Kapitel 20

Eigenschaftsseiten und anderes

Ich muß Ihnen ein Geständnis machen. Ich mag die Eigenschaftsseiten. Ja, ich bearbeite meine Eigenschaften hauptsächlich im Eigenschaftenfenster von VB. Und ja, Microsoft empfiehlt, nicht zu viel auf den Eigenschaftsseiten zu tun, außer die Eigenschaften eines Steuerelements zu setzen. Aber ich kann mir nicht helfen – ich liebe dieses Konzept der Eigenschaftsseiten.

Sie sollten jetzt schon daran gewöhnt sein, daß Sie als Autor eines Steuerelements für die Laufzeit- als auch die Entwurfszeit-Umgebung des Containers schreiben. Das Verhalten des Steuerelements, das Sie für die Laufzeit definieren, ist für den Endbenutzer vorgesehen, also die Person, für die der Entwickler, der Ihr Steuerelement einsetzt, ein Programm schreibt. Aber das Verhalten, das Sie zur Entwurfszeit definieren, ist ausschließlich für den Entwickler vorgesehen. Im Steuerelement selbst ist nicht viel Platz für Kreativität und Flexibilität, insbesondere weil Visual Basic es kompliziert macht, zur Entwurfszeit mit dem Steuerelement zu interagieren. Sie können eine solche Interaktion erlauben, indem Sie die `EditAtDesignTime`-Eigenschaft auf `True` setzen, aber selbst dann ist es für die Entwickler noch schwierig, in den Bearbeitungsmodus eines Steuerelements zu wechseln und diesen wieder zu verlassen. Ich mache es selbst nur ganz selten.

Die Eigenschaftsseite dagegen wurde speziell dafür vorgesehen, dem Entwickler eine Interaktion mit dem Steuerelement zu ermöglichen, egal, welche Benutzeroberfläche Sie dafür definieren. Es ist fast so, als könnten Sie Ihr eigenes Programm erzeugen, um den Entwicklern die Konfiguration Ihres Steuerelements zu ermöglichen. Und das ist genau das, was die Eigenschaftsseiten bieten: Die Möglichkeit, Ihre eigene, eindeutige Entwicklungsumgebung für Ihr Steuerelement zu schaffen.

Denken Sie doch einmal darüber nach. Was bedeutet »eindeutige Entwicklungsumgebung für Ihr Steuerelement« eigentlich? Die VB-Dokumentation läßt grüßen, die die Eigenschaftsseiten als »eine Alternative zum Eigenschaftenfenster für die Anzeige der Eigenschaften von ActiveX-Steuerelementen« bezeichnet. Sie werden mich verstehen, wenn Sie das restliche Kapitel lesen, aber bevor Sie damit anfangen, möchte ich Ihnen noch einen Gedanken offenbaren: Es gibt keinen technischen Grund, der fordern würde, überhaupt Eigenschaftsseiten für die Eigenschaften Ihres Steuerelements anzulegen. Faszinierend, oder?

20.1 Eigenschaftsseiten – Grundlagen

Als erstes sollten Sie über Eigenschaftsseiten wissen, daß Sie sie für Ihre Steuerelemente immer implementieren sollten. Das hat einen einfachen Grund: Es gibt keine Garantie, daß ein Container, der Ihr Steuerelement einsetzt, ein eigenes Eigenschaftenfenster verwendet wie das in Visual Basic. Bei einem solchen Container stellen die Eigenschaftsseiten die einzige Möglichkeit dar, die Eigenschaften Ihres Steuerelements zu bearbeiten.

Als nächstes sollten Sie wissen, daß es Eigenschaftsseitenfenster in einem Eigenschaftsseiten-Containerfenster gibt. Dieses Fenster enthält die Schaltflächen OK,

ABBRECHEN und ÜBERNEHMEN, die von allen Eigenschaftsseiten verwendet werden. Das Eigenschaftsseiten-Contaienrfenster ist moduslos für das Steuerelement unter Visual Basic, aber das muß nicht für jeden Container zwingend der Fall sein.

Nun wollen wir einen Moment lang die Technik, die Schlüsseleigenschaften und die Ereignisse von Eigenschaftsseiten betrachten.

Jedes `UserControl`-Objekt hat eine `PropertyPages`-Auflistung. Wenn Sie im `PropertyPages`-Eintrag im VB-Eigenschaftenfenster auf die Dialogschaltfläche klicken, sehen Sie einen Dialog, ähnlich dem in Abbildung 20.1 gezeigten.

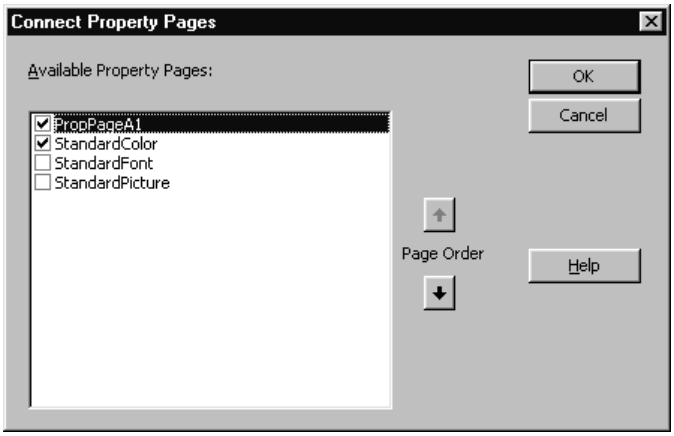

Abb. 20.1: Das Dialogfeld Eigenschaftsseiten.

Dieses Dialogfeld listet alle Eigenschaftsseiten auf, die Ihrem Steuerelement zur Verfügung stehen. Drei Standard-Eigenschaftsseiten stehen immer zur Verfügung: für die Standardfarbe, die Standardschrift und das Standardbild.

Der Container Ihres Steuerelements bietet immer eine Möglichkeit, die Eigenschaftsseiten für Ihr Steuerelement anzuzeigen. In Visual Basic verwenden Sie dazu den Befehl EIGENSCHAFTEN im Kontextmenü oder die Schaltfläche für die benutzerdefinierten Eigenschaften im Eigenschaftenfenster.

Visual Basic ordnet Eigenschaften automatisch Standard-Eigenschaftsseiten zu, wenn das möglich ist. Schrift-Eigenschaften werden den Standard-Schrift-Eigenschaftsseiten zugeordnet, Bild-Eigenschaften der Standard-Bild-Eigenschaftsseite, und die Eigenschaften mit dem Typ `OLE_COLOR` der Standard-Farb-Eigenschaftsseite. Aber diese Standardzuordnung bedeutet nicht, daß VB die Eigenschaftsseite automatisch auch der Dialogschaltfläche im VB-Eigenschaftenfenster zuordnet.

Wenn Sie möchten, daß der Dialogschaltfläche für die Eigenschaft im VB-Eigenschaftenfenster eine Eigenschaft zugeordnet wird, verwenden Sie EXTRAS/PRO-

ZEDURATTRIBUTE. Wählen Sie im Kombinationsfeld EIGENSCHAFTENKATALOG-SEITE aus den verfügbaren Seiten aus.

Was das bedeutet, kann recht verwirrend sein. Betrachten Sie etwa die `BackColor`-Eigenschaft im Beispiel `Trivial.vbp` für dieses Kapitel. Wenn die `BackColor` der Standard-Farb-Eigenschaftsseite nicht zugeordnet ist, wenn Sie auf die Schaltfläche im VB-Eigenschaftenfenster klicken, sehen Sie eine kleine Palette, die aufgeklappt wird. Wenn sie der Standard-Eigenschaftsseite jedoch zugeordnet wurde, sehen Sie statt dessen die Eigenschaftsseite. Welche sollten Sie verwenden? Das bleibt Ihnen überlassen, aber größtenteils weisen die Steuerelemente den Standard-Eigenschaftsseiten keine Steuerelementeigenschaften zu, wenn der Container eine Alternative in seinem Eigenschaftenfenster bereitstellt.

