
学校的理想装备
电子图书·学校专集
校园网上的最佳资源
Visual Interdev 6.0
程序员指南（六）
BOOK
内部资料 非卖品

返回总目录

第五篇　编辑与脚本编程

第五篇　编辑与脚本编程

第五篇提供有关页面编辑、模型编程和脚本调试诸方面的内容。

第二十三章　脚本编程概念

这一章包括 Visual InterDev 编辑器和和下述各节所涉及的概念：“Web 应用中的脚本”、“脚本对象模型”、“文档元素”和“脚本调试过程”。

第二十四章　用设计期间控件和脚本对象编写脚本

这一章讨论使用 Visual InterDev 设计期间控件为你的 Web 应用创建用户界面，以及扩展 Visual InterDev 脚本对象模型。

第二十五章　使用 HTML 元素编写脚本

即便在你的 Web 页面中不使用设计期间控件，你可以通过创建 HTML 元素和为 HTML 元素编写脚本的办法来创建全功能 Web 应用。这一章描述编写脚本同原本 HTML 元素之间的相互作用。

第二十六章　调试页面

这一章讨论在 Visual InterDev 中调试客户机脚本和服务器脚本。

第二十七章　把脚本包装成对象

Microsoft Script Components（scriptlet，脚本部件，小脚本）为你提供一种轻松方式来创建功能强可重复使用的控件与 COM 部件。这一章讨论创建和使用小脚本。

第二十三章　脚本编程概念

23.1　编　辑　模　型

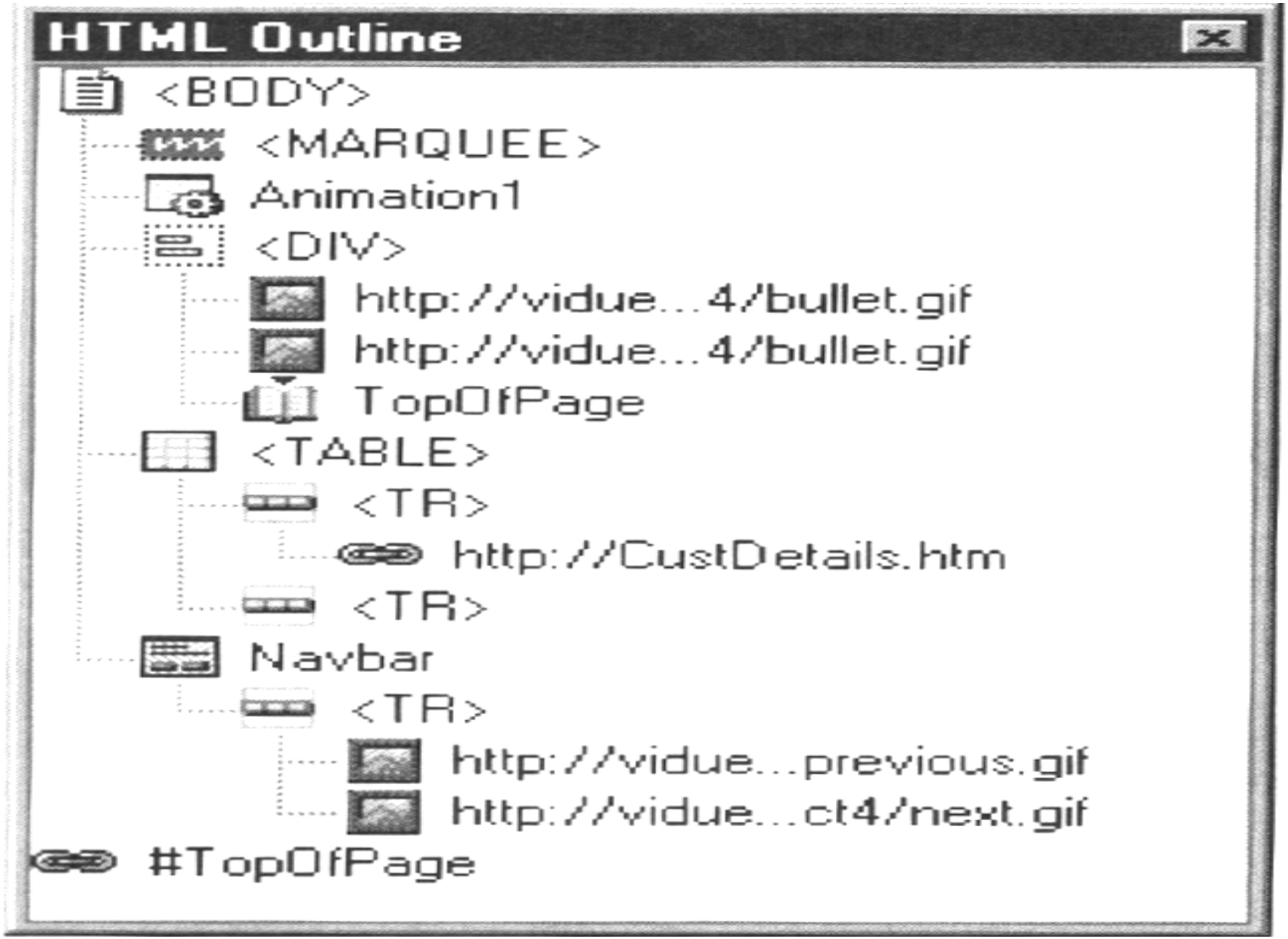
默认的 Visual InterDev HTML 编辑器允许你以不同模式加工 Web 页面：

- **Design view（设计视图）**　以字符和分段窗体显示文本的编辑器，很像文字处理器。
- **Source view（源视图）**　显示 HTML 标记、文本和脚本，并且亮显 HTML 标记和文本。
- **Queick view（快速视图）**　显示 .htm 文件的编辑器，文件出现的样子就像出现在 Internet Explorer 中一样。

在编辑器中导航

当你使用 HTML 编辑器的时候，Visual InterDev 允许你显示各种文档大纲，帮助你快速移动，轻松通过你的文档。

- 在 Design 和 Source 视图中，HTML Outline（大纲）窗口显示页面上 HTML 元素和对象的分层视图。


HTML Outline
<BODY>
<MARQUEE>
Animation1
<DIV>
http://vidue...4/bullet.gif
http://vidue...4/bullet.gif
TopOfPage
<TABLE>
<TR>
http://CustDetails.htm
<TR>
Navbar
<TR>
http://vidue...previous.gif
http://vidue...ct4/next.gif
#TopOfPage

HTML 大纲窗口

- 在 Source（源）视图中，Script Outline（脚本大纲）窗口给出页面上所有脚本元素的树状图；对每个元素给出你可以编写脚本的事件。已经存在的脚本用黑体表示。

脚本大纲窗口

当你在任何一个大纲窗口中选择一个元素的时候，你就移到页面上那个元素的位置。

工具框

当你用编辑器工作时，你可以把对象加到你的页面上，使用的办法是把对象从 Toolbox（工具框）拖到页面上。你可以使用 Design 视图和 Source 视图中的 Toolbox。Toolbox 在你的计算机上显示一组预选控件，其中包括标准 HTML 控件（例如文本框和按钮）、Visual InterDev 设计期间控件、ActiveX 控件和服务器对象。

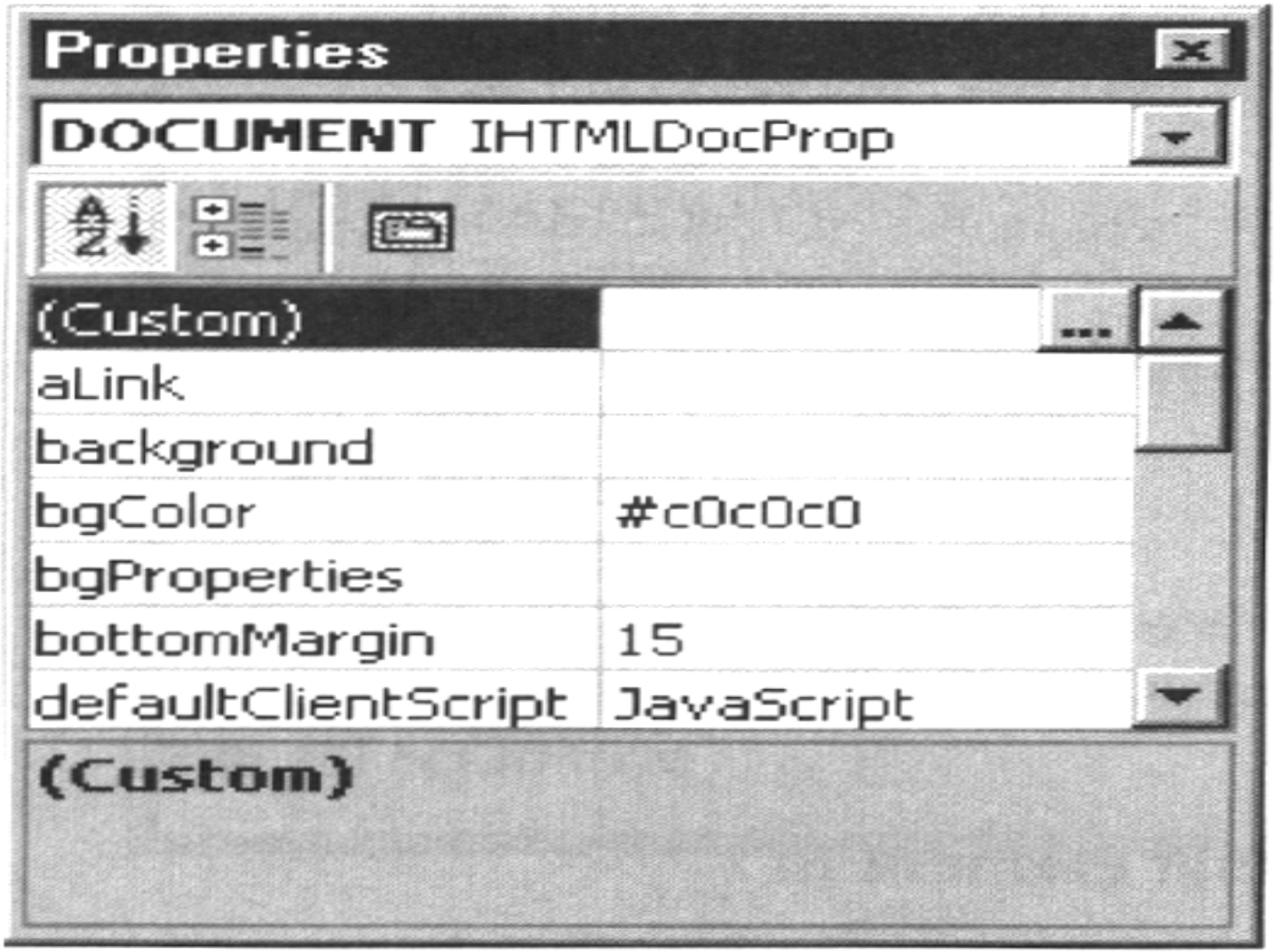
然而，Toolbox 只不过是添加对象的另一种手段。你可以使用它存储编辑器中的任何文本，其中包括脚本和 HTML 文本等。你在 Toolbox 中存储的这些文本二进制位称作“片断”(fragment)。此外，你可以在 Toolbox 中编辑选项卡，添加、删除和重新安排这些选项卡，以适应你的开发环境需要。

设计视图

设计视图（Design view）使得查看和编辑 HTML 文本变得轻松容易，所使用的窗体看上去同浏览器的窗体类似。你可以添加 HTML 文本并用 HTML 文本进行工作，还可以添加和使用一些元素，如图像、表格、设计期间控件、HTML 控件（窗体、按钮等）、ActiveX 控件和 Java 小程序等。你的页面全是用字符显示的，你指定的分段窗体很像在浏览器中的窗体。如果你正使用级联样式图表(CSS)或者已经把样式信息加到 HTML 标记中，设计视图就能反映出来。

因为你在设计视图中工作，你可以为页面上的元素设置特性。如果显示的

是属性窗口(Properties windows)，页面上当前选中元素的属性就会显示出来。一些元素，例如表格，还允许是用客户属性对话框设置属性(attribute)。



在设计视窗中的属性窗口

注意:在设计视图中，属性窗口显示(在 Source 视图中不显示的)样式信息。

尽管设计视图看上去很像浏览器中的一个页面，但在以下诸方面是不同的：

- 字符与分段窗体可能不同，因为每种浏览器可以使用不同格式实现。
- 客户脚本不运行。
- 链接不活化。
- 某些元素，例如脚本、注解和不能识别的 HTML 标记，都可以作为图示符(glyphs)显示出来，这样你便知道页面上有这些元素。
- 你可以显示元素周围的在浏览器中看不见的边框，这样你就可以看见边框在什么地方。例如，即便一个表格没规定边框，你可以从设计工具栏中选择“可见边框”，使编辑器在表格周围显示一个像素宽的边框，以便你在表格上工作。

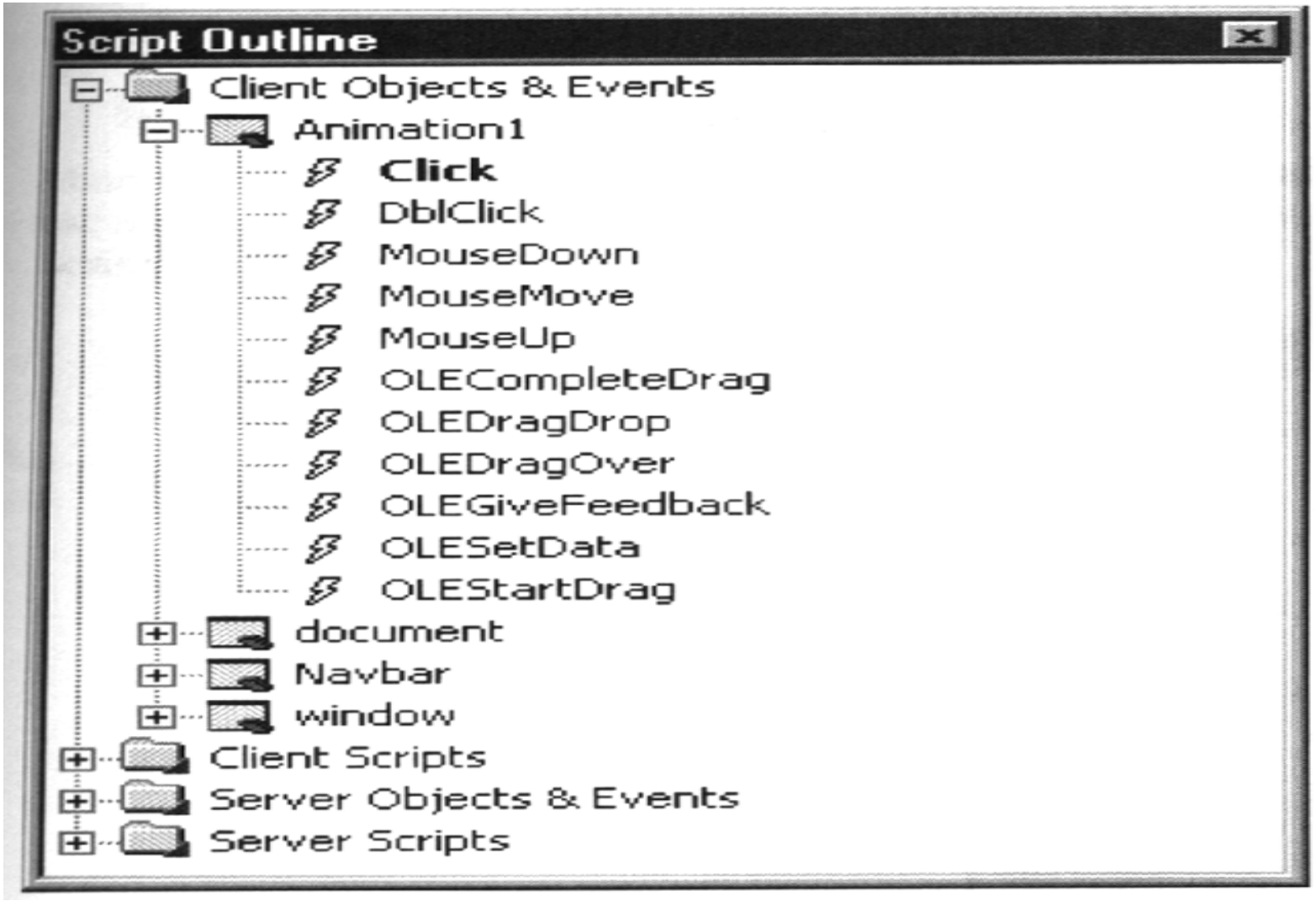
关于设计视图的更多细节，参见在线文档中“Design View,HTML Editor”和“Design Toolbar”。

源视图

源视图（Source view）允许你在页面上查看并编辑原始 HTML，并直接用脚本进行工作。你还可以使用图形表示的或文本表示的 Visual InterDev 设计期间控件、ActiveX 控件和 Java 小程序进行工作。文本是彩色的，因而你可以很容易区分脚本关键字、HTML 标记、属性、注解等。这些性质使你很容易开发 Web 页面，并全面控制其内容。

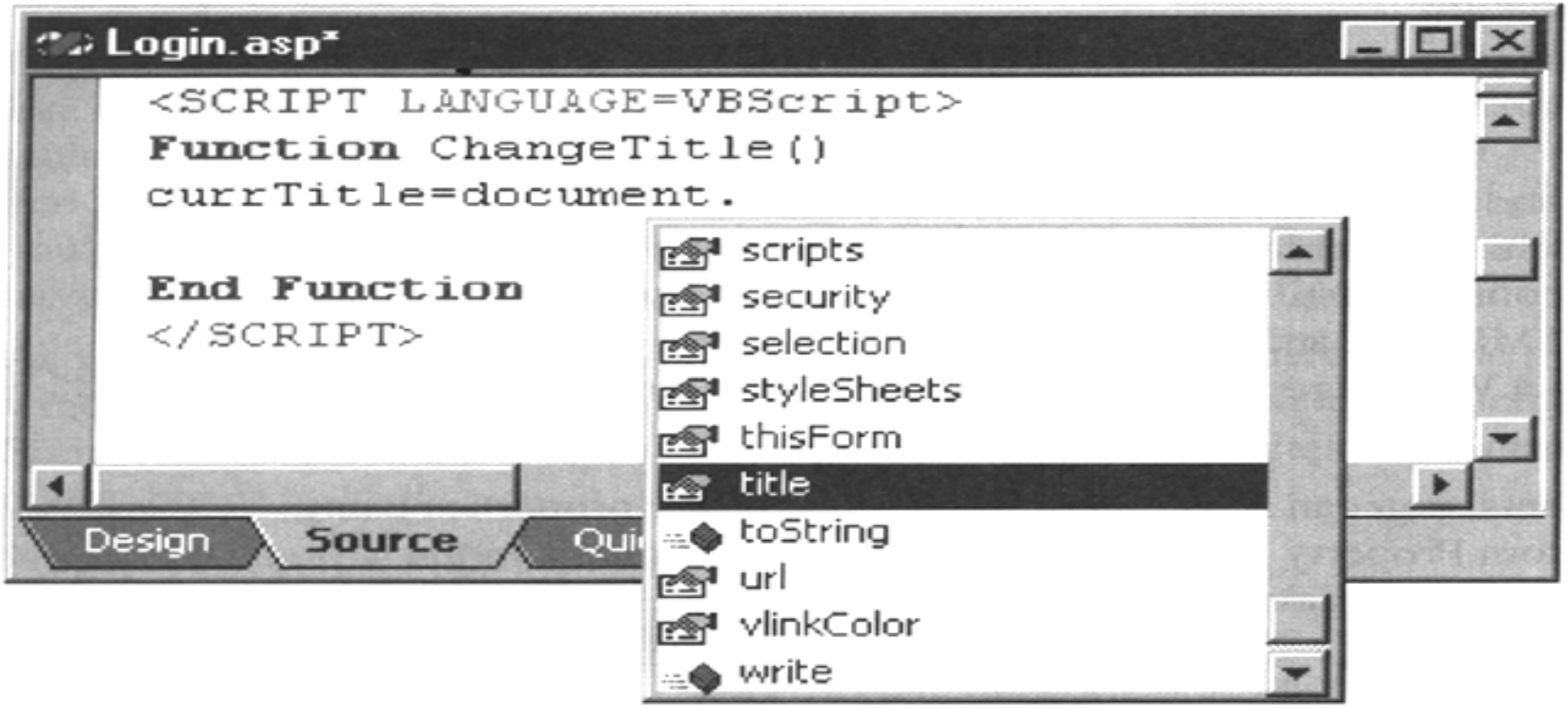
为编辑对象或 HTML 元素，你可以选择它们，然后使用 Properties（属性）窗口或客户 Property Pages（属性页面）窗口设置它们的属性。在属性窗口中或在属性页面对话框中所做的改变都反映在这些对象的 HTML 源代码中。

如果你的页面包含脚本，你就可以在源视图中看见脚本的源代码。在源视图中，你可以使用 Script Outline（脚本大纲）窗口查看你页面上的脚本元素，看看已经创建了什么脚本。



脚本大纲窗口

随着你键入语句，针对你使用的对象，IntelliSense 显示属性和方法列表，帮助你完成语句。



IntelliSence 完成语句

当你在“源视图”中编辑一个脚本的时候，你可以调出一个调试器，这种调试器使你设置断点，一步步通过脚本，并完成其它典型调试任务。

快速视图

要想事先看看 .htm 文件在 Internet Explorer 中看上去是什么样的，你

可以使用“Quick view”（快速视图）。图像和链接的工作就像文档在目的 Web 服务器上自己的位置上工作一样。

快速视图

注意：快速视图不通过服务器处理页面，因而它不能对 ASP 文件看上去应当

是什么样提供精确视图。如果你希望预览包含服务器元素的文档，要使用 View 菜单中的 View in Browser 命令。

因为快速视图为你展示浏览器中的一个页面，因而你不能编辑或调试这个视图中的页面。

23.2 Web 应用中的脚本

为显示一个页面上的文本、图像或链接，你要添加文本，并用 HTML 标记赋予它格式。然而，要控制页面的表现方式，你就要创建嵌入到页面中的脚本或程序，完成指定功能，例如：

- 在用户敲击按钮、键入文字或提交窗体时，控制出现的内容。
- 根据条件，例如用户的喜好，导航到特定页面。
- 收集并存储用户信息，以便动态定制 Web 应用。
- 查询数据库，显示结构。

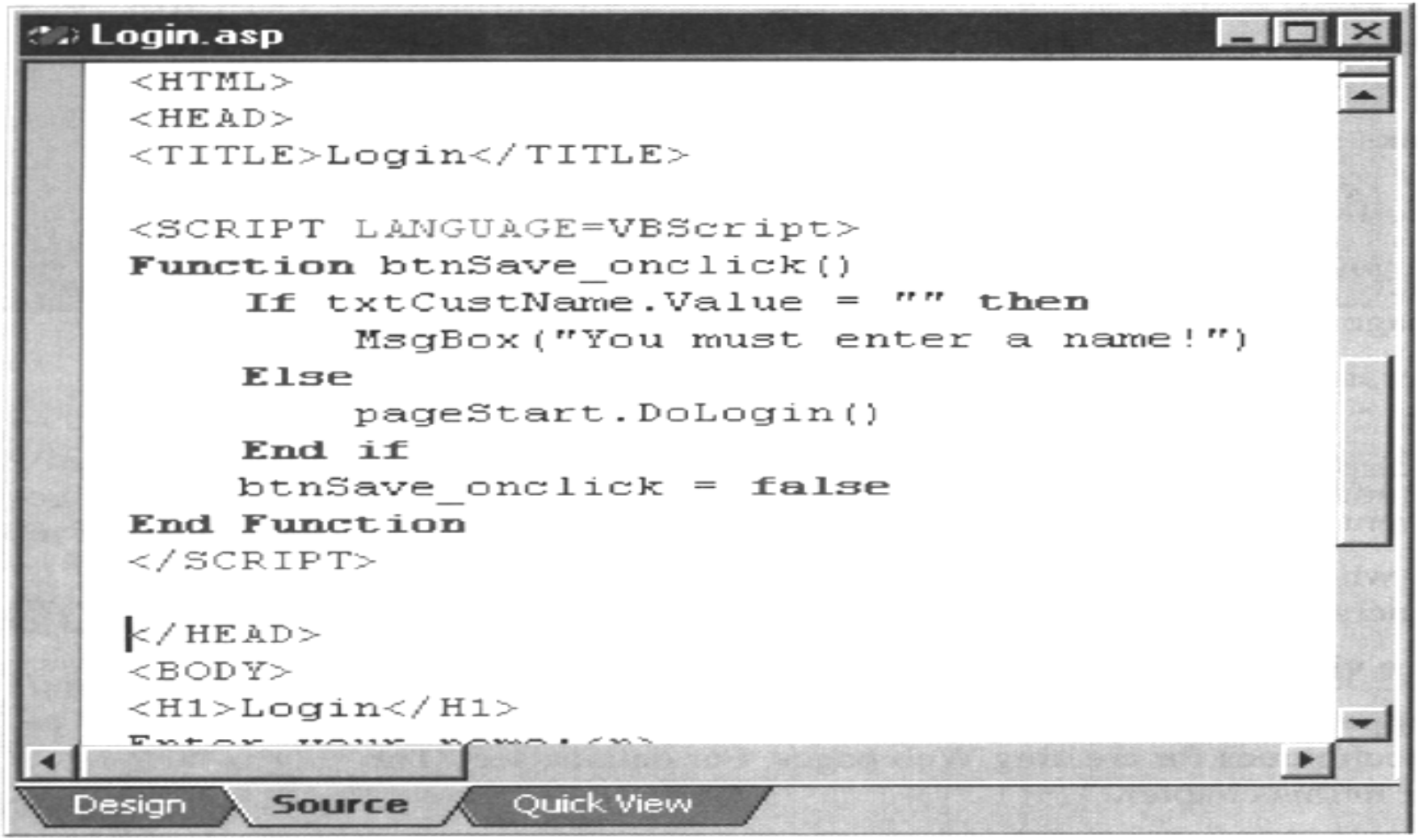
你可以用不同方式创建脚本。

- 使用设计期间控件，允许你设置属性值，并在对话框中键入值，然后为你生成脚本。
- 在 HTML 编辑器中的源视图中编写你的脚本。

Visual InterDev 支持完整的脚本对象模型，允许你使用标准的面向对象的技术创建 Web 页面。详见本章后边“脚本对象模型”一节。

脚 本 如 何 工 作

脚 本 源 代 码 出 现 在 页 面 上 ， 如 下 边 例 子 所 示 。



脚 本 源 代 码

当 请 求 页 面 时 ， 页 面 上 的 脚 本 同 其 它 所 有 文 本 一 起 读 出 来 。 服 务 器 或 浏 览

器读取脚本，检查是否有错误，然后运行这个脚本。

因为脚本是简单的文本块，你可以在一个页面上编写脚本，然后把它放到多个其它页面上。详见第二十五章“用 HTML 元素编写脚本”中的“编写可重复使用的脚本”。

脚本语言

你可以使用任何你感觉舒适的脚本语句编写脚本。常见的脚本语言有 Microsoft Visual Basic、Script Edition(VBScript)和 ECMAScript(一种标准的脚本语言)。实现 ECMAScript 的流行语言是 Microsoft JScript 和 JavaScript。

