Visual-Basic 5.0

Christian Zahler

6 Datenaustausch: OLE und DDE

6.1 Das Clipboard-Objekt

Das Objekt "`Clipboard`" entspricht der Windows-Zwischenablage und dient zum Austausch von Text oder Grafik mit anderen Windows-Applikationen.

```
Clipboard.SetText „Text“ (1)
```

schreibt "`Text`" in die Zwischenablage

```
Clipboard.SetData Bild1.Picture
```

kopiert die Grafik `Bild1.Picture` in die Zwischenablage

Die Zahl in Klammer hat folgende Bedeutung:

1	Text (Konstante `vbCFTEXT`)
2	Bitmap (`vbCFBITMAP`)
3	Windows Metafile (vbCFMETAFILE)
4	DIB-Datei (device independent bitmap) (vbCFDIB)
&HBF00	`vbCFLINK` = DDE-Information zum Datenaustausch (siehe später!)

```
a$ = Clipboard.GetText (1)
```

holt den in der Zwischenablage befindlichen Text und legt ihn in der Variablen `a$` ab

```
Bild1.Picture = Clipboard.GetData ()
```

holt die in der Zwischenablage befindliche Grafik und legt sie im Bildfeld `Bild1` ab

```
Clipboard.Clear
```

löscht den Inhalt der Zwischenablage

Beispiel: Die folgende Anweisung kopiert den Inhalt der Datei `chess.bmp` in die Zwischenablage:

```
Clipboard.SetData LoadPicture
„c:\windows\chess.bmp“
```

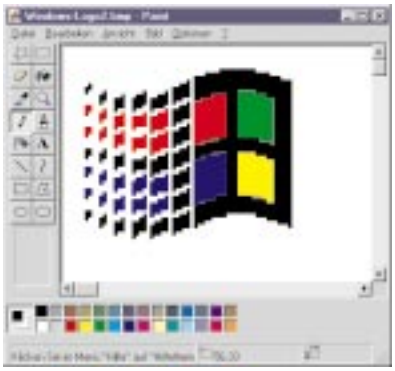
6.2 OLE = Object Linking and Embedding:

Object Linking: Objekt aus einer anderen Windows-Anwendung (etwa Word, Excel) wird nur eingebunden, d.h. angezeigt. Es kann in der VB-Anwendung *nicht bearbeitet* werden.

Object Embedding: Objekt aus einer anderen Windows-Anwendung (etwa Word, Excel) wird verknüpft (eingebettet), d.h. es wird angezeigt und kann in der VB-Anwendung ("OLE-Client") *bearbeitet* werden. Dazu wird das OLE-Programm (Word, Excel usw. – der "OLE-Server") gestartet.

OLE-Server: liefert die Daten.

Beispiel: MS-Paint ist hier Server; das Windows-Logo als Bitmap wird in ein Word-Dokument eingefügt.



OLE-Client („Rahmenanwendung"): übernimmt Daten vom OLE-Server

Beispiel: MS-Word ist hier Client, da es Daten von Paint übernommen hat.

Herstellen von OLE-Verbindungen

Dafür steht ein eigenes Steuerelement zur Verfügung:

OLE Container. Damit können Objekte anderer Windows-Applikationen (Excel, Word usw.) in Ihre Visual-Basic-Anwendung eingebettet oder verbunden werden.

Nachdem ein OLE-Steuerelement angelegt ist, erscheint automatisch ein Dialogfeld, in welchem die Art der Verbindung (Einbetten oder Verknüpfen) und der OLE-Server angegeben werden muss:

Für die Erstellung einer Verknüpfung ist das Kästchen „***AIs Symbol***" anzuhaken.

Wenn das Objekt besteht, so steht mit der rechten Maustaste ein Kontext-Menü zur Verfügung:

Insert Object (Objekt einfügen)	Immer aktiv; blendet obiges Dialogfenster ein
Paste Special (Inhalte einfügen)	Fügt Daten über die Zwischenablage ein (nur aktiv, wenn sich gültige Daten im Clipboard-Objekt befinden)
Delete Embedded Object (Eingebettetes Objekt löschen)	entfernt eingebettetes Objekt
Delete Linked Objekt (Verknüpftes Objekt löschen)	entfernt verknüpftes Objekt und beendet die OLE-Verbindung
Create Link (Verknüpfung herstellen)	erstellt eine Verknüpfung, indem die SourceDoc-Eigenschaft gesetzt wird
Create Embedded Object (Eingebettetes Objekt erstellen)	erstellt ein eingebettetes Objekt

OLE-Steuerelemente haben natürlich eine Reihe von Eigenschaften und Methoden.

