

26 Das Anwendungsframework

733 Die Optionen des Anwendungsframeworks

734 Eine Codedatei implementieren, die Anwendungsereignisse (Starten, Beenden, Netzwerkzustand, generelle Fehler) behandelt

Das, was Microsoft als »Anwendungsframework« bezeichnet, sind eine Reihe von Funktionalitäten, die unter anderem auch dafür verantwortlich sind, dass Anwendungseinstellungen nach dem Beenden eines Programms abgespeichert werden oder auch das Anwendungsereignisse zur Verfügung gestellt werden (siehe ► Abschnitt »Eine Codedatei implementieren, die Anwendungsereignisse (Starten, Beenden, Netzwerkzustand, generelle Fehler) behandelt« ab Seite 734).

Um das zu erreichen, bedient sich Visual Basic 2005 übrigens zwei Mitteln: Zum einen ergänzt der Visual Basic-Compiler Ihren Quellcode um weiteren Code, den Sie selbst aber nie zu Gesicht bekommen. Dieser Quellcode ist quasi fest im Compiler hinterlegt, und werden bestimmte Funktionalitäten benötigt, die Sie durch die Optionen des Anwendungsframework in den Projekteigenschaften für Ihr Projekt aktivieren, ergänzt der Compiler dafür während des Kompilierens den entsprechenden Quellcode, der dann wiederum dafür sorgt, dass Ihr Programm die entsprechenden Funktionalitäten zur Verfügung stellt. Zum anderen benötigt Visual Basic 2005 dazu bestimmte Funktionsaufrufe, die durch die *VisualBasic.dll* des Frameworks zur Verfügung gestellt werden.

Für Windows-Anwendungen ist das Anwendungsframework standardmäßig aktiviert. Sie können die Aktivierung des Anwendungsframeworks in den Projekteigenschaften unter der Registerkarte Anwendung steuern (siehe Abbildung 26.1).

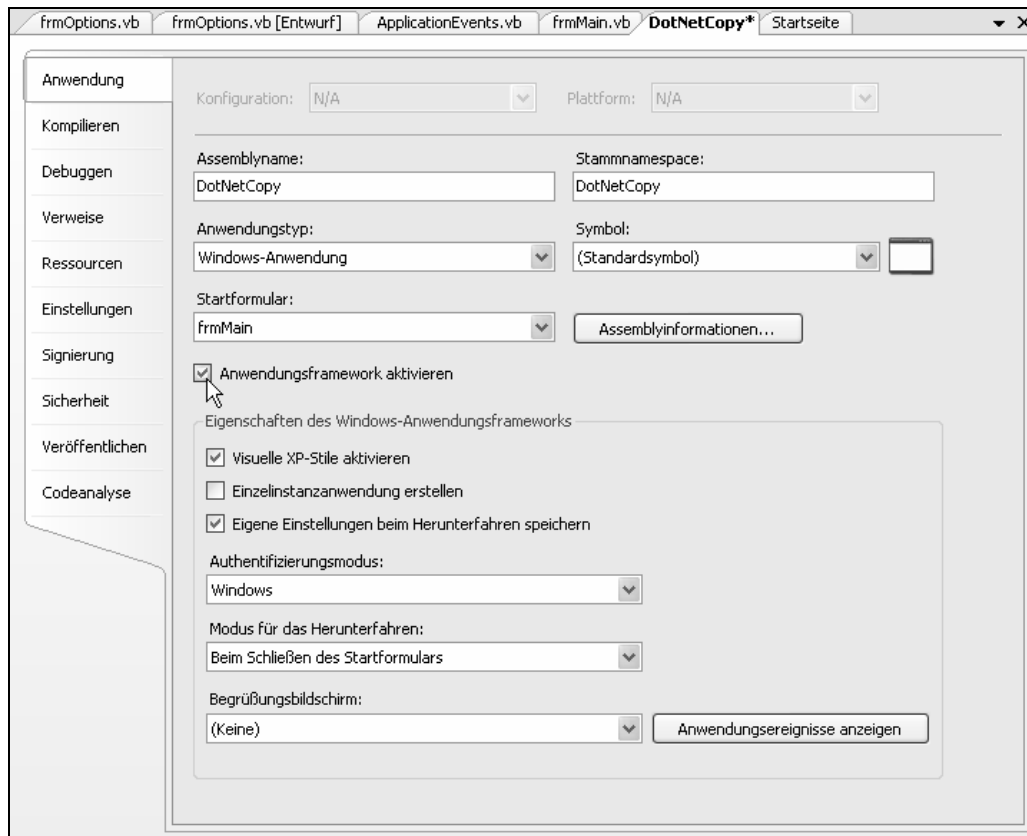


Abbildung 26.1: Erst durch das aktivierte Anwendungsframework stehen viele der hier beschriebenen Funktionalitäten zur Verfügung. Bei Windows Forms-Anwendungen ist das Anwendungsframework standardmäßig aktiviert.

