

Vulcan.NET



Anfängerkurs (VNA051)

Beschreibung des Kurs-Inhalts und -Begleitmaterials

Kursmaterial von

Dieter Crispian Software-Entwicklung und Vertrieb

dcSE

Dieter Crispian Software Entwicklung und Vertrieb

1 **Vorspann**

2 **Portierung von Visual Objects Quelltext**

Was ist kompatibel und was nicht, ist hier Thema. Eines der wichtigsten Leistungsmerkmale für Vulcan.NET-Anwender, die aus dem Visual Objects-Bereich kommen, ist, mit wieviel Aufwand und unter welchen Bedingungen ihr existierender Visual Objects-Code nach Vulcan.NET übernommen werden kann. Dieses Kapitel beschreibt die wenigen Ecken und Kanten beim Übergang von Visual Objects nach Vulcan.NET und bietet in vielen Fällen Hintergrund-Informationen, die weit über die Darstellung in der Hilfedatei hinausgehen.

Das Kapitel 2 widmet (fast) jedem der in Visual Objects vorkommenden Datentypen einen Abschnitt. Dazu gehören unter anderem Array, Codeblock, Date, Float, Klassen, String, Symbol und Usual. Aber auch auf nicht mehr unterstützte Leistungen wie Publics und Privates wird eingegangen. Der folgende längere Abschnitt wurde aus der aktuellen Fassung des Kursmaterials übernommen.

2.2 Unterstützte Leistungen von Visual Objects

In diesem Abschnitt geht es vor allem um spezielle Leistungen von Visual Objects, die in anderen Sprachen nicht vorkommen, und wie weit diese in Vulcan.NET nachempfunden wurden. Da Vulcan.NET vor allem den Anforderungen der CLS für die .NET-Welt entsprechen will, ist hier bei der Implementierung in manchen Fällen eine Kompromisslösung notwendig. Die Grundaussage ist jedoch immer: Ja, es gibt auch in Vulcan.NET die VO-spezifischen Arrays, Codeblocks usw. Aber es sind teilweise (meist unbedeutende) Einschränkungen hinzunehmen. Auch auf die entsprechenden Erweiterungen wird hingewiesen.

2.3 Arrays

In der .NET-Welt sind Arrays 0-basiert implementiert, d.h. das erste Array-Element hat den Index 0 und nicht 1 wie in Visual Objects. Vulcan.NET ist auch hier kompatibel mit VO, muss aber "unter der Haube" den Index 0 verwenden, damit die Zusammenarbeit (Interoperabilität) mit anderen .NET-Sprachen gewährleistet ist. Das wird nur dann für den Vulcan.NET-Programmierer offensichtlich, wenn er Vulcan.NET-Quelltext im Debugger sieht oder wenn er Quelltext von anderen .NET-

Sprachen nach Vulcan.NET portiert. Mit einem Compiler-Schalter /az kann man auch in Vulcan.NET 0-basierte Arrays erzwingen (standardmäßig ist dieser Schalter natürlich nicht gesetzt!). Dies würde jedoch für die ganze Anwendung gelten, d.h. es empfiehlt sich, diesen nur für komplett neu geschriebene Anwendungen zu setzen. In Vulcan.NET kann auch weiterhin die vertrautere Notation für die Deklaration von DIM-Arrays verwendet werden: `LOCAL DIM aItems[2] AS STRING`

In diesem Fall wird jedoch kein Array-Objekt (`System.Array`) erzeugt. VO meint in diesem Fall lediglich einen zusammenhängenden Bereich auf dem Stack, der fortlaufend adressierbar ist. In Vulcan.NET wird empfohlen, wenn immer möglich, streng typisierte (.NET-)Arrays zu verwenden, da diese sehr viel effizienter sind als dynamische Arrays, die in Visual Objects vorzugsweise eingesetzt werden. Eine Besprechung der .NET-Arrays finden Sie im Kapitel 9.

