第三部分

Visual C++与Internet

返回总目录

目　　录

第 8 章　设计 Web 页

以任何方式接触 Internet 都需要在一定程度上了解 HTML（超文本标记语言）知识。对于这一点，无论你怎样努力，都不能躲开它。以任何形式正常进入 Internet，都要涉及到访问 Web 页，在你领悟到这一点之前，这种说法看起来好像太笼统。在键入 http:时，就会发现你自己正在进入 Web 页，这种 Web 页通过 HTTP（超文本传输协议）使用了 HTML，而不键入这些字母，就不能进行任何信息交换。

技巧　如果愿意的话，大多数情况下可以避免学习 HTML 标记(tags)，诸如 Visual InterDev 之类的产品使你很容易地创建含有 ActiveX 控件的 Web 页面。当然，这就意味着，为了创建并测试 ActiveX 控件，在购买 Visual C++ 时不得不将另一个附加产品买回来。显然，使用诸如 Visual InterDev 之类的产品会使你受益，而购买并学习另一个产品则对你产生不利影响，根据你实际要创建的 Web 页数量，你需要对此权衡利弊。如果仅仅是偶尔地创建 Web 页，那么，手工写出代码可能仍是上策；　反之，如果大部分时间你都是在创建 Web 页，那么就应该投资购买诸如 Visual InterDev 之类的产品了。Osborne/McGraw-hill 公司将出版我的新著《Visual 工作室：参考手册全集》（Visual Studio: The Complete Reference），从这本书中你可以就怎样使用

Visual C++及诸如 Visual InterDev 之类产品，学到更多的东西，这本书介绍了怎样使用 Visual 工作室包含的各种语言产品在大环境下（特别是在团队环境下工作时）去创建应用程序。

作为对使用 ActiveX 感兴趣的 Visual C++程序员，你在创建 Web 页方面可能会花费大量时间，也可能不在这方面耗费时间，但是，你需要了解它们是如何工作的，以便于你能帮助他人使用你的控件。学习 HTML 起初看起来有点难，有大量的标记（就象程序语句一样）可以用于创建 Web 页。查看这些标记不成问题——市场上的任何浏览器都提供了 View/Source 命令，用它们就可以看到 Web 站点的任何 Web 页的原始 HTML。图 8.1 是一个典型例子。

注释 和本书中其它图示一样，对浏览器的配置不一样，可能看到的显示画面外观会有所变化。另外，Microsoft 和 Netscape 经常会改变它们各自的 Web 站点和浏览器的画面外观。最后，增加插件也会稍稍改变浏览器的外观。请记住，把本章中所有的图示及说明看作是你将看到的画面的一个基础，你的实际的屏幕显示可能会与本书所说明的有所不同。

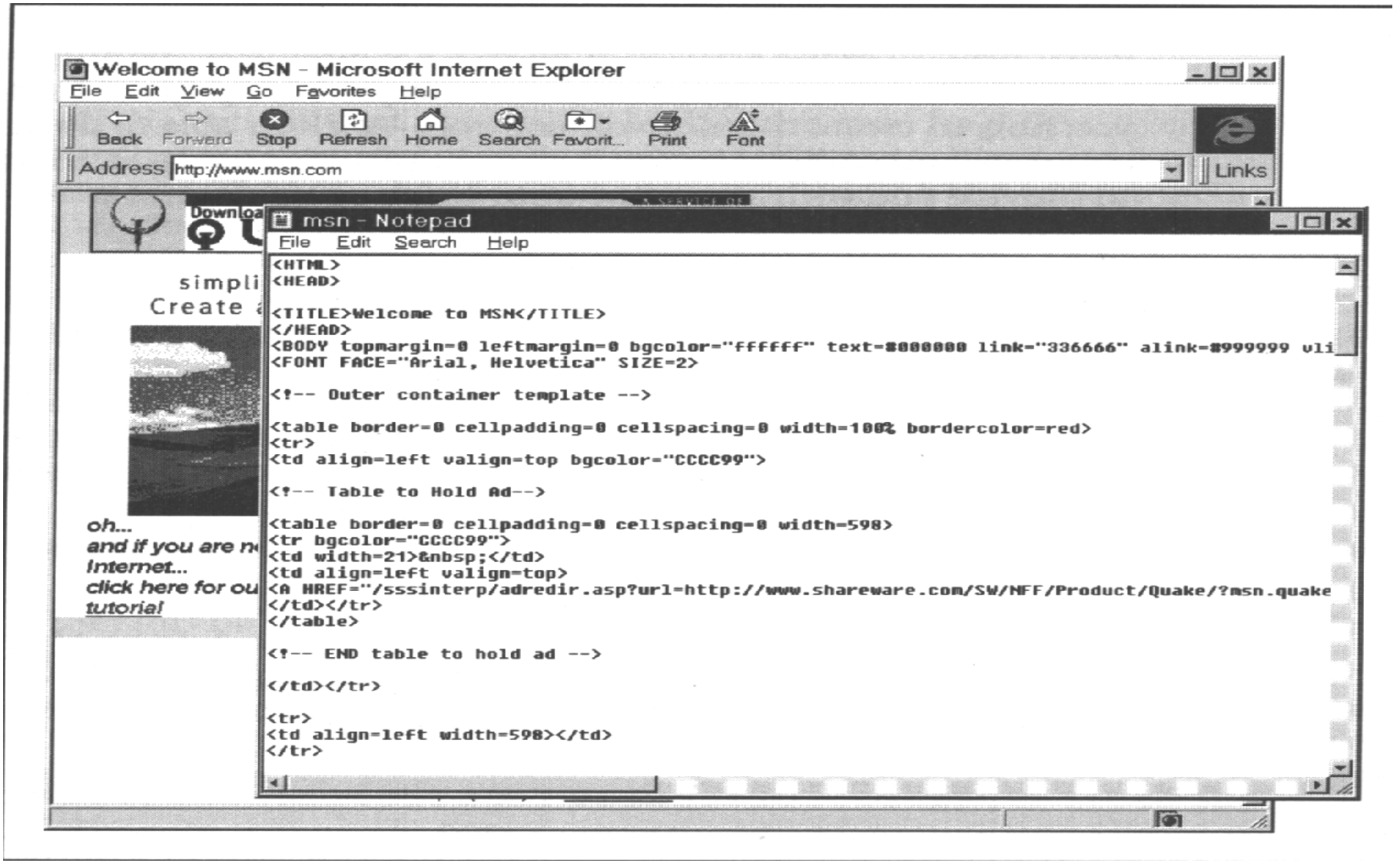


图 8.1　你能够查看大多数 Web 页面的源代码——当你想学点新东西时这是个方便的特色

至少有一种简单的途径开始学习过程，这正是我们这儿将要讨论的问题。仅仅使用 12 种 HTML 标记就可以构成一个 Web 站点，以后再用另外的标记对它们进行修饰。这就是本章第一节的内容。我们考察这 12 种标记，在任何 Internet 站点——甚至是提供最小级别功能的站点——都可看到它们。

技巧 如果要设计大量的 Web 页，那么，能利用 View/Source 命令而获益。先简单地找到一个含有你想使用的那种特色的 Web 站点，然后查看其基础源代码，从而知道别的程序员是如何完成这一任务的。在其它许多编程情况中，代码或被隐蔽，或被加密，或用其它不可访问形式。而 HTML 却保留了自己的类似英语的语法结构，即便是在一个 Internet 站点使用时也是如此。

一旦记住了这 12 种标记，我们就看一下需要使 ActiveX 工作的另外一个标记。我们将把你对一个 ActiveX 控件知道的每一样东西组合起来，产生一个非常简单的 Web 页（第 11 章将说明怎样创建这个 ActiveX 控件），然后，使用这个 Web 页就能干点事了，甚至可以把它作为以后创建新的 Web 页的模板。我的想法就是提供 HTML 的基本知识，而不是使你成为一名 HTML 编程权威者。

注释 本章不打算把需要知道的有关 HTML 的每一个细节都作介绍。有些人抱怨，就这个题目即便写一大本书也不足以完成这项工作。我们想做的是在一个非常基本的级别上——大多数同类书籍中漏掉的级别——考察一下 HTML，看完本章，就有了足够多的信息，使得学习其它 HTML 标记更加容易。

ASCII 文本文件能包含足够多的使 ActiveX 工作的信息，这看起来有些奇怪。学习了 ActiveX-专用标记之后，我们将研究一下浏览器是如何解释这一标记并使 ActiveX 工作的了。一旦知晓了 Microsoft 在 Internet Explorer3.x 和 4.x 中包含的小秘密后，就会发现完成上述任务并不是一件难事。

Microsoft 还引入了称之为 COM+（组件对象模型 Plus）的新技术。Microsoft 正是最终利用这一技术在 Internet 上扩大编程的。它不是 ActiveX 的替代物，ActiveX 仍保持作为创建组件的选择技术，但 COM+将最终会在 Internet 上取代 ActiveX。本章后面的一节将提供这一激动人心的新技术的概况，从而认识到在将来它会怎样对你有所帮助。

注 COM+最终会在 Internet 上取代 ActiveX，而 ActiveX 将仍保留作为 Microsoft 的应用程序组件技术。

掌握了基本的 HTML 标记和 ActiveX 标记之后，我们就对简单的 HTML 页进行两个容易完成的增强，这两个增强，一个是声音，一个是动画 GIF，它们摇旗呐喊，避免了枯燥。重要的一点是为你的 Web 站点增加情趣时，别把它搞得太大，以至于不能在合适的时间内下载。

本章的最后一节，讨论 ActiveX SDK 提供的一个实用程序：DIANTZ.EXE。

它允许压缩一个或多个 ActiveX 控件，减少下载时间。Microsoft 正是使用这个实用程序产生了用于发布 Windows95 的 CAB 文件，可以发现，DIANTZ 实用程序的功能远远不止是用于 ActiveX 开发。

8.1 HTML 概要

关于 HTML，即使写上一大厚本书，也不足以涵盖所有标记的全部变化。让我们首先讨论一下“标记”这个术语。每一个 HTML 语句由一个或多个标记组成，一个标记就象一个程序设计语句。有的标记用于标明 HTML 文档各个章节的开始与结束，而其它的标记则用于标明进行其它处理。由于公司更多的涉足 Internet，标记的复杂性也在变化，我们将讨论落入本章范围的两种变化，但是，由于篇幅所限，我们将不讨论其他的变化（本章将指出在什么地方可以找到一些另外的信息）。

注 标记类似于微编程语句，允许格式化 HTML 文档或者处理信息。

最基本的 HTML 文档大概包含 12 种不同的标记，其中一些是强制性使用的，另一些仅用于提供指定类型的功能。我们仅讨论标记的最简单形式——但能够对大多数标记添加限定词以影响它们工作的方式，例如：可以添加一个字体描述，或者让标记对其所在页的文本进行居中处理。表 8.1 是将要讨论的 12 种基本标记的概要。

注 据经验估算，包括图形在内，Web 页要保持小于 60KB，否则，下载会占用太多的时间。

表 8.1　12 种基本 HTML 标注概要

标记	使用方式	说明
<HTML>　和 </HTML>	强制使用	每个 HTML 文档都以<HTML>开始，以</HTML>结束，这样浏览器就可以知道从什么地方开始和结束读取
<HEAD>　和 </HEAD>	正常使用	当查看 HTML 文档时，你将看到一个头部和一个正文，头部通常标识 Web 站点并定义页面设置
<BODY>　和 </BODY>	正常使用	将把 Web 页内容的主要部分放在这两个标记之间，正文部分通常包含着访问站点的用户要查找的信息
<Hx>和</Hx>	选择使用	书籍和其它文本形式，通常把它们所包含的信息，用标题分成章节，本标记允许对 Web 站点添加标题
<P>	选择使用	HTML 总是假定：你键入的所有文本，不论它们是否在一行，都是同一段的一部分。使用本标记增加一个回车符和两个换行符（标识段结束）
 	选择使用	常常想要在不增加新行的情况下回到页的左端，本标记可以完成这一工作，它添加一个回车符和一个换行符

续表

<HR>	选择使用	水平标尺标记允许一行可以跨页，它还在段落之间增加空白
<A HREF="Destination Document">和</A>	选择使用	与其它文档的超链接是 Web 页最普通的组成部分
<A NAME= "Anchor Name">和</A>	选择使用	锚地标记允许从页的一点移动到另一点。超级链接使用<A HREF= "Current Document #Anchor Name">来访问这一标记
<EM>和</EM>	选择使用	不同的浏览器对强调标注反应不同，大多数浏览器都是用斜体显示文本
<STRONG>和</STRONG>	选择使用	和强调标注一样，本标记由浏览器决定如何对<STRONG>标记作出反应。大多数浏览器看到这一标记时用黑体显示文本
<PRE>和</PRE>	选择使用	经常不想让浏览器对文本重新格式化，预格式化标记指示浏览器保持文本格式不变

现在已经对这些标记的作用有了基本了解，让我们再对它们中的几个进行详细的讨论。下面的章节讨论一些标记，它们会使你在多少有些一致的基础上进行使用，并且在大多数情况下使用时要做一些工作。不必担心在讨论中不能完全掌握它们，在“创建简单的 HTML 文档”那一节中会详细地讨论实际使用过程。了解标记的工作方式，有助于你弄明白它们与浏览器是如何相互作用的。

理解<HEAD>标记

头部让浏览器在开始显示时知道你对该页做些什么，但它本身并不提供任何形式的实际内容。例如，请看图 8.2 中 http://www.microsoft.com 的源代码。

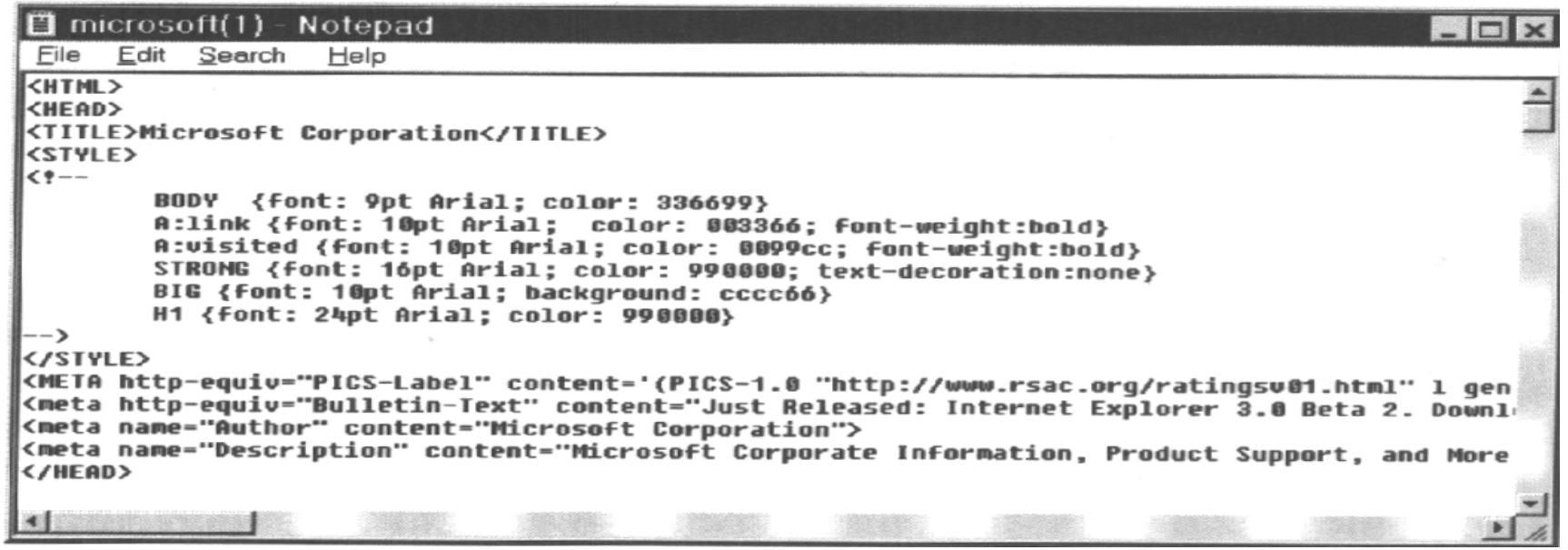


图 8.2 <HEAD>标记不影响页的内容，但确实定义了页的外观

注意到头部会有一个标题，一些体例命令和几个元命令。事实上，这三个元素代表了通常可以在 HTML 页头部中找到的仅有的三种标记。

不要以为<META>标记是在头部部分找到的脚本的唯一形式。例如，如果访问 http://www.netscape.com，就可以在其头部找到 JavaScript 命令。不用在乎页设计者使用哪种脚本语言，关键是脚本的作用是为页的某些元素添加指令。

一种方法是把头部当作页的其余部分的定义区。它影响的是页的整体外观而不提供任何形式的内容。事实上，通常总是想对访问站点的访问者隐藏掉这一部分。

使用<Hx>标记添加标题

与一本书中的标题把文本分隔成小节一样，HTML 文档中的标题标记（<Hx>）把内容部分分隔成不同部分。HTML 规范提供了 6 个级别的标题，通常，这 6 个级别间的唯一差别是显示标题时使用的字体大小。标题的审美需求是人们所要求的，使用标题并无什么神秘可言，它们仅在文本的小节之间创建物理分隔，这有助于访问你的站点的人理解你所提供的材料。

技巧 数字小的标题级别使用的字体，比数字大的标题级别的使用的字体大。例如，<H1>比<H2>使用户可以看到更大的字体。

怎样创建一个小节标题呢？ 在<Hx>标记中，简单地用 1 到 6 之中的一个数字代替 x，然后键入标题文本，之后在行尾加上</Hx>。例：

```
<H1>This is b heading </H1>
```

要记住的另一点是，不必在标题尾部为标题添加诸如<P>，
或<HR> 之类的行结束标记，这是因为结束标记</Hx>自动地添加了所需数量的空白，当然，这并不阻止你加入所需的其它额外空白。

链接与锚地

链接与锚地提供了访问一个Web站点的特殊手段。每次访问一个Web站点时，总会看到一些文本加了下划线，这时大概就是看到了一个链接。一个链接把当前位置与某个另外的位置相连接。事实上，大多数Web页使用最常见的链接形式。第一种是创建到另一页的链接（用Internet术语来说是一个URL），这种形式更常见；另一种是创建到当前页的另一小节或到另一页的某一指定部分的链接，使用它可以找到一个锚地（一会儿会讨论它）。下面是两种常见的链接形式的例子，注意到它们都在<A>标记中使用了HREF关键字。

```
This is a <A HREF="http://www.microsoft.com">Document</A>Link
This is a <A HREF="http://www.microsoft.com#MyAnchor">Anchor</A>Link
```

注 链接将一页与另一页相连接；锚地创建到一页中指定位置的一种连接。

在这两种情形中，标记<A>与标记</A>间出现的文本都是Web站点上加了下划线的句子的一部分。在<A>与</A>之间应该总是要放上某些文本，否则你的链接对于用户是不可见的！注意到这两种链接看起来几乎完全一样，但有一个差别，那就是锚地链接使用#号将URL与锚地名分割开，上例中，使用MyAnchor作为锚地名，第二种链接首先处理URL，然后在向用户显示内容之前，先处理页上指定的锚地位置。

技巧 若创建到当前页锚地的一个锚地链接，则不必提供<A>标记中的URL部分，可缩写为<A HREF="#*Anchor Name*">。

锚地链接要求在<A>标记中指定页的某个地方创建锚地（除非是在当前页

找一个指定位置，这时不必提供 URL）。例如，我们看到了一个到 http: //www.microsoft.com 上的 MyAnchor 的锚地链接，但是这一位置的锚地看起来是什么样的？

```
This is the <A NAME="MyAnchor">Anchor<A> for this page.
```

如你所见，锚地和链接之间的差别是 NAME 关键字，锚地总是用 NAME 代替 HREF。另外，大部分浏览器都不对<A>与</A>之间的文本作高亮或加下划线处理，这是因为确实不用它们做任何事。

8.2 创建一个简单的 HTML 文档

在前面的小节中，我们讨论了创建 HTML 页所用的 12 种基本标记。每个人都可以按这种方式使用这些标记。现在，该讨论一下怎样在 Web 站点上使用这些标记创建一个页了。清单 8-1 列出了一个简单 Web 页的源代码。显然，决不会在实际应用程序中创建这样（简单的）一个页，这一页的全部目的在于说明 12 种标记是如何工作的。

程序列表 8.1

```
<HTML>
<HEAD>
<TITLE> Sample HTML Page </TITLE>
</HEAD>
```

<BODY>
<H1>This is a <EM> Level 1 </EM> heading。 </H1>
Notice that the "Level 1"portion of the heading is emphasized using the
EM tag.<P>
We ended the previous paragraph with a < STRONG> P< /STRONG> tag.
This paragraph will end with a <STRONG>BR</STRONG>tag.
Notice how we used the <STRONG>STRONG</STRONG> tag to add bold text to
the previous paragraph.<HR>
Horizontal rules also have a place on Web sites. You'll use them most
often to provide separations between major areas of text . Of course,
most Web sites are starting to use frames because they're more flexible .<HR>
Here are the other five levels of headings.<P>
<H2>Level 2</H2>
<H3>Level 3</H3>
<H4>Level 4</H4>
<H5>Level 5</H5>
<H6>Level 6</H6><HR>
Let's look at some other HTML tags.<P>
Click <A HREF="LISTS.HTM">Here</A> to display the list page.
Click <A HREF="GRAPH.HTM">Here</A> to display the graphics page.
Click <A HREF="TABLE.HTM">Here</A> to display a page with tables.

```
Click <A HREF="FORM.HTM">Here</A> to display a page with forms.<BR>
Click <A HREF="#MyAnchor">Here</A> to see another area of this page.<P><P><P><P><P>
This is an <A NAME="MyAnchor">Anchor</A> link area.
</BODY>
</HTML>
```

这些源码初看起来有些唬人，但这里没有任何前面小节中没有讲到的东西。首先，第一个标记<HTML>定义了页的开始，在源代码程序列表的末尾可以看到它的配对符号</HTML>。其次，我们把文档用<HEAD>及<BODY>（及其配对符号）将文本分成了两部分。唯一没见过的标记是<TITLE>，它定义了浏览器标题栏中的标题，这一标记并不是绝对必要的，但给Web页一个标题很好，使其它人知道它们在看什么。另外，标题还可以帮助在自己的页间航行以找到出问题的部分。

注 <TITLE>标记在浏览器标题栏中显示你的Web页的名称。

页的正文部分以一个1级标题开始，接着是加了特殊属性的一些文本，其后是一条水平标尺标记，在这里，还可以看到能表明<P>与
标记有什么差别的一些文字，过一会儿就会看到它们是怎么一回事。

程序列表的下一节说明了标题的其它五个级别的差别，大多数浏览器只是简单地使用不同的字号来改变标题的级别。不幸地是，大多数浏览器没有足够多的可读的字体大小来对应全部标题级别。可能需要避免使用第6级标题，因为它们太小，以至于某些人看不清。第5级标题已经是相当小了。

在这个示例的最后一小节含有一些链接，现在还不能看到链接上去的页，这仅是后面章节的一个练习。但是，请看第五个链接，可以注意到它是当前页的一个锚地。正文部分的最后一个工作行包含着这个锚地。一定要花些时间在你的浏览器上看一看它们是如何工作的。

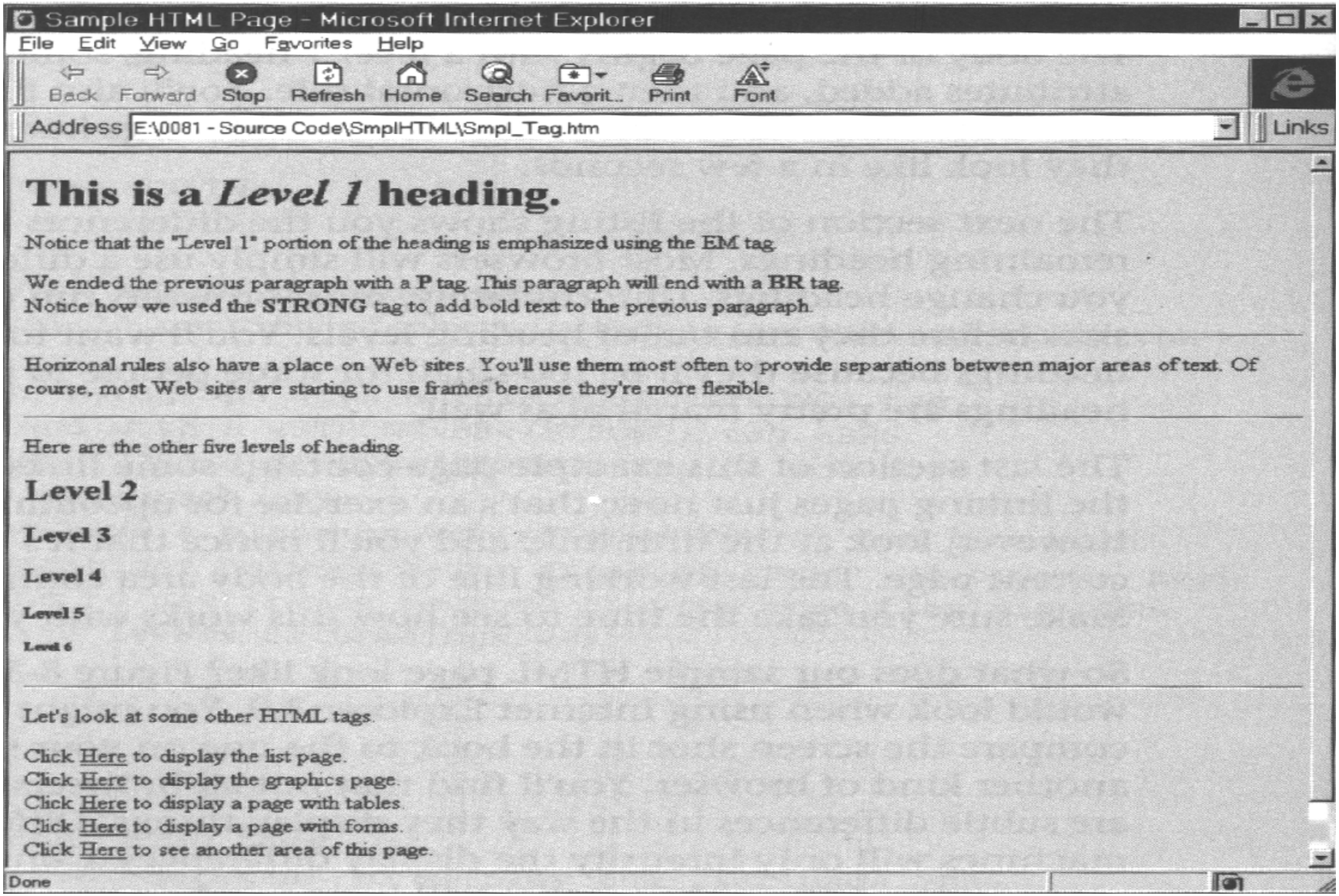


图 8.3　HTML 页的样本，用以说明在 Internet Explorer 3.0 中 12 个基本标记的作用

那么，我们的 HTML 样页看起来是什么样子呢？图 8.3 是使用 Internet Explorer 3.0 时可以看到的样子。如果你使用其它浏览器，就要用些时间对书中列出的屏幕显示与你自己的屏幕进行比较了。可以发现，并不是所有的浏览器都是一样的，在显示东西时，它们会有一些细小的差别。机器的差别只能加剧显示的差别。考虑到这些都是常见标记，想象一下，如果使用了某个浏览器支持而其它浏览器不支持的一些更奇异的标记时会发生什么呢?!

现在到了扩充我们的 HTML 知识的时候了。下面几节将要说明一些其他标记，尽管在这些章节不准备讨论全部细节，但它们提供的足够信息，已经可以使你在访问其它 Web 站点并查看源码时能看懂它们。

使用列表

你看到过不包括任何形式的公告式列表或过程程序的技术书籍吗？把复杂的思想用容易理解的语句片段表达出来，恐怕是作者们能够进行的最基本的任务之一。列表——是枚举想法的小段文本——是大多数情况下解决问题的答案。考虑到大多数 Web 页的大小限制，列表就变得更加重要。HTML 支持三种不同形式的列表：

- 无序列表　这种列表使用公告牌形式，以<UL>标记开始，并以</UL> 标记结束，在列表中的项以<LI>标记开头。
- 有序列表　过程或其它具有编号的文本形式考虑使用有序列表，这种列表格式使用<OL>和</OL>标记，和无序列表一样，每一项以<LI>标记开头。
- 术语汇编　这种特殊的列表使用由<DL>和</DL> 标记括起来的两类项元

素。被定义的项前辍以<DT>标记，而项的定义则前辍以<DD>标记。

尽管只有三种列表形式，但就如同其它程序设计方案中一样，可以用嵌套方法实现多种功效。讨论页的样本的代码时，就会看到列表 8.2 中是如何实现这一点的了。

注 总要对 HTML 页加入大量的注释，因为可能有不是程序员的人们来不时地编辑它们。

在我们讨论的三种形式的列表中，术语汇编是最灵活的。它使用的两种类型的列表元素使显示的文本看起来更美观。<DT>（项）标记和<DD>标记（定义）间的差别在于其中之一是被缩进的。可以用这一特色，作出与术语汇编类型元素显示无关的特殊功效。和本章中讨论的一些其它特殊文本标记一样，列表标记（<LI>，<DT>和<DD>）在行尾自动插入一个
标记。使用
标记使得文本变成下一行，但保持列表仍在一起。

现在又该看一下列表代码的例子了。程序列表 8.2 显示的是程序列表 8.1 中看到的第一页的第一个链接页。它包含着三种基本列表形式，以及一个说明怎样对它们进行嵌套的例子。请确保将产生的文件命名为 LISTS.HTM，这样才能使链接按所预期地那样工作。特别要注意这是第一个使用注释（<!-文本>标记）的页面。

程序列表 8.2

```
<!-本页目的是说明如何使用列表标注。->
<HTML>
<HEAD>
```

```
<TITLE>Creating Lists</TITLE>
</HEAD>
<BODY>
<H2>There are three different kinds of Lists:</H2>

<!-显 示 无 序 列 表 ->
<H3>Unordered</H3>
<UL>
<LI>Item 1
<LI>Item 2
</UL><HR>

<!-显 示 有 序 列 表 ->
<H3>Ordered</H3>
<OL>
<LI>Item 1
<LI>Item 2
</OL> <HR>

<!-显 示 说 明 两 种 列 表 元 素 类 型 的 术 语 汇 编 列 表 ->
<H3>Glossary</H3>
```

```
<DL>
<DT>A Term
<DD>this is the definition for it.
<DT>Another Term
<DD>Yet another definition.
</DL><HR>

<!-显示的是嵌套列表的一种方法->
<H3>Nested List</H3>
<OL>
<LI>Item 1
     <UL>
     <LI>Subitem 1
     <LI>Subitem 2
     </UL>
<LI>Item 2
</OL>
</BODY>
</HTML>
```

注　尽管有其它方式创建注释，但通常习惯于用<!－文本>标记。

你已很好地了解了怎样对各种列表标记写出代码了，现在就能够看到这一

页的样子了。图 8.4 显示的是怎样使用列表的一个典型例子，请注意，三种列表形式都没有怎么进行修饰，但它们却都正常有效。

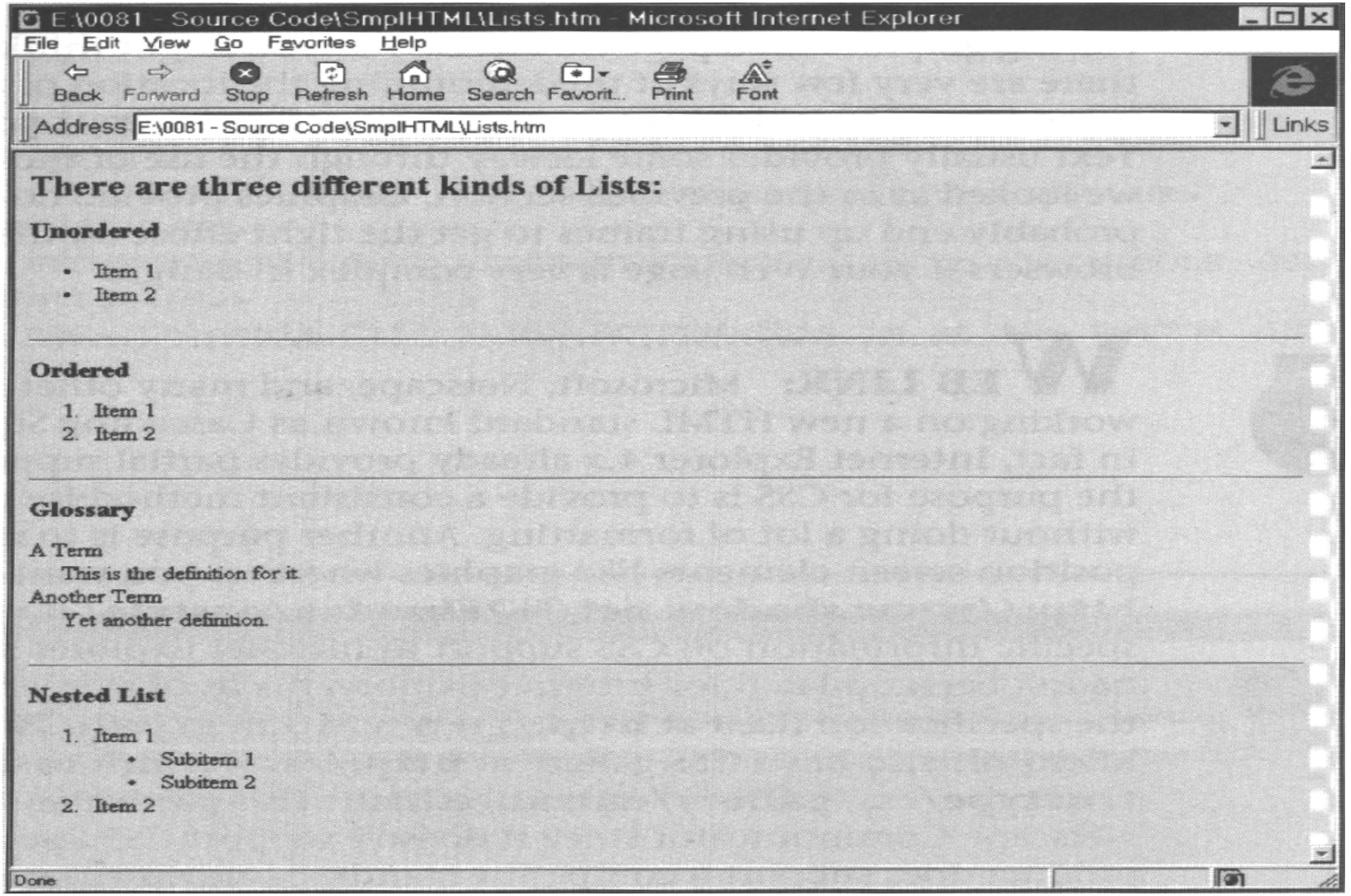


图 8.4 列表标记：容易地在 Web 页中枚举指定要点，或提供术语汇编类型的元素

添加图形

图形确实可以对文本页进行乔装打扮。我们大多数人都会同意这样的说法：如果作者在技术手册中添加了足够多的正确的图形图像，理解起来就会容易得多。也可以向 Web 页中加入图形，然而，添加的方式上却有一些限制。

技巧 一些人过分热衷于向它们的 Web 站点添加图形，导致了大多数人甚至不能看到这些图形的大部分，当用户在下载图形时，如果用了一分钟还不能完成下载，他们就会按下浏览器的 Stop（停止）按钮。事实上，一个良好的估算是使 Web 页的大小（包括图形和 ActiveX 控件）保持在 60KB 以下。

你要面对的最大限制之一是能够显示的图形种类，许多浏览器只处理 GIF（又称为 CompuServe 格式）和 JP（JPG）图像这两种格式——它们是由 HTML 支持的。你的浏览器上显示的其它类型的图像，都是使用了插件的结果。Netscape 和 Internet Explorer 都支持各种插件，例如，更流行的插件之一允许用浏览器观看 AVI（电影）文件。Web 页设计者的问题是不能指望别人都拥有这些插件。于是，如果计划创建一个受欢迎的 Internet 站点，那么大概就要坚持使用 GIF 和 JPEG 图像了。

注 使用 GIF 和 JPG 图像可以获得最大的兼容性，而在知道了用户使用什么浏览器时，使用其它图像类型则可以获得最大的美观效果。

显示图形图像是相当容易的，必须做的全部工作就是使用<IMG>标记，这一标记使用 SRC 属性来接受一个文件名，例如：

```
<IMG SRC="Figure1.Gif">
```

把图形放入一个页面就不像你想像的那么容易了，像文本情况时一样，由于不能定义它们的确切位置，除了使用框架(frame)或表格之外，没有多少办法来预先确定图形的位置。通过使用诸如在前一小节中看到的列表标记技巧，文本通常提供了某些灵活性，而图形却不提供这样的灵活性。如果你设计中的 Web 页非常复杂，那么大概一直要使用框架(frame)来获取大部分浏览器的正确响应了。

Web 链接 Microsoft、Netscape 和其它供货商正在制定称之为层叠式电子表格（CSS）的 HTML 新标准。事实上，Internet Explorer 4.x 已部分地支持 CSS 。CSS 的一个目的是提供前后连贯的方法，从而不用做大量格式化工作就可以显示数据。另一个目的是，使诸如图形这样的屏幕元素的定位更容易。如果想找到 Internet Explorer3.x 支持的 CSS 的具体信息（写本书时，Internet Explorer 4.x 的 Web 站点还没有更新），那么请查看 http://www.shadow.net/%7Ebraden/nostyle/ 。可以在 http://www.w3.org/pub/WWW/Style/CSS/，找到规范本身。Microsoft 在 http://www.microsoft.com/turetype/css/gallery/entrance.htm 也有一个 CSS 展台，这一特定的站点不能用 Netscape Communicator 访问，因为到写本书时它还不支持 CSS。这一产品也支持称之为 Javascript Style Sheets 的竞争标准。

使用<IMG>标记能够定义文本和图形混合显示的方式，所要做的一切工作

就是加入 ALIGN 属性，有三种方式对齐文本：

- ALIGN=TOP
- ALIGN=MIDDLE
- ALIGN=BOTTOM

注 当图形与文本相邻显示时，ALIGN 属性允许你改变文本的显示位置。

让我们看一下向某个 Web 站点加入图形的简单代码。在“使用窗体(FORM)”一节中，我们将更深入地考察图形的定位问题。程序列表 8.3 中的代码显示了图形图像，并说明怎样使用 ALIGN 属性（这一程序列表使用了我作的一个 GIF，你可以用手边的任何 GIF 文件代替它）。这种方法对于显示公司形象标志或在一个页上显示其它简单图形是适用的。图 8.5 则显示这些代码显示出的图像。

程序列表 8.3

```
<HTML>
<HEAD>
<TITLE>Using Graphics</TITLE>
</HEAD>
<BODY>
<H2>You can align text and graphics in one of three ways.</H2>
<H3>Top</H3>
<IMG SRC="ColorBlk.Gif",ALIGN=TOP>This is at the TOP.</HR>
<H3>Middle</H3>
<IMG SRC="ColorBlk.Gif",ALIGN=MIDDLE>This is in the MIDDLE.</HR>
```

```
<H3>Bottom</H3>
<IMG SRC="ColorBlk.Gif",ALIGN=BOTTOM>This is in the BOTTOM.</HR>
</BODY>
</HTML>
```

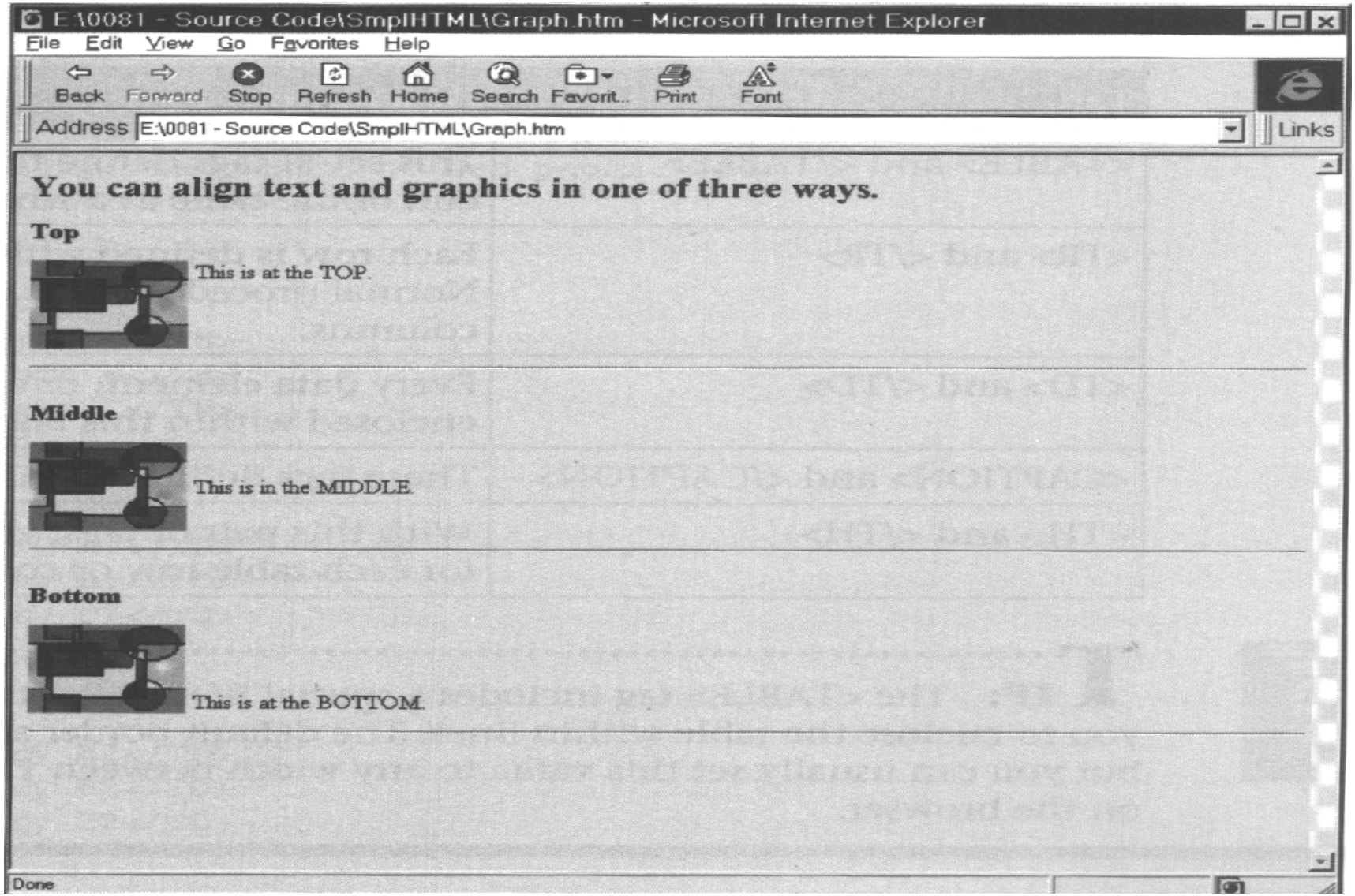


图 8.5　添加简单图形是相当容易的

Web 链接 你可能已经注意到某些人用图形而不是用文本来创建链接（详见本章前面一节“链接及锚地”）。这种技术称之为映射。如果没有实用程序的帮助，在 Web 页中使用映射很快就会糊涂起来。在 http://www,ecaeta.ohio-state.edu/tc/mt/ 上可以找到一个称为 Map This 的非常出色的共享实用程序，可以帮助你创建图形映射。（由于这个站点十分繁忙，也许会碰上麻烦，但坚持试下去，因为这个实用程序非常值得去努力得到它。）

创建表格

表格化数据差不多总是所有商务展示的一部分。电子表格帐页（spreadsheets）仅仅是会计师分类帐帐页在计算机中实现的一个例子。在 HTML 页上创建表格是有意义的，但是需要有一整套标记来完成这项工作。这里列出了所有必须的制表标记。

标记	说明
<TABLE>和</TABLE>	这两个标记定义表格的头和尾
<TR>和</TR>	用这两个标记定义每个行，正规步骤是先定义行，再定义列
<TD>和</TD>	每一个数据元素，或称做列，都用这一对标记括起来
<CAPTION>和</CAPTION>	这些标记定义表格的标题
<TH>和</TH>	使用这一对标记创建表格中每个行或每个列的标题

技巧 <TABLE>标记有一个特殊的 BORDER 属性，允许你用线条把表格围起来，缺省的边界（线条）值是宽度 1，也可以根据浏览器设置为 1 到 6 之间的任一个数字。

我们来看一份简单的代码，它定义了两种不同的表格。程序列表 8.4 说明了怎样使用本节谈到的那些简单标记来实现两种不同的效果。第一个表格包含的是简单文字元素，而第二个则以复杂的方式混合了文字和图形。

技巧 可用<CENTER>和</CENTER>这对标记对表格中的文本进行居中处理。这一对标记对放入文档的任何文本都可以起作用，而且它们对图形也起作用，请参考程序列表 8.4。

程序列表 8.4

```
<HTML>
<HEAD>
<TITLE>Creating Tables</TITLE>
</HEAD>
<BODY>

<!- 创建简单的文本表格 ->
<H3 Tables are an important part of Web pages.</H3>
<CAPTION> A Simple Text Table </CAPTION>
<TABLE BORDER>
      <TR>
```

```
        <TH></TH>
        <TH>Column A</TH>
        <TH>Column B</TH>
        <TH>Column C</TH>
</TR>
<TR>
        <TH>Row 1</TH>
        <TD>Entry A1</TD>
        <TD>Entry B1</TD>
        <TD>Entry C1</TD>
</TR>
<TR>
        <TH>Row 2</TH>
        <TD>Entry A2</TD>
        <TD>Entry B2</TD>
        <TD>Entry C2</TD>
</TR>
<TR>
        <TH>Row 3</TH>
        <TD>Entry A3</TD>
        <TD>Entry B3</TD>
        <TD>Entry C3</TD>
</TR>
```

```
</TABLE><HR>

<!-  创建图文混合的复杂表格  ->
<H3>You aren't Limited to text in a table.</H3>
<TABLE BODER=3>
      <TR>
            <TD><IMG SRC="ColorBlk.Gif"></TD>
            <TD><IMG SRC="ColorBlk.Gif" ALIGN=MIDDLE>Text and Graphics.</TD>
            <TD>Text Alone</TD>
      </TR>
      <TR>
            <TD>Some more text. </TD>
            <TD><CENTER><IMG SRC="ColorBlk.Gif"></CENTER></TD>
            <TD>Some more text. </TD>
      </TR>
      <TR>
            <TD><IMG SRC="ColorBlk.Gif"></TD>
            <TD><CENTER>Some more text .</CENTER></TD>
            <TD><IMG SRC="ColorBlk.Gif" ></TD>
      </TR>
</TABLE>
</BODY>
</HTML>
```

注 用缩进格式写代码可以使表格中的行和列分开，而且容易看清。

这些代码看起来真不少，手工写起来令人发烦（类似于这样的代码从另一角度说明，如果计划写出大量HTML页面代码的话，就应该多用些精力让GUI前端工具来帮你的忙）。图8.6是在浏览器看到的屏幕显示。上述代码还说明了一个要遵守的基本规则：工作时总是先行后列。如果遵守这一简单规则，在制表时就不会遇到什么麻烦。注意到第二个表格是图文混合的，尽管表格看起来有些复杂，但先行后列的同一原则仍然成立。请注意，第一个表格使用的1宽度边界与第二个表使用的3宽度边界有所差别，3宽度边界占用更多的空间，但也更引人注目。

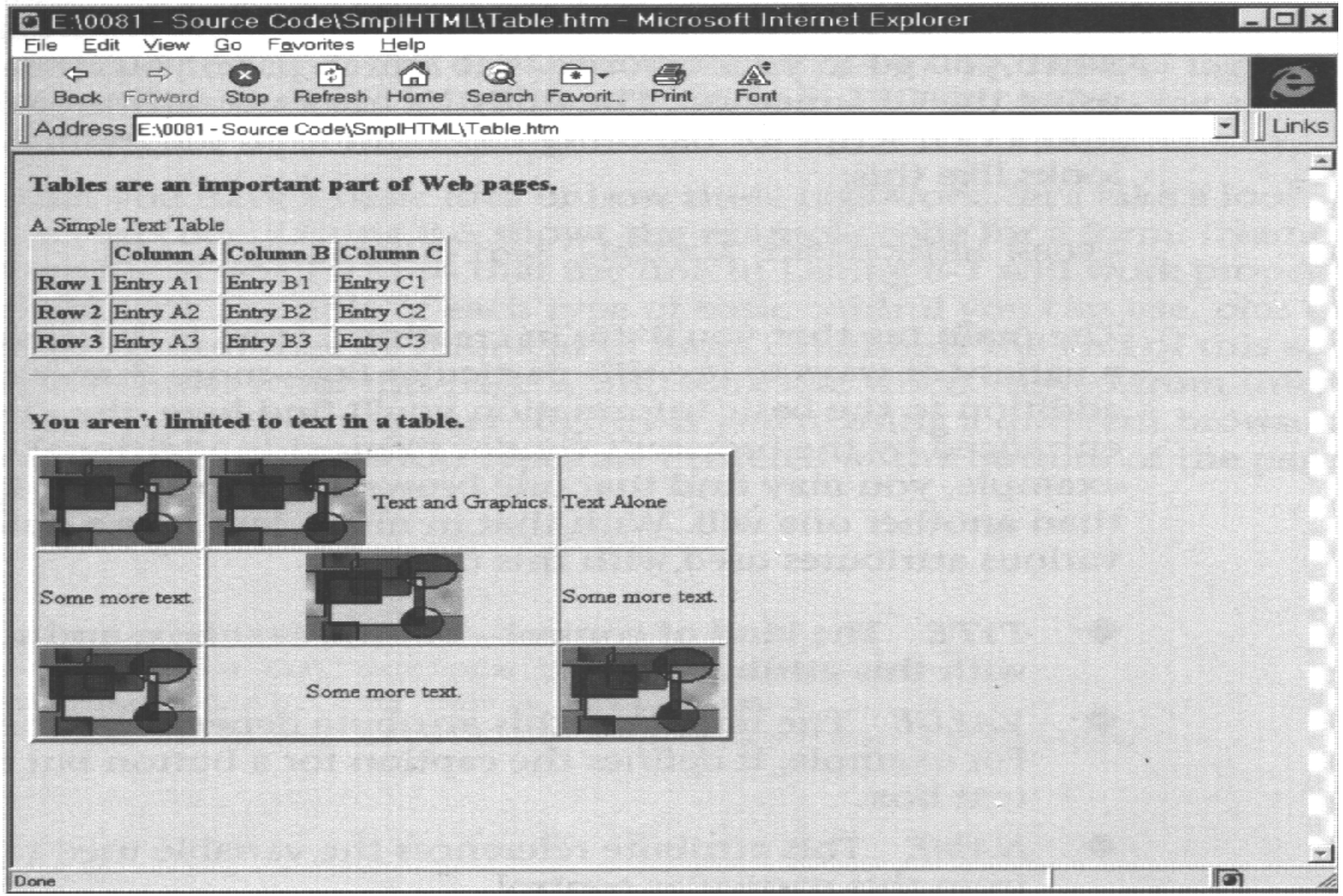


图 8.6　表格容易实现：先指定行，再指定列

第一个表格中行和列中都使用了<TH>。标准文本和表头文本有两个主要差别：第一，标准文本通常是左对齐的，而表头文本则是居中的。可像以程序列表 8.4 中那样，用<CENTER>来改变它们的行为。第二，表头文本使用粗体字。

注：用<CENTER>可以对 Web 页上任意位置的文本（包括表格和窗体中的文本）进行居中处理。

使用窗体

我们要讨论的最后一段基本的 HTML 代码是窗体，窗体有多种用途——大部分都是和数据项（Entry）打交道而不是用于数据传播（dissemination)。到目前为止，我们所看到的全部技术中，在使用标准的 HTML 向 Web 站点添加类似于单选按钮和复选框之类的项时，窗体提供了唯一的方法。因为 ActiveX 将从根本上大量地代替了使用窗体的需求，尽管了解窗体是使用 ActiveX 的另一种选择是重要的，但我们在这里并不打算介绍使用窗体的所有难以预测的变化。

每个窗体都以<FORM>标记开头。除了标记本身，还必须包括进两个属性：METHOD 和 ACTION。METHOD 属性定义了你打算怎样处理窗体收集数据的方式，这有两种标准的方法：POST 和 GET。POST 方法更多地用于数据项（Entry）窗体，它允许你从客户机向主机发送数据。GET 方法用于数据查询窗体。例如，当你要到你喜爱的 Web 搜索页时，实际上你就是要填一份使用 GET 方法的窗体。ACTION 属性则让 Web 服务器知道到哪里去找一个 CGI 脚本，用它处理来自窗体的输入。一个典型的<FORM>标记如下：

```
<FORM METHOD=GET ACTION="/cgi-bin/query">
```

创建一个窗体使用的主要标记是<INPUT>标记。有许多方式——多于这里介绍的——使用这一特定的标记。除了在这里可以找到的基本信息外，浏览器供货商通常对<INPUT>标记进行增强，使之提供附加功能。例如，你或许会发现某个浏览器比另一个浏览器支持更多种类的按钮。了解了这些后，让我们快速地看一下这一标记使用的各种属性吧。

TYPE 这一属性定义控件的类型——按钮、文本框等等。

VALUE 这一属性的效果取决于控件的类型。例如，它定义按钮的标题或文本框的内容。

NAME 这一属性指定引用该控件的变量名称。

SIZE 这一属性定义控件的大小——通常是按字符计算的宽度。

CHECKED 具有选中或未选中两种状态的单选按钮或其它控件使用这个属性。

ROWS 这一属性定义了为一个特定控件分配的行数——通常是控件的高度。

除了<INPUT>标记，还可以使用<SELECT>标记创建弹出式列表框或菜单，它工作的方式与本章前面创建的列表相同。要用</SELECT>标记结束列表框。在<SELECT>和</SELECT>标记之间，使用<OPTION>或<OPTION SELECTED>标记为弹出式列表框创建一系列表项（entries），<OPTION SELECTED>标记指示出当用户最初看到这一页时你想选择哪一个选项。

最后的窗体专用标记允许创建一个文本区。这允许你在一个小区中显示大量文本，或为用户的注释及评论提供一个大显示区。文本区使用<TEXTAREA>

和</TEXTAREA>标记对，也能够为<TEXTAREA>标记添加一个可选的属性NAME，以便于后面恢复其内容。

现在你已经基本了解了这些标记是如何工作了。让我们来看看实际代码。程序列表 8.5 列出了一个窗体的代码示例（将产生的文件命名为 FORM.HTM 以使程序列表 8.1 中的链接正确工作）。它对能够使用的每种类型的基本控件都给出了例子，还有一个在本节末尾要更详细讨论的扩充控件。图 8.7 则显示的是从 Internet Explorer 上看到的这一页的外观，如果使用不同的浏览器，你的视图也许会稍有不同。注意，在页底部的三个按钮，使用了<CENTER>标记进行居中处理。

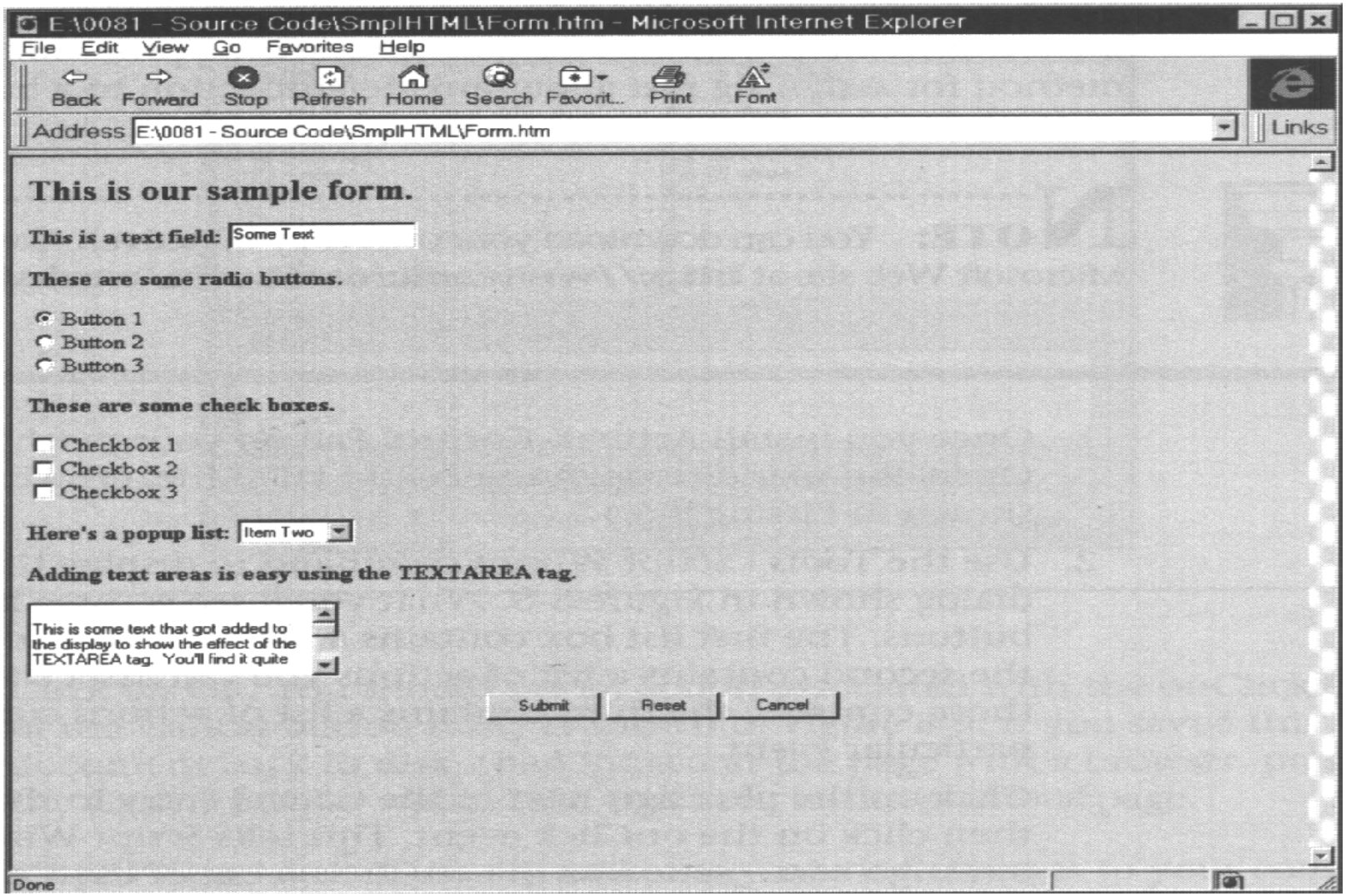


图 8.7　窗体使你从浏览者那里得到反馈

程序列表 8.5

```
<!- 这是一个非常简单的但非功能性窗体，它不是说明怎样写出通常  ->
<!- 需要的 CGI脚本使窗体可以使用，为了看到完整的窗体的实现， ->
<!- 请看类似于 http://www-msn.lycos.com 的 Web 站点->

<HTML>
<HEAD>
<TITLE>New Page<TITLE>
</HEAD>
<BODY>
<H2>This is our sample form.</H2>

<!- 使用标准控件，创建一个简单的窗体 ->
<FORM METHOD=GET ACTION="/cgi-bin/query">

<!- 创建一个文本字段 ->
<STRONG>This is a text field:</STRONG>
<INPUT TYPE="text" VALUE="Some Text" NAME="Text1" SIZE=20><P>

<!- 添加一些单选按钮。->
<STRONG>These are some radio buttons.</STRONG><P>
```

```
<INPUT TYPE="radio" NAME="Radio1" VALUE="Button 1" CHECKED>Button 1<BR>
<INPUT TYPE="radio" NAME="Radio1" VALUE="Button 2">Button 2<BR>
<INPUT TYPE="radio" NAME="Radio1" VALUE="Button 3">Button 3<P>

<!- 添加一些复选框。 ->
<STRONG>These are some check boxes.</STRONG><P>
<INPUT TYPE="checkbox" NAME="Checkbox 1" VALUE="Check 1" >Checkbox 1<BR>
<INPUT TYPE="checkbox" NAME="Checkbox 2" VALUE="Check 2" >Checkbox 2<BR>
<INPUT TYPE="checkbox" NAME="Checkbox 3" VALUE="Check 3" >Checkbox 3<P>

<!- 弹出式列表和菜单需要不同的格式 ->
<STRONG>Here's a popup list:</STRONG>
<SELECT NAME="Popup 1">
      <OPTION>
Item One
      <OPTION SELECTED>
Item Two
      <OPTION>
Item Three
</SELECT>
<P>
```

```
<!- 还能够在屏幕上创建文本区 ->
<STRONG>Adding text areas is easy using the TEXTAREA tag.</STRONG><P>
<TEXTAREA ROWS=3 NAME="Text Area 1">
This is some text that got added to the display to show the effect of
the TEXTAREA tag. You'll find it quite handy as you create Web pages
with a lot of information to convey. It's very important to provide the
users of your site with complete information, and this tag helps you do
it.</TEXTAREA><P><P>

<!- 每个窗体都需要一个提交按钮向服务器发送数据 ->
<CENTER>
<INPUT TYPE=submit VALUE="Submit">

<!- 这一个可选的复位按钮能节省用户的时间 ->
<INPUT TYPE=reset VALUE="Reset">

<!-这是一个标准的按钮类型，我们用它来取消窗体 ->
<INPUT TYPE=button VALUE="Cancel">

</CENTER>
```

</FORM>
</BODY>
</HTML>

如你所见，显示的大多数控件使用了<INPUT>标记。控件类型间的主要差别由 TYPE 属性定义，若想知道它们如何工作，可以选择任一控件。不能正确工作的唯一按钮是提交（Submit）按钮，这是因为我们还没有写出 CGI 脚本来处理窗体的信息。取消按钮还不能用，但过一会儿我们就定义它。

在源代码中还应注意到两个独特的按钮类型：提交（Submit）与复位（Reset）。提交按钮总是执行<FORM>标记中 ACTION 属性指出的动作。若打算用标准方法处理窗体中的信息，就必须提供一个提交按钮。复位按钮总是将窗体中的全部控件恢复成初始状态，这使得用户可以从头开始重新启动窗体，而不必先离开该站点再重新进入该站点。

这个例子中的取消按钮还不能用，可以单击它，但什么也不发生。取消按钮代表了 HTML 文档的标准按钮类型，为使其工作，不得不添加一个调用某个脚本或执行某个其它动作的 ONCLICK 参数。一旦创建了这个按钮本身，ActiveX Control Pad 就提供一种容易的办法让你创建这一脚本。下列程序说明了为一个按钮（或其它控件）指定几乎任何缺省动作的快速方法。

注释　可以从 http://www.microsoft.com/workshop/author/cpad/ 的 Microsoft Web 站点为自己下载一个 ActiveX control Pad。

---

1. 一旦在你的机器上安装了 ActiveX Control Pad，使用 File（文件）|Open （打

开）命令打开 FORM.HTM 文件（该文件代码如程序列表 8.5 所示）。

2. 使用 Tools（工具）|Script Wizard（脚本向导）命令显示脚本向导对话框，如图 8.8 所示。看到的是三个列表框及一些按钮。第一个列表框包含了你定义的控件；第二个则包含了你可以与这些控件的事件相联系的动作；第三个包括的是当前赋给一个特定事件的动作。

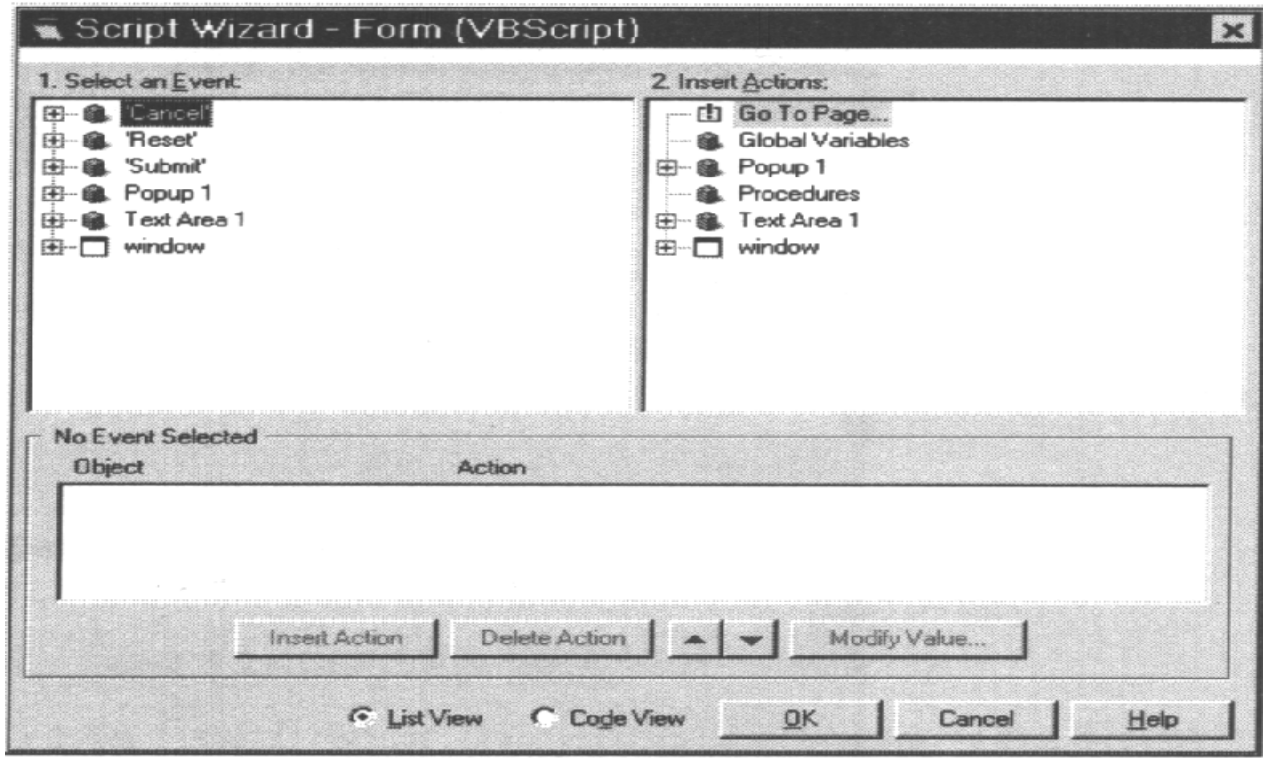


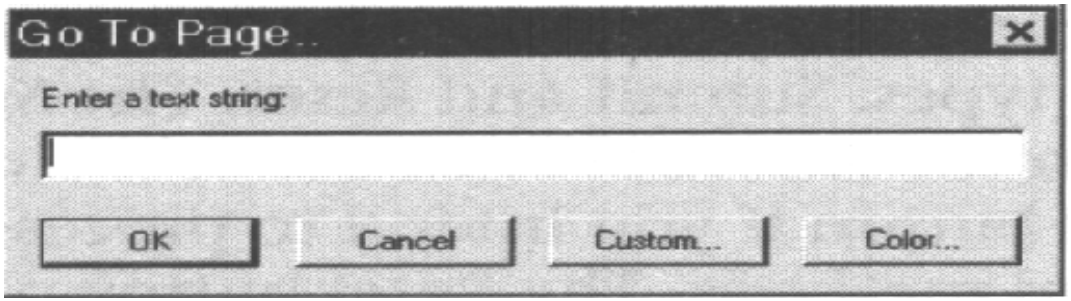
图 8.8　脚本向导显示出你的控件列表和与控件相联系的动作

3. 单击第一个列表框中 Cancel（取消）元素左边的加号，再单击 onClick 事件，

这就是让脚本向导知道你想为 Cancel 按钮的 onClick 事件（单击事件）定义某个动作了。

4. 单击第二个列表框中的 Go To Page（到.....页）项，这表示当你单击窗体中的 Cancel（取消）按钮时，你将到另一页去。这时，你将到我们例中 Web 站点的前一页去。

5. 单击 Insert Action（插入动作）按钮，则看到如下所示的 Go To Page 对话框，在这里可以告诉脚本向导你想到哪一页去。



6. 在 Enter a Text String（键入字符串）域中键入你的 Web 站点某页名称，本例中是 SMPL_TAG.HTM。请注意不必有双引号或单引号——脚本向导自动地处理这些事。

7. 单击 OK，就可以看到 Cancel 按钮的 onClick 事件与一个动作联系起来，如图 8.9 所示。将这一文档存盘，然后用浏览器查看该页，单击 Cancel 按钮，将回到前一页去。

脚本向导自动地对你的代码作了所需的变化，在这时，它向程序列表 8.5 中我们定义的取消按钮添加了一个 ONCLICK 属性，代码如下：

```
<!- 这是标准的按钮类型，我们将用它取消窗体 ->
```

```
<INPUT LANGUAGE="VBScript" TYPE=button VALUE="Cancel"
        ONCLICK="Window.location.href   'SMPL_TAG.HTM'">
```

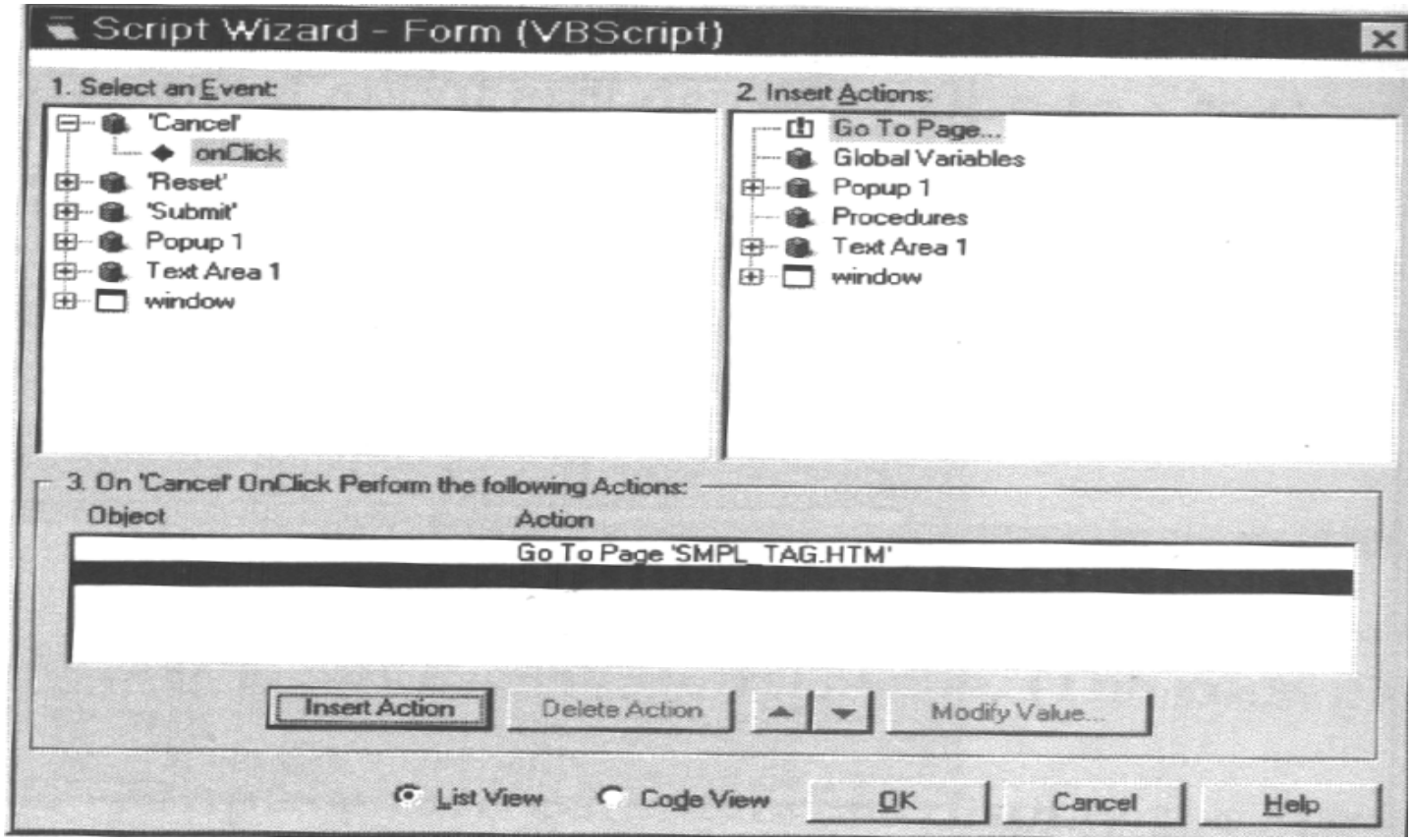


图 8.9　脚本向导通过在第三个列表框中显示事件及其相联系的动作完成你所创建的脚本

8.3 ActiveX 适用的场合

差不多整整一章用于讨论在任何浏览器上都可以使用的编码技术，这些技术对于使用 ActiveX 进行工作的 Visual C++开发人员同样重要，因为想为 Web 站点的每个方面都写出控件是不可能的。即便确实写出了所需的控件，也不可能每个人都乐意等待着将它们全部下载。到目前为止我们讨论的编码技术都是很通用的，不仅它们差不多可以在每个可用的浏览器上工作，而且它们还是与平台无关的。但 ActiveX 却不提供同一级别的支持，暂时它还是非常紧密地绑在 Internet Explorer（或带适当插件的 Netscape Navigator）上。

你可以把现在已学到的知识作为使用 ActiveX 控件进行工作的基础。和前面的几节中添加其它的控件一样，向 HTML 文档添加一个 ActiveX 控件，也需要一个标记，这个标记就是<OBJECT>。让我们来看一个基本的<OBJECT>标记。程序列表 8.6 中含有一个用 ActiveX Control Pad 创建的<OBJECT>标记的典型例子，但你可以同样容易地手工创建它。这个标记设置成使用第 10 章（“以 MFC 为基础的一个基本按钮编程例子”一节）中创建的控件进行工作，而且还需要改变 CODEBASE 特性，使之指向你的 Web 站点。我们将在以后的段落中更多地讨论 ActiveX 特性。

程序列表 8.6

```
<HTML>
<HEAD>
```

```
<TITLE>This Page Contains an ActiveX Control </TITLE>
</HEAD>
<BODY>

<OBJECT ID="OCXEXMPL1" WIDTH=75 HEIGHT=25
CLASSID="CLSID:D8D77E03-712A-11CF-8C70-00006E3127B7"
CODEBASE="http://AUX/CONTROLS/OCXEXMPL.OCX">
        <PARAM NAME ="_Version" VALUE="65536">
        <PARAM NAME ="_ExtentX" VALUE="1976">
        <PARAM NAME ="_ExtentY" VALUE="670">
        <PARAM NAME ="_StockProps" VALUE="70">
        <PARAM NAME ="_Caption" VALUE="On">
        <PARAM NAME ="_OnOff" VALUE="-1">
        <PARAM NAME ="_ModalResult" VALUE="8">
If you see this message without a corresponding pushbutton, then your browser
doesn't support ActiveX conponents. （如果看到这条消息而没有对应的按钮，
那么，你的浏览器不支持 ActiveX 组件。）
</OBJECT>

</BODY>
</HTML>
```

注　和其它的复杂标记一样，<OBJECT>是定义的开始，而</OBJECT>是定义的结束。

先看<OBJECT>标记的第一行。ID 属性指出我们正在处理哪一种控件。我们注意在这里并没写出扩展名，Internet Explorer 缺省扩展名是 OCX。Internet Explorer 目前支持三种扩展名：OCX，CAB 和 INF。我们在本节后边将讨论各种类型的优缺点。还可以指定一个 URL 做为 ID 属性串的一部分。请注意，第一行中还包含着 WIDTH 和 HEIGHT 属性，用它们来定义控件的大小。

注　<OBJECT>标记能识别 OCX、CAB 和 INF 这三种文件扩展名。

和本地机器中每个 ActiveX 控件一样，从 Internet 下载的 ActiveX 控件也具备 CLASSID 属性，这是 Windows 识别一个特定控件的唯一编号，该编号存贮在 Windows 注册表中。在后面将会看到，如果你使用的控件在用户的机器中已经安装，这时，这一特色就能帮助减少控件的下载时间。

添加 CODEBASE 属性的作用是，如果主机上还没安装该控件的话，就指示到你的因特网站点的某处去找到这个控件。如同已在 AUTOEXEC.BAT 已放入的远程 PATH 语句那样，把它也想像成一个远程 PATH。如果不向<OBJECT>标记添加 CODEBASE 属性，那么，就不能有效地使别人从你的 Internet 站点下载该控件。下一节我们会看到这个特色是如何工作的。

现在我们已定义一个对象，一大串<PARAM>标记紧随其后。这些标记定义了在向用户显示该控件时如何配置它。<PARAM>标记总是包括两个属性：NAME 和 VALUE。NAME 特性定义了你想设置的参数名称，而 VALUE 属性则向参数赋一个值。所有值都用双引号括起来，既便是数值也不例外。

程序列表中最后三个参数，是任何环境下使用该控件时能够设置的永久性特性。Caption（标题）特性改变按钮顶部的文本，就好象你是在 Visual C++ 中使用该控件一样， OnOff 特性是这个控件所特有的——它允许你创建一个 On/Off 按钮，第 11 章中我们要创建一个实际控件，到那时就会看到它如何工作。ModalResult 参数允许控件返回一个基于其当前标题的值，例如，按钮标题是 On 时，控件返回为 8。

还有一些通常你不无须关心的参数，但把控件作为 Web 页一部分时却需要进行处理。_Version 参数在下载过程中起作用，它帮助浏览器对照服务器上的控件版本来确定客户机的控件版本是否已经过时。_EntentX 和 _EntentY 特性则在 Web 页上将控件定位，最后，_StockProps 属性定义控件的族系特性——通常不需设置它们。

接下来你将看到的是一个串。支持 ActiveX 的浏览器将会忽略这个串。只有当客户机不支持 ActiveX 时，这个串才被显示出来。实质上它从另一方面告诉用户，如果它们的浏览器提供了正确的支持，那么，它们立即就能看到一个 ActiveX 控件。你不必受限于一个串，尽管你总是可以在这里放上一个等价的 HTML 控件。例如，这个 ActiveX 控件是一个能用 HTML 代码代替的按钮，尽管它不再这样灵活了。

前面我们谈到可以从 Internet 上下载的文件有三种类型：OCX、CAB 和 INF。OCX 文件格式，又称为可移植的可执行文件（PE）格式，使你能用最终形式向客户机发送控件。在另一边不需附加处理，从长远看来，意味着你遇到的支持问题会更少。CAB 文件格式与 Microsoft 发行(shipping)产品(如 Windows)时使

用一样的格式。它独到的优点是让你在一个单一包中发行多个控件。这种格式还提供了文件压缩功能，从而减少用户的下载时间。缺点是太复杂，你不得不创建几个安装文件，还包括一个 INF 文件。另外，一旦下载完成之后，用户需要等待 CAB 文件解压缩，然后安装它们，这就使得这种方法更可能会使用户忍受不了，并在它们试图取消这种处理过程时引起许多问题。INF 格式允许选择性地安装一个或多个控件，它使用的 INF 文件的格式类似于 Windows。用户访问你的站点时，就会看到某种类型的安装屏幕显示。它的缺点（downside）是仍然要创建 INF 文件。使用 INF 文件，意味着当用户等待页下载时，让他感到完全厌烦的机会会更少，而且他们至少是过程的一部分。我们将在本章后面讨论 INF 格式（参见“使用 DIANTZ.EXE 建立组件下载(CAB)文件”）。

8.4 COM+：Internet 的未来

对 Windows 进行的任何技术讨论，必然要包括对组件对象模型 COM 的讨论。COM 是一种对象技术，已成为 Windows 的全部（技术）的基础，并且还是 Microsoft 向 Internet 进军的一部分。COM 不仅与用户相互作用，而且还是创建类似于应用程序中使用的按钮之类的组件的技术基础。换言之，不以某种方式讨论一下 COM，就不能开始对 Internet 的 Windows 或 Microsoft 版本进行讨论。

就在一年前，ActiveX 还是表达 Microsoft 进军 Internet 的新版 COM 的中心词汇。但是，现在已经证明，ActiveX 太庞大（尽管 ActiveX 控件远远小于被它

取代的 OCX），太依赖于浏览器，而且对大多数人来说又有太多的安全风险。另外，Microsoft 市场部正在用 ActiveX 来描述 Microsoft 正在发布的几乎每一项新技术，这样只能使客户更加迷惑。

当 COM+及分布式网际网应用程序（Distributed interNet Application，DNA）出现在 Internet 时，就又有了两个新鲜而又激动人心的术语让你学习了。ActiveX 不再是描述 Microsoft 的 COM 技术诸个方面的术语，它仅关系到在应用程序或内部网站点中创建并使用的组件。如果你正在谈论使能技术或 DNA 技术，如果你正在谈论 COM+技术的进展情况，那么在 Internet 上干任何事现在都称为 COM+。

WEB 链接 可以在 http://www.microsoft.com/com/default.htm 上得到 Microsoft 种种最新的 COM 技术。这个站点可使你一直跟得上 Microsoft 正在引入的各种与 COM 相关的技术发展，以及预期它们是如何发展的。

事实上，作为 C++程序员，需要了解相当多的基于 COM 的技术。下列章节将快速地浏览一下这些技术，并讨论如何将它们放入一般方案中。显然，在 Microsoft 升级其 COM 技术时，这种讨论也会有所变化。

分布式 COM（DCOM）——网络接口 COM 的这一版允许组件和应用程序在网络上进行通信。DCOM 设计得可以在相当多的网络协议——包括 Internet 使用的 TCP/IP——上进行通信。DCOM 为开发人员所能做到的，是允许客户的应用程序只要使用标准的 COM 接口，就可和运行于服务器上的服务直接通信。实际上，这就意味着你可以设计一个控件或一小段程序，

