

Kapitel II

Aktive Symbolleisten

In diesem Kapitel:

Symbolleisten mit dynamischen Schaltflächen

2

Zusammenfassung

7

In Kapitel 5 werden die Grundlagen zum Word-Objektmodell vorgestellt. Das Kapitel 8 befasst sich unter anderem mit dem Einbinden von Applikationsereignissen in das Projekt. Dieses Lösungsbeispiel baut auf den nachfolgenden Objekten und Ereignissen auf.

Im Blickpunkt:

API-Funktion

GetKeyState

Word-Ereignisse

DocumentChange, WindowSelectionChange

Word-Objekte

CommandBars.Controls

ActiveDocument.PrintOut, ActiveDocument.AutoHyphenation

Application.StatusBar

Selection.Type, Selection.ShapeRange, Selection.InlineShapes, Selection.Tables

Im Gegensatz zu den integrierten Symbolleisten von Word sind die benutzerdefinierten Symbolleisten passiv. Dies bedeutet, dass die einzelnen Symbolleistenschaltflächen, ohne zutun des Entwicklers, stets aktiviert sind. Dies ist auch dann der Fall, wenn die hinterlegten Funktionen gar nicht zum Kontext der Einfügemarke passen und somit eigentlich nicht gestartet werden können.

Dieses Lösungsbeispiel zeigt, wie aus einer passiven Symbolleiste eine dem Kontext entsprechende Symbolleiste aufgebaut wird, deren Status der einzelnen Symbolleistenschaltflächen sich dynamisch verändert. Die einzelnen Symbolleistenschaltflächen werden in Abhängigkeit von der Position der Einfügemarke aktiviert bzw. deaktiviert.

Symbolleisten mit dynamischen Schaltflächen

Das vorliegende Beispiel baut auf einer statischen Symbolleiste auf, die in eine Dokumentvorlage eingebunden wurde. Die Dokumentvorlage wird anschließend als Add-In geladen. Bei Bedarf kann die Symbolleiste auch dynamisch beim Starten von Word angelegt werden. Dies ist in Kapitel 16 beschrieben. Das Einbinden einer Dokumentvorlage als Add-In wird in Kapitel 1 näher erläutert.

In der Datei *BspII_Symbolleiste.dot* ist neben der eigentlichen Symbolleiste die Logik zur Steuerung der kontextabhängigen Symbolleistenschaltflächen und die eigentlichen Makros hinterlegt.

Abbildg. II.1 Symbolleiste *DasHandbuch_BspII* mit aktivierten bzw. deaktivierten Symbolleistenschaltflächen



HINWEIS

Damit die integrierte Symbolleiste in allen Dokumenten zur Verfügung steht, muss die Datei *Bspll_Symbolleiste.dot* in den *Startup*-Ordner von Word kopiert werden. Nach dem erneuten Aufruf der Applikation steht die Symbolleiste automatisch am rechten Rand des Programmfensters zur Verfügung.

Zusammenspiel der einzelnen Module

Um auf Ereignisse reagieren zu können, muss eine entsprechende Klasse in das Projekt eingebunden werden. Eine neue Instanz dieser Klasse wird dann beim Starten von Word angelegt. In dieser Klasse wird die »Überwachung« der Ereignisse bearbeitet.

Klassenmodul

Damit die Symbolleiste mit aktiven Symbolleistenschaltflächen versehen werden kann, muss das Add-In auf die Ereignisse der Applikation reagieren können. In Kapitel 8 wird gezeigt, wie diese aktiviert werden können.

Das Programmmodul


Innerhalb des Programmmoduls muss an geeigneter Stelle sichergestellt werden, dass eine neue Instanz der entsprechenden Klasse mit den hinterlegten Applikationsereignissen angelegt wird. Wie eine neue Instanz auf diese Klassen erzeugt wird, ist in Kapitel 8 detailliert beschrieben.

Um beim Laden des Add-Ins die automatische Überwachung der Ereignisse zu gewährleisten, ist ein Makro mit der Bezeichnung *AutoExec* implementiert. Dieses wird automatisch von Word abgearbeitet, sobald das Add-In aktiviert wird. Die Funktionsweise der Auto-Makros ist in Kapitel 8 behandelt.

