在HTML表格中。

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))

set rs = Server.CreateObject("ADODB.recordset")
rs.Open "SELECT Companyname, Contactname FROM Customers", conn
%>

<table border="1" width="100%">
<%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
       <td><%Response.Write(x.value)%></td>
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
```

```
%>
</table>

</body>
</html>
```

向 *HTML* 表格添加标题

如何向HTML表格添加标题，以使其可读性更强。

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers"
rs.Open sql, conn
%>

<table border="1" width="100%">
<tr>
<%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
next%>
</tr>
<%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
       <td><%Response.Write(x.value)%></td>
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
%>
</table>

</body>
</html>
```

向 *HTML* 表格添加颜色

如何向HTML表格添加颜色，以使其更加美观。

```
<html>
<body>
```

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers"
rs.Open sql, conn
%>

<table border="1" width="100%" bgcolor="#fff5ee">
<tr>
<%for each x in rs.Fields
    response.write("<th align='left' bgcolor='#b0c4de'>" & x.name & "</th>")
next%>
</tr>
<%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
       <td><%Response.Write(x.value)%></td>
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
%>
</table>

</body>
</html>
```

显示字段名称和字段值

我们有一个名为 "Northwind" 的数据库，并且我们希望显示出 "Customers" 表中的数据（记得以 .asp 为扩展名来保存这个文件）：

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

set rs = Server.CreateObject("ADODB.recordset")
rs.Open "SELECT * FROM Customers", conn

do until rs.EOF
  for each x in rs.Fields
    Response.Write(x.name)
```

```
    Response.Write(" = ")
    Response.Write(x.value & "<br />")
  next
  Response.Write("<br />")
  rs.MoveNext
loop

rs.close
conn.close
%>

</body>
</html>
```

在一个 **HTML** 表格中显示字段名称和字段的值

我们也可以通过下面的代码把表 "Customers" 中的数据显示在一个 HTML 表格中：

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

set rs = Server.CreateObject("ADODB.recordset")
rs.Open "SELECT Companyname, Contactname FROM Customers", conn
%>

<table border="1" width="100%">
<%do until rs.EOF%>
   <tr>
   <%for each x in rs.Fields%>
      <td><%Response.Write(x.value)%></td>
   <%next
   rs.MoveNext%>
   </tr>
<%loop
rs.close
conn.close
%>
</table>

</body>
</html>
```

向 **HTML** 表格添加标题

我们希望为这个 HTML 表格添加标题，这样它就更易读了：

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers"
rs.Open sql, conn
%>

<table border="1" width="100%">
  <tr>
  <%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
  next%>
  </tr>
  <%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
      <td><%Response.Write(x.value)%></td>
    <%next
    rs.MoveNext%>
    </tr>
  <%loop
  rs.close
  conn.close
  %>
</table>

</body>
</html>
```

ADO 查询

我们可以使用 **SQL** 来创建查询，这样就可以指定仅查看选定的记录和字段。

实例

显示 ***"Companyname"*** 以 ***A*** 开头的记录

如何仅仅显示 "Customers" 表的 "Companyname" 字段中以 A 开头的记录。

```
<html>
<body>
```

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers WHERE CompanyName LIKE 'A%'"
rs.Open sql, conn
%>

<table border="1" width="100%">
<tr>
<%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
next%>
</tr>
<%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
       <td><%Response.Write(x.value)%></td>
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
%>
</table>

</body>
</html>
```

显示 ***"Companyname"*** 大于 ***E*** 的记录

如何仅仅显示 "Customers" 表的 "Companyname" 字段中大于 E 的记录。

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers WHERE CompanyName>'E'"
rs.Open sql, conn
%>

<table border="1" width="100%">
<tr>
<%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
```

```
next%>
</tr>
<%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
       <td><%Response.Write(x.value)%> </td>
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
%>
</table>

</body>
</html>
```

仅显示西班牙的客户

如何仅仅显示 "Customers" 表中的西班牙客户。

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers WHERE Country='China'"
rs.Open sql, conn
%>

<table border="1" width="100%">
<tr>
<%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
next%>
</tr>
<%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
       <td><%Response.Write(x.value)%> </td>
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
%>
```

```
</table>

</body>
</html>
```

让用户来选择筛选标准

让用户根据国别来选择客户

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))

set rs=Server.CreateObject("ADODB.recordset")
sql="SELECT DISTINCT Country FROM Customers ORDER BY Country"
rs.Open sql,conn

country=request.form("country")

%>

<form method="post">
Choose Country <select name="country">
<%  do until rs.EOF
    response.write("<option")
    if rs.fields("country")=country then
      response.write(" selected")
    end if
    response.write(">")
    response.write(rs.fields("Country"))
    rs.MoveNext
loop
rs.Close
set rs=Nothing %>
</select>
<input type="submit" value="Show customers">
</form>

<%
if country<>"" then
   sql="SELECT Companyname,Contactname,Country FROM Customers WHERE country='" & countr
   set rs=Server.CreateObject("ADODB.Recordset")
   rs.Open sql,conn
%>
   <table width="100%" cellspacing="0" cellpadding="2" border="1">
   <tr>
     <th>Companyname</th>
```

```
    <th>Contactname</th>
    <th>Country</th>
  </tr>
<%
do until rs.EOF
  response.write("<tr>")
  response.write("<td>" & rs.fields("companyname") & "</td>")
  response.write("<td>" & rs.fields("contactname") & "</td>")
  response.write("<td>" & rs.fields("country") & "</td>")
  response.write("</tr>")
  rs.MoveNext
loop
rs.close
conn.Close
set rs=Nothing
set conn=Nothing%>
</table>
<%  end if %>

</body>
</html>
```

显示选定的数据

我们希望仅仅显示 "Customers" 表的 "Companyname" 字段中以 A 开头的记录：

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

set rs=Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers
WHERE CompanyName LIKE 'A%'"
rs.Open sql, conn
%>

<table border="1" width="100%">
  <tr>
  <%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
  next%>
  </tr>
  <%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
```

```
      <td><%Response.Write(x.value)%></td>
    <%next
    rs.MoveNext%>
    </tr>
  <%loop
  rs.close
  conn.close%>
</table>

</body>
</html>
```

ADO 排序

我们可以使用**SQL**来规定如何对记录集中的数据进行排序。

实例

根据指定的字段名处对记录进行升序排序

如何根据指定字段名对数据进行排序

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers ORDER BY CompanyName"
rs.Open sql, conn
%>

<table border="1" width="100%">
<tr>
<%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
next%>
</tr>
<%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
       <td><%Response.Write(x.value)%></td>
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
```

```
%>
</table>

</body>
</html>
```

根据指定的字段名处对记录进行降序排序

如何根据指定字段名对数据进行排序

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM Customers ORDER BY CompanyName DESC"
rs.Open sql, conn
%>

<table border="1" width="100%">
<tr>
<%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
next%>
</tr>
<%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
       <td><%Response.Write(x.value)%> </td>
    <%next
    rs.MoveNext%>
    </tr>
<%loop
rs.close
conn.close
%>
</table>

</body>
</html>
```

让用户来选择根据哪列进行排序

让用户来选择根据哪列进行排序

```
<html>
<body>
```

```
<table border="1" width="100%" bgcolor="#fff5ee">
<tr>
<th align="left" bgcolor="#b0c4de">
<a href="/example/adoe/demo_adoe_sort_3.asp?sort=companyname">Company</a>
</th>
<th align="left" bgcolor="#b0c4de">
<a href="/example/adoe/demo_adoe_sort_3.asp?sort=contactname">Contact</a>
</th>
</tr>
<%
if request.querystring("sort")<>"" then
   sort=request.querystring("sort")
else
   sort="companyname"
end if

set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs=Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname,Contactname FROM Customers ORDER BY " & sort
rs.Open sql,conn

do until rs.EOF
   response.write("<tr>")
   for each x in rs.Fields
     response.write("<td>" & x.value & "</td>")
   next
   rs.MoveNext
   response.write("</tr>")
loop
rs.close
conn.close
%>
</table>

</body>
</html>
```

对数据进行排序

我们希望显示 "Customers" 表中的"Companyname"和"Contactname"字段，并根据"Companyname"进行排序（请记得用.asp为后缀保存）：

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
```

```
conn.Open "c:/webdata/northwind.mdb"

set rs = Server.CreateObject("ADODB.recordset")
sql="SELECT Companyname, Contactname FROM
Customers ORDER BY CompanyName"
rs.Open sql, conn
%>

<table border="1" width="100%">
  <tr>
  <%for each x in rs.Fields
    response.write("<th>" & x.name & "</th>")
  next%>
  </tr>
  <%do until rs.EOF%>
    <tr>
    <%for each x in rs.Fields%>
      <td><%Response.Write(x.value)%></td>
    <%next
    rs.MoveNext%>
    </tr>
  <%loop
  rs.close
  conn.close%>
</table>

</body>
</html>
```

ADO 添加记录

我们可以使用 **SQL** 的 **INSERT INTO** 命令向数据库中的表添加记录。

向数据库中的表添加记录

我们希望向 Northwind 数据库中的 Customers 表添加一条新的记录。我们首先要创建一个表单，这个表单包含了我们需要从中搜集数据的输入域：

```
<html>
<body>

<form method="post" action="demo_add.asp">
<table>
<tr>
<td>CustomerID:</td>
<td><input name="custid"></td>
</tr><tr>
<td>Company Name:</td>
<td><input name="compname"></td>
```

```
</tr><tr>
<td>Contact Name:</td>
<td><input name="contname"></td>
</tr><tr>
<td>Address:</td>
<td><input name="address"></td>
</tr><tr>
<td>City:</td>
<td><input name="city"></td>
</tr><tr>
<td>Postal Code:</td>
<td><input name="postcode"></td>
</tr><tr>
<td>Country:</td>
<td><input name="country"></td>
</tr>
</table>
<br /><br />
<input type="submit" value="Add New">
<input type="reset" value="Cancel">
</form>

</body>
</html>
```

当用户按下确认按钮时，这个表单就会被送往名为 "demo_add.asp" 的文件。文件 "demo_add.asp" 中含有可向 Customers 表添加一条新记录的代码：

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

sql="INSERT INTO customers (customerID,companyname,"
sql=sql & "contactname,address,city,postalcode,country)"
sql=sql & " VALUES "
sql=sql & "('" & Request.Form("custid") & "',"
sql=sql & "'" & Request.Form("compname") & "',"
sql=sql & "'" & Request.Form("contname") & "',"
sql=sql & "'" & Request.Form("address") & "',"
sql=sql & "'" & Request.Form("city") & "',"
sql=sql & "'" & Request.Form("postcode") & "',"
sql=sql & "'" & Request.Form("country") & "')"

on error resume next
conn.Execute sql,recaffected
if err<>0 then
```

```
  Response.Write("No update permissions!")
else
  Response.Write("<h3>" & recaffected & " record added</h3>")
end if
conn.close
%>

</body>
</html>
```

重要事项

在您使用 INSERT command 命令时，请注意以下事项：

- 如果表含有一个主键，请确保向主键字段添加的值是唯一且非空的（否则，provider 就不会追加此记录，亦或发生错误）
- 如果表含有一个自动编号的字段，请不要在 INSERT 命令中涉及此字段（这个字段的值是由 provider 负责的）

关于无数据字段

在 MS Access 数据库中，假如您将 AllowZeroLength 属性设置为 “Yes”，您可以在文本、超链接以及备忘字段输入零长度的字符串 ("")。

注释：并非所有的数据库都支持零长度的字符串，因而当添加带有空白字段的记录时可能会产生错误。因此，检查您使用的数据库所支持的数据类型是很重要的。

ADO 更新记录

我们可使用 **SQL** 的 **UPDATE** 来更新数据库表中的某条记录。

更新数据库表中的记录

我们希望更新 Northwind 数据中 Customers 表的某条记录。首先我们需要创建一个表格，来列出 Customers 中的所有记录。

```
<html>
<body>
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
set rs=Server.CreateObject("ADODB.Recordset")
rs.open "SELECT * FROM customers",conn
%>

<h2>List Database</h2>
<table border="1" width="100%">
```

```
<tr>
<%
for each x in rs.Fields
  response.write("<th>" & ucase(x.name) & "</th>")
next
%>
</tr>
<% do until rs.EOF %>
<tr>
<form method="post" action="demo_update.asp">
<%
for each x in rs.Fields
  if lcase(x.name)="customerid" then%>
    <td>
    <input type="submit" name="customerID" value="<%=x.value%>">
    </td>
  <%else%>
    <td><%Response.Write(x.value)%></td>
  <%end if
next
%>
</form>
<%rs.MoveNext%>
</tr>
<%
loop
conn.close
%>
</table>

</body>
</html>
```

如果用户点击 "customerID" 列中的按钮，会打开一个新文件 "demo_update.asp"。此文件包含了创建输入域的源代码，这些输入域基于数据库中记录的字段，同时也含有一个保存修改的“更新按钮”：

```
<html>
<body>

<h2>Update Record</h2>
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

cid=Request.Form("customerID")

if Request.form("companyname")="" then
  set rs=Server.CreateObject("ADODB.Recordset")
  rs.open "SELECT * FROM customers WHERE customerID='" & cid & "'",conn
```

```
  %>
  <form method="post" action="demo_update.asp">
  <table>
  <%for each x in rs.Fields%>
  <tr>
  <td><%=x.name%></td>
  <td><input name="<%=x.name%>" value="<%=x.value%>"></td>
  <%next%>
  </tr>
  </table>
  <br /><br />
  <input type="submit" value="Update record">
  </form>
<%
else
  sql="UPDATE customers SET "
  sql=sql & "companyname='" & Request.Form("companyname") & "',"
  sql=sql & "contactname='" & Request.Form("contactname") & "',"
  sql=sql & "address='" & Request.Form("address") & "',"
  sql=sql & "city='" & Request.Form("city") & "',"
  sql=sql & "postalcode='" & Request.Form("postalcode") & "',"
  sql=sql & "country='" & Request.Form("country") & "'"
  sql=sql & " WHERE customerID='" & cid & "'"
  on error resume next
  conn.Execute sql
  if err<>0 then
    response.write("No update permissions!")
  else
    response.write("Record " & cid & " was updated!")
  end if
end if
conn.close
%>

</body>
</html>
```

ADO 删除记录

我们可使用 **SQL** 的 **DELETE** 命令来删除数据库表中的某条记录。

删除表中的记录

我们希望删除 Northwind 数据库的 Customers 表中的一条记录。首先我们需要创建一个表格，来列出 Customers 中的所有记录。

```
<html>
<body>
<%
```

```
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
set rs=Server.CreateObject("ADODB.Recordset")
rs.open "SELECT * FROM customers",conn
%>

<h2>List Database</h2>
<table border="1" width="100%">
<tr>
<%
for each x in rs.Fields
  response.write("<th>" & ucase(x.name) & "</th>")
next
%>
</tr>
<% do until rs.EOF %>
<tr>
<form method="post" action="demo_delete.asp">
<%
for each x in rs.Fields
  if x.name="customerID" then%>
    <td>
    <input type="submit" name="customerID" value="<%=x.value%>">
    </td>
  <%else%>
    <td><%Response.Write(x.value)%></td>
  <%end if
next
%>
</form>
<%rs.MoveNext%>
</tr>
<%
loop
conn.close
%>
</table>

</body>
</html>
```

假如用户点击 "customerID" 列中的按钮，会打开新文件 "demo_delete.asp"。此文件包含了创建输入域的源代码，这些输入域基于数据库中记录的字段，同时也含有一个删除当前记录的“删除按钮”：

```
<html>
<body>

<h2>Delete Record</h2>
<%
```

```
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

cid=Request.Form("customerID"

if Request.form("companyname")="" then
  set rs=Server.CreateObject("ADODB.Recordset")
  rs.open "SELECT * FROM customers WHERE customerID='" & cid & "'",conn
  %>
  <form method="post" action="demo_delete.asp">
  <table>
  <%for each x in rs.Fields%>
  <tr>
  <td><%=x.name%></td>
  <td><input name="<%=x.name%>" value="<%=x.value%>"></td>
  <%next%>
  </tr>
  </table>
  <br /><br />
  <input type="submit" value="Delete record">
  </form>
<%
else
  sql="DELETE FROM customers"
  sql=sql & " WHERE customerID='" & cid & "'"
  on error resume next
  conn.Execute sql
  if err<>0 then
    response.write("No update permissions!")
  else
    response.write("Record " & cid & " was deleted!")
  end if
end if
conn.close
%>

</body>
</html>
```

ADO 通过 GetString() 加速脚本

请使用 **GetString()** 方法来加速您的 **ASP** 脚本（来代替多行的 **Response.Write**）。

实例

使用 *GetString()*

如何使用 GetString() 在 HTML 表格中显示记录集中的数据。

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open(Server.Mappath("/db/northwind.mdb"))
set rs = Server.CreateObject("ADODB.recordset")
rs.Open "SELECT Companyname, Contactname FROM Customers", conn
str=rs.GetString(,,"</td><td>","</td></tr><tr><td>"," ")
%>

<table border="1" width="100%">
   <tr>
     <td><%Response.Write(str)%></td>
   </tr>
</table>

<%
rs.close
conn.close
set rs = Nothing
set conn = Nothing
%>

</body>
</html>
```

多行 **Response.Write**

下面的例子演示了在 HTML 表格中显示数据库查询的一种方法：

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

set rs = Server.CreateObject("ADODB.recordset")
rs.Open "SELECT Companyname, Contactname FROM Customers", conn
%>

<table border="1" width="100%">
<%do until rs.EOF%>
   <tr>
      <td><%Response.Write(rs.fields("Companyname"))%></td>
      <td><%Response.Write(rs.fields("Contactname"))%></td>
```

```
   </tr>
<%rs.MoveNext
loop%>
</table>

<%
rs.close
conn.close
set rs = Nothing
set conn = Nothing
%>

</body>
</html>
```

对于一个大型的查询来说，这样做会增加脚本的处理时间，这是由于服务器需要处理大量的 Response.Write 命令。

解决的办法是创建全部字符串，从 <table> 到 </table>，然后将其输出 - 只使用一次 Response.Write。

GetString() 方法

GetString() 方法使我们有能力仅使用一次 Response.Write，就可以显示所有的字符串。同时它甚至不需要 do..loop 代码以及条件测试来检查记录集是否处于 EOF。

语法

```
str = rs.GetString(format,rows,coldel,rowdel,nullexpr)
```

如需使用来自记录集的数据创建一个 HTML 表格，我们仅仅需要使用以上参数中的三个（所有的参数都是可选的）：

- coldel - 用作列分隔符的 HTML
- rowdel - 用作行分隔符的 HTML
- nullexpr - 当列为空时所使用的 HTML

注释：GetString() 方法是 ADO 2.0 的特性。您可从下面的地址下载 ADO 2.0：http://www.microsoft.com/data/download.htm

在下面的例子中，我们将使用 GetString() 方法，把记录集存为一个字符串：

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
```

```
set rs = Server.CreateObject("ADODB.recordset")
rs.Open "SELECT Companyname, Contactname FROM Customers", conn

str=rs.GetString(,,"</td><td>","</td></tr><tr><td>"," ")
%>

<table border="1" width="100%">
  <tr>
    <td><%Response.Write(str)%></td>
  </tr>
</table>

<%
rs.close
conn.close
set rs = Nothing
set conn = Nothing
%>
</body>
</html>
```

