Für alle Programm-versionen von 2007 bis 2013

michael KOFLER
ralf NEBELO

EXCEL PROGRAMMIEREN

ABLÄUFE AUTOMATISIEREN, APPS UND ANWENDUNGEN ENTWICKELN

HANSER

Kofler/Nebelo

Excel programmieren

Michael Kofler
Ralf Nebelo

Excel programmieren

Abläufe automatisieren, Apps und Anwendungen entwickeln mit Excel 2007 bis 2013

HANSER

Die Autoren:

Michael Kofler, Graz
Ralf Nebelo, Bocholt

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Brigitte Bauer-Schiewek
Copy editing: Petra Kienle, Fürstenfeldbruck
Herstellung: Irene Weilhart
Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München
Umschlagrealisation: Stephan Rönigk
Layout: Manuela Treindl, Fürth
Druck und Bindung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-43866-8
E-Book-ISBN: 978-3-446-43912-2

Inhalt

Vorwort

Excel bietet von Haus aus ein riesiges Spektrum von Funktionen. Wozu sollten Sie dann noch selber Makros, Add-ins, Apps und andere Programmerweiterungen mit VBA, Visual Studio oder anderen Werkzeugen entwickeln? Weil Sie damit ...

- ... eigene Tabellenfunktionen programmieren können, die einfacher anzuwenden sind als komplizierte Formeln.
- ... Excel nach Ihren Vorstellungen konfigurieren und auf diese Weise eine einfachere und effizientere Programmbedienung erreichen können.
- ... komplexe Arbeitsschritte wie etwa das Ausfüllen von Formularen durch „intelligente" Formulare (alias Mustervorlagen) strukturieren und erleichtern können.
- ... immer wieder auftretende Arbeitsvorgänge automatisieren können. Das empfiehlt sich besonders dann, wenn regelmäßig große Datenmengen anfallen, die verarbeitet, analysiert und grafisch aufbereitet werden sollen.
- ... eigenständige und leistungsfähige Excel-Lösungen erstellen können, die sich durch maßgeschneiderte Bedienelemente nahtlos in das Menüband, die sogenannte Backstage-Ansicht (im Datei-Menü) oder andere Teile der Excel-Oberfläche integrieren.

  Damit lassen sich Excel-Anwendungen in ihrer Bedienung so weit vereinfachen, dass sie von anderen Personen (auch von Excel-Laien) ohne lange Einweisung verwendet werden können.

Das notwendige Know-how für alle diese Formen der Excel-Programmierung finden Sie in diesem Buch. *Übrigens: Auch wenn Excel 2013 auf dem Titel steht, so gilt das Gesagte – oder besser: Geschriebene – doch für alle Programmversionen ab 2007 (und zum größten Teil auch für die Versionen davor).*

Wenn es Dinge gibt, die in einer älteren Version anders funktionieren als in Excel 2013, so wird das ausdrücklich erwähnt. Falls das wider Erwarten einmal nicht der Fall sein sollte, bitten wir schon jetzt um Verzeihung. Bei so vielen Versionen verlieren auch erfahrene Autoren manchmal den Überblick.

Neues in Excel

Mit der radikal neuen Multifunktionsleiste, die die früheren Menüs und Symbolleisten plötzlich sehr alt aussehen ließ (und letztlich in Rente schickte), war Excel 2007 eine echte Revolution. Excel 2010 ließ es entwicklungstechnisch deutlich ruhiger angehen und bescherte uns statt einer großen *Re*volution viele kleine *E*volutionen.

Eine davon war der neue *Oberflächeneditor*, mit dem wir nicht mehr nur die unscheinbare „Symbolleiste für den Schnellzugriff" nach unseren Wünschen konfigurieren dürfen, sondern

die komplette Multifunktionsleiste. Die heißt nun übrigens *„Menüband“* (siehe Abschnitt 8.2) und beschränkt sich auf solche Befehle, die der Bearbeitung von Dokumentinhalten dienen. Für alle anderen Befehle, die das Dokument als Ganzes betreffen (Speichern, Drucken etc.), hat Microsoft die sogenannte *Backstage-Ansicht* (siehe Abschnitt 8.5) erfunden, die das Office-Menü von Excel 2007 ersetzt. Menüband und Backstage-Ansicht bilden seither die Kommandozentrale von Excel und zeichnen sich durch eine konsequente Aufgabenteilung aus.

Konsequenz zeigte Microsoft auch bei der *Oberflächenprogrammierung*. Hier gilt seit Excel 2010 für alle Bestandteile – Menüband, Backstage-Ansicht, Symbolleiste für den Schnellzugriff *und* Kontextmenüs – das gleiche „duale Prinzip“: XML-Code bestimmt das Design, VBA-Code die Funktion. Mit dem Know-how, das Sie sich womöglich schon bei der Anpassung der früheren Multifunktionsleiste erworben haben, können Sie jetzt also die gesamte Excel-Oberfläche verändern und eigene Lösungen integrieren (Kapitel 8).

Evolutionär präsentierte sich Excel 2010 auch bei der Visualisierung von Zahlen. So fanden die *SmartArt-Diagramme* (Abschnitt 10.7), die mit der Version 2007 eingeführt wurden, Eingang in das Objektmodell, so dass man sie nun programmatisch erstellen oder verändern kann. Darüber hinaus hat Excel 2010 der Welt die sogenannten *Sparklines-Diagramme* (Abschnitt 10.6) beschert, ein seinerzeit völlig neuer und ebenfalls programmierbarer Diagrammtyp, der in eine einzelne Zelle passt und sich insbesondere für die Visualisierung von Trends eignet.

Wo Licht ist, ist bekanntlich auch Schatten. Und das gilt insbesondere für die Tatsache, dass es Excel seit der Version 2010 auch in einer *64-Bit-Version* zu kaufen gibt. Dass die nicht nur Vorteile hat, sondern auch massive Nachteile in Form von diversen Inkompatibilitäten, zeigt der Abschnitt 15.7 (und was Sie dagegen tun können, natürlich auch).

Und was bringt uns Excel 2013? Aus Anwendersicht zunächst mal einen nüchternen, von Schatten und Transparenzeffekten befreiten Look, der sich an der Optik von Windows 8 orientiert. Und dazu passend eine neuerlich aufgeräumte und entschlackte Menüband- und Backstage-Oberfläche, in der man so manchen Befehl aus früheren Versionen leider nicht mehr findet.

