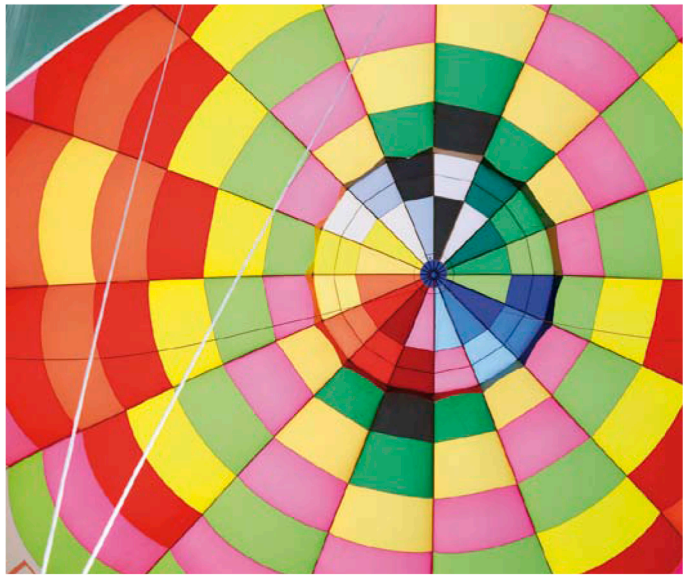
2. Auflage

Richtig einsteigen:

Excel VBA-Programmierung

Für Excel 2007 bis 2016



Bernd Held

O'REILLY®

Bernd Held

Richtig einsteigen: Excel VBA-Programmierung

– Für Microsoft Excel 2007 bis 2016

O'REILLY®

Bernd Held

Lektorat: Ariane Hesse
Korrektorat: Sibylle Feldmann
Satz: Gerhard Alfes, www.mediaservice.tv
Herstellung: Susanne Bröckelmann
Umschlaggestaltung: Michael Oréal, www.oreal.de
unter Verwendung eines Fotos von francescodemarco/Fotolia.com
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH,
www.mediaprint-druckerei.de

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.

ISBN:
Print 978-3-96009-003-8
PDF 978-3-96010-027-0
epub 978-3-96010-028-7
mobi 978-3-96010-029-4

1. Auflage 2016

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«. O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Inhalt

Einleitung

Wie dieses Buch aufgebaut ist

Sie wollten schon immer einmal mit Makros arbeiten? Sie möchten noch tiefer in Excel einsteigen, um damit Abläufe zu automatisieren, zu sichern und zu optimieren?

Sie haben Freude daran, sich mit zukunftsträchtigen und wertvollen Techniken zu beschäftigen? Wenn ja, dann ist dieses Buch das Richtige für Sie!

Dieses Buch führt Sie Schritt für Schritt richtig in die VBA-Programmiersprache ein. Die Grundlage für dieses Buch ist ein dreitägiger VBA-Kurs, den ich seit über 10 Jahren fast jeden Monat in Vaihingen-Enz und an anderen Standorten in Deutschland durchführe.

Nach Durcharbeiten dieses Buchs werden Sie in der Lage sein, große Teile Ihrer täglichen Arbeit zu automatisieren und lästige Routinearbeiten über Makros erledigen zu lassen. Mit diesem Buch als Grundlage werden Sie viel Spaß daran haben, auf diesem Gebiet weiter zu machen. Da Excel in fast jeder Firma vertreten und Automatisierung beinahe immer notwendig ist, werden Sie sich durch Ihr neu erworbenes Know-how viele Freunde machen.

Lernen Sie zu Beginn des Buchs zunächst einmal Ihr Werkzeug kennen. Dabei erfahren Sie, wie Sie sich in der Entwicklungsumgebung von Excel zurechtfinden, und Sie werden erfolgreich Schleifen und Verzweigungen anwenden können. Danach lernen Sie die Programmierung der wirklich wichtigen Excel-Objekte (Zellen, Tabellen und Mappen) kennen. Erfahren Sie, welche Methoden und Eigenschaften dieser Objekte dringend gebraucht werden, um die wichtigsten Arbeiten automatisiert und schnell erledigen zu können.

Am Ende eines jeden Kapitels erwarten Sie Übungsaufgaben und Verständnisfragen, sodass Sie sicher sein können, den optimalen Nutzen aus diesem Buch zu ziehen.