因为脚本语言是解释性语言，你必须保证在用户请求一个页面的时候，用户的浏览器（和服务器，如果你正在编写服务器脚本的话）可以使用你在编写脚本时使用的语言。例如，如果你使用 VBScript 语言编写你的所有脚本，你必须保证用户浏览器可以解释 VBScript。微软的 Internet Explorer 支持 VBScript，但是不是所有浏览器都支持。有关浏览器能力的细节，可查阅浏览器文档，参见第二十五章“用 HTML 元素编写脚本”中的“创建可移植脚本”一节。

如果需要，你可以在同一个页面上使用不同脚本语言。例如，如果把你自己的脚本加到用设计期间控件产生的脚本就会出现这种情况。当使用设计期间控件的时候，你规定一种目标编写脚本平台（客户机或服务器），这种平台是由脚本中默认语言确定的。有关细节，参见第二十四章“用设计期间控件和脚本对象编写脚本”中的“用设计期间控件创建窗体”一节。

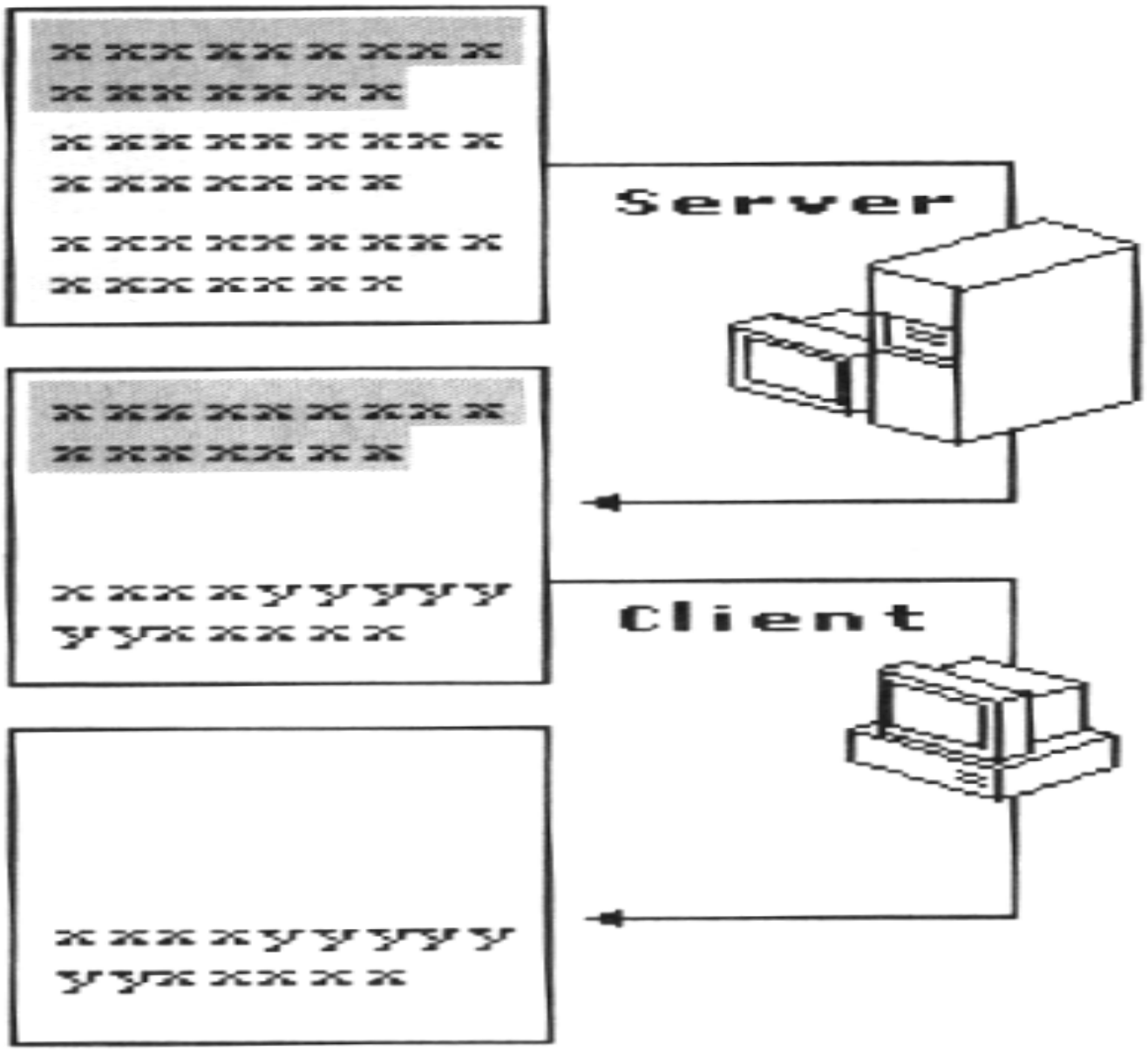
如果你不用设计期间控件编写脚本，则通常只用一种语言进行工作，因而你可以为你创建的每个新页面指定一种默认语言。如果需要，你可以为单个脚本指定一种默认语言。有关细节，参见第二十五章“用 HTML 元素编写脚本”中的“选择一种脚本语言”一节。

客户机和服务器脚本

如果你的服务器是 Microsoft Internet Information Server(IIS)，你便可以创建包含客户机和服务器两种脚本的 ASP 文件。两种类型的脚本可以出现同一个页面上。

客户机脚本是一个页面的一部分，在用户请求这个页面的时候发送给浏览器，并由浏览器运行。

服务器脚本也是一个页面的一部分，但是不发送给浏览器。相反，这些脚本在页面被请求之后，但在页面传递给浏览器之前由 IIS 运行。在页面发送给浏览器的时候，服务器已经运行了这个服务器脚本，并且已经从这个页面上去掉。



执 行 客 户 机 和 服 务 器 脚 本

规定脚本是在客户机上运行还是在服务器上运行，这是 Web 脚本的一个重要特性。Web 脚本允许你为希望完成的任务规定正确的运行期间环境。例如，通常由客户机脚本完成下列任务：

- 在页面装载到浏览器的时候或对诸如单击按钮的事件作出响应的时候，改变文本或页面的样子。
- 对键入到 HTML 窗体中的数据，在发送给服务器之前进行验证，例如要保证雇员的 ID 号码包含正确的数字位数。相比之下，对数据库的数据进行验证是服务器的常见任务。
- 显示信息，响应用户事件，例如单击按钮。

相反，有些任务应当用服务器脚本去完成：

- 查询数据库，把结果送给一个 HTML 页面。有关细节，参见第十九章“查看数据”。
- 根据某种条件，例如口令查阅，把用户的请求重定向给特定的页面。关于使用脚本在页面之间进行移动的更一步的信息，参见第二十四章“使用设计期间控件和脚本对象编写脚本”中的“跨越页面扩展脚本对象模型”和第二十五章“使用 HTML 元素编写脚本”中的“有条件导航”。
- 处理用户键入到 HTML 窗体中的信息。关于创建脚本处理 HTML 窗体进一步内容，参见第二十四章“用设计期间控件和脚本对象编写脚本”中的“为脚本对象编写脚本”和第二十五章“使用 HTML 元素编写脚本”中的“用 HTML 窗体收集信息”。

尽管上边列出的任务是客户机脚本或服务器脚本完成的典型情况，但这不是严格的规则。例如，你可以使用 Visual InterDev 设计期间控件去创建服务

器脚本，这个脚本对客户事件，如单击按钮，作出响应。另一个例子，如果你的用户使用 Internet Explorer 4.0 或其它支持动态 HTML(DHTML)的浏览器，你可以编写一个应用，从客户机浏览器那里访问数据库。

因此，使用客户机脚本还是使用服务器脚本，不仅取决于你要完成的任务，还取决于你的应用运行的环境，特别是约束条件（如性能）等。

客户脚本处理

客户脚本是由诸如微软 Internet Explorer 一类的浏览器处理的，浏览器调出相应的运行期间模块，执行这个脚本。客户脚本置于<SCRIPT>和</SCRIPT>标记之间。下边的简单例子给出在页面上打印当前时机的脚本：

```
<SCRIPT LANGUAGE=" VBScript " >
    Document.Write time
</SCRIPT>
```

你的页面可以包含许多脚本块，你需要多少就可以包含多少。你可以把多个功能和子程序放到一个脚本块中，或者把它们中的每一个放到一个分开的脚本块中。

客户脚本是在不同时间得到处理的，具体什么时间，取决于脚本是如何编写的：

- 语句出现在一个脚本块中，但是不是过程（函数或子程序）的组成部分。这些语句称作“全局”（Global）脚本或“内联”（inline）脚本，而且在浏览器读出页面时按顺序处理。例如，下边给出一个全局脚本：

  ```
  <SCRIPT LANGUAGE=" VBScript " >
  ```

```
    Document.Write time
</SCRIPT>
```

- 语句作为过程的组成部分而出现，这里的过程指诸如函数或子程序。这些语句不立刻执行，而是在页面运行时和检查语法差错的时候进行语法分析。然而，它们在过程调出之前是不运行的。
- 事件处理过程不立即执行，而是在用户完成触发脚本的事件时，如单击按钮，才执行这个过程。例如，下边脚本说明事件处理子程序只在用户单击表单中 Submit（提交）按钮才执行：

```
<SCRIPT LANGUAGE="VBScript">
Function btnTest_onclick
    If Len(Document.frmTest.txtName.value) < 1 then
        Alert("You must enter a name!")
    End If
End Function
</SCRIPT>
```

事件处理过程、子程序或函数的脚本可以出现在页面的任何地方，因为它们只在需要的时候才处理。不过，通常是把这些类型的脚本放在页面的前头。

在你编写客户机脚本的时候，你可以访问页面上的对象，获得对象的特性，或者编写对象的事件处理程序。在工作中可以使用的对象准确列表取决于你的用户要使用的浏览器类型。更多内容，参见第二十四章“用设计期间控件和脚本控件编写脚本”和本章后边的“文档元素”。

客户脚本可以直接同客户相互作用，具体情况是，对于输出要发出消息框；对于输入就使用对话框或窗体。例如，如果用户在一种窗体中键入信息时产生一个错误，客户脚本就显示出一条出错消息（如前例所示）。

关于显示信息的更详细内容，参见第二十五章“用 HTML 元素编写脚本”中的“为用户显示信息”。关于使用客户脚本和服务器脚本窗体的更多信息，参见第二十四章“使用设计期间控件和脚本控件编写脚本”和第二十五章“使用 HTML 元素编写脚本”中的“用 HTML 窗体收集信息”。

服务器脚本处理

在 Visual InterDev 中，你把服务器脚本放到 Active Server Pages（ASP 文件）中。ASP 扩展版本告知 IIS：页面可以包含服务器脚本。当 IIS 读到该页面时，它便查找服务器脚本并处理这个脚本。当 ASP 文件中的服务器脚本被处理之后，便把这个脚本从文件中去掉，然后再把文件发送给浏览器（该文件中可能包含某种客户机脚本）。浏览器把这个 ASP 文件作为原来的 .htm 文件来对待。

在页面发送到浏览器之前，服务器可以对页面的任何方面进行修改。典型情况是要完成任务，再把任务的输出结合到页面的 HTML 文本中。但是服务器脚本可以较容易地创建客户脚本，因为客户脚本除了在这个页面上再添加一些文本就没有别的事了。

ASP 文件处理的一种特殊情况是 Global.asa 文件。这个文件所包含的文本要负责响应范围广泛的应用事件：每次 ASP 应用启动和关闭，和每次一个用户启动一个会话。

你可以在 Global.asa 文件中为这些事件创建服务器脚本，这对保存应用设置（例如默认脚本语言）、广泛应用变量初始化、计数器维护等一些任务都是很有用的。关于在 Global.asa 文件中为存储全局信息而创建事件句柄的细节，参见第二十五章“使用 HTML 元素编写脚本”中的“共享动态信息”一节。

当你在 ASP 文件中编写服务器脚本的时候，使用下述两种方式之一把这种脚本同其它文本区分开来：

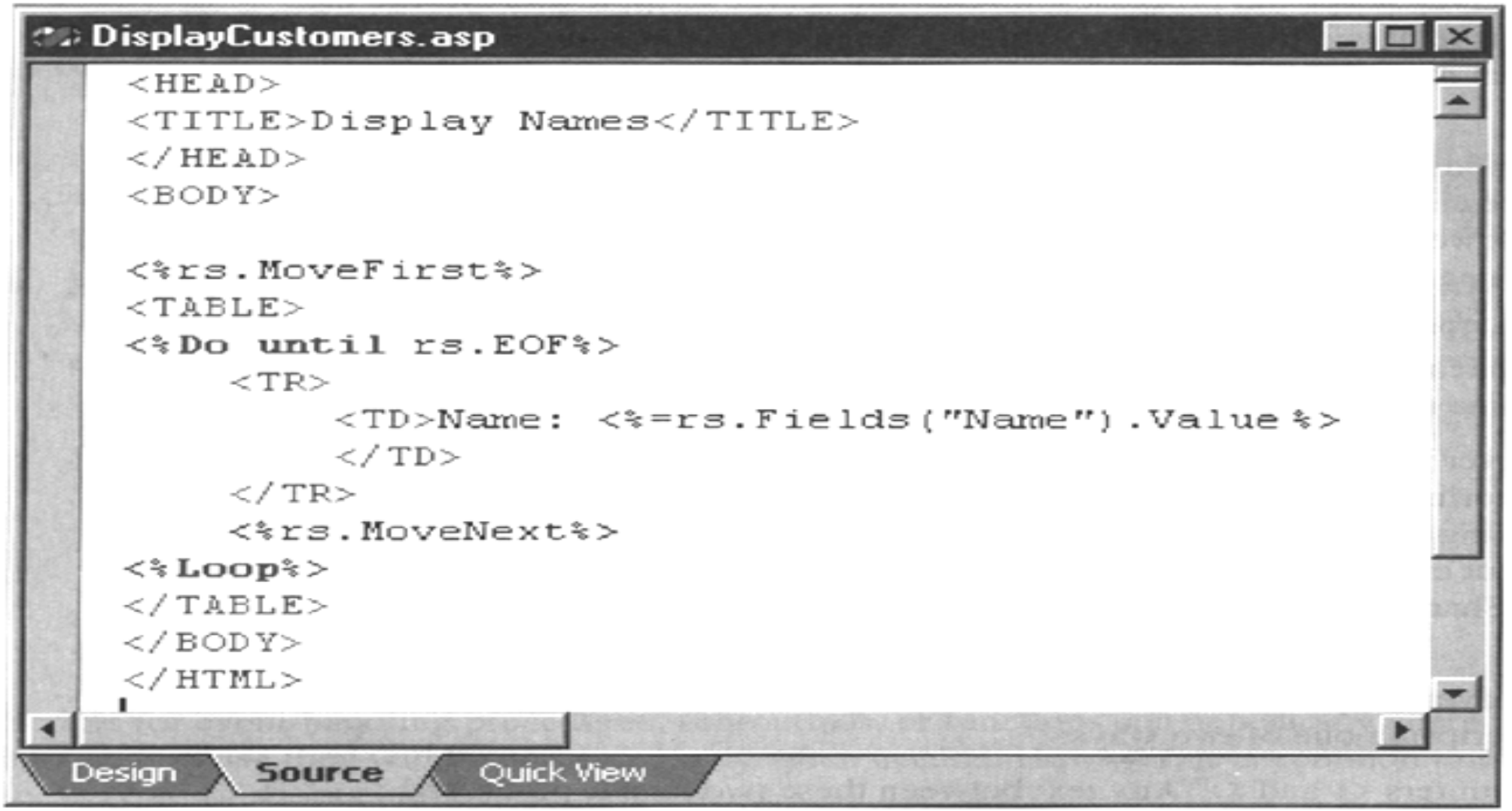
- 在分割符<%和%>内。这两个标记之间的任何文本是作为内联服务器脚本进行处理的。<% %>分割符用于放在计算表达式的两边，并且插入到页面的 HTML 文本中。例如，下述服务器脚本在页面上显示当前时间：

```
<% response.write time%>
```

- 在<SCRIPT>标记（连同客户脚本）中，带上 RUNAT=SERVER 属性，用于把单独的过程，例如函数和子程序，括起来。下边例子给出 RUNAT 属性：

```
<SCRIPT RUNAT=SERVER>
    Function GetDate
      [some script lines here]
    End Function
</script>
```

在 Visual InterDev HTML 编辑器中，服务器脚本呈现黄色，以便与交换机脚本区别开来。下边的图给出包括服务器脚本和客户脚本的一个页面。



编辑服务器脚本

服务器脚本通常不是事件驱动的，除了用设计期间控件创建的服务器事件处理程序和在 Global.asa 文件中创建的事件处理程序。在 ASP 页面被请求时，服务器读取页面并从上到下处理整个服务器脚本。这个脚本完成你编写的计算

和数据库访问，并且计算所有表达式和变量。按照需要，调用独立过程。

因为这个脚本在服务器上运行，因而它对服务器上可用的对象拥有访问权。例如，在 IIS 上运行的服务器脚本可以引用 ASP 应用、会话、请求和响应对象。然而，服务器脚本却不能使用浏览器中已经有的对象，例如，服务器脚本不能使用 Internet Explorer 文档或窗口对象。

在你编写服务器对象的时候，必须小心，要只使用服务器上下文中可用的对象。关于在服务器脚本中可以使用的对象的更多内容，参见本章后边的“文档元素”和第二十四章“用设计期间控件和脚本控件编写脚本”

如果服务器脚本产生一些输出，例如，如果你希望显示变量的数值或显示从一个数据库收到的一些记录，你可以使用内联脚本中的 Response.Write 方法（或使用缩写窗体，“＝”操作符）把输出放到该页面上。例如下边的简单页面说明服务器脚本计算当前时间，并把时间放到一个变量中。接着，在该页面上，服务器脚本变量的值被集成到一些 HTML 文本中：

```
<%
vDate = date
vTime = time
%>

<HTML>
<BODY>
When the script ran, it was <%Response.Write vTime%> o'clock on <%=vDate%>.
</BODY>
</HTML>
```

当页面被处理时，服务器便计算 Response.Write 或“＝”之后的表达式，结果值放到 HTML 页面流指定的位置。当在浏览器中显示这个页面时，看上去有些像下边这个样子：

When the script ran,it was 10:46:30 o’clock on 12/31/97.

如果你注意到页面的源代码，它看上去同该页面输出是一样的。你可能没看见源代码中的表达式<%Response.Write Time%>或<%-vDate%>，因为这些表达式是在页面送到浏览器之前由服务器计算的。

服务器脚本可以产生任何类型的输出，不仅包括变量和表达式的值，还包括 HTML 标记、文本以及事件客户脚本。

23.3 脚本对象模型

为了使 Web 应用的开发又快又轻松，Microsoft Visual InterDev 提供“脚本对象模型”(script object model)，这个模型通过引入常见面向对象编程模型用于对 HTML 和脚本编程的办法，简化 Web 应用的开发。这个模型大大简化了编写客户（浏览器）和服务器之间相互作用应用脚本的复杂性和数量：。

Visual InterDev 脚本对象模型以事件、特性和方法定义了一组对象，你可以用来创建你的应用或为应用编写脚本。你可以使用设计期间控件为你的应用创建可视界面，然后编写脚本，以便用熟悉的面向对象的技术控制应用。

在创建 Web 应用的时候，脚本对象模型允许你使用同你在诸如微软 Visual Basic 和微软 Access 环境中创建应用所使用的非常类似的方式。

如果你把脚本对象模型同使用 ASP 和 HTML 的结合形式创建 Web 应用情况比较，这种脚本对象模型是最容易理解的。例如要创建一种窗体，你就把 HTML 元素放到一个页面上，包括文本框、列表框和按钮。其中有一种按钮是典型的提交按钮，它可以产生向服务器送一种窗体的动作，并指定一个包含可以处理这种窗体服务器脚本的 ASP 页面。在目的页面上必须手动检查浏览器提交的状态，而且同创建这种状态的对象没有关系。

注意：在动态 HTML(DHTML)中，页面上的控件（和该页面本身）都是文档对象模型的组成部分，但是这个模型仅仅施加于客户脚本；这个模型不包括服务器方的处理。此外，这种 DHTML 文档对象模型只能由支持 DHTML 的浏览器所使用。

成为对照的是，这种脚本对象模型允许你以控件进行工作，也允许你以使

用标准面向对象技术的页面进行工作。例如，代替使用 ASP 和 HTML 所要求的复杂窗体提交过程，你可以简单地把一个按钮放到一个页面上，再为处理这种窗体的敲击方法编写一个处理程序。这种脚本对象模型分离出一些事件，例如敲击事件，这样你就可以在客户脚本或服务器脚本中为这些事件编写处理程序。

脚本对象模型的优点

这种脚本对象模型具有如下优点：

- **使用熟悉的面向对象模型进行快速应用开发**。你可以采用标准面向对象技术开发 Web 应用。因为你要编写的脚本较少且可以使用较简单的脚本，这样创建应用就快得多，错误也很少。
- **浏览器和平台彼此独立**。你可以使用这种脚本对象模型，而不管将访问你应用的浏览器如何。工作总是一样的，不管你设计的应用是用于服务器脚本的（最大范围）或是用于客户脚本（对用户是较好的试验，例如闪烁较少，到服务器往返次数也很少）。
- **简化的窗体**。你可以把设计期间控件拖放到页面上创建一种窗体，这种方式就同在 Visual Basic 环境中使用的方式相同。把控件拖放到页面上之后再使用属性改变它们的样子和表现。你可以通过编写标准事件处理程序的办法处理窗体；也可以通过调用方法来处理窗体：脚本对象模型处理复杂的窗体投递，并分派正确的窗体处理逻辑。
- **隔离应用逻辑**。你可以很容易地在不同过程中，其中包括其它页面上的过程中，把你的应用逻辑隔离开，而不是把这些应用同用户界面及导航元素混在一起。

- **状态维护**。脚本对象模型提供一种机制，按照在客户机和服务器之间传递的控件维持对象的状态。你不必手动管理隐蔽在字段中或会话中的状态。
- **数据绑定**。脚本对象可以绑定在数据库字段上，这样你就可以按照数据录入窗体和数据编辑窗体使用它们。
- **简化页面导航**。这种脚本对象模型提供一种机制，这种机制允许你导航到其它页面上使用名字而不是指定目标页面的 URL，这样就可以直接跳到那个页面上的任何过程上。
- **远程脚本**。你可以从一个页面上执行服务器脚本，而不是导航离开这个页面。在传统的客户/服务器应用中，这样就允许浏览器脚本向服务器要求一组结果，然后在浏览器中进一步计算或通过 HTML 改变这个页面。

脚本对象模型中的部件

在你使用脚本对象模型的时候，不管你编写基于服务器的应用(在 ASP 文件中)，还是编写可在客户浏览器上执行的脚本页面时(在.htmz 文件中)，你都使用相同的对象。

注释:脚本对象模型是使用存储在“脚本库”(Script Library)中的脚本实现的。不要改变库中的内容，否则脚本对象模型中部件将不能正确工作。

开启脚本对象模型

在使用脚本对象模型之前，你必须开启它，它为页面构造脚本对象模型构

架。

注释:Visual InterDev 设计期间控件需要脚本对象模型。如果你把设计期间控件加到尚未使脚本对象模型开启的页面上，Visual InterDev 就提醒你开启它。

为一个页面开启脚本对象模型

1.用右键单击对象或控件以外页面上任何地方，选择 Properties,然后选择 General 选项卡。

2.在 ASP settings 之下，选择 Enable scripting object model（开启脚本对象模型）。

Visual InterDev 编辑器把脚本对象模型框架加到顶部和底部的页面上。你不要改变这些页面的内容。

设计期间控件和脚本对象

要为你的应用创建用户界面，你可以使用 Visual InterDev“设计期间控件”(design-time controls)。设计期间控件是你在编辑器中进行工作创建 Web 应用功能时使用的控件。设计期间控件产生 HTML 文本、脚本，有时候还用其它部件在运行时实现你设计的功能。

Visual InterDev 设计期间控件包括标准用户界面元素：文本框、检查框、列表框、命令按钮等等，根据脚本对象模型产生运行期间文本。使用设计期间控件可以很容易地创建窗体，并使用脚本对象模型的特点优势绑定到数据上。关于设计期间控件的细节，参见 Visual InterDev 在线文档中“设计期间控件”

一节。

设计期间控件有两级相互作用。在设计期间，这些相互作用就像控件一样，你可以在诸如 Visual Basic 环境中把它们放到一种窗体上。你可以在 HTML 编辑器中同它们进行可视相互作用，设置它们的属性，规定它们的表现行为。然而实际目的是产生在你运行页面时可执行的脚本。因此，在你设置设计期间控件的属性时，你实际是改变设计期间控件产生的代码。

在设计期间控件产生的代码运行时，它动态产生一个“脚本对象”。这个脚本对象是你编写脚本的对象，而且在编写脚本时设置对象的特性，调用它的方法并响应它的事件。

在编辑器中工作的时候你为设计期间控件设置的特性不同于运行时响应脚本对象可用的特性。但是这些特性是相关的。例如，设计期间控件的 ID 特性是用来创建一个脚本在运行时的 ID 特性。

在设计你的用户界面的时候，你可以在属性窗口中或在客户属性页面上设置设计期间控件的属性。脚本对象属性不在属性窗口中显示出来，这是因为这些属性是运行时的属性。然而，在你编写脚本时，这些属性出现在 IntelliSense 语句完全下拉列表中。

此外设计期间控件没有方法或事件，因为这种控件从不在运行时工作。脚本对象有预定义的方法和事件。使用脚本对象的特性，你可以在 IntelliSense 语句完全下拉列表中看见方法和事件的一个列表。另外，你还可以在 Script Outline（脚本大纲）窗口中看见这些为一个脚本对象预定义的方法和事件。