Wenn das Kontrollkästchen *Anwendungsframework aktivieren* aktiviert ist, verwendet die Anwendung dann eine quasi »versteckte« Sub Main (die durch den versteckten Quellcode des Compilers in Ihr Projekt »gelangt«), mit der Ihre Windows-Anwendung startet, und die es erst ermöglicht, dass alle weiteren Optionen des Anwendungsframeworks funktionieren können. In diesem Fall müssen Sie ein Startformular auswählen, das von dieser versteckten Sub Main aufgerufen wird, wenn alle nötigen Vorbereitungen für das Zur-Verfügung-Stellen aller anderen Funktionen des Anwendungsframeworks abgeschlossen sind.

Wenn dieses Kontrollkästchen deaktiviert ist, verwendet die Anwendung die benutzerdefinierte Sub Main, die Sie unter Startformular angegeben haben, und die Sie in einem Modul Ihres Projektes platzieren müssen. In diesem Fall können Sie entweder ein Startobjekt (eine benutzerdefinierte Sub Main in einer Methode oder Klasse) oder ein Formular festlegen. Die Optionen im Bereich *Eigenschaften des Windows-Anwendungsframeworks* sind in diesem Fall nicht verfügbar.

Die Optionen des Anwendungsframeworks

Die folgenden Einstellungen dienen der Konfiguration des Abschnitts Eigenschaften des Windows-Anwendungsframeworks. Diese Optionen sind nur verfügbar, wenn das Kontrollkästchen Anwendungsframework aktivieren aktiviert ist.

Windows XP Look and Feel für eigene Windows-Anwendungen – Visuelle XP-Stile aktivieren

Aktivieren oder deaktivieren Sie die visuellen Windows XP-Stile, die auch Windows XP-Designs genannt werden. Die visuellen Windows XP-Stile lassen z. B. Steuerelemente mit gerundeten Ecken und dynamischen Farben zu. Die Option ist standardmäßig aktiviert.

Verhindern, dass Ihre Anwendung mehrfach gestartet wird – Einzelinstanzanwendung erstellen

Aktivieren Sie dieses Kontrollkästchen, um zu verhindern, dass Benutzer mehrere Instanzen der Anwendung ausführen. Dieses Kontrollkästchen ist standardmäßig deaktiviert, wodurch mehrere Instanzen der Anwendung ausgeführt werden können.

MySettings-Einstellungen automatisch sichern – Eigene Einstellungen beim Herunterfahren speichern

Durch das Aktivieren dieses Kontrollkästchens wird festgelegt, dass die My.Settings-Einstellungen der Anwendung beim Herunterfahren gespeichert werden. Die Option ist standardmäßig aktiviert. Wenn diese Option deaktiviert wird, können Sie diese Festlegung durch Aufrufen von My.Settings.Save manuell durchführen.

Bestimmen, welcher Benutzer-Authentifizierungsmodus verwendet wird

Wählen Sie Windows (Standard) aus, um die Verwendung der Windows-Authentifizierung für die Identifikation des gerade angemeldeten Benutzers festzulegen. Diese Informationen können zur Laufzeit übrigens mit dem My.User-Objekt abgerufen werden. Wählen Sie *Anwendungsdefiniert* aus, wenn Sie eigenen Code anstatt der Windows-Standardauthentifizierungsmethoden für die Authentifizierung von Benutzern verwenden möchten.

Festlegen, wann eine Anwendung »zu Ende« ist – Modus für das Herunterfahren

Wählen Sie *Beim Schließen des Startformulars (Standard)* aus, um festzulegen, dass die Anwendung beendet wird, wenn das als Startformular festgelegte Formular geschlossen wird. Dies gilt auch, wenn andere Formulare geöffnet sind. Wählen Sie *Beim Schließen des letzten Formulars* aus, um festzulegen, dass die Anwendung beendet wird, wenn das letzte Formular geschlossen oder wenn `My.Application.Exit` oder die End-Anweisung explizit aufgerufen wird.