Die Beispiele, die im Buch vorgestellt werden, wurden mit der Excel-Version 2016 geschrieben. Alle hier vorgestellten Makros können aber genauso gut in allen Versionen ab Excel 2002 genutzt werden.

Sollten Sie Fragen oder Anregungen haben, dann scheuen Sie sich nicht, mich unter meiner Mailadresse

info@held-office.de

zu kontaktieren.

Download der Beispieldateien

Die für dieses Buch bereitgestellten Dateien können Sie sich unter folgender Adresse herunterladen:

www.Held-office.de/Downloads/RichtigEinsteigen.zip

Sie finden sie auch direkt auf der Verlagsseite:

http://downloads.oreilly.de/9783960090038

Die Icons

Dieses Buch führt Sie in die VBA-Programmierung mit Excel ein. Schon nach kurzer Einarbeitungszeit werden Sie über das nötige Know-how verfügen, um ansprechende Ergebnisse zu erzielen. Zugleich erwerben Sie das entsprechende Know-why, das heißt, Sie erfahren immer auch, warum etwas in einer bestimmten Weise funktioniert. Hin und wieder empfiehlt es sich für eine bessere Übersicht, bestimmte Informationen in eigenen Absätzen auszuzeichnen.

Der Textmarker weist Sie auf etwas hin, worauf Sie unbedingt achten sollten.

Hier erfahren Sie, wie Sie etwas besonders einfach und elegant erledigen können.

Absätze mit diesem Icon geben Ihnen wichtige Hintergrundinformationen und erklären, warum etwas in einer bestimmten Weise funktioniert.

Unterstützung für dieses Buch

Der Verlag hat auch von seiner Seite alles unternommen, um die Richtigkeit des Buchinhalts sicherzustellen. Etwaige Korrekturen und Änderungen finden Sie unter folgender Adresse:

http://downloads.oreilly.de/9783960090038

Kommentare, Fragen und Anregungen können Sie ebenfalls an den Verlag schicken. Wenden Sie sich dazu an kommentar@oreilly.de

Bitte beachten Sie, dass der Verlag keinen Support für Software-Produkte anbieten kann.

Über den Autor

Mein Name ist Bernd Held. Ich wurde am 02.04.1969 geboren, bin verheiratet und Vater von zwei Kindern. Während meines Abiturs und in der Zeit danach war ich Leistungssportler, wurde unter anderem zwei Mal Deutscher Jugendmeister über 400 Meter Hürden und nahm an Europa- und Weltmeisterschaften teil. Nach meiner sportlichen Laufbahn habe ich mich ins Berufsleben gestürzt und mich auf die Themen Excel/Access, individuelle VBA-Schulungen/VBA-Kurse sowie VBA-Programmierung spezialisiert.

Von Haus aus bin ich gelernter Informatiker. Zunächst war ich zwei Jahre bei einer kleinen Softwarefirma in der Entwicklung und danach sechs Jahre bei T-Systems im Controlling beschäftigt. Dort war ich verantwortlich für das Berichtswesen, die Leistungsverrechnung, das Erstellen von betrieblichen Auswertungen und Wirtschaftlichkeitsrechnungen sowie für die Erstellung neuer Controlling-Tools auf Basis von Microsoft Office. Im Januar 2002 folgte dann der Schritt in die Selbstständigkeit.

Seit dieser Zeit konzentriere ich mich auf die Auftragsprogrammierung, die Unternehmensberatung sowie das Schreiben von Fachartikeln und Computerbüchern. Einige Bücher von mir wurden bereits ins Russische, Tschechische und Englische übersetzt. Weitere Aufgabengebiete sind das fachliche Überarbeiten von Computerbüchern sowie die Durchführung von VBA-Schulungen. Zu meinen Spezialgebieten zählen Excel, VBA-Programmierung und Access. Acht Jahre in Folge wurde ich als MVP (Most Valuable Professional) für den Bereich Excel von Microsoft ausgezeichnet.

Seit 2008 arbeite ich neben meinen drei Angestellten mit einem eigenen Team aus Experten erfolgreich zusammen. Wir führen in erster Linie Programmierprojekte und Schulungen durch, sind in der Beratung tätig und schreiben Bücher sowie Artikel für diverse Fachbuchverlage.