Die Zuordnung der Eigenschaftsseite wird besonders wichtig für Ihre eigenen Eigenschaften. Sie sollten Eigenschaften Seiten zuordnen, wenn Sie direkten Zugriff darauf bieten wollen.

Das in Listing 20.1 gezeigte Beispiel `Trivial.vbp` zeigt eine einfache Eigenschaftsseite, die eine einzige `String`-Eigenschaft für ein Steuerelement verarbeitet. Die Seite enthält ein einziges Textfeld, in dem die `Text`-Eigenschaft Test bearbeitet wird.

```
Option Explicit
     Private Sub txtTest_Change()
         Changed = True
     End Sub

     Private Sub PropertyPage_ApplyChanges()
         SelectedControls(0).Test = txtTest.Text
     End Sub

     Private Sub PropertyPage_SelectionChanged()
         txtTest.Text = SelectedControls(0).Test
     End Sub
```

Listing. 20.1: Die Eigenschaftsseite TrivialPg

Dieses Listing demonstriert die wichtigsten Ereignisse und Eigenschaften für eine Eigenschaftsseite. Die Auflistung `SelectedControls` enthält Verweise auf die Steuerelemente, denen die Eigenschaftsseite zugeordnet ist. Das `SelectionChanged`-Ereignis tritt immer dann auf, wenn der Seite ein oder mehrere Steuerelemente zugeordnet werden. Zu diesem Zeitpunkt wird das Text-Steuerelement mit dem Wert der `Text`-Eigenschaft des Steuerelements geladen.

Wenn der Entwickler den Wert des Textfelds ändert, wird das Ereignis `txtTest_Change` ausgelöst und die `Changed`-Eigenschaft der Eigenschaftsseite wird auf `True` gesetzt. Diese Eigenschaft erledigt zweierlei Dinge: Wenn sie auf `True` gesetzt ist, wird die Übernehmen-Schaltfläche für die Eigenschaftsseite

aktiviert. Außerdem löst die Eigenschaftsseite automatisch das `ApplyChanges`-Ereignis aus, wenn die OK-Schaltfläche angeklickt oder eine andere Eigenschaftsseite ausgewählt wird.

Wenn der Entwickler auf die ÜBERNEHMEN-Schaltfläche klickt oder die OK-Schaltfläche anklickt, während die `Changed`-Eigenschaft gleich `True` ist, wird das Ereignis `ApplyChanges` ausgelöst. Dieses Ereignis zeigt an, daß die Eigenschaftswerte in das Steuerelement geschrieben werden sollen. Nun wollen wir die Eigenschaften und Ereignisse des `PropertyPage`-Objekts genauer betrachten.

20.1.1 Eigenschaftsseiten-Eigenschaften

Wenn Sie das VB-Eigenschaftenfenster für ein `PropertyPage`-Objekt betrachten, sehen Sie, daß es fast alle Eigenschaften, Methoden und Ereignisse eines Standardformulars unterstützt. Es kann sowohl Standardelemente als auch ActiveX-Steuerelemente enthalten. Das bedeutet, Sie sind in der Programmierung einer Eigenschaftsseite so flexibel, wie Visual Basic das erlaubt. Die einzige Beschränkung ist, daß Ihnen wirklich ans Herz gelegt werden sollte, ist, Ihre Benutzeroberfläche innerhalb der Eigenschaftsseite beizubehalten. Sie können andere Formulare von der Eigenschaftsseite aus anzeigen, aber das weicht völlig von dem ab, was die Entwickler gewöhnt sind. Sie sollten das also nur dann tun, wenn es unumgänglich ist.

Es gibt mehrere Eigenschaften und Ereignisse, die es nur für Eigenschaftsseiten gibt. Sie sind im restlichen Abschnitt beschrieben.

Die `Changed`-Eigenschaft. Wenn diese Eigenschaft gleich `True` ist, wird die Übernehmen-Schaltfläche im Containerfenster der Eigenschaftsseite aktiviert. Außerdem löst das Anklicken der OK-Schaltfläche oder ein Wechsel auf eine andere Eigenschaftsseite automatisch das `ApplyChanges`-Ereignis der Eigenschaftsseite aus.

Wenn Sie diese Eigenschaft während des `ApplyChanges`-Ereignisses auf `True` setzen und das Ereignis ausgelöst wurde, wird die Seite nicht geschlossen.

Die `SelectedControls`-Eigenschaft. Die `SelectedControls`-Auflistung enthält zwei Eigenschaften. Die `Count`-Eigenschaft gibt die Anzahl der Steuerelemente an, die aktuell ausgewählt sind. Die `Item`-Eigenschaft, die gleichzeitig die Standardeigenschaft für das Objekt ist, enthält Verweise auf die ausgewählten Steuerelemente.

Wie wählen Sie mehr als ein Steuerelement für eine Eigenschaftsseite aus? In Visual Basic ist das möglich, indem Sie zuerst die Eigenschaftsseiten für ein Steuerelement anzeigen, und dann auf weitere Steuerelemente klicken, während Sie die Strg- oder die Shift-Taste gedrückt halten. Andere Container unterstützen vielleicht andere Techniken für die Auswahl mehrerer Steuerelemente.

Sie sollten herausfinden, wie Sie die Auswahl mehrerer Steuerelemente am besten verarbeiten, während Sie Ihre Eigenschaftsseite entwerfen. Sie sind viel-

leicht versucht, nur auf das erste Steuerelement in der Liste zuzugreifen, aber das könnte die Entwickler verwirren, gar nicht davon zu sprechen, daß sie dann nicht mehr die Möglichkeit hätten, Eigenschaftsänderungen für mehrere Steuerelemente gleichzeitig vorzunehmen. Ein Beispiel dafür, wie diese Situation verarbeitet wird, finden Sie später in diesem Kapitel.

Die `StandardSize`**-Eigenschaft**. Wenn Sie eine leere Eigenschaftsseite erzeugen, wird diese mit einer Größe von 395x233 Pixel angelegt. Das ist eine gebräuchliche Größe, die in Visual-Basic-Containern ausgezeichnet funktioniert. Es gibt zwei Standardgrößen, die mit dieser Eigenschaft gesetzt werden können, 375x179 und 375x101 Pixel. Ein gut vorbereiteter Container erweitert die Größe seines Eigenschaftsseitenfensters, um die von Ihnen gewählte Größe zu unterstützen. Wenn Sie also nicht die ganze Standardgröße brauchen, können Sie problemlos eine der kleineren Größen auswählen. Die Entwickler schätzen es sehr, wenn Steuerelement-Anbieter nicht unnütz Bildschirmplatz vergeuden. Die Microsoft-Dokumentation empfiehlt, diese Standardeinstellungen zu vermeiden, weil sie sich möglicherweise nicht korrekt an die Bildschirmauflösung auf dem System des Entwicklers anpassen.

20.1.2 Eigenschaftsseiten-Ereignisse

Eigenschaftsseiten unterstützen fast alle Ereignisse der Standardformulare. Sie unterstützen jedoch kein `Load`- oder `Unload`-Ereignis, weil diese ja auch recht wenig Sinn machen würden. Die Ereignisse `Initialize` und `Terminate` verhalten sich für Eigenschaftsseiten genau wie für Formulare. Mit Hilfe dieser beiden Ereignisse können Sie auf Steuerelemente auf den Eigenschaftsseiten zugreifen. Eine Eigenschaftsseite wird nicht zerstört, wenn Sie mit Hilfe der Registerkarten für das jeweilige Steuerelement zwischen den Eigenschaftsseiten wechseln. In der Regel wird sie jedoch zerstört, wenn der Eigenschaftsseiten-Container geschlossen wird, oder wenn Sie auf die Eigenschaftsseiten für ein anderes Steuerelement wechseln.

