Befehlsreferenz - Interaktion: SendKeys-Anweisung

**VB-Versionen:**   http://www.vbarchiv.net/images/16x16/yes.gif VB4   |   http://www.vbarchiv.net/images/16x16/yes.gif VB5   |   http://www.vbarchiv.net/images/16x16/yes.gif VB6

Beschreibung:  
Die **SendKeys**-Anweisung sendet eine Tastenfolge (dies können auch mehrere Tastenanschläge sein) an das aktive Fenster, als ob sie ganz normal über die Tastatur eingegeben worden wären.

Syntax:

SendKeys (tastenfolge,  *[wait]* )

Parameter:

|  |  |
| --- | --- |
| tastenfolge | (String) Zeichenfolge (Tastenfolge), die gesendet werden soll. |
| wait | **Optional.** (Boolean) *False* (Voreinstellung), wenn die Anwendung sofort nach dem Senden der Tastenfolge fortgesetzt werden soll. *True*, wenn die Anwendung warten soll, bis die Tastenfolge verarbeitet wurde. |

**Besonderheiten**  
Das Pluszeichen (+), Caret-Zeichen (^), Prozentzeichen (%), die Tilde (~) und die Klammern ( ) haben bei der SendKeys-Anweisung eine spezielle Bedeutung. Um diese verwenden zu können, müssen Sie sie in geschweifte Klammern setzen.  
  
**Sondertasten** (Tastenkombinationen)

|  |  |
| --- | --- |
| **Taste** | **Tastencode** |
| Rücktaste | {BACKSPACE}, {BS} oder {BKSP} |
| Pause | {BREAK} |
| Feststelltaste | {CAPSLOCK} |
| Entf | {DELETE} oder {DEL} |
| Nach unten, Nach oben | {DOWN}, {UP} |
| Nach links, Nach rechts | {LEFT}, {RIGHT} |
| Ende | {END} |
| Eingabetaste | {ENTER} |
| ESC | {ESC} |
| Hilfe | {HELP} |
| Pos1 | {HOME} |
| Einfg | {INSERT} oder {INS} |
| NUM-Feststelltaste | {NUMLOCK} |
| Bild auf, Bild ab | {PGUP}, {PGDN} |
| Rollen-Festelltaste | {SCROLLLOCK} |
| TAB | {TAB} |
| F1, F2, ... F16 | {F1}, {F2}, ... {F16} |

Beispiel:

' Notepad starten und Dialog "Seite einrichten" aufrufen

Dim AppID As Long

AppID = Shell("c:\winnt\system32\notepad.exe", vbNormalFocus)

DoEvents

' NotePad aktivieren

AppActivate AppID

' Alt+d (Menü DATEI)

SendKeys "%d", True

' r (Seiten einrichten)

SendKeys "r", True

' Alt+k (Kopfzeile)

SendKeys "%k", True

' Text schreiben

SendKeys "Test-Kopfzeile", True

' Dialog beenden (OK-Schaltfläche per Alt+O auslösen)

SendKeys "%o"

' oder anstelle OK, Dialog per Alt+F4 schließen

SendKeys "%{F4}"

**Anmerkung**  
Die DRUCK-Taste (PRTSC) kann von der **SendKeys**-Anweisung **nicht** simuliert werden.

|  |  |
| --- | --- |
| **Siehe auch:** | http://www.vbarchiv.net/images/pfeil_rot.gif[AppActivate-Anweisung](http://www.vbarchiv.net/commands/details.php?id=appactivate)  http://www.vbarchiv.net/images/pfeil_rot.gif[Shell-Funktion](http://www.vbarchiv.net/commands/details.php?id=shell) |

Befehlsreferenz - Interaktion: AppActivate-Anweisung

**VB-Versionen:**   http://www.vbarchiv.net/images/16x16/yes.gif VB4   |   http://www.vbarchiv.net/images/16x16/yes.gif VB5   |   http://www.vbarchiv.net/images/16x16/yes.gif VB6

Beschreibung:  
Mit der **AppActivate**-Anweisung können andere Anwendungsfenster aktiviert werden.