Kapitel 1
Die Entwicklungsumgebung von Excel

In diesem Kapitel:

Excel für die Programmierung vorbereiten

Bevor Sie richtig mit Excel-VBA einsteigen können, müssen Sie Microsoft Excel erst einmal für den Gebrauch der zukünftigen Makros einrichten.

Die Sicherheitsstufe heruntersetzen

Seit der Version Excel 2007 ist der Umgang mit Makros in Office etwas erschwert worden. Standardmäßig ist nach der Installation des Office-Pakets in Excel die höchste Sicherheitsstufe eingestellt. In dieser Einstellung sind die Nutzung und das Schreiben von Makros nicht möglich.

Bereiten Sie Excel für die zukünftigen Aufgaben einmalig wie folgt vor:

1. Starten Sie Microsoft Excel.
2. Klicken Sie im Menüband unter *Datei* auf den Befehl *Optionen*.

3. Im Dialog *Excel-Optionen* wählen Sie die Rubrik *Trust Center* aus.
4. Klicken Sie auf die Schaltfläche *Einstellungen für das Trust Center.*
5. Wählen Sie im Dialog *Trust Center* die Rubrik *Makroeinstellungen* aus.
6. Aktivieren Sie in den *Makroeinstellungen* die Option *Alle Makros aktivieren*.
7. Setzen Sie einen Haken in das Kontrollkästchen *Zugriff auf das VBA-Projektmodell vertrauen.*

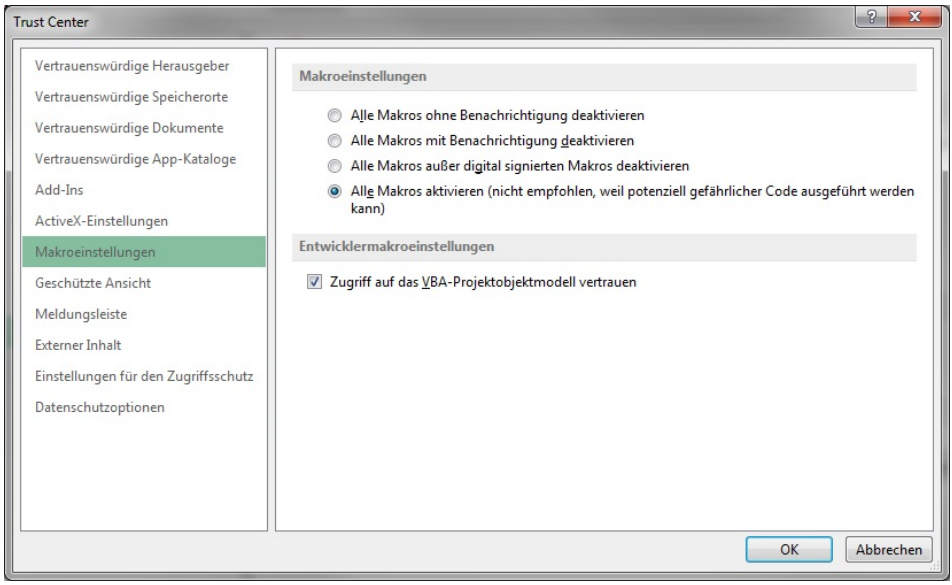


Abbildung 1.1: Die Makroeinstellungen anpassen und den Zugriff auf das VBA-Objektmodell erlauben.

8. Bestätigen Sie diese Einstellung, indem Sie die geöffneten Dialoge jeweils mit *OK* beenden.

Nachdem Sie diese Einstellungen vorgenommen haben, können Sie zukünftig mit Makros arbeiten, außerdem können Sie Dateien, die Makros enthalten, ebenfalls nutzen.

Der Haken im Trust Center bei *Zugriff auf das VBA-Projektmodell vertrauen* versetzt Sie in die Lage, zu einem späteren Zeitpunkt, wenn Sie schon etwas geübter im Umgang mit VBA sind, Makros zu schreiben, mit denen Sie beispielsweise Makros aus anderen Excel-Dateien ersetzen und bearbeiten können. Sie können aufgrund dieser Trust-Einstellung dann im Prinzip zur Laufzeit auf Makros zugreifen sowie ganze Programmteile austauschen und durch neue Makros ersetzen.