Das `SelectionChanged`**-Ereignis**. Dies ist eines von zwei wirklich sehr wichtigen Ereignissen für jede Eigenschaftsseite (das andere ist `ApplyChanges`). Dieses Ereignis wird immer dann ausgelöst, wenn ein Steuerelement über die Eigenschaftsseite zur Bearbeitung ausgewählt wird. Es sollte irgendwie wie das `Load`-Ereignis eines Formulars behandelt werden. Während dieses Ereignisses können Sie mit Hilfe der `SelectedControls`-Auflistung auf das zugehörige Steuerelement zugreifen und die aktuellen Werte für die Eigenschaften laden. Während dieses Ereignisses sollten Sie die Mehrfachauswahl von Steuerelementen verarbeiten.

Wenn die `PropertyChanged`-Methode für eine Eigenschaft des Steuerelements aufgerufen wird, wird dieses Ereignis ausgelöst. Das kann zu interessanten Nebeneffekten führen, wie Sie gleich noch sehen werden.

Um dieses Ereignis wirklich zu verstehen, müssen Sie erkennen, wann es nicht auftritt. Dieses Ereignis wird nicht ausgelöst, wenn Sie zwischen Eigenschaftsseiten wechseln, außer wenn eine Eigenschaft geändert wird. Außerdem können Sie die Ereignisse `GotFocus` oder `LostFocus` nicht verwenden, um den Wechsel zwischen den Seiten zu erkennen. Diese Ereignisse werden nicht ausgelöst, wenn Ihre Seite Steuerelemente enthält (und das ist in der Regel der Fall). Glücklicherweise ist dies ein relativ seltenes Problem. Aus Ihrer Perspektive ist der Wechsel auf eine andere Seite in etwa vergleichbar damit, als ob Ihre Seite vorübergehend durch ein anderes Fenster verdeckt würde.

Das `ApplyChanges`**-Ereignis**. Dieses Ereignis wird immer dann ausgelöst, wenn die Einstellungen auf der Eigenschaftsseite in das Steuerelement geschrieben werden müssen. Es tritt auf, wenn der Entwickler auf die ÜBERNEHMEN-Schaltfläche klickt, oder wenn die Changed-Eigenschaft `True` ist und der Entwickler auf die OK-Schaltfläche klickt, die Seite über das Systemmenü schließt oder eine andere Seite auswählt. Es wird nicht aufgerufen, wenn die Seite über die ABBRECHEN-Schaltfläche geschlossen wird.

Das `EditProperties`**-Ereignis**. Einige Container zeigen, wenn Sie auf eine einzelne Eigenschaft im Eigenschaftenfenster klicken, nicht alle verfügbaren Eigenschaftsseiten an, sondern genau die Eigenschaftsseite, die der Eigenschaft zugeordnet ist, auf deren Dialogschaltfläche Sie geklickt haben. In diesem Fall wird das Ereignis `EditProperties` ausgelöst, um anzuzeigen, welche Eigenschaft die Eigenschaftsseite aufgerufen hat. Eine typische Verwendung für dieses Ereignis ist, den Fokus auf das Steuerelement auf der Eigenschaftsseite zu setzen, die für die Bearbeitung der betreffenden Eigenschaft verantwortlich ist.

Sie sollten sich nicht auf dieses Ereignis verlassen. Ein Container hat die Möglichkeit, alle Eigenschaftsseiten anzuzeigen, die dem Steuerelement zugeordnet sind, wobei dieses Ereignis normalerweise nicht ausgelöst wird.

20.2 Eigenschaftsseiten – Techniken

Bisher haben Sie zwei Ansätze kennengelernt, Eigenschaftsseiten zu verwenden. Sie wissen, daß die Eigenschaften für ein Steuerelement auf Eigenschaftsseiten bearbeitet werden können. In Kapitel 19 haben Sie eine Eigenschaftsseite gesehen, die für die Bearbeitung einer komplexen Eigenschaft oder eines Objekts verwendet werden kann. In beiden Fällen folgten die Eigenschaftsseiten einem Standardansatz:

- Die Eigenschaftsseite lädt Eigenschaftswerte aus dem Steuerelement.
- Der Entwickler hat die Möglichkeit, die Eigenschaftswerte auf der Seite zu ändern.
- Der Entwickler kann die Zuweisung der geänderten Eigenschaftswerte an das Steuerelement übernehmen oder abbrechen.

Die Programmgruppe PrpPage1.vbg enthält zwei Projekte: Das Projekt PropPgs1.vbp enthält ein Steuerelement, PropPageCtlA, und eine Eigenschaftsseite, PropPageA1. Das Projekt PropPageTest1 enthält ein Formular, das zwei Instanzen des Steuerelements PropPageCtlA verwendet.

Das Steuerelement PropPageCtlA besitzt vier interessante Eigenschaften. Die Eigenschaften Label1Caption und Label2Caption werden in den Standardelementen Label1 und Label2 für das Steuerelement angezeigt. Das Standardelement Label3 demonstriert die Konfiguration des Steuerelements ohne die Verwendung von Eigenschaften. Die Eigenschaft ButtonVisible legt fest, ob das Standardelement Command1 sichtbar ist. Den Code für dieses Steuerelementmodul sehen Sie in Listing 20.2.

```
' Guide to the Perplexed
    ' PropPgs1 Beispiel aus Kapitel 20
    ' Copyright (c) 1997-1998 by Desaware Inc. All Rights Reserved

    Option Explicit

    'ACHTUNG! DIE AUSKOMMENTIERTEN ZEILEN NICHT LÖSCHEN ODER ÄNDERN!
    'MappingInfo=Label1,Label1,-1,Caption
    Public Property Get Label1Caption() As String
       Label1Caption = Label1.Caption
    End Property

    Public Property Let Label1Caption(ByVal New_Label1Caption As String)
       Label1.Caption() = New_Label1Caption
       PropertyChanged "Label1Caption"
    End Property

    'ACHTUNG! DIE AUSKOMMENTIERTEN ZEILEN NICHT LÖSCHEN ODER ÄNDERN!
    'MappingInfo=Label2,Label2,-1,Caption
    Public Property Get Label2Caption() As String
       Label2Caption = Label2.Caption
    End Property

    Public Property Let Label2Caption(ByVal New_Label2Caption As String)
       Label2.Caption() = New_Label2Caption
       PropertyChanged "Label2Caption"
    End Property

    'Eigenschaftswerte aus dem Speicher laden
    Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
       UserControl.BackColor = PropBag.ReadProperty("Backcolor", &H8000000F)
       Label1.Caption = PropBag.ReadProperty("Label1Caption", "Label1")
       Label2.Caption = PropBag.ReadProperty("Label2Caption", "Label2")
       Command1.Visible = PropBag.ReadProperty("ButtonVisible", True)
    End Sub
```

```
'Eigenschaftswerte in den Speicher schreiben
Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
   Call PropBag.WriteProperty("Backcolor", UserControl.BackColor, &H8000000F)
   Call PropBag.WriteProperty("Label1Caption", Label1.Caption, "Label1")
   Call PropBag.WriteProperty("Label2Caption", Label2.Caption, "Label2")
   Call PropBag.WriteProperty("ButtonVisible", Command1.Visible, True)
End Sub

Public Property Get BackColor() As OLE_COLOR
   BackColor = UserControl.BackColor
End Property

Public Property Let BackColor(ByVal vNewValue As OLE_COLOR)
   UserControl.BackColor = vNewValue
   PropertyChanged "BackColor"
End Property

Friend Property Get InternalLabel3() As Label
   Set InternalLabel3 = Label3
End Property

Public Property Get ButtonVisible() As Boolean
   ButtonVisible = Command1.Visible
End Property

Public Property Let ButtonVisible(ByVal vNewValue As Boolean)
   Command1.Visible = vNewValue
End Property

Public Sub About()
   frmAbout.Show vbModal
End Sub
```

Listing. 20.2: Das Steuerelement PropPageCtlA

Dieses Steuerelement hat mehrere interessante Funktionen. Es verwendet ein drittes Bezeichnungsfeld, `Label3`, das nicht durch eine öffentliche Eigenschaft bereitgestellt wird. Das Steuerelement enthält jedoch eine `Friend`-Funktion, die einen Verweis auf das `Label3`-Element bietet. Damit kann eine Eigenschaftsseite direkt auf das Bezeichnungsfeld zugreifen, weil es Teil desselben Projekts ist. Es wäre ein größerer Fehler, ein Standardobjekt öffentlich auf diese Weise bereitzustellen, weil damit jeder, der Ihr Steuerelement verwendet, Zugriff auf das Objekt hätte. Es ist jedoch sicher, das innerhalb Ihres eigenen Projekts so zu handhaben, weil Sie die volle Kontrolle darüber haben, was mit dem Objekt passiert.