上面的变量 str 包含着由 SELECT 语句返回的所有列和行的一个字符串。在每列之间会出现 </td><td>，在每行之间会出现 </td></tr><tr><td>。这样，仅使用一次 Response.Write，我们就得到了需要的 HTML。

ADO Command 对象

Command 对象

ADO Command 对象用于执行面向数据库的一次简单查询。此查询可执行诸如创建、添加、取回、删除或更新记录等动作。

如果该查询用于取回数据，此数据将以一个 RecordSet 对象返回。这意味着被取回的数据能够被 RecordSet 对象的属性、集合、方法或事件进行操作。

Command 对象的主要特性是有能力使用存储查询和带有参数的存储过程。

ProgID

```
set objCommand=Server.CreateObject("ADODB.command")
```

属性

属性	描述
ActiveConnection	设置或返回包含了定义连接或 Connection 对象的字符串。
	设置或返回包含提供者（provider）命令（如 SOL 语句、表格

CommandText	名称或存储的过程调用）的字符串值。默认值为 ""（零长度字符串）。
CommandTimeout	设置或返回长整型值，该值指示等待命令执行的时间（单位为秒）。默认值为 30。
CommandType	设置或返回一个 Command 对象的类型
Name	设置或返回一个 Command 对象的名称
Prepared	指示执行前是否保存命令的编译版本（已经准备好的版本）。
State	返回一个值，此值可描述该 Command 对象处于打开、关闭、连接、执行还是取回数据的状态。

方法

方法	描述
Cancel	取消一个方法的一次执行。
CreateParameter	创建一个新的 Parameter 对象
Execute	执行 CommandText 属性中的查询、SQL 语句或存储过程。

集合

集合	描述
Parameters	包含一个 Command 对象的所有 Parameter 对象。
Properties	包含一个 Command 对象的所有 Property 对象。

ADO Connection 对象

Connection 对象

ADO Connection 对象用于创建一个到达某个数据源的开放连接。通过此连接，您可以对一个数据库进行访问和操作。

如果需要多次访问某个数据库，您应当使用 Connection 对象来建立一个连接。您也可以经由一个 Command 或 Recordset 对象传递一个连接字符串来创建某个连接。不过，此类连接仅仅适合一次具体的简单的查询。

ProgID

```
set objConnection=Server.CreateObject("ADODB.connection")
```

属性

属性	描述
Attributes	设置或返回 Connection 对象的属性。
CommandTimeout	指示在终止尝试和产生错误之前执行命令期间需等待的时间。
ConnectionString	设置或返回用于建立连接数据源的细节信息。
ConnectionTimeout	指示在终止尝试和产生错误前建立连接期间所等待的时间。
CursorLocation	设置或返回游标服务的位置。
DefaultDatabase	指示 Connection 对象的默认数据库。
IsolationLevel	指示 Connection 对象的隔离级别。
Mode	设置或返回 provider 的访问权限。
Provider	设置或返回 Connection 对象提供者的名称。
State	返回一个描述连接是打开还是关闭的值。
Version	返回 ADO 的版本号。

方法

方法	描述
BeginTrans	开始一个新事务。
Cancel	取消一次执行。
Close	关闭一个连接。
CommitTrans	保存任何更改并结束当前事务。
Execute	执行查询、SQL 语句、存储过程或 provider 具体文本。
Open	打开一个连接。
OpenSchema	从 provider 返回有关数据源的 schema 信息。
RollbackTrans	取消当前事务中所作的任何更改并结束事务。

事件

注释：您无法使用 VBScript or JScript 来处理事件（仅能使用 Visual Basic、Visual C++ 以及 Visual J++ 语言处理事件）。

事件	描述

BeginTransComplete	在 BeginTrans 操作之后被触发。
CommitTransComplete	在 CommitTrans 操作之后被触发。
ConnectComplete	在一个连接开始后被触发。
Disconnect	在一个连接结束之后被触发。
ExecuteComplete	在一条命令执行完毕后被触发。
InfoMessage	假如在一个 ConnectionEvent 操作过程中警告发生，则触发该事件。
RollbackTransComplete	在 RollbackTrans 操作之后被触发。
WillConnect	在一个连接开始之前被触发。
WillExecute	在一条命令被执行之前被触发。

集合

集合	描述
Errors	包含 Connection 对象的所有 Error 对象。
Properties	包含 Connection 对象的所有 Property 对象。

ADO Error 对象

Error 对象

ADO Error 对象包含与单个操作（涉及提供者）有关的数据访问错误的详细信息。

ADO 会因每次错误产生一个 Error 对象。每个 Error 对象包含具体错误的详细信息，且 Error 对象被存储在 Errors 集合中。要访问这些错误，就必须引用某个具体的连接。

循环遍历 Errors 集合：

```
<%
for each objErr in objConn.Errors
  response.write("<p>")
  response.write("Description: ")
  response.write(objErr.Description & "<br />")
  response.write("Help context: ")
  response.write(objErr.HelpContext & "<br />")
  response.write("Help file: ")
  response.write(objErr.HelpFile & "<br />")
  response.write("Native error: ")
  response.write(objErr.NativeError & "<br />")
  response.write("Error number: ")
```

```
  response.write(objErr.Number & "<br />")
  response.write("Error source: ")
  response.write(objErr.Source & "<br />")
  response.write("SQL state: ")
  response.write(objErr.SQLState & "<br />")
  response.write("</p>")
next
%>
```

语法

```
objErr.property
```

属性

属性	描述
Description	返回一个错误描述。
HelpContext	返回 Microsoft Windows help system 中某个主题的内容 ID。
HelpFile	返回 Microsoft Windows help system 中帮助文件的完整路径。
NativeError	返回来自 provider 或数据源的错误代码。
Number	返回可标识错误的一个唯一的数字。
Source	返回产生错误的对象或应用程序的名称。
SQLState	返回一个 5 字符的 SQL 错误码。

ADO Field 对象

Field 对象

ADO Field 对象包含有关 Recordset 对象中某一列的信息。Recordset 中的每一列对应一个 Field 对象。

ProgID

```
set objField=Server.CreateObject("ADODB.field")
```

属性

属性	描述
ActualSize	返回一个字段值的实际长度。

Attributes	设置或返回 Field 对象的属性。
DefinedSize	返回Field 对象被定义的大小
Name	设置或返回 Field 对象的名称。
NumericScale	设置或返回 Field 对象中的值所允许的小数位数。
OriginalValue	返回某个字段的原始值。
Precision	设置或返回当表示 Field 对象中的数值时所允许的数字的最大数。
Status	返回 Field 对象的状态。
Type	设置或返回 Field 对象的类型。
UnderlyingValue	返回一个字段的当前值。
Value	设置或返回 Field 对象的值。

方法

方法	描述
AppendChunk	把大型的二进制或文本数据追加到 Field 对象
GetChunk	返回大型二进制或文本 Field 对象的全部或部分内容。

集合

集合	描述
Properties	包含一个 Field 对象的所有 Property 对象。

ADO Parameter 对象

Parameter 对象

ADO Parameter 对象可提供有关被用于存储过程或查询中的一个单个参数的信息。

Parameter 对象在其被创建时被添加到 Parameters 集合。Parameters 集合与一个具体的 Command 对象相关联，Command 对象使用此集合在存储过程和查询内外传递参数。

参数被用来创建参数化的命令。这些命令（在它们已被定义和存储之后）使用参数在命令执行前来改变命令的某些细节。例如，SQL SELECT 语句可使用参数定义 WHERE 子句的匹配条件，而使用另一个参数来定义 SORT BY 子句的列的名称。

有四种类型的参数：input 参数、output 参数、input/output 参数 以及 return 参数。

语法

```
objectname.property
objectname.method
```

属性

属性	描述
Attributes	设置或返回一个 Parameter 对象的属性。
Direction	设置或返回某个参数如何传递到存储过程或从存储过程传递回来。
Name	设置或返回一个 Parameter 对象的名称。
NumericScale	设置或返回一个 Parameter 对象的数值的小数点右侧的数字数目。
Precision	设置或返回当表示一个参数中数值时所允许数字的最大数目。
Size	设置或返回 Parameter 对象中的值的最大大小（按字节或字符）。
Type	设置或返回一个 Parameter 对象的类型。
Value	设置或返回一个 Parameter 对象的值。

方法

方法	描述
AppendChunk	把长二进制或字符数据追加到一个 Parameter 对象。
Delete	从 Parameters 集合中删除一个对象。

ADO Property 对象

Property 对象

ADO 对象有两种类型的属性：内置属性和动态属性。

内置属性是在 ADO 中实现并立即可用于任何新对象的属性，此时使用 MyObject.Property 语法。它们不会作为 Property 对象出现在对象的 Properties 集合中，因此，虽然可以更改它们的值，但无法更改它们的特性。

ADO Property 对象表示 ADO 对象的动态特性，这种动态特性是被 provider 定义的。

每个与 ADO 对话的 provider 拥有不同的方式与 ADO 进行交互。所以，ADO 需要通过某种方式来存储有关 provider 的信息。解决方法是 provider 为 ADO 提供具体的信息（动态属性）。ADO 把每个 provider 属性存储在一个 Property 对象中，而 Property 对象相应地也被存储在 Properties 集合中。此

集合会被分配到 Command 对象、Connection 对象、Field 对象 或者 Recordset 对象。

例如，指定给提供者的属性可能会指示 Recordset 对象是否支持事务或更新。这些附加的属性将作为 Property 对象出现在该 Recordset 对象的 Properties 集合中。

ProgID

```
set objProperty=Server.CreateObject("ADODB.property")
```

属性

属性	描述
Attributes	返回一个 Property 对象的属性
Name	设置或返回一个 Property 对象的名称
Type	返回 Property 的类型
Value	设置或返回 一个 Property 对象的值

ADO Record 对象

Record 对象 (ADO version 2.5)

ADO Record 对象用于容纳记录集中的一行、或文件系统的一个文件或一个目录。

ADO 2.5 之前的版本仅能够访问结构化的数据库。在一个结构化的数据库中，每个表在每一行均有确切相同的列数，并且每一列都由相同的数据类型组成。

Record 对象允许访问行与行之间的列数且/或数据类型不同的数据集。

语法

```
objectname.property
objectname.method
```

属性

属性	描述
ActiveConnection	设置或返回 Record 对象当前所属的 Connection 对象。
Mode	设置或返回在Record 对象中修改数据的有效权限。
ParentURL	返回父 Record 的绝对URL。
RecordType	返回 Record 对象的类型。

Source	设置或返回 Record 对象的 Open 方法的 src 参数。
State	返回 Record 对象的状态。

方法

方法	描述
Cancel	取消一次 CopyRecord、DeleteRecord、MoveRecord 或 Open 调用的执行。
Close	关闭一个 Record 对象。
CopyRecord	把文件或目录拷贝到另外一个位置。
DeleteRecord	删除一个文件或目录。
GetChildren	返回一个 Recordset 对象，其中的每一行表示目录中的文件或子目录。
MoveRecord	把文件或目录移动到另外一个位置。
Open	打开一个已有的 Record 对象或创建一个新的文件或目录。

集合

集合	描述
Properties	特定提供者属性的一个集合。
Fields	包含 Record 对象中的所有 Field 对象。

Fields 集合的属性

属性	描述
Count	返回 fields 集合中的项目数。起始值为 0。 例子： `countfields = rec.Fields.Count`
Item(named_item/number)	返回 fields 集合中的某个指定的项目。 例子： `itemfields = rec.Fields.Item(1)`

或者

```
itemfields = rec.Fields.Item("Name")
```

ADO Recordset 对象

实例

GetRows

本例演示如何使用 GetRows 方法。

Recordset 对象

ADO Recordset 对象用于容纳一个来自数据库表的记录集。一个 Recordset 对象由记录和列（字段）组成。

在 ADO 中，此对象是最重要且最常用于对数据库的数据进行操作的对象。

ProgID

```
set objRecordset=Server.CreateObject("ADODB.recordset")
```

当您首次打开一个 Recordset 时，当前记录指针将指向第一个记录，同时 BOF 和 EOF 属性为 False。如果没有记录，BOF 和 EOF 属性为 True。

Recordset 对象能够支持两种更新类型：

> 立即更新 - 一旦调用 Update 方法，所有更改被立即写入数据库。 批更新 - provider 将缓存多个更改，然后使用 UpdateBatch 方法把这些更改传送到数据库。

在 ADO，定义了 4 中不同的游标（指针）类型：

- 动态游标 - 允许您查看其他用户所作的添加、更改和删除
- 键集游标 - 类似动态游标，不同的是您无法查看有其他用户所做的添加，并且它会防止您访问其他用户已删除的记录。其他用户所做的数据更改仍然是可见的。
- 静态游标 - 提供记录集的静态副本，可用来查找数据或生成报告。此外，由其他用户所做的添加、更改和删除将是不可见的。当您打开一个客户端 Recordset 对象时，这是唯一被允许的游标类型。
- 仅向前游标 - 只允许在 Recordset 中向前滚动。此外，由其他用户所做的添加、更改和删除将是不可见的。

可通过 CursorType 属性或 Open 方法中的 CursorType 参数来设置游标的类型。

注释：并非所有的提供者（providers）支持 Recordset 对象的所有方法和属性。

属性

属性	描述
AbsolutePage	设置或返回一个可指定 Recordset 对象中页码的值。
AbsolutePosition	设置或返回一个值，此值可指定 Recordset 对象中当前记录的顺序位置（序号位置）。
ActiveCommand	返回与 Recordset 对象相关联的 Command 对象。
ActiveConnection	如果连接被关闭，设置或返回连接的定义，如果连接打开，设置或返回当前的 Connection 对象。
BOF	如果当前的记录位置在第一条记录之前，则返回 true，否则返回 fasle。
Bookmark	设置或返回一个书签。此书签保存当前记录的位置。
CacheSize	设置或返回能够被缓存的记录的数目。
CursorLocation	设置或返回游标服务的位置。
CursorType	设置或返回一个 Recordset 对象的游标类型。
DataMember	设置或返回要从 DataSource 属性所引用的对象中检索的数据成员的名称。
DataSource	指定一个包含要被表示为 Recordset 对象的数据的对象。
EditMode	返回当前记录的编辑状态。
EOF	如果当前记录的位置在最后的记录之后，则返回 true，否则返回 fasle。
Filter	返回一个针对 Recordset 对象中数据的过滤器。
Index	设置或返回 Recordset 对象的当前索引的名称。
LockType	设置或返回当编辑 Recordset 中的一条记录时，可指定锁定类型的值。
MarshalOptions	设置或返回一个值，此值指定哪些记录被返回服务器。
MaxRecords	设置或返回从一个查询返回 Recordset 对象的的最大记录数目。
PageCount	返回一个 Recordset 对象中的数据页数。
PageSize	设置或返回 Recordset 对象的一个单一页面上所允许的最大记录数。
RecordCount	返回一个 Recordset 对象中的记录数目。
Sort	设置或返回一个或多个作为 Recordset 排序基准的字段名。

Source	设置一个字符串值，或一个 Command 对象引用，或返回一个字符串值，此值可指示 Recordset 对象的数据源。
State	返回一个值，此值可描述是否 Recordset 对象是打开、关闭、正在连接、正在执行或正在取回数据。
Status	返回有关批更新或其他大量操作的当前记录的状态。
StayInSync	设置或返回当父记录位置改变时对子记录的引用是否改变。

方法

方法	描述
AddNew	创建一条新记录。
Cancel	撤销一次执行。
CancelBatch	撤销一次批更新。
CancelUpdate	撤销对 Recordset 对象的一条记录所做的更改。
Clone	创建一个已有 Recordset 的副本。
Close	关闭一个 Recordset。
CompareBookmarks	比较两个书签。
Delete	删除一条记录或一组记录。
Find	搜索一个 Recordset 中满足指定某个条件的一条记录。
GetRows	把多条记录从一个 Recordset 对象中拷贝到一个二维数组中。
GetString	将 Recordset 作为字符串返回。
Move	在 Recordset 对象中移动记录指针。
MoveFirst	把记录指针移动到第一条记录。
MoveLast	把记录指针移动到最后一条记录。
MoveNext	把记录指针移动到下一条记录。
MovePrevious	把记录指针移动到上一条记录。
NextRecordset	通过执行一系列命令清除当前 Recordset 对象并返回下一个 Recordset。
Open	打开一个数据库元素，此元素可提供对表的记录、查询的结果或保存的 Recordset 的访问。
	通过重新执行对象所基于的查询来更新 Recordset 对象中的

Requery	数据。
Resync	从原始数据库刷新当前 Recordset 中的数据。
Save	把 Recordset 对象保存到 file 或 Stream 对象中。
Seek	搜索 Recordset 的索引以快速定位与指定的值相匹配的行，并使其成为当前行。
Supports	返回一个布尔值，此值可定义 Recordset 对象是否支持特定类型的功能。
Update	保存所有对 Recordset 对象中的一条单一记录所做的更改。
UpdateBatch	把所有 Recordset 中的更改存入数据库。请在批更新模式中使用。

事件

Note: You cannot handle events using VBScript or JScript (only Visual Basic, Visual C++, and Visual J++ languages can handle events).