Syntax:

AppActivate (Titel,  *[Wait]* )

Parameter:

|  |  |
| --- | --- |
| Titel | (String) Titel der Anwendung. Dieser erscheint im Normalfall in der Titelleiste (Caption-Eigenschaft einer Form). Man kann auch die von der Shell-Funktion zurückgegebene AppID angeben. Es reicht, wenn man ein Wort oder die ersten paar Buchstaben aus dem Titel angibt, dies kann aber auch zu unerwünschten Ergebnissen führen. |
| Wait | **Optional.** (Boolean) Wenn *True*, dann wird mit der Ausführung der Anweisung gewartet, bis die Anwendung den Fokus hat. Wenn *False* (Voreinstellung), wird der Befehl sofort ausgeführt. |

Beispiel:

' Alle 3 Aufrufe aktivieren die Visual Basic Entwicklungsumgebung,

' wenn kein anderes Fenster mit dem gleichen Namen im Text

' (z.B. "Visual Basic-Hilfe") existiert.

AppActivate "Visual"

AppActivate "Visual Basic"

AppActivate "Vis"

|  |  |
| --- | --- |
| **Siehe auch:** | http://www.vbarchiv.net/images/pfeil_rot.gif[Shell-Anweisung](http://www.vbarchiv.net/commands/details.php?id=shell)  http://www.vbarchiv.net/images/pfeil_rot.gif[SendKeys-Anweisung](http://www.vbarchiv.net/commands/details.php?id=sendkeys) |

Befehlsreferenz - Dateisystem: Shell-Funktion

**VB-Versionen:**   http://www.vbarchiv.net/images/16x16/yes.gif VB4   |   http://www.vbarchiv.net/images/16x16/yes.gif VB5   |   http://www.vbarchiv.net/images/16x16/yes.gif VB6

Beschreibung:  
Die **Shell**-Funktion startet eine Anwendung (ausführbare Datei, z.B. .EXE) und gibt die TaskID zurück. Konnte die Anwendung nicht ausgeführt werden, wird Null zurückgegeben.

Syntax:

Shell (Filename,  *[WindowStyle]* )

Parameter:

|  |  |
| --- | --- |
| Filename | (String) Legt den Dateinamen der zu startenden Anwendung fest (ink. Pfadangabe). Wird kein Pfad angegeben, wird versucht die Anwendung aus dem aktuellen Verzeichnis zu starten. |
| WindowStyle | **Optional.** (Konstante) Legt fest, wie die Anwendung ausgeführt werden soll. Wenn WindowStyle nicht angegeben wird, erhält das Programm den Fokus und wird im minimierten Zustand gestartet. |

Mögliche Werte für **WindowStyle** sind:

|  |  |  |
| --- | --- | --- |
| **vbHide** | 0 | Das Fenster ist ausgeblendet, und das ausgeblendete Fenster erhält den Fokus. |
| **vbNormalFocus** | 1 | Das Fenster hat den Fokus, und die ursprüngliche Größe und Position wird wiederhergestellt. |
| **vbMinimizedFocus** | 2 | Das Fenster wird als Symbol mit Fokus angezeigt. |
| **vbMaximizedFocus** | 3 | Das Fenster wird maximiert mit Fokus angezeigt. |
| **vbNormalNoFocus** | 4 | Die zuletzt verwendete Größe und Position des Fensters wird wiederhergestellt. Das momentan aktive Fenster bleibt aktiv. |
| **vbMinimizedNoFocus** | 6 | Das Fenster wird als Symbol angezeigt. Das momentan aktive Fenster bleibt aktiv. |

Wird die **Shell**-Funktion erfolgreich ausgeführt, d.h. die Anwendung konnte gestartet werden, wird die TaskID der Anwendung zurückgegeben. Mittels der TaskID lässt sich dann jederzeit die gestartete Anwendung über Ihr Programm wieder aktivieren (natürlich nur solange es noch geöffnet ist).

Beispiel:

' Windows-Editor starten

Dim TaskID As Long

TaskID = Shell("c:\winnt\system32\notepad.exe", vbNormalNoFocus)

' Weitere Ausführung Ihres Programms

...