在一些实例中，设计期间控件在设计时是可视的，但是相应脚本对象是不可视的。例如，用于控制数据访问的“记录集”（Recordset）控件在设计时是

可视的，因而你可以看见它的属性。但是在页面运行的时候，由“记录集”脚本对象创建的脚本对象却没有可视的部件，即便它像任何其它脚本对象一样有属性、方法和事件。

设计期间控件和脚本对象的优势之一是：不管你希望在服务器上工作还是在客户机上运行，使用它们都以相同的方式进行工作。例如，如果以许多不同浏览器为目标，你可以把控件的平台设置为“服务器”，并且使所有设计期间控件产生的代码都在这个服务器上运行。如果你的目标是把客户机作为平台，所产生的代码便在客户机上运行（当然你要保证这个客户机有运行它的能力）。但是，你同脚本对象的相互作用，例如调用它们的方法，一样是可视的。使用服务器脚本响应客户事件（例如敲击按钮）的复杂任务是建立在脚本对象之中的。

注意:组合窗体的一种轻松方式是使用 FormManager。有关细节，参见第五章“演练”中的“简化数据录入页面”和第三章“数据库基础”中的“建立事件驱动窗体”。

页面对象

脚本对象模型允许你用 Web 页面作为在你的脚本中可以引用的对象，就像任何其它对象一样。按照默认设置，当前页面可作为一种对象，并称作 thisPage。

页面对象可以从其它页面上引用。对于页面导航，尤其是如果你的应用要求你在其它页面上处理一个指定的脚本时，这种引用能力是很有用的。例如，在你的应用中，你可能要导航到其它页面上处理一个窗体，而导航到的其它页面上包含的脚本要进行验证、数据库更新及其它操作。在 HTML 模型中，你把你的窗体寄送给其它页面，然后在目的页面上编写脚本，这就需要确定如何调用

这个页面以及如何进行处理。若是使用页面对象，你可以在那个页面对象上调用一种方法，而脚本对象模型完成所有导航并为你进行分派。

例如，在一个称作 EmployeeList 的页面对象中编写称作 DoQuery 的功能，然后作为一种方法展示出来。从另一个页面，你可以像下边这样简单调用一种方法，执行这个功能：

```
EmployeeList.navigate.DoQuery(“Accounting”)
```

使用页面对象还可以提供远程脚本编写手段。远程脚本编写允许你在 ASP 页面上调用一个脚本功能进程（在服务器上运行）。完成这个调用不用离开这个页面就可以在客户脚本上进行（通常的方式是你要离开当前页面，在服务器上执行一个过程，即使这个过程在当前页面上也是一样）。因此客户页面保持了它的状态，包括在调用服务器之前脚本是在那里执行的。

因为远程脚本还可以异步执行，你可以提供丰富的用户经验。例如，你可以执行数据库查阅来验证用户录入，而同时用户可以用窗体继续进行工作。详见第二十四章“用设计期间控件和脚本对象编写脚本”中的“远程执行服务器脚本”一节。

文档元素

在编写脚本的时候，你可以操纵不同对象以便完成应用任务。例如，你可以在一个客户脚本中：

- 测试一个复选框，然后设置文本框的值，
- 导航到其它页面上，
- 当一个页面第一次显示的时候，显示 Java 小程序的一个特定图形，

- 通过把文本在整个屏幕上移动、重定文本或图形的大小等操作，创建多媒体效果。

在服务器脚本中，你可以用不同的对象进行工作，你可以按如下顺序操纵这些脚本：

- 获取用户录入到窗体中的信息，
- 同数据库建立一条连接，并对这条连接运行一次查询，
- 确定当前浏览器是否支持你应用所要求的性质。

在脚本中可以使用的对象取决于脚本所运行的上下文。如果是正在服务器脚本中工作，你只能使用服务器上可用的对象。相反，如果你在客户脚本上工作，则只能使用作为页面组成部分并且在浏览器中可用的对象，或者你知道存在于客户计算机上的对象。

注意：如果使用脚本对象模型和设计期间控件，你就可以使用同样的脚本对象在客户脚本中工作，或者在服务器脚本中工作。详见第二十四章“用设计期间控件和脚本对象编写脚本”。

客户对象

在客户脚本中你可以获得页面上对象的属性并调用它们的方法，为这些对象编写事件处理程序。例如下边小脚本就是使用窗口对象的更替方法和文档对象的标题属性显示当前文档的抬头：

```
<SCRIPT LANGUAGE=" VBScript " >
   alert(document.title)
</SCRIPT>
```

每个对象都有自己的方法和特性，你可以按照要求来使用它们。在客户脚本中，大多数对象也支持你可以编写处理程序的事件。例如，你可以为按钮敲击事件编写一个处理程序，这个事件在敲击按钮时出现。

在客户脚本中可以使用的确切对象和事件取决于你的用户要使用的浏览器中可用的对象模型。例如，在一些浏览器中，你可以编写一种脚本用于改变页面上的文本，但是不是所有浏览器都支持这种特性。一般说来，你可以依赖下列事项：

- 大多数浏览器(包括 Internet Explorer 3.0)至少支持 HTML 3.2 级。在 HTML 3.2 对象模型中，一般你不能改变页面上已经有的对象的样子。然而你可以为 HTML 控件编写事件处理程序，例如为按钮、窗体、Java 小程序、ActiveX 控件编写事件处理程序，以及为整个文档编写处理程序。下边的表中给出 HTML 3.2 中的可编写脚本对象的列表。
- 某些浏览器(包括 Internet Explorer 4.0)支持动态 HTML(DHTML)，运行时属性、方法和页面上赋名对象的事件三方面提供全面实施。你还可以为计时器事件编写处理程序，按照规定时间间隔移动文本或对象，或重定文本或对象的大小。DHTML 为你提供对 HTML 页面的控件级别，同你对 Visual Basic 中窗体的控制级别或在 Visual Basic for Applications(VBA)中对文档的控制级别，差不多是一样的。
- Visual InterDev 支持一种脚本对象模型和设计期间控件，允许你使用标准面向对象的技术为页面上控件编写脚本。详见第二十四章“用设计期间控件和脚本对象编写脚本“。你可以用任何浏览器使用这种脚本对象模型，只要你使用“Internet 信息服务器”作为你的服务器就行。

在你编写脚本的时候，一个比较好的办法是把你限制在你的用户浏览器所支持的对象模型上。例如，如果你知道你的大多数用户都使用 Internet Explorer 3.0（或支持 HTML 3.2 的另一种浏览器），你就不应当使你的应用依靠 HTML 特性。另一种办法是你可以编写你的应用，测试一种具体浏览器，然后根据用户拥有的浏览器展示特性。关于测试浏览器能力的内容，参见第二十五章“使用 HTML 元素编写脚本”中的“创建可移植脚本”。

对象	说明
窗口	浏览器对象，允许你确定有关浏览器信息、提示用户信息以及显示消息。两个常用的装载和卸载事件允许你在装载文档时完成对任务的初始化
文档	浏览器对象，允许你设置文档颜色，确定当前和推荐文档的 URL，获得文档抬头以及把文本写到页面中。关于编写文本的更多内容，参见第二十五章“使用 HTML 元素编写脚本”中的“显示用户信息”
窗体	浏览器对象，允许你确定窗体方法和动作的信息，并列举窗体中的元素。你可以为窗体的提交事件编写处理程序，以便指定在用户敲击提交按钮时出现什么事情。关于使用窗体的更多内容，参见第二十五章“使用 HTML 元素编写脚本”中的“用 HTML 窗体收集信息”一节 **提示**：Visual InterDev 提供一些设计期间控件和一个 FormManager 控件，允许你轻松创建窗体，而不用为 HTML<form>元素编写脚本。详阅第二十四章“用设计期间控件和脚本对象编写脚本”和第三章“数据库基础”中的“建立事件驱动窗体”

续表

元素	单个窗体元素，如按钮、文本框等等。你可以为元素事件编写处理程序，例如进行敲击、得到或丢掉焦点，和修改。更多内容，参见第二十五章“使用 HTML 元素脚本脚本”中的“用 HTML 窗体收集信息” **提示：**你可以使用 Visual InterDev 设计期间控件取而代之。详见第二十四章“用设计期间控件和脚本对象编写脚本”
Java 小程序，ActiveX 控件	把外部创建的对象加到页面上。小程序放到页面上<APPLET>块中，对象放到<OBJECT>块中。Java 小程序和 AcitveX 控件支持它们自己的一组属性、方法和你可以为之编写处理程序的事件

服务器对象

如果你正编写的脚本将在微软 Internet 信息服务器上运行，你可以使用这个服务器中内在的对象，例如服务器的 Request 和 Response 对象。你可以使用服务器绑定的部件，但不是它内在的部分，例如 AdRotator 部件、BbowserCapability 部件和 ActiveX Data Object(ADO)。最后，你可以使用在这种服务器中登记的任何其它对象，包括你创建的部件和登记用的寄存器本身。

使用服务器对象和部件是 Visual InterDev 强有力的属性。在你创建一个 Web 应用的时候，通常你不能控制用户拥有什么浏览器，也不能控制在用户计算机上登记什么。在服务器上安装控件并在服务器脚本中使用这些控件，你可以使得这些控件的特性被任何用户所使用，不管使用的是什么浏览器。

在你编写服务器脚本时，你不能直接操纵客户对象，例如浏览器窗口、HTML窗体、Java小程序或者DHTML对象，因为这些对象都不常驻于服务器上。

如果你用服务器的内在控件进行工作，例如Request或Response对象，你可以在你的脚本中简单地引用这些对象，如下例所示：

```
<%Server(“starttime”)=time%>
```

然而，对所有其它对象而言，你创建这个对象，可以使用一个<OBJECT>标记做这件事，而在这个标记中你指定属性 RUNAT=SERVEWR。创建一个<OBJECT>标记就允许你在任何服务器脚本中，在这个页面上引用该对象，把这个对象及其编号加到InlelliSense语句完全下拉列表上。例如，下边创建一个引用一个ADO连接对象的对象：

```
<OBJECT RUNAT=" Server " ID=cn PROGID=" ADODB.Connection " >
<OBJECT>
```

在你的脚本中可以使用你在ID属性中分配的名字引用这个对象。下边的语句使用在<OBJECT>元素中定义的对象：

```
cn.Open Application( " ConnectionString " )
‘ other processing here
cn.Close
```

另一方面，你可以使用该服务器的CreateObject创建对象，创建对象的实例如下边语句所示：

```
<%
Set Ad = Server.CreateObject( " MSWC.Adrotator " )
```

```
Ad`.GetAdvertisement(" /ads/adrot.txt " )
%>
```

下表列出常见服务器和部件

对象	说明
请求	内在 IIS 对象，通过一次 HTTP 请求对下列信息进行访问，这些信息包括窗体信息、搜索串、浏览器信息和存储在串存器（cookies）中的信息。参见第二十五章“用 HTML 元素编写脚本”中的“用 HTML 窗体收集信息”、“共享动态信息”和“创建可移植脚本”
响应	内在 IIS 对象，通过向 Web 页面流写入信息的办法把信息发送给用户。关于使用响应对象的详细内容和实例，参见第二十五章“使用 HTML 元素编写脚本”中的“条件导航”“共享动态信息”
会话，应用	内在 IIS 对象，允许你设置或获取 Web 应用中页面之间存在的值。更多细节，参见第二十五章“用 HTML 元素编写脚本”中的“共享动态信息”
服务器	内在 IIS 对象，允许你创建在服务器上登记的实例对象，包括绑定的部件和你创建的对象

续表

AdRotator，BrowserCapability 和 TextStream 和 NextLink	用 IIS 绑定的部件，允许你显示图像集的变更、存储和获得特定浏览器的信息、读出和写入文本文件，以及创建一条通过页面的有序路径
ActiveX 数据对象（ADO）	绑定的部件，连接数据库或查询数据库

脚本调试过程

你可以使用 Visual InterDev 调试器测试在 Visual Basic Scripting Edition(VBSript)和 JScript 中编写的脚本。调试 Web 页面同在传统开发环境中进行调试有以下几方面不同：

- 大多数 Web 应用都由在客户机上运行的脚本（在 .htm 文件中）和在服务器上运行的脚本（在 ASP 文件、Global.asa 文件和 .cdf 文件中）组成。
- 脚本可以使用不同语言。
- 脚本可以同 HTML 文本混合编写。
- 许多 Web 应用不仅包含脚本，而且包含 Java 部件，如小程序和 COM 对象。

Visual InterDev 允许你在所有这些情况中进行调试。你可以调试在 Internet Explorer 本地版本中运行的脚本

注意：在你进行调试的时候，建议不要使用 Internet Explorer 4.0 的 Active

Desktop 模式。

要调试在“Internet 信息服务器”（IIS）4.0 中运行的脚本，可以在你的计算机上运行这个调试器，并把调试器同该服务器中运行的一个脚本连接起来。如果该服务器软件正在另一个计算机上运行，你可以使用“远程调试”调试在这里运行的脚本。

注意：关于调试你 Web 页面上 Java 部件的有关内容，参见 Java 调试文档。

错误类型

调试就是查找错误。当你使用脚本进行工作时，你可能遇到下列各类的错误需要调试：

- syntax error（句法错） 在下列情况下出现：如键入错误关键字，多行命令忘掉关闭（例如 DO...LOOP）或者产生类似的错误。如果一个脚本包含一个句法错误，这个脚本便不能执行，而且错误信息立刻显示在浏览器上或服务器处理的页面上。

  注意：在某些编程环境中，句法错误是指预处理错误、编译错误和编译时间错误。
- run-time error（运行期间错误） 当命令企图执行一个非法动作时出现。例如，如果你企图用尚未初始化的变量进行计算，便会出现运行期间错误。如果出现运行期间错误，脚本要么停止，要么执行一个例外例行程序。
- logic error（逻辑错误） 在一个脚本执行的时候没出现句法错误和运行期间错误，但是结果不是所期望的结果。例如，脚本可能提示用户

键入一个口令，但是在键入错误口令之后仍允许访问应用。

用调试器进行工作

调试脚本的基本过程由下述任务组成：

- 在你的 Web 项目中运行你当前正工作的文档，开始调试；或者把调试器连接到正在浏览器中或者在服务器中运行的文档上，开始调试。你可以及时启动调试器响应脚本错误，这就是所谓“及时”(just-in-time)调试。
- 发出中断命令，停止脚本执行。你还可以在脚本中设置断点，在断点处调试器将自动停止。在你停止一个过程的时候，它的原代码就显示出来。
- 观察脚本的状态：检查变量或属性的值，列出运行过程（“调用堆栈”）。
- 控制单个语句或过程的执行（步进），观察你应用的效果，也观察变量值或属性值。
- 跳过（“跨过”（step over））或步进通过（“步入”（step into））当前过程所调用的过程。如果有多个过程或线程在活跃，你可以移到另一个过程中，再从那里接着进行。

  注意： 你不能在 HTML 编辑器中的“设计视图”（Design view）或“快速视图”（Quick view）中用调试器进行工作。要进行调试，应转换到“源视图”（Source view）中。

要想允许你完成上述任务，调试器就要包含下述命令和窗口：

- “进程对话框”允许你把调试器连接到浏览器中或服务器上运行的文档上。这个文档可以包含在不是作为 Visual InterDev 项目组成部分的页

面中。你可以调试在本地计算机运行着的脚本，或者把脚本连接到在远程计算机上运行的进程上进行调试，例如连接到在服务器上运行的 ASP 文件上。

- “运行文档”（Running Documents）窗口，允许你查看调试器可以使用的文档列表。
- HTML 编辑器的“源视图”（Source view） 显示你正在调试的脚本源代码或部件源代码。如果你正进行调试的脚本或部件是你当前 Visual InterDev 项目的组成部分，你就可以排除错误，然后重新运行文档，再一次进行测试。
- 在 Visual InterDev HTML 编辑器中，你可以设置和清除断点。在调试器到达断点之后，你可以使用“调试”（Debug）菜单上的命令步进到你脚本的各个行中。如果你到达你脚本中要调出另一个过程（一个函数、子程序或小程序）的某一点，你就进入(“步入”)这个过程或运行它(“跨过”),并且停在下一行上。在任何一点你都可以跳到当前过程的末点(“步出”)，前进到下个断点。
- 在“本地”窗口中，你可以看到当前过程中变量的值。你还可以规定希望调试器显示指定表达式的值，例如属性的值，这要在“观察”（Watch）窗口中组建“观察表达式”。
- 要设置和改变值，你就使用“立即”（Immediate）窗口。在这个窗口中你可以计算任何表达式，也可以键入脚本命令，看它们的效果。你还可以观察和改变“观察”窗口中的值。
- 使用“调用堆栈”（Call Stack）窗口，你可以移到任何当前激活的过

程中。

关于使用调试器命令和窗口的细节，参见 Visual InterDev 在线文档。

了解脚本处理

了解客户脚本如何处理，错误如何解决，有助于你成功地调试客户脚本。

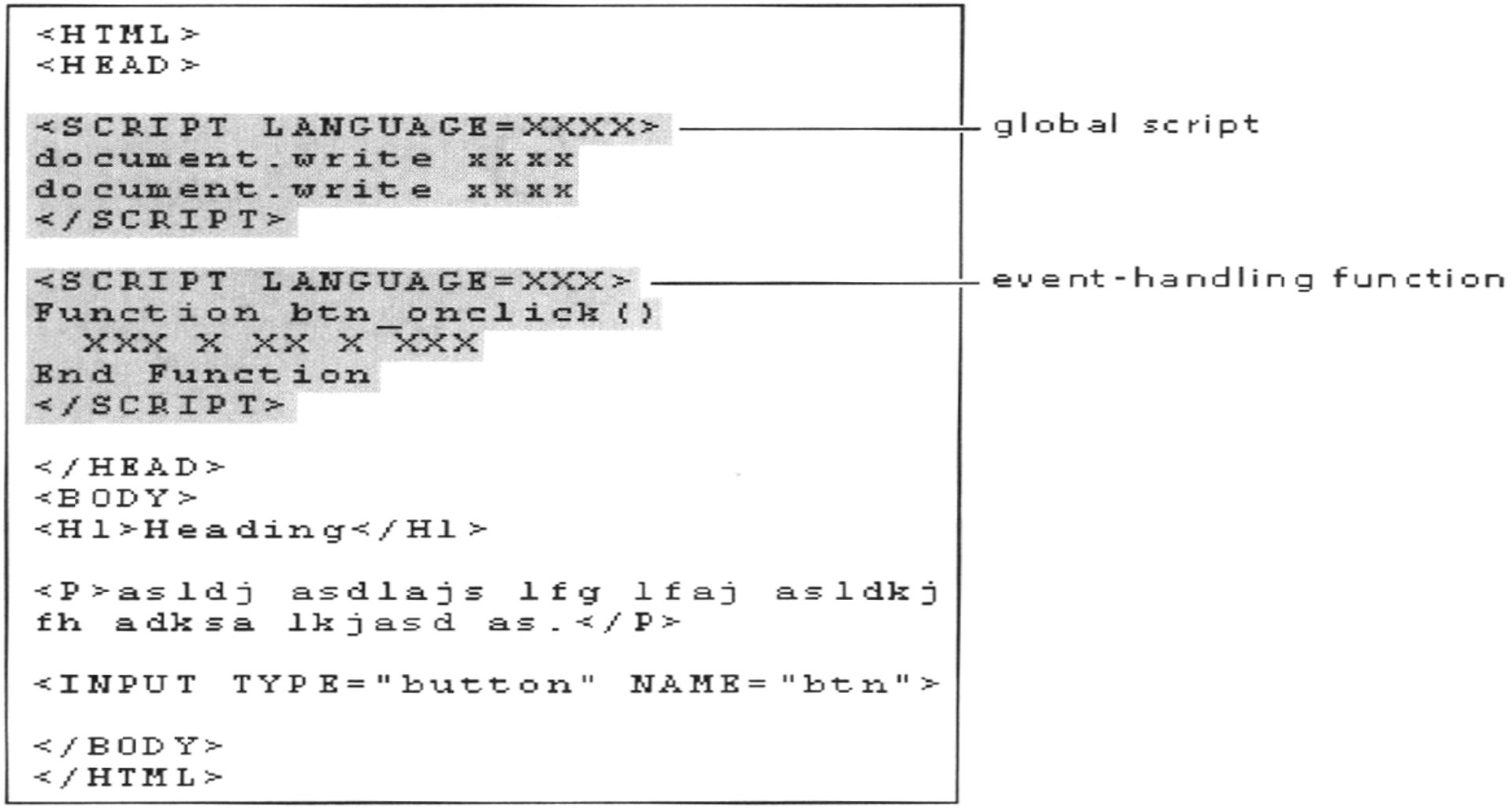
处理客户脚本

客户脚本是由 Internet Explorer 处理的。这个浏览器调用相应的运行期间模块，处理 VBScript 脚本或 JScript 脚本。

在 Web 文档装入到浏览器的时候，脚本要进行语法分析。在这个语法分析阶段，浏览器要报告它发现的任何句法错误。

在完成一段脚本的语法分析之后，浏览器就执行它。“全局”（Global）或“内联”（inline）脚本将立即执行。这两种脚本不是事件处理子程序或函数的组成部分。事件处理子程序、函数以及由其它过程调用的过程，都立即进行语法分析，但是在一个事件触发之前或被另一个过程调用之前都不执行。

Client script processing




```
<HTML>
<HEAD>

<SCRIPT LANGUAGE=XXXX>
document.write xxxx
document.write xxxx
</SCRIPT>

<SCRIPT LANGUAGE=XXX>
Function btn_onclick()
  XXX X XX X XXX
End Function
</SCRIPT>

</HEAD>
<BODY>
<H1>Heading</H1>

<P>asldj asdlajs lfg lfaj asldkj
fh adksa lkjasd as.</P>

<INPUT TYPE="button" NAME="btn">

</BODY>
</HTML>
```

客户脚本处理

如果在执行一个客户脚本时出现运行期间错误，则显示出一条错误消息，并且含有错误的脚本停止下来，但文档中的其它客户脚本可以继续运行（除非

你启动调试器）。如果重新调用含有错误的脚本，错误消息就再一次显示出来。

根据你使用的语言，你可以在脚本中包含一些语句以便俘获运行期间错误，并运行你自己的错误过程。例如在 VBScript 中，你可以使用 ON ERROR 语句建立自己的错误陷井。有关细节，参见你的脚本语言文档。

处理服务器脚本

大多数服务器脚本都不是事件驱动的。在 ASP 文件被请求的时候，服务器读出页面，并从头到尾处理所有服务器脚本。这个页面包含属于内联的脚本，其中带有 HTML 文本。如下页图所示。

服务器脚本处理

不是所有服务器脚本都立即执行。同客户脚本一样，服务器脚本可以包含函数和子程序，这些功能和子程序只在被其它过程调用时才执行。

Global.asa 文件是一种特殊情况。在这个文件中的 Application_OnStart

和 Session_OnStart 过程只对一个应用或一个会话执行一次。因此，要想使调试这些事件容易一些，你就必须在文件中嵌入调试语句。有关细节，参见第二十六章“调试页面”中的“调试 Global.asa 文件”一节。

第二十四章　用设计期间控件和脚本对象编写脚本

要为你的应用创建用户界面，你可以使用 Visual InterDev 脚本对象模型和 Visual InterDev 设计期间控件。Visual InterDev 设计期间控件包含标准用户界面元素：文本框、复选框、列表框、命令按钮等等。

在你的 Web 页面运行的时候，设计期间控件以脚本对象模型为基础创建脚本对象。你可以为应用添加功能，办法是编写一些脚本，使你编写的脚本同这些脚本对象一起工作。

24.1　用设计期间控件创建窗体

使用脚本对象模型的主要优点之一就是简化处理 Web 页面窗体中信息所需要的脚本。借助于使用脚本对象模型和设计期间控件，你可以为你的应用创建用户界面，而且与你在 Visual Basic 和 Access 中创建窗体所使用的方式非常相似。

你可以使用下表列出的设计期间控件创建用户界面：

Button	Textbox	Listbox
Label	Grid	Checkbox
OptionGroup	Recordset	RecordsetNavBar

注意:设计期间控件是用存储在 Script Library(脚本库)中的脚本实现的。不要改变这个库中的内容,否则控件不能正确工作。

选择目标平台

在把控件加到你的页面上之前,你必须决定你的应用将使用什么平台:服务器还是客户机。你的选择将决定设计期间控件如何创建,以及事件在那里出现。请参考下表内容。

目标平台	结果	如果选中
服务器	● 你的页面是 .asp 文件。 ● 脚本对象在服务器脚本中创建。它的属性、方法和事件只在服务器脚本中可用。 ● 脚本对象模型事件（例如敲击）在服务器脚本中处理。更多内容，参见本章后边“为脚本对象编写脚本”。 ● 数据绑定出现在服务器上。	你的应用将可由各种浏览器使用。优点包括： ● 应用在所有浏览器上同样运行。 ● 你希望可以对应用逻辑和数据库连接串信息的访问进行控制。
客户机	● 你的页面可以是一个.asp 文件或一个.htm 文件 ● 使用客户脚本创建脚本对象。它们的属性、方法和事件只在服务器脚本中可以使用。 ● 事件在客户脚本中处理。更多内容，参见本章后边“为脚本对象编写脚本”。 ● 数据绑定出现在客户机和服务器两个地方。	应用将只能由支持动态 HTML(DHTML)的浏览器，如微软的 Internet Explorer 4.0，所使用。典型的优点是用户使用方便。此外，你的应用可以使用 DHTML 对象模型。

了解目标平台的重要性不仅仅决定脚本对象在哪里创建，还决定你如何为

它编写脚本。例如，对某个控件的平台是服务器，它相应的脚本对象就在服务器脚本中创建。你只能访问服务器脚本中的脚本对象属性、方法和事件，脚本对象对客户脚本而言是不可用的。相反情况也是对的：如果控件的目标平台是客户机，相应脚本对象的属性、事件和方法，在服务器脚本中是不可用的。

选中一种目标平台并不意味着所有设计期间控件都为这种平台生成脚本。某一个控件的脚本平台，和某个脚本的脚本平台，取决于该控件是如何使用的。例如，对窗体中 Save 按钮的目标平台可以是服务器，于是它的脚本就可以把信息保存在一个数据库中。但是，另一个脚本(甚至对同一个按钮)，可以在客户机上运行，以便在数据保存以前对数据进行验证。

你可以为单个页面选中一个目标平台，或者使用项目的默认平台。

为页面指定目标平台

1. 转换到 Source 视图上。
2. 右键单击该页面，选中 Properties。
3. 在 Property Pages 窗口中，选中 General 选项卡。
4. 在 DTC scripting platform 下，选择 Server 或 Client。