事件	描述
EndOfRecordset	当试图移动到超过 Recordset 结尾的行时被触发。
FetchComplete	当异步操作中的所有记录均被读取后被触发。
FetchProgress	在异步操作期间被定期地触发，报告已读取多少记录。
FieldChangeComplete	Field 对象的值更改被触发。
MoveComplete	Recordset 中的当前位置更改后被触发。
RecordChangeComplete	一条记录更改之后被触发。
RecordsetChangeComplete	在 Recordset 更改之后被触发。
WillChangeField	在 Field 对象的值更改之前被触发
WillChangeRecord	在一条记录更改之前被触发。
WillChangeRecordset	在 Recordset 更改之前被触发。
WillMove	在 Recordset 中的当前位置更改之前被触发。

集合

集合	描述
Fields	指示在此 Recordset 对象中 Field 对象的数目。

Properties	包含所有 Recordset 对象中的 Property 对象。

Fields 集合的属性

属性	描述
Count	返回 fields 集合中项目的数目。以 0 起始。 例子： `countfields = rs.Fields.Count`

Item(named_item/number)	返回 fields 集合中的某个指定的项目。
	例子：
	`itemfields = rs.Fields.Item(1)`
	或者
	`itemfields = rs.Fields.Item("Name")`

Properties 集合的属性

属性	描述
Count	返回 properties 集合中项目的数目。以 0 起始。 例子： `countprop = rs.Properties.Count`

Item(named_item/number)	返回 properties 集合中某个指定的项目。
	例子：
	`itemprop = rs.Properties.Item(1)`
	或者
	`itemprop = rs.Properties.Item("Name")`

ADO Stream 对象

Stream 对象 (ADO version 2.5)

ADO Stream 对象用于读写以及处理二进制数据或文本流。

Stream 对象可通过三种方法获得：

- 通过指向包含二进制或文本数据的对象（通常是文件）的 URL。此对象可以是简单的文档、表示结构化文档的 Record 对象或文件夹。
- 通过将 Stream 对象实例化。这些 Stream 对象可用来存储用于应用程序的数据。跟与 URL 相关联的 Stream 或 Record 的默认 Stream 不同，实例化的 Stream 在默认情况下与基本源没有关联。
- 通过打开与 Record 对象相关联的默认 Stream 对象。打开 Record 时便可获取与 Record 对象相关联的默认流。只需打开该流便可删除一个往返过程。

语法

```
objectname.property
objectname.method
```

属性

属性	描述
CharSet	指定用于存储 Stream 的字符集。
EOS	返回当前位置是否位于流的结尾。
LineSeparator	设置或返回用在文本 Stream 对象中的分行符。
Mode	设置或返回供修改数据的可用权限。
Position	设置或返回从 Stream 对象开始处的当前位置（按字节计算）。
Size	返回一个打开的 Stream 对象的大小。
State	返回一个描述 Stream 是打开还是关闭的值。
Type	设置或返回 Stream 对象中的数据的类型。

方法

方法	描述
Cancel	取消对 Stream 对象的 Open 调用的执行。
Close	关闭一个 Stream 对象。
CopyTo	把指定数目的字符/比特从一个 Stream 对象拷贝到另外一个 Stream 对象。

Flush	把 Stream 缓冲区中的内容发送到相关联的下层对象。
LoadFromFile	把文件的内容载入 Stream 对象。
Open	打开一个 Stream 对象。
Read	从一个二进制 Stream 对象读取全部流或指定的字节数。
ReadText	从一个文本 Stream 对象中读取全部流、一行或指定的字节数。
SaveToFile	把一个 Stream 对象的二进制内容保存到某个文件。
SetEOS	设置当前位置为流的结尾 (EOS)
SkipLine	在读取一个文本流时跳过一行。
Write	把二进制数据写到一个二进制 Stream 对象。
WriteText	把字符数据写到一个文本 Stream 对象。

ADO 数据类型

下面的表格列出了 Access、SQL Server 与 Oracle 之间的数据类型映射：

DataType Enum	Value	Access	SQLServer	Oracle
adBigInt	20		BigInt (SQL Server 2000 +)	
adBinary	128		Binary TimeStamp	Raw *
adBoolean	11	YesNo	Bit	
adChar	129		Char	Char
adCurrency	6	Currency	Money SmallMoney	
adDate	7	Date	DateTime	
adDBTimeStamp	135	DateTime (Access 97 (ODBC))	DateTime SmallDateTime	Date
adDecimal	14			Decimal *
adDouble	5	Double	Float	Float
adGUID	72	ReplicationID (Access 97 (OLEDB)), (Access 2000 (OLEDB))	UniqueIdentifier (SQL Server 7.0 +)	

adIDispatch	9			
adInteger	3	AutoNumber Integer Long	Identity (SQL Server 6.5) Int	Int *
adLongVarBinary	205	OLEObject	Image	Long Raw * Blob (Oracle 8.1.x)
adLongVarChar	201	Memo (Access 97) Hyperlink (Access 97)	Text	Long * Clob (Oracle 8.1.x)
adLongVarWChar	203	Memo (Access 2000 (OLEDB)) Hyperlink (Access 2000 (OLEDB))	NText (SQL Server 7.0 +)	NClob (Oracle 8.1.x)
adNumeric	131	Decimal (Access 2000 (OLEDB))	Decimal Numeric	Decimal Integer Number SmallInt
adSingle	4	Single	Real	
adSmallInt	2	Integer	SmallInt	
adUnsignedTinyInt	17	Byte	TinyInt	
adVarBinary	204	ReplicationID (Access 97)	VarBinary	
adVarChar	200	Text (Access 97)	VarChar	VarChar
adVariant	12		Sql_Variant (SQL Server 2000 +)	VarChar2
adVarWChar	202	Text (Access 2000 (OLEDB))	NVarChar (SQL Server 7.0 +)	NVarChar2
adWChar	130		NChar (SQL Server 7.0 +)	

* 在 Oracle 8.0.x 中 - decimal 和 int 等于 number 和 number(10)。

ASP 快速参考

来自 **W3School** 的 **ASP** 快速参考。打印出来，放入口袋，以备随时使用。

基础语法

ASP 脚本由 <% 和 %> 包围。这样向浏览器输出内容：

```
<html>
<body>
<% response.write("Hello World!") %>
</body>
</html>
```

ASP 中的默认语言是 VBScript。如需使用其他脚本语言，请在 ASP 页面顶端插入一段语言声明：

```
<%@ language="javascript" %>
<html>
<body>

<%
....
%>
```

表单和用户输入

Request.QueryString 用户收集 method="get" 的表单中的值。从表单通过 GET 发送的信息对所有人都可见（将显示在浏览器的地址栏中），对所发送的数据量也有限制。

Request.Form 用于收集 method="post" 的表单中的值。从表单通过 POST 发送的信息对其他人是不可见，对所发送的数据量没有限制。

ASP Cookies

cookie 常用语识别用户。cookie 是服务器嵌到用户计算机上的小文件。每当相同的计算机通过浏览器请求摸个页面时，也会发送 cookie。

Response.Cookies 命令用于创建 cookie：

```
<%
Response.Cookies("firstname")="Alex"
Response.Cookies("firstname").Expires="May 10,2012"
%>
```

注释：Response.Cookies 命令必须位于 <html> 标签之前！

"Request.Cookies" 命令用于取回 cookie 值：

```
<%
fname=Request.Cookies("firstname")
response.write("Firstname=" & fname)
%>
```

引用文件

通过 #include 指令，在服务器执行前，您能够把一个 ASP 文件的内容插入另一个 ASP 文件中。#include 指令用于创建函数、页头、页脚，或多个页面上重复使用的元素。

语法：

```
<!--#include virtual="somefile.inc"-->
```

或者

```
<!--#include file ="somefile.inc"-->
```

请使用关键词 virtual 来指示以虚拟目录开始的路径。如果名为 "header.inc" 的文件位于名为 /html 的虚拟目录中，那么下面的代码会插入 "header.inc" 的内容：

```
<!-- #include virtual ="/html/header.inc" -->
```

请使用关键词 file 来指示相对路径。相对路径以包含该引用文件的目录开头。如果您的文件位于 html 目录中，而文件 "header.inc" 位于 html\headers 中，下面的代码将在您的文件中插入 "header.inc" 的内容：

```
<!-- #include file ="headers\header.inc" -->
```

请使用关键词 file 与语法 (..\) 来引用更高层级目录中的文件。

Global.asa

Global.asa 文件是可选文件，可包含能够由 ASP 应用程序中的每个页面访问的对象声明、变量以及方法。

注释：Global.asa 文件必须存放在 ASP 应用程序的根目录中，而且每个应用程序只能有一个 Global.asa 文件。

Global.asa 文件只能包含以下内容：

- Application 事件
- Session 事件
- <object> 声明
- TypeLibrary 声明
- #include 指令

Application 和 Session 事件

在 Global.asa 中，您可以告诉 application 和 session 对象当 application/session 开始时做什么，当 application/session 结束时做什么。完成该任务的代码位于事件处理程序中。

注释：在 Global.asa 文件中插入代码时，我们并不使用 <% 和 %>，我们需要在 HTML <script> 标签内部放置子程序：

```
<script language="vbscript" runat="server">
sub Application_OnStart
  ' some code
end sub
sub Application_OnEnd
  ' some code
end sub
sub Session_OnStart
  ' some code
end sub
sub Session_OnEnd
  ' some code
end sub
</script>
```

<object> 声明

通过使用 <object> 标签，也可以在 Global.asa 中创建带有 session 或 application 作用域的对象。

注释：<object> 标签应该位于 <script> 标签之外！

语法：

```
<object runat="server" scope="scope" id="id"
{progid="progID"|classid="classID"}>
.......
</object>
```

TypeLibrary 声明

TypeLibrary 是与 COM 对象对应的 DLL 文件的内容容器。通过在 Global.asa 文件中包含对 TypeLibrary 的调用，就能够访问 COM 对象的常量，同时 ASP 代码也能够更好地报告错误。如果您的 Web 应用程序依赖已在类型库中声明了数据类型的 COM 对象，您可以在 Global.asa 中声明该类型库。

语法：

```
<!--
METADATA TYPE="TypeLib"
file="filename"
```

```
uuid="typelibraryuuid"
version="versionnumber"
lcid="localeid"
-->
```

Session 对象

Session 对象用于存储有关用户 session 的信息，或者更改其设置。Session 对象中存储的变量存有关于单个用户的信息，并且能够由一个应用程序中的所有页面进行访问。

集合

- Contents - 包含所有通过脚本命令追加到 session 的条目
- StaticObjects - 包含了所有使用 HTML 的 <object> 标签追加到 session 的对象
- Contents.Remove(*item*/*index*) - 从 Contents 集合删除一个项目
- Contents.RemoveAll() - 从 Contents 集合删除全部项目

属性

- CodePage - 规定显示动态内容时使用的字符集
- LCID - 设置用于显示动态内容的区域标识符
- SessionID - 返回 session id
- Timeout - 设置或返回 session 的超时时间

方法

- Abandon - 撤销 session 对象中的所有对象。

Application 对象

在一起工作以完成某项任务的一组 ASP 文件被称为一个应用程序。ASP 中的 Application 对象用于将这些文件捆绑在一起。所有用户捆绑一个 Application 对象。Application 对象应该存有被应用程序中的许多页面使用的信息（例如数据库连接信息）。

集合

- Contents - 包含所有通过脚本命令追加到应用程序中的项目
- StaticObjects - 包含所有使用 HTML 的 <object> 标签追加到应用程序中的对象
- Contents.Remove - 从 Contents 集合中删除一个项目
- Contents.RemoveAll - 从 Contents 集合中删除所有的项目

方法

- Lock - 防止用户修改 Application 对象中的变量
- Unlock - 允许用户修改 Application 对象中的变量

Response 对象

Response 对象用于从服务器将输出发送给用户。

集合

Cookies(name) - 设置 cookie 的值。假如不存在，就创建 cookie ，然后设置指定的值。

属性

- Buffer - 规定是否缓冲输出。当输出设置缓存时，服务器会阻止向浏览器的响应，直到所有的服务器脚本均被处理，或者直到脚本调用了 Flush 或 End 方法。如果要设置此属性，它应当位于 .asp 文件中的 <html> 标签之前。
- CacheControl - 设置代理服务器是否可以缓存由 ASP 产生的输出。如果设置为 Public，则代理服务器会缓存页面。
- Charset(*charset_name*) - 将字符集的名称追加到 Response 对象中的 content-type 报头。
- ContentType - 设置 Response 对象的 HTTP 内容类型。（比如 "text/html", "image/gif", "image/jpeg", "text/plain"）。默认是 "text/html"
- Expires - 设置页面在失效前的浏览器缓存时间（分钟）
- ExpiresAbsolute - 设置浏览器上页面缓存失效的日期和时间
- IsClientConnected - 指示客户端是否已从服务器断开
- Pics(*pics_label*) - 向 response 报头的 PICS 标志追加值
- Status - 规定由服务器返回的状态行的值

方法

- AddHeader(*name*, *value*) - 向 HTTP 响应添加新的 HTTP 报头和值
- AppendToLog string - 向服务器记录项目（server log entry）的末端添加字符串
- BinaryWrite(*data_to_write*) - 在没有任何字符转换的情况下直接向输出写数据
- Clear - 清除已缓冲的输出。使用该方法来处理错误。如果 Response.Buffer 未设置为 true，该方法将产生 run-time 错误
- End - 停止处理脚本，并返回当前的结果
- Flush - 立即发送已缓存的输出。如果 Response.Buffer 未设置为 true，该方法将产生 run-time 错误
- Redirect(*url*) - 把用户重定向到另一个 URL
- Write(*data_to_write*) - 向用户写文本

Request 对象

当浏览器从服务器请求页面时，就被称为 request。request 对象用于获取来自用户的信息。

集合

- ClientCertificate - 包含了在客户证书中存储的字段值
- Cookies(*name*) - 包含 cookie 值
- Form(*element_name*) - 包含表单值。该表单必须使用 post 方法
- QueryString(*variable_name*) - 包含查询字符串中的变量值
- ServerVariables(*server_variable*) - 包含服务器变量值

属性

- TotalBytes - 返回在请求正文中客户端所发送的字节总数

方法

- BinaryRead - 取回作为 post 请求的一部分而从客户端送往服务器的数据

Server 对象

Server 对象用于访问服务器上的属性和方法。

属性

ScriptTimeout - 设置或返回一段脚本在终止前所能运行多长时间。

方法

- CreateObject(*type_of_object*) - 创建对象的实例
- Execute(*path*) - 从 ASP 文件内部执行另一个 ASP 文件。在被调用的 ASP 文件执行完毕后，控制权返回原先的 ASP 文件
- GetLastError() - 返回描述所发生错误的 ASPError 对象
- HTMLEncode(*string*) - 对字符串应用 HTML 编码
- MapPath(*path*) - 把相对或虚拟路径映射为物理路径
- Transfer(*path*) - 把所有状态信息发送到另一个文件，以备处理。在传送之后，程序的控制权不会返回原先的 ASP 文件
- URLEncode(*string*) - 对字符串应用 URL 编码规则

来源：http://www.w3school.com.cn/asp/asp_quickref.asp

VBScript Date/Time 函数

函数	描述
CDate	把一个有效的日期或时间表达式转换为日期类型。
Date	返回当前的系统日期。
DateAdd	返回已添加指定时间间隔的日期。
DateDiff	返回两个日期之间的时间间隔数。
DatePart	返回给定日期的指定部分。
DateSerial	返回日期的指定年、月、日
DateValue	返回日期
Day	返回代表一月中一天的数字 （介于并包括1至31之间）

FormatDateTime	返回以日期或时间格式化的表达式。
Hour	返回可代表一天中的小时的数字 （介于并包括0至23之间）
IsDate	返回可指示计算表达式能否转换为日期的布尔值。
Minute	返回一个数字，代表小时的分钟 （介于并包括0至59）
Month	返回一个数字，代表年的月份 （介于并包括1至12之间）。
MonthName	返回指定月份的名称。
Now	返回当前的系统日期和时间。
Second	返回一个数字，代表分钟的秒 （介于并包括0至59之间）
Time	返回当前的系统时间。
Timer	返回自 12:00 AM 以来的秒数。
TimeSerial	返回特定小时、分钟和秒的时间。
TimeValue	返回时间。
Weekday	返回一个数字，代表星期的一天（介于并包括1至7）
WeekdayName	返回星期中指定的一天的星期名。
Year	返回一个代表年份的数字。

VBScript CDate 函数

定义和用法

CDate 函数可把一个合法的日期和时间表达式转换为 Date 类型，并返回结果。

提示：请使用 IsDate 函数来判断 date 是否可被转换为日期或时间。

注释：IsDate 函数使用本地设置来检测字符串是否可被转换为日期。

语法

```
CDate(date)
```

参数	描述
date	必需的。任何有效的日期表达式。（比如 Date() 或者 Now()）

实例

例子 **1**

```
d="April 22, 2001"
if IsDate(d) then
  document.write(CDate(d))
end if
```

输出：

```
2/22/01
```

例子 **2**

```
d=#2/22/01#
if IsDate(d) then
  document.write(CDate(d))
end if
```

输出：

```
2/22/01
```

例子 **3**

```
d="3:18:40 AM"
if IsDate(d) then
  document.write(CDate(d))
end if
```

输出：

```
3:18:40 AM
```

VBScript Date 函数

定义和用法

Date 函数可返回当前的系统日期。

语法

```
Date
```

提示和注释

重要事项：

如果同时读取 Date、Time 以及 Now，那么 Now = Date + Time，但是实际上，我们不可能同时调用这三个函数，因为执行完一个函数之后，才能执行另一个函数，所以如果您在程序中必需同时取得当时的日期和时间，必需调用 Now，再利用 DateVale 及 TimeValue 分别取出日期和时间。

实例：取得某一时间点的日期和时间：

```
N = Now '这个时间点的日期和时间
D = Datevalue(N) '同一时间点的日期部分
T = TimeValue(N) '同一时间点的时间部分
D2 = Date '时间点1的日期
T2 = Time '时间点2的时间
```

问题思考

连续执行 Response.write Now 及 Response.Write Date + Time，则可能出现的最大误差值有多大？假设：

```
时间点1取得的     Now = #7/1/95 23:59:59#
时间点2取得的     Date = #7/1/95#
```

而如果“时间点3”刚好跨过一日，所以 Time = #0:00:00，于是 Now 与 Date+Time 的差距便成了 23:59:59。

实例

例子 **1**

```
document.write("The current system date is: ")
document.write(Date)
```

输出：

```
The current system date is: 1/14/2002
```

TIY

Date

如何使用 Date 函数来显示当前日期。

VBScript DateAdd 函数

定义和用法

DateAdd 函数可返回已添加指定时间间隔的日期。

语法

```
DateAdd(interval,number,date)
```

参数	描述
interval	必需的。需要增加的时间间隔。 可采用下面的值： • yyyy - 年 • q - 季度 • m - 月 • y - 当年的第几天 • d - 日 • w - 当周的第几天 • ww - 周 • h - 小时 • n - 分钟 • s - 秒
number	必需的。需要添加的时间间隔的数目。可对未来的日期使用正值，对过去的日期使用负值。
date	必需的。代表被添加的时间间隔的日期的变量或文字。

实例

例子 **1**

给 January 31, 2000 增加一个月：

```
document.write(DateAdd("m",1,"31-Jan-00"))
```

输出：

```
2/29/2000
```

例子 **1**

给 January 31, 2001 增加一个月：

```
document.write(DateAdd("m",1,"31-Jan-01"))
```

输出：

```
2/28/2001
```

例子 **1**

从 January 31, 2001 减去一个月：

```
document.write(DateAdd("m",-1,"31-Jan-01"))
```

输出：

```
12/31/2000
```

TIY

DateAdd

如何使用 DateAdd 函数为日期增加一个月。

DateAdd

如何使用 DateAdd 函数从日期减去一个月。

VBScript DateDiff 函数

定义和用法

DateDiff 函数可返回两个日期之间的时间间隔数。

DateDiff 函数用于计算两日期时间的差值，计算方法是 date2 - date1。

若比较年份，则不管月份以下的数值，若比较月份，则不管天数以下的数值..... 以此类推。

注释：firstdayofweek 参数会对使用“w”和“ww”间隔符号的计算产生影响。

语法

```
DateDiff(interval,date1,date2[,firstdayofweek[,firstweekofyear]])
```

参数	描述
interval	必需的。计算 date1 和 date2 之间的时间间隔的单位。 可采用下面的值： • yyyy - 年 • q - 季度 • m - 月 • y - 当年的第几天 • d - 日