Neue Funktionen wie Schnellanalyse und Empfohlene Diagramme beschleunigen die Erstellung von Diagrammen. Arbeitsmappen lassen sich nun standardmäßig „in der Cloud“ und somit online speichern, manche Diagramme in animierter Form anzeigen und Pivot-Tabellen auf der Basis mehrerer Listen beziehungsweise Tabellen generieren. Unter der Haube gibt es die eine oder andere Tabellenfunktion zu entdecken, unter anderem für das direkte Anzapfen von Webdiensten (siehe Abschnitt 15.4).

Der wichtigste und aus Entwicklersicht interessanteste Neuzugang aber ist die *App für Office*. Dabei handelt es sich um ein völlig neues Erweiterungskonzept, das Webtechniken an die Stelle von VBA-Makros setzen möchte. Wie (und ob) das funktioniert, ist detailliert im Kapitel 15.9 beschrieben.

Warum dieses Buch?

Im Gegensatz zu anderen Titeln, die sich mit dem Thema Excel-Programmierung beschäftigen, liefert Ihnen dieses Buch *keine* systematische Beschreibung von Objekten, ihren Eigenschaften und Methoden oder VBA-Befehlen. Wer so etwas sucht, ist mit der Hilfefunktion des VBA-Editors und mit zahlreichen Internetquellen besser bedient.

Anstelle einer umfassenden Referenz stehen bei diesem Buch praktische Lösungen und Anwendungsmöglichkeiten im Vordergrund. Die zugehörigen Code-Beispiele lassen sich relativ leicht an eigene Bedürfnisse anpassen, was die Entwicklungszeit für manches berufliche oder private Programmiervorhaben spürbar verkürzen kann.

Dass man bei praxisbezogenen Projekten natürlich auch sehr viel über Objekte (die wichtigsten sogar!), vor allem aber über sinnvolle Formen ihres programmierten Zusammenarbeitens erfährt, ist quasi ein Nebeneffekt. Gleichzeitig nennen wir aber auch die Probleme Excels beim Namen, um Ihnen die langwierige Suche nach Fehlern zu ersparen, die Sie gar nicht selbst verursacht haben.

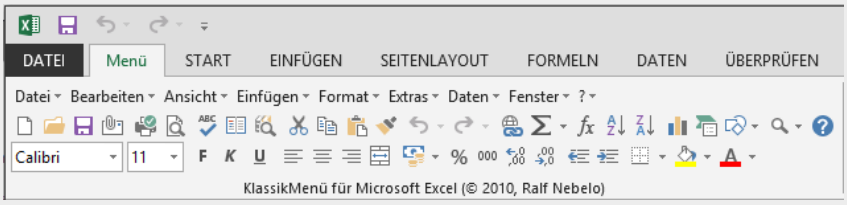
Neben konkreten Programmierlösungen liefert Ihnen dieses Buch aber auch sehr viel Insider-Wissen über die Bedienung von Excel. Damit werden Sie so ganz nebenbei zum „Power-User“ und können so manches Anwendungsproblem manuell lösen, für das Sie womöglich sonst ein Programm geschrieben hätten ... ;-)

Jenseits von VBA

Obwohl VBA immer noch das wichtigste Werkzeug für die Entwicklung von Excel-Lösungen ist (und daher im Mittelpunkt dieses Buchs steht), stellen sich zunehmend mehr Aufgaben, die mit der „eingebauten“ Programmiersprache des Kalkulationsprogramms nur noch teilweise oder gar nicht mehr zu lösen sind. Beispiele sind etwa die Anpassung von Menüband (siehe Abschnitt 8.2.2) und Backstage-Ansicht (8.5.1), die Programmierung individueller Aufgabenbereiche (15.8.4.1), die Abfrage von Web Services (15.4) oder die Integration von Webtechniken in Form der neuen Office-Apps (15.9).

Damit Sie solche Aufgaben dennoch meistern können, stellt Ihnen dieses Buch die erforderlichen Werkzeuge vor und liefert Ihnen das notwendige Know-how für den erfolgreichen Einsatz. Das erspart Ihnen mühsame Recherchen im Internet, den Kauf weiterer Bücher und lässt Sie mitunter auch kleine programmiertechnische „Wunder“ vollbringen – die Wiederbelebung der mit Excel 2003 „entschlafenen“ Menüs und Symbolleisten (siehe Abschnitt 8.2.5) beispielsweise. Darüber dürften sich insbesondere altgediente Excel-Anwender freuen, die sich im Menüband immer noch nicht zurechtfinden.

Tipp



Die Datei *KlassikMenü.xlam* im Unterordner 8 der Beispieldateien enthält eine vollständige Nachbildung der Menü- und Symbolleiste von Excel 2003. Sie können diese Datei als sofort nutzbares Add-in in Excel ab Version 2007 einbinden. Abschnitt 8.2.6 verrät, wie Sie dazu vorgehen müssen.

Viel Erfolg!

Die Beispiele dieses Buchs zeigen, wie weit Excel-Programmierung gehen kann. Die Möglichkeiten sind beinahe unbegrenzt! Wer sie nutzen will, muss sich aber nicht mehr nur im komplexen Objektmodell von Excel und in VBA zurechtfinden, sondern zunehmend auch in angrenzenden Programmierwelten.

Dabei will Ihnen dieses Buch eine praktische Orientierungshilfe sein. Mit zunehmender Übersicht und Erfahrung beginnt dann die Office-Programmierung mit VBA, Visual Studio, XML und diversen anderen Werkzeugen richtig Spaß zu machen.

Und wenn das der Fall ist, lässt auch der gewünschte Erfolg nicht lange auf sich warten. Genau den wünschen wir Ihnen von Herzen!

