Word mit VBA noch leistungsfähiger gemacht

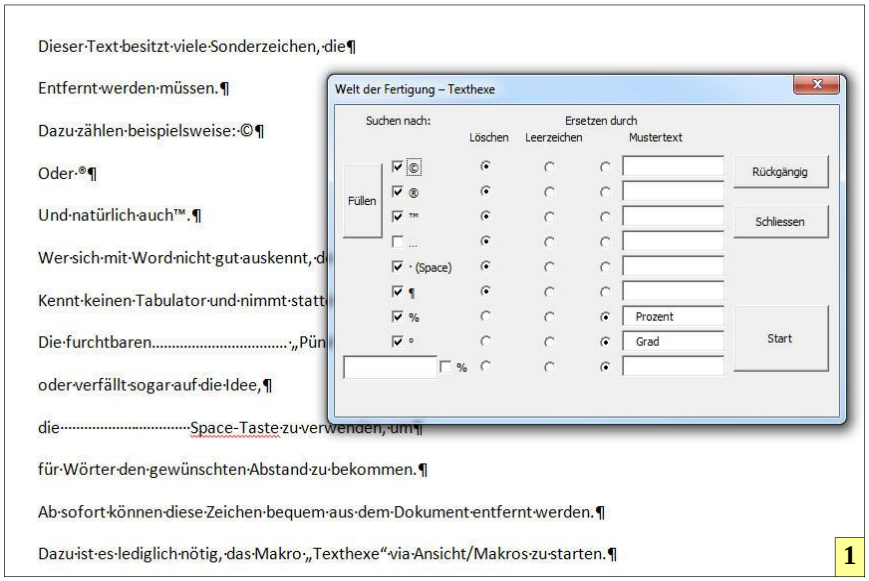
Mit eigenen VBA-Makros den Turbo in Word zünden

Die in Microsoft Office integrierte Programmiersprache VBA (Visual Basic for Applications) ist bestens geeignet, nicht im Office-Befehlsumfang vorhandene Funktionen flugs selbst zu schreiben. Wer sich mit dieser Sprache ein wenig anfreundet, wird im Laufe der Zeit erstaunt feststellen, das sich damit völlig neue Anwendungen auftun.

Wer Word oder Excel intensiv nutzt, kann sich über einen Mangel an Befehlen und Funktionalitäten eigentlich nicht beklagen. Dennoch gibt es immer mal wieder Fälle, in denen man gerne einen ganz bestimmten Befehl zusätzlich hätte. Auch bereits Vorhandenes findet nicht immer die Zustimmung des Users.

Zum Beispiel ist die Suchen und Ersetzen-Funktion von Word alles andere als einfach handhabbar, wenn es darum geht, etwa bestimmte Sonderzeichen zu entfernen oder Großbuchstaben in Kleinbuchstaben umzuwandeln. In diesen Fällen lohnt ein Griff in die Programmier-Kiste, da via VBA sich rasch entsprechende Funktionen erstellen lassen. VBA ist nicht besonders schwer zu erlernen, da dahinter ein Basic-Dialekt steht, der über Schlüsselwörter verfügt, die auf der englischen Sprache fusen. So lässt beispielsweise der Befehl ›MsgBox „Heute ist der „ & Date‹ ein Fenster erscheinen, in dem der genannte Text und das aktuelle Datum ausgegeben werden. ›Msg‹ ist die Abkürzung für Message, was Nachricht bedeutet.

Besagter Befehl blendet also ein Nachrichtenfenster ein, in dem der genannte Text steht. Das ›&‹-Zeichen ist eine logische UND-Verknüpfung, die das aktuelle Datum mit dem vorherstehenden Text verknüpft, sodass beides in einer Zeile ausgegeben wird. Um ein neues Makro einzugeben, muss man sich zunächst in die VBA-Entwicklungsumgebung begeben, was über die Tastenkombination ALT+F11 rasch erledigt ist. Sobald man sich dort befindet, gilt es, ein neues Modul anzulegen. Dazu wird aus dem Menü ›Einfügen‹ der Befehl ›Modul‹ gewählt. Nach dem Erstellen des Moduls wird es im Projekt-Baum dargestellt.