	• w - 当周的第几天 • ww - 周 • h - 小时 • n - 分钟 • s - 秒
date1,date2	必需的。日期表达式。在计算中需要使用的两个日期。
firstdayofweek	可选的。规定一周的日数，即当周的第几天。 可采用下面的值： • 0 = vbUseSystemDayOfWeek - 使用区域语言支持 (NLS) API 设置。 • 1 = vbSunday - 星期日 (默认) • 2 = vbMonday - 星期一 • 3 = vbTuesday - 星期二 • 4 = vbWednesday - 星期三 • 5 = vbThursday - 星期四 • 6 = vbFriday - 星期五 • 7 = vbSaturday - 星期六
firstweekofyear	可选的。规定一年中的第一周。 可采用下面的值： • 0 = vbUseSystem - 使用区域语言支持 (NLS) API 设置。 • 1 = vbFirstJan1 - 由 1 月 1 日所在的星期开始（默认）。 • 2 = vbFirstFourDays - 由在新年中至少有四天的第一周开始。 • 3 = vbFirstFullWeek - 由在新的一年中第一个完整的周开始。

实例

例子 **1**

```
document.write(Date & "<br />")
document.write(DateDiff("m",Date,"12/31/2002") & "<br />")
document.write(DateDiff("d",Date,"12/31/2002") & "<br />")
document.write(DateDiff("n",Date,"12/31/2002"))
```

输出：

```
1/14/2002
11
351
505440
```

例子 **2**

请注意在下面的代码中，date1>date2：

```
document.write(Date & "<br />")
document.write(DateDiff("d","12/31/2002",Date))
```

输出：

```
1/14/2002
-351
```

例子 **3**

```
'How many weeks (start on Monday),
'are left between the current date and 10/10/2002
document.write(Date & "<br />")
document.write(DateDiff("w",Date,"10/10/2002",vbMonday))
```

输出：

```
1/14/2002
38
```

VBScript DatePart 函数

定义和用法

DatePart 函数可返回给定日期的指定部分。

注释：firstdayofweek 参数会对使用“w”和“ww”间隔符号的计算产生影响。

语法

```
DatePart(interval,date[,firstdayofweek[,firstweekofyear]])
```

参数	描述
interval	必需的。计算 date1 和 date2 之间的时间间隔的单位。 可采用下面的值： • yyyy - 年 • q - 季度 • m - 月 • y - 当年的第几天 • d - 日

	• w - 当周的第几天 • ww - 周 • h - 小时 • n - 分钟 • s - 秒
date	必需的。需计算的日期表达式。
firstdayofweek	可选的。规定一周的日数，即当周的第几天。 可采用下面的值： • 0 = vbUseSystemDayOfWeek - 使用区域语言支持 (NLS) API 设置。 • 1 = vbSunday - 星期日 (默认) • 2 = vbMonday - 星期一 • 3 = vbTuesday - 星期二 • 4 = vbWednesday - 星期三 • 5 = vbThursday - 星期四 • 6 = vbFriday - 星期五 • 7 = vbSaturday - 星期六
firstweekofyear	可选的。规定一年中的第一周。 可采用下面的值： • 0 = vbUseSystem - 使用区域语言支持 (NLS) API 设置。 • 1 = vbFirstJan1 - 由 1 月 1 日所在的星期开始（默认）。 • 2 = vbFirstFourDays - 由在新年中至少有四天的第一周开始。 • 3 = vbFirstFullWeek - 由在新的一年中第一个完整的周开始。

实例

例子 **1**

```
d = #2/10/96 16:45:30#
document.write(DatePart("yyyy",d)) '输出：1996
document.write(DatePart("m",d)) '输出：2
document.write(DatePart("d",d)) '输出：10
document.write(DatePart("h",d)) '输出：16
document.write(DatePart("n",d)) '输出：45
document.write(DatePart("s",d)) '输出：30
document.write(DatePart("q",d)) '输出：1，2月是第1季
document.write(DatePart("y",d)) '输出：41，2月10日是1996年的第41日。
document.write(DatePart("ww",d)) '输出：6，2月10日是1996年的第6周。
document.write(DatePart("w",d)) '输出：7，2月10日在在1996年是第6周的第7日（星期六）。
```

VBScript DateSerial 函数

定义和用法

DateSerial 函数可返回指定的年、月、日的子类型 Date 的 Variant 。

也就是说，**DateSerial** 函数可以把年、月、日合并为日期。

语法

```
DateSerial(year,month,day)
```

参数	描述
year	必需的。介于100到9999的数字，或数值表达式。介于 0 到 99 的值被视为 1900–1999。对于所有其他的 year 参数，请使用完整的4位年份。
month	必需的。任何数值表达式。若大于12，则日期从12月起向后推算mouth-12个月，若小于1，则日期从1月起向前推算1-month个月。
day	必需的。任何数值表达式。若大于当月的日数，则日期从当月日数起，向后推算day-当月日数；若小于1，则日期从1日起向前推算1-day日。

实例

例子 **1**

```
document.write(DateSerial(1996,2,3)) '普通的调用方法
```

输出：

```
1996/2/3
```

例子 **2**

```
document.write(DateSerial(95,13,10)) '13月=1年+1月
```

输出：

```
1996/01/10
```

例子 **3**

```
document.write(DateSerial(96,-1,10)) '-1月要从1月起向前推算1-（-1）=2个月
```

输出：

```
1995/11/10
```

例子 **4**

```
document.write(DateSerial(95,2,30)) '95年2月有28日，所以30日=1月+2日
```

输出：

```
1995/03/02
```

例子 **5**

```
document.write(DateSerial(95,2,-2)) '-2日要从1日起向前推算1-（-2）=3日
```

输出：

```
1995/01/29
```

例子

```
document.write(DateSerial(1990-20,9-2,1-1))
'1990-20=1970年，9-2=7月，1-1=0日，0日要从1日起向前推算1-0=1日。
```

输出：

```
1970/6/30
```

VBScript DateValue 函数

定义和用法

DateValue 函数可返回一个日期类型。

也就是说，**DateValue** 函数可从表达式中取回日期。

注释：如果日期中的年份部分不省略，那么函数会使用来自计算机系统日期的当前年份。

注释：如果日期参数含有时间信息，那么时间信息不会被返回。如果日期中含有无效的时间信息，那么会发生 run-time 错误。

注释：如果参数中不含日期，那么 DateValue 函数将返回0 ，若输出则是12:00:00 AM，即0。

语法

```
DateValue(date)
```

参数	描述
date	必需的。一个介于100年1月1日到9999年12月31日的日期，或者任何可表示日期、时间或日期时间兼而有之的表达式。

实例

例子 **1**

```
document.write(DateValue("31-Jan-02"))
```

输出：

```
1/31/2002
```

例子 **2**

```
document.write(DateValue("31-Jan")) '假设当年是2002年，则输出的是系统日期的年份
```

输出：

```
1/31/2002
```

例子 **3**

```
document.write(DateValue("31-Jan-02 2:39:49 AM"))
'如果参数同时包含日期和时间，则只输出日期。
```

输出：

```
1/31/2002
```

例子 **4**

```
document.write(DateValue("2:39:49 AM")) '输出 12:00:00 AM ，相当于0。
```

输出：

```
12:00:00 AM
```

VBScript Day 函数

定义和用法

Day 函数可返回介于 1 到 31 之间的一个代表月的天数的数字。

语法

```
Day(date)
```

参数	描述
date	必需的。可表示一个日期的表达式。

实例

例子 **1**

```
document.write(Date & "<br />")
document.write(Day(Date))
```

输出：

```
1/14/2002
14
```

VBScript FormatDateTime 函数

定义和用法

FormatDateTime 函数可格式化并返回一个合法的日期或时间表达式。

语法

```
FormatDateTime(date,format)
```

参数	描述
date	必需的。任何合法的日期表达式。(比如 Date() 或 Now())
format	可选的。规定所使用的日期/时间格式的格式值。

format 参数：

常数	值	描述
vbGeneralDate	0	显示日期和/或时间。如果有日期部分，则将该部分显示为短日期格式。如果有时间部分，则将该部分显示为长时间格式。如果都存在，则显示所有部分。

vbLongDate	1	使用计算机区域设置中指定的长日期格式显示日期。
vbShortDate	2	使用计算机区域设置中指定的短日期格式显示日期。
vbLongTime	3	使用此格式显示时间：hh:mm:ss PM/AM
vbShortTime	4	使用 24 小时格式 (hh:mm) 显示时间。

实例

例子 **1**

```
D = #2001/2/22#
document.write(FormatDateTime(D))
```

输出：

```
2001-2-22
```

例子 **2**

```
D = #2001/2/22#
document.write(FormatDateTime(D,1))
```

输出：

```
2001年2月22日
```

例子 **3**

```
D = #2001/2/22#
document.write(FormatDateTime(D,2))
```

输出：

```
2001-2-22
```

例子 **4**

```
D = #2001/2/22#
document.write(FormatDateTime(D,3))
```

输出：

```
2001-2-22
```

VBScript Hour 函数

定义和用法

Hour 函数可返回介于 0 到 23 之间的代表天的小时数的数字。

语法

```
Hour(time)
```

参数	描述
time	必需的。任何可表示时间的表达式。

实例

例子 **1**

```
T = #1/15/2002 10:07:47 AM#
document.write(Hour(T))
```

输出：

```
10
```

例子 **2**

```
T = #10:07:47 AM#
document.write(Hour(T))
```

输出：

```
10
```

VBScript IsDate 函数

定义和用法

IsDate 函数可一个布尔值，指示经计算的表达式是否可被转换为日期。如果表达式是日期，或可被转换为日期，则返回 True 。否则，返回 False 。

注释：IsDate 函数使用本地设置来检测字符串是否可以转换为日期。

语法

```
IsDate(expression)
```

参数	描述
expression	必需的。要计算的表达式。

实例

例子 **1**

```
document.write(IsDate("April 22, 1947"))
```

输出：

```
True
```

例子 **2**

```
document.write(IsDate(#11/11/01#))
```

输出：

```
True
```

例子 **3**

```
document.write(IsDate("#11/11/01#"))
```

输出：

```
False
```

例子 **4**

```
document.write(IsDate("Hello World!"))
```

输出：

```
False
```

VBScript Minute 函数

定义和用法

Minute 函数可返回表示小时的分钟数的数字，介于0到59。

语法

```
Minute(time)
```

参数	描述
time	必需的。表示时间的表达式。

实例

例子 **1**

```
D = #1/15/2002 10:34:39 AM#
document.write(Minute(D))
```

输出：

```
34
```

例子 **2**

```
T = #10:34:39 AM#
document.write(Minute(T))
```

输出：

```
34
```

VBScript Month 函数

定义和用法

Month 函数可返回表示年的月份的数字，介于 1 到 12。

语法

```
Month(date)
```

参数	描述
date	必需的。任何可表示日期的表达式。

实例

例子 **1**

```
D = #1/15/2002#
document.write(Month(D))
```

输出：

```
1
```

VBScript MonthName 函数

定义和用法

MonthName 函数可返回指定的月份的名称。

语法

```
MonthName(month[,abbreviate])
```

参数	描述
month	必需的。规定月的数字。（比如一月是1，二月是2，依此类推。）
abbreviate	可选的。一个布尔值，指示是否缩写月份名称。默认是 False。

实例

例子 **1**

```
document.write(MonthName(8))
```

输出：

```
八月
```

例子 **2**

```
document.write(MonthName(8,true))
```

输出：

```
Aug
```

注释：在中文系统中，仍然输出为“八月”。

VBScript Now 函数

定义和用法

Now 函数可根据计算机系统的日期和时间设置返回当前的日期和时间。

语法

```
Now
```

提示和注释

重要事项：

如果同时读取 Date、Time 以及 Now，那么 Now = Date + Time，但是实际上，我们不可能同时调用这三个函数，因为执行完一个函数之后，才能执行另一个函数，所以如果您在程序中必需同时取得当时的日期和时间，必需调用 Now，再利用 DateVale 及 TimeValue 分别取出日期和时间。

实例：取得某一时间点的日期和时间：

```
N = Now '这个时间点的日期和时间
D = Datevalue(N) '同一时间点的日期部分
T = TimeValue(N) '同一时间点的时间部分
D2 = Date '时间点1的日期
T2 = Time '时间点2的时间
```

问题思考

连续执行 Response.write Now 及 Response.Write Date + Time，则可能出现的最大误差值有多大？假设：

```
时间点1取得的     Now = #7/1/95 23:59:59#
时间点2取得的     Date = #7/1/95#
```

而如果“时间点3”刚好跨过一日，所以 Time = #0:00:00，于是 Now 与 Date+Time 的差距便成了 23:59:59。

实例

例子 **1**

```
document.write(Now)
```

输出：

```
2007-10-1 14:10:06
```

注释：输出的结果可能由于不同的计算机设置而略有差异。

VBScript Second 函数

定义和用法

Second 函数可返回表示分钟的秒数的数字，介于 0 到59 之间。

语法

```
Second(time)
```

参数	描述
time	必需的。任何可表示时间的表达式。

实例

例子 **1**

```
D = #1/15/2002 10:55:51 AM#
document.write(Second(D))
```

输出：

```
51
```

例子 **2**

```
T = #10:55:51 AM#
document.write(Second(T))
```

输出：

```
51
```

例子 **3**

下面这个例子是一个可输出中文格式时间的 ASP 函数，其中使用到了 VBScript 的 Hour、Minute 以及 Second 函数。

```
<%
Sub HHMMSS(T)
H = Hour(T)
If H<12 then
  Response.Write "上午"&H&"时"
Else
  Response.Write "下午"&(H-12)&"时"
End If
Response.Write Minute(T)&"分"
Response.Write Second(T)&"秒"
End Sub
%>
```

VBScript Time 函数

定义和用法

Time 函数可返回当前的系统时间。

语法

```
Time
```

提示和注释

重要事项：

如果同时读取 Date、Time 以及 Now，那么 Now = Date + Time，但是实际上，我们不可能同时调用这三个函数，因为执行完一个函数之后，才能执行另一个函数，所以如果您在程序中必需同时取得当时的日期和时间，必需调用 Now，再利用 DateVale 及 TimeValue 分别取出日期和时间。

实例：取得某一时间点的日期和时间：

```
N = Now '这个时间点的日期和时间
D = Datevalue(N) '同一时间点的日期部分
T = TimeValue(N) '同一时间点的时间部分
D2 = Date '时间点1的日期
T2 = Time '时间点2的时间
```

问题思考

连续执行 Response.write Now 及 Response.Write Date + Time，则可能出现的最大误差值有多大？假设：

```
时间点1取得的     Now = #7/1/95 23:59:59#
时间点2取得的     Date = #7/1/95#
```

而如果“时间点3”刚好跨过一日，所以 Time = #0:00:00，于是 Now 与 Date+Time 的差距便成了 23:59:59。

实例

例子 **1**

```
document.write(Time)
```

输出：

```
14:34:38
```

注释：输出的结果可能由于不同的计算机设置而略有差异。

VBScript Timer 函数

定义和用法

Timer 函数可返回午夜 12 时（12:00 AM）以后已经过去的秒数。

语法

```
Timer
```

实例

例子 **1**

```
document.write(Timer)
```

输出：

```
52744.64
```

VBScript TimeSerial 函数

定义和用法

TimeSerial 函数可把时、分、秒合并成为时间。

注释：时分秒若超过应有的范围，其推算的原理与 DateSerial 相同。若经推算后得到的时间小于 #00:00:00#，则自动将负时间变为正时间；若经推算后得到的时间大于等于 #24:00:00#，则时间向前增加，使数据变成一个含有日期时间的数据，其中日期的起算日是 #12/30/1899#。

语法

```
TimeSerial(hour,minute,second)
```

参数	描述
hour	必需的。介于 0-23 的数字，或数值表达式。
minute	必需的。介于 0-59 的数字，或数值表达式。
second	必需的。介于 0-59 的数字，或数值表达式。

要指定一时刻，如 11:59:59，TimeSerial 的参数取值应在可接受的范围内；也就是说，小时应介于 0-23 之间，分和秒应介于 0-59 之间。但是，可以使用数值表达式为每个参数指定相对时间，这一表达式代表某时刻之前或之后的时、分或秒数。

当任何一个参数的取值超出可接受的范围时，它会正确地进位到下一个较大的时间单位中。例如，如果指定了 75 分钟，则这个时间被解释成一小时十五分钟。但是，如果任何一个参数值超出 -32768 到 32767 的范围，就会导致错误。如果使用三个参数直接指定的时间或通过表达式计算出的时间超出可接受的日期范围，也会导致错误。

实例

例子 **1**

```
document.write(TimeSerial(9,30,50)) '正常的调用方法
```

输出：

```
9:30:50 或 9:30:50 AM
```

例子 **2**

```
document.write(TimeSerial(0,9,11)) '正常的调用方法
```

输出：

```
0:09:11 或 12:09:11 AM
```

例子 **3**

```
document.write(TimeSerial(14+2,9-2,1-1)) '根据数值表达式的结果来输出时间
```

输出：

```
16:07:00 或 4:07:00 PM
```

例子 **4**

```
document.write(TimeSerial(26,30,0)) '日期从#12/30/1899#起向后增加1日
```

输出：

```
1899-12-31 2:30:00 AM
```

VBScript TimeValue 函数

定义和用法

TimeValue 函数可返回包含时间的日期子类型的变量。

TimeValue 函数可用来取出参数的时间部分。

注释：若参数不含时间，则 TimeValue 将返回0，若输出结果则是 12:00:00 AM。

语法

```
TimeValue(time)
```

参数	描述
time	必需的。介于 0:00:00 (12:00:00 A.M.) - 23:59:59 (11:59:59 P.M.) 的时间，或任何可表示此范围内时间的表达式。

实例

例子 **1**

```
D = #7/1/96 13:30:00#
document.write(TimeValue(D))
```

输出：

```
13:30:00 或 1:30:00 PM
```

例子 **2**

```
D = "7/1/96 13:30:00"
document.write(TimeValue(D)) '只要能够判断，字符串也能接受。
```

输出：

```
13:30:00 或 1:30:00 PM
```

例子 **3**

```
D = "7/1/96"
document.write(TimeValue(D)) '只有日期
```

输出：

```
12:00:00 AM '相当于0
```

VBScript Weekday 函数

定义和用法

Weekday 函数可返回表示一周的天数的数字，介于 1 和 7 之间。

注释：

语法

```
Weekday(date[,firstdayofweek])
```

参数	描述
date	必需的。需计算的日期表达式。
firstdayofweek	可选的。规定周的第一天。 可采用下面的值： • 0 = vbUseSystemDayOfWeek - 使用区域语言支持 (NLS) API 设置。 • 1 = vbSunday - 星期日（默认） • 2 = vbMonday - 星期一 • 3 = vbTuesday - 星期二 • 4 = vbWednesday - 星期三 • 5 = vbThursday - 星期四 • 6 = vbFriday - 星期五 • 7 = vbSaturday - 星期六

实例

例子 **1**

```
D = #10/1/2007#
document.write(Weekday(D)) '2007年10月1日是星期一，即一周的第二天。
```

输出：

```
2
```

VBScript WeekdayName 函数

定义和用法

WeekdayName 函数可返回一周中指定一天的星期名。

注释：在英文 Windows 环境下，返回值是 "Sunday"、"Monday" 、... ...