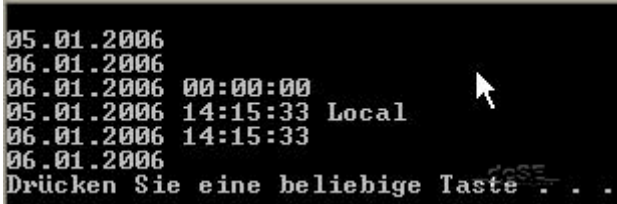
2.4 Datum

Der Datentyp DATE ist in Vulcan.NET kompatibel zu VO implementiert. Diese Implementierung basiert auf der .NET Framework-Struktur `DateTime`. `DateTime` ist sehr viel leistungsfähiger als der VO-Datentyp DATE und sollte deshalb in neu geschriebenem Quelltext vorgezogen werden. `DateTime`-Angaben müssen in Feldern vom Typ String in DBFs gespeichert werden. SQL-Systeme haben oft auch einen `DateTime`-Datentyp. Das .NET-Framework 2.0 unterscheidet jetzt auch zwischen UTC-Zeitangaben (`UtcNow`) und lokalen Zeitangaben (`Now`). Diese kann über die `Kind`-Property abgefragt werden und über die Methoden `ToUniversalTime()` bzw. `ToLocalTime()` umgewandelt werden. Zum Speichern von Zeitangaben wird UTC empfohlen, für die Anzeige auf dem Bildschirm lokale Zeit.

Das folgende Vulcan.NET-Programm zeigt die unterschiedliche Verwendung:

```
1 FUNCTION Start() AS VOID
2 LOCAL dt AS DateTime
3 LOCAL d AS DATE // kompatibel zu VO
4 d := Today() // wie in VO
5 ? d // 1.Zeile in flg. Abb.
6 d := d+1 // wie in VO
7 ? d // 2.Zeile in flg. Abb.
8 dt := d // sichere Konvertierung, kein Cast notwendig
9 ? dt // 3.Zeile in flg. Abb.
10 dt := system.DateTime.Now // mit Zeitangabe
11 ? dt, dt:kind // 4.Zeile in flg. Abb.
12 //dt := dt+1 // Compiler-Fehlermeldung:
13 // + is not defined for Types 'System.DateTime' and
14 // System.Int32
15 dt:= dt:AddDays(1) // einen Tag hinzufügen
16 ? dt // 5.Zeile in flg. Abb.
```

```
17 d := (DATE) dt // explicit cast required
18 // die Zeitangabe geht verloren
19 ? d // 6.Zeile in flg. Abb.
20 RETURN VOID
```



```
05.01.2006
06.01.2006
06.01.2006 00:00:00
05.01.2006 14:15:33 Local
06.01.2006 14:15:33
06.01.2006
Drücken Sie eine beliebige Taste . . .
```

(Anm.: der obige Quelltext enthält neue Casting-Syntax - z.B. in Zeile 17 -, die im Kurs ausführlich erläutert wird!)

2.5 Fließkommazahlen (FLOATs)

Im Gegensatz zu Visual Objects sind Fließkommazahlen in Vulcan.NET keine Verweistypen sondern Werttypen. In den allermeisten Fällen ist dieser Unterschied für den Entwickler jedoch bedeutungslos, da die Art der Verwendung identisch mit der in Visual Objects ist..... (Ende des Textauszugs)

3 Die wichtigsten visuellen Werkzeuge

Hier werden die Entwicklungsumgebungen VIDE und Visual Studio 2005 in ihren wesentlichen Leistungen beschrieben. Und genauso wie man ganz ohne auskommt.

4 Hybrid Programmierung - Nach der Portierung was dann?

In diesem Kapitel geht es um den sanften Übergang nach .NET. Wie kann man .NET-Leistungen in Visual Objects nutzen und umgekehrt. Was ist bei Vulcan/Visual Objects DLLs zu beachten, die man in der jeweils anderen Sprache nutzen möchte.

5 Die Beispiele von Vulcan.NET

Fast alle der mit Vulcan.NET ausgelieferten Beispiele werden in diesem Kapitel ausführlich kommentiert. Mit Vulcan.NET werden viele Beispiele ausgeliefert, die als Muster für eigene Programme genommen werden können. Sie enthalten auch viele Programmiertricks, die für den Anfänger hilfreich sein können.

6 Der Compiler und mehr

Hier geht es um Compiler-Einstellungen und Warnungen. Wie man diese in VIDE einstellt und was es mit MSBuild auf sich hat.