20.2.1 Die Eigenschaftsseite PropPageA1

Die Seite `PropPageA1` demonstriert mehrere Techniken für die Arbeit mit Eigenschaftsseiten:

- Die Standardmethode, eine Eigenschaft zu bearbeiten, die für die `Label1Caption`-Eigenschaft angewendet wird.
- Die unmittelbare Aktualisierung bei der Bearbeitung einer Eigenschaft, die für die `Label2Caption`-Eigenschaft eingesetzt wird.
- Die Konfiguration eines Steuerelements ohne die Verwendung einer Eigenschaft, die genutzt wird, um das Standardelement `Label3` zu konfigurieren.
- Die Bearbeitung von Eigenschaften auf mehreren Seiten, wie es bei der `ButtonVisible`-Eigenschaft der Fall ist.
- Die Verarbeitung ungültiger Eigenschaftseinstellungen.

Listing 20.3 zeigt den Code für die Eigenschaftsseite `PropPageA1`. Eine detaillierte Erklärung folgt.

```
' Guide to perplexed:
    ' Beispiel für Eigenschaftsseiten, Kapitel 20
    ' Copyright (c) 1997-1998, by Desaware Inc.  All Rights Reserved
    Option Explicit

    Private Declare Function GetParent Lib "user32" (ByVal hwnd As Long) As Long
    Private Declare Function EnableWindow Lib "user32" (ByVal hwnd As Long, _
    ByVal fEnable As Long) As Long

    Private m_ReadInProgress As Boolean
    Private m_IgnoreSelectionChanged As Boolean
    Private m_SavedChangedState As Boolean
    Private m_LastControlBad As Boolean


    Private Sub PropertyPage_Initialize()
       Debug.Print "Property page initialized"
    End Sub

    Private Sub PropertyPage_Terminate()
       Debug.Print "Property page terminated"
    End Sub

    Private Sub PropertyPage_EditProperty(PropertyName As String)
       lblProp.Caption = "Editing property: " & PropertyName
       Select Case PropertyName
          Case "Label1Caption"
             txtLabel1Caption.SetFocus
```

```
            Case "Label2Caption"
               txtLabel2Caption.SetFocus
            Case "ButtonVisible"
               chkButtonVisible.SetFocus
         End Select
      End Sub

      ' Label2 wird sofort aktualisiert
      Private Sub txtLabel2Caption_Change()
         If Not m_ReadInProgress Then
            m_IgnoreSelectionChanged = True
            m_SavedChangedState = Changed
            SelectedControls(0).Label2Caption = txtLabel2Caption.Text
         End If
      End Sub

      Private Sub chkButtonVisible_Click()
         If Not m_ReadInProgress Then
            Changed = True
         End If
      End Sub

      Private Sub txtLabel1Caption_Change()
         If Not m_ReadInProgress Then
            Changed = True
         End If
      End Sub

Private Sub txtLabel1Caption_Validate(Cancel As Boolean)
   ' Sehen Sie die Probleme bei dieser Auswertung?
   If txtLabel1Caption.Text = "bad2" Then
      m_LastControlBad = True
      Cancel = True
      MsgBox "Bad property value for label1"
      ' Restore the original value
      txtLabel1Caption.Text = SelectedControls(0).Label1Caption
      txtLabel1Caption.SetFocus
   End If
End Sub


      Private Sub txtLabel3Caption_Change()
         If Not m_ReadInProgress Then
            Changed = True
         End If
      End Sub
```

```
     Private Sub PropertyPage_ApplyChanges()
        Dim ctl As PropPageCtlA
        If SelectedControls.Count = 1 Then
            m_LastControlBad = False
       On Error Resume Next
       ValidateControls
       On Error GoTo 0
       If m_LastControlBad Then
    Changed = True
Exit Sub
      End If


         If LCase$(txtLabel1Caption.Text) = "bad" Then
              MsgBox "Bad property value on Label1"
              txtLabel1Caption.SetFocus
              ' Ursprünglichen Wert wiederherstellen
              txtLabel1Caption.Text = SelectedControls(0).Label1Caption
              Changed = True
              Exit Sub
           End If
           SelectedControls(0).Label1Caption = txtLabel1Caption.Text
           SelectedControls(0).Label2Caption = txtLabel2Caption.Text
           Set ctl = SelectedControls(0)
           ' Frühe Bindung, um auf den Friend zuzugreifen
           ctl.InternalLabel3.Caption = txtLabel3Caption.Text
        End If
        For Each ctl In SelectedControls
           ctl.ButtonVisible = chkButtonVisible
        Next
     End Sub



     Private Sub PropertyPage_SelectionChanged()
        Dim ctl As PropPageCtlA

        If m_IgnoreSelectionChanged Then
           ' Ausgelöst durch die unmittelbare Aktualisierung der Eigenschaft, die
           ' PropertyChanged aufruft
           m_IgnoreSelectionChanged = True
           Changed = m_SavedChangedState
           Exit Sub
        End If

        m_ReadInProgress = True
```

```
    SetControlsVisibility   ' handles multiple selection
    If SelectedControls.Count = 1 Then
       txtLabel1Caption.Text = SelectedControls(0).Label1Caption
       ' Wert initialisieren
       txtLabel2Caption.Text = SelectedControls(0).Label2Caption
       Set ctl = SelectedControls(0)
       ' Go early bound to access friend
       txtLabel3Caption.Text = ctl.InternalLabel3.Caption
    End If
    ' Hier erfolgt die späte Bindung
    If SelectedControls(0).ButtonVisible Then
       chkButtonVisible.Value = 1
    Else
       chkButtonVisible.Value = 0
    End If

    m_ReadInProgress = False
End Sub


Private Sub SetControlsVisibility()
    Dim ctl As Object
    Dim EnableCtl As Boolean
    If SelectedControls.Count = 1 Then EnableCtl = True
    For Each ctl In PropertyPage.Controls
       If TypeOf ctl Is Label Or TypeOf ctl Is TextBox Then
          ' Wir könnten die Sichtbarkeit auch ändern
          ctl.Enabled = EnableCtl
       End If
    Next
End Sub

Private Sub cmdAbout_Click()
    Dim containerwnd&
    containerwnd = GetParent(PropertyPage.hwnd)
    containerwnd = GetParent(containerwnd)
    if containerwnd Then Call EnableWindow(containerwnd, False)
    frmAbout.Show vbModal
    if containerwnd Then Call EnableWindow(containerwnd, True)
End Sub
```

Listing. 20.3: Die Eigenschaftsseite PropPageA1

Die Ereignisse `Initialize` und `Terminate` haben `debug.print`-Anweisungen, die Ihnen ermöglichen, das Erzeugen und Zerstören der Eigenschaftsseite zu verfolgen.