语法

```
WeekdayName(weekday[,abbreviate[,firstdayofweek]])
```

参数	描述
weekday	必需的。一周的第几天的数字。
abbreviate	可选的。布尔值，指示是否缩写星期名。
firstdayofweek	可选的。规定周的第一天。 可采用下面的值： • 0 = vbUseSystemDayOfWeek - 使用区域语言支持 (NLS) API 设置。 • 1 = vbSunday - 星期日（默认） • 2 = vbMonday - 星期一 • 3 = vbTuesday - 星期二 • 4 = vbWednesday - 星期三 • 5 = vbThursday - 星期四 • 6 = vbFriday - 星期五 • 7 = vbSaturday - 星期六

实例

例子 **1**

```
document.write(WeekdayName(3))
```

输出：

```
星期二
```

例子 **2**

```
D = #2007/10/1#
document.write(WeekdayName(Weekday(D)))
```

输出：

```
星期一
```

例子 **3**

```
D = #2007/10/1#
document.write(WeekdayName(Weekday(Date),true))
```

输出：

```
星期一 或 Mon
```

VBScript Year 函数

定义和用法

Year 函数可返回表示年份的一个数字。

语法

```
Year(date)
```

参数	描述
date	必需的。能表示日期的任何表达式。

实例

例子 **1**

```
D = #2007/10/1#
document.write(Year(D))
```

输出：

```
2007
```

VBScript Conversion 函数

函数	描述
Asc	把字符串中的首字母转换为 ANSI 字符代码。
CBool	把表达式转换为布尔类型。
CByte	把表达式转换为字节（Byte）类型。
CCur	把表达式转换为货币（Currency）类型。
CDate	把有效的日期和时间表达式转换为日期（Date）类型。
CDbl	把表达式转换为双精度（Double）类型。
Chr	把指定的 ANSI 字符代码转换为字符。
CInt	把表达式转换为整数（Integer）类型。
CLng	把表达式转换为长整形（Long）类型。
CSng	把表达式转换为单精度（Single）类型。
CStr	把表达式转换为子类型 String 的 variant 。
Hex	返回指定数字的十六进制值。
Oct	返回指定数字的八进制值。

VBScript Asc 函数

定义和用法

Asc 函数可把字符串中的第一个字母转换为对应的 ANSI 代码，并返回结果。

语法

```
Asc(string)
```

参数	描述
string	必需的。字符串表达式。不能为空字符串！

实例

例子 **1**

```
document.write(Asc("A"))
```

```
document.write(Asc("F"))
```

输出分别是：

```
65
70
```

例子 **2**

```
document.write(Asc("a"))
document.write(Asc("f"))
```

输出分别是：

```
97
102
```

例子 **3**

```
document.write(Asc("W"))
document.write(Asc("W3School.com.cn"))
```

输出分别是：

```
87
87
```

例子 **4**

```
document.write(Asc("2"))
document.write(Asc("#"))
```

输出分别是：

```
50
35
```

VBScript CBool 函数

定义和用法

CBool 函数可把表达式转换为布尔类型。

语法

```
CBool(expression)
```

参数	描述
expression	必需的。任何合法的表达式。非零的值返回 True，零返回 False。如果表达式不能解释为数值，则将发生 run-time 错误。

实例

例子 **1**

```
dim a,b
a=5
b=10
document.write(CBool(a))
document.write(CBool(b))
```

输出分别是：

```
True
True
```

VBScript CByte 函数

定义和用法

CByte 函数可把表达式转换为字节（Byte）类型。

语法

```
CByte(expression)
```

参数	描述
expression	必需的。任何合法的表达式。

实例

例子 **1**

```
dim a
a=134.345
document.write(CByte(a))
```

输出：

```
134
```

例子 **2**

```
dim a
a=14.345455
document.write(CByte(a))
```

输出：

```
14
```

VBScript CCur 函数

定义和用法

CCur 函数可把表达式转换为货币（Currency）类型。

语法

```
CCur(expression)
```

参数	描述
expression	必需的。任何合法的表达式。

实例

例子 **1**

```
dim a
a=134.345
document.write(CCur(a))
```

输出：

```
134.345
```

例子 **2**

```
dim a
a=1411111111.345455
document.write(CCur(a)) '请注意，此函数会把结果四舍五入为4位的小数。
```

输出：

```
1411111111.3455
```

VBScript CDate 函数

定义和用法

CDate 函数可把一个合法的日期和时间表达式转换为 Date 类型，并返回结果。

提示：请使用 IsDate 函数来判断 date 是否可被转换为日期或时间。

注释：IsDate 函数使用本地设置来检测字符串是否可被转换为日期。

语法

```
CDate(date)
```

参数	描述
date	必需的。任何有效的日期表达式。（比如 Date() 或者 Now()）

实例

例子 **1**

```
d="April 22, 2001"
if IsDate(d) then
  document.write(CDate(d))
end if
```

输出：

```
2/22/01
```

例子 **2**

```
d=#2/22/01#
if IsDate(d) then
  document.write(CDate(d))
end if
```

输出：

```
2/22/01
```

例子 **3**

```
d="3:18:40 AM"
if IsDate(d) then
  document.write(CDate(d))
end if
```

输出：

```
3:18:40 AM
```

VBScript CDbl 函数

定义和用法

CDbl 函数可把表达式转换为双精度（Double）类型。

语法

```
CDbl(expression)
```

参数	描述
expression	必需的。任何合法的表达式。

实例

例子 **1**

```
dim a
a=134.345
document.write(CDbl(a))
```

输出：

```
134.345
```

例子 **2**

```
dim a
a=14111111113353355.345455
document.write(CDbl(a))
```

输出：

```
1.41111111133534E+16
```

VBScript Chr 函数

定义和用法

Chr 函数可把指定的 ANSI 字符代码转换为字符。

注释：从0到31的数字代表不可打印的 ASCII 代码，举例，Chr(10) 会返回一个换行符。

语法

```
Chr(charcode)
```

参数	描述
charcode	必需的。标识某个字符的数字。

实例

例子 **1**

```
document.write(Chr(65))
document.write(Chr(97))
```

输出分别是：

```
A
a
```

例子 **2**

```
document.write(Chr(37))
document.write(Chr(45))
```

输出分别是：

```
%
-
```

例子 **2**

```
document.write(Chr(50))
```

```
document.write(Chr(35))
```

输出分别是：

```
2
#
```

VBScript CInt 函数

定义和用法

CInt 函数可把表达式转换为整数（Integer）类型。

注释：值必须是介于 -32768 与 32767 之间的数字。

注释：CInt 不同于 Fix 和 Int 函数删除数值的小数部分，而是采用四舍五入的方式。 当小数部分正好等于 0.5 时, CInt 总是将其四舍五入成最接近该数的偶数。例如， 0.5 四舍五入为 0, 以及 1.5 四舍五入为 2.

语法

```
CInt(expression)
```

参数	描述
expression	必需的。任何有效的表达式。

实例

例子 **1**

```
dim a
a=134.345
document.write(CInt(a))
```

输出：

```
134
```

例子 **2**

```
dim a
a=-30000.24
document.write(CInt(a))
```

输出：

```
-30000
```

VBScript CLng 函数

定义和用法

CLng 函数可把表达式转换为长整形（Long）类型。

注释：值必须是介于 -2147483648 与 2147483647 之间的数字。

注释：CLng 不同于 Fix 和 Int 函数删除小数部分， 而是采用四舍五入的方式。 当小数部分正好等于 0.5 时， CLng 函数总是将其四舍五入为最接近该数的偶数。如， 0.5 四舍五入为 0, 以及 1.5 四舍五入为 2 。

语法

```
CInt(expression)
```

参数	描述
expression	必需的。任何有效的表达式。

实例

例子 **1**

```
dim a,b
a=23524.45
b=23525.55
document.write(CLng(a))
document.write(CLng(b))
```

输出分别是：

```
23524
23526
```

VBScript CSng 函数

定义和用法

CSng 函数可把表达式转换为单精度（Single）类型。

注释：如果表达式在 Single 子类型允许的范围之外，则发生错误。

语法

```
CSng(expression)
```

参数	描述
expression	必需的。任何有效的表达式。

实例

例子 **1**

```
dim a,b
a=23524.4522
b=23525.5533
document.write(CSng(a) & "<br />")
document.write(CSng(b))
```

输出分别是：

```
23524.45
23525.55
```

VBScript CStr 函数

定义和用法

CStr 函数可把表达式转换为字符串（String）类型。

注释：若表达式的类型不同，那么 CStr 输出的结果也会有所不同。

语法

```
CStr(expression)
```

参数	描述
expression	必需的。任何有效的表达式。

当表达式为不同的值时，**CStr** 返回的结果：

表达式可能的值	**CStr** 相应的返回结果

Boolean	字符串，包含 True 或 False。
Date	字符串，包含系统的短日期格式日期。
Null	会发生 run-time 错误。
Empty	零长度字符串 ("")。
Error	字符串，包含跟随有错误号码的单词 Error。
其他数值	字符串，包含此数字。

实例

例子 **1**

```
dim a
a=false
document.write(CStr(a))
```

输出分别是：

```
false
```

例子 **1**

```
dim a
a=#01/01/01#
document.write(CStr(a))
```

输出分别是：

```
2001-1-1
```

VBScript Hex 函数

定义和用法

Hex 函数可返回一个表示指定数字的十六进制值的字符串。

注释：如果 number 参数不是整数，则在进行运算前将其四舍五入为最接近的整数。

语法

```
Hex(number)
```

参数	描述

expression	必需的。任何有效的表达式。 如果 number 是： • Null - 那么 Hex 函数会返回 Null。 • Empty - 那么 Hex 函数会返回零 (0)。 • 其他数 - 那么 Hex 函数会返回最多 8 个十六进制字符。

实例

例子 **1**

```
document.write(Hex(3))
document.write(Hex(5))
document.write(Hex(9))
document.write(Hex(10))
document.write(Hex(11))
document.write(Hex(12))
document.write(Hex(400))
document.write(Hex(459))
document.write(Hex(460))
```

分别输出：

```
3
5
9
A
B
C
190
1CB
1CC
```

VBScript Oct 函数

定义和用法

Oct 函数可返回表示指定数字八进制值的字符串。

注释：如果 *number* 参数不是整数，则在进行运算前将其四舍五入为最接近的整数。

语法

```
Oct(number)
```

参数	描述
number	必需的。任何有效的表达式。 如果 *number* 是： • Null - 那么 Oct 函数会返回 Null。 • Empty - 那么 Oct 函数会返回零 (0)。 • 其他数 - 那么 Oct 函数会返回最多 11 个十六进制字符。

实例

```
document.write(Oct(3))
document.write(Oct(5))
document.write(Oct(9))
document.write(Oct(10))
document.write(Oct(11))
document.write(Oct(12))
document.write(Oct(400))
document.write(Oct(459))
document.write(Oct(460))
```

分别输出：

```
3
5
11
12
13
14
620
713
714
```

VBScript Format 函数

函数	描述
FormatCurrency	返回作为货币值进行格式化的表达式。
FormatDateTime	返回作为日期或时间进行格式化的表达式。
FormatNumber	返回作为数字进行格式化的表达式。
FormatPercent	返回作为百分数进行格式化的表达式。

VBScript FormatCurrency 函数

定义和用法

FormatCurrency 函数可返回作为货币值被格式化的表达式，使用系统控制面板中定义的货币符号。

语法

```
FormatCurrency(Expression[,NumDigAfterDec[,
IncLeadingDig[,UseParForNegNum[,GroupDig]]]])
```

参数	描述
expression	必需的。需被格式化的表达式。
NumDigAfterDec	指示小数点右侧显示位数的数值。默认值为 -1（使用的是计算机的区域设置）。
IncLeadingDig	可选的。指示是否显示小数值的前导零（leading zero）： • -2 = TristateUseDefault - 使用计算机区域设置中的设置。 • -1 = TristateTrue - True • 0 = TristateFalse - False
UseParForNegNum	可选的。指示是否将负值置于括号中。 • -2 = TristateUseDefault - 使用计算机区域设置中的设置。 • -1 = TristateTrue - True • 0 = TristateFalse - False
GroupDig	可选的。指示是否使用计算机区域设置中指定的数字分组符号将数字分组。 • -2 = TristateUseDefault - 使用计算机区域设置中的设置。 • -1 = TristateTrue - True • 0 = TristateFalse - False

实例

例子 **1**

```
document.write(FormatCurrency(20000))
```

输出：

```
￥20,000.00
```

例子 **2**

```
document.write(FormatCurrency(20000.578,2))
```

输出：

```
￥20,000.58
```

例子 **3**

```
document.write(FormatCurrency(20000.578,2,,,0))
```

输出：

```
￥20000.58
```

VBScript FormatDateTime 函数

定义和用法

FormatDateTime 函数可格式化并返回一个合法的日期或时间表达式。

语法

```
FormatDateTime(date,format)
```

参数	描述
date	必需的。任何合法的日期表达式。(比如 Date() 或 Now())
format	可选的。规定所使用的日期/时间格式的格式值。

format 参数：

常数	值	描述
vbGeneralDate	0	显示日期和/或时间。如果有日期部分，则将该部分显示为短日期格式。如果有时间部分，则将该部分显示为长时间格式。如果都存在，则显示所有部分。
vbLongDate	1	使用计算机区域设置中指定的长日期格式显示日期。
vbShortDate	2	使用计算机区域设置中指定的短日期格式显示日期。

vbLongTime	3	使用此格式显示时间：hh:mm:ss PM/AM
vbShortTime	4	使用 24 小时格式 (hh:mm) 显示时间。

实例

例子 **1**

```
D = #2001/2/22#
document.write(FormatDateTime(D))
```

输出：

```
2001-2-22
```

例子 **2**

```
D = #2001/2/22#
document.write(FormatDateTime(D,1))
```

输出：

```
2001年2月22日
```

例子 **3**

```
D = #2001/2/22#
document.write(FormatDateTime(D,2))
```

输出：

```
2001-2-22
```

例子 **4**

```
D = #2001/2/22#
document.write(FormatDateTime(D,3))
```

输出：

```
2001-2-22
```

VBScript FormatNumber 函数

定义和用法

FormatNumber 函数可返回作为数字被格式化的表达式。

语法

```
FormatNumber(Expression[,NumDigAfterDec[,
IncLeadingDig[,UseParForNegNum[,GroupDig]]]])
```

参数	描述
expression	必需的。需被格式化的表达式。
NumDigAfterDec	指示小数点右侧显示位数的数值。默认值为 -1（使用的是计算机的区域设置）。
IncLeadingDig	可选的。指示是否显示小数值的前导零（leading zero）： • -2 = TristateUseDefault - 使用计算机区域设置中的设置。 • -1 = TristateTrue - True • 0 = TristateFalse - False
UseParForNegNum	可选的。指示是否将负值置于括号中。 • -2 = TristateUseDefault - 使用计算机区域设置中的设置。 • -1 = TristateTrue - True • 0 = TristateFalse - False
GroupDig	可选的。指示是否使用计算机区域设置中指定的数字分组符号将数字分组。 • -2 = TristateUseDefault - 使用计算机区域设置中的设置。 • -1 = TristateTrue - True • 0 = TristateFalse - False

实例

例子 **1**

```
document.write(FormatNumber(20000))
```

输出：

```
20,000.00
```

例子 **2**

```
document.write(FormatNumber(20000.578,2))
```

输出：

```
20,000.58
```

例子 **3**

```
document.write(FormatNumber(20000.578,2,,,0))
```

输出：

```
20000.58
```

VBScript FormatPercent 函数

定义和用法

FormatPercent 函数可返回作为百分数被格式化的表达式（尾随有 % 符号的百分比（乘以 100 ））。

语法

```
FormatPercent(Expression[,NumDigAfterDec[,
IncLeadingDig[,UseParForNegNum[,GroupDig]]]])
```

参数	描述
expression	必需的。需被格式化的表达式。
NumDigAfterDec	指示小数点右侧显示位数的数值。默认值为 -1（使用的是计算机的区域设置）。
IncLeadingDig	可选的。指示是否显示小数值的前导零（leading zero）： • -2 = TristateUseDefault - 使用计算机区域设置中的设置。 • -1 = TristateTrue - True • 0 = TristateFalse - False
UseParForNegNum	可选的。指示是否将负值置于括号中。 • -2 = TristateUseDefault - 使用计算机区域设置中的设置。 • -1 = TristateTrue - True • 0 = TristateFalse - False

GroupDig	可选的。指示是否使用计算机区域设置中指定的数字分组符号将数字分组。 • -2 = TristateUseDefault - 使用计算机区域设置中的设置。 • -1 = TristateTrue - True • 0 = TristateFalse - False

实例

例子 **1**

```
' 6 是 345 的百分之几?
document.write(FormatPercent(6/345))
```

输出：

```
1.74%
```

例子 **2**

```
' 6 是 345 的百分之几?
document.write(FormatPercent(6/345,1))
```

输出：

```
1.7%
```

VBScript Math 函数

函数	描述
Abs	返回指定数字的绝对值。
Atn	返回指定数字的反正切。
Cos	返回指定数字（角度）的余弦。
Exp	返回 e（自然对数的底）的幂次方。
Hex	返回指定数字的十六进制值。
Int	返回指定数字的整数部分。
Fix	返回指定数字的整数部分。

Log	返回指定数字的自然对数。
Oct	返回指定数字的余弦值。
Rnd	返回小于1但大于或等于0的一个随机数。
Sgn	返回可指示指定的数字的符号的一个整数。
Sin	返回指定数字（角度）的正弦。
Sqr	返回指定数字的平方根。
Tan	返回指定数字（角度）的正切。

VBScript Abs 函数

定义和用法

Abs 函数可返回指定的数字的绝对值。

注释：如果 number 参数包含Null，则返回 Null 。

注释：如果 number 参数是一个未初始化的变量，则返回 0 。

语法

```
Abs(number)
```

参数	描述
number	必需的。一个数值表达式。

实例

例子 **1**

```
document.write(Abs(1))
document.write(Abs(-1))
```

输出：

```
1
1
```

例子 **2**

```
document.write(Abs(48.4))
```

```
document.write(Abs(-48.4))
```

输出：

```
48.4
48.4
```

VBScript Atn 函数

定义和用法

Atn 函数可返回指定数字的正切。

语法

```
Atn(number)
```

参数	描述
number	必需的。一个数值表达式。

实例

例子 **1**

```
document.write(Atn(89))
```

输出：

```
1.55956084453693
```

例子 **2**

```
document.write(Atn(8.9))
```

输出：

```
1.45890606062322
```

例子 **3**

```
' 计算 pi 的值：
dim pi
pi=4*Atn(1)
```

```
document.write(pi)
```

输出：

```
3.14159265358979
```

VBScript Exp 函数

定义和用法

Exp 函数可e（自然对数的底）的幂次方。

注释：值不能超过 709.782712893 。常数 e 的值约为 2.718282。

提示：请参阅 Log 函数。Exp 函数完成 Log 函数的反运算，并且有时引用为反对数形式。

语法

```
Cos(number)
```

参数	描述
number	必需的。有效的数值表达式。

实例

例子 **1**

```
document.write(Exp(6.7))
```

输出：

```
812.405825167543
```

例子 **2**

```
document.write(Exp(-6.7))
```

输出：

```
1.23091190267348E-03
```

VBScript Int 函数

定义和用法

Int 函数可返回指定数字的整数部分。

注释：若 number 参数包含 Null ， 则返回 Null 。

提示：请参阅 Fix 函数。Int 和 Fix 函数都删除 number 参数的小数部分并返回以整数表示的结果。

Int 和 Fix 函数的区别在于如果 number 参数为负数时，Int 函数返回小于或等于 number 的第一个负整数，而 Fix 函数返回大于或等于 number 参数的第一个负整数。例如，Int 将 -8.4 转换为 -9，而 Fix 函数将 -8.4 转换为 -8。