7 Debugging und Fehlerbehandlung

Die zur Verfügung stehenden Debugger sind neue Werkzeuge. Man muss sich also erst eingewöhnen, um damit effizient zu arbeiten. Tracing unter .NET bietet neue Möglichkeiten - lange bevor man mit den schwersten Geschützen, den Debuggern, auf Fehlersuche geht. Auch um die Fehlerbehandlung auf die .NET-Art kommt man früher oder später nicht herum, wenn man mit der Framework Class Library arbeitet.

8 Neue Konzepte in Vulcan.NET

Als Visual Objects-Entwickler hat man bisher nichts mit Konstruktoren im .NET-Sinn zu tun gehabt. Was bieten die Möglichkeiten von STATIC und abstrakten Klassen/Methoden? Was steckt hinter dem Konzept von versiegelten Methoden? Wie nutzt man Interfaces?

9 Neue Datentypen

Eine Auswahl der mit .NET zur Verfügung stehenden neuen Datentypen wird hier mit Beispiel-Quelltexten beschrieben.

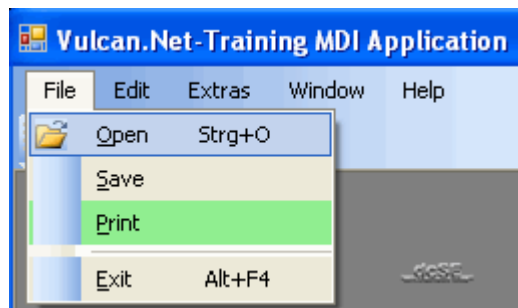
10 Einsatz von .NET-Leistungen

Dieses Kapitel stellt ein paar Leistungen der Framework Class Library (FCL) im Vulcan.NET Quelltext vor, die sicher für Erweiterungen einmal portierter Visual Objects-Anwendungen in Erwägung gezogen werden. Dieses Kapitel könnte leicht zu einer mehrbändigen Enzyklopädie erweitert werden, so umfangreich sind die Leistungen der FCL.

10.1 Vergleich von Visual Objects OLE und .NET InterOp

10.2 Moderne GUI-Elemente

Hier sind Toolstrips und Menus im Office-Look Thema



Die moderne Weiterentwicklung der Toolbar heißt *ToolStrip*. Mehrere ToolStrips können in einem *ToolStripPanel* zusammengefasst werden und innerhalb dieses Panels verschoben werden - so wie man es von den ToolStrips in den Office-Anwendungen kennt.

Der folgende Quelltext zeigt wie man ein ToolStripPanel erzeugt und konfiguriert.

```
// Create the "Top" ToolStrip control and add
// to the corresponding ToolStripPanel.
SELF:tsTop := ToolStrip{}
SELF:tsTop:Items:Add("Top")
SELF:tsTop:ShowItemToolTips := TRUE
SELF:tsTop := SELF:ConfigTsTop(SELF:tsTop)
SELF:tspTop:Join(tsTop)
```

Im obigen Quelltext wird aus Lernzwecken der Prefix `SELF` angegeben, der jedoch bei Instanzvariablen überflüssig wäre. Die folgenden Quelltextzeilen stellen einen Ausschnitt aus der selbst geschriebenen Methode `ConfigTsTop()` dar:

```
// ToolStrip konfigurieren
ts:ImageScalingSize := Size{20,20}
ts:BackgroundImage := Bitmap.FromFile("d:/bgo.bmp")
ts:Items:Add( (ToolStripItem) bNew )
ts:Items:Add( (ToolStripItem) bOpen )
// Button konfigurieren
bNew := ToolStripButton{}
bNew:Image := Bitmap.FromFile("d:\\NewItem.bmp")
bNew:DisplayStyle := ToolStripItemDisplayStyle.Image
bNew:ImageAlign := ContentAlignment.MiddleLeft
bNew:ImageScaling := ToolStripItemImageScaling.SizeToFit
bNew:Name := "bNew"
bNew:Text := "&New"
bNew:TextAlign := ContentAlignment.MiddleRight
bNew:Click += EventHandler{SELF,@StandardMDIShell.ButtonNew}
```