Michael Kofler und *Ralf Nebelo*, August 2013

http://www.kofler.info

Konzeption des Buchs

Visual Basic für Applikationen (oder kurz: VBA) ist eine sehr leistungsfähige Programmiersprache. Die große Zahl von Schlüsselwörtern bringt aber auch viele Probleme mit sich. Während des Einstiegs ist es so gut wie unmöglich, auch nur halbwegs einen Überblick über VBA zu gewinnen. Und selbst nach monatelanger Programmierung mit VBA wird die Hilfe der wichtigste Ratgeber zu den Details eines bestimmten Schlüsselworts bleiben. Dieses Buch versucht deswegen ganz bewusst, das zu bieten, was in der Originaldokumentation bzw. in der Hilfe zu kurz kommt:

- detaillierte Informationen für die Entwicklung eigener VBA-Lösungen und deren Integration in Menüband, Backstage-Ansicht und andere Bestandteile der Excel-Oberfläche
- „echte“ Anwendungen in Form von konkreten, realitätsbezogenen Beispielen
- themenorientierte Syntaxzusammenfassungen (z. B. alle Eigenschaften und Methoden zur Bearbeitung von Zellbereichen)
- aussagekräftige Beschreibungen der wichtigsten Objekte von VBA und ihre Einordnung in die Objekthierarchie

Darüber hinaus liefert Ihnen dieses Buch sehr viel Know-how für fortgeschrittene Programmierthemen, bei denen VBA nicht unbedingt im Mittelpunkt steht:

- Einsatz von DLL-Funktionen
- ActiveX-Automation
- Programmierung eigener Add-ins
- Verwendung von Web Services
- 64-Bit-Programmierung
- Realisierung von Office-Anwendungen mit den Visual Studio Tools for Office (VSTO)
- Entwicklung von Office-Apps mit Webtechnologien

Einem Anspruch wird das Buch aber ganz bewusst nicht gerecht: dem der Vollständigkeit. Es erscheint uns sinnlos, Hunderte von Seiten mit einer Referenz aller Schlüsselwörter zu füllen, wenn Sie beinahe dieselben Informationen auch in der Hilfe finden können. Anstatt zu versuchen, auch nur den Anschein der Vollständigkeit zu vermitteln, haben wir uns bemüht, wichtigeren Themen den Vorrang zu geben und diese ausführlich, fundiert und praxisorientiert zu behandeln.

Formalitäten

Die Namen von Menüs, Befehlsregisterkarten, Symbolen, Buttons und anderer Dialog- und Oberflächenelemente werden in Kapitälchen dargestellt: Datei|Öffnen, Abbruch oder OK. Die Anweisung Überprüfen|Blatt schützen|Zellen formatieren meint, dass Sie zuerst die Befehlsregisterkarte Überprüfen öffnen, den Befehl Blatt schützen anklicken und im daraufhin erscheinenden Dialog das Kontrollkästchen Zellen formatieren auswählen sollen.

VBA-Schlüsselwörter, Variablen- und Prozedurnamen sowie Datei- und Verzeichnisnamen werden *kursiv* angegeben, etwa *Application*-Objekt, *Visible*-Eigenschaft, *strName*-Variable oder *C:\Muster.xlsm*. Tabellenfunktionen wie *WENN()* erscheinen in der gleichen Schrift, aber in Großbuchstaben. (Tabellenfunktionen sind auch anhand der Sprache von VBA-Schlüsselwörtern zu unterscheiden: VBA-Schlüsselwörter sind grundsätzlich englisch, Tabellenfunktionsnamen immer deutsch.)

Beispielcode, Beispieldateien

Aus Platzgründen sind in diesem Buch immer nur die wichtigsten Code-Passagen der Beispielprogramme abgedruckt. Den vollständigen Code finden Sie auf der beiliegenden CD. Die Beispieldateien sind in Verzeichnissen angeordnet, deren Namen den Kapitelnummern entsprechen. VBA-Code in diesem Buch beginnt immer mit einem Kommentar im Format Verzeichnis\Dateiname, der auf die entsprechende Beispieldatei verweist:

```
' 01\format.xlsm
Sub FormatAsResult()
   Selection.Style = "result"
End Sub
```

Im Fall von XML-Code (den Sie hauptsächlich im Kapitel 8 finden) haben die Kommentare die folgende Form:

```
<!-- 08\Menüband_Button.xlsm -->
```

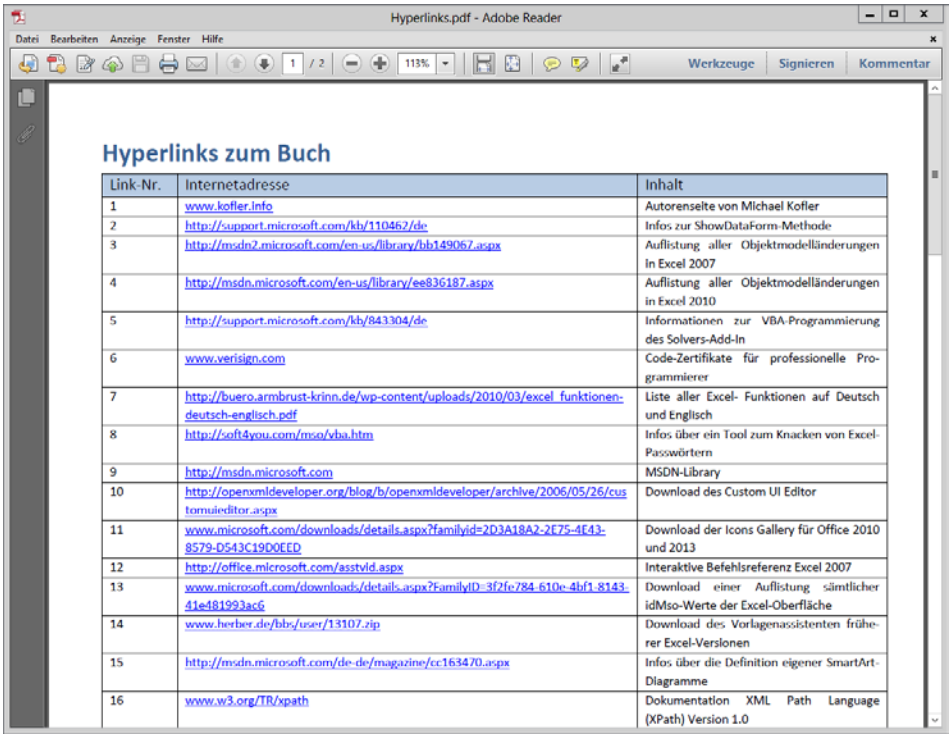
Kommentare in JavaScript-Dateien (Kapitel 15.9) schließlich sehen so aus:

```
/* 15\OfficeApps\SimpleApp\ComplexApp.js */
```

Internetadressen (Hyperlinks.pdf)

Der Text dieses Buchs enthält zahlreiche Verweise auf Internetadressen, wo Sie weiterführende Informationen finden, Tools herunterladen können etc. Da viele dieser „Links“ zu kompliziert sind, um sie abzutippen, haben wir sie in einem PDF-Dokument zusammengefasst. Es trägt den Namen *Hyperlinks.pdf* und ist ebenfalls auf der beiliegenden CD im Ordner *Info* zu finden.

