Kapitel 24

ActiveX-Dokumente und das Internet

Ich habe die Dokumentation für die ActiveX-Dokumente mehrmals durchgelesen, um die Bedeutung dieser Technologie zu verstehen. Aber erst als ich diese Dokumente wirklich einsetzte, begann ich, all ihre Möglichkeiten zu verstehen. In diesem Kapitel will ich Ihnen nichts mehr über ActiveX-Dokumente erzählen, sondern sie Ihnen vielmehr zeigen. Dazu hatte ich das Vergnügen, eine fiktive Immobilienfirma zu erfinden. Ich hoffe, die Site gefällt Ihnen. Sie setzt wirklich viele ActiveX-Dokumente ein. Ich empfehle Ihnen jedoch äußerste Vorsicht, bevor Sie einen Scheck oder eine Kreditkartennummer dorthin senden!

24.1 Das ActiveX-Dokument Realty

Really Realty Services Inc. ist ein kleiner, frei erfundener Laden mit großen Zielen. Seine Besitzer wollen nicht nur Immobilien im Web anbieten; sie wollen die aktuellen Informationen über eine Immobilie darstellen, und gleichzeitig ein interaktives Programm bereitstellen, das potentiellen Kunden erlaubt, ihre Hypothekenzahlungen zu berechnen, und sogar Online-Gebote zu machen. Sie hätten dazu auch ein paar komplexe, formularbasierte Web-Seiten verwenden können, aber sie haben beschlossen, daß wirklich nichts dafür spricht, ihren eigenen Server mit Dingen wie der Zinsberechnung zu belasten. Viel besser verwendet man dazu die Programme, die schon auf der Client-Maschine vorhanden sind.

Darüber hinaus sind sie Profis in der Visual-Basic-Programmierung, und sie wissen nicht viel über komplizierten, formularbasierten HTML-Code und serverseitige Programmierung. Mit Hilfe eines ActiveX-Dokuments sind sie in der Lage, alle Ressourcen des lokalen Systems zu nutzen. Sie sind noch nicht sicher, was sie mit diesen Möglichkeiten anfangen werden, aber sie wissen, daß das Ganze irgendwann sicher sehr praktisch sein wird.

Abbildung 24.1 zeigt die Seite, die die Firma angelegt hat, um Informationen über eine Immobilie anzuzeigen. Es gibt ein Bildfeld, das ein Bild der Immobilie zeigt, ein Beschriftungsfeld, das den aktuellen Preis anzeigt, sowie ein Textfeld, in dem die Immobilie beschrieben wird.

Das ActiveX-Dokument hat drei Eigenschaften: Preis, Bild und Beschreibung (`Price`, `Picture` und `Description`). Die Eigenschaftsprozeduren für diese Eigenschaften und die Persistenzfunktionen sehen wie folgt aus:

```
' Guide to the Perplexed
' Realty-Beispiel
' Copyright © 1997 by Desaware Inc. All Rights Reserved.
' Really Realty Services Inc. ist eine frei erfundene Firma
' etwaige Ähnlichkeiten zu einer echten Firma sind völlig zufällig
Option Explicit

Dim m_Price As Currency
```