使得客户机的应用程序从服务器接收数据，然后在本地对这些数据进行计算。另外，DCOM 使得我们可以在客户机和服务器间进行安全通信，而不需关心所用的协议。

COM+——功能　标准 COM 接口扩充。它是现有的 COM 标准的扩充，使开发人员能利用特殊的 COM+调用来设计高级的控件。COM+提供的最重要的 COM 扩充是数据捆绑，允许组件提供数据库访问。添加数据库访问，意味着可以使用 COM+为能访问远程数据的 Internet 建立组件。COM+还是分布式网际网（DNA）体系结构的基础，而 DNA 是 Microsoft 的最新的 Internet 版本。

理论上，DNA 允许 COM 组件可以在任意平台上执行，并且允许开发人员可以使用任意语言写出该组件。可以使用任意语言和任意平台进行工作的这部分能力，来源于一个称之为侦听器的 COM+新特色。使用侦听器允许 COM+ 组件可以在运行期间调用多种服务，而不是依赖于某一具体服务。你将使用侦听器接收和处理与实例创建、调用与返回、出错以及实例删除有关的事件。侦听器还提供了处理事务及系统监督的机制。

Microsoft 事务服务器（MTS）——功能　COM 服务器。信不信由你，MTS 是 COM 的一部分，它允许开发人员创建“轻型(lite)”控件，这种控件集中于处理组件背后的事务逻辑，而不是处理组件所需要的全部接口。建立轻型控件意味着能够更快地创建它们，使用时下载时间更少。MTS 将成为 IIS4.x 的一部分。

ActiveX——功能　组件建立。ActiveX 最初是作为 Internet 的一项新技术而

引入的，它是一种组件建立技术，它是 Microsoft 组件技术的第三个版本。将 ActiveX 从以前的 OLE 控件扩充（OCX）中分离出来的原因之一是，可以在类似于 Internet 的高级潜在网络中使用它们。它们还设计得可以与浏览器一起工作，尽管在写出本书时真正使用它们的唯一浏览器只有 Internel Explorer。ActiveX 控件的特色还包括增量绘制和代码签名，使得用户可以在控件执行前识别出它们的作者。

8.5 从 Internet 上下载 ActiveX

本章中介绍的大部分 HTML 标记的作用是很明显的，浏览器看到标记后就把标记的要求显示出来。整个过程简单易懂。但是 ActiveX <OBJECT>标记却相当不简单。当浏览器看到一个<OBJECT> 标记后会发生什么事？我们知道它不会只是简单地在屏幕绘制控件——ActiveX 控件提供了比这多得多的功能。

Microsoft 已作出了一个新的 Windows API 调用供浏览器使用——CoGetClassObjectFromURL。当浏览器看到一个 < OBJECT> 标记时，它先分析出 CLASSID 、CODEBASE 和 _Version 参数，并把这些参数传送给 CoGetClassObjectFromURL。如有必要，这个 API 可以来完成控件的下载、验证和安装。这个 API 调用做的第一步，是检查当前注册表中是否包含着对 CLASSID 的引用。你可以在注册表的 HKEY_CLASSES_ROOT|CLSID 关键字中找到这一引用。如图 8.10 所示。

WEB 链接 关于<OBJECT>标记的规范及其相联系的 API 调用处于经常变动之中，若想得到有关<OBJECT>标记的最新信息，请参看 http://www.w3.org/TR/WD-object.html。

如果 CoGetClassObjectFromURL 找到了安装在客户上的 ActiveX 控件的实例，它就检查其版本号。当安装在客户机上的控件版本号大于等于 HTML 页上指出的版本号时，CoGetClassObjectFromURL 就装载本地控件，而不再从 Internet 上下载这个控件。控件装载完后，CoGetClassObjectFromURL 就为它创建一个类工厂（Class Factory），并把类工厂传送回浏览器（一个类工厂和其它工厂一样工作，它产生出一些项，在上述情况下，它允许浏览器产生对象的一个实例）。否则，API 调用就请求浏览器从 Internet 站点下载代码。

下载是一个异步处理过程——Windows 等待下载完成的过程中能完成其它与 ActiveX 有关的任务。一旦浏览器完成下载并在必要时解压缩后，就调用 Windows 信任提供者服务函数 WinVerifyTrust。这个服务到 ActiveX 控件内部检查并确定是否有签名块。签名块包含有作者姓名、公开密钥以及控件内容加密摘要等。可以把 ActiveX 控件签名块看成是支票的签名、驾驶执照或一份合约，它不仅指出你是谁，而且还验证你是不是你说的你是谁(you are who you say you are)。若 WinVerifyTrust 调用找到了一个签名块（也称为证书），它就对证书进行证实。每个证书能够包含一个父证书的名称。WinVerifyTrust 沿着证书的分层树上溯直至到达根证书。然后它就在可信任的根证书列表中找这份根证书。若能找到，CoGetClassObjectFromURL 自动地装入控件并为它创建一个类工厂。否则，用户就会收到一条该控件不可信的消息（当我们在第 10 章中创建

一个 ActiveX 控件并在 Web 页中使用时你会看到这一消息的）。

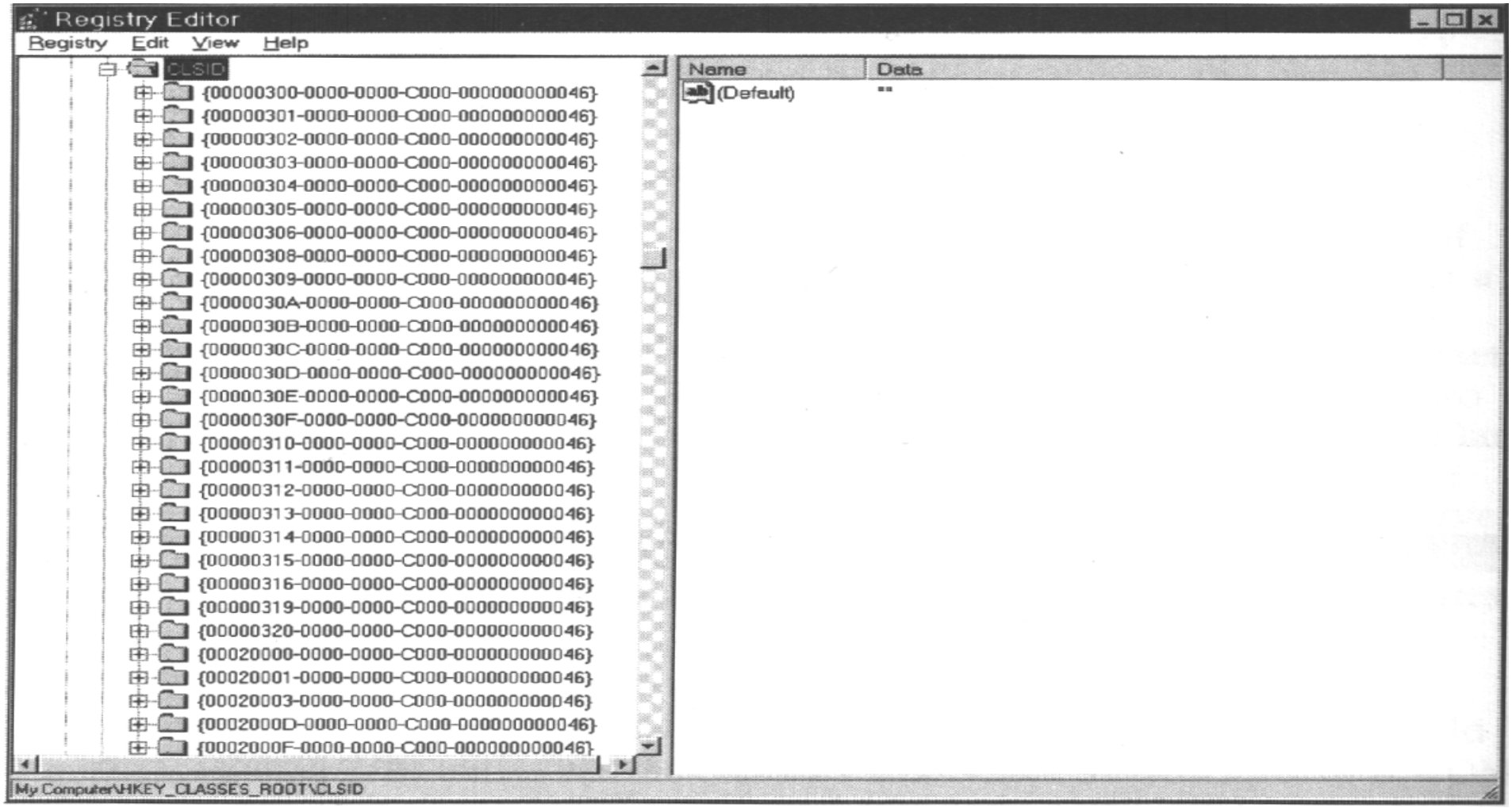


图 8.10　Windows 在 HKEY_CLASSES_ROOT|CLSID 关键字中存放 ActiveX 控件的类标识符

注　公开密钥的作用如同共用房屋或公寓的安全门钥匙——它使你能进入

大楼。私有密钥则如同公用房屋或公寓的室门钥匙——它使你进入你的起居室。

即使一个 ActiveX 控件不出现在可信任的列表中，用户仍可选择是否安装这个 ActiveX 控件。若选择安装这个控件，用户就会被问及是否想把控件作者添加到他们的信任列表中去。向列表添加作者姓名，意味着同一作者的任一新控件将会立即得到承认。我们将在第十四章中更详细地讨论安全问题。

这些控件安装在什么地方呢？理论上，控件设计者可以把控件放在任何位置。但是，大部分从 Internet 下载的 ActiveX 控件，都不会自动地出现在用户的类似 System（系统）文件夹这样的地方。实际上，它们被添加到名为 OCCACHE 的一个特殊文件夹中，这个文件夹可以出现在用户机器的任何位置。通常你可以在 Windows 主文件夹、/SYSTEM 文件夹或用户的 Internet 文件夹中找到这个文件夹。（一些浏览器还使用 ActiveX Control Cache 文件夹，它位于存放 ActiveX 控件的 Window 主文件夹中）。

注 可以找到从 Internet 上下载的 ActiveX 控件的最普遍的位置是 OCCAHE 文件夹。

装入控件还不是工作的全部，控件在安装过程中还要对自身进行注册。在大多数情况下，这意味着要调用 DllRegisterServer API 函数。完成安装及注册后，CoGetClassObjectFromURL 函数把与控件相联系的类工厂传回浏览器。浏览器使用类工厂创建一个对象实例，并对<OBJECT>标记传送的参数进行初始化，（必要的话）将控件显示到屏幕上。

ActiveX 与 Netscape Navigator

如果你管理着一个内部网站点，你总是能控制让谁访问站点以及它们应该用什么浏览器来进行访问。这时，如果想使用 ActiveX，那么，你就只需要简单地要求每个人都使用 Internet Explorer 就可以了。但是，对于 Internet 站点来说却做不到这一点。所以就不得不确保站点支持尽可能多的浏览器。Internet Explorer 直接支持 ActiveX，而 Netscape Navigator 却不是这样。幸运地是，有一种可供选择的办法使 ActiveX 可以在这样的环境中工作。

NCompass Labs,Inc.公司生产了一种称为 ScriptActive 的插件，它允许使用 Netscape Navigator 3.0 看到含有 ActiveX 控件的 HTML 页。在写本书时，NCompass 的插件和 Internet Explorer 3.0 的 ActiveX 控件在能力上是类似的。在使用这个插件之前，可能需要花些时间研究一下它。NCompass 在 http: //www.ncompasslabs.com/为 ScriptActive（及其它产品）提供支持。

不幸的是，使用当前版本的 ScriptActive 不太简单。你不得不操作<OBJECT>标记的一些特殊格式来让它工作。还算幸运，这个插件中还包含有可以完成大部分工作的 HTML 转换实用程序，名称是 NConvert。程序列表 8.7 显示的是使用这个实用程序创建标准的 ActiveMovie 控件所产生的结果。

程序列表 8.7

```
<!- 插入一个 ActiveMovie 控件 ->
<OBJECT ID="ActiveMovie1" WIDTH=267 HEIGHT=73
        CLASSID="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A"
```

```
        CODEBASE="http://aux/controls">
        <PARAM NAME="_ExtentX" VALUE="7038">
        <PARAM NAME="_ExtentY" VALUE="1931">
<EMBED NAME="ActiveMovie1" WIDTH=267 HEIGHT=73
        CLASSID="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A"
        CODEBASE="http://aux/controls"
        TYPE="application/oleobject"
      PARAM__ExtentX="7038"
      PARAM__ExtentY="1931"
></OBJECT>
```

警告 一旦从一个特定的<OBJECT>标记向一个<OBJECT>标记和<EMBED>标记组合的转换，就不要通过单击 HTML 页上邻接元素的<OBJECT>标记来编辑 ActiveX 控件。在某些情况下，编辑器会改变<EMBED>标记，使之不能正常工作。最好创建自己的 Web 页，并在 Internet Explorer 中测试它，然后添加所需的<EMBED>标记，使它通过 NCompass ScriptActive 插件来进行工作。

请注意，NConvert 采纳了<OBJECT>标记的信息并在同一信息上添加一个<EMBED>标记。Navigator 能识别出<EMBED>标记是要向插件中传达某种信息，插件用 ScriptActive 完成它的工作。因为 Internet Explorer 不能理解上下

文中的<EMBED>标记，多出来的这些信息不引起任何问题。这一部分额外的代码的（存在）理由是相当明显的。ScriptActive插件需要一个特殊的TYPE属性，使用它指定对象的因特网MIME类型。根据供货商的说法，ScriptActive将来的版本中不再需要这一附加信息，并且能够不加修改地使用<OBJECT>标记。

技巧 一旦NCompass ScriptActive插件安装成功，用右键单击文件时，就会看到NConvert关联菜单项，这个项包含两个选项：NConvert（转换）和 Set Destination Folder(目的文件夹设置)。使用Set Destination Folder这个选项，可以告诉实用程序，你想把转换过的文件放于何处，如果不想用转换过的文件复盖原来的文件，那么，就要指定一个特殊目录。Convert选项的作用是插入产品的当前版本所需的所有<EMBED>标记。

ScriptActive还支持VBScript，但是它是通过添加一个外部的AXS文件来实现的。一定要保证AXS文件与包含VBSCript的HTML文件在同一个目录中。NConvert添加一些JavaScript行，从而建立Navigator和ScriptActive之间的联系。ScriptActive收到请求后，就在AXS文件中寻找合适的VBScript代码。不幸地是，ScriptActive对VBscript的支持相当有限，在使AXS文件工作之前，请对这个文件中的VBScript代码进行再加工。如果计划在一个站点上同时使用Internet Explorer和Navigator，那么，最好的方法是优先使用JavaScript而尽量避免使用VBScript。

使用ScriptActive带来的另一个问题是，它不能总是让你访问与对象相联系的属性。事实上，支持的级别甚至会随机器而变，具有相同配置且使用相同硬

件的两台机器，在一个具体的 Web 页上，也可能显示出不同的结果。不同的控件看起来也提供可变化的访问级别，复杂的控件往往比简单的控件出的问题更多。考虑到 Internet 处于不断变化之中且 ActiveX 技术相对比较年轻，出现这些问题也不值得大惊小怪。你能够期盼获得的最好结果是显示控件于屏幕，至于它能不能实际工作则是另一回事了。

尽管存在这些问题，使用 ActiveX 控件的前途是太光明了，不允许忽视它。类似于 ScriptActive 的第三方厂商的插件，差不多是在 Web 站点上能够使用 ActiveX 的唯一途径，并且仍能保证访问站点的大多数人，能够实实在在地看到你所提供的内容。有传言说，Netscape Navigator 的下一版将直接支持 ActiveX，但在今天，ScriptActive 依然是你最好的解决方案。

8.6 增强 HTML 的功能

我们可以对一个 Web 站点进行许多方面的增强(Enhancement)。实际上，你将在 Internet 上看到的各种增强，大概能塞满整整这一本书。对 Web 页使用增强，会产生两方面的问题：第一，大部分技术在数据格式上需要一些约定，你不得不考虑，一个访问者是否真想下载数据，从而产生所要的效果；第二，可以发现，一些增强在这个浏览器工作得不错，而在另一个浏览器却不行。在后面的章节中我们将陷入这些不兼容性之中，所以从现在开始，我不再对这一点进行反复说明。

对 Web 站点能够进行的最流行的两种增强是声音和动画。到目前为止，声

音在二者之中更为流行。但这里我们并不是要看完整的(full-fledge)CD 或优雅的传真图像。声音以声音字节流的形式出现，声音字节流是一小段声音，可以添加到你试图在 Web 站点上获得的效果中去。大多数情况下，人们使用 WAV 文件向站点添加声音，也在一些人使用 MIDI 文件添加音乐。不论你使用声音支持的哪一种形式，当决定添加多少声音时要小心谨慎，少一些为好。应该给人关闭声音的选择，以便于它们在办公室工作时能访问你的站点。

使用字幕是向 Web 页添加动画形式的一种途径。字幕所做的一切就是显示一个滚动的消息或图形图像。这一标记的文本处理能力受到一些限制，任何添加到<MARQUEE>标记中的“花体”(fancy)文本将不在屏幕上显示，你开始使用的文本就是一直不变的文本（字体）。这就意味着，在实际显示字幕之前，要进行你想做的格式化（这也正是许多程序员宁愿使用图像的原因）。Microsoft 还提供了字幕的 ActiveX 版本，作为 ActiveX Control Pab 实用程序的一部分。

警告 使用<MARQUEE>标记时总是要使用 LOOP 属性，因为如果不使用 LOOP 属性，支持<MARQUEE>的有些浏览器就会做出奇怪的反应。你第一次使用这个特定标记时，注意收集用户反馈是非常重要的，因为在旧式的浏览器上，它非常容易引起麻烦。虽然有些不完美，但进行一些小的改动，也许就能使这个标记在你的站点上确实支持这个标记的每个浏览器工作。

在不用增加太多开销（但确实要增加一些）的前提下，动画确实能为站点增光。问题当然是要花费时间来创建某种动画例程。一些人也许试图写一个脚本或创建一个 ActiveX 控件来完成这一工作，但这可用更好的方式来解决。如

果你不想创建花哨的动画方案的话，就不必被迫去这样做。只需使用一个动画 GIF 即可。如果你和我一样，也许会问动画 GIF 到底是什么呢？因为大多数程序员从未见到过它。实际上，如果你在 Internet 上花费了很多时间，也许已经看到了一两个这样的文件。一个动画 GIF 是由一组图像组成的，这些图像在你的 Web 站点上一次显示一幅。如果非常细心，你就能够使用动画 GIF 创建动画效果，既增加了快乐，又收藏了资源。最好的消息是绝大多数浏览器都完全支持动画 GIF。

WEB 链接 在许多 Internet 站点可以看到活动的动画 GIF 。一个更有趣的站点是 http://www.wanderers2.com/rose/animate.html，这个站点提供了一个站点索引，访问这些站点就能看到多种动画 GIF。到各种站点去转一转，会帮助你明白什么能做，什么做不了。你还可以下载动画 GIF 向导，从而使你自己的动画 GIF 在线，并能学习有关怎样制作动画 GIF 的全部知识。

利用声音和其它效果使 Web 页面更引人入胜

(WAV 文件中的)声音片段(bites)对于装饰一个 Web 站点大有好处。但是，如果每次用户完成一个最小的任务时，你都连续地演奏声音，就会令用户心烦意乱。因此声音片段要短且要用在点子上。添加声音要像厨师添加调料一样，少许的调料对烧出味道鲜美的菜肴大有裨益，用多了调料反而使人倒胃口。

Web 站点的东西越小越好。最近到一些站点的访问表明：一些站点五彩缤纷、非常有趣且下载又非常之快。看了一下代码，反映了一个共同现象，那就

是所有作者都用表格和框架(frame)将页面分成更小的区，每个小区使用小图标和图形，使页面看起来非常漂亮。另外，作者还用某种谐调的绘制方案，使得每个框架(frame) 有独特的外观，并且，页面也没有显得俗气。

下面来看一个示例页面，这一页对我们在本章中已讨论过的概念作了示范性说明。程序列表 8.8 到 8.12 显示的是怎样制作这一页，注意到这里包含了一些小的图形和框架(frame)来组织信息。在这里还可以看到本章中学习过的其它技巧（也有一些新东西）。图 8.11 显示了这一页看起来是什么样——至少从灰度上，你需要适当地使用颜色来构造页面。

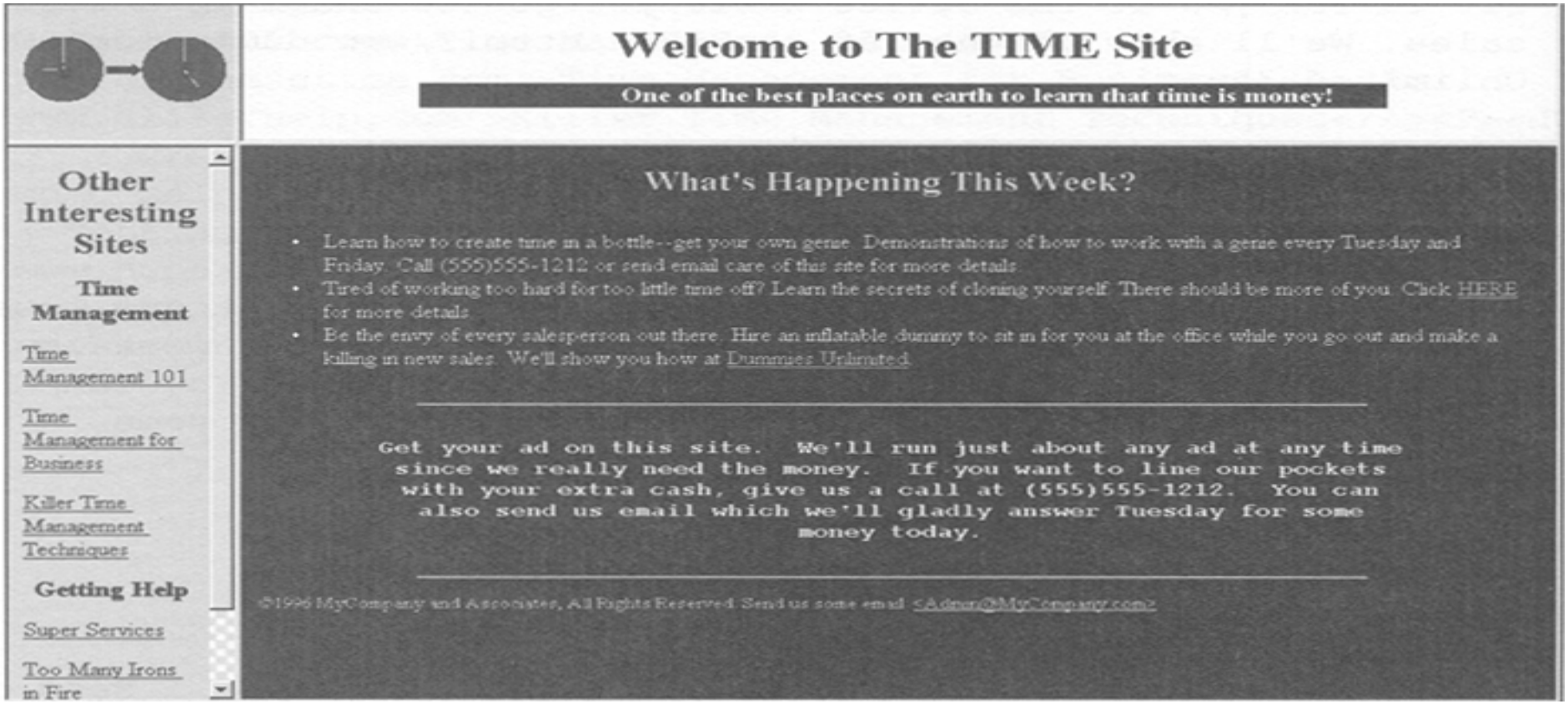


图 8.11　我们例子中的 Web 页表明：不必使用大量的资源，只要将一些简单标记组合起来就能创建一个令人感兴趣的 Web 页面

程序列表 8.8

```
<!-FRAME.HTM->
<HTML>
<HEAD>
```

```
<TITLE>Sample Web Page</TITLE>
</HEAD>

<!-设置框架参数->
<FRAMESET COLS="15%,*" ROWS="20%,80%">

<!-加入一个图标->
<FRAME SCROLLING=NO SRC=Icon.HTM>

<!-创建一个标题->
<FRAME SCROLLING=NO SRC=Heading.HTM>

<!-列出可能会感兴趣的一些其它站点->
<FRAME SRC=OthrSite.HTM>

<!-显示页的主要内容->
<FRAME SRC=Main.HTM>

</BODY>
```

程序列表 8.9

```
<!-HEADING.HTM->
<HTML>
<HEAD>
<TITLE>New Page</TITLE>
</HEAD>
<BODY BGCOLOR=WHITE>

<!-为我们的站点定义标题和字幕->
<CENTER><H1>Welcome to The TIME Site</H1>
<H3><FONT COLOR=WHITE>
<MARQUEE BGCOLOR=BLUE BEHAVIOR=ALTERNATE WIDTH="75%"
LOOP=INFINITE>
   One of the best places on earth to learn that time is money!
</MARQUEE>
</H3><FONT COLOR=BLACK>
</CENTER>

</BODY>
</HTML>
```

程序列表 8.10

```
<!-ICON.HTM->
<HTML>
<HEAD>
<TITLE>New Page</TITLE>
</HEAD>
<BODY BGCOLOR=YELLOW>

<!-显示我们的图标->
<IMG SRC=TimeIt.GIF ALT="From 9 to 5">

</BODY>
</HTML>
```

程序列表 8.11

```
<!-OTHRSITE.HTM->
<HTML>
<HEAD>
<TITLE>New Page</TITLE>
</HEAD>
<BODY BGCOLOR=YELLOW>
```

```
<!-显 示 标 题 ->
<CENTER><FONT COLOR=BLUE>
<H2>Other Interesting Sites</H2>
</CENTER><FONT COLOR=BLACK>

<!-显 示 时 间 管 理 链 接 ->
<P><P></CENTER><H3>Time Management</H3></CENTER>
<A HREF="http://www.time101.com">Time Management 101</A><P>
<A HREF="http://www.businesstime.com">Time Management for Business</A><P>
<A HREF="http://www.killerhelp.com">Killer Timer Management Techniques</A><P>

<!-显 示 添 加 的 帮 助 链 接 ->
<P><P><CENTER><H3>Getting Help</H3></CENTER>
<A HREF="http://www.superserve.com">Super Services</A><P>
<A HREF="http://www.irons.com">Too Many Irons in Fire</A><P>
<A HREF="http://www.business.com">Business Help to Go</A><P>
</BODY>
</HTML>
```

程序列表 8.12

```
<!-MAIN.HTM->
<HTML>
<HEAD>
<TITLE>New Page</TITLE>
</HEAD>
<BODY BGCOLOR=BLUE LINK=YELLOW VLINK=SILVER>

<!-为进入站点的人演奏声音->
<!-Internet Explorer 标记>
<BGSOUND SRC="TwilZone.WAV" LOOP=1>
<!Navigator 标记>
<EMBED SRC="TwilZone.WAV" HIDDEN=TRUE LOOP=1>

<!-显示标题->
<CENTER><FONT COLOR=YELLOW>
<H2>What't Happening This Week?</H2>
</CENTER><FONT COLOR=WHITE>

<!-显示当前事件列表->
<UL>
```

<LI>Learn how to creat time in a bottle--get your own genie.
demonstrarions of how to work with a genie every Tuesday and Friday.
Call (555)555-1212 or send email care of this site for more details.
<LI>Tired of working too hard for too little time off? Learn the
secrets of cloning yourself. There should be more of you. Click<A HREF
="Clone">HERE</A> for more details.
<LI>Be the envy of every salesperson out there. Hire an inflatable
dummy to sit in for you at the office while you go out and make killing
in new sales. We'll show you how at<A HREF="http://www.dummy.com"> Dummies
Unlimited</A>
</UL>

<!-为将来的潜在销售提供一个广告->
<CENTER><HR WIDTH=75%></CENTER>
<STRONG><H3><CENTER><PRE>
Get your ad on this site. We'll run just about any ad at any time since
we really need the money. If you want to line our pockets with your
extra cash, give us a call at (555)555-1212. You can also send us email
which we'll gladly answer Tuesday for some money today.
</PRE></CENTER></H3></STRONG>

```
<!-请给我们发电子邮件->
<CENTER><HR WIDTH=75%></CENTER>
<FONT COLOR=YELLOW SIZE=-1>
&#169;1996 MyCompany and Associates, All Rights Reserved.  Send us  some
email:<A HREF="mailto:admin@mycompany.com">&lt;Admin@MyCompany.com&gt;
</A>
</BODY>
</HTML>
```

注释　这个例子并没有涵盖你所能做的一切事情，下一节中我们将讨论动画 GIF。

正如你所看到的，这个站点的画面还不是很完美，但是，它大概比你过去访问过的一些地方更令人感兴趣。让我们用些时间把这些代码分析一下，看看其中包括的更具诱惑力的部分。你注意到的第一件事是站点分成了 5 个文件，可以发现，通过使用分离开的文件，使得对具有框架的站点进行维护更加容易。另外，站点上的所有页可以使用相同的标题及图标文件，这样可以为用户节约下载时间。程序列表 8.8 中的代码完成的全部工作，就是把浏览器工作区分成若干框架，然后再定义每个单元属于谁。应注意，能够明确地用百分数或*号来定义每个框架的区域。浏览器首先按百分比把可见的视屏分给定义的区域，然后再把剩余区域平均分配给其它单元。使用这一特色能够在 Web 页上强制实现某一方面的比例。

注　使用(*)号定义某框架时，其意义是浏览器为已指定的其它框架分配空

间后，该框架占用剩下的窗口空间。

程序列表 8.9 含有 Web 站点的标题。这里的代码中，唯一令人感兴趣的就是字幕区域颜色的使用。在整个站点谐调着色可以使之美观，而且又不花费用户多少下载时间。还应注意到，这个字幕在前后跳跃而不是兜着圈转。BEHAVIOR 属性就是负责这种变化方式的。使用多种属性来提供特殊效果，能够使你的站点更加令人注目。还应注意这是站点上唯一使用动画的地方，再多了就会使人反感。

可移植性 <MARQUEE>标记仅在 Internet Explorer 中工作。若使用 Navigator，仍可以看到标题区，但没有滚动的字幕。解决这一问题的较好的方案是，创建一个 Java applet 来激活<MARQUEE>标记的行为。也可以创建一个 ActiveX 控件来完成这件事，但不得不使用类似于 NCompass ScriptActive 之类的插件使控件与 Navigator 兼容。

程序列表 8.10 中，只有一个关于我们站点的图标的<IMG>标记。须注意<BODY> 标记一定含有一个 BGCOLOR 属性，从而使页的背景色与图标能匹配起来。这时，若用户改变了显示区的大小，它们就将看不到图标下面的页面颜色了。

许多站点包含有到其它站点的链接，但总体上不妨碍站点的工作。程序列表 8.11 中代码的目的就在于此。它在不占用 Web 页本身空间的前提下，组织了从当前站点到因特网其它站点的链接。这样做，不仅使信息更容易找到，而且还可以把它作为你自己站点的一个分类索引。

在程序列表 8.12 中可以找到一些有趣的编码技巧。第一，标记的属性如何

才能用于全页以获得特殊效果。例如，应注意如何使用<PRE>标记为页面中间的注释提供特殊格式。还应注意到这一页使用了反视频色（暗背景，亮文字，而不是通常的那样）这一特殊效果，使页面的主要部分从它周围的辅助区域中突出出来。Internet的几个站点都使用了这种相同的效果。例如：

http://www.pacbell.com.

这一页还使用了<BGSOUND>这个新标记，使得用户进入你的Web站点时，就为他演奏一小段声音。特殊的LOOP属性用于定义声音片段应该演奏几遍。仅使用<BGSOUND>标记本身，只演奏一次，通常最好包含进LOOP属性，以防止出现不兼容的问题。Microsoft也许会决定，今后某时开始，你必须明确地指示<BGSOUND>标记应把声音演奏几遍。

可移植性问题 请注意，<BGSOUND>标记是Internet Explorer专用的，必须用<EMBED>标记使Navigator也能演奏声音。有大量的与此相类似的标记——Internet Explorer不认识Navigator标记，而Navigator也不认识Internet Explorer的标记。作为程序员，必须通过添加所需的对应代码来弥补这一问题上的缺陷。

在程序列表8.12尾部，还可以看到完成某些任务的两种新方法。第一个新技巧是使用&#<数字>来表示特殊字符，例如，用&#169产生版权符号。如果想把这些特殊字符紧挨着一个普通字符放置，只须用分号将它们分开即可，例如：©1996。第二个新技巧是显示如果让别人向你发电子邮件，你的Web站点是什么?这只须提供一个HREF即可：

```
<A HREF="mailto:admin@mycompany.com">&lt;Admin@MyCompany.Com&gt;</A>
```

这里的秘密在链接的“mailto”部分中。请注意链接文本使用了特殊字符的另一种形式，&lt与 &gt提供了将链接文本括起来的“小于”及“大于”号。

技巧 ActiveX Control Pad 提供的 HTML 参考，在某些方面可能不全，但它提供了这些特殊字符的完整列表。查看一下主索引下的 Character Set 类即可。

添加动画 GIF

如果说前面几节中还没有介绍足够多的技术使你的 Web 站点光彩夺目的话，那么，还有许多其它你能够使用的方法。世界上的 Web 站点所有者最钟爱的方法，就是使用动画 GIF，一个动画 GIF 所做的事，就是把几幅图片打包成为一个文件，浏览器则一幅一幅地连续放映这些图片，形成连续动画的错觉。还能够使用特殊效果，来创建使用 GIF 的滑动演示，这种方法带来的问题是下载时间——滑动演示会造成用户的下载能力相当紧张。

注释 本节教你如何使用 Alchemy Mind Works 的 GIF Construction Set 来创建 GIF。这个软件可从几个地方下载，最好的地点是直接从供货商那儿下载：

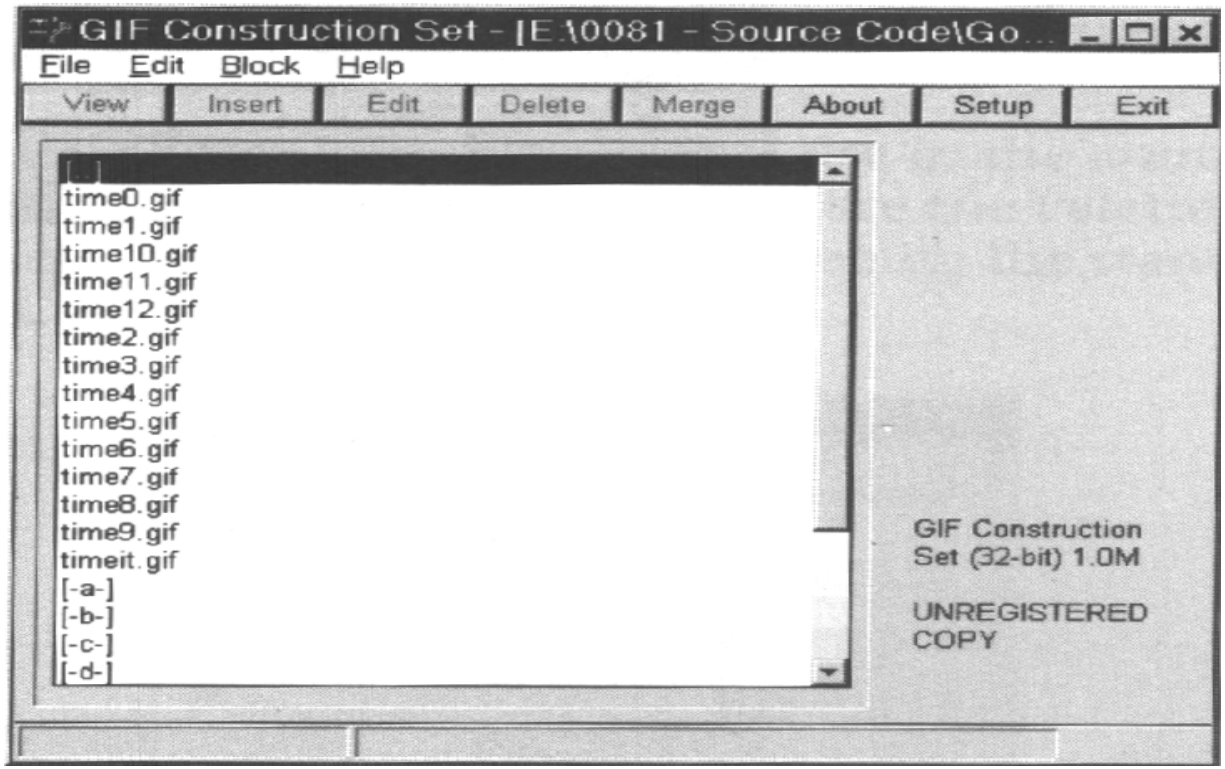
http://www.mindworkshop.com/alchemy/gifcon.html。还可以从本章前面提到的动画 GIF 观看站点中下载：

http://www.wanderers2.com/rose/animate .html。

本例中使用 GIF Construction Set 的理由有两条。第一，它是共享软件，你可以把它从 Internet 上整个地下载回来，然后用到本例中。第二，它确实是一个相当好的程序，大多数人发现用它创建动画 GIF 非常好。但是你会注意到，这个程序缺少一个实际的画图程序，但 Windows 已经提供了 Paintbrush 或 MS Paint。

如果画图程序不直接支持 GIF 文件格式(Paintbrush 及 MS Paint 都不支持），则还要有一个图形转换的实用程序。Alchemy Mind Works 的 Graphics Workshop 和 JASC.Inc 公司的 Paint Shop Pro 都是出色的图形转换程序。这两家厂商都提供产品的共享软件版本。可以在前面注释中提供的 Internet 站点中找到 Alchemy Mind Works 的产品，JASC 产品则出现在各种 BBS 和 CompuServe 论坛上（你读到本书时，也许它们也有了自己的 Internet 站点）。

打开 GIF Construction Set ，首先可以看到的是 File|Open 对话框，如下图所示：



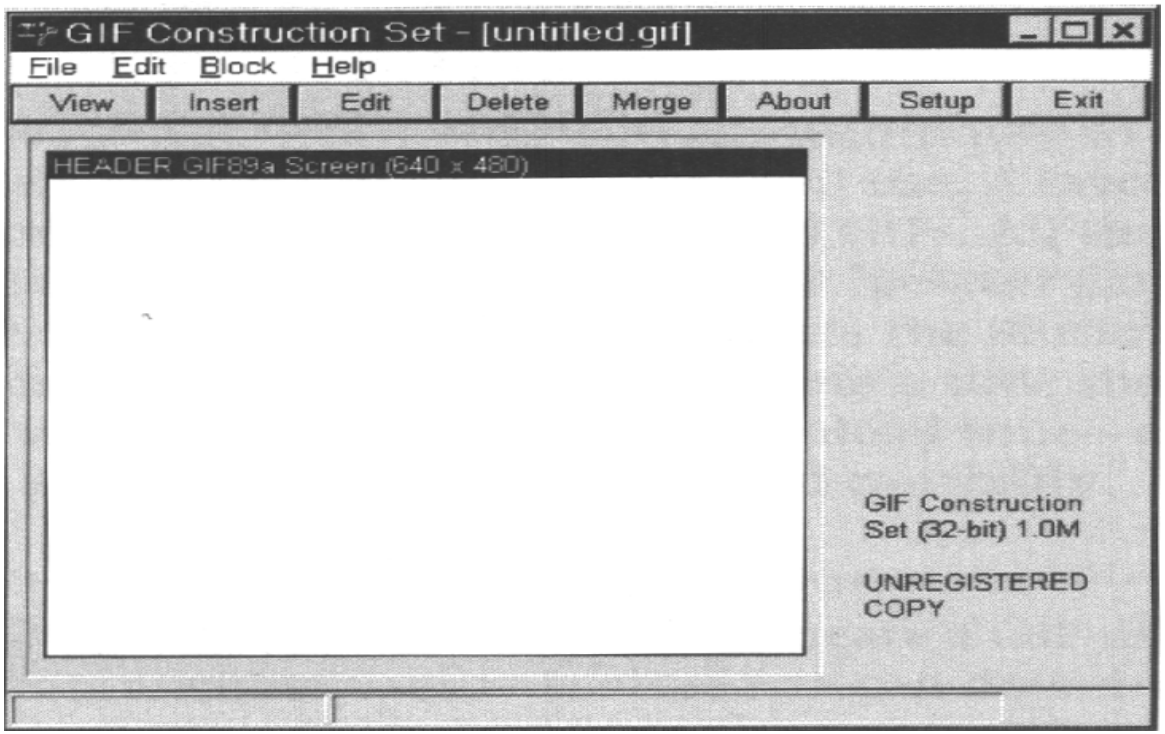
注释 和本书中所有其它的源代码材料一样，可从 http://www.osborne.com 这个 Osborne Web 站点下载 TIME.GIF。

请注意，该目录下已经有了几个 GIF 文件。TimeIt.GIF 是你在图 8.11 中所看到的静态文件，Time0.GIF 是基文件——创建动画效果使用的空白文件。创建动画时先创建这样一个空白文件，可以节省大量的时间。事实上，卡通作家也使用这一技术！他们把一个动画的公用元素一次性地画在各个单页上，然后，将它们组合起来就构成动画。只有那些唯一的项才需要每次都画。Time1.GIF

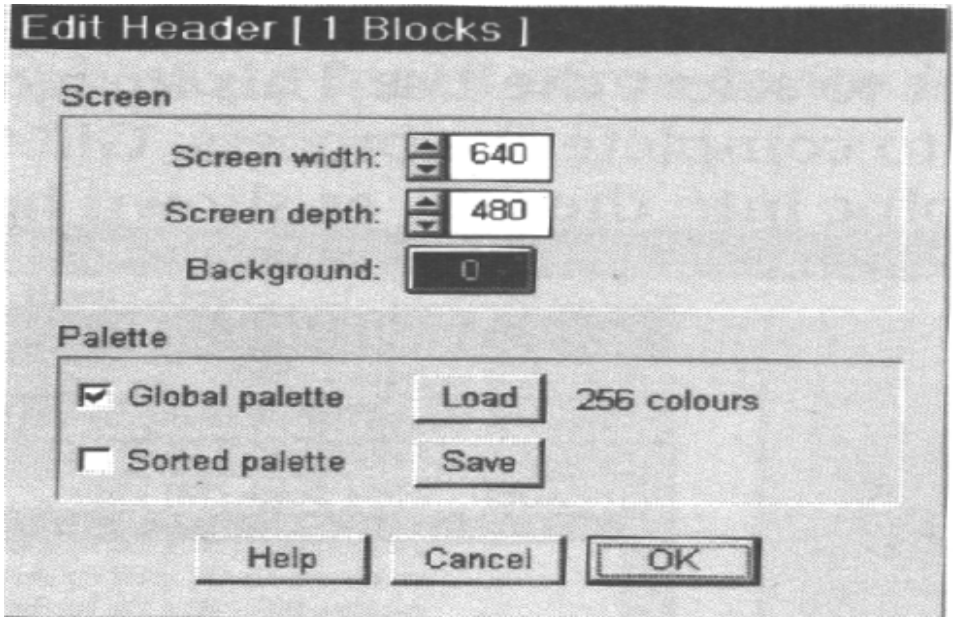
到 Time12.GIF 是实际的动画文件，把每一个看成一个动画单元好了。

让我们用这些动画单元文件来制作动画 GIF。下列操作过程并不意味着要把你束缚到一个特定模式上，但它确实说明了使用 GIF Construction Set 制作 GIF 的一种方法。

1. 用 File|New 命令创建一个新的 GIF，你将看到如下图所示的空白 GIF 对话框。GIF Construction Set 总是假定使用标准的 640×480 像素的显示区。我们需要改变其大小。



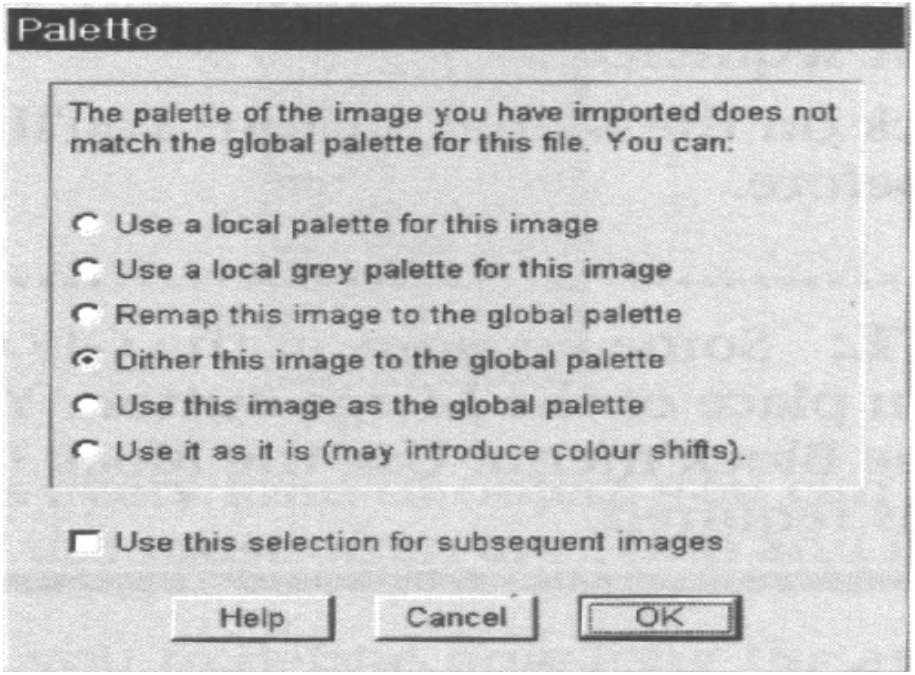
2. 双击 Header 项，就可以看到 Edit Header 对话框，如下图所示。这个对话框允许改变与 GIF 有关的特征，比如说它的大小。



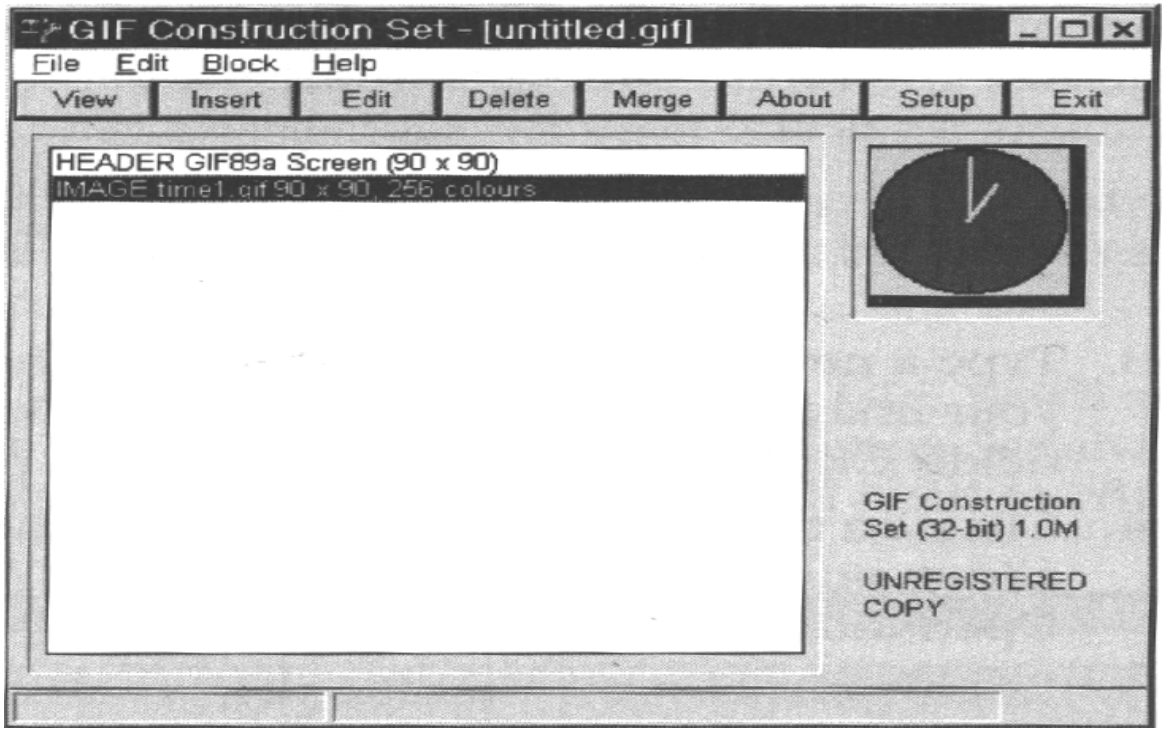
3. 在宽度和高度域中，键入新的大小，使之与图像的大小相匹配。例如，在宽度和高度域中都键入 90，单击 OK 使改变生效并保留下来。

4. 单击 Merge 按钮(或用 Block|Merge 命令)，它使你可以向 GIF 中添加一幅图像，这时将看到标准的 File|Open 对话框。

5. 双击你想在动画中使用的第一个文件，在本例中，双击 Timel.GIF，将会看到 Pallete(调色板)对话框，如下图所示。用于本图形的调色板与 GIF Construction Set 使用的标准调色板并不匹配。



6. 因为这个动画中所有图像使用同一个调色板，所以选择“Use This Image as the global pallete（把这一图像作为全局调色板）”选项，然后单击 OK 就完成了这一工作。GIF Construction Set 将把一个新的图形插入到 GIF 中，如下图所示。



如果单击 Image 项，就会看到图像的实际副本。此时，我们无须关心该软件的这一功能；但是如果想确保你的动画文件的顺序不乱，你迟早就会用到它。

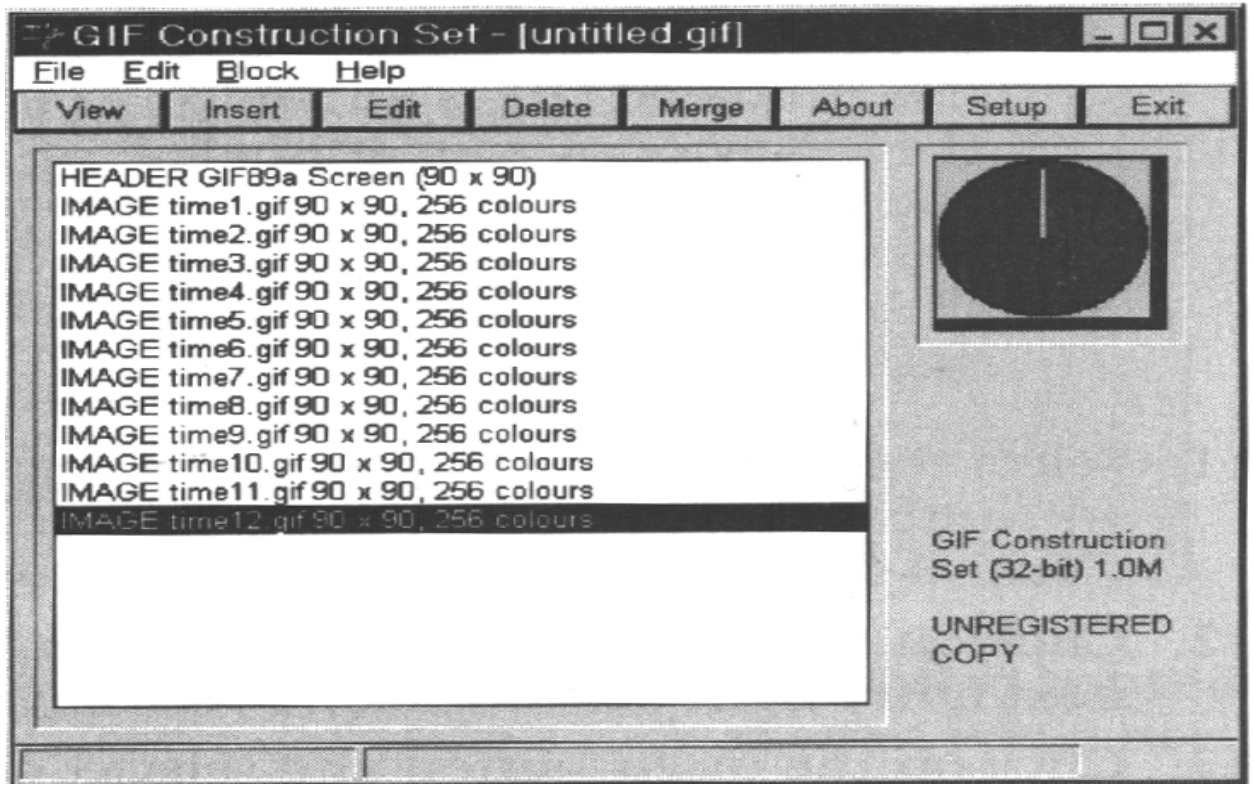
7. 单击 Merge 按钮，将看到与前面介绍过的 File|Open 对话框相同的对话框。

注释 GIF Construction Set 的某些版本在 Merge 按钮的位置上是 Manage 按钮。但你仍能在 Block 菜单中找到 Merge 选项。在需要用 Merge 按钮的地方使用 Block|Merge 命令，即可达到相同目的。

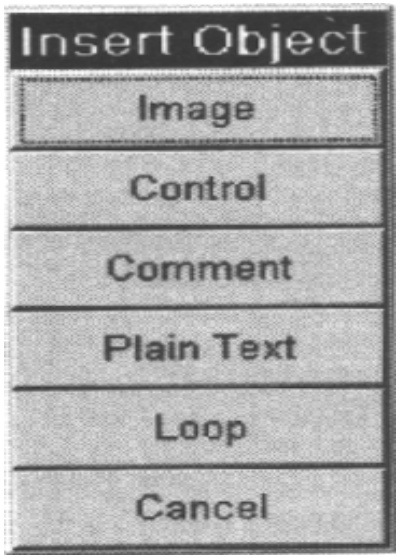
---

8. 顺次选择下一图像，并单击 OK，若 GIF Construction Set 再次询问调色板设置，则单击 OK 。GIF 构造器则自动地把这个图像放入动画序列的下一位置。

9. 对这一动画剩余的 GIF(Time2.GIF，Time3.GIF 等等)，重复 7、8 两步。完成后，对话框如下图所示。请注意这 12 个图像的次序。现在要插入一些控件，以使得图像正确地工作。



10. 单击 Insert 按钮，看到 Insert Object（插入对象）菜单，如下图所示。



这一菜单包含了可以插入到动画 GIF 中去的各种对象。使用最多的是 Loop（循环）和 Control（控制）。

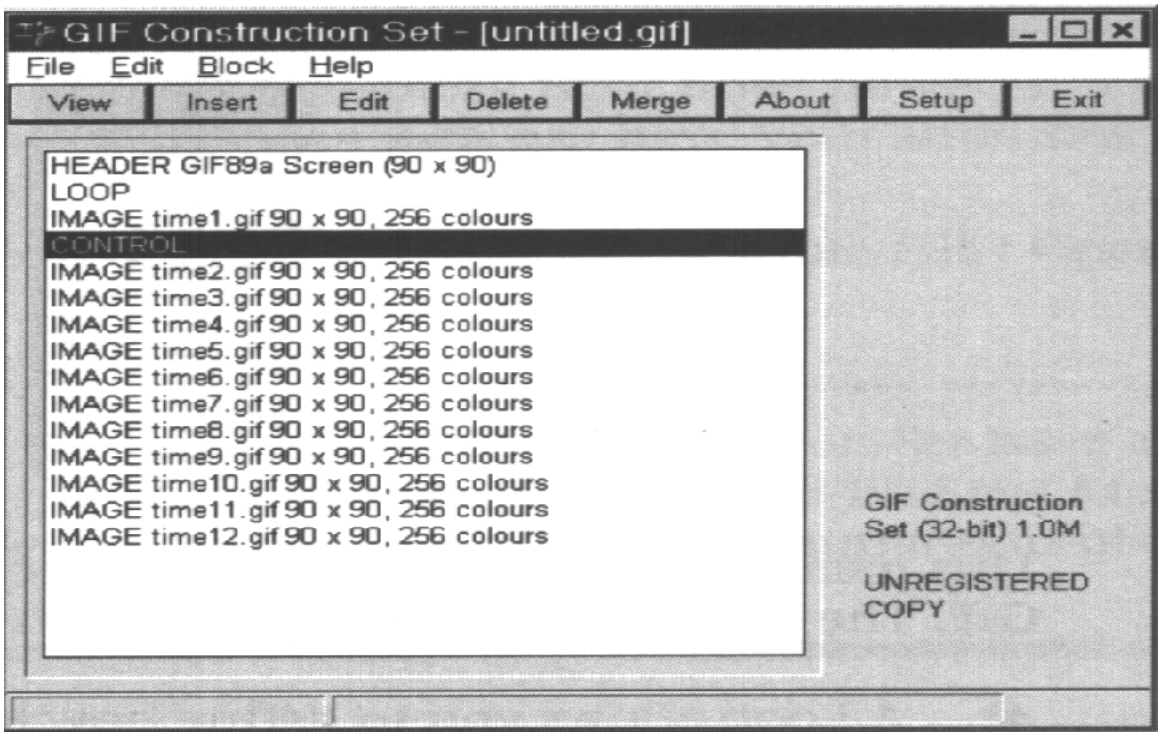
- Loop（循环）定义动画连续播放时用多长时间，Loop 对象的合理使用能制造出一些相当有趣的效果。不幸地是，许多浏览器不理睬这一特定项，所以你应该尽量少用它。（若浏览器忽略 Loop 对象，则它只是简单地保持动画 GIF 永远循环下去。）
- Control（控制）能改变动画 GIF 的行为，例如可以使用一个 Control 设置图片间隔时间。
- Image（图像）项的含义是相当明显的，打算加到动画 GIF 中去的每一幅图片都是一个 Image。使用注释将动画 GIF 的行为作成文档，也是必要的，这在你允许他人使用这个动画 GIF 时显得尤为重要。

◆ Plain Text（明文）很简单，这种文本将在动画中作为一部分显示出来。

11. 在 Insert Object（插入对象）菜单选择 LOOP（循环），GIF Construction Set 将自动地把它放到 Header 项之下。然后，在每两个图片之间放上 Control 对象，这样可以安排动画序列的速度了。

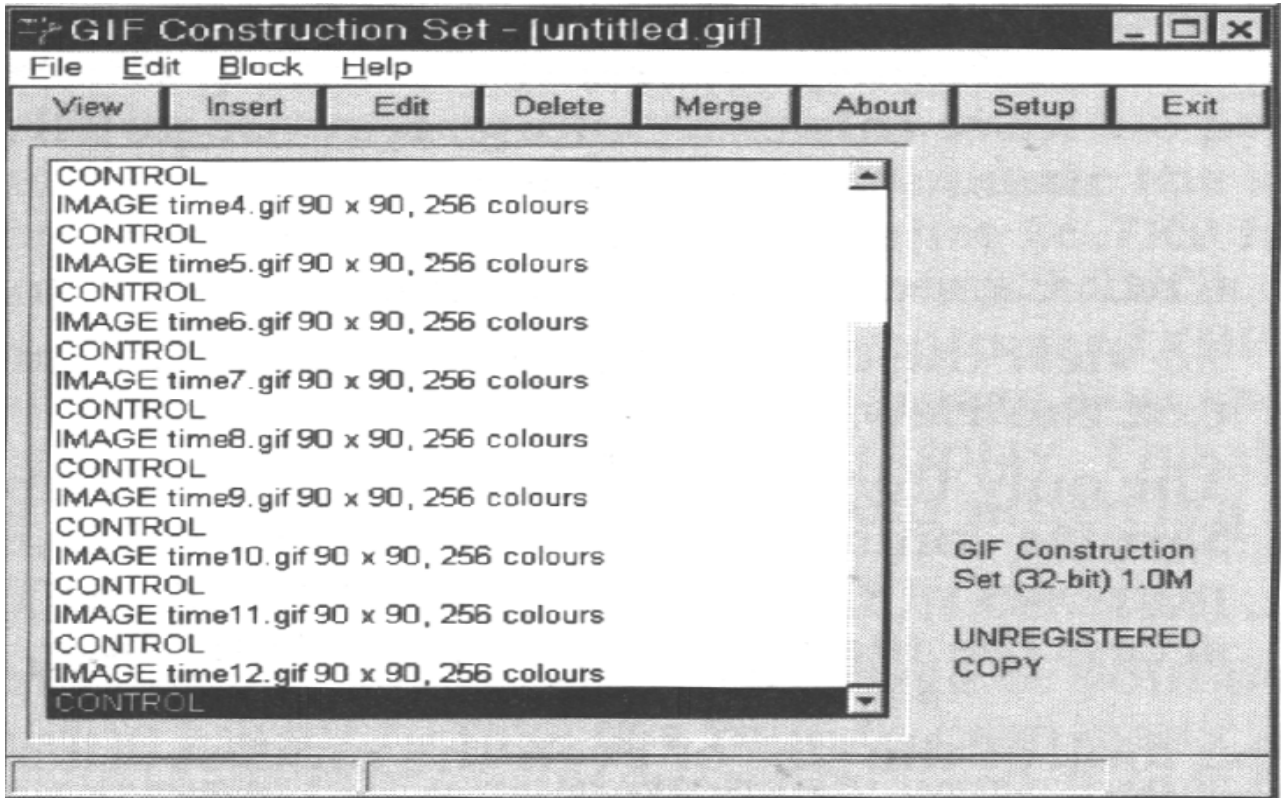
12. 单击第一个 Image（图像）项，正常情况下，GIF Construction Set 把下一个项放在你单击的项之后。

13. 单击 Insert 按钮，从 Insert Object 菜单中选择 Control，则会看到一个 Control 项添加到了列表中，如下图所示。

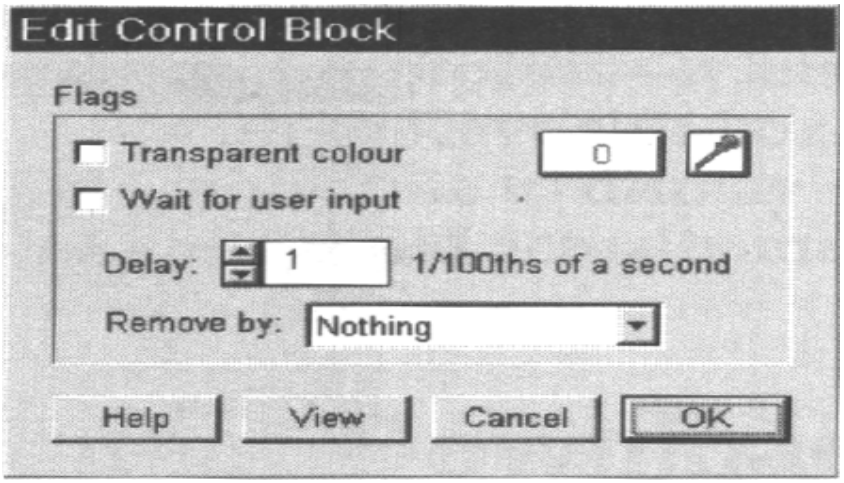


14. 单击下一个 Image 项。

15. 对每一个 Image 项重复 13、14 步。这些都完成后，对话框则如下图所示。（在最后一个 Image 项后，一定也要加入一个 Control 对象，因为动画 GIF 会自动地循环到第一个图像去。）



16. 正常情况下，你可以发现使用图片间的缺省时间间隔时动画显示得非常好。但是，还是让我们看一看改变设置时需要做些什么。双击最后一个 Control 对象项，可以看到 Edit Control Block 对话框，如下图所示：



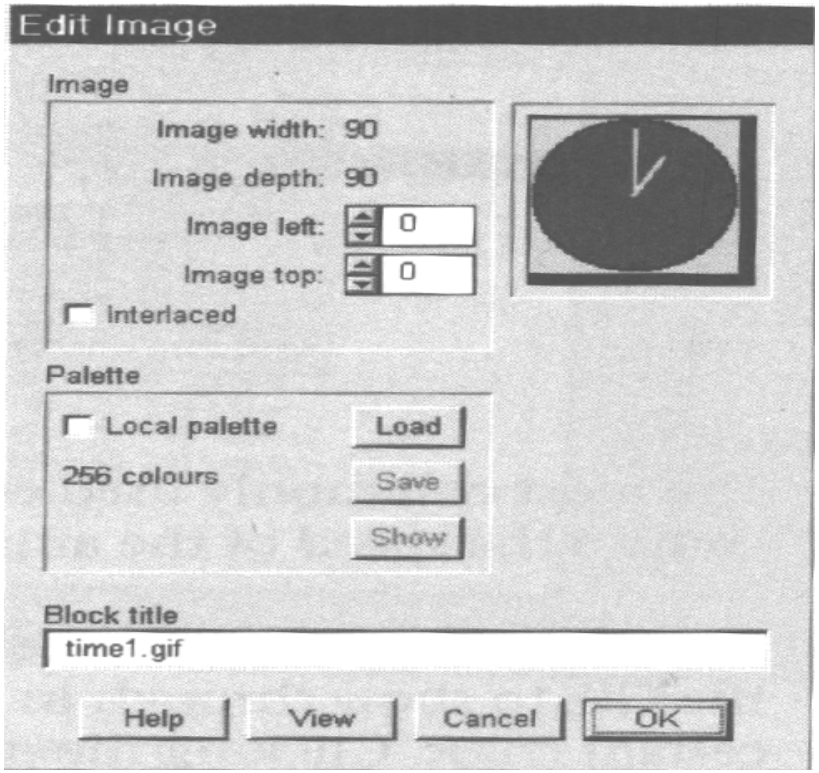
- 最常用的项是 Delay（延迟）域，使用它可以控制动画的速度。
- Transparent Color（透明色）复选框的功能为，使得在 GIF 之下出现的内容，都在使用某种特殊颜色显示的 GIF 所占区域中显示出来。单击该按钮，则可以选择透明色。
- Wait for User Input（等待用户输入）复选框，告诉动画在这点上暂停，等待用户输入某种信息。通常是用户击一个键。
- Remove by（用..... 删除）域允许你确定：一旦浏览器显示这一动画单元之后，对它做些什么。请确保不要随便改动这一项，原因在于改动这一项后，在某些浏览器上会产生不可预测的结果。

17. 单击 Cancel（取消）关闭对话框。
18. 为了观看整个动画，单击 View（观看）按钮。键入 ESC 键，退出浏览区。
19. 唯一还没完成的事是保存动画 GIF 文件。使用 File（文件）|Save（保存）命令来完成即可。可以用任意文件名，但本例把它保存为 TimeIt.GIF，随后的

HTML 页面中我们还要访问它。

如果看一下本例中创建的这种有些简朴的动画文件的大小，就会发现它也已经用了大约 16.7KB 的内存。这差不多是这一页内存预算的三分之一，这时应该考虑一下，动画增加这么多的内存负担是否值得？

幸运的是，还有一种方法处理这一问题，双击任一个 Image（图像）对象项，则可以看到如下图所示的 Edit Image（编辑图像）对话框。



请留意，Image Left（图像左端）和 Image Top（图像顶部）域，它们允许你为动画中每个图像选择一个起点。确实不必要在每次动画单元改变时都显示

整个时钟。必须要做的仅是将前一个时针位置覆盖掉，并用系列中的下一个时针位置代替。对每个图像进行剪裁，使得前一个时针被覆盖且新的时针显示出来，这样能把动画 GIF 的大小减少 75%，换言之，16.7KB 的文件将减少到 4.7KB，需要你做的事仅仅是用画图程序剪裁图像，用刚讲过的程序将图像插入到动画 GIF 文件中，然后，使用 Image Top 和 Image Left 域，将裁过的图像正确地放到显示区中去。这个对话框还能为图片选择新的调色板，并给它起个不同的名称。

技巧 除非动画 GIF 非常大，或者你的下载预算非常小，否则不必花时间对每个图像都单独地进行剪裁。例如，这里的样本 GIF，只要它是页面上唯一的图像，就会工作得很好。

InterLaced 复选框也非常有趣。如果这个 GIF 设计成静态图像，就应该选中这个复选框。交错图像每次显示一行，这也是在大多数站点下载大图形时看到的效果。使用交错图形，可以在下载过程中给用户一点可视反馈，让它们知道没有死机。对于动画 GIF，则不要选这个框，因为显示交错图像用的时间更多。本例如果选择了交错复选框，实际上会使动画看起来好象是指针在跳跃。

8.7 使用 DIANTZ.EXE 创建组件下载(CAB)文件

前面谈到过将 ActiveX 控件放入 CAB 文件后，可以节省用户的时间。这种文件格式使得在一次下载中，可以传送多个文件并同时完成文件压缩。创建 CAB

文件不一定十分困难，但是，在投入产品环境中使用之前，一定要彻底测试它。

注释 你可能会看到，提到 DIANTZ.EXE 文件时，根据你是在看什么资源，还会用另外两个名称中的一个来称呼它。Visual Basic 用户把这个实用程序称之为 MAKECAB.EXE。ActiveX SDK 的一些版本则有一个名为 DIAMOND.EXE 的实用程序，无论你使用的实用程序名称是什么，目的却都是创建 CAB 文件，你能用它们来发布你的应用程序。

第一步是确定要把哪些文件发送给用户，例如，用户在使用其它程序时，也许拥有了全部 MFC（Microsoft 基础类库）文件。这样就不要让他们浪费时间一遍又一遍地下载这些文件。在大多数情况下，要把自己的 Web 站点上的文件都限制为独一无二的。

第二步，创建 INF 文件。包括这个文件的理由有下列几条。第一条是它能包含安装指令，从而使用户的工作更容易，而且也会减少你的支持电话。另一条是它包含了一些公用文件，ActiveX 控件需要这些文件，但不一定要作为 CAB 文件的一部分下载。你能够预先把下载这些文件的指令包括进去，当用户确实需要这些文件时，就可以从你的 Internet 站点上下载它们。浏览器中的 Internet 组件下载服务部分，并不懂得所有的 INF 文件规范，它只能使用其标准元素的一个子集，表 8.2 列出了其中可用的元素以及运用它们时的使用顺序。

表 8.2 Internet 组件下载服务使用的 INF 文件格式

项目	描述
[Add.Code]	
<Filename1>= <Section-Name1> <Filename2>= <Section-Name2> <Filename n>= <Section-Name n>	[ADD.Code]节提供了要安装文件的完整列表。它并非包含了 INF 文件中的全部文件，原因在于你并不希望把 INF 文件安装成最小模式。项目的<Section-Name>部分告诉因特网组件下载服务到什么地方寻找包含在特定文件中的安装指令。（参见下节）
[Section-Name1]	
Key1=Value1 Key2=Value2 Keyn=Valuen	文件中的每个特定节都由一个或多个关键字组成，这些关键字就象以前在 INI 文件中使用过的关键字一样，跟随其后的元素则指定了 Internet 组件下载服务能理解的关键字的值
File=[<URL>| ThisCAB]	这个关键字指出，是从 Internet 的一个具体地址下载文件，还是从这个 CAB 文件中提取文件。对于 ActiveX 控件需要但未包括到 CAB 文件中去的那些文件，使用这一关键字，可以定义出它们的位置。通常情况下，这些附加支持文件存放在 Internet 服务器中

续表

File Version=<a>,<b>,<c>,<d>	这个关键字指明了某文件可接受的最小版本号。如果不指定一个值，则 Internet 组件下载服务认为任意版本都可以。每个字母指定版本的一个层次，所以若控件版本为 1.0，则使用 Version=1,0,0,0
File-[Mac|Win32]-[x86|PPC|Mips|Alpha]=[<URL>|IGNORE]	这个关键字让你对各种平台所需的支持加以区分。先定义操作系统，然后是 CPU 类型。<URL>参数则可以指定一个文件位置。IGNORE 参数则告诉 Internet 组件下载服务，该文件对于指定的平台并不是必要的
CLSID=[<Class ID>]	这个关键字让你定义文件的类标识符。你没有必要实际创建 CLSID，因为 Visual C++根据 Microsoft 研究的一个复杂公式自动地完成这一步工作。从两个地方可以得到这个值：C++源代码和 Windows 注册表。为了在 Windows 注册表中找到它。只要打开注册表编辑器，用 Edit|Find 命令，在 HKEY_CLASSES_ROOT|CLSID 关键字下，寻找 OCX 文件名即可。第二章示例中的 CLSID 的注册项是{D8D77E03-712A-11CE-8C70-00006E3127B7}。C++源代码的位置在控制文件中，例如，第二章中的源代码文件是 OCXEXMPLCTL.CPP。可以用这样的调用找到标识符： ////////////////////// 初始化类工厂和指南

续表

	IMPLEMENT_OLECREATE_EX(COCXEXMPLCtrl,"OCXEXMPL.OCXEXMP-LCtrl.1",0xd8d77e03,0x712a,0x11cf,0x8c,0x70,0,0,0x6e,0x31,0x27,0xb7) ////////////////////////////////////// ActiveX 控件的 CLSID 值不会改变（除非对于像名称这样非常基本的东西进行改变——代码改变无效）,所以只需找到这个值一次即可

创建 INF 相当简便。确实需要做的是，你要想清楚需要包括什么，以及它们的位置。程序列表 8.13 展示的是安装两个文件的典型 INF 文件。第一个文件 OCXEXMPL.OCX，实际存放在 CAB 中。用户还需要 MFC40.DLL，但有可能已经安装到客户机中了。例子中包括了在 Internet 服务器上找到这个文件的一个站点，只有在客户机上没有安装这个所需文件时，这个站点才会有用。创建了文件列表和 INF 文件后，还要创建 Diamond Directive File (DDF)，DIANTZ.EXE 创建 CAB 文件时要用到它。这个文件的格式不难理解，程序列表 8.14 是创建 DDF 的一个典型例子。

程序列表 8.13

```
;OCXEXMPL.OCX 的 INF File
[Add.Code]
OCXEXMPL.OCX=OCXEXMPL.OCX
```

```
MFC40.DLL=MFC40.DLL

[OCXEXMPL.OCX]
File=thiscab
Clsid=[D8D77E03-712A-11CF-8C70-00006E3127B7]
FileVersion=1,0,0,0

[MCF40.DLL]
File=http://aux/files/MFC40.DLL
FileVersion=4.0.0.5
```

程序列表 8.14

```
;Diamond 命令文件，生成变量类型错误的完全列表。
.OPTION EXPLICIT
;定义 CAB 文件的名称
.SET CABINETNAMETEMPLATE=Activex.CAB
;创建一个 cabinet
.SET CABINET=ON
;压缩这个文件
.SET COMPRESS=ON
;文件列表
```

Install.INF
File1.OCX
file2.OCX
ù

一旦做到这一步，创建CAB文件就不困难了，只需使用下列命令行即可：

DIANTAZ /F MY.DDF

第 9 章　JavaScript 概览

如果你认为在 Web 站点上使用 ActiveX 控件可以摆脱脚本，那么，就请忘掉这个不切实际的想法吧。事实上，ActiveX 控件比起其它种类的 Web 接口元素来说，更加依赖于脚本。理论上，你可以在没有脚本的情况下把一个 ActiveX 控件放进一个页面，它能在那儿挺美观地呆着，但也仅此而已。事实上，在第 10 章中，我们将会看到，即使你还没有为一个 ActiveX 控件写出脚本，它也可以执行某些功能，这些功能仅限于使用 Visual C++ 写出这个控件时，指定它要完成的事。换言之，控件在没有脚本时是内部激活的，而脚本却可以让它与外部世界打交道。

幸运的是，除了使用 ActiveX 控件需要脚本语言外，使用脚本语言还有一些好处。对你键入的数据，找出其不完整或有点错这样一些小事，要靠 CGI 脚本在主机（Internet 服务器）上完成，这时，你是不是曾经坐在那里傻等？ 许多人都曾有过这样的经历。由于 VBScript 和 JavaScript 都是在主机（这时是用户的机器）上执行，可以发现，用户一打入信息，马上就能把有效性验证工作完成。这节约了网络带宽，并且做到了即时向用户反馈。任何使用户高兴的事，也要受限于节约你一些时间这件事。

注　使用脚本可以节约网络带宽，而且可以使页面上的 ActiveX 控件与外

部世界打交道。

显然，如果你打算要花些时间使用脚本语言，那么就值得去了解一下这个舞台的主角。某些脚本语言比另一些要更流行一些，现在，VBScript 和 JavaScript 看起来都引起了大部分媒体的注意（Javascript 的 Microsoft 版本称为 JScript，我在全书中都使用 JavaScript 表示任何形式的脚本语言），JavaScript 可以在 Internet Explorer 和 Netscape Navigator 上工作，而 VBScript 只能在 Internet Explorer 上工作。（用于 Netscape Navigator 的 NCompass ScriptActive 插件仅能完成让这种浏览器支持 VBScript 这类事情。）ActiveX Control Pad 目前支持 VBScript 和 JavaScript。可以发现，市场上可以见到的大部分其它工具也支持它们。

注释 尽管你可能要选择 VBScript 或 JavaScript 与 ActiveX 控件一起使用，但本章将集中讨论 JavaScript。做出这种抉择的理由有三：第一，JavaScript 比 VBScript 在语法上更接近 Visual C++，所以，你的学习曲线的陡峭度就小（即学习起来难度小）；第二，JavaScript 天生就被两大主要浏览器：Internet Explorer 和 Netscape Navigator 支持；第三，最新的 Microsof 的出版物及商业杂志之类的其它来源已经在说明 VBScript 正在失宠，而 JavaScript 却受到喜爱。

我们再看一下 ActiveX Scripting，它是一种 OLE 通信技术，而不是脚本语言。一旦 ActiveX Scripting 可以使用后，就能得到它是涉及到什么技术的概要，从而使你准备使用 ActiveX Scripting。总起来说，可以发现 ActiveX Scripting 依赖于目前的 OLE 技术，它加入了一些新的界面元素和一些从其它地方借鉴来

的功能。这项技术的实际面目还有点复杂。（无疑我们将会看到这些新的界面元素。）

技巧 如果计划采用 ActiveX 控件或本章中讨论的任一种脚本技术、耗费大量时间去开发 Web 页的话，你肯定需要得到一种比 ActiveX Control Pad 功能更强的工具。诸如 Microsoft Visual InterDev 之类的产品，则使你不必对基本的 HTML 标记了解太多，而且很容易地创建 Web 页。Visual InterDev 还使你可以在 Web 页内使用 ActiveX 控件，ActiveX Control Pad 对于实验用途是最有用的，使你可以创建包含脚本的小小的 Web 测试页面。

本章用大部分时间来考察 ActiveX 控件和脚本怎样配合起来进行工作。让我们从一个简单例子开始，这个例子使用了 ActiveX Control Pad 提供的一些控件。例如，也许你已经注意到，我们不能通过把下拉列表框粘贴到页面上并通过访问属性对话框来定义下拉列表框的列表项。道理很简单：需要使用脚本在运行时添加列表项。本章各节均讨论 JavaScript，但是，也可以使用 VBScript 完成类似的任务。

走过打基础阶段后，我们将用些时间讨论一些中等难度的脚本例子。例如，讨论一下怎样使用脚本与已建立的控件打交道。虽然本章缺少一个实际上更复杂的控件来演示深入的概念，但我们依然研究了足够多的基本概念，说明了如何在复杂情况下使用脚本的方法。

9.1 ActiveX Scripting 概要

目前，ActiveX Scripting 还不能像我们谈论过的 VBScript 和 JavaScript 语言那样，成为一种语言（宏或其它形式的语言，本章下一节将讨论 JavaScript 语言）。最好将它定义为客户机与服务器之间的一种通信方法。实质上，它是 Microsoft 为 OLE 自动化起的新名称。说 OLE 自动化与 ActiveX Scripting 完全相同也许不够准确，因为 Microsoft 没有把 ActiveX Scripting 的能力停留在应用级别上。（大多数 Microsoft 产品和诸如 CorelDRAW 之类的一些其它产品，在应用级别上支持 OLE 自动化!）ActiveX Scripting 还包括了 Internet。

可移植性问题 本书前面的例子中，MFC AppWizard（应用程序向导）提供了 OLE 自动化复选框，这一点也许你已注意到。这一复选框，像诸如 CorelDRAW 之类的产品一样，向你的应用程序添加相同种类的脚本能力。换言之，可以在本地的机器或网络中使用脚本能力。但是，这一能力仍然不是完整的 ActiveX Scripting，因为它不包括 Internet。你仍然需要手工地向 Visual C++应用程序添加这种功能。因为 ActiveX Scripting 标准还没有最后定下来，所以你看到的仍然是进展中的东西。

既然 ActiveX Scripting 不是语言，而是通信规范，那么，脚本语言，包括语法，这一类问题就留给了脚本供货商来提供。这一标准的目的在于定义一种方法，使脚本主机可以调用各种脚本引擎，并且用于 OLE 容器中对象间的通信。

从实质上看，脚本更像是可执行代码的大杂烩，它可能含有文本、伪代码，甚至还有原始二进制格式的可执行代码。ActiveX Scripting 的实质是，它是基于 OLE 的、可用于应用程序和 Internet 的通信媒介。

WEB 链接 写作本书时，ActiveX Scripting 说明书仍在变动之中。如果打算用 ActiveX Scripting 作为其它技术的替代或补充，也许要用些时间看一下最新动向。进而，本节并不完全涵盖 OLE 主机和脚本引擎需要的所有接口方法的调用语法。欲了解目前 ActiveX Scripting 说明更多的信息，请查看 http://www.microsoft.com/intdev/sdk/docs/olescrpt/ 要查找有关 ActiveX Scripting 主机的信息，下列两个站点不错：http://www.microsoft.com/msdn/sdk/inetsdk/help/wsh/wobj_2.htm 和 http://www.microsoft.com/msdn/sdk/inetsdk/help/compdev/scripting/hosts.htm。如果想多了解一些关于建立一个 ActiveX Scripting 引擎的需求，请查看：http://www.microsoft.com/msdn/sdk/inetsdk/help/compdev/engines/engines.htm。

让我们花点时间定义几个术语。首先，精确地说，什么是脚本主机呢？它是支持脚本引擎的一种应用程序。定义脚本时，是脚本主机接受这个脚本并向引擎发出命令。ActiveX Scripting 主机的最通用的例子是 Internet Explorer 3.0。使用插件可以向 Netscape Navigator 添加这种能力。第 8 章讨论过一种插件

——NCompass ScriptActive。脚本引擎，当然不限于这些产品中的任何一种。你可以在一个定制的浏览器应用程序中建立它，或者在一些应用程序中找到它。例如，很有可能，在不远的将来，Microsoft会使其全部应用程序产品（如 Microsoft Word）都符合这一标准。