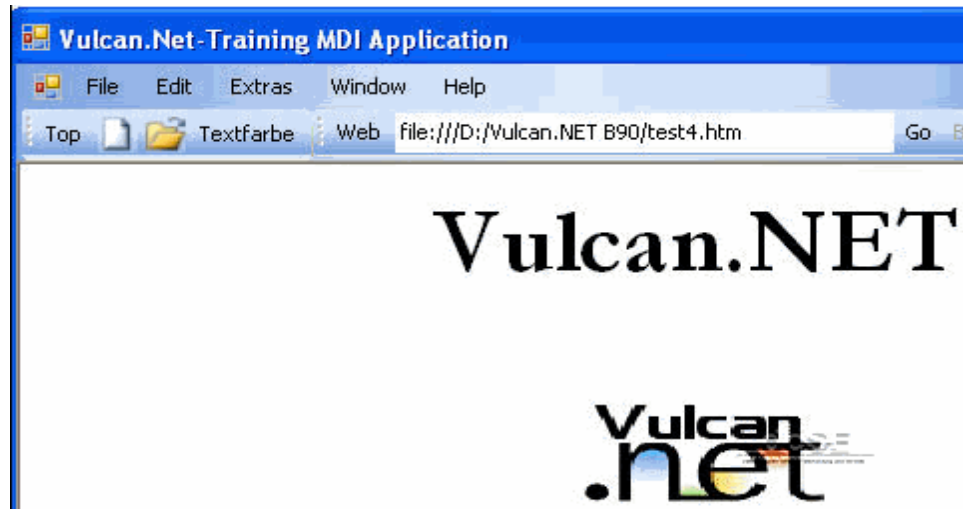
Wem das zu komplex erscheint, sei gesagt, dass sich fast alle Zeilen von Visual Studio 2005 generieren lassen. Natürlich wird auch der oben stehende Quelltext im Begleitmaterial ausführlich kommentiert (siehe „Beispielseiten aus dem Begleitmaterial“¹). Übrigens gibt es neben den Buttons als ToolStrip-Element auch Comboboxen, Progressbalken und TextBox (entspricht SingleLineEdit), die in beliebiger Anzahl auf einem ToolStrip platziert werden können und denen ohne Mühen der passende Eventhandler zugewiesen werden kann. All das in diesem Abschnitt des umfangreichen Kapitels 10 zu den Leistungen von .NET, die man sicher bald nutzen will.

10.3 Das WebBrowser-Control

Internetzugriff aus der eigenen Anwendung, Drucken von HTML-Seiten oder Auffüllen von HTML-Code mit DBF-Daten. All das leicht gemacht mit dem WebBrowser-Control.

¹ Auch diesem Auszug aus den Kursunterlagen kann man entnehmen, dass es Einiges zu lernen gibt, sobald es darum geht, die Klassen der Framework Class Library zu nutzen. Das trifft ja auch eigentlich immer zu, wenn man eine neue Klassenbibliothek nutzt, oder?

Die folgende Abbildung zeigt das WebBrowser-Control integriert in eine MDI-Anwendung. Eine lokale HTML-Seite wird angezeigt.



Man kann jeden HTML-Text in diesem "managed wrapper" um das WebBrowser ActiveX verwenden. Es arbeitet automatisch mit der "FullTrust"-Sicherheitseinstellung. Eine Anwendung, die weniger als FullTrust-Modus hat (z.B. PartiallyTrust-Modus) kann keine "shared managed" DLLs aufrufen. Anhand des WebBrowser-Beispiels wird gezeigt, wie Sicherheitseinstellungen auf Klassen-Ebene realisiert werden, denn das WebBrowser-Control erfordert, dass der aufrufende Quelltext ebenfalls die "FullTrust"-Erlaubnis hat. Diese gewährt man einer Methode über sogenannte Attribute. In unserem Fall schaut die Klassendeklaration des Fensters, zu dem das WebBrowser-Control gehört,

wie folgt aus:

```
[PermissionSet(SecurityAction.Demand,Name := "FullTrust")];  
[ComVisibleAttribute(TRUE)];  
CLASS BrowserForm INHERIT Form
```

Ein ausführlicher Exkurs zu Attributen ist ebenfalls in den Unterlagen enthalten.