' Windows-Editor anhand TaskID aktivieren

AppActivate TaskID

' Windows-Editor schließen

SendKeys "%{F4}", True

|  |  |
| --- | --- |
| **Siehe auch:** | http://www.vbarchiv.net/images/pfeil_rot.gif[AppActivate-Anweisung](http://www.vbarchiv.net/commands/details.php?id=appactivate)  http://www.vbarchiv.net/images/pfeil_rot.gif[SendKeys-Anweisung](http://www.vbarchiv.net/commands/details.php?id=sendkeys) |

### Visual Basic Quellcode

Public Class Form1

Private Declare Function SetCursorPos Lib "user32" (ByVal X As Integer, ByVal Y As Integer) As Integer

Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Dim x As Integer = 100

Dim y As Integer = 50

SetCursorPos(x, y)

End Sub

End Class

**Visual Basic Quellcode**

Windows.Forms.Cursor.Position = New Point(600, 600)

'API-Funktion deklarieren  
Public Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long  
  
Public Type POINTAPI 'Variablentyp deklarieren  
   X As Long  
   Y As Long  
End Type

Dim CursorPos As POINTAPI 'Variable deklarieren  
  
Call GetCursorPos(CursorPos) 'API-Funktion aufrufen

Nun enthalten die Variablen "CursorPos.X" und "CursorPos.Y" die jeweilien Koordinaten. Beachten Sie bitte, das diese Werte **in Pixeln** angegeben sind. Um diese in die standardmäßig in VB verwendeten Twips umzurechnen, schauen Sie sich bitte den Tipp "[Pixel in Twips und umgekehrt](http://www.shadoware.de/vb/pixeltwips.html)" an.

#### Auslesen der aktuellen Mausposition

**Problem:** Die aktuelle Mausposition soll ausgelesen werden.

StandardModule: basMain

[Private](http://www.herber.de/mailing/vb/html/vakeyprivate.htm) [Declare](http://www.herber.de/mailing/vb/html/vastmdeclare.htm) [Function](http://www.herber.de/mailing/vb/html/vastmfunction.htm) GetCursorPos Lib "user32" (lpPoint [As](http://www.herber.de/mailing/vb/html/vakeyas.htm) POINTAPI) [As](http://www.herber.de/mailing/vb/html/vakeyas.htm) [Long](http://www.herber.de/mailing/vb/html/vadatlong.htm)

[Private](http://www.herber.de/mailing/vb/html/vakeyprivate.htm) [Type](http://www.herber.de/mailing/vb/html/vastmtype.htm) POINTAPI

x [As](http://www.herber.de/mailing/vb/html/vakeyas.htm) [Long](http://www.herber.de/mailing/vb/html/vadatlong.htm)

y [As](http://www.herber.de/mailing/vb/html/vakeyas.htm) [Long](http://www.herber.de/mailing/vb/html/vadatlong.htm)

[End](http://www.herber.de/mailing/vb/html/vastmend.htm) [Type](http://www.herber.de/mailing/vb/html/vastmtype.htm)

[Sub](http://www.herber.de/mailing/vb/html/vastmsub.htm) WhereAmI()

[Dim](http://www.herber.de/mailing/vb/html/vastmdim.htm) pTargetPoint [As](http://www.herber.de/mailing/vb/html/vakeyas.htm) POINTAPI

[Dim](http://www.herber.de/mailing/vb/html/vastmdim.htm) lRetVal [As](http://www.herber.de/mailing/vb/html/vakeyas.htm) [Long](http://www.herber.de/mailing/vb/html/vadatlong.htm)

lRetVal = GetCursorPos(pTargetPoint)

MsgBox "Meine [Position](http://www.herber.de/mailing/vb/html/xlproposition.htm):" & vbLf & \_

pTargetPoint.x & "," & pTargetPoint.y

[End](http://www.herber.de/mailing/vb/html/vastmend.htm) [Sub](http://www.herber.de/mailing/vb/html/vastmsub.htm)

mouseover, mousemove, mousedown, mousepointer, mouseup, mouse click, mouse position

|  |  |  |
| --- | --- | --- |
| Maus positionieren und Klick simulieren  Mit Hilfe dieser Routine lassen sich Mausklicks an beliebiger Stelle simulieren. | | |
| **Autor:**  http://www.vbarchiv.net/images/16x16/user.gif [Torsten Kerz](http://www.vbarchiv.net/tipps/index.php?id=autor114) | **Bewertung:**  http://www.vbarchiv.net/images/16x16/star1.gifhttp://www.vbarchiv.net/images/16x16/star1.gifhttp://www.vbarchiv.net/images/16x16/star1.gifhttp://www.vbarchiv.net/images/16x16/star1.gifhttp://www.vbarchiv.net/images/16x16/star.gif   [ [Jetzt bewerten](javascript:showhide('voting');) ] | **Views:**  37.088 |
| ohne Homepage | **System:**  Win9x, WinNT, Win2000, WinXP, Vista | http://www.vbarchiv.net/images/16x16/cd.gif [Beispielprojekt auf CD](http://www.vbarchiv.net/home/vbcdvol5.php) |