Das erste wirklich interessante Ereignis ist SelectionChanged. Wir wollen das Flag m_IgnoreSelectionChanged für den Augenblick ignorieren. Als erstes setzt das SelectionChanged-Ereignis die Modulsvariable m_ReadInProgress auf True, um damit anzuzeigen, daß Eigenschaften geladen werden. Die Funktion lädt die Eigenschaftswerte und zeigt sie in den Textfeldern der Eigenschaftsseite an, die ihre Change-Ereignisse auslösen (oder Clicked-Ereignisse, wenn es sich um Kontrollkästchen handelt). Standardmäßig setzt das Change-Ereignis eines Steuerelements die Changed-Eigenschaft auf True, so daß die ÜBERNEHMEN-Schaltfläche aktiviert wird. Es ist jedoch nicht sinnvoll, die ÜBERNEHMEN-Schaltfläche zu aktivieren, wenn noch keine Änderungen stattgefunden haben. Letztlich könnte das den Entwickler sogar verwirren. Die Variable m_ReadInProgress wird also im Change-Ereignis für Textfelder bzw. Click-Ereignis für Kontrollkästchen verwendet, um zu verhindern, daß die Changed-Eigenschaft auf True gesetzt wird, wenn die entsprechenden Steuerelemente mit ihren Ausgangswerten geladen werden.

Verarbeitung der Mehrfachauswahl. Als nächstes kümmert sich der Code des SelectionChanged-Ereignisses um mögliche Mehrfachauswahlen. In diesem Beispiel wurde die Eigenschaftsseite so angelegt, daß sie nur für die Eigenschaft ButtonVisible eine Mehrfachauswahl verarbeitet. Die private Funktion SetControlVisibility prüft, ob mehrere Steuerelemente ausgewählt sind. Wenn das der Fall ist, verbirgt sie die Label- und Text-Steuerelemente, die nicht für Mehrfachauswahlen verwendet wurden.

Beachten Sie, daß dieses Beispiel nur einen sehr einfachen Ansatz für die Verarbeitung von Mehrfachauswahlen zeigt. Andere Ansätze könnten die folgenden Dinge beinhalten:

- Neupositionierung des ButtonVisible-Kontrollkästchens, um das Feld ausgeglichener erscheinen zu lassen.
- Warnungen, Statusanzeigen oder Listenfelder mit den ausgewählten Steuerelementen anzeigen.
- Bereitstellung eines völlig anderen Aussehens der Eigenschaftsseite bei Mehrfachauswahl.

Mit anderen Worten, Sie sind bei der Verarbeitung der Situation völlig flexibel. Eine interessante Anwendung finden Sie später im Beispielprojekt ProgPgT2.vbp.

Der Ereigniscode von SelectionChanged greift nicht auf Eigenschaften zu, die bei der Mehrfachauswahl nicht verwendet werden. Es würde nichts weiter passieren, aber es entsteht ein Overhead beim Zugriff auf die Eigenschaften eines Steuerelements, insbesondere von Eigenschaftsseiten, wo der Zugriff in der Regel spät gebunden ist. Die Vermeidung unnötiger Zugriffe auf Eigenschaften verbessert also die Performance auf Kosten einer einzigen If...Then-Anweisung.

Die ButtonVisible-Eigenschaft ist die einzige auf dieser Seite, die für Mehrfachauswahlen vorgesehen ist. Wie können Sie diese Situation verarbeiten, wenn mehrere Steuerelemente unterschiedliche Werte für diese Eigenschaft haben dürfen? Die chkButtonVisible-Eigenschaft wird abhängig vom Status des ersten Steuerelements in der SelectedControls-Auflistung geladen. Das ist eine zufällige Auswahl. D.h. dieses Beispiel erzwingt, daß alle Steuerelemente den Wert des ersten Steuerelements verwenden. Ein komplexerer Ansatz wäre, alle ButtonVisible-Eigenschaftswerte für alle Steuerelemente zu durchlaufen. Wenn sie alle gleich sind, setzen Sie das Kontrollkästchen auf den entsprechenden Wert. Wenn sich einer davon jedoch unterscheidet, könnten Sie das Kontrollkästchen ausgegraut darstellen, d.h. es gibt unterschiedliche Werte. Sie könnten die Logik für das Kontrollkästchen-Element dann so abändern, daß es von markiert in nicht markiert und dann ausgegraut wechselt. Wenn das ApplyChanges-Ereignis ausgelöst wird, setzen Sie die Steuerelementeigenschaften auf markiert oder nicht-markiert, oder lassen sie unverändert, falls das Kontrollkästchen ausgegraut ist. Dieser Ansatz ist in vielen Dialogfeldern gebräuchlich und wird hier dem Leser als Übung überlassen.

Die eigentliche Technik, wie die Eigenschaft im Steuerelement gesetzt wird, wird für das ApplyChanges-Ereignis gezeigt. Sie durchlaufen einfach alle Steuerelemente der SelectedControls-Auflistung und setzen die Eigenschaft auf den gewünschten Wert.

Unmittelbare Aktualisierung der Eigenschaften. Die Label1Caption-Eigenschaft arbeitet so, wie Sie es gewohnt sind. Sie bearbeiten die Eigenschaft auf der Seite, und wenn Sie fertig sind, können Sie die Änderungen auf das Steuerelement anwenden. Die Label2Caption-Eigenschaft dagegen wird unmittelbar aktualisiert, während Sie Text auf der Eigenschaftsseite eingeben (probieren Sie es aus!). Wie ist das möglich? Eigentlich ganz einfach. Die Control-Eigenschaft wird während des Ereignisses txtLabel2Caption_Change gesetzt.

Beachten Sie, daß die Changed-Eigenschaft in diesem Fall nicht auf True gesetzt wird. Und es ist auch nicht erforderlich, die Eigenschaft während des ApplyChanges-Ereignisses zu schreiben, weil die Aktualisierungen sofort stattfinden. Aber was, wenn mehrere Steuerelemente ausgewählt sind? Das ist eine Fangfrage, und Sie müssen sich keine Gedanken darüber machen, weil in diesem Beispiel das Text-Steuerelement verborgen wird, sobald mehrere ActiveX-Steuerelemente ausgewählt werden!

Wichtig bei diesem Ansatz ist, daß wenn die Label2Caption-Eigenschaft geändert wird, die Property Let-Prozedur die Funktion PropertyChanged aufruft. Diese Funktion bewirkt, daß das SelectionChanged-Ereignis der Eigenschaftsseite ausgelöst wird, das alle Eigenschaften neu lädt und das Changed-Flag löscht. Alle anstehenden Änderungen anderer Eigenschaften wären verloren. Um das zu verhindern, werden während des Ereignisses txtLabel2Caption_Change zwei Flags gesetzt. Das eine signalisiert, daß das nächste SelectionChanged-Ereignis ignoriert werden soll. Das andere enthält den aktuellen Wert der Changed-Eigen-

schaft. Wird das Flag `m_IgnoreSelectionChanged` während des `SelectionChanged`-Ereignisses gesetzt, ignoriert die Eigenschaftsseite das Ereignis und stellt den Wert der `Changed`-Eigenschaft wieder her.

Wie können Sie diese Probleme bei der unmittelbaren Aktualisierung verhindern? Eine Möglichkeit wäre, die `PropertyChanged`-Funktion im Steuerelement nicht aufzurufen, wenn die Eigenschaft von der Eigenschaftsseite aus gesetzt wird (um das zu erkennen, setzen Sie ein Flag). Ein anderer Ansatz wäre, die direkte und die verzögerte Aktualisierung von Eigenschaften nicht auf derselben Seite zu kombinieren. Das ist vielleicht die sauberste Lösung, und vermeidet, daß die Entwickler einen Eigenschaftstyp mit dem anderen verwechseln.

Dies ist wirklich eine sehr vereinfachte Darstellung einer ganzen Klasse von Eigenschaftsseiten-Lösungen, wo die Eigenschaftsseite unmittelbar mit dem Steuerelement interagieren kann. Es gibt Situationen, wo Sie möchten, daß das Steuerelement eine Operation ausführt oder aktualisiert wird, wenn der Entwickler Änderungen vornimmt, statt auf das `ApplyChanges`-Ereignis zu warten. Ein weiteres Beispiel dafür finden Sie im Projekt `PropPgT2.vbp` später in diesem Kapitel.