Die Links in diesem Dokument sind jeweils mit einer Nummer gekennzeichnet, die Sie auch im Buchtext in der Form *[Link x]* finden, wobei „x“ für die konkrete Link-Nummer steht. Zum Öffnen eines Links genügt ein Mausklick. Beim ersten Mal müssen Sie Ihrem Reader-Programm unter Umständen die Erlaubnis dazu erteilen.



Link-Nr.	Internetadresse	Inhalt
1	www.kofler.info	Autorenseite von Michael Kofler
2	http://support.microsoft.com/kb/110462/de	Infos zur ShowDataForm-Methode
3	http://msdn2.microsoft.com/en-us/library/bb149067.aspx	Auflistung aller Objektmodelländerungen in Excel 2007
4	http://msdn.microsoft.com/en-us/library/ee836187.aspx	Auflistung aller Objektmodelländerungen in Excel 2010
5	http://support.microsoft.com/kb/843304/de	Informationen zur VBA-Programmierung des Solvers-Add-In
6	www.verisign.com	Code-Zertifikate für professionelle Programmierer
7	http://buero.armbrust-krinn.de/wp-content/uploads/2010/03/excel_funktionen-deutsch-englisch.pdf	Liste aller Excel- Funktionen auf Deutsch und Englisch
8	http://soft4you.com/mso/vba.htm	Infos über ein Tool zum Knacken von Excel-Passwörtern
9	http://msdn.microsoft.com	MSDN-Library
10	http://openxmldeveloper.org/blog/b/openxmldeveloper/archive/2006/05/26/customuieditor.aspx	Download des Custom UI Editor
11	www.microsoft.com/downloads/details.aspx?familyid=2D3A18A2-2E75-4E43-8579-D543C19D0EED	Download der Icons Gallery für Office 2010 und 2013
12	http://office.microsoft.com/asstvid.aspx	Interaktive Befehlsreferenz Excel 2007
13	www.microsoft.com/downloads/details.aspx?FamilyID=3f2fe784-610e-4bf1-8143-41e481993ac6	Download einer Auflistung sämtlicher idMso-Werte der Excel-Oberfläche
14	www.herber.de/bbs/user/13107.zip	Download des Vorlagenassistenten früherer Excel-Versionen
15	http://msdn.microsoft.com/de-de/magazine/cc163470.aspx	Infos über die Definition eigener SmartArt-Diagramme
16	www.w3.org/TR/xpath	Dokumentation XML Path Language (XPath) Version 1.0

Die Internetadressen in der Datei *Hyperlinks.pdf* können Sie direkt per Mausklick öffnen.

Und eine Entschuldigung

Wir sind uns bewusst, dass unter den Lesern dieses Buchs auch zahlreiche Frauen sind. Dennoch ist in diesem Buch immer wieder von *dem Anwender* die Rede, wenn wir keine geschlechtsneutrale Formulierung gefunden haben. Wir bitten dafür alle Leserinnen ausdrücklich um Entschuldigung. Wir sind uns des Problems bewusst, empfinden Doppelgleisigkeiten der Form *der/die Anwender/in* oder kurz *AnwenderIn* aber sprachlich als nicht schön – und zwar sowohl beim Schreiben als auch beim Lesen.

TEIL I

Intuitiver Einstieg

1 Das erste Makro

Im Verlauf des Kapitels lernen Sie Begriffe wie Makro oder Visual Basic für Applikationen kennen, zeichnen selbst Makros auf, integrieren diese in die Symbolleiste für den Schnellzugriff, stellen eine einfache Datenbankanwendung zusammen etc. Das Kapitel gibt Ihnen einen ersten Einblick in Themen, die im Buch später viel ausführlicher aufgegriffen werden.

1.1 Begriffsdefinition

Makro

Dieses Kapitel steht unter dem Motto „Das erste Makro“. Daher soll zunächst einmal der Begriff „Makro“ erklärt werden:

> *Ein Makro bezeichnet eine Reihe von Anweisungen an den Computer, die dieser ausführt, sobald er dazu aufgefordert wird.*

Welchen Zweck haben Makros? Hier ein paar Antworten:

- Sie können Arbeitsschritte, die sich häufig wiederholen, automatisieren und vereinfachen.
- Sie können Excel und dessen Benutzeroberfläche ganz an Ihre Bedürfnisse anpassen.
- Sie können die Bedienung von Excel für andere Anwender vereinfachen, sodass diese praktisch ohne Schulung konkrete Excel-Anwendungen bedienen können.
- Und schließlich können Sie echte „Programme“ (Anwendungen) schreiben, denen man kaum mehr anmerkt, dass sie in Excel entstanden sind.

Da der Computer natürlichsprachliche Anweisungen wie „Speichere diese Datei!“ oder „Stelle die drei markierten Zellen in einer größeren Schrift dar!“ leider nicht versteht, müssen Makroanweisungen in einer speziellen Sprache formuliert werden. Aus Kompatibilitätsgründen stellt Excel zwei Sprachen zur Auswahl:

- Die *herkömmliche Makroprogrammiersprache* hat sich im Verlauf der ersten Excel-Versionen gebildet. Makros in dieser Sprache heißen herkömmliche Makros oder Excel-4-Makros, weil die Grundkonzepte dieser Sprache seit der Programmversion 4 nicht mehr verändert oder erweitert wurden. Die Schlüsselwörter der herkömmlichen Makroprogrammiersprache werden in deutscher Sprache angegeben.

- Mit Excel 5 wurde die Sprache *Visual Basic für Applikationen* (kurz VBA) eingeführt. Sie bietet mehr und ausgefeiltere Möglichkeiten der Programmsteuerung, wirkt aber vielleicht auf den ersten Blick etwas umständlich (insbesondere dann, wenn Sie schon einmal ein herkömmliches Makro geschrieben haben).

