Kapitel 23

ActiveX-Dokumente – Grundlagen

23.1 Was ist ein ActiveX-Dokument? 732
23.2 Programmierung von ActiveX-Dokumenten 737
23.3 Das UserDocument-Objekt 742

Vom Programmieraspekt her gesehen, sind ActiveX-Dokumente und ActiveX-Steuerelemente fast gleich. Die größte Herausforderung, der Sie gegenüberstehen, ist, genau zu verstehen, was ActiveX-Dokumente sind und wie sie sich von Steuerelementen unterscheiden.

Wir beginnen dieses Kapitel also mit der Philosophie von ActiveX-Dokumenten, wobei wir davon ausgehen, daß die wahrscheinlichste Verwendung dieses Komponententyps auf Web-Sites im Internet und in Intranets ist. Darauf folgt ein erstaunlicher kurzer Abschnitt über die Programmierung von ActiveX-Dokumenten – kurz, weil Sie aus unseren Beschreibungen anderer Komponententypen bereits fast alles wissen, was Sie brauchen.

Was ist, wenn Sie direkt in dieses Kapitel geblättert haben, weil Sie nur ActiveX-Dokumente entwickeln möchten, und Ihre Zeit nicht damit vergeuden wollen, zuvor ActiveX-Codekomponenten und Steuerelemente zu schreiben? Sorry, dann werde ich Sie wirklich stark verwirren, weil ich definitiv voraussetze, daß Sie zu diesem Zeitpunkt bereits das gesamte Buch gelesen haben.

23.1 Was ist ein ActiveX-Dokument?

Um ActiveX-Dokumente wirklich verstehen zu können, müssen Sie noch einmal in Kapitel 2 zurückblättern und sich unsere Diskussion über das Konzept der dokumentationsbezogenen Programmierung verinnerlichen. In diesem Ansatz unterscheiden wir zwischen einem Dokument, wie beispielsweise einem Word-Dokument oder einem Videoclip, und der Applikation, die das Dokument bedient (Öffnen, Bearbeiten, Anzeigen, Speichern). Wenn Sie über ActiveX-Dokumente sprechen, dann geht es letztlich um zwei unterschiedliche Dinge: das eigentliche Dokument und den ActiveX-DLL- oder -EXE-Server, der es unterstützt.

Wenn Sie ein neues ActiveX-Dokument-Projekt erstellen, erstellen Sie eigentlich diese beiden Elemente: den Server und ein leeres Dokument. Ein einziger Server kann beliebig viele Dokumente unterstützen, so wie Microsoft Word beliebig viele Dokumentdateien unterstützt. Der Server für ein ActiveX-Dokument kann ein EXE- oder ein DLL-Server sein. Das Dokument hat im allgemeinen die Erweiterung `.VBD`, aber ActiveX-Dokumente können auch innerhalb anderer Dateien abgelegt werden, wozu ein Mechanismus verwendet wird, der auch als OLE-strukturiertes Speichern bezeichnet wird.

Diese Unterteilung zwischen Server und Dokument scheint vielleicht offensichtlich, aber ich finde, daß man schnell die Übersicht verlieren kann, wenn man in der Visual-Basic-Dokumentation über ActiveX-Dokumente liest. Tabelle 23.1 zeigt die Ähnlichkeiten zwischen Word-Dokumenten und ActiveX-Dokumenten.

Wenn Sie einen neuen ActiveX-Dokument-Server kompilieren oder ausführen, erzeugt Visual Basic eine leere `.VBD`-Datei, für die keine Eigenschaften gesetzt sind. Wenn Sie eine leere `.VBD`-Datei laden, wird das `InitProperties`-Ereignis des `UserDocument` aufgerufen. Wenn Sie das Objekt speichern, schreibt das

Operation	Microsoft Word	ActiveX-Dokument
Ein Dokument öffnen	Doppelklick auf eine Dokumentdatei.	Doppelklick auf eine HTML-Seite, die auf eine .VBD-Datei verweist, oder Öffnen der .VBD-Datei mit dem Microsoft Explorer.
Der Server ist gefunden	Windows prüft die Registrierung und stellt fest, daß Dateien mit der Erweiterung .DOC mit Word geöffnet werden.	OLE sucht in der .VBD-Datei nach der CLSID des Servers.
Das Dokument wird geöffnet	Word wird geladen und öffnet die .DOC-Datei. Der Inhalt des Dokuments wird geladen.	Der ActiveX-Dokument-Server wird geladen und öffnet die .VBD-Dokumentdatei. Ein ActiveX-Dokument wird erzeugt, das das Dokument lädt, indem es die Eigenschaften aus der Datei liest.
Das Dokument wird angezeigt	Word zeigt die .DOC-Datei an.	Der Browser alloziert ein Fenster, in dem das ActiveX-Dokument-Objekt plaziert wird. Der Server zeigt dieses Objekt an.
Das Dokument wird bearbeitet	Sie können die gesamte Funktionalität von Word nutzen, um die Daten für das Dokument zu manipulieren.	Sie können die gesamte Funktionalität nutzen, die im Server programmiert wurde, um die Daten für das Objekt zu manipulieren.
Der Server wird geschlossen	Word fordert Sie auf, das veränderte Dokument gegebenenfalls in der .DOC-Datei zu speichern.	Der Explorer fordert Sie auf, das veränderte Dokument gegebenenfalls zu speichern. Wenn Sie zustimmen, speichert das Objekt die Dokumentdaten, indem es Eigenschaften in die .VBD-Datei schreibt (oder in eine separate Datei, falls es nicht in die aktuelle Dokumentdatei schreiben kann.