Konfiguration ohne Eigenschaften. Die Eigenschaft `Label3Caption` demonstriert noch eine weitere Technik, wie eine Eigenschaftsseite eine Konfiguration für ein Steuerelement vornimmt, ohne öffentliche Eigenschaften zu verwenden.

Wie Sie sich erinnern, stellt das Steuerelement `PropPageCtlA` eine nur lesbare `Friend`-Eigenschaft bereit, `InternalLabel3`, die einen Verweis auf das Standardelement `Label3` enthält. Der Zugriff auf diese `Friend`-Funktion ist etwas kompliziert. Sie können nicht direkt über die `SelectedControls`-Auflistung vorgehen, weil die Elemente in der Auflistung als `As Object` angesprochen werden, `Friend`-Funktionen aber nur über eine früh gebundene Schnittstelle zur Verfügung stehen. Glücklicherweise ist es ganz einfach, diese Schnittstelle zu erhalten, wie im folgenden Code gezeigt:

```
Dim ctl As PropPageCtlA
Set ctl = SelectedControls(0)
txtLabel3Caption.Text = ctl.InternalLabel3.Caption
```

Nachdem Sie die frühgebundene Schnittstelle haben, können Sie auf das `Label3`-Objekt und seine Eigenschaften zugreifen. Der Prozeß ist umgekehrt, wenn die Eigenschaft während des `ApplyChanges`-Ereignisses geschrieben wird.

Warum sollte jemand diesen einigermaßen seltsamen Ansatz nutzen, um ein Steuerelement zu konfigurieren? Schließlich ist es ganz einfach, eine Eigenschaft für den Zugriff zur Entwurfszeit zu konfigurieren, und dabei zu verhindern, daß sie im VB-Eigenschaftenfenster angezeigt wird (im Dialogfeld EXTRAS/PROZEDURATTRIBUTE).

Die Wahrheit ist, daß dieser Ansatz in unserem Fall reichlich blöd ist. Der zusätzliche Aufwand bietet keinen wirklichen Vorteil. Die Vorteile kommen in den folgenden Situationen zu Tage:

- Ihr Steuerelement enthält zusätzliche komplexe Objekte, deren Bearbeitung schwierig ist und nicht im VB-Eigenschaftenfenster erfolgen kann.
- Ihr Steuerelement muß sehr viele Eigenschaften bereitstellen, die zur Entwurfszeit nicht benutzt werden, und die das VB-Eigenschaftenfenster völlig unübersichtlich machen würden.
- Ihre Eigenschaftsseite muß komplexe Operationen direkt auf Standardelemente oder das `UserControl`-Objekt ausführen.

Im aktuellen Beispiel ist die Verwendung eines Standardelements nur zur Konfiguration der `Caption`-Eigenschaft unsinnig. Aber wollte ich eine Eigenschaftsseite erzeugen, die mehrere Eigenschaften des Bezeichnungsfelds anpaßt, wäre dieser Ansatz schon sehr viel passender.

Verarbeitung unzulässiger Eigenschaftswerte. Die `Label1Caption`-Eigenschaft scheint auf den ersten Blick eine völlig gebräuchliche Methode zu demonstrieren, die Eigenschaften auf einer Eigenschaftsseite zu verarbeiten. Auf den zweiten Blick sehen Sie, daß sie wirklich völlig normal ist. Um zu verhindern, in diesem Beispiel nur Platz zu vergeuden, war ich so frei, es dazu zu verwenden, Ihnen zu zeigen, wie unzulässige Einträge verarbeitet werden.

Es gibt mehrere verschiedene logische Fehler, die in diesem Steuerelement demonstriert werden. Der erste tritt auf, wenn Sie das Wort »bad« in das Textfeld für die `Label1Caption`-Eigenschaft eintragen. Diese Logik finden Sie im `ApplyChanges`-Ereignis, das die typischen Aktionen demonstriert, die Ihr Code in so einer Situation ergreifen sollte:

- Anzeige eines Meldungsfeld mit der Erklärung des Fehlers.
- Rückgabe des Fokus an das Steuerelement, das den Fehler enthält. Damit ist dem Entwickler schneller deutlich, welcher Wert korrigiert werden sollte.
- Wiederherstellung des alten Eigenschaftswerts. Das ist praktisch, wenn der Entwickler den vorherigen Wert vergessen hat. Ohne irgendeine Information, wie ein korrekter Wert aussehen könnte, halten Sie den Entwickler möglicherweise für immer auf Ihrer Eigenschaftsseite gefangen (wenigstes so lange, bis er an die ABBRECHEN-Schaltfläche denkt).
- Setzen Sie die `Changed`-Eigenschaft auf `True`. Daran erkennt Visual Basic, daß es die Eigenschaftsseite nicht schließen soll.

Es gibt noch eine Technik für die Auswertung. Statt den Eigenschaftswert zu überprüfen und innerhalb der Eigenschaftsseite auszuwerten, können Sie eine Fehlerverarbeitung aktivieren und versuchen, den Eigenschaftswert zu setzen – und sich darauf verlassen, daß das Steuerelement einen Fehler aufwirft, wenn der

Eigenschaftswert nicht zulässig ist. Wenn dieser Fehler erkannt wird, setzen Sie die Changed-Eigenschaft auf True und brechen die Aktualisierung ab. Der einzige Nachteil bei diesem Ansatz ist, daß eine einzige fehlerhafte Eigenschaft verhindern könnte, daß alle anderen, gültigen Eigenschaften für das Steuerelement gesetzt werden, abhängig von der Reihenfolge, in der Sie die Eigenschaften setzen, und von der Logik Ihres Codes. (Beispielsweise könnten Sie beim ersten Fehler abbrechen, oder es weiter versuchen, alle anderen Eigenschaften zu setzen.)

Eine dritte Technik sehen Sie, wenn Sie das Wort »bad2« in das Textfeld für die Label1-Überschrift eingeben. Dieser Ansatz verwendet das Validataion-Ereignis für die Überprüfung des Werts, bevor der Fokus an ein anderes Steuerelement auf der Eigenschaftsseite weitergegeben wird. Schlägt die Auswertung fehl, wird eine Meldung angezeigt, und der Fokus wird auf das betreffende Steuerelement zurückgesetzt. Das einzige Problem ist, daß das Validation-Ereignis nicht aufgeworfen wird, wenn Sie die Eigenschaftsseite verlassen, beispielsweise indem Sie auf die ÜBERNEHMEN-Schaltfläche klicken.

Um dieses Problem zu lösen, verwendet das ApplyChanges-Ereignis ein Flag auf Modulebene, m_LastControlBad, das anzeigt, ob das letzte Steuerelement gültig war. Wird die ValidateControl-Methode aufgerufen, wird das Validate-Ereignis für das letzte Steuerelement auf der Eigenschaftsseite aufgeworfen. Liegt ein Problem vor, wird das Steuerelement auf seinen vorherigen Wert zurückgesetzt, ein Meldungsfeld zeigt eine Meldung für den Benutzer an, der Fokus wird auf das Steuerelement zurückgesetzt, das Flag m_LastControlBad wird auf True gesetzt und der Cancel-Parameter des Ereignisses wird auf True gesetzt. Das dient auch dazu, einen Fehler aufzuwerfen, der in diesem Fall ignoriert wird. ApplyChanges erkennt, daß das Flag m_LastControlBad auf True gesetzt wurde, und verläßt die Funktion:

```
Private Sub txtLabel1Caption_Validate(Cancel As Boolean)
   ' Sehen Sie die Probleme bei der Auswertung?
   If txtLabel1Caption.Text = "bad2" Then
      m_LastControlBad = True
      Cancel = True
      MsgBox "Bad property value for label1"
      ' Ursprünglichen Wert wiederherstellen
      txtLabel1Caption.Text = SelectedControls(0).Label1Caption
      txtLabel1Caption.SetFocus
   End If
End Sub
```

Anzeige zusätzlicher Dialogfelder. Die INFO-Schaltfläche auf der Eigenschaftsseite zeigt, daß Sie von dort aus eine modale Eigenschaftsseite anzeigen können. Wenn Sie den Show-Befehl mit der Option vbModal verwenden, ist das Formular modal für Ihre Eigenschaftsseite. Es ist jedoch in der Regel nicht modal für das

Container-Fenster der Eigenschaftsseite, was zu dem eher unangenehmen Effekt führt, daß das Eigenschaftenfenster unter dem modalen Dialogfeld neu positioniert werden kann.