一个 ActiveX 脚本引擎是一个对象，这个对象才真正地对脚本进行解释。对精确的语言语法甚至脚本的形式都没有限制。理论上，写 ActiveX 脚本，可使用任意语言，包括 VBScript、Java 和 JavaScript 。和本节后面要讨论的任何一种 OLE 对象一样，一个 ActiveX 脚本引擎提供的是特殊接口。另外，可以使用 OLE 自动化包装工具写出 ActiveX 脚本引擎（Microsoft 计划为 ActiveX Scripting 提供一种特殊形式的 OLE 自动化包装器）。使用包装器格式的好处是，它保持了代码的短小精炼——这对于浏览器和在线脚本引擎都是理想选择。不利之处是，如果直接写出 ActiveX Scripting 接口，你将失去对运行时你可以有的名字空间、持久模型以及其它著作元素的控制。

ActiveX Scripting 要求有四种新的接口元素。可选项 IActiveScriptSite 和可选项 IActiveScriptSiteWindows 接口为主机专用，而 IActiveScript 和可选项 IActiveScriptParse 接口则为脚本引擎专用（若不实现 IActiveScriptParse，则需要实现某种形式的 IPersist 接口）。每个接口元素执行下面列出的一种具体功能：

IactiveScriptSite 这个接口的主要目的是为 ActiveX Scripting 引擎创建一个站点。这与其它的 OLE 实现中的容器的概念(idea)相对应——容器容纳的是诸如 ActiveX 控件之类的东西。因为这是一个容器接口，所以它监测脚本的开始、终止及脚本出错等等一类的事件。

IactiveScriptSiteWindow 任一个提供用户接口的 ActiveX Scripting 主机，

也需要提供本接口（服务器不一定应用它）。实际上，如果需要用于脚本目的的窗口，则也需要实现这一接口。IActiveScriptSiteWindow 提供了两种主要方法（尽管可以提供其它所需的方法）：GetWindow()用于创建窗口，EnableModeless( ) 则设置窗口的模式(modal)条件。GetWindow()的功能类似于 OLE 自动化专用的 IOleWindow::GetWindow()函数，而 EnableModeless()的功能则类似于 IOleInPlaceFrame::EnableModeless()函数。

IactiveScript 这一接口使你与脚本本身一起工作，用它可得到脚本的站点、启动和终止脚本、使用脚本项目、关闭脚本、或者建立线程状态与参数。

IactiveScriptParse 这一接口从脚本主机接受脚本，它使你能创建一个新脚本、 向一个现存脚本中添加原始脚本片段或对一个现存脚本进行分析。

Visual C++提供了一个方便的实用程序，称为 OLE/COM Object Viewer（OLE/COM 对象浏览器），使用它可以更详细地看到这些接口（Microsoft 在其 Visual C++的最新版本中将名称缩写为 OLE View（OLE 浏览器））。本书中我们将多次用到这个实用程序，如果你还没有安装的话，应该安装上它。让我们先看一看 ActiveX Scripting issue 的主机端。和以前提到的一样，Visual C++仍然提供一个 OLE 自动化接口，这个接口不提供与 Internet 连接的功能。接着请打开 OLE/COM Object Viewer，就会看到一叠文件夹。打开 Document Objects（文档对象）文件夹，再打开 XYZ Single Document 文件夹，就会看到接口列表，与图 9.1 类似，这些接口是 MFC 实现的，建立应用程序时可使用它们。

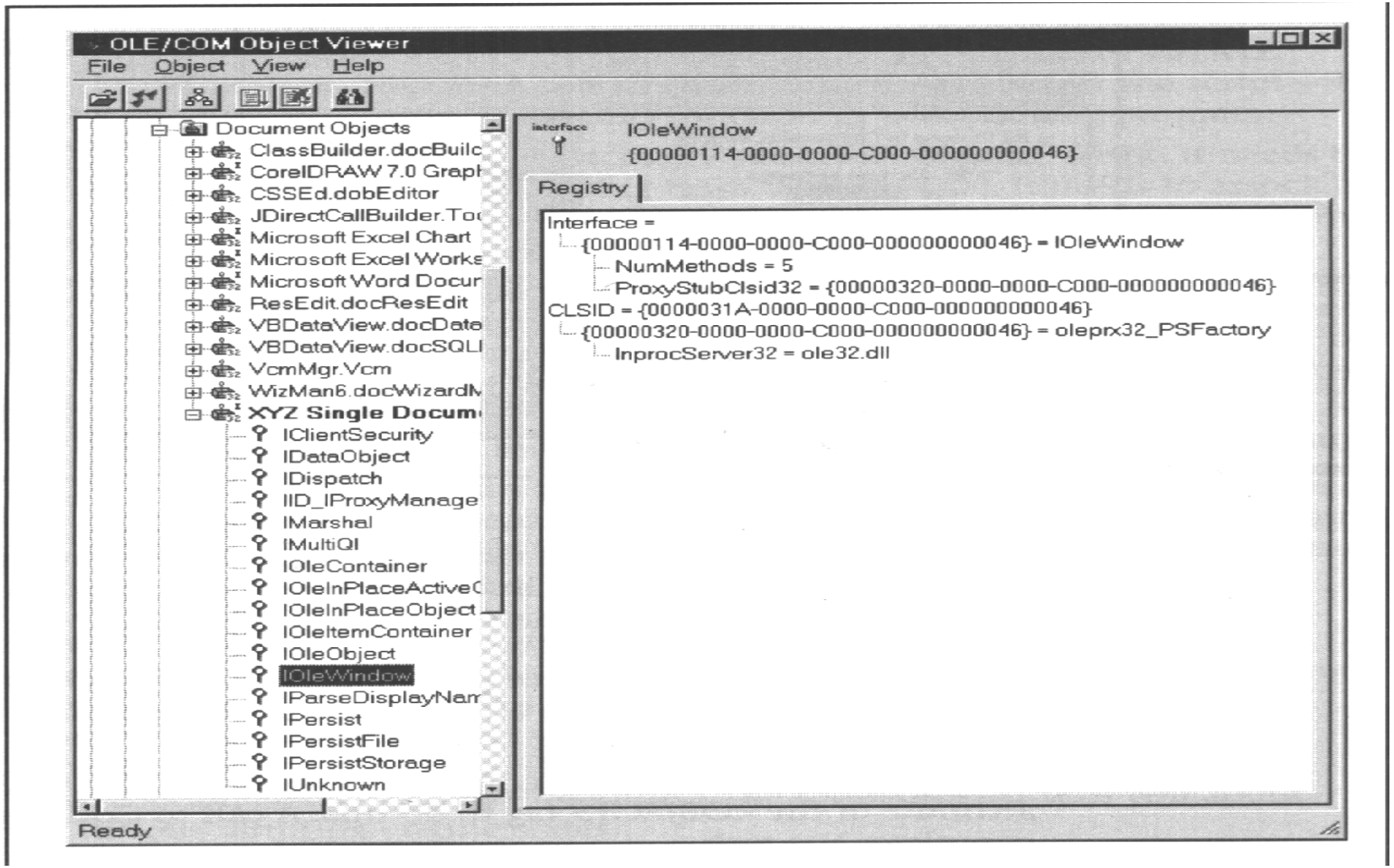


图 9.1　在 OLE View 中查看 ActiveX 接口

请注意在左边的窗格中的 IOle Window 接口已被选中。看一下右边的窗格，可以看到这个接口提供五种方法。进一步研究表明，这些方法之一是 GetWindow()。前面提到过，没有 IActiveScriptSiteWindow 接口，而 IOleWindow 接口却存在，这意味着你的应用程序支持 OLE 自动化，而不支持 ActiveX Scripting。现在，如果你看一下这个列表中的每个其它应用程序，（如果有的话）包括 Office97，就会发现，没有一个应用程序是支持 ActiveX Scripting 的。

警告 如果在 OLE/COM Object Viewer 中创建了应用程序或控件的实例，而后没有不释放相应实例，那么就会发生奇怪的事情。例如，你的机器会意外地死机！每当你查看一个应用程序或控件支持的接口时，就不得不创建一个实例来完成这个操作。通过查看应用程序的名称，也能够知道是否存在一个对象的实例。OLE/COM Object Viewer 用粗体显示打开的对象。为了释放你创建的实例，用右键单击对象名（比如是 XYZ Single Document），然后在上下文相关菜单中选择 Release Instance（释放实例）即可完成。幸运的是，OLE/COM Object Viewer 在你离开该工具时总是会关闭对象的实例，但是如果在查看会话期间机器的内存开始不够用时，就应该关闭对象的实例。应该记住，每创建一个实例，就要占用一些内存。

客户端对 ActiveX Scripting 也是支持的，至少在一种初步形式上是这样。释放 XYZ Single Document 的实例，然后关闭 Document Object 文件夹。接着打开 ActiveX Scripting Engine 文件夹，可以看到两个项，一个是 JavaScript，另一个是 VBScript。这两个项都是 Internet Explorer 支持的，但我们期望看到 Netscape

Navigator 也支持它们。打开 JScript Language(JavaScript)项，可以看到如图 9.2 所示的典型的接口列表（你的列表也许会稍有不同）。

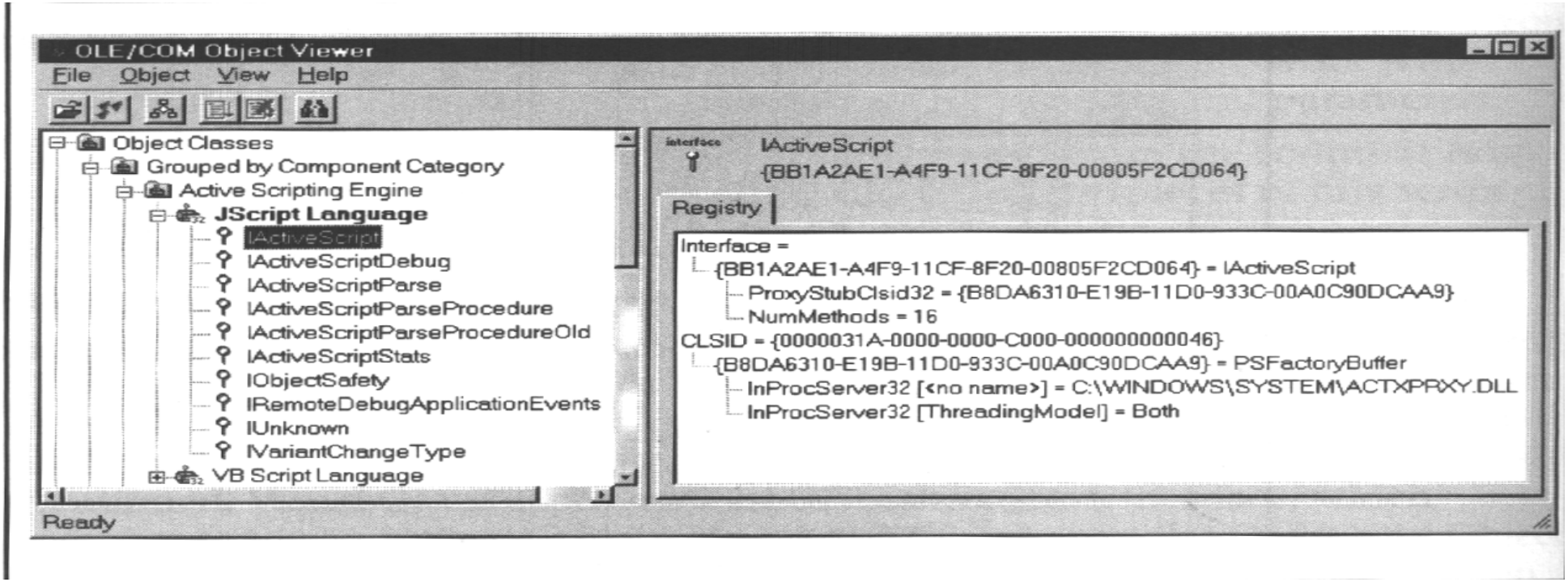


图 9.2　在 OLE 视图中查看 JavaScript

请留意被选中的项 IActiveScript，它是我前面描述过的第一个客户机端接口，在它下面，可以看到 IactiveScript-Parse 项，它是我前面描述的第二个客户机端接口。这两个元素的存在，意味着 Internet Explorer 当前把 ActiveX Scripting 做为一个客户机来支持它。但是，在 OLE/COM Object Viewer 的这一部分中却连一个相联系的服务器项都找不到。

让我们对于 ActiveX Scripting 的主机与客户机的关系再多说几句。一个 ActiveX Scripting 会话期，要涉及到三个不同的元素：主机、引擎以及包含代码和控件的窗口（页）。在这三个元素之间建立通信的过程有 8 个步骤：

1. 主机根据需要装入文档或工程。这不是 ActiveX Scripting 专用任务，因为任意环境都需要执行这一步。

2. 主机创建一个脚本引擎。通常，它要使用<OBJECT>标记中的 PROGID 属性，并通过调用 CoCreatInstance()来完成这个任务。

3. 然后装入脚本。主机完成这件事的方式有几种。如果脚本存贮为驻留(Persisted)数据，那么，主机就使用引擎的 IPersist::Load()把它再次装入内存。如果主机需要装入一个新脚本，那么它就调用 IActiveScriptParse::InitNew()或者 IPersist::InitNew()创建一个空脚本。把脚本当成文本来维护的主机，然后调用 IActiveScriptParse::ParseScriptText()把脚本装入引擎（使用其它格式维护脚本的主机则不得不设计自己的装入方法）。

4. 主机把脚本装入后，必须用实体填入引擎的名字空间。实体可以是窗体、 页或其它文档。主机使用 IActiveScript::AddNameItem()方法完成这一任务。显然，它不必装入存贮为脚本驻留内存一部分的那些实体。低级项（比如 ActiveX 控件）需要不同的装入技术，主机使用 ITypeInfo 和 IDispatch 接口来完成这项工作。

5. 现在，ActiveX 脚本引擎已取得了它需要运行脚本所需的一切。主机发出一个 IActiveScript::SetScriptState(SCRIPTSTATE_CONNECTED)调用，这就是让脚本引擎启动脚本。它相当于标准 C 程序中调用 Main()函数。

6. 在脚本引擎真正开始运行脚本之前，它还要先干几件家务活。例如，需要为每个顶级项联系上一个符号。脚本引擎干这件事时使用的是IActiveScriptSite::GetItemInfo()方法。
7. 脚本引擎真正运行脚本之前的最后一个步骤是，在脚本元素与与对象相关联的事件之间建立适当连接。例如，如果存在一个脚本项，它处理一个按钮的onClick（单击）事件，那么脚本引擎就需要在运行脚本之前建立这个连接。脚本引擎使用IConnectionPoint接口完成这一工作。
8. 最后，脚本引擎开始运行脚本，运行脚本时，脚本引擎需要访问与HTML页上或其它文档中的对象相关联的方法和属性。它使用IDispatch::Invoke()或其它标准的OLE捆绑方法来干这件事。

现在，你对脚本如何工作这一点已有了相当好的理解。让我们快速地看一下，脚本引擎运行时作出反应的几种方式。脚本引擎不只是具有简单的两种操作状态：ON/OFF，实际上它具有好几种状态，使用这些状态，可以确定需要干些什么才能使它运行。（使用IActiveScript::GetScriptState()方法可以得到当前状态。）以下列表列出了脚本引擎的各种状态：

- Uninitialized（未初始化） 这表明脚本引擎还没有准备好，通常，主机在脚本引擎离开这一状态前，要装入运行脚本所需的脚本和实体。
- Initialized（已初始化） 脚本已装入（通常用IPersist），运行站点已设定（使用IActiveScriptSite），但脚本引擎还没有真正地与一个主机相联系。脚本引擎处于这种状态时，不能运行任何代码。一旦脚本引擎进入已启动状态，则由IActiveScriptParse::ParseScriptText()执行的代码将开始运行。

- Started（已启动） 这是一种过渡状态。从这种状态使用 IDispatch 就能运行代码了。所有代码与对象一起装入。但是，脚本引擎尚未在脚本元素与对象事件之间建立连接。如果脚本运行到需要执行事件处理的地方，它就被阻塞（停止），直至脚本引擎完成所需的初始化。
- Connected（已连接） 脚本引擎做好了去执行需要由它来执行的每项任务的准备时，就是进入了这种状态。
- Disconnected（连接暂停） 脚本引擎进入这一状态后，实际上脚本已暂停。脚本仍处于装入状态，所有连接依然存在，但脚本引擎却不准备回答主机的请求。脚本引擎可以在不丢失脚本当前运行位置的情况下，离开这一状态并返回已连接状态。
- Closed（关闭） 在这种状态中的脚本引擎，不再回答调用。你发出 IActiveScript::Close()调用后，就进入这一状态。

9.2 Java Scripting 基础知识

在深入讨论之前，我们先澄清一件事。Java 和 JavaScript 是两种完全不同的实体。如果你看到的是一种功能强大的编程环境，那么，你看到的语言是 Java(或 Mictosoft 版的 Java，称为 J++)，而不是 JavaScript。反之，如果你看到的是与 VBScript 一样外型的宏语言（尽管功能不同），而且是面向 C 的，那么这是 JavaScript。另外，Java 是 Sun 公司的发明，而 JavaScript 却是由 Netscape 的杰作。肯定是 Netscape 的市场人员造成了这种容易混淆的概念——对两种产品赋

予了相同的名称。JavaScript 原来的名称是 LiveScript。

可移植性 JavaScript 一直是浏览器大战中的受害者之一。Internet Explorer 和 Netscape Navigator 在各自浏览器中，使用类似的但又不完全兼容的 JavaScript 版本。有一个小故事说，Netscape 为它的 JavaScript 版本推出公开说明书时慢了一步，而 Microsoft 就抓住时机设计了自己 JavaScript 版本。有传言说，两个公司正在制定 JavaScript 标准的联合版本——对于这一点，我不会屏住呼吸等待它。同时，使用 JavaScript 时，你不得不面对兼容性问题。好消息是，通过努力，可以做出能在两个浏览器上都行得通的你自己的一个脚本版本。当然，你肯定需要在两种产品中测试你的脚本，从而确保他们像所预期地那样工作起来。

现在我们已把 Java 和 JavaScript 区分清楚了。我们来看一下语言本身。可以发现，JavaScript 在相当程度上与 VBScript 一样，都是用于相同的目的——作为在一个 HTML 页上用 ActiveX 控件和 Java applet 工作的一种手段。事实上，使用 JavaScript 可以处理任意对象，唯一的制约因素是弄清楚该怎样去做。对于我们在本章后面几节中要看到的许多其它目的的大部分，JavaScript 迟早会有用。JavaScript 提供了一套与 VBScript 截然不同的特色，这些特色能增强 Web 站点的表达能力，弄明白这一点才是重要的。

注 除了名称相似外，JavaScript 和 Java 实际上是两种截然不同的编程环境。

与 VBScript 不同，对于 JavaScript 来说，不存在任何与它直接兼容的版本。

把它与 C(或 Java)相比，不会使你得到更多的信息，因为 JavaScript 设计为面向 Internet 使用的具有一套独立特色的产品。使 JavaScript 在 Internet 出名的主要是它的兼容性。肯定地是，不能把 JavaScript 应用程序搬到 C 编译器去进行编译，并期望它做事。但是，你能指望它比 VBScript 能在更多的原始浏览器环境下工作。本节中我们快速地浏览一下 JavaScript 提供了些什么，然后在本章中我们可以看到它是如何工作的。

可移植性 请记住，JavaScript 使用与 C 类似的语法。这意味着在有些情况下，JavaScript 看起来有点像 C，而在其它方面，JavaScript 将使用通常不能在 C 中使用的新结构。当你在 JavaScript 环境中，能从 C 应用程序中使用一些应用程序逻辑时，不要以为不做任何的修改就能这样做。JavaScript 毕竟与 Visual C++不直接兼容。

JavaScript 处理 HTML 文档时，采用了对象映射法(approach)。它建立了对象的一种层次结构，从窗口开始，文档在窗口中，窗体在文档中，最后，对象在窗体中。使用 JavaScript，可以通过定义路径的方法与每个对象通信。例如，如果想对某个窗体做点事，你要从窗口名中分离出窗体名：Window1.Form1。注意窗口名与窗体名用一个圆点分开，许多程序员把这叫做点语法引用。

JavaScript 存在一些与 C 完全不同的内容。首先，JavaScript 是一种松散型语言。这意味着不一定要对变量进行说明。其次，JavaScript 中存在对象，但不存在类或继承。这意味着不能创建新的对象类型。但是，能够把一个现有对象类型扩充到某种程度。你使用的对象，就是 HTML 文档或窗体中包含的那些对象。例如，甚至是基本的 JavaScript 应用程序也能够访问 Window 对象——每个

HTML 文档必须有这样的对象。在 Window 对象中，可以找到缺省对象，如文档、框架、历史、位置等等。

还可以扩充对象的数目供你使用，但不要用创建新类的通常方法。本章中讨论的方法是使用 ActiveX 控件。使用 ActiveX 控件的扩充方法，依赖于控件作者让你使用什么样的属性与方法。从实质上看，一个 ActiveX 控件，属性出现于属性页中，方法就是暴露出来的事件。（若要对建立控件方面了解更多的信息，请看第 10 章中例子。）重要的是要记住，JavaScript 是基于对象的而不是面向对象的。最后，JavaScript 是动态约束(bound)的，这意味着它在运行时才核查对象。C 和 Java 则都是静态约束的，他们都是在编译时核查对象。

WEB 链接 有许多站点可以学习 JavaScript。首先是找一份语言指南，语言指南最好到 http://home.netscape.com/eng/mozilla/Gold/handbook/javascript/index.html 去找。可以发现，在那里不仅有语言指南，而且还有对各种语言元素的极好讨论。幸好你可以从 http://www.jchelp.com/javahelp/javahelp.htm 得到一份具有 Windows 帮助格式的 JavaScript 文档的教学版。

关于 JavaScript，要记住的最最重要的事情是，和 Java 一样，你在操作系统硬件方面受到严格的限制。其原因是（尽可能多的）防止出现安全缺口。不幸的是，实际能从 JavaScript 得到的保护层次是有限的（参见本节末尾高级技巧“三个普通的 JavaScript 安全缺陷”中的详细讨论）。一些人把这些限制真当成了一回事来对待，但是你能够使用 ActiveX 控件绕开这些限制中的大部分。

因为用户在下载前不得不请求 ActiveX 控件，所以安全性得到了加强。ActiveX 控件要求一定要签名，也有助于改善安全性。但不论你使用何种策略，确实想毁坏你机器的人总有办法去达到他们的目的。

遇到的硬件访问限制之一是，不能写文件。这一规则的唯一例外是能够向 cookie 中写数据。cookie 是保存在浏览器缓冲区目录下的一部分数据。cookie 的目的是，从一个会话转向下一个会话时存贮配置信息。一些程序员也把 cookie 用作其它的小型存贮目的，例如，可以用它来存贮用户到达的最后一个站点的 URL，以便于在下一会话中让用户再返回到那个站点。

JavaScript 支持四种不同的变量类型：数字、字符串、布尔值和一个特殊的 null（空）值。可以使用函数调用将一种变量类型转换成另一种。但是，你能够很容易地混合使用变量类型。例如，可以创建一个名为 nANumber，给它赋一个数值的值。就在代码的下一行，你就能够赋给这个变量一个串值，JavaScript 对此不会抱怨。还能够在同一行代码中混合使用变量类型。例如，nANumber="SomeAnswer Equals "+42 在 JavaScript 中是完全合法的。这一点是下列做法的理由之一：通过使用（引言中解释过的）匈牙利表示法强制使用你输入的某种变量。

WEB 链接 与在其它领域中进行程序设计一样，从 JavaScript 程序员伙伴那里取得反馈信息是相当重要的。如果想得到关于 JavaScript 程序设计技术相当公正的观点，Comp.Lang.JavaScript 是最好的地方之一。如果要知道 Microsoft 的看法，那么就看一下 microsoft.public.frontpage.client 或者 microsoft.public.

activex. programming.scripting.jscript。（其它的 Microsoft 新闻组偶尔也有 JavaScript 方面的消息，但从这两个新闻组可以得出大部分消息。） comp.infosystems.www.authoring.html 偶尔也会提到 JavaScript。

在 JavaScript 中，函数是使用模块化编码的唯一形式。事实上，这也是 JavaScript 实际使用类似 C 的外观的第一点。和大多数程序设计语言一样，函数调用接受参数，也可以有返回值。下面的例子是一个典型的 JavaScript 函数，请注意，它调用了 Document.Write()，这个函数是脚本开始时用得最多的方法之一，使用它不仅可以显示文本，而且如同我们这里所做的一样，也可以写出 HTML 代码。

```
function MyFunction(nSomeValue)
{
    Document.Write("The value received was: ",nSomeValue,".","<BR>")
    Return nSomeValue +1
}
```

大多语言都支持某种条件语句，JavaScript 也不例外。它使用的最简单的条件形式是：(<条件>)?<值 1>:<值 2>，使用它把两个值中的一个赋给变量。例如，lISGreater=(值 1>值 2)?true:false，如果值 1 大于值 2，那么 lIsGreater 就等于 TRUE。还可以像下面例子那样使用 If.....Else 条件语句。JavaScript 目前不支持 Case(switch)语句，这样，当核查许多不同值时，写出其代码就要困难一些。

```
if (Value1>Value2)
{
        Document.Write("Value 1 is greater than Value 2.")
        Return true
}
else
{
        Document.Write("Value 2 is greater than Value 1.")
        Return false
}
```

循环语句是所有程序设计语言都支持的另一种操作。可以发现 JavaScript 支持两种不同的循环结构：for 和 while。下面对每一种循环形式都给出一个例子，请注意他们看起来几乎和 C 中的等价结构一模一样。

```
for (nCounter = 0, nCounter < 10,nCounter++)
{
        Doucument.Write(nCounter)
}

while (nCounter < 10)
{
        Document.Write(nCounter)
```

```
        nCounter++
    }
```

以上给出了JavaScript最基本的概貌。本章中会看到大量的JavaScript代码，所以不用担心你还有很多东西没弄明白。需要记住的主要事情是，JavaScript很像C，但无论你怎样富于想象，它都不是C（现在你该已注意到相当多的区别了）。

高级技巧

三个普通的JavaScript安全缺陷

恐怕你认为脚本是很容易监测的，但是与脚本相关的许多安全问题一直存在，对于这些问题，近几个月来我在贸易新闻与新闻组上已指出过。简单地说，无论VBScript还是JavaScript都不是安全的（尽管做了大量工作以找出JavaScript

存在的安全缺陷），在http://www.osf.org/~loverso/javascript/上能够找到目前已知的JavaScript的安全问题。下面列出当前三种最普通的问题：

- 欺骗用户上载(upload)文件。尽管 JavaScript 不得不请求得到用户的允许来上载一个文件，但黑客可以用各种方式隐藏这一请求。黑客真正需要的全部东西，只不过是具有令人感兴趣的标题的按钮。像 Windows95 使用的那种上载口令字文件，是很容易被破译的——这使得黑客很轻易地就能闯入你的系统。
- 获取文件目录。JavaScript 应用程序不必请求得到任何人的允许就能加载你机器中的目录。事实上，它还能加载你访问过的任意网络机器的目录。一个知道你的硬盘组织方式的黑客，闯入你机器时就要容易得多了。
- 追踪访问过的站点。黑客们通过追踪你访问过的站点，可以对你了解很多。JavaScript 应用程序使得追踪很容易实现，它能追踪你访问的每个 URL 并把地址发向黑客的机器。和文件上载问题一样，用户不得不指出是否允许追踪，但黑客能够把这一允许伪装成其它东西。

9.3 使用独立的脚本

如果你熟悉 C 语言，那么 JavaScript 是相当简单的。由于在本章上节中已初步介绍过 JavaScript 语法，所以现在让我们考察一个脚本样本。这是 JavaScript 的一个典型应用程序，用于计算定货单上用户项的结果，使得服务器不必在

Internet 上来来往往的传送数据。

WEB 链接 写作本书时，Microsoft 还没有在其任一个程序设计产品中包括 ActiveX Control Pad 这个实用程序。可以从几个地方得到这个实用程序，可以由你的 MSDN (Wicrosoft 开发者网络)预订处向你供应这个实用程序，还可以从 http://www.microsoft.com/workshop/author/cpad/下载这个实用程序。还要确信你能找到 Site Builder Workshop 工具库中的其它工具。在 http://www. microsoft. com/workshop/default.asp 能找到关于这个工具的全部细节。在学习本章中其它例子时，最好先得到这个实用程序，因为，这些例子都假定你拥有 ActiveX Control Pad（从第 8 章开始，我们就讨论了这个实用程序，如果你没读第 8 章中“ActiveX 放在什么地方”这一节的话，现在去读一下吧）。

程序列表 9.1 是一个相当简单的 JavaScript（与 HTML 有关联的）脚本，用于显示一份订货单，并自动计算用户应付费用的总额。仔细看一下程序列表，有一两个标记以前还没见过，以及一些虽然你已经见过、但现在才第一次使用的标记。图 9.3 则显示了这张订货单，请注意表格这时是全部都能够看到的。这个特定的窗体也不包括 Submit（提交）按钮。本例的重要信息是如何在本地处理窗体中包含的数据。

程序列表 9.1

```
<HTML>
```

```
<HEAD>
<TITLE>Typical Entry Form</TITLE>

<SCRIPT LANGUAGE="JavaScript">
<!--
function CalculateItem1(nAmount)
{
      window.document.DataEntry.Item1Ex.value = nAmount * window.document.
          DataEntry.Item1.value
      CalculateTotals()
}

function CalculateItem2(nAmount)
{
      window.document.DataEntry.Item2Ex.value = nAmount * window. Document.
          DataEntry.Item2.value
       CalculateTotals()
}
function CalculateItem3(nAmount)
{
      window.document.DataEntry.Item3Ex.value = nAmount * window.document.
```

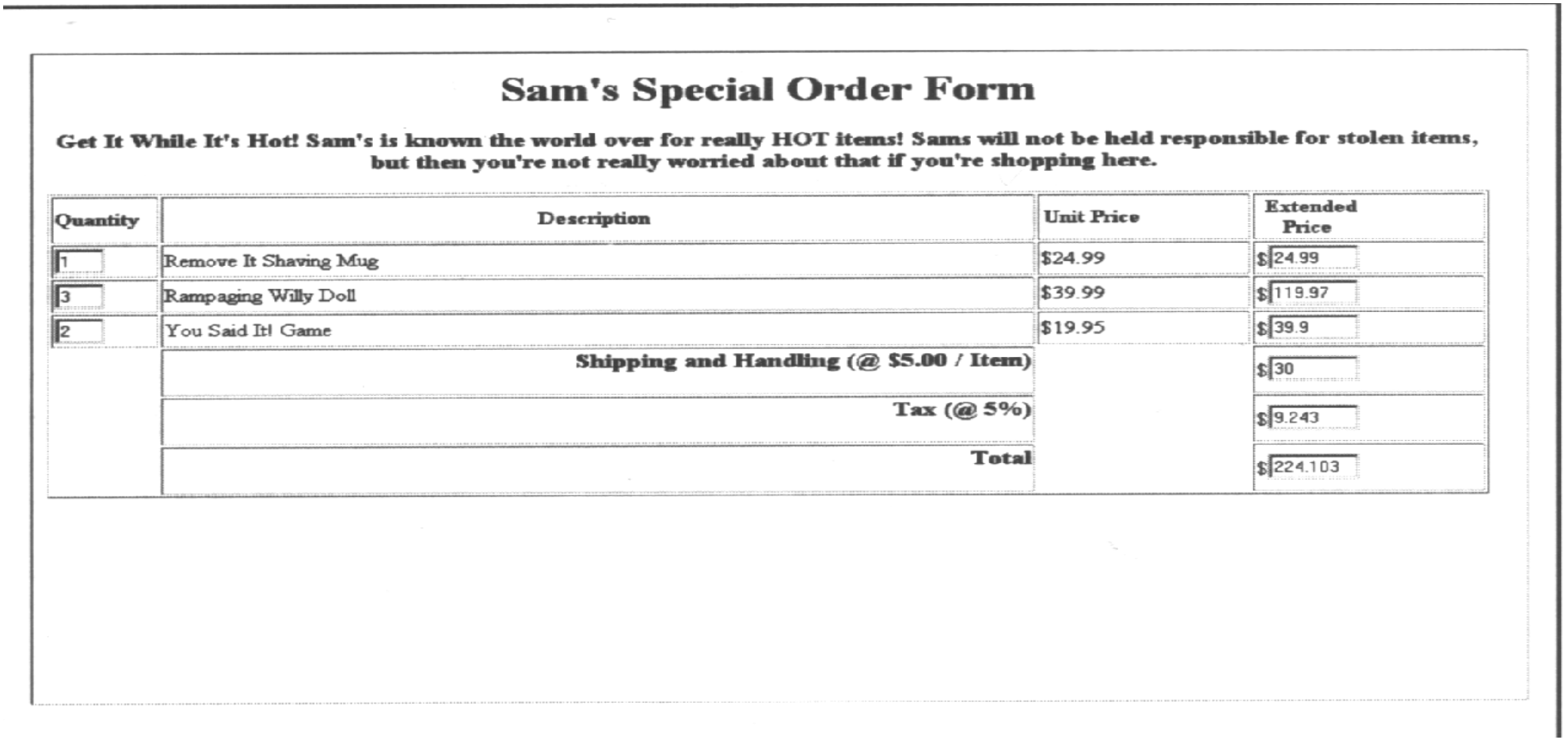
```
        DataEntry.Item3.value
    CalculateTotals()
}

function CalculateTotals()
{
    with (window.document.DataEntry)
    {
    //Calculate the item subtotals.
    nTotalItems = parseInt(Item1.value) + parseInt(Item2.value) + parseInt(Item3.value)
    nTotalAmount = parseFloat(Item1Ex.value) + parseFloat(Item2Ex.value)+
parseFloat(Item3
Ex.value)

    // Calute the total amounts.
    nHandling = nTotalItems * 5
    nTax = nTotalAmount * .05
    nTotal = parseFloat(nTotalAmount) + parseFloat(nHandling) + parseFloat(nTax)

    // Store the data.
    Handling.value = nHandling
```

```
        Tax.value = nTax
        Total.value = nTotal
        }
}
//-->
</SCRIPT>

</HEAD>
<BODY>
<!加入一个标题 .->
<CENTER><H1>Sam's Special Order Form</H1><CENTER>
<H3><MARQUEE WIDTH=50% LOOP=INFINITE>
        Get It While It's Hot! Sam's is known the world over for really HOT
items! Sams will not be held responsible for stolen items, but then you're
not really worried about that if you're shopping here.
</MARQUEE></H3>

<!创建一个窗体 .->
<FORM NAME="DataEntry">
<!-把订货项目表格放入窗体，表格包括四列：quantity(数量), description(说明),
<!-unit price(单价)和 extended price(合计)，用户只需在 quantity(数量)栏中->
```

```
<!-打 入 数 量 即 可 ->
<TABLE WIDTH=100% BORDER=1>
<TR>
      <TH WIDTH=30> Quantity</TH>
      <TH>Description</TH>
      <TH WIDTH=75> Unit Price</TH>
      <TH WIDTH=75> Extened Price</TH>
<TR>
      <TD><INPUT  TYPE= "text"  NAME= "Item1"  VALUE= "0"  SIZE= 3
         ONCHANGE=CalculateItem1(24.99)></TD>
      <TD>Remove It Shaving Mug</TD>
      <TD>$24.99</TD>
      <TD>$<INPUT TYPE="text" NAME="Item1Ex" VALUE="0" SIZE=7></TD>
<TR>
      <TD><INPUT  TYPE= "text"  NAME= "Item2"  VALUE= "0"  SIZE= 3
         ONCHANGE=CalculateItem2(39.99)></TD>
      <TD>Rampaging Willy Doll</TD>
      <TD>$39.99</TD>
      <TD>$<INPUT TYPE="text" NAME="Item2Ex" VALUE="0" SIZE=7></TD>
<TR>
      <TD><INPUT  TYPE= "text"  NAME= "Item3"  VALUE= "0"  SIZE= 3
```

```
        ONCHANGE=CalculateItem3(19.95)></TD>
    <TD>You Said It! Game</TD>
    <TD>$19.95</TD>
    <TD>$<INPUT TYPE="text" NAME="Item3Ex" VALUE="0" SIZE=7></TD>
<TR>
    <TD><INPUT  TYPE= "hidden"></TD>
    <TD ALIGN=Right><H3>Shipping and Handling (@ $5.00/Item)</H3></TD>
    <TD><INPUT  TYPE= "hidden"></TD>
    <TD>$<INPUT TYPE="text" NAME="Handling" VALUE="0" SIZE=7></TD>
<TR>
    <TD><INPUT  TYPE= "hidden"></TD>
    <TD ALIGN=Right><H3>Tax (@ $5%)</H3></TD>
    <TD><INPUT  TYPE= "hidden"></TD>
    <TD>$<INPUT TYPE="text" NAME="Tax" VALUE="0" SIZE=7></TD>
<TR>
    <TD><INPUT  TYPE= "hidden"></TD>
    <TD ALIGN=Right><STRONG><H3> Total</H3></STRONG></TD>
    <TD><INPUT  TYPE= "hidden"></TD>
    <TD>$<INPUT TYPE="text" NAME="Total" VALUE="0" SIZE=7></TD>
</TABLE>
</FORM>
```

</BODY>
</HTML>



Sam's Special Order Form

Get It While It's Hot! Sam's is known the world over for really HOT items! Sams will not be held responsible for stolen items, but then you're not really worried about that if you're shopping here.

Quantity	Description	Unit Price	Extended Price
1	Remove It Shaving Mug	$24.99	$ 24.99
3	Rampaging Willy Doll	$39.99	$ 119.97
2	You Said It! Game	$19.95	$ 39.9
	Shipping and Handling (@ $5.00 / Item)		$ 30
	Tax (@ 5%)		$ 9.243
	Total		$ 224.103

图 9.3　Sam 的订货单窗体：自动计算出用户全部订货的费用总额

让我们从讨论本例中的 HTML 代码开始。首先，它含有一个新标记<MARQUEE>，用于显示滚动的文本。用这一标记还能滚动图像——绝大多数程序员都这样做。这一标记的文本处理能力受到一些限制，加到<MARQUEE>标记中的“花体”文本不在屏幕上出现。开始时文本什么样，则文本一直保持这个样。这意味着在实际显示字幕之前，先要做些格式化（这也正是许多程序员宁可使用图像的原因）。

警告 <MARQUEE>标记不能在 Netscape Navigator 浏览器中工作。如果对标记进行的格式化正确，那么，在 Netscape Navigator 浏览器中看到的是如图 9.3 中所示的居中化了的字幕文本。使用<MARQUEE>标记时总是要使用 LOOP 属性，因为如果不使用 LOOP 属性，支持<MARQUEE>的有些浏览器就会做出奇怪的反应。你第一次实施这个特定标记时，注意收集用户反馈是非常重要的，这是由于在旧式的浏览器上，它非常容易引起麻烦。进行一些小改动，也许就能使你的站点上使用的这个标记在确实支持这个标记的浏览器中工作。

后面一段 HTML 代码是如何使用表格的范例。这里用表格创建了订货单。请注意几个部分都用了 WIDTH 属性，使用这一属性的方式有两种：要么给出占用整个屏幕的百分比，要么给出占用实际屏幕的具体数量。如果使用百分比，当用户改变显示的大小时，占用空间仍保持这一比例。还要注意，四个表头标题中的三个都用了 WIDTH 属性。Description（说明）标题则占用其它三项剩余的那一部分。可以发现，无论用户屏幕的分辨率是多少，也不管他们怎样改变浏览器的窗口大小，这种非常简便的方法都能向用户提供一个具有吸引力的窗

体。

<INPUT>标记中也有几个变化。代码中使用了 SIZE 属性，可以保持输入的文本有具体的大小。还可以看到<INPUT>标记的新类型。“隐藏”控件能够跳过表格中那些凸起部分。再把图 9.3 看一遍，就会注意到。在计算总额时，隐藏控件，通过恢复凸起的外观来保持 Quantity 数量域。

现在来看一下 JavaScript。我们把所有函数都放进了一个 HTML 注释中。理由非常简单，那就是防止旧式的浏览器读窗体时发生问题。还应注意到，HTML 注释标记的第二部分是//-->。如果不用双斜线（双斜线是 JavaScript 和 Visual C++ 中的注释），那么 Netscape Navigator 的 JavaScript 解释程序就试图对这一行做些事——好象它是一行代码；而 Internet Explorer 忽略这一注释行，它意味着能够简单地结束像-->这样的行。像注释标记这样的小事也能产生很大差别：你的代码是仅能在一种浏览器上工作，还是可以在多个浏览器上工作，理解这一点非常重要。

前三个函数相当简单。每一个都是监督着 Quantity 行元素的变化。发生一个变化后，就计算该项订货的合计金额，然后调用函数来计算订单中底部的总计栏。把三个计算组合到一个函数中也是可能的，这样做在一定程度上使程序更快，能减少代码的复杂度，而且也不增加函数调用开销。

可移植性 你会发现，Netscape Navigator 的 JavaScript 版本比 Internet Explorer 所提供的 JavaScript 版本要严格得多。你必须重视浏览器间存在的两个极重要的差别。前面例子的 JavaScript 代码（程序列表 9.1 中 CalculateItem1() 函数）中有一行：

Window.DataEntry.Item1Ex.Value=nAmount * Window. DataEntry. Item1. Value ，它在 Internet Explorer 上工作得很好。如果看一下程序列表 9.1 中的同一行，就会发现两个差别：首先，代码的这个版本不区分大小写。看一下 Window 对象——当你在 Activex Control Pad 的脚本向导中查看时它是小写。另一方面，Netscape 却区分大小写，你绝对要打入小写的“window”才能使脚本工作。还应注意到，这时的 DataEntry 直接跟在 Window 之后，即 Internet Explorer 不要求使用完全限定的对象路径，而 Netscape Navigator 却要求使用完全限定的对象路径。到达 DataEntry 窗体的对象路径的完全限定版本应该是 window.document.DataEntry。在写出代码时，一定要牢记这些差别，否则，就要为找出脚本中的莫名其妙的错误而花费一个星期天了。

有必要对 CalculateTotals()函数看仔细些，因为在使用 JavaScript 窗体时，用了几个开始时你也许未注意到的技巧。使用 With( ) 函数，能够减少键入代码的数量：对于“window.document.DataEntry”，使用了这个函数后，只须打一遍即可，而不必每次用到时都打一遍。

由于 JavaScript 是一种松散类型的语言，本节开始时它直接保持变量实际类型的处理方式，也许会使你感到惊奇。事实上，在有些地方，它不能很好地完成任务，这时就需要做一些额外的工作。看一下第一组计算，可以注意到三个对象的值都被放到 ParseInt()函数调用之中，这个调用告诉 JavaScript 要把串转换成数值。如果不用这个函数，那么就是把三个串联结成一个长串，而不是按

要求将它们加起来。下一行也是这样的问题，这时，因为是三个带有美元符号的值，所以使用了 Parsefloat()函数。这样做虽然不是最好的方案，但它确实有效。

最后应该注意到，CalculateTotal( ) 函数使用了许多中间变量。中间变量帮助 JavaScript 直接保持变量的类型，并且使得代码可读性增强。如果担心中间变量使用的内存太大，少用一两个中间变量也是完全可以的。

技巧 在 Activex Control Pad 中，如果打算使用 JavaScript 语言取代 VBScript 语言，一定要用 Tools|Options|Script 命令进行。当前版本倾向于在每个会话完成后，就丢失这一设置，把 Script Wizard 所作变化逆转过来，需要用很多时间。

9.4 使用 ActiveX 控件

信不信由你，将脚本、HTML 和 ActiveX 控件组合起来使用，实际地创建一些令人震撼的 Web 页，你已万事俱备。现在需要做的唯一一件事，就是把你所学的全部知识装配起来，再附加一些信息就足够了。

和前面小节中一样，我们来看一个使用 JavaScript 的例子。差别是，现在我们把一切东西跟对象装配在了一起，因为我们在本书余下部分要研究更复杂的问题，所以这里的例子的目的非常简单。非常重要的是从现在开始要把基本知识牢记在心。

本节中，我们会看到使用 ActiveX Control Pad 将工作变得更容易完成的一种方式（尽管开头看起来是更复杂了）。我们全部使用 ActiveX 控件（从现在开始不再是<INPUT>标记）来创建一个半功能性的窗体。本节中假定使用 ActiveX Control Pad，但是也包含了全部所需的源代码，这样做潜在地使你能够用其它方法创建它。要做的第一件事是新创建另一个 HTML 页和一个使用 ActiveX Control Pad 的 HTML 框架。New（新建）按钮允许创建文档。

保存空的 HTML 框架。使用 Edit（编辑）|Insert HTML Layout（插入 HTML 框架）命令将框架插入到 HTML 页中，再添加一个标题以解释这一文档是干什么用的。HTML 页这时就完成了，在这个例子中不用再做什么事了。程序列表 9.2 列出了这个 HTML 页的代码。

程序列表 9.2

```
<HTML>
<HEAD>
<TITLE>Using ActiveX Control with JavaScript and HTML Layout</TITLE>
</HEAD>
<BODY>

<!-加入标题->
<CENTER><H1>Using JavaScript with ActiveX Controls On an HTML Layout</H1>

<!-插入框架->
```

```
<OBJECT CLASSID="CLSID:812AE312-8B8E-11CF-93C8-00AA00C08FDF"
      ID="Layout1_alx" STYLE="LEFT:0;TOP:0">
      <PARAM NAME="ALXPATH" REF VALUE="Layout1.alx">
</OBJECT>

</BODY>
</HTML>
```

注释 这个例子的 Web 页仅适用于 Internet Explorer。需要使用类似 NCompass ScriptActive 的插件使这个页在 Netscape Navigator 上工作。本书后面我们讨论这个过程。现在，仅适用于 Internet Explorer 的示例让代码简单些，从而使你能够清楚地看到它们的工作原理。

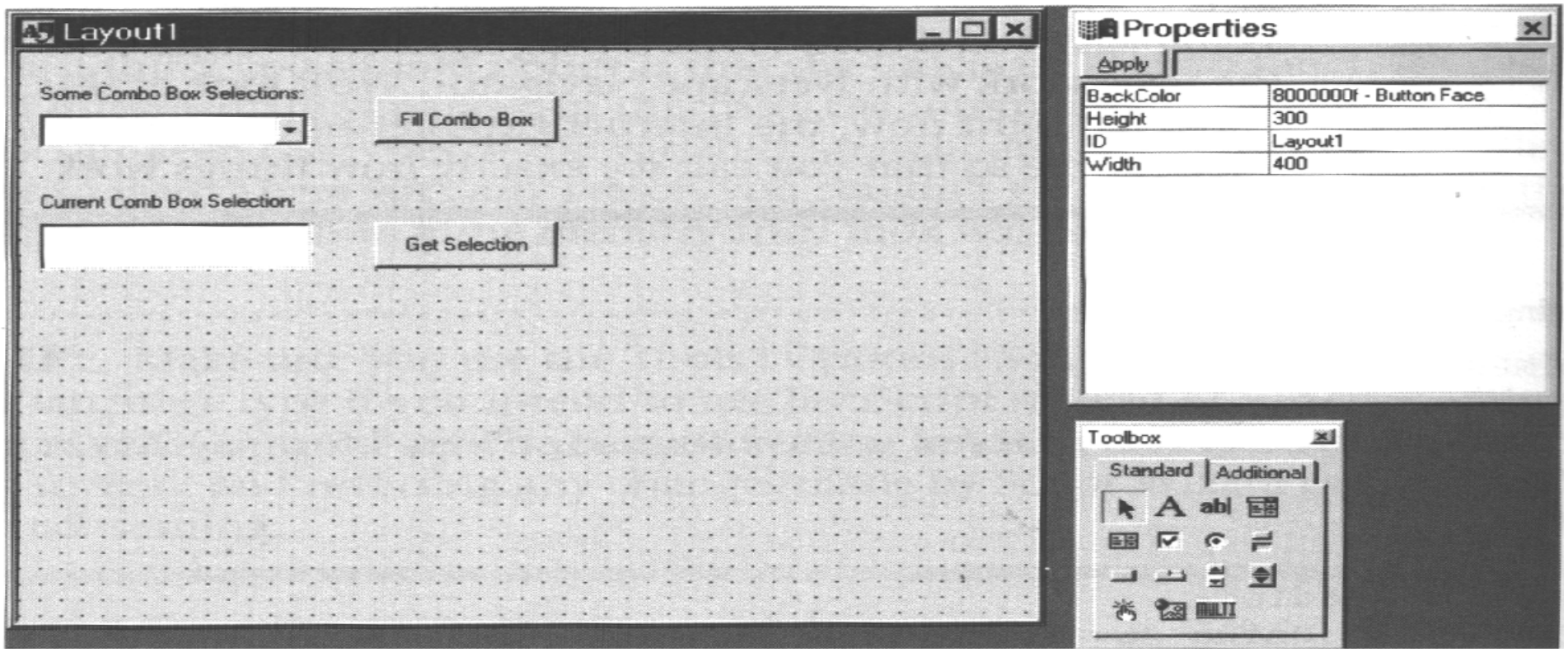


图 9.4　使用 HTML 框架控件能够将 ActiveX 控件精确地定位到 Web 页上

现在来干些容易干的事。首先看一下 HTML 框架。图 9.4 显示了本例中使用的一组 ActiveX 控件。你要完成的工作就是在 Toolbox（工具箱）中单击一个控件，然后把它放到画布上。（如果喜欢的话，ActiveX Control Pad 还允许使用拖放操作向画布添加控件。）控件放到画布后，可以把它拖曳到任何需要的位置，然后再改变其大小，使之与同一区域中的其它控件相匹配。功能框架上与图 9.4 中所示的有些不一样，但该框架的设计目的是向你展示组合使用 ActiveX

控件和 JavaScript 的具体特色。请留意从 Toolbox 中取用的控件都是 ActiveX 控件。

该给这个示例增加一些功能了。和 VBScript 一样，JavaScript 也要求在向按钮或其它控件派发函数时，要执行一些特殊的函数名称格式化。还有一种容易的方法来产生函数名，这就是使用 Script Wizard（脚本向导）来产生函数名。使用 Tools（工具）|Script Wizard（脚本向导）命令，打开 Script Wizard（脚本向导），然后为一个按钮选择 Click（单击）事件，在 Script Wizard（脚本向导）对话框底部就会看到 List View（列表视图）和 Code View（代码视图）这两个单选按钮。单击 Code View 这个单选按钮，就会看到对话框底部发生变化，如图 9.5 所示。看一下对话框底部三分之一处的灰色条，注意到它清楚地说明了如何格式化你的 JavaScript 函数调用。若选择了 VBScript 代替 JavaScript 作为语言选择，那么，将显示出过程调用的 VBScript 版本。

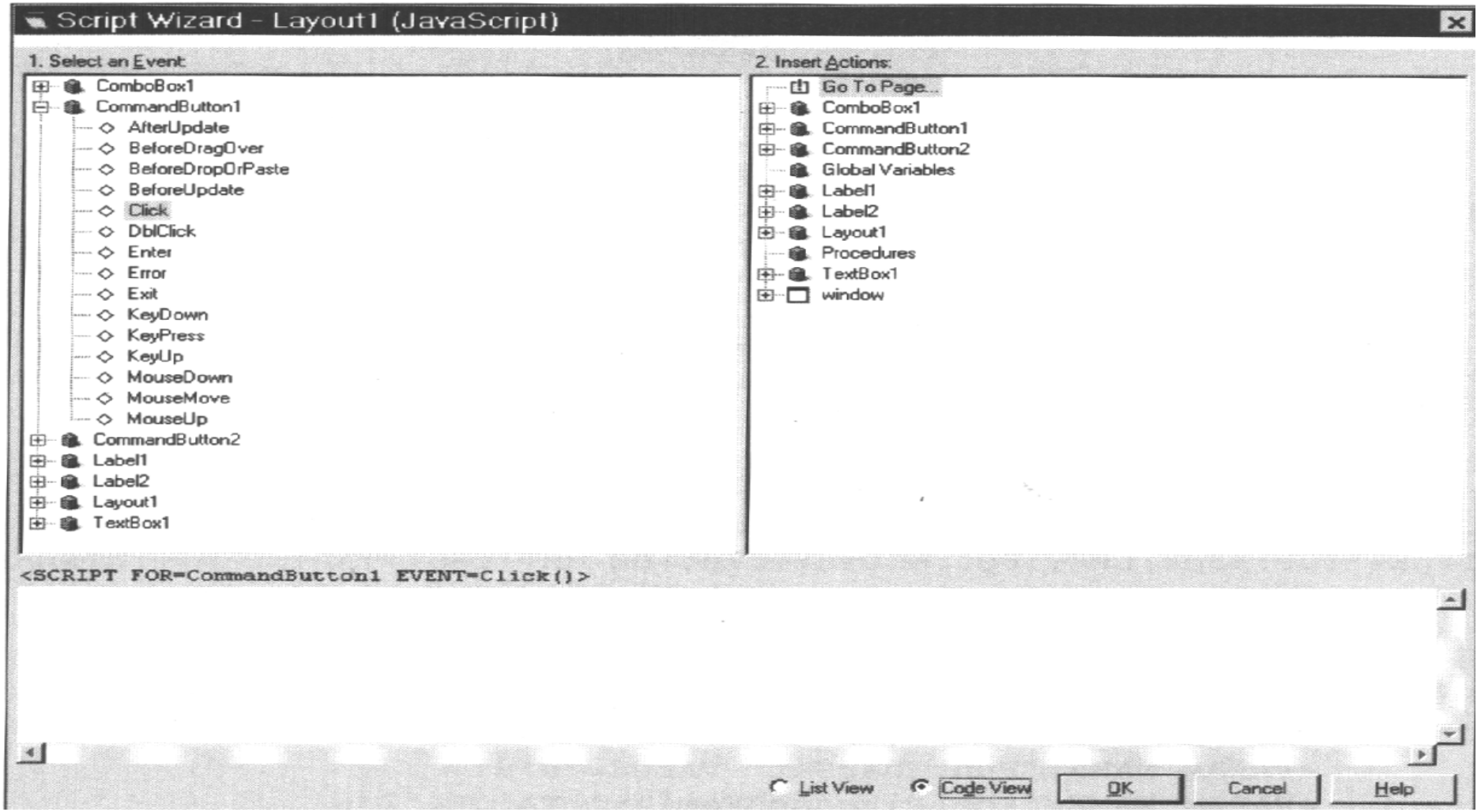


图 9.5　Script Wizard 帮助你处理一个对象所需的过程或函数的名称

现在到了添加一些 JavaScript 使控件生效的时候了。关闭 Script Wizard（脚本向导）（如果还没关闭的话），用右键单击框架，从关联菜单中选择 View Source

Code（查看源代码）。ActiveX Control Pad 可能要求你先保存框架，那么，按要求进行保存。然后就会看到一个 NotePad 或 WritePad（这取决于框架有多大）文本文件。不要破坏已存在的<OBJECT>标记，添加如程序列表 9.3 所示的代码（程序列表也列出了< OBJECT>标记。因为也许你会手工创建这个文件），本章最后的图 9.6 显示的是 HTML、脚本和框架页组合起来在浏览器所看到的样子。

程序列表 9.3

```
<!-这个脚本用数据填充组合框->
<SCRIPT FOR="CommandButton1" EVENT="Click()">
    for (nCount = 1; nCount <ComboBox1.ListRows; nCount++)
        {
        ComboBox1.AddItem("Item " + nCount)
        }
    ComboBox1.ListIndex = 0;
</SCRIPT>

<!-这个脚本提取当前组合框的值->
<SCRIPT FOR="CommandButton2" EVENT="Click()">
    TextBox1.Text = ComboBox1.Text
</SCRIPT>
<!-这些对象都是机器创建的，不要修改->
```

```
<DIV ID="Layout1" STYLE="LAYOUT:FIXED;WIDTH:400pt;HEIGHT:300pt;">
    <OBJECT ID="Label1"
      CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0"
      CODEBASE="http://aux/controls"
STYLE="TOP:17pt;LEFT:8pt;WIDTH:124pt;HEIGHT:17pt;ZINDEX:0;">
        <PARAM NAME="Caption" VALUE="Some Combo Box Selections:">
        <PARAM NAME="Size" VALUE="4374;600">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="ComboBox1"
      CLASSID="CLSID:8BD21D30-EC42-11CE-9E0D-00AA006002F3"
      CODEBASE="http://aux/controls"
STYLE="TOP:33pt;LEFT:8pt;WIDTH:107pt;HEIGHT:18pt;TABINDEX:1;ZINDEX:1;">
        <PARAM NAME="VariousPropertyBits" VALUE="746604571">
        <PARAM NAME="DisplayStyle" VALUE="3">
        <PARAM NAME="size" VALUE="3775;635">
        <PARAM NAME="MatchEntry" VALUE="1">
        <PARAM NAME="ShowDropButtonWhen" VALUE="2">
        <PARAM NAME="FontCharSet" VALUE="0">
```

```
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="CommandButton1"
      CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"
      CODEBASE="http://aux/controls"
STYLE="TOP:25pt;LEFT:140pt;WIDTH:72pt;HEIGHT:24pt;TABINDEX:2;ZINDEX:2;">
        <PARAM NAME="Caption" VALUE="Fill Combo Box">
        <PARAM NAME="Size" VALUE="2540;847">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="Label2"
      CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0"
STYLE="TOP:74pt;LEFT:8pt;WIDTH:107pt;HEIGHT:17pt;ZINDEX:3;">
        <PARAM NAME="Caption" VALUE="Current Combo Box Selection:">
        <PARAM NAME="Size" VALUE="3775;600">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
```

```
      <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="TextBox1"
     CLASSID="CLSID:8BD21D10-EC42-11CE-9E0D-00AA006002F3"
STYLE="TOP:91pt;LEFT:8pt;WIDTH:107pt;HEIGHT:25pt;TABINDEX:4;ZINDEX:4;">
      <PARAM NAME="VariousPropertyBits" VALUE="746604571">
      <PARAM NAME="Size" VALUE="3775;882">
      <PARAM NAME="FontCharSet" VALUE="0">
      <PARAM NAME="FontPitchAndFamily" VALUE="2">
      <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="CommandButton2"
     CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"
STYLE="TOP:91pt;LEFT:140pt;WIDTH:72pt;HEIGHT:24pt;TABINDEX:5;ZINDEX:5;">
      <PARAM NAME="Caption" VALUE="Get Selection">
      <PARAM NAME="Size" VALUE="2540;847">
      <PARAM NAME="FontCharSet" VALUE="0">
      <PARAM NAME="FontPitchAndFamily" VALUE="2">
      <PARAM NAME="ParagraphAlign" VALUE="3">
      <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
```

</DIV>

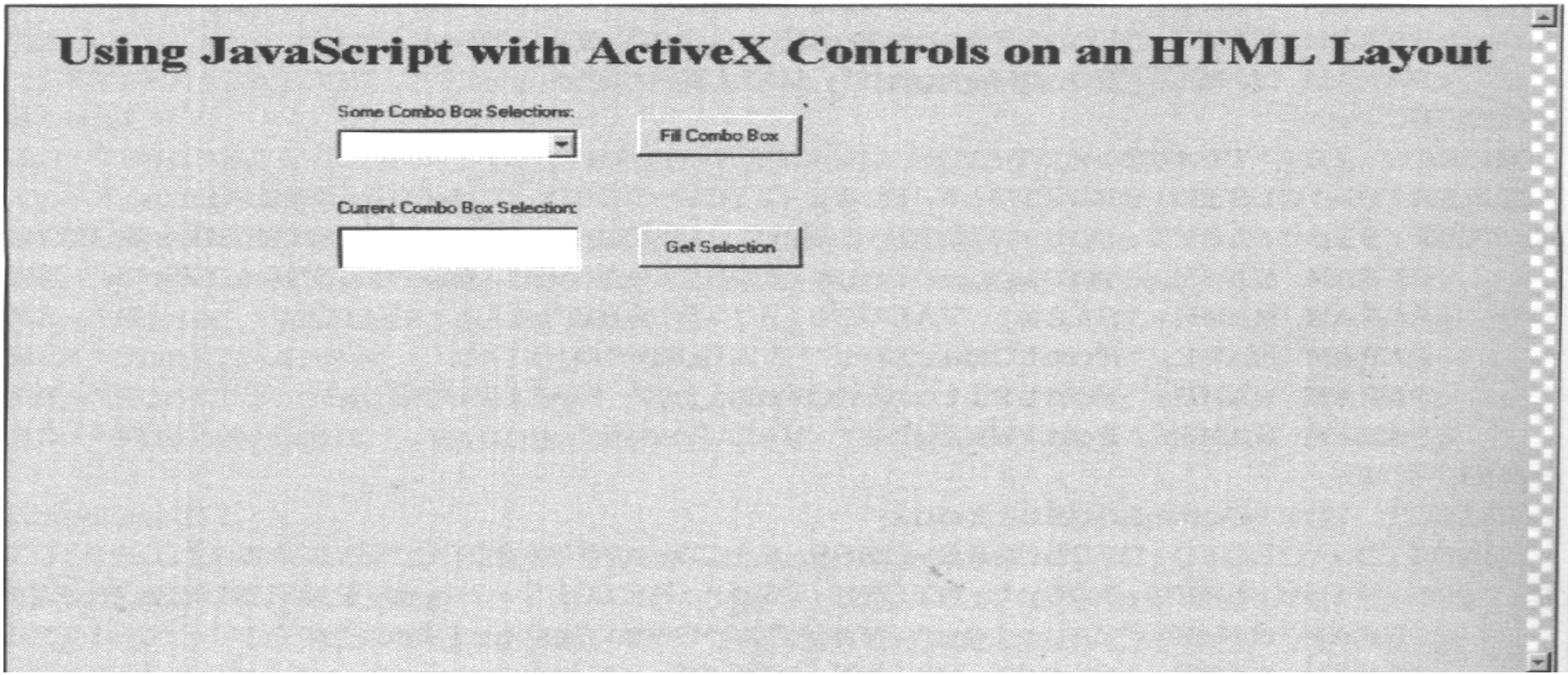


图 9.6　HTML 框架在浏览器中是不可见的，但使用它的效果是可见的

程序列表 9.3 中的两个脚本是很简单的，第一个脚本说明怎样填充一个组合框。真正要你做的，只不过是使用 AddItem（添加项）方法告诉组合框你想添加些什么。填充了组合框后，使用 ListIndex（列表索引）属性设置缺省的选择。如果不这样做，甚至在你填充之后用户还会继续看到一个空白的组合框。列表索引从 0 开始。

使用 ActiveX 对象的问题之一是，不运行就不能做一些安装工作。例如，运行之前不能用值去填充组合框（用属性对话框试一试）。使用脚本语言操作 ActiveX 的整个想法是给你一种以前从未见到过的灵活选择。

存在多种方法能估计出用户做的选择是什么。若需要一个数值的选择数，那么可以从 ListIndex 属性中得到它。我们这儿需要一个串值，所以最好使用 Text 属性。这正是第二个脚本所完成的工作。它只不过是把 ComboBox1.Text 属性的值拷贝到 TextBox1.Text 属性中去而已。

第 10 章　创建 ActiveX 控件

某些人不愿意学习新东西，特别是学习看起来很难很复杂时更是这样。对许多人说来，OCX 就是这一类的。和在 DOS 环境下写设备驱动程序一样，写 OLE 控件也属于这一类。现在 Microsoft 出于某种考虑，已把 OCX 的新版本的名称改为 ActiveX，许多人会为不想试一试这种技术而找到更多的借口。

注　ActiveX 控件使用了 OCX 扩展名，就象他们设计用来代替的 OLE 控件一样。

因为你已经在阅读这一章了，所以我不能不假定你至少已对 ActiveX 控件背后的技术产生了较浓的兴趣。在本章中我们不会做出任何迷惑你们这些小伙伴的事。事实上，我创建的两个程序设计例子（非常有用）都非常普通。我们来看看如何创建一个新型按钮：on/off 开关。

选择这个例子的理由很容易理解，大多数人发现，与一些熟悉的事物打交道，可以缓解学习新东西的压力。另外，这个特定的编码例子实现起来极简单，只需用少量时间就能弄明白它是如何工作的。

注释　本章中的例子是用 Microsoft Visual C++6.0 开发的，使用 Microsoft Visual C++ 5.x 版或 4.x 版都不会有太多麻烦。用 Borland C++或比 Microsoft C++4.x 还老的版本编译写出的代码，不经修改是不能工

作的。创建能在每个产品下都能工作的万能例子，已经证明是行不通的。无论你决定使用哪一种程序设计语言产品，本章的例子都向你提供了关于开发 OCX 的有价值的提示或技巧点拨。

一旦写出了例子的第一个版本，应该学几种方法去测试这个新的 ActiveX 控件，看其是否完全安全（“安全”一词在这里意味着在一个产品环境中能使用这个控件，而不是对其安全漏洞进行核查，尽管你肯定想这样干）。有些人想一下子就把控件放入最近的 HTML 页或产品应用程序中，这实际上不是干事情的最好办法。通常在内部环境中(in-house) 测试三次，投入实际实现中时再测试一次。因为 Windows Scripting Host（WSH，窗口脚本主机）提供了使用 ActiveX 控件的另外机会，所以也要在这个环境中对它们进行测试。（如果 ActiveX 控件在 Internet 上不成功，你会发现可以限制它们用于公司内部网、WSH 附件或作为一个应用程序组件）。

注 一个 ActiveX 控件的完全测试，意味着对它能否在应用程序、本地 HTML 和远程 HTML 环境中工作。还要测试它在 WSH 环境下能否工作。

本章还要花费时间去讨论两个 API，它们是比照着 ActiveX 控件的参照物 OCX 开始定义 ActiveX 的，可以发现，绝大部分的 OCX 已经工作得很像是 ActiveX 控件了。阻碍你使用它们的唯一障碍是，供货商用什么方式许可使用它们。Internet 的公共特性使得使用你可能拥有的控件的当前版本更加困难。——大多数供货商允许把它们包括到你创建的应用程序中，但不能做为 Web 页的一部分。这些日子里，更多的关注是构造这些控件的方法，它们太大了，以至于不能在合理的时间内把它们下载。必须记住，用户不会仅仅是为了看到一个

花哨的屏幕按钮而乐意于下载一个很大的文件。

WEB 链接 附录 A 列出了也许会考虑添加到工具箱的第三方的 ActiveX 控件。在那里除了可以找到供货商外，还可以看一下由 NCompass 推出的 CaptiveX 控件。CaptiveX 包括 6 个控件，使你可以使用广告牌(billboard)和轻型招贴画(lightpanel)之类的方式显示信息。在 http://www.ncompasslabs.com/captivex/index.htm 可以获得 CaptiveX 控件的完全演示版。与许多演示站点不一样，如果安装了 ScriptActive 插件(可以从这个演示站点下载),就可以使用 Netscape Navigator 在这个站点工作。

本章的最后一节，打算考察一下 RealAudio。我发现，看一下别人在做什么，至少与本章中我们钻研学习程序设计技术一样重要。RealAudio 提供一个控件，允许你在 Internet 上演奏音乐或其它种类的声音。事实上，可以发现这个控件的使用遍及 Internet，原因是它满足了你要知道的两条规则：第一，这个控件很小，用极少的下载时间就可完成下载任务；第二，它提供的是流型声音，这意味着用户可以得到即时反馈，实质上，用户不会注意下载整个声音文件要用多少时间，因为 RealAudio 控件只要得到文件的一小部分就可以开始演奏音乐（或其它声音信息）了。

10.1 理解 ActiveX 控件的一些背景知识

在真正开始创建 ActiveX 控件之前，需考虑三个问题。第一个当然是 ActiveX 控件是什么，这个问题引起了不少混乱，所以对它下个定义是很重要的。请记住这是本书的定义，一旦你开始成为迎合 ActiveX 程序员的新闻组的常客时，就会发现各种其它答案。第二件事是，我们要讨论一下，一旦你创建了一个 ActiveX 控件，你能从它得到些什么，这个讨论中还要看用户期待些什么。最后，我们需要看一下 ActiveX 控件与使用过的其它控件有什么区别，最重要的是 OCX 和 ActiveX 控件有什么区别。下面每一节帮助你探讨刚才提到的一个问题。

什么是 ActiveX

这里讲得是你发现的对 ActiveX 所作的最简单的定义。ActiveX 是 OCX 的高级形式（也许把它看作 OLE 的较简单形式更合适）。但是，这个简单定义不能表达 ActiveX 控件的实质，用户认为 ActiveX 就是 OLE 。对程序员来说，ActiveX 还是一套用于 Internet 的实能技术。它提供了过去没有的一种信息交换方法。

注释 Microsoft 目前正在开发一种新的 Internet 专用的组件技术，称为 COM+ 。尽管许多内部网使用了 ActiveX，而且 ActiveX 依然将用作应用程序组件，但 ActiveX 从未在 Internet 上广泛被采用过。ActiveX

在 Internet 上失败的理由有三条：组件大小，安全考虑以及缺乏对非 Microsoft 浏览器的兼容性。请谨记，你或许应该考虑把 ActiveX 控件作为公司内部使用的解决方案，而不是做为 Internet 的实现技术。

作为真正欣赏 ActiveX 的程序员，就不得不从程序员的角度去看一下 OLE，这就是意味着要看一下 OCX。从用户的角度看，OCX 所做的都是在两个应用程序（或操作系统与一个应用程序）之间交换数据。OCX 比数据交换的内涵要丰富得多。它们包括着一种称之为组件对象模型(COM)的概念。COM 是定义对象模块间标准二进制接口的规范。这个接口定义了函数调用方法、基于标准结构的数据传送技术以及几个标准的函数调用。使用 COM 意味着，你写应用程序模块（如 OCX）时使用何种语言都无关紧要，因为模块接口在二进制层次上是相同的。

注释 写作本书时，Microsoft 正在设计 WebView，它是 Internet Explorer 与 Windows 95 系统的 Explorer 之间的一种集成技术。这一新技术使得对 Web 站点的访问就像对 Explorer 中列出的驱动器或其它资源的访问一样容易。还可以看到你比较熟悉的 URL 已被普通英文名称所代替。你可以在 Windows 98 中看到这一技术的实际运用。

COM 怎样对你写的应用程序产生影响呢?答案是相当复杂的，原因在于使用 COM 的方式众多，倒不是因为技术本身是多么复杂。当用户把一个图形图像对象放入你的应用程序控制的容器中时，你对这个对象知道些什么呢？实际

上你只知道是谁首先创建了它。知道这一信息就允许你在这个应用程序中调用许多服务，这些服务包括显示这个图形或允许用户编辑它。实际中，你在做的是共享应用程序的代码。

程序员也能从使用 COM 中受益。当把一个 OCX 安装到你的程序设计环境中时，实际上你完成了什么呢?大多数情况下，你拥有了一个粘贴于窗体某处的一个新控件。你不必真正地知道控件的内部工作原理，唯一重要的因素是，控件能为你的应用程序做些什么，以及如何与它打交道。你可以使用标准接口来调用安装到你机器中的特定代码模块，这就是 COM。

ActiveX 是这种思想的扩充，仍然使用标准接口，但是，不再简单地从本地机器环境或 LAN/WAN 网的常设连接上调用代码，而是从 Internet 调用它。另外，这一新的代码能采用 applets（包含内容的小程序）或微小应用程序的形式。

ActiveX 将为你做些什么

OCX 能为桌面系统做些什么，ActiveX 就能为 Internet 做同样的事。但是，你将在以前从没想到的地方发现 ActiveX 控件。例如，NetManage,Inc 公司计划创建一个新的电子邮件客户机，称之为 Z-Mail Pro。这一软件包支持 ActiveX 技术的方式是，允许用户直接地在消息查看窗口中交换、创建以及查看 HTML 文档。这就意味着用户拥有了创建动态 Web 页的能力，这种能力正是你今天不得不实际完成的任务。

远程连接也会从 ActiveX 的使用中受益。例如，Proginet Corporation 目前正在开发把主机(Mainframe)数据带到桌面系统去的 ActveX 技术。它的 Fusion

FTMS（文件传输管理系统）可以使用支持 OLE 容器的任何开发语言，比如 Delphi，Visual C++以及 PowerBuilder。从本质上看，你将把一个 ActiveX 控件放入一个窗体，定义到哪里去找数据，然后依靠控件进行连接。Internet 上的远程访问不再要求用户去钻过马戏团中的大铁圈了（即不用在连接中跳来跳去了）。主机上一个特殊的传输服务器通过自动化地处理所有传输请求而完成打包。操作员在客户能访问公司站点之前再也不用人工地为站点下载一个必需的文件了。

甚至于 Microsoft Exchange 也会从 ActiveX 受益。Wang Laboratories,Inc. 和其它公司正在创建将 Exchange 和 ActiveX 混合在一起的新的附件。Wang 的产品是客户机/服务器图像附件，它允许用户扫描、查看、注释、操作或打印图形图像，而不用考虑这些图像位于何处。正是这个产品，还包括了一个层次存贮管理的 ActiveX 控件。将这两种技术组合在一起，使得在大公司中使用和访问图形变得容易了。它们还使得寻找一个所需的图形变得容易了，这最终导致公司节约了存贮空间。

Microsoft 自己也发布了许许多多的 ActiveX 控件。其中有些可以从 Microsoft 的 Internet 站点(http://www. microsoft. com) 免费下载，这些新的控件有用于 PowerPoint 的 Animation Player 以及用于 Access 和 Schedule+ 的 Internet Assistant（Internet 助手）。Access 的 Internet 助手能为要上载的数据库表格拍出快照作为静态图像，每当用户访问该页时，快照都会自动更新。Schedule+ 的 Internet 助手允许把日程表信息加载到 Web 页，因为每当用户访问站点时都会自动更新这些数据，所以再也不用担心在家上班的雇员会缺席会议了。最后，

PowerPoint 的动画放映机则允许你在任何兼容 ActiveX 的浏览器中演示 PowetPoint 的演示文稿。

最后，如果认为 ActiveX 无助于安全性，那还要再推敲一下。这些日子里，许多的新的防火墙和证书策略正在轮番出现，其中一个是 Net2000 。Net2000 是由许多 API 构成的，允许开发人员把 NetWare 内核服务（包括目录、 安全和特许证书）放入自己的应用程序中。你可以在内部网中通过 ActiveX 控件来选择这个 API。这种做法对用户和开发人员有什么帮助呢？这意味着，在适当的程序设计结构下，网络管理员能够追踪整个网络中的特许使用，这种追踪甚至于跨越 Internet 连接。当更多的人开始在家中而不是在办公室中进行计算工作时，这就变成了一个非常重要的课题。

ActiveX 与 OCX 控件

就绝大多数情况来说，ActiveX 控件和 OCX 控件是完全可以互换的。可以看到与 Internet 风马牛不相及的 ActiveX 控件广告，仔细一看，就会发现，在不久以前，这些控件大概还出现在 OCX 控件列表中。当然，你要仔细观察这些控件。即使 ActiveX 控件与 OCX 控件确实共享了同一继承，ActiveX 控件与 OCX 控件也不完全是一回事。请记住，ActiveX 控件是能够在 Internet 上工作的。

Internet 对程序设计环境带来了一些特殊的挑战。你不再能够奢求高速装入，OCX 在因特网环境中使用时，其规模成了一个严重的问题，下载一个 60KB 的 OCX 就对用户的耐心带来了考验，而试图下载一个 200KB 的 OCX 大概会导致用户中断下载。而 ActiveX 控件是 OCX 的小型版本。

注　OCX 和 ActiveX 控件的一个主要差别是 ActiveX 控件通常要小一些（轻一些）。

ActiveX 控件也在受到各种机器专用需求的牵制。在机器上安装 OCX 时，安装程序能对机器进行检测并作出所需的许可。对 ActiveX 控件却不能这样要求。在这里你不能对客户机作任何假定，客户机可以是新的奔腾，也许是昨日的 80386。（如果你的 ActiveX 控件确存在某种平台限制，那么，要么设法解决它，要么肯定每个使用它的人都清楚这一限制。）

你还需要应付 OCX 程序员从未想到的情况。比如，如果浏览器根本不支持 ActiveX 时会发生什么?目前处理这一问题的方法是，浏览器对那些不知作如何处理的 HTML 标记只是简单地忽略掉。在这种情况下很容易应付一个与 ActiveX 不兼容的浏览器——只需打出一条消息，让用户知道他们的浏览器不能在当前页工作并指导用户换一个浏览器就可以了。

基于 MFC 与基于 ATL 的控件

Visual C++为了说明它比其它程序设计语言有完善的开发环境，它包容了两种创建 ActiveX 控件的方法：MFC 和 ATL（Active 模板库）。但是，这一灵活性也带来了其它开发人员没遇到的一些问题。比如，怎样确定创建哪一种类型的控件？一些开发人员通过在他创建的所有控件中使用同一种技术来解决这个问题，但这样做本身就表明，你还没有真正地钻研并利用 Visual C++的全部潜力。

注　用 Visual C++创建 ActiveX 控件有两种方法：ATL 和 MFC。

确实没有任何理由能肯定地说，在给定的情况下，创建控件使用哪种方法

会更好。真正需要做的是弄清楚：你期待控件做些什么，你打算对什么进行开发从而获得所需功能，以及你的专业知识水平。显然，有些情况下，创建控件用这种方法比另一种方法更好，因为，两种方法确实各有利弊。当把一个 ATL ActiveX 控件与一个 MFC ActiveX 控件作对比时，需要考虑一些因素，为了给你一些对这个问题的基本看法，请读完下面列出的各条注意事项，从这里会发现一些对你有用的基本思路，从而帮助你选择一种最好的方案。

开发速度　使用 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）是创建控件的最快方法。这个向导将大部分接口细节管理起来，使你最终得到控件逻辑的一个轮廓。事实上，使用 ATL 方法创建控件通常要使开发人员用两倍（于 MFC 方法）的时间。显然，这还取决于诸如控件复杂度以及程序设计经验之类的因素。

维护　MFC 和 ATL 共有的特色是使得用它们产生的代码，比那些从头开始写出的代码，维护起来要容易得多。但是，当对一个原来的控件作一些与维护有关的变化（比如找出小毛病后修改掉它）时，就要考虑一下了。ATL 控件通常包含不多的由向导生成的“锅炉钢板型”（即难懂的）代码。因为对自己写出的代码会更加熟悉，因而改变起来就更快更容易。另一方面，Microsoft 维护着你使用的 MFC 中所有控件，这就是说，许多小毛病的改正和其它种类的更新，只要简便地再把代码编译一次，就可以自动地完成。在这种情况下作何种选择，确实难以回答，因为没有办法能确切地确定在将来你需要对你的控件作什么样的维护变化。

控件大小　如果要创建最小的 ActiveX 控件，那么采取 ATL 方案最好。ATL 把控件的各方面控制权都留给你，使得你可以方便地手工调节控件的每个

元素，而不至于陷于 MFC 专用代码的泥潭。基于 MFC 的控件不仅规模大，而且在用户使用这一控件之前还必须下载 MFC 库，这个库也有相当多的代码。

学习难度　由于创建 ATL 控件时不得不考虑更多的事情（比如接口），所以创建起来比 MFC 控件就更困难。多数情况下，先用 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）创建几个控件，从中学习创建控件逻辑的规则，这样做是值得的。

技巧　如果你用惯了 Visual C++5.0 的 ATL，那么就尝试一下 6.0 版中新的 ATL COM AppWizard（ATL COM 应用程序生成向导）。新的 ATL COM AppWizard（ATL COM 应用程序生成向导）帮助你以比过去手工技术快得多的速度创建控件。但是，使用 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）创建控件仍然要比使用 ATL COM AppWizard（ATL COM 应用程序生成向导）快得多。

- 兼容性　从定义可以看出，基于 MFC 的 ActiveX 控件要求客户将 MFC 库安装到他们的机器上。但是，有好多种版本的 MFC 库在使用，而它们并不都是相互兼容的。如果用户下载了你的控件及其相关的库，但不能使用某些重要的应用程序，因为新库与它们的应用程序不兼容，这时该怎么办呢？因为 MFC 库存贮于 SYSTEM 目录下，客户机仅能拥有一个版本，到那里找到 MFC 库就能解决兼容性问题了。
- 使用的难易程度　MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）对你写控件代码的能力不作多少要求，所以它把它所拥有的一

切都放进了控件之中。这意味着你最终可以得到十分丰富的接口，但其中有的对你来说或许用不着。这一功能性浪费，导致了控件规模的膨胀，并且使得它难于使用。

- **代码修改的难易程度** 开始时，创建基于 MFC 的控件是非常容易的。因为向导把许多代码都添加好了，所以应用程序开发进程很快，你只需关注把你的控件做得出色的细节。如果创建的控件投入使用后不需作什么修改，选择 MFC 显然是上策。但是，如果决定要更新控件时该怎么办？你的源代码文件包括了大量代码，这些代码不是由程度员写出来的，需要用额外的时间去研究、理解它们。因此，如果计划经常会对代码作修改，那么选择 ATL 将是上策，因为，当控件中由程序员生成的代码出了什么事时，程序员处理起来就容易多了。

10.2 一个基于 MFC 的基本按钮编程示例

在我的机器上，我想不出有哪一个程序不用到一两个按钮。事实上，按钮是为数不多的、可以说是每个应用程序都要用到的 Windows 控件之一。即使是在一个简单对话框中显示信息的应用程序中，通常也要在用完对话框后关闭它时使用 OK 按钮。只要说按钮（或某些程序设计环境中所说的命令按钮）是使你机器正常运转的一个控件就够了。它还是绝对能为你提供全部所需特色，而且程序员改变最多的一个控件。

注 按钮是每个应用程序都使用的一个控件，也是对定制控件的潜力进行

讨论时最好的入口点。

程序员对一个控件的用途作了这么多的强调，但这并没有用去我多少时间，使我不能在本章的这一节中，确定用哪一种控件向你说明如何去对它进行修改。我还想增加一个新特色，你也许在按钮中并没有发现这一特色，而这个特色最终是每个人都必需的。正是为了说明下述特色，我才选择 On/Off 按钮作为本章中控件的基础——就好象在进行应用程序设计中每个人要打开/关闭某个东西一样。将这种特色内置于你使用的控件之中是非常好的。对于从 Visual C++中得到的缺省控件来说，我还增加了其它几个特色。尽管我的这个按钮也许不能将商业化程序中所用控件的花哨特色全部囊括，但它能帮你开始创建你自己的定制控件。