1 Die Suchfunktion von Word ist zwar sehr leistungsstark, doch alles andere als einfach in der Handhabung. Was liegt da näher, als sich eine eigene Suchen-und Ersetzen-Funktion zu schneidern? Dieses Projekt vermittelt einen tiefen Einblick in VBA und macht Lust auf mehr.

In das auf diese Weise erzeugte Befehlsfenster sind nun diejenigen Befehle einzugeben, die den Weg beschreiben, um automatisiert zum gewünschten Resultat zu gelangen. Das ist unter Umständen eine längere Tipperei, die zudem Fehleranfällig ist. Besser ist es, für ein Programmgerüst den Makro-Rekorder einzusetzen und Feinheiten von Hand nachzutragen. Dies hat zudem den Vorteil, dass man auch als VBA-Einsteiger relativ rasch zu ganz beachtlichen Makros kommt.

Der Makro-Rekorder wird über den nach unten weisenden Pfeil im Button ›Makros‹ zugänglich. Ein Klick auf ›Makro aufzchn.‹ startet den Aufzeichnungsvorgang. Ist alles aufgezeichnet, genügt der gleiche Weg, um die Aufzeichnung zu beenden. Das so erstellte VBA-Programm kann nun nach eigenen Wünschen abgeändert werden.

Ein Dialog mit Qualität

Natürlich sind nicht alle Makros dergestalt programmiert, dass man sie sofort starten könnte. Vielfach sind vor dem Ablauf des Makros Attribute festzulegen, die den Programmlauf steuern. Damit dies komfortabel erfolgen kann, gibt es in VBA die Möglichkeit, eigene Dialogfelder zu erstellen. Dazu ist über Einfügen/UserForm ein Fenster zu erzeugen, das mit Buttons, Texten oder Auswahlfeldern nach eigenen Wünschen gestaltet werden kann.

Diese Elemente sind über das Einblenden eines Werkzeugkastens nutzbar, der über Bearbeiten/Werkzeugkasten aktiviert wird. Damit dieses selbstgestaltete Fenster in Office genutzt werden kann, muss eine Möglichkeit geschaffen werden, es aufzu-

rufen. Dies geschieht über den Befehl ›UserForm2.show‹, der in ein Modul geschrieben werden muss. Die Zahl in diesem Befehl ist fortlaufend, sodass beliebige Fenster eingeblendet werden können. Das Modul muss nun noch einen passenden Namen bekommen, der dann automatisch in der Auswahlliste der zur Verfügung stehenden Makros auftaucht.

Tricks

Die Elemente aus dem Werkzeugkasten haben vorab festgelegte Eigenschaften, die jederzeit geändert werden können, was sinnigerweise im Eigenschaften-Fenster erfolgt. Zum Beispiel kann die Eigenschaft ›Caption‹ durch überschreiben des vorgegebenen Textes ein passender Text eingesetzt werden. Über die Zeichenfolge ALT+0174 kann beispielsweise das ®-Zeichen erzeugt werden, wenn nach diesem Zeichen gesucht werden soll, um es aus dem Text automatisch zu löschen.

Die Elemente können zueinander sauber ausgerichtet werden. Zu diesem zweck sind diese anzuklicken, während die STRG-Taste gedrückt wird. Dadurch werden alle Elemente, die zusammen ausgerichtet werden sollen, zusammengefasst. Ein Rechtsklick ermöglicht dann den Zugriff auf die Ausrichtoptionen, die sich stets nach dem letzten selektierten Element richten. Damit OptionButtons sich nicht gegenseitig stören, beziehungsweise, damit mehrere zusammengehörende Optionbuttons erzeugt werden können, müssen diese den gleichen GroupName haben, sind als zu einer Gruppe zusammenzuschließen. Dazu können die fraglichen Optionbuttons bei gedrückter STRG-Taste zusammen markiert und benannt werden. Dadurch wird es nun möglich, mehreren Optionbuttons den Wert ›True‹ im Feld ›Value‹ zuzuweisen. Dieser Effekt zeigt sich in den schwarzen Punkten, die die Aktivität der Buttons anzeigen.