Hinweis

In Excel 5 war der deutsche VBA-Dialekt die Default-Einstellung. Bereits mit Version 7 vollzog Microsoft dann aber eine Kehrtwendung: Plötzlich wurden englische Schlüsselwörter bevorzugt. In Version 97 wurde der deutsche Dialekt dann vollständig gestrichen. Deutschsprachiger VBA-Code wird beim Laden alter Dateien automatisch konvertiert. Dieses Buch beschreibt daher ausschließlich die englische VBA-Variante!

Zu den Makrosprachen gleich ein Beispiel, das die aktuelle Arbeitsmappe speichert (zuerst als herkömmliches Makro, dann in VBA):

```
=SPEICHERN()                'herkömmliches Makro (Excel 4)
AktiveArbeitsmappe.Speichern 'VBA deutsch  (Excel 5 und 7)
ActiveWorkbook.Save          'VBA englisch (Excel 97 und folgende)
```

Das zweite Beispiel stellt in zuvor markierten Zellen eine etwas größere Schriftart ein (wiederum zuerst als herkömmliches Makro, dann in VBA):

```
=SCHRIFTART.EIGENSCHAFTEN(;;ZELLE.ZUORDNEN(19)+2)    'Excel 4
Selection.Font.Size = Selection.Font.Size + 2        'VBA englisch
```

Aus dem Begriff Makro allein geht die gewählte Makrosprache nicht hervor. In diesem Buch meint Makro aber immer ein VBA-Makro.

Anmerkung

Die obigen Beispiele sind in dieser Form nicht verwendbar. Ein Excel-4-Makro muss mit dem Namen des Makros beginnen und mit dem Kommando *=RÜCKSPRUNG()* enden. VBA-Makros müssen zwischen *Sub Name()* und *End Sub* eingeklammert werden. Die Syntaxkonventionen von Visual Basic gehen aus den Beispielen dieses Kapitels hervor. Eine detaillierte Beschreibung der VBA-Syntax bietet Kapitel 4.

Makros aufzeichnen

Generell gibt es zwei Möglichkeiten, Makros zu erstellen: Entweder Sie tippen die Kommandos über die Tastatur ein oder Sie lassen sich das Makro von Excel „aufzeichnen“. Damit ist gemeint, dass Sie (über Maus und Tastatur) Daten eingeben, Zellen formatieren, Kommandos ausführen etc. Excel verfolgt Ihre Aktionen und schreibt die entsprechenden VBA-Anweisungen in ein Modul. Wenn Sie das so erzeugte Makro später ausführen, werden

exakt dieselben Arbeitsschritte, die Sie vorher manuell ausgeführt haben, durch das Makro wiederholt.

In der Realität erfolgt die Erstellung von Makros zumeist in einem Mischmasch aus beiden Methoden. Sie werden zwar immer wieder einzelne Arbeitsschritte von Excel aufzeichnen lassen, ebenso wird es aber auch oft notwendig sein, diese Makros später zu ändern oder zu erweitern.

Makros ausführen

Die unbequemste Form, ein Makro auszuführen, bietet das Kommando Ansicht | Makros | Makros anzeigen. Es stellt Ihnen eine Liste mit allen definierten Makros in allen geladenen Arbeitsmappen zur Auswahl. Sobald Sie einen der Makronamen anklicken, wird das entsprechende Makro ausgeführt.

Daneben bestehen aber elegantere Varianten: Sie können ein Makro mit einem beliebigen Symbol in der Symbolleiste für den Schnellzugriff oder mit einer Tastaturabkürzung Strg+Anfangsbuchstabe verbinden. Sobald Sie das Symbol anklicken oder die Tastaturabkürzung eingeben, wird das Makro sofort ausgeführt. Solcherart definierte Makros können eine enorme Arbeitserleichterung bedeuten, wie die Beispiele der folgenden Abschnitte beweisen.

Bis Excel 2003 bestand zudem die Möglichkeit, Makros über selbst definierte Menübefehle zu aktivieren. Da es seit Excel 2007 keine klassische Menüleiste mehr gibt, fällt diese Möglichkeit weg. Ersatzweise können Sie aber das Menüband um eigene Makrostartbefehle erweitern. Das dazu notwendige Verfahren erfordert ein wenig Programmierung und wird ausführlich im Abschnitt 8.2 beschrieben.

Es besteht sogar die Möglichkeit, Makros automatisch beim Eintreten bestimmter Ereignisse ausführen zu lassen. Excel kennt eine ganze Menge solcher Ereignisse – etwa den Wechsel des aktiven Arbeitsblatts, die Neuberechnung des Blatts, das Speichern der Arbeitsmappe etc. Ereignisprozeduren werden in Abschnitt 4.4 ausführlich behandelt.

Makros und Programme

Vielen Excel-Anwendern – sogar solchen, die bereits Makros erstellt haben – stehen die Haare zu Berge, wenn sie den Begriff „programmieren“ hören. Programmieren, das sei nur etwas für Profis mit Fach- oder Hochschulausbildung, lautet eine immer wieder vertretene Ansicht. In Wirklichkeit sind Sie bereits ein Programmierer, sobald Sie das erste – vielleicht nur dreizeilige – Makro erstellt haben. Jedes Makro stellt im Prinzip ein echtes Programm dar.

In diesem Buch wird der Begriff Programm zumeist etwas weiter gefasst. Ein Programm meint eine eigenständige Excel-Anwendung, die sich zumeist durch eigene Menübandkommandos, eigene Dialoge und eine oft große Anzahl von Makros auszeichnet. Dieses Buch leitet Sie vom ersten Makro (in diesem Kapitel) bis zu umfangreichen Anwendungen.

1.2 Was ist Visual Basic für Applikationen?

Visual Basic für Applikationen ist eine Makroprogrammiersprache. Mit VBA können Sie Excel-Anwendungen automatisieren oder in ihrer Bedienung vereinfachen. Die Einsatzmöglichkeiten von VBA reichen so weit, dass Sie damit vollkommen eigenständige Programme erstellen können, denen kaum mehr anzumerken ist, dass es sich eigentlich um Excel-Anwendungen handelt. Einführungs- und Anwendungsbeispiele einfacher Makros finden Sie in diesem Kapitel.