定义工程

从用户和程序员两个角度来说，本章前面讨论过开发 ActiveX 控件的一些规则。现在再从程序员角度深入探讨一下。在开始讨论第一个程序设计例子之前，我们快速地看一下，定义一个具体的 ActiveX 控件必须记住哪些规则。下列的建议可以加深你对创建 ActiveX 控件必须遵守的一些特殊约束的理解。

技巧 尽管使用能生成 OCX 的编译器就能创建 ActiveX 控件，但你会发现使用 Microsoft Visual C++（4.2 版以上）产品将会为你节约许多时间。你在本章还会看到，要使 OCX 像 ActiveX 控件那样正确地工作，哪怕在最好的环境中，也需要一些额外的步骤。使用旧版编译器意味

着你需要用 DOS 命令行工具来处理额外的步骤。Microsoft Visual C++ 编译器的 4.2 及 5.x 都把这些工具作为软件包的一部分，而且将它们的使用自动化了。

- **保持代码小型化** 经验告诉我们，应该保持 ActiveX 控件的体积在 40KB 以下。通常用户不想下载庞大的、仅能用于对页面图形作动画处理的控件。实在没办法时，把一个大的组件分成几个小的功能块来实施。

技巧 还可以像第 8 章讨论过的那样，把控件压缩进 CAB 文件也能够减少下载规模。

- **使用最少的持久数据** 一些 OCX 要求大量的持久数据来完成任务。例如，你可能会把一个电子表格控件粘贴到窗体上，而没想到它所包含的持久数据的数量。ActiveX 控件没有这般奢侈，因为你不能对客户机有太多的假设。持久数据不仅增加装入时间和内存需求，而且扩大了控件自身的规模。
- **将功能特色限制在最低限度** 众多的功能的确能为屏幕演示增光添彩。为本地机器编写 OCX 时，几个额外特色不会增加什么问题。事实上，如果不包括它们就会使多数程序员感到吃惊。但传输时间的确是 Internet 的一大问题。向控件中每增加一个不是真正必需的图形或其它特殊声音效果，都要增加装入时间，并且降低控件的价值。ActiveX 控件甚至不能假定一些特殊效果是否能够在客户机上正确工作（举例来说，客户机可能没有声卡）。
- **功能单一是关键** 建库的原始动机之一是，用容易访问的形式，存贮大量预编译过的函数调用。作为 OCX 的前辈，DLL 存在的理由正在于此。你会

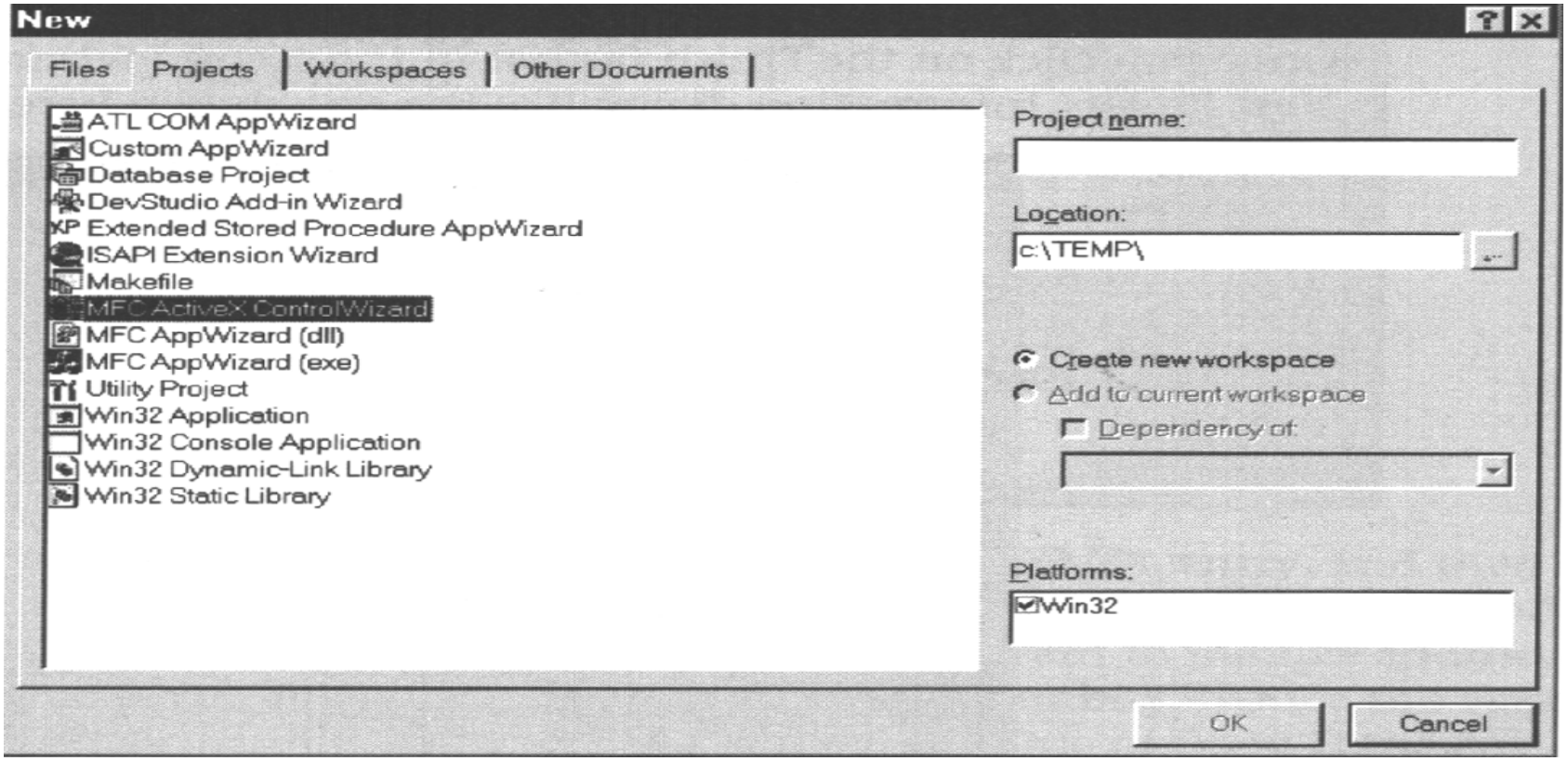
发现市场上的许多 OCX（如 DLL）都不止包含一个对象（如按钮）。事实上它们包含了一簇对象。这一策略对于桌面系统来说当然很不错，但到了 Internet 上就行不通了。要保证在一个控件中保持一个对象。遵循这样的原理去模块化控件，确保了用户不必下载他们绝对需要功能以外的东西。

注释 本章中还将对对象进行相当多的讨论。本章中每个对象都是 Windows 对象（或 COM 使用的特殊形式）。一些 C++程序员可能认为 Windows 对象就是 C++ 对象。因为 COM 使用一种特殊形式的 Windows 对象，这些程序员就认为，使用手边的 C++对象写 OCX 不会出问题。越过真理一步就是谬误。尽管可以用 C++对象创建 Windows 对象，但存在一些限制。本章篇幅不足以全面讨论 C++程序设计中对象的种种复杂性——这个题目，一些作者要用一章或两章的篇幅来作综述性讨论——但是，了解一下对于用 C++写 OCX 时能做些什么以及应做些什么、有哪些限制，都是很重要的。最好按照本章例子中的方法去写 OCX，这样才不会出问题。

测试、测试、再测试 仅仅在本地或仅仅在网络上测试 ActiveX 控件都是不够的。应该使用不同的连接，在多种情况下进行测试。本章最后会看到三个层次的本地测试以及一个层次的针对 Internet 的测试。可能你想再加进几个层次的测试，根本上说，你对一个 ActiveX 测试永远不嫌多。（在发布控件之前测试它是重要的，但在出问题后一定要去查出缺陷，所以要使 ActiveX 控件正常工作，维护精确的问题日志也很重要的。）

编写代码

现在到了看一下简单编码示例的时候了。本章使用 Microsoft Visual C++6.0（尽管你能够使用 4.2 以上版本），这在前面已提到过。让我们开始创建一个C++新工程。但是，与你创建过的其它工程不一样，现在要用 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）来创建工作区。现在开始吧，使用 Flie（文件）|New（新建）命令，系统显示 New（新建）对话框，选择 Project Tab（工程页），则会看到如下所示的对话框。



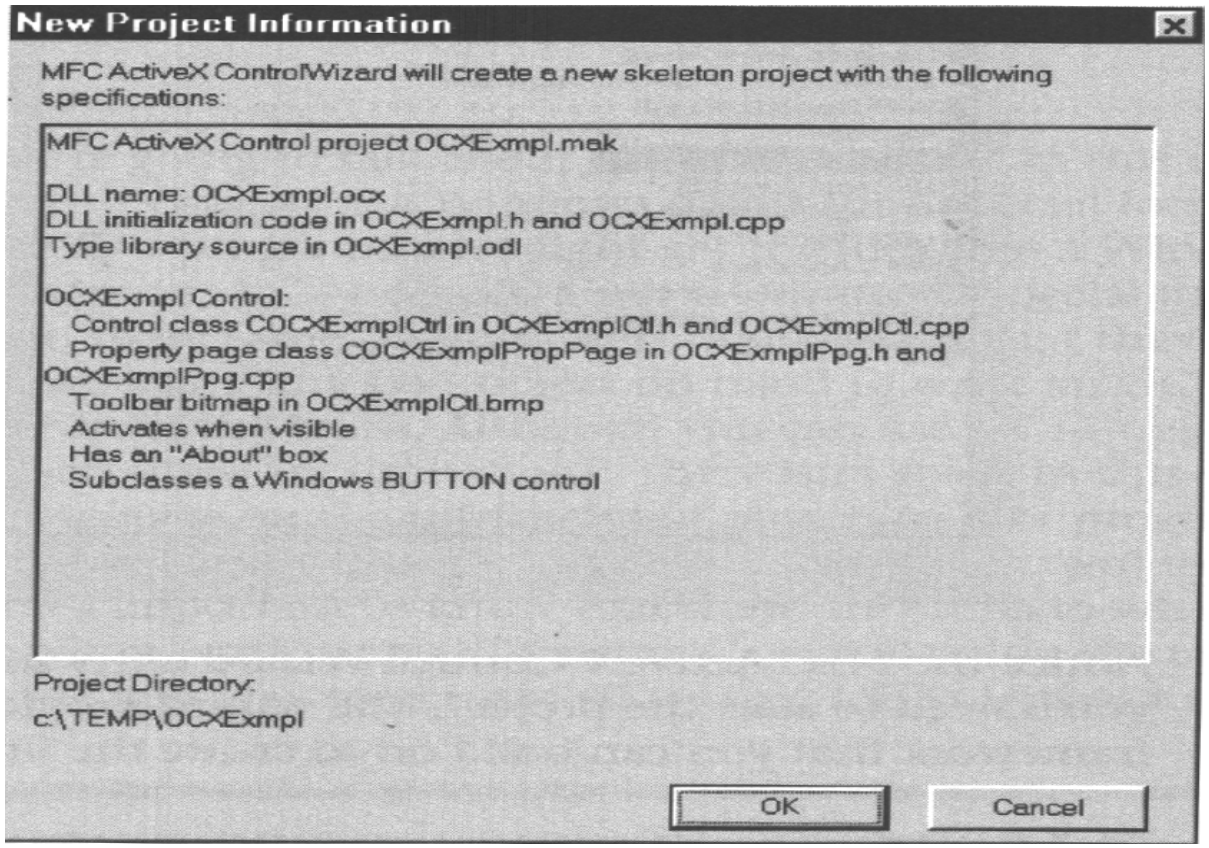
该对话框中的 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）就是开始创建本工程的那个选项。这个向导提供了一个 OCX 框架，通过建立这个框架来创建本例的最后版本。

注释 本例中我使用了新的 Microsoft Developer Studio（开发人员工作室）。看到的所有的屏幕图像都是 Windows 95 下的。如选择使用老的界面，你的屏幕图像就会与这儿的图像不一样。即便你也在使用 Developer Studio 接口，因为这一产品提供的配置选项可以修改，因此，仍可能产生一些小的差别。

工程开始时，在 Project Name（工程名）域中键入工程名。本例用 OCXExmpl 作为工程名。在工程列表框中单击 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导），然后单击 OK。Microsoft Visual C++自动选择了 Win32 选项，它还创建了一个工程目录。

接下来你看到的是 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）屏幕的两个对话框。我采用了这两个对话框的缺省设置，但对第二个对话框的 subclass（子类）域作了选择，如果你也想创建像这个例子一样的控件，那么，在这个域中选择 BUTTON 类。否则，可以浏览可用类的列表框以确定用什么作为你的控件的基础。还要注意到 Visual C++允许创建自己的基类。

在第二个向导屏幕上单击 Finish（完成）按钮后，则会看到如下的 New Project Information（新工程信息）对话框。

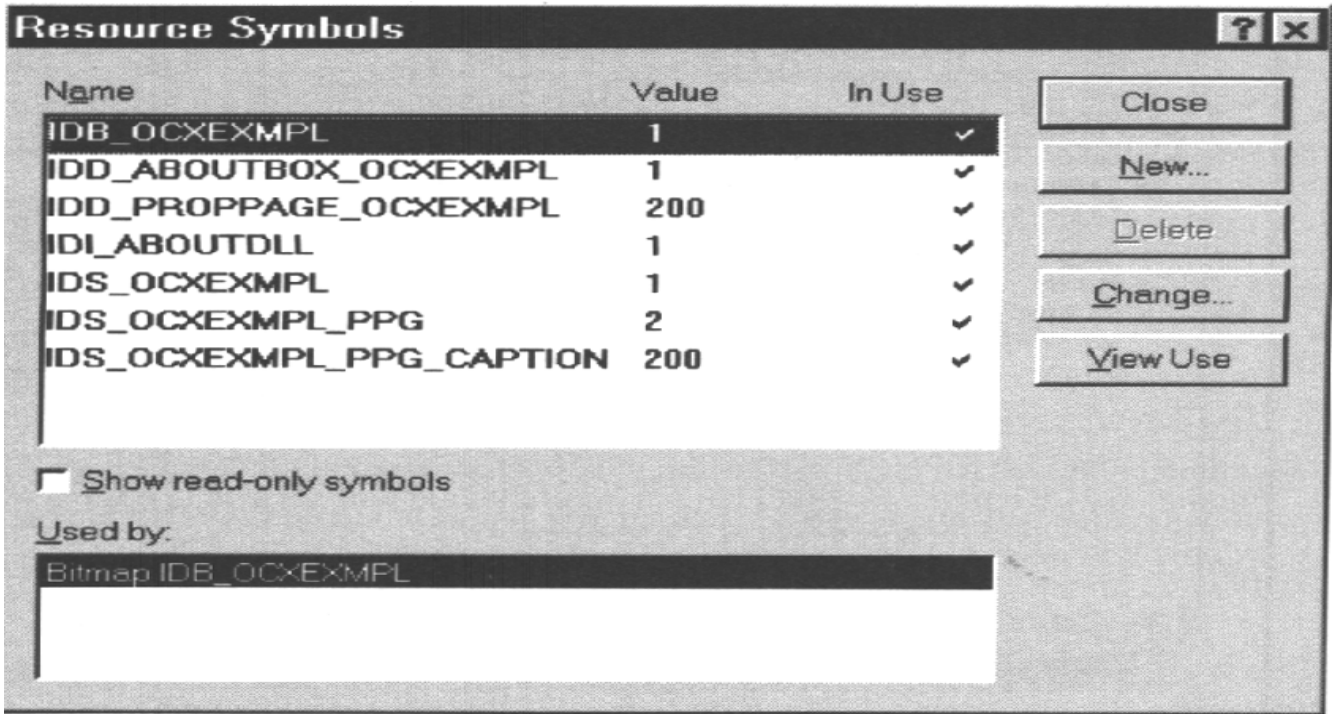


浏览一下所提供的功能列表，确认工程中包含了你所需要的一切内容。核对工程建立无误后，单击 OK，则启动了工程。Visual C++将使硬盘转动一会儿，然后就会看到工程框架。

修改缺省的 About 对话框

现在已有了一个框架，我们把它填充好。我总是先处理简单的事情（谁不是这样呢？）。首先修改 About 对话框。Visual C++ 自动为你创建这个对话框，

你需要做的事就是定制它。访问 About 框很容易，使用 View（视图）|Resource Symbols（资源符号）命令则显示出如下图所示的 Resource Symbols（资源符号）对话框。



选择 IDD_ABOUTBOX-OCXEXMPL 项，单击 View Use 按钮显示该对话框。图 10.1 显示了修改示例中 About 对话框的一种方法。也许你想把版权信息以及公司信息也包含到 About 对话框中。请留意 Microsoft 为对话框提供了多种工具，其中之一是定制控件按钮，你能够把另一个 OCX 贴到 About 对话框或其它你创建的对话框中。

技巧 用右键单击一个控件，然后从上下文相关菜单中选择 Properties(属性)，于是 Properties(属性)对话框显示出来。通过改变 Properties(属性)对话框中 General(普通的)属性页上的 Caption(标题)属性，可以改变静态文本控件的文本。用右键单击 Visual C++的大部分对象，都会显示一个上下文相关菜单——大部分这样的菜单中都包含 Properties(属性)选项。用右键单击 Visual C++中某个对象的目的，就在于让你看到能够对这个特定对象做些什么。请记住，对象不仅仅指控件，它也可以是代码行、工具栏、甚至可以是各种窗口。

Visual C++的最新版本可能要求你对 About 对话框做一些过去不必做的事。你可能需要为 About 对话框创建一个类——Visual C++的老版本只是假定你会创建新类。双击 About对话框，则看到 Adding a Class(添加类)对话框，如下图所示(若没看到该对话框，则说明已有一个类赋给了 About 对话框，你不必再往下继续做了)。

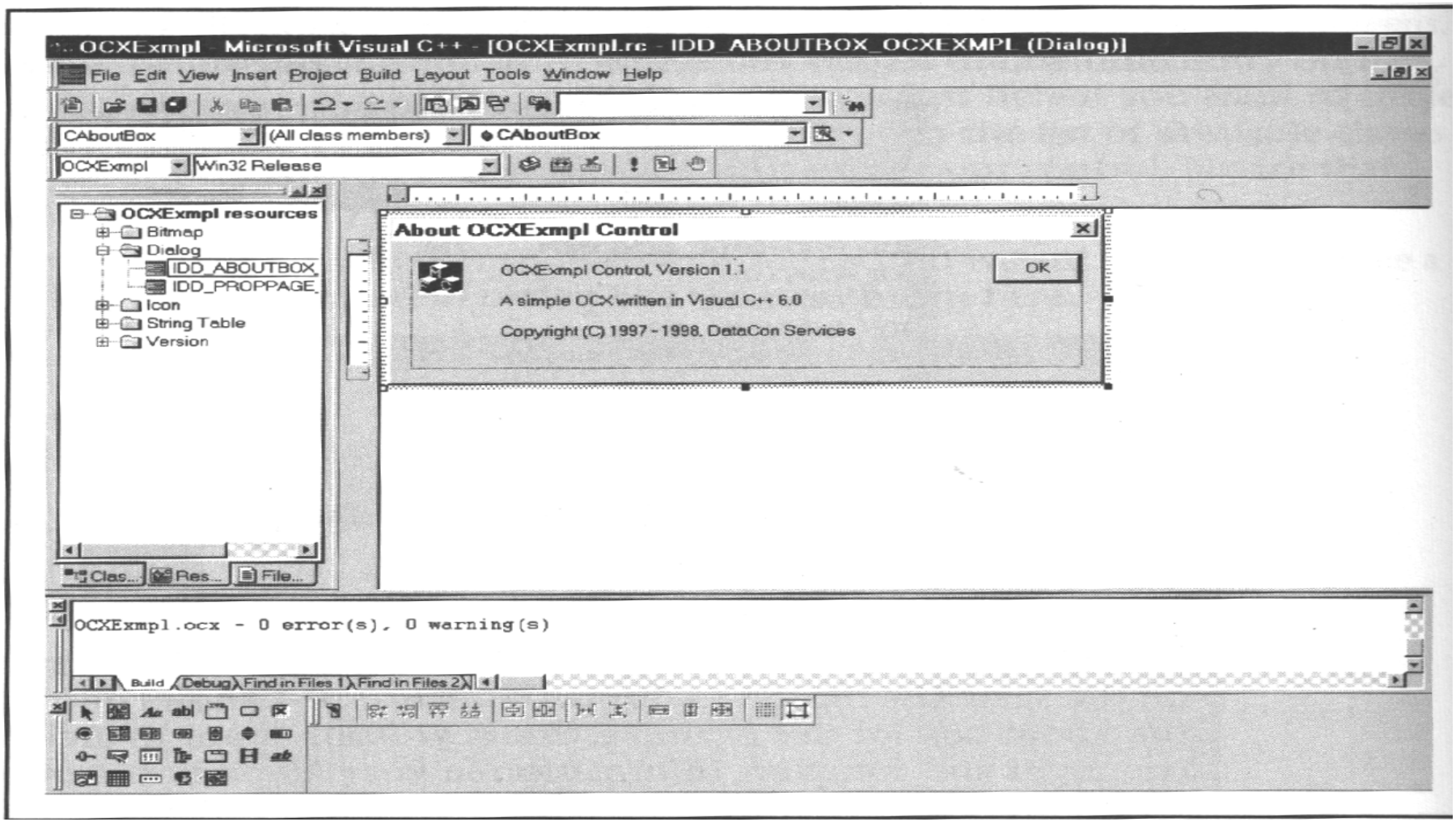
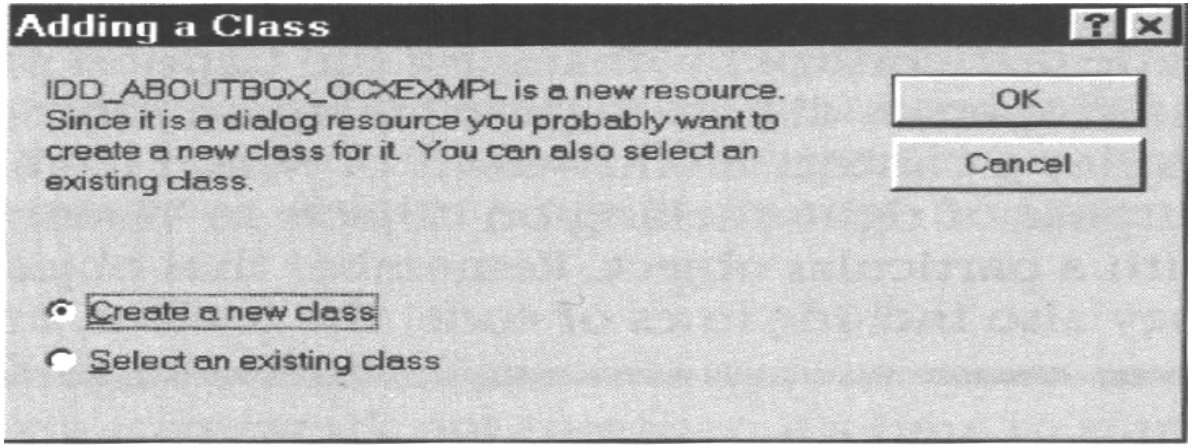
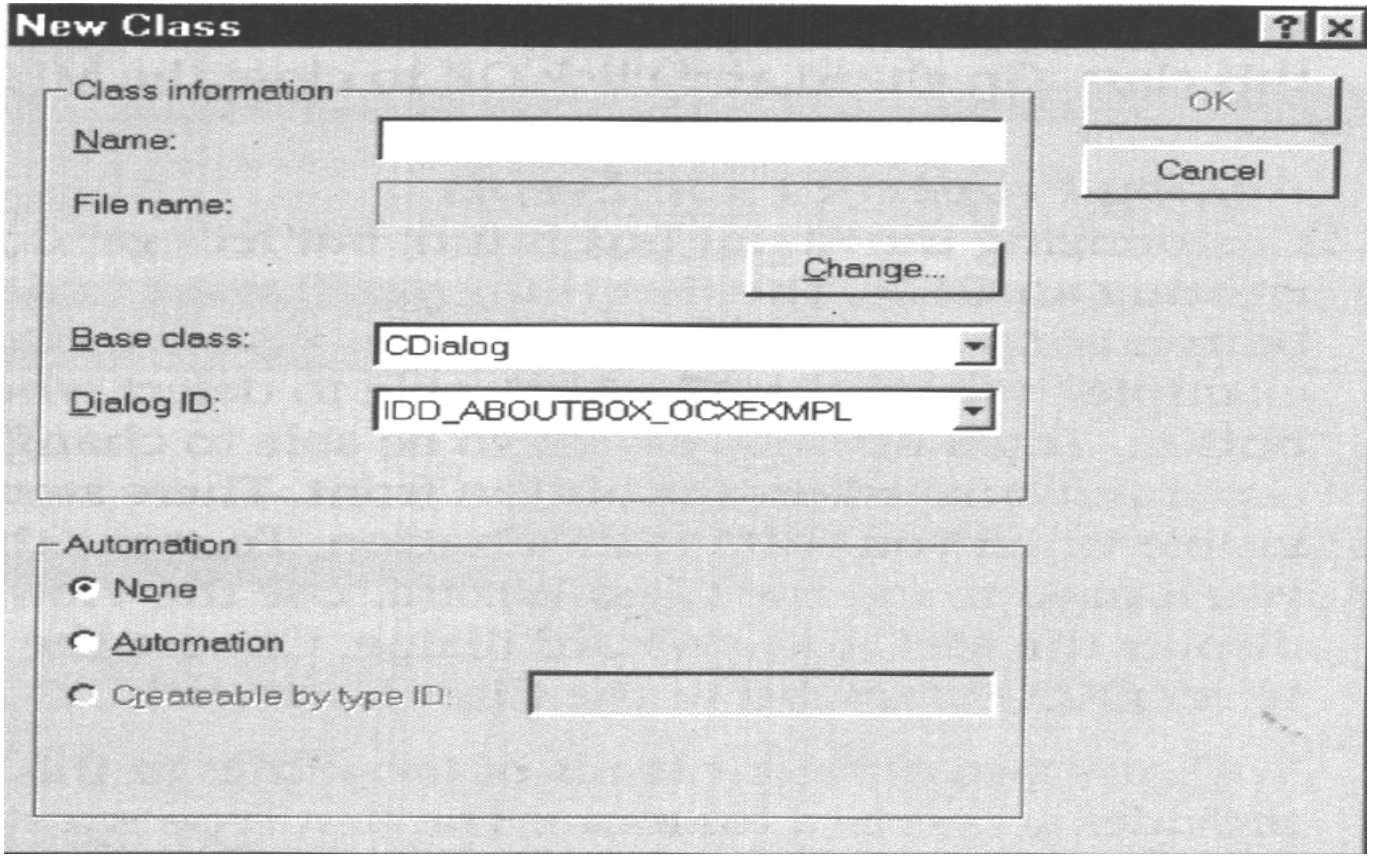


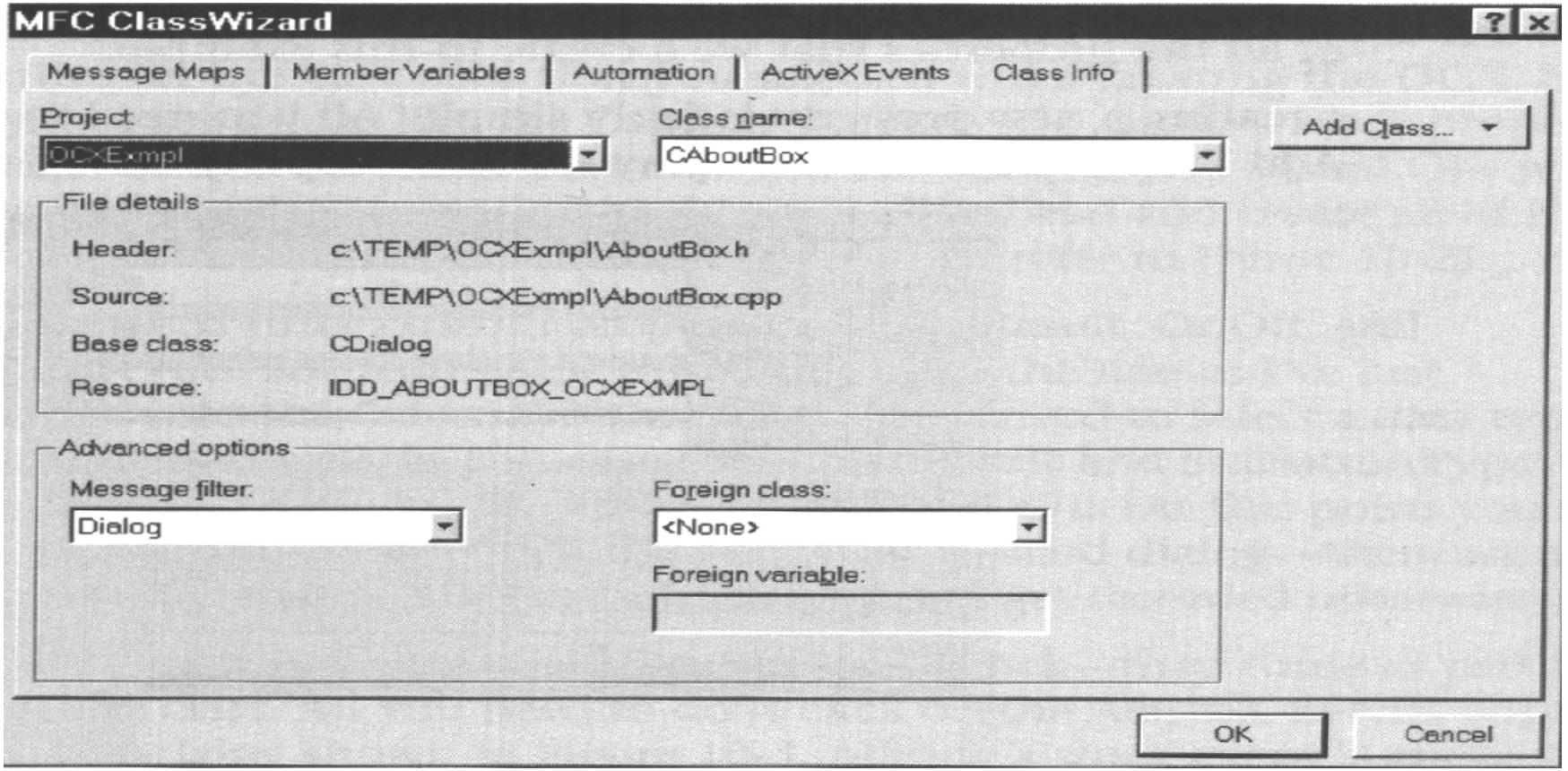
图 10.1　该对话框编辑器看起来和 Visual Basic 的一样，区别在于你必须在主编辑器屏幕中单独访问它



选择 Creat a New Class（创建新类）选项，单击 OK。Visual C++ 则显示如下的 New Class（新类）对话框。



我们这儿用 CAboutBox 作为类名，把它键入到 Name（名称）域。这时这个例子所需的一切都准备好了。单击 OK，则创建所需的类。这时，就会看到在 MFC ClassWizard （类向导）对话框中出现了一个新项，如下图所示（如果屏幕显示不是这样，那么选一下 ClassInfo 选项卡即可）。



这个对话框把新类的有关信息都显示出来。Header（头文件）及 Source（源码）两项是与这个类相关的文件的存放位置。你还会看到，这个类的对话框资源是 IDD_ABOUTBOX_OCXEXMPL。单击 OK，关闭 MFC ClassWizard(类向导）对话框。

添加属性及事件

定制对话框的工作很有趣，现在就让我们继续创建 OCX 吧。首先要完成的工作是使按钮控件的某些属性和事件让使用该 OCX 控件的人能够访问（即可见）。例如，在用户单击按钮时要能检测到这个事件。肯定你想改变缺省的属性，如按钮正面上显示的标题等。第一次创建这个按钮时，没有多少属性是可见的。为了使这些元素成为可见状态，需要使用 Class Wizard（类向导）。使用 View（视图）|ClassWizard（类向导）命令，系统显示 MFC ClassWizard （类向导）对话框。选取 Automation（自动化）这一页，然后在 Class Name（类名）域选择 OCXExmplCtrl。

注 Visual C++对你创建的控件作的假设非常少，甚至于使用什么属性和方法使控件可见也不做任何假定。

在这个例子中，使用了两类不同的属性（实际上 Microsoft 提供了许多供你使用的属性）。第一种是常备(Stock)属性。你会发现，第一次创建 OCX 时，就连我们都认为是理所当然的 Caption（标题）属性等等一类事物都是不可见的。常备属性（在图 10.2 中用 S 表示）是由父类缺省支持的一种属性。另一种是定制属性（在图 10.2 中用 C 表示）。定制属性是把一个特定类作为子类时你向它添加的属性，其中之一是我们将用于创建一个 OnOff 控件的 OnOff 属性。本章下面来讨论做这些事的过程。图 10.2 显示了创建我们例子时所有属性的完整列表。

注 常备属性是基类的一部分，定制属性是把控件作成子类时创建的属性。

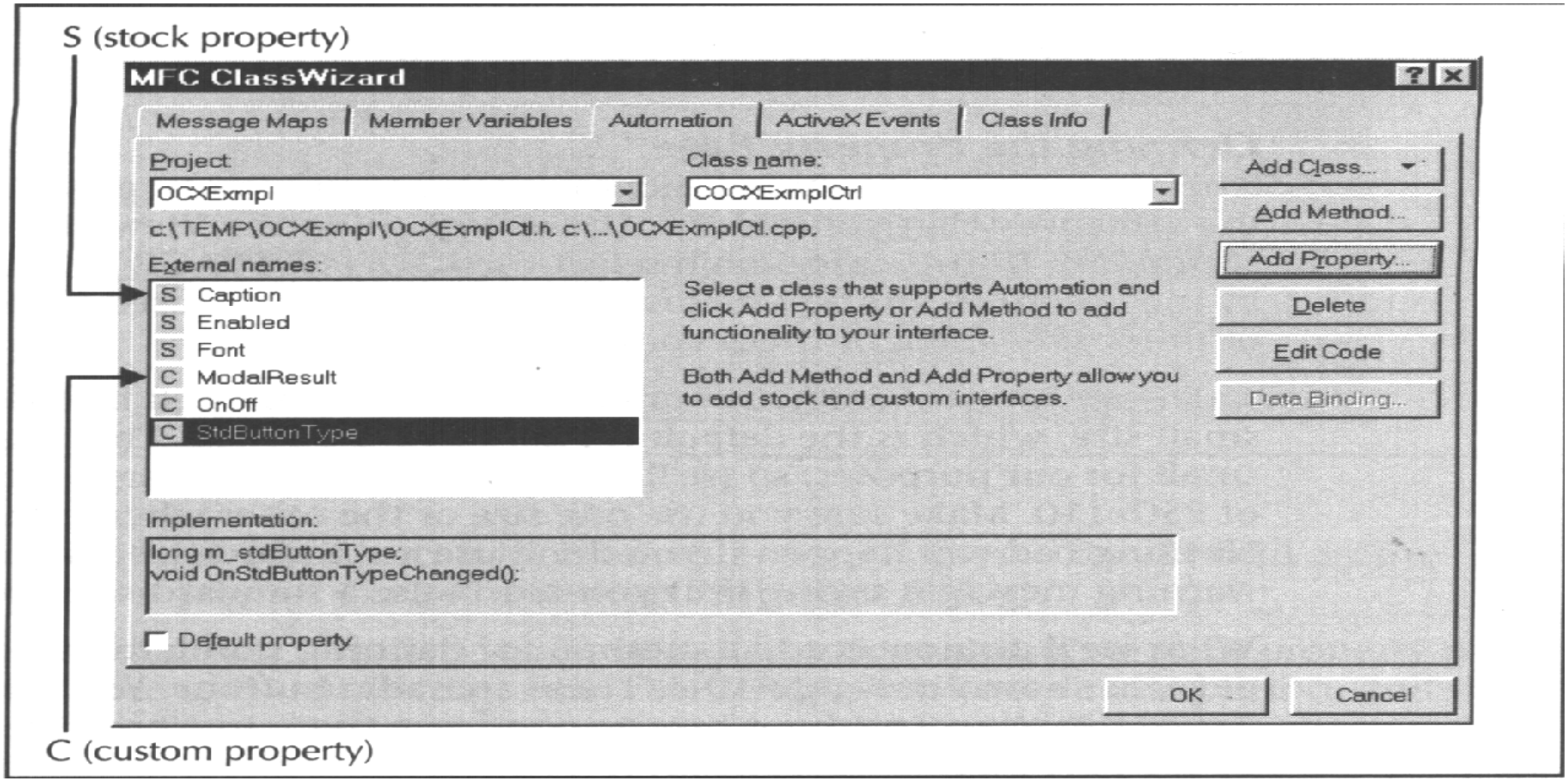
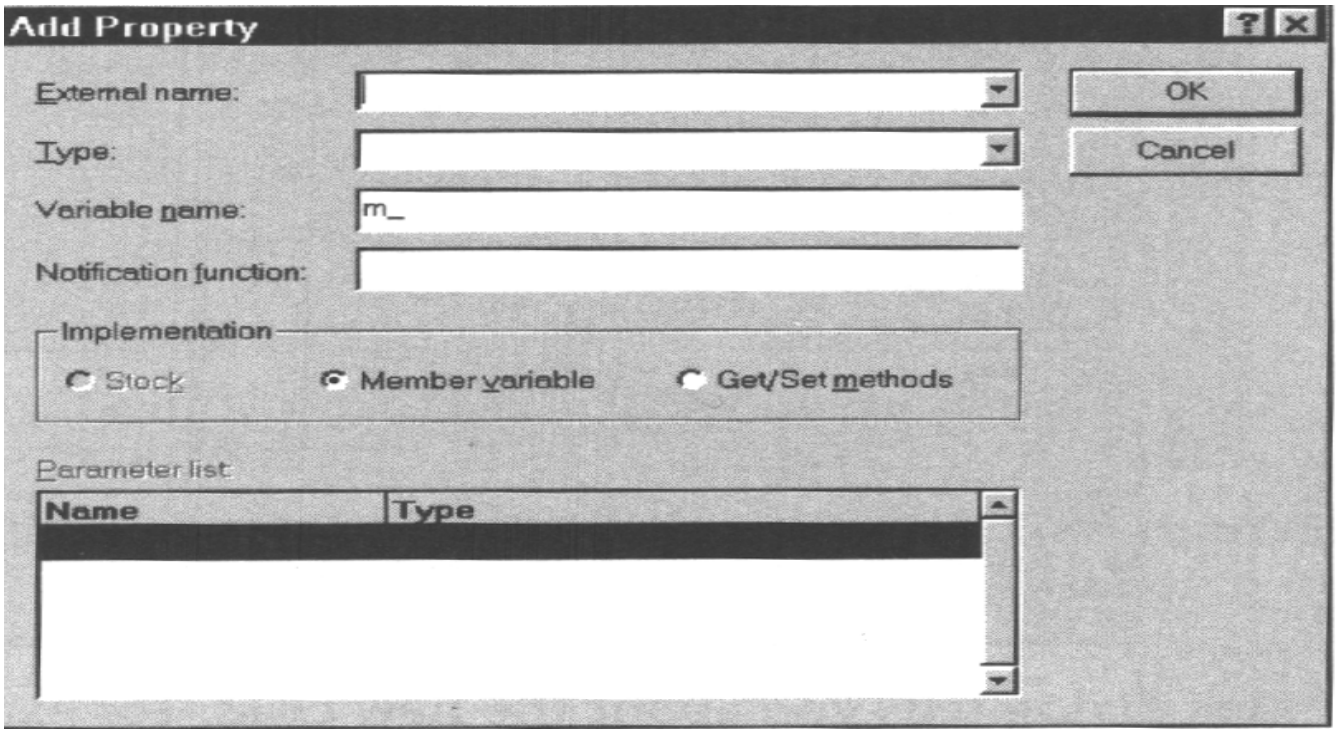


图 10.2　MFC ClassWizard（类向导）使属性及事件对 OCX 用户成为可见状态

创建新属性的过程很简单。只要单击一个 Add Property（添加属性）按钮就可以显示出下图所示的 Add Property 对话框。



Add Property（添加属性）对话框中包含一些以前我们未曾见到过的重要特性。External Name 组合框包含了创建 OCX 时选择的基类的全部缺省属性，在我们的例子中，就可见到 Caption 属性等。为了创建常备属性，从这个列表中选一项并单击 OK 即可，Visual C++将处理细节。现在接着创建出本例子程序中的全部常备属性（参见图 10.2）。

我们还需要三个定制属性：ModalResult、OnOff 及 StdButtonType。为了创建它们，把它们键入到刚才提到的 External Name（外部名称）域中。然后需要在 Type（类型）字段中为它们选择数据类型。在我们例子中，ModalResult 和 StdButtonType 属性是 long 型，而 OnOff 是 BOOL 型。（在这里，你可能要先

关闭 MFC ClassWizard（类向导）对话框，然后再打开它，否则在有些时候，你选择的属性可能没记录下来。）

这个例子中我们使用的事件都是常备的，他们来自于按钮基类的一部分。要做的是单击 ActiveX Events（事件）页，系统显示如图 10.3 所示。添加常备事件的做法与添加一个常备属性的方法一样，只需单击 Add Event（添加事件）按钮显示出 Add Event（添加事件）对话框，从 Extenal Name（外部名称）组合框中选取一个常备的名称，然后单击 OK 即可。图 10.3 显示了需要为这个例子选择的全部常备事件。

定义属性页

现在该向属性页中添加一些功能了，访问属性页的方法与访问 About 对话框的方法相同，就是使用 View（视图）|Resource Symbols（资源符号）命令。在 Resource Symbols（资源符号）对话框选择 IDD_PROPPAGE_OCXEXMPL 项。属性页用途广泛，其中绝大多数是面向配置的。

Visual C++支持两种标准规格的属性页，OCX 的缺省选择是 250×62 的小号规格。这种规格对我们的目的说来太小了，所以需要把它变成 250×110 的大号属性页。创建一个控件时一定要确定用这一种还是另一种。你不用标准尺寸时也不会发生什么坏事，但用户可能会收到警告信息，说你没使用标准尺寸的属性页。

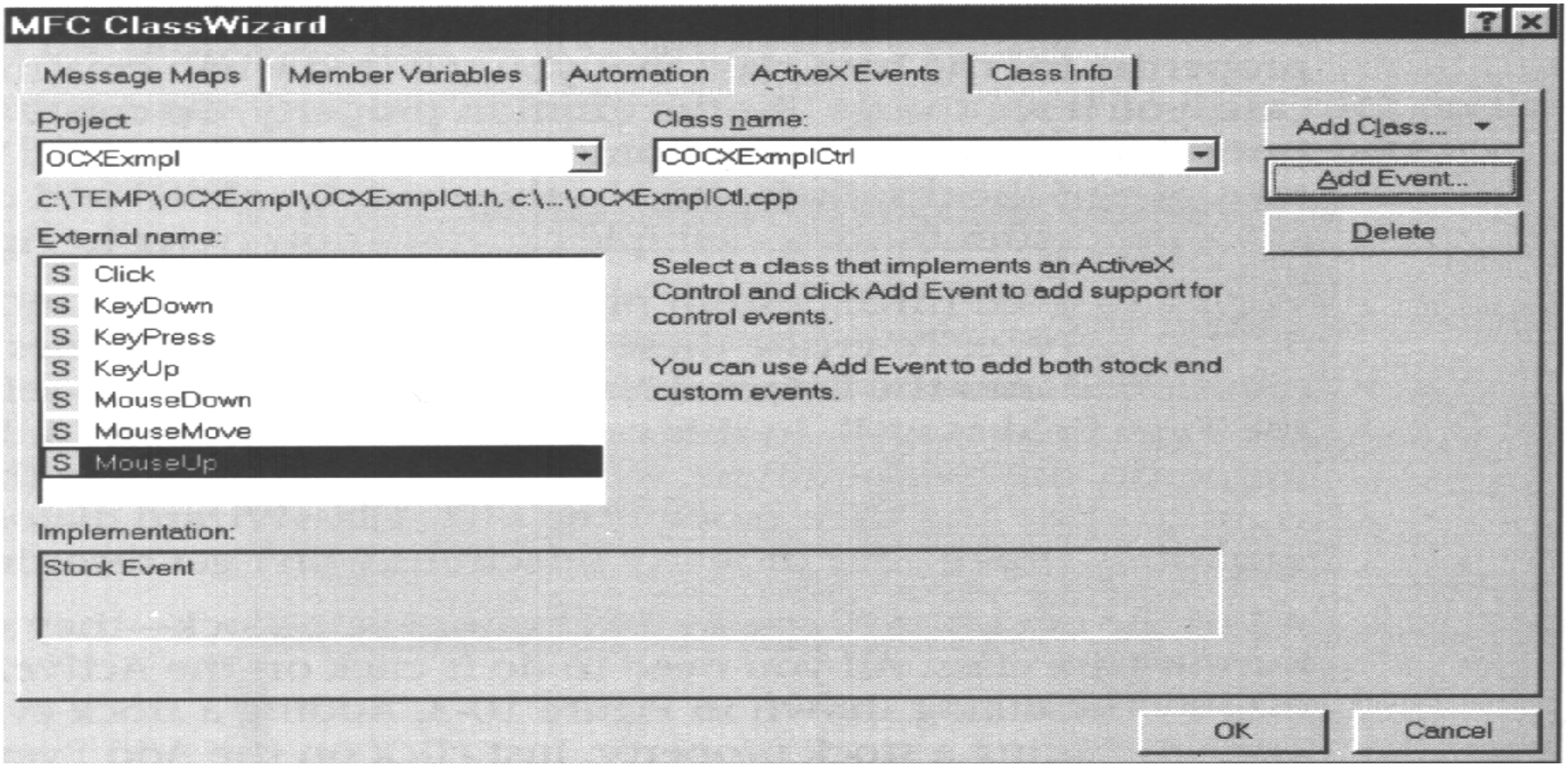


图 10.3　ActiveX 事件页显示了在我们的 OCX 程序例子中要添加的全部常备事件

现在要完成的工作是，为向页中定义标准的按钮类型来添加一种方法，如图 10.4 所示。它们是单选按钮，你需要 10 个按钮（现在不必担心如何配置它们，下述段落中会进行讲解）。每个单选按钮应有不同的 ID，以便于能检测出用户单击的是哪一个。（参见 Radio Button Properties（单选按钮属性）对话框

中 General 页上的 ID 域）。

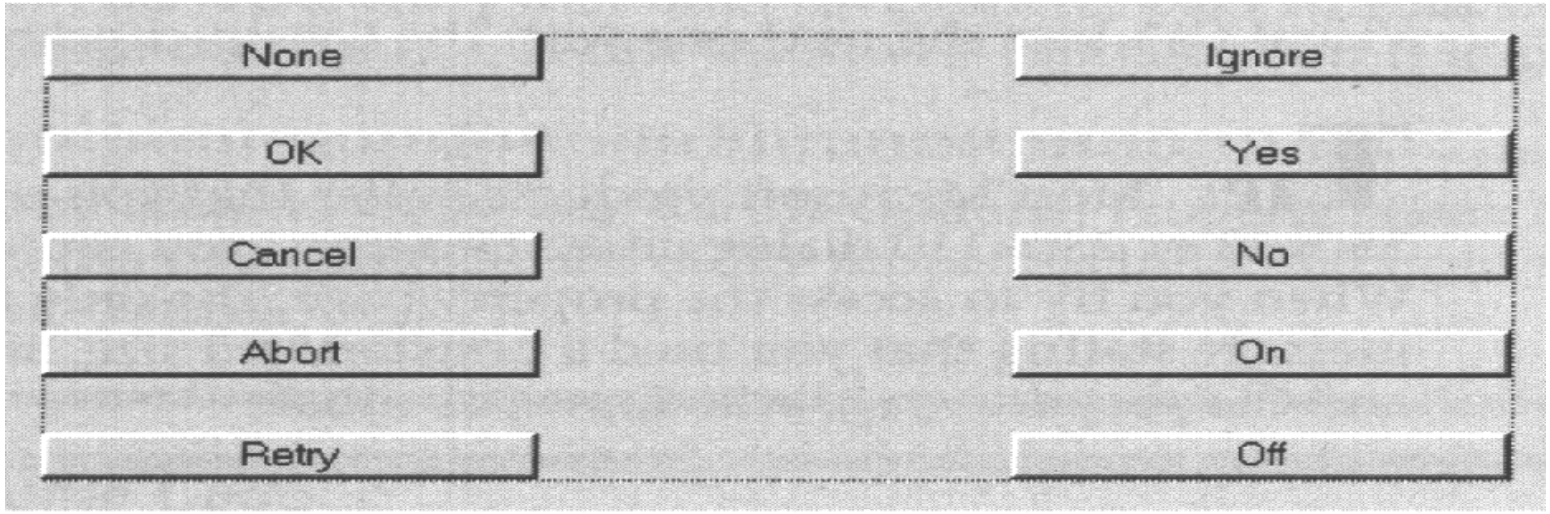
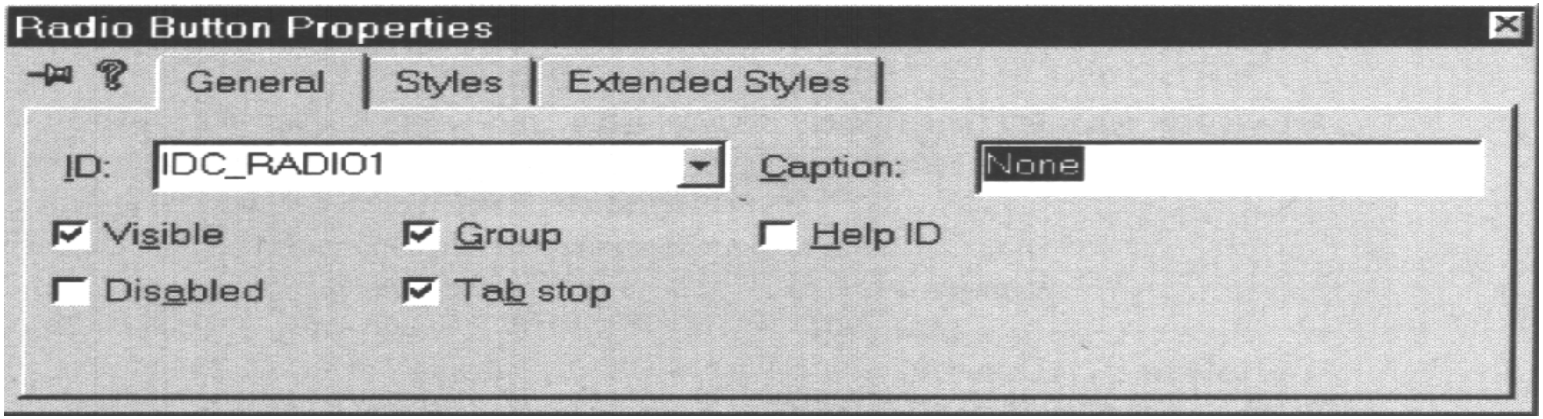


图 10.4　Property Page（属性页）对话框允许用户创建除 on/off 按钮之外的标准按钮类型

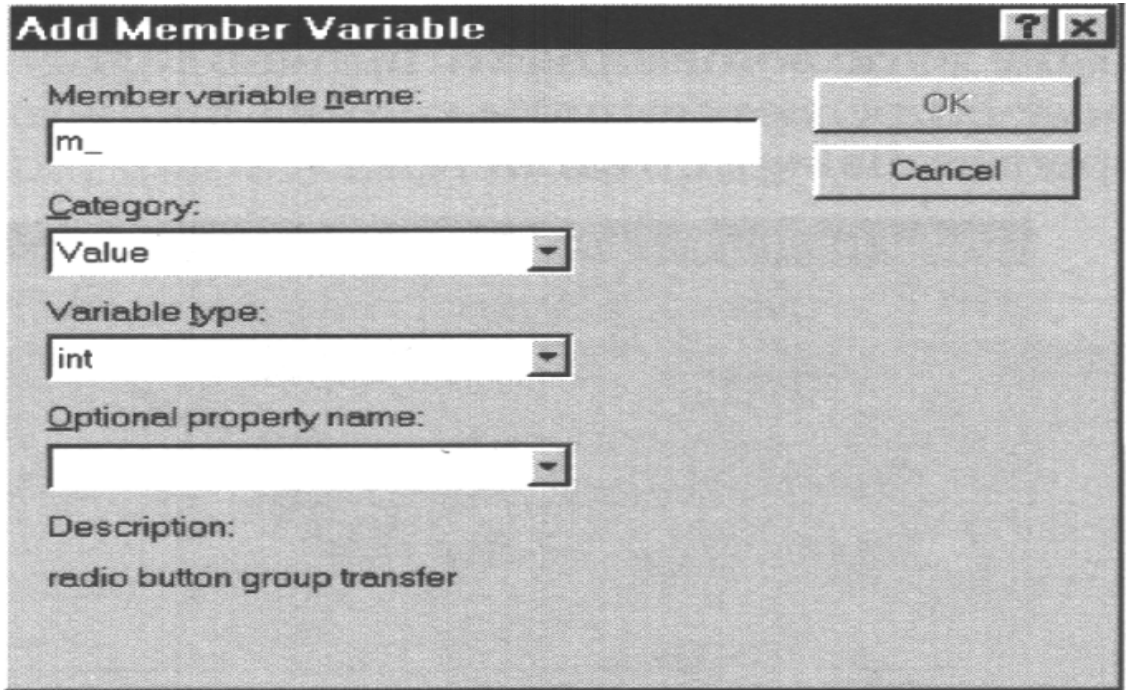
属性页上已有了 10 个标准的单选按钮，现在让它们完成一些任务。用右键单击其中一个单选按钮，从上下文菜单中选择 Properties（属性），系统显示 Radio Button Properties（单选按钮属性）对话框，如下图所示。



为了使单选按钮看起来与如图 10.4 所示的外观相同，要对它们作些小的改动。首先，在 Radio Button Properties（单选按钮属性）对话框选择 Style（风格）页，并为每个按钮选中 Push-like 复选框。还需把这些单选按钮放入一个组，使得当你选择一个新按钮时，当前选择能消除掉。为了完成这个任务，先对使用 IDC_RADIO1 缺省 ID 的组中的第一个单选按钮，复选 Group 及 Tabstop 复选框，而对（该组中）其它的单选按钮都只复选 Tabstop 复选框。否则，最后得到的是每组只有一个按钮的 10 个组而不是有 10 个按钮的一个组。Visual C++把它看到的第一个选择了 Group 复选框的按钮作为组的起点，顺序下去，在 Visual C++看到选择了 Group 复选框的下一个按钮之前，所有按钮都归为这一个组。

技巧 大多数 Microsoft 产品推荐使用尺寸为 250×62 或 250×110 对话框单元的属性页，但你可以按照需要使用任意尺寸的属性页，当你试图访问这个属性页时，你就会看到一条消息，说你没使用标准尺寸。清除这一消息，则属性按通常那样显示出来。

在这个对话框中，我们要对单选按钮再做一件事。为了在单选按钮和 OCX 控件之间建立 OLE 连接，要对它们的输出赋以 OLE 属性。按下 CTRL 键并双击该组中第一个单选按钮(None)，系统显示 Add Member Variable（添加成员变量）对话框，如下图所示：



技巧 键入 CTRL-W 显示 MFC ClassWizard 对话框，选择 Member Variable（成员变量）页，再单击 Add Variable（添加变量）按钮，也可以访问这个对话框。

这儿定制的各个项目都是十分关键的，原因在于 Visual C++不检查它们的

对错，并且也不能从列表中选择它们。在 Member Variable Name（成员变量名）域，键入 m_stdButtorType，这是前面我们创建的定制属性之一的内部名称。保持 Category（范畴）域及 Variable Type（变量类型）域不变。在 Optional Property Name（可选属性名称）域中，键入 StdButtonType，这是把属性页链接到你的 OCX 控件的一个项。记住 C++是区分大小写的，所以大写很重要。

技巧 Optional Property Name（可选属性名称）域的下拉列表中通常包含了从控件基类继承来的属性的完整列表。

添加代码

直到现在，我们还没有向我们的应用程序中添加一行代码。因为我们一直在忙于建立代码框架。现在该向 OCX 中添加代码了。首先要添加一些代码，使得我们的控件能与使用这个控件的用户交换数据，例如，当看到控件属性对话框时，通常想看到这些属性的当前值，类似地，当改变了一个属性值时，就想确信实际控件也随之改变。程序列表 10.1 是需要添加的代码。

程序列表 10.1

```
VOID COCXExmplCtrl::DoPropExchange(CPropExchange* pPX)
{
    //Default actions on the part of the Class Wizard.
    ExchangeVersion(pPX,MAKELONG(_wVerMinor,_wVerMajor))
    COleControl::DoPropExchange(pPX);
```

```
    //Make all of our properties persistent.
    PX_Bool(pPX,"OnOff",m_onOff,FALSE);
    PX_Loog(pPX,"ModalResult",m_modalResult,mrNone);
    PX_Loog(pPX,"StdButtonType",m_stdButtonType,0);
}
```

现在假定你不喜欢按钮的缺省大小，并且想在用户把按钮插入到 Web 页或其它框架中时，显示一个具体的标题。可以在 OnReset()函数中改变这两个属性。程序列表 10.2 中显示了需要修改的代码。请注意，我们使用 COleControl 类的函数来完成所需的修改。SetText()函数修改按钮的标题。每当用户插入这一控件时，标题“Button”就会显示在该控件上。SetControlSize()函数把控件大小变成 75×25 个像素。显然，可以按你自己的意愿设置这些属性，甚至能够选择其中一个按钮作为缺省按钮。

程序列表 10.2

```
VOID COCXExmplCtrl::OnResetState()
{
    COleControl::OnResetState();   //Resets defaults found in DoPropExchange

    //Modify the Microsoft control to match custom size settings.
    COleControl::SetText("Button");
    COleControl::SetControlSize(75,25);
```

```
}
```

现在我们有了交换信息的方法，并且把控件按我们的要求进行了设置。下面来实现我们创建的三个定制属性。是的，每当你创建一个定制属性时，就需要定义一些代码，使得这种属性能完成一些任务。否则，它就只能呆在那里干不成事。程序列表 10.3 列出了用以实现 ModalResult、OnOff 和 StdButtonType 属性所需添加的代码。下一节我来解释一下这些代码的内部操作方式。在目前，你只要知道这些代码实现了我们创建的属性就够了。

程序列表 10.3

```
void COCXExmplCtrl::OnModalResultChanged()
{
        //We don't need to do anything here except set the modified flag.
        SetModifiedFlag();
}

void COCXExmplctrl::OnOffChanged()
{
        //If the programmer set the OnOff property true,take appropriate action.
        if (m_onOff)
        {
                COleControl::SetText("On");     //Change the Caption
```

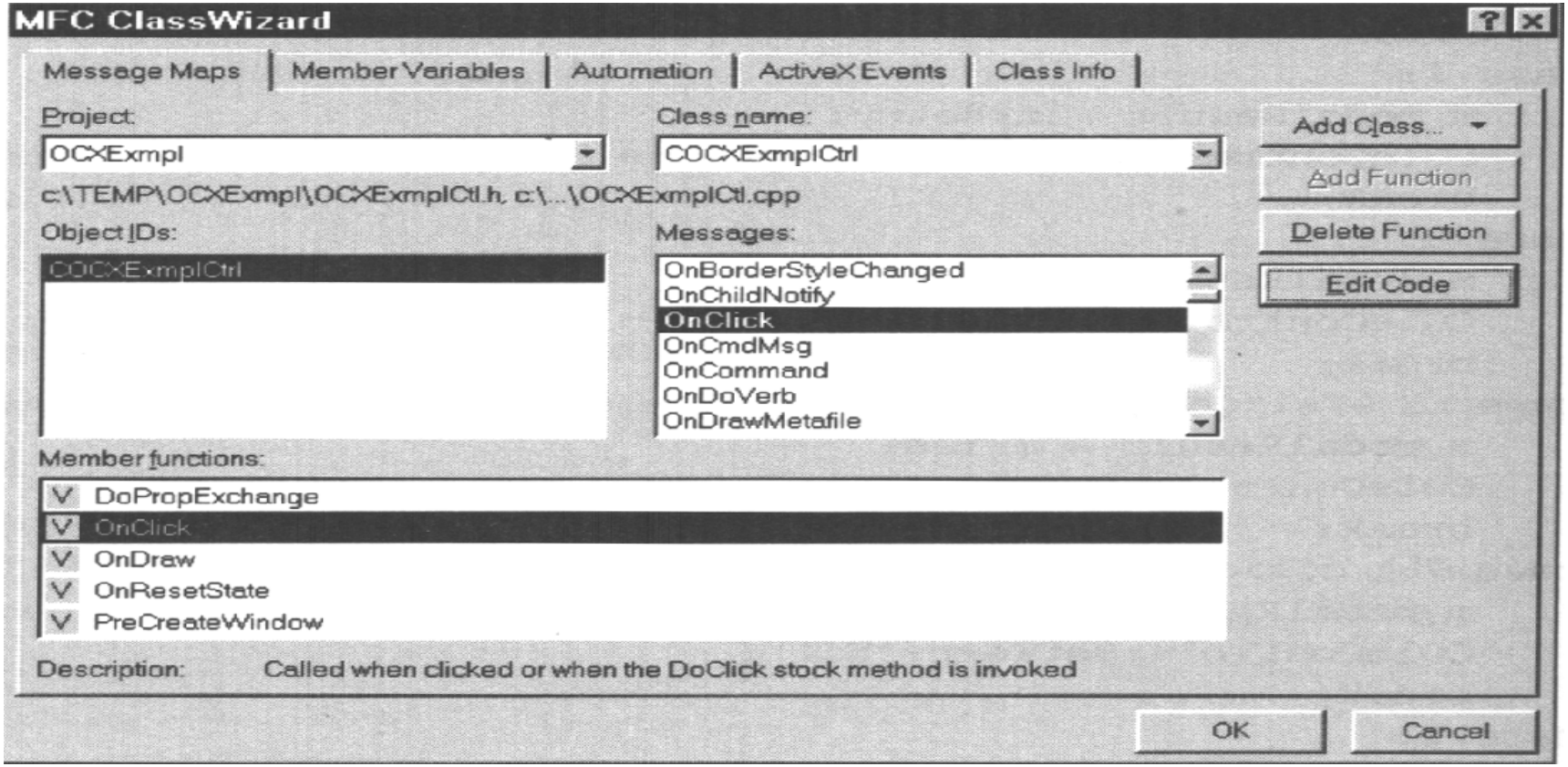
```
            m_SetOn = TRUE;                 //Set an internal caption flag
            m_modalResult = mrOn;           //Set the modal result value.
        }
        else
        {
            COleControl::SetText("Button");     //Restore default Caption
            m_SetOn = FALSE;                    //Turn our caption flag off
            m_modalResult = mrNone;             //Use the default modal result.
        }

        //Perform the default action.
        SetModifiedFlag();
}

void COCXExmplCtrl::OnStdButtonTypeChanged()
{
   // Change the modal result and button caption to match the user selection.
        switch (m_stdButtonType)
        {
        case 0:
            m_modalResult = mrNone;
```

```
        COleControl::SetText("Button");
        break;
case 1:
        m_modalResult = mrOK;
        COleControl::SetText("OK");
        break;
case 2:
        m_modalResult = mrCancel;
        COleControl::SetText("Cancel");
        break;
case 3:
        m_modalResult = mrAbort;
        COleControl::SetText("Abort");
        break;
case 4:
        m_modalResult = mrRetry;
        COleControl::SetText("Retry");
        break;
case 5:
        m_modalResult = mrIgnore;
        COleControl::SetText("Ignore");
```

```
        break;
case 6:
        m_modalResult = mrYes;
        COleControl::SetText("Yes");
        break;
case 7:
        m_modalResult = mrNo;
        COleControl::SetText("No");
        break;
case 8:
        m_modalResult = mrOn;
        COleControl::SetText("On");
        break;
case 9:
        m_modalResult = mrOff;
        COleControl::SetText("Off");
        break;
}

//Set the OnOff property to false since the user selected another type.
m_onOff = FALSE;
```

```
        //Set the modified flag.
        SetModifiedFlag();
}
```

我们现在需要在 OCXEXMPLCtl.cpp 文件中做最后的一些编码工作。当用户单击按钮时怎么办？如果他使用的是标准按钮类型之一，OnOff 控件将返回一个标准的模式结果值。但是，OnOff 控件还有一个特殊行为。如果你把 OnOff 属性设置为 True，按钮则随着用户的单击而在 On 和 Off 之间切换。我们需要添加一些特殊的事件代码来处理这种情况。使用 View（视图）|ClassWizard（类向导）命令显示出 MFC ClassWizard（MFC 类向导）对话框，选取 Message Maps（消息映射）页，选择 Class Name（类名）域中的 COCXExmplCtrl 项，在 Messages（消息）列表中选中 OnClick，单击 Add Function（添加函数）向类中添加一个函数框架，这时 MFC ClassWizard（MFC 类向导）对话框如下图所示：



现在向 OnClick()函数中添加代码。单击 Edit Code（编辑代码）按钮，Visual C++就把你带入到新函数中。程序列表 10.4 列出了需要添加的代码。

程序列表 10.4

```
void COCXExmplCtrl::OnClick(USHORT iButton)
{
        //see if the OnOff flag is set. If so, change the caption and internal
```

```
        //caption flag.The effect you should see from this code is a toggling
        //of the caption text.
        if (m_onOff)
        {
                if (m_SetOn)
                {
                        COleControl::SetText("Off");
                        m_SetOn = FALSE;
                        m_modalResult = mrOff;
                }
                else
                {
                        COleControl::SetText("On");
                        m_SetOn = TRUE;
                        m_modalResult = mrOn;
                }
         }
         //Call the default OnClick processing.
        COleControl::OnClick(Button);
}
```

现在我们已经实现了函数编码部分，但我们还需要向 OCXEXMPLCtl.h 文

件中添加两个支持项。第一个项是枚举类型，其唯一目的是使源代码容易阅读。每个元素对应一个标准按钮类型。第二个项是一个特殊变量。如果注意一下代码，就会看到我们一直引用一个名为 m_SetOn 的成员变量，但这个变量现在还不是类的一部分。程序列表 10.5 显示了如何向头文件中添加所需的枚举类型及特殊变量——就在 Event maps 和 Dispatch and event IDs 项之间添加。

程序列表 10.5

```
//Event maps
        //{{AFX_EVENT(COCXExmplCtrl)
        //}}AFX_EVENT
        DECLARE_EVENT_MAP()
        //Creat a new enumerated type for the modal result.
        typedef enum
        {
                mrNone = -1L,
                 mrOk = 1L,
                mrCancel = 2L,
                mrAbort = 3L,
                mrRetry = 4L,
                mrIgnore = 5L,
                mrYes = 6L,
                mrNo = 7L,
```

```
        mrOn = 8L,
        mrOff = 9L,
}MODALTYPE;

//Special On/Off state variable.
    BOOL    m_SetOn;

//Dispatch and event IDs
public:
    enum{
    //{{AFX_DISP_ID(COCXExmplCtrl)
    dispidModalResult = 1L,
    dispidOnOff = 2L,
    dispidStdButtonType = 3L,
    //}}AFX_DISP_ID
    };
```

详细解析代码

你对上述代码作出的最初反应肯定是觉得太可怕了，但是，如果每次处理一个函数，这实际上又相当容易。事实上，作为 ActiveX 控件定义处理过程的一部分，Visual C++已为你写出了大部分的代码，所以你写出的代码并没有比

为一个标准应用程序所写的代码多多少。我们添加的函数，只不过是定义我们要求这个控件所要完成的特殊任务。

让我们把代码分解开来看一下。你修改的第一个函数是 DoPropExchange()。这个函数在这个例子中只完成一种服务——把你的定制属性变成持久属性。实际上，PX_ 系列函数调用允许把一个特定属性的值，从一个会话期存贮到下一个会话期。对你定义的每一个变量类型都调用一次该类型的函数。每次调用都接受类似于下文所述的四个变量：

```
PX_Bool(pPX,"OnOff", m_OnOff,FALSE);
```

第一个变量是指向属性交换结构的一个指针，Visual C++自动定义这一结构，你只管使用就是了。第二个参数包含了属性的外部名，用户在 Property Inspector 中可以看到它。（对于 Property Inspector，Visual C++称之为属性对话框，Delphi 称为对象观察器，等等）。第三个参数是属性的内部名。使用它在整个程序中确定该属性。最后的参数是定义属性的缺省值（除非你想让用户在 Properties（属性）对话框中看到一个空白域）。

你修改的下一个函数是 OnResetState()。这个函数提供的是，当用户把组件添加到窗体上时，用户看到的一些美学细节。在这儿，我们给予组件一个缺省标题，并改变其尺寸使之与能在 Web 页上正常工作的定制尺寸相匹配。如果你设计一个 ActiveX 控件用于其它目的，就需要改变这一设置，以满足你使用最多的程序设计语言的需要。要记住重要的是 OnResetState()函数允许你对使用控件所需要的任何设置进行操作。

技巧 Microsoft 使用的组件缺省尺寸差不多是 Borland 产品（如 DelPhi）的两倍。Internet 控件的尺寸可变，但本例中使用的 75×25（宽×高）规格在绝大多数情况下都能工作。

在代码的消息处理程序 (handler) 部分，三个修改过的函数中有两个需要进一步修改。ModalResultChanged()函数不需作修改，所以这里就不讨论它了。与 ModalResultChange()函数相关的属性 ModalResult 由其它两个函数进行修改。先看一看 OnOffChange()函数。我们要做的是设置内部标题标志和初始标题。如果程序员设置 OnOff 属性为 True，我们通过设置它的标题为 On 把控件设置为一个 on/off 开关按钮。当按钮用作 on/off 开关时，还提供了不同的模式结果值。请留意，内部属性变量 m_onOff 追踪着标志的状态，而内部属性 m_SetOn 追踪着 OnOff 控件的当前状态(on 或 off)，因为这个按钮初始时为 On，所以一开始设置 m_SetOn 标志为 True。

现在看一下这个 OCX 的 Property Page（属性页）中功能所需的处理过程。OnStdButtonTypeChanged 函数只不过是一个 case 语句，它按照创建各种缺省按钮类型的需要来改变按钮的 Caption 和 ModalResult 属性。请留意，如果用户选择了缺省的按钮类型，我们还不得不关闭 OnOff 的按钮处理过程。

OnClick()消息处理函数在运行期间是激活的，这里有两个层次的动作。第一，需要确定程序员是不是把这个按钮定义为 on/off 开关，如果是，我们就改变内部状态变量(m_SetOn)和按钮标题。该函数让按钮状态在 On 和 Off 之间按需要切换。一旦我们完成了使按钮工作所需的内部处理，我们就调用缺省的 OnClick 处理例程。如调用失败，则导致 OCX 跳过你为按钮事件附加的专用于

程序设计环境的代码。例如，如果你在 Visual C++应用程序中使用这个控件，附加于 Visual C++中 exposed 事件的任何代码都会被忽略。

在能够使用这个组件之前，还得在 Visual C++中建立它。建立过程的一部分是用 Windows 自动为你注册 OCX。我确实喜欢这一特色，因为在测试 OCX 时它为我节省了不少时间。唯一的缺憾是，预先注册污染了工作环境，从 Internet 上使用 ActiveX 控件的角度来看，你必须到另一台机器上去测试这一组件。

测试控件

一旦创建了一个任意种类的 ActiveX 控件（不论是否你的编译器是否将它称为 OCX），都要进行一些测试，从而保证它按预期的方式工作。对于想用于 Internet 的 ActiveX 控件来说，其测试工作最好分为四个阶段：三个层次的内部测试和第四层次的外部测试。下述列表说明了各个阶段的重要性。

- 内部测试阶段 1　在标准环境中使用这个控件。在将控件从标准程序设计平台移走之前，看一下它是否能够工作正常。这样做的理由很简单：在浏览器内测试 ActiveX 控件时没有调试程序可用。在把它从 C++（或其它 OCX/ActiveX）程序设计环境中移走之前测试 ActiveX 控件的基本功能，使你拥有调试程序，从而能随时找出真正严重的问题。
- 内部测试阶段 2　在浏览器内进行本地测试。在你的本地机器上建立简短测试，看一看 ActiveX 控件能否装入到一个 HTML 页中，这样可以使后面的工作节省一些时间，你还应该验证一下，是否有足够多的属性在实际使用控件时可用，并且在你浏览测试页时它能否工作。

- 内部测试阶段 3　使用网络连接测试整个 Web 页。一旦你对 ActiveX 控件的基本功能完成了测试工作，并验证了它能在浏览器中正常使用，就要确定它是否能够在一整页 HTML 标记中正常工作。总之，不能与同在一页上的其它控件一起工作的控件有什么用呢？控件间的相互作用能引起一些真正奇怪的问题。可以在应用程序中使用标准窗体来测试相互作用问题，但也帮不了多少忙，问题是浏览器并不是像你所喜爱的编译器那样来处理 ActiveX 控件。
- 外部测试　在未被污染的机器上核查控件功能。你遇到的最大问题是污染。请记住，大部编译器总是自动地对你所创建的 OCX 或 ActiveX 控件进行注册。而进入你的 Internet 站点的人却没有这个便利。非常重要的是，用未被污染的客户机和服务器在 Web 页的上下文环境中测试你的新控件。换言之，测试的最后阶段，要像任何访问你的 Internet 站点的人那样来看待这个控件。

现在你已经明白我们要做些什么了，让我们看一下我们创建的控件的测试工作吧。在下面我将使用 Visual C++ 6.0。你可以使用支持 OCX 的任意程序设计环境，例如，可以使用 Visual Basic 或 Delphi 来测试 OnOff 控件，看看在这些语言中控件是如何工作的。重要的考虑不在于你用什么语言来测试，而在于要使用包含了全部调试支持的一些程序设计语言来完全地测试这个控件。在预定要使用控件的最安全的环境中，要确保创建的控件能够真正地按预期方式工作。由于许多 Internet 工具处于测试阶段，你会发现看起来好象控件能工作却又工作不了。在一个已知的环境中，通过对控件进行测试来消除控件本身的干扰，

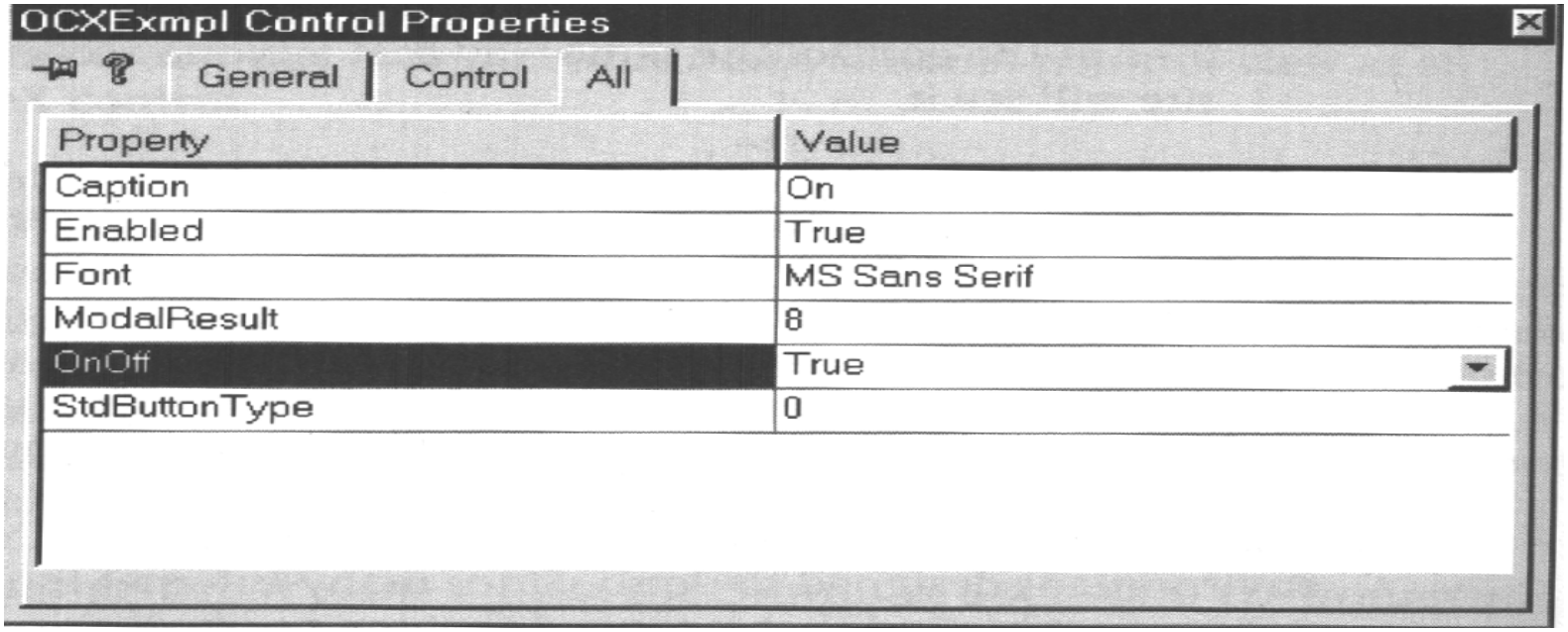
这是迈向找出问题的第一步。

内部测试阶段 1：在标准环境中使用控件

内部测试的第一阶段不会是一个既长又累人的过程。真正需要完成的任务是使用标准程序设计环境创建一个工程，然后把你创建的 ActiveX 控件添加进去。要保证所有属性如预期的那样工作起来，花些时间彻底地核查属性页。

注 记住需要使用 Project（工程）|Add to Project|Components and Controls（组件与控件）命令显示出 Components and Controls Gallery（组件与控件展台）对话框，在这个对话框中把 OCXExmpl 控件添加到当前工程中去。

在你喜欢的（支持 OCX 工作的）程序设计环境中创建一个新工程。对于本例来说，在 Visual C++中创建一个基于对话框的应用程序就能容易地测试该控件。MFC AppWizard（MFC 应用程序向导）为你做了大部分工作。（在第 2 章中已讨论过用 Visual C++ 创建基于对话框的应用程序的过程）。我们把这个工程命名为 OCXTest。



创建了新工程之后，必要时再创建一个窗体，并把 ActiveX 控件添加进去。Visual C++自动地将 ActiveX 控件注册，所以前一节中创建的控件就会出现在可用控件列表中。（其它程序设计环境可能需要你单独地注册你的 ActiveX 控件）。图 10.5 显示了为调试这一例子而创建的测试程序对话框，它还显示了 Property（属性）对话框中选中的 Control（控件）属性页。