Die Ereignisse

Durch das Einbinden des Schlüsselworts *WithEvents* in die Deklaration der globalen Variable innerhalb des Klassenmoduls werden die Ereignisse aus einer ActiveX-Komponente überwacht. In unserem Fall ist dies die Komponente *Word.Application*.

Damit eine aktive Symbolleiste erzeugt werden kann, sind in erster Line zwei Ereignisse von Bedeutung:

- Das *DocumentChange*-Ereignis wird ausgelöst, wenn eine Datei geöffnet oder geschlossen wird. Es wird auch ausgelöst, wenn zwischen zwei verschiedenen Dokumenten hin und her gewechselt wird.
- Das *WindowSelectionChange*-Ereignis wird ausgelöst, wenn die Einfügemarke verschoben wird. Dies ist in erster Linie dann der Fall, wenn die Einfügemarke mittels Pfeiltasten oder eines Mausklicks verschoben wird. Ein Wechseln der Zellen innerhalb einer Tabelle mittels der -Taste löst das Ereignis ebenfalls aus.

Das Ereignis wird jedoch nicht ausgelöst, wenn Text erfasst wird. Aus Sicht von Word wird in diesem Fall die Einfügemarke nicht verschoben, da diese immer *vor* dem gleichen Zeichen steht.

Diese beiden Ereignisse reichen aus, um die Symbolleiste »zum Leben zu erwecken« und die einzelnen Symbolleistenschaltflächen dem Kontext der Einfügemarke entsprechend zu aktivieren bzw. zu deaktivieren.

Diskussion zum Code

Die eigentliche Funktionalität hinter den einzelnen Symbolleistenschaltflächen wurde dreistufig aufgebaut:

- Die erste Stufe ist die allgemeine Prozedur, die den aktuellen Kontext der Einfügemarke auswertet und die Symbolleistenschaltflächen entsprechend aktiviert bzw. deaktiviert.
- Die zweite Stufe dient der Fehlerbehandlung. Hier wird überprüft, ob die entsprechende Symbolleistenschaltfläche zu Recht aktiviert war. Wurden alle Prüfungen erfolgreich bestanden, wird das eigentliche Makro aufgerufen.
- In der dritten Stufe ist das eigentliche Makro, also die gewünschte Funktionalität, der Symbolleistenschaltfläche hinterlegt.

Durch das dreistufige Konzept können die einzelnen Stufen problemlos ergänzt oder angepasst werden, ohne dass dies einen Einfluss auf die restliche Funktionalität des Add-Ins hat.

So kann beispielsweise die Funktion zum Ausdrucken eines Entwurfs erweitert werden, indem die Papierzufuhr beim Ausdruck aus einem bestimmten Papierschacht erfolgt. Auf die vorherigen Prüfungen, ob überhaupt ein Ausdruck erfolgen kann, hat dies keinen Einfluss.

HINWEIS

Die Fehlerbehandlung in der zweiten Stufe wird bewusst eingebaut, da die einzelnen Ereignisse nicht in jedem Fall so ausgeführt werden, wie sich dies der Programmierer wünscht.

Als besonderer Stolperstein kann sich ab und zu das `WindowSelectionChange`-Ereignis erweisen, denn dieses wird beim Erfassen des Textes nicht ausgelöst.

Kontext der Einfügemarke auswerten – Stufe 1

In Listing II.1 wird der Kontext der Einfügemarke analysiert. Für jeden benötigten Fall wird eine eigene Variable gesetzt. Am Schluss der Prozedur wird den einzelnen Symbolleistenschaltflächen die zugehörige Kombination von Variablen zugewiesen.