Das Werkzeug Entwicklertools einblenden

Um praktikabel mit Makros umgehen zu können, bietet Microsoft eine eigene Registerkarte für die Verwaltung und Programmierung von Makros in der Oberfläche von Excel an, die zunächst jedoch dem Standardanwender verborgen bleibt. Diese Registerkarte trägt den Namen *Entwicklertools*. Mithilfe der Werkzeuge auf dieser Registerkarte können Sie beispielsweise Schaltflächen in Tabellen einfügen und diesen Schaltflächen Makros zuweisen, um diese später mit einem Klick auf die Schaltfläche starten zu können. Des

Weiteren beinhaltet dieser Werkzeugkasten die Möglichkeit, Makros mittels des Makrorekorders automatisch aufzuzeichnen.

Blenden Sie das Menüband *Entwicklertools* wie folgt ein:

1. Klicken Sie im Menüband unter *Datei* auf den Befehl *Optionen*.
2. Im Dialog *Excel-Optionen* wählen Sie die Rubrik *Menüband anpassen* aus.
3. Aktivieren Sie im Feld *Hauptregisterkarten* das Kontrollkästchen *Entwicklertools*.

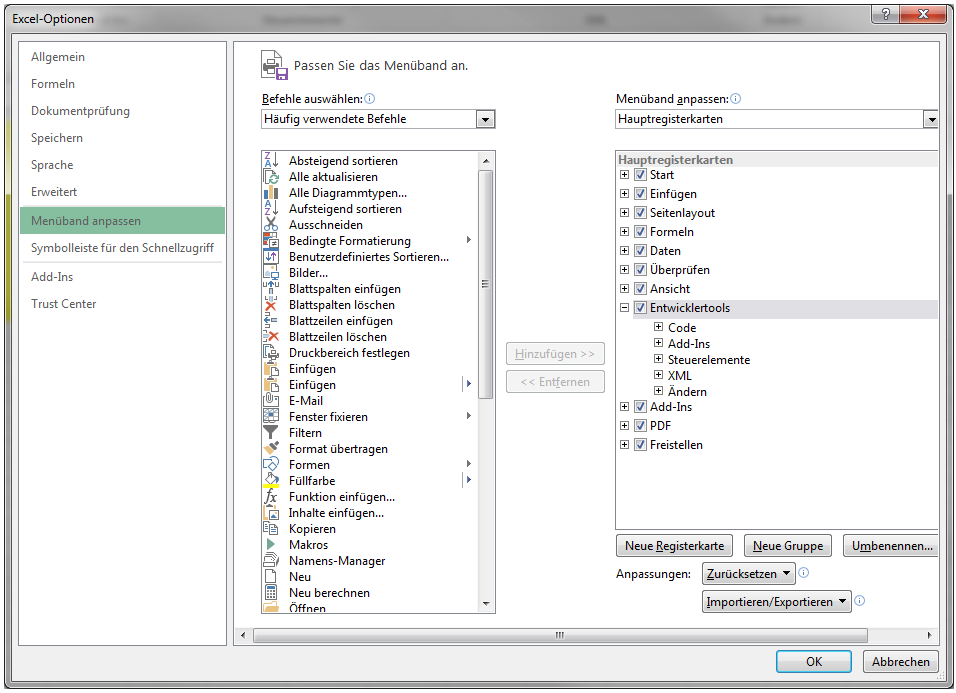


Abbildung 1.2: Das Werkzeug Entwicklertools einblenden.

4. Bestätigen Sie Ihre Einstellung mit einem Klick auf *OK*.

In der Excel-Oberfläche wird jetzt ein neues Menüband mit dem Namen *Entwicklertools* angeboten. Die wichtigsten Funktionen aus diesem Register werden auf den folgenden Seiten nach und nach beschrieben.