Tab. 23.1: Vergleich zwischen Microsoft Word und einem ActiveX-Dokument

Objekt seine Eigenschaften während des `WriteProperties`-Ereignisses des `UserDocument` in die `.VBD`-Datei. Wenn Sie jetzt die `.VBD`-Datei öffnen, wird ein `ReadProperties`-Ereignis ausgelöst, um den Server aufzufordern, die Eigenschaften aus der `.VBD`-Datei zu lesen. Hört sich genau wie bei einem ActiveX-Steuerelement an, oder?

23.1.1 ActiveX-Dokumente und ActiveX-Steuerelemente

Obwohl ActiveX-Dokumente und ActiveX-Steuerelemente sehr ähnlich in Hinblick auf die Programmierung sind, gibt es wesentliche Unterschiede in ihrer Arbeitsweise und ihrer Verwendung.

Tabelle 23.2 vergleicht ActiveX-Steuerelemente und ActiveX-Dokumente. Die Tabelle beschränkt sich auf einen Vergleich der Anwendung der beiden Technologien auf Web-Seiten, weil wir in der Einleitung dieses Kapitels festgestellt haben, daß das Internet und Intranets die wahrscheinlichsten Einsatzbereiche für ActiveX-Dokumente sind.

Operation	ActiveX-Steuerelement	ActiveX-Dokument
Die Komponente suchen	Das Steuerelement wird durch ein Objekt-Tag in einer HTML-Datei definiert.	Der Dokument-Server DLL oder EXE wird durch ein Objekt-Tag in einer HTML-Datei definiert.
Herunterladen des Servers	Der Browser lädt das Steuerelement oder die `.CAB`-Datei herunter. Die Datei kann anhand der Codesignatur identifiziert werden.	Der Browser lädt den Dokumentserver DLL oder EXE herunter, oder die `.CAB`-Datei. Die Datei kann anhand der Codesignatur identifiziert werden.
Initialisierung des Server-Objekts	Eigenschaften werden durch PARAM-Befehle in der HTML-Datei definiert. Das Steuerelement befindet sich auf dem Container im Ausführungsmodus und muß deshalb niemals Eigenschaften schreiben. Jede Objektreferenz erzeugt eine Instanz des Steuerelements auf dem Container.	Der angeforderte Server und die aktuellen Eigenschaftseinstellungen werden aus einer `.VBD`-Dokumentdatei geladen. Die Objektreferenz lädt den Server DLL oder EXE, aber erzeugt keine Instanz des `Document`-Objekts. `Document`-Objekte werden nur erzeugt, wenn Verweise auf eine `.VBD`-Dokumentdatei erfolgen.

Tab. 23.2: Vergleich von ActiveX-Steuerelementen und ActiveX-Dokumenten auf einer Web-Seite

Operation	ActiveX-Steuerelement	ActiveX-Dokument
Einrichtung des Servers	Das `Object`-Referenzobjekt definiert eine Position für das Steuerelement auf der aktuellen Web-Seite.	Das ActiveX-Dokument stellt man sich am besten als eigene Web-Seite vor. Sie können einen Link darauf anlegen wie auf jede andere Seite. Sie können auf einer Seite nicht HTML und ein ActiveX-Dokument kombinieren. Ein ActiveX-Dokument kann jedoch in einen Browser-Frame geladen werden. Jeder Frame enthält eine unabhängige Web-Seite.
Ändern der `Object`-Eigenschaften	Sie können die Steuerelementeigenschaften durch Interaktion mit dem Benutzer oder ein HTML-Skript ändern. Änderungen werden nicht gespeichert, es sei denn, Sie stellen einen Mechanismus bereit, um das selbst zu tun – und niemals durch das WriteProperties-Ereignis. Das Steuerelement kann die HTML-Seite, die es geladen hat, nicht neu schreiben.	Sie können die Eigenschaften des ActiveX-Dokuments durch Interaktion mit dem Benutzer ändern, oder, abhängig vom Container, durch Skripting. Der Browser gibt Ihnen die Möglichkeit, die Änderungen zu speichern, bevor Sie die Seite verlassen. Wenn Sie zustimmen, wird eine `.VBD`-Datei mit den neuen Eigenschaften geschrieben.