Listing II.1

Prozedur zum Auswerten des Kontextes der Einfügemarke und Definieren der Symbolleistenschaltflächen

```
Private Sub procSchaltflächeAktivieren()
    Dim objCContext As Object
    Dim bDocumentOffen As Boolean
    Dim bDocumentOhneSchutz As Boolean
    Dim bTabelleMarkiert As Boolean
    Dim bShapeMarkiert As Boolean
    Dim bInlineShapeMarkiert As Boolean

    'Ist ein Document geöffnet?
    If (Application.Windows.Count > 0) Then
        bDocumentOffen = True

    'Ist Document geschützt?
    If (ActiveDocument.ProtectionType = wdNoProtection) Then
        bDocumentOhneSchutz = True
    End If

    'Ist eine Tabelle markiert?
```

Listing II.1 Prozedur zum Auswerten des Kontextes der Einfügemarke und Definieren der Symbolleistenschaltflächen (Fortsetzung)

```

    If (Selection.Tables.Count > 0) Then
        bTabelleMarkiert = True
    End If

    'Ist ein Shape markiert?
    If Selection.Type = wdSelectionShape Then
        bShapeMarkiert = True
    End If

    'Ist ein InlineShape markiert?
    If Selection.Type = wdSelectionInlineShape Then
        bInlineShapeMarkiert = True
    End If
End If

'Schaltflächen ein-/ausschalten
Set objCContext = Application.CustomizationContext
Application.CustomizationContext = ThisDocument

With CommandBars(cbrSYMBOLLEISTE).Controls
    .Item(cbbDOKUMENT_DRAFT_DRUCK).Enabled = bDocumentOffen
    .Item(cbbDOKUMENT_FELDFUNKTION).Enabled = bDocumentOffen And bDocumentOhneSchutz
    .Item(cbbDOKUMENT_SILBENTRENNUNG).Enabled = bDocumentOffen And bDocumentOhneSchutz
    .Item(cbbTABELLE_FORMATIEREN).Enabled = bDocumentOffen And bDocumentOhneSchutz _
        And bTabelleMarkiert
    .Item(cbbSHAPE_BREITE_SETZEN).Enabled = bDocumentOffen And bDocumentOhneSchutz _
        And (bShapeMarkiert Or bInlineShapeMarkiert)
    .Item(cbbSHAPE_VERANKERN).Enabled = bDocumentOffen And bDocumentOhneSchutz _
        And bShapeMarkiert
End With
Application.CustomizationContext = objCContext

'Add-In als gespeichert kennzeichnen
ThisDocument.Saved = True
End Sub

```

PROFITIPP

Ist in einem Add-In eine Symbolleiste eingebunden, wird manchmal beim Beenden von Word die Rückfrage gestellt, ob das entsprechende Add-In gespeichert werden soll. Dies geschieht vor allem dann, wenn, wie in unserem Beispiel, durch das Programm selber oder durch den Anwender die Symbolleiste geändert wurde.

Um diese Rückfrage zu verhindern, wird nach erfolgter Manipulation der Symbolleiste das Add-In als bereits gespeichert gekennzeichnet:

```
ThisDocument.Saved = True
```

Um sicherzustellen dass die besagte Rückfrage unter keinen Umständen gestellt wird, wurde zusätzlich ein Makro mit der Bezeichnung *AutoExit* eingebaut. Dieses Makro wird automatisch abgearbeitet, sobald Word beendet wird.

Wird das Add-In vom Entwickler geändert und er beendet Word, ohne das Add-In zuvor manuell gespeichert zu haben, gehen sämtliche Änderungen verloren, da das erwähnte Makro die erwartete Rückfrage unterdrückt und die Datei geschlossen wird, ohne es zu speichern.

Mögliche Fehlerbehandlung – Stufe 2

In Listing II.2 ist eine Programmsequenz aufgezeigt, die für die Fehlerbehandlung einer einzelnen Symbolleistenschaltfläche zuständig ist. Das Makro *ShapeVerankern* wurde der Symbolleistenschaltfläche zugewiesen. Für jede Symbolleistenschaltfläche wurde eine eigene Prozedur, mit ähnlicher Funktionalität, aufgebaut.