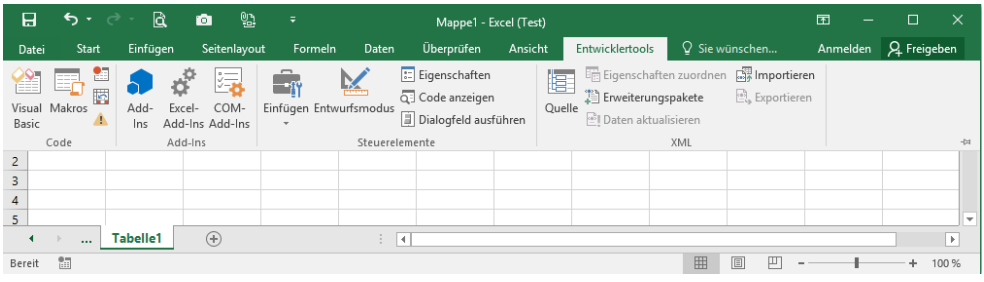


Abbildung 1.3: Das Register Entwicklertools steht für die Bearbeitung und Verwaltung der Makros bereit.

Die Entwicklungsumgebung kennenlernen

Erfahren Sie auf den nächsten Seiten alles, was Sie von Beginn an brauchen, damit Sie Ihre Makros schnell und sicher erfassen, starten und testen können.

Die fertig ausgefüllte Arbeitsmappe *Start.xlsm* mit allen folgenden Beispielen können Sie unter dieser Adresse herunterladen: *http://downloads.oreilly.de/9783960090038*.

Die Entwicklungsumgebung aufrufen

Wagen Sie den ersten Sprung in die Entwicklungsumgebung von Excel, indem Sie im Register *Entwicklertools* den Befehl *Visual Basic* anklicken. Alternativ dazu können Sie sich die Tastenkombination Alt+F11 merken, die Sie ebenfalls direkt in die Entwicklungsumgebung von Microsoft Excel bringt.

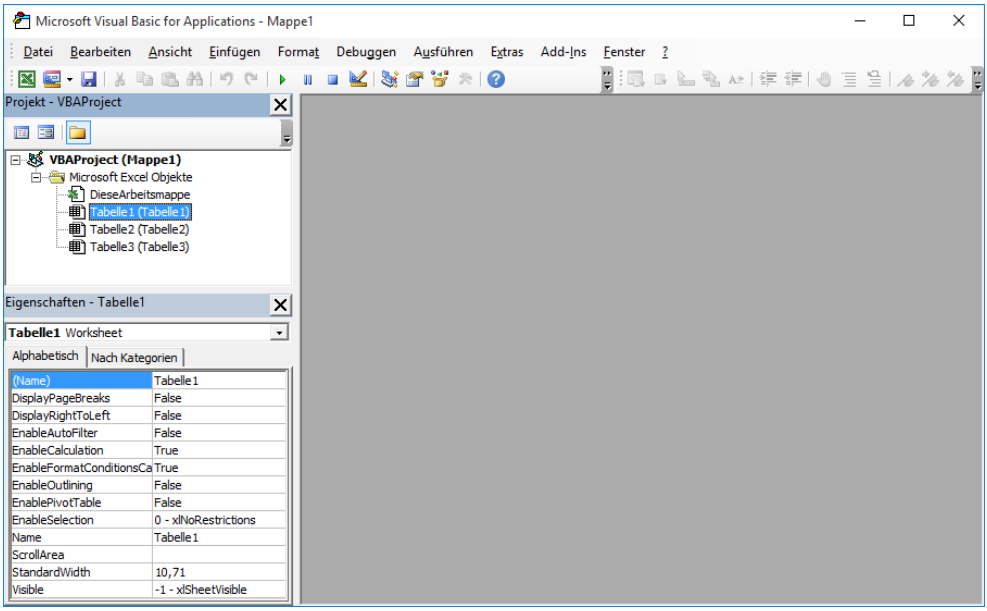


Abbildung 1.4: Die Entwicklungsumgebung von Microsoft Excel.

Die Entwicklungsumgebung in Excel-VBA beinhaltet mehrere Fenster, über die Sie Makros einsehen, erfassen und testen können.

Der Projekt-Explorer

Standardmäßig oben links ist der sogenannte Projekt-Explorer zu finden. Sollte dieser Explorer nicht angezeigt werden, können Sie ihn über das Menü *Ansicht* und den Befehl *Projekt-Explorer* einblenden. Alternativ rufen Sie den Projekt-Explorer über die Tastenkombination Strg+R auf.

Der Projekt-Explorer zeigt Ihnen alle aktuell geöffneten Arbeitsmappen sowie die darin enthaltenen Tabellen an.