Einen Splash-Dialog beim Starten von komplexen Anwendungen anzeigen – Begrüßungsdialog

Wählen Sie das Formular aus, das als Begrüßungsbildschirm angezeigt werden soll. Zuvor müssen Sie einen Begrüßungsbildschirm mithilfe eines Formulars oder einer Vorlage erstellt haben.

Mehr zum Thema Splash-Dialoge finden Sie im nächsten Kapitel, ► Kapitel 27.

Eine Codedatei implementieren, die Anwendungsereignisse (Starten, Beenden, Netzwerkzustand, generelle Fehler) behandelt

In den vorangegangenen Abschnitten haben Sie schon hier und da am Rande mitbekommen können, dass es offensichtlich – anders als noch in Visual Basic 2002 und 2003 – möglich sein muss, bestimmte Anwendungsereignisse »mithören« zu können. `DotNetCopy` klingt sich so beispielsweise in den Programmstart ein, und steuert das weitere Schicksal des Programms aufgrund der übergebenen Optionen sogar.

Wenn bei bestimmten Anwendungsereignissen Ihr eigener Code ausgeführt werden soll, müssen Sie dazu eine kleine Vorbereitung treffen:

- Rufen Sie dazu die Eigenschaften Ihres Projektes auf, und wählen Sie die Registerkarte *Anwendung*.
- Klicken Sie auf die Schaltfläche *Anwendungsereignisse anzeigen* (siehe Abbildung 26.1).
- Die Visual Studio-IDE fügt Ihrem Projekt nun eine Codedatei namens *ApplicationEvents.vb* hinzu und öffnet diese im Editor.
- Um den Funktionsrumpf für die Behandlungsroutine in eine der fünf existierenden Anwendungsereignisse einzufügen, wählen Sie im Codeeditor aus der linken Aufklappliste am oberen Rand des Editors zunächst den Eintrag `myApplicationEvents`.
- Wählen Sie anschließend in der rechten Liste das Ereignis, das Sie behandeln wollen, und für das der Funktionsrumpf in den Editor eingefügt werden soll. Orientieren Sie sich dabei auch an Abbildung 26.2.

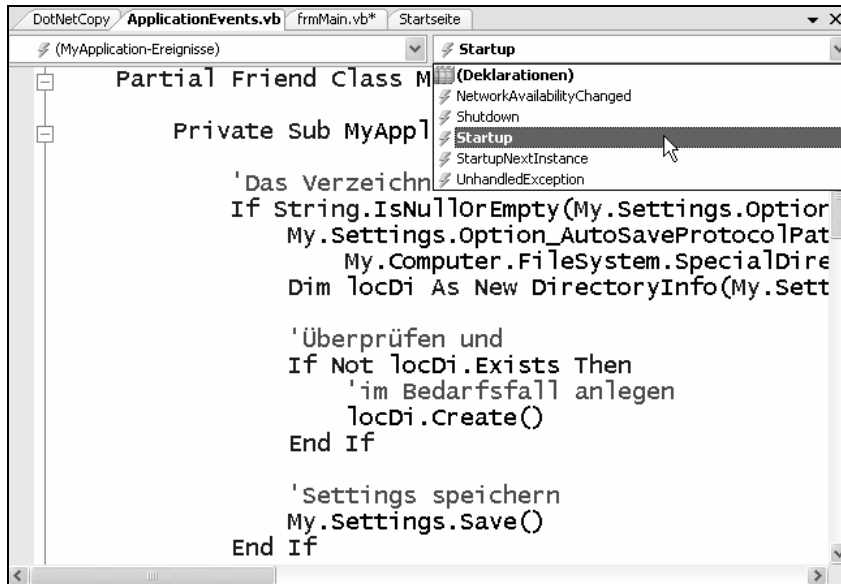


Abbildung 26.2: Wählen Sie zum Einfügen des Rumpfs der Ereignisbehandlungsroutine eines der fünf existierenden Ereignisse aus der Aufklappliste aus

- Formulieren Sie den Behandlungscode aus.