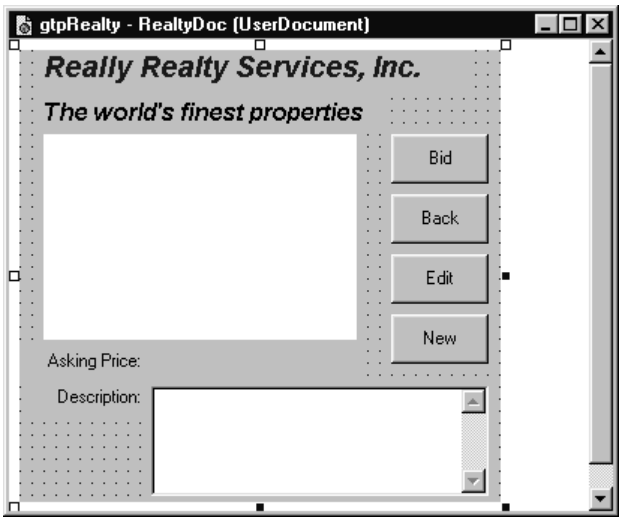

Abb. 24.1: Das ActiveX-Dokument realty.dob zur Entwurfszeit

```
Public Property Get Picture() As Picture
Set Picture = Picture1.Picture
End Property

Public Property Set Picture(ByVal vNewValue As Picture)
   Set Picture1.Picture = vNewValue
   PropertyChanged "Picture"
End Property

Public Property Get Price() As Currency
   Price = m_Price
End Property

Public Property Let Price(ByVal new_Price As Currency)
   If new_Price < 0 Then Exit Property
   m_Price = new_Price
   lblPrice.Caption = Format$(m_Price, "Currency")
   PropertyChanged "Price"
End Property

Public Property Get Description() As String
   Description = txtDescription.Text
End Property

Public Property Let Description(new_Description As String)
   txtDescription.Text = new_Description
   PropertyChanged "Description"
```

```
End Property

Private Sub UserDocument_InitProperties()
   cmdEdit.Visible = True
   cmdNew.Visible = True
End Sub

Private Sub UserDocument_ReadProperties(PropBag As PropertyBag)
   txtDescription.Text = PropBag.ReadProperty("Description")
   m_Price = PropBag.ReadProperty("Price")
   lblPrice.Caption = Format$(m_Price, "Currency")
   Set Picture1.Picture = PropBag.ReadProperty("Picture")
End Sub

Private Sub UserDocument_WriteProperties(PropBag As PropertyBag)
   Call PropBag.WriteProperty("Description", txtDescription.Text)
   Call PropBag.WriteProperty("Price", m_Price)
   Call PropBag.WriteProperty("Picture", Picture1.Picture)
   ' Nachdem die Daten geschrieben sind, kann die Seite
   ' nicht mehr bearbeitet werden
   cmdEdit.Visible = False
   cmdNew.Visible = False
End Sub
```

Es gibt auch vier Schaltflächen. Die BID-Schaltfläche wird genutzt, um ein Formular für die Abgabe eines Angebots anzuzeigen. Dabei handelt es sich um ein separates, modales Visual-Basic-Formular, das dem Client nicht nur erlaubt, Gebote für eine Immobilie abzugeben, sondern die auch andere Aufgaben übernimmt, die für die Abgabe eines Gebots erforderlich sind. Die Schaltfläche BACK stellt nur eine Navigationshilfe dar, womit der Client zur vorherigen Browser-Seite zurückkehren kann, ohne dazu die Symbolleiste oder das Menü des Browsers zu benötigen.

Dieses ActiveX-Dokument soll in zwei verschiedenen Modi arbeiten. Wenn der Browser eine leere `.VBD`-Datei öffnet, wird das `InitProperties`-Ereignis ausgelöst und die Schaltflächen EDIT und NEW werden angezeigt (zunächst sind sie nicht sichtbar). Das passiert im folgenden Code:

```
Private Sub UserDocument_InitProperties()
   cmdEdit.Visible = True
   cmdNew.Visible = True
End Sub
```

Dieser Modus ist für die eigene Arbeit der Realty-Firma gedacht. Er wird bei der Entwicklung der Site eingesetzt. Man verwendet dort den Internet-Browser von Microsoft, um die leere Datei `RealtyDoc.vbd` zu öffnen, und nutzt dann die Schaltflächen EDIT und NEW, um neue `.VBD`-Dokumente zu erzeugen, in denen das Bild, der Preis und die Beschreibung der Immobilie enthalten sind. Beim

Laden dieser Dokumente wird das `ReadProperties`-Ereignis ausgelöst, das die Schaltflächen EDIT und NEW nicht sichtbar macht. Diese beiden Schaltflächen bleiben auch verborgen, nachdem das `WriteProperties`-Ereignis aufgetreten ist, um zu verhindern, daß die Seiten erneut verändert werden.

Die Bearbeitung der aktuellen Seite ist ganz einfach. Für die Bearbeitung der Dokumenteigenschaften wird das separate Formular `frmEdit` verwendet. Es wird mit Hilfe des folgenden Codes angezeigt, nachdem die Schaltfläche EDIT angeklickt wurde. Abbildung 24.2 zeigt das Edit-Formular.

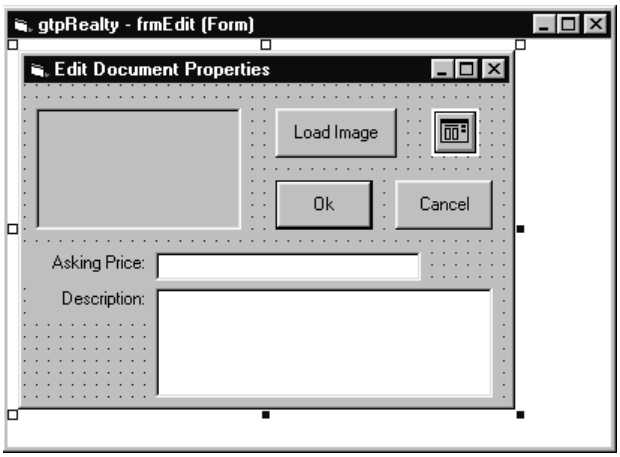

Abb. 24.2: Das Formular zur Dokumentbearbeitung

```
Private Sub cmdEdit_Click()
 Set frmEdit.OwnerDoc = Me
 frmEdit.Show vbModal
 Set frmEdit = Nothing
End Sub
```