语法

```
Int(number)
```

参数	描述
number	必需的。有效的数值表达式。

实例

例子 **1**

```
document.write(Int(6.83227))
```

输出：

```
6
```

例子 **2**

```
document.write(Int(6.23443))
```

输出：

```
6
```

例子 **3**

```
document.write(Int(-6.13443))
```

输出：

```
-7
```

例子 **4**

```
document.write(Int(-6.93443))
```

输出：

```
-7
```

VBScript Fix 函数

定义和用法

Fix 函数可返回指定数字的整数部分。

注释：若 number 参数包含 Null ，则返回 Null 。

提示：请参阅 Int 函数。Int 和 Fix 函数都删除 number 参数的小数部分并返回以整数表示的结果。

Int 和 Fix 函数的区别在于如果 number 参数为负数时，Int 函数返回小于或等于 number 的第一个负整数，而 Fix 函数返回大于或等于 number 参数的第一个负整数。例如，Int 将 -8.4 转换为 -9，而 Fix 函数将 -8.4 转换为 -8。

语法

```
Int(number)
```

参数	描述
number	必需的。有效的数值表达式。

实例

例子 **1**

```
document.write(Fix(6.83227))
```

输出：

```
6
```

例子 **2**

```
document.write(Fix(6.23443))
```

输出：

```
6
```

例子 **3**

```
document.write(Fix(-6.13443))
```

输出：

```
-6
```

例子 **4**

```
document.write(Fix(-6.93443))
```

输出：

```
-6
```

VBScript Log 函数

定义和用法

Log 函数可返回指定数据的自然对数。自然对数是以 e 为底的对数。

注释：不允许使用负值。

提示：请参阅 Exp 函数。

语法

```
Log(number)
```

参数	描述
number	必需的。大于 0 合法的数值表达式。

实例

例子 **1**

```
document.write(Log(38.256783227))
```

输出：

```
3.64432088381777
```

VBScript Rnd 函数

定义和用法

Rnd 函数可返回一个随机数。数字总是小于 **1** 但大于或等于 **0** 。

因每一次连续调用 Rnd 函数时都用序列中的前一个数作为下一个数的种子，所以对于任何最初给定的种子都会生成相同的数列。

在调用 Rnd 之前，先使用无参数的 Randomize 语句初始化随机数生成器，该生成器具有基于系统计时器的种子。

要产生指定范围的随机整数，请使用以下公式：

```
Int((upperbound - lowerbound + 1) * Rnd + lowerbound)
```

这里， upperbound 是此范围的上界，而 lowerbound 是此范围内的下界。

注释：要重复随机数的序列，请在使用数值参数调用 Randomize 之前，立即用负值参数调用 Rnd。使用同样 number 值的 Randomize 不能重复先前的随机数序列。

语法

```
Rnd[(number)]
```

参数	描述
number	可选的。合法的数值表达式。 如果数字是： • <0 - Rnd 会每次都返回相同的值。 • >0 - Rnd 会返回序列中的下一个随机数。 • =0 - Rnd 返回最近生成的数。 • 省略 - Rnd 会返回序列中的下一个随机数。

实例

例子 **1**

```
document.write(Rnd)
```

输出：

```
0.7055475
```

例子 **2**

如果您使用例子 1 中的代码，相同的随机数会重复出现。

可以使用 Randomize 语句在页面每次重新载入的时候生成一个新的随机数：

```
Randomize
document.write(Rnd)
```

输出：

```
0.4758112
```

例子 **3**

```
dim max,min
max=100
min=1
document.write(Int((max-min+1)*Rnd+min))
```

输出：

```
71
```

VBScript Sgn 函数

定义和用法

Sgn 函数可返回指示指定数字的符号的整数。

语法

```
Sgn(number)
```

参数	描述
number	必需的。合法的数值表达式。 如果数字是： • >0 - Sgn 会返回 1。 • <0 - Sgn 会返回 -1。 • =0 - Sgn 会返回 0。

实例

例子 **1**

```
document.write(Sgn(15))
```

输出：

```
1
```

例子 **2**

```
document.write(Sgn(-5.67))
```

输出：

```
-1
```

例子 **3**

```
document.write(Sgn(0))
```

输出：

```
0
```

VBScript Sin 函数

定义和用法

Sin 函数可返回指定数字（角度）的正弦。

Sin 函数取某个角并返回直角三角形两边的比值。此比值是直角三角形中该角的对边长度与斜边长度之比。结果的范围在 -1 到 1 之间。

注释：将角度乘以 pi/180 即可转换为弧度，将弧度乘以 180/pi 即可转换为角度。

语法

```
Sgn(number)
```

参数	描述
number	必需的。将某个角表示为弧度的有效表达式。

实例

例子 **1**

```
document.write(Sin(47))
```

输出：

```
0.123573122745224
```

例子 **2**

```
document.write(Sin(-47))
```

输出：

```
-0.123573122745224
```

VBScript Sqr 函数

定义和用法

Sqr 函数可返回一个数的平方根。

注释：number 参数不能是负值。

语法

```
Sqr(number)
```

参数	描述
number	必需的。大于等于 0 的有效数值表达式。

实例

例子 **1**

```
document.write(Sqr(9))
```

输出：

```
3
```

例子 **2**

```
document.write(Sqr(0))
```

输出：

```
0
```

例子 **3**

```
document.write(Sqr(47))
```

输出：

```
6.85565460040104
```

VBScript Tan 函数

定义和用法

Tan 函数可返回指定数字（角度）的正切。

Tan 取某个角并返回直角三角形两个直角边的比值。此比值是直角三角形中该角的对边长度与邻边长度之比。

注释：将角度乘以 pi /180 即可转换为弧度，将弧度乘以 180/pi 即可转换为角度。

语法

```
Sqr(number)
```

参数	描述
number	必需的。将某个角表示为弧度的有效表达式。

实例

例子 **1**

```
document.write(Tan(40))
```

输出：

```
-1.1172149309239
```

VBScript Array 函数

函数	描述
Array	返回一个包含数组的变量
Filter	返回下标从零开始的数组，其中包含基于特定过滤条件的字符串数组的子集。
IsArray	返回一个布尔值，可指示指定的变量是否是数组。
Join	返回一个由数组中若干子字符串组成的字符串。
LBound	返回指定数组维数的最小下标。
Split	返回下标从0开始的一维数组，包含指定数目的子字符串。
UBound	返回指定数组维数的最大下标。

VBScript Array 函数

定义和用法

Array 可返回一个包含数组的变量。

注释：数组中的第一个元素是零。

语法

```
Array(arglist)
```

参数	描述
arglist	必需的。数组中元素值的列表（由逗号分割）。

实例

例子 **1**

```
dim a
a=Array(5,10,15,20)
document.write(a(3))
```

输出：

```
20
```

例子 **2**

```
dim a
a=Array(5,10,15,20)
document.write(a(0))
```

输出：

```
5
```

VBScript Filter 函数

定义和用法

Filter 函数可返回一个基于 0 的数组，此数组包含以特定过滤条件为基础的字符串数组的子集。

注释：如果找不到与 value 参数相匹配的值，Filter 函数会返回一个空数组。

注释：若参数 inputstrings 为 Null 或者不是一维数组，则会发生错误。

语法

```
Filter(inputstrings,value[,include[,compare]])
```

参数	描述
inputstrings	必需的。需检索的一维字符串数组。
value	必需的。要搜索的字符串。
include	可选项。Boolean 值，指定返回的子字符串是否包含 Value。如果 Include 为 True，Filter 将返回包含子字符串 Value 的数组子集。如果 Include 为 False，Filter 将返回不包含子字符串 Value 的数组子集。
compare	可选的。规定所使用的字符串比较类型。

参数 **compare** 的值：

常数	值	描述
vbBinaryCompare	0	执行二进制比较。
vbTextCompare	1	执行文本比较。

实例

例子 **1**

```
dim a(5),b
a(0)="Saturday"
a(1)="Sunday"
a(2)="Monday"
a(3)="Tuesday"
a(4)="Wednesday"
b=Filter(a,"n")
document.write(b(0))
document.write(b(1))
document.write(b(2))
```

输出：

```
Sunday
Monday
Wednesday
```

例子 **2**

```
dim a(5),b
a(0)="Saturday"
a(1)="Sunday"
a(2)="Monday"
a(3)="Tuesday"
a(4)="Wednesday"
b=Filter(a,"n",false)
document.write(b(0))
document.write(b(1))
document.write(b(2))
```

输出：

```
Saturday
Tuesday
```

VBScript IsArray 函数

定义和用法

IsArray 函数可返回一个指示指定的变量是否为数组的布尔值。如果变量为数组，则返回 True，否则返回 False。

语法

```
IsArray(variable)
```

参数	描述
variable	必需的。任何变量。

实例

例子 **1**

```
dim a(5)
a(0)="Saturday"
a(1)="Sunday"
a(2)="Monday"
a(3)="Tuesday"
a(4)="Wednesday"
document.write(IsArray(a))
```

输出：

```
True
```

例子 **2**

```
dim a
a="Saturday"
document.write(IsArray(a))
```

输出：

```
False
```

VBScript Join 函数

定义和用法

Join 函数可返回一个由某个数组中一系列子字符串组成的字符串。

语法

```
Join(list[,delimiter])
```

参数	描述
list	必需的。一维数组，其中包含需被连接的子字符串。
delimiter	可选的。用于在返回的字符串中分割子字符串的字符。默认是空格字符。

实例

例子 **1**

```
dim a(5),b
a(0)="Saturday"
a(1)="Sunday"
a(2)="Monday"
a(3)="Tuesday"
a(4)="Wednesday"
b=Filter(a,"n")
document.write(join(b))
```

输出：

```
Sunday Monday Wednesday
```

例子 **2**

```
dim a(5),b
a(0)="Saturday"
a(1)="Sunday"
a(2)="Monday"
a(3)="Tuesday"
a(4)="Wednesday"
b=Filter(a,"n")
document.write(join(b,", "))
```

输出：

```
Sunday, Monday, Wednesday
```

VBScript LBound 函数

定义和用法

LBound 函数可返回指示数组维数的最小下标。

注释：任何维数的 LBound 总是 0。

提示：LBound 函数与 UBound 函数共同使用，可以确定数组的大小。使用 UBound 函数可以找到数组某一维的上界。

语法

```
LBound(arrayname[,dimension])
```

参数	描述
arrayname	必需的。数组变量的名称。
dimension	可选的。要返回哪一维的下界。 1 = 第一维， 2 = 第二维，以此类推。默认是 1 。

实例

例子 **1**

```
dim a(10)
a(0)="Saturday"
a(1)="Sunday"
a(2)="Monday"
a(3)="Tuesday"
a(4)="Wednesday"
a(5)="Thursday"
document.write(UBound(a))
document.write("<br />")
document.write(LBound(a))
```

输出：

```
10
0
```

VBScript Split 函数

定义和用法

Split 函数可返回基于 0 的一维数组，此数组包含指定数目的子字符串。

语法

```
Split(expression[,delimiter[,count[,compare]]])
```

参数	描述
expression	必需的。包含子字符串和分隔符的字符串表达式。
delimiter	可选的。用于识别子字符串界限的字符。默认是空格字符。
count	可选的。需被返回的子字符串的数目。-1 指示返回所有的子字符串。

compare	可选的。规定要使用的字符串比较类型。 可采用下列的值： • 0 = vbBinaryCompare - 执行二进制比较。 • 1 = vbTextCompare - 执行文本比较。

实例

例子 **1**

```
dim txt,a
txt="Hello World!"
a=Split(txt)
document.write(a(0) & "<br />")
document.write(a(1))
```

输出：

```
Hello
World!
```

VBScript UBound 函数

定义和用法

UBound 函数可返回指示数组维数的最大下标。

提示：LBound 函数与 UBound 函数共同使用，可以确定数组的大小。使用 UBound 函数可以找到数组某一维的上界。

语法

```
UBound(arrayname[,dimension])
```

参数	描述
arrayname	必需的。数组变量的名称。
dimension	可选的。要返回哪一维的上界。 1 = 第一维， 2 = 第二维，以此类推。默认是 1 。

实例

例子 **1**

```
dim a(10)
a(0)="Saturday"
a(1)="Sunday"
a(2)="Monday"
a(3)="Tuesday"
a(4)="Wednesday"
a(5)="Thursday"
document.write(UBound(a))
document.write("<br />")
document.write(LBound(a))
```

输出：

```
10
0
```

VBScript String 函数

函数	描述
InStr	返回字符串在另一字符串中首次出现的位置。检索从字符串的第一个字符开始。
InStrRev	返回字符串在另一字符串中首次出现的位置。检索从字符串的最末字符开始。
LCase	把指定字符串转换为小写。
Left	从字符串的左侧返回指定数目的字符。
Len	返回字符串中的字符数目。
LTrim	删除字符串左侧的空格。
RTrim	删除字符串右侧的空格。
Trim	删除字符串左侧和右侧的空格。
Mid	从字符串返回指定数目的字符。
Replace	使用另外一个字符串替换字符串的指定部分指定的次数。
Right	返回从字符串右侧开始指定数目的字符。
Space	返回由指定数目的空格组成的字符串。
StrComp	比较两个字符串，返回代表比较结果的一个值。
String	返回包含指定长度的重复字符的字符串。
StrReverse	反转字符串。

UCase	把指定的字符串转换为大写。

VBScript InStr 函数

定义和用法

InStr 函数可返回一个字符串在另一个字符串中首次出现的位置。

InStr 函数可返回下面的值：

- 如果 string1 为 ""（零长度） - InStr 返回 0
- 如果 string1 为 Null - InStr 返回 Null
- 如果 string2 为 "" - InStr 返回 start
- 如果 string2 为 Null - InStr 返回 Null
- 如果 string2 没有找到 - InStr 返回 0
- 如果在 string1 中找到 string2，InStr 返回找到匹配字符串的位置。
- 如果 start > Len(string1) - InStr 返回 0

提示：请参阅 InStrRev 函数。

语法

```
InStr([start,]string1,string2[,compare])
```

参数	描述
start	可选的。规定每次搜索的起始位置。默认是搜索起始位置是第一个字符。如果已规定 compare 参数，则必须有此参数。
string1	必需的。需要被搜索的字符串。
string2	必需的。需搜索的字符串。
compare	必需的。规定要使用的字符串比较类型。默认是 0 。可采用下列值： • 0 = vbBinaryCompare - 执行二进制比较。 • 1 = vbTextCompare - 执行文本比较。

实例

例子 **1**

```
dim txt,pos
txt="This is a beautiful day!"
pos=InStr(txt,"his")
document.write(pos)
```

输出：

```
2
```

例子 **2**

```
dim txt,pos
txt="This is a beautiful day!"
'A textual comparison starting at position 4
pos=InStr(4,txt,"is",1)
document.write(pos)
```

输出：

```
6
```

例子 **3**

```
dim txt,pos
txt="This is a beautiful day!"
'A binary comparison starting at position 1
pos=InStr(1,txt,"B",0)
document.write(pos)
```

输出：

```
0
```

VBScript InStrRev 函数

定义和用法

InStrRev 函数可返回一个字符串在另一个字符串中首次出现的位置。搜索从字符串的末端开始，但是返回的位置是从字符串的起点开始计数的。

InStrRev 函数可返回下面的值：

- 如果 string1 为 ""（零长度） - InStr 返回 0
- 如果 string1 为 Null - InStr 返回 Null
- 如果 string2 为 "" - InStr 返回 start
- 如果 string2 为 Null - InStr 返回 Null
- 如果 string2 没有找到 - InStr 返回 0
- 如果在 string1 中找到 string2，InStr 返回找到匹配字符串的位置。
- 如果 start > Len(string1) - InStr 返回 0

提示：请参阅 InStr 函数。

语法

```
InStrRev(string1,string2[,start[,compare]])
```

参数	描述
start	可选的。规定每次搜索的起始位置。默认是搜索起始位置是第一个字符。如果已规定 compare 参数，则必须有此参数。
string1	必需的。需要被搜索的字符串。
string2	必需的。需搜索的字符串。
compare	必需的。规定要使用的字符串比较类型。默认是 0 。可采用下列值： • 0 = vbBinaryCompare - 执行二进制比较。 • 1 = vbTextCompare - 执行文本比较。

实例

例子 **1**

```
dim txt,pos
txt="This is a beautiful day!"
pos=InStrRev(txt,"his")
document.write(pos)
```

输出：

```
2
```

例子 **2**

```
dim txt,pos
txt="This is a beautiful day!"
'textual comparison
pos=InStrRev(txt,"B",-1,1)
document.write(pos)
```

输出：

```
11
```

例子 **3**

```
dim txt,pos
txt="This is a beautiful day!"
'binary comparison
pos=InStrRev(txt,"T")
document.write(pos)
```

输出：

```
1
```

例子 **4**

```
dim txt,pos
txt="This is a beautiful day!"
'binary comparison
pos=InStrRev(txt,"t")
document.write(pos)
```

输出：

```
15
```

VBScript LCase 函数

定义和用法

LCase 函数可把指定的字符串转换为小写。

提示：请参阅 UCase 函数。

语法

```
LCase(string)
```

参数	描述
string	必需的。需要被转换为小写的字符串。

实例

例子 **1**

```
dim txt
txt="THIS IS A BEAUTIFUL DAY!"
document.write(LCase(txt))
```

输出：

```
this is a beautiful day!
```

例子 **2**

```
dim txt
txt="This Is a Beautiful Day!"
document.write(LCase(txt))
```

输出：

```
this is a beautiful day!
```

VBScript Left 函数

定义和用法

Left 函数可从字符串的左侧返回指定数目的字符。

提示：请使用 Len 函数来确定字符串中的字符数目。

提示：请参阅 Right 函数。

语法

```
Left(string,length)
```

参数	描述
string	必需的。从其中返回字符的字符串。
length	必需的。规定需返回多少字符。如果设置为 0，则返回空字符串("")。如果设置为大于或等于字符串的长度，则返回整个字符串。

实例

例子 **1**

```
dim txt
txt="This is a beautiful day!"
document.write(Left(txt,11))
```

输出：

```
This is a b
```

例子 **2**

```
dim txt
txt="This is a beautiful day!"
document.write(Left(txt,100))
```

输出：

```
This is a beautiful day!
```

例子 **3**

```
dim txt,x
txt="This is a beautiful day!"
x=Len(txt)
document.write(Left(txt,x))
```