Geschichtliches

Die herkömmliche Makrosprache von Excel hat sich ursprünglich aus dem Wunsch heraus entwickelt, neue Tabellenfunktionen zu definieren und wiederholt auftretende Kommandos zu einer Einheit (zu einem Makro) zusammenzufassen. Um die Bedienung von Excel-Anwendungen möglichst einfach zu gestalten, wurden außerdem die Veränderung der (früheren) Menüs und die Definition eigener Dialoge ermöglicht. Verbunden mit dem riesigen Funktionsspektrum von Excel hat sich daraus bis Version 4 eine ziemlich unübersichtliche Makrosprache entwickelt.

Diese Makrosprache hat zwar eine fast uneingeschränkte Programmierung aller Excel-Funktionen erlaubt, viele Programmierprobleme ließen sich allerdings nur umständlich lösen. Die resultierenden Programme waren fehleranfällig und langsam. Bei größeren Projekten traten die Grenzen dieser Makrosprache besonders deutlich zum Vorschein. Für Anwender, die gleichzeitig mehrere Microsoft-Programme (Excel, Word, Access) verwenden, kam als weiteres Problem hinzu, dass jedes Programm mit einer eigenen Makrosprache ausgestattet ist.

Aufgrund all dieser Unzulänglichkeiten beschloss Microsoft, eine seinerzeit vollkommen neue Makroprogrammiersprache zu entwickeln, die zuerst für Excel zur Verfügung stand, inzwischen aber längst in alle Office-Anwendungen integriert wurde.

Die besonderen Merkmale von VBA

VBA ist im Gegensatz zu bisherigen Makrosprachen eine vollwertige Programmiersprache: VBA kennt alle in „echten“ Programmiersprachen üblichen Variablentypen, kann mit Zeichenketten umgehen, dynamische Felder verwalten, zur Definition rekursiver Funktionen eingesetzt werden etc.

- VBA ist **objektorientiert**: Als *Objekte* gelten beispielsweise markierte Zellbereiche, Diagramme etc. Typische Merkmale von Objekten – etwa die Ausrichtung des Zellinhalts, die Hintergrundfarbe eines Diagramms – werden über sogenannte *Eigenschaften* eingestellt. Eigenschaften sind also vordefinierte Schlüsselwörter, die zur Manipulation von Objekten vorgesehen sind. Neben den Eigenschaften gibt es noch *Methoden*, die zur Ausführung komplexer Operationen vorgesehen sind: etwa zum Erzeugen von Objekten (neuen Diagrammen, Pivot-Tabellen etc.) oder zum Löschen vorhandener Objekte. Methoden lassen sich am ehesten mit herkömmlichen Kommandos vergleichen. Der wesentliche Unterschied besteht darin, dass Methoden nur auf speziell dafür vorgesehene Objekte angewendet werden können.

- VBA ist **ereignisorientiert**: Das Anklicken eines Buttons oder eines Symbols führt zum automatischen Aufruf des dazugehörigen Makros. Als Programmierer müssen Sie sich nicht um die Verwaltung der Ereignisse kümmern, sondern lediglich Makros erstellen, die dann von Excel selbstständig aufgerufen werden.
- VBA stellt professionelle Hilfsmittel zur **Fehlersuche** zur Verfügung: Programmteile können Schritt für Schritt ausgeführt und die Inhalte von Variablen können überwacht werden. Die Programmausführung kann beim Eintreffen von bestimmten Bedingungen unterbrochen werden.
- VBA ist **erweiterungsfähig**: In jedem VBA-Dialekt kann auf Objekte anderer Anwendungen zugegriffen werden. Beispielsweise ist es möglich, in einem Excel-VBA-Programm auch die Schlüsselwörter (im Fachjargon: die *Objektbibliothek*) von Access oder Word zu nutzen. Mit Add-ins können Sie neue Excel-Funktionen und Objekte erstellen.
- Zur Ausstattung von VBA gehört ein leistungsfähiger **Dialogeditor**. Die Verwaltung von Dialogen erfolgt nach dem gleichen objekt- und ereignisorientierten Schema wie die Verwaltung von Excel-Objekten.

Hinweis

Gelegentlich stiftet der Umstand Verwirrung, dass es bei Microsoft mehrere Produkte gibt, die mit Visual Basic zu tun haben. Thema dieses Buchs ist Excel, das über die integrierte Sprache VBA gesteuert werden kann. Es gab und gibt aber auch die eigenständigen Produkte „Visual Basic 6“ und dessen Nachfolger „Visual Basic .NET“. Dabei handelt es sich um Programmiersprachen, mit denen Sie unabhängig vom Office-Paket Programme entwickeln können; die Ausführung solcher Programme setzt also nicht voraus, dass beim Anwender ebenfalls das Office-Paket installiert ist. VBA auf der einen Seite und VB6 bzw. VB.NET auf der anderen Seite weisen zwar Ähnlichkeiten auf, sind aber durchaus nicht immer kompatibel. (Insbesondere bei VB.NET gibt es sehr viele Änderungen.)

1.3 Beispiel: eine Formatvorlage mit einem Symbol verbinden

Im ersten Beispiel wird zuerst eine Formatvorlage definiert. (Eine Formatvorlage sammelt ein Bündel von Formatinformationen zu Schriftart, Ausrichtung, Rahmen und Farben. Formatvorlagen können zur Formatierung von Zellen verwendet werden.) Anschließend wird ein Makro aufgezeichnet, das den markierten Zellen diese Formatvorlage zuweist. Dann wird in die Symbolleiste für den Schnellzugriff ein neues Symbol eingefügt und diesem Makro zugewiesen. Damit besteht die Möglichkeit, die zuvor markierten Zellen durch einen Klick auf das neue Symbol mit der definierten Formatvorlage zu formatieren.

Tipp

Alle Beispiele dieses Kapitels befinden sich natürlich auch in den Beispieldateien der beiliegenden CD (Verzeichnis 01 für das erste Kapitel).

Bevor Sie beginnen

Vorweg einige Tipps, die Ihnen das Leben und die Programmierung mit Excel erleichtern:

- Machen Sie die Befehlsregisterkarte ENTWICKLERTOOLS im Menüband sichtbar. Dazu öffnen Sie die Registerkarte DATEI und wählen OPTIONEN. Klicken Sie links im Dialogfeld auf MENÜBAND ANPASSEN, schalten Sie im rechten Listenfeld das Kontrollkästchen vor ENTWICKLERTOOLS ein und schließen Sie das Dialogfeld mit OK.
- Die nun sichtbare Befehlsregisterkarte stellt Ihnen alle Werkzeuge für die Aufzeichnung und Bearbeitung von Makros, den Entwurf von Formularen sowie die XML-Programmierung zur Verfügung.

