

10.2 Moderne GUI-Elemente

Das .NET Framework 2.0 ist vor allem eine riesige Bibliothek, die dem Vulcan.NET-Entwickler kostenlos zur Verfügung steht. Während in Visual Objects nur zögerlich – wenn überhaupt – Neuerungen bei den GUI-Elementen verfügbar waren, stehen dem Vulcan.NET-Entwickler die neuesten Leistungen direkt zum Einbau in eigene Programme zur Verfügung. Auch wenn es nicht zu den wesentlichen Leistungen eines Programms gehört, so ist eine moderne Benutzeroberfläche nicht selten ein zusätzliches und attraktives Argument für den Anwender, auf eine neuere Programmversion aufzusteigen.

10.2.1 ToolStrips – Eine Einführung mit Portierung von C#

Dieser Abschnitt zeigt nicht nur, wie man moderne Menus in sein Programm einbaut, sondern gibt auch eine ausführliche Anleitung für die Übernahme von C#-Quelltext aus der Dokumentation des .NET Frameworks 2.0. Wir werden sehen, dass es sehr einfach ist. Und das bedeutet, dass alle Beispiele in der genannten Doku (und das sind Hunderte) schnell und einfach nutzbar sind.

```

1      /* MDI2.PRG */
2      REFERENCES "System.Windows.Forms"
3      REFERENCES "System.Drawing"
4      USING System.Windows.Forms
5      USING System.Drawing

6      FUNCTION Start() AS VOID
7      LOCAL oShell AS StandardMDIShell

8      oShell := StandardMDIShell{}
9      oShell :Show()
10     Application.Run(oShell)

11     RETURN

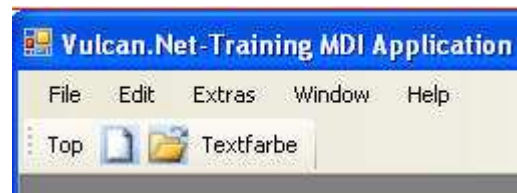
```

Einen Überblick darüber, was man alles mit den in diesem Kapitel beschriebenen *ToolStrips* machen kann, findet man in der .NET Framework SDK 2.0 Dokumentation unter

ms-help://MS.NETFramework.v20.en/dv_fxmclict/html/81d067ed-297c-4dad-90de-1bcac15336ec.htm

Ausführliche Beschreibungen der einzelnen Leistungsmerkmale mit kurzen Beispielen findet man hier

ms-help://MS.NETFramework.v20.en/dv_fxmclict/html/e5ef455a-e049-429c-8005-30c93132fb79.htm



Natürlich ist es auch immer entscheidend mit welchem Aufwand welche neuen Leistungen in existierenden

Anwendungen eingebaut werden können. An dieser Stelle ist nicht nur wichtig, zu wissen, wie man das Standard Erscheinungsbild wie in obiger Abbildung auf das modernere



Aussehen nach MS Office-Art ändert (siehe zweite Abbildung). Die Antwort darauf lautet Es sind gerade mal zwei Zeilen Quelltext vorausgesetzt man hat die richtigen Menu Klassen verwendet nämlich MenuStrip und

ToolStrip. Ein Programm, das klassische VO-Menus in diesen neuen Klassen umwandelt, wäre also angebracht.

```

12  CLASS StandardMDIShell INHERIT Form
13  CONSTRUCTOR() CLASS StandardMDIShell
14  LOCAL tspTop AS ToolStripPanel
15  LOCAL tspBottom, tspLeft, tspRight AS Tool-
StripPanel
16  LOCAL tsTop AS ToolStrip
17  LOCAL tsBottom, tsRight, tsLeft AS ToolStrip
18  LOCAL ms AS MenuStrip
19  LOCAL windowMenu, windowNewMenu AS ToolStrip-
MenuItem
20
21  SELF:IsMdiContainer := TRUE
22  SELF:Text :=;
23  "Standard Vulcan.Net MDI Application"
24  SELF:Size := Size{800,600}
25  SELF:StartPosition :=;
26  FormStartPosition.CenterScreen
27
28  tspTop := ToolStripPanel{}
29  tspBottom := ToolStripPanel{}
30  tspLeft := ToolStripPanel{}
31  tspRight := ToolStripPanel{}
32
33  tspTop:Dock := DockStyle.Top
34  tspBottom:Dock := DockStyle.Bottom
35  tspLeft:Dock := DockStyle.Left
36  tspRight:Dock := DockStyle.Right
37
38  tsTop := ToolStrip{}
39  tsTop:Items:Add("Top")
40  tsTop:Join(tspTop)
41
42  tsBottom := ToolStrip{}
43  tsBottom:Items:Add("Bottom")
44  tsBottom:Join(tspBottom)
45  // Entsprechender Quelltext für tsRight
46  // u. tsLeft hier ausgelassen
47
48  ms := MenuStrip{}
49  windowMenu := ToolStripMenuItem{"Window"}
50  windowNewMenu := ToolStripMenuItem{"New", ;
51  NULL, EventHandler{SELF, ;
52  @StandardMDIShell.windowNewMenu_Click}}

```

In nebenstehendem Constructor (Zeile 13 ff) für ein ShellWindow fügen wir `ToolStripPanels` und `ToolStrips` hinzu.

Zunächst machen wir das Fenster zu einem ShellWindow, indem wir `IsMdiContainer` auf `TRUE` setzen (Zeile 21).

Wir erzeugen vier `ToolStripPanel`-Objekte (Zeile 27-30) und „kleben“ diese an die vier Ränder des Fensters, indem wir der Property `Dock` den entsprechenden Wert aus der `DockStyle`-Enumeration zuweisen (Zeile 31-34). Ein `ToolStripPanel` ist ein zunächst leerer Container für `ToolStrip`-Objekte. Wir erzeugen dann ein `ToolStrip` (Zeile 35), das ist ein verschiebbares Toolbar-Band. Diesem fügen wir ein `ToolStripItem` hinzu mit (Zeile 36)

```
tsTop:Items:Add("Top")
```

Ein `ToolStrip` wird dann mit `Join` einem `ToolStripPanel` zugewiesen (Zeile 37):

```
tspBottom:Join(tsBottom)
```

Entsprechend verfahren wir bei den anderen `ToolStripPanels`.



Die obige Abbildung zeigt das Ergebnis. Der Pfeil zeigt auf den `ToolStrip` `tsTop`, der ein `ToolStripItem` (ein Label „Top“) enthält.