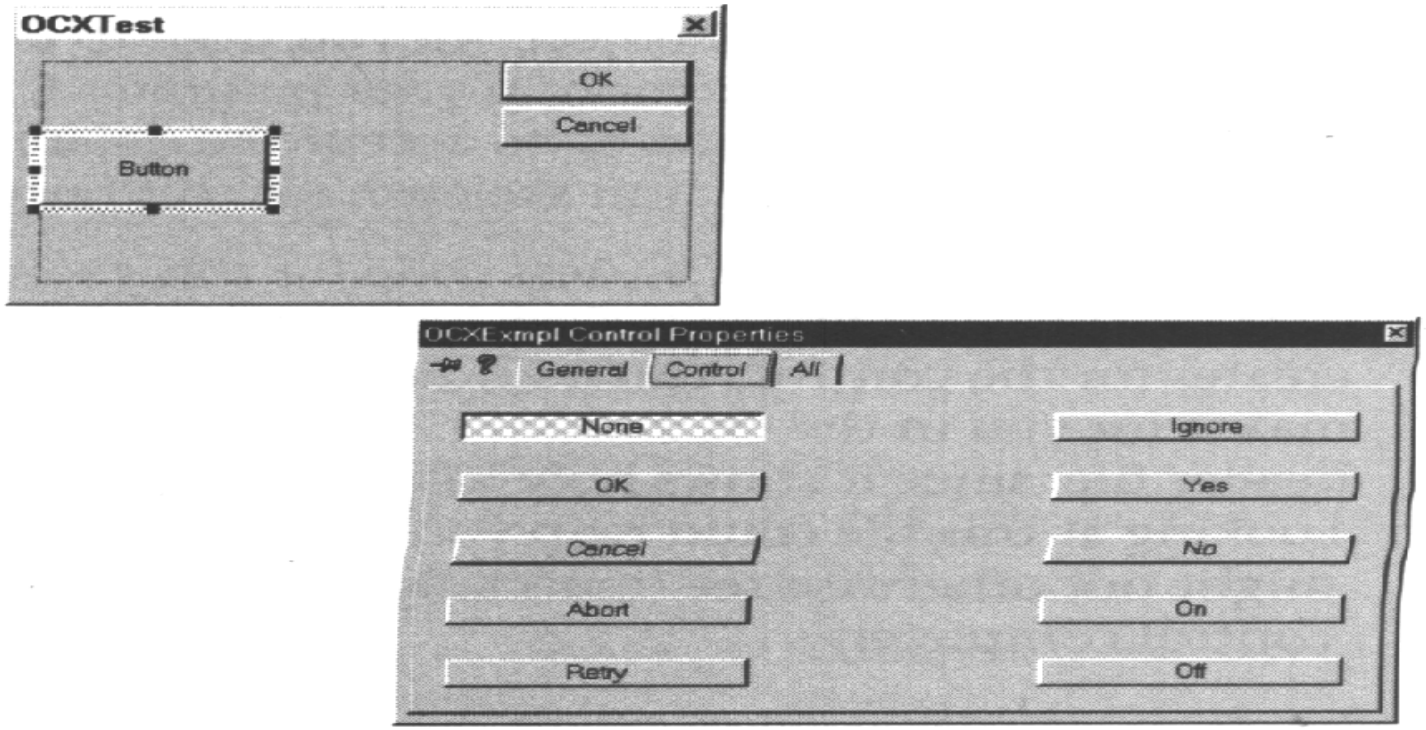


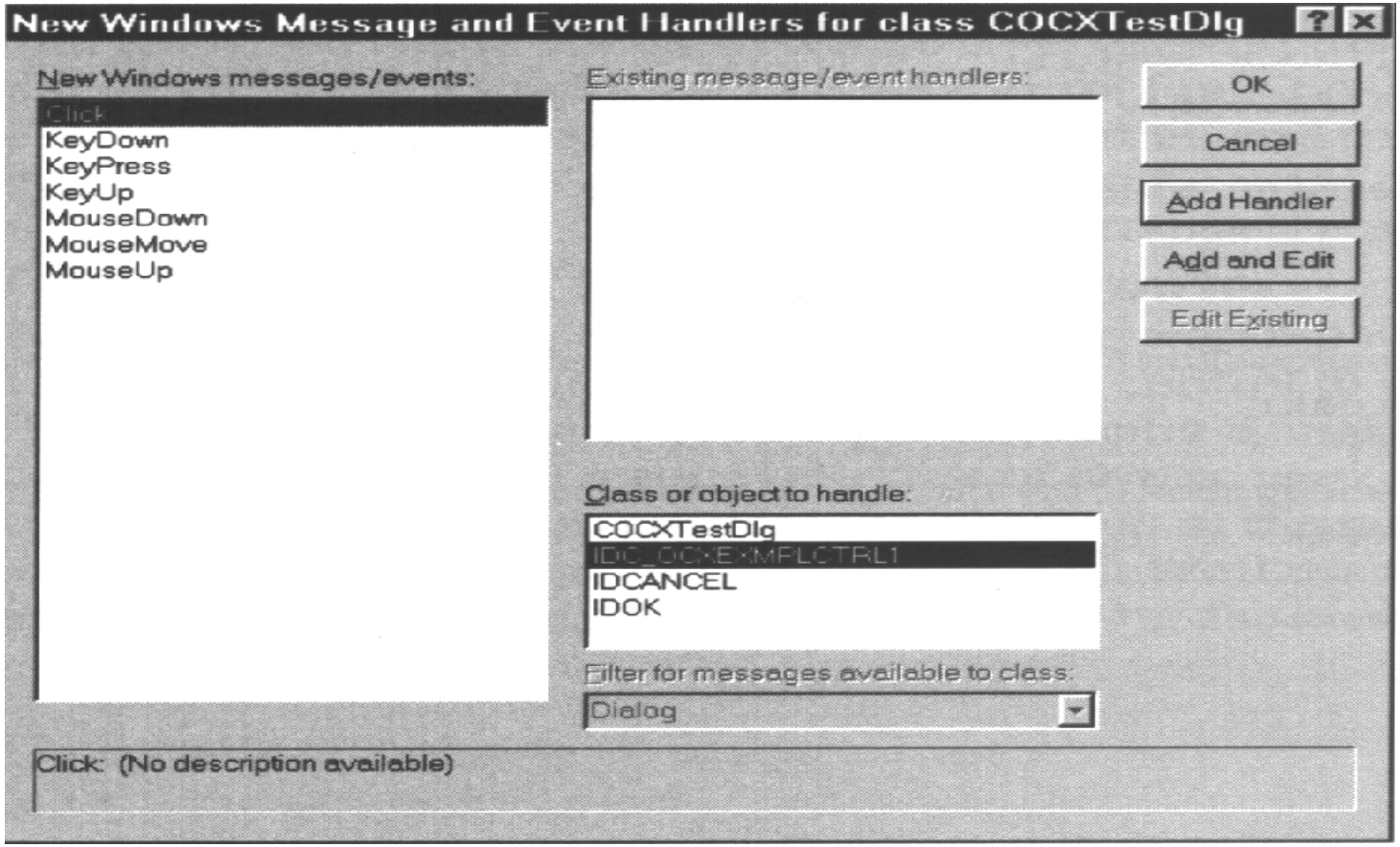
图 10.5　控件属性页对话框表明了这个按钮的特殊特色

绝大多数程序设计环境还提供了一种方法，使得你可以查看与一个控件相关联的所有属性。下图显示了 OCXExmpl Control Properties（控件属性）对话框的 All 页，其中 OnOff 属性设置为 True。

也许你想向程序中添加一些测试代码，从而能够核查各个控件事件的作用。例如，本例中的 On/Off 开关按钮提供了许多依赖于怎样设置按钮属性的模式结果返回值。把 OnOff 属性设置为 True 则创建了一个开关按钮，ModalResult 属

性在两个值间切换。但是，也可以从 Properties（属性）对话框的 Control （控件）页中很容易地选择一种标准按钮值。

第一件要做的事是按下 CTRL 键并双击控件，然后看到 Add Member Variable（添加成员变量）对话框，在 Member Variable Name（成员变量名）域中键入 m_OnOffButton，还要保证 Category（范畴）域设置成 Control（控件），并且 Variable Type（变量类型）域设置成 COCXExmpl。现在就能够在测试应用程序中访问控件的属性了。用右键单击控件，在上下文相关菜单中选中 Events，就能看到如下图所示的 New Window Message and Event Handlers（新建窗口消息和事件处理程序）对话框。



请注意，Visual C++自动地选择了 Click（单击）事件和 IDC_OCXEXMPLCTRL1 对象。需要你做的工作是单击 Add and Edit（添加与编辑）按钮来向你的程序中添加一个新函数。看到 Add Member Function（添加成员函数）对话框后，接受缺省函数名并单击 OK。这时就会看到这个按钮的函数框架。

程序列表 10.6 是这个例子的 C++测试代码。请注意，Visual C++自动为控

件创建的 GetModalResult()封装类函数的使用。在 OCXEXMPLE.H 文件中，可以找到 Visual C++ 为你制作的全部说明。因为头文件说明了 Visual C++是如何与控件打交道的，所以看一下这个头文件是有启发意义的。看一下这个文件，能帮助你找出可能漏掉的接口问题，（特别是如果你不对控件的每个属性进行完全测试时更是如此）。

注释 创建 OnClickOcxexmplctrl1()函数的方法有多种，最简单的方法是使用 MFC ClassWizard（类向导）（使用 View（视图）|ClassWizard（类向导）命令显示这个向导）。在 Object IDs（对象标识）列表框中选中 IDC_OCXEXMPLCTRL1 项，并在 Messages（消息）列表框中选中 Click 项。这时单击 Add Function（添加函数）按钮，就把函数添加到了你的程序中。

程序列表 10.6

```
void COCXTestDlg::OnClickOcxexmplctrl1()
{
        //Get the current ModalResult value
        long liModalResult;
        liModalResult = m_OnOffButton.GetModalResult();

        //Determine which modal result was returned and display a message.
        switch(liModalResult)
```

```
{
case -1:
	Messagebox("None button pressed","State of control",MB_OK);
	break;
case 1:
	Messagebox("OK button pressed","State of control",MB_OK);
	break;
case 2:
	Messagebox("Cancel button pressed","State of control",MB_OK);
	break;
case 3:
	Messagebox("Abort button pressed","State of control",MB_OK);
	break;
case 4:
	Messagebox("Retry button pressed","State of control",MB_OK);
	break;
case 5:
	Messagebox("Ignore button pressed","State of control",MB_OK);
	break;
case 6:
	Messagebox("Yes button pressed","State of control",MB_OK);
```

```
            break;
        case 7:
            Messagebox("No button pressed","State of control",MB_OK);
            break;
        case 8:
            Messagebox("Button is On","State of control",MB_OK);
            break;
        case 9:
            Messagebox("Button is Off","State of control",MB_OK);
            break;
        }
    }
```

我们已创建了一个简单窗体并把你的控件添加到该窗体上面，试着测试一下它吧。示例程序将显示一个简单的带有 ActiveX 控件的对话框。单击控件，就会看到另一个对话框，其中显示了按钮的状态，如图 10.6 所示。单击 OK 或 Cancel 控钮（由 Visual C++自动提供）就结束本程序。如前所述，这是对控件基本功能的简单测试。到现在为止，我们核查了属性页、属性和使用控件的结果。

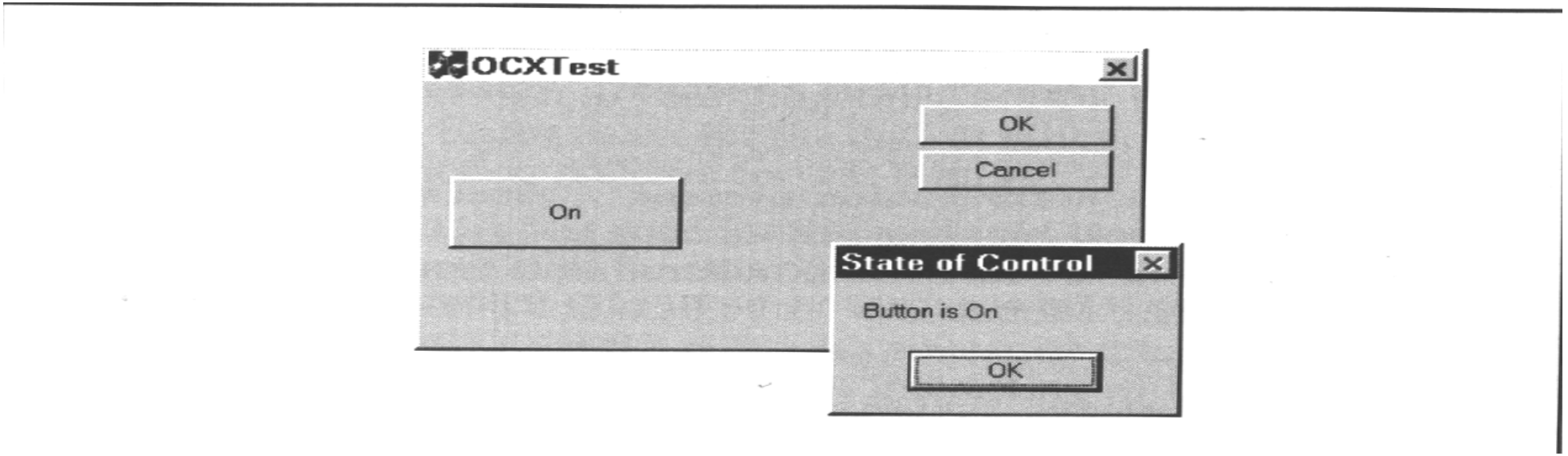


图 10.6　OnOff 控件按预期方式工作：返回模式结果值

内部测试阶段 2:　在浏览器内进行本地测试

到现在为止，我们还没有在本章中做一些哪怕是远远地接近一下在 Web 页中工作的事，至于向 Web 页中添加 ActiveX 控件，涉及得就更少了（在第 8 章中我们确实研究过使用 Web 页问题，如果想了解使用 Internet 的简单情况时，一定去读一下这一章）。该变化一下了，测试过程的下一步就是把控件插入到一个 HTML 文档中，我们再次使用一个非常简单的设置。

注释 在这一节中，我们使用一个非常简单而且容易使用的实用程序。如果计划设计许多普通的或高难度的 Web 页，那么，应该考虑使用某个工具，比如 Visual InterDev 。但 ActiveX Control Pad 用作测试工具或用作设计简单 Web 页更好使用些。

Microsoft 提供了一个 ActiveX Control Pad 实用程序，可用于测试控件时快速建立 HTML 页。图 10.7 是第一次打开这个实用程序时的样子。还可以用它创建功能强大的 Web 页（第 8 章我们讨论过创建过程）。

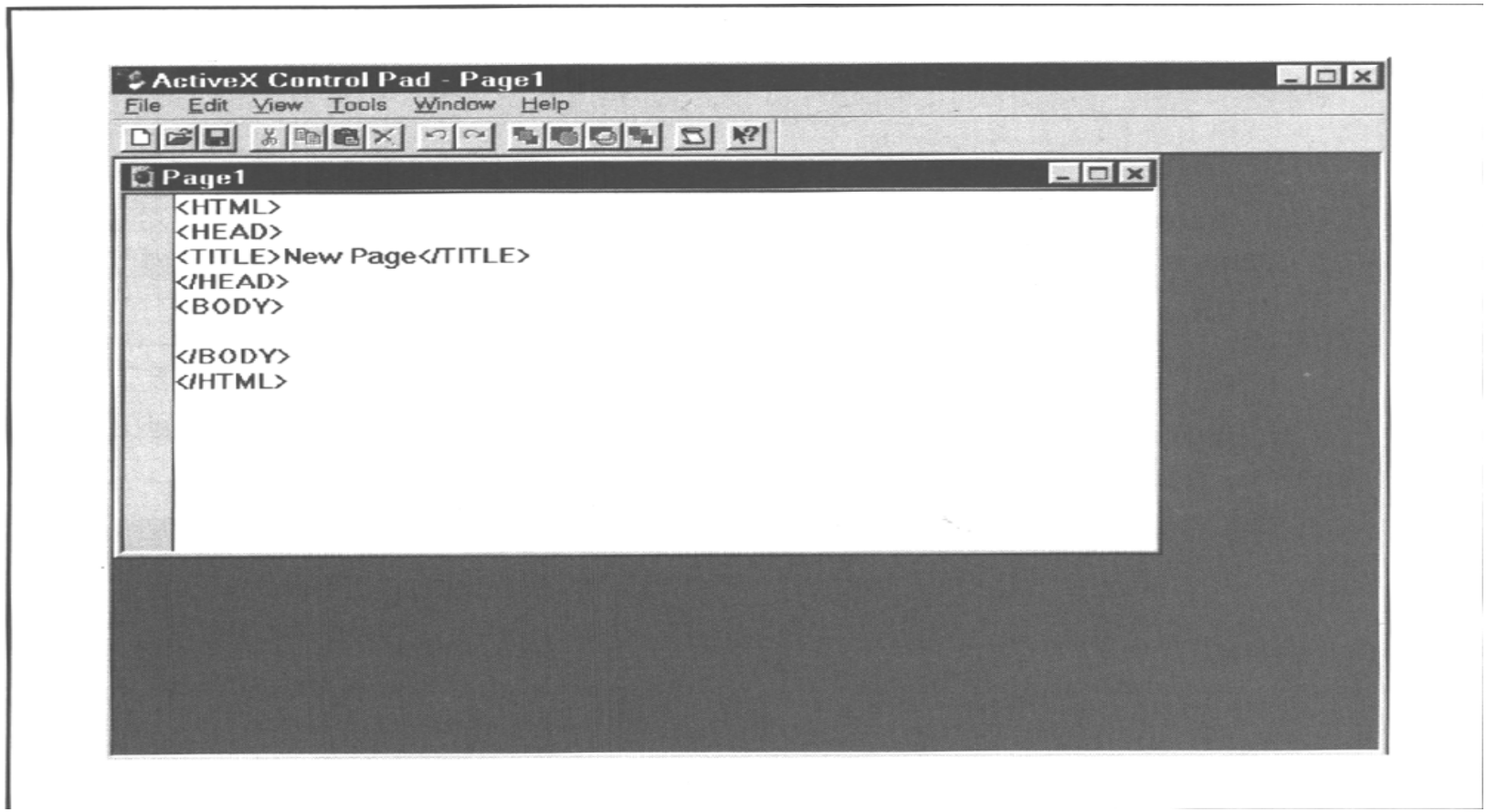


图 10.7　ActiveX Control Pad 允许快速地设计一个 HTML 页来测试你的控件

为了创建用于测试 ActiveX 控件的 HTML 页，使用 Edit(编辑)|Insert ActiveX Control(插入 ActiveX 控件)命令显示出 Insert ActiveX Control(插入 ActiveX 控件)对话框。从提供的列表中选择你的控件，单击 OK，就会看到 ActiveX 控件已装入，如图 10.8 所示。正常情况下，应该把该控件放入一个窗体中，但这

儿唯一的目的只是测试它而已。

每装入一个新控件，ActiveX Control Pad 都会自动地显示出 Properties （属性）对话框，如图 10.8 所示。这个对话框包含了这个控件的已发布的(published)属性的标准列表。请留意图 10.8 还显示出了 General 属性页，通过用右键单击控件并在关联菜单中选择所需的项，就可显示出这个页。对我们的示例来说，要选择第二个 Properties（属性）项。

我们需要改变的唯一一个属性是 Properties(属性）对话框中的 OnOff 属性，把它变成 True，然后关闭包含控件的对话框。ActiveX Control Pad 就向 HTML 页脚本添加一个标记（如果看不懂 HTML 页上的专用术语的话，请到第 8 章找出对 HTML 标记的说明 ）。请注意该项包含了控件的 CLASSID 和所有需要设置的属性。

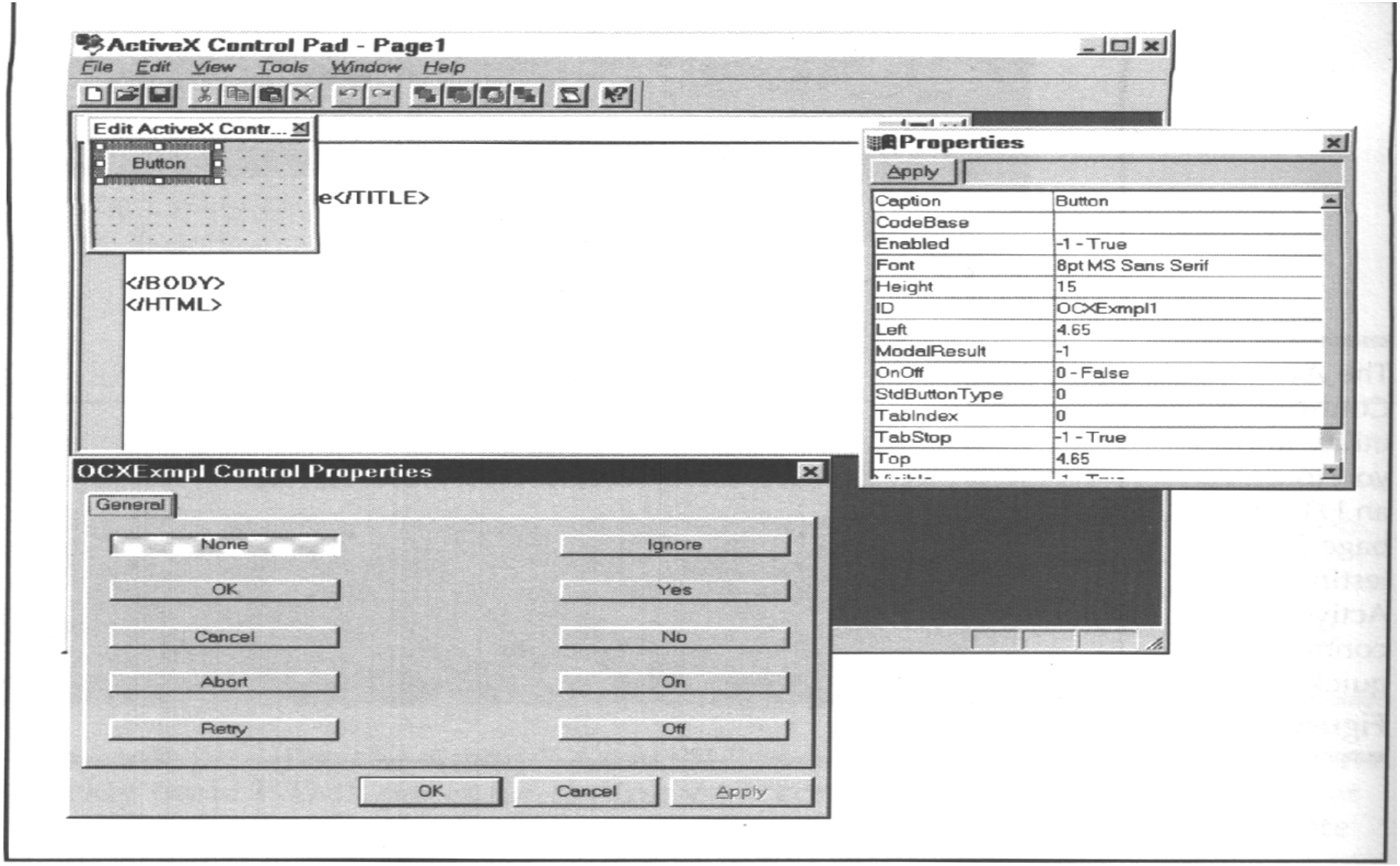


图 10.8　ActiveX Control Pad 在装入你的控件后就显示它

技巧 你在 HTML 脚本中紧挨着<OBJECT>标记的地方会看到一个小按钮。单击这个按钮，再次显示这个控件，从而可以编辑它的属性。每当你把一个 ActiveX 控件或一个 HTML 页框架放入脚本时，ActiveX Control Pad 就显示这个按钮，使你能容易地按需要编辑控件或页面布局。

使用工具栏上的 Save（保留）按钮保存示例 HTML 页面，我将其命名为 TestPage.HTM 。关闭 ActiveX Control Pad，然后，使用你喜欢的与 ActiveX 兼容的 Web 浏览器（写作本书时，仅限于 Internet Explorer 和使用使用 NCompass ScriptActive 插件的 Netscape Navigator）。打开这个 HTML 页面。如果你使用 Internet Explorer，要保证把浏览器的安全层次设为中等。设置安全层次的步骤是：使用 View（视图）|Options （选项）命令，选择 Security（安全）页，在这个页中单击 Safety Level（安全层次）按钮即可。

注 要保证保存的测试用 Web 页的文件扩展名为 HTM。

装入 TestPage.HTM 文件后，所看到的如图 10.9 所示。现在看上去并不使人肃然起敬，但控件却能做那些标准 HTML 页按钮做不到的事。单击按钮，就会看到标题在 On 和 Off 间切换。你还能监测按钮的模式结果值并确定它处于什么状态，所有这一切，一行 CGI 脚本代码也不用写就做到了！（本章后面将讨论从 ActiveX 控件中得到信息的方法。）

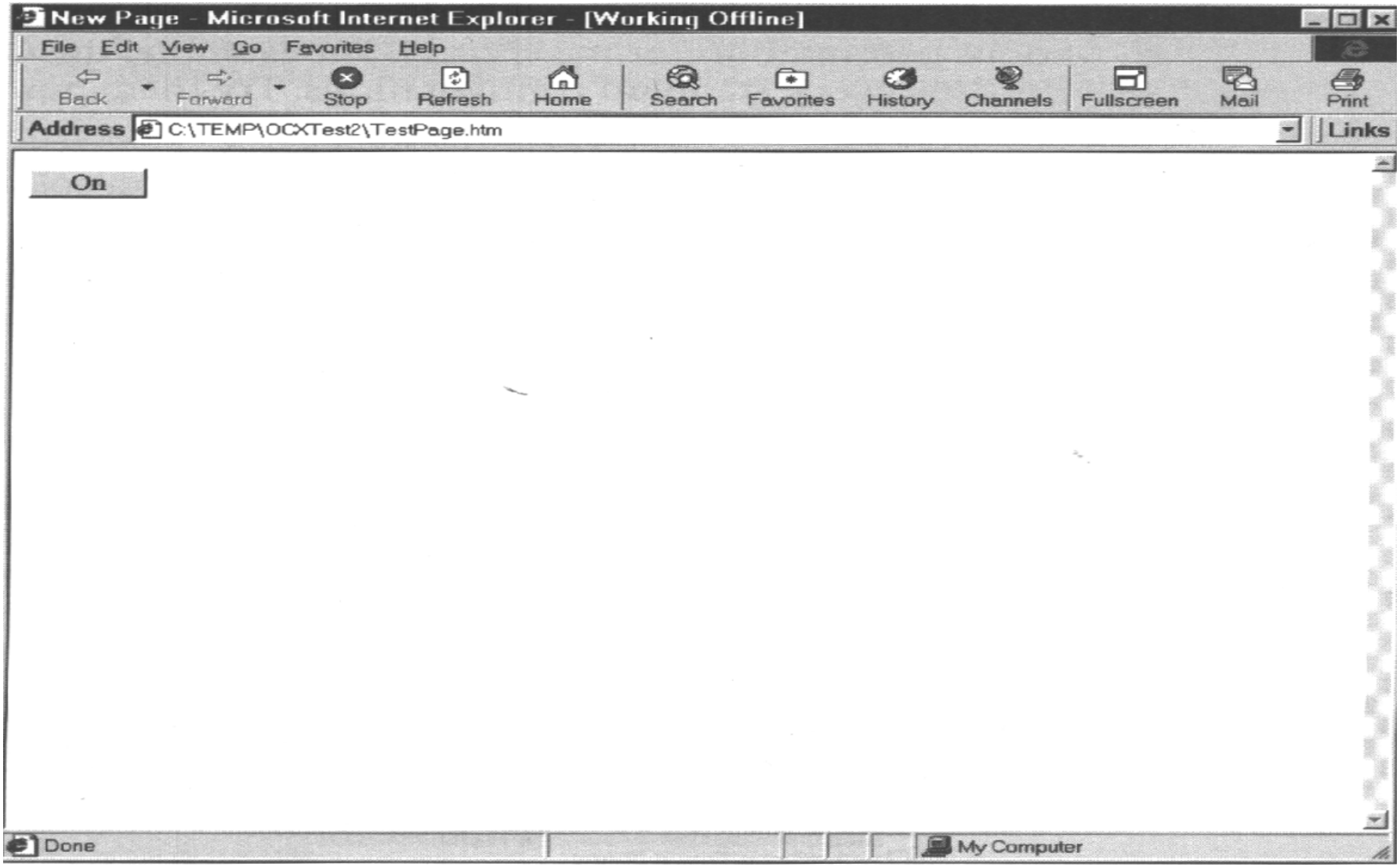


图 10.9　ActiveX 控件在这个页上能做通常控件做不到的事，单击它时，它在两种状态之间切换

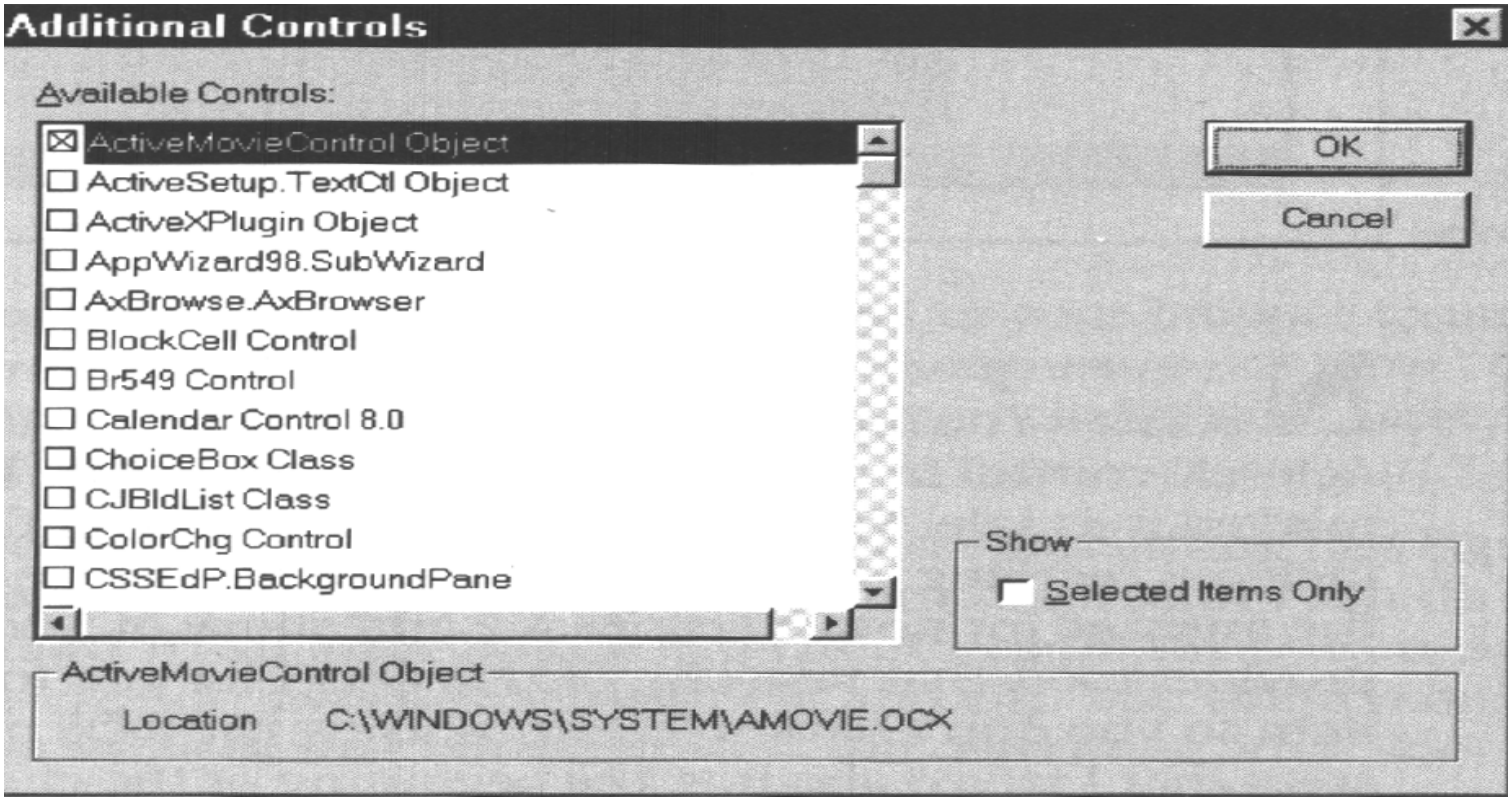
注释 也许你会看到 Satety Violation（安全被破坏）对话框，它告诉你本页的 ActiveX 控件含有不安全代码。这是因为我们还没有把它注册(mark)为安全的。在第 14 章中会学到关于安全性的知识，第 14 章中包括了一旦控件完全测试完成后，怎样把它注册为安全的，现在暂且不用理会这条消息。4.2 版以上的 Microsoft Visual C++ 提供的工具负责处理这些细节（至少提供一个菜单使你不必回到 DOS 提示符下）。本章开头我就说过，可以用旧编译器编译 ActiveX 控件，但新的编译器使得做这类事更方便，上面的情形就是其中的一个方面。

现在，已在 C++和本地 ActiveX 文档中测试了控件的基本功能。在移到网络上去之前，应该试试控件的各种变换，比如，OnOff 控件应能在其它几个标准配置中工作，试一下看看它们是否真能工作。

现在要把 OnOff 控件与其它控件集成在一起使用了。我们在 ActiveX Control Pad 中新建一个 HTML 页。我们先创建一个新的 HTML 框架，然后把 OnOff 控件放入这个框架。使用 File（文件）|New HTML Layout（新建 HTML 框架）命令，创建一个如图 10.10 所示的窗体（这个图还显示了在下面段落中要添加的 On/Off 控件）。就是在这里，你可以拖几个组件来与 OnOff 控件集成在一起。注意到示例框架中为了进行测试，只把几个控件放到了窗体上。在这时，确实不必添加太多的复杂性，只要有足够的信息来保证控件能在一个标准的 HTML 环境中工作就可以了。

注 有趣的是，我们在工具箱中看到的所有控件，实际上都是 ActiveX 控件而不是标准 HTML 控件。

这次添加 ActiveX 控件的过程与以前有些不同。右击 Toolbox（工具箱），就会看到上下文菜单，选择 Additional Controls （附加控件）选项，就会看到如下图所示的 Additional Controls（附加控件）对话框。



Additional Controls（附加控件）对话框包含了你的机器上全部 ActiveX 控件的完整列表（实际上不用管这些控件会不会是 OCX）。还能够选择显示出可插入对象的完整列表。例如，你可以把 Microsoft Word 文档或一个 Excel 电子表格放入窗体。在第 11 章我们将讨论这个特殊过程。

从列表中选中 OCXExmpl 可选项（或者是你命名的控件名称）。要保证确实把挨着这个控件的复选框选中，否则，ActiveX Control Pad 不安装它。单击 OK 则大功告成。你会看到一个新控件添加到了你的 Toolbox（工具箱）中。抓取住它并把它挨着 Command（命令）按钮放下，如图 10.10 所示。和以前一样，如果使用本章中的示例控件，就要保证把它的 OnOff 属性设置成 True。

完成编辑后，保存你的框架，然后关闭本页。ActiveX Control Pad 将向 HTML 脚本添加一个<OBJECT>标记，就如同前面为我们测试的 ActiveX Control 页所做的工作那样。当然，这时的不同点是我们看到了一整页控件，而不是仅仅只有一个控件在页上呆着。保存新的测试页，然后再用浏览器打开它看看控件能不能工作。在这种环境下对控件进行测试后，你就可以使用我们刚才建立的两个页，移兵到网络中进行测试了。

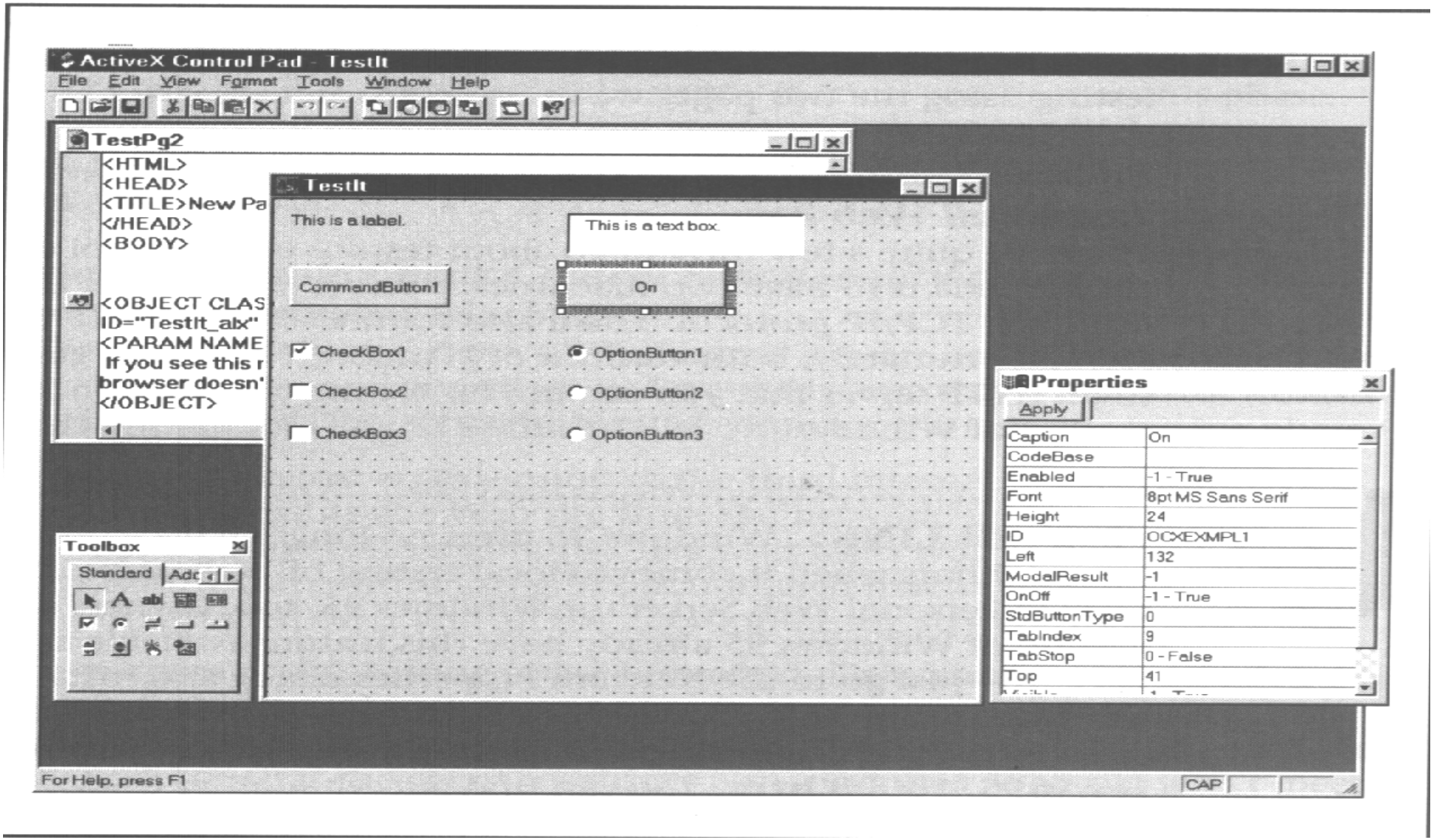


图 10.10　测试 ActiveX 控件的第二步工作是看一下它能否与其它现存的 HTML 控件一起工作

内部测试阶段 3: 使用网络连接测试整个 Web 页

在网络上测试控件的方法相当多，唯一的要求是要有两台机器，一个是服务器，一个是客户机，并使用 TCP/IP 协议将它们连接起来。你还可以在服务器上运行 HTTP 服务程序。幸运的是，Windows NT 4.0 的两个版本都自带有小型的 HTTP 服务器，你可以用它进行测试。事实上，Windows NT 的新版本还将包括 FTP 和 Gopher 服务器。

Web 链接 Windows 95 用户进行 Web 页的本地测试时，不用再羡慕 Windows NT 邻居了。现在也可以得到 Windows 95 的个人 Web 服务器了。Windows95 的 OSR2 版的拥有者已把这一特色作为软件包的一部分使用了，这只需打开 Network Properties（网络属性）对话框，并安装它就可以了。如果你的 Windows95 是老版本，你可以从 http://www .microsoft.com/msdownload/ieplatform/iewin95/10000.htm 下载所需的支持。在 http://www.microsoft.com/windows/pr/win95osr.htm 可以发现为 Windows95 可以下载哪些 OSR 功能。

你要保证能用标准浏览器和你指定的域名与服务器通信。不要错误地认为，可以通过像浏览器（如 Explorer）这样的东西访问服务器，然后双击测试页。不错，这将打开浏览器，你将看到你的测试页。但是，你将使这一测试阶段的目的遭受挫折！通过 Explorer 或文件管理器(File Manager)打开这个页，将使浏览器置于文件模式，它不允许你测试 HTTP 服务程序的能力。这一测试阶段的

目的是，尽可能紧密地模拟典型用户访问你的站点时的环境。

一旦与服务器建立了链接，就能看到它提供的缺省的 Web 页。在其中添加一个到你的测试页的链接，使用这个链接就能测试前一节中创建的单个控件和整个 HTML 页。多数情况下，你会发现，对于测试的这个阶段而言，简单地测试一下链接就很充分了。换言之，如果控件在一种模式下能工作，那么，它就应该在你提供给它的全部其它模式下也能工作。例如，在我们的示例控件中，我把它作为一个 on/off 开关进行简单的测试，那么，标准按钮模式也应工作得很好，因为，在测试的前两个阶段，我已经彻底地测试过它们了。这正是本地测试之所以重要的原因，它能在这个阶段节省时间。（想象一下，不得不反复创建同一 Web 页的几个版本，并在服务器连接上测试它们，对于程序说来，这是多么没有效率的方法呀！）

外部测试：在未被污染的机器上检查控件的功能

在我们的私有环境中，已对控件进行了彻底的测试。考察了每种控件访问，我们确信控件可以与标准的 HTML 控件一起工作。现在该进入外部测试阶段了。

实际上，这一阶段仅仅是第三个内部阶段的扩充而已。但是，不再是从网络上访问 Web 站点，而是像你的用户那样，通过电话连接来访问站点。另外，你对控件进行测试一直使用的是受了污染的机器，在这种机器上，已安装了被测试控件，而且，拥有了所有适当的注册表项。

这个测试阶段的目的分为两个方面。第一方面，想确信，当从一个未被污染的机器上进行访问时，控件仍能工作；否则，就会发现你的控件依赖于某些注册表项，或者依赖于客户机访问你的 Web 站点时不能满足的其它要求。第二，

想检测你的 Web 页的下载时间，到现在为止，我们一直通过高速连接访问测试页，而使用你的站点的人就没有这种便利了。他们中的大多数人是通过拨号连接来访问你的站点的。

技巧 可能你想把你的测试页安装到你的 Web 站点中一个私有部分，从而防止其它人访问他们。与网络测试阶段不一样，不要在你的主 Web 页上放上对测试页的链接，因为你已经在以前的测试阶段中测试过连接性能，所以可以用直接地址访问测试页。

为了完成这一部分的测试工作，把测试页移到你的 Web 服务器的一个私有部分。从一个未被污染的机器上，使用一个支持 ActiveX 的浏览器，拨号进入你的 Web 服务器。你需要使用你的网络的私有部分的 URL 而不是主页的 URL 。先试一试单一控件测试页，看看 ActiveX 控件是否能工作，还要看一看在没有一个整页实现的影响时下载控件用多长时间。一旦检测完单一控件测试页，就转向使用整个 HTML 框架的第二页。若这个连接成功，你就获得了一个可以工作的 ActiveX 控件，这个控件能够用于增强你的正规 HTML 页面的功能。

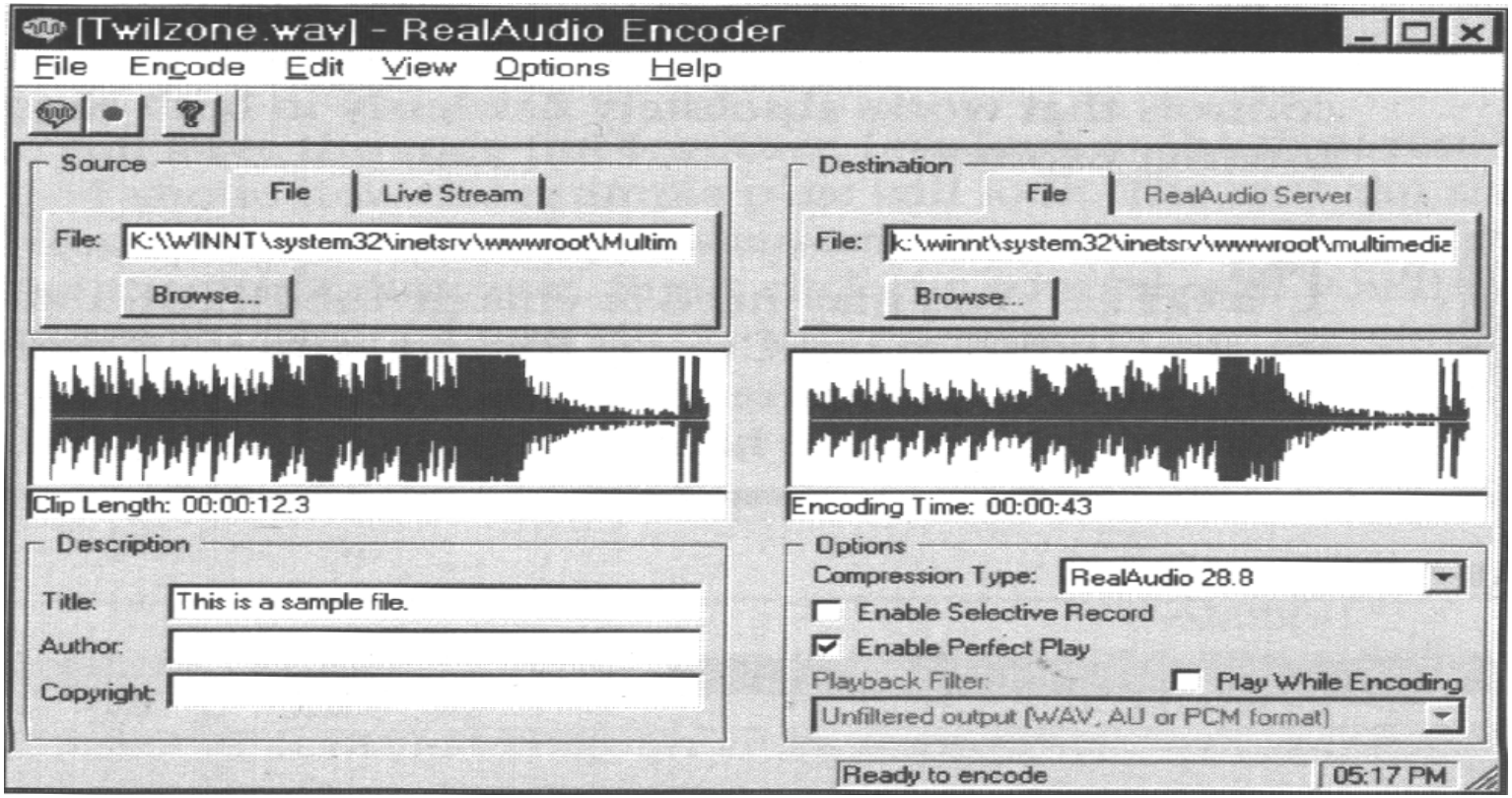
10.3 在 Netscape Navigator 和 Internet Explorer 中使用 ActiveX 控件

演奏音乐以及其它形式的音响效果，大概是使 Web 页更具吸引力的最普通的方法。RealAudio 在提高你获得的声音质量时，至少使得这个过程容易些。

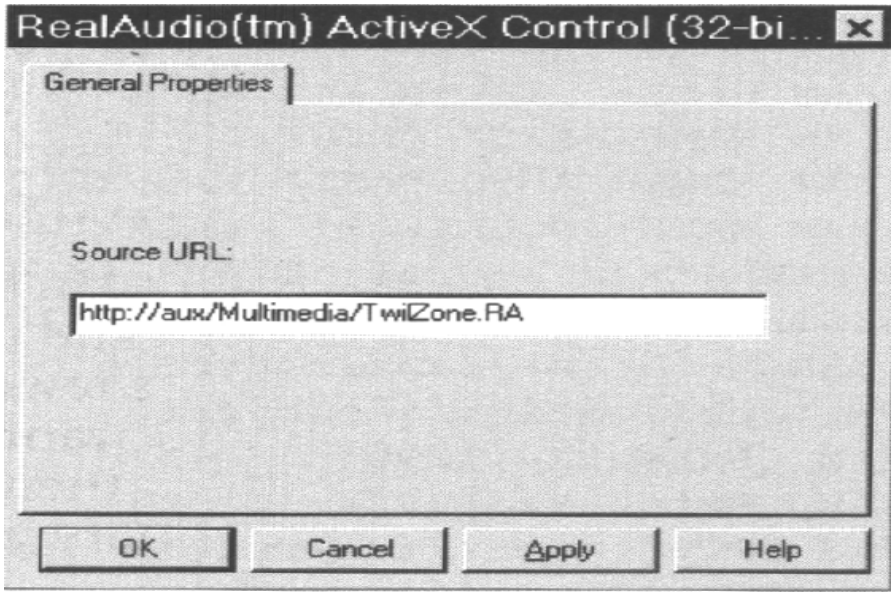
RealAudio 有好几种不同形式，而不仅仅是一种 ActiveX 控件。这个多媒体的当前版本还可以做为 Netscape Navigator 的插件。在 http://www.realaudio.com/可以与 RealAudio 的制作者联系，从而了解播放器、SDK 和支持的实用程序的详细情况。SDK、RealAudio Encoder（编码器）、RealAudio TimeLine Editor（在线编辑器）、RealAudio Personal Sever（个人服务器）以及 RealAudio Content Creation Guide（内容创建指南）是可以免费下载的。RealAudio Personal Sever 支持两种数据流——这对于用于演示和试验目的来说是太多了点。

这个播放器提供的多层次支持，只是使用这一特定技术的原因之一。相当多的站点都使用了它。最有趣的站点是 http: //www. cdnow. com/ （它与 Magazine Warehouse 相连接）。这是一个使用多媒体效果的 CD/movie（电影）商店。例如，如果你打算买一张 CD，那么在购买之前可以先在你的 PC 扬声器上听一听所选的乐曲片段。它还卖 T 恤衫，你可以在购买之前先看一看货。唯一做不到的是不能在购买电影之前先看一些电影片段，但几乎可以肯定的是它最终会添加这一功能。

启动 RealAudio 不成问题，要做的只不过是安装服务程序，对你的文件编码，然后安装 Web 页。RealAudio Encoder 实用程序对话框如下图所示。只需选择将要编码的文件，在 Description（说明）区键入所需的信息，然后单击 Encode 按钮对文件编码（RealAudio Encoder 实用程序自动地为你指定一个目的文件名）。



把这个控件添加到 Web 页也相当容易。在 ActiveX Control Pad 的 Insert ActiveX Control（插入 ActiveX 控件）对话框的控件列表中，选择 RealAudio 控件——以前我们所做的也是这样一个过程。除了 CodeBase 这一个属性需要改变外，其它需要改变的属性如下图所示，把声音文件所在的 URL 打入即可。



程序列表 10.7 是这个示例的代码。注意它包含了 NCompass ScriptActive 所需的<EMBED>标记。它是在 Netscape Navigator 和 Internet Explorer 上都能无毛病地进行工作的少数控件之一。

注释 也许需要改变一下 RealAudio1<OBJECT>标记中 HEIGHT 和 WIDTH 属性，使它能在你的特定的显示配置中工作。当前值代表的是缺省尺寸，也许有些人会发现在高分辨率显示中它太小了。

程序列表 10.7

```
<HTML>
<HEAD>
<TITLE> RealAudio Test </TITLE>
```

```
</HEAD>
<BODY>

<!-显 示 标 题 ->
<CENTER><H2>Test of RealAudio ActiveX Control </H2></CENTER>

Click here to test the control:<BR>
<!-显 示 控 件 ->
<OBJECT ID="RealAudio1" WIDTH=49 HEIGHT=39
        CLASSID="CLSID:CFCDAA03-8BE4-11CF-B84B-0020AFBBCCFA"
        CODEBASE="http://aux/controls">
        <PARAM NAME="_ExtentX" VALUE="1296">
        <PARAM NAME="_ExtentY" VALUE="1032">
        <PARAM NAME="SRC" VALUE="http://aux/multimedia/TwilZone.RA">
        <PARAM NAME="AUTOSTART" VALUE="0">
        <PARAM NAME="NOLABELS" VALUE="0">
<EMBED NAME="RealAudio1" WIDTH=49 HEIGHT=39
        CLASSID="CLSID:CFCDAA03-8BE4-11CF-B84B-0020AFBBCCFA"
        CODEBASE="http://aux/controls">
        TYPE="application/oleobject"
        PARAM__ExtentX="1296">
```

```
        PARAM__ExtentY="1032">
        PARAM__SRC="http://aux/multimedia/TwilZone.RA">
        PARAM__AUTOSTART="0">
        PARAM__NOLABELS="0">
</OBJECT>
</BODY>
</HTML>
```

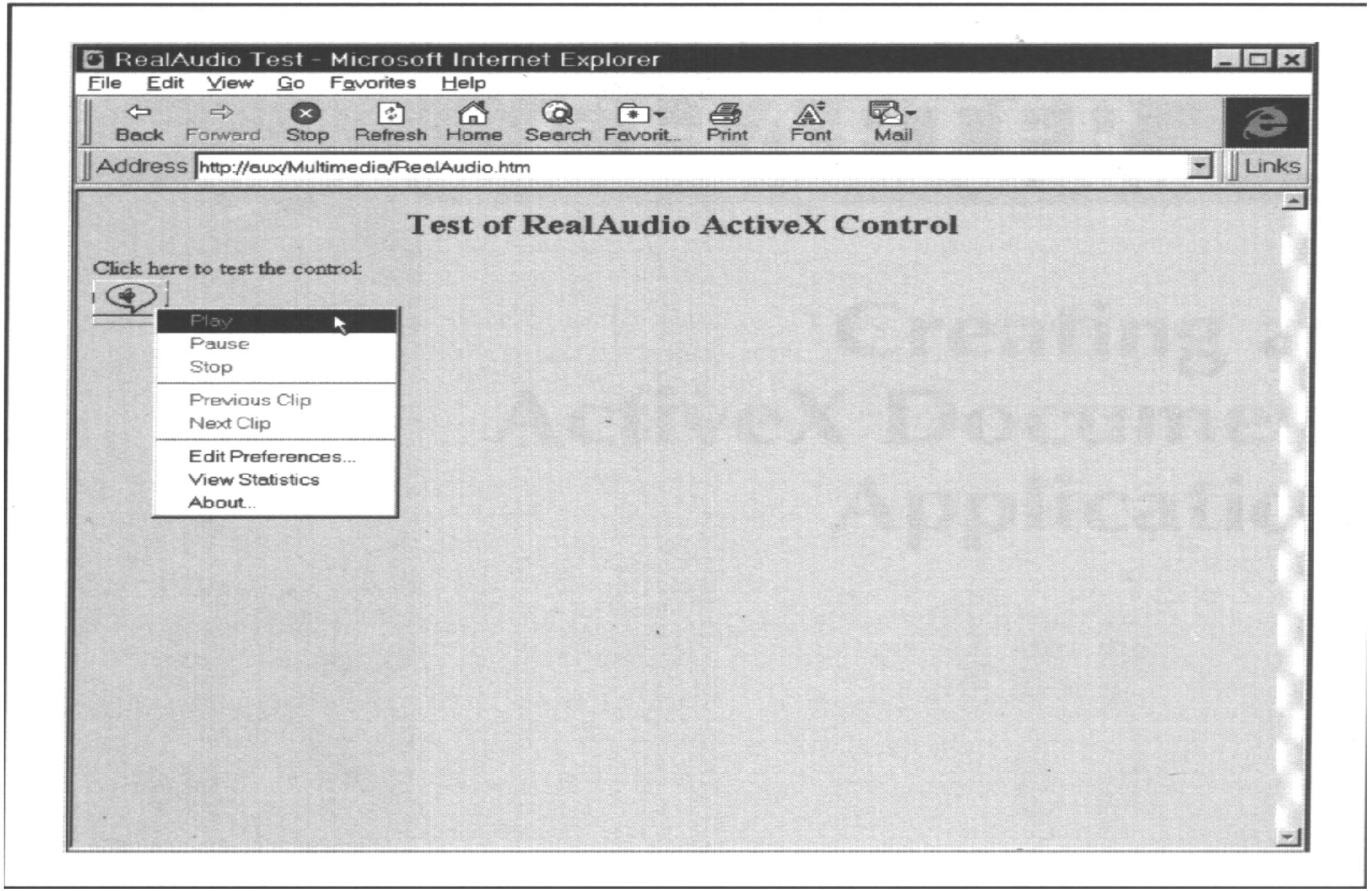
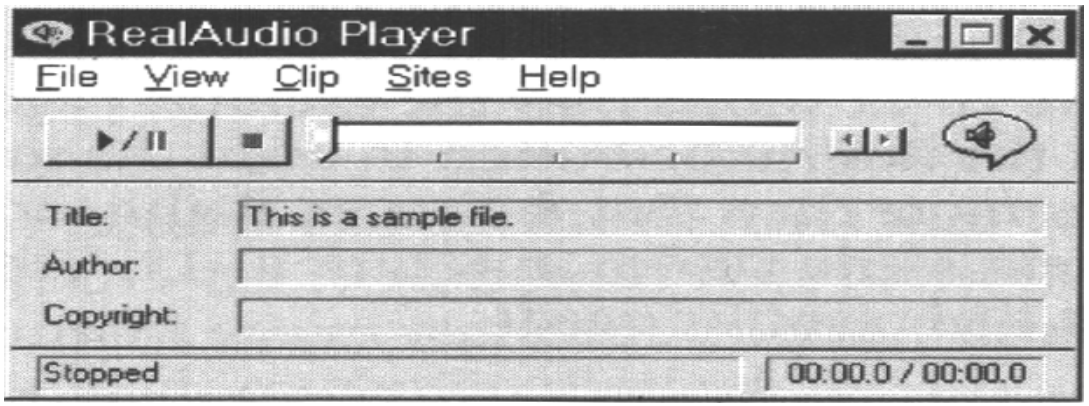


图 10.11　RealAudio 控件提供了独特的按钮接口以及关联菜单供用户进行设置

现在把代码写好了。来看一下最后结果吧。图 10.11 显示的是这个例页的外观。用户只要单击 RealAudio 按钮就可以打开声音文件。这个控件还有别的功能。右击该控件，就会看到如图 10.11 所示的上下文菜单，用户可以定制控件以满足专门需要。

那么，该控件操作起来是什么样子呢？下图显示了用户眼中示例 WAV 文件的显示情况。



请注意，这个对话框中显示了在 RealAudio Encoder 实用程序中添加的说明信息。遗憾的是，你添加的信息量受到限制。最好除了控件所提供的缺省域外再添加一个用户可配置的域。

第 11 章　创建 ActiveX 文档应用程序

到目前为止，我们已讨论了各种新技术，包括 ActiveX 和 COM+，这些技术允诺要把 Internet 变成每个人的业务工具。使用 ActiveX 的方式多种多样（COM+ 在将来会用到）。例如，Microsoft 发布了一个称之为 ActiveX Accessibility 的 ActiveX API。这个 API 设计用于向 Internet 添加那些已添加到 Windows 95 中的 Accessibility applet（含有内容的小程序）。类似于 Sticky Keys（一种每次按一个键而不是同时按住所有键的方式来创建 CTRL 组合键的方法）这样的方法，不仅出现于 Windows 95 和 Windows NT 4.0 桌面系统，而且正在出现于与 ActiveX 兼容的 Internet 浏览器中。

还有其它一些（由 Microsoft 和其它公司开发的）基于 ActiveX 的技术已经投入使用——这样的技术实在太多了，这里难以一一罗列。更重要的 ActiveX 技术之一是 ActiveMovie，事实上，MSNBC Internet 站点已经使用 ActiveMovie 提供 NBC 新闻网络的新闻剪辑了。实际上，ActiveMovie 使你能在 Internet 上看电影剪辑、放映 AVI 和其它种类的电影文件。

注　ActiveX 提供了把令人激动的新技术从桌面系统转移到 Internet 的手段。

WEB 链接 刚才讨论的新技术可以查到更多的可用信息。可以在 http://www.microsoft.com/enable/dev/msdn4.htm 中找到 ActiveX Accessibility。ActiveMovie 控件的使用说明和技术手册出现在 http://www.microsoft.com/msdn/sdk/inetsdk/help/complib/activemovie.htm 中。

最重要的 ActiveX 技术并不是新颖而令人激动的技术，而是涉及到一个老生常谈的问题，即在 Internet 上共享文档。因为接口是静态的，这确实一直是一个大问题。读过第 8 章后，考虑一下为了创建任意种类的动态接口所需的程序复杂度和所需的标记数量，就会认识到这一点。是的，你能把字处理文档结合一些现实的数据嵌入到 HTML，但是数据并未变化，用户还是难于编辑它。对于电子表格也是一样。显示数据一般说来并不太难，只要你肯于牺牲最新信息。ActiveX 提供了解决办法，它允许你创建动态文档，你可以编辑它并且看到它的实时变化。

那么，这一技术是从什么地方神秘地冒出来的呢？其实它一点也不新颖，Microsoft 只不过简单地修改了 Microsoft Office 中的技术而已。ActiveX Document 原来的名称为 OLE Document Objects（文档对象）（大多数人，包括 Microsoft 的人，已把 ActiveX Document 简记为 ActiveDocument，本书中我们将一直使用这个术语，原因在于它既清楚又简单）。它是 Microsoft Office Binder 技术的一部分，从未打算成为一个公开的规范。最初甚至在能得到 OLE Document Objects 规范之前，必须申请 Office 兼容的程序。只是到了 Windows 95 出现之

后，这一规范才成为每个人都能得到的东西。有迹象表明，Microsoft 会把这一规范公开的，原因在于它是 OLE 演变中的下一个逻辑步骤。

本章的中心内容是 ActiveDocument。我们要考察一下，除了数据应用之外，什么是 ActiveX 的最重要的用途。另外你还将看到，创建浏览器使用的 ActiveXDocument 是多么容易。另外，本章还将介绍一下创建你自己的定制文档的一些设置时，你需要做些什么工作。

注 你可能会听到有人将 ActiveX 文档称为 OLE 文档对象，另外有些人将它们称为 ActiveDocument 文档。

本章将介绍三个主要的 ActiveDocument 主题。首先让我们看一下 Web Publishing Wizard。如果你不了解从用户角度看 ActiveDocument 是如何工作的，那么你应该密切关注这一节的内容了。总而言之，怎么能期望写出你自己都不理解的应用程序呢？接下来我们考察一下 ActiveDocument 的理论基础，最重要的是你要实现的接口。最后，我们创建一个简单程序，它实现了 ActiveDocument 所需的接口。

注释 本章的例子全部采用 Visual C++ 6.0，但是，除了一些小的程序性变动之外，也可以容易地使用 4.2 以上版本的 Visual C++（代码本身应该工作的一样好）。你必须拥有 Visual C++ 的某种版本（推荐 6.0 版），以便于从头至尾地学习本章的例子。也可用 Visual C++4.1 来学习这些例子，但你在工作中却要尽量少用为好。例如，可能需要添加对 Visual C++4.2 以上版本作为缺省支持的各种新类的引用。另外，创建例子时，也不如使用新一些的产品那样，自动化程度高一些。

理论上，使用 Visual C++4.0 创建这些例子也可以。但所需的工作量很大，无疑使用该编译器的升级版显得更容易些。

11.1 什么是 ActiveDocument（OLE 文档对象）

到现在为止，对于用户在 Internet 比较熟悉的文档种类，我们还没有进行太多的讨论。例如，用户非常熟悉刚在本地硬盘上修改过的 Word 文档，但他们在 Internet 上使用真正干练的 ActiveX 控件却是不完全熟悉的。对于使用只有程序员喜欢的方式来显示专用页元素的脚本语言，我们也要讨论（用户大概不会全都对创建 JavaScript 的最新技术感兴趣）。当然，这就会带来差不多每个人都会问到的问题：对于用户而言，除了一点点信息、几个窗体以及那眼花缭乱的 ActiveX 控件，Internet 到底是个什么玩艺儿？那么，ActiveDocument 为你提供一个答案，它们是对每个人说来都可用的在 Web 页上创建内容并显示数据的一种手段。

但是，ActiveDocument 却远远不止是共享信息的一种方式，你可以使用它们正儿八经的干点事。考虑一下这种场景吧：你的公司拥有自己的内部网，公司的雇员们需要时可以在世界上任何地方进入这个网，他们使用这个网获得自己的 e-mail 并完成各种各样的其它任务。现在你的老板告诉你，要求市场开发部提交报告，而这时大部分销售代表都在外工作着，你怎么样才能完成这一工作呢？

注释 Netscape Navigator 和早期版本的 Internet Explorer，通过使用帮手(helpers)让你看到非标准文档。浏览器启动应用程序的一个完整拷贝，并把 Web 站点上的文件内容传递给它。这种办法的问题是，启动另一个应用会使用更多的内存，而且你与 Internet 服务器也没有现场连接，所以对文件所作的改变不会立即反映到服务器中去。尽管 Microsoft 在 Internet Explorer3.0 及以上版本中仍然使用“帮手”(helpers)一词，但从用户的观点来看两者的意义却是完全不同的。

在这种情况下，ActiveDocument 就是答案。使用第 8 章中讨论过的标记，在 HTML 文档中建立到该文档的链接。当用户单击文档引用时，他们在浏览器中看到的是文档的可编辑版本，这个浏览器的菜单和工具栏发生了一些变化。图 11.1 中显示的是在 Internet Explorer 浏览器窗口中显示的 Word for Windows 文档。请注意浏览器所有特色仍然完整无损，变化了的只是文档显示的方式。这种特定技术并不是什么新货色，它被称之为在位编辑(in-place editing)。大多数 OLE2 服务器现在都能向 OLE2 客户机提供这种功能，你可以试一下 Word for Windows 或 CoreDRAW！单击 OLE 对象时，菜单和工具栏会随时改变得与客户机上的菜单和工具栏匹配。最大的差别在于，这是跨越 Internet 连接在浏览器中正在发生的事。

注 ActiveDocument 使用称之为在位(inplace)编辑的 OLE2 特色，它让你在客户端中编辑文档而不是作为一个独立的服务器进行编辑。

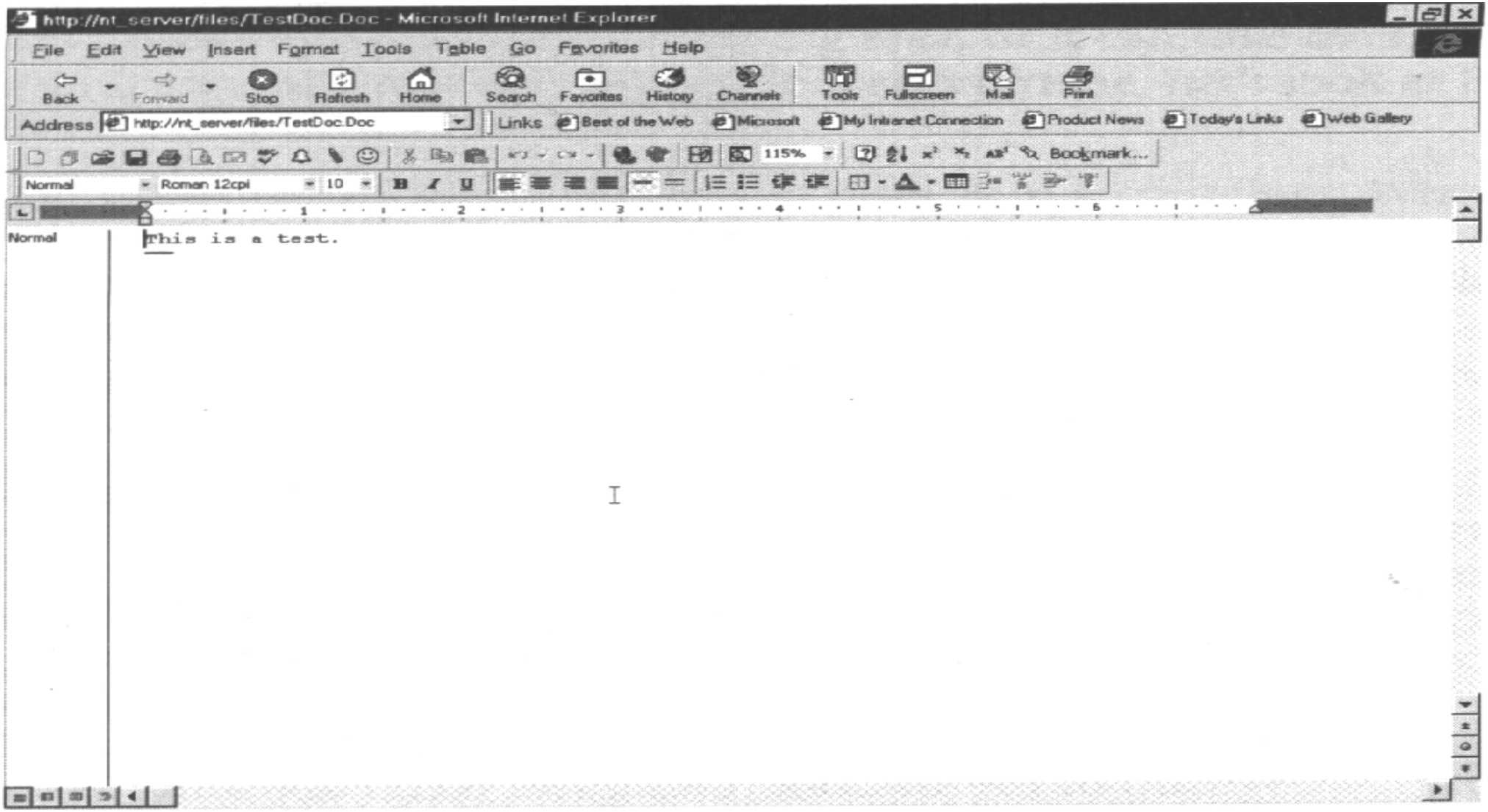
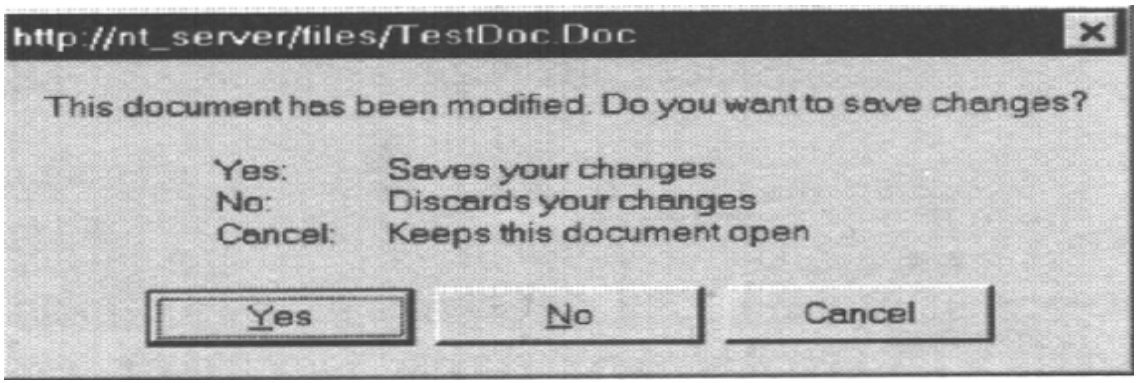


图 11.1　Internet Explorer 4.0 显示了在位(in-place)编辑的效果，这是 COM 的 OLE2 特色

注释 如果在基于 LAN 的 Internet 服务器上测试这个示例的话，可能会出现一个对话框，问你是否想打开或保存文件。多数情况下，在一个现场(live)Internet 连接的连接中测试时，肯定会出现这个对话框（取决于你是如何设置浏览器安全性的）。简单打开文件，就会看到如图 11.1 所示的效果。

你还会注意到另外一个差别。如果改变了这个文本的内容，然后在 Internet Explorer 浏览器工具栏上单击 Back 按钮，就会看到如下图所示的对话框，它询问你是否想改变文件。如果单击 Yes，就会看到典型的 File Save（保存文件）对话框。遗憾的是，使用这一技术保存文件将把文件保存到你的本地硬盘中——在我们这里这项特性没有什么用处，但也许在别的地方有用。还有其它保存文件的方法，可以使用 File（文件）|Save （另存为）命令，你还能够使用 File（文件）|Send To（发往）| Web Publishing Wizard（发布向导）命令，真正地将文件送回 Web 服务器。这正是我们所要的。（Internet Explorer 与 Windows 的某种组合不提供 File|Send To|Web Publishing Wizard 命令，这就是说，必须先把文件存在本地硬盘，然后再用 Web Publishing Wizard 命令将它发往 Web 服务器）。本章后面一节“使用 Web Publishing Wizard”中，将讨论使用 Web Publishing Wizard 所需的步骤。



技巧 对某些情况来说，本节讨论的每种技术的帮助都不太大。例如，你可能想向 Web 服务器邮寄文档，而且希望每个邮件单独到达。这时，你会告诉用户使用 File|SendTo|Mail Recipient（邮件收件人）命令。显然，这就意味着你需要整理所有收到的文档，但是，当其它方法无效时，这种方法却能够使用。

技巧 当前的 HTTP1.0 规范不允许用户向 WWW 服务器发布文档，新的 HTTP1.1 规范修正了这一点。如果你希望允许用户从 Web 页面上向服务器发布文档，就应提供一个使用 1.1 规范的服务器。因为不必编写通常的脚本或者在连接中跳来跳去，所以你不仅可以节省时间和精力，而且，可以减少电话支持数量，从而对用户而言事情也就更简单了。（Netscape 和 Microsoft 产品的所有当前版本都支持该 1.1 规范，如果你拥有任一家供货商产品的老版本，可以查看一下自己的文档。）

ActiveXDocument 的变化也使得程序员必须考虑对 OLE 的支持。在客户机有三级对象参与。对象可以简单地出现在查看区，也可以占据整个窗口，还可以占据整个应用框架。现在来比较一下这三级对象的不同之处。

以前，你可以创建一个对象并将它放在容器中。OLE1 出现后，容器只是简单地显示一个图标，说明对象的存在。如果希望编辑对象，可以双击容器中的对象图标。Windows 将在另一个窗口中形成该文档的完整拷贝。这正是某些人所说的对象简单地出现在浏览区的意思。

OLE2 改变了客户机和服务器交互的方式。现在，你可以真正看到对象的内容了。例如，如果在字处理文档中放入一个电子表格对象，不必双击该对象就能看到它的内容。它就是窗口级的参与者。客户机和服务器共享窗口；客户机显示它的数据，然后依靠服务器显示对象内部信息。OLE2 还提供了在位替换操作。大部分情况下，双击对象都会启动一个进程之外的(out-of-process)服务程序，它占据整个客户机框架。服务器占据了一般留作客户机使用的菜单和工具栏。从用户角度来看，应用程序是一样的；只是改变了工具，以满足用户编辑对象的需求。

注 当你创建了一个与外部文档的连接时，OLE1 只是显示一个图标；OLE2 则显示真正的文档内容并允许进行在位编辑。

通过使用浏览器，ActiveDocument 将该技术从桌面系统扩展到了 Internet 上。现在浏览器工具将会自动更新，以适应用户的需要，用户不必再打开一个其它的程序来编辑文档。一个进程之外的服务程序将负责更新浏览器菜单和工具栏，以适应应用程序的一般需要。实际上，Windows 的未来版本将会做得更好。

注　当你单击窗口应用程序中的对象时，进程外的服务程序实现你看到的菜单和工具栏的修改。

Windows 95 和 Windows NT 4.0 使用的都是资源管理器式界面。如果你用右键单击与某个已注册的应用程序相关的文档，就能看到一个有多种选项的菜单。最常见的选项是打开或打印文档。大部分情况下你能看到使用 Quick View 实用程序浏览文档内容的选项。

用户界面在不久的将来也可能有所改变。双击文档能进行在位操作。服务器将真正接管 Explorer 的菜单和工具栏。用户再也不必离开 Explorer 去打开另外一个应用程序窗口。另外，Internet 站点就像硬盘一样出现在 Explorer 中，并组合了目前的浏览器的功能。

程序员们很欢迎这些变化吗？并不全是。有些人已经在抱怨该技术只是加强了 Microsoft 对计算机世界的控制。当然，只有在你不能安装另一个服务器来替代 Internet Explorer 时才会发生这种情况——这几乎是不可能的事。随着计算技术日趋成熟，ActiveDocument 将会成为一种特别重要的技术。这也正是本章为什么重要的原因——本章的目的正是使你跟上这种日新月异的技术的发展，以便你可以使用它来满足目前的计算需求。

创建连接

现在，你已经了解了使用 ActiveDocument 的结果，并讨论了该技术为什么重要的原因，本节将介绍实现它的 HTML 代码。程序列表 11.1 显示了用于创建该示例的代码，这个示例虽然很短但很实用。它没有增加任何特别功能，从而

使你可以看到创建 ActiveX 文档链接的最小需求。可以看出，代码中只是使用了一个简单的链接。所有该应用程序后面的“魔法”都存在于 Internet Explorer 中。第 8 章已经用这种链接来显示 Web 站点上的其它页了。

程序列表 11.1

```
<HTML>
<HEAD>
<TITLE>ActiveX Document</TITLE>
<BODY>
<CENTER>
<H2>ActiveX Document Test Page<H2>
<EM>Requires Microsoft Word or WordPad</EM><P>
</Center>
<A HREF ="http://nt_server/files/TestDoc.Doc">Test Document</A>
</BODY>
</HTML>
```

好了，现在浏览器中已经有了一个可以编辑的文档了。这确实不是一件特别困难的事情。由于文档仍在浏览器中。所以可以节省内存。只有一个应用程序处于运行状态，所以尽管实际中要花费一些额外的处理时间和内存来浏览及处理文档，但比起运行两个应用程序来说，这样做的花费要少得多。例如，在位激活功能这一特色就是使用进程外服务程序的结果。进程外服务程序实际上

是 DLL 的一种“漂亮”的形式，它能提供与客户机应用程序进行通讯的适宜接口。关键在于：DLL 比完整的应用程序占用的内存要少得多，并且也没有理由担心什么显示问题（这是客户机的工作）。本章“ActiveDocument 结构概述”一节将介绍这种 DLL 的一些需求情况。

注 ActiveDocument 使用进程外服务程序（通常是一个 DLL）来实施在位编辑。

还有两种其它的方法可以创建 ActiveDocument 连接，不过本章将不用多少时间来讨论这些方法。Microsoft Web Browser Control 允许你浏览 Internet 上任意类型的文档，包括那些一般与 Internet 并无关系的文档，如 Word for Windows 文档。另外，在这方面还有一些高级的<OBJECT>标记属性能提供很大帮助。你可能要花费一些时间去了解第 8 章中有关<OBJECT>标记的知识，然后就可以到所列的 Web 站点上去下载有关的技术规范。实际上<OBJECT>标记的高级属性目前正处于不断发展变化中：这也正是本章不作介绍的原因。

使用 Web Publishing Wizard

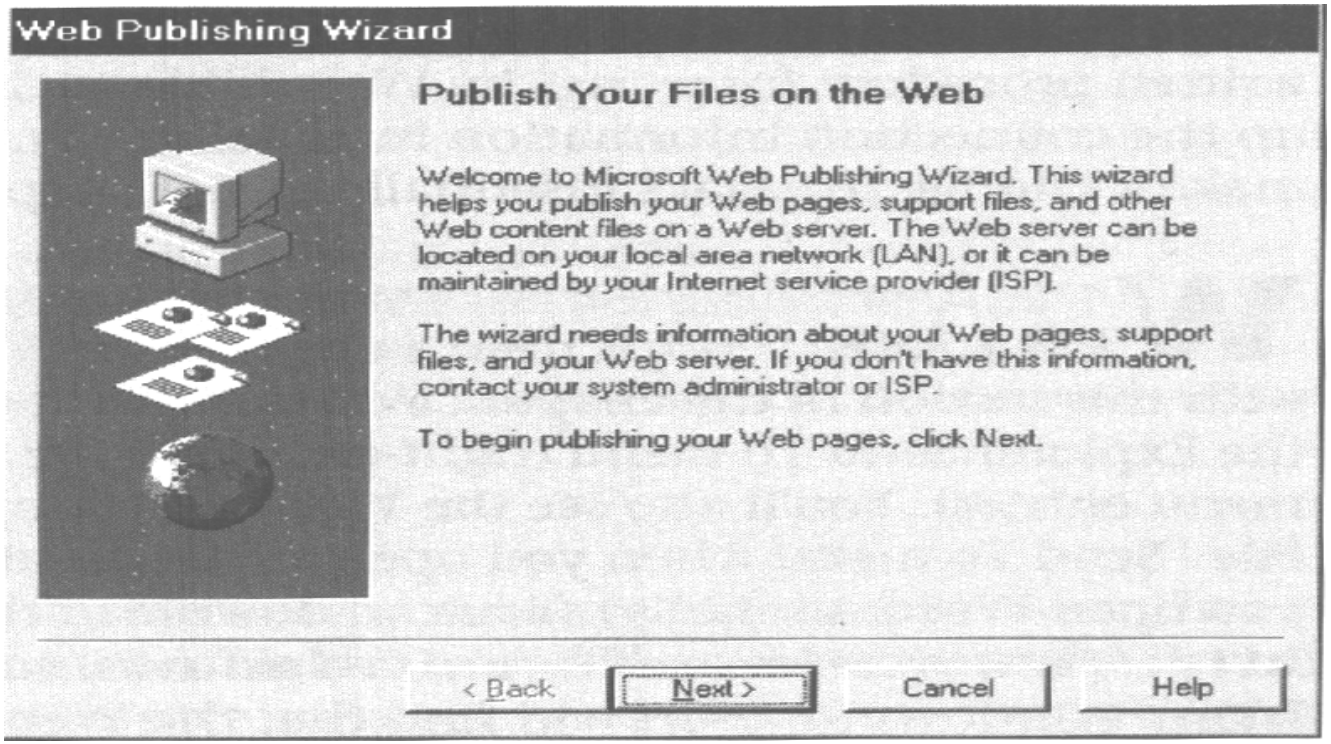
你不必依赖那些老方法来保持文档为最新状态。利用 Web Publishing Wizard，只要稍作努力就能使 Web 服务器上的文档保持最新状态。现在对文档的更改将不会像原来的浏览器技术那样只出现在自己的本地机器上。实际上它们会出现在 Internet 服务器上。由于提供了使用 Web Publishing Wizard （以及可能的高级的设置连接信息）的已经编写好的过程，因此，就可以允许雇员从甚至很远的地方对 HTML 页进行修改了。

Web 链接 学习本章的这一节时，你需要有一份 Web Publishing Wizard 的拷贝。你在“Explorer Send to”菜单上应该能够看到“Web Publishing Wizard”（右击某文件，然后在上下文相关菜单中查找）。另外，如果你在浏览器中打开一个文档，那么在 Word 的“File|Send To”菜单中也能看到“Web Publishing Wizard”项（但正常打开 Word 时不会看到这个菜单项）。你可以从 http://www.microsoft.com/windows/software/webpost/default.htm 下载 Web Publishing Wizard。Windows 98 用户在添加/删除程序属性对话框的 Windows Setup 标签中，可能会发现 Web Publishing Wizard 出现于 Internet 文件夹中。Web Publishing Wizard 也是作为不同的 Internet 编程工具的一部分由 Microsoft 提供安装的，所以在安装之前应看看 Web Publishing Wizard 是否可用。

那么，怎么开始呢？下面的过程将帮助你进行第一次的实践。第一次尝试之后，你可以将它们总结为简单的四步过程，一旦通过第一阶段后，我们将看一下这个过程的部分内容。我们先从图 11.1 所示的文档起步，并且假定该文档已经编辑过了。现在要做的是将所做的修改保存到 Internet 站点上去。