In einem früheren Tipp haben wir Ihnen schon einmal gezeigt, wie man einen Mausklick per Programmcode simulieren kann, so als wäre tatsächlich auf eine Maustaste gedrückt worden. Unser heutiger Tipp zeigt zusätzlich noch, wie man den Mauszeiger an eine ganz bestimmte Position auf dem Bildschirm positionieren kann.

' zunächst die benötigten API-Deklarationen

Private Declare Sub mouse\_event Lib "user32" ( \_

ByVal dwFlags As Long, \_

ByVal dx As Long, \_

ByVal dy As Long, \_

ByVal cButtons As Long, \_

ByVal dwExtraInfo As Long)

Private Const MOUSEEVENTF\_ABSOLUTE = &H8000

Private Const MOUSEEVENTF\_LEFTDOWN = &H2

Private Const MOUSEEVENTF\_LEFTUP = &H4

Private Const MOUSEEVENTF\_MIDDLEDOWN = &H20

Private Const MOUSEEVENTF\_MIDDLEUP = &H40

Private Const MOUSEEVENTF\_MOVE = &H1

Private Const MOUSEEVENTF\_RIGHTDOWN = &H8

Private Const MOUSEEVENTF\_RIGHTUP = &H10

**Maus positionieren und Mausklick simulieren**

' Mausklick simulieren mit optionaler Angabe der

' Koordinaten, an welche der Mauszeiger autom.

' positioniert werden soll

Public Sub Mausklick(Optional Button As \_

MouseButtonConstants = vbLeftButton, \_

Optional XPos As Long = -1, \_

Optional YPos As Long = -1)

' Mauszeiger positionieren

If XPos <> -1 Or YPos <> -1 Then

mouse\_event MOUSEEVENTF\_ABSOLUTE + MOUSEEVENTF\_MOVE, \_

XPos / Screen.Width \* 65535, \_

YPos / Screen.Height \* 65535, 0, 0

End If

' Mausklick simulieren

Select Case Button

' linke Maustaste

Case vbLeftButton

mouse\_event MOUSEEVENTF\_LEFTDOWN, 0, 0, 0, 0

mouse\_event MOUSEEVENTF\_LEFTUP, 0, 0, 0, 0

' mittlere Maustaste

Case vbMiddleButton

mouse\_event MOUSEEVENTF\_MIDDLEDOWN, 0, 0, 0, 0

mouse\_event MOUSEEVENTF\_MIDDLEUP, 0, 0, 0, 0

' rechte Maustaste

Case vbRightButton

mouse\_event MOUSEEVENTF\_RIGHTDOWN, 0, 0, 0, 0

mouse\_event MOUSEEVENTF\_RIGHTUP, 0, 0, 0, 0

End Select

End Sub

**Beispiele**

' Maus auf dem Bildschirm zentrieren ohne Klick

Mausklick 0, Screen.Width / 2, Screen.Height / 2

' Mit rechts auf die Bildschirmmitte klicken

Mausklick vbRightButton, Screen.Width / 2, \_

Screen.Height / 2

' Klicken, ohne Positionsveränderung

Mausklick vbRightButton

Option Explicit  
  
Private Declare Function SetCursorPos Lib "user32.dll" ( \_  
    ByVal x As Long, \_  
    ByVal y As Long) As Long  
  
Private Declare Sub Sleep Lib "kernel32.dll" ( \_  
    ByVal dwMilliseconds As Long)  
  
Private Declare Function GetCursorPos Lib "user32.dll" ( \_  
    ByRef lpPoint As POINTAPI) As Long  
  