输出：

```
This is a beautiful day!
```

VBScript Len 函数

定义和用法

Len 函数可返回字符串中字符的数目。

语法

```
Len(string|varname)
```

参数	描述
string	字符串表达式。
varname	变量名称。

实例

例子 **1**

```
dim txt
txt="This is a beautiful day!"
document.write(Len(txt))
```

输出：

```
24
```

例子 **2**

```
document.write(Len("This is a beautiful day!"))
```

输出：

```
24
```

VBScript LTrim 函数

定义和用法

LTrim 函数可删除字符串左侧的空格。

语法

```
Len(string|varname)
```

参数	描述
string	字符串表达式。

实例

例子 **1**

```
dim txt
txt="  This is a beautiful day!  "
document.write(LTrim(txt))
```

输出：

```
"This is a beautiful day!  "
```

VBScript Trim 函数

定义和用法

Trim 函数可删除字符串两侧的空格。

语法

```
Trim(string)
```

参数	描述
string	字符串表达式。

实例

例子 **1**

```
dim txt
txt="  This is a beautiful day!  "
document.write(Trim(txt))
```

输出：

```
"This is a beautiful day!"
```

VBScript Mid 函数

定义和用法

Mid 函数可从字符串中返回指定数目的字符。

提示：请使用 Len 函数来确定字符串中的字符数目。

语法

```
Mid(string,start[,length])
```

参数	描述
string	必需的。从其中返回字符的字符串表达式。如果字符串包含 Null，则返回 Null。
start	必需的。规定起始位置。如果设置为大于字符串中的字符数目，则返回空字符串("")。
length	可选的。要返回的字符数目。如果省略或 length 超过文本的字符数（包括 start 处的字符），将返回字符串中从 start 到字符串结束的所有字符。

实例

例子 **1**

```
dim txt
txt="This is a beautiful day!"
document.write(Mid(txt,1,1))
```

输出：

```
T
```

例子 **2**

```
dim txt
txt="This is a beautiful day!"
document.write(Mid(txt,1,11))
```

输出：

```
This is a b
```

例子 **3**

```
dim txt
txt="This is a beautiful day!"
document.write(Mid(txt,1))
```

输出：

```
This is a beautiful day!
```

例子 **4**

```
dim txt
txt="This is a beautiful day!"
document.write(Mid(txt,10))
```

输出：

```
beautiful day!
```

VBScript Replace 函数

定义和用法

Replace 函数可使用一个字符串替换另一个字符串指定的次数。

语法

```
Replace(string,find,replacewith[,start[,count[,compare]]])
```

参数	描述
string	必需的。需要被搜索的字符串。
find	必需的。将被替换的字符串部分。
replacewith	必需的。用于替换的子字符串。
start	可选的。规定开始位置。默认是 1。
count	可选的。规定指定替换的次数。默认是 -1，表示进行所有可能的替换。
compare	可选的。规定所使用的字符串比较类型。默认是 0。

参数 **compare** 的值：

常数	值	描述
vbBinaryCompare	0	执行二进制比较。
vbTextCompare	1	执行文本比较。

Replace 可能返回的值：

参数可能的取值	**Replace** 返回的值
expression 为零长度	零长度字符串 ("")。
expression 为 Null	错误。
find 参数为零长度	expression 的副本。
replacewith 参数为零长度	expression 的副本，其中删除了所有由 find 参数指定的内容。
start > Len(expression)	零长度字符串。
count 为 0	expression 的副本。

实例

例子 **1**

```
dim txt
txt="This is a beautiful day!"
document.write(Replace(txt,"beautiful","horrible"))
```

输出：

```
This is a horrible day!
```

VBScript Right 函数

定义和用法

Right 函数可从字符串右侧返回指定数目的字符。

提示：请使用Len 函数来确定字符串中的字符数目。

提示：请参阅 Left 函数。

语法

```
Right(string,length)
```

参数	描述
string	必需的。从其中返回字符的字符串。
length	必需的。规定返回多少字符。如果设置为 0，则返回空字符串 ("")。如果设置为大于或等于字符串的长度，则返回整个的字符串。

实例

例子 **1**

```
dim txt
txt="This is a beautiful day!"
document.write(Right(txt,11))
```

输出：

```
utiful day!
```

例子 **2**

```
dim txt
txt="This is a beautiful day!"
document.write(Right(txt,100))
```

输出：

```
This is a beautiful day!
```

例子 **3**

```
dim txt,x
txt="This is a beautiful day!"
x=Len(txt)
document.write(Right(txt,x))
```

输出：

```
This is a beautiful day!
```

VBScript Space 函数

定义和用法

Space 函数可返回一个由指定数目的空格组成的字符串。

语法

```
Space(number)
```

参数	描述
number	必需的。字符串中的空格数目。

实例

例子 **1**

```
dim txt
txt=Space(10)
document.write(txt)
```

输出：

```
"          "
```

VBScript StrComp 函数

定义和用法

StrComp 函数可比较两个字符串，并返回表示比较结果的一个值。

StrComp 函数可返回下面的值：

- -1 (如果 string1 < string2)
- 0 (如果 string1 = string2)
- 1 (如果 string1 > string2)
- Null (如果 string1 或 string2 为 Null)

语法

```
StrComp(string1,string2[,compare])
```

参数	描述
string1	必需的。字符串表达式。
string2	必需的。字符串表达式。
compare	可选的。规定要使用的字符串比较类型。默认是 0。Optional. Specifies the string comparison to use. Default is 0 可采用的值： • 0 = vbBinaryCompare - 执行二进制比较 • 1 = vbTextCompare - 执行文本比较

实例

例子 **1**

```
document.write(StrComp("VBScript","VBScript"))
```

输出：

```
0
```

例子 **2**

```
document.write(StrComp("VBScript","vbscript"))
```

输出：

```
-1
```

例子 **3**

```
document.write(StrComp("VBScript","vbscript",1))
```

输出：

```
0
```

VBScript String 函数

定义和用法

String 函数可返回包含指定长度的重复字符的一个字符串。

语法

```
String(number,character)
```

参数	描述
number	必需的。被返回字符串的长度。
character	必需的。被重复的字符。

实例

例子 **1**

```
document.write(String(10,"#"))
```

输出：

```
##########
```

例子 **2**

```
document.write(String(4,"*"))
```

输出：

```
****
```

例子 **3**

```
document.write(String(4,42))
```

输出：

```
****
```

例子 **4**

```
document.write(String(4,"XYZ"))
```

输出：

```
XXXX
```

VBScript StrReverse 函数

定义和用法

StrReverse 函数可反转一个字符串。

语法

```
StrReverse(string)
```

参数	描述
string	必需的。需被反转的字符串。

实例

例子 **1**

```
dim txt
txt="This is a beautiful day!"
document.write(StrReverse(txt))
```

输出：

```
!yad lufituaeb a si sihT
```

VBScript UCase 函数

定义和用法

UCase 函数可把指定的字符串转换为大写。

提示：请参阅 LCase 函数。

语法

```
UCase(string)
```

参数	描述
string	必需的。需被转换为大写的字符串。

实例

例子 **1**

```
dim txt
txt="This is a beautiful day!"
document.write(UCase(txt))
```

输出：

```
THIS IS A BEAUTIFUL DAY!
```

例子 **2**

```
dim txt
txt="This Is a Beautiful Day!"
document.write(UCase(txt))
```

输出：

```
THIS IS A BEAUTIFUL DAY!
```

VBScript 其他函数

函数	描述
CreateObject	创建指定类型对象。
Eval	计算表达式，并返回结果。
GetLocale	返回当前区域设置 ID 值。
GetObject	返回对文件中 automation 对象的引用。
GetRef	允许您把 VBScript 子程序连接到页面上的一个 DHTML 事件。

InputBox	可显示对话框，用户可在其中输入文本，并/或点击按钮，然后返回结果。
IsEmpty	返回一个布尔值，指示指定的变量是否已被初始化。
IsNull	返回一个布尔值，指示指定的变量是否包含无效数据 (Null)。
IsNumeric	返回一个布尔值，指示指定的表达式是否可作为数字来计算。
IsObject	返回一个布尔值，指示指定的表达式是否是一个 automation 对象。
LoadPicture	返回一个图片对象。仅用于32位平台。
MsgBox	显示消息框，等待用户点击按钮，并返回指示用户点击了哪个按钮的值。
RGB	返回一个表示 RGB 颜色值的数字。
Round	对数进行四舍五入。
ScriptEngine	返回使用中的脚本语言。
ScriptEngineBuildVersion	返回使用中的脚本引擎版本号。
ScriptEngineMajorVersion	返回使用中的脚本引擎的主版本号。
ScriptEngineMinorVersion	返回使用中的脚本引擎的次版本号。
SetLocale	设置地区 ID ，并返回之前的地区 ID。
TypeName	返回指定变量的子类型。
VarType	返回指示变量子类型的值。

VBScript UCase 函数

定义和用法

CreateObject 函数可创建一个指定类型的对象。

语法

```
CreateObject(servername.typename[,location])
```

参数	描述
servername	必需的。提供此对象的应用程序名称。
typename	必需的。对象的类型或类（type/class）。

location	可选的。在何处创建对象。

实例

例子 **1**

```
dim myexcel
Set myexcel=CreateObject("Excel.Sheet")

myexcel.Application.Visible=True

...code...

myexcel.Application.Quit

Set myexcel=Nothing
```

VBScript GetLocale 函数

定义和用法

GetLocale 函数可返回当前的 locale ID 。

locale 是用户参考信息集合，比如用户的语言、国家和文化传统。locale 决定键盘布局、字母排序顺序和日期、时间、数字与货币格式等等。

返回值可以是任意一个 32-位 的值，如 区域设置 ID 所示。

语法

```
GetLocale()
```

实例

例子 **1**

```
dim c
c=GetLocale
document.write(c)
```

输出：

```
1033
```

Locale ID 表

Locale 描述	简写	十六进制值	十进制值
Afrikaans	af	0x0436	1078
Albanian	sq	0x041C	1052
Arabic ?United Arab Emirates	ar-ae	0x3801	14337
Arabic - Bahrain	ar-bh	0x3C01	15361
Arabic - Algeria	ar-dz	0x1401	5121
Arabic - Egypt	ar-eg	0x0C01	3073
Arabic - Iraq	ar-iq	0x0801	2049
Arabic - Jordan	ar-jo	0x2C01	11265
Arabic - Kuwait	ar-kw	0x3401	13313
Arabic - Lebanon	ar-lb	0x3001	12289
Arabic - Libya	ar-ly	0x1001	4097
Arabic - Morocco	ar-ma	0x1801	6145
Arabic - Oman	ar-om	0x2001	8193
Arabic - Qatar	ar-qa	0x4001	16385
Arabic - Saudi Arabia	ar-sa	0x0401	1025
Arabic - Syria	ar-sy	0x2801	10241
Arabic - Tunisia	ar-tn	0x1C01	7169
Arabic - Yemen	ar-ye	0x2401	9217
Armenian	hy	0x042B	1067
Azeri ?Latin	az-az	0x042C	1068
Azeri ?Cyrillic	az-az	0x082C	2092
Basque	eu	0x042D	1069
Belarusian	be	0x0423	1059
Bulgarian	bg	0x0402	1026
Catalan	ca	0x0403	1027
Chinese - China	zh-cn	0x0804	2052
Chinese - Hong Kong S.A.R.	zh-hk	0x0C04	3076

Chinese ?Macau S.A.R	zh-mo	0x1404	5124
Chinese - Singapore	zh-sg	0x1004	4100
Chinese - Taiwan	zh-tw	0x0404	1028
Croatian	hr	0x041A	1050
Czech	cs	0x0405	1029
Danish	da	0x0406	1030
Dutch ?The Netherlands	nl-nl	0x0413	1043
Dutch - Belgium	nl-be	0x0813	2067
English - Australia	en-au	0x0C09	3081
English - Belize	en-bz	0x2809	10249
English - Canada	en-ca	0x1009	4105
English ?Carribbean	en-cb	0x2409	9225
English - Ireland	en-ie	0x1809	6153
English - Jamaica	en-jm	0x2009	8201
English - New Zealand	en-nz	0x1409	5129
English ?Phillippines	en-ph	0x3409	13321
English - South Africa	en-za	0x1C09	7177
English - Trinidad	en-tt	0x2C09	11273
English - United Kingdom	en-gb	0x0809	2057
English - United States	en-us	0x0409	1033
Estonian	et	0x0425	1061
Farsi	fa	0x0429	1065
Finnish	fi	0x040B	1035
Faroese	fo	0x0438	1080
French - France	fr-fr	0x040C	1036
French - Belgium	fr-be	0x080C	2060
French - Canada	fr-ca	0x0C0C	3084
French - Luxembourg	fr-lu	0x140C	5132

French - Switzerland	fr-ch	0x100C	4108
Gaelic ?Ireland	gd-ie	0x083C	2108
Gaelic - Scotland	gd	0x043C	1084
German - Germany	de-de	0x0407	1031
German - Austria	de-at	0x0C07	3079
German - Liechtenstein	de-li	0x1407	5127
German - Luxembourg	de-lu	0x1007	4103
German - Switzerland	de-ch	0x0807	2055
Greek	el	0x0408	1032
Hebrew	he	0x040D	1037
Hindi	hi	0x0439	1081
Hungarian	hu	0x040E	1038
Icelandic	is	0x040F	1039
Indonesian	id	0x0421	1057
Italian - Italy	it-it	0x0410	1040
Italian - Switzerland	it-ch	0x0810	2064
Japanese	ja	0x0411	1041
Korean	ko	0x0412	1042
Latvian	lv	0x0426	1062
Lithuanian	lt	0x0427	1063
FYRO Macedonian	mk	0x042F	1071
Malay - Malaysia	ms-my	0x043E	1086
Malay ?Brunei	ms-bn	0x083E	2110
Maltese	mt	0x043A	1082
Marathi	mr	0x044E	1102
Norwegian - Bokm錶	no-no	0x0414	1044
Norwegian ?Nynorsk	no-no	0x0814	2068
Polish	pl	0x0415	1045
Portuguese - Portugal	pt-pt	0x0816	2070

Portuguese - Brazil	pt-br	0x0416	1046
Raeto-Romance	rm	0x0417	1047
Romanian - Romania	ro	0x0418	1048
Romanian - Moldova	ro-mo	0x0818	2072
Russian	ru	0x0419	1049
Russian - Moldova	ru-mo	0x0819	2073
Sanskrit	sa	0x044F	1103
Serbian - Cyrillic	sr-sp	0x0C1A	3098
Serbian ?Latin	sr-sp	0x081A	2074
Setsuana	tn	0x0432	1074
Slovenian	sl	0x0424	1060
Slovak	sk	0x041B	1051
Sorbian	sb	0x042E	1070
Spanish - Spain	es-es	0x0C0A	1034
Spanish - Argentina	es-ar	0x2C0A	11274
Spanish - Bolivia	es-bo	0x400A	16394
Spanish - Chile	es-cl	0x340A	13322
Spanish - Colombia	es-co	0x240A	9226
Spanish - Costa Rica	es-cr	0x140A	5130
Spanish - Dominican Republic	es-do	0x1C0A	7178
Spanish - Ecuador	es-ec	0x300A	12298
Spanish - Guatemala	es-gt	0x100A	4106
Spanish - Honduras	es-hn	0x480A	18442
Spanish - Mexico	es-mx	0x080A	2058
Spanish - Nicaragua	es-ni	0x4C0A	19466
Spanish - Panama	es-pa	0x180A	6154
Spanish - Peru	es-pe	0x280A	10250
Spanish - Puerto Rico	es-pr	0x500A	20490

Spanish - Paraguay	es-py	0x3C0A	15370
Spanish - El Salvador	es-sv	0x440A	17418
Spanish - Uruguay	es-uy	0x380A	14346
Spanish - Venezuela	es-ve	0x200A	8202
Sutu	sx	0x0430	1072
Swahili	sw	0x0441	1089
Swedish - Sweden	sv-se	0x041D	1053
Swedish - Finland	sv-fi	0x081D	2077
Tamil	ta	0x0449	1097
Tatar	tt	0X0444	1092
Thai	th	0x041E	1054
Turkish	tr	0x041F	1055
Tsonga	ts	0x0431	1073
Ukrainian	uk	0x0422	1058
Urdu	ur	0x0420	1056
Uzbek ?Cyrillic	uz-uz	0x0843	2115
Uzbek ?Latin	uz-uz	0x0443	1091
Vietnamese	vi	0x042A	1066
Xhosa	xh	0x0434	1076
Yiddish	yi	0x043D	1085
Zulu	zu	0x0435	1077

VBScript GetObject 函数

定义和用法

GetObject 函数可返回对文件中 Automation 对象的引用。

语法

```
GetObject([pathname][,class])
```

参数	描述
pathname	可选的。包含 automation 对象的文件的完整路径和名称。如果此参数被忽略，就必须有 class 参数。
class	可选的。automation 对象的类。此参数使用此语法：appname.objectype。

VBScript GetRef 函数

定义和用法

GetRef 函数可把一段 VBScript 过程（子程序）连接到页面的一个 DHTML 事件上。

语法

```
Set object.event=GetRef(procname)
```

参数	描述
object	必需的。事件所关联的对象的名称。
event	Required. 要与函数绑定的事件的名称。
procname	Required. 与事件关联的 Sub 或 Function 过程的名称。

实例

```
Function test()
dim txt
txt="GetRef Test" & vbCrLf
txt=txt & "Hello World!"
MsgBox txt
End Function

Window.Onload=GetRef("test")
```

VBScript InputBox 函数

定义和用法

InputBox 函数可显示一个对话框，用户可在其中输入文本并/或点击一个按钮。如果用户点击点击确认按钮或按键盘上的回车键， 则 InputBox 函数返回文本框中的文本。如果用户点击取消按钮，函数返回一个空字符串("")。

注释：若同时规定helpfile 和 context 参数，则会向对话框添加一个帮助按钮。

提示：请参阅 MsgBox 函数。

语法

```
InputBox(prompt[,title][,default][,xpos][,ypos][,helpfile,context])
```

参数	描述
prompt	必需的。现实在对话框中的消息。prompt 的最大长度大约是 1024 个字符，这取决于所使用的字符的宽度。如果 prompt 中包含多个行，则可在各行之间用回车符 (Chr(13))、换行符 (Chr(10)) 或回车换行符的组合 (Chr(13) & Chr(10)) 以分隔各行。
title	可选的。显示在对话框标题栏中的字符串表达式。如果省略 title，则应用程序的名称将显示在标题栏中。
default	可选的。显示在文本框中的字符串表达式，在没有其它输入时作为默认的响应值。如果省略 default，则文本框为空。
xpos	可选的。数值表达式，用于指定对话框的左边缘与屏幕左边缘的水平距离（单位为缇）。如果省略 xpos，则对话框会在水平方向居中。
ypos	可选的。数值表达式，用于指定对话框的上边缘与屏幕上边缘的垂直距离（单位为缇）。如果省略 ypos，则对话框显示在屏幕垂直方向距下边缘大约三分之一处。
helpfile	可选的。字符串表达式，用于标识为对话框提供上下文相关帮助的帮助文件。如果已提供 helpfile，则必须提供 context。
context	可选的。数值表达式，用于标识由帮助文件的作者指定给某个帮助主题的上下文编号。如果已提供 context，则必须提供 helpfile。