Abfrage If then Else

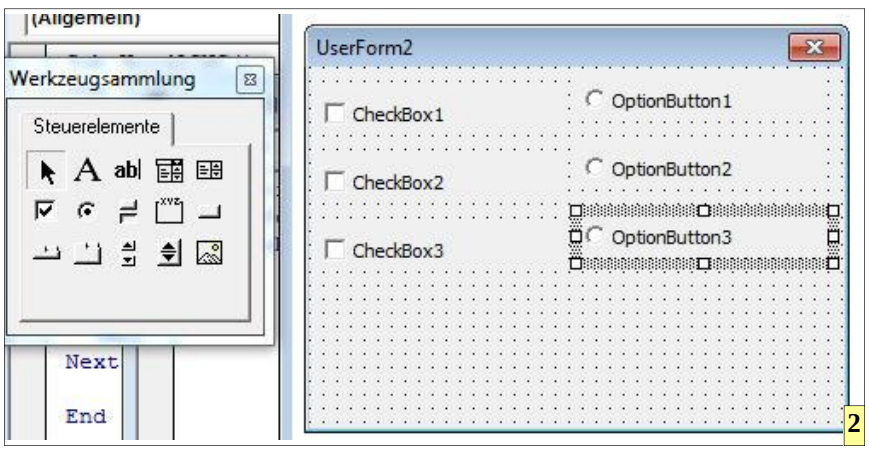
Beim Programmieren einer Abfrage mit If Then Else ist peinlich genau darauf zu achten, den Befehlstext in die richtige Reihenfolge zu bringen, da sonst Fehlermeldung ungewöhnlicher Art produziert werden. Zum Beispiel ist unbedingt darauf zu achten, dass alles, was nach Then folgt, in eine neue Zeile geschrieben wird. Zudem ist jede If Then-Sequenz mit End If abzuschließen. Beispiel:

```
If OptionButton3.Value = True Then
    Neutext1 = " " ' Kommentar
    ElseIf OptionButton24.Value = True Then
    Neutext1 = TextBox1.Text ' Kommentar
    End If
```

Wie anhand des Beispiels ersichtlich ist, werden Kommentare mit einem Hochkomma eingeleitet. Grundsätzlich sollte Makros ausführlich dokumentiert werden, damit später kein Rätselraten über dessen Funktion aufkommt.

UserForm öffnen und schliessen

Sobald eine UserForm angelegt ist, soll diese in der Regel geöffnet werden, sobald das Makro startet. Dies erledigt der Befehl ›UserForm1.Show‹.



2 Aus einem Werkzeugkasten lassen sich verschiedenste Steuerelemente zu einem passenden „Cockpit“ für die spätere Textmanipulation zusammenstellen.

Ist die Arbeit erledigt, könnte man das Fenster mit dem X-Button schließen. Schöner ist es allerdings, dies mit einem eigenen Button direkt im selbsterstellten Formular zu bewerkstelligen. Dazu wird einfach ein neuer Button vom Werkzeugfenster in die UserForm gezogen und per Doppelklick darauf ein neuer Makro-Kopf erzeugt. Hier muss lediglich der Befehl ›Unload Me‹ eingetragen werden und schon kann über diesen Button die UserForm geschlossen werden.

Dokument wieder herstellen

Gerade bei umfangreichen Manipulationen am Text passiert es schon einmal, dass das Ergebnis nicht den Erwartungen entspricht. In Word gibt es dazu eine Undo-Funktion. Leider ist es mittels VBA nicht direkt möglich, so einen Befehl zu nutzen.