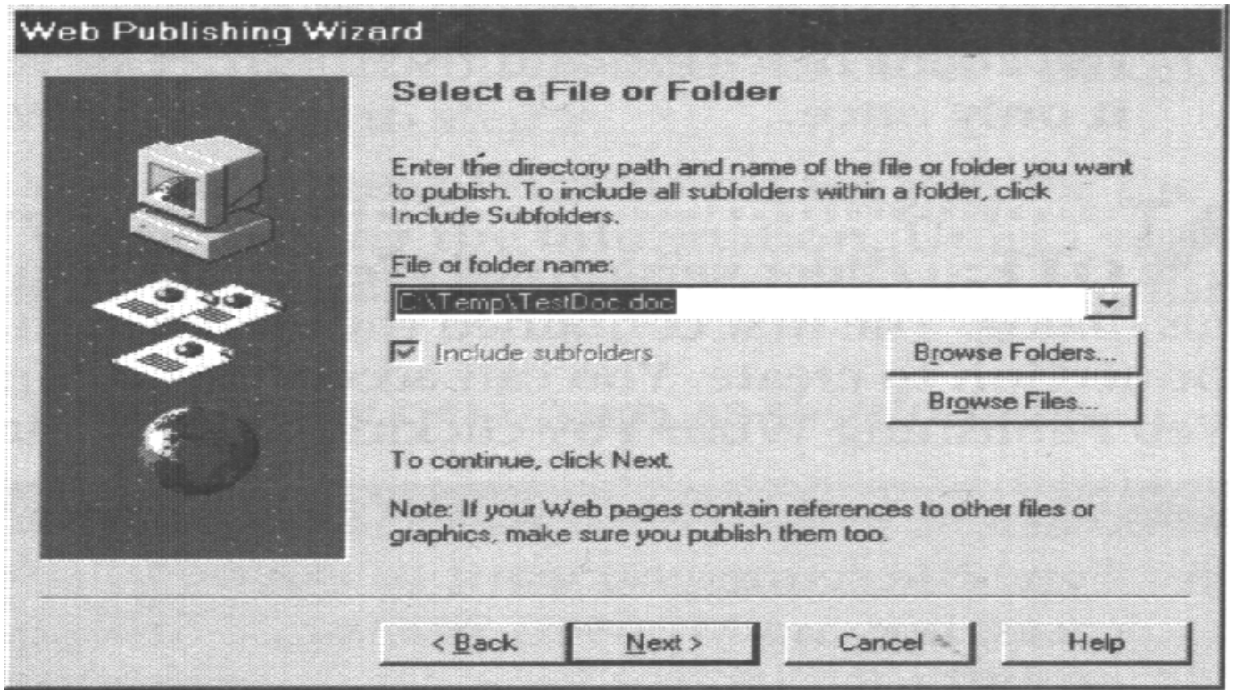
1. 使用“File|Save As”命令显示“File Save”对话框。在将文件发送给 Web 服务器之前，必须先在本地机上保存它。可能以后 Microsoft 会修改这个过程，但现在你还是需要花点时间做一个本地拷贝。将文件命名为 Web 站点页的名字。在本例的 Internet 站点中，文件名为 TESTDOC.DOC（如图 11.1 中 Internet Explorer 的标题栏所示）。

2. 在 Word 或 Windows Explorer 中使用 File|Send To|Web Publishing Wizard 命令来显示 Web Publishing Wizard 对话框（如果使用 WindowsExplorer 发送它，要首先选中文件或文件夹）。



3. 单击 Next，如果你使用的是 Web Publishing Wizard 的老版本，或者从开始菜单启动 Web Publishing Wizard，你将看到下一页，如下图所示。在某些情况下，Web Publishing Wizard 将自动把一个缺省的文件名放在 File（文件）或 Folder（文件夹）域中。现在，你还用不到该文件名。（如果你使用的是新版本的产品，那么在启动 Web Publishing Wizard 之前，就要选择一个文件，你就会看到一个

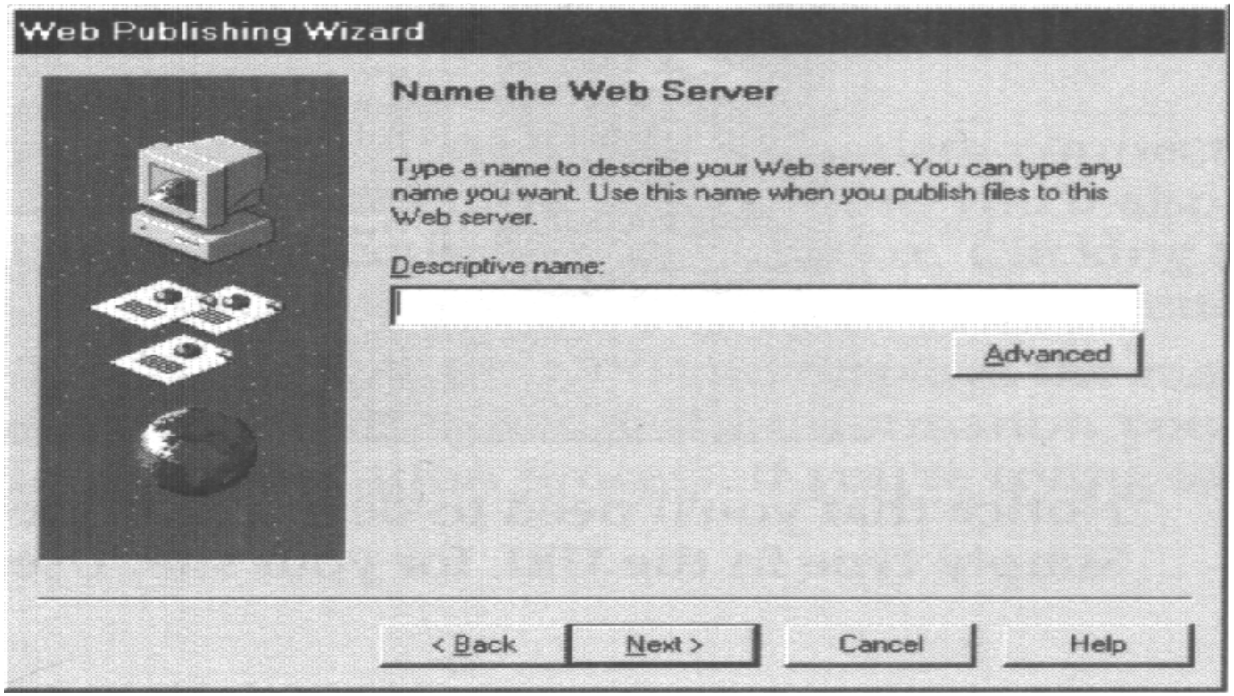
对话框，询问你将使用哪一个 Web 服务器连接，这时请跳过第 5 步）。



4. 单击 Browse Files（浏览文件）按钮，系统显示 Browse 对话框（它的外观与标准的 File Open 对话框相似）。找到刚才保存的文档的本地拷贝，然后单击 Open。现在就能在 File（文件）或 Folder（文件夹）域中看到 Web 文档的名字了。单击 Next。

5. 如果你先前曾定义了 Web 服务器连接，就能看到一个包含着连接名称的下拉式列表框。你可以选择其中之一，或者单击 New 按钮，定义一个新的连接。该例中，假定需要一个新连接。（如果确实不需要新连接，可以选择已有的一个连接然后跳到第 14 步）无论你单击了 New 按钮，或是以前从未定义过服务

器连接，都会看到如下图所示的对话框（该对话框来自于 Web Publishing Wizard 的最新版本，它包含有一个描述域和一个命令按钮，而老版本有两个域）。该对话框是你开始定义客户机与 Internet 服务器之间连接的地方。幸运的是，这些工作只要做一次就可以了。

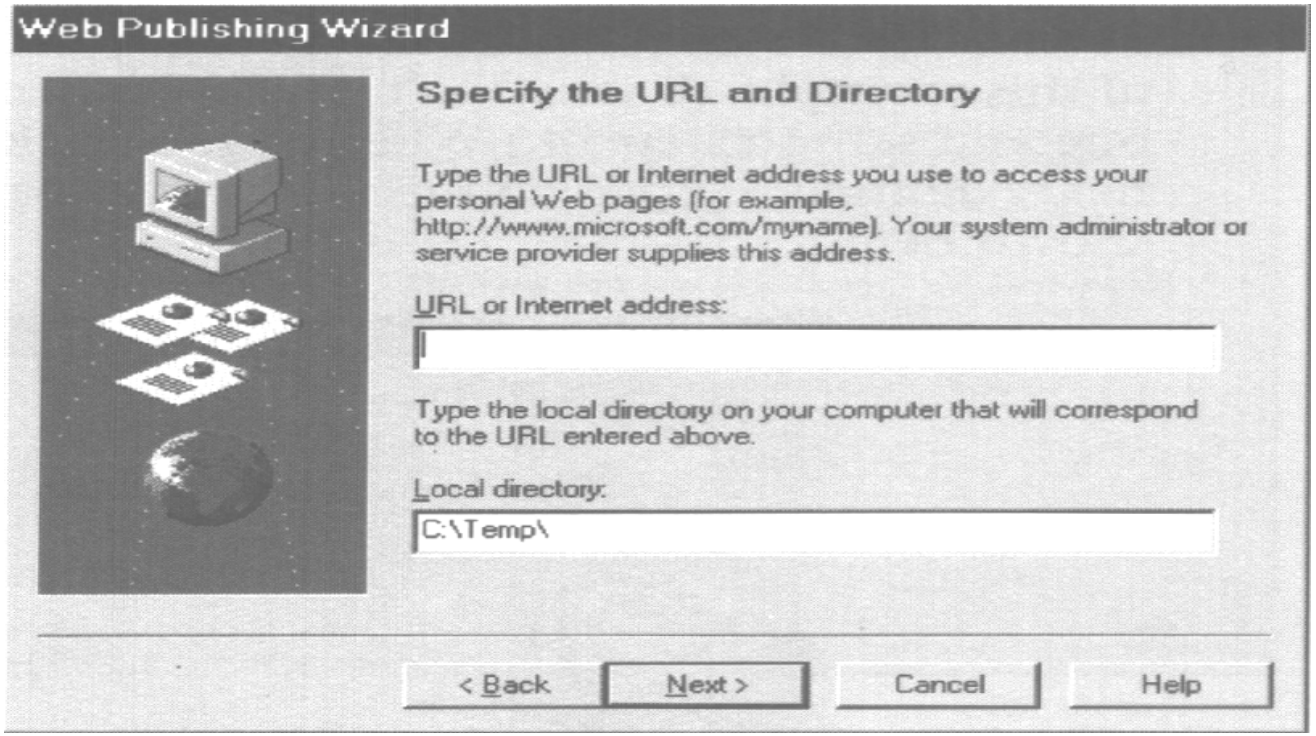


注释 老版本的 Web Publishing Wizard 在这个对话框中提供了两个域，第一个是连接名，第二个是创建的连接类型。可在 Web Publishing Wizard 的新版本上单击对话框中的 Advanced 按钮来访问第二个域。

6. 为 Internet 连接输入一个名字。在绝大多数情况下，如果只需要一项内容的

话，保持 My Web Site 不变就可以了。从第二个列表框中选择 Internet Service Provider（Internet 服务供应者）（Web Publishing Wizard 的最新版的用户需要单击 Advanced 按钮）。如果是为一个 LAN 内部网站点建立连接，那么，当使用老版本的 Web Publishing Wizard 时，可以选择 Other Internet Provider 选项，或者在使用新版的 Web Publishing Wizard 时自动提供 Select Service Provider 选项。

7. 单击 Next，就可以看到下一页面，如下图所示。

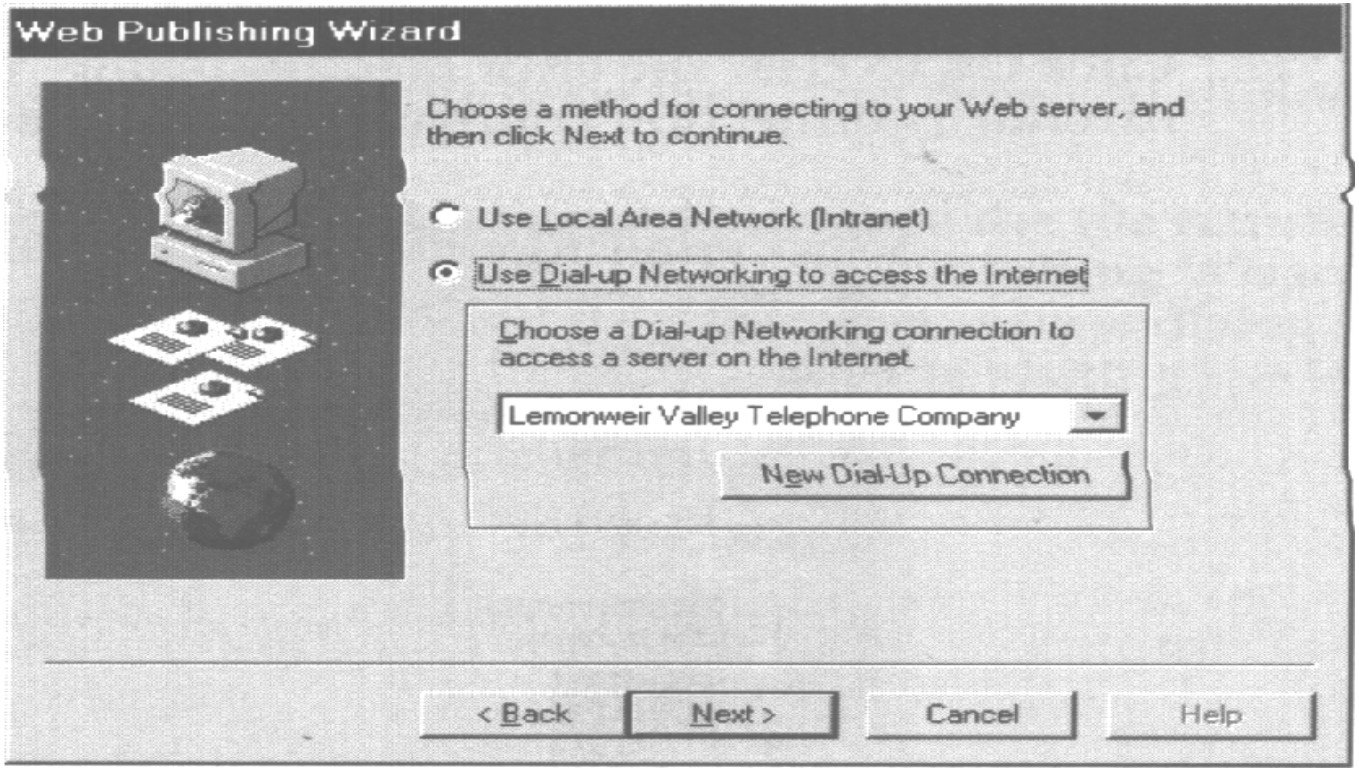


注意，你需要定义一个到 Internet 服务器的连接。在 URL 域中简单地输入自己的站点名。使用 Web Publishing Wizard 新版本的用户还需要提供与远程 Web 站

点相对应的本地目录。这是 Web Publishing Wizard 寻找发送到 Web 服务器上文件的地方。绝大多数情况下使用缺省目录也就可以了。如果你使用 Web Publishing Wizard，那么请跳到第 9 步。

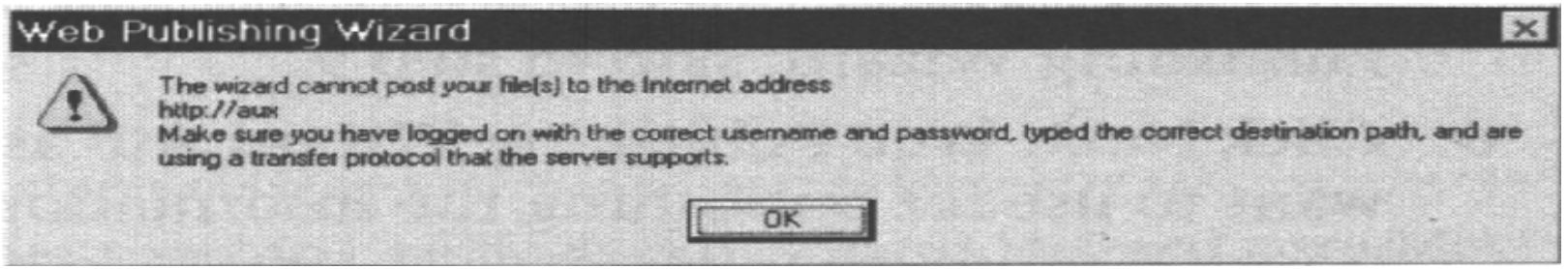
8. 单击 Next，Web Publishing Wizard 就会询问你用于发布信息的 URL，除非使用非正常安装，否则 Web Publishing Wizard 提供的缺省 URL 就工作得很好。

9. 单击 Next，就能看到下一页，如下图所示。该页允许你选择连接的类型：LAN 或拨号方式。不要被对话框中的描述弄糊涂，就像为 Internet 建立连接一样，你可以很容易地为内部网建立拨号连接。

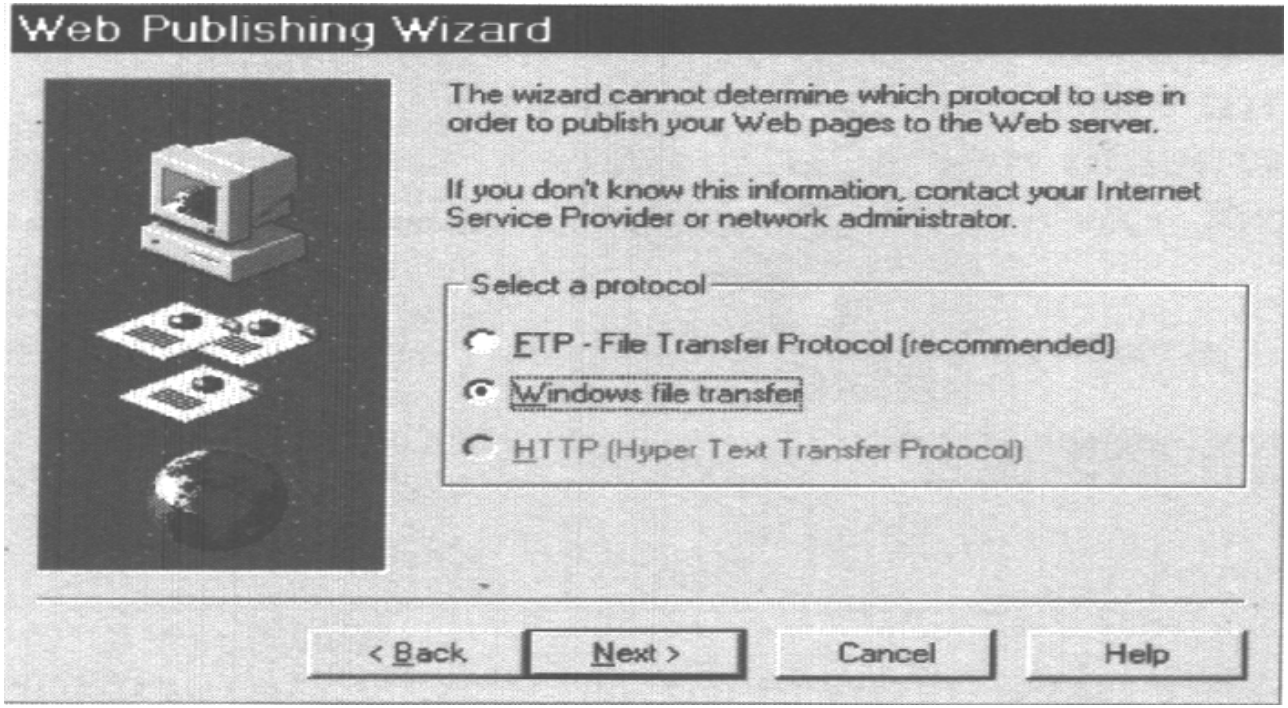


10. 在 LAN 连接（没有调制解调器的连接）或拨号连接之间选择一项。如果你选择的是拨号连接方式，那么还需要在列表框中选择一种拨号连接。单击 New Dial-UP Connection 按钮，可以创建一个新的连接定义。

11. 单击 Next，Web Publishing Wizard 会显示下一页，该页简单说明你所提供的信息中需要验证的信息。单击 Next 将开始对信息进行验证。如果你要建立 LAN 连接，几乎立刻就会得到如下图所示的错误消息。别担心，下面几步会告诉你如何解决这个问题。如果 Web Publishing Wizard 成功地找到了你的站点，你就可以跳转到第 15 步了。



12. 单击 OK，关闭错误消息对话框，你会看到扩展连接配置的第一页，如下图所示。



这就是你选择文件传输方式的地方。如果你是通过拨号连接方式访问 Internet 站点，就可以选择 FTP 或 HTTP 文件传输方式。其中的 HTTP 方法只对使用 HTTP 版本 1.1 及以上版本的 Web 站点可用。如果你使用的是 LAN，那么 FTP 和 Windows File Transfer 选项都可以使用。这里不同的连接还需各种不同的定义。接着，系统将继续显示 Windows File Transfer 选项，这也正是你经常需要使用的选项。

技巧 如果使用的是 LAN 连接，应尽可能使用 Windows File Transfer 方式，因为它比较快。FTP 连接需要一个附加的文件传输层，这在 LAN 环境中是不需要的。不过，在 WAN 上，FTP 方式能提供附加的安全保护层。

13. 单击 Next，就能看到如图 11.2 所示的下一页。在这里，连接问题变得明显了。Web Publishing Wizard 几乎总是不能找到文件正确的目标 UNC（统一命名约定）。原因很简单：大部分情况下该名字都被服务器隐藏起来了。你需要给自己的存贮目录提供一个完整的 UNC，如图 11.2 所示。确保提供的是一个 UNC 名，而不是标准的 DOS 驱动器和目录位置。使用 UNC 的原因是，它允许你使用同一种登记项技术，而不管服务器使用什么文件系统。

14. 单击 Next，Web Publishing Wizard 显示下一页，该页简单说明要验证你所提供的信息。单击 Next 开始验证过程。这次，你将会看到一条操作成功的消息。

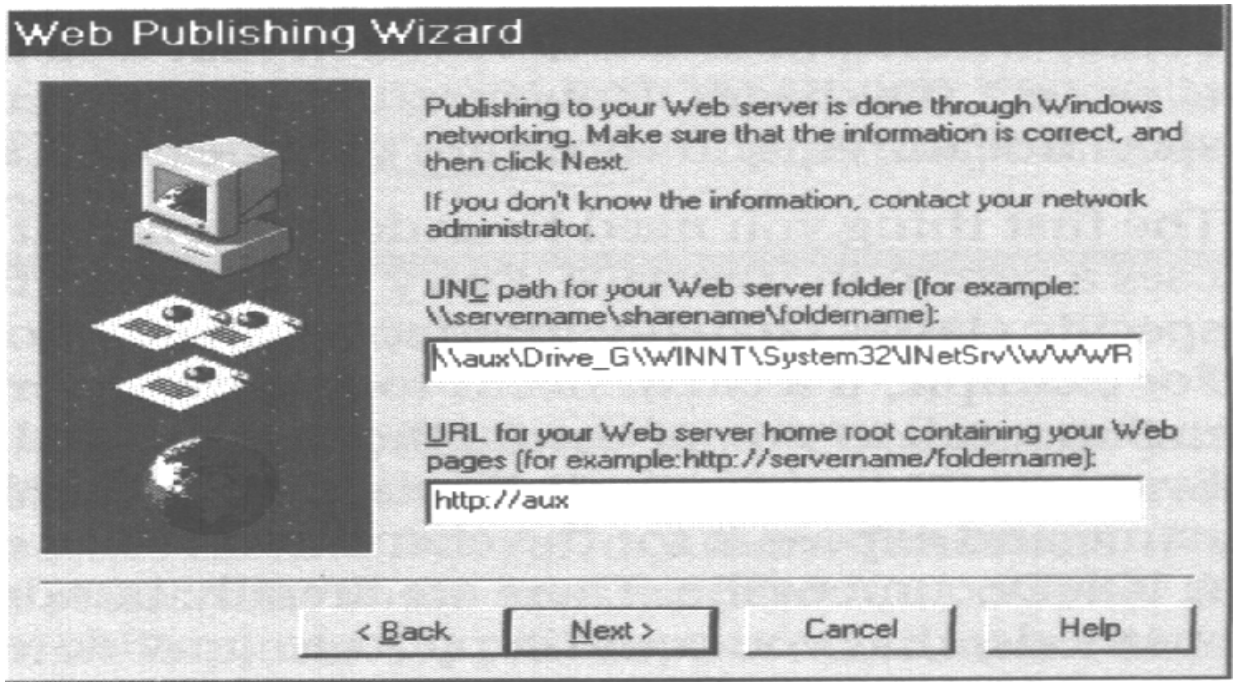
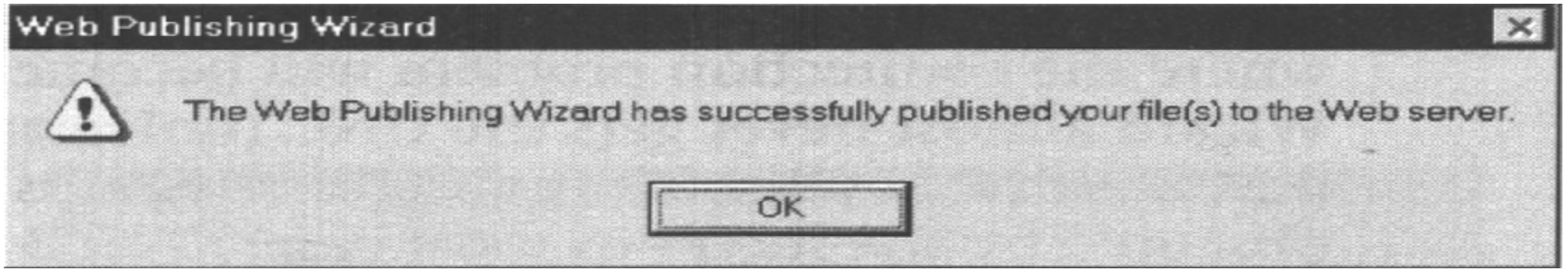


图 11.2　本页显示了你配置 Web Publishing Wizard 时几乎所有问题的产生根源

15. 单击 Finish，完成文件传输过程。在 Web Publishing Wizard 为你拷贝完文件时，你可以看到一个文件传送对话框。文件传送完后，可以看到如下图所示的成功对话框。此时你已经修改了该文档，访问 Web 站点的任何人都能自动看到所做的修改。



一旦你首次完成上述设置步骤，用户就可以只用简单的四步完成文件传送过程。而你需要做的就是将文档保存到本地驱动器上。利用 File|Send To|Web Publishing Wizard 命令启动该 Wizard，选中文件并选择一个连接，然后单击 Finish 就可以完成整个过程（即上面所列步骤 1 到步骤 4）。

技巧 网络管理员可以在每台机器上只执行这个任务一次，用以减少用户的支持电话。遗憾的是（至少到本书写作时），还没有办法像修改注册表或向一个目标机器拷贝文件那样自动完成上述任务。

11.2 ActiveDocument 结构概述

理解自己所使用控件的工作方式十分很重要的——至少要在一定程度上理解——只有这样在出现问题时你才能够解决它。本书无法详细介绍 OLE 的基本理论；即使整本书都只讨论这个问题，恐怕写上 1000 页都还不够。不过，ActiveDocumcnt 控件只需要 OLE 的一部分特殊特性，这就是本节要介绍的内容。

首先需要理解的是，OLE 控件利用一个公用接口进行工作。你使用的每个 OLE 控件都必须支持特定的类（实际上是接口元素），以便让应用程序访问它。例如，如果客户机需要服务器提供在位操作支持，那么它就会调用与 IOleInplaceActivateObject 相关联的方法之一。每个支持在位操作的服务器都必须提供该名字的类，并提供给客户机使用。本例中，客户机的等价类是 IOleDocumentSite。实际上，你可以支持几百种类（及相关方法）以便提供各种 OLE 功能（不过不必支持所有类，因为某些形式的功能是专用控件不需要的）。图 11.3 显示了在 Excel Worksheet 中使用的 OLE/COM Object Viewer（在第九章中已经介绍过这个实用程序）类（接口元素）。请注意，每个类都以 I 开头，而不是通常使用的 C。I 表示接口类，即客户机访问它们以获得服务器的某种功能。

注 由于 OLE 控件使用客户机了解和使用的标准接口，所以 OLE 控件能够工作。

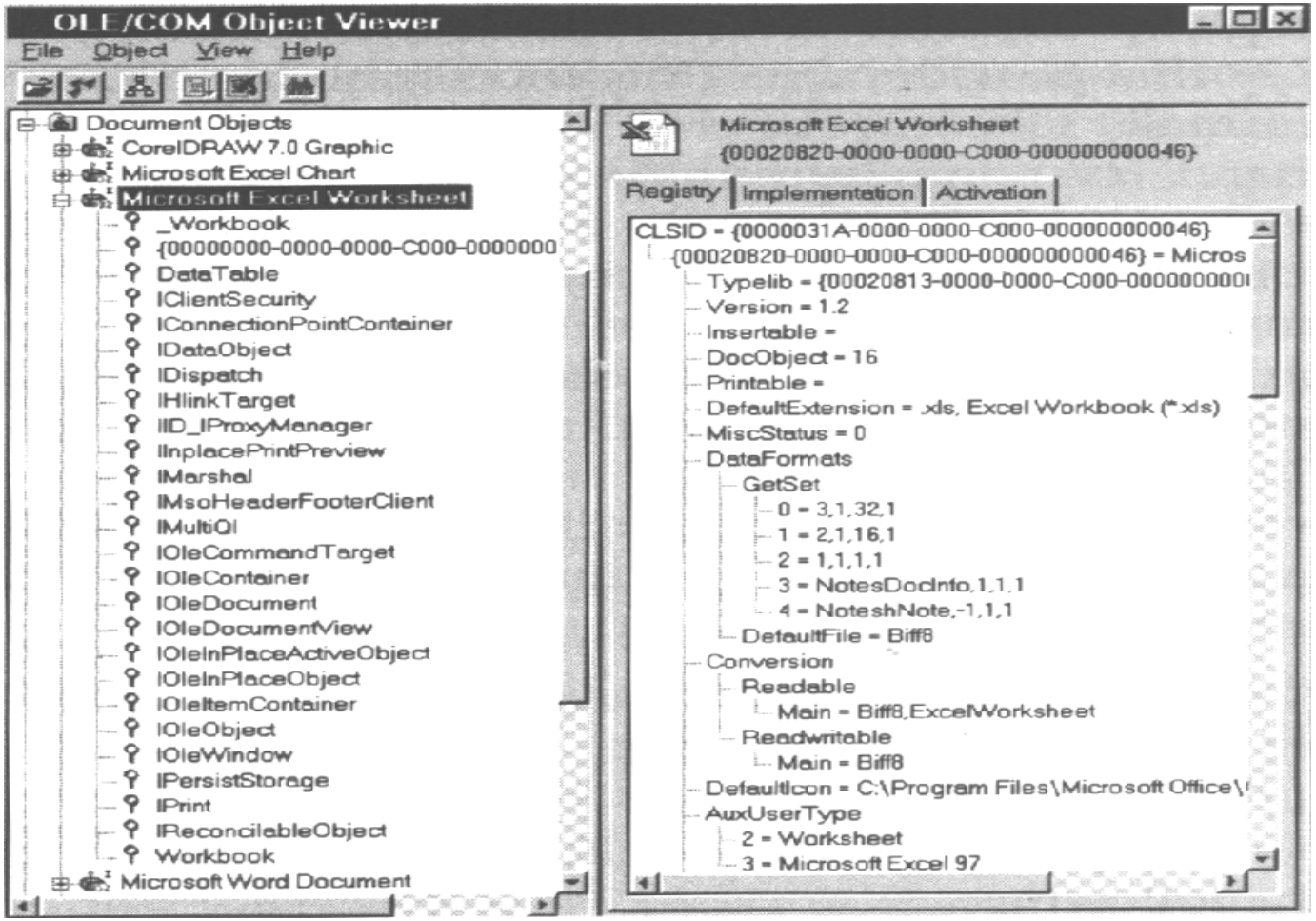


图 11.3　OLE/COM Object Viewer 提供了一种浏览创建控件时需要支持的接口元素方便的方法

注释　OLE/COM Object Viewer 的老版本称为 OLE/COM Object View。这两个实用程序都能提供同样的信息。

注释 图 11.3 显示了 Microsoft Excel 的 ActiveDocument 的最新功能，你可以看到四个 IMso 函数，这是 ActiveDocument 函数的预定义名称。ActiveX 规范现在声称这些是标准的 IOle 类。幸运的是，好象 Internet Explorer3.0 将支持使用两种接口的文档对象。不过，这种支持不会持续很长时间的，你最好还是使用本章讨论的 IOle 版本调用。

显然，试图记住你是否在自己的控件中实现了所有这些调用是不现实的。幸运的是，像 Visual C++这样对建立 ActiveX 控件（或 OCX）提供直接支持的 IDE，通常负责创建大量的类。（现在已有很多编译器厂家为他们的产品创建 ActiveX 附加功能。例如 Borland 的 Delphi 就已经包含了一个 Internet 控件包附加功能，以帮助你建立适用于 Internet 的应用程序）。另外，他们还为每个所需类实现了一种缺省动作。只有当你需要加入某种编译器不支持的特殊功能时，才需要改变它的动作。在第 10 章中你已经看到，如果建立了一个合适的应用程序框架，那么建立一个简单的控件并不需要做太多工作。本示例在加入一行代码之前，花费那么大力气来设置好一切配置，其原因正在于此。

实际上，有一种简单的方式来讨论创建 ActiveDocument 所需的 OLE 接口函数。我们可以用本书中学到的知识来总结 ActiveDocument 的功能。如果希望自己的应用程序支持这种规范，就必须遵守下述四条原则。

- 实现 IPersistentStorage 。应用程序必须支持该类及其相关方法以便能使用 OLE 复合文件作为存贮介质。
- 支持 OLE 文档嵌入特性。该特性可以用多种方式实现。目前流行的方式是

通过菜单功能给用户提供两种嵌入方法：Insert Object（插入对象）和 Paste Special（粘贴特殊项）。用于完成这项工作的函数是 IPersistFile 、IOleObject 和 IDataObject。

- 提供在位(in-place)操作支持。你必须实现两个类来支持在位操作：IOleInplaceObject、IOlePlaceActivateObject。为实现这些类，必须用 IOleInPlaceSite 和 IOleInPlaceFrame 类所提供的方法获取有关容器的信息。
- 加入 ActiveDocument 扩展。现在大部分 OLE 2 服务程序都能完成前面 3 步工作。为了使它能在 Internet 上工作，必须加入我们将简单讨论的这四个函数：IOleDocument、IOleDocumentView、IOleCommandTarget 和 Iprint。现在，让我们详细说明一下第四条。ActiveDocument 在某些方面相当新。由于 Microsoft 对这些规范很长时间没有公开，所以几乎很少有编译器能提供对 ActiveDocument 服务器的直接支持。也就是说，对于程序员，或者必须自己建立附加的接口（不很困难），或者升级编译器。下面四小节要介绍的内容是，为创建一个功能完整的 ActiveX 文档服务器，你必须要支持的四个附加的接口调用。（幸运的是，并不是所有这四个接口调用都需要——图 11.3 的显示说明，在 Excel 中，只需要实现其中三个）。另外 ActiveX SDK 也提供了创建这些接口元素所需的头文件和其它支持文件。本节也将对此作些介绍。

技巧 Microsoft Visual C++的4.2及以上版本，对于创建大部分ActiveX对象类型提供了直接支持，其中包括对ActiveDocument的直接支持。也就是说，这些版本实际上会创建我们本节所讨论的所有元素，从而减少创建自己新控件或应用程序时所需要的时间。

IOleDocument类

无论什么时候，当某个Internet客户看到实现了IOleDocument类的服务器时，它就会知道该服务器能作为一个ActiveDocument服务器。Internet Explorer和其它Internet客户机应用程序，在看到你的应用程序相关联的文档时，首先要查询的内容也正是这些。不要把标准的在位操作和ActiveDocument服务器所用的操作混淆起来。你可以建立一个只支持在位操作的应用程序，它在本地机器上会运行得很好，但它将不支持Internet上的在位活动。

注 IOle Document通知客户程序你的应用程序可以用作ActiveDocument服务器并提供了通讯服务。

注释 许多情况下，是否实现该接口并没什么关系，因为没有它，你的应用程序也能支持OLE2。忽略该接口的服务器，即使它对类似Word for Windows这样的本地客户支持在位操作时，也只是简单地在一个独立的窗口中被打开。

那么，除了让Internet Explorer知道应用程序支持ActiveDocument之外，该类还能做什么呢？每当客户需要创建新的服务器视图（CreateView方法）、枚

举视图（EnumViews 方法）、获取与 ActiveDocument 相关的 MiscStatus 位（GetDocMiscStatus 方法）时，都会调用这个类所支持的方法。实际上，这个类能帮助你从整体上管理服务器。它提供了客户机和服务器进行通讯的底层功能。

服务器视图并不等同于用户单击某个连接到你的文档链接时用户所看到的视图（那是由下节要介绍的 IOleDocumentView 界面管理的）。服务器视图提供的是服务器自身的一个单一实例。客户应用程序利用该视图进行通讯。例如，如果客户机需要找出服务器支持哪些特性，它就会利用服务器视图来完成。在介绍实际的应用程序代码时，你会很清楚地了解这一点。

注 服务器视图是服务器自身的一个单一实例——它支持在客户机和服务器间进行通信。

存在四种标准的混合(miscellaneous)状态位。下面逐一详细介绍。

- DOCMISC_CANCREATMULTIPLEVIEWS 让客户机知道服务器是否能创建多个视图。也就是说，该位定义了你是否能同时运行应用程序的多份拷贝。大多数情况下，现在的服务器都能做到这一点。唯一的例外是 CAD 或绘画程序，它们的内存需求实在太大了。另一个可能不支持该位的应用程序类是通讯程序，原因在于绝大多数人都只有一个调制解调器。
- DOCMISC_SUPPORTCOMPLEXRECTANGLES 用于指出服务器能否支持复杂的视图区域命令。例如，服务器是否允许客户机确定滚动条和缩放框的位置。
- DOCMISC_CANOPENEDIT 用于向客户机指出，服务器能否打开一个文

档用于编辑。设置该位能防止用户在线编辑文档（当你基于安全方面的考虑，或者希望创建一个只允许浏览的服务器时可以这么做）。

- DOCMISC_NOFILESUPPORT 用于指出服务器是否支持任意类型的文件操作。设置该位一般会使客户机显示一条错误消息，原因在于用户甚至不能读取所选的文件。

IOledocumentView 类

与 IOleDocument 类相似，你必须实现 IOleDocumentView 类才能使用 ActiveDocument。这个特殊的接口元素依赖于 IOleDocument 类。打开文档前必须已有一个服务器在运行。另外，客户机通过从 IOleDocument 类的 GetDocMiscStatus 方法获得的信息，才能知道如何与该类进行交互。IOleDocumentView 类的每个拷贝都控制着一个 ActiveDocument 视图的实例。大部分情况下，这意味着 IOleDocumentView 类的单个实例控制着单个文档。不过，同样可以很容易为自己打开的文档的每个视图创建一个类的实例。

注 服务器视图是服务器自身的单一实例——它支持客户与服务器之间的通讯。

IOleDocumentView 类支持许多方法。表 11.1 列出了最普遍应用的方法。可以看出，这些方法允许你改变屏幕大小、创建你正在看到的视图的另一份拷贝，重置视图的边界区域或者确定哪个文档在当前视图中显示等。

表 11.1　与 IOleDocumentView 视图类项关联的方法

方法	动作
SetInplaceSite	将视图站点对象与相应视图联系起来。客户机提供视图站点对象。本质上，这是将文档与当前视图联系起来的方法
GetInplaceSite	返回指向与视图相关联的视图站点对象的指针
GetDocument	返回指向与视图相关的文档的指针
SetRect	定义视图的边界区域。也就是说，该方法设置用户将看到的窗口的大小
GetRect	返回视图的边界区域坐标
SetRectComplex	定义视图的复杂边界区域。该方法不仅确定用户将看到的窗口的大小，而且确定诸如滚动条和其它视图元素的位置。视图不一定非要支持该特色。如果你的应用程序不提供这种支持，就需要设置一个混合状态位（参见 IoleDocument 类描述中的详细介绍）
Show	让客户机显示或隐藏视图
UIActivate	确定用户接口是否活动。正常情况下，仅在视图得到焦点时用户接口才是活动的，其它情况下，用户借口均不处于活动状态，这样可以防止与已经得到焦点的视图冲突
Open	请求服务器在另一窗口中打开视图。可以用混合状态位关闭该特色（参见 IoleDocument 类中的详细说明）
CloseView	关闭视图

续表

SaveViewState	将当前视图状态信息写入 Istream
ApplyViewState	请求视图将其状态设置为先前保存在 IStream 中定义的设置值
Clone	创建当前视图的一个拷贝。复制的视图与原视图有相同的上下文关系，但使用不同的视图端口（IoleDocumentView 类的实例）

IOleCommandTarget 类

在生成 ActiveX 文档时，可以不实现 IOleCommandTarget 类。但它却不仅仅是为了方便才设置的类。IOleCommandTarget 类允许客户机和服务器进行交谈，而不必借助于像设置固定的菜单 ID 那样进行工作，但这一通讯受到限制。例如，通讯仍然只限于某些命令（下面将进一步介绍）。为完成接口工作的这一部分，必须分成两个步骤进行。

注 IOleCommandTarget 类允许服务器和客户机进行通讯而不必使用像固定菜单 ID 这样的技术。

这种客户机与服务器通讯的第一个步骤，是找出服务器支持哪些命令。客户机必须利用 Query（询问）方法来完成。表 11.2 列出了服务器支持的所有命令以及它们的标识符的完整列表。首先要注意的是，大部分命令都是标准的菜单项。

表 11.2　IOleCommandTarget 类支持的常用命令

命令	标识符
Edit Clear	OLECMDID_CLEARSELECTION
Edit Copy	OLECMDID_COPY
Edit Cut	OLECMDID_CUT
Edit Paste	OLECMDID_PASTE
Edit Paste Special	OLECMDID_PASTERSPECIAL
Edit Redo	OLECMDID_REDO
Edit Select All	OLECMDID_SELECTALL
Edit Undo	OLECMDID_UNDO
File New	OLECMDID_NEW
File Open	OLECMDID_OPEN
File Page Setup	OLECMDID_PAGESETUP
File Print	OLECMDID_PRINT
File Print Preview	OLECMDID_PRINTPREVIEW
File Properties	OLECMDID_PROPERTIES
File Save	OLECMDID_SAVE
File Save As	OLECMDID_SAVEAS
File Save Copy As	OLECMDID_SAVECOPYAS

续表

非标准命令。该标识符询问服务器是否能完成下面三个任务：返回缩放值、显示缩放对话框以及设置缩放值。如果服务器支持这三项任务，则该标识符一般与 View 菜单命令（或等价命令）相对应	OLECMDID_ZOOM
非标准命令。该标识符获取服务器支持的缩放范围。如果服务器支持它，则该标识符一般与 View Zoom 命令（或等价命令）相对应	OLECMDID_GETZOOMRANGE
Tools spelling	OLECMDID_SPELL

客户/服务器通讯的第二个步骤是，使用 Exec 方法，客户机给服务器发送一个或多个 OLECMD 结构。每个结构将只包含一条命令、该命令所需要的输入参数以及存放调用返回信息标志的位置。你不必提供什么输入参数，因此结构的该部分应为 NULL。标准的选项请参见表 11.3。表 11.4 描述了 Exec 调用返回的标志。

表 11.3 标准 Exec 方法的输入参数

标志	动作
OLECMDEXECOPT_PROMPTUSER	在执行命令前提示用户输入一些信息。例如，使用 File Open 命令时，你就会使用该选项
OLECMDEXECOPT_DONTPROMPUSER	不要求用户作任何输入。例如，当用户请求打印文档时，可以使用该选项
OLECMDEXECOPT_DODEFAULT	当你不清楚是否要提示用户时使用该选项。这种情况下可以让应用程序执行缺省动作。大部分情况下这意味着将提示用户进行输入
OLECMDEXECOPT_SHOWHLP	根本不执行命令；而是显示它的帮助屏。如果你的 ActiveDocument 能提供另外的帮助按钮，就可以使用该命令

表 11.4　标准 Exec 方法的返回值

标志	动作
OLECMDF_SUPPORTED	视图对象支持请求的命令
OLECMDF_ENABLED	该命令可以使用且视图对象已把它激活
OLECMDF_LATCHED	该命令使用了一个 On/Off 开关并且现在开关被置为 On
OLECMDF_NINCHED	视图对象不能确定使用 On/Off 开关的命令的状态。大多数情况下，意味着该命令使用的是三态配置并且正处于不确定状态。例如，如果用户为某安装程序选择了一些子项但没选其它项的话，一个三态复选框就可能返回该值。（该复选框在屏幕上显示为灰色。）

Iprint 类

IPrint 是另一个你能够实现的可选类。该类允许某个对象编程打印。IPrint 支持三种方法：打印(Print)、获取与打印相关的信息(GetPageInfo)以及设置打印作业的初始页号(SetInitialPageNum)。三种方法中，只有 Print 方法接收任意个数的标志作为输入。表 11.5 列出了这些标志以及它们的用法。

注　IPrint 的作用不言自明——允许对象支持可编程方式打印。

表 11.5 IPrint 的 Print 方法支持的标志

标志	动作
PRINTFLAG_MAYBOTHERUSER	告诉服务器，客户机允许用户进行交互操作。如果没设该标志，任何打印请求都必须自己完成。大部分情况下客户机都允许用户交互——唯一的例外可能涉及批处理打印作业或者打印操作处于后台的情况
PRINTFLAG_PROMPTUSER	提示用户在标准打印对话框（类似于 Windows 支持的打印框）中输入有关打印作业的信息。例如，如果指定该选项，则用户可以选择拷贝份数。使用该选项还必须同时指定 PRINTFLAG_PROMPTMAYBOTHER-USER 标志
PRINTFLAG_USERMAYCHANGE-DPRINTER	允许用户修改打印机设置。不过有些情况下可能不希望指定该选项——比如在网络环境中用户就不太容易访问到打印机。使用该选项必须同时指定 PRINTFLAG_PROMPTUSER 标志
PRINTFLAG_RECOMPOSETODEV-ICE	告诉目标打印机进行打印作业重组，例如，如果目标打印机支持比目前打印作业指定的分辨率更高，打印作业就应使用更高的分辨率
PRINTFLAG_DONTACTUALLYPR-INT	测试打印作业，但并不真正创建任何输出。该选项允许你在不浪费纸张的情况下测试用户接口的功能
PRINTFLAG_PRINTTOFILE	将打印输出送往文件而不是打印机

11.3 创建 ActiveDocument

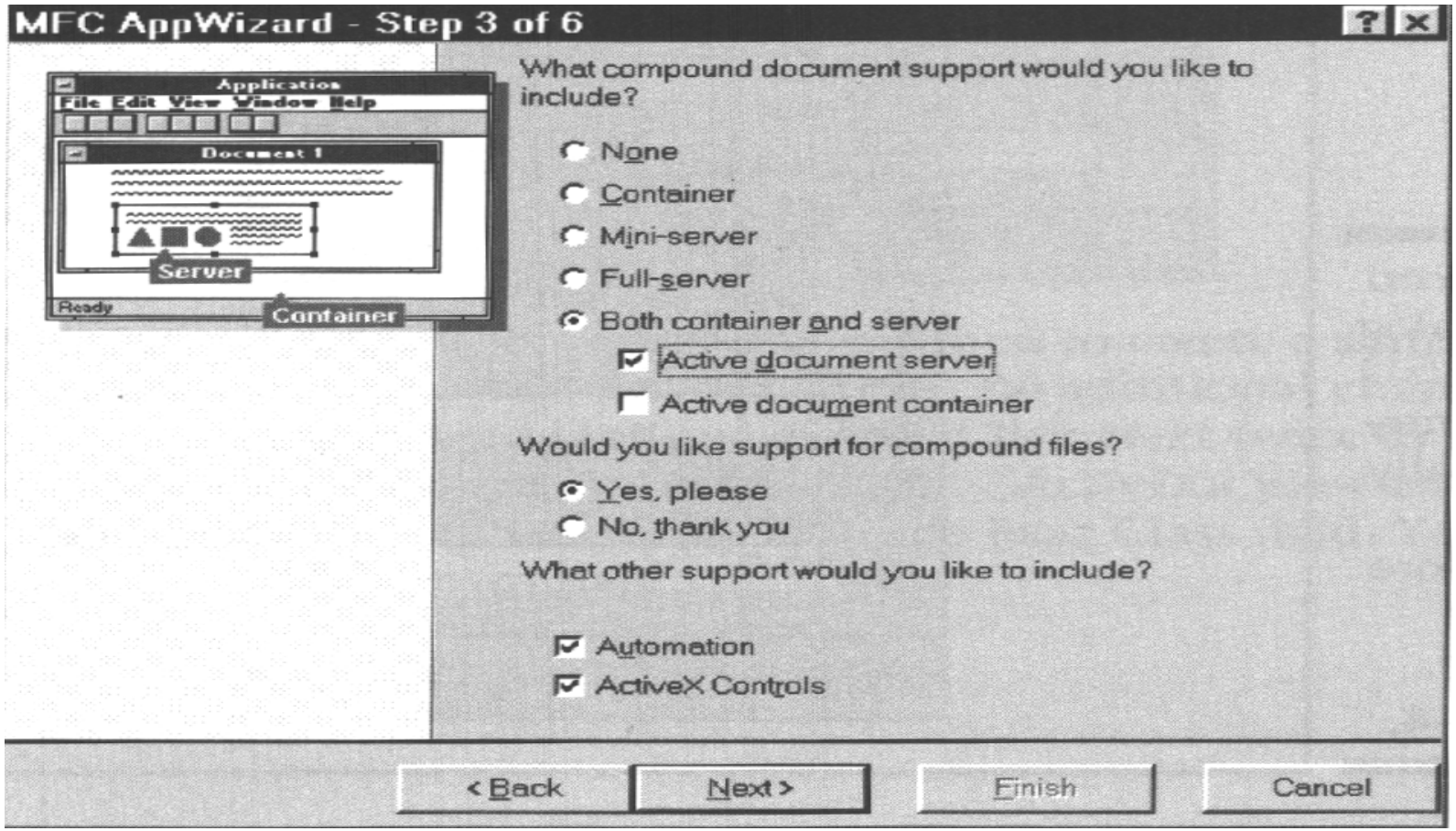
前面几节已经介绍了利用 ActiveDocument 应用程序（如 Word for Windows 或 Excel ）可以做些什么。如果你是一个只需要得到一些增强性能的高级用户，或者是一个只想优化自己的 Web 站点的 Web 管理员，那么，这些就足够了；但毕竟你是一个需要用这些功能创建应用程序的程序员。

即使对最好的程序员而言，手工加入前文所述的所有四个接口也是相当困难的。而且，由于 Microsoft 已经发布了 C++的新版本（4.2 以上版本都可以，不过 5.0 比 4.2 更好用一些，6.0 版则已天衣无缝了），实际上也没有必要去找这个麻烦。本节要介绍的就是创建支持 ActiveDocument 的应用程序的快速方法。创建其它任何应用程序都有的步骤这里就不再详述了，而着重介绍工程中涉及 ActiveX 的特定部分。

首先，你需要完成的任务是创建一个新的工程工作区。第二章我们已经介绍了完成这项工作的步骤。你可以选择 MFC AppWizard，然后给应用程序命名——本例把它命名为 ActivDoc。单击 OK 开始一个创建过程，可以看到 MFC AppWizard 的第一页。为简单起见，选择 Wizard 第一页上的 Single Document 选项。两次单击 Next 跳过 Wizard 的第一和第二页，下面你将看到第三页对话框。

第三页是为应用程序完成大部分 ActiveDocument 设置的地方。你可以为应用程序提供五个不同层次的 OLE 支持，最后的三个层次允许你添加 ActiveDocument 支持。Mini-Server 选项不允许单独运行应用程序——必须在

Word for Windows、Internet Explorer 3.0 或其它什么容器中运行它。如果你想创建文件浏览器，这一级支持确实不错。下一选项是 Full-Server，它允许应用程序自己运行。你可以利用这种应用程序来支持对象，但不显示它们。绘图程序经常是服务器应用程序的好例子，但是，不必作为一个容器来运行。最后一级，Both Container and Server（容器与服务器）是本示例要选择的选项。它允许你在应用程序中提供完整的 OLE 2 功能，包括嵌入对象。另外，你还可以选择 ActiveX Document Server（文档服务器）复选框，如下图所示。另外，要确保选中 Automation(自动化）和 ActiveX Controls（控件）复选框。下图是选择选项后对话框的外观。



注释 ActiveX Document container（文档容器）是 Visual C++ 6.0 的新特色。它允许建立的应用程序包含其它的 ActiveDocument。例如，利用这种特色可以在没有真正提供 HTML 能力的应用程序中显示 Internet Explorer 文档。

单击 Next，系统将显示 MFC AppWizard 的第四页。该页上的大部分设置都

已完成。你可能想把 Recent File List（最新文件列表）的值设置的再高一点，因为大多数人都喜欢该特性（它当然要占用硬盘空间）。大部分情况下设为 9 或 10 就足够了；本示例中设置为 10。

该页还包括一个 Advanced（高级选项）按钮——如果不仔细的话，大部分程序员都会忽略该按钮。不幸的是，该按钮实在不应该标识为 Advanced，（也许 Microsoft 应考虑对 Application Wizard（应用程序向导）做点修改使某些设置的意义更明确一些）。单击 Advanced 按钮，即会显示如图 11.4 所示的对话框。

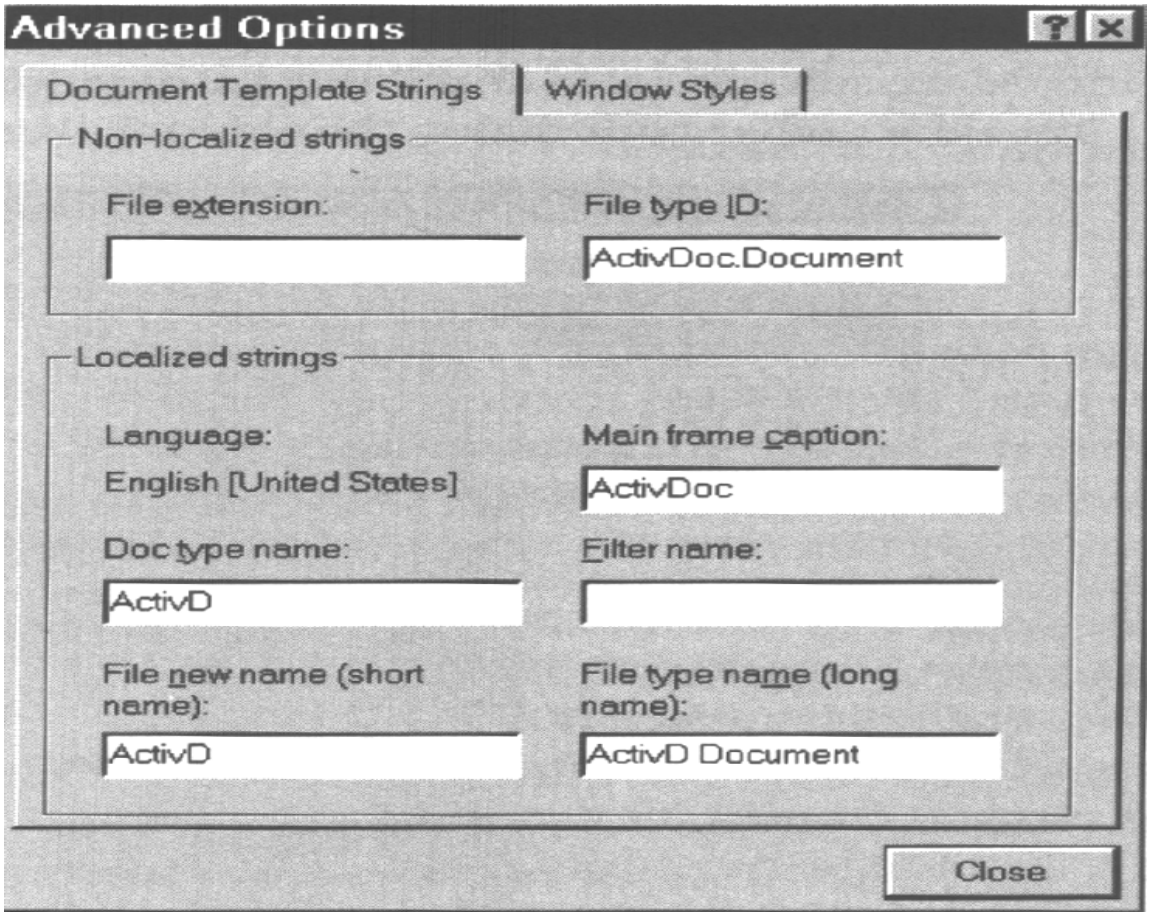
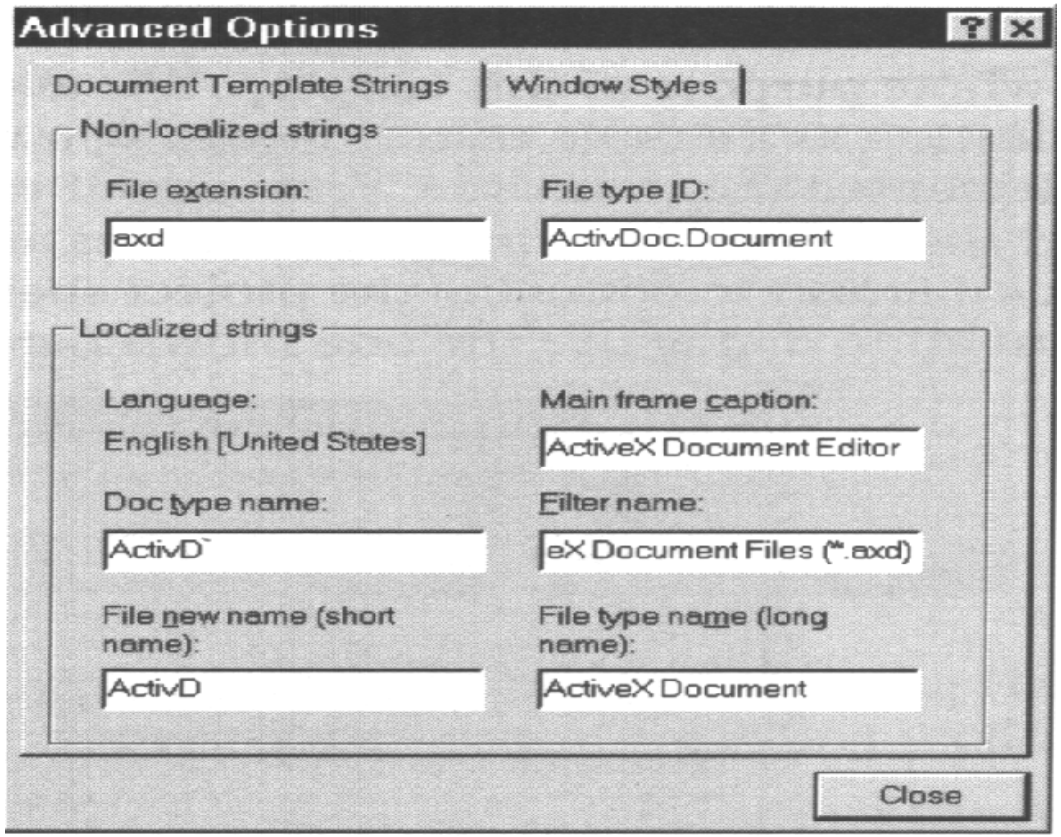


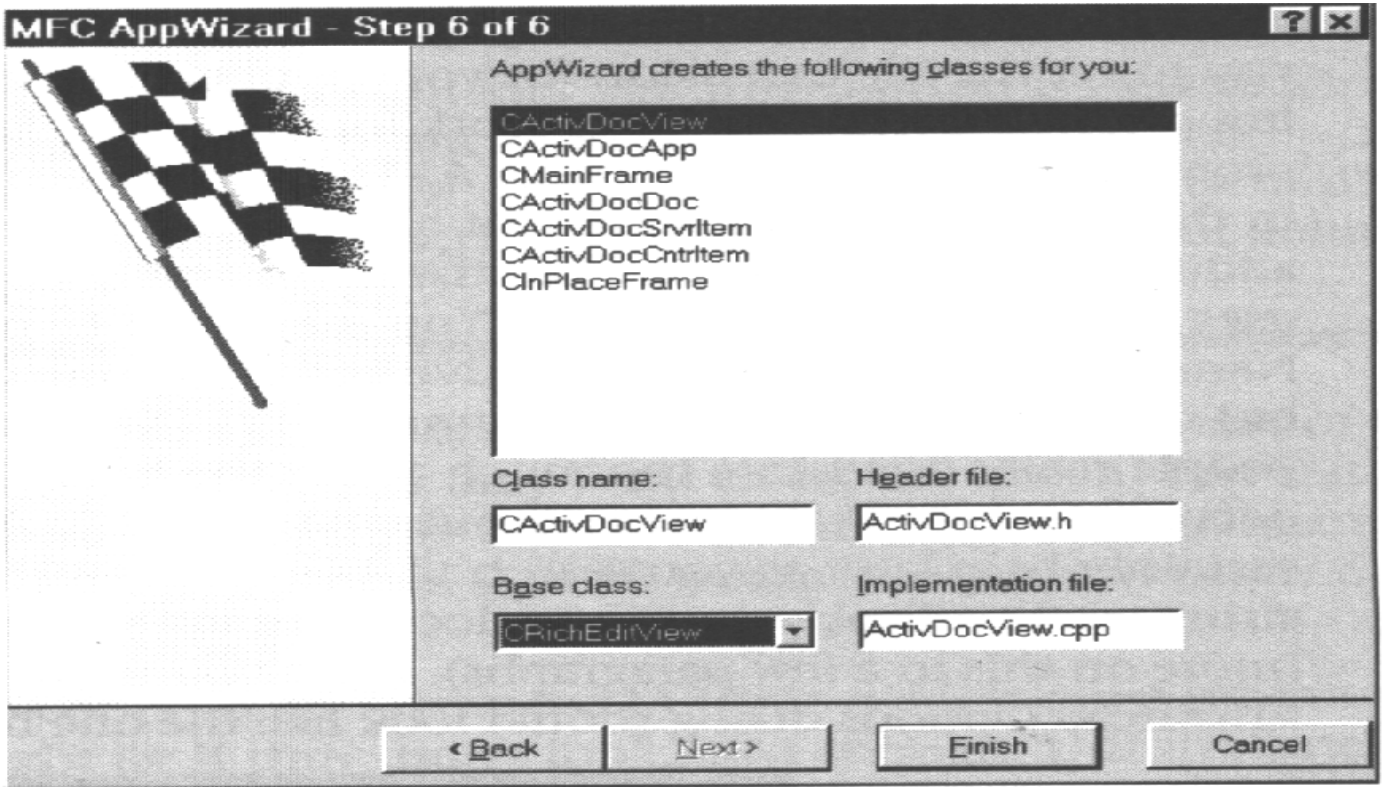
图 11.4　Advanced Options 对话框中至少包含了一个需要在创建应用程序前修改的选项

Advanced Options（高级选项）对话框的 Document Template Strings（文档模板串）属性页允许你为应用程序设置文件扩展名。另外，它还自动完成一些幕后的工作。本例使用的文件扩展名为 AXD。你要做的工作就是在第一个域（开始为空）中输入扩展名。也许你还想修改其它字符串，如 Frame Caption（主框

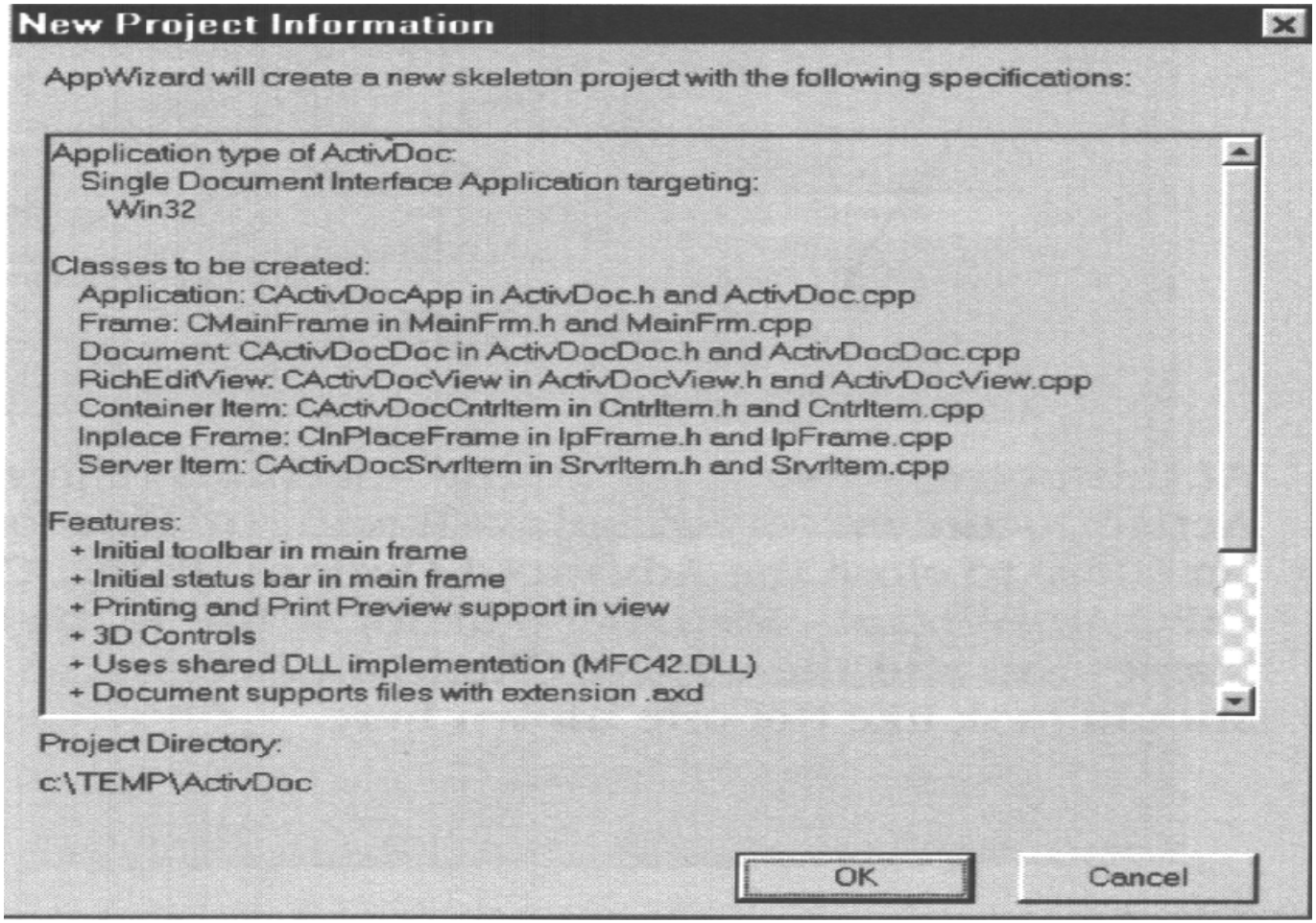
架标题）域。该例使用 ActiveX Document Editor（文档编辑器）。另外，你可能想让 Filter Name(过滤器名称)域中的输入值更富于描述性。初始时它是 ActivD Files(*.axd)，把它更改为 ActiveX Document Files(*.axd)时可读性会更好些。有些人确实不太在意 File Type Name（文件类型名称）域的长短（缺省为 ActiD Document），不过如果把它更改为 ActiveX Document 一定会对以后注册时的检索大有帮助。另外，它也是在 Windows 上下文相关菜单中用于显示你的新文档的字符串（稍后还将进一步介绍）。做完这些修改后，Advanced Options 对话框应如下图所示。



至此，我们已经完成了创建一个简单的 ActiveDocument 服务器的所有选项设置，但还有一点需要修改。单击 Close 关闭 Advanced Options 对话框。单击 Next 两次就可以到达 MFC AppWizard-Step6-6 对话框。在类列表中选择 CActiveDocView 项，然后在 Base Class(基类）域中选择 CRichEditView。此时的对话框如下图所示。



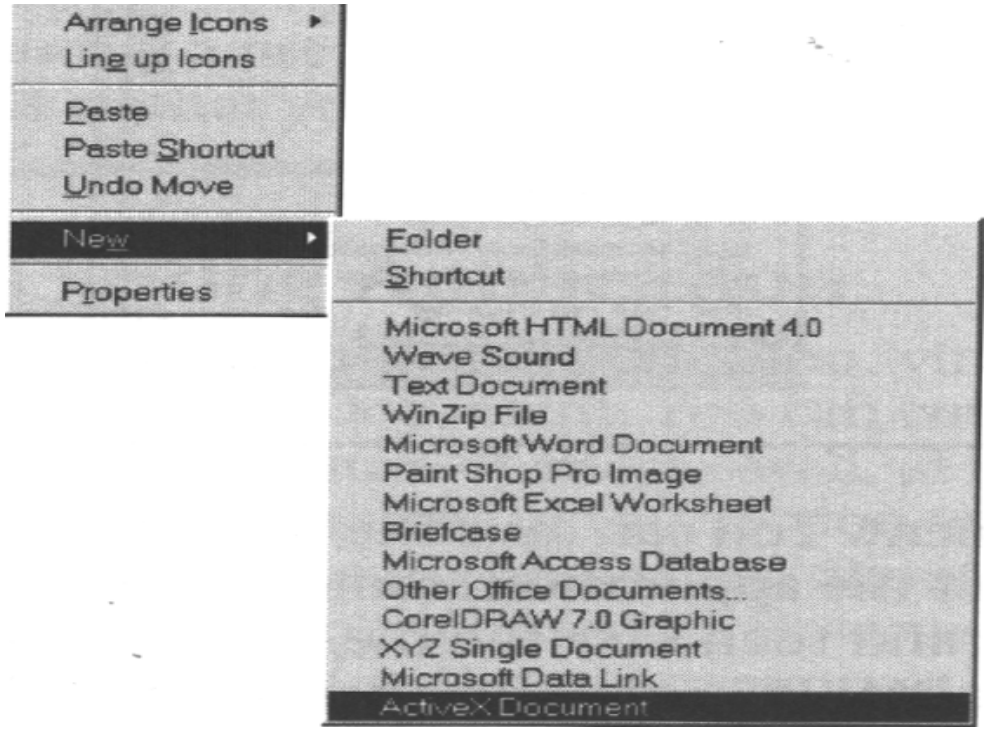
最后一步的目的是，在你愿意时，允许把该例子的应用程序作为简单的编辑器。如果愿意的话，你还可以实现标准的 CView 类，它不影响该程序作为 ActiveDocument 服务器的功能。现在单击 Finish 完成该工程，就会看到如下图所示的 New Project Information（新工程信息）对话框：



花点时间浏览一下特性列表，以确定应用程序所需的 ActiveX 支持是否完整（做过几个工程后，你就能通过该对话框迅速发现问题了）。单击 OK 生成工程。

测试缺省的应用程序

现在，虽然我们的示例应用程序还做不了太多的事，但它确实能在 Wizard 之外做一些什么了。当 MFC AppWizard 创建完成之后，你可以编译并运行该应用程序。运行应用程序很重要，因为第一次运行时，应用程序将生成一些注册表项。可以发现的第一个变化是 Windows 上下文相关菜单现在包含了应用程序文件类型的一个表项，如下图所示，并且它与前面章节中我们在 Advanced Options 对话框的 File Type Name 域中输入的名称一样。



你还可以发现另一个变化。图 11.5 显示了在 OLE/COM Object Viewer 实用

程序（前面已用过它）中的 ActiveX 文档应用程序类型。可以看出，它与其它的文档对象——比如 Word for Windows——列在了一起。你应该能够立即发现，本章介绍的四个接口，在这里一个也没有出现，就像 Word for Windows 或其它 Binder 程序一样。在程序已经准备好进行测试之前检查接口，本节稍后会让你了解为什么这样做会大大节省时间和精力。

ActiveDoc 程序还能够创建一个基本的容器文件。你需要做的工作只是用 Insert/Object 命令给当前文档加入一个已有的对象。你可以将该文件保存到磁盘中。现在试着创建一个容器文件，以便能用 Internet Explorer 测试应用程序框架。确保已插入了一个对象，然后保存文件，否则打开文档时将什么也看不见。该例使用 ColorBlk. BMP 作为对象；而文件本身保存为 TESTDOC.AXD。一旦你创建了测试文档，就应该创建一个 HTML 页面来测试它。程序列表 11.2 显示了该例使用的代码。

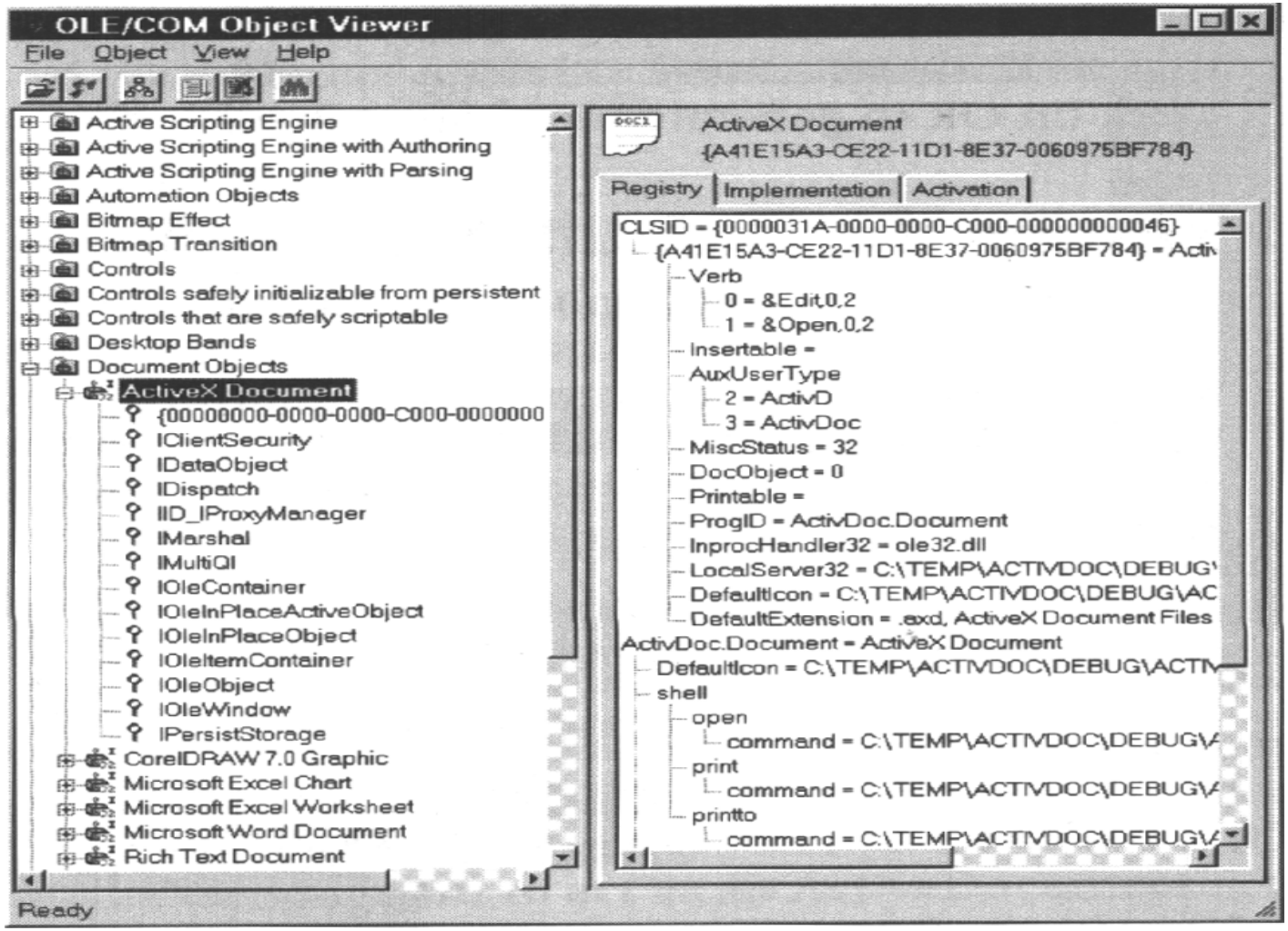


图 11.5　ActivDoc 应用程序作为 Document Object（文档对象）显示在 OLE/COM Object Viewer 实用程序中

程序列表 11.2

```
<HTML>
<HEAD>
<TITLE>new Page</TITLE>
</HEAD>
<BODY>

<!-显示标题->
<CENTER><H2>ActiveX Document Test</H2></CENTER>

<!-创建一个到测试文档的链接->
Click < A HREF="TestDoc.AXD">Here</A> to test the AXD file

</BODY>
</HTML>
```

既然我们已经有了一个测试基础，那么让我们看一下应用程序的工作情况。在 Internet Explorer 中打开 Web 测试页，然后单击测试链接。你看到的是在 Internet Explorer 中我们的测试应用程序的一个副本，如图 11.6 所示，就如同在 Word for Windows 中工作时一样。请注意示例应用程序已经接管了浏览器的菜单和工具栏，就如同在 Word for Windows 文档中一样。另外，使用菜单可以插入新对象，还可以执行其它任务，就好象应用程序是在本地文档中工作一样。你还可以使

用 Web Publishing Wizard（本章前面已详细介绍过它）来保存这些修改。

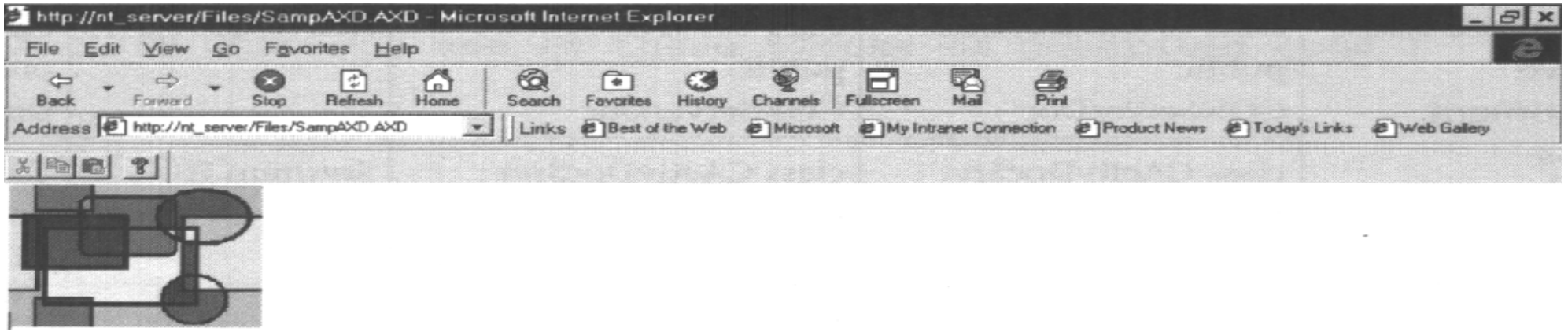


图 11.6　如任何 ActiveX 文档一样我们的文档显示在浏览器窗口中

注释　如果一切正常，系统会显示一个对话框，询问是保存还是打开文件（当然假定你在前面的会话中没有关掉对话框）。一定要告诉浏览器是打开文件，这样才能真正浏览它。

转换现存的应用程序

如果你有一个非常出色的应用程序，但它却不支持 ActiveDocument，请别担心，只要稍微做点工作就能将它转换成能够提供这种支持的应用程序了——至少比从头开始重写该程序要省事多了。下面就将介绍转换过程的 5 个步骤。如下所述，你可以利用它将任何一个已有的 OLE 服务器转换为基本的 ActiveDocument 服务器。不过要记住只能提供基本的支持。某些应用程序会和

这个例子一样工作正常，有些却不行。在开始加入支持项之前，应再花点时间去看看本章“ActiveDocument 结构概述”一节并记下要点。做完附加的支持工作后，一定要用实际的 Internet 环境而不是本地驱动器来测试它们。这样做能保证测试所有接口，并且保证应用程序不会因为使用了标准的 OLE 接口，而没有使用你真正想测试的 ActiveDocument 接口来工作。

步骤 1: 实现所需的类

首先并且最明显的一步是实现创建 ActiveX 文档服务器所需的类。如果你比较一下前面创建的例子和其它 C++ 4.2 版之前的应用程序，就会注意到声明类的方式有点不同。这是因为 Microsoft 已经从原来的 MFC 类中导出了许多子类，并且给它们加入了一些功能。表 11.6 列出了哪些类作了修改以及作了什么修改。第一列是在 C++ 4.2 版之前的应用程序中使用的类声明。第二列是现在的声明。第三列告诉你影响了哪些文件。第四列则是在以前 MFC 类文件中可能会发现的另外的类声明——它们是基于 Microsoft Office Binder 声明。

表 11.6 Active 文档类声明

最初的类说明	C++4.2 以上版本的新说明	ActivDoc 例子程序文件	MFC 类说明
class CInPlaceFrame: public COleIPFrameWnd	class CInPlaceFrame: public ColeFrameIPWnd	IPFrame.H	class CInPlaceFrame: public CDocObjectIPFrameWnd

续表

class CActivDocDoc: public ColeServerDoc	class CActivDocDoc: public ColeServerDoc	ActivDocDoc.H	class CActivDocDoc: public DocObjectServerDoc:
class CactivDocSrvr: public ColeServerItem	class CActivDocServr: public CDocObjectServer	SrvrItem.H	class CActivDocSrvr: public CDocObjectServerItem

在头文件中替换了这些声明后，你还需要在相应的 CPP 文件中替换它们。利用 Microsoft Developer 的检索和替换功能会使工作变得容易一点。你不必担心会漏掉使用新类名的地方，在你做完头文件中的修改后，编译器会自动指出不一致的地方。所以必须先做头文件修改并且在往下做之前好好检查。（注意，如果你想用 ActiveX SDK 和老的 MFC 文件集来实现 ActiveDocument 服务器，那么必须修改三个头文件，而如果用 Visual C++ 4.2 以上版本只需要修改两个头文件）。

步骤 2: 添加一个简单的声明

第二步是给应用程序的 STDAFX.H 文件中添加一个简单的声明：

```
#include<afxdocob.h>
```

该头文件包含了使应用程序支持文档对象所需要的所有声明。实际上，略

微看看该文件就很有启发，因为它准确地声明了四个 ActiveDocument 接口元素是怎样实现的。

步骤 3: 修改注册表项

第三步是修改你使用注册表项的方式。你正在创建的应用程序已不再仅仅是一个在位（In-place）服务器，所以需要在 CWinApp.CPP 文件（本例是 ActivDoc.CPP 文件）中修改注册表项，把 OAT_INPLACE_SERVER 改为 OAT_DOC_OBJECT_SERVER。代码如下所示：

```
m_server.UpdateRegistry(OAT_DOC_OBJECT_SERVER);
```

步骤 4: 修改语法分析映射(Parse Map)

第四步是修改某些语法分析映射，因为必须让应用程序知道将打印命令和其它 OLE 相关信息送往何处。此时你需要对两个文件进行修改。第一个是应用程序文档头文件，本例中即为 ActivDocDoc.H。你必须加入下列代码行：

```
//Generated message map functions
protected:
        //{{AFX_MSG(CActivDocDoc)
                //NOTE - the ClassWizard will add and remove member functions here.
                //   DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
DECLARE_MESSAGE_MAP()
```

```
DECLARE_OLECMD_MAP()
};
```