注意：如果你改变目标平台，已有的事件处理脚本不能自动改变以反映新的设置，但是你可以手动做这件事。例如，当把目标平台从服务器改成客户机时，服务器<SCRIPT RUNAT=”Server”>块中的事件处理程序需要移到客户机的<SCRIPT>块中。详见本章后边的“为脚本对象编写脚本”一节。

如果你希望在任何时间都使用同一种平台，你可以为项目设置默认目标平

台。此后你创建的所有页面都将反映你的默认设置。

为项目指定目标平台

1.在 Project Explorer 中，右键单击项目，然后选择 Properties。
2.在 Property Pages 窗口中，选中 Editor Defaults 选项卡。
3.在 DTC scripting platform 下，选择 Server 或 Client。

注意:如果你已经有了一些页面，上边有设计期间控件，改变项目默认并不改变对这些页面的设置。

按照默认设置，页面上的所有设计期间控件都从当前页面遗传给目标平台。然而你可以按单个控件改变目标平台。例如，你页面上的一些设计期间控件可以绑定到一个数据库上，并用服务器作为目标平台。其它控件不能绑定数据并使用客户机作为目标平台。

注意:数据绑定设计期间控件必须使用同一个目标平台作为要绑定的记录集对象。

为单个控件指定目标平台

1.转换到 Source 视图。
2.右键单击该控件，然后选中 Properties。
3.在 Property Pages 窗口中，选中 General 选项卡。
4.在 scripting platform 下，选择 Client 或 Server。

向页面添加控件

你可以使用拖放办法把设计期间控件加到页面上，然后设置它们的属性。

添加设计期间控件

1.保证你设置的选项可以查看图形控件。从 View 菜单中选中 View Controls Graphically。

注意:想把这个选项设置为默认选项，要使用 Options 对话框中的 HTML 节点。

2.从 Toolbox 的 Design-Time Controls 选项卡把设计期间控件拖到你的页面上。

如果对某个页面的目标平台是服务器，而脚本对象模型还没有对这个页面开启，则提示你开启它。你必须对设计期间控件开启脚本对象模型，工作才能正常进行。

注意:不要使用 HTML<FORM>标记定义窗体。脚本对象模型为你创建一种窗体。

3.右键单击该控件，然后选中 Properties。出现该控件的客户 Properties 窗口。

4.按要求为控件设置选项。

注意:当你在 HTML 编辑器中打印一个页面时，某些设计期间控件可能不显示你所设置的值。如果你在一个页面上放上多个设计期间控件例子，例如，如果你把多个 Textbox 或者 Button 控件放到一个页面上，第二个控件和后边的控件将打印显示它们的默认值。

按照默认设置，设计期间控件使用它们的图形表示形式在 Design 视图和 Source 视图中显示出来。例如 Button 设计期间控件就作为按钮显示出来。

你可以选择以文本形式观察控件。然而，这种控件在文本模式中的时候不

能同页面上的其它控件进行通信，这就使得该控件不能正常作用。

注意:ASP 页面不能正常出现在“快速”(Quick)视图中，因为“快速”视图不运行服务器脚本。因而，如果脚本对象的目标平台是服务器，“快速”视图就不允许你预览脚本对象。

你可以切除、复制和粘贴设计期间控件。然而，只有在复制控件的图形版本而不是文本版本的情况下，你才能成功地做这些事。

为了便于规定设计期间控件的选择，可使用客户特性页面来设置它们的属性。

设置设计期间控件的属性

- 右键单击该控件，然后选中 Properties。

  或者

  在 Properties 栅格中，移到(Custom)，然后单击该属性数值段中的按钮。

每个设计期间控件都支持不同的选项。要了解你可为其设置属性的细节，请按 Property Pages 对话框中的 F1 键。

设计期间控件在页面上产生脚本，以实现相应的脚本对象。你可预览产生的文本。

预览产生的文本

1. 转换到 Source 视图。
2. 右键单击设计期间控件，然后，选中 Show Run-time Text。

你可以永久地把设计期间控件转换成运行期间文本，以便离开所需要的脚本去创建脚本对象，但舍去用于同其它控件进行通信的信息以显示图形控件。

如果你希望定制所生成的文本，你就可能这样做，但是这不是使用设计期间控件进行工作的推荐方式。

把设计期间控件永远转换成文本

- 在 Source 视图中，右键单击设计期间控件，然后选中 Convert to Run-time Text。

告诫：转换成运行期间文本是一种单向操作。你不能反向操作返回到图形视图。只有你感到定制成控件运行期间文本很方便的，你才应进行这样的转换。

用设计期间控件创建窗体

在使用脚本对象模型和设计期间控件的时候，不要使用 HTML<FORM>标记创建窗体。相反，你要把你需要的设计期间控件拖到页面上。

提示：使用设计期间控件组建一种窗体的轻松方式是使用 FormManager。有关细节，参见第五章“演练”中的“简化数据录入页面”和第三章“数据库基础”中的“建立事件驱动窗体”。

要想对单个控件定义特定的行为，你可以在脚本中编写事件处理程序。不要添加 HTML Submit 按钮，而是添加 Save 按钮去调用窗体处理方法，如下边 VBScript 例子所示：

```
Sub btnSave_onclick
    ProcessForm()
End sub
```

窗体处理过程可以在页面的任何地方。可以用请求页面上控件数值的办法从窗体中提取数值。下边的例子测试称为 txtName 文本框的数值：

```
Sub ProcessForm()
  If txtName.value = “” then
     TxtName.value = “Please enter a name!”
  Else
     ‘ etc.
  End if
End sub
```

24.2 为脚本对象编写脚本

使用脚本对象模型和设计期间控件的主要好处之一是便于编写脚本以定义应用的行为。脚本对象模型使得你的 Web 页面使用起来就像 Visual Basic 或 Access 中的窗体一样。把设计期间控件拖到页面上之后，你就可使用熟悉的脚本技术设置它们的属性，并为事件编写处理程序。

但是用脚本对象模型进行工作和在其它环境中进行工作还是有一些不同。在下边的几小节中，将介绍如何为脚本对象编写脚本，其中包括与其它环境的差别。

为目标平台编写相应脚本

目标平台规定了脚本在哪里运行，因而也就决定了你的脚本可以做什么。当目标平台是服务器的时候，你可以使用脚本对象模型和 ASP 编程模型，其中包括 Microsoft Internet Information Server (IIS)（Internet 信息服务器）对象。相反，如果你的目标平台是客户机，这个脚本对象模型就用动态 HTML(DHTML)扩展文档对象模型。

若编写脚本，你必须清楚它在哪里运行，你不要企图使用与这些上下文无关的过程。例如，如果你的目标平台是服务器，就不要试着使用类似于 MsgBox 功能或使用告警方法为用户直接显示消息。即便这些功能可以正常工作（一般而言它们都会产生错误结果），它们还是把消息显示在服务器上，而不是显示给用户。

改变目标平台

如果目标平台是服务器，脚本块的 RUNAT 属性(attribute)将设置为 SERVER。如果目标平台是客户机，则没有 RUNAT 属性。

如果你对一个页面改变目标平台，则必须手动加上或去掉影响脚本块的 RUNAT 属性。例如，如果目标平台设置为客户机的情况下，你一开始就把设计期间控件加到一个页面上，脚本块看上去可能是这样：

```
<SCRIP LANGUAGE="VBScript">
```

如果接着把目标平台改变为服务器，你就必须加上一个 RUNAT 属性，如下边例子所示：

<SCRIP LANGUAGE="VBScript" RUNAT=SERVER>

这就可能不仅仅是调整问题，还须了解其它特定平台的特性，其中包括：

- 客户脚本可以为用户显示消息。如果你把平台转换成服务器，你必须去掉或更换 MsgBox 或告警呼叫。
- 客户脚本可以从 DHTML 对象模型中引用对象，包括窗口、文档和其它。如果你的脚本使用这些对象，你必须在改变成服务器平台时发出告警。
- 服务器脚本必须用默认页面语言（在页面顶部有@LANGUAGE=attribute 规定的语言）编写，或者必须在<SCRIPT>块中包含一种 LANGUAGE 属性。
- 服务器脚本不能访问某些调用返回的值，例如 onkeyup 事件返回的值。

测试脚本对象

要在合适的位置测试你的页面。HTML 编辑器中的 Quick View（快速视图）在本地运行，因而它不处理服务器脚本。所以，如果你的目标平台是服务器，就不会有脚本对象出现在 Quick View 中，而应使用浏览器从 Web 服务器查看页面。

使用浏览器进行测试

- 在 Project Explorer 中，右键单击文件的名字，进行测试，然后选择 View in Browser。

引用脚本对象和属性

在你的脚本中，你可以使用在创建设计期间控件的时候为脚本对象分配的

名字引用脚本对象。你可以像通常那样读写大多数属性，并把它们的名字带上圆点(.)加到对象引用子上。

例如，如果你已经把一个 Textbox 设计期间控件拖到页面上，并且为它起名叫做 txtName，你可以通过获得其特性值的办法读到其当前内容，如下所示：

```
<SCRIPT LANGUAGE=”JAVASCRIPT”
function getlName
{
    fname = txtName.value;
}
</SCRIPT>
```

注意:对象和特性，在 JScript 和 JavaScript 中是区分大小写的。

按照默认设置，在你键入脚本对象名字并在后边跟一个圆点(.)的时候，IntelliSense 的弹出将显示与你工作使用的脚本对象相对应的所有属性和方法。

当脚本对象模型开启的时候，你可以使用对象名字 thisPage 引用当前页面；或者，如果你已经指定它作为页面对象，就可以使用其页面对象名字。例如，下列语句设置当前页面的 cancelEven 属性值：

```
thisPage.cancelEvent=true
```

注释:thisPage 对象不出现在 Script Outline(脚本大纲)窗口中或 IntelliSense 下拉菜单中，除非你把 PageObject 设计期间控件加到页面上。但是，这个 thisPage 对象在运行期间是可用的，并且你可用对它编写脚本，甚至你没有使用 PageObject 控件也是一样。

某些值按照成对的方法进行访问：一个获取方法，获取特性值；一个写入方法，写入属性值。例如，你可以检查 Checkbox 脚本对象的状态，办法是调用其 getCheck 方法，并使用 setCheck 方法设置它的值。下边的例子说明这种属性的类型：

```
Function toggLeCheckBox
  If checkbox1.getChecked() then
     Checkbox1.setChecked(0)
  Else
     Checkbox1.setChecked(1)
  End if
End Function
```

对应每个脚本对象所支持的属性列表，参见 Visual InterDev 在线文档中的“Script Object”（脚本对象）。

调用方法和功能

调用脚本对象的方法所使用的方式同你对待任何脚本都是一样的。例如，如果你把 Listbox 设计期间控件拖到页面上，你可以通过调用其 addItem 方法的方式植入它，如下列脚本所示：

```
<SCRIPT LANGUAGE="VBSCRIPT">
Function LoadListBox()
    ListBox1.additem "Paris"
    ListBox1.additem "London"
    ListBox1.additem "Cairo"
End Function
</SCRIPT>
```

注意：JScript 和 JavaScript 对方法名字是区分大小写的。

响应脚本对象事件

每个脚本对象都可以产生一组预先确定的（或隐含的 implicit）事件。例如，对应于 Button 设计期间控件的脚本对象可以产生一个敲击事件，而对应于 Textbox 设计期间控件的脚本对象可以产生一个 onchange 事件。

注意:你可以把这组事件加以扩充，使一个对象给予响应，这样就可以使用浏览器产生的任何事件。详见本章后边“扩展一个对象的事件”一节。

要想为脚本对象编写处理程序，就要使用对象的名字和要处理的事件创建一个过程。你可以使用你的应用使用的浏览器所支持的任何语言编写事件处理程序。

例如，如果你创建称作 btnDisplay 的按钮，你可以对它的 onclick（敲击）

事件编写句柄，在 VBScript 中，处理程序看上去像下边的样子：

```
<SCRIPT LANGUAGE="VBSCRIPT">
Sub btnDisplay_onclick()
    TextBox1.value = "Button has been clicked"
End Sub
</SCRIPT>
```

提示：使用“脚本大纲”窗口创建事件处理程序。在“脚本大纲”窗口中，为你所使用的对象扩展节点，然后双击你希望编写处理程序的事件名字。有关细节，参见 Visual InterDev 在线文档中的“脚本大纲窗口”。

注意：脚本对象事件只在响应用户一个动作时才出现，在程序改变时不出现。例如，用户在 Listbox 脚本对象中选择一项，onchange 事件就激发。然而，如果你使用脚本语句改变选择，则不激发事件。

设计期间控件的一个重要特性是你可以根据目标脚本平台的位置(可以在客户机上也可以在服务器上)为脚本对象编写事件处理程序。在用户单击一个按钮的时候，事件实际上是在客户机上出现。然而，如果你的目标脚本平台是服务器，你在服务器脚本中编写处理程序，响应那个事件，就好像事件在服务器上出现一样。

首先事件被捕获，然后同其它脚本对象值信息一起，作为邮递内容的组成部分发送出去。然后，在服务器处，辅助脚本对象模型过程决定要处理一个事件，并调用你的事件处理程序。在服务器脚本完成的时候，被刷新的页面送回到浏览器中。在服务器脚本中每处理一次事件，就执行一次类似的循环。

对大多数情况，这个过程对用户和对你的脚本都是不可见的，但下列情况除外：

- 在服务器脚本中处理事件比在客户机脚本中要慢，因为涉及到来往于服务器。
- 你必须知道事件处理程序在哪里运行，不要企图使用不在服务器中或不在客户机上工作的过程。

扩展一个对象的事件

每个脚本对象有一组预先确定的或隐含的可响应事件。但是你可能希望发挥其它事件的优势，这些事件指由浏览器产生的并由你的脚本对象使用的事件。

最典型的情况是，如果你使用设计期间控件，并且你的目标脚本平台是客户机，你可能希望发挥在 DHTML 文档对象模型中可用事件的优势。

例如，textbox 脚本对象支持一种隐含的你可以为其编写处理程序的 onchange 事件，然而，在客户机上，你还可能希望为 onkeypress 事件或其它事件编写处理程序。

你可以扩展这组对一个对象可用的事件，办法是“通告”（advising）一个事件，或在事件出现的时候登记一个被通告的对象。在你通告一个事件之后，你可以为那个事件（或对象）编写事件处理程序，就像你对任何其它对象一样。你可以用撤销对事件通告的办法注销事件通告。

每个对象都支持一种通告方法，从而允许你登记一个特定事件。在你发出通告的时候，你指定事件名字和一个功能名字。这个功能是在该事件出现的时候要调用的，实际上是用于这个事件的事件处理程序名字。

提示：一般说来，通告事件只有在你的脚本目标平台是客户机时才是现实的。如果你的平台是服务器，并且你通告诸如 onkeypress 之类的事件，每次事件出现都会往返一次服务器（在这种情况下，每次都击键）。

在任何时候你都可以通告一个事件或撤销通告一个事件。通常是在装载一个页面的时候发出通告或撤销通告。对应客户脚本目标平台而言，你为窗口对象的装载事件创建一个处理程序，并在这里调用通告方法。

下边的例子说明如何在页面开始的时候发出通告，把 DHTML onkeypress 事件发送给称作 Textbox1 的文本框。当这个 onkeypress 事件激发并发送给 Textbox 的时候，将调用 checkkeys 功能。这种通告方法的结果是把称作 objAdviseTextbox1 的对象通告出去，如果你过一会儿需要撤销这个事件通告的话，你还要用到这个对象。

```
<SCRIPT LANGUAGE="VBScript">
Function window_onload()
   objAdviseTextbox1 = Textbox1.advise("onkeypress", "checkkeys()")
End function
</SCRIPT>
```

你在通告方法中指定的功能，工作形式很像任何一个事件处理程序。如果该事件传递参数，你便可以使用 DHTML 对象事件方法得到这些参数。下边示出在前边的例子中 onkeypress 事件的处理程序。这个处理程序检查在 Textbox1 对象中出现每次击键，并且只把击键数复制给对象 Textbox2。

```
Function checkkeys()
    character = Chr(window.event.keycode)
    If character >= "0" and character <= "9" then
       Textbox2.value = Textbox2.value & character
    End if
End function
```

在你不再需要该事件的通告时，就调用对象撤销通告方法注销它。这种撤销通告方法需要通告方法返回通告对象和事件名字。下边示出调用撤销通告的例子。

Textbox1.unadvise(“onkeypress”,objAdviseTextbox1)

俘获客户机事件

如果页面的目标平台是服务器，事件处理程序便作为服务器脚本来执行。然而，在一些实例中，你可能希望俘获事件，即在事件传递给服务器之前先处理客户脚本中的事件。例如，你可能希望俘获客户机脚本中的 Save 按钮的击键事件，以便在发送给服务器之前验证用户信息。

俘获事件只在目标平台是服务器的情况下才是有用的。如果事件已经开始在客户机上处理，就不需要俘获这个事件，因为它也不需要传递给服务器。

俘获服务器上的事件

- 为当前页面的 onbeforserverevent 事件编写处理程序。这个事件激发

之后立刻转发到服务器。该处理程序的语法如下：

function thispage_onbeforeserverevent(*objectName. EventName*)

objectName 参数中包含激发该事件对象的名字，而 eventName 参数包含要转发到服务器上的事件名字。

注销服务器上的事件

- 把 cancelEvent 特性设置为真。

下边是针对 Delete（删除）按钮说明如何俘获按钮敲击事件的例子。该脚本提示用户在进行下一步之前要确认是否删除。

```
<SCRIPT LANGUAGE="Javascript">
function thisPage_onbeforeserverevent( obj, event ){
if (obj=="btnDelete"){
        if(evnt=="onclick"){
            if (confirm("Are you sure you want to Delete?")){
                alert("Deleted per your request");
            }
            else {
                alert("Delete cancelled");
                thisPage.cancelEvent = true;
            }
        }
    }
}
</SCRIPT>
```

转变参数值

一些事件处理程序和方法用事件调用来接收参数。如果你的目标脚本平台是服务器，并且过程调用是到服务器上往返一次的结果，那么参数数据类型就转换成字符串。到服务器上往返一次，会在下述情况中出现：

- 客户机上的一个动作（例如敲击按钮）是在服务器脚本中处理的。
- 客户机脚本中的过程调用在服务器脚本中实现的一种方法。
- 你在另一个页面上调用一个页面对象方法。有关细节，参见下一节“跨越页面扩展脚本对象模型”。
- 客户产生远程脚本调用，去调用 ASP 页面。有关细节，参见本章后边的“远程执行服务器脚本”。

在上述任何一种情况中，接收参数的过程都必定把参数转换成所需要的相应数据类型。例如，布尔值转换成字符串“真”和“假”。如果你编写一个接收布尔参数的一个过程，你应当使用如下所示的脚本对它进行测试：

```
Sub TestValue( boolFlag )
    dim flag
    If boolFlag = "true" then
        flag = True
    Else
        flag = False
    End If
    If flag then
        ' processing here
    End If
End Sub
```

脚本对象实例

下边示出包含属性、方法和事件的完整脚本块实例。这个页面是一个简单的计算器，带有两个文本框，分别叫做 txtNumber1 和 txtNumber2，用户向文本框中键入数字。用户从 lstOperators 中选择一个操作符。当用户敲击 btnCalculate 的时候，计算结果就显示在 lblResult 中，如下所示。

```
<SCRIPT RUNAT=SERVER LANGUAGE=VBSCRIPT>
Sub thisPage_onenter()
   If thisPage.firstEntered then
      txtNumber1.value = ""
      txtNumber2.value = ""
      lstOperators.addItem "+", 10
      lstOperators.addItem "-", 20
      lstOperators.addItem "÷", 30
      lstOperators.addItem "x", 40
      lstOperators.selectByValue(10)
   End if
```

24.3 跨越页面扩展脚本对象模型

使用设计事件控件和脚本对象模型，允许你使用面向对象的标准技术创建页面并编写脚本。然而，Web 应用在一个关键性方面不同于其它开发环境：这些应用通常都建造一种页面集合。

在一个简单应用中，你可以把一些链接简单加到其它页面上。当用户敲击一条链接时，浏览器将顺着链接读取页面，并显示这个页面。然而对更复杂的应用，须使用更加完善的链接技术。例如你的应用可能要求链接：

- 有条件地跳到不同页面上。
- 在用户另一个页面的时候处理这个页面上的指定脚本。
- 在多个页面上导航，为你的应用提供服务器处理（例如数据库查询）。

脚本对象模型跨越页面进行扩展，帮助你用这些类型需求来设计应用。你可以使用脚本对象模型设计页面作为“页面对象”（page objects）。

页面对象是一种 ASP 页面，上面包含你在应用中使用的服务器脚本。这个页面上的过程，即函数或子程序，可以变成页面对象的方法。

例如，你可能在应用中有一个 ASP 页面，用一种窗体调用这个页面以显示雇员列表。目标页面上的过程为显示这个列表建造不同的查询：按不同顺序、使用不同字段，或者使用不同选择依据。当你把这个页面转换成页面对象的时候，你可规定这些过程中的一种过程作为一种方法。然后可从应用的其它页面上激活调用这些方法。

页面对象还允许你创建属性，这些属性维持到服务器上多次往返的状态。

页面对象为你提供：

- 简化的导航。可以使用标准对象引用子在应用中的其它页面上进行导航，不必追踪页面的 URL。
- 在另一个页面上执行指定脚本的简易方式。借助于把过程作为方法来输出，你可以直接跳到另一个页面的指定过程上，不用编写脚本去对隐蔽窗体元素或查询字符串进行语法分析。
- 维护状态信息的手段。你可以在一个页面对象上定义属性，用来规定页面寿命、会话或应用期间维护它们的值。
- 从浏览器中显示的一个页面上执行服务器脚本的方式。

创建一个页面对象的基本步骤是：

- 指定一个页面为一个页面对象，
- 把过程作为方法输出到页面上，
- 定义该页面对象将支持的属性，
- 引用在你的脚本中你将使用的任何其它页面对象，这些对象可能在该页面上，也可能在其它页面上。

把页面规定为对象

你可以把任何 ASP 页面规定作为页面对象。为此，你要使用页面对象设计期间控件。

注意：页面对象是用存储在“脚本库”（Script Library）中的脚本实现的。不要改变这个库中的内容，否则页面对象会工作不正常。

规定一个页面作为一个对象

1. 在 HTML 编辑器中创建或打开一个 .asp 文件。
2. 为这个页面开启脚本对象模型。
   要保证你已经把选项设置为观察图形控件。从 View 菜单中，选择 View Controls Graphically。要想把这个选项设置为默认值，便使用选项对话框中的 HTML 节点。
3. 从 Toolbox 的 Design-Time Controls 选项卡上，把 PageObject 控件拖到你的页面上。你可以把这个控件拖到该页面的任何地方，但是它必须在脚本对象模型块的框架之中。
4. 在 PageObject 控件上的 Name 框中，键入这个页面对象的名字。这便是你可以用来引用脚本中对象的名字。

你给出页面对象的名字被登记在 Visual IntrDev 项目中，因此这些名字在任何其它页面上都可以使用。甚至可把这个页面移到其它位置，它的页面对象名字仍然不变。

过程作为方法输出

使用页面对象的主要好处之一是你可以把应用任务作为可以轻松地从其它脚本调用的方法来输出。这样做可大大简化代码的结构，消除在页面之间传送变量所需要的时间和工作，分配相应的例行程序.....。

页面对象支持两类方法：

- 导航方法（Navigate methods）是由一个客户页面调用的，这个客户页面希望跳到 ASP 页面，运行这里的一个过程，然后可能跳到其它什

么地方。导航常用于处理某种窗体。

- 执行方法（Execute methods）是由一个客户页面调用的，这个客户页面希望使用远程脚本去执行一个脚本功能。经常使用执行方法验证用户键入的用于检索数据库的值；或者执行一次数据库查询把一个控件植入到一个页面上。

所有页面对象都有一个默认方法，叫做 show()，这个方法显示页面的内容。这个方法是在页面上其它方法被处理之后才调用的。

为页面对象创建方法

1. 如果页面已经没有一个控制，就把 PageObject 控件加到页面上，并给 PageObject 控件一个名字。
2. 在一个拥有属性 RUNAT=SERVER 的脚本块中编写过程（函数或子程序）。这个过程可以有任意数目的参数，但是所有参数都是用值传递的。
   例如，下列函数创建一次查询字符串：