Es gibt jedoch auch dazu einen Trick: Vor der eigentlichen Manipulation wird das Word-Dokument in der Zwischenablage gespeichert und kann von dort jederzeit wieder hervorgeholt werden, sodass sich Fehler schnell ausbügeln lassen.

Die Zwischenablage lässt sich mit wenigen Befehlen ansprechen:

```
Selection.WholeStory ( Text markieren )
Selection.Copy ( Text inZwischenablage)
```

Im Bedarfsfall wird das Dokument mit diesem Befehl wieder hergestellt:

```
Selection.Paste
```

Es ist problemlos möglich, VBA-Programme auch auf andere Rechner zu installieren und zu nutzen. Dazu sind lediglich folgende Dateien zu sichern:

Das VBA-Programm mit der Endung .BAS und die UserForm-Dateien mit den Endungen .frm und .frx.

Dabei ist anzumerken, dass die frx-Datei automatisch mit der frm-Datei erzeugt wird, wenn die UserForm-Datei exportiert wird.

Die Dateien mit den Endungen .Bas

und .frm sind lesbare Dateien, die mit dem Windows-Editor gelesen werden können und den eigentlichen VBA-Code enthalten.

Sichern des Programms und Überspielen auf einen anderen PC

Die genannten Dateien werden in ein Verzeichnis etwa eines USB-Sticks kopiert und danach per Import-Funktion des VBA-Editors auf den Zielrechner überspielt. Ab sofort sind auch für diesen Rechner die VBA-Makros verfügbar.

Zum Exportieren muss die gewünschte Datei angeklickt werden, denn es wird ausschließlich die blau markierte Datei exportiert. Wird darauf nicht geachtet, kann es passieren, dass die falsche Datei den Weg auf den Stick nimmt.

Beim Importieren, was direkt vom USB-Stick vorgenommen werden kann, kann es passieren, dass es eine Fehlermeldung gibt, wenn das neue Makro und ein bereits auf dem Rechner Vorhandenes den gleichen Namen besitzen. Daher unbedingt darauf achten, einen eindeutigen, mit Sicherheit nicht zweimal vorkommenden Namen für Module und UserForms zu verwenden. Dies kann im Eigenschaften-Feld vorgenommen werden.

Dazu wird im Feld mit der Bezeichnung (Name) der automatisch vergebene Standardname geändert. Für die UserForm bietet sich der Name ›UserFormTexthexe‹ an. Wichtig: Der Name darf keine Leerzeichen enthalten. Zudem muss unbedingt darauf geachtet werden, dass der Aufruf der UserForm mit dem neuen Namen geschieht. Das muss manuell vorgenommen werden, da dies nicht automatisch geschieht. Es muss also ›UserForm2.show‹ in ›UserFormTexthexe.show‹ geändert werden. Der Name für das Modul wird auf die gleiche Weise geändert, sodass es beim Import keine Probleme mehr geben dürfte.

Damit der Import von VBA-Makros möglich ist, muss bereits eines vorhanden sein, welches man dann mit ›Bearbeiten‹ öffnen kann. Nur bei geöffneten VBA-Editor bekommt man Zugriff auf die Import-Funktion. Sollte noch kein Makro am Zielrechner vorhanden sein, erstellt man einfach eines. Dazu wird schlicht ein beliebiger Makro-Name eingegeben und der Button ›Erstellen‹ betätigt. Die eigentliche Import-Funktion findet sich unter ›Datei‹. Der Import ist genauso einfach, wie das Öffnen einer Word-Datei. Man muss nicht einmal auf die Endungen achten, da der VBA-Editor die einzelnen Dateien anhand ihrer Endung erkennt und richtig zuordnet.