Private Type POINTAPI  
    x As Long  
    y As Long  
End Type  
  
**Public Sub maus\_spazieren\_Fahren()**  
Dim myPos As POINTAPI  
Dim Ziel\_X As Long  
Dim Ziel\_Y As Long  
Dim L As Long  
Ziel\_X = 10  
Ziel\_Y = 100  
GetCursorPos myPos  
If myPos.x >= Ziel\_X Then  
    For L = myPos.x To Ziel\_X Step -1  
        Sleep 5  
        SetCursorPos L, myPos.y  
    Next  
    Else:  
    For L = Ziel\_X To myPos.x  
        Sleep 5  
        SetCursorPos L, myPos.y  
    Next  
End If  
If myPos.y >= Ziel\_Y Then  
    For L = myPos.y To Ziel\_Y Step -1  
        Sleep 5  
        SetCursorPos Ziel\_X, L  
    Next  
    Else:  
    For L = myPos.y To Ziel\_Y  
        Sleep 5  
        SetCursorPos Ziel\_X, L  
    Next  
End If  
**End Sub**

<nobr>Option Explicit  
Private Declare Sub Sleep Lib "kernel32.dll" ( \_  
    ByVal dwMilliseconds As Long)  
  
Private Declare Function GetCursorPos Lib "user32.dll" ( \_  
    ByRef lpPoint As POINTAPI) As Long  
  
Private Declare Function SetCursorPos Lib "user32.dll" ( \_  
    ByVal x As Long, \_  
    ByVal y As Long) As Long  
  
Private Declare Function GetAsyncKeyState Lib "user32.dll" (ByVal vKey As Long) As Long  
  
Private Type POINTAPI  
    x As Long  
    y As Long  
End Type  
  
Const VK\_LBUTTON = &H1 ' Linker Mausbutton  
Const VK\_RBUTTON = &H2 ' Rechter Mausbutton  
Const KEYEVENTF\_KEYUP = &H2 ' Die angegebene Taste wird losgelassen  
  
  
  
  
**Public Sub maus\_spazieren\_Fahren()**  
Dim myPos As POINTAPI  
Dim Ziel\_X As Long  
Dim Ziel\_Y As Long  
Dim L As Long  
Dim Click  
Ziel\_X = 10  
Ziel\_Y = 100  
GetCursorPos myPos  
If myPos.x >= Ziel\_X Then  
    For L = myPos.x To Ziel\_X Step -1  
        Sleep 5  
        SetCursorPos L, myPos.y  
    Next  
    Else:  
    For L = Ziel\_X To myPos.x  
        Sleep 5  
        SetCursorPos L, myPos.y  
    Next  
End If  
If myPos.y >= Ziel\_Y Then  
    For L = myPos.y To Ziel\_Y Step -1  
        Sleep 5  
        SetCursorPos Ziel\_X, L  
    Next  
    Else:  
    For L = myPos.y To Ziel\_Y  
        Sleep 5  
        SetCursorPos Ziel\_X, L  
    Next  
End If  
Click = GetAsyncKeyState(&H1)  
MsgBox Click  
**End Sub**  
  
</nobr>

1. Benutze "Early Binding" (google mal danach) oder programmiere Outlook-Dinge erstmal in Outlook, da gibt es dann auch Hilfe dazu. Und, lies die Outlook-VBA-Hilfe zu GetInspector.   
     
   2. Was willst Du damit überhaupt erreichen? In Excel warten bis Outlook gesendet hat??   
     
   Falls ja, vergiss es. Mache das eher so:   
     
   <http://www.office-loesung.de/ftopic74117_0_0_asc.php>   
   <http://www.office-loesung.de/ftopic56207_0_0_asc.php>   
   <http://www.office-loesung.de/ftopic106897_0_0_asc.php>   
   <http://www.office-loesung.de/ftopic267822_0_0_asc.php>

Hi,   
  
die Display-Funktion von Outlook besitzt einen Parameter namens "Modal", den man auf True setzen kann. In dem Fall wartet VBA mit der Weiterverarbeitung, bis klar ist, was der Benutzer getan hat (Mail gesendet oder nicht gesendet).   
  
Wenn man Modal auf TRUE setzt kann man anschließend die Eigenschaft Sent des Mailitems auswerten und entscheiden, ob ein Kommentar geschrieben werden soll.   
  