Die Microsoft-Dokumentation rät Ihnen ab, modale Formulare in Eigenschaftsseiten anzuzeigen. Ich stimme dem grundsätzlich zu, was die Verwendbarkeit betrifft, aber es stellt sich heraus, daß es eine einfache Methode gibt, das Verhalten des Containers zu verbessern, zumindest, wenn es sich dabei um Visual Basic handelt. Dies sehen Sie im `CmdAbout_Click`-Ereignis im Listing `PropPageA1`. Die Routine verwendet API-Funktionen, um den Handle des Fensters zwei Ebenen oberhalb Ihrer Eigenschaftsseiten zu erhalten (die erste Ebene ist das Registerkartenfenster, die nächste das eigentliche Container-Fenster). Anschließend deaktiviert sie das Fenster, und aktiviert es erst wieder, nachdem das modale Formular geschlossen wurde.

Sie sollten jedoch wissen, daß diese Technik nicht dokumentiert oder von Microsoft anerkannt ist, es gibt also keine Garantie, daß sie für jeden Container funktioniert, und auch nicht, daß sie in zukünftigen Versionen von Visual Basic noch eingesetzt werden kann. Wenn es jedoch nicht funktioniert, ist das auch nicht weiter schlimm.

Was immer Sie tun, zeigen Sie keine nicht-modalen Formulare von einer Eigenschaftsseite aus an (oder von einem ActiveX-Steuerelement oder einer anderen ActiveX-DLL-Komponente). Neben dem schlechten Programmierstil ist das eigentliche Problem, daß viele Container keine nicht-modalen Formulare unterstützen, die von ActiveX-DLLs aller Art erzeugt werden.

Noch ein paar Gedanken. Der Assistent für die Eigenschaftsseiten ist ebenfalls sehr praktische, insbesondere, wenn Sie seine Arbeitsweise genau verstanden haben. Der Assistent ist einfach in der Anwendung, und ermöglicht Ihnen, neue Seiten zu erzeugen, Seiten Eigenschaften zuzuordnen, und Seiten auszuwählen und zu gruppieren.

Der Assistent konzentriert sich jedoch auf die einfachste der Standardtechniken zur Verarbeitung von Eigenschaften. Sie sollten den erzeugten Code sorgfältig überprüfen, ob er wirklich Ihren Anforderungen entspricht, insbesondere, wenn die Mehrfachauswahl von Steuerelementen korrekt sein soll.

20.2.2 Das Projekt PropPgT2.vbp

Das Projekt `PropPgT2` demonstriert eine völlig andere Verwendung von Eigenschaftsseiten, der zeigt, wie sie genutzt werden können, um Entwurfswerkzeuge und Utilities zu erzeugen. Das Projekt enthält ein Formular, ein privates Steuerelement und eine Eigenschaftsseite, die vom Steuerelement verwendet wird. Der einzige Grund dafür, warum in diesem Beispiel ein privates Steuerelement verwendet wurde, ist die Bequemlichkeit – es hätte genauso gut mit einem Standalone-Steuerelement funktioniert. Listing 20.4 zeigt den Code für das Steuerelementmodul `PropCtlB`.

```
' Guide to the Perplexed
    ' Beispiel für die Verwendung von Eigenschaftsseiten - Kapitel 20
    ' Copyright (c) 1997 by Desaware Inc. All Rights Reserved
    Option Explicit

    Public Property Get Font() As Font
       Set Font = UserControl.Font
    End Property

    Public Property Let Font(ByVal vNewValue As Font)
       Set UserControl.Font = vNewValue
       PropertyChanged "Font"
    End Property

    Private Sub UserControl_InitProperties()
       Set UserControl.Font = Ambient.Font
    End Sub

    Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
       Set UserControl.Font = PropBag.ReadProperty("Font", Ambient.Font)
    End Sub

    Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
       Call PropBag.WriteProperty("Font", UserControl.Font, Ambient.Font)
    End Sub

    Private Sub UserControl_Resize()
       UserControl.Size Label1.Width, Label1.Height
    End Sub

    Friend Function Container() As Object
       Set Container = UserControl.Extender.Parent
    End Function
```

Listing. 20.4: Das Steuerelement PropCtlB

Wie Sie sehen, enthält das Steuerelement eine einzige öffentliche `Font`-Eigenschaft. Das Steuerelement enthält ein einziges Standardelement, ein Bezeichnungsfeld, das die Meldung »Aligner-Use Property Page to Align« ausgibt. Das Bezeichnungsfeld hat eine feste Größe, und das Steuerelement wird automatisch daran angepaßt. Das bedeutet, der Entwickler hat keine Möglichkeit, dem Steuerelement eine neue Größe zu geben.

Einzig komisch in diesem Steuerelement ist die Container-Funktion, die einen Verweis auf den Container des Steuerelements zurückgibt. Der Container, das Formular frmtest2.frm, besitzt keinen eigenen Code. Er enthält eine Instanz des Steuerelements `PropCtlB` sowie fünf zufällig positionierte Bezeichnungsfelder.

Die Eigenschaftsseite PropPg2A. Diese Eigenschaftsseite enthält ein Listenfeld und eine Schaltfläche. Listing 20.5 zeigt den Code für diese Eigenschaftsseite. Wie Sie sehen, erinnert sie kaum an eine typische Eigenschaftsseite. Das liegt zum einen daran, daß es gar keine Eigenschaften gibt!

```
' Guide to the Perplexed
    ' Ausrichtungs-Eigenschaftsseite - Kapitel 20
    ' Copyright (c) 1997, by Desaware Inc. All Rights Reserved
    Option Explicit

    Private m_LeftPosition As Long

    Private Sub cmdExecute_Click()
       AdjustLabels
    End Sub

    Private Sub PropertyPage_SelectionChanged()
       ' Vorherige Werte können beibehalten werden
       lstLabels.Clear
       If SelectedControls.Count > 1 Then
          lstLabels.Visible = False
          ' Statt dessen Warnung anzeigen
          lstWarning.Visible = True
       Else
          ' Das wäre Overkill (zumindest für VB)
          lstLabels.Visible = True
          ' Statt dessen Warnung anzeigen
          lstWarning.Visible = False
       End If

       Set lstLabels.Font = SelectedControls(0).Font

       LoadListBox

    End Sub


    Private Sub LoadListBox()
       Dim MyControl As PropCtlB
       Dim ContainerObj As Object
       Dim InternalControl As Object
       Set MyControl = SelectedControls(0)
       Set ContainerObj = MyControl.Container
       For Each InternalControl In ContainerObj.Controls
          If TypeOf InternalControl Is Label Then
             lstLabels.AddItem InternalControl.Name
          End If
```

```
   Next
End Sub

Private Sub AdjustLabels()
   Dim MyControl As PropCtlB
   Dim ContainerObj As Object
   Dim InternalControl As Object
   Dim lstidx As Long
   Set MyControl = SelectedControls(0)
   Set ContainerObj = MyControl.Container
   m_LeftPosition = 0
   For Each InternalControl In ContainerObj.Controls
      If TypeOf InternalControl Is Label Then
         For lstidx = 0 To lstLabels.ListCount - 1
            ' Wir haben ein Bezeichnungsfeld mit dem korrekten Namen gefunden
            If InternalControl.Name = lstLabels.List(lstidx) _
            And lstLabels.Selected(lstidx) Then
               AdjustThisLabel InternalControl
            End If
         Next
      End If
   Next
End Sub

' Note we QI for the Label IDispatch interface during the call
Private Sub AdjustThisLabel(lbl As Label)
   If m_LeftPosition = 0 Then
      m_LeftPosition = lbl.Left
   Else
      lbl.Left = m_LeftPosition
   End If
End Sub
```