如果消息映射包括 DECLARE_MESSAGE_MAP( ) 消息映射函数，就一定要在其后加入 DECLARE_OLECMD_MAP()。第二个文件是应用程序文档 CPP 文件。在文件的一开头，你会看到至少有一个用于消息的映射区。还要加入另外一个映射区，如下所示：

```
BEGIN_OLECME_MAP(CActiveDocDoc,COleServerDoc)
    ON_OLECMD_PAGESETUP()
    ON_OLECMD_PRINT()
END_OLECMD_MAP()
```

正如你所看到的那样，增加的代码允许应用程序使用 ID_FILE_PAGE_SETUP 和 ID_FILE_PRINT 标准标识符，在它们的处理函数中，通过安排打印函数来完成打印作业。你要做的工作只是给真正的句柄函数加入命令映射，如下所示：

```
ON_COMMAND(ID_FILE_PRINT,OnFilePrint)
```

步骤 5: 添加新函数

第五步（也是最后一步）用来给你的应用程序文档头文件及 CPP 文件增加新函数。

一旦完成这些修改后，必须编译并测试修改后的应用程序。大多数情况下，

可以先在本地机上进行测试，以保证这些修改不会造成破坏。确信这些修改不会影响本地机性能后，可以试着从浏览器中打开文档。文档将在浏览器中打开而不是在另一窗口中（就像图 11.6 所示）。一定要保证该应用程序如同在本地机上使用时那样，能提供层次的功能（比如 File|Save 命令）。

第 12 章　使用 URL 和 Moniker

美国东北部的人们因为这样一句话而著称：“你不能从这儿直接到那儿，你必须首先途经另外一个地方。”有些人对 Internet 的看法也是如此，看起来总是要至少跨越四、五个 Web 页面，才能找到自己所需的起始站点。

正是这个原则，导致有些 Web 拥有者在他们的 Web 站点上，放置了许多链接。每个链接都指向与当前主题相关的信息——即从某个方面扩展该站点的信息。没有一个用户抱怨已有的这些链接，事实上，如果去掉这些链接他们反而会不满意。

不过当前这种链接系统确实存在一些问题。就像 Internet 上其它东西一样，它们都是静态的。不管如何进入站点，也不管是在什么样的条件设置下进行操作，对于一个特定的站点得到的链接集合总是相同。如果 Web 站点能有一点智能，从而可以帮助用户找到他们所需的东西，那时又会怎么样呢？另外，如果用户能输入一些条件，使得从一个站点到另一站点更加容易，不是挺好吗？例如，可以按照目前需求输入某种判断条件来定位某种特殊的站点。通过使用 URL（它定义出查找的目的地字符串）和 Moniker（它是由系统提供的实际的资源对象），ActiveX 提供了一种处理这种动态需求的方法。

注　统一资源定位器(URL)指出了在 Internet 上的什么地方进行检索。

注 在 Internet 上下文中，Moniker 是指用于保存位置对象的资源。

12.1 URL Moniker 概述

这一节中我们先介绍一下 URL 和 Moniker 这两个术语。URL（统一资源定位器）是一种方法，我们对它已经比较熟悉了，我们用它从一个站点走到另一站点。它就是使用浏览器时打入的 http://www.mycompany.com，URL 有三个主要部分，它们是协议（上例中是 http）、主机（www）和域（mycompany.com），你打入的 URL 的其余部分即是你标识的资源位置（如目录）。这种机制只所以非常有用，是因为可以无限制地扩展它。例如，如果你想用另一种协议，只需在 URL 的开头指明即可。Internet 有好几个这样的协议，包括 HTTP、FTP、Gopher 和 News。

注 URL 由三个主要部分组成：协议、主机名和域。

注释 本节中讨论的 URL，包括了全部的 URL，而不仅仅讨论一般情形。例如，一些 URL 把 host（主机）称为 home 或别的什么名称，如：http:/home.netscape.com。另外，一些域正变得越来越复杂了，从而可以提供更精确的信息，如，http:/www.wisconsin.edu.k6.us 是美国威斯康星州小学校的 URL。一些浏览器不要求用户打入协议名，他们简单地为资源核对所有合适的协议。总之，除了 URL 不同于他们提供的一般例子外，在这一节中，不对它作任何的假定。

Moniker 起源于 OLE 2 规范。实际上一个 Moniker 就是系统生成的一个对象，你可以利用该对象找到另一对象或者从中获取数据。某些定制的 Moniker 还支

持其它操作，但任何 Moniker 都提供两种基本操作。OLE 2 规范推出了同步 Moniker，也就是说应用程序必须等待，一直到系统已经从指定对象获取了所查找的数据为止。

Web 链接 如果你想在商业性的应用程序中使用 URL Moniker，最好先进行一点专业训练。Developer Solutions（开发者解决方案）在 http://www.devsolutions.com/ 上开设了一门课程，专门讲述 Moniker 的问题。三天的课程还讨论了各种其它与 OLE 相关的主题如 DCOM 和编组(marshalling)。如果你确实想了解 Microsoft 提交给 ANSI 申请批准的 OLE 标准的所有信息，可以浏览 http://info.gte.com/ftp/doc/activities/x3h7/by_model/ OLE 2.html。

将 URL 和 Moniker 的思想组合起来，就得到了 ActiveX API（从 Visual C++4.2 版至 6.0 版的一部分）提供的 URL Moniker。那么 ActiveX 使用的 URL Moniker 与 OLE 2 规范所说的同步有什么不同呢？一方面，URL Moniker 同时提供了同步和异步捆绑，这在 Internet 上很重要，因为不知道要花多长时间才能找到所需数据。如果对 Internet 检索使用同步捆绑，应用程序将被挂起并给客户机造成许多问题（更不用说用户必须坐在那里等待，傻看着没有变化的应用程序）。URL Moniker 还提供了一种建立并使用 URL 的框架，这是 OLE 2 规范没涉及的领域。如果你要用标准的 OLE 2 应用程序访问远程数据，那么它必须支持网络上的 UNC（统一命名约定）设备定位。这意味着可能有某种映射的设备，但很显然，Internet 定位是不可能做到的。

创建 URL Moniker 需要三部分：客户机、系统和传输。客户机部分已经常驻于应用程序中了，本章“使用 URL Moniker”一节将进一步讨论客户机端。系统组件指的是操作系统的一部分（你不必担心这部分，因为 Microsoft 负责处理它）。传输简单地指出你想怎样将信息从服务器发送给客户机。这部分是可变的，它取决于你要实现什么传输方式。在 Internet 上一般使用 TCP。

注 URL Moniker 的三部分是客户机、系统和传输。

就像本书中到目前为止讨论的其它特殊的 OLE 功能一样，实现 URL Moniker 要求你创建或使用某些特殊接口。URL Moniker 的客户机、系统和传输三个组件中各有两个接口。下面将对每个接口一一说明。

注 和其它每种 ActiveX 编程技术一样，使用 URL Moniker 也需要增加特殊接口。

注 请谨记，Windows 还提供了实现 URL Moniker 所需要的两个系统级接口。

- IEnumFormatETC（客户机） 它是可选接口，允许你提供影响捆绑操作的特定协议信息。例如，要提供 MIME 功能，就需要提供该接口，以便列举应用程序所支持的格式。
- IBindStatusCallback（客户机） 传输将使用该接口来通知客户机的特定事件（如下载过程进度）。更重要的是，一旦从客户机经由系统到传输的捆绑完成，传输即利用 IBindSystemCallback::OnStartbinding 方法传回一个 IBinding 接口。
- IBindCtx（系统） 你在开始查找 Internet 资源时，都必须使用 CreatAsyncBindctx 这个 ActiveX API 调用，给 IBindStatusCallback 接口

（也可经给 IEmumFromatETC 接口）传送一个指针。它将返回一个指向 IBindCtx 接口的指针，负责完成客户机和传输之间的实际捆绑。

- IMoniker（系统） 该接口元素有许多用途。它的主要功能包括：使用 GetDisplayName 方法，按人们可读的格式，获取 moniker 名称，以及使用 BindToStorage 或 BindToObject 方法传输具体例子。
- IParseDisplayName(传输） 由 ActiveX 支持的 MkParseDisplayNameEx() 函数的当前版本允许你根据文件或者 URL 创建 Moniker。不过你可能会发现，需要根据其它类型的对象创建 moniker。该接口允许你的应用程序与传输一起创建新的 moniker 类型。你还应该将这些新 moniker 类注册到注册表中。
- IBinding（传输） 系统创建一个由应用程序想要访问的协议类型所定义的传输。IBinding 就是产生的专用协议传输接口。它通过 IBindStatusCallback 接口，负责分析协议串、 驱动任何下载的数据、以及给应用程序提供状态信息。应用程序能利用 IBinding 接口启动、终止、暂停或恢复捆绑操作。

高级技巧

端口号与 URL 的关系

我不打算花费过多的时间去让你们学习那些可以留给工程师们解决的问题。但是，如果能在碰到硬件/软件连接问题时，知道一点关于 URL 中的“http”是如何工作的信息，那么，会使你工作起来更方便。Internet 上使用端口号的概念。也就是说，你在与 IP 地址 200.100.100.1 连接时，可以选择这个地址的某个特定端口。

实际上，这个概念是从人们在 Internet 上使用主机排斥一切的年代遗留下来的。你可以将某个终端插入主机背后的特定端口以获得特定服务。幸运的是，你可以对 PC 使用同样的概念。如同机器上的串口和并口一样，Internet 上的端口也有专用目的。

从软件角度也很好理解。任何在机器上做过汇编级编程的人都知道，必须访问特定端口号才能获取键盘信息。即使你没做过汇编程序，也可能使用特定端口来设置某个卡（如声卡）。任何想使用该声卡的软件（如游戏），都需要知道声卡的端口号才能发出声音。同样如果你想从 Internet 上获取特定信息，也需要端口号。

在 URL 中指定特定协议等同于申请特殊端口号。老的通讯程序要求你输入端口号；谢天谢地，URL 已经解决这个问题。例如，当你在 URL 中输入“http”时，实际上是请求访问主机上端口号 80。(其它如 TelNet 使用端口 23，某些交互游戏使用端口 4201。总之，每个协议都有一个特定端口号）。记住“http”比记住使用端口 80 要容易得多。由于 Internet 上的每个人都使用同样的端口号请求特定的公共服务，所以在 URL 中可以使用 http。

好了，现在来看看你为什么要知道这些信息。网络管理员并不总是使用服务的缺省端口号。你可以为 Internet 上特定的公司服务加入一个特殊端口号——只要服务器软件允许这样设置就行。尽管使用特殊端口号并不能代替防火墙和其它安全帮助，它仍不失为在 Internet 服务器上设定一块不容别人窥视的领地的一种好方法。在做某些配置工作如设置防火墙的过滤器时，知道端口号也会带来方便。最后，尤其对于程序员来说，如果一切看似正常，但却不能正常运行，那么了解 Internet 连接是如何工作的，将有助于解决应用程序的这种问题。

12.2 创建 URL Moniker

和其它类型的 OLE 通信一样，创建 URL Moniker 也遵循着一个相当标准的过程。这里存在着几级通信，可以选择是否完整地实现某些接口

(IEnumFormatETC)。下面的步骤使用最少的接口元素概述了一个典型的通信会话。换言之，该过程讲述的只是经常看到的东西，也包括了提高灵活性或提供附加信息交换等内容。

1. 创建捆绑环境。可以利用 ActiveX API 提供的 CreatAsyncBindCtx()函数。该调用要求为应用程序的 IBindStatusCallback 函数提供一个指针。没有该接口，系统和传输就没办法与应用程序进行通信。如果应用程序需要实现特殊功能，如 MIME，那么你还需要给可选的 IEnumFormatETC 接口提供一个指针。格式枚举器允许服务器和传输确定应用程序能处理什么数据格式（它们总假定应用程序能处理文本数据）。

2. 系统将一个指针返还给它的 IBindCtx 接口。该接口还通过捆绑环境将 IBindStatusCallback 进行注册。

3. 一旦应用程序有一个 IBindCtx 需要处理，它就能通过调用 CreatURLMoniker( )或 MkParseDisplayNameEx()函数来创建 URL Moniker。选择使用哪个函数很简单。如果你有 moniker，则使用 CreateURLMoniker()函数。否则就使用 MkParseDisplayNameEx() 函数来分析文本字符串，并根据它创建一个 Moniker，然后将信息传给 CreatURLMoniker( )函数。

> **注释** Visual C++老版本的用户如果要实现 MkParseDisplayNameEx()函数，需要 MSDN 杂志提供的 Win32API 升级版，这种支持已经封装在 Visual C++ 5.0 和 6.0 软件包中了。

4. 下一步是利用系统的 IMoniker 接口创建能处理 URL Moniker 的传输。有两种方法可以使用：BindToObject()和 BindToStorage()调用允许你将指向

BindToObject() 调用允许你将指向 moniker 的对象实例化。该对象允许你与 Moniker 进行交互。一般是在 Web 站点上或其它活动连接中使用它。调用 BindToStorage()可获取由 moniker 指向的数据并保存它。该过程通常称为“捆绑”Moniker。实际上是你获取信息并存放到磁盘上的。你使用该调用类似于使用 FTP 站点。两情况下都要把在第二步中获得的指针作为调用的一部分传给 IBindCtx。

5. 系统的 IMoniker 接口将为你装载一个专用传输服务程序，而不管你请求什么样的协议。它会根据你传送的 Moniker 来确定使用什么协议。一旦专用传输服务程序被激活，它就使用 IBindStatusCallback::OnStartBinding()方法将指向 IBinding 接口的指针传送给应用程序。

6. 现在，在所有三个元素：应用程序、系统和传输间建立了通信，并且为应用程序提供了处理其它元素的两个接口（IBindCtx 和 IBinding）。应用程序能够开始下载文件、Web 页面或其它资源了。关于通信难题还有一点要说明。无论何时应用程序申请数据时，传输都会用 IBindStatusCallback::OnDataAvailable()方法告诉它下载的进度。应用程序利用该方法传递的状态信息确定下载是否完成（也可以用这些状态信息为用户更新进度栏或其它状态指示器）。

12.3 超链接基础

URL moniker 是从 Internet 上一个地方到达 Internet 上另一个地方的 OLE 实现的底层部分。还有另一个高层接口，它就是用户可以看到的接口。无论何时，如果在浏览器中看见带下划线的文本，那么就可能是看到了另一站点的超链接。

本书好几章都已经讨论了该过程的HTML部分（参见第八章HTML概述）。

注 URL moniker接口的用户部分可以和Web页上的超链接一样简单。

Microsoft提供了一种ActiveX超链接接口，你可以利用它模拟ActiveX控件内专用的浏览器活动。和其它形式的OLE一样，这意味着可以将浏览器技术扩展到应用程序中或者简单地用于HTML页面，以便给ActiveX控件提供另一附加级别的功能。无论在什么地方使用ActiveX超链接，都可以完成包括下述内容在内的一些基本服务。

- 应用程序到应用程序。没有浏览器时，你也能在两个应用程序间创建超链接。如果需要在自己的LAN上实现某种内部网站点，这种功能就特别方便。
- 应用程序到ActiveX/HTML文档。第11章已经介绍了ActiveDocument的工作原理。文档实际上是在浏览器窗口中显示的。使用标准的HTML<A HREF>标记完成从Internet到文档的连接。如果能在应用程序中放一个控件，并且它能自动与Internet上的文档相连接，会是什么样的情况呢？也就是说，该链接的作用等同于反向的HTML< A HREF> 标记。ActiveX超链接控件就能帮助你完成这个任务。例如，可以利用该功能提供从公司政策文档到Internet上其它信息源的链接。
- ActiveDocument到ActiveDocument。你一般是在浏览器中查看文档。如果能在文档中使用锚地，简单的从一个地方转移到另一地方（就像在HTML页上移动一样），事情会变得更好。ActiveX超链接允许你完成这样的功能。你利用该功能从一个文档转移到另一文档，就像在Internet

上从一个 HTML 页面转移到另一 HTML 页面一样。

- 从浏览器 ActiveDocument 到 Microsoft Office Binder ActiveDocument。公司在本地内部网上对使用该功能有潜在的需求。它允许用户把浏览器中显示的文档转移到本地机器上的一个 Microsoft Ofiice Binder 文档。例如，可以利用该功能提供与本地术语汇编的链接。无论何时，如果用户不知道某个词的意思，都可以沿着位于 Internet 上的链接，到达术语汇编的本地拷贝，从而弄清那个词的含义。

注 导航栈(navigation stack)是在当前会话中访问的 URL 列表，它由浏览器维护。

不过，在你使用上述任一种技巧之前，都必须首先创建 URL moniker。在前两节已经讨论过这种过程。另外，你还需要 ActiveX API 提供的超链接接口。幸运的是，你不必求助于诸如为 URL moniker 等提供的那些低级接口。如果你只是希望从 A 点到 B 点，你就可以利用 Microsoft 已经提供的下列四个函数来完成。

函数	描述
HlinkSimpleNavigateToString()	该函数允许你到达基于字符串的新地点，该字符串的格式与你在浏览器中输入的字符串格式一样。通常可用该函数到达一个新的 HTML 文档或者同一 HTML 文档中的另一位置，但它也适用于对象
HlinkSimpleNavigateToMoniker()	该函数利用 moniker 作为信息源，允许你从一个地方转移到另一地方，通常它适用于对象
HlinkGoBack()	该函数模拟浏览器工具栏上的 Go Back（退回）按钮。它将你送回导航栈所指的先前位置
HlinkGoForward()	该函数模拟浏览器工具栏上的 Go Forward（向前）按钮。它将你送到导航栈中的下一位置

技巧：这些简单的超链接函数都放在 URLHLINK.H 头文件中，（对 Visual C++ 5.0 以下版本的用户，这个文件由 ActiveX SDK 提供），你会在 URLMON.DLL 中看到实际的函数。

这些函数的使用是很简单的。首先介绍比较容易理解的函数：HLinkGoBack()和 HLinkGoForward()，这两个函数都只需要一个参数，即指向当前文档的 IUnknown 接口的指针。可以利用 ExternalQueryInterface( )函数来获取该指针，其中第一个参数是 IID_IUnknown，第二个参数是一个指针变量。

HLinkSimpleNavigateToString()和 HLinkSimpleNavigateToMoniker()函数稍复

杂一点。下面的命令行说明了它们所需的参数。

```
HLinkSimpleNavigateToString(szTarget, szLocation, szTargetFrame, pIUnknown,
        pIBindCtx, pIBindStatusCallback,grfHLNF, dwReserved)
HLinkSimpleNavigateToMoniker(pmkTarget, szLocation, szTargetFrame, pIUnknown,
        pIBindCtx, pIBindStatusCallback,grfHLNF, dwReserved)
```

除第一个参数外，这两个函数使用了同样的参数表。HLinkSimpleNavigateToString()的第一个参数是一个包含着将要访问站点的 URL（或其它标识信息）的串。HLinkSimpleNavigateToMoniker()的第一个参数是指向 moniker 的指针。

现在看看公共参数。szLocation 包含同一页上某个位置的名称。第 8 章已经介绍了怎样做到这一点。szTargetFrame 参数包含文档内某个框架的名称。它是 HTML 专用的，并且只适用于使用框架的页。这两个参数都不是必须的，仅在需要修饰一些扩展的导航信息时才需要提供它们。不想包括它们时可以将它们的值设置为 NULL。

另外，还有三个接口专用参数：pIUnknown，pIBindCtx，IBindStatusCallback。这里只有 pIUnknown 参数是必须的，它可以利用 ExternalQueryInterface()函数得到。如果你想提供另外的处理信息，就要指定 pIBindCtx 值。例如，如果你的应用程序提供 MIME 功能，就可能利用该参数传递 IEnumFormatETC 接口的位置（它是作为 IBindCtx 参数的一部分传递的，但是，不要给 IEnumFormatETC 接口本身传递指针）。如果在传送中需要状态信息，那么，还要给应用程序的 IBindStatusCallback 接口传递一个指针。

剩下的就是 grfHLNF 参数了。它允许你指定浏览器的特定行为。例如，如果将该参数设置为 HLNG_OPEN_INNEWWINDOW，那么浏览器会在新窗口中显示指定站点。在 HLINK.H 头文件（对于 Visual C++ 5.0 以下版本的用户，该文件由 ActiveX SDK 提供）中有 HLNF 值的完整列表。

12.4 理解超链接接口

在某些情况下，前面讨论的简单的超链接函数调用还显得不够用。例如，你可能想实现一个完整的浏览器，把它作为应用程序的一部分。另一种情形是，如果你想提供这些调用不支持的某些扩展功能，比如操作历史列表等，这时简单调用就显得不够用了。前面讨论的调用特别适用于只关心导航的情形。

对任何 ActiveX 控件，都需要提供一些附加的接口，以便在应用程序中实现超链接。实际上有五个新接口，如下所述：

- IHLink（必选项） 这是 ActiveX 超链接接口的中心。它是一个提供了另一个需要进行超链接的应用程序所需的所有信息的接口，包括目标 moniker（实际对象标识）、URL 风格的字符串以及一个友好的名称。它还能提供其它类型信息，但创建简单的超链接一般不需要。所有简单链接都是通过 Navigate 方法完成的，它正是前面讨论的简单函数使用的方法。
- IHLinkTarget（可选项） 利用该接口可以在文档中定位或者下载其它信息。例如，该接口允许你在电子表格中指向特定单元格。IHLinkTarget

还提供了一种将指向文档的 IBrowseContext 接口的指针传递给调用的应用程序的手段。应用程序可以链接到一个不提供支持该接口的某一文档，但也只能如此。

- IHLinkFrame（可选项） 记住，在 OLE 中，框架是 OLE 容器中的一个应用程序容器。它包含菜单和外部框。该接口允许框架维护与 OLE 容器（通常为某种文档）的联系。
- IHLinkSite（可选项） 这是管理超链接站点的接口。它有两项基本任务，并能提供并发服务。该接口的第一个主要任务是，获得对文档的 IHLink 对象的访问权。访问该对象能提供许多信息，如当前站点名。接口的第二个主要目的是，提供对客户机应用程序的反馈。例如，OnNavigationComplete 方法能告诉客户机下载或其它任务已经结束。
- IBrowseContext（可选项） 应用程序通过该接口获得对浏览器环境对象的访问。所有浏览环境对象的全部工作就是追踪导航栈——即当前会话期间访问的站点。前文所述的 IHLinkGoFoward()和 IHLinkGoBack()函数正是根据该接口来完成它们的工作的。

实际上，在这一组接口中有两级实现。我们认为第一级是应用程序，在这里是指浏览器。第二级是一个 OLE 容器，在这里是指文档。浏览器实现 IHLinkFrame 和 IBrowseContext 接口。这两个接口只会有一种情况发生，即浏览器要么实现它们、要么不实现。文档负责实现 IHLinkTarget、IHLink 和 IHLinkSite 接口。

用接口术语来定义文档有点困难。文档的容器部分实现 IHLinkTarget 接口，

所以，即使是处理一个包含多个对象的文档，你也只能看到其中之一。同样，无论文档包含多少对象，每个文档也只有一个 HLinkSite 接口。但你可以在一个文档中包含多个超链接。

让我们快速浏览一下超链接。每个超链接都需要单独的 IHLink 接口，可以用四个函数来建立：IHLinkCreatFromMoniker( )、IHLinkCreatFromString( )、IHLinkCreatFromData()以及 IHLinkQueryCreatFromData()。HLink 对象可以使用持久数据，也就是说控件必须实现一个 IPersistStream 接口。因为你可以从数据中创建一个 HLink，所以就可能利用 Clipboard 来剪切，拷贝并粘贴它。另外，因为 IHLink 可以使用 moniker，所以你可以指向任意类型 OLE 文档，包括那些一般不在浏览器中显示的文档，如 Word for Windows 文件。

12.5 使用 URL Moniker

今天，许多公司特别关注 Internet。因此，在不久的将来，访问 Internet 上的信息，一定会像访问本地硬盘数据一样容易。实际上，随着 Internet Explorer 4.0 中包含了部分 Visual C++ 6.0 特性，Microsoft 已经向上述目标迈进了一步。你将会发现硬盘和 Internet 上你最喜欢的站点正共享着一个公用的 Explorer 视图。

WEB 链接 获取自己需要的信息经常是指访问自己的本地新闻组，以获得其它的程序员的帮助。不必多说，你会发现有些新闻组比别人花更多的时间在讨论 URL moniker；毕竟它还算是一个神秘的课题。例如，microsoft.public.win32

.programmer.ole 新闻组很长时间以来一直在讨论将 Moniker 捆绑到专用控件上的话题。甚至还有一个新闻组专门用于 URL moniker: microsoft.public.activex.programming.urlmonikers。如果你想从新闻组中的人那里获得 Visual C++有关的帮助,可以看一看 microsoft public.vc.mfcole。

当然，数据站点的这种界线模糊倾向(blurring)对用户和程序员都会提供一些新机遇。例如，作为用户你会发现，将你需要的数据放入你需要的应用程序中去进行编辑要省事一点。作为程序员你会发现，现在还有许多别的方法来处理一个程序，并使它产生特殊的效果。

注释 如果你不是使用 Visual C++5.0 以上版本来创建本节的例子，那么，就必须安装 Activex SDK。SDK 提供了能使示例代码正常工作所需的 URLMON.H MLINK.LIB 和 URLMON.LIB 文件。

使应用程序具有特殊效果的一种最普通方法就是给它添加超链接功能，而最容易的添加方法是，在应用程序的工具栏上，放置一个可以使用户进入公司内部网的按钮。同样，你可以使用另一按钮，允许用户选择特定站点或在公司授权的 Internet 站点列表中进行选择。这正是本节要介绍的内容。样本程序将说明如何给某个典型的应用程序加入两个按钮，从而使用户可以快速简单地链接到 Internet 上。

第一步当然是创建一个新的应用程序。就像第 11 章中的 ActiveDocument 例子一样，你也可以使用老版本的 Microsoft Visual C++编译器来创建这些代码。本节假定你使用的是 5.0 以上版本，尽管用 4.2 版本创建它相对容易些。由于

第 2 章已经介绍了使用 MFCApplication Wizard 创建应用程序的过程，所以这里只进行简短的描述。在创建过程中，你需要选择一些选项才能使例子正常工作。首先，例子的名称为 ViewURL（尽管你可以选择自己喜欢的名字）。其次，在 Wizard 的第 1 页应选择 Single Document 。第三，在 Wizard 的第 4 页上应选择 Windows Sockets 支持。最后，在 MFC AppWizard 的第 6 页将 Base Class 域设置为 CEditView，这将允许你在示例应用程序的主窗口中编辑文本。

技巧 还有许多情况像本例一样不需要 CRichEditView 类的全部功能。与 CEditView 相比，CRichEditView 类不仅相当大，而且使用 CRichEditView 类还要求给应用程序添加 OLE 容器支持功能，这就进一步增大了应用程序的规模。CEditView 类既能让你显示无格式文本，又能使应用程序比较小——这正是实用类应用程序一个很重要的考虑因素。

添加库支持

一旦 Wizard 完成了应用程序创建工作后，你就可以开始做进一步的修改了。首先要完成的工作就是给应用程序加入超链接支持。为此，可在 STDAFX.H 文件中加入下述的#include。

```
//Added for URL support.
#include "URLMon.h"
```

该头文件包含了你在使用各种 URL moniker 有关命令时所需的所有#define

语句。本章前面已经介绍了其中一些命令，这里将介绍它们是如何工作的。

注释 老版本的 Visual C++ 用户要做的下一件事就是添加一些静态的支持（但是 Visual C++ 5.0 以上的用户不必做这一步）。HLINK.LIB 和 URLMON.LIB 这两个库都在 ActiveX SDK 的 LIB 文件夹中。可以利用 Visual C++ 的 Insert（插入）| Files Into Project（文件到项目）命令给应用程序添加所需的库支持。通过查看 FileView 中的包括文件列表，来检查是否真的添加了支持（只需单击显示工作区左边的 FileView 标志即可）。

技巧 Microsoft 打算将来把 HLINK.LIB 和 URLMON.LIB 中的静态库支持放入 URLMON.DLL 中。一定要在编写支持 URL moniker 的应用程序之前检查该动态库的支持功能。

创建所需资源

现在已经加入了所需的库支持，让我们给工具栏添加一些按钮。单击 Resource View（资源视图），打开 Toolbar 文件夹，然后双击 IDR_MAINFRAME 项，就能看到如图 12.1 所示的标准工具栏。添加新按钮很容易，你只需单击工具栏末尾的空白按钮，然后开始在绘画区域中显示的空白按钮上绘制。把这个按钮稍微向右移动一点，就能将它与工具栏上已显示的其它按钮区分开。图 12.1 显示的是这个例子中添加的两个按钮。

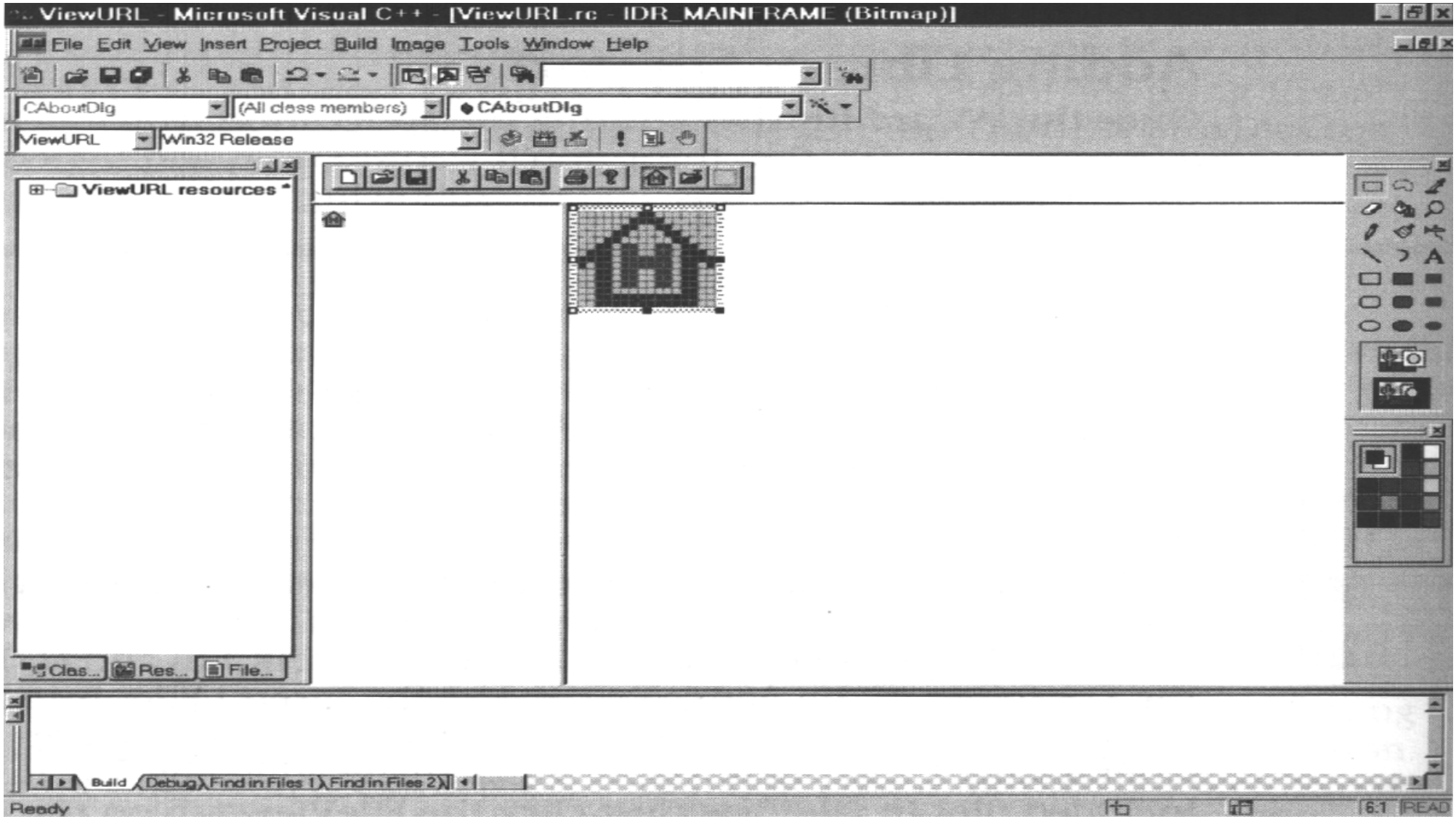
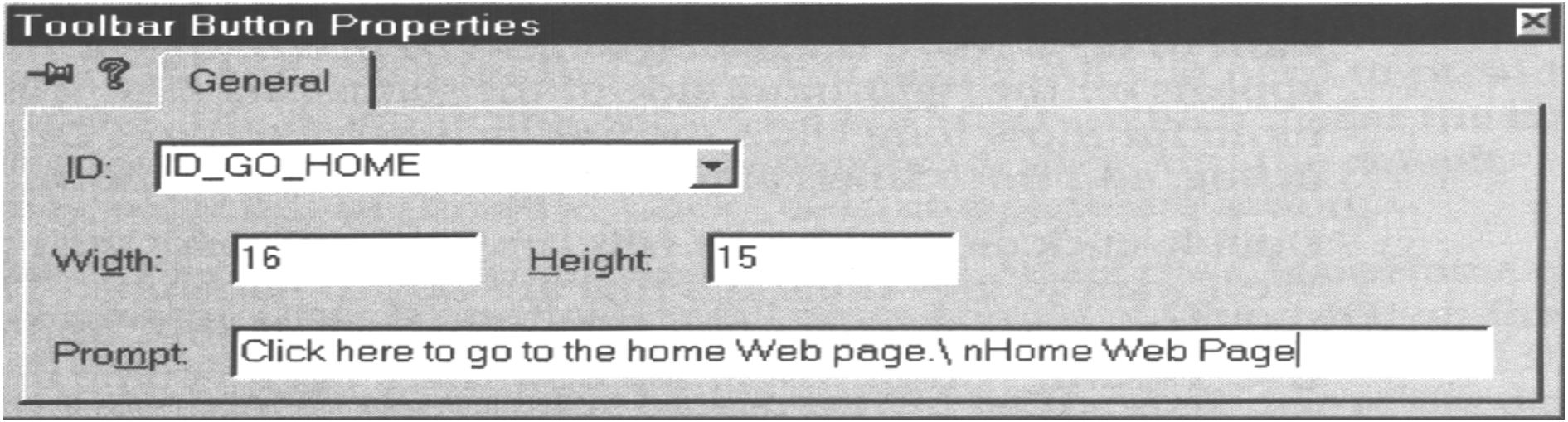


图 12.1　具有 Home Page 和 Any Web Page 按钮的标准工具栏

注释 增加按钮可以给用户提供一种可视图化的显示，另外，还要给应用程序添加一些按钮标识信息。简单地双击将要设置的按钮的工具栏(不要单击刚创建的按钮图标)，就会看到如下图所示的对话框。对两个按钮使用下表进行的设置。

标识符	描述	高度	宽度
ID_GO_HOME	Click here to go to home Web page.\nHome Web Page	15	16
ID_GO_SITE	Click here to go to any Web page.\nAny Web Page	15	16

技巧 注意 Caption 包含了一个较长的描述、一个\n 以及一个较短的描述。较长的描述将出现在应用程序的状态栏上。较短的描述用于当鼠标停留在该控件时的弹出帮助。一定要用“\n”分隔这两项。在 Visual C++ 应用程序中任何看到弹出式帮助的地方都可以使用这种特殊的输入方法。



单击第一个按钮将把用户带入公司的主页。按钮已经起作用，在资源方面不必再做什么了。你需要添加的只是完成实际工作的一点代码。但是，第二个按钮将允许用户输入他们自己的站点，如果愿意的话，还可以输入该站点中的某个位置。该按钮还支持那些使用框架的站点上的框架，这就需要添加一个特殊的对话框。打开 ResourceView 中的 Dialog 文件夹，用右键单击 Dialog 文件夹，就能看到一个关联菜单。选择 Insert Dialog 项，就能看到一个名为 IDD_DIALOG1 的新对话框添加到了文件夹的列表中（应用程序当前提供的唯一一个对话框是 About Box 对话框）。

现在给对话框改名。用右键单击 IDD_DIALOG1 项，然后从关联菜单中选择 Properties。在 ID 字段中输入 IDD_SITE_SELECT，然后单击属性对话框，就能看到新名字出现在 Dialog 文件夹的对话框中。

给对话框添加所需控件相当容易。首先要做的是将对话框大小改为 25×120

像素点（即出现在状态栏右边的当前对话框大小）。这可能要占一点空间，因为用户可能想输入很长的 URL。改变对话框大小后，可以添加三个标签和三个编辑控件，如图 12.2 所示。

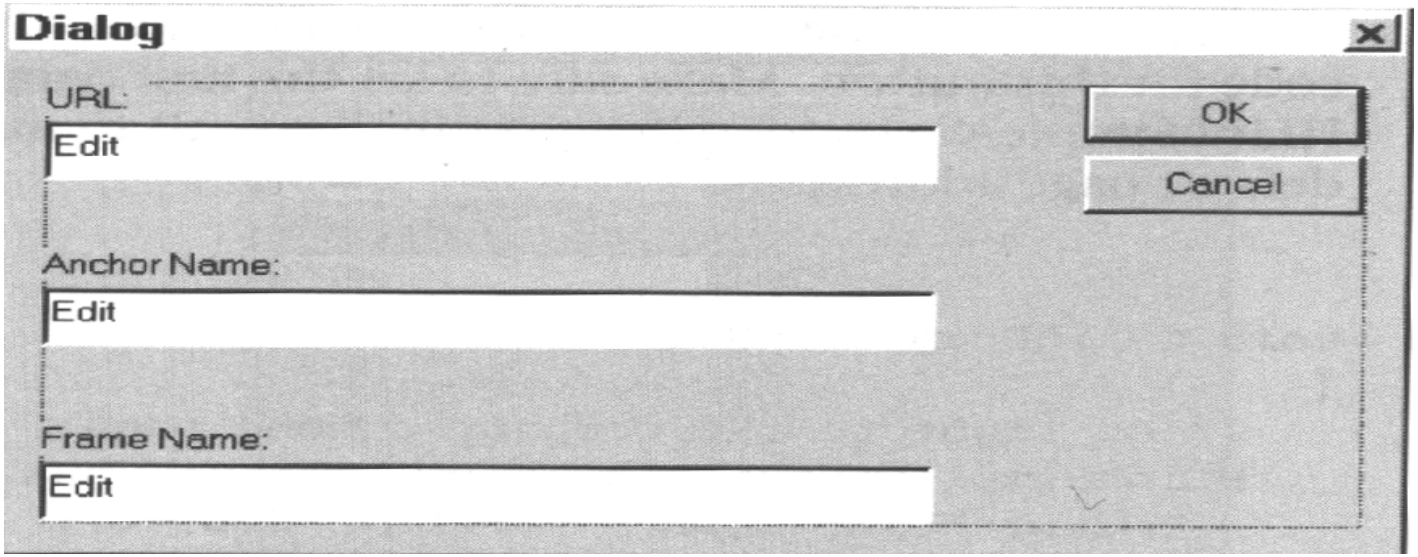
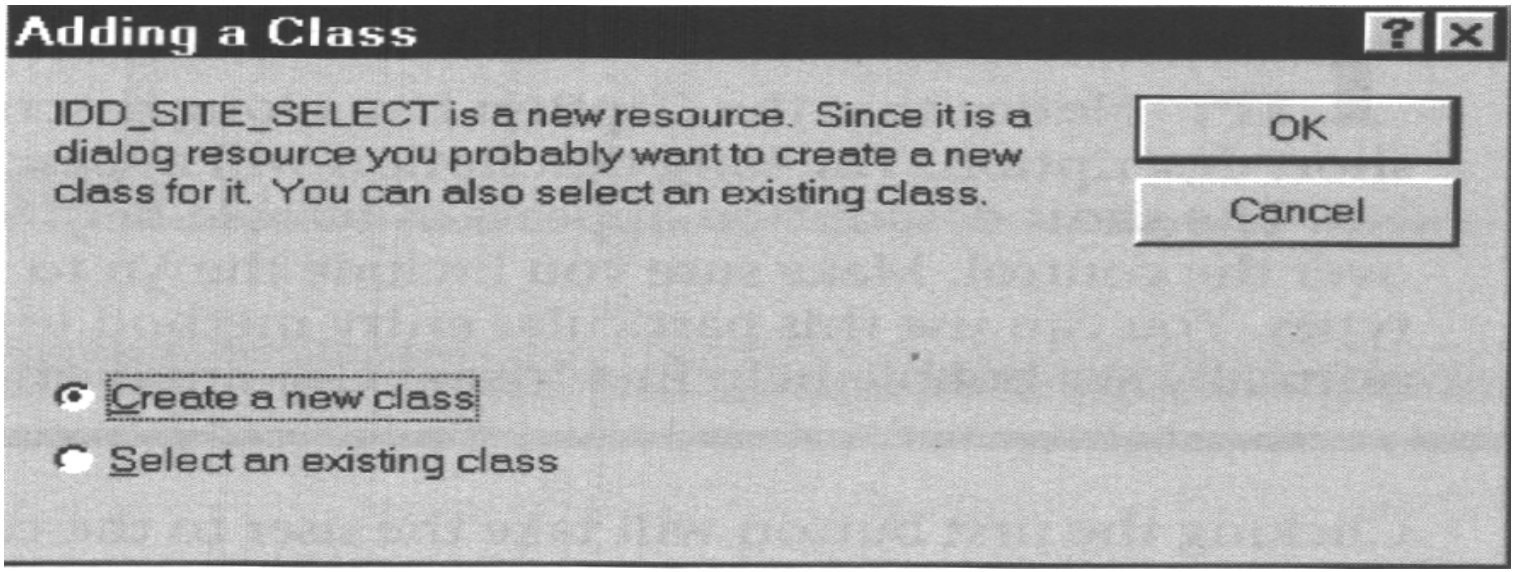


图 12.2　如果用户选择访问主 Web 站点之外的地方，示例程序将显示该对话框

双击每个编辑控件显示它们的属性对话框，设置一个易于记忆的 ID。第一个编辑控件的 ID 为 IDC_URL，第二个编辑控件的 ID 为 IDC_ANCHOR_NAME，第三个编辑控件的 ID 为 IDC_FRAME_NAME。稍后将会介绍这些名称的作用。

定义新类并编写代码

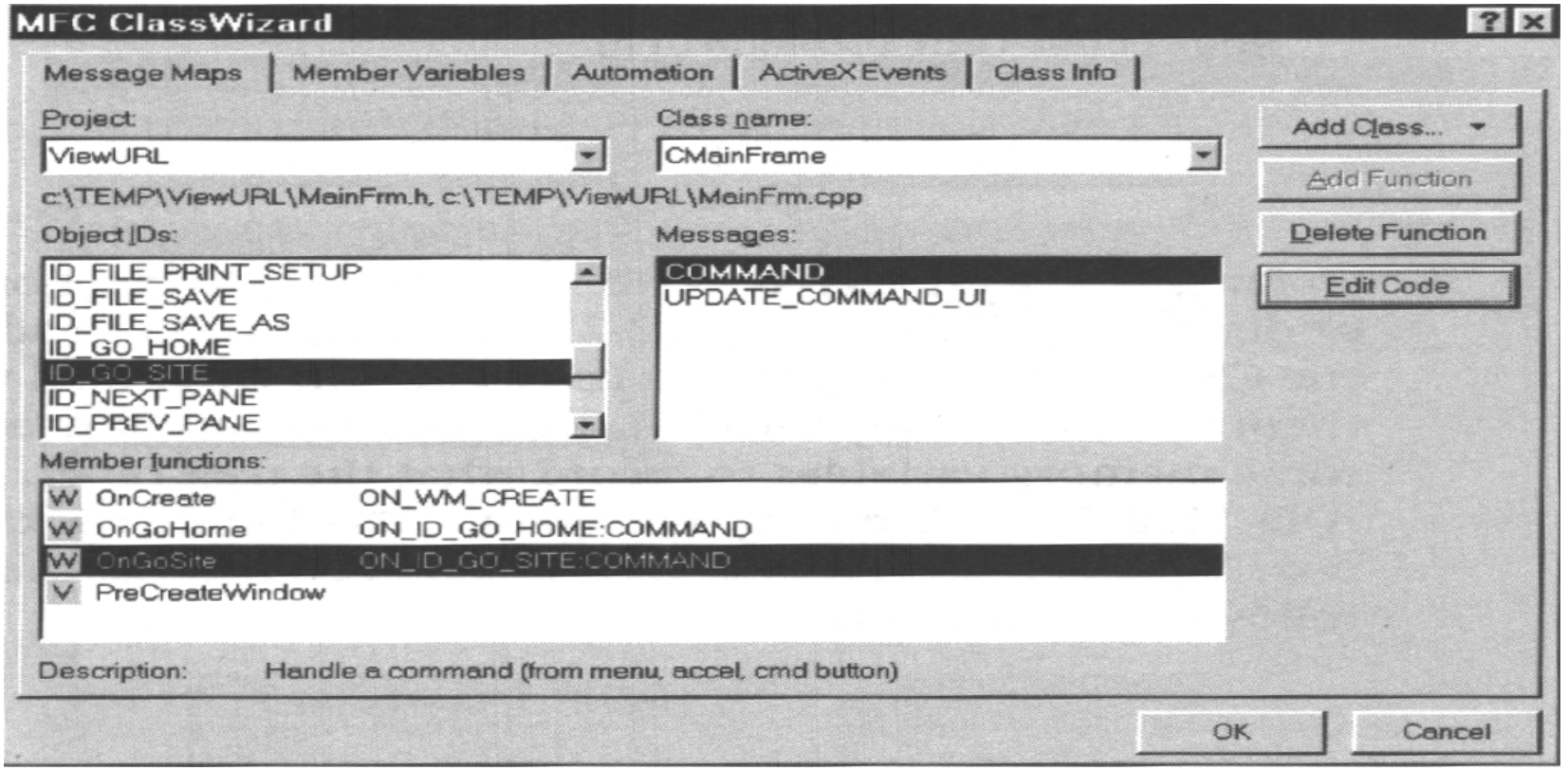
现在已经定义了所有所需的资源，应该给添加到工具栏中的两个按钮增加代码了。确保已经选择了 IDD_SITE_SELECT 对话框，然后用 View|ClassWizard 命令显示 ClassWizard 对话框。在这种情况下，你将看到如下图所示的对话框，它表明 IDD_SITE_SELECT 是一个新对话框，需要为它创建一个类。



单击 OK，你将看到 New Class 对话框。你需要提供的只是类名。示例程序使用的是 CSiteSelect，它使得类名及其相关资源很易于识别。本例使用的是缺省的 CDialog 基类。单击 OK 完成操作。

选择 ClassWizard 的 Message Map 页，然后从 Class Name 字段中选择 CMainFrame 类名。滚动下拉对话框左边的 Object IDs 直至找到 ID_GO_HOME。

单击该项，然后单击对话框右边的 Message 字段中的 COMMAND 项。单击 Add function 给应用程序添加所需函数。此时会看到 Add Member Function 对话框。单击 OK 接受缺省函数名。同样处理 ID_GO_SITE 对象标识符。完成之后，MFC ClassWizard 对话框如下图所示。



选择 OnGoHome 项并单击 Edit Code。MFC ClassWizard 就会直接进入新函数框架。程序列表 12.1 显示了该按钮的代码，确保将 HlinkSimpleNavigateToString()函数调用的第一个参数与 Web 服务器的缺省页面

地址相匹配。你会看到 OnGoSite()函数就在 OnGoHome() 函数之后。给它添加程序列表 12.2 所示的代码。确保在 MainFrm.Cpp 文件开头包括了 SiteSelect.H（如果愿意的话，还可在 STDAFX.H 文件中包含它）。

程序列表 12.1

```
void CMainFrame::OnGoHome()
{
        // Go right to the company's home page.
        HLindSimpleNavigateToString(L"http://aux/default.htm",
                NULL,NULL,NULL,0,NULL,NULL,0);
}
```

程序列表 12.2

```
void CMainframe::OnGoSite()
{
        //Create a copy of the dialog.
        CSiteSelect   NewSiteSelect;

        //Display it.
        NewSiteSelect.DoModal();
}
```

现在已有足够的代码可以将用户送往主页（程序列表 12.1）或显示对话框（程序列表 12.2）上了。不过还需要给 CSiteSelect 对话框增加一些代码。当用户单击 OK 时，你可能想把他们带到他们选择的其它站点。首先要添加一些内存变量来记录用户的输入。再来看一看 IDD_SITE_SELECT 对话框。按下 CTRL 键并双击第一个编辑框，就能看到如下图所示的 Add Member Variable 对话框。

它允许你给 IDC_URL 编辑框赋予一个变量。本例中命名为 URL（该名字将会在对话框中以 m_URL 出现）。选择 Category 域中的 Control，把 Variable Type 域改变成 CEdit，CEdit 就是该例所需的变量类型。单击 OK 完成操作。同样处理另两个编辑框，并将第二个变量命名为 Anchor，第三个命名为 Frame。

有了变量，现在还要添加另一个函数，用 View|ClassWizard 命令打开 MFC ClassWizard。对话框应该仍然显示 Message Map 页；否则的话就选中该页。在 Class Name 域中选择 CSiteSelect 项，就会看到对话框左边的 Object IDs 列表变为 CSiteSelect 对话框提供的内容。选中 IDOK 对象标识符和 Message 域中的 BN_CLICKED 项。单击 Add Function，然后单击 Add Member Function 对话框中的 OK。这时 MFC ClassWizard 对话框如下图所示。

现在，单击 Edit Code 显示函数的框架。该函数接收用户在对话框中的输入信息，然后将它们带到 Internet 上的特定位置。程序列表 12.3 显示了完成该任务的代码。

程序列表 12.3

```
void CSiteSelect::OnOk()
{
```

```
LPCWSTR     pszURL = NULL;       //local copy of URL string.
LPCWSTR     pszAnchor = NULL;    //local copy of Anchor string.
LPCWSTR     pszFrame = NULL;     //local copy of Frame string.
LPTSTR      pszBuffer = "";      //Conversion buffer.

// Get the URL string and convert it .
if (m_URL.LineLength(0) > 0)
{
        m_URL.GetLine(0,pszBuffer,m_URL.LineLength(0));
        pszURL = LPCWSTR(pszBuffer);
}
else
{
        MessageBox("You must enter a URL","Error",MB_OK);
        return;
}

//Get the Anchor string and convert it.
if (m_Anchor.LineLength(0) > 0)
{
        m_Anchor.GetLine(0,pszBuffer,m_Anchor.LineLength(0));
```

```
            pszAnchor = LPCWSTR(pszBuffer);
        }

        //Get the Frame string and convert it.
        if (m_Frame.LineLength(0) > 0)
        {
            m_Frame.GetLine(0,pszBuffer,m_Frame.LineLength(0));
            pszFrame = LPCWSTR(pxzBuffer);
        }

        //Go right to the selected page.
        HLinkSimpleNavigateToString(pszURL,
            pszAnchor, pszFrame, NULL, 0, NULL,NULL,0);
        CDialog::OnOK();
    }
```

上述代码中有一些小技巧，但大部分 C 语言程序员使用过这些技巧。首先，要将三个用于保存位置信息的变量置为 NULL。这样，即使它们没有什么信息可保存，也能正常运行。这时，代码仍要强制用户提供 URL，只有一种情况不必这么做，即，如果你允许用户跳到他们当前所在页的一个锚地，就可以让 URL 参数仍然是 NULL 并简单地让它们指定锚地名称。保存对话框中信息的方法需要使用缓冲变量。但从 LPTSTR（指向 Windows 或 Unicode 的以 NULL 结尾的

长指针）到 LPCWSTR（指向以 NULL 结尾的 Unicode 串常量的长指针）的转换很简单。这个特殊例程的核心是调用 HLinkSimpleNavigateToString，个函数负责将位置信息转换为真正的 URL。

现在，让我们看一下应用程序是什么样子。和通常一样编译并运行应用程序。如果单击 Home Web Page 按钮，就能看到 Internet Explorer（或其它缺省浏览器）启动了，并把你带到设为主页的任何 Internet 站点。图 12.3 显示了本例按下该按钮的结果（图中显示的主页是作者的 Internet 服务器设置，你的屏幕上应该显示自己的设置）。一定要将程序列表 12.1 中 HLinkSimpleNavigateToString() 调用的地址设置为自己的主 Web 站点，否则就会出现意外结果。

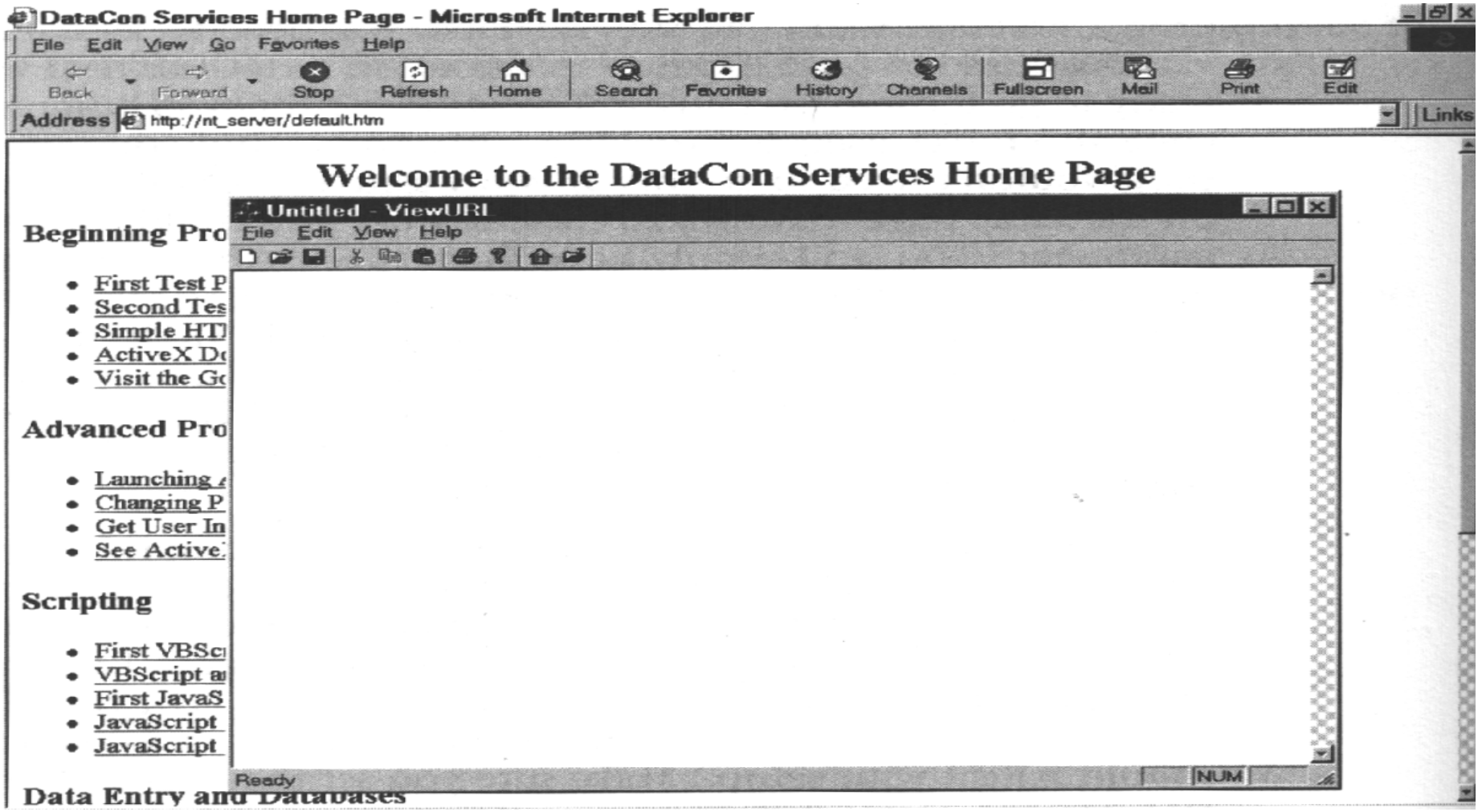
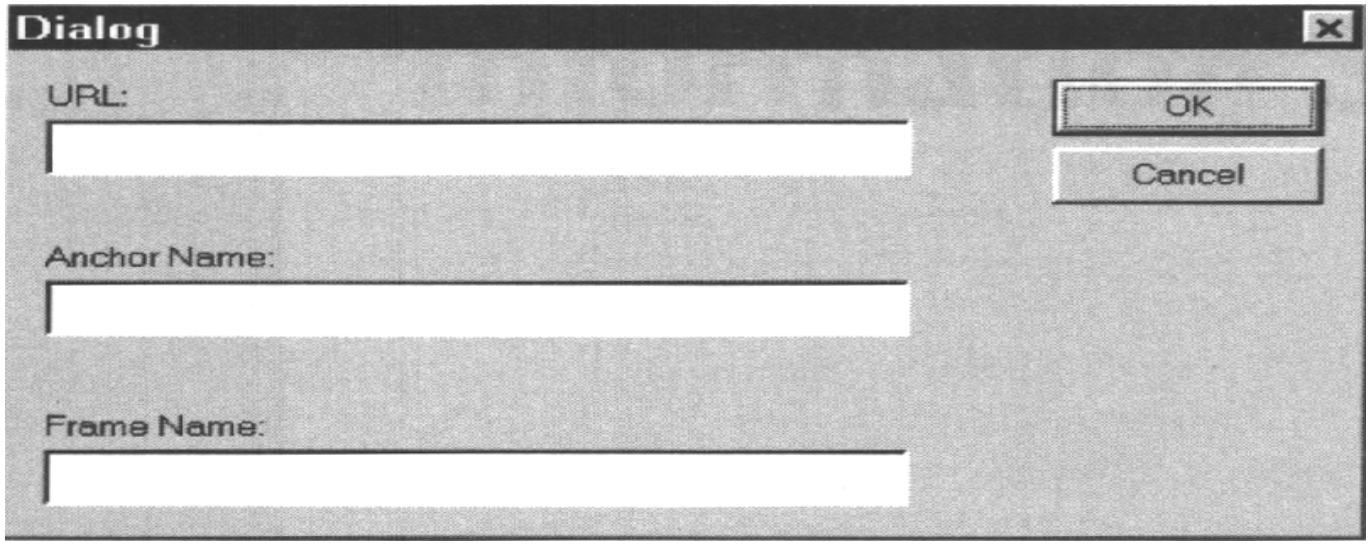


图 12.3　可以给任何应用程序添加这样的按钮，从而允许用户很容易地访问 Internet

单击 Any Web Page 按钮会显示如下图所示的对话框。



它允许输入一个URL、可选的锚地名或框架名。目标Web页必须支持所指定的锚地名和框架名，否则函数调用会失败。如果输入本地测试机器支持的地址之外的URL值，那么应用程序还需要一个活动的Internet连接。

第 13 章　使用 Internet 信息服务器(IIS)

在工作中有许多种不同的方法使用 IIS(Internet Information Server)，其中大部分都是通过传统的静态 HTML 方案，它需要使用各种类型的脚本。例如，可以在 C/C++ 程序中使用 CGI（公共网关接口）脚本查询一个数据库。脚本将会形成一个包含查询结果的 HTML 页并将它返回给客户机。本章将不再介绍这些传统的方法，因为这方面的内容已经有许多书籍介绍过了。需要着重指出的一点是，传统方法之所以仍然存在，一方面是因为它们确实能够有效地工作，另一方面，当我们本章讨论的新技术还没来得及担当重任时，仍然需要使用旧方法。例如，你可能已有一个现存的基础结构，那么试着用新技术对它彻底重新编程就可能不合算。

WEB 链接　有一种很快捷的方式可供学习包括 ODBC 在内的某些 IIS 访问技术的基础知识。Dynamic Systems Internaitonal（国际动态系统）提供了一系列有关 IIS 的课程，你可以从 http://www.dsi.org/dsi/iis.htm 下载它们，这些课程包括了使一些旧的技术访问方法能进行工作的全部内容的概述。你还可以访问 Microsoft IIS FAQ 站点（由 Stephen Genusa 负责主办）。该站点位于 http://www.genusa.com/iis/，它包括了一些像 CGI 这样的论

题。另外，位于 http://www.Adiscon.com/IIS 的 KLV 站点也包括了一些关于在使用老技术时的 Visual C++编程中各种注意事项的链接，这个站点还包括一些其它链接，据此可以找到有关 Internet 制定规范的活动以及安全问题（如用户认证等）的最新信息。

另一方面，使用像 Active Server Pages(ASP)这样的新技术还能提供更大的灵活性并极大地提高你在 Internet 上提供内容的能力。这就是本章要介绍的内容——可用来在自己的 Web 站点上提供更好内容的新技术。另外本章还将介绍一种给 Web 站点用户提供更大灵活性的特殊方法。本章的主要内容就是介绍使用 Internet 服务器应用程序设计接口(ISAPI)的方法，另外还要介绍与 ISAPI 相关的一些支持技术，比如 ISA（ISAPI 服务器应用程序）。学完本章后，对这种技术在现实世界中的工作会从总体上有个很好的了解，清楚自己在什么时候需要使用那些老技术。另外，我肯定要把在自己的服务器上实现这些新技术时应注意的一些问题介绍一下给你。知道这些问题将有助于你决定使用新技术还是使用以前用过的旧方法。

那么什么是 ISAPI 呢？ 就本书而言，ISAPI 是一系列 MFC 扩展，它允许你直接使用 IIS。本章将使用一种新型工程来实现 ISAPI，即 ISAPI Extension Wizard（ISAPI 扩展向导）。另外还将介绍五个新类：CHttpSever、CHttpServerContext、CHttpFilter 、CHttpFilterContext 和 CHtmlStream。我们将使用这些类来创建 ISA——它还有许多其它名称，例如在 Microsoft 文档中就被称为 ISAPI 服务器扩展 DLL。为简单起见本书将使用 ISA 这个名称。

注 ISAPI 是允许你在工作中使用 IIS 的一系列特殊 MFC 扩展。

ISA 必然依赖于 ISAPI。你可以用 ISAPI 类来创建 ISA 扩展以及 IIS 的过滤器。不过并不仅限于 ISAPI 类，还有用于控制 Internet 通信的 WinInet 类以及提供像接口这些东西的所有标准 MFC 类。过滤器允许你通过监测服务器上的事件决定是让它进入还是将它拒之门外。例如，可以创建一个 ISA 过滤器，除非用户输入了正确的口令，否则，将被 Web 站点拒之门外。另一种类型的过滤器，对于文件规模大于某值的文件，拒绝将它装入 Web 站点上的 FTP 服务器。扩展更像是应用程序或后台进程。例如，可以创建一个扩展允许用户不必使用脚本就能与数据库进行交互。同一个扩展可以根据用户的输入和服务器上数据库的内容，动态创建 Web 页。

注 ISA 既可以对信息访问进行过滤，也能扩展 IIS 提供的服务。

在阅读本章前，还应回顾一下第 8 章到第 12 章中讨论的关于客户机的信息。第 8 章帮助你理解 HTML 的基础，第 9 章帮助你理解 JavaScript，第 10 章介绍 ActiveX 控件编程的基础知识，第 11 章帮助你理解 ActiveX Document 的编程，而第 12 章则讨论了非常重要的 URL moniker。所有这些章节都能帮助你了解客户机对服务器的需求——如果你计划修改一下服务器，那么这些信息都是重要信息。第 12 章特别重要，因为它能帮助你理解客户机与服务器之间的连接——即进行对话的支持机制。

注释 本章的代码都是使用 Visual C++ 6.0 开发的，并且没在老平台上进行测试（在 C++ 5.0 中肯定没问题）。尽管这些代码可能不经修改就可以在 Visual C++ 4.2 下工作，但你应在我提供的过程中加以考虑。代码本身如果要在 4.2 版以前的 Visual C++ 上编译，需要进行某种修改。

下面各节将逐一介绍 ISAPI 编程的五个重要方面。首先是 ISAPI 自身简介。这是理论部分，如果你已经对 ISAPI 的工作原理很清楚，并想直接开始创建自己的程序，就可跳过这一节。本书特意将有关过滤器和扩展的创建与它们的实现分开论述，以便更通用。而且创建一个 ISA 与实现它是不同的步骤。本章第二部分介绍如何创建 ISAPI 扩展，第三部分介绍如何在服务器上实现它。类似地，第四部分介绍如何创建 ISAPI 过滤器，第五部分是如何实现 ISAPI 过滤器。

本章最后两节介绍怎样创建其他类型的 ISAPI 扩展和过滤器。第 6 节提供了另一个 ISAPI 扩展的程序设计例子，而第 7 节则提供了第二个 ISAPI 过滤器程序设计的例子。这些例子使你更好地掌握怎样使用 ISAPI 来增强服务器的操作性能、向用户提供更好的输出或者使你的网络更加安全。

WEB 链接 如果你遇到有关 IIS、ASP 或 ISA 的问题，可以访问许多新闻组以获得帮助。实际中的问题太多了，本章不可能将它们一一列出，所以你还是花点时间自己去找吧。要得到 Microsoft 对 IIS 专门的最佳支持，可参阅 microsoft.public.inetserver.iis 。在 microsoft.public.inetserver 区还有其它一些新闻组，不过刚才说的新闻组消息最多。你还可以在

microsoft.public.inetserver.iis.activeServerPages 找到 Microsoft 专用的 ASP 帮助。另一个更有趣的非 Microsoft 站点是 comp.lang.java，在该站点上有关于从 ASP 到 ActiveX 的各种事情的消息线索，这一点真让我感到惊奇。另一个很好的有关 IIS 专用帮助的非 Microsoft 站点是 comp.infosystems.www.servers.ms-windows。在写本书时作者就在该新闻组中找到一条非常棒的 ISAPI 信息。毫无疑问还有其它 comp.infosystems.www 新闻组可以查看。如果你使用 FrontPage 作为自己 Web 页维护工具之一，就应看看有关 ISAPI 专用帮助的 microsoft.public.frontpage.client。通常要想找出软件中的错误是非常困难的，但是如果你看看 comp.os.ms-windows.nt.software.compatibility，那么就会发现 IIS 并不困难。最后，如果你想检索一些隐藏的 ASP 新闻组，可以访问 microsoft.public.activex
.programming.scripting.vbscript。

13.1 ISAPI 概述

了解一下 ISAPI(Internet Server API)在 Microsoft 产品中处于什么位置，对于理解它本身不无帮助。实际上 Visual C++提供了五个层次的支持，其中三个层次位于服务器上。另外两个层次支持是客户机专用的——你永远不会在服务器上看到它们。下面将逐一定义各个层次支持并告诉你到哪里去找到它们。

- ISAPI（服务器） 这是本章要讨论的支持层。你需要给服务器本身提供一个扩展或过滤器。也就是说，客户机不会直接与该层次支持打交道。它只能看到交互的结果。
- WinInet（服务器或客户机） 本书不直接涉及该支持层次，但确实会间接地涉及到它。这些类允许你在客户机和服务器之间使用专用的数据传送方法。其中包含三种协议支持：HTTP、FTP 和 Gopher。从本质上看，你需要用这些类来创建一个会话(CInternetSession)，它是服务器的一个连接，然后指定连接类型(CFtpConnection 或者 CHttpConnection)。建立连接后，你就可以进行工作了，比如查找文件等（CFtpFileFind 或 GopherFileFind）。通常你不必直接与这些类打交道，因为 Visual C++已经做好了一切，就像第 12 章创建 URL moniker 应用程序时一样。
- 异步 URL moniker(服务器或客户机） 第 12 章已经介绍了 Visual C++ 的 Internet 支持的特定领域。要记住的是异步 URL moniker 允许你在 Internet 上完成任务而不必等待。只要简单地告诉目标应用程序你想要什么，然后就可以干其它事情了。这里的主要思想是，大部分情况下 Internet 不提供即时响应，如果不使用异步 URL moniker 方式，那么对于长时间的下载，会使用户的机器长时间地等待而无所事事。
- ActiveX document（客户机） 该层次的支持在浏览器中显示文档并允许用户编辑它。这些内容第 11 章都已经介绍过了。
- ActiveX 控件（客户机） 过去创建一个 Web 页的基本元素要涉及到编写大量的脚本，即使那样，得到的也只是静态的画面。在 Web 站点上

使用 ActiveX 控件意味着老的技术和静态显示再也不拖你的后腿了——现在的 Web 页可以根据具体情况发生变化。第 10 章介绍的就是该技术。

Web 链接 Internet 上有许多地方可以获得 IIS、ISAPI 以及其它与 Internet 服务器有关的技术（如 Active Server Page(ASP) ）。访问 IIS 信息的主要站点是 http://www.microsoft.com/ntserver/Basics/WebServices/default.asp 。Microsoft 支持的该站点包括了一些链接，它们指向你使用 IIS 本身所需的一切知识，甚至还有一些公用链接指向使能(Enabling)技术，（例如 ISAPI）。如果你想获得非 Microsoft 的帮助，可以访问位于 http://www.genusa.com/asp/的 ASP 开发者站点。它包含许多关于在 Internet 服务器上使用 Active Server Pages 的有价值的信息，另外它还有非 Microsoft 链接指向的 IIS 和 ISAPI 站点。

在现实世界中使用 ISAPI

在现实世界产品环境中，你需要了解一些与 ISAPI 有关的事情。例如，你的 ISA 是要装入服务器的一个 DLL,这与其它 DLL 完全一样。该 DLL 将与 HTTP 服务器共享同一地址空间，如果需要把该 DLL 占用的内存用于其它用途，还可以卸载这个 DLL。正是由于这一点，ISAPI 才赢得了你的青睐。与其它技术（如 CGI）比较起来，使用 ISA 有诸多好处，下面逐一予以介绍。

◆ 内存占用少 由于 ISA 是作为 DLL 装在服务器上的，并和 HTTP 服务

器共用同一内存空间，所以不必浪费那些使用 CGI 时用到的内存。你真正要装载的只是要求 ISA 完成任务所需的业务逻辑。

- **速度** 第一次装载 DLL 与第一次装载 C 应用程序所花费的时间基本相同，尽管 DLL 因为较小可能要快一些。不过，一旦第一次装载了 DLL 后，它就常驻内存直至卸载它，这就意味着可以不必再花费装载时间。而 CGI 脚本是每次调用时都装载一次 C 应用程序。这可不是好消息。因为 ISA DLL 与 HTTP 服务器共享同一内存空间，就不会为进程间调用浪费时间了，也不会遇到在不同地址空间中使用 C 应用程序时遇到的其它时间耗费问题。
- **代码共享** 服务器的任务就是一次性装入你的 DLL。申请 DLL 服务的任意应用程序都可以访问它。显然，代码共享是导致前两点中提到的对 CGI 进行改进来获得低内存消耗和提高速度的因素之一，而且，代码共享还带来不那么明显看到的益处。例如，代码共享减少了服务器的管理时间，因为管理员只需要对给定的 DLL 再产生一个副本去影响使用它的每一个应用程序。CGI 通常使用的 C 应用程序有大量的冗余代码，改变一个单一例程，就需要对服务器上使用该例程的每一个 C 应用程序进行改变。这就意味着增加管理员的时间和程序员追踪附加应用程序的需求。
- **可靠性** CGI 脚本使用的 C 应用程序在装入服务器并在服务器上执行时，并没有对服务器本身作多少访问。因此，就难以创建一个 C 应用程序来监督服务器事件并从出错中恢复。通常服务器总是终止出了错的

CGI，而客户机却什么也不知道。ISA 拥有对服务器的完全访问，这就是说它们可以更容易地从出错中恢复。于是，客户机二次申请服务器的情况（即使有的话）非常少见。

- **过滤能力** 使用 CGI 和 C 应用程序，你不能提供与 ISA 一样的事件驱动功能。原因很简单：C 应用程序是被调用、做工作、然后卸载。所以它没有办法长时间一直监视服务器事件。
- **在一个 DLL 中的多任务能力** 要 CGI 完成的每个任务都需要你有完成每个任务的单独的可执行程序。结果导致过度调用每个例程。而 ISA 却能包含多个命令，每个命令是 CHttpServer 类的一员。

获得这六项功能并不意味着在学习或写出代码方面付出昂贵代价。ISA 就像 CGI 一样好用。下面两行代码看上去就像是在应用程序中的代码一样。

```
<!-这是带有一个参数 CGI 例程的调用。->
http://aux/controls/sample.exe?Param

<!-这是带有一个参数 ISA 例程的调用。->
http://aux/contorls/sample.dll?Param
```

可以看出操作 ISAPI 一点也不难。我们使用了与调用 CGI 例程几乎完全一样的代码来调用 ISA 控件。实际上，从编码角度来看，唯一的不同就是 ISA 控件使用 DLL 扩展名，而 CGI 例程使用 EXE 扩展名。从理论上来说，你可以将服务器转换为 ISA，对 Web 页面作些检索替换性的修改，这从用户界面角度几乎看不出什么区别。当然，每个人都会注意到，因为使用 ISAPI 会带来 Web

站点的更高效率。

上面我提到的使用 ISA 的最后一个好处是，你可以用一个 ISA 完成多个任务。稍后将会介绍如何实现这种机制。现在你只需要知道，调用语法与以前调用标准 CGI 没什么太多不同。下面就是一个 ISA 例程调用，它指明了你要使用 DisplayStr 函数。

```
<!- 用一个参数在一个 ISA 例程中调用 DisplayStr 函数。->
http://aux/controls/sample.DLL?DisplayStr?Param
```

可以看出，在调用串中，我们使用第二个问号(?)调用了缺省任务之外的某种东西。第一个参数告诉服务器在 SAMPLE.DLL 中要调用哪个函数，第二个参数则包含一系列参数。这种调用函数的方法称为分析映射，本章“创建 ISAPI 扩展”一节将对如何创建它们作进一步介绍。

注 给调用语法再加第二个问号就可以在 ISA 中调用缺省函数之外的东西。

ISA 确实还有一些与 CGI 相同的特性。例如，应用程序是在服务器而不是客户机上运行的。这就使升级应用程序变得很容易——你要做的工作只是在服务器上更换一个 DLL 文件（一定要终止服务才能更新 DLL，不过这是方便地对机器进行更新所需付出的最小代价）。显然，这比在访问 Web 站点的每个客户机上都替换一个应用程序要容易多了。这也正是目前各家公司都开始重视把内部网作为承担像帮助系统和定制数据库应用程序这样的主要任务的原因之一：更新一个服务器比更新许多客户机要容易得多。

注 ISA 在服务器上运行，意味着每次更新一个定制应用程序时只需更新服务器而不用更新各个客户机。

技巧 IIS 第 4 版添加了一些新功能，使得 ISA 更容易使用。例如，可以让服务器在每次调用之后就卸载 ISA。这意味着你能试试 ISA，看它能否工作，如有必要，用新版本去替换一下，而这一切都是在不停止服务的条件下进行的。这一新功能的不足之处是，由于每当客户机调用 ISA 时，服务器都需要重载它，这样，就带来一些操作问题。

在过滤器与扩展之间进行抉择

你不必花太多时间就可以决定出是创建 ISA 过滤器还是创建 ISA 扩展（或者两个都创建）。这两种 ISA 之间的区别很明显。知道它们的差别后选择所需的 ISA 就是件简单的事了。

过滤器总是对服务器本身的事件作出反应。如果用户试图登录 Web 站点，就会生成 ISA 过滤器能监测到的一个事件。通常用过滤器来修改客户机与服务器之间的数据流。例如，如果应用程序发现有 SF_NOTIFY_AUTHENTICATION 事件发生，就可以显示一个对话框，要求用户输入名称和口令。如果名称和口令都正确，用户即可获得对 Web 站点的访问权。

扩展适用于 CGI 同样的情形。例如，如果想创建一个订货系统，就可以使用扩展。扩展将接收用户填入窗体中的数据，处理信息并将它们加入数据库，最后将某种形式的收据发还给用户。这正是使用 CGI 脚本时遵循的同一过程；唯一的区别是现在使用的是 DLL。与过滤器不同，扩展不监测服务器上的事件，无论从哪一方面看它都像应用程序那样工作。

还有一些情况，不管选择过滤器还是扩展都没什么关系。例如，考虑一个简单的 Web 计数器。每次有人访问 Web 站点时，都更新计数器并显示它们是第几位访问者。如果愿意的话，你可以从 Web 页面上调用 ISA 扩展来更新计数器，也可以用过滤器监测服务器上的 SF_NOTIFY_LOG 事件并自动发送信息。这只是个人喜好问题。

注 像访问计数器这类的 ISA 应用程序，我们既可以选择创建过滤器，也可选择创建扩展。

你可能还会遇到需要同时使用扩展和过滤器的情形。例如，你可能想为授权的 Netscape 用户显示一种 Web 页，而给非授权 Netscape 用户显示另一种 Web 页。针对授权的和非授权的 Internet Explorer 用户也有两个 Web 页。这种情况下需要在 HTML 格式中编写一些脚本来检测浏览器类型，需要用过滤器来检查用户是否授权，还需要用扩展来生成正确的页面。在服务器中将客户机脚本和 ISA 扩展进行组合，将大大丰富你的 Web 站点并使它便于每个人使用。

注 某些 ISA 应用程序（如专用的浏览器支持），要求创建既包括过滤器又包括扩展的复杂的 DLL。

使用 ISAPI 的五个类

前面已经说过本章将使用五个新的 MFC 类。这并不是说你不再使用那些以前已经熟悉的类了，不过这些类确实能提供设计 ISA 过滤器或扩展所需的特殊功能。下面将对下面章节中要用的这些类作简单的介绍。显然在开始设计真正的示例代码时你的了解会更深入一些的。

- CHttpServer 这是创建过滤器或扩展所需的主要类。CHttpServer 类定义了一个作为服务器扩展 DLL 或 ISA 的对象。该 CHttpServer 对象的主要目的是处理由服务器和客户机生成的消息。所以任何 ISA DLL 中只有一个 CHttpServer 对象。
- CHttpServerContext CHttpServer 真正创建了该对象，它是来自客户机的某个单一请求的单一实例。也就是说，对每个活动的客户机请求，都有一个 CHttpServerContext 对象。当客户机请求完成后该对象即被清除。
- CHttpFilter ISA 过滤器要求使用特殊的 CHttpFilter 对象来管理服务器的事件。该对象负责查找所有客户机与服务器交互时由服务器生成的 SF 消息。例如，每次用户试图访问 Web 站点时，客户机都会生成一个 SF_NOTIFY_LOG 消息。
- CHttpFilterContext CHttpFilter 对象将为每个活动的服务器事件创建一个 CHttpFilterContext 对象。CHttpFilterContext 类表示在特定会话中由单个客户机形成的单个事件。
- CHtmlStream 利用该类来管理客户机和服务器间的数据。无论何时当 CHttpServer 类需要在客户机和服务器之间传递信息时，都会创建该类的对象。大部分情况下每个 DLL 只有一个这样的对象。不过，CHttpServer 对象为了安全传递数据可以创建所需的任意多的 CHtmlStream 对象。典型的 CHtmlStream 对象包含创建 HTML 页内容所需的数据和所有标记。例如，如果你是检索数据库，那么结果的 CHtmlStream 对象就不仅包含数据，还包括实际创建的用来显示数据 Web 页所需的 HTML 标记（参

见第 8 章有关 HTML 标记的介绍）。

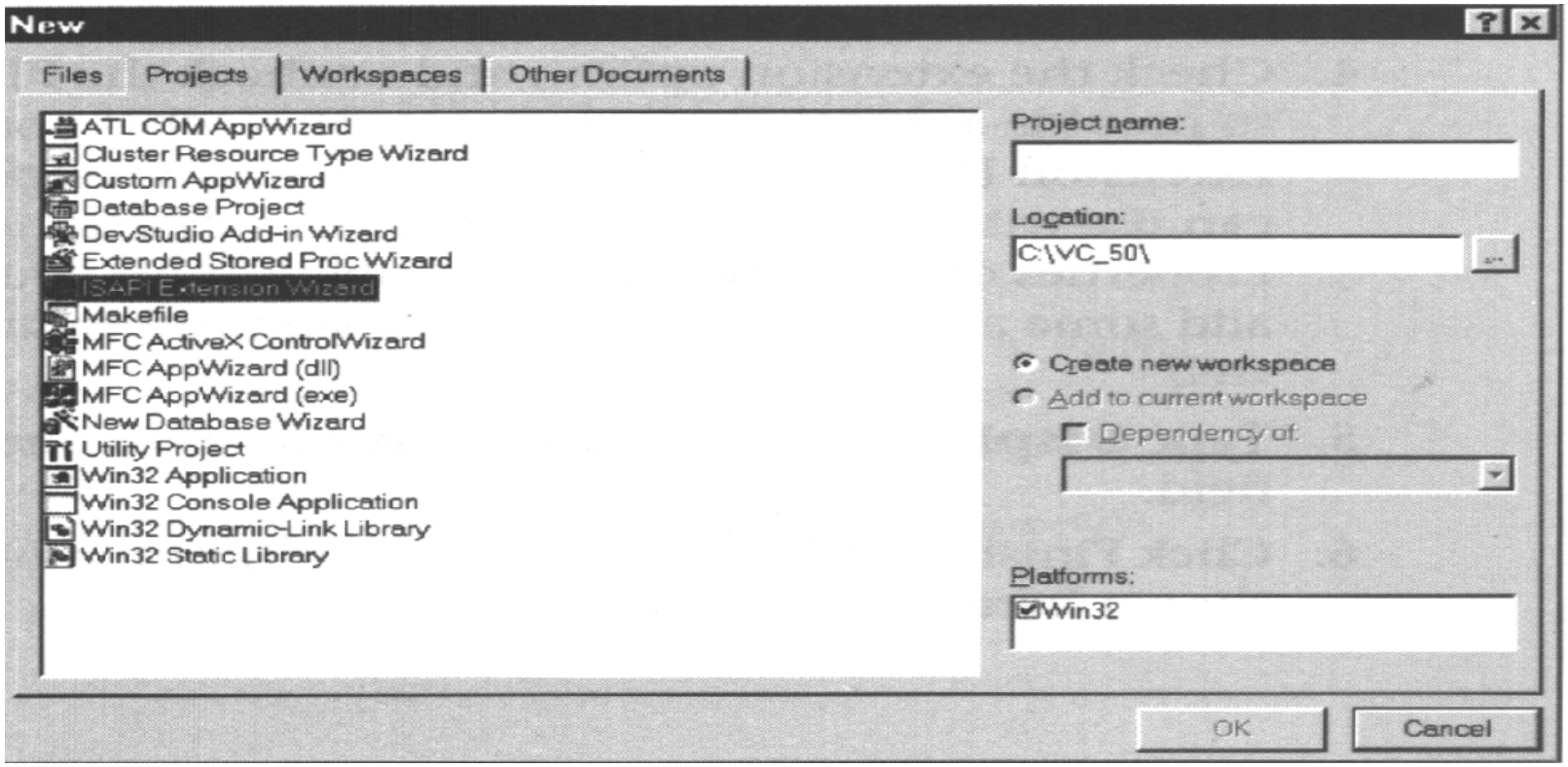
13.2 创建 ISAPI 扩展

好了，现在让我们看一下我们的第一个 ISA。这个例子中将利用 ISAPI Extension Wizard（ISAPI 扩展向导）来创建一个简单的 ISAPI 扩展。全部目的在于让你熟悉概念并介绍一些使用 ISAPI 的技术。例子程序接收来自客户机的一个串，稍作修改，然后用一个新页显示客户机发送来的这个串。下面介绍的过程将带着你一步步地创建示例 ISA。你可以使用同样的过程来做需要创建的任何 ISA 扩展。我们假定你已经启动 Visual C++ 并且使用的是 6.0 版。