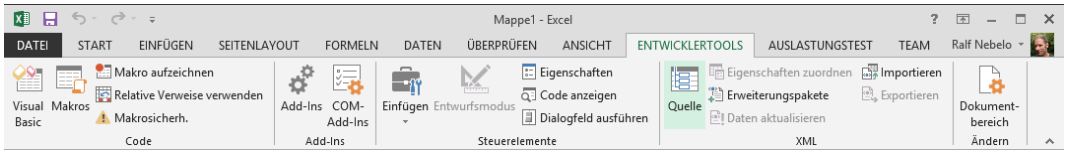


BILD 1.1 Die Registerkarte Entwicklertools

- Standardmäßig deaktiviert Excel sämtliche Makros. Beim Öffnen eines „Makrohaltigen“ Dokuments erscheint unterhalb des Menübands eine Sicherheitswarnung, über deren INHALT-AKTIVIEREN-Schaltfläche Sie die Makroausführung ausdrücklich „freischalten“ müssen. Was als Schutz vor möglichen Makroviren gedacht ist, kann die Entwicklung eigener Makros aber ungemein behindern.
- Wenn Sie möchten, dass Excel alle Ihre Makros ohne Sicherheitswarnung aktiviert, stellen Sie die Makrosicherheitsstufe vorübergehend für die Zeit der Makroentwicklung auf den niedrigsten Level ein. Dazu wählen Sie ENTWICKLERTOOLS | MAKROSICHERH., aktivieren das Optionsfeld ALLE MAKROS AKTIVIEREN und wählen OK. Sofern Sie einen Virenscanner installiert haben (was dringend zu empfehlen ist), überprüft Excel nun immer noch alle Dokumente auf mögliche Makroviren, sodass Sie kein allzu großes Risiko eingehen. (Weitere Informationen zur Sicherheit von Makros folgen in Abschnitt 4.7.)
- In der VBA-Entwicklungsumgebung (die Sie mit Alt+F11 erreichen) sind einige Optionen verquer voreingestellt. Den Optionsdialog erreichen Sie dort mit EXTRAS | OPTIONEN.
- Dort deaktivieren Sie AUTOMATISCHE SYNTAXÜBERPRÜFUNG. (Die Syntax wird weiterhin überprüft, fehlerhafte Zeilen werden rot markiert. Es entfällt nur die lästige Fehlermeldung samt Piepston.)
- Dann aktivieren Sie die Option VARIABLENDEKLARATION ERFORDERLICH. (Eine ausführliche Begründung folgt in Abschnitt 4.1.)
- Im Dialogblatt ALLGEMEIN deaktivieren Sie die Option KOMPILIEREN | BEI BEDARF (siehe Abschnitt 3.2).

▸ Schritt 1: Definition der Formatvorlage „Result“

Zellen, die das (Zwischen-)Ergebnis einer Berechnung beinhalten, sollen folgendermaßen aussehen:

- Schrift: Arial, 14 Punkt, fett
- Rahmen: doppelte Linie unten
- Zahlenformat: zwei Dezimalstellen

Wenn Sie möchten, können Sie natürlich auch andere Formatierungsmerkmale auswählen. Es geht in diesem Beispiel nur darum, ein neues, eindeutig erkennbares Format zu definieren.

Zur Definition der Formatvorlage öffnen Sie eine neue Arbeitsmappe mit Datei | Neu | Leere Arbeitsmappe, schreiben in eine beliebige Zelle eine Zahl und formatieren diese Zelle anschließend mit den oben aufgezählten Merkmalen. Anschließend führen Sie das Kommando Start | Zellenformatvorlagen | Neue Zellenformatvorlage aus. Im nun erscheinenden Dialog geben Sie als Formatvorlagenname „Result“ ein und klicken OK an.

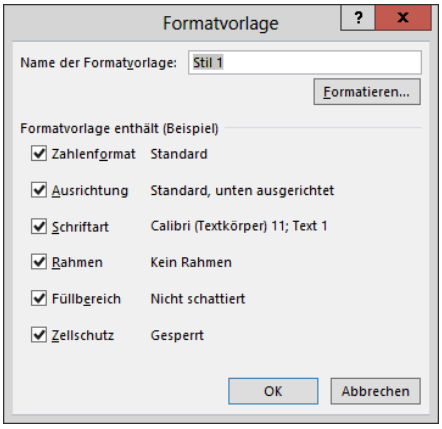


BILD 1.2
Die Definition einer neuen Formatvorlage

Geben Sie in Ihrer Tabelle in einer beliebigen Zelle eine weitere Zahl ein, und testen Sie die neue Formatvorlage: Führen Sie Start | Zellenformatvorlagen aus, und wählen Sie als Vorlage „Result“. Die zweite Zelle sollte nun ebenso formatiert sein wie die erste.

▸ Schritt 2: Makro aufzeichnen

Die Arbeitsschritte, die Sie gerade zur Formatierung einer Testzelle ausgeführt haben, sollen in Zukunft automatisch von einem Makro erledigt werden. Dazu müssen diese Arbeitsschritte in einem Makro aufgezeichnet werden. Bewegen Sie den Zellzeiger in eine neue Zelle, und geben Sie dort (um das Ergebnis zu kontrollieren) eine Zahl ein. Schließen Sie die Eingabe mit Return ab, und bewegen Sie den Zellzeiger gegebenenfalls zurück in die gerade veränderte Zelle. Wählen Sie in Excel (nicht in der VBA-Entwicklungsumgebung) das Kommando Entwicklertools | Makro aufzeichnen, und geben Sie als Makronamen „FormatAsResult“ ein.