HINWEIS: Beachten Sie auch, dass einige Ereignisse besondere Ableitungen der EventArgs-Klasse als Instanz übergeben, mit denen Sie das weitere Programmverhalten beeinflussen können. Die folgende Tabelle verrät mehr zu den vorhandenen Ereignissen, und wie Sie in ihren Rahmen den weiteren Programmverlauf beeinflussen können:

Ereignisname	Beschreibung	Steuerungsmöglichkeiten
Startup	Wird beim Starten der Anwendung noch vor dem Erstellen des Startformulars und dessen Binden an den Anwendungskontext ausgelöst.	Ja: Der Start der Anwendung kann durch Setzen von <code>Cancel</code> der <code>StartupEventArgs</code> -Instanz verhindert werden. In <code>CommandLineArgs</code> dieser Instanz werden ferner die Befehlszeilen-Argumente an die Anwendung als <code>String-Array</code> übergeben.
ShutDown	Wird nach dem Schließen aller Anwendungsformulare ausgelöst. Dieses Ereignis wird nicht ausgelöst, wenn die Anwendung nicht normal beendet wird, also beispielsweise durch einen Fehler, der eine Ausnahme ausgelöst hat.	Nein.

Ereignisname	Beschreibung	Steuerungsmöglichkeiten
StartupNextInstance	Wird beim Starten einer Einzelinstanzanwendung ausgelöst, wenn diese bereits aktiv ist. HINWEIS: Sie bestimmen, dass eine Anwendung nur in einer Instanz (also nicht mehrmals gleichzeitig) gestartet werden darf, wenn Sie in den Projekteigenschaften Ihres Windows Forms-Projektes auf der Registerkarte <i>Anwendungen</i> die Option <i>Einzelinstanzanwendung erstellen</i> aktivieren.	Ja: Mit dem Setzen der BringToFront-Eigenschaft der StartupNextInstanceEventArgs-Instanz können Sie festlegen, dass nach Verlassen der Ereignisbehandlungsroutine die schon laufende Instanz Ihrer Anwendung wieder zur aktiven Anwendung wird. In CommandLineArgs dieser Instanz werden ferner die Befehlszeilen-Argumente an die Anwendung als String-Array übergeben (und zwar für die, die zum erneuten Startversuch und damit auch zum Auslösen des Ereignisses führten).
NetworkAvailabilityChanged	Wird beim Herstellen oder Trennen der Netzwerkverbindung ausgelöst.	Nein. Sie können jedoch mit der Nur-Lesen-Eigenschaft IsNetworkAvailable abfragen, ob das Netzwerk zur Verfügung steht oder nicht.
UnhandledException	Wird ausgelöst, wenn in der Anwendung auf Grund eines nicht behandelten Fehlers eine unbehandelte Ausnahme auftritt.	Ja: Das Beenden der Anwendung aufgrund des Fehlers kann durch Setzen von Cancel der UnhandledExceptionEventArgs-Instanz verhindert werden. Darüber hinaus steht Ihnen durch die Exception-Eigenschaft dieser Instanz ein Exception-Objekt zur Verfügung, das Auskunft über die aufgetretene unbehandelte Ausnahme gibt.

Der folgende Beispielcode demonstriert den Umgang mit den Anwendungsereignissen:

```

Private Sub MyApplication_NetworkAvailabilityChanged(ByVal sender As Object, _
    ByVal e As Microsoft.VisualBasic.Devices.NetworkAvailableEventArgs) _
    Handles Me.NetworkAvailabilityChanged
    MessageBox.Show("Die Netzwerkverfügbarkeit hat sich geändert!" &
        "Das Netzwerk ist " & IIf(e.IsNetworkAvailable, "verfügbar", "nicht verfügbar").ToString)
End Sub

Private Sub MyApplication_Shutdown(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Shutdown
    Debug.Print("Die Anwendung wird jetzt beendet!")
End Sub

Private Sub MyApplication_StartupNextInstance(ByVal sender As Object, _
    ByVal e As Microsoft.VisualBasic.ApplicationServices.StartupNextInstanceEventArgs) _
    Handles Me.StartupNextInstance
    MessageBox.Show("Sie können keine 2. Instanz dieser Anwendung starten!")
    e.BringToFront = True
End Sub

Private Sub MyApplication_UnhandledException(ByVal sender As Object, _
    ByVal e As Microsoft.VisualBasic.ApplicationServices.UnhandledExceptionEventArgs) _
    Handles Me.UnhandledException