Tab. 23.2: Vergleich von ActiveX-Steuerelementen und ActiveX-Dokumenten auf einer Web-Seite

Damit gelangen wir zu einer anderen Sichtweise auf ActiveX-Dokumente.

Stellen Sie sich Ihren Internet-Browser als Fenster auf das Internet oder ein Intranet vor. Wenn er eine Web-Seite anzeigt, interpretiert er eigentlich eine einfache Text-Markup-Sprache, HTML. Ein komplexer Browser, etwa der Internet Explorer von Microsoft oder Netscape's Navigator, kann Skripts ausführen, die auf der HTML-Seite enthalten sind. Diese Skripts können in JavaScript oder in VBScript vorliegen, aber es spricht nichts gegen die Definition anderer Skripting-Sprachen.

Ein Internet-Browserfenster kann jedoch auch andere Dinge anzeigen. Es könnte in der Lage sein, ein Verzeichnis anzuzeigen, wenn Sie eine FTP-Site besuchen, oder vielleicht ein Bild, wenn Sie es auffordern, eine Bitmap-Datei aufzunehmen.

Microsoft's Internet Explorer kann ActiveX-Dokumente anzeigen. Anders als ein ActiveX-Steuerelement, das als Objekt auf einer HTML-Seite erscheint, ist ein ActiveX-Dokument ein völlig unabhängiger Objekttyp, den der Browser anstelle einer Web-Seite anzeigen kann. Das führt uns zu einer erschütternden Möglichkeit: Weil ActiveX-Dokumente das Hyperlink-Objekt unterstützen, das die Navigation zwischen Seiten erlaubt, ist es theoretisch möglich, eine Web-Site zu entwickeln, die nur ActiveX-Dokumente enthält, ohne irgendwelchen HTML-Code (anders als vielleicht eine einzige Ausgangsseite zum Herunterladen der Server-Komponenten).

Jetzt berücksichtigen Sie neben dieser Tatsache, daß ein ActiveX-Dokument fast dieselben Möglichkeiten bietet wie Visual Basic, auch die Fähigkeit, zusätzliche Formulare anzuzeigen, und daß das UserDocument-Objekt und die Formulare des Dokuments zusätzliche Steuerelemente enthalten können. Mit anderen Worten, ein ActiveX-Dokument ist eine Visual-Basic-Applikation. Der logische Schluß ist unvermeidbar: ActiveX-Dokumente erlauben, eine Site zu entwickeln, bei der es sich um eine Applikation handelt, die auf dem Client-System läuft.

Hmmm. Vielleicht hatte Carl doch recht mit seiner Einschätzung des Potentials der ActiveX-Dokumente. Andererseits könnte eine Site, die auf diese Weise entwickelt wird, zum Zeitpunkt der Drucklegung dieses Buchs nur Benutzer unterstützen, die den Microsoft Internet Explorer 3.01 oder neuer unter Windows 95/98 oder Windows NT 4 installiert haben. Damit gibt es mehrere mögliche Strategien für den Einsatz dieser Technologie:

- Sie können speziell für diese Leute eine Site erzeugen. Dieser Ansatz ist gut geeignet für Microsoft-spezifische Sites, wo Sie keine Rücksicht auf Leute nehmen, die andere Werkzeuge verwenden, oder für Firmen-Intranets, wo Sie die Client-Umgebung selbst festlegen können.
- Sie könnten eine Site entwickeln, in der Hoffnung, daß ActiveX-Dokumente irgendwann von mehr Betriebssystemen und mehr Browsern unterstützt werden, und mit der Erwartung, daß Ihr Zielpublikum so bald wie möglich einen Upgrade für die erforderlichen Werkzeuge durchführt.
- Sie könnten eine Site entwickeln, in der Erwartung, daß es nicht mehr lange dauert, bis jeder Benutzer den Microsoft Internet Explorer unter Windows 95/98 oder NT 4 einsetzt. (Ich glaube, das ist die Perspektive, die Microsoft am besten gefallen würde.)
- Sie könnten eine Hybrid-Site anlegen, wo Scripting-Code auf der HTML-Seite den verwendeten Browser erkennt und den Benutzer an die ActiveX-Dokument-Seiten weiterleitet, wenn dieser die richtigen Werkzeuge einsetzt. Das wäre eine gute Lösung, aber sie bedingt, daß Sie zwei Sites unterstützen – die eine auf den ActiveX-Dokumenten basierend, die andere auf anderen Technologien.