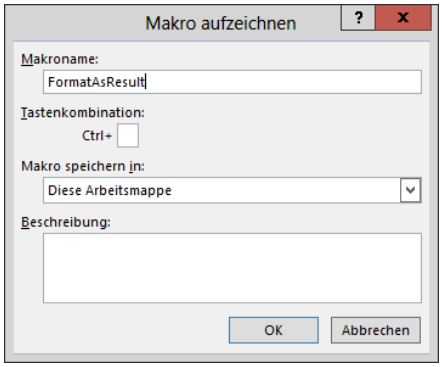


BILD 1.3
Der Dialog zum Aufzeichnen von Makros

Sobald Sie das Dialogfeld mit OK schließen, beginnt Excel mit der Aufzeichnung des neuen Makros. Formatieren Sie die gerade aktive Zelle mit dem Druckformat „Result“ (ebenso wie am Ende von Schritt 1, als Sie die Formatvorlage getestet haben). Beenden Sie die Makroaufzeichnung mit ENTWICKLERTOOLS | AUFZEICHNUNG BEENDEN oder durch das Anklicken des kleinen Quadrats, das seit Beginn der Aufzeichnung in der Statuszeile von Excel am unteren Bildschirmrand angezeigt wird.

Jetzt können Sie sich das fertige Makro ansehen, indem Sie mit Alt+F11 in die Entwicklungsumgebung wechseln und dort „Modul1“ ansehen. (Dieses Modul wurde im Zuge der Makroaufzeichnung automatisch erzeugt. Wenn „Modul1“ schon existiert, legt Excel ein neues Modul mit dem Namen „Modul2“ an.) Das neue Modul sollte folgendermaßen aussehen:

```
Sub FormatAsResult()
'
' FormatAsResult Makro
'
   Selection.Style = "result"
End Sub
```

Jetzt sollten Sie das neue Makro noch testen: Wechseln Sie wieder zurück in das Tabellenblatt, geben Sie in einer beliebigen Zelle eine weitere Zahl ein, und schließen Sie die Eingabe mit Return ab. Wählen Sie mit dem Kommando ANSICHT | MAKROS | MAKROS ANZEIGEN das Makro „FormatAsResult“ aus. Excel führt Ihr Makro aus, die Zelle sollte anschließend in dem nun schon vertrauten Ergebnisformat erscheinen.

▸ Schritt 3: Definition eines neuen Symbols

Damit das Makro bequemer aufgerufen werden kann, soll jetzt ein neues Symbol in die Symbolleiste für den Schnellzugriff (die Sie links in der Titelleiste des Excel-Programmfensters finden) eingefügt werden. Dazu klicken Sie mit der rechten Maustaste auf die Symbolleiste und wählen den Befehl PASSEN SIE DIE SYMBOLLEISTE FÜR DEN SCHNELLZUGRIFF AN. Stellen Sie das Listenfeld BEFEHLE AUSWÄHLEN auf „Makros“ ein, markieren Sie „FormatAsResult“ in der Liste darunter, und klicken Sie auf HINZUFÜGEN >>. Der Name Ihres aufgezeichneten Makro sollte nun in dem rechten Listenfeld erscheinen.

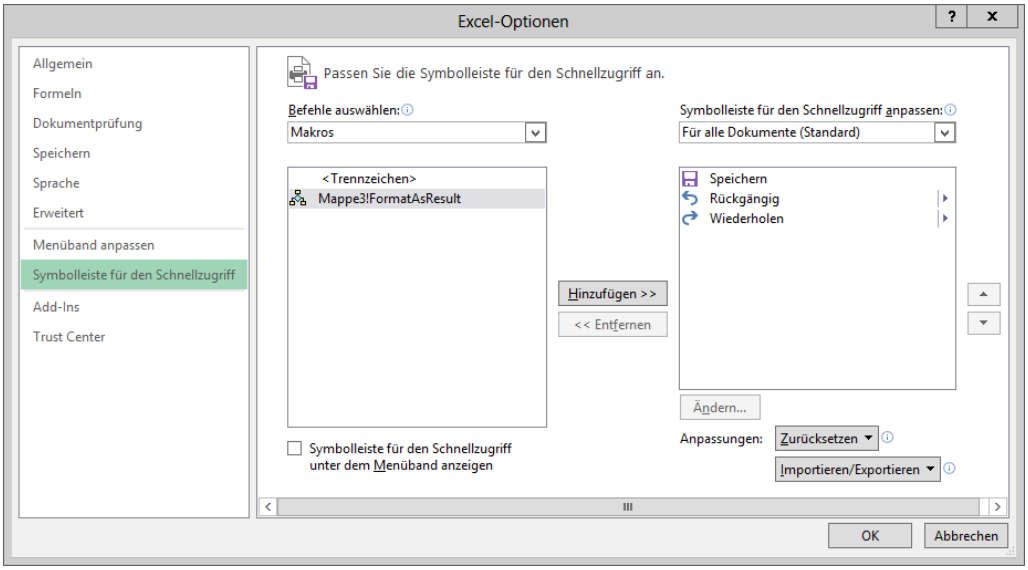


BILD 1.4 Der Dialog zur Anpassung der Symbolleiste für den Schnellzugriff

Würden Sie das ANPASSEN-Dialogfeld jetzt per OK-Schaltfläche schließen, wäre das neue Makrostartsymbol bereits in der Symbolleiste für den Schnellzugriff sichtbar. Beim „Überfahren" des Symbols mit dem Mauszeiger würde allerdings nur der vielleicht etwas kryptische Makroname „FormatAsResult" als sogenannter Tooltip-Text erscheinen. Sie sollten diesen Text gegen eine aussagekräftigere Bezeichnung ersetzen, welche die Funktion des Makros auch anderen Anwendern offenbart.

Und wo Sie schon gerade beim Anpassen sind, sollten Sie dem neuen Symbol auch gleich eine individuelle Grafik zuweisen, die das Standardbild, das voreinstellungsgemäß für alle Makros Verwendung findet, ersetzt.

Um beides zu bewerkstelligen, klicken Sie im immer noch geöffneten ANPASSEN-Dialog im rechten Listenfeld auf den Makronamen „FormatAsResult" und anschließend auf die Schaltfläche ÄNDERN. Markieren Sie nun ein Symbol Ihrer Wahl, und ändern Sie den Anzeigenamen im Textfeld darunter, beispielsweise in „Formatvorlage Result zuweisen". Nach einem Klick auf OK erscheint das neue Symbol in seiner endgültigen Fassung in der Symbolleiste für den Schnellzugriff.

Im Unterschied zu früheren Excel-Versionen gibt es keine Möglichkeit, einem Symbol eine selbst erstellte Grafik zuzuweisen. Die Auswahl ist somit auf die angebotenen Standard-Icons beschränkt.