VBasic 5.0

Christian Zahler

9 Einbinden von EXE-Dateien

```
Private Sub Form_Click ()
  Dim X
  ' Instanzen-Handle für das laufende
Programm
  AppActivate „Microsoft Visual Basic“
  ' Fokus auf Visual Basic
  SendKeys „%{ }{Down 3}{Enter}“, True
  ' Auf Symbolgröße verkleinern
  X = Shell(„c:\bp\bin\dbank.exe“, 1)
  ' externes Programm starten
  AppActivate „Microsoft Visual Basic“
  ' Fokus auf Visual Basic.
  SendKeys „%{ }{Enter}“, True
  ' Visual Basic auf Vollbild vergrößern.
  Unload Form1
  ' Programm beenden
End Sub
Wichtig ist der Befehl Shell, der ein
ausführbares Programm ausführt.
```

Syntax: `Shell(Befehlstext [, Fensterart])`

Befehlstext enthält den Namen des auszuführenden Programms und alle erforderlichen Argumente oder Parameter der Befehlszeile. Enthält der Dateiname in Befehlstext keine der Dateinamenerweiterungen *.COM, *.EXE, *.BAT oder *.PIF, wird *.EXE vorausgesetzt.

Fensterart bestimmt durch eine Zahl die Art des Fensters, in dem das Programm ausgeführt werden soll. Wird `Fensterart` nicht angegeben, wird das Programm standardmäßig in Symbolgröße geöffnet und gleichzeitig aktiviert. Dies entspricht `Fensterart = 2`.

Die nachfolgende Tabelle zeigt die möglichen Werte für Fensterart und die daraus resultierende Form des Fensters:

Wert	Fensterart
1, 5, 9	Normales Fenster mit Fokus
2	Symbol mit Fokus (Voreinstellung)
3	Vollbild mit Fokus
4, 8	Normales Fenster ohne Fokus
6, 7	Symbol ohne Fokus

Wenn die `Shell`-Funktion das angegebene Programm erfolgreich ausgeführt hat, liefert sie das Instanzen-Handle des gestarteten Programms. Das Instanzen-Handle ist eine eindeutige Kennung für das laufende Programm. Kann die `Shell`-Funktion das angegebene Programm nicht starten, tritt ein Fehler auf.

Hinweis: Die `Shell`-Funktion operiert andere Programme asynchron. Sie können sich daher nicht darauf verlassen, dass ein mit `Shell` begonnenes Programm mit der Ausführung fertig ist, bevor der Code, der der `Shell`-Funktion in Ihrer Visual Basic-Anwendung folgt, ausgeführt wird.

Die `AppActivate`-Anweisung verschiebt den Fokus zum genannten Anwendungsfenster. Sie hat jedoch keinen Einfluß darauf, ob das Fenster in Vollbild- oder Symbolgröße erscheint. Das Anwendungsfenster besitzt solange den Fokus, bis der Benutzer diesen einem anderen Fenster zuordnet oder das Fenster schließt.

Syntax: `AppActivate Titeltext`

Das Titeltext-Argument ist ein Zeichenfolgenausdruck, der den Namen in der Titelleiste des zu aktivierenden Anwendungsfensters bezeichnet. Ist die gewünschte Anwendung mehrmals vorhanden, dann wählt die Betriebsumgebung willkürlich eine Anwendung aus und aktiviert diese. Der Name, der in der Titelleiste erscheint, muss dem Namen in Titeltext genau entsprechen. Die Groß-/Kleinschreibung muss jedoch nicht übereinstimmen. Die Wörter „Rechner“ und „rechner“ sind also in diesem Sinne identisch.

`SendKeys` sendet Tastaturanschläge zur anderen Anwendung. Steuertasten werden mit geschwungenen Klammern dargestellt. % bedeutet die ALT -Taste, der Parameter `True` erzwingt ein sofortiges Senden.

10 Dynamic Link Libraries (DLLs)

Dynamic Link Libraries stellen eines der wichtigsten Konzepte von Windows dar. DLLs sind Bibliotheken, die aus Prozeduren bestehen. Anwendungen haben die Möglichkeit, auf diese Bibliotheken **zur Laufzeit** zuzugreifen (dynamisch). Das bedeutet, dass diese Bibliotheken unabhängig von der Anwendung aktualisiert werden können und dass mehrere Anwendungen auf dieselbe Bibliothek zugreifen können.

Arten von DLLs

- **Windows-DLLs:** Windows besteht selbst aus DLLs; die drei wesentlichsten sind:

Windows 3.1x:	Windows 95:
USER.EXE	USER32.DLL
KRNL386.EXE	KERNEL32.DLL
GDI.EXE	GDI32.DLL

  Diese drei Dateien befinden sich standardmäßig im `C:\Windows\SYSTEM-Verzeichnis`. Oft werden diese DLLs auch als **Windows-API** (= Application Programming Interface) bezeichnet.

- DLLs anderer Anwendungen
- eigene DLLs (können z.B. in C programmiert werden; in Visual Basic ist die Erstellung eigener DLLs *erst ab Version 5.0 Professional Edition möglich!*)
- MSVBVM50.DLL (frühere Versionen: `VB40032.DLL` bzw. `VB40016.DLL`, `VBRUN300.DLL`, `VBRUN200.DLL`, `VBRUN100.DLL`)

Wo findet man die Beschreibung der Systemfunktionen?

- Windows 3.1x (16 bit-Anwendungen): Dateien `WIN31API.TXT` und `WIN31API.HLP`
- Multimediafunktionen von Windows 3.1x: Datei `WINMMSYS.TXT`
- Windows 95 (32 bit-Anwendungen): Datei `WIN32API.TXT`

Es steht ein eigener **„API-Viewer“** zur Verfügung, mit dem die nötigen Declare-Statements einfach eingebunden werden können!

TIPP: Vor dem Testen der VB-Applikation ***unbedingt speichern***! Bei der Verwendung von DLL-Aufrufen kann es **sehr leicht** zu **Systemabstürzen** kommen!

Zugriff auf DLLs

Bevor eine Funktion einer DLL verwendet werden kann, muss sie in Visual Basic deklariert werden.

Beispiel: Anzeige von Systeminformationen

Modul

```
Option Explicit
Declare Function GetVersionEx Lib „kernel32"
Alias “GetVersionExA" (ByVal
lpVersionInformation As OSVERSIONINFO) As
Long
Type OSVERSIONINFO
        dwOSVersionInfoSize As Long
        dwMajorVersion As Long
        dwMinorVersion As Long
        dwBuildNumber As Long
        dwPlatformId As Long
        szCSDVersion As String * 128      '
Maintenance string for PSS usage
End Type
```

Form

```
Dim OSInfo as OSVERSIONINFO
Private Sub Form_Click()
  Dim Wert As Long
  Wert = GetVersionEx(OSInfo)
  OSInfo.dwOSVersionInfoSize = 148
  OSInfo.szCSDVersion = Space (128)
  Wert = GetVersionEx(OSInfo)
  Print OSInfo.dwMajorVersion
  Print OSInfo.dwMinorVersion
  Print OSInfo.dwBuildNumber
  Print OSInfo.dwPlatformID
End Sub
```

Anmerkung: Beschäftigen Sie sich doch einmal mit dem Zusatzsteuerelement `SysInfo.OCX`!