- Sie können Abstand von ActiveX-Dokumenten nehmen und Technologien einsetzen, die möglicherweise weniger leistungsfähig sind, aber dafür auf jedem möglichen System eingesetzt werden können. (Ich glaube, das ist der Ansatz, der Netscape am besten gefallen würde.)
- Und natürlich können Sie die oben aufgeführten Möglichkeiten auch nach Belieben kombinieren.

Sie können natürlich auch nach alternativen Ansätzen suchen, die ähnlich sind. Beispielsweise könnten Sie darauf setzen, daß DHTML-Komponenten erfolgreicher sein werden als ActiveX-Dokumente. DHTML-Komponenten sind ActiveX-Dokumenten vergleichbar, weil sie ebenfalls aus client-seitigen Programmen bestehen. Sie lesen einfach statt VBD-Dateien HTML. Und natürlich verwenden Sie ein völlig anderes Objektmodell für die Programmierung.

Hmmm. Vielleicht ist meine Skepsis bezüglich der Bedeutung von ActiveX-Dokumenten doch berechtigt.

Sie erkennen vielleicht, warum dieses Kapitel wenig gute Ratschläge enthält. Ich sehe die Vorteile von ActiveX-Dokumenten. Ich erkenne ihr Potential. Ich sehe, daß sie eine völlig coole Technologie darstellen, die das Potential hat, das Internet zu revolutionieren, so wie die Erfindung von HTML. Mehr und mehr erweisen sie sich jedoch als Konzept, das nie weitere Akzeptanz finden wird, und das im Vergleich zu konkurrierenden Technologien keine Rolle spielt.

Ich überlasse also die Entscheidung für oder gegen diese Technologie völlig Ihnen, während ich Ihnen zeige, wie alles aufgebaut wird.

23.2 Programmierung von ActiveX-Dokumenten

Sie wissen bereits 99 Prozent von allem, was Sie wissen müssen, um ActiveX-Dokumente zu erzeugen. Hier folgt ohne weitere Kommentare das fehlende Prozent.

23.2.1 Sollten Sie einen DLL- oder einen EXE-Server verwenden?

Sie kennen die Unterschiede zwischen den beiden Ansätzen bereits sehr gut. Alle Themen, die in diesem Buch bisher angesprochen wurden, gelten auch für die Anwendung auf ActiveX-Dokumente.

DLL-Server sind in der Regel effizienter, weil sie im Prozeßraum des Browsers ausgeführt werden. EXE-Server laufen in ihrem eigenen Thread, wodurch einige Hintergrundoperationen einfacher zu implementieren sind. Ein EXE-Server kann so konfiguriert werden, daß er als Standalone-Programm läuft (obwohl Sie nie in der Lage sein werden, das UserDocument zu nutzen, können Sie während der Sub Main-Prozedur Formulare laden und anzeigen). Ein einziger EXE-Server unterstützt alle Benutzerdokumente eines bestimmten Typs, die geladen werden.

Ich persönlich muß erst noch in die Situation geraten, wo ich einen EXE-Server für ein ActiveX-Dokument brauche. Der wahrscheinlichste Fall wäre, wenn Sie den Zugriff auf eine gemeinsam genutzte Ressource steuern möchten, wie es beispielsweise bei dem StockQuote-Server in Kapitel 15 der Fall war.

23.2.2 Anzeige von ActiveX-Dokumenten

ActiveX-Dokumente unterscheiden sich von Formularen und Steuerelementen in der Weise, wie sie angezeigt werden.

Als Programmierer definieren Sie die Größe Ihres Dokuments. Sie haben keine Kontrolle über den Teil des Bildschirms, der für die Anzeige Ihres Dokuments reserviert wird, aber Sie haben alle Informationen über den verfügbaren Platz. Ist der Platz kleiner als die Dokumentgröße, ermöglicht der Container, einen Teil des Dokuments in einem Viewport anzuzeigen (siehe Abbildung 23.1).

Die Eigenschaften `ScaleLeft`, `ScaleTop`, `ScaleWidth` und `ScaleHeight` verhalten sich wie bei Formularen und ActiveX-Steuerelementen. Sie reflektieren die Koordinaten des Steuerelements aus Sicht des Programmierers. Der Viewport definiert den Bereich, der tatsächlich angezeigt wird. Die Eigenschaften `ViewportLeft` und `ViewportTop` definieren die Position innerhalb des Dokumentbereichs, die in der oberen linken Ecke des Viewport-Bereichs erscheint. Die Eigenschaften `ViewportWidth` und `ViewportHeight` definieren die Breite und die Höhe des Viewports.

Die `SetViewport`-Methode von `UserDocument` kann genutzt werden, um die Anzeigeposition des Dokuments innerhalb des Viewports festzulegen. Damit wiederum werden die Eigenschaften `ViewportLeft` und `ViewportTop` gesetzt.