实例

```
dim fname
fname=InputBox("Enter your name:")
MsgBox("Your name is " & fname)
```

VBScript IsEmpty 函数

定义和用法

IsEmpty 函数可返回指定的变量是否被初始化的布尔值。如果未被初始化，则返回 true ，否则返回 False 。

语法

```
IsEmpty(expression)
```

参数	描述
expression	必需的。表达式（通常是一个变量名）。

实例

```
dim x
document.write(IsEmpty(x))
x=10
document.write(IsEmpty(x))
x=Empty
document.write(IsEmpty(x))
x=Null
document.write(IsEmpty(x))
```

分别输出：

```
True
False
True
False
```

VBScript IsNull 函数

定义和用法

IsNull 函数可返回指示指定表达式是否无效数据的布尔值。如果表达式是 Null，则返回 True，否则返回 False。

语法

```
IsNull(expression)
```

参数	描述
expression	必需的。表达式。

实例

```
dim x
document.write(IsNull(x))
x=10
document.write(IsNull(x))
x=Empty
document.write(IsNull(x))
x=Null
```

```
document.write(IsNull(x))
```

分别输出：

```
False
False
False
True
```

VBScript IsNumeric 函数

定义和用法

IsNumeric 函数可返回指示指定的表达式是否可作为数字来计算的布尔值。如果作为数字来计算，则返回 True ， 否则返回 False。

注释：如果表达式是日期表达式，则 IsNumeric 返回 False。

语法

```
IsNumeric(expression)
```

参数	描述
expression	必需的。表达式。

实例

```
dim x
x=10
document.write(IsNumeric(x))
x=Empty
document.write(IsNumeric(x))
x=Null
document.write(IsNumeric(x))
x="10"
document.write(IsNumeric(x))
x="911 Help"
document.write(IsNumeric(x))
```

分别输出：

```
True
True
False
True
```

```
False
```

VBScript IsObject 函数

定义和用法

IsObject 函数可返回指示是否指定的表达式是一个 automation 对象的布尔值。如果表达式是对象，则返回 True 。否则返回 False。

语法

```
IsObject(expression)
```

参数	描述
expression	必需的。表达式。

实例

例子 **1**

```
dim x
set x=me
document.write(IsObject(x))
```

输出：

```
True
```

例子 **2**

```
dim x
x="me"
document.write(IsObject(x))
```

输出：

```
False
```

VBScript LoadPicture 函数

定义和用法

LoadPicture 函数可返回一个图片对象。

可被 LoadPicture 函数识别的图形格式有：

- bitmap 文件 (.bmp)
- icon 文件 (.ico)
- run-length encoded 文件 (.rle)
- metafile 文件 (.wmf)
- enhanced meta文件 (.emf)
- GIF 文件 (.gif)
- JPEG 文件 (.jpg)

注释：此函数仅供 32-位 平台。

语法

```
LoadPicture(picturename)
```

参数	描述
picturename	必需的。需被载入的图片文件的文件名。

VBScript MsgBox 函数

定义和用法

MsgBox 函数可显示一个消息框，等待用户点击某个按钮，然后返回指示被点击按钮的值。

MsgBox 函数可返回下面的值：

- 1 = vbOK - 确定按钮被单击。
- 2 = vbCancel - 取消按钮被单击。
- 3 = vbAbort - 终止按钮被单击。
- 4 = vbRetry - 重试按钮被单击。
- 5 = vbIgnore - 忽略按钮被单击。
- 6 = vbYes - 是按钮被单击。
- 7 = vbNo - 否按钮被单击。

注释：当 helpfile 和 context 参数均被规定后，用户可按 F1 键来查看帮助。

提示：请参阅 InputBox 函数。

语法

```
MsgBox(prompt[,buttons][,title][,helpfile,context])
```

参数	描述

prompt	必需的。作为消息显示在对话框中的字符串表达式。prompt 的最大长度大约是 1024 个字符，这取决于所使用的字符的宽度。如果 prompt 中包含多个行，则可在各行之间用回车符 (Chr(13))、换行符 (Chr(10)) 或回车换行符的组合 (Chr(13) & Chr(10)) 分隔各行。
buttons	数值表达式，是表示指定显示按钮的数目和类型、使用的图标样式，默认按钮的标识以及消息框样式的数值的总和。如果省略，则 buttons 的默认值为 0。 button 的取值： • 0 = vbOKOnly - 只显示确定按钮。 • 1 = vbOKCancel - 显示确定和取消按钮。 • 2 = vbAbortRetryIgnore - 显示放弃、重试和忽略按钮。 • 3 = vbYesNoCancel - 显示是、否和取消按钮。 • 4 = vbYesNo - 显示是和否按钮。 • 5 = vbRetryCancel - 显示重试和取消按钮。 • 16 = vbCritical - 显示临界信息图标。 • 32 = vbQuestion - 显示警告查询图标。 • 48 = vbExclamation - 显示警告消息图标。 • 64 = vbInformation - 显示信息消息图标。 • 0 = vbDefaultButton1 - 第一个按钮为默认按钮。 • 256 = vbDefaultButton2 - 第二个按钮为默认按钮。 • 512 = vbDefaultButton3 - 第三个按钮为默认按钮。 • 768 = vbDefaultButton4 - 第四个按钮为默认按钮。 • 0 = vbApplicationModal - 应用程序模式：用户必须响应消息框才能继续在当前应用程序中工作。 • 4096 = vbSystemModal - 系统模式：在用户响应消息框前，所有应用程序都被挂起。 第一组值 (0 - 5) 用于描述对话框中显示的按钮类型与数目；第二组值 (16, 32, 48, 64) 用于描述图标的样式；第三组值 (0, 256, 512) 用于确定默认按钮；而第四组值 (0, 4096) 则决定消息框的样式。在将这些数字相加以生成 buttons 参数值时，只能从每组值中取用一个数字。
title	显示在对话框标题栏中的字符串表达式。如果省略 title，则将应用程序的名称显示在标题栏中。
helpfile	字符串表达式，用于标识为对话框提供上下文相关帮助的帮助文件。如果已提供 helpfile，则必须提供 context。在 16 位系统平台上不可用。
context	数值表达式，用于标识由帮助文件的作者指定给某个帮助主题的上下文编号。如果已提供 context，则必须提供 helpfile。在 16 位系统平台上不可用。

实例

```
dim answer
answer=MsgBox("Hello everyone!",65,"Example")
document.write(answer)
```

VBScript RGB 函数

定义和用法

RGB 函数可返回表示 RGB 颜色值的数字。

语法

```
RGB(red,green,blue)
```

参数	描述
red	必需的。介于 0 - 255 之间（且包括）的数字，代表颜色红色部分。
green	必需的。介于 0 - 255 之间（且包括）的数字，代表颜色绿色部分。
blue	必需的。介于 0 - 255 之间（且包括）的数字，代表颜色蓝色部分。

实例

例子 **1**

```
document.write(rgb(255,0,0))
```

输出：

```
255
```

例子 **2**

```
document.write(rgb(255,30,30))
```

输出：

```
1974015
```

VBScript Round 函数

定义和用法

Round 函数可对数字进行四舍五入。

语法

```
Round(expression[,numdecimalplaces])
```

参数	描述
expression	必需的。需要被四舍五入的表达式。
numdecimalplaces	可选的。规定对小数点右边的多少位进行四舍五入。默认是 0。

实例

例子 **1**

```
dim x
x=24.13278
document.write(Round(x))
```

输出：

```
24
```

例子 **2**

```
dim x
x=24.13278
document.write(Round(x,2))
```

输出：

```
24.13
```

VBScript ScriptEngine, ScriptEngineBuildVersion, ScriptEngineMajorVersion, 以及 ScriptEngineMinorVersion 函数

ScriptEngine 函数

ScriptEngine 函数可返回当前使用的脚本语言。

此函数可返回下面的字符串：

- VBScript - 指示当前使用的脚本引擎是 Microsoft(R) Visual Basic(R) Scripting Edition
- JScript - 指示当前使用的编写脚本引擎是 Microsoft Visual Basic Scripting Edition(R)。
- VBA - 指示当前使用的脚本引擎是 Microsoft Visual Basic for Applications。

ScriptEngineBuildVersion 函数

ScriptEngineBuildVersion 函数可返回当前在用的脚本引擎的版本号。

ScriptEngineMajorVersion 函数

ScriptEngineMajorVersion 函数可返回当前在用的脚本引擎的主版本号。

ScriptEngineMinorVersion 函数

ScriptEngineMinorVersion 函数可返回当前在用的脚本引擎的副版本。

语法

```
ScriptEngine

ScriptEngineBuildVersion

ScriptEngineMajorVersion

ScriptEngineMinorVersion
```

实例

```
document.write(ScriptEngine)
document.write(ScriptEngineBuildVersion)
document.write(ScriptEngineMajorVersion)
document.write(ScriptEngineMinorVersion)
```

输出：

```
VBScript
8827
5
6
```

VBScript SetLocale 函数

定义和用法

SetLocale 函数可设置 locale ID，并返回之前的 locale ID。

locale 是用户参考信息集合，比如用户的语言、国家和文化传统。locale 可决定键盘布局、字母排序顺序和日期、时间、数字与货币格式等等。

语法

```
SetLocale(lcid)
```

参数	描述
lcid	必需的。任意一个在 Locale ID 表 中的短字符串、十六进制值、十进制值，该值必须唯一标识一个地理区域。如果 lcid 参数被设置为 0 ， 则 locale 将由系统设置。

实例

```
document.write(SetLocale(2057))
document.write(SetLocale(2058))
```

输出：

```
1033
2057
```

Locale ID 表

Locale 描述	简写	十六进制值	十进制值
Afrikaans	af	0x0436	1078
Albanian	sq	0x041C	1052
Arabic ?United Arab Emirates	ar-ae	0x3801	14337
Arabic - Bahrain	ar-bh	0x3C01	15361
Arabic - Algeria	ar-dz	0x1401	5121
Arabic - Egypt	ar-eg	0x0C01	3073
Arabic - Iraq	ar-iq	0x0801	2049
Arabic - Jordan	ar-jo	0x2C01	11265
Arabic - Kuwait	ar-kw	0x3401	13313
Arabic - Lebanon	ar-lb	0x3001	12289
Arabic - Libya	ar-ly	0x1001	4097
Arabic - Morocco	ar-ma	0x1801	6145
Arabic - Oman	ar-om	0x2001	8193
Arabic - Qatar	ar-qa	0x4001	16385

Arabic - Saudi Arabia	ar-sa	0x0401	1025
Arabic - Syria	ar-sy	0x2801	10241
Arabic - Tunisia	ar-tn	0x1C01	7169
Arabic - Yemen	ar-ye	0x2401	9217
Armenian	hy	0x042B	1067
Azeri ?Latin	az-az	0x042C	1068
Azeri ?Cyrillic	az-az	0x082C	2092
Basque	eu	0x042D	1069
Belarusian	be	0x0423	1059
Bulgarian	bg	0x0402	1026
Catalan	ca	0x0403	1027
Chinese - China	zh-cn	0x0804	2052
Chinese - Hong Kong S.A.R.	zh-hk	0x0C04	3076
Chinese ?Macau S.A.R	zh-mo	0x1404	5124
Chinese - Singapore	zh-sg	0x1004	4100
Chinese - Taiwan	zh-tw	0x0404	1028
Croatian	hr	0x041A	1050
Czech	cs	0x0405	1029
Danish	da	0x0406	1030
Dutch ?The Netherlands	nl-nl	0x0413	1043
Dutch - Belgium	nl-be	0x0813	2067
English - Australia	en-au	0x0C09	3081
English - Belize	en-bz	0x2809	10249
English - Canada	en-ca	0x1009	4105
English ?Carribbean	en-cb	0x2409	9225
English - Ireland	en-ie	0x1809	6153
English - Jamaica	en-jm	0x2009	8201
English - New Zealand	en-nz	0x1409	5129

English ?Phillippines	en-ph	0x3409	13321
English - South Africa	en-za	0x1C09	7177
English - Trinidad	en-tt	0x2C09	11273
English - United Kingdom	en-gb	0x0809	2057
English - United States	en-us	0x0409	1033
Estonian	et	0x0425	1061
Farsi	fa	0x0429	1065
Finnish	fi	0x040B	1035
Faroese	fo	0x0438	1080
French - France	fr-fr	0x040C	1036
French - Belgium	fr-be	0x080C	2060
French - Canada	fr-ca	0x0C0C	3084
French - Luxembourg	fr-lu	0x140C	5132
French - Switzerland	fr-ch	0x100C	4108
Gaelic ?Ireland	gd-ie	0x083C	2108
Gaelic - Scotland	gd	0x043C	1084
German - Germany	de-de	0x0407	1031
German - Austria	de-at	0x0C07	3079
German - Liechtenstein	de-li	0x1407	5127
German - Luxembourg	de-lu	0x1007	4103
German - Switzerland	de-ch	0x0807	2055
Greek	el	0x0408	1032
Hebrew	he	0x040D	1037
Hindi	hi	0x0439	1081
Hungarian	hu	0x040E	1038
Icelandic	is	0x040F	1039
Indonesian	id	0x0421	1057
Italian - Italy	it-it	0x0410	1040
Italian - Switzerland	it-ch	0x0810	2064

Japanese	ja	0x0411	1041
Korean	ko	0x0412	1042
Latvian	lv	0x0426	1062
Lithuanian	lt	0x0427	1063
FYRO Macedonian	mk	0x042F	1071
Malay - Malaysia	ms-my	0x043E	1086
Malay ?Brunei	ms-bn	0x083E	2110
Maltese	mt	0x043A	1082
Marathi	mr	0x044E	1102
Norwegian - Bokm錶	no-no	0x0414	1044
Norwegian ?Nynorsk	no-no	0x0814	2068
Polish	pl	0x0415	1045
Portuguese - Portugal	pt-pt	0x0816	2070
Portuguese - Brazil	pt-br	0x0416	1046
Raeto-Romance	rm	0x0417	1047
Romanian - Romania	ro	0x0418	1048
Romanian - Moldova	ro-mo	0x0818	2072
Russian	ru	0x0419	1049
Russian - Moldova	ru-mo	0x0819	2073
Sanskrit	sa	0x044F	1103
Serbian - Cyrillic	sr-sp	0x0C1A	3098
Serbian ?Latin	sr-sp	0x081A	2074
Setsuana	tn	0x0432	1074
Slovenian	sl	0x0424	1060
Slovak	sk	0x041B	1051
Sorbian	sb	0x042E	1070
Spanish - Spain	es-es	0x0C0A	1034
Spanish - Argentina	es-ar	0x2C0A	11274
Spanish - Bolivia	es-bo	0x400A	16394

Spanish - Chile	es-cl	0x340A	13322
Spanish - Colombia	es-co	0x240A	9226
Spanish - Costa Rica	es-cr	0x140A	5130
Spanish - Dominican Republic	es-do	0x1C0A	7178
Spanish - Ecuador	es-ec	0x300A	12298
Spanish - Guatemala	es-gt	0x100A	4106
Spanish - Honduras	es-hn	0x480A	18442
Spanish - Mexico	es-mx	0x080A	2058
Spanish - Nicaragua	es-ni	0x4C0A	19466
Spanish - Panama	es-pa	0x180A	6154
Spanish - Peru	es-pe	0x280A	10250
Spanish - Puerto Rico	es-pr	0x500A	20490
Spanish - Paraguay	es-py	0x3C0A	15370
Spanish - El Salvador	es-sv	0x440A	17418
Spanish - Uruguay	es-uy	0x380A	14346
Spanish - Venezuela	es-ve	0x200A	8202
Sutu	sx	0x0430	1072
Swahili	sw	0x0441	1089
Swedish - Sweden	sv-se	0x041D	1053
Swedish - Finland	sv-fi	0x081D	2077
Tamil	ta	0x0449	1097
Tatar	tt	0X0444	1092
Thai	th	0x041E	1054
Turkish	tr	0x041F	1055
Tsonga	ts	0x0431	1073
Ukrainian	uk	0x0422	1058
Urdu	ur	0x0420	1056
Uzbek ?Cyrillic	uz-uz	0x0843	2115

Uzbek ?Latin	uz-uz	0x0443	1091
Vietnamese	vi	0x042A	1066
Xhosa	xh	0x0434	1076
Yiddish	yi	0x043D	1085
Zulu	zu	0x0435	1077

VBScript TypeName 函数

定义和用法

TypeName 函数可指定变量的子类型。

TypeName 函数可返回的值：

值	描述
Byte	字节值
Integer	整型值
Long	长整型值
Single	单精度浮点值
Double	双精度浮点值
Currency	货币值
Decimal	十进制值
Date	日期或时间值
String	字符串值
Boolean	Boolean 值；True 或 False
Empty	未初始化
Null	无有效数据
<object type>	实际对象类型名
Object	一般对象
Unknown	未知对象类型
Nothing	还未引用对象实例的对象变量
Error	错误

语法

```
TypeName(varname)
```

参数	描述
varname	必需的。变量的名称。

实例

```
dim x
x="Hello World!"
document.write(TypeName(x))
x=4
document.write(TypeName(x))
x=4.675
document.write(TypeName(x))
x=Null
document.write(TypeName(x))
x=Empty
document.write(TypeName(x))
x=True
document.write(TypeName(x))
```

输出：

```
String
Integer
Double
Null
Empty
Boolean
```

VBScript VarType 函数

定义和用法

VarType 函数可返回指示指定变量的子类型的值。

VarType 函数可返回的值：

常数	值	描述
vbEmpty	0	未初始化（默认）
vbNull	1	不包含任何有效数据

vbInteger	2	整型子类型
vbLong	3	长整型子类型
vbSingle	4	单精度子类型
vbDouble	5	双精度子类型
vbCurrency	6	货币子类型
vbDate	7	日期或时间值
vbString	8	字符串值
vbObject	9	字符串子类型
vbError	10	错误子类型
vbBoolean	11	Boolean 子类型
vbVariant	12	Variant （仅用于变量数组）
vbDataObject	13	数据访问对象
vbDecimal	14	十进制子类型
vbByte	17	字节子类型
vbArray	8192	数组

注释：这些常数是由 VBScript 指定的。所以，这些名称可在代码中随处使用，以代替实际值。

注释：假如变量是数组，则 VarType() 会返回 8192 + VarType(数组元素)。举例：整数数组的 VarType() 会返回 8192 + 2 = 8194 。

语法

```
VarType(varname)
```

参数	描述
varname	必需的。变量的名称。

实例

```
dim x
x="Hello World!"
document.write(VarType(x))
x=4
document.write(VarType(x))
x=4.675
document.write(VarType(x))
```

```
x=Null
document.write(VarType(x))
x=Empty
document.write(VarType(x))
x=True
document.write(VarType(x))
```

分别输出：

```
String
Integer
Double
Null
Empty
Boolean
```

免责声明