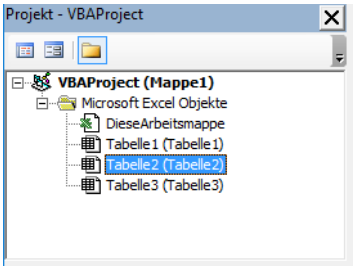


Abbildung 1.5: Der Projekt-Explorer gibt Auskunft über den Inhalt einer Mappe.

Die beiden ersten Symbole *Code anzeigen* und *Objekt anzeigen* werden später für Sie von Interesse sein, wenn Sie beispielsweise eigene benutzerfreundliche Dialoge entwerfen und zwischen den beiden Ebenen Objekt und Code hin- und herspringen müssen.

Das doppelt belegte Symbol *Ordner wechseln* ist eine reine Ansichtsoption. Damit wird die Ansicht der Objekte in der Arbeitsmappe angezeigt. Die standardmäßig eingestellte Ansicht ordnet die Objekte schön ordentlich in Rubriken an. Mit einem weiteren Klick auf dieses Symbol werden alle Objekte alphabetisch und ohne Darstellung in Rubriken angezeigt. Hier ist der Übersichtlichkeit halber die Standardeinstellung zu empfehlen.

Der Projekt-Explorer arbeitet sehr eng mit dem darunterliegenden Fenster, dem Fenster *Eigenschaften*, zusammen. Diese beiden Werkzeuge müssen in einem Atemzug genannt werden, da sie sich einander bedingen. Genau aus diesem Grund sind beide Fenster standardmäßig untereinander angeordnet.

Das Eigenschaftenfenster

Direkt unterhalb des Projekt-Explorers befindet sich standardmäßig das Eigenschaftenfenster. Sollte es nicht eingeblendet sein, können Sie es über den Menübefehl *Ansicht/Eigenschaftenfenster* einblenden. Alternativ dazu können Sie auch die Taste F4 drücken, um das Eigenschaftenfenster einzublenden.

Wie gerade schon erwähnt, sind Explorer und Eigenschaftenfenster als Einheit zu verstehen. Wenn Sie beispielsweise im Projekt-Explorer die *Tabelle1* anklicken, werden im darunterliegenden Eigenschaftenfenster alle Eigenschaften zu dieser Tabelle angezeigt. Durch Setzen dieser Eigenschaften können Verhalten und Aussehen der Tabellen beeinflusst werden.

Generell können Eigenschaften entweder im Eigenschaftenfenster eingestellt oder auch direkt durch Makros gesetzt werden.

Die wichtigsten Eigenschaften lernen Sie anhand von praktischen Aufgaben auf den folgenden Seiten kennen.

Tabellen ein- und ausblenden

Mithilfe des Eigenschaftenfensters lassen sich in Excel Tabellen ein- und ausblenden. Dabei erleben Sie gleich zu Beginn eine kleine Überraschung. Achten Sie darauf, dass Ihre Arbeitsmappe mehr als nur eine Tabelle enthalten muss.

Gehen Sie wie folgt vor, um beispielsweise *Tabelle1* auszublenden:

1. Klicken Sie im Projekt-Explorer auf das Objekt *Tabelle1 (Tabelle1)*.
2. Klicken Sie im Eigenschaftenfenster ganz unten rechts neben die Eigenschaft *Visible*.