Das `Resize`-Ereignis von `UserDocument` wird immer dann ausgelöst, wenn sich die Größe des Containers für das Dokument ändert. Es tritt auch auf, wenn der Container größer ist, als durch die aktuellen Einstellungen von `ScaleWidth` und `ScaleHeight` vorgegeben.

Wenn der Container schmaler ist als in der `MinWidth`-Eigenschaft von `UserDocument` angegeben, und die `ScrollBars`-Eigenschaft von `UserDocument` horizontale Bildlaufleisten festlegt, erscheinen um das Container-Fenster herum Bildlaufleisten. Damit ist es möglich, zu der horizontalen Position des Dokuments im Viewport zu scrollen. Wenn der Container niedriger als die `MinHeight`-Eigenschaft von `UserDocument` ist, und die `ScrollBars`-Eigenschaft von `UserDocument` vertikale Bildlaufleisten vorgibt, werden um das Containerfenster herum Bildlaufleisten angezeigt. Sie erlauben Ihnen, zur vertikalen Position des Dokuments im Viewport zu scrollen. `HScrollSmallChange` und `VScrollSmallChange` geben die Distanz an, um die das Dokumentfenster gescrollt wird, wenn der Benutzer auf eine der Bildlaufleisten klickt.

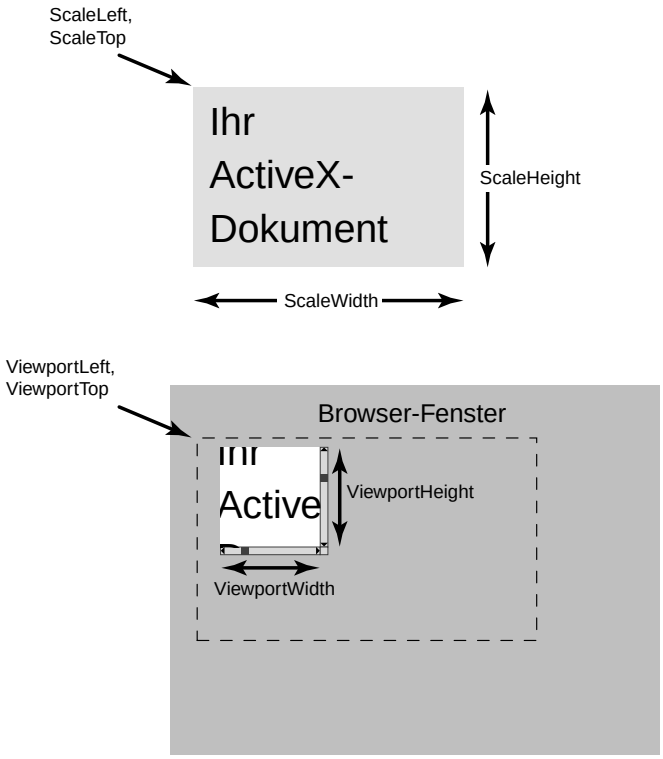

Abb. 23.1: Viewports für ein ActiveX-Dokument

Das `Scroll`-Ereignis wird ausgelöst, wenn der Benutzer das Dokument im Container scrollt. Wenn die `ContinuousScroll`-Eigenschaft `True` ist, werden `Scroll`-Ereignisse ausgelöst, sobald der Benutzer den Schieberegler der Bildlaufleiste verstellt.

Das Projekt `scroll.vbp` demonstriert ein einfaches ActiveX-Dokument, das die Eigenschaftswerte des Viewports anzeigt und Ihnen erlaubt, mit dem Scrolling zu experimentieren, während sich die Größe des Containers ändert:

```
' Guide to the Perplexed
' Einfaches Scrolling-Beispiel
Option Explicit

Private m_ScrollCount&

Private Sub UpdateLabels()
   lblViewportLeft.Caption = ViewportLeft
   lblViewportTop.Caption = ViewportTop
   lblViewportWidth.Caption = ViewportWidth
```

```
    lblViewportHeight.Caption = ViewportHeight
    lblScrollCount.Caption = m_ScrollCount
End Sub

Private Sub UserDocument_Resize()
    Debug.Print "Resize"
End Sub

Private Sub UserDocument_Scroll()
    m_ScrollCount = m_ScrollCount + 1
    UpdateLabels
End Sub

Private Sub UserDocument_Show()
    UpdateLabels
End Sub
```

23.2.3 Der Lebenslauf eines ActiveX-Dokuments

Die Visual-Basic-Dokumentation beschreibt relativ detailliert den Lebenszyklus eines ActiveX-Dokuments. In gewisser Weise ist das müßig, weil die wichtigsten Eigenschaften (`Initialize`, `InitProperties`, `Show`, `ReadProperties`, `WriteProperties`, `Hide` und `Terminate`) genau so funktionieren wie bei ActiveX-Steuerelementen. Der größte Unterschied ist, daß der Container keine separaten Entwurfszeit- und Laufzeitmodi unterstützt.