Wenn ein Makro nicht mehr gewünscht ist, so kann es rückstandlos gelöscht werden. Die erfolgende Abfrage bezüglich der vorherigen Sicherung sollte man nicht leichtfertig überspringen. Ebenso einfach ist die UserForm gelöscht, sodass sich der PC danach genauso präsentiert, als hätte er nie mit den selbst erstellten Makros Kontakt gehabt. Natürlich kann dieses VBA-Programm nur als Grundlage dienen, selbst Hand anzulegen, um für die eigenen Zwecke passende Funktionen einzubinden.

Ein Vorschlag wäre, einen Button einzubauen, der das rasche Setzen und Löschen der Checkboxen gestattet. Diese Funktion benötigt nur wenig Zeitaufwand, um implementiert zu werden. Dazu wird zunächst ein Button etwa neben die Checkboxen platziert.

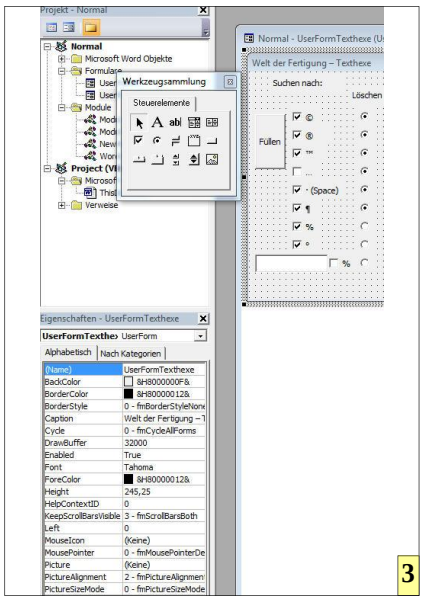
Steuern leicht gemacht

Die Steuerung übernimmt eine globale Variable, die über zwei If Then Else-Abfragen unterschiedliche Ereignisse auslöst. Einmal sollen alle Checkboxen deaktiviert und beim nächsten Druck wieder aktiviert werden. Umgesetzt wird dies über die Eigenschaften-Variable ›Value‹, die wechselweise die logischen Zustände True und False erhält. Der dazugehörende Programmtext sieht so aus:

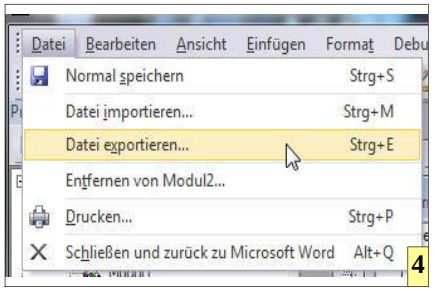
```
If Fuellen = 0 Then
  CheckBox1.Value = False
  CheckBox2.Value = False
  CheckBox3.Value = False
  CheckBox4.Value = False
  CheckBox5.Value = False
  CheckBox6.Value = False
  CheckBox7.Value = False
  CheckBox8.Value = False
  CheckBox9.Value = False
  Fuellen = 1
  ElseIf Fuellen = 1 Then
  CheckBox1.Value = True
  CheckBox2.Value = True
  CheckBox3.Value = True
  CheckBox4.Value = True
  CheckBox5.Value = True
  CheckBox6.Value = True
  CheckBox7.Value = True
  CheckBox8.Value = True
  CheckBox9.Value = True
  Fuellen = 0
  End If
```

Wie man sieht, werden hier zwei Blöcke wechselweise durchlaufen, je nachdem, welchen Wert die Variable ›Fuellen‹ gerade besitzt. Am Ende jeden Unterprogramms wird der Wert von Fuellen geändert, damit der Ablauf korrekt funktioniert.

Nur die Fantasie setzt Grenzen, was das Aufbohren dieses Beispiels betrifft. Gutes Gelingen!



3



4

3 Den einzelnen Elemente können über die Eigenschaften-Palette sehr einfach Merkmale, wie etwa ›aktiv‹ oder ›sichtbar‹ zugewiesen werden.

4 Per Export kann das VBA-Makro auf einen Stick kopiert werden, von wo es auf andere Rechner übertragen werden kann.