Dieses Formular enthält ein Vorschaufenster, in dem ein Bild der Immobilie angezeigt wird. Es verwendet zwei Textfelder für die Bearbeitung des Preises und der Beschreibung, sowie ein Standarddialog-Steuerelement zum Laden der Bilddatei. Der Code für dieses Formular sieht wie folgt aus:

```
Option Explicit

Public OwnerDoc As RealtyDoc

Private Sub cmdCancel_Click()
   Unload Me
End Sub
```

```
Private Sub cmdLoadImage_Click()
   Dim newfile$
   CmnDialog1.ShowOpen
   If CmnDialog1.FileTitle = "" Then Exit Sub
   newfile = CmnDialog1.filename
   On Error GoTo badimage
   Set Picture1.Picture = LoadPicture(newfile)
   Exit Sub
badimage:
End Sub

Private Sub cmdOk_Click()
   OwnerDoc.Price = txtPrice.Text
   OwnerDoc.Description = txtDescription.Text
   Set OwnerDoc.Picture = Picture1.Picture
   Unload Me
End Sub

Private Sub Form_Load()
   txtPrice.Text = OwnerDoc.Price
   txtDescription.Text = OwnerDoc.Description
   Set Picture1.Picture = OwnerDoc.Picture
End Sub
```

Die Variable `OwnerDoc` ist eine öffentliche Eigenschaft des Formulars, die durch das `Document`-Objekt gesetzt wird, bevor das Formular angezeigt wird. Damit ist es möglich, daß das Formular auf die öffentlichen Eigenschaften des Dokuments zugreift. Sie können auch auf die `Friend`-Eigenschaften und Methoden des `Document`-Objekts zugreifen, was aber in diesem Beispiel nicht gezeigt ist.

Wenn Sie die Einstellungen für das Standarddialog-Steuerelement betrachten, stellen Sie fest, daß seine `Filter`-Eigenschaft auf Bilder gesetzt ist (`*.bmp`; `*.ico`; `*.wmf`; `*.gif`). WMF? Das ist eine Windows-Metadatei. Seit wann kann ein Web-Browser eine Windows-Metadatei herunterladen und anzeigen? Die meisten können das nicht. Aber das ist egal. Sie werden sehen, der Web-Browser lädt die Metadatei nicht herunter und zeigt sie nicht an. Er lädt die `.VBD`-ActiveX-Dokumentdatei herunter. Diese Dokumentdatei enthält die Metadatei, und Visual Basic weiß, wie man Metadateien liest und in Bildfeldern anzeigt.

Bedeutet das, daß ActiveX-Dokumente Ihnen erlauben, fast jedes Dateiformat zu verarbeiten, oder sogar selbst benutzerdefinierte Dateiformate anzulegen, die Sie bearbeiten und anzeigen können, unabhängig von den Fähigkeiten des Browsers? Ja. Dieses Beispiel verwendet Metadatei-Bilder für die gesamte Grafik. Im Rahmen dieses Buchs ist es nicht möglich, die Metadateien vollständig zu beschreiben. Kurz gesagt, unterscheiden sie sich in zweierlei Hinsicht von Bitmaps:

- Sie enthalten Grafikzeichenbefehle statt Bitmapinformationen. Das bedeutet, sie können wesentlich kleiner als eine Bitmap sein.
- Sie sind vollständig skalierbar und verlieren bei einer Größenänderung keine Bildqualität.

Die EDIT-Schaltfläche bearbeitet das aktuelle Dokument. Sie wollen aber nicht das leere `RealtyDoc`-Dokument ändern. Sie müssen es als leere Schablone für alle weiteren Dokumente aufbewahren. Aus diesem Grund sollte es als read-only deklariert werden. Und man braucht eine Möglichkeit, neue Dokumente zu erzeugen.

Eine Methode dafür ist im folgenden Befehlscode für `cmdNew_Click` demonstriert:

```
Private Sub cmdNew_Click()
   Dim newname$
   Dim newpath$, oldpath$
   Dim searchloc&, curloc&, curchar$
   newname$ = InputBox("Enter new document name", "New")
   If newname$ = "" Then Exit Sub
   oldpath = Parent.locationname
   newpath = oldpath
   curloc = Len(newpath)
   Do While curloc > 0
 curchar$ = Mid$(newpath, curloc, 1)
 If curchar = "/" Or curchar = "\" Then
    newpath = Left$(newpath, curloc)
    Exit Do
 End If
 curloc = curloc - 1
   Loop
   newpath = newpath & newname$ & ".vbd"
   ' Hier Test auf exist. Datei einfügen
   FileCopy oldpath, newpath
   m_HyperLinked = True
   Hyperlink.NavigateTo "file://" & newpath
End Sub
```

Die `Locationname`-Eigenschaft des `Parent`-Objekts stellt den Dokumentnamen bereit. Es ist unter Visual Basic nicht möglich, die Eigenschaft `App.Path` zu verwenden, weil Sie damit die Position des Servers erhalten, nicht des aktuellen Dokuments.