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
Sub ListEmployees(sortOrder)
    sqlText = "SELECT lname, fname FROM Employees"
    sqlText = sqlText & " ORDER BY " & sortOrder
    DoQuery(sqlText)
End sub
</SCRIPT>
```

注意:参数转变成字符串的时间是在你调用一个页面对象方法的时候，这样这些参数就可以成功地跨越 Web 进行传递。在你的页面对象脚本中，应当把参数值转变成所需要的相应数据类型。

3.右键单击 PageObject 控件，然后选择 Properties 以显示 Property Pages 对话框。

4.确定哪种方法将在导航中使用或被执行。然后在 Navigate methods 或 Execute methods 之下的列表中寻找第一个空行。从下拉列表中选择你希望展现作为方法的过程。

注意:有可能展现同一种方法既作为导航方法又作为执行方法。然而，方法的基本过程必须编写成可以适应两种调用类型。

为页面对象定义属性

页面对象允许你展现属性，这些属性基本上是全局变量。你可以在三个级别上查看这些属性：页面、会话和应用。

- 页面作用域属性（page-scope properties）使得一种属性在一个页面上的任何地方都可以被脚本所使用，直到你导航到另一个页面位置。
- 对话作用域属性（session-scope properties）在你的项目中，你导航到任何页面上都可使用。对话值使用 IIS 会话变量进行存储。
- 应用作用域属性（application-scope properties）是你应用中任何用户都可使用的属性。应用值使用 IIS 应用变量进行存储。

你还可以规定在客户脚本、服务器脚本、或者两种脚本中使用属性。如果你规定一种属性作为一种客户属性，你可以进一步规定它可读可写，或者只读。

服务器属性总是可读可写的。

为一个页面对象创建属性

1. 如果页面已经没有一个控件，就把 PageObject 控件加到页面上，并给 PageObject 控件一个名字。
2. 右键单击 PageObject 控件，选择 Properties 显示 Property Pages 对话框，然后选择 Properties 选项卡。
3. 在 Name 栏中，寻找第一个空行，然后键入你希望创建的属性名字。
4. 从其它栏中选择新属性性质：
   - 寿命（lifetime） 选择属性在页面退出以前都可用，还是按照会话或应用值决定是否可用。
   - 客户（client） 选择属性在客户脚本中是只读还是可写又可读。如果你选择 None，则属性在客户脚本中不可用。
   - 服务器（server） 选择 Read/Write 使属性在服务器脚本中可用；如果选择 None，则不可用。

要使得特性可在你的脚本中访问，页面对象就要对这些脚本实现 get()和 set()方法。例如，如果你定义一个称为 Color（颜色）的特性，你可以使用 getColor()读取其值，用 setColor()方法设置属性值。有关细节，参见本章后边“访问页面对象属性”。

引用其它页面

你可以用 thisPage 的默认页面对象名字调用方法和使用属性。然而，如果

你希望从其它页面对象上访问这些方法和属性，你必须首先在当前页面上为那个页面创建一个引用子。

引用其它页面对象

1. 如果页面已经没有一个控件，就把 PageObject 控件加到页面上，并给 PageObject 控件一个名字。如果你的脚本目标是服务器，脚本对象模式必须对该页面开启。
2. 右键单击 PageObject 控件，选择 Properties,以便显示 Property Pages 对话框，然后选中 Reference 选项卡。
3. 在 Name 栏，单击“三点”按钮以显示 Create URL 对话框。
4. 选择你希望作为页面对象而引用的 .asp 文件。键入选项，决定页面对象应当如何调用，然后单击 OK。要了解这个选项的帮助，在 Create URL 对话框中按 F1 键。

调用页面对象方法

你可以用两种方式调用页面对象的方法：用导航方式和执行方式。当你完成一个页面，希望跳到另一个页面并在那里处理脚本的时候，用导航方式调用一种方法。这同在 HTML 脚本中导航到一个 URL 的情况很类似：一次单向跳跃，除非在导航之前使用一个页面对象保存当前页面的状态。例如，你可能从一个页面上的用户那里收集查询参数，然后跳到另一个页面，你实际在那里创建并执行查询。

与此相对应的，用执行方式调用方法类似于熟悉的面向对象的方法调用：

你在某个地方调用脚本，而且执行任务并返回。然而，用执行方式调用的方法是异步运行的。调出方法的页面依然保留在浏览器中，用户可以继续用这个页面进行工作。

只在运行客户脚本的情况下，你才用执行方法调用方法，而且调出的方法在服务器上。例如，在用户填充表格的时候，一个客户脚本可能调用服务器页面上的一种方法去完成一次数据库查阅。

调用页面对象方法

1. 在当前页面上对你希望使用的页面对象创建一个引用子。有关细节参见本章前边的“引用其它页面”一节。
2. 在你的脚本中，使用下述调用类型中的一种类型调用页面对象方法：

   pageObject.navigate.*methodName(parameters)*

   pageObject.execute.*methodName(parameters)*

   这里：

   - *pageObject* 是你在前边对一个页面对象创建的引用子。
   - 导航和执行都是确定如何处理方法调用的页面对象的子对象。
   - *methodName* 是你希望调用的 pageObject 方法名字。
   - *parameters* 是你要传递给方法的参数列表。所有参数都用值传递。

注意:参数可以在你调用一个页面对象方法的时候转变成字符串，这样参数就可以成功地在 Web 之间进行传递。在你的页面对象脚本中，你应当把参数值转变成与要求所适应的数据类型。

下边的脚本示出一些简单的处理窗体。当用户敲击客户页面上 List Now 按

钮时，这个按钮的 onclick 处理程序就提取该页面上一个列表框中的值，然后调用对象 poListEmployees 上的一种方法，以便进行处理。在这个例子中，用户信息作为参数传递给页面对象的 CreateList 方法。

```
<SCRIPT LANGUAGE="VBScript">
Sub btnListNow_onclick()
    department = lstDept.gettext()
    poListEmployees.navigate.CreateList(department)
End Sub
</SCRIPT>
```

访问页面对象特性

当你为一个页面对象定义一个属性的时候，脚本对象模型创建一个 get()方法和一个 set()方法，你可以用这些方法访问属性。例如，如果你定义了一种叫做 UserName 的属性，你就可以使用方法 getUserName 读取这个属性的值，并且使用 setUserName 设置这个属性的值，如下边例子所示：

```
newUser = PageObj1.Navigate.getUserName()
PageObj1.Navigate.setUserName(txtUserName.Value)
```

在使用这些属性进行工作的时候，你需要知道它们的寿命。如果你已经定义了属性的寿命，例如定义为“页面”，只有当你离开这个页面并显示另一个页面的时候才获取并设置这个属性的值（再一次调出同一个页面，执行一种方法，以便返回作用于页面上的属性值）。然而，在你导航到其它页面之后，该属性将复原。

24.4 远程执行服务器脚本

如果你已经创建了一些 ASP 页面作为页面对象，你就可以远程调用这些页面上的方法，也就是说，你可以在一个客户页面装载到一个浏览器的情况下，无需导航离开这个客户页面就可以执行一个 ASP 页面上的一种方法。

因为你没有离开客户页面，它的值依然保留着，客户脚本也可以继续处理，不需要复杂的策略去保存页面之间的值。在这个期间，服务器脚本可以执行服务器相应的过程，例如数据库查阅。在一个过程完成的时候，服务器可以把这些刚刚得到的结果发送给客户机，而不是重新窗体化或者重新发送整个页面。这样就降低了服务器的负荷。

你可以用两种方式执行服务器脚本：

- **同步方式**：你的脚本调用远程过程，并等待返回。如果你在继续进行以前等待远程过程的结果，这种方式非常有用。
- **异步方式**：你的脚本调用一个远程脚本，然后继续进行处理。这个页面对进行工作的用户保持可用。在 Web 应用中，异步调用是很有用的，因为请求到达服务器并且返回来，这样一个远程过程可能要很长时间。

产生同步远程过程调用

远程脚本编程使用 ASP 页面对象技术。被调用的页面是一个 ASP 页面对象，而且这个页面上有你已经展现出来的方法。这个客户页面可能是一个 .htm 文件，也可能是 .asp 文件，上面有调用上述页面对象方法的客户脚本。

要使远程脚本从一个客户页面调用一个服务器页面，就要使用页面脚本执

行子脚本。执行子脚本不从你调用的方法中返回单个值，而是返回一个“调用对象”(call object)，这个调用对象是包含返回信息和有关调用过程状态信息的对象。

最常用的属性是调用对象的 return_value 特性，这个属性包含由远程过程计算出来的或查阅出来的单个值。其它调用对象属性允许你获取远程过程调用状态有关的更多信息。

产生同步远程过程调用

1. 在当前页面上创建你希望使用页面对象的引用子。有关细节，参见本章前边的“引用其它页面”和“跨越页面扩展脚本对象模型”两节。
2. 在你的客户脚本中，使用如下语法调用过程：

pageObject.execute.*methodName*

注意:远程脚本是用存储在“脚本库”(Script Library)中的脚本实现的。不要改变库的内容，否则远程脚本将不能正常工作。

例如，下述函数是在客户脚本中调用的，用于验证信用卡。要完成验证，需要在页面对象 poSignIn 中调用验证方法。远程过程返回的值在调用对象验证的 return_value 属性中是可用的。

```
<SCRIPT LANGUAGE="JAVASCRIPT">
function checkCreditCard(){
   retObj = poSignIn.execute.validate(txtCC.value);
   if (retObj.return_value == "OK"){
      alert("Accepted");
      }
      else{
      alert("Rejected");
   }
}
</SCRIPT>
```

异步调用方法

在你异步调用远程脚本过程的时候，你必须在调用脚本中包括额外处理，以便确定远程调用是什么时候完成的。为此，你要指定“调用返回过程”（callback procedure），这个过程是在远程过程完成时调用的功能。

要使用调用返回功能，你需要在客户页面中创建一种辅助功能，去处理远程脚本的结果。你还可用在客户脚本中有选择地创建一个错误处理进程，如果远程脚本遇到错误条件，就调用这个进程。

产生异步远程过程调用

- 在使用 execute 对象调用一种方法的时候，要使用类似下面的语法，其中包含调用返回功能的名字：

  co = PageObject.execute.Method(*p1, p2, callBack*)

  其中

  co，是调用对象。

  p1,p2 是被调用过程所要求的参数。你可以按要求传递许多参数。

  callBasck 在远程过程完成的时候被调用方法的名字。在远程方法完成的时候，立即跳到你指定的进程。

例如，下边给出一个按钮，它的 onclick 属性指出为了验证信用卡而应当调用一个远程过程。调用远程过程，把 displayResults 函数指定为过程的调用返回。

```
<BUTTON
   ID=""btnValidate"
   TYPE="button"
   LANGUAGE="JavaScript"
   ONCLICK="poSignIn.execute.validate(txtCC.value, displayResults) ">
   "Validate Credit Card"
</BUTTON>
```

DisplayResults 函数把远程过程调用对象接收过来，作为一个参数，并测试这个参数，确定信用卡是否有效。

```
<SCRIPT LANGUAGE="JavaScript">
function displayResults(retObj)
{
   if (retObj.return_value == "OK"){
      alert ("Your order has been accepted. ");
   }
   else{
      alert("Invalid credit card number, please re-enter. ");
   }
}
</SCRIPT>
```

注意：你还可以使用远程脚本以调用页面脚本执行方法的外部作用域。有关

远程脚本的细节和更多内容，参见微软公司的脚本 Web 站点 http://www.microsoft.com/ scripting/。

第二十五章　使用 HTML 元素编写脚本

即便你在 Web 页面中不使用设计期间控件，还可以使用创建和编写 HTML 元素脚本的办法创建全功能 Web 应用。做这件事不涉及使用设计期间控件，这对下述情况是有用的：如果你愿意亲自编写 HTML 元素的脚本，或者如果你工作所使用的页面是与非 Visual InterDev 的应用共享的页面。

对专门任务创建脚本通常意味着你一定要了解各个 Web 页面的不同元素如何配合在一起，以及浏览器和服务器如何相互作用。此外，还必须了解不同脚本环境的能力，例如知道脚本是否运行在某些浏览器上。

25.1　选择脚本语言

Visual InterDev 允许你使用你觉得最方便的脚本语言设计 Web 应用。你的应用可以包含使用 VBScript 和 JScript 的混合文件。你甚至可以在同一个页面上使用不同的语言，但是在一个单个脚本块中的脚本必须包含一种单一的语言。

为脚本块设置脚本语言

- 在<SCRIPT>块中，把 LANGUAGE 属性设置为你希望为那个脚本所使用的

语言，如下边例子所示：

```
<SCRIPT LANGUAGE=" VBSCript " >
    [some scripting statemets here]
</SCRIPT>
```

如果你没为脚本块设定一种语言，Microsoft “Internet 信息服务器”（IIS）为服务器脚本运行的默认语言就是服务器设置的语言（通常是VBScript）。对于客户脚本，默认语言由浏览器决定。大多数浏览器都使用JavaScript 作为它们的默认语言。Microsoft Internet Explorer 承认把JavaSctipt 作为它的默认语言，但是却把当前页面的默认重新设置成它在<SCRIPT>块中遇到的第一个 LANGUAGE 属性(attribute)（在第一个 LANGUAGE 属性之后不再重新设置默认语言）。

为 ASP 文件选择语言

因为 ASP 页面中可以有不在<SCRIPT>块内的内联脚本，因而这些页面提供对整个页面重新设置默认语言的手段。在每个 ASP 页面的顶部，<%@LANGUAGE=%>指令规定默认脚本语言。你可以手动改变这种设置，也可以通过设置属性值的办法改变。

改变 ASP 文件的脚本语言

1. 在编辑器中不做选择，在 View 菜单中选择 Property Pages。
2. 在 Property Pages 对话框中，选择 General 选项卡，然后在 Default scripting language 之下选择 Server 列表中的语言。

设计期间控件还使用这个 LANGUAGE 指令了解将使用什么语言生成脚本。如果你在一个文件中删除<%@LANGUAGE=%>指令，Visual InterDev 将使用当前在 Properties 对话框中规定的语言为页面生成脚本。此外，当这些页面在 Web 服务器上运行的时候，它们将使用 Web 服务器记录中 ASP 条目下的 DefaultScriptLanguage 参数中设置的语言进行处理。

注意：如果你有一些页面上面没有<% @ LANGUAGE=%>指令，就要保证在 Options 对话框中你的语言设置同服务器上默认语言设置一样，否则控件生成的代码语言是错误的。

你可以为创建的所有新 ASP 页面设置默认语言。

为新 ASP 页面重新设置默认脚本语言

1. 右键单击 Project Explorer 中的一个项目，然后选择 Properties。
2. 在 Property Pages 对话框中，选择 Editor Defaults 选项卡。
3. 在 Default script language 之下，在 server 列表中选择你希望的语言。

如果你改变默认脚本语言，你创建的新文件将反映这种改变，但是在你已经选择的文件中规定的语言将不改变。

为 Global.asa 文件选择语言

在创建一个新项目的时候，你在 Options 对话框中规定默认脚本语言用于定义 Global.asa 文件中的事件。你可以改变用于这些事件的语言。

改变 Global.asa 文件使用的脚本语言

1. 在编辑器中打开 Global.asa 文件。

2.在每个脚本块中改变 LANGUAGE 属性。

为生成的脚本选择语言

如果是手动编写脚本，你可以使用上述过程设置你自己的脚本语言。然而，你还可以为你使用 Script Outline（脚本大纲）窗口生成的脚本设置默认脚本语言。

改变已生成脚本的脚本语言

1.在编辑器中不做选择，在 View 菜单中选中 Property Pages。
2.在 Property Pages 对话框中，选中 General 选项卡。
3.在 Default scripting language 之下，为客户或服务器脚本选择默认语言。

注意:这种改变只影响新脚本，不影响已经有的脚本。

你还可以改变为整个项目改变脚本生成的默认语言。

为新 ASP 页面重新设置默认脚本语言

1.右键单击 Project Explorer 中的项目，然后选择 Properties。
2.在 Property Pages 对话框中，选中 Editor Defaults 选项卡。
3.在 Default scripting language 之下，为客户或服务器脚本选择默认语言。

注意:这种改变只影响新脚本，不影响已经有的脚本。

25.2 用 HTML 元素处理事件

你可以通过编写响应事件脚本的办法规定 HTML 元素在页面上表现如何。例如，你可以编写客户脚本，以便在下列情况下使变量初始化：在装载文档时（窗口事件的 onload 事件）、在用户向文本框中键入文本时（文本框的 onchange 事件），或用户敲击按钮时（按钮的 onclick 事件）。

注意：把对象加到你应用上或为应用编写脚本的最容易的方式是使用微软 Visual InterDev 设计期间控件。有关细节，参见第二十四章“用设计期间控件和脚本对象编写脚本”中的“用设计期间控件创建窗体”和“为脚本对象编写脚本”。

你编写作为过程的事件处理程序（功能或子程序），这些过程出现在你页面中的特定<SCRIPT>块中。要规定一个过程是一个特定事件的处理程序，你可以

- 创建隐含处理程序，办法是为处理程序对应事件之后的过程命名。
- 使用对象或脚本属性使过程同事件显示相关。

如果你正使用微软 Visual Basic，Script Edition，你可以创建作为子程序或功能的过程。如果你希望返回或设置过程的值，可使用功能，在一些情况下消除事件的影响。

创建隐含处理程序

隐含处理程序使用命名约定，把该句柄链接到对象或事件上。

创建隐含处理程序

1.在 Script Outline（脚本大纲）窗口中扩展节点。这个节点包含你希望为之编写脚本的对象。

注意:只当“脚本大纲”的 ID 或 NAME 属性(attribute)被设置的时候，对象才出现在“脚本大纲”窗口中。只当窗体的 ID 或 NAME 属性被设置的时候，对象才出现在一种窗体中。

2.扩展项目的名字。

3.双击你希望为之编写处理程序的事件名字。

编辑器用下述窗体创建骨架处理程序：

function *objectName_event*

例如，如果你的页面包含一种窗体，这种窗体有一个称为 btnNext 的按钮，则使用名字 btnNext_onclick 创建处理程序。如果你创建一个 JavaScript 处理程序，编辑器还把一个事件属性加到你编写脚本的控件上。有关细节，参见下一节“使用属性创建处理程序”。

处理程序是在下列各<SCRIPT>块中的一个块内创建的，具体是哪个块，取决于你是为客户对象创建处理程序还是为服务器对象创建处理程序，以及这个对象的默认语言是什么。

- clientEventHandlersJS
- clientEventHandlersVBS
- serverEventHandlersJS
- serverEventHandlersVBS

对于生成脚本设置默认语言的细节，参见本章前边的“选择脚本语言”。

创建骨架之后，你就可以填充它。例如，下边对应于名为 txtName 文本框的客户脚本功能是在用户敲击按钮时或离开时调用的。

```
Function txtName_onblur()
   If frmMyForm.txtName.value = "" then
      alert("Name is required!")
   End if
End Function
```

如果你正在 VBScript 中编写脚本，你可以消除一些事件的影响。例如，你可以为窗体的 onsubmit（提交）事件编写处理程序，而这个事件在用户敲击 Submit 按钮时出现。你的脚本可以检查数据是否有效，如果无效，就注销 onsubmit 事件。有关是否能注销一个事件的细节，参见你所使用的事件文档。

消除事件的影响

- 创建作为功能的 VBScript 过程，并把该功能返回值设置为 False（假），如下边脚本所示：

```
<SCRIPT LANGUAGE="VBSCRIPT">
Function frmMyForm_onsubmit()
    If frmMyForm.txtName.value = "" then
        alert("You must enter a name.")
        frmMyForm_onsubmit = false
    End if
End function
</SCRIPT>
```

使用属性(attribute)创建处理程序

把一个过程链接到一个事件上的另外一种方式是设置对象的属性或脚本的属性，显式把两者链接起来。

设置对象属性(attribute)

在创建一个对象的时候，你可以设置一个属性，用于把一个处理程序分配给这个对象。

使用属性把处理程序分配给事件

- 在创建对象的标记中给事件命名，并把过程分配给事件，所使用的语法如下：

<INPUT TYPE="ObjectType"LANGUAGE="language" EventNmame="Procedure">

-or-

<FORM LANGUAGE=”language” NAME=”FormName” EventName=”Procedure”>

这种语言属性并不是所要求的，但可以保证处理程序将以你希望的方式进行工作。如果不规定语言，便承认页面的默认语言。

例如，下边的页面包含带有两个按钮的一种窗体。每个按钮都明确规定把onclick事件链接到一个过程上。

```
<FORM NAME="Form1">
   <INPUT TYPE="button" NAME="btnVB" VALUE="VB" onClick="pressed"
      LANGUAGE="VBScript">
   <INPUT TYPE="button" NAME="btnJS" VALUE="JS" onClick="pressed2()"
      LANGUAGE="JavaScript">
</FORM>

<SCRIPT LANGUAGE="VBSCRIPT">
Sub Pressed()
   alert ("Pressed the VBScript button")
End Sub
</SCRIPT>

<SCRIPT LANGUAGE="JavaScript">
   function pressed2() {
      alert("Pressed the JavaScript button.");
   }
</SCRIPT>
```

在客户脚本中，对于像窗口对象这样不是明显使用标记创建出来的对象，要使用<BODY>标记，如下边的例子所示：

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="VBScript">
Sub newWindow
    'script statements here
End Sub
</SCRIPT>
</HEAD>

<BODY LANGUAGE="VBScript" onLoad="newWindow">
    [...]
</BODY>
</HTML>
```

设置脚本属性

链接对象和处理程序的另外一种方式是设置脚本的属性(attribute)，并把这个脚本分配给一个事件。

使用脚本属性把脚本分配给事件

- 在<SCRIPT>标记中使用 FOR 属性标识对象，使用 EVENT 属性标识事件，所使用的语法是

```
<SCRIPT LANGUAGE="language" FOR="object" EVENT="EventName">
```

你可以对任何命名的元素和用<OBJECT>标记插入的元素使用这种方法。你不需要创建作为子程序或功能的脚本，因为脚本属性规定脚本什么时间运行。

下边的例子同以前的例子类似，但是它使用一种不同语法。

```
<SCRIPT LANGUAGE="VBScript" FOR="Button1" EVENT="onclick">
   alert("Button has been pressed")
   document.Form1.Button1.value="Pressed"
</SCRIPT>

<FORM NAME="Form1">
   <INPUT TYPE="button" NAME="Button1" VALUE="Click">
</FORM>
```

25.3 用 HTML 窗体收集信息

提示用户给出信息的常见方式是把一种窗体放到页面上。用户可以键入信息，也可以使用文本框、选择按钮、下拉列表和复选框选择信息。然后用户用

敲击按钮的办法提交窗体，信息随之变得对你的应用可用。

你可以用下述方式定义窗体：

- 使用 Visual InterDev 设计期间控件，然后为它们的时间编写处理程序。这种方法允许你同你的窗体控件进行可视性工作，并把面向对象的技术用于脚本编程。这种方法还能很容易地把控件绑定在数据库上。
- 创建一种HTML窗体。这是创建窗体的传统HTML方式。如果你想用Visual InterDev以外的工具编辑页面，你就可以使用这种方法。

使用设计期间控件是很容易的，也是强而有力的，因为这种方式允许你使用脚本目标模型进行窗体设计。关于使用设计期间控件和脚本对象模型的有关内容，参见第二十四章“用设计期间控件和脚本对象编写脚本”中的“用设计期间控件创建窗体”。

下边几节描述以传统方式如何创建 HTML 窗体并用这种窗体进行工作。

定义 HTML 窗体

使用<FORM>标记创建一种 HTML 窗体。

注意:不要把 HTML 窗体加到包含 Visual InterDev 设计期间控件的页面上，因为这种页面已经包含用于管理这些控件的窗体。有关细节，参见第二十四章“用设计期间控件和脚本对象编写脚本”中的“用设计期间控件创建窗体”。

创建 HTML 窗体

1.选择你希望包含在窗体中的文本，然后从 HTML 菜单中选择 Form。

编辑器把<FORM>和</FORM>标记放到你所选文本的两边。窗体的 ID 和 NAME 属性自动分配唯一的值，如果你希望的话，这些值可以改变。

注意:如果你把已经有的 HTML 控件放到一种窗体中，这些控件就出现在窗体节点的 Script Outline（脚本大纲）窗口中。

2.规定 .asp 文件的名字，这个文件用于处理窗体，并为窗体分配 ACTION 属性：

<FORM NAME="*FormName*" ACTION="*ASPFileName*.asp">

3.规定窗体信息如何用 METHOD 属性发送给服务器：

<FORM NAME="FormName" ACTION="ASPFileName.asp" METHOD="method">

想发送 HTML 抬头的部分信息（以便使用服务器 Request 对象的 Forms 集合提取），就要把 METHOD 设置成“POST”。

想把字符串信息发送给 .asp 文件（以便可以使用服务器 Request 对象的 QueryString 集合提取），就要把 METHOD 设置成“GET”。

在定义窗体之后，你就可以定义控件，如文本框、按钮等等。

注意:可以使用设计期间控件为你的应用创建用户界面。有关细节，参见第二十四章“用设计期间控件和脚本对象编写脚本”中的“用设计期间控件创建窗体”。

在窗体中创建控件

- 把控件从 Toolbox 的 HTML 选项卡上拖到页面上。

控件的 ID 和 NAME 属性自动分配唯一的值。如果你希望，可以改变这些值。

注意：当你把 HTML 控件拖到 HTML 窗体上的时候，这些控件出现在窗体节点 Script Outline（脚本大纲）窗口中。

例如，一种窗体包含文本框和 Submit（提交）按钮，在你把这些控件从 Toolbox 拖出并进行编辑之后，看上去像下边的样子。

```
<FORM NAME="Form1" NAME="frmMyForm" ACTION="process.asp" METHOD="post">
   Enter your name:
   <INPUT TYPE=text NAME=text1 ID=Text1>
   <INPUT TYPE="submit" VALUE="Submit" ID=Submit1 NAME=Submit1>
</FORM>
```

提交之前处理窗体信息

在简单窗体中在客户机上不要求处理：用户敲击 Submit（提交）按钮，而浏览器完成收集数据的任务，并把数据发送给服务器。然后，你可以通过编写脚本处理按钮敲击等来实现对窗体的处理。

典型的例子是编写一个脚本，对用户的数据进行验证，然后把窗体寄送出去。但是在客户脚本中，你对激发所有窗体控件的时间拥有访问权，这样你便可以对你需要的任何目的，包括替换服务器的全面处理，编写处理程序。

注意：脚本对象模型允许你轻松编写设计期间控件的脚本。有关细节，参见第二十四章“用设计期间控件和脚本对象编写脚本”中的“用脚本对象编写脚本”。

处理客户脚本中的窗体

- 为下列事件编写处理程序：窗体的 onsubmit（提交）事件，或同单个控件有关的事件，如按钮敲击事件或文本框 onblur（变模糊）事件。

想确定特定控件的值，你就要引用一个层次对象，这个对象来自控制属性的有窗体文档。

下边给出一个简单验证例程，这个例程在一种窗体的 onsubmit 事件处理程序中。

```
<SCRIPT LANGUAGE="VBScript">
Function form1_onsubmit()
   If Len(document.form1.text1.value) < 1 then
      MsgBox("You must enter a name!")
      form1_onsubmit = False    ' This cancels the submission
   Else
      ' Information automatically sent to server
   End If
End Function
</SCRIPT>
```

要为其它事件编写处理程序，你可以使用“脚本大纲”（Script Outline）窗口。关于为同 HTML 窗体控件有关的事件编写脚本更多内容，参见本章前边“用 HTML 元素处理事件”。

处理服务器上的窗体

在 Visual InterDev 中窗体通常用服务器上的.asp 文件处理。

处理服务器脚本中的窗体

- 在目标 ASP 页面上，使用服务 Request 对象的 Forms（窗体）集合从窗体中提取值。

窗体集合像一个数列，里面有窗体中所有控件的值。你可以用 Forms 集合中的名字引用它们来提取单个控件的值。

例如，下边脚本使用在窗体中分配给<INPUT>标记的名字提取窗体上一些字段的值，再把这些值放到 Session（会话）对象变量中，以便被其它地方的应用所使用。

```
<%
Session("LastName") = Request.Form("LastName")
Session("BirthDate") = Request.Form("Birthdate")
%>
```

在隐蔽字段中创建动态信息

如果你希望传递不是由用户直接键入的信息，则可以使用隐蔽字段。

注意:如果你使用脚本对象模型和设计期间控件，信息在服务器脚本中是自动可用的。有关更多内容，参见第二十四章“用设计期间控件和脚本对象编写脚本”中的“用设计期间控件创建窗体”。

在隐蔽字段中创建信息

1. 在一个窗体中，创建<INPUT>标记并把它的 TYPE 属性设置为 HIDDEN，如下边例子所示：

```
<FORM NAME=frmEmployee METHOD=POST ACTION=process.asp>
   [...]
   <INPUT TYPE="Hidden" NAME="ExtraInfo">
</FORM>
```