Methoden

CreateEmbed	Erstellt ein eingebettetes Objekt
CreateLink	Erstellt ein verknüpftes Objekt aus einer Datei
Copy	Legt eine Kopie eines Objekts in der Zwischenablage ab
Paste	Fügt den Inhalt der Zwischenablage das OLE-Steuerelement ein
Update	Update (aktualisiert das Objekt)
DoVerb	Öffnet ein OLE-Objekt z.B. zum Bearbeiten
Close	Schließt ein OLE-Objekt und trennt die Verbindung mit der Server-Anwendung.
Delete	Löscht ein OLE-Objekt
SaveToFile	Speichert ein OLE-Objekt als Datei
ReadFromFile	Lädt ein mit `SaveToFile` gespeichertes OLE-Objekt
InsertObjDlg	Zeigt den Dialog „***Objekt einfügen***“ (Insert Object) an
PasteSpecialDlg	Zeigt den Dialog „***Inhalte einfügen***“ (Paste Special) an
FetchVerbs	Aktualisiert die Liste der Verben, die von einem Objekt unterstützt werden
SaveToOle1File	Speichert das Objekt im OLE1-Format

Eigenschaften

`OleTypeAllowed = Einstellung%`

`OleTypeAllowed = 0`: Object Linking
`OleTypeAllowed = 1`: Object Embedding
`OleTypeAllowed = 2`: Wahlweise Object Linking oder Embedding

`SourceDoc = Dateiname$`

Bestimmt den Dateinamen der Datei, in welcher ein OLE-Objekt gespeichert wurde.

`SourceItem = Daten$`

Bestimmt, auf welche Daten zugegriffen werden soll.

`Class = Klassenname$`

Legt die Anwendung fest, in der das Objekt erzeugt hat. Beispiele: `WordDocument ExcelWorkSheet`

`PasteOK`

Bestimmt, ob der Inhalt der Zwischenablage in ein OLE-Client-Steuerelement eingefügt werden kann

Beispiel: Excel-Tabelle ohne OLE-Container einfügen (Embedding)

```
OLE1.OLETypeAllowed = 1
  'Embedding
OLE1.SourceDoc = „TABELLE.XLS“
  'Datei mit Excel-Tabelle
OLE1.Class = „ExcelWorkSheet“
  'Klassenname für Excel-Tabelle
OLE1.CreateEmbed
  'eingebettetes Objekt erstellen
```

Beispiel: Excel-Tabelle direkt einfügen (Linking)

```
OLE1.OLETypeAllowed = 0
  'Linking
OLE1.SourceDoc = „TABELLE.XLS“
  'Datei mit Excel-Tabelle
OLE1.SourceItem = „Z10S5"
  'Feld Zeile 10, Spalte 5 (= E10)
OLE1.Class = „ExcelWorkSheet“
  'Klassenname für Excel-Tabelle
OLE1.CreateLink
  'verknüpftes Objekt erstellen
```

Beispiel: Objekt aus der Zwischenablage einfügen

```
OLE1.OLETypeAllowed = 2
  'Linking und Embedding zulässig
Inf OLE1.PasteOK Then
  'sind gültige Daten in Clipboard?
  OLE1.Paste
  'falls ja, Daten einfügen
End If
```

Beispiel: Objekt in die Zwischenablage kopieren

```
OLE1.Copy
```

6.3 DDE = Dynamic Data Exchange:

DDE beschreibt die Kommunikation zweier Windows-Anwendungen, die gegenseitig Daten austauschen.

Das Programm, das die Kommunikation beginnt und Daten anfordert, wird als **DDE-Client** bezeichnet; dasjenige, das diese Daten liefert, heißt **DDE-Server**.

Jede Anwendung, die als DDE-Server dienen soll, benötigt einen eigenen **DDE-Namen**, der aber meist der eigentlichen Anwendungsbezeichnung sehr ähnlich ist:

Anwendung	DDE-Name
Word für Windows	`WinWord`
Excel	`Excel`

Fungiert ein Visual Basic-Programm als DDE-Server, so ist der DDE-Name der Hauptname der Projektdatei (`*.VBP`) oder – falls eine `*.EXE`-Datei erstellt wurde – der Dateiname dieser ausführbaren Datei.

Man unterscheidet:

aktive Verbindung („Hot Link“)	passive Verbindung („Cold Link“)
Eine aktive Verbindung ist dann vorhanden, wenn Client und Server in ständigem Kontakt miteinander stehen. Werden die Server-Daten (z.B. in einer Excel-Tabelle) aktualisiert, so erhält der Client gleichzeitig die aktualisierten Daten.	Eine passive Verbindung tauscht Daten nur aus, wenn der Client dies beim Server anfordert. Dazu dient die *LinkRequest*-Methode.