```

```

'Eine bislang nicht behandelte Ausnahme ist aufgetreten!
Dim locDr As DialogResult = _
    MessageBox.Show("Eine unbehandelte Ausnahme ist aufgetreten!" & _
        vbNewLine & "Soll die Anwendung beendet werden?", _
        "Unbehandelte Ausnahme", MessageBoxButtons.YesNo, _
        MessageBoxIcon.Error, MessageBoxDefaultButton.Button1)
If locDr = DialogResult.No Then
    e.ExitApplication = False
End If
End Sub

```

Ein dokumentiertes Beispiel für das Ereignis `Startup`, das beim Starten einer Windows Forms-Anwendung ausgelöst wird, finden Sie auch im Beispielprogramm `DotNetCopy` – der Vollständigkeit halber finden Sie im Folgenden den kompletten Code dieser Ereignisbehandlungsroutine:

BEGLEITDATEIEN: Viele der hier gezeigten Features können Sie sich direkt am Beispiel von `DotNetCopy` (vorgestellt in ► Kapitel 24) anschauen. Sie finden dieses Projekt unter dem Namen `DotNetCopy.sln` im Verzeichnis `.\VB 2005 - Entwicklerbuch\F - Vereinfachung\Kap25\DotNetCopy`. So Sie ► Kapitel 24 noch nicht gelesen haben, sollten Sie es zunächst tun, um das Beispiel besser nachvollziehen zu können.

```

Private Sub MyApplication_Startup(ByVal sender As Object,
    ByVal e As Microsoft.VisualBasic.ApplicationServices.StartupEventArgs) Handles Me.Startup

'Das Verzeichnis für die Protokolldatei beim ersten Mal setzen...
If String.IsNullOrEmpty(My.Settings.Option_AutoSaveProtocolPath) Then
    My.Settings.Option_AutoSaveProtocolPath = _
        My.Computer.FileSystem.SpecialDirectories.MyDocuments & "\DotNetCopy Protokolle"
    Dim locDi As New DirectoryInfo(My.Settings.Option_AutoSaveProtocolPath)

    'Überprüfen und
    If Not locDi.Exists Then
        'im Bedarfsfall anlegen
        locDi.Create()
    End If

'Settings speichern
My.Settings.Save()
End If

Dim locFrmMain As New frmMain

'Kommandozeile auslesen
If My.Application.CommandLineArgs.Count > 0 Then
    For Each locString As String In My.Application.CommandLineArgs

        'Alle unnötigen Leerzeichen entfernen und
        'Groß-/Kleinschreibung 'Unsensibilisieren'
        'HINWEIS: Das funktioniert nur in der Windows-Welt;
        'kommt die Kopierlistendatei von einem Unix-Server, bitte darauf achten,
        'dass der Dateiname dafür auch komplett in Großbuchstaben gesetzt ist,
        'da Unix- (und Linux-) Derivate Groß-/Kleinschreibung berücksichtigen!!!
    
```

```

        locString = locString.ToUpper.Trim

        If locString = "/SILENT" Then
            locFrmMain.SilentMode = True
        End If

        If locString.StartsWith("/AUTOSTART") Then
            locFrmMain.AutoStartCopyList = locString.Replace("/AUTOSTART:", "")
            locFrmMain.AutoStartMode = True
        End If
    Next
End If

'Silentmode bleibt nur "an", wenn AutoStart aktiv ist.
locFrmMain.SilentMode = locFrmMain.SilentMode And locFrmMain.AutoStartMode

'Und wenn Silentmode, erfolgt keine Bindung des Formulars an den Anwendungskontext!
If locFrmMain.SilentMode Then
    'Alles wird in der nicht sichtbaren Instanz des Hauptforms durchgeführt,
    locFrmMain.HandleAutoStart()
    'und bevor das "eigentliche" Programm durch das Hauptformular gestartet wird,
    'ist der ganze Zauber auch schon wieder vorbei.
    e.Cancel = True
Else
    'Im Nicht-Silent-Modus wird das Formular an die Anwendung gebunden,
    'und los geht's!
    My.Application.MainForm = locFrmMain
End If
End Sub
End Class

End Namespace

```