2.使用客户脚本把信息放到隐蔽字段中。

例如，你可以用为窗体 onsubmit（提交）事件编写的句柄提交窗体，在刚好提交窗体之前装载这个隐蔽字段。下边例子就是把当前时间放到隐蔽字段中：

```
<SCRIPT Language="VBScript">
Function frmEmployee_onsubmit
   document.frmEmployee.ExtraInfo = time
End Function
</SCRIPT>
```

在提交窗体的时候，你可以使用服务器脚本提取隐蔽字段的信息，其方式与你从任何其它窗体元素中提取信息一样，如下边例子所示：

```
<%Session("UpdateTime") = Request.Form("ExtraInfo")%>
```

向同一个文件寄送信息

为取代总是创建一个单独的 .asp 文件处理一种窗体的 .htm 文件内容，你可以把窗体和处理这个窗体的脚本放到同一个 ASP 页面上。这样可以简化你的应用，并使得用户更轻松。

例如，一种典型情况是有三个或更多个与同一种窗体相关的页面：一个页面带有窗体，另一个页面带有处理这个页面的脚本，第三个页面带有错误报文。然而，如果这种窗体寄送给持续显现的文件上，你可以把信息报文连同窗体环境一起寄送出去，并且只需创建一个页面。

向同一个文件寄送信息

1. 在 .asp 文件中创建窗体。
2. 在窗体的 ACTION 属性中，规定作为目标 URL 的文件名字。
3. 在服务器中 .asp 文件的顶部，用 Request（请求）对象检查信息是否传递到该文件。如果传递到了，窗体已经提交了，你就可以处理它；否则，假设这个窗体第一次显示出来。

下边的 GetEmail.asp 文件就是这样的例子。脚本决定用户是否已经键入电子邮件地址。如果键入了，检查这个地址是否有效。在每一种情况中，脚本都产生相应的报文，然后报文出现在窗体下边，作为对用户的反馈。

```
<HTML>
<BODY>
<!-- GetEmail.asp -->
<%
    If IsEmpty(Request("Email")) Then
       Msg = "Please enter your email address."
    ElseIf InStr(Request("Email"), "@") = 0 Then
       Msg = "Please enter an email address" & _
       " in the form username@location."
    Else
       ' In a real application, the following message
       ' would be replaced by actual processing.
       Msg = "This script could process the " & _
       "valid Email address now."
    End If
%>
<FORM METHOD="POST" ACTION="GetEmail.asp">
<PRE>
Email: <INPUT TYPE="TEXT" NAME="Email" SIZE=30
➥ VALUE="<%= Request("Email")%>">
<%= Msg %><P>
<INPUT TYPE="Submit" VALUE="Submit">
</PRE>
</FORM>
</BODY>
</HTML>
```

25.4 为用户显示信息

Web 页面填充用户信息，但是这些信息大部分都是静态的。你可以使用脚本为用户显示动态信息。

从客户脚本显示信息

因为客户脚本在浏览器上运行，这就为你希望如何为用户显示信息提供了灵活性。一种方式是显示报文框，这是独立应用经常使用的方式。

显示客户脚本中的报文框

- 在客户脚本中调用窗口对象的告警方法：

```
<SCRIPT LANGUAGE="VBScript">
   window.alert("Hello, World.")
</SCRIPT>
```

要引用当前窗口，你可以省略引用窗口对象。

如果你在 VBScript 中编写脚本，你可以使用 MsgBox 功能，这个功能允许你规定具体按钮。

你可以直接在页面上显示客户脚本的信息，这样的页面是用 HTML 文本实现的。

用客户脚本显示页面上的信息

- 调用文档编写方法，把文本放到页面中脚本执行的位置上。例如

```
<SCRIPT LANGUAGE="VBScript">
   document.write "The current time is " & time
</SCRIPT>
```

如果你的页面包含 HTML 文本框或文本域控件，你可以改变这个框的内容，用于显示信息。

显示 HTML 控件中的信息

- 设置文本框或文本域控件的值的属性，如下例所示：

```
<SCRIPT LANGUAGE="VBScript">
Function btnShowTime_onclick()
   document.Form1.txtTime.value = time
End Function
</SCRIPT>
```

如果你的应用在支持动态 HTML（例如 Internet Explorer 4.0）的浏览器中运行，你就可以直接设置拥有名字或 ID 的任何标记的文本。

使用 DHTML 设置标记文本

- 设置标记的 innertext 属性。这个标记必须有一个 ID 或者有你可以在

脚本中引用的名字。下边示出一个标记和如何设置标记。

```
<HEAD>
<SCRIPT LANGUAGE="VBScript">
Function btnChangeText_onclick()
   para1.innerText = "The new time is: " & time
End Function
</SCRIPT>

</HEAD>
<BODY>
<P ID=para1>This text will be replaced.</P>
<P><INPUT TYPE="button" NAME="btnChangeText" VALUE="Change text"></P>
</BODY>
```

要对你想显示的文本赋予格式，你可以在其中包含 HTML 标记，如下例所示：

```
<SCRIPT LANGUAGE="VBScript">
   document.write "<P>The current time is <B>" & time & "</B></P>"
   document.write "<P>The current date is <B>" & date & "</B></P>"
</SCRIPT>
```

如果你希望显示的信息中包含 HTML 保留的字符，例如<and>，你就不能把它们放在字符串中直接显示。

显示保留符号

- 把 HTML 语法用于 ASCII 字符，例如&1t，或&#60，要用一个开始尖括号（<）：

```
<SCRIPT LANGUAGE="VBSCRIPT">
    document.write "&lt;Click here&gt;"
</SCRIPT>
```

从服务器脚本显示信息

要使客户从服务器脚本上显示信息，通常你可以使这些信息成为发送到浏览器的页面的一部分。

从服务器脚本显示页面上的信息

- 调用 Response.Write 方法，把它传递给你希望显示的信息。
  <% Response.Write “The current time on the server is” & Time %>
  或者
- 使用“＝”操作符，这是同种方法的缩短形式：
  <% = “The current time on the server is” & Time %>
  或者
- 如果信息不在变量中，或者不是用表达式计算出来的，就使它成为页面的组成部分。要显示文本，就要保证文本在<% %>定界符之外，而且不是<SCRIPT RUNAT=”SERVER”>块的组成部分。你可以在这种方式中包

含 HTML 标记，如下例所示：

```
<BODY>
<H1>The Time Page</H1>
<% t = time %>
<P>
The current time at the server is <B><%=t%></B>
</P>
</BODY>
```

服务器不能直接显示报文框，但是服务器脚本可以建立显示信息的客户脚本。

显示服务器脚本中的报文框

- 创建一个客户脚本，然后把它放在服务器脚本中，以便按条件显示它。在下边例子中，只当服务器上的错误旗标为真时，这个脚本块才发送到浏览器中。

```
<%if fError = True then%>
   <SCRIPT Language="VBSCRIPT">
      alert("An error occured on server.")
   </SCRIPT>
<%End If%>
```

如果你希望显示的信息包含 HTML 保留的字符，例如<and>，则不能在字符

串中显示它们。

显示保留符号

- 调用 HTMLEncode 方法，把这些字符转换成用于 ASCII 字符的 HTML 语法。

```
<%= Server.HTMLEncode(" The paragraph tag: <p> " ) %>
```

25.5 有条件导航

在静态 Web 页面中，你用<A>标记把页面链接在一起，如下例所示：

```
Click <A HREF= home.asp>here</A>to return to the home page.
```

但是，使用脚本，你可以根据应用中的条件指定目标页面，动态地在应用中实现导航。例如，你应用中的页面可能要用户提供口令。如果用户给出正确的口令，应用便显示欢迎页面，否则显示错误信息。

注意:你可以使用脚本对象模型和为 Web 页面设计导航的页面对象。有关细节，参见第二十四章“用设计期间控件和脚本对象编写脚本”中的“跨越页面扩展脚本对象模型”。

从客户脚本进行导航

如果你正在编写客户脚本，则可以通过调用一种方法或设置一种属性的办法控制浏览器显示文本的页面。

在客户脚本中导航到另一个页面

- 在 VBScript 中调用浏览器窗口对象的导航方法，并把这种方法传递给要去的页面 URL。

  或者

- 在任何脚本语言中，把窗口对象的 location.href 属性设置成目标页面的 URL。

  或者

- 在任何脚本语言中，调用窗口对象的打开方法，把这个方法同其它参数一起传递给要去的页面 URL。

  调用这种方法允许你规定一个窗口或帧，在这个窗口或帧中，显示一个页面。启动浏览器的一个新实例，并控制浏览器的外观。

例如，下边客户脚本读出窗体中复选框中的值，并根据用户的喜好，导航到两个可能的页面之一：

```
<SCRIPT LANGUAGE="VBScript">
Sub NextPage()
   ' The following reads the value of a checkbox called Checkbox1
   fReviewIntro = document.frmNavBar.Checkbox1.checked
   If fReviewIntro then
      window.navigate("http://MyServer/intro.htm")
   Else
      window.navigate("http://MyServer/page2.htm")
   End If
End Sub
</SCRIPT>
```

下边是一个简单例子，但是把窗口对象的 href 属性设置成导航：

```
If fReviewIntro Then
   window.location.href = "http://MyServer/intro.htm"
End If
```

从服务器脚本进行导航

你还可以从服务器脚本控制在浏览器上显示的文本页面。如果移到一个指定页面的条件只取决于服务器上可用的信息，例如，来自数据库、会话或应用对象变量等的信息，你就可以使用服务器脚本进行导航。

在服务器脚本中导航

- 调用 Response.Redirect 方法，并传递给要去的页面 URL。

例如，你可能希望在观察页面之前，保证用户登录到你的应用上。如果用户企图导航到应用中较深的页面上，而不事先登录，你便会希望检测用户，并向用户发送登录页面。在下边的例子中，一个会话对象变量包含一个旗标，指明用户是否已经登录。

```
<%
If Not Session("Been_to_Home_Page") Then
   Response.Redirect "homepage.asp"
End If
%>
```

缓冲响应

作为默认方式，在服务器把一个页面发送给浏览器的时候，一旦组成页面，就以流式进行输出。然而，你可以使服务器对输出进行缓冲。如果你这样做，页面就在被发送之前在服务器里完成处理。其优点是你可以调用诸如 Response.Redirect 的方法发送不同的页面。如果页面不被缓冲，且你企图在当前页面发送之后对某个页面进行重定向，则服务器就报告出错。

设置缓冲

- 把 Response（响应）对象的 Buffer（缓冲器）特性设置成 True（真）或 False（假），如下边例所示：

  <% Response.Buffer = True %>

管理顺序页面导航

如果以服务器方式进行导航，你可以使用一种绑定部件，它可以在整个页面列表中管理导航。用不着维护数个 ASP 页面中的 URL 引用子，你可以在一个容易编辑的文本文件中指定多个页面之间的顺序编排。

创建顺序页面导航

1.创建一个文本文件，每一行包含一个文件列表。
2.创建一个 NextLink 对象。
3.使用 NextLink 对象的方法在列表中多个页面之间进行移动。
下边例子从文本文件中读出链接顺序，并在一个页面上创建一个内容表格。

```
<% Set NextLink=Server.CreateObject("MSWC.NextLink") %>
<% count=NextLink.GetListCount("/Nextlink.txt") %>
<% i=1 %>
<UL>
<% For i = 1 to count %>
   <LI><A HREF="<%= NextLink.GetNthUrl("/nextlink.txt",i) %>">
   <%= NextLink.GetNthDescription("/nextlink.txt",i) %></A>
<% Next %>
</UL>
```

25.6 共享动态信息

在你的 Web 应用中，你可能需要多次使用两个页面或多个页面上的相同信息。例如，你的应用可能：

- 提示用户给出名字，然后把名字传递给另一个页面并在那里显示出来。
- 显示用户最后访问你站点的时间和日期。
- 要求用户登录。当用户登录的时候，应用就为用户分配安全级别，即在你应用中到处检查的安全级别。

注释：你可以使用脚本对象模型和页面对象去自动维护信息状态。有关细节，参见第二十三章“脚本编程概念”中的“脚本对象模型”。

共享动态信息

- 把信息存储在由应用或会话对象维护的全局变量中。而应用和会话变量中的信息保存在多个页面之中。

  或者
- 把一个字符串从一个页面直接传递给另一个页面，这另一个页面是一个链接上 URL 的组成部分。

  或者
- 把信息存储在允许永久存储数据的数据库中，与其它应用共享这些信息，并且使用数据库功能以过滤和分析应用数据。详见第二十章“修改数据”。
- 把信息存储到用户计算机的串存单元（cookie）中，因为你可以读写

串存单元，因而你可以使用串存单元维护用户专门信息。

或者

- 把信息存储到 ASP 对象中。

每种方法都有不同的用法，具体用法取决于若干因素，如多少个页面需要这个信息，和你是否需要硬件保留这些信息等。

存储全局信息

你可以使用 Microsoft Intrnet Information Services(IIS，Internet 信息服务)的两个对象定义全局变量：

- **应用对象** 服务器为每个基于 ASP 的应用维护一个应用对象。这个对象是在第一个用户访问应用的时候初始化的，然后保持不变，直到应用关掉为止。

  当你在服务器应用对象中定义变量时，其值对应用中所有页面都是可用的。典型例子是应用命中一个计数器，每当用户开始一个会话时，这个计数器就更新一次。

- **会话对象** 会话(session)对象是在用户第一次请求服务器中一个页面时初始化的。该对象保持不变，直到放弃会话为止，这可能是用户手动关掉并重新打开浏览器（调用 Session.Abandon 方法），也可能是用户的浏览器停止访问该应用的页面持续时间达到了规定的时间间隔。

  会话对象对任何用户都是全局性的。你可以使用会话对象维护用户的安全级别。

获取和设置应用中的值和会话对象变量

- 在服务器脚本中使用如下语法：

```
Application("VariableName") = value
Session("VariableName") = value
variable = Applicaton("VariableName")
variable = Session ("VariableName")
```

应用和会话对象变量初始化

- 在全局 Global.asa 文件中，为 Application_OnStart 和 Session_OnStart 事件编写处理程序，为你希望使用的每个变量设置初始值。

例如，你可以在应用对象变量中维护一个命中的计算机。你可以使用下边的脚本对 Application_OnStart 事件中的计数器进行初始化：

```
Sub Application_OnStart()
    Application("counter") = 0
End Sub
```

用户每次启动一个会话，这个计数器就更新一次。然后你还可以在会话对象变量中为用户产生一个计数器的本地副本。做这件事的最好位置是在 Global.asa 文件的 Session_OnStart 事件中，要使用下边这样的脚本：

```
Sub Session_OnStart()
    iCount = Application("counter")
    iCount = iCount + 1
    Application("counter") = iCount' Global counter
    Session("counter") = iCount ' User's copy of counter
End Sub
```

打开你应用的页面，便以下述形式显示这个计数器：

```
<BODY>
<H1>Welcome</H1>
You are visitor number <%=Session("counter")%>
out of <%=Application("Counter")%>.
</BODY>
```

当用户第一次看见打开的页面时，这两个数是相同的。如果用户再一次访问页面时，“out of”数（应用对象计数器）可能改变(如果第一个用户开启一个会话之后，其他用户又开启会话的话)。

应用和会话对象变量通常都只存储动态信息。如果你希望永久保留这些信息，你必须采取一种方式保存应用和会话之间的信息。一种方式是把 Application_OnEnd 处理程序或 Session_OnEnd 处理程序的值全部写到 Global.asa 文件中。

把查询串加到链接上

你可以把信息作为链接（<A>标记）的一部分直接传递给另一个页面。例如，一个页面可能显示雇员名字列表，列表中的每个名字都是一个链接。当用户选择其中的一个名字的时候，链接便调用一个页面，并把这个页面传递给相应雇员 ID。

把查询串加到链接上

- 在一条链接定义（<A>标记）中，把一个查询串加到目标 URL 的后边，要在 URL 和这个查询串之间用“？”作为定界符。语法如下：
  <*targetURL*> ? <*parm1*>=<*value*> & <*parm2*>=<*value*> & ...

下例说明一个显示雇员名字列表的页面。每个名字是一条链接。所有链接都到同一个页面上，但是每个链接都传递一个不同的雇员 ID。

```
<BODY>
<H1>List of employees</H1>
<P>Click the name of an employee to see information about that employee.</P>
<A HREF=EmplInputForm.asp?empid=1>Ann</A><BR>
<A HREF=EmplInputForm.asp?empid=2>John</A><BR>
<A HREF=EmplInputForm.asp?empid=3>Susan</A><BR>
<A HREF=EmplInputForm.asp?empid=4>Michael</A><BR>
</BODY>
```

在 EmplInputForm.asa 文件中，你可以使用服务器 Request（请求）对象的 QueryString（查询串）集合确定要传递什么值：

```
<% vEmp1ID = Request.QueryString(“empid”) %>
```

你可以使用服务器脚本，创建多个先进的动态链接，以便提供值。例如，下行脚本中雇员 ID 和雇员名字是从对数据库的一次查询得到的：

```
<A HREF=EmplInputForm.asp?empid= <%=RS.Fields(" EemID ")%> > _
  <%=RS.Fields(" EmpFName ")%> </A>
```

在用户计算机上存储信息

维护用户信息的一种方便方式是使用一个串存单元（cookie）。当第一次启动一个会话时，服务器便把一个串存单元发送给客户浏览器。客户浏览器每次从那个服务器请求一个页面的时候，浏览器都把这个串存单元发回到服务器，服务器接着就可以阅读这个串存单元，并识别客户浏览器。

你可以使用串存器存储你自己的应用信息，例如用户的喜好。串存单元一直可用到串存器 Expire（到期）属性中规定的日期。

注意：因为串存单元可能写到用户的硬盘中，大多数浏览器通常都允许用户关闭它们，或者在接收一个串存器之前显示一条警告。串存单元对于一些应用是不现实的，例如对于拥有范围广泛的浏览器和安全设置的用户访问的公共信息。如果你使用串存单元，你的应用必须提供另一种方式维护动态信息，因为用户浏览器可能拒绝串存单元。

在串存单元中存储信息

- 要把信息存储到串存单元（cookie）中，可使用 Response.Cookie 集合。要取得一个串存单元的信息，就使用 Request.Cookie 集合。例如

```
<% Response.Cookies ("FavoriteColor")="Red" %>
<% txtFavorite = Request.Cookies("FavoriteColor")%>
```

串存单元可以存储多个值，对串存单元中的每个值都分配一个键以便识别。

要把一个值存储到串存单元中，你要使用 Response 对象，指定要更新的串存单元的名字、要更新的键以及值。如果串存单元尚未存在，Response 对象就创建它。例如，下边脚本就更新一个串存器，把其键 FavoriteColor 设置成“Red”:

```
<% Response.Cookies("Preferences") ("FavoriteColor") ="Red"
%>
```

注意:你必须在一个 .asp 文件的<HEAD>部分中更新串存单元，否则将导致错误结果。

如果一个已经存在的串存单元有键值，但是 Response.Cookies 方法没指定键名字，那么这个已经存在的键值便被检测到。同样，如果一个已经存在的串存单元没有键值，但是 Response.Cookies 方法指定了键名字和值，那么这个已经存在的串存单元值便被检测到，而且创建新的键值对。

要从串存单元取得这个值，可要用类似的语法使用 Request 对象：

```
<% vColor= Request.Cookies("Preferences") ("FavoriteColor") %>
```

有关如何使用串存单元存储信息，参见 Visual InterDev 在线文档中的 User Preference sample。

阅读与编写文件

在服务器脚本中，维护信息的另一种方式是把信息存储到服务器上的一个文本文件中。

阅读与编写文本文件

1. 在服务器脚本中，创建一个 TextStream 对象。
2. 使用该对象的 CreateTextFile、WriteLine 和 ReadLine 方法管理这个文件中的信息。

下边的例子创建一个新文件，并把一行文本写入该文件中：

```
<HTML>
<BODY>
<H3>Textstream test</H3>
<%
  Set OutStream = Server.CreateObject("MS.TextStream")
  OutStream.CreateTextFile "tsworks.txt", , True
  OutStream.WriteLine "This line is written to the file."
%>
</BODY>
</HTML>
```

25.7 编写可重复使用的脚本

在大多数 Web 应用中，你会希望显示多个页面上的 HTML 内容块或多个文件中的进程脚本。

使用多个文件中的内容

- 使用服务器方的#INCLUDE 指令，把其它文件内容动态插入到你的文件中。

你可以使用这个#INCLUDE 指令：

- 共享各个页面之间的 HTML 内容。
- 共享脚本库。

注意：当你把其它文件的内容插入你的页面中的时候，Script Outline（脚本大纲）窗口不反映它们的内容。同样，在所包含文件中定义的对象在 IntelliSense 语句的下拉列表中不出现。

共享 HTML 内容

你的应用中可能包含你希望在多个页面上使用的 HTML 元素：页面导航按钮（Next、Previrous 和 Home）、版权信息、公司徽标等。

为取代把文本复制和传递到多个页面上，你可以把信息放到文件上，然后引用文件，被引用的文件包含你在其它页面相应位置处所希望的文本。然后，如果你需要更新这些文本，你可以在文件中更新它，包含这个文件的所有页面会自动显示更新的文本。

引用其它文件中的信息

- 使用服务器命令#INCLUDE FILE（对于路径同当前文件的关系）或#INCLUDE VIRTUAL（对于路径同虚拟根的关系）。

例如，下边一行包括在页面顶部名字为 Header.inc 的一个文件和在页面底部名字为 Footer.inc 的一个文件。

```
<!-- # INCLUDE FILE = " Header.inc "  -->
Normal HTML text and script goes here.
<!-- # INCLUDE FILE = " Footer.inc "  -->
```

当这个文件被 Web 服务器处理的时候，Header.inc 和 Footer.inc 文件的全部内容都插入到位于#INCLUDE 指令位置的文件中。Header.inc 和 Footer.inc 中的任何脚本都将被处理，但是处理之前包含的所有内容都要归并到一个主文件之中。你甚至可以用使文件包含其它#INCLUDE 指令的办法，把文件嵌套起来。

你可以更新你产品服务器上被包含的文件，无需使 Web 服务器停下来。然而，你不能把#INCLUDE 包含在一个循环中，也不能递归式地包含文件。例如文件 A.inc 可能有如下一行：

```
<!-- # INCLUDE FILE = " B.inc " -->
```

在这种情况下，你不能在 B.inc 文件中包含如下一行：

```
<!-- # INCLUDE FILE = " A.inc "  -->
```

企图这样做将导致出错。

共享脚本库

在 ASP 页面中，你可以使用#INCLUDE 指令共享脚本库。这就允许你在多个页面之间共享子程序和函数。例如，下边文件包含一个函数和一个子程序：该功能返回由单引号(')定界的串，而该子程序取一个数列自变量，并在 HTML 表中显示这个数列值：

```
<!-- shared.inc -->
<%
Function SQLString(cString)
    SQLString = "'" & cString & "'"
End Function
%>
```

```
<%
Sub Array2Table(aArray)
If IsArray(aArray) Then %>
   <TABLE>
   <% For i = 0 to UBound(aArray,1) - 1%>
      <TR>
      <% For j = 0 to UBound(aArray,2) - 1 %>
         <TD> <%= aArray(i,j) %> </TD>
      <% Next %>
      </TR>
   <% Next %>
   </TABLE>
<%End If
End Sub %>
```

为取代在你的许多页面上重复这个脚本，你可以使用下边一行使子程序和函数可用：

```
<!-- # INCLUDE FILE = " shared.inc "  -->
```

要保证这个#INCLUDE 行在任何调用之前进入文件的子程序或功能。

接着，如果你希望改变 Array2Table 子例程中的某些窗体选项（如给表格边框为 1，或宽度为 100%），你可以在 Shared.asp 中的子例程内建立这种改变，则所有页面都将立即以新特征显示这个表。

在客户脚本中，你可以在<SCRIPT>块中包含对其它文件的引用子。使用 SRC 属性对你希望包含脚本的文件规定名称。例如，下边<SCRIPT>块包含一个引用

子，指向包含出错报文例程的页面。

```
<SCRIPT SRC="Errmsg.htm"></SCRIPT>
```

以这种方式把一个引用子放到一个文件之后，你就可以调用这个页面上的脚本，就像脚本在当前页面上一样。

25.8 创建可移植脚本

如果你编写一个 Web 应用供公司的内部网使用，或供其它已知环境使用，你通常可以依靠特定 Web 浏览器的特性。但是如果你的应用将被可自选浏览器的用户访问，则必须小心编写这个应用，使尽可能多的浏览器能正常工作。

在创建一个尽可能独立于浏览器的应用时，你必须可以：

- 识别用户要使用的浏览器和它的功能。
- 创建在不同浏览器上运行的脚本。或者说，你可以提供适当的方式，以便适应多种浏览器，包括不能支持你应用所有特性的浏览器。

注意：当你在 Visual InterDev 中创建一个新 Web 页面的时候，你可以为页面设置的属性之一是 TargetBrowser。这个属性履行任何检查，但是，可防止你创建不适合于某种浏览器的脚本。

这种 TargetBrowser 属性只是向编辑器告警：你是否为 Internet Explorer 4.0 编写脚本。如果是，它便告诉编辑器在完成脚本语句的时候显示 Internet Explorer 4.0 对象。

识别浏览器和浏览器能力

你可以使用各种不同属性和对象以获取用户当前使用浏览器的有关信息。

获取客户脚本中浏览器信息

- 查询窗口导航器对象的属性。
  例如，下边客户脚本显示当前浏览器的名字：

```
<SCRIPT LANGUAGE="VBScript">
Sub Button1_OnClick()
   MsgBox "Current browser is " & window.navigator.appname
End Sub
</SCRIPT>
```

在服务器脚本中，你可以从 HTTP 抬头中获取某个浏览器的基本信息，并使用绑定部件得到细节。

在服务器脚本中从 HTTP 抬头获取基本浏览器信息

- 使用服务器 Request 对象的 ServerVariables 集合，并查询 HTTP_USER_AGENT 的值。

HTTP_USER_AGENT 变量的值是一个串，它列出浏览器的兼容性、名字和版本。例如，这个脚本显示请求一个页面的浏览器功能，这个页面上面有该脚本。

```
<% browser = Request.ServerVariables("HTTP_USER_AGENT")%>
<H1>Browser Identification</H1>
<P>Your browser identifies itself as:</P>
<%=browser%>
```