DDE-Verknüpfungen können entweder zur Entwicklungszeit oder zur Laufzeit hergestellt werden:

6.3.1 zur Entwicklungszeit:

Hier wird eine „starre“ Verknüpfung während der Entwicklungszeit eingerichtet, die dann während der Laufzeit in gleicher Weise wieder aufgebaut wird.

Die so hergestellten Verbindungen sind aktive Verbindungen. Sie werden mit der Form gespeichert. Sie benötigen zwar keinen Code, da sie ausschließlich mit Menübefehlen hergestellt werden, können aber erst mit Programmende abgebrochen werden!

Vorgangsweise: ***Herstellen einer Client-Verbindung***

- Auswahl der zu übertragenden Daten im DDE-Server (Word, Excel, ...)
- im DDE-Server: Menüpunkt ***Bearbeiten – Kopieren*** (Edit – Copy)
- im VB-Programm: Steuerelement aussuchen, das die Daten aufnehmen soll (Bezeichnungsfeld, Bildfeld, Textfeld)
- in VB: Menüpunkt ***Bearbeiten – Verbinden und Einfügen*** (Edit – PasteLink)

Vorgangsweise: ***Herstellen einer Server-Verbindung***

- in VB: Steuerelement aussuchen, das die Daten an den Client übergeben soll (Bezeichnungsfeld, Bildfeld, Textfeld)
- in VB: Menüpunkt ***Bearbeiten – Kopieren*** (Edit – Copy)
- im Client-Programm: Ziel aussuchen (wohin sollen die Daten kommen, z.B. in Excel: Zelle markieren)
- im Client-Programm: Menüpunkt ***Bearbeiten – Verbinden und Einfügen*** (Edit – PasteLink)

6.3.2 zur Laufzeit

Hier wird nicht über das Menü gearbeitet; die Verbindung muss programmiert werden. Dafür ist es nur nötig, einige spezielle Eigenschaften richtig zu setzen:

`LinkTopic`

Diese Eigenschaft legt das **Thema** der DDE-Kommunikation fest, d.h. meist der Name der Datei des Programms, auf dessen Inhalt sich die Kommunikation bezieht.
Syntax:
`Text1.LinkTopic = "Excel|c:\excel\umsatz.xls"`
Wichtig: Zwischen DDE-Name der Anwendung und Dateiname (Thema) muss ein senkrechter Strich (|) stehen.

`LinkItem`

Diese Eigenschaft legt das zu übertragende Element fest, etwa bei Excel das Feld `Z1S1 (= A1)`.

`LinkMode`

Christian Zahler christian.zahler@telecom.at

Gibt die Art der DDE-Verbindung an:
`Text1.LinkMode = 0`: kein DDE-Dialog
`Text1.LinkMode = 1`: Hot Link, aktiver DDE-Dialog
`Text1.LinkMode = 2`: Cold Link, passiver DDE-Dialog (wird nur geführt, wenn durch LinkRequest-Methode angefordert)

`LinkRequest`

Diese Methode fordert während einer DDE-Kommunikation den Server auf, den Inhalt eines Steuerelements zu aktualisieren.

`LinkSend`

Überträgt den Inhalt eines Bildfeldes an den Client. *Beispiel:*

```
Private Sub Form_ ()
  If Text1.LinkMode = 0 Then
      ' Falls kein
    Text1.LinkTopic = _
       „Excel|[Mappe1]Tabelle1"
      ' Anwendung und Thema (Datei)
    Text1.LinkItem =  "Z1S1"
      ' Zu übertragendes Element
    Text1.LinkMode = 2
      ' Passiver DDE-Dialog
    Text1.LinkRequest
      ' Textfeld aktualisieren
  Else
    If Text1.LinkItem  = "Z1S1" Then
      ' übrigens: auch "A1" geht!
      Text1.LinkItem = "Z2S1"
      Text1.LinkRequest
      ' Textfeld aktualisieren
    Else
      Text1.LinkItem = "Z1S1"
      Text1.LinkRequest
      ' Textfeld aktualisieren
    End If
  End If
End Sub
```

Ereignisse im Zusammenhang mit DDE: (Syntax und Werte siehe Online-Hilfe!)