Listing II.2 Fehlerprüfung für die Symbolleistenschaltfläche *Autoform verankern ein/aus*

```
Public Sub ShapeVerankern()
    Dim cbtn As Office.CommandBarButton

    Set cbtn = CommandBars(cbrSYMBOLLEISTE).Controls.Item(cbbSHAPE_VERANKERN)
    cbtn.Enabled = False

    If Windows.Count = 0 Then
        Beep
    ElseIf Not (ActiveDocument.ProtectionType = wdNoProtection) Then
        MsgBox "Aktuelles Dokument ist geschützt." & vbCr & _
            "Verankerung der Shape-Objekte kann nicht geändert werden.", _
            vbOKOnly + vbExclamation, "Warnung"
    ElseIf (Not Selection.Type = wdSelectionShape) Then
        Beep
    Else
        fktShapeVerankern
        cbtn.Enabled = True
    End If
End Sub
```

Nur wenn die Prüfung auf mögliche Fehler erfolgreich war, wird die eigentliche Funktion, welche die markierten Shape-Objekte verankert, aufgerufen und die zugehörige Symbolleistenschaltfläche wieder aktiviert:

```
fktShapeVerankern
cbtn.Enabled = True
```

Ausführen der gewünschten Funktion – Stufe 3

In Listing II.3 wird der letzte Schritt ausgeführt. In dieser Funktion wurde die eigentliche Programmsequenz für die Symbolleistenschaltfläche hinterlegt.

Listing II.3 Die Programmzeilen zur Symbolleistenschaltfläche *Autoform verankern ein/aus*

```
Public Function fktShapeVerankern()
    Dim shp As Word.Shape
    Dim bFlag As Boolean

    bFlag = True
    For Each shp In Selection.ShapeRange
        bFlag = bFlag And shp.LockAnchor
    Next shp

    Selection.ShapeRange.LockAnchor = Not bFlag
```

Listing II.3 Die Programmzeilen zur Symbolleistenschaltfläche *Autoform verankern ein/aus* (Fortsetzung)

```
Application.StatusBar = "Verankerung bei " & CStr(fktShapesInShapeRangeZählen) & _
    " Shape-Objekten auf '" & CStr(Not bFlag) & "' gesetzt."
End Function
```

Die Funktion *fktShapeVerankern* verhält sich ähnlich wie andere Standardfunktionen von Word (beispielsweise markierten Text auf Fett setzen). Mittels einer Schleife wird zuerst der Status aller markierten Shape-Objekte ermittelt und in der Variable *bFlag* zwischengespeichert. Solange bei allen Objekten die Verankerung aktiviert ist, bleibt *bFlag* auf True. Ist jedoch nur bei einem einzelnen Objekt die Verankerung nicht aktiviert, so wird *bFlag* auf False gesetzt:

```
For Each shp In Selection.ShapeRange
    bFlag = bFlag And shp.LockAnchor
Next shp
```

In einem zweiten Schritt wird der umgekehrte Wert der *LockAnchor*-Eigenschaft zugewiesen. Auf diese Weise ist sichergestellt, dass allen markierten Shape-Objekten der gleiche Status der *LockAnchor*-Eigenschaft zugewiesen wird:

```
Selection.ShapeRange.LockAnchor = Not bFlag
```

HINWEIS

Bei den aufgeführten Listings handelt es sich nur um eine Auswahl von Programmsequenzen, die benötigt werden, um die Funktionalität der sechs Symbolleistenschaltflächen zu ermöglichen.

Es würde jedoch den Rahmen dieses Buches sprengen, wenn jede Prozedur abgebildet und detailliert erläutert würde. Die restlichen Programmzeilen können direkt in der Beispieldatei *Bspll_Symbolleiste.dot* eingesehen werden.



Das dargestellte Beispiel mit der aktiven Symbolleiste finden Sie in der Beispieldatei *Bspll_Symbolleiste.dot* auf der CD-ROM zum Buch im Ordner *\Beispiele\KapII*.

Zusammenfassung

In diesem Lösungsbeispiel wurde gezeigt, wie Ereignisse genutzt werden können, um die Schaltflächen einer Symbolleiste dem Kontext entsprechend aktivieren bzw. deaktivieren zu können:

- Es wurde dargestellt, wie ein Klassenmodul aufgebaut sein muss, um auf die Ereignisse reagieren zu können, und wie eine Instanz dieser Klasse in das Projekt eingebunden wird (Seite 3).
- Außerdem wurde erklärt, weshalb es sinnvoll ist, die eigentliche Funktionalität von einer zusätzlichen Fehlerbehandlung zu entkoppeln (Seite 4).