Beispiele für den Einsatz des Controls sind HTML-Hilfetexte, oder Internetzugriff aus Ihrem Programm und unter Kontrolle Ihres Programms. Oder man verwendet das Control einfach im Hintergrund, um HTML-Seiten auszudrucken. Neben der WebBrowser-Klasse im verwalteten Code enthält die Framework Class Library auch die Möglichkeit, das WebBrowser Control auf die herkömmliche ("unmanaged") Weise zu nutzen. Dies ist für Quelltext, der das alte DOM (über COM) anspricht, vorzuziehen. Die Kursunterlagen zeigen wie einfach es ist, DBF-Daten in HTML-Format anzuzeigen.

10.4 Exkurs zu Attributen

10.5 Dateizugriff auf die .NET-Art

10.6 DataGridView für tabellarische Datenpräsentation

Manch einer hätte sich den DataBrowser von Visual Objects so gewünscht.



	bestellen	Titel	Künstler	Album
	<input type="checkbox"/>	Revolution 9	Beatles	<i>The Beatles [...]</i>
	<input type="checkbox"/>	One of These ...	Pink Floyd	<i>Meddle</i>
	<input checked="" type="checkbox"/>	Where Is My ...	Pixies	<i>Surfer Rosa</i>
▶	<input type="checkbox"/>	Fools Rush In	Frank Sinatra	<i>Nice 'n' Easy</i>

Die DataGridView ist ein äußerst leistungsfähiges Control der .NET Framework Class Library. Ihre Anwendung erhält sofort ein attraktives modernes Erscheinungsbild. Und Sie können sicher sein, wenn Microsoft entscheidet, weitere Leistungen diesem Control hinzuzufügen, sind Sie sofort in Vulcan.NET verfügbar.

Einer DGV-Zelle können unter anderem folgende Controls für Anzeige bzw. Bearbeitung zugeordnet werden.

- TextBox (Voreinstellung)
- CheckBox
- Image

- Link
- Button
- ComboBox

Eine DGV kann Array-Daten, DBF-Daten oder SQL-Daten darstellen. Sie können im Eingabemodus auf beliebige gültige Zeichen beschränken und den eingegebenen Wert auf Gültigkeit prüfen. Wie aufwendig ist das alles? Nun, Sie müssen wissen, welche Ereignisroutinen in welcher Weise programmiert werden müssen. Aber das und noch viel mehr lernen Sie ja in dem Kurs-Abschnitt über die DataGridView. Hier allein die Themen, die für eine DataGridView derzeit im Kursmaterial enthalten sind.

- Spaltenformatierung in Abhängigkeit vom Datentyp
- Virtuelle Spalten (siehe CheckBox in der Abb. oben)
- Zellspezifische Tooltips
- Gestaltungsmöglichkeiten
- Kopieren eines ausgewählten Zellbereichs in die Zwischenablage

- Eingaben in Zellen und deren Validierung
- Beschränkung der Eingabemöglichkeiten
- Bilddarstellung in Zellen
- Bindung an Datenquellen

Zur DataGridView gibt es derzeit sechs ausführlich kommentierte Quelltextbeispiele (zum großen Teil zusätzlich mit C#-Quelltext).

10.7 Tooltips

11 Fortgeschrittene Themen

Dieses Kapitel beschäftigt sich mit XML, Unicode, Webservices, PInvoke und 'Codeblocks vs. Delegates'

12 Nutzung vorhandener Quellen

Was bringt mir das .NET Framework SDK? Welche Internetseiten bieten nützliche Informationen zu .NET? Wie portiert man C#-Quelltext nach Vulcan.NET? Nützliche Hilfsprogramme, Literatur-Hinweise.

13 Anhang A: Installation von Vulcan.NET

Anforderungen an die Entwickler-Maschine, Hinweise zur Installation

14 Anhang B: Allgemeines zu .NET

Was ist managed und unmanaged Code? Wie vertragen die sich? Plattform-Unabhängigkeit: Gibt es die wirklich? Unter welchen Bedingungen?

Impressum:	Dieter Crispian Software-Entwicklung, -Schulung und -Vertrieb		
	Frauenlobstr. 24	80337 München	Tel: 089/51519608
			Fax: 089/54349983
	E-Mail:	yosales@visualobjects.de	
Internet:	www.visualobjects.de (Hauptseite)	www.vulcandotnet.de (Vulcan.NET)	www.vodc.visualobjects.de (Konferenzen)
	www.support.visualobjects.de (Support)		

Version: 1.0 vom 04.06.06