Während das Verhalten des Microsoft Internet Explorer in der VB-Dokumentation als sehr interessant beschrieben wird, ist es auch größtenteils irrelevant. Das ist so, weil Sie nicht voraussetzen können, daß zukünftige Versionen des Browsers dieselbe Cache-Größe und dasselbe Verhalten aufweisen. Und Sie können nicht davon ausgehen, daß sich andere Container ähnlich verhalten. Entwerfen Sie also Ihre Steuerelemente unter den folgenden Voraussetzungen:

- Der Container kann Ihr Dokument unter Verwendung des `WriteProperties`-Ereignisses auffordern, sich selbst zu speichern, wenn er meint, es sei dazu an der Zeit. Allgemein ausgedrückt passiert das, wenn Sie von einer Seite weggehen, auf der das Dokument enthalten ist, und wenn die `PropertyChanged`-Methode von `UserDocument` aufgerufen wurde.
- Der Container kann von einer beliebigen Position aus Ihr Dokument ansteuern, und von Ihrem Dokument aus an jede beliebige Position weiterspringen. Das Dokument wird aus der `.VBD`-Datei neu geladen, wenn es von irgendwoher erreicht wird.
- Der Container kann Ihr Dokument jederzeit verbergen und neu anzeigen.

Wie Carl in seiner Einleitung darlegt, kann diese fehlende Kontrolle einer der wichtigsten Gründe sein, warum sich die ActiveX-Dokument-Technologie nicht besser durchsetzen konnte.

23.2.4 Versionsverwaltung

Kapitel 25 beschreibt die Versionsverwaltung und die Bewahrung der Abwärtskompatibilität, wenn Sie Upgrades für Ihr Steuerelement vornehmen. Dieser Aspekt ist in Hinblick auf ActiveX-Dokumente absolut kritisch, denn wenn Sie die Abwärtskompatibilität verlieren, funktionieren alle Ihre existierenden .VBD-Dateien, die von diesem Server unterstützt werden, nicht mehr.

23.2.5 Container

Visual Basic hat die Möglichkeit, ActiveX-Steuerelemente zu testen, wesentlich verbessert. Sie führen das Steuerelement einfach aus, und Visual Basic erzeugt eine leere VBD-Datei und zeigt sie im Internet Explorer an. Sie können ein ActiveX-Dokument auch laden, indem Sie die .VBD-Datei mit dem Browser-Befehl DATEI/ÖFFNEN öffnen.

Die Microsoft Sammelmappe hat eine eigene Metapher für die Arbeit mit ActiveX-Dokumenten. Dort werden sie als Abschnitte (Sections) bezeichnet, und die Eigenschaften werden in `.OBD`-Dateien (Office BinDer) abgelegt, statt in `.VBD`-Dateien. Weitere Informationen über die Verwendung von ActiveX-Dokumenten in diesem Container finden Sie in Ihrer Dokumentation zur Sammelmappe.

23.2.6 Menüs

ActiveX-Dokumente können ein Menü haben. Das Menü erscheint jedoch nicht separat auf der Dokumentseite. Statt dessen wird es in die Menüleiste der Container-Applikation eingegliedert. Um das zu erzielen, müssen Sie die Option `NegotiatePosition` auf einen Wert ungleich 0 setzen.

Wenn der Name des übergeordneten Menüs `&Hilfe` ist, dann erscheint das Menü als Untermenü im HILFE-Menü des Containers, zumindest im Internet Explorer.

23.2.7 Fehlerverarbeitung

Es ist unabdingbar, Laufzeitfehler in Ihrem Projekt aufzufangen. Wenn ein Laufzeitfehler auftritt, beendet Ihr Server die Ausführung. Der Browser gibt an, daß er keine Dokumente anzeigen kann, die von diesem Server verarbeitet werden, in der Regel, indem eine Schraffur über der Seite oder eine Fehlermeldung angezeigt wird. Der Browser lädt den Server nicht neu, bis er beendet wird, oder alle im Cache befindlichen Seiteninformationen entfernt werden. Damit ist es für die Clients schwierig zu erkennen, was falsch gelaufen ist.

23.3 Das UserDocument-Objekt

Das `UserDocument`-Objekt ist für die ActiveX-Dokument-Server das, was das `UserControl`-Objekt für ActiveX-Steuerelement ist. Es ist die Seele des Servers.

Und noch wichtiger ist, daß das `UserDocument`-Objekt fast genau wie das `UserControl`-Objekt arbeitet. Aus diesem Grund geht dieser kurze Abschnitt auch nur auf die Unterschiede zwischen den beiden ein.