Vom Dokumentnamen wird vorausgesetzt, daß er sich auf dem lokalen System befindet. Wenn Sie ihn über eine Web-Site laden, ergibt dieser Code einen Fehler. In dieser speziellen Applikation wird jedoch vorausgesetzt, daß neue Eigenschaften nur auf dem lokalen System definiert werden. Die Routine extrahiert den endgültigen Dokumentnamen aus dem Pfad und hängt einen neuen Dokumentnamen

an, der über ein Eingabefeld eingegeben wird. Der Name sollte ohne Erweiterung angegeben werden; ein robusteres Beispiel wird das zeigen. Der Befehl `FileCopy` wird verwendet, um die neue Dokumentdatei zu erzeugen. Beachten Sie, daß es sich dabei um ein leeres Dokument handelt.

Im nächsten Schritt gehen Sie in das neue Dokument. Das erfolgt, indem die globale Variable `m_HyperLinked` auf `True` gesetzt wird. Diese Variable zeigt einem neu geladenen Dokument an, daß es nach einer `New`-Operation geladen wurde, und teilt ihm mit, daß es beim `Show`-Ereignis das Bearbeitungsformular anzeigen soll. Das `Hyperlink`-Objekt wird anschließend genutzt, um auf die neue Seite zu gehen.

Die Variable `m_HyperLinked` ist als Standardformular definiert, es ist also global für alle `Document`-Objekte. Wenn sie `True` ist, wird das Bearbeitungsformular angezeigt, wie Sie im folgenden sehen:

```
Private Sub UserDocument_Show()
   If m_HyperLinked Then
 m_HyperLinked = False
 Set frmEdit.OwnerDoc = Me
 frmEdit.Show vbModal
 Set frmEdit = Nothing
   End If
End Sub
```

Das ActiveX-Dokument enthält ein ABOUT-Feld, das auf die übliche Weise implementiert wird:

```
Private Sub mnuAbout_Click()
   frmAbout.Show vbModal
End Sub
```

Das Dokument hat ein Menü, das die Befehle HELP und ABOUT beinhaltet. Das `Negotiate`-Attribut ist für das Menü gesetzt, so daß es in das Menü des Containers eingefügt wird.

Für die Beschreibung wird ein Textfeld erzeugt, weil dieses das Scrolling unterstützt. Das Dokument erlaubt jedoch nicht, die Beschreibung auf der eigentlichen Dokumentseite zu bearbeiten. Um Änderungen zu verhindern, werden Tastencodes aufgefangen, so daß im Textfeld nichts geändert oder gelöscht werden kann:

```
Private Sub txtDescription_KeyDown(KeyCode As Integer, Shift As Integer)
   If KeyCode = vbKeyDelete Then
 KeyCode = 0
   End If
End Sub
Private Sub txtDescription_KeyPress(KeyAscii As Integer)
   KeyAscii = 0
End Sub
```

Die Implementierung des Browser-Befehls BACK ist trivial:

```
Private Sub cmdBack_Click()
   Hyperlink.GoBack
End Sub
```

Das Bid-Formular wird mit dem folgenden Code angezeigt:

```
Private Sub cmdBid_Click()
   Set frmBid.OwnerDoc = Me
   frmBid.Show vbModal
   Set frmBid = Nothing
End Sub
```

Abbildung 24.3 zeigt das Formular.

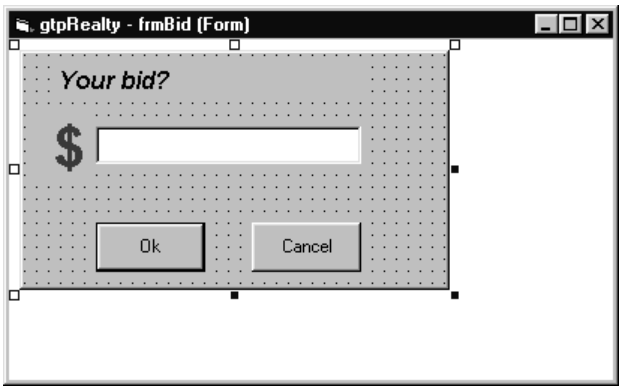

Abb. 24.3: Das Bid-Formular zur Entwurfszeit

Noch macht das Formular nichts mit der eingegebenen Information. Es handelt sich jedoch um ein Standardformular von Visual Basic, Sie können also an dieser Stelle alles tun, was immer Sie möchten. Sie könnten eine E-Mail-Nachricht für das Gebot erstellen. Sie könnten eine Zinszahlung berechnen und abhängig vom Ergebnis bestimmte Informationen senden. Sie könnten dem Benutzer erlauben, eine Audio-Nachricht aufzunehmen und sie irgendwo auf eine FTP-Site laden.