在服务器脚本中获取有关指定浏览器功能的信息

- 创建 BrowserType 对象实例，然后查询它的属性。

例如，你可以确定浏览器是否支持帧，或者是否支持 VBScript。下边例子使用 BrowserType 对象部件以显示当前浏览器的功能。

```
<% Set bc = Server.CreateObject("MSWC.BrowserType") %>
Browser: <%= bc.browser %><BR>
Version: <%= bc.version %><BR>
Supports frames?
   <% If (bc.frames = "true") then %>
      Yes<BR>
   <% Else %>
      No<BR>
   <% End If %>
 Supports tables?
   <% If (bc.tables = "true") then %>
      Yes<BR>
   <% Else %>
      No<BR>
   <% End If %>
 Supports background sounds?
 <% If (bc.BackgroundSounds = "true") then %>
      Yes<BR>
   <% Else %>
      No<BR>
   <% End If %>
```

```
Supports VBScript?
   <% If (bc.vbscript = "true") then %>
      Yes<BR>
   <% Else %>
      No<BR>
   <% End If %>
Supports JavaScript?
   <% If (bc.javascript = "true") then %>
      Yes<BR>
   <% Else %>
      No<BR>
   <% End If %>
```

指定浏览器类型的有关信息保存在服务器上 Browscap.ini 文件中。Browscap.ini 文件的内容决定对 BrowserType 对象可用的属性。

创建独立于浏览器的脚本

除非你知道你的用户将使用什么类型的浏览器，否则你就应当预料范围广泛的浏览器，创建的脚本应能在尽可能多的浏览器上运行。

使你的脚本独立于浏览器

- 避免使用只能在某些浏览器上可用的特性，如特定对象。
  或者

- 允许用户规定他们希望看见的内容。

  或者

- 对特定浏览器功能进行测试，对于浏览器可能不支持的脚本，要慎重使用其分支的内容。

提示:适应特定浏览器的简单方式是把你的 Web 页面创建成两个（或更多）版本。根据查询用户浏览器情况（或者明确问用户），你可以提供通过你应用的不同途径，每种途径都包含同用户浏览器相兼容的页面。

避免特定浏览器特性

特定浏览器使用不同对象模型，尽管各种浏览器之间的对象模型常常有重叠之处。例如，Internet Explorer 4.0 支持动态 HTML（DHTML），这种特性允许在页面上添加动画片和文本效果。然而，DHTML 特性不是对所有浏览器都是必需的。所以，如果你使用这些特性，就必须保证使用不同浏览器的用户不要试图观察有 DHTML 页面的页面。

有关特定浏览器可以使用哪些对象的细节，参见你的浏览器文档。

允许用户规定内容

你可以查明用户浏览器在你脚本中的能力，但是你不能确定其它因素，例如调制解调器的速度，因为这会影响用户对你的 Web 站点的感受。允许用户规定他们希望接收的内容类型，把控制权交到用户手中是很有好处的。

允许用户规定内容

1. 使用一个窗体或动态链接，询问用户的喜好。有关细节，参见本章前边“用 HTML 窗体收集信息”和“共享动态信息”。
2. 把这些喜好存储到一个服务器会话变量中，或者串存单元（cookie）中。详参见本章前边“共享动态信息”。
3. 使用分支显示或包括用户指定信息。有关内容，参见本章前边“编写可以重复使用的脚本”。

例如，下边两个链路允许一个用户规定显示首选。

```
<A HREF="setpref.asp?type=basic">Basic Display<A>
<A HREF="setpref.asp?type=rich">Rich Display<A>
```

Setpref.asp 中的脚本把一个会话对象变量设置成存储用户的首选。

```
<% Session("type") = Request("type")
Response.Redirect "home.asp" %>
```

你页面上的脚本可以检查这个会话对象变量的设置，确定如何显示你的内容：

```
<% If Session("type") = "rich" Then %>
        Display Frames, Tables, and Images.
<% Else %>
        Display text only
<% End If %>
```

测试浏览器能力

你还可以使用脚本中的逻辑，确定可用的特性。例如，下边的脚本测试浏览器是否支持 JavaScript。如果支持，就插入一个小脚本，跳到首页上；否则就插入 HTML 文本，显示用户要敲击的跳跃。

```
<% Set bc = Server.CreateObject("MSWC.BrowserType") %>
<% If bc.JavaScript = true then %>
   <SCRIPT LANGUAGE="JavaScript">
      top.location.href = "home.asp"
   </SCRIPT>
<%Else%>
   Click <A HREF="home.asp">here</A> to return to the home page.
<%End If%>
```

你可以把对浏览器的测试同服务器#INCLUDE 指令结合起来，为用户全面显示不同的页面。下边的脚本测试浏览器。如果浏览器同 Internet Explorer 3.0 或更高版本兼容，它就把各种颜色选项包括在一个页面中；否则，就给出一个较一般化的页面。

```
    <% browser = Request.ServerVariables(" HTTP_USER_AGENT " )%>
    <%If browser =  " Mozilla/2.0 (compatible; MSIE 3.08;Windows NT) "
Then%>
        <!--#INCLUDE FILE=" /myapp/ColoredTable.asp " -->
    <%Else%>
        <!--#INCLUDE FILE=" /myapp/PlainTable.asp " -->
    <%End If%>
```

第二十六章　调　试　页　面

既然你为 Web 应用编写脚本，你就可以测试它，找出错误。为了帮助你，Microsoft Visual InterDev 提供一种全性能的调试器，你可以用来追踪客户脚本和服务器脚本的错误。

26.1　调试客户脚本

你可以用下述诸方式中任何一种方式调试客户脚本：

- 在你的 Visual InterDev 解决方案中运行一个页面，页面上包含要调试脚本。
- 把 Visual InterDev 调试器连接到一个（页面）进程上，这个进程在 Internet Explorer 中运行着。
- 对脚本中语法错误或运行期间错误作出响应，这便是“及时”（just-in-time）调试。
- 在脚本中包含一个语句，这个语句启动调试器。

注意：要在 Internet Explorer 中调试客户脚本，必须使用 Internet Explorer 4.0。调试必须在 Internet Explorer 中开启（这是默认状况）。

我们建议，在调试的时候不要使用 Internet Explorer 的 ActiveDesktop 模式，或者把 Internet Explorer 中的选项设置为在一个新进程中启动浏览器的每个新实例。

如果 Web 页面包含客户和服务器两种混合脚本，你可以用 Visual InterDev 调试器对两种脚本进行调试。有关细节，参见本章后边“调试客户与服务器混合脚本”。

对 ASP 页面开启客户脚本调试

在可以对 ASP 页面中客户脚本进行调试之前，你必须开启调试。你可以对你的 ASP 应用手动开启调试，如附录 A“排除故障”中的“在服务器上开启 ASP 调试”所描述的那样，另外，Visual InterDev 可以按照需要自动开启服务器上的调试。

注意:如果你企图只用 .htm 文件中的客户脚本进行工作，则不需要开启服务器调试或执行下述过程。

自动开启 ASP 页面的客户脚本调试

1. 在 Project Explorer 中，右键单击这个项目，并选择 Properties 以显示 Property Pages 对话框。
2. 选择 Launch 选项卡。
3. 在 Server script 下检查 Automatically enable ASP server-side debugging on launch（自动开启 ASP 服务器方的调试。）

注意:要调试 ASP 页面上的脚本，必须运行 Internet Information

Server(IIS) 4.0 或更高版本。

在设置这个选项的情况下，你每次启动调试会话，Visual InterDev 就检查服务器是否为调试做了配置。这方面的配置包括：

- 把 IIS 应用设置成在其自己的存储空间中运行（按照 COM 术语，在“进程外”(out of process)运行）。
- 开启 IIS 应用的调试选项。
- 组建一个“微软事务处理服务器”(Microsoft Transaction Server，MTS)包，以便把调试器连接到 Web 应用上。包的识别是在你第一次启动调试会话时设置的，设置的时候要求提供名字和口令。

在你退出调试会话时，Visual InterDev 把服务器调试设置和“进程外”设置恢复成以前的值。

使用一种解决方案调试客户脚本

如果你使用 Vusual InterDev 的一种解决方案，你可以启动调试器来调试文件。

使用一种解决方案调试脚本

1.在 Visual InterDev 中，打开包含你希望调试页面的项目，再把页面装到编辑器中。

2.使得这个页面就是你项目开始的页面。在 Project Explorer 中，右键单击这个页面，并选择 Set as Start Page。

3.在脚本中设置断点，以便进行调试。有关细节，参见第五章“演练”中

的“设置断点”一节。

4. 在 Debug 菜单中，选择 Start。

5. 如果对 ASP 页面中的客户脚本进行调试没有像上述那样予以开启，Visual InterDev 就显示一条消息。你的选择将是：

   - 如果你用一个 .htm 文件（不是 .asp 文件）中的客户脚本进行工作，并且在调试会话期间你将不导航到 ASP 页面上，则选择 No 继续进行调试。
   - 如果你用一个 .asp 文件进行工作，或者在调试会话期间你将导航到 .asp 文件上，则选择 Yes 使 Visual InterDev 自动开启客户方的 ASP 调试。

6. 如果在 ASP 页面中开启了客户调试，并且这是打开项目以来第一次启动调试会话，你就会得到提示：提供标识服务器上调试进程的用户信息（即便你当前以用.htm 文件进行工作，在你导航到 ASP 页面的情况下也会得到提示。）

注意：在 Visual InterDev 对服务器建立了适当配置的情况下，你若第一次对当前项目启动调试会话，就会感到迟后。

Visual InterDev 启动 Internet Explorer，并把页面装载到里面。当 Internet Explorer 达到断点的时候，就停下来，并在编辑器窗口中显示原代码。如果断点在事件处理程序脚本中，你必须触发这个事件，以便达到断点。

7. 如果你找到一个错误，则解决它，然后保存这个文件。如果你没有文件的工作副本，就用右键单击 Project Explorer 中的文件名字，在进行修

改以前选择 Get Latest Version。

8.在 Debug 菜单中，选择 Restart。

注意:使用调试器命令的有关细节，参见 Visual InterDev 在线文档。

在运行文档中调试客户脚本

如果一个客户脚本已经在 Internet Explorer 中运行，并且你发现了一个问题，你可以把脚本停下来，在这个地方调试它。你可以在 Visual InterDev 中或在 Internet Explorer 中调试一个运行文档。

注意:如果是以一个 ASP 页面进行工作，必须在服务器上开启调试。有关细节，参见附录 A“排除故障”中的“在服务器上开启 ASP 调试”。

只当你开启连接的时候，你才可以连接到一个运行文档上。

开启及时调试

1.在 Tools 菜单中，选择 Options。

2.在 Options 对话框中，选择 Debugger。

3.在 Script 下，检查 Attach to programs running on this machine（连接到这个机器上运行的程序上）。

你可以直接在 Visual InterDev 中连接到一个文档上。

调试 Visual InterDev 中运行的文档

- 在 Visual InterDev 中，选择 Debug 菜单上的 Processes。在 Processes 对话框中，选择 Microsoft Internet Explorer，然后选择 Attach，然后选择你希望调试的脚本。

或者

如果你已经连接到运行文档上，在 Visual InterDev 中，选择 Debug 菜单中的 Break。调试器便停在脚本下一个要执行的语句上。

如果 Visual InterDev 尚未运行，你可以从 Internet Explorer 启动调试器。

从 Internet Explorer 调试运行文档

1. 在 Internet Explorer 中，选择 View 菜单中的 Script Debugger，然后选择 Break at Next Statement。在浏览器中，触发一个事件，这个事件调用你希望调试的脚本。

   或者

   在 Internet Explorer 中，选择 View 菜单中的 Script Debugger，然后选择 Open。

2. 启动 Visual InterDev 中的一个新实例，出现打开一个项目的提示。如果 Visual InterDev 已经在运行，就启动第二个实例。打开其中包含要调试文件的项目。
3. 如果这个项目已经在 Visual InterDev 另一个实例中打开，你就不能再一次打开项目，因而 Visual InterDev 创建一个新解决方案，作为替代项目。
4. 要调试的页面装到编辑器中。如果需要的话，就获取这个页面的工作副本。如果项目已经打开，页面就作为只读文件装载到这个新项目中。

如果你改变了这个文件，就保存它，并再把它放到服务器上。在再一次运

行该脚本之前，在 Internet Explorer 中刷新该文件。

注意：有关使用调试器命令，详见 Visual InterDev 在线文档。

在响应错误或调试器语句的过程中调试客户脚本

如果 Internet Explorer 遇到一个语法错误或运行期间错误，你可以用及时(just-in-time)调试找到并解决它。你可以在脚本中放一个语句，例如 Visual Basic 的 Stop 语句、Scripting Edition (VBScript)或 JScript 的调试器语句，以便在一个脚本中启动调试器。

你可以启动调试器，对错误或调试器语句作出响应，但是只有在开启及时调试的情况下才行。

开启及时调试

1. 在 Tools 菜单中选择 Options。
2. 在 Options 对话框中，选择 Debugger。
3. 在 Script 下，检查 Just-In-Time debugging。

在响应错误或调试器语句中调试

1. 如果 Internet Explorer 遇到一个语法错误或启动调试器的语句，就显示一个错误消息，提示你进行调试。选择 Yes。
2. 启动 Visual InterDev 中的一个新实例。如果 Visual InterDev 已经在运行，就启动第二个实例。
3. 打开包含要调试文件的项目。如果这个项目已经在 Visual InterDev 的另一个实例中打开，你就不能再一次打开它，因而 Visual InterDev 创建

一个替代的新项目。

把要调试的页面装载到编辑器中。如果需要的话，就获取这个页面的工作副本。如果该项目已经打开，页面就作为只读文件装载到这个新项目中。

注意：如果你在调试由 .asp 文件生成的客户脚本，在错误消息中报告的行号便是指当前显示在浏览器中 HTML 文档中的行。这些行通常不对应于原来 .asp 文件中的行号，这是因为服务器脚本不出现在 .asp 文件的 HTML 输出中。详细内容，参见本章后边“调试客户与服务器混合脚本”。

如果你改变了这个文件，就保存它，并再把它放到服务器上。在再一次运行该脚本之前，在 Internet Explorer 中刷新该文件。

注意：有关使用调试器命令，参见 Visual InterDev 在线文档。

26.2 调试服务器脚本

从 Microsoft Visual InterDev 你可以调试在 Internet 信息服务器(IIS)上运行的服务器脚本。如果 IIS 在你的计算机上运行，你可以以调试客户脚本类似的方式调试服务器脚本。如果服务器是在另外一台计算机上，则可以从你的计算机上使用“远程调试”(remote debugging)，查找服务器脚本中的错误。有关细节，参见第二十九章“综合任务”中的“远程调试”一节。

你可以使用下述方式中的任何一种调试服务器脚本：

- 从你的 Visual InterDev 解决方案中运行一个包含要调试脚本的页面。
- 把 Visual InterDev 调试器连接到一个（页面）进程上，这个进程在

Internet Explorer 中运行着。

- 对脚本中语法错误或运行期间错误作出响应。这便是所谓“及时”（just-in-time）调试。
- 在脚本中包含一个语句，这个语句启动调试器。

注意:在 ASP 页面中调试脚本，你必须运行“Internet 信息服务器”(IIS)的 4.0 版本或更高版本。

如果 Web 页面包含客户和服务器两种混合脚本，你可以用 Visual InterDev 调试器对两种脚本进行调试。有关细节，参见本章后边“调试客户与服务器混合脚本”。

对 ASP 页面开启服务器脚本调试

在可以对 ASP 页面中客户脚本进行调试之前，你必须开启调试。你可以对你的 ASP 应用手动开启调试，如附录 A“排除故障”中的“在服务器上开启 ASP 调试”所描述的那样。另外，Visual InterDev 可以按照需要自动开启服务器上的调试。

自动开启 ASP 页面的脚本调试

1. 在 Project Explorer 中，右键单击这个项目，并选择 Properties，以便显示 Property Pages 对话框。
2. 选择 Launch 选项卡。
3. 在 Server script 下，确保选中 Automatically enable ASP server-side debugging on launch（自动开启 ASP 服务器方的调试。）

在设置这个选项的情况下，你每次启动调试会话，Visual InterDev 就检查服务器是否为调试做了配置。这方面的配置包括：

- 把 IIS 设置成在其自己的存储空间中运行（按照 COM 术语，在“进程外”(out of process)运行）。
- 开启 IIS 应用的调试选项。
- 组建一个 Microsoft Transaction Server(MTS)包以便把调试器连接到 Web 应用上。包的识别是在你第一次启动调试会话时设置的，设置的条件是要求你提供名字和口令。

  注意：你可以手动在服务器上执行前两个步骤。有关细节，参见附录 A“排除故障”中的“在服务器上开启 ASP 调试”。

在你退出调试会话时，Visual InterDev 把服务器调试设置和“进程外”设置恢复成以前的值。

在解决方案中调试服务器脚本

如果你以 Visual InterDev 解决方案进行工作，则可以启动调试器调试服务器脚本。

注意：在调试服务器脚本之前，保证已经按上边所述，已经开启调试，或处于附录 A“排除故障”中的“在服务器上开启 ASP 调试”状况之下。

在服务器中调试服务器脚本

1. 在 Visual InterDev 中，打开包含你希望调试服务器脚本的项目。
2. 使得这个页面就是你项目开始的页面。在 Project Explorer 中右键单击

这个页面，并选择 Set as Start Page。

3. 在服务器脚本中设置断点，以便进行调试。
4. 在 Debug 菜单中，选择 Start，以便启动项目。
   Visual InterDev 试图把调试器连接到在服务器上运行的文档上。
5. 如果这是打开当前项目以来你第一次启动调试器，便提示你提供用户信息，以便标识在服务器上的调试过程。键入你的域名、名字（窗体是：域名\名字）和口令。
6. 浏览器打开，你可以进行调试。当服务器执行到有断点的行时，调试器就以加亮的行显示编辑器中的页面。
7. 解决任何错误，保存文件，然后在 Debug 菜单中选择 Restart。如果你没有文件的工作副本，就用右键单击 Project Explorer 中这个文件的名字，选择 Get Latest Version，然后进行修改。

在运行文档中调试服务器脚本

如果调试已经在服务器上对你的项目开启，并且在应用运行的时候你检测到一个错误，你就可以把调试器连接到这个应用上。有关在服务器上开启调试的细节，参见附录 A“排除故障”中这“在服务器上开启 ASP 调试”。

只有连接开启的情况下，你才可以连接到运行的文档上。

开启及时调试

1. 在 Tools 菜单中选择 Options。
2. 在 Options 对话框中选择 Debugger。

3. 在 Script 下，选中 Attach to programs running on this machine（连接到这个机器上运行的程序上）。

调试运行的脚本

1. 在 Visual InterDev 中，选择 Debug 菜单中的 Processes。在 Processes 对话框中，选择 Active Pages，再选 Atach。
2. 在 Running Documents 窗口中，选择你希望调试的脚本。
3. 设置断点，然后选择 Debug 菜单中的 Restart，或刷新浏览器器中的文档。

注意:有关使用调试器命令的细节，参见 Visual InterDev 在线文档。

在响应错误或调试器语句过程中调试服务器脚本

如果对服务器上的 IIS 应用开启了调试，而且这个服务器遇到服务器脚本的一个语法错误或运行期间错误，你可以使用及时调试找到并解决这个错误。你还可以在你的脚本中放入一个语句，例如 VBScript 的 Stop 语句或 JScript 的调试器语句，以便在脚本中启动调试器。有关在服务器上开启调试的细节，参见附录 A“排除故障”中的“在服务器上开启 ASP 调试”。

注意:如果调试器安装在服务器计算机上，服务器不把错误信息传递给客户机。相反，在服务器计算机的监视器上显示一条错误消息。更多内容，参见附录 A“排除故障”中的“及时调试服务器页面”一节。

只当在开启及时调试的条件下你才可以启动调试器，响应错误或调试器语句。

开启及时调试

1.在 Tools 菜单中，选择 Options。
2.在 Options 对话框中，选择 Debugger。
3.在 Script 下，检查 Just-In-Time debugging（及时调试）。

在响应错误或调试器语句过程中调试服务器脚本

1.当出现一个错误消息，提示你启动调试器的时候，选择 Yes。
2.启动 Visual InterDev 的新实例，提示你打开项目。如果 Visual InterDev 已经在运行着，就启动第二个实例。
3.打开包含要调试文件的项目。如果项目已经在 Visual InterDev 的另一个实例中打开，你就不能再一次打开它，Visual InterDev 就创建一个新解决方案以替代项目。

把要调试的页面装入到编辑器中。如果需要，获取这个页面的工作副本。如果项目已经打开，页面就作为只读文件装入到新项目中。

注意:有关使用调试器命令的细节，参见 Visual InterDev 在线文档。

如果服务器已经对应用开启，错误就作为页面文本显示在浏览器上。在这种情况下，按照上边所述，打开包含 Visual InterDev 中页面的项目，并在这里启动调试器。

26.3 调试客户和服务器混合脚本

许多 ASP 页面包含客户和服务器两种脚本。你可以在客户和服务器两种脚本中设置断点。每种脚本在执行中都可以在断点处调用调试器。

如果服务器没在你的计算机上运行，你可使用“远程调试”(remote debugging)调试页面中的服务器脚本。有关如何组建远程调试的细节，参见第二十九章“综合任务”中的“远程调试”。

注意:建议你在调试的时候不要使用 Microsoft Internet Explorer 4.0 的 Active Desktop 模式。

为 ASP 页面开启调试

在 ASP 页面中调试脚本以前，你必须开启调试。你可以手动为 ASP 应用开启调试，如附录 A“排除故障”中的“在服务器上开启 ASP 调试”描述的那样。此外，Visual InterDev 还可以按照需要自动开启调试。

注意:要在 ASP 页面上调试脚本，必须运行微软公司的“Internet 信息服务器”(IIS) 4.0 版本或更高版本。

在 ASP 页面上开启脚本调试

1. 在 Project Explorer 中，右键单击项目并选择 Properties，以便显示 Property Pages 对话框。
2. 选择 Launch 选项卡。
3. 在 Server script 下，确保选中 Automatically enable ASP server-side

debugging（自动开启服务器方的调试）。

当这些选项都设置完毕时，你每启动一次调试会话，Visual InterDev 就检查一次服务器是否已做调试配置，其中包括：

- 把 IIS 应用设置成在其存储空间中运行（按照 COM 术语，在“进程外”(out of process)运行）。
- 开启 IIS 应用的调试选项。
- 组建一个 Microsoft Transaction Server(MTS)包，从而允许你把调试器连接到 Web 应用上。包的识别是在你第一次启动调试会话时设置的，设置的时候要求你提供你名字和口令。

在你退出调试会话时，Visual InterDev 把服务器调试设置和“进程外”设置都恢复成以前的值。

调试包含客户和服务器脚本的页面

若一个页面包含服务器和客户两种脚本，而你只在这个页面上的客户脚本中设置一个断点，调试器便追踪这个断点事件，即使在服务器脚本执行之后这个断点在文件中的位置可能变化很大。如果服务器脚本把客户脚本在这个页面上写入多次，调试器就分别追踪每个断点。

你可以在服务器脚本、客户脚本或两种脚本上设置断点。如果你在两种脚本上设置断点，调试器就先停在服务器脚本断点上。当你继续运行时，页面发送到浏览器上，调试器便接着停在客户脚本断点上。

调试包含客户和服务器两种脚本的页面

1.在 Visual InterDev 中，在客户脚本和服务器脚本上你希望调试的行中设置断点。

2.要使得一个页面就是你项目开始的页面。在 Project Explorer 中用右键单击该页面，选择 Set as Start Page。

3.在 Debug 菜单中选择 Start。

Visual InterDev 企图把调试器连接到服务器上运行的文档上。

4.如果这是打开当前项目以来你第一次启动调试器，便提示你提供用户信息，以便标识在服务器上的调试过程。键入你的域名、名字（窗体是：域名\名字）和口令。

5.进行调试。当服务器执行到有断点的行时，调试器就以加亮的行显示编辑器中的页面。

当你离开服务器脚本时，页面将继续执行，直到进入另一个服务器脚本为止。当调试器完成对服务器脚本的调试时，服务器就把页面发送给浏览器，显示出来。

6.如果需要，就触发一个事件（例如敲击按钮），于是运行你希望调试的客户脚本。

调试器停在断点处，显示正被浏览器处理的页面型式。

注意:关于使用调试器命令的细节，参见 Visual InterDev 在线文档。

当你调试包含服务器和客户两种脚本的页面时，要记住，服务器脚本有可能把 HTML 文本客户脚本插入到这个页面中。例如，用几行服务器脚本就可以动态创建数据库信息之外的大型表格，或者把客户脚本写到这个页面上或在这个

页面上重新进行安排。

在处理这个页面之后，服务器便去掉这个服务器脚本，因此正被浏览器处理的页面同它在 Visual InterDev 编辑器中包含有服务器脚本和客户脚本时的样子看上去会完全不同。有关如何处理服务器脚本的细节，参见第二十三章“脚本编程概念”中的“了解脚本处理”一节。

调试嵌入的服务器脚本

有一种特殊情况，就是你希望调试的服务器脚本出现在客户脚本块之内。在下述情况中，服务器脚本先从一种窗体中提取一个值，再把它存储到一个变量中。接着，在客户脚本块中，一个嵌入的服务器脚本块动态地把这个变量值插入到一个客户语句中。

```
<%loginName = = Request.Form("loginName")%>

<SCRIPT LANGUAGE="VBScript">
Sub ShowWelcome
   txt = "<%=loginName%>"
   MsgBox("Welcome, " & txt)
End sub
</SCRIPT>
```

注意：当你在客户脚本中编写服务器脚本时，编辑器不遵从服务器脚本的彩色编码约定。

如果你在嵌入的服务器脚本行上设置断点，它不清楚你是希望把调试器停在服务器脚本上（<%=loginName%>），还是希望过一会儿停在客户脚本中（txt=），因此 Visual InterDev 为你提供两种选择：

- **只是客户断点**。这个断点被解释成客户断点，因而在服务器处理嵌入的服务器脚本时，调试器不停下来。
- **服务器和客户断点**。调试器停两次，第一次是处理服务器脚本的时候，再一次是到达客户脚本中同一行的时候。

默认选择仅仅是客户断点。要停两次，你就要开启调试器选项。

为嵌入的服务器脚本开启断点

1. 在 Tools 菜单中选择 Options，然后打开 Debugger 节点。
2. 选择 Insert breakpoints in Active Server Pages for breakpoints in client script。

在你运行调试器并在嵌入的服务器脚本上中断的时候，不直接停在包含服务器脚本的行上，而是停在紧接着服务器脚本行后边的客户脚本行或 HTML 行上。在有些情况中，这会使断点在嵌入的服务器脚本之前许多行上出现。

例如，如果你在嵌入的服务器脚本上设置一个断点（<%=loginName%>），如下例所示，调试器就停在<HTML>标记处。

```
<%loginName = Request.Form("loginName")%>
<HTML>
<HEAD>
<SCRIPT LANGUAGE="VBScript">
Sub ShowWelcome
    txt = "<%=loginName%>"
    MsgBox("Welcome, " & txt)
End sub
</SCRIPT>
</HEAD>
```