23.3.1 Ambient- und Extender-Eigenschaften

Ein ActiveX-Steuerelement befindet sich in einem Container, der auch andere Steuerelemente und Objekte enthält. Das Steuerelement muß also in der Lage sein, auf die Umgebungseigenschaften des Containers zuzugreifen.

Ein ActiveX-Dokument übernimmt die gesamte Seite bzw. den Frame, in die bzw. den es geladen wurde. Ob das wirklich ein ausreichender Grund dafür ist, das Konzept der Umgebungseigenschaften zu verwerfen, kann ich nicht sagen. Jedenfalls hat das `UserDocument`-Objekt keine `AmbientProperties`-Eigenschaft und auch kein Ambient-Objekt, auf das es zugreifen könnte. Und es unterstützt auch kein `Extender`-Objekt, sehr wohl jedoch eine `Parent`-Eigenschaft.

23.3.2 Parent und andere Client-Eigenschaften

Das `UserDocument`-Objekt in einem ActiveX-Dokument-Server hat eine `Parent`-Eigenschaft, die Ihnen ermöglicht, auf Eigenschaften oder Methoden zuzugreifen, die der Container bereitstellt. Die Eigenschaft `UserDocument.Parent.Name` beispielsweise gibt den Namen des Containers zurück, etwa Microsoft Internet Explorer.

Das `Parent`-Objekt und andere Eigenschaften, die sich auf den Container beziehen, stehen nicht zur Verfügung, bis sich das Dokument auf der Site befindet. Die Eigenschaft ist gültig, nachdem das `Show`-Ereignis aufgetreten ist (vorausgesetzt, sie wird im betreffenden Container unterstützt), kann aber während der Ereignisse `InitProperties` und `ReadProperties` gültig sein, abhängig vom jeweiligen Container. Sie ist nicht gültig während des `Initialize`-Ereignisses.

Die `Parent`-Eigenschaft bietet Zugriff auf das Objektmodell des Containers. Das kann Ihren Server mit einer enormen Funktionalität ausstatten, abhängig vom Container. Die Besonderheiten einzelner Container können in diesem Buch nicht beschrieben werden. Das Objektmodell für den Microsoft Internet Explorer finden Sie im ActiveX SDK, der von Microsoft bereitgestellt wird.

Natürlich ist jeder Code, den Sie schreiben, und der das Objektmodell des Containers nutzt, im höchsten Maße von diesem abhängig.

23.3.3 Eigenschaftspersistenz

Die .VBD-Dokumentdatei, die mit Ihrem ActiveX-Dokument-Server erzeugt wird, ist zunächst leer. Sie enthält im wesentlichen nichts anderes als die Informationen, die für ein Standarddokument benötigt werden. Immer wenn dieses Dokument geladen wird, wird das InitProperties-Ereignis von UserDocument ausgelöst. Erst nachdem Informationen in das Dokument geschrieben wurden, wird das ReadProperties-Ereignis ausgelöst. Nachdem das passiert ist, wird das InitProperties-Ereignis nie wieder für die betreffende .VBD-Dokumentdatei ausgeführt.

Der Container versucht nur, Eigenschaften mit Hilfe des WriteProperties-Ereignisses zu schreiben, wenn die PropertyChanged-Methode für das UserDokument-Objekt aufgerufen wurde. Das unterscheidet sich von ActiveX-Steuerelementen, wo das InitProperties-Ereignis nur einmal auftritt, und zwar in der Entwicklungsumgebung.

Dokumente, die mit einem Browser von einer Web-Site heruntergeladen werden, können nicht direkt auf die Site zurückgesichert werden, und es ist nicht klar, welche Art Applikation davon Nutzen tragen sollte, daß ein Benutzer ein heruntergeladenes ActiveX-Dokument ändert und es lokal speichert. Damit kommen wir zu zwei sehr wahrscheinlichen Szenarios für Ihre Entwicklung von ActiveX-Dokumenten auf einer Web-Site:

- Der Ansatz mit dem leeren Dokument – Nirgends ist gesagt, daß ein ActiveX-Dokument persistente Informationen speichern muß. Das gilt insbesondere für Dokumente, die über das Internet heruntergeladen wurden. Der Dokument-Server ist schließlich ein voll funktionales Programm, und es gibt beliebig viele mögliche Applikationen, die überhaupt keine Eigenschaften verwenden wollen. Mit diesem Ansatz tritt bei jedem Laden des Dokuments das InitProperties-Ereignis auf.
- Der Ansatz mit dem vordefinierten Dokument – Bei diesem Ansatz stellen Sie Ihrem Server zwei verschiedene Betriebsmodi bereit. Der eine ist ein Bearbeitungsmodus, der ausgelöst wird, wenn das leere Dokument geladen wird, wie durch das InitProperties-Ereignis angezeigt. Der andere ist ein Standardmodus, der durch das ReadProperties-Ereignis angezeigt wird. Im Bearbeitungsmodus öffnen Sie die Datei lokal auf Ihrem System, bearbeiten die Eigenschaften und speichern die Änderungen, nachdem Sie fertig sind. Im Standardmodus kann das Dokument über einen Web-Server heruntergeladen werden. Der ActiveX-Server lädt Eigenschaften während des ReadProperties-Ereigniss und deaktiviert alle Optionen, die die Eigenschaftswerte ändern könnten. Dieser Ansatz wird im Beispiel in Kapitel 24 demonstriert.