WEB 链接 在创建自己的 ISAPI 扩展或过滤器时，最好的资源就是其它程序员编写的免费软件。alt.comp.freeware 新闻组中列出了相当多的这样的软件。例如，在写本书的时候，AAIT Incorporated 就推出了一个新的免费软件，称为 CGI Expert（CGI 专家）。它同时支持 CGI、Win-CGI、ISAPI 以及 NSAPI 接口。显然谁都想用免费产品，但是就像使用其它产品一样，一定要小心。不过它们确实对如何创建自己的定制扩展提供了极好的想法（或者可以提供一种解决方案使得你自己不必去编程了）。

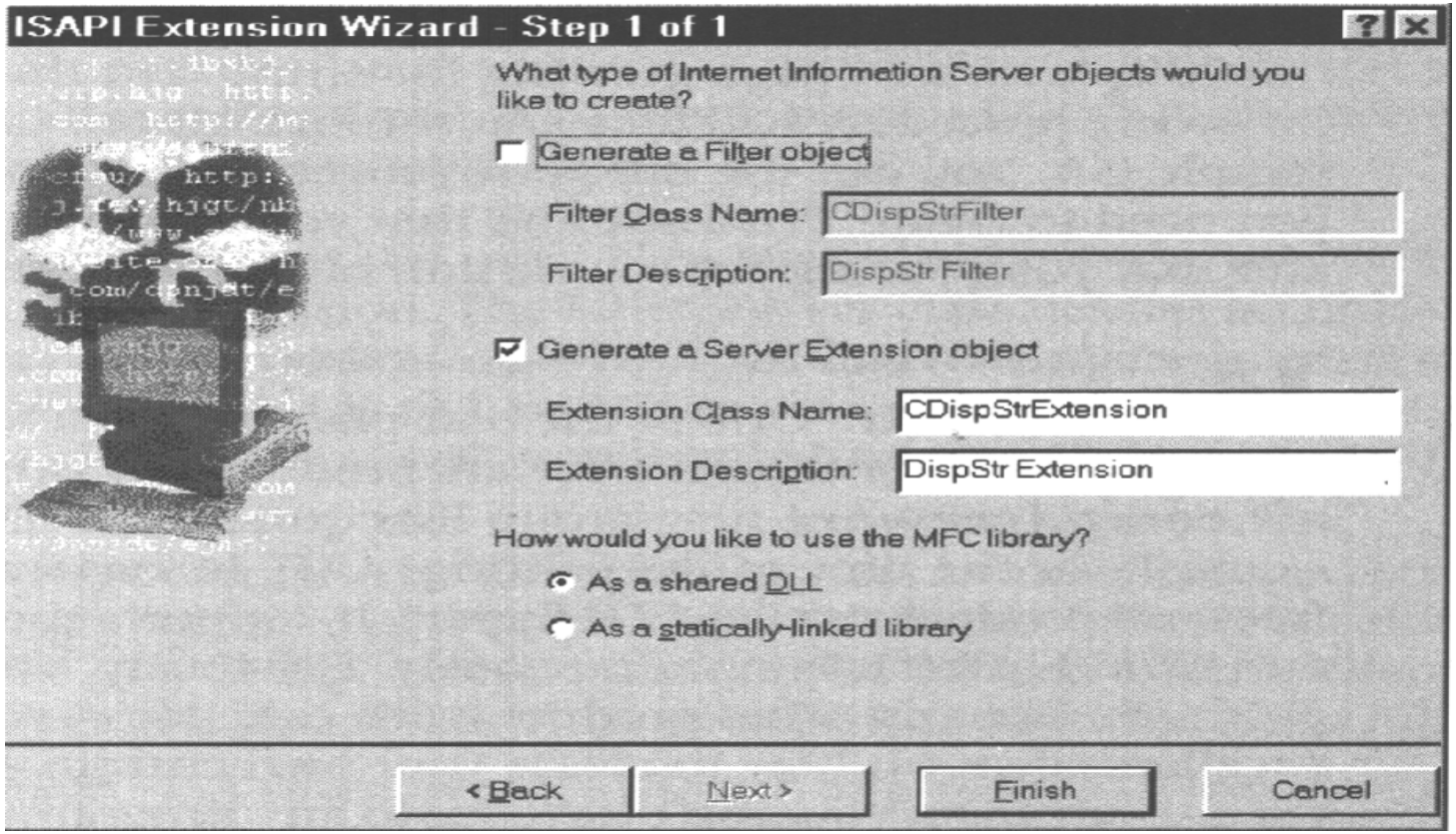
---

1. 使用 File（文件）|New（新建）命令显示如下图所示的 New（新建）对话框。请注意，我已经选择了 ISAPI Extension Wizard（ISAPI 扩展向导）。



2. 在工程名称域中键入想创建的 ISA 的名字。这个例子中使用的是 DispStr，不过可以使用你想使用的任何名称。

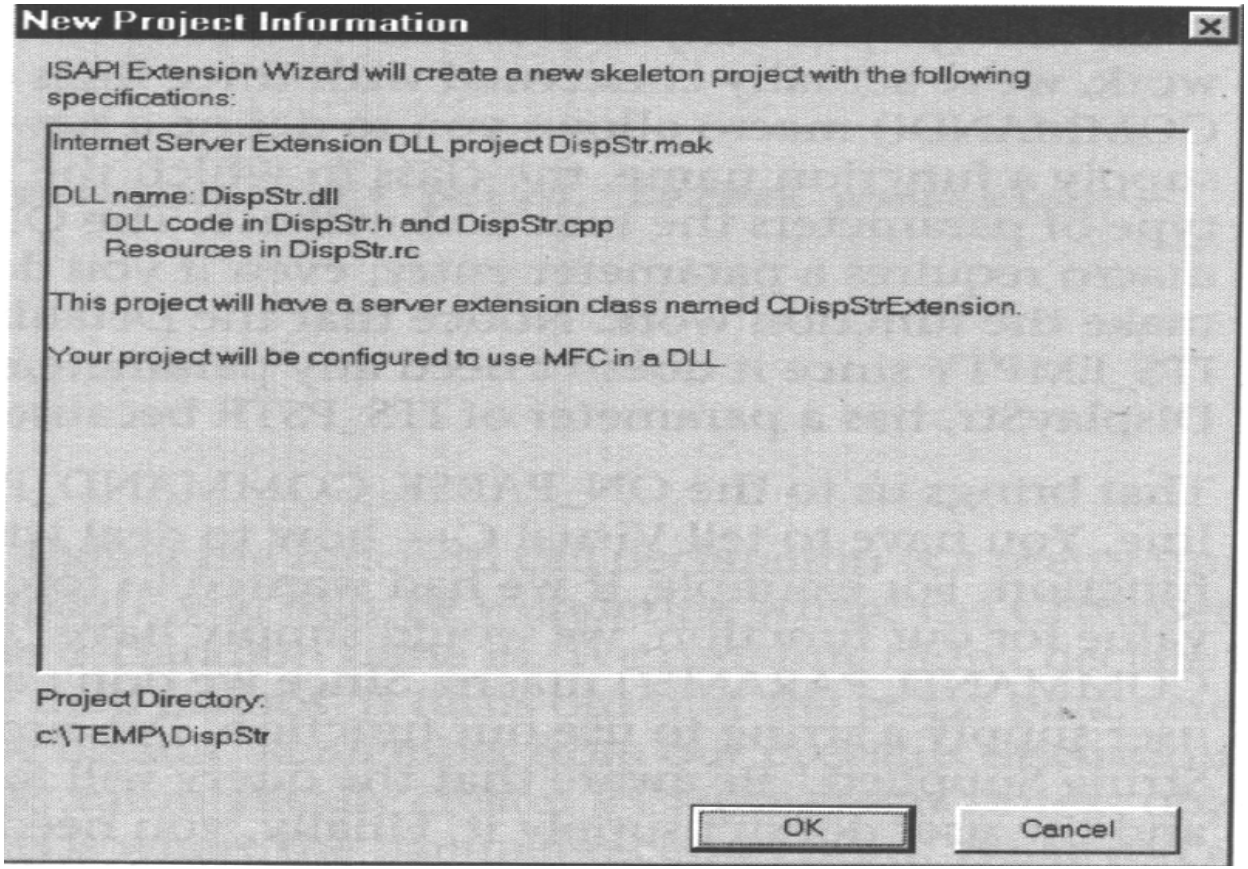
3. 单击 OK，就会看到如下图所示的 ISAPI Extension Wizard（ISAPI 扩展向导）Step 1 of 1（第一步）对话框。在这里可以为你的 ISA 选择各种特性。注意对话框中有三个主要区域。选中第一个复选框，就可以创建过滤器；选中第二个复选框，就可以创建扩展。第三个区域定义如何将 MFC 与你的应用程序链接起来。



4. 选中扩展选项，而不是过滤器选项。在 Extension Description （扩展描述）域中，你应对你的 ISA 提供简短扼要的说明。该说明将以一个字符串的形式出现，你可在 DLL 中需要时使用它。当某人打开你的 ISA 的 Properties(属性)对话框时，该说明并不出现在对话框中，所以需要给 DLL 的版本信息再添加点附加文本。

5. 在 Extension Description（扩展描述）域输入 Display a string from the client（显示来自客户机的串）。

6. 单击 Finish（完成）就会看到如下图所示的 New Project Information （新建工程信息）对话框。



7. 单击 OK，ISAPI Extension Wizard（ISAPI 扩展向导）就完成了所需的程序框架的创建。

现在，为了使扩展具有实用功能，还要对 ISAPI Extension Wizard（ISAPI 扩展向导）创建的框架做三件事。首先是为添加的新函数创建一个分析映射 (Parse

map)。在 DispStr.cpp 文件的顶部可以找到该分析映射。程序列表 13.1 列出了应该给该分析映射加入的代码，以便我们能够从 HTML 页面访问这个新增函数。

程序列表 13.1

```
//////////////////////////////////////////////////////////////////////
// command-parsing map

BEGIN_PARSE_MAP(CDispStrExtension,CHttpServer)
    //Our special DisplayStr parse command.
    ON_PARSE_COMMAND(DisplayStr,CDispStrExtension,ITS_PSTR)
    ON_PARSE_COMMAND_PARAMS("string='No String Supplied'")

    //Default parse command
    ON_PARSE_COMMAND(Default,CDispStrExtension,ITS_EMPTY)
    DEFAULT_PARSE_COMMAND(Default,CDispStrExtension)
END_PARSE_MAP(CDispStrExtension)
```

尽管只需要增加两行代码就能使这个例子工作，但我们真正关心的却是下面三行代码。ON_PARSE_COMMAND()宏允许定义一个新函数。注意，应提供函数名、所提供函数所在的类以及该函数使用的参数的类型。ON_PARSE_COMMAND()宏需要一个参数项，尽管你实际上可能并不需要任何参数就能使该函数正常运行。注意，因为缺省函数不需要任何参数，所以使用了 ITS_EMPTY 值。由于我们的新函数，DispStr——需要一个字符串指针，所

以它带有一个参数 ITS_PSTR。

然后下一行是 ON_PARSE_COMMAND_PARAMS()宏。你必须告诉 Visual C++如何处理你的函数的参数。例如，如果我们想强制用户为函数提供一个串值，就应在 ON_PARSE_COMMAND_PARAMS()宏中，简单的放上 String。因为实际上并不一定要求用户提供一个字符串才能使用函数工作，所以这里提供了一个缺省值 。注意如果你需要一个参数而用户没有提供的话，查询就会失败。最后还要用 DEFAULT_PARSE_COMMAND() 宏告诉 Visual C++用哪个函数作为缺省函数。由于这个例子的 Default 函数挺合适，所以就没有更改这个缺省的设置。

其次，你需要给 DispStr.H 文件增加一个函数项。除非通过修改类说明来包含你的新函数，否则 Visual C++对此将会一无所知，于是 DLL 也不会被编译。幸运的是，我们只要增加程序列表 13.2 中黑体显示的那一行代码就可以了。程序列表 13.2 列出了该行代码周围的代码行，以便于你知道新的一行应添加在什么地方。

程序列表 13.2

```
{
public:
        CDispStrExtension();
        ~CDispStrExtension();

//Overrides
        //ClassWizard generated virtual function overrides
```

```
        //NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //{{AFX_VIRTUAL(CDispStrExtension)
    public:
    virtual BOOL GetExtensionVersion(HSE_VERSION_INFO* pVer);
    //}}AFX_VIRTUAL
    //TODO: Add handlers for your commands here.
    // For example:

    void Default(CHttpServerContext* pCtxt);
    void DisplayStr(CHttpServerContext* ptext,LPTSTR pszString);

    DECLARE_PARSE_MAP()
    //{{AFX_MSG(CDispStrExtension)
    //}}AFX_MSG
}
```

可以看出，增加函数调用声明很简单。不过现在你可能还不清楚声明的CHttpServerContext* pCtxt部分是从哪里来的。我们确实没有在前面的任何分析映射宏中对它进行说明。实际上pCtxt参数是缺省地传递给函数的。记住前面已经介绍过CHttpServer类可以为每个用户请求创建一个CHttpServerContext对象，这就是该参数的来源。你得到的是指向与用户对函数的调用相联的CHttpServerContext对象的指针。这也是对函数的多个调用进行区分的办法——

每个调用都有一个完全不同的对象与之相关联。

制作这个 DLL 函数还剩最后一件事——给 DispStr.CPP 文件添加函数代码。这个例子是将函数代码放在已有的 Default()函数代码之后。程序列表 13.3 显示的是这个例子使用的非常短的函数。这里没有什么花里胡哨的东西，其全部目的在于说明如何完成这件任务。

程序列表 13.3

```
void CDispStrExtension::DisplayStr(CHttpServerContext *pCtxt,LPTSTR pszString)
{
        // Start sending information to the new Web including a defaullt title.
        StartContent(pCtxt);
        WriteTitle(pCtxt);

        //Display the body of the new page.
        * pCtxt << _T("<H3><CENTER>ISAPI Server Extension
Example</CENTER></H3><P><P>");
        * pCtxt << _T("This was the string you entered:<P><EM>);
        * pCtxt << _T(pszString);
        * pCtxt << _T(pszString);
        * pCtxt << _T("</EM>");

        //End the display area.
```

```
    EndContent(pCtxt);
}
```

这个函数本身很容易理解。我们做的第一件事是告诉 Visual C++启动一个 Web 页，这就像给文档加入<HTML>和<HEAD> 标记一样。我们做的第二件事是输出一个标题，就像给文档加入<TITLE>标记一样。重载缺省标题的唯一途径是重载 WriteTitle()函数，这是一件想干就能干成的事。现在有了标题，就该创建内容了。需要使用流操作符来发送信息。注意这个例子很轻松自如地使用了第 8 章介绍的所有标记。对标准 HTML 文档能做的事也同样能够对 ISAPI 扩展做。待会儿将会说明这些标记是如何一起工作从而产生了 Web 页的。另外还应注意我们将从 Web 页得到的字符串发送回新 Web 页，甚至不必将值转换为文本，就可以使它们工作了。这个例子中使用的最后一个函数调用是 EndContent()，它告诉 Visual C++已经完成发送信息的任务了，这就象在文档末尾加上</HTML>一样。

现在你就可以编译我们创建的 ISAPI 扩展了。一旦编译成功，就可以将 DLL 移到 Web 服务器上。有好几个逻辑位置可以存放该 DLL，不过最常用的两个位置是 Scripts 目录或一个特殊的 Controls 目录。我通常将自己的所有控件都放在 Controls 目录中以便于查找，不过实际的存放位置并不重要。唯一的原则是用户应该能够通过 Web 站点来访问包含该 DLL 的目录，而且你应该用 Web 服务器提供的 Internet 服务管理器将该目录标记为可执行的。

13.3 在 Web 页中使用 ISAPI 扩展

现在 Web 服务器上已经有了一个新的 ISAPI 扩展，可以进行测试了。记住该 ISA 有两个函数：由 ISAPI Extension Wizard（ISAPI 扩展向导）提供的缺省函数(Default)和我们加入的函数(DisplayStr)。另外，DisplayStr()函数还将依据用户是否为函数的工作提供数据来决定采取两种不同的工作方式中的哪一种方式。程序列表 13.4 显示了创建这个例子的 ISA 的测试 Web 页所需的 HTML 代码。存放该代码的文件采用.HTML 作为扩展名。

程序列表 13.4

```
<HTML>
<HEAD>
<TITLE>ISAPI Extension Example</TITLE>
</HEAD>
<!- 创建显示我们的按钮的窗体 ->
<FORM NAME="MyForm">
<!- 显示一个标题 ->
<H3><CENTER>Display a String ISAPI Extension</CENTER></H3>

<!- 显示一个编辑框 ->
Type a string value:
```

```
<INPUT TYPE=TEXTBOX NAME="StringValue" SIZE=40><P>

<!- 显示一些按钮 ->
<CENTER>

<!- 执行缺省行动 ->
<INPUT LANGUAGE="JavaScript"
       TYPE=BUTTON
       VALUE="Default"
       ONCLICK="window.location.href = 'controls/dispStr.dll?'">
<!- 用一个串执行缺省行动 ->
<INPUT LANGUAGE="JavaScript"
       TYPE=BUTTON
       VALUE="Display String"
       ONCLICK="window.location.href = 'controls/dispStr.dll?DisplayStr?'
                +MyForm.StringValue.value">
</CENTER>
</FORM>
</HTML>
```

可移植性 尽管 Internet Explorer 能很好地显示在 HTLM 文档主体中<BODY>和</BODY>标记之间的<INPUT>标记，但 Netscape Navigator 甚至都不认识它们。如果想做一个适用于所有浏览器的格式，就必须将<INPUT>标记放在<FORM>和</FORM>标记对之间的窗体中。

样本代码并不复杂。它包括一个标题、一个编辑框和两个按钮，其中编辑框将包含用户想显示的串的值。注意，我们的格式中并没有包括提交动作，而是利用标准的按钮和 Java Script 来启动该动作。在使用像这个例子一样包含多个函数的 ISA 时，这样安排比提交似乎要更好一些，我们就是这样做的。

另外，对每个按钮的 ONCLICK 事件所提供的 JavaScript 还要注意以下两点。对 Default（缺省）按钮只是调用 DispStr.DLL 本身，而没有指定一个特殊的函数。另一方面，Display String（显示字符串）按钮却同时包含一个函数名和一个串参数作为 DispStr.DLL 调用的一部分。待会儿将简单地交待一下它是如何工作的。

技巧 HTML 并不把编辑框中空格之后的文本发送给服务器。例如，如果在编辑框中输入 Hello World，服务器只会接收到 Hello。必须在单词间放上加号(+)才能将它们发给服务器。例如，如果给服务器发送 Hello+World，服务器看到的就是 Hello World（Web 服务器日志仍会显示+符号，它表示是在到达服务器后加号才被空格替换的）。这意味着在向服务器发送串之前，用在例子程序中添加的一些不被显示的文本操作符代码（如加号）去替换编辑框中的空格。

打开测试 Web 页，看到的是如图 13.1 显示的页面。如前所述，这是一个测试页，所以为简单起见，并没有添加太多的格式方面的代码。

注释 一定要注意，把服务器上用来保存 DispStr.DLL 的目录，设置成 Web 服务器在因特网服务管理器内可执行的特权，否则，如果浏览器试图执行该文件时就会出现许多错误——而大部分都不告诉你是由于它不能执行该文件（最常见的错误是说该服务未被支持）而引起的。将目录置为在操作系统层次可执行的特权还不足以允许 Web 站点的访问者使用你创建的 ISA。大部分情况下，将 ISA DLL 放在 Web 服务器的脚本目录（假定有一个）下，就能避免出现以上问题。

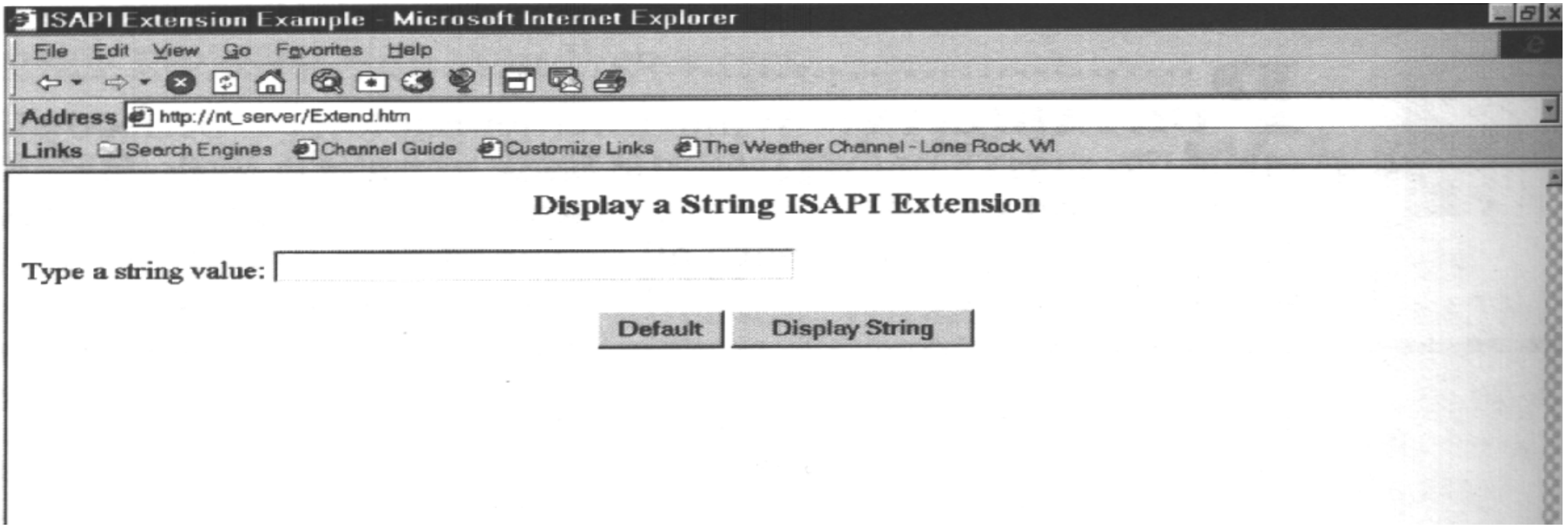
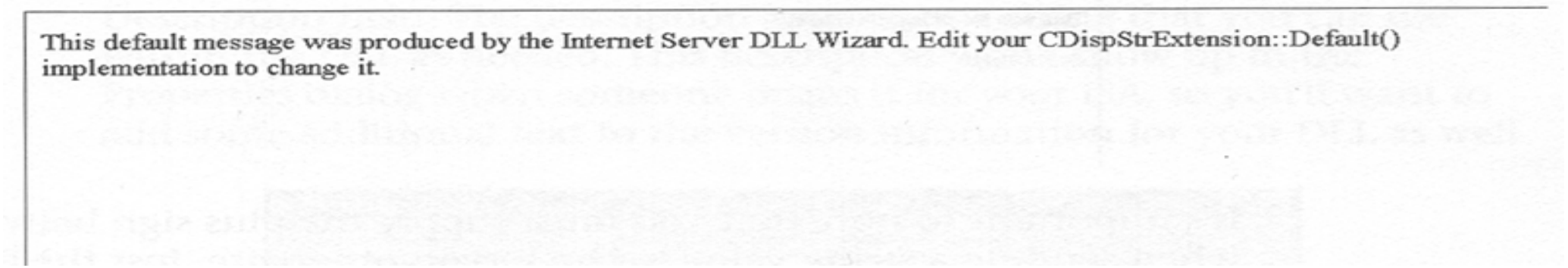
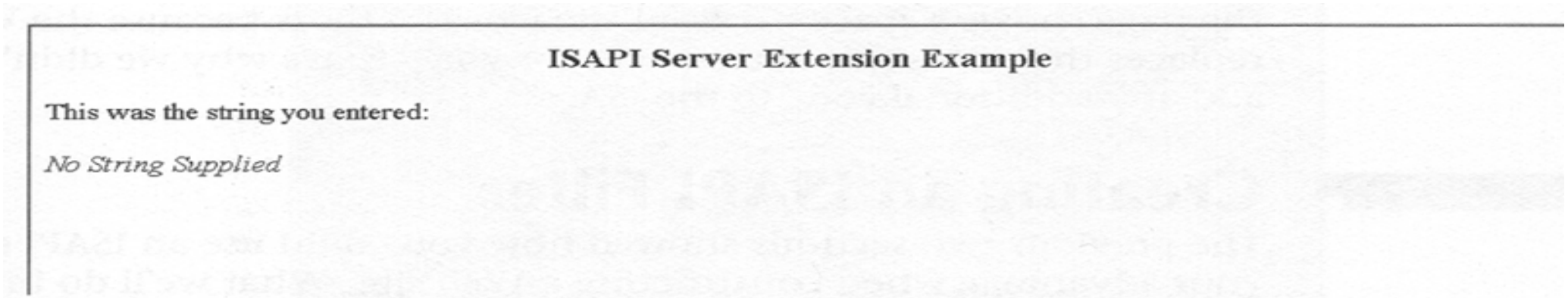


图 13.1　这个简单的测试页面允许我们检查 DispStr.DLL 的所有功能

现在开始测试 DispStr.DLL。首先单击 Default 按钮，就会看到如下图所示的由 ISAPI Extension Wizard（ISAPI扩展向导）提供的缺省字符串（实际上，它通知你必须给缺省函数增加一些代码）。



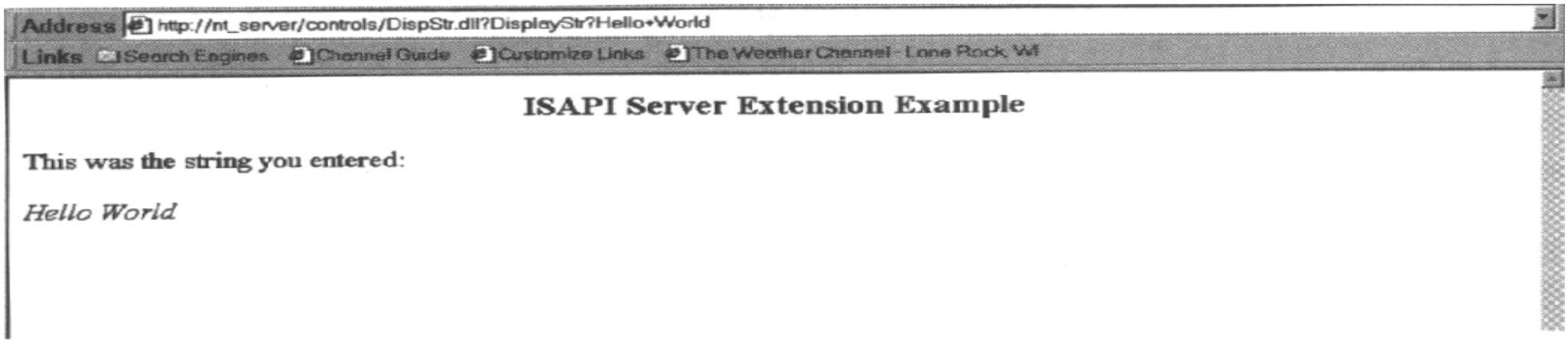
单击 Back 按钮（或浏览器上与之功能相同的部件），就回到 Extend.HTM 页。现在单击 Display String 按钮，就能看到如下图所示的结果。



这个例子用 DisplayStr 函数调用 DispStr.DLL，但是因为编辑框中没有任何值，所以函数必须显示前面作为 ON_PARSE_COMMAND_PARAMS()宏的一部分而定义的缺省字符串。现在你应该明白大多数情况下提供一个缺省字符串的重要性了。如果用户忘记了提供一个需要的值，也不必担心会使 ISA 产生问题。相反它们得到的将是一个可以帮助解决 Web 页问题的返回值。这样用户就会确切知道哪里错了，从而只需花几分钟就能解决问题，而不是像通常那样花几个

小时去进行调试。

再次单击 Back 按钮回到 Extend.HTM 显示页面，这次在 Type a String Value（键入一个串值）域中输入 Hello+World。单击 Display String（显示串）按钮就能看到如下图所示的结果。这里我们使用 DispStr.DLL 的 DisplayStr 函数并提供 Hello+World 作为字符串值。



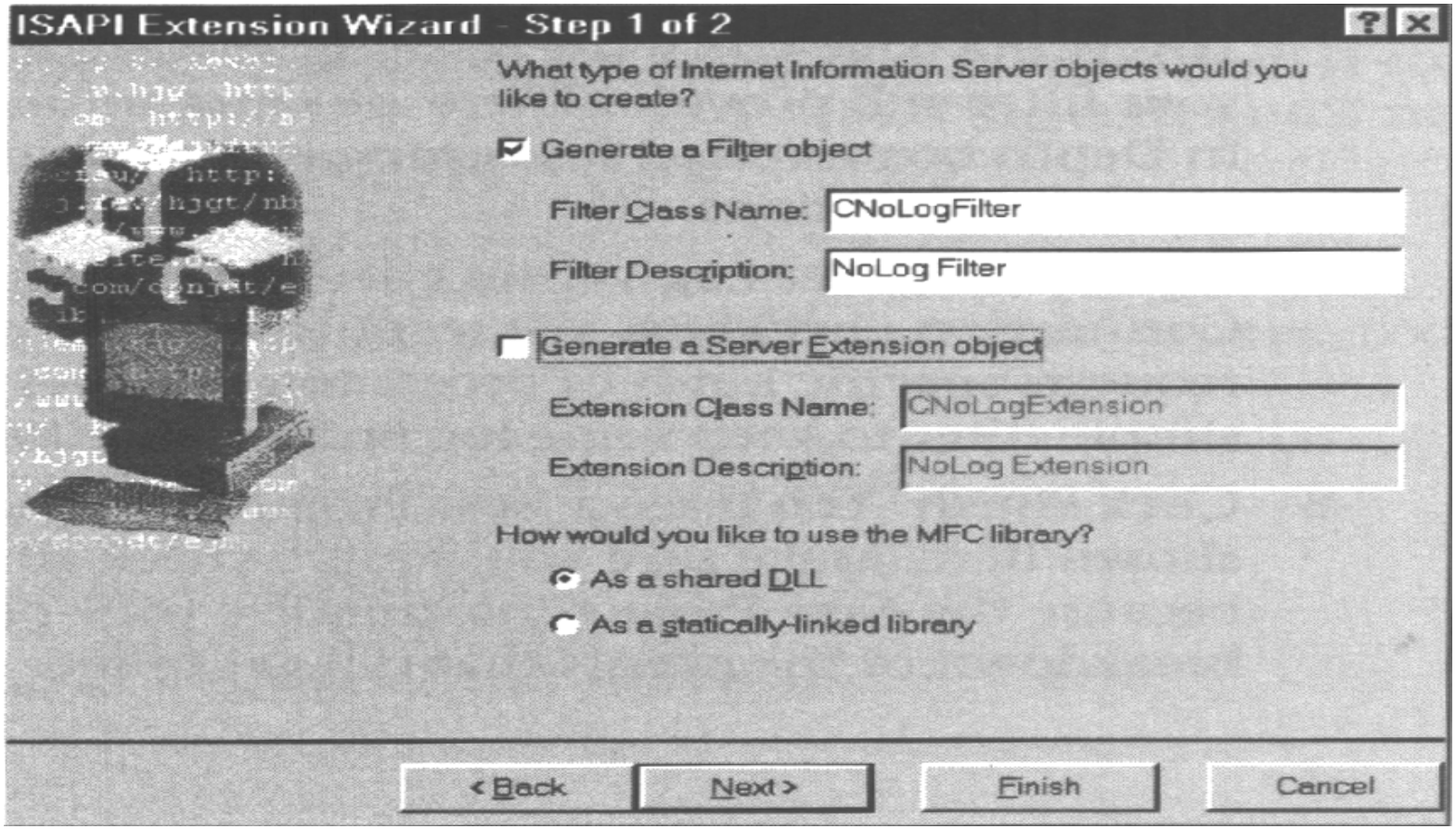
在向服务器发送字符串值时，你必须在单词间加上加号，否则将只显示第一个单词（你可以亲自做个实验。尝试不同的组合时，一定要检查 Web 服务器的日志）。如果看一下浏览器上的 Location 域，就会注意单词间有加号，但页内容中单词间却是空格。这是因为 Web 服务器用空格替换了加号，这也正是不必给 ISA 再添什么代码的原因。

13.4 创建 ISAPI 过滤器

前面两节介绍了如何利用 ISAPI 扩展更好地构造 Web 站点。本节将介绍构建自己的 ISAPI 过滤器的过程。创建 ISAPI 过滤器过程与创建扩展的过程的起始部分是一样的。下面的步骤将帮助你创建一个程序框架。

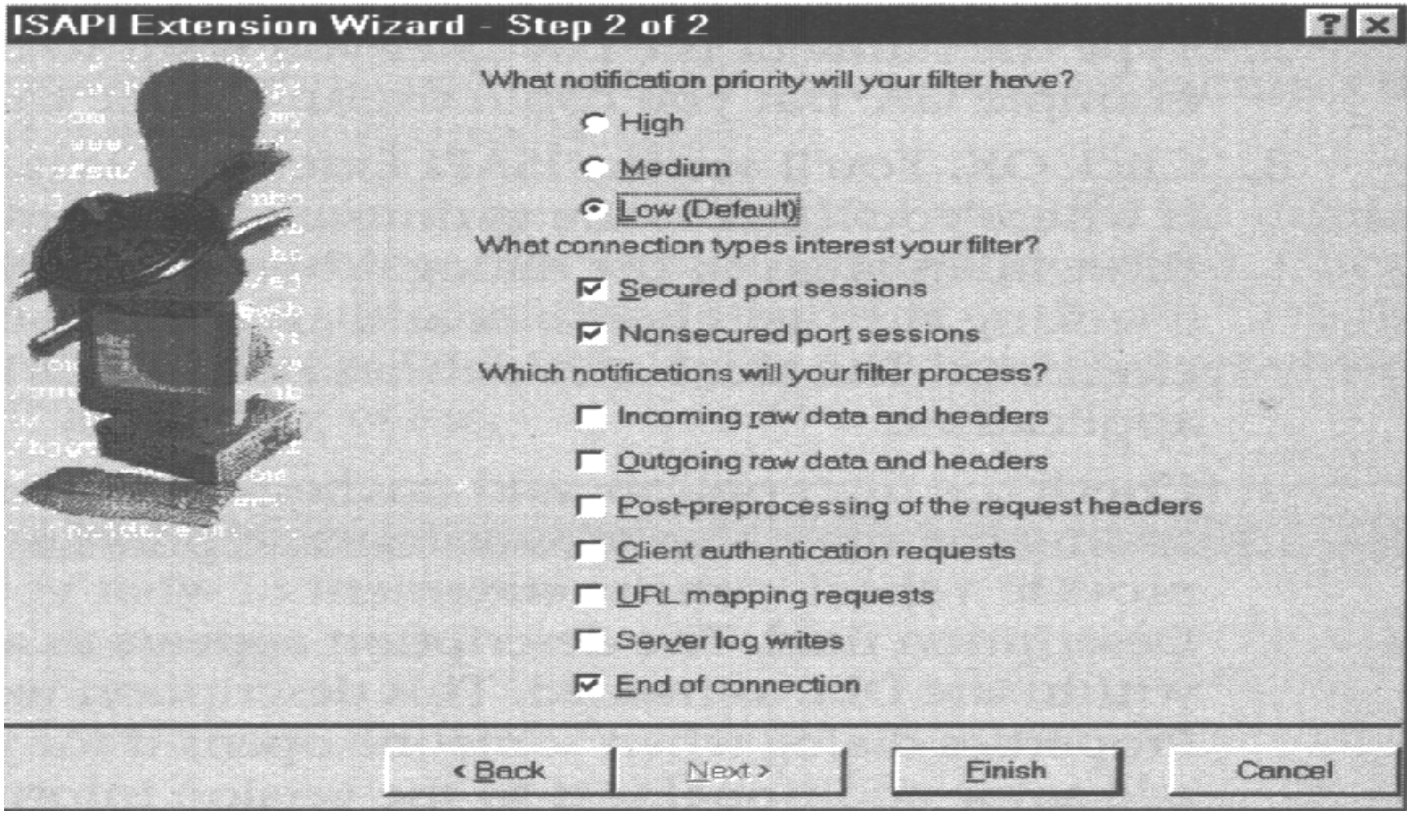
注 记住 ISAPI 过滤器与扩展不同，过滤器在特定事件发生时自动做出反应，而扩展更像是一个应用程序那样被调用。

1. 使用 File（文件）|New（新建）命令显示 New 对话框。选中 ISAPI Extension Wizard（ISAPI 扩展向导）（即使要创建的是过滤器，也要用 ISAPI Extension Wizard 来创建）。
2. 输入想创建的过滤器的名称。对这个 ISA 示例我们使用了 NoLog，但你可以任意选用其它名称。
3. 单击 OK，即出现 ISAPI Extension Wizard（ISAPI 扩展向导)——Step 1 of 2 对话框。在这里为你的 ISA 选择各种特性。注意对话框有三个主要区域。选中第一个复选框表示创建过滤器，选中第二个复选框表示创建扩展。第三部分定义如何将 MFC 链接到你的应用程序中。
4. 选中 Filter（过滤器）选项而不选中 Extension （扩展）选项（注意加入过滤器也就增加了一些其它步骤）。在 Filter Description（过滤器描述）域中，提供一个描述你的 ISA 功能的短小扼要的说明。该描述将作为字符串来显示，需要时可以在 DLL 中使用它。当别人为了使用 ISA 打开它时，该描述不会出现在 Properties（属性）对话框中，所以还要给 DLL 的版本信息中加入一些文本。



5. 在 Filter Description(过滤器描述)域中输入 Classify some log entries for security reasons(为了安全原因而对日志项进行密级划分)。

6. 单击 Next，就能看到如下图所示的 ISAPI Extension Wizard(ISAPI 扩展向导)——Step 2 of 2 对话框。

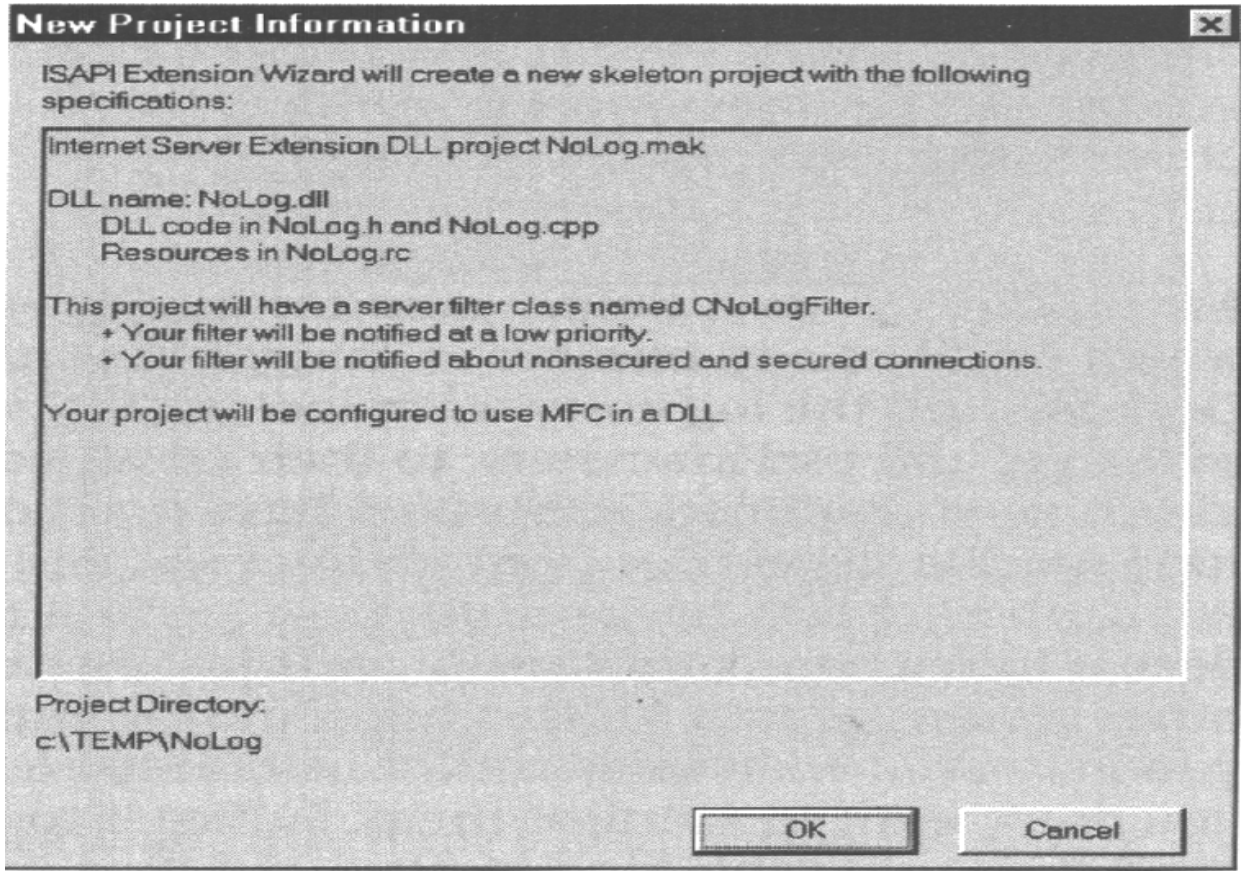


这是选择过滤器能够提供的事件以及监测类型的属性页。有三个方面的情况需要考虑。这些选项的详细介绍请参见“高级技巧”中的“选择过滤器选项”部分。

7. 选中 Server log Writes（服务器日志写入）复选框，而不选 End of Connection（连接结束）复选框。由于我们创建了一个简单的过滤器来保持某些日志项的密级划分，所以每当用户申请服务器的特殊类型服务时就激活过滤器。

8. 单击 Finish，就会看到如下图所示的 New Project Information （新建工程信

息）对话框。一定要二次核对过滤器的这些设置项，因为 New Project Information 页提供了对过滤器将要看到的事件进行的详细分类。



9. 单击 OK，ISAPI Extension Wizard（ISAPI 扩展向导)就会创建所需的程序框架。

这个示例的代码部分简单明了。过滤程序很容易弄得复杂化，从而导致调试很困难。大多数情况下，你应该让过滤器程序尽可能简短。在对出错的过滤

器查错时，如果程序是模块化的，将会提供极大的帮助。程序列表 13.5 列出了为使 OnLog() 函数工作而需要添加的代码。

程序列表 13.5

```
DWORD CNoLogFilter::OnLog(CHttpFilterContext *pCtxt,PHTTP_FILTER_LOG pLog)
{
        //Check for a specific notification.
        if (strstr(pLog->pszTarget,"NoLog.htm") !=NULL)
        {
                //Change the target information to Classified.
                pLog->pszTarget = "Classified Target";
                pLog->pszParameters = "Classified Parms";
        }

        //Pass control to the next filter.
        return SF_STATUS_REQ_NEXT_NOTIFICATION;
}
```

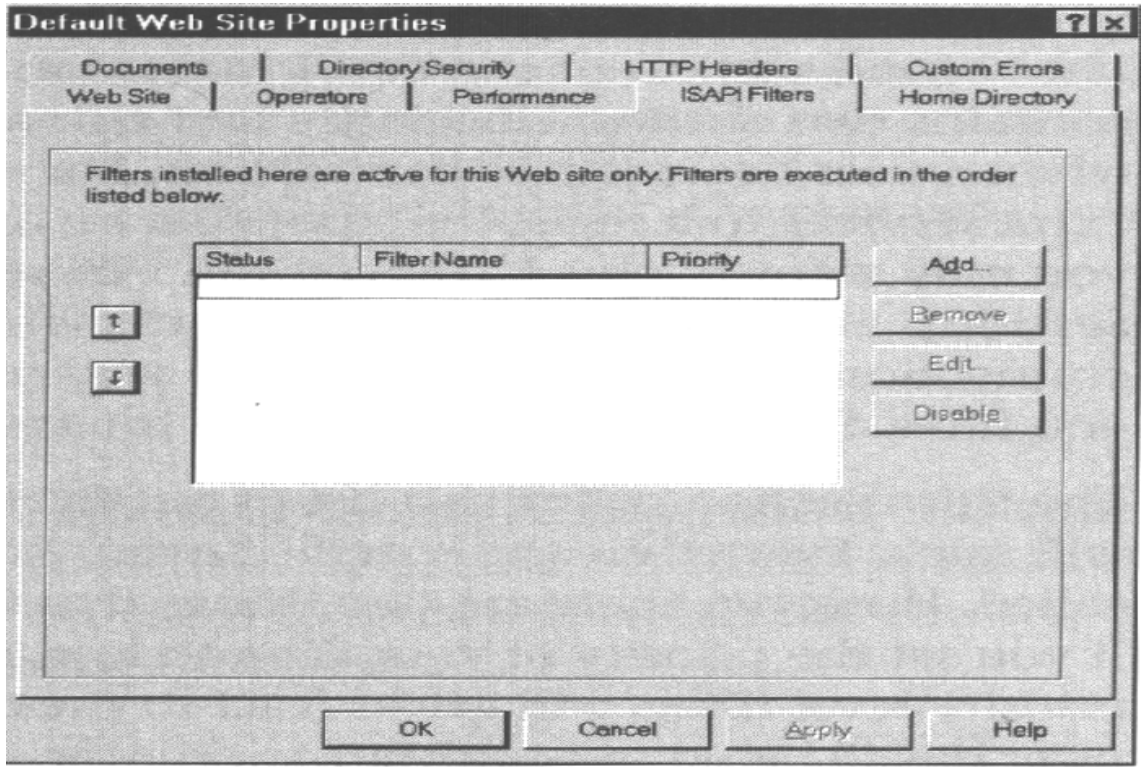
可以看出，为了某个特定目标，我们对日志项进行简单的监测，这个特定目标可以是除另一个 Web 页之外的任意数量的事情。一旦找到所需的日志项，我们将按照分类设置修改两个成员：pszTarget 和 pszParameters。显然这种思路已大大简化，不过都有实际的用途。你可能不想让某个特定日志项被别人偷看，

这里介绍的方法就可以实现你的目的。记住黑客们经常利用它们在服务器上找到的目志文件，来盗窃公司的内部资源。不仅如此，对某些日志项进行密级划分还有其它原因。在某些情况下，你可能想完全删除某些日志项只是因为不想再监测它们。好了，我们已经详细说明了如何小心使用这种过滤器，不过这将是一个好的开端。

现在可以编译新的ISAPI过滤器了，但是，在你将它移到服务器上的Controls或Scripts目录之后，仍要做一件事。与扩展不同，ISA过滤器在你启动服务器时即被装载。也就是说，如果你使用的是老版本的IIS，那么，就停止目标服务器，生成一个注册表项，然后再次启动该服务器。过滤器将作为启动进程的一部分被装载。

WWW服务在如下的注册值中保存它的过滤器项：HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\W3SVC\Parameters\Filter DLLs。你可能已经在该位置找到一个或多个值。这里要做的是加上一个逗号，然后输入自己新的ISA过滤器的位置。一定要在学习下节之前做这一步，否则过滤器就不会被装载。实际上，如果过滤器工作出了问题，这里是你首先应该检查的地方。

对IIS 4.0以上版本的用户来说，不必为装载ISAPI过滤器而手工编辑注册表了。此时所要完成的工作就是打开Microsoft Management Console（管理控制台），用右键单击需添加过滤器的Web站点，从关联菜单中选择Properties（属性），再选中ISAPZ Filter（过滤器）项，就会看到如下所示的对话框。



单击 Add（添加）按钮，就会看到 Filter Properties（过滤器属性）对话框，在 Filter Name（过滤器名）域中键入名称，使用 Browse（浏览）按钮找到 Nolog.DLL 文件（或想添加到 Web 站点的随便什么别的 ISAPI 过滤器）。选定文件后，其名称将出现于 Executable（可执行）域。单击 OK 关闭 Filter Properties（过滤器属性）对话框；再次单击 OK 关闭 Default Web Site Properties（缺省 Web 站点属性）对话框。IIS 将为你自动加载该过滤器。

高级技巧

选择过滤器选项

ISAPI Extension Wizard（ISAPI 扩展向导）——Step 2 of 2 页对于过滤器设计者是非常重要的，这是因为它包含了需要为过滤器设置的监测选项。这一页上的选项分三部分。第一部分确定过滤器优先级，缺省设置为 Low（低），这在大多数情况下运行良好，一般后台任务使用这一层次的优先级。安全过滤器需要设置为 medium（中）优先级，毕竟你不想让过滤器在事件之后再作出反应，在事件发生期间过滤器就作出反应就好得多了。最后，High（高）优先级设置应为关键层次的过滤器保留着，例如，向每个人发出一条消息，声明由于电源故障，服务器即将关机，这时就应使用 High 优先级了。

ISAPI Extension Wizard（ISAPI 扩展向导）- Step 2 of 2 对话框的第二部分包含着两个选项。如果用户拥有安全的连接就应复选第一项，例如，如果计划对注册用户提供额外服务，而不让匿名用户得到这些额外服务，那么就应复选第一项了。第二项是不安全端口会话，允许你在用户不拥有对服务器的安全连接时监测事件。如果你有一个 Internet 一般 Web 站点，大多数用户都是这样去访问你的站点的。

第三部分是过滤器追踪的事件列表，每当这里指定的事件发生，就调用过滤器，但是，过滤器被调用时有两件事要影响它。如果过滤器的优先级为 Low（低），那么 High（高）或 medium（中）优先级的过滤器将先对事件作出反应。另外，你的过滤器不得不设置成用户当前安全级来监测事件，换言之，如果你把过滤器设置成监测非安全级行动而用户处于安全模式，则过滤器就不会被调用。

13.5 使用 ISAPI 过滤器划分 Web 站点的密级

现在 ISA 过滤器已经完成并且编译好了，应将它放在自己通常存放过滤器的位置，可以是 Controls、Scripts、甚至 System32 目录下。记住要在服务器停止运行时生成所需的注册表项，完成后再重新启动服务器。现在我们开始测试过滤器。

我们首先需要完成的工作是创建三个 Web 页。第一个是中心 Web 页，我们使用它在另两个页（一个正常注册，另一个按密级注册）之间切换。第一个 Web 页称为 Filter.HTM。程序列表 13.6 显示了该测试部分的源代码。

程序列表 13.6

```
<HTML>
<HEAD>
<TITLE>ISAPI Filter Test</TITLE>
</HEAD>
<BODY>
```

```
<!- 显示标题 ->
<H3><CENTER>ISAPI(NoLog)Filter Tester</CENTER></H3>
<!- 显示链接和某些解释性文本 ->
Choosing the first link will create a classified log entry,
while the second one creates a standard log entry.<P>

Log this Link as <A HREF="NoLog.htm">Classified</A><P>
Log this Link <A HREF="LogIt.htm">Normally</A>

</BODY>
</HTML>
```

这些代码并没有什么神秘可言，我们前面都曾使用过。该页包含一个标题，一些解释性文本以及几个链接。我们将使用它在两个测试页之间切换而无需太多额外的工作。

两个测试页的名字很重要，第一个测试页必须命名为 NoLog.HTM，否则过滤器将不会工作（因为过滤器通过该名称来寻找目标，因此你必须提供一个名称）。程序列表 13.7 包含该页的代码。第二个测试页名为 LogIt.HTM。程序列表 13.8 包含了创建它所需的源代码。两个测试页都包含一个标题和一些文本，但也仅此而已。

程序列表 13.7

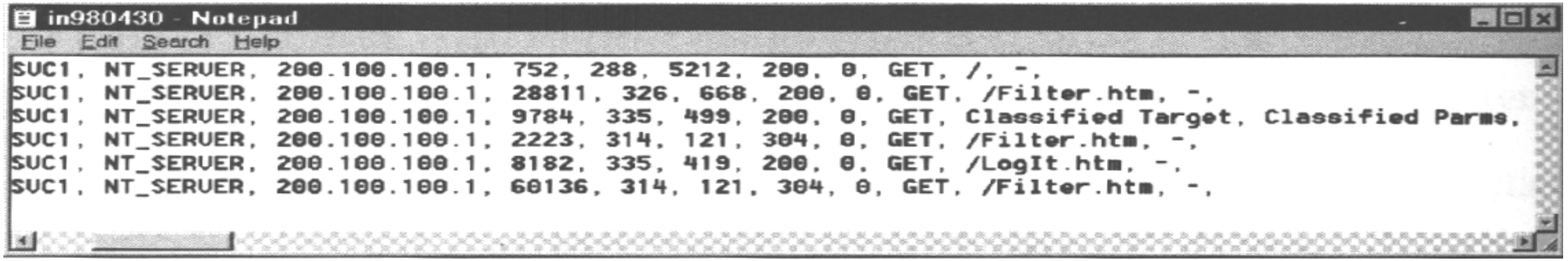
```
<HTML>
<HEAD>
<TITLE>ISAPI Filter Test</TITLE>
</HEAD>
<BODY>
<H3><CENTER>This Page is <EM>NOT</EM> Logged</CENTER></H3>
Which means that you won't see any of the normal entries in the log.<P>
What you will see are a bunch of classified entries instead.
</BODY>
</HTML>
```

程序列表 13.8

```
<HTML>
<HEAD>
<TITLE>ISAPI Filter Test</TITLE>
</HEAD>
<BODY>
<H3><CENTER>This Page is Logged</CENTER></H3>
Which means that you will see all of the normal entries in the log.
</BODY>
```

</HTML>

现在已经作好准备对 ISA 过滤器进行核查，你要做的就是用自己的浏览器打开 Filter.HTM web 页。一定要通过内部网连接访问 Web 服务器。在 Explorer 中双击 Filter.HTM 并将它作为文件打开，这样就不会迫使 Web 服务器做出日志项。在两个测试页之间切换几次后可以打开日志文件。如果你的 Web 服务器设置和我的一样，那么你的 Web 服务器的 System32\LogFiles 目录中每天都应该有一个日志文件。下图就是测试 NoLog ISA 过滤器时的日志项。



请注意，除了 NoLog.HTM 的日志项之外，所有日志项都显示了正确的 Web 页。图中唯一的 NoLog.HTM 项位于第三行（从上往下看），由于 NoLog 过滤器已经修改了日志项，因此，这些项均按密级划分进行标记。另外，LogIt 和 Filter 项都能正常显示。显然这些都是自动完成的，过滤器在后台对日志事件自动作出反应。

13.6 使用 ISAPI 扩展转发服务器信息

几乎每个人都知道本地日期和时间，看一下时钟就够了。如果你是网络管理员或 Web 站点拥有者，管理远程资源时，只知道本地时间通常就不够用了。实际上，更重要的是要知道所辖位置的日期和时间。

下面的示例说明了如何从服务器上取得日期和时间并在浏览器窗口中显示它们。使用这个特定 ISAPI 的方式有多种，但我们讨论的是在一个简单 Web 页上使用它们。你顺便可以获得的最重要概念是，使用 ISAPI 扩展，允许你访问那些你坐在控制台前使用正规程序访问的每一条信息。换言之，坐在计算机前能做的一切也可以远程地管理它们。实际上唯一的限制是你应考虑到标准 Windows NT 的安全性。

技巧 灵活性是一把双刃剑。对于 ISAPI 扩展要考虑一些对普通程序不必考虑的限制。例如，除非你拥有实际管理整个进程的全部能力，否则就要对是否允许远程备份进行再三考虑。坐在 Windows NT 控制台前的管理员能够通过物理地验证磁带编号或其他识别信息看到，你打算使用的磁带不是你用于备份应该使用的那盘磁带，问题是，你不得不问问自己，远程管理员是否也具有这种能力，多数情况下，答案都是否定的。

创建得到时间/日期的 ISAPI 扩展

让我们设计基本程序吧。由于本章中已设计过一个基本的 ISAPI 扩展，本节中不再花费时间讨论程序的细节。请你按下列步骤进行操作吧。

1. 创建一个新的 ISAPI 扩展工程，我把它命名为 GetTD，至于你怎么命名就随你的便了。
2. 对于 ISAPI Extension Wizard（ISAPI扩展向导）Step 1 of 1 没有什么特别可做的事，单击 Finish（完成）即可。
3. 单击 OK，关闭 New Project Information（新建工程信息）对话框，在这之前应先确认所有程序参数。

下面创建代码。和其他 ISAPI 扩展一样，这个例子也要添加三段主要代码。下面的列表就是要添加的东西。

- Parse map（分析映射） 分析映射对 DLL 中的函数如何调用作出定义，它还定义了所需变量以及这些变量映射到你的函数中的方式。
- Function（函数） 显然，如果没有某些向浏览器提供输出的代码，你的 DLL 就没用了。
- Function definition（函数定义） 需要在 ISAPI 扩展的头文件中添加函数说明。

正如你要看到的，我们将对 GetTD.CPP 文件作两处改动，还要对 GetTD.H 文件作一处改动。程序列表 13.9 是 GetTD.CPP 文件的有关代码。对分析映射及 GetTD()函数都要看一下。还要注意对 Default()函数所作的改动，它使得当你在一个 ISAPI 扩展文件中有多个函数时这个函数更有用。程序列表 13.10 列出了

整个 GetTD.H 文件。

程序列表 13.9

```
//////////////////////////////////////////////////////////////////
//command-parsing map

BEGIN_PARSE_MAP(CGetTDExtension,CHttpServer)
        // Get the system time and date.
        ON_PARSE_COMMAND(GetTD,CGetTDExtension,ITS_EMPTY)

        // Default parse map.
        ON_PARSE_COMMAND(Default,CGetTDExtension,ITS_EMPTY)
        DEFAULT_PARSE_COMMAND(Default,CGetTDExtension)
END_PARSE_MAP(CGetTDExtension)

//////////////////////////////////////////////////////////////////
// The one and only CGetTDExtension object

CGetTDExtension theExtension;

//////////////////////////////////////////////////////////////////
// CGetTDExtension implemenation
```

```
CGetTDExtension::CGetTDExtension();
{
}

CGetTDExtension::~CGetTDExtension();
{
}

BOOL CGetTDExtension::CGetTDExtensionVersion(HSE_VERSION_INFO* pVer)
{
        // Call default implementation for initialization
        CHttpServer::GetExtensionVersion(pVer);

        // Load description string
        TCHAR sz[HSE_MAX_EXT_DLL_NAME_LEN+1];
        ISAPIVERIFY(::LoadString(AfxGetResourceHandle(),IDS_SERVER, sz,
                                HSE_MAX_EXT_DLL_NAME_LEN));
        _tcscpy(pVer->lpszExtensionDesc,sz);
        return TRUE;
}
```

```
//////////////////////////////////////////////////////////////////////
// CGetTDExtension command handlers

void CGetTDExtension::Default(CHttpServerContext* pCtxt)
{
	StartContent(pCtxt);
	WriteTitle(pCtxt);

	// display an informational message on how to use this extension.
	* pCtxt << _T("<H2><CENTER>How to Use This ISAPI Extendion </CENTER>
</H2>");
	* pCtxt << _T("Use the <STRONG><EM>GetTD</EM></STRONG>function to");
	* pCtxt << _T("retrieve the current Web server date and time.<P>");

	EndContent(pCtxt);
}

void CGetTDExtension::GetTD(CHttpServerContext* pCtxt)
{
	// Creat a couple of local variables.
```

```
CString cDate;                    // String representation of date.
CString cTime;                    // String representation of time.
CTime oCurDate;                   // CUrrent date and time.

// Get the current date and store it in a string.
oCurDate = CTime::GetCurrentTime();
cDate = oCurDate.format("%A,%d %B %Y");
cTime = oCurDate.format("%I:%M %p");

// Start an HTML page.
StartContent(pCtxt);
WriteTitle(pCtxt);

// Display the time and date.
* pCtxt << _T("The system date is: ");
* pCtxt << _T("cDate);
* pCtxt << _T("<P>");
* pCtxt << _T("The system time is: ");
* pCtxt << _T("cTime");
* pCtxt << _T("<P>");
```

```
        // End the HTML page.
        EndContent(pCtxt);
}
```

程序列表 13.10

```
if !defined(AFX_GETTD_H_E8B71025_2785_11D1_8E36_444553540000__INCLUDED_
)
define AFX_GETTD_H_E8B71025_2785_11D1_8E36_444553540000__INCLUDED_

// GETTD.H - Header file for your Internet Server
//    An ISAPI Extension that retrieves the time  and date.

#include "resource.h"

class CGetTDExtension : public CHttpServer
{
public:
        CGetTDExtension()
        ~CGetTDExtension()

// Overrides
```

```
	// ClassWizard generated virtual function overrides
		//NOTE - ClassWizard will add and remove member functions here.
		// DO NOT EDIT what you see in these blocks of generated code !
	//{{AFX_VIRTUAL(CGetTDExtension)
	public:
	virtual BOOL GetExtensionVersion(HSE_VERSION_INFO* pVer);
	//}}AFX_VIRTUAL

	// Default handler.
	void Default(CHttpServerContext* pCtxt);

	// Special time and date handlers.
	void GetTD(CHttpServerContext* pCtxt);

	DECLARE_PARSE_MAP()
	//{{AFX_MSG(CGetTDExtension)
	//}}AFX_MSG
};
//{{AFX_INSERT_LOCATION}}
//Microsoft Developer Studio will insert additional declarations immediately before the
previous line.
```

```
#endif                                                                    //!defined
(AFX_GETTD_H_E8B71025_2785_11D1_8E36_444553540000__INCLUDED)
```

让我们先讨论一下程序列表 13.9 中的代码。这个分析映射没有什么出人意料之处。我们不需要把来自客户机的数据传递给服务器，所以对 GetID 函数使用了 ITS_EMPTY 常量。

请注意我是如何修改 Default()函数的。通常如果 DLL 中只有一个函数时就要在代码中使用 Default()函数。但是，我发现，当 DLL 中多于一个函数时，作为生成 ISAPI 扩展接口文件的一种方法，使用 Default()函数也是相当方便的，这样，合作者(coworker) 能快速地构造一个 Web 页，查看 ISAPI 扩展提供的文档，然后去处理创建一个实际页的事务。

GetTD()函数是相当基础的一个函数。它只是获取系统时间并把它格式化成两个字符串，一个是时间，一个是日期。请注意我是如何使用 Format()函数来得到这两个字符串的。有了这两个字符串变量后，就可以创建包含其值的基本 Web 页面了。

程序列表 13.10 与我们以前的例子有很大区别。我做的一切工作就是说明 GetTD 函数。这是因为我没有向函数传递任何数据，所以唯一的数据是 pCtxt。

创建测试 GetTD 的 Web 页

现在看一看访问 ISAPI 扩展的 HTML 代码，程序列表 13.11 包含了一个非常简单的测试页面。

程序列表 13.11

```
<HTML>
<HEAD>
<TITLE>Get Time/Date ISAPI Extension Example</TITLE>
</HEAD>
<BODY>

<!- 显示标题 ->
<CENTER><H2>Getting Time and Date from Your Server</CENTER></H2>

<!- 显示到 Default()函数的链接 ->
Click <A HREF="Controls/GetTD.DLL?Default">Here</A> to see the Default( )
function message.<P>

<!- 显示到取得时间/日期页的链接 ->
Click <A HREF="Controls/GetTD.DLL?GetTD">Here</A> to see the Time( ) and
 Date() function message.<P>

</BODY>
</HTML>
```

应该注意到程序列表 13.11 比起以前的例子（程序列表 13.4）有一个主要的

变化。这一次是使用了一个锚地去访问 ISAPI 扩展。用户单击这一链接后，得到的是由 ISAPI 生成的页，而不是像通常那样链接到 Web 服务器的一个静态页。这样做有几个重要的优点：

- 能够通过把检测到的浏览器类型作为参数传递给 ISAPI 扩展来创建浏览器专用页。大多数浏览器都提供了唯一识别信息。
- ISAPI 扩展允许定制 Web 页的内容来反映特定用户的需求。
- 使用 ISAPI 扩展，允许创建基于某些规则（如星期中某天）的链接。例如，能够把星期一欢迎页和星期五会见页连接到同一个主页链接上。

显然，使用类似于 ASP 或老版本的其他技术也能得类似的结果，但事实证明，在 Web 站点通常不使用的许多方式中，你也能使用锚地技术。

注释 记住先在内部网上测试这个 ISAPI 扩展。你不能在本地打开扩展，所以如果作为浏览器中的文件来简单地打开测试 Web 页的话，链接就会失败。

现在看一下这个 ISAPI 扩展的输出。图 13.2 显示了第一次看到的 Web 页。如你所见，它含有一个简单标题和两个链接，ISAPI 扩展中每个函数都有一个链接。

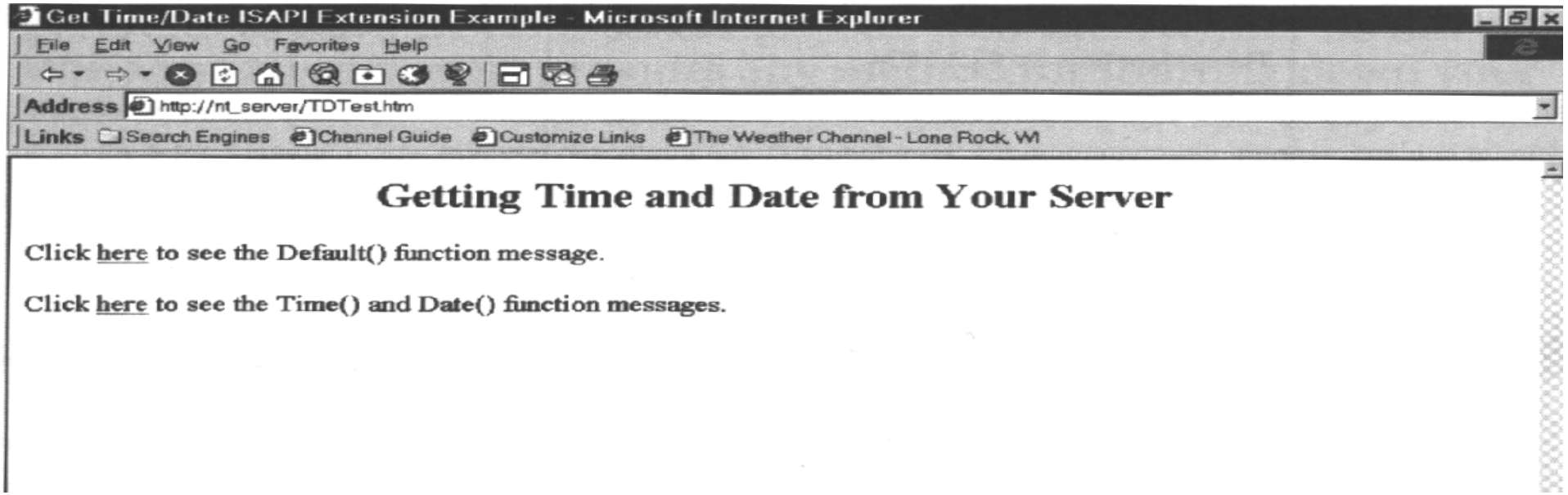
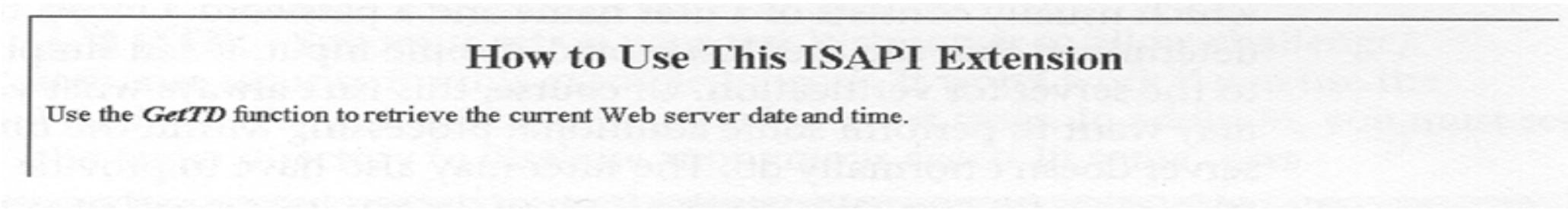


图 13.2　初始的测试页有两个选项，一个去查看 Default( )函数的输出，另一个去查看 GetTD()函数的输出

如果单击第一个链接，你会看到如下图所示的显示内容：



尽管这一页没有给出 ISAPI 扩展的明显的使用指令，但你却知道了它能做些什么。显然，在产品型扩展中可以提供更多的细节，可以包括一个参数列表

（如果有的话）作为 Default()函数页的一部分，有时，甚至可以包括一些与 ISAPI 扩展创建者进行联系的信息。

单击浏览字符串上的 Back（后退）按钮，退回到图 13.2 所示的初始 Web 页。现在试一试第二个链接。在试第二个链接之前，在测试服务器上变更一下时间和日期，这样做是为了证实 ISAPI 扩展确实是按所预期的那样完成了工作。下图是 GetTd()函数输出时间和日期的典型例子。如你所见，Web 页上显示的是服务器上变更了的系统时间，而不是本地时间。

```
The system date is : Saturday,02 May 1998
The system time is : 03: 13 PM
```

13.7 使用 ISAPI 过滤器请求用户名和口令字

存在许多不同的方式来使用 IIS 安全过滤器，但在完成其它工作前，让我们先了解一下如何制作一个简单的过滤器。本节帮助你理解如何创建一个过滤器，该过滤器在服务器上传递值，并使用这些值对用户的合法性进行证实。

过滤器需要做的第一件事是从用户那里得到输入，这些输入通常包含用户名和口令字。一旦过滤器确定用户已提供了某些输入，它就简单地把控制移交给服务器去进行验证工作。当然这并不总是你所想的结果。你也许想在过滤器内完成一些服务器通常不做的附加处理。过滤器也许还要提供某种形式的附加解密功能，或许还要有专门的错误消息处理功能。下列内容是安全过滤器能提供的附加特性。

- 解密　我们总是选择在客户端对用户名和口令字进行加密，然后在服务器端对它们解密。ISAPI过滤器恰好能够胜任这一工作。
- 认证　Windows NT不强制使用内置的认证功能，你总是可以选择自己去认证用户并给予用户适当的安全级。于是，可以进行认证，然后把用户分到某一个组，之后让Windows NT负责细节。有许多不同的认证场合。
- 定制化错误处理　也许你不喜欢Windows NT错误处理的方式，ISAPI过滤器提供了限制最少的错误处理功能。当然，你提供的错误处理的精确层次由客户机和服务器来确定，这意味着你想对用户访问使用的浏览器实施控制。
- 专门化访问处理　有时不想让每个人都能在24小时内随时访问Web站点。无论是IIS还是Windows NT目前都没有办法使你的服务器对于一般用户是关闭的而对另外的用户仍然是开放的。而ISAPI过滤器能基于当前时间或其它规则提供访问，甚至可以基于服务器加载来限制访问某些站点。

现在我们对过滤器能干些什么有了一个一般的了解。我们下面给出一个例子。首先是使用本章前一节中创建ISAPI过滤器的过程创建一个框架。创建工程时有一点差别，这个差别是该程序命名为AuthUser，你可以随便起个其它什么名称。在ISAPI Extension Wizard（ISAPI扩展向导）- Step2 of 2对话框中，选中Client Authentication Request（客户机认证请求）选项，并且不选中End of Connection（连结终止）选项。程序列表13.12是要向OnAuthenticate()函数添加

的代码。随后我们将对这些代码的作用进行说明。

注释 为了能够让本示例正常运行，必须把测试的 Web 服务器设置成允许 Chanllenge / Reponse 安全层次。如果你使用绝大多数情况下使用的 Windows NT Authentication 方法，那么该示例将不能正常工作。另外，还必须把目标目录设置成不允许匿名访问的方式。

程序列表 13.12

```
// AUTHUSER.CPP - Implementation file for your Internet Server
// AuthUser Filter

#include "stdafx.h"
#include "AuthUser.h"

////////////////////////////////////////////////////////////////////
// The one and only CWinApp object
// NOTE: You may remove this object if you alter your project to no
// longer use MFC in a DLL

CWinApp theApp;

////////////////////////////////////////////////////////////////////
```

```
// The one and only CAuthUserFilter object

CAuthUserFilter;

//////////////////////////////////////////////////////////////////
// CAuthUserFilter implementation

CAuthUserFilter::CAuthUserFilter()
{
}

CAuthUserFilter::~CAuthUserFilter()
{
}

BOOL CAuthUserFilter::CGetFilterVersion(PHTTP_FILTER_VERSION pVer)
{
        // Call default implementation for initialization
        CHttpFilter::GetFilterVersion(pVer);

        // Clear the flags set by base class
```

```
        pVer->dwFlags &= ~SF_NOTIFY_ORDER_MASK;

        //Set the flags we are interested in
        pVer->dwFlags |= SF_NOTIFY_OFDER_LOW | SF_NOTIFY_SECURE_PORT |
                                SF_NOTIFY_NONSECURE_PORT                    |
SF_NOTIFY_AUTHENTICATION;

        // Load description string
        TCHAR sz[SF_MAX_FILTER_DESC_LEN+1];
        ISAPIVERIFY(::LoadString(AfxGetResourceHandle(),IDS_FILTER,sz,
                                        SF_MAX_FILTER_DESC_LEN));
        _tcscpy(pVer->lpszFilterDesc,sz)
        return TRUE;

}

DWORD CAuthUserFilter::OnAuthentication(CHttpFilterContext* pCtxt
                , PHTTP_FILTER_AUTHENT pAuthent)
{
        CString oBuffer;                    // Buffer for client output
        DWORD    dwSize;                        // Size of the buffer.
```

```
LPVOID pvInOut;                    // Client input or output.

// See if we're gettint an anonymous request.
if(strlen(pAuthent->pszUser)==0)
{
        // Get the user's name.
        dwSize = 100;
        pvInOut = " ";
        pCtxt->GetServerVariable("REMOTE_USER", pvInOut, &dwSize);
        dwSize = strlen(LPTSTR(PvInOut));
        if (strncmp(LPTSTR(pvInOut), " ", 1) == 0)
        {
                // Set an error condition.
                pCtxt->ServerSupportFunction(SF_REQ_SEND_RESPONSE_HEADER,
                        "401 Access Denied",
                         NULL,
                         NULL);
        }
        else
        {
                // Indicate the user has supplied a password.
```

```
                pCtxt->ServerSupportFunction(SF_REQ_SEND_RESPONSE_HEADER,
                        NULL,
                        NULL,
                        NULL);

                // Store the user's name
                strncpy(pAuthent->pszUser, LPTSTR(pvInOut), dwSize);

                // Get and store the user's password.
                dwSize = 100;
                pvInOut = " ";
                pCtxt->GetServerVariable("AUTH_PASS", pvInOut, &dwSize);
                dwSize = strlen(LPTSTR(vInOut));
                strncpy(pAuthent->pszPassword, LPTSTR(pvInOut), dwSize);
        }
    }

    // return the appropriate status code
    return SF_STATUS_REQ_NEXT_NOTIFICATION;
}
```

```
// Do not edit the following lines, which are needed by ClassWizard.
#if 0
BEGIN_MESSAGE_MAP(CAuthUserFilter,CHttpFilter)
        //{{AFX_MSG_MAP(CAuthUserFilter)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
#endif       // 0

//////////////////////////////////////////////////////////////////////
// If your extension will not use MFC, you'll need this code to make
// sure the extension objects can find the resource handle for the
// module,   If you convert your extension to not be dependent on MFC,
// remove the comments around the following AfxGetResourceHandle()
// and DllMain() functions, as well as the g_hInstance global.

/****

static HINSTANCE g_hInstance;

HINSTANCE AFXISAPI AfxGetResourceHandle()
{
```

```
        return g_hInstance;
}

BOOL WINAPI DllMain(HINSTANCE hInst, ULONG ulReason, LPVOID lpReserved)
{
        if(ulReason == DLL_PROCESS_ATTACH)
        {
                g_hInstance = hInst;
        }

        return TRUE;
}

****/
```

正如你所看到的，比起前一个 ISAPI 过滤器示例来，这儿的代码复杂多了。其中我们需要完整的一件工作是从过滤器直接向服务器传递信息，之后再把它传给客户机。这个过滤器完成了三步操作：

1. 使用 GetServerVariable()函数取得用户名，确定用户是否是匿名地访问服务器。（用户名返回值为空白，表明用户是匿名地访问服务器）。如果是，使用 SeverSupportFuction()函数告诉服务器向客户机发出 401 错误信息。这使得显示出浏览器口令对话框，在这个对话框中用户输入口令字和用户名。

2. 用户输入口令字和用户名之后，使用 GetServerVariable() 函数从输入流中提取它们，把用户名和口令字传递给 Windows NT 进行安全认证。
3. 使用 SF_STATUS_REQ_NEXT_NOTIFICATION 返回值告诉服务器，过滤器已成功地完成使命。

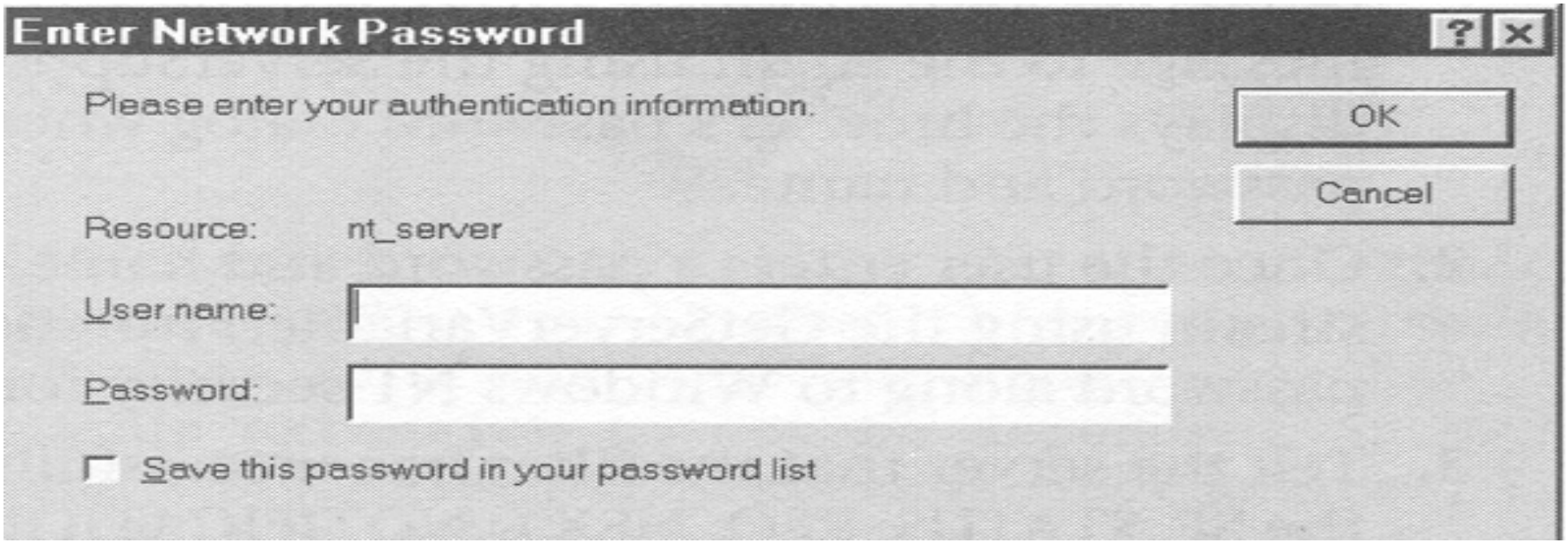
由于需要使用三个步骤，因此，使用安全过滤器时总有些令人吃惊的东西，例如，在使用 GetServerVariable()函数提取用户名与口令字和使用 Strncpy()函数把这一信息拷贝到 pAuthent 结构之间的时间内，对过滤器进行第二次操作期间将会进行解密。在第三个步骤中也要发生一些事情，例如，要对你需要直接与用户交互作出决定，这通常发生于对过滤器进行的第三次操作中。

请注意，我提供了 SF_STATUS_REQ_NEXT_NOTIFICATION 这个返回值。这是让服务器处理认证用户和显示 Web 页等细节的一个信号。如果想自己完成全部认证工作，就需要显示请求的信息。

技巧 IIS 通常在报错前提供三次机会让你正确地输入口令字。有两种办法可以防止这一行为的发生。第一种，在对过滤器进行的第二、第三次操作之间，自己完成认证。这样就可以给用户以另外的机会来提供正确的口令字，并将正确的用户名和口令字传递给服务器。第二种方法是使用 SF_STATUS_REQ_ERROR 返回值在第一次企图访问 Web 服务器失败后就终止进一步的访问企图。这一个返回值将显示一个服务器错误消息，并且可能会欺骗一些不那么精明的用户，让它们以为服务器发生了问题，从而不再企图访问服务器。

一旦编译了这个过滤器，使用本章第一个 ISAPI 过滤器相同的技术安装它。

运行这个示例就会看到如下图所示的口令对话框。



如果你键入了并不存在的用户名或口令字，则会看到如下图所示的服务器生成的报错消息。

输入一个有效的用户名和口令字就可以访问该 Web 站点了。

HTTP Error 401

401.1 Unauthorized: Logon Failed

This error indicates that the credentials passed to the server do not match the credentials required to log on to the server.

Please contact the Web server's administrator to verify that you have permission to access the requested resource.