之所以出现这种情况，是因为在服务器处理页面的时候，除了服务器脚本，其它任何东西一律忽略。服务器不“看”嵌入服务器脚本之前的 HTML 行和客户脚本行，因此包含嵌入服务器脚本的开始行是服务器脚本紧接着的一行。

要移动嵌入服务器脚本，就要使用调试器的 Step Into 命令。

26.4 调试 Global.asa 文件

调试 Global.asa 文件同调试 .asp 文件有以下诸方面不同：

- Global.asa 文件不能是开始页面。要调试 Global.asa 文件，你必须请求一个 ASP 页面。在请求 ASP 页面的时候，服务器处理 Global.asa 页

面。

- Global.asa 文件中的过程是事件驱动的。这与 .asp 文件中的内联脚本不同。
- Global.asa 文件中的过程通常对每个应用或对每次会话只运行一次：
  - Application_OnStart 过程在基于 ASP 应用中任何页面被第一访问的时候执行。
  - Application_OnEnd 过程只在应用停止的时候执行。
  - Session_OnStart 过程只对每个用户会话执行一次。
  - Session_OnEnd 过程只在用户会话出现超时时执行，或在脚本明确调用会话对象的 Abandon（放弃）方法时执行。

在 Global.asa 文件开启调试

在 Global,asa 文件中调试脚本之前，你必须为 ASP 页面开启调试。你可以手动为 ASP 应用开启调试，如附录 A“排除故障”中的“在服务器上开启 ASP 调试”所述的那样。此外，Visual InterDev 可以按照需要自动开启调试。

在 ASP 页面中自动开启脚本调试

1. 在 Project Explorer 中，右键单击项目并选择 Properties，以便显示 Property Pages 对话框。
2. 选择 Launch 选项卡。
3. 在 Server script 下，确保选中 Automatically enable ASP server-side debugging on launch（自动开启 ASP 服务器方的调试。）

在设置这个选项的情况下，你每次启动调试会话，Visual InterDev 就检查服务器是否为调试作了配置。这方面的配置包括：

- 把“Internet 信息服务器”（IIS）应用设置成在其存储空间中运行（按照 COM 术语，在“进程外”(out of process)运行）。
- 开启 IIS 应用的调试选项。
- 组建一个 Microsoft Transaction Server(MTS)包，从而允许你把调试器连接到 Web 应用上。包的识别是在你第一次启动调试会话时设置的，设置的时候要求你提供名字和口令。

注意:你可以手动在服务器上执行前两个步骤。有关细节，参见附录 A“排除故障”中的“在服务器上开启 ASP 调试”。

在你退出调试会话时，Visual InterDev 把服务器调试设置和“进程外”设置都恢复成以前的值。

在 Global.asa 文件中调试脚本

要调试 Global.asa 文件，必须组建对该文件中脚本的一次调试器调用，然后启动 ASP 应用。

从 Global.asa 文件中调用调试器

1. 在编辑器中打开 Global.asa 文件，然后在脚本中设置一个断点。

   或者

   选取一个明确启动调试器的语句，例如 VBScript 中的 Stop 或 JavaScript 中的调试器。在你希望步入的任何语句之前，把这个语句放到过程的开

头。

2.从浏览器中的当前项目中请求一个 ASP 页面。这会使 IIS 运行 Global.asa 文件。

在对错误响应中调试 Global.asa 文件

如果 Global.asa 文件中有一个错误（语法错误或运行事件错误），服务器便停止含有这个错误的过程。如果对这个 ASP 应用开启调试，服务器便提示你启动调试器并显示一条错误消息。如果没对这个 ASP 应用开启调试，错误消息便送到客户浏览器上。不管在哪种情况下，含有错误的过程都要停下来。

注意:如果调试器安装到服务器计算机上，服务器便不把错误信息传递给客户，而是把错误消息显示在服务器计算机的监视器上。详细内容，参见附录 A“排除故障”中的“及时调试服务器页面”。

只在开启“及时调试”情况下,你可以启动调试器对错误或调试器语句作出响应。

开启及时调试

1.从 Tools 菜单中选择 Options。
2.在 Options 对话框中选择 Debugger。
3.在 Script 下选中 Just-In-Time debugging（及时调试）。

在对错误响应过程中调试 Global.asa 脚本

1.在出现错误消息提示你启动调试器动时候，选择 Yes。
2.启动 Visual InterDev 的一个新实例,并提示你打开一个项目。如果 Visual

InterDev 已经在运行，就启动第二个实例。

3. 打开包含要调试 Global.asa 文件的项目。如果这个项目已经在另一个 Visual InterDev 实例中打开，你不能再一次打开它，于是 Visual InterDev 创建一个新解决方案，取代项目。
   Global.asa 文件被装载到编辑器中。如果需要，便获取该页面的工作副本。如果项目已经打开，Global.asa 文件就作为只读文件装到这个新项目中。

注意：关于使用调试器命令的细节，参见 Visual InterDev 在线文档。

如果没对应用开启服务器调试，错误就按照页面的文本显示在浏览器上。在这样情况下，包含这个页面的项目在 Visual InterDev 中打开，并开始调试，如上所述。

重新启动 Global.asa 文件

你不能停下来用刷新 Global.asa 文件的办法来重新启动脚本。要返回到 Application_OnStart 和 Session_OnStart 过程，你就必须刷新这个文件或再一次触发该事件，否则就要重新启动应用。

所有过程都返回到 Global.asa 文件

- 在编辑器中，改变 Global.asa 文件，然后把它用到服务器上。
  或者
  停下来重新启动 Web 服务器。
  这便重新启动应用和会话，再一次在 Global.asa 文件中运行过程。

第二十七章　把脚本包装成对象

微软的“脚本部件”（scriptlets，小脚本）为你提供一种轻松方式，用来创建强有力又可重复使用的控件。你可以使用 Web 脚本语言，例如 Visual Basic Script Edition(VBScript)和 ECMAScript(一种基于 JScript 2.0 和 JavaScript 1.1 的标准语言）语言创建小脚本。在创建小脚本之后，你就可以登记并像使用 ActiveX 控件一样使用它。

小脚本：

- 为脚本作者提供创建控件的能力。
- 为脚本作者提供范围广泛的系统服务的访问权力。
- 易于创建与维护。
- 付出少效率高。

27.1　小脚本类型

你可以创建两类小脚本（scriptlets）：

- **DHTML 小脚本**　一种基于动态 HTML（DHTML）语言的 Web 页面，可以在任何支持控件的应用中作为一种控件使用，如 Visual Basic 或 Internet Explorer 4.0 等。

- **服务器小脚本** 你可以用作一种 COM 部件的小脚本。服务器小脚本只包括脚本（不包括 HTML 或其它界面元素）。你可以把服务器小脚本作为下述应用的 COM 部件使用：如微软公司的“Internet 信息服务器”（IIS）、“窗口脚本宿主”（Windows Scripting Host）和可以支持 COM 部件的任何应用。

注意：服务器小脚本不在这里讨论。关于建设和使用服务器小脚本的信息，可访问微软脚本编程 Web 站点：http://www.microsoft.com/scripting/。

DHTML 小脚本用作应用中的可视控件，适应于在客户应用中，例如在 Internet Explorer 和 Visual Basic 应用中，作为显示控件。用 DHTML 小脚本，你便可以使用 Web 页面的图形和超文本功能作为下述应用的可视界面：例如你可在 Web 页面上、Visual Basic 中或在其它基于 COM 的环境中显示日历控制。

DHTML 小脚本如何工作

如果你创建一个 DHTML 小脚本（scriptlet），必须注意两个方面。第一个方面是可视界面。你可以创建这样的界面用于使用 DHTML 的 Web 页面。

第二个方面是小脚本控件界面：属性、方法和事件，这些允许你使用小脚本作为控件。使用脚本创建这些控件元素要遵从下述诸方面的约定：方法和属性如何显示，事件如何俘获并转发给宿主(host)应用。

例如，一个 DHTML 小脚本可以定义一个动画片，这个动画片用于移动页面上文本并改变文本的尺寸。你可以使用 DHTML 定义文本并设置它的颜色、尺寸和位置。然后你可以使用脚本定义其它应用 用来设置速度的属性、动画文本的

方向，还可以定义其它应用启动、停止和暂停动画的方法。

注意:有关如何在 Web 页面中使用动态 HTML 的知识，可以查阅览下述位置的 Internet Client SDK 文档：http://www.microsoft.com/msdn/sdk/inetsdk/help/default.htm/。

在创建一个 DHTML 小脚本之后，你就可以把它作为任何控件来使用。你可以把它加到工具条上，在窗体中下拉它，或者把它加到一个 Web 页面上。在宿主应用中，这种 DHTML 是由“小脚本容器对象”（scriptlet container object）置于宿主地位的。这种容器对象为 DHTML 小脚本创建一个窗口，并为宿主应用提供一种方式，让宿主应用规定小脚本在什么地方显示，以多大尺寸显示等。小脚本容器对象还为你提供一种界面，允许你设置和获取 DHTML 小脚本的属性、调用它的方法以及响应它的事件。

27.2 创建 DHTML 小脚本

DHTML 小脚本是一个 HTML 文件，文件中包含普通的用于定义小脚本外观和感觉的 HTML 语言。DHTML 小脚本还包含用于定义 DHTML 小脚本如何提交属性、方法和事件的脚本。

创建 DHTML 小脚本

1. 创建一个新文件。
2. 使用 DHTML 为你的小脚本定义可视界面。

3.为你的小脚本定义属性、方法和事件。有关细节，参见本章后边“在 DHTML 小脚本中定义属性和方法”。你还可以为你的小脚本定义事件处理。有关细节，参见本章后边的“揭示 DHTML 小脚本中的事件”。

在你创建 DHTML 小脚本之后，你可以把它作为控件来使用。Visual InterDev 包含一种特性，这种特性允许你把小脚本轻松地加到 Toolbox 上。有关细节，参见本章后边的“把 DHTML 小脚本加到你的应用上”。

27.3 在 DHTML 小脚本中定义属性和方法

DHTML 小脚本可以展现任何数目的属性和方法。你可以用下述两种方法做这件事：

- **创建一个 JavaScript public_description（公共描述）对象** 对象构造函数（object's constructor function）明确定义小脚本可以展现什么特性和方法。
- **使用默认界面描述** 你可以遵从特定命名约定只创建函数的办法展现属性和方法。

一般说来，使用一个 public_description 对象同使用下述内容有关：使用 JavaScript 和使用默认界面描述，而后者又使用 VBScript。根据严格的语言要求，如果你使用一个 public_description 对象，则 public_description 对象本身必须用 JavaScript 创建出来。任何其它函数，例如定义属性和方法的函数，可以使用脚本编程语言编写。同样，尽管使用默认界面描述同使用 VBScript 有关，你还是可以使用任何语言编写函数。

使用 public_description 对象有几种好处。你可以对要现示其属性和方法的变量和函数使用任何名字，这是因为你可以在 public_description 对象中为它们分配公共名字。此外，使用 public_description 对象可以为你提供方便的汇总方式，为小脚本显示的属性和方法提供方便的文档方式。

创建 public_description 对象

public_description 对象是一种 JavaScript 对象，这个对象对由对象构造函数定义的属性和方法提供运行期间访问机制。没有在这个构造函数中明确声明的任何性能都是不可以使用的。

注意：必须使用小写字母为 public_description 命名，如下所示，否则小脚本的属性和方法将得不到正确揭示。

骨架 public_description 连同它的构造函数看上去如下：

```
<SCRIPT LANGUAGE="JavaScript">
   var public_description = new CreateScriptlet();
   function CreateScriptlet(){
      // statements here to define properties and methods
   }
</SCRIPT>
```

注意：你不要使用构造函数定义事件。细节参见本章后边的“揭示 DHTML 小脚本中的事件”。

在你创建 public_description 对象的时候，你分配给它的构造函数可以用任何名字，只要相应的函数出现在这个小脚本中的某个地方就行。在构造函数中声明你希望用语法揭示的属性和方法，如下表所示。

在构造函数中宣布	创建
This.*PropertyName=expression*	以表达式声明属性。在用户获取这个属性时，表达式的值传递给调用容器。用户设置属性时，表达式值被更新
This.get_*PropertyName=function*	以函数宣布属性。用函数定义调用的属性可以使用任何有效的脚本语言。要使属性成为只读的，不要宣布 put_function；要使属性成为只写，不要宣布 get_function
This.put_*PropertyName=function*	
This.*method=methodFunction*	方法

例如，下边的小脚本包括一个选取框（marquee），用来揭示用于设置文本和颜色的属性和方法。这个构造函数定义了两种属性：版本和“选取框文本”（marqueetext）。版本是一个简单值；选取框文本是按照一组函数创建的，这组函数描述如何按照条件获取和设置属性值。

构造函数还定义了方法 togglecolor（转换颜色），它在 setcolor 的小脚本中调用一个函数。

```
<HTML>
<HEAD>
<TITLE>Scriptlets!</TITLE>
<SCRIPT LANGUAGE="JavaScript">
// public_description object used to declare scriplet
var public_description = new scriptletobject();

// general object description
function scriptletobject()
{
   this.version = "1.0";                  //property
   this.get_marqueetext = readtext;       //property (read)
   this.put_marqueetext = writetext;      //property (write)
   this.togglecolor = setcolor;           //method
}

function readtext(){
   if (mrq1.innerText == ""){
      return "No text";}
   else{
      return mrq1.innerText;}
}
```

```
function writetext(newtext){
   if (newtext != ""){
      mrq1.innerText = newtext;}
}

function setcolor(){
   if (f1.color == "#ff0000"){
      // red, make it blue
      f1.color = "#0000ff";}
   else{
      // not red, make it red
      f1.color = "#ff0000";}
}
</SCRIPT>
</HEAD>

<BODY>
<B>Sample scriptlet</B><br>

<FONT ID="f1" COLOR="red">
<MARQUEE ID="mrq1">Scriptlets are fun and easy!</MARQUEE>
</FONT>
</BODY>
</HTML>
```

创建默认界面说明

如果在小脚本中没有定义的 public_description 对象，小脚本容器对象就使用该小脚本中的变量和函数，按照命名约定揭示属性和方法。

为揭示小脚本的属性和方法，要使用下述约定：

- 为创建读/写属性，要声明作用于页面级（即不在函数内定义）的一个变量，并且给它一个 public_前缀。
- 要用函数创建一种可读属性，则须用前缀 public_get_定义一个函数。
- 要用函数创建一种可写属性，则须用前缀 public_put_定义一个函数。
- 为创建一种方法，则须用前缀 public_定义一个函数。

注意：在揭示一种属性的时候，其在宿主（host）应用中的名字没有 public_前缀。例如，如果你在小脚本中定义了一种名字叫做 public_MyTitle 的属性，它在宿主应用中的名字就是 MyTitle。

下表给出在一个小脚本中的变量和函数的例子，还给出在宿主应用中揭示的结果界面。

例子	揭示的内容	在容器中使用
ver public_Color=" red "	属性	vColor=SC1.Color SC1.Color=" blue "
function public_get_C()	属性（读）	X=SC1.C
function public_put_C(param)	属性（写）	SC1.C=" test "
function public_look(param)	不可用（无 public_前缀）	SC1.look(param)
function look()	不可用（无 public_前缀）	
function get_C()	不可用（无 public_前缀）	
ver Color=red;	不可用（无 public_前缀）	
ver get_Color=red;	不可用（无 public_前缀）	

下边例子给出一个简单小脚本（scriptlet），其中包含名字叫做“p1”的一个段落。这个段落后边的脚本块揭示叫做 Text（文本）的一种属性和一个叫做 color（颜色）的属性。这些属性都是用 get（获取）和 set（设置）函数定义的。这些函数都表示如何按条件设置属性。这个小脚本还揭示了叫做 settext 的一种方法。

```
<HTML><HEAD></HEAD>
<BODY>
<FONT ID="f1" color="black">
<P ID="p1">This is a paragraph of text.</P>
</FONT>

<SCRIPT LANGUAGE="JavaScript">
var public_text = p1.innerText;
function public_get_color(){
   return f1.color;
}
function public_put_color(color){
   f1.color = color;
}
function public_settext(newtext){
   p1.innerText = newtext;
}
</SCRIPT>
</BODY></HTML>
```

注意：这个小脚本保留函数名字前缀 public_get_和 public_put_来定义特性。例如，如果小脚本包含一个名为 public_get_MyText 的函数，它就被当作 MyText 属性来对待。如果你企图用语法 SC1.get_MyText，

作为一种方法来调用函数 public_get_MyTex，就会产生错误。这是因为这个函数本身只能看作是名为 MyText 的一种属性。

27.4 揭示 DHTML 小脚本中的事件

在你的宿主应用中使用 DHTML 小脚本（scriptlet）的时候，这个应用就会通告在这个小脚本中出现的有关事件。一个 DHTML 小脚本可以揭示两类事件：

- **标准 DHTML 事件**，例如 onclick(敲击)事件或 onkeypress(按键)事件。
- **客户事件**，你定义的事件或标准事件中没提供的 DHTML 事件。例如，小脚本在属性值改变的时候可以激发一种事件。

处理标准事件

DHTML 小脚本可以揭示下述标准 DHTML 事件：

Onclick	onkeypress	Onmousemove
Ondbclick	Onkeyup	Onmouseup
Onkeydown	onmousedown	

注意：有关小脚本事件、属性和方法的内容，参见在线文档中“Visual InterDev Reference”。

在小脚本中出现的事件不能自动发送给宿主应用。要俘获宿主应用中的标准事件，你必须在两个地方编写处理程序：一个地方是在该小脚本中，用于俘获和转发事件；另一个地方是在宿主应用中，用于截获事件。

把事件从小脚本传递给宿主应用

1.把事件处理程序脚本连接到你希望传递的事件上。

2.在事件处理程序脚本中调用 bubbleEvent 方法，把事件转发给宿主应用。

注意:把事件传递给容器对象以前，你可以检查该小脚本冻结的特性，保证容器对象已经准备好处理事件。

如果小脚本没包含对应于一个特定事件的事件处理程序，那个事件便不传递给宿主应用。同样，如果小脚本包括对应于特定事件的处理程序，但是不调用 bubbleEvent 方法，宿主应用就见不到这个事件。

注意:小脚本容器对象揭示所有设计期间的标准事件，即使这个小脚本不包含把标准事件传递给小脚本容器的脚本也是如此。例如，在 Visual Basic 中，对应于小脚本容器的代码窗口列出所有标准事件，即使在特定小脚本中不是所有事件都可用的。

下边的小脚本说明如何把文本框中的 onkeyup 事件传递给宿主应用。

```
<INPUT TYPE=text ONKEYUP="passKeyUp()" NAME="t1" VALUE="">
<SCRIPT LANGUAGE="JavaScript">
function passKeyUp() {
    // script statements here if required
     window.external.bubbleEvent();
    // further script statements here if required
}
</SCRIPT>
```

注意：当标准事件出现的时候，宿主应用便从整体角度看待相应于小脚本容器对象所触发的事件。例如小脚本可以包含几个按钮。如果用户敲击其中一个按钮（并且如果这个小脚本正在转发事件），宿主应用仅仅接收一个总敲击事件。

要准确指出是小脚本中的哪个控件触发了这个事件，你可以在小脚本中创建一个客户事件，其中包含有关事件的辅助信息。有关细节，参见下边的“创建并处理客户事件”。

关于标准事件的其它信息，如鼠标器光标的位置或事件被触发时键的状态，在脚本容器对象事件特性中都是可用的。例如，下边的 Visual Basic 子程序示出如何截获小脚本的 onkeypress 事件，如何在小脚本文本框中键入字符的键代码：

```
Sub ScriptContainer1_onkeyup()
   MsgBox "The character typed was " & ScriptContainer1.event.keyCode
   MsgBox "The shift state was " & ScriptContainer1.event.shiftKey
End Sub
```

在 Internet Explorer 中，下边脚本完成同样的事情：

```
<SCRIPT LANGUAGE=JavaScript FOR=document EVENT=onkeyup>
   alert("Key code = " + window.event.keyCode)
   alert("Shift status  = " + window.event.shiftKey)
</SCRIPT>
```

创建并处理客户事件

你可以揭示 DHTML 中或自动小脚本中的客户事件。客户事件允许你：

- 发送有关标准事件的更多细节。例如在小脚本中多个按钮中某个按钮被敲击而出现的事件。
- 把小脚本中非标准改变的有关信息通知给宿主应用。例如特性值改变时出现的信息。
- 把有关 DHTML 事件有关的信息通知给宿主应用。这样的事件不在由 bubbleEvent 方法处理的标准事件之列。

对于标准事件，你必须把小脚本中的事件和宿主应用截获的事件发送出去。

发送小脚本中的客户事件

- 调用小脚本的 raiseEvent 方法。

注意:在把事件传递给容器(container)对象之前,你可以检查小脚本冻结的特性,要保证这个容器对象已经准备好建立事件。

例如,下边脚本示出当重置小脚本的 backgroundColor 属性时,如何发送称作 oncolorchange 的客户事件。这第二个参数是引用 window.event 对象的对象,它允许容器处理程序获取对象激发事件的有关信息。

```
<SCRIPT LANGUAGE="JavaScript">
function public_put_backgroundColor(value)
{
   window.document.bgColor = value;
   window.external.raiseEvent("event_onbgcolorchange",window.event);
}
</SCRIPT>
```

处理宿主应用中的客户事件

- 为 onscriptletevent 事件创建事件处理程序。

下边是 Visual Basic 中的一个例子,说明如何确定是什么控件触发事件:

```
Sub ScriptletContainer1_onscriptletevent( ByVal txtTitle As String,
    ByVal eventData As Variant)
    objName = eventData.srcElement.id
    MsgBox "The event " & txtTitle & " occurred in " & objName
End Sub
```

如果你的宿主应用是 Internet Explorer，就使用如下边给出的脚本截获小脚本事件：

```
<SCRIPT LANGUAGE="JavaScript"
        FOR="s1"
        EVENT="onscriptletevent (event, obj)">
    msg = "Event fired in the scriptlet was " + event
    msg = msg + "\nand the ID of the object was " + obj.srcElement.id
    alert(msg);
</SCRIPT>
```

你可以使用 onscriptletevent 事件中的 Select Case 结构，依据不同事件产生不同动作。

27.5 把 DHTML 小脚本加到你的应用上

在你创建一个 DHTML 小脚本之后，你可以在应用中使用小脚本。你必须首先创建一个小脚本容器对象。这个对象为小脚本创建一个窗口，并提供一种方式，让宿主应用规定小脚本在哪里显示，使用什么尺寸等。

把 DHTML 小脚本加到宿主应用上

1.在你的应用中创建一个小脚本容器对象，并设置它的 Name 属性。

2.把小脚本容器对象的 url（统一资源地址）属性设置成你希望使用的小脚本的 URL。

重要事项:如果你把小脚本加到一个 Web 页面上，不要把 url 属性设置成当前页面的 URL。这样做会产生循环调用页面，也会引起浏览器停止工作。

把 DHTML 小脚本加到工具框中

- 在 Project Explorer 中，右键单击小脚本的 .htm 文件，然后选择 Mark As Scriptlet。

  包含指向小脚本指针的<OBJECT>标记被加到 Toolbox 的 Scriptlet 选项卡上（如果这是 Toolbor 的第一个小脚本，就创建 Scriptlet 选项卡）。然后你可以把小脚本从 Toolbox 拖到其它页面上，并自动创建实现小脚本所需要的<OBJECT>选项卡。

注意:当你把一个小脚本加到 Toolbox 的时候，它就包含这个小脚本的绝对 URL。当把小脚本拖到你的页面上之后，你可能需要在 Properties 窗

口中或在 Source 视图中修改<OBJECT>标记的 URL 属性，从而产生相关的链接。

另外，你可创建一个<OBJECT>标记，以便引用该小脚本。

在<OBJECT>标记中访问 DHTML 小脚本

- 依照下列语法创建一个<OBJECT>标记，替换小脚本的 URL 和名字 *url/scriptletName*:

```
<OBJECT ID="MyScriptlet" TYPE="text/x-scriptlet" WIDTH=300 HEIGHT=200>
  <PARAM NAME="url" VALUE="url/scriptletName">
</OBJECT>
```

用小脚本容器对象进行工作

在你的应用中，DHTML 小脚本出现在小脚本容器对象中。这个对象为你提供到该小脚本的界面。例如，在该小脚本中的一个事件出现的时候（和该小脚本转发事件的时候），你的应用会看见在该容器对象中出现的事件。

你可以通过设置小脚本容器对象属性的办法处理小脚本的外观。在某些情况下，你可以在小脚本的脚本中设置属性以处理其外观。

你还可以从小脚本改变容器对象的尺寸，办法是使用一个脚本设置 DHTML 脚本对象的 pixelHeight 和 pixelWeidth 属性。例如，下例示出在小脚本第一次装载时如何改变小脚本容器的尺寸。

```
<HTML ID="MyPage">
<HEAD>
<SCRIPT LANGUAGE="VBScript">
Sub window_onload()
    MyPage.style.pixelHeight = 300
    MyPage.style.pixelWidth = 400
End Sub
</SCRIPT>
</HEAD>
```

如果你在创建一个控件之后改变 .htm 文件，控件中的显示便不更新，直到下一次页面读出为止。这种情况在应用运行的时候出现，或者在你再一次改变控件的 url 属性时出现。

注意：在启动小脚本之后，在 Internet Explorer 中用于刷新页面的 F5 键在小脚本容器对象中不再起作用。

在创建小脚本实例之后，你就可以为它编写脚本，并作为任何其它控件。你用其属性和方法进行工作的对象是小脚本容器对象。你所使用的特性和方法是由在这个容器 url 属性中标识的小脚本定义的。如果你在可以显示对象属性和方法中的环境中工作，例如 Visual Basic，你就看不见这些属性，因为这些属性不在这样的开发环境中揭示出来。

例如，下边的代码在 Visual Basic 窗体中设置一个属性，并调用一种方法，这种方法在由 ScriptContainer 控件引用的页面中：

```
Sub cmdColor_Click()
    ScriptContainer1.BackgroundColor="red"
    ScriptContainer1.UpdateText (Text1.Text)
End Sub
```

注意：在 Visual Basic 中，你必须把一个参数传递给一种小脚本方法，即使这种方法不需要参数或可以出现差错也要传递。例如，下边的语句把 0 值的占位符参数传递给一个不要求参数的小脚本方法：

```
ScriptContainer1.ToggleColor (0)
```

在获得一个小脚本的属性或调用它的方法之前，你必须保证这个脚本已经由测试容器对象的 onreadystatechange 事件和 readyState 属性以及小脚本的冻结属性完全装载。