Listing. 20.5: Eigenschaftsseite PropPg2A

Das `SelectionChanged`-Ereignis lädt keine Eigenschaften, weil es keine zu laden gibt. Als erstes löscht es das Listenfeld und prüft dann, ob mehrere Seiten ausgewählt sind. Diese Seite ist nur auf die Arbeit mit einem einzigen Steuerelement ausgelegt, wenn also mehrere Steuerelemente ausgewählt sind, verbirgt sie das Listenfeld und die Schaltfläche und zeigt ein Beschriftungsfeld mit einer Warnung an. (Dieses Beschriftungsfeld liegt zur Entwurfszeit hinter dem Textfeld. Wenn Sie es sehen möchten. laden Sie das Projekt und schieben das Listenfeld einfach weg.)

Das Listenfeld wird dann auf die Schrift des Steuerelements gesetzt. Beachten Sie hier folgendes: Diese Eigenschaftsseite bearbeitet die Schrift nicht; sie verwendet sie einfach. Sie können auf die `Font`-Eigenschaftsseite springen, um die Schrift

zu bearbeiten, und dann zurück auf diese Seite gehen, und Sie sehen, daß die Schrift im Listenfeld mit der Font-Eigenschaft des Steuerelements aktualisiert wurde.

Das demonstriert einen Mechanismus, wie Eigenschaftsseiten miteinander kommunizieren können.

Schließlich wird die private Funktion `LoadListBox` aufgerufen, um die Namen aller Bezeichnungsfelder aus dem Container des Steuerelements in das Listenfeld zu laden. Sie sehen das in Abbildung 20.2.

Hoffentlich erkennen Sie, was in der Funktion `LoadListBox` in diesem Codebeispiel falsch ist! Richtig – es gibt keine Fehlerprüfung. Ich habe die Tatsache genutzt, daß dieses Steuerelement unter Visual Basic getestet wurde, um Operationen auszuführen, die mit Sicherheit funktionieren. Falls Sie möchten, daß diese Routine auch in anderen Containern funktioniert (oder wenigstens das System nicht zum Absturz bringt), sollten Sie eine Fehlerprüfung einfügen.

Die `Changed`-Eigenschaft wird für diese Eigenschaftsseite nie auf `True` gesetzt. Das ApplyChanges-Ereignis wird überhaupt nicht verwendet. Diese Seite soll die Bezeichnungsfelder im Container des Steuerelements am linken Rand des ersten Steuerelements auf der Liste ausrichten.

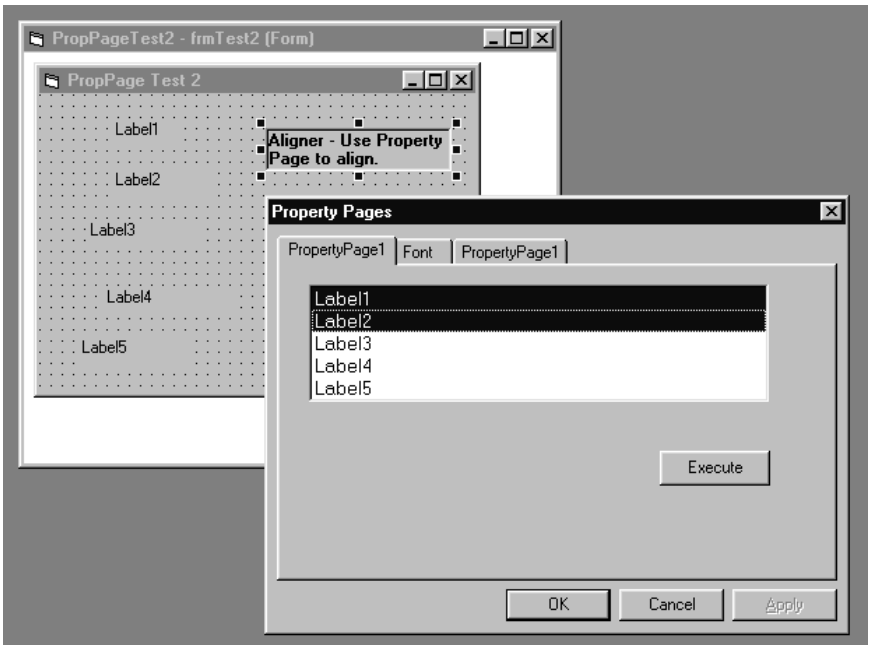

Abb. 20.2: Die Eigenschaftsseite PropPg2A zur Laufzeit.

Dazu wählt der Entwickler zuerst im Listenfeld die Steuerelemente aus, die ausgerichtet werden sollen. Diese Routine durchsucht alle Steuerelemente, die auf dem Container Ihres Steuerelements vorhanden sind. Ist das Steuerelement ein Bezeichnungsfeld und der Name stimmt mit dem in dem Listenfeld ausgewählten Namen überein, wird es links von dem ersten gefundenen Bezeichnungsfeld angeordnet.

Bevor wir fortfahren, betrachten Sie kurz die Eigenschaftsseite PropPg2B. Wie Sie sehen, enthält sie keinen Code. Sie erscheint jedoch trotzdem, weil sie dem Steuerelement über das Dialogfeld Eigenschaftsseiten von UserControl zugeordnet wurde.

Meine Meinung? Sie können fast alles mit Eigenschaftsseiten machen. Sie können den Entwicklern leistungsfähige Werkzeuge für die Konfiguration Ihres Steuerelements und möglicherweise sogar der Umgebung, in der es verwendet wird, an die Hand geben.

20.3 Info-Felder und anderes

Das Projekt `PropPgs1.vbp` zeigt auch, wie Ihrem Steuerelement ein Info-Feld hinzugefügt wird. Das ist einfach, indem man eine Methode einfügt, die ein modales Formular anzeigt. Die Methode muß im Dialogfeld EXTRAS/PROZEDURATTRIBUTE die Prozedur-ID AboutBox erhalten. Dadurch zeigt das VB-Eigenschaftenfenster zur Entwurfszeit eine About-Eigenschaft für das Steuerelement an. Die Prozedur wird immer aufgerufen, wenn die Schaltfläche für die About-Eigenschaft angeklickt wird. Die Prozedur kann einen beliebigen Namen erhalten. Wichtig ist nur, daß sie die korrekte Prozedur-ID besitzt.

Microsoft stellt eine AboutBox-Schablone bereit, die die Möglichkeit bietet, Systeminformationen anzuzeigen. Es handelt sich um ein sehr gutes Info-Feld, und die Option zur Anzeige der Systeminformation ist praktisch, ich sehe also keinen Grund, es nicht zu verwenden.

Wie bereits erwähnt, sollten Sie für Eigenschaftsseiten vermeiden, alle anderen Arten von Dialogfeldern (modale Formulare) anzuzeigen. Sie sollten definitiv alle nicht-modalen Formulare vermeiden, weil diese nicht von jedem Container unterstützt werden.

Damit schließen wir unsere Betrachtung der Eigenschaftsseiten und Dialogfelder in Hinblick auf ActiveX-Steuerelemente. Und entgegen all meiner guten Vorsätze, das Unvermeidbare hinauszuzögern, werde ich Ihnen als nächstes über (Trommelwirbel!) das Internet erzählen.