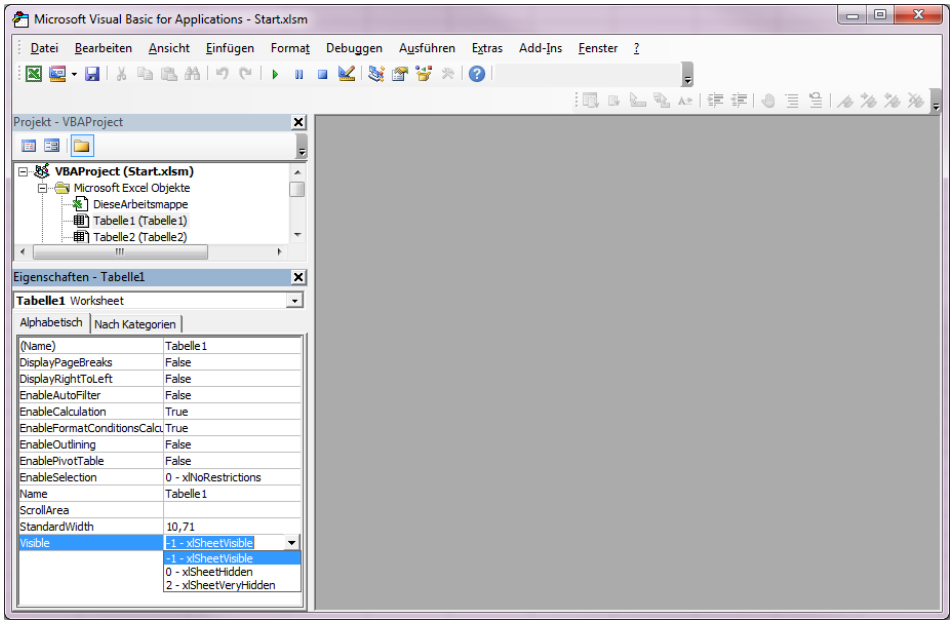


Abbildung 1.6: Die Eigenschaft Visible bietet drei Zustände der Sichtbarkeit an.

3. Wählen Sie aus dem Drop-down die Konstante *0 – xlSheetHidden*.

Jetzt ist *Tabelle1* ausgeblendet. Der Mauszeiger springt dabei, was zunächst vielleicht etwas irritiert, automatisch auf *Tabelle2*. Um den Status von *Tabelle1* wieder einzusehen bzw. zu ändern, klicken Sie einfach auf das Objekt *Tabelle1 (Tabelle1)* im Projekt-Explorer und weisen im Eigenschaftenfenster wieder die Konstante *-1- xlSheetVisible* zu.

Sicher ist Ihnen nicht entgangen, dass das Eigenschaftenfenster bei der Eigenschaft *Visible* eine weitere Konstante anbietet: *2 - xlSheetVeryHidden*. Diese Konstante sorgt dafür, dass die so eingestellte Tabelle sicher ausgeblendet wird. Das bedeutet, dass der »normale« Anwender keine Möglichkeit hat, diese Tabelle über die Oberfläche von Excel einzublenden. Tabellen werden in der Praxis gern sicher versteckt, wenn es darum geht, Daten zu verbergen, damit diese nicht gelöscht werden können.

In einer Arbeitsmappe muss mindestens eine Tabelle immer eingeblendet sein. Sie können daher nicht alle Tabellen ausblenden!

Standardspaltenbreite festlegen

Über die Eigenschaft *StandardWidth* im Eigenschaftenfenster können Sie die Standardspaltenbreite der Tabelle festlegen. Standardmäßig ist die Breite einer Spalte mit 10,71 festgelegt.

Eine Einheit der Spaltenbreite entspricht der Breite eines Zeichens in der standardmäßig eingestellten Schriftart. Je nach eingestellter Schriftart ist eine Umrechnung mehr oder weniger genau. Wenn Sie eine Umrechnung in Millimeter durchführen möchten, kann man gedanklich näherungsweise die Breite nach der Formel –0,71 + 5,1425 * Wunsch in mm / 10 berechnen. Hier wäre natürlich eine einfachere Umrechnung wünschenswert.

Zu jeder Eigenschaft können Sie übrigens die Taste F1 drücken, um weiterführende Informationen in der Onlinehilfe abzurufen.

Die ScrollArea setzen

Unter einer ScrollArea versteht man den Bereich einer Tabelle, in dem sich der »normale« Anwender aufhalten darf. Dieser Aufenthaltsbereich kann über das Eigenschaftenfenster festgelegt werden.

Setzen Sie einmal testweise die Eigenschaft *ScrollArea*, indem Sie wie folgt vorgehen:

1. Erfassen Sie im Eigenschaftenfenster bei der Eigenschaft *ScrollArea* den Bezug *A1:D10*.
2. Bestätigen Sie die Eingabe mit ↵. Die Eingabe wird augenblicklich mit absoluten Bezügen umfasst. So wird aus der Eingabe *A1:D10* die Eingabe *$A$1:$D$10*.
3. Wechseln Sie, nachdem Sie die ScrollArea gesetzt haben, über die Tastenkombination Alt+F11 in die Standardoberfläche von Microsoft Excel.
4. Wechseln Sie zu *Tabelle1*.
5. Versuchen Sie, den Bereich A1:D10 zu verlassen