`LinkClose`	Zeigt an, dass ein Steuerelement eine beendet.
`LinkError`	Zeigt an, dass während der Kommunikation ein DDE-Fehler aufgetreten ist.
`LinkExecute`	Zeigt die Übertragung einer Befehlsfolge vom Client an der Server, wobei der Server diese Befehlsfolge ausführen soll
`LinkOpen`	Zeigt an, dass ein Steuerelement oder ein Client-Anwendungsprogramm die DDE-Kommunikation beginnt

Begleitende Dokumentation in VB-Projekten

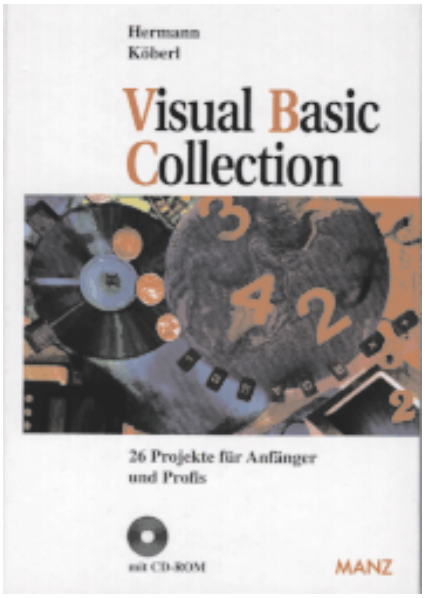
Martin Wertjanz

Als Unterrichtsbehelf für VisualBasic habe ich ein Formular entwickelt. Es besteht auf Tabellen, die den Entwickler die begleitende Dokumentation erleichtern sollen. Für die ersten Lektionen hat es sich jedenfalls sehr bewährt.

Möglicherweise möchten auch andere VB-Lernende und -lehrende dieses als Kopiervorlage verwenden. Das WinWord-Dokument enthält auf der ersten Seite das Formular, auf der zweiten Seite ein Muster für ein Projekt. PCNEWS-Leser können das Dokument aus dem Web laden: http://pcnews.at/ins/pcn/62/~62.htm.

Visual Basic Collection

Martin Weissenböck



Hermann Köberl hat in seinem Buch "Visual Basic Collection" (Manz-Verlag, ISBN 3-7068-0562-6, 224 Seiten) einen originellen Zugang zu Visual Basic gewählt: in 26 ansprechenden Projekten (Beispielen) werden die Eigenschaften und Möglichkeiten der Sprache dargestellt. Auf der beigefügten CD sind die Beispiele auch zu finden. Darunter sind Aufgaben wie ein "Spielcasino", der "Biorhythmus", eine "Watchlist", das Spiel "Hangman" und andere. Nur beim "Kalorienplan" hätte ich mir einen "Jouleplan" gewünscht...

VBasic 5 Control Creation Edition (CCE)

ftp://pcnews.at/freesoft/vb5cce/

Kostenlos verfügbare, stark reduzierte Version von Visual Basic 5.0. Die Dokumentation ist in vielen WinWord-Dokumenten ausdruckbar. Es fehlt das Lernprogramm und die Möglichkeit, eigenständige exe-Dateien zu erstellen. Es ist aber möglich, ActiveX-Steuerelemente zu erstellen. ca 13MB inklusive Doku.

vb5ccein.exe	VBasic 5.0 CCE
ccedoc01.exe	VBasic 5.0 CCE Doc1
ccedoc02.exe	VBasic 5.0 CCE Doc2
ccedoc03.exe	VBasic 5.0 CCE Doc3
ccedoc04.exe	VBasic 5.0 CCE Doc4
ccedoc05.exe	VBasic 5.0 CCE Doc5
ccedoc06.exe	VBasic 5.0 CCE Doc1
ccehelp.exe	VBasic 5.0 CCE Online Help

Visual Basic-Kurs in den PCNEWS-Ausgaben

Der Visual-Basic-Kurs als WinWord-Datei: http://pcnews.at/ins/prj/vb50.zip

PCNEWS	Seite	Einführung in Visual Basic 5.0
58	82	1. Einleitung, 2.Grundlagen der Programmierung
59		Anhang A: Befehlsübersicht, 3 Grafikprogrammierung in Visual Basic
60		Anhang B: ANSI-Zeichensatz 4 Das Standarddialogobjekt 5 Dateizugriff
62		6 Datenaustausch: OLE und DDE
65		7 Zugriff auf Datenbanken 8 ODBC (Open DataBase Connectivity)
63		9 Einbinden von EXE-Dateien in Visual Basic-Programme
64		10 Verwendung von Dynamic Link Libraries (DLLs)
64		11 Erstellung von Online-Hilfe-Dateien