Damit kommen wir zu einem wichtigen Aspekt. Standard-Web-Sites sind darauf ausgelegt, kleine Datenmengen vom Client entgegenzunehmen, während große Datenmengen zum Client zurück geschickt werden. In HTML werden relativ kurze Text-Befehlszeilen verwendet, um Informationen von einem Server anzufordern. Wenn Sie einmal in Visual Basic sind, haben Sie sehr viel mehr Flexibilität an der Hand, Informationen an den Server zu senden. Sie könnten beispielsweise Daten auf eine FTP-Site laden, und dann einen Befehl mit dem Dateinamen senden. Sie könnten aber auch über ein Winsock-Steuerelement eine separate direkte Internet-Verbindung verwenden.

ActiveX-Dokumente können eine ausgezeichnete Lösung für diese Art Situation darstellen. Wenn die fiktiven Angestellten, die bei Really Realty Services, Inc. arbeiten, irgendwann nicht mehr genug Arbeit haben, könnten sie beispielsweise einen Dating-Service anbieten, wo Sie ein VB-Formular nutzen, um Ihr Bild und vielleicht einen kurzen, persönlichen Videoclip über das Internet direkt auf das Computersystem Ihres Services senden.

Nun betrachten wir, was erforderlich ist, damit das Dokument auf einer Web-Site eingesetzt werden kann.

24.2 ActiveX-Dokumente und HTML

Der Prozeß, ein ActiveX-Dokument für die Veröffentlichung im Internet aufzubereiten, ist identisch zu dem für ActiveX-Steuerelemente und wird hier nicht noch einmal beschrieben. Der einzige Unterschied ist, daß Sie hier den Dokument-Server veröffentlichen, nicht das Dokument selbst.

Wenn Sie im `Setup`-Unterverzeichnis im Beispielverzeichnis Chapter 23 auf der CD zum Buch nachsehen, finden Sie die Datei `Realty.htm`, die HTML-Seite, die durch den Setup-Assistenten von Visual Basic erzeugt wird:

```
<HTML>
<HEAD>
<TITLE>realty.CAB</TITLE>
</HEAD>
<BODY>

<a href=RealtyDoc.VBD>RealtyDoc.VBD</a>
<!--********** Comment Begin **********
    Internet Explorer Version 3.x HTML
    ==================================
    The following HTML code has been commented
    out and provided for ActiveX User Documents
    download support in IE 3.x only.  This
    HTML script may not work properly in later
    versions of Internet Explorer.

    Additional information about downloading
    ActiveX User Documents in IE 3.x can be
    found in Microsoft's online support on the
    internet at http://support.microsoft.com.
    ********** Comment End   ********** -->

<!--********** Comment Begin **********
<HTML>
<OBJECT ID="RealtyDoc"
CLASSID="CLSID:18C62902-7B03-11D0-91BB-00AA0036005A"
CODEBASE="realty.CAB#version=1,0,0,9">
```

```
</OBJECT>

<SCRIPT LANGUAGE="VBScript">
Sub Window_OnLoad
    Document.Open
    Document.Write "<FRAMESET>"
    Document.Write "<FRAME SRC=""RealtyDoc.VBD"">"
    Document.Write "</FRAMESET>"
    Document.Close
End Sub
</SCRIPT>
</HTML>
    ********** Comment End   ********** -->

</BODY>
</HTML>
```

Hier sehen Sie den vielleicht wichtigsten Aspekt bei der Entwicklung clientseitigen Codes. Sie können nie sicher sein, was die Clients auf ihrem System installiert haben. Bei Versionen des Internet Explorers vor 4.1 SP1 mußte man eine `.CAB`-Datei von einer Web-Seite laden, und die Web-Seite mußte eine Umleitung zu der `.VBD`-Datei vornehmen. Neuere Versionen des Internet Explorers sind in der Lage, die `CODEBASE`-Information direkt aus der `.VBD`-Datei zu lesen – mehr darüber später.

Nun wollen wir betrachten, was für frühere Versionen des Internet Explorers passiert. Das <`OBJECT`>-Tag sollte Ihnen aus dem Beispiel mit dem ActiveX-Steuerelement bekannt sein. Die `.CAB`-Datei kann signiert werden, so wie im Beispiel mit dem ActiveX-Steuerelement. Die Server innerhalb der Datei können als sicher für die Initialisierung und sicher für das Skripting gekennzeichnet werden, entweder unter Verwendung der Registrierungstechnik oder mit der `IObject-Safety`-Technik, die in Kapitel 21 beschrieben wurde. Beachten Sie, daß dieser Beispielcode das Skripting verwendet, deshalb erhalten wir eine Warnung, wenn die Komponente nicht als sicher für das Skripting gekennzeichnet ist.

Weil wir schon beim Skripting sind: was für ein Skript erscheint unterhalb des <`OBJECT`>-Tags? Wenn diese HTML-Seite durch den Browser geladen wird, prüft dieser zuerst, ob das Objekt auf dem System vorhanden ist und lädt es gegebenenfalls herunter. Anschließend führt sie den gezeigten VBScript-Code aus. Dieser Code öffnet die aktuelle Seite und beginnt, neuen HTML-Code in die Seite zu schreiben. Als erstes schreibt er ein <`FRAMESET`>-Tag, das definiert, daß es sich beim Nachfolgenden um einen Frame handelt, der das gesamte Browser-Fenster übernimmt. Anschließend bestimmt er, daß der Frame mit dem ActiveX-Dokument geladen wird, in diesem Fall ist das die Standarddatei `RealtyDoc.VBD`. Danach schließt er das Frameset und das Dokument. Der Browser wird also angewiesen, die .VBD-Datei zu laden und sie anzuzeigen.

Die hier gezeigte Seite lädt – falls nötig – einen Dokument-Server herunter und zeigt dann ein Dokument an. Das Ganze ist jedoch etwas irreführend. Ich fürchte, viele Entwickler von ActiveX-Dokumenten haben den Eindruck, daß dies die einzige Möglichkeit ist, ein ActiveX-Dokument anzuzeigen, und daß für jedes ActiveX-Dokument eine eigene HTML-Seite zum Laden und zum Anzeigen erforderlich ist. Das kann der Fall sein, wenn Sie ein einzelnes ActiveX-Dokument verwenden, aber wenn Ihre Site viele Dokumente einsetzt, dann ist das alles andere als effizient.

Die Programmierer der Firma Really Realty Services wußten es besser. Sie sehen ihr Werk, indem Sie die Datei `default.htm` im Verzeichnis für Kapitel 24 auf der CD zum Buch mit dem Microsoft Internet Explorer öffnen. Die Seite ist hier gezeigt:

```
<!DOCTYPE HTML PUBLIC "-//W3O/DTD HTML//EN">

<html>
<!-- *** Comment
    The following HTML is needed when using this application
    from Internet Explorer versions before 4.01 SP1.
    It loads the ActiveX Doc server.
    IE 4.01 SP1 and later can read the codebase info directly
    from the VBD file if VBSetCodeBase is used.
<OBJECT ID="RealtyDoc"
CLASSID="CLSID:18C6292-7B03-11D0-91BB-00AA0036005A"
CODEBASE="realty.CAB#version=1,0,0,8">
</OBJECT>
     *** Comment ends -->
<head>
<title>mainpage</title>
<meta name="FORMATTER" content="Microsoft FrontPage 1.1">
</head>
<frameset rows="23%,77%">
    <frame src="frbanner.htm" name="banner" marginwidth="1"
    marginheight="1">
    <frameset cols="30%,70%">
        <frame src="frconten.htm" name="contents" marginwidth="1"
        marginheight="1">
        <frame src="frmain.htm" name="main" marginwidth="1"
        marginheight="1">
    </frameset>
    <noframes>
    <body>
    <p> </p>
    <p>This web page uses frames, but your browser doesn't
    support them.</p>
    </body>
```

```
    </noframes>
</frameset>
</html>
```

Das ist die Root-Seite für die Site. Die Firma erwartet, daß jeder, der die Site besucht, auf diese Seite gelangt, wenigstens beim ersten Zugriff. Hier findet also das Herunterladen des Dokument-Servers statt. Die Seite hat einen Überschrifts-Frame mit dem Firmennamen und einer Liste der Immobilien auf der linken Seite. Der Hauptteil der Seite, rechts, enthält zunächst einen einführenden Text. Die Inhaltsseite, frconten.htm, enthält Links, die direkt auf die ActiveX-Dokumente gehen. Beispielsweise erscheint die folgende Zeile als Big Ben mit blauem Text auf der Web-Seite (oder welche Farbe Ihr Browser eben verwendet, um einen Link darzustellen):

```
<h3><a href="bigben.vbd">Big Ben</a></h3>
```

Wenn Sie auf den Link klicken, lädt der Browser die Datei `bigben.vbd` von der Site herunter. Er wertet die Datei aus und stellt fest, daß es sich um eine strukturierte Speicherdatei handelt, die dem Standard von ActiveX-Dokumenten entspricht. Er extrahiert die CLSID des Servers und prüft den Server, wonach er feststellt, daß dieser als `realty.dll` registriert ist, der zuvor heruntergeladen wurde. Der Browser fordert den Server auf, ein `RealtyDoc`-Objekt zu erzeugen. Das `ReadProperties`-Ereignis des Objekts wird ausgelöst, so daß die Eigenschaftsinformationen aus der Datei `bigben.vbd` gelesen werden können, die zuvor heruntergeladen wurde. Der Browser zeigt das ActiveX-Dokument im Haupt-Frame an, dem Standardziel für alle Links auf der aktuellen Seite.

Der Code für die Inhaltsseite sieht wie folgt aus:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">

     <html>

     <head>
     <title>Table of Contents Frame in mainpage</title>
     <meta name="GENERATOR" content="Microsoft FrontPage 1.1">
     <base target="main">
     </head>

     <body>
     <h3>Choose from among <br>
     these fine properties:</h3>
     <h3><a href="bigben.vbd">Big Ben</a></h3>
     <h3><a href="capitol.vbd">U.S. Capitol</a></h3>
     <h3><a href="eiffel.vbd">Eiffel Tower</a></h3>
     <h3><a href="empire.vbd">Empire State</a></h3>
     <h3><a href="sydney.vbd">Opera House</a></h3>
     <h3><a href="tajmahal.vbd">Taj Mahal</a></h3>
```

```
</body>

</html>
```

Es ist vorstellbar, daß ein Client Probleme bekommt, wenn er seine Lesezeichen direkt auf eines dieser Dokumente setzt und den Zugriff auf seinen Server verliert, aber wenn das passiert, wird er vom Browser benachrichtigt. Der Client braucht einfach nur auf die Homepage zurückzugehen, um den Dokument-Server neu zu laden. Andererseits haben die Leute bei Realty, die zutiefst gläubig immer die allerneuesten Technologien implementieren, beschlossen, daß jeder, der auf ihre Site zugreift, die neueste Version des Internet Explorers braucht, sie haben also den <OBJECT>-Abschnitt einfach völlig auskommentiert und die CODEBASE-Information in den .VBD-Dateien plaziert.

Wie haben sie das gemacht? Sie haben im What's New-Abschnitt für VB6 gelesen, daß es ein Codebase Fixup Utility gibt, das Codebase- und Versionsinformationen in die .VBD-Datei einbetten kann. Dort wird nicht viel darüber gesagt, und auch nicht darauf hingewiesen, wo man es findet, aber es gab einen Verweis auf eine Readme-Datei. Die Readme-Datei beschrieb diese Möglichkeit etwas genauer und erklärte, daß das Packaging-Utility die Codebase automatisch in die .VBD-Datei einbetten würde.

Leider macht dieses Packaging-Utility das nur für die leere .VBD-Datei, die bei der Kompilierung erzeugt wird. Die Leute bei Realty mußten diese Information allen anderen .VBD-Dateien hinzufügen, die sie angelegt hatten. Dazu brauchten sie ein weiteres Utility, SetCodeBase, das in der Readme-Datei beschrieben wurde, und das jeder .VBD-Datei CODEBASE-Informationen hinzufügen kann. Dieses Utility war leider nicht Bestandteil von Visual Studio, man durchsuchte also vertrauensvoll eine Stunde lang die Web-Site von Microsoft, bis man den Abschnitt für Visual-Basic-Anwender gefunden hatte, wo es als Download bereitgestellt wurde. Die Web-Site von Microsoft ändert sich häufig, aber Sie könnten die Suche beispielsweise unter `http://msdn.microsoft.com` beginnen.

Nachdem die .VBD-Dateien markiert waren, begannen die Leute von Realty, die Downloads des ActiveX-Dokuments für ihre Site zu testen. Sie verbrachten Stunden damit und hatten nur mäßigen Erfolg. Der Internet Explorer wurde nämlich sorgfältig definiert, alle Downloads im Hintergrund auszuführen, ohne den Benutzer mit Informationen über deren Fortschritt zu langweilen. Leider ist der Internet Explorer ebenso schweigsam, wenn etwas fehlschlägt, ohne den geringsten Hinweise darauf, wo das Problem liegen könnte (das ist eine Funktion). Die Realty-Leute entdeckten schließlich, daß eine der Dateien, auf die in der .CAB-Datei verwiesen war, nicht an der vorgegebenen URL zur Verfügung stand (in .CAB-Dateien können Sie die Anweisung geben, eine Datei herunterzuladen, statt sie direkt in die .CAB-Datei einbetten zu müssen). Nachdem man die Datei eingebettet hatte, funktionierte alles einwandfrei.

Schließlich doch ein Erfolg! Das Ergebnis sehen Sie in Abbildung 24.4.

Ich empfehle Ihnen, diese kleine Site genauer zu betrachten und ein paar eigene Dokumente zu erzeugen.

Ach ja, ich wollte Ihnen noch diese Geschichte über die Probleme, die die Realty-Leute hatten, erzählen. Das ist genau die Geschichte, mit der ich meine letzten acht Stunden verbracht habe.

Die Geschichte wirft eine ernsthafte Frage auf: Haben die Realty-Leute den richtigen Ansatz gewählt? Hätten sie eine serverseitige Lösung mit IIS-Applikationen verwenden sollen? Hätten sie DHTML verwenden können? Sollte man einfach nur eine Standalone-Applikation entwickeln, die direkte Internet-Verbindungen herstellen könnte? Ich werde in Kapitel 28 noch genauer darauf eingehen.

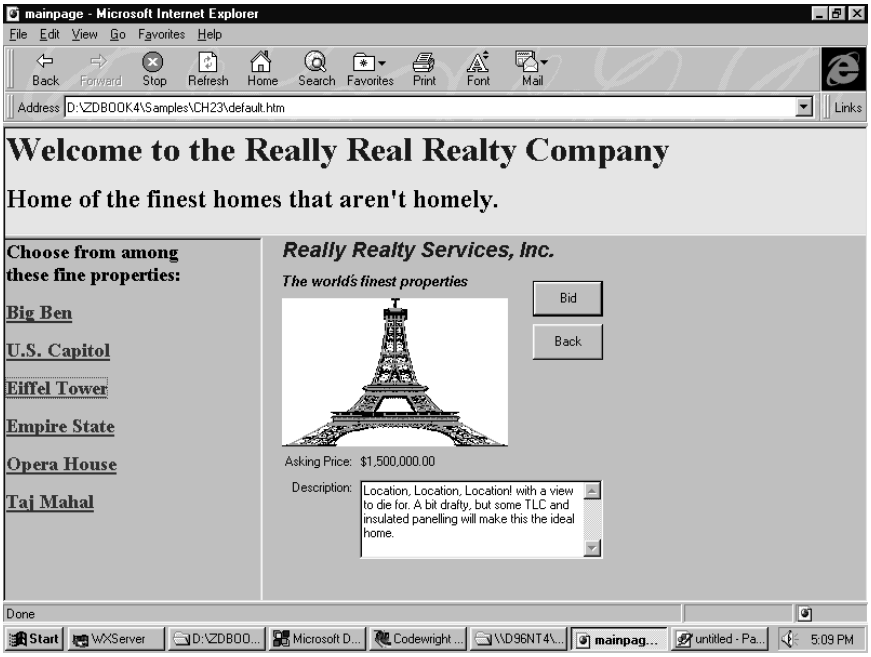

Abb. 24.4: Einfache Seite von der Realty-Site

Ich habe versucht, die ActiveX-Dokument-Technologie so unvoreingenommen wie möglich darzustellen, im Zweifel für den Angeklagten. Jetzt muß ich schließen, daß diese Technologie noch nicht »bereit für den Ernstfall« ist, so wie sie jetzt ist. Wenn Sie diese Technologie einsetzen möchten, sollten Sie unbedingt auch die erforderlichen Werkzeuge verwenden, um eine Remote-Diagnose und eine Prüfung von Abhängigkeiten ausführen zu können (es gibt Drittanbieterwerkzeuge, unter anderem VersionStamper von Desaware, die das für Sie erledigen). Sie können sich nicht auf die Installations-Technologie verlassen, die in den Internet Explorer eingebaut ist, um die Probleme, die Ihnen auf echten Client-

Systemen begegnen, zu erkennen und zu verarbeiten, weil es fehlerhafte Konfigurationen, fehlerhafte Registrierungen und Komponentenkonflikte geben kann.

Damit beenden wir unsere Betrachtung der ActiveX-Dokumente. Nachdem ich die Gelegenheit hatte, sie auszuprobieren, bin ich froh, daß ich die Information im Buch belassen habe. Ich bin immer noch versucht, vom Einsatz dieser Technologie abzuraten, aber nichtsdestotrotz ist sie interessant, und es geht nichts über eine genauere Betrachtung eines möglichen clientseitigen Ansatzes, um die Abwägungen für den serverseitigen Ansatz zu verstehen, der in Kapitel 28 beschrieben wird.