Ungetestet etwa so:

|  |
| --- |
| **Code:** |
| Sub MailBodyDialog()   Dim rng As Range  Dim olApp  As Object  Dim olMail As Object  Dim bolSent As Boolean  Dim KoNr, Im, Absender, Betreff As String   On Error GoTo ErrHandler       Betreff = Cells(Selection.Row, 11)      If Selection.Rows.Count > 1 Or MsgBox("Wollen Sie ein E-Mail bez. " & Betreff & " schicken?" & vbNewLine & vbNewLine & \_      Cells(Selection.Row, 2) & " - " & Cells(Selection.Row, 6), vbYesNo, Betreff) = vbNo Then Exit Sub           KoNr = Cells(Selection.Row, 2)      Im = Cells(Selection.Row, 4)           Application.ScreenUpdating = False      Set rng = Range("A8:M8,A" & Selection.Row & ":M" & Selection.Row)           Set olApp = CreateObject("Outlook.Application")      Set olMail = olApp.CreateItem(0)      bolSent = False           With olMail          .HtmlBody = RangetoHTML(rng)          .to = "claudia.wesenauer@derrenner.at" 'Empfänger          '.cc = "mail1@server.de,mail2@server.de,mail3@server.de" 'optional Kopie an          '.bcc = "mail1@server.de,mail2@server.de,mail3@server.de" 'optional Blindkopie an          '.ReadReceiptRequested = True ' optional Lesbestätigung anfordern          .Subject = Betreff & " " & KoNr & " Import " & Im & " - Bitte um Lieferschein Nr.!"          .Display True          '.Send                   On Error Resume Next          bolSent = olMail.Sent          If Err.Number <> 0 Then            bolSent = True            Err.Clear          End If          On Error GoTo ErrHandler               End With           If bolSent Then        With Cells(Selection.Row, 11)          On Error Resume Next          .AddComment          On Error GoTo ErrHandler          .Comment.Visible = False          .Comment.Text Text:="E-Mail " & Betreff & " an WES geschickt" & Chr(10) & "am " & Date & " durch " & ADName() & Chr(10), Start:=Len(.Comment.Text) + 1          .Comment.Shape.TextFrame.AutoSize = True        End With      End If           ErrExit:    On Error Resume Next    Application.ScreenUpdating = True    Set rng = Nothing    Set olApp = Nothing    Set olMail = Nothing    Exit Sub   ErrHandler:    MsgBox Err.Description, vbCritical, "Fehler " & Err.Number & " aufgetreten!"    Resume ErrExit           End Sub |

HTH,   
Eric

Private Declare

Sub mouse\_event Lib "user32" \_

(ByVal dwFlags As Long, ByVal dx As Long, \_

ByVal dy As Long, ByVal cButtons As Long, \_

ByVal dwExtraInfo As Long)

Public Const MOUSE\_LEFT = 0

Public Const MOUSE\_MIDDLE = 1

Public Const MOUSE\_RIGHT = 2

'Die nachfolgende Prozedur simuliert den gewünschten Mausklick.

Public

Sub SendMausklick(ByVal mButton As Long)

Const MOUSEEVENTF\_LEFTDOWN = &H2

Const MOUSEEVENTF\_LEFTUP = &H4

Const MOUSEEVENTF\_MIDDLEDOWN = &H20

Const MOUSEEVENTF\_MIDDLEUP = &H40

Const MOUSEEVENTF\_RIGHTDOWN = &H8

Const MOUSEEVENTF\_RIGHTUP = &H10

If (mButton = MOUSE\_LEFT) Then

Call mouse\_event(MOUSEEVENTF\_LEFTDOWN, 0, 0, 0, 0)

Call mouse\_event(MOUSEEVENTF\_LEFTUP, 0, 0, 0, 0)

ElseIf (mButton = MOUSE\_MIDDLE) Then

Call mouse\_event(MOUSEEVENTF\_MIDDLEDOWN, 0, 0, 0, 0)

Call mouse\_event(MOUSEEVENTF\_MIDDLEUP, 0, 0, 0, 0)

Else

Call mouse\_event(MOUSEEVENTF\_RIGHTDOWN, 0, 0, 0, 0)

Call mouse\_event(MOUSEEVENTF\_RIGHTUP, 0, 0, 0, 0)

End If

End Sub