Natürlich können Sie die beiden Ansätze beliebig kombinieren. Dabei sollten Sie unbedingt beachten, wie Ihr ActiveX-Dokument genutzt werden soll, und dann die Ereignisse InitProperties und ReadProperties entsprechend setzen.

23.3.4 Tips und Tricks

Hier folgt die Beschreibung einiger zusätzlicher Techniken, die Sie für ActiveX-Dokumente nutzen können.

Sie können die Eigenschaften `MinHeight` und `MinWidth` auf einen sehr großen Wert setzen, um damit zu erzwingen, daß immer Bildlaufleisten angezeigt werden.

Sie steuern das Erscheinungsbild des Dokuments! Angenommen, Sie brauchen ein Dokument mit 100 Textfeldern in einer vertikalen Spalte (das ist natürlich ein völlig hypothetischer Fall!). Sie könnten versuchen, ein einziges, riesiges Dokument zu erzeugen. Eine bessere Vorgehensweise wäre die Verwendung eines Steuerelementfelds, um nur so viele Textfelder zu erzeugen, daß der verfügbare Viewport damit gefüllt wird, und dann die Bildlaufleisten zu aktivieren. Wenn der Benutzer scrollt, können Sie programmtechnisch den Inhalt der sichtbaren Textfelder ändern, so daß der Eindruck entsteht, sie wären gescrollt worden. Anschließend können Sie mit Hilfe der `SetViewport`-Eigenschaft erzwingen, daß das Dokument korrekt im Fenster erscheint.

Ein ActiveX-Dokument kann in einem Browser-Frame aber auch in einem unabhängigen Fenster erscheinen. Damit ist es mit jedem Web-Designer möglich, den Navigationsteil Ihrer Site zu erzeugen (die Links zwischen den verschiedenen Seiten), statt Navigations-Links in dem Dokument-Server zu codieren. In Kapitel 24 erfahren Sie, wie das funktioniert.

Das `Show`-Ereignis tritt auf, nachdem sich das Dokument vollständig in dem Container befindet. Außerdem tritt es immer dann auf, wenn Sie zu der Seite zurückkehren, nachdem Sie zu einer anderen Seite weitergesprungen waren. Sie können die Performance Ihres Dokuments verbessern, indem Sie die nötige Initialisierung ausführen, wenn das `Show`-Ereignis zum ersten Mal aufgerufen wurde, und dann eine Modul- oder eine statische Variable setzen, um damit anzuzeigen, daß die Initialisierung abgeschlossen ist, so daß Sie sie in allen folgenden `Show`-Ereignissen nicht mehr wiederholen zu brauchen.

Visual Basic besitzt einen Assistenten, der Ihnen einen Großteil der Arbeit abnimmt, eine Standalone-Applikation in ein ActiveX-Dokument umzuwandeln.

Visual Basic 6 beinhaltet ein neues Utility, die sogenannte Codebase (die in den Paket- und Weitergabe-Assistenten eingebaut ist), die die Position des ActiveX-Dokument-Servers in eine VBD-Datei einbettet. Bei früheren Versionen des Internet Explorers war es erforderlich, eine HTML-Seite zu erstellen, die die Server-Komponente unter Verwendung des <`OBJECT`>-Tags und der `CODEBASE`-Attribut lud. Diese Seite wäre dann in die `.VBD`-Datei gesprungen. Der Internet Explorer 4 kann die `CODEBASE`-Information direkt aus der `.VBD`-Datei lesen, so daß die zusätzliche Seite nicht mehr nötig ist. Weitere Informationen finden Sie in der Online-Hilfe von Visual Basic 6 bei der Beschreibung der neuen Internet-Funktionen.

Mit unseren Erkenntnissen, daß ActiveX-Dokumente hauptsächlich auf Web-Sites verwendet werden, scheint klar zu sein, daß wir im nächsten Schritt genauer untersuchen müssen, wie ActiveX-Dokumente auf einer Internet- oder Intranet-Site eingesetzt werden können. Dies ist der Schritt, der die Grundlage für Kapitel 24 bildet, »ActiveX-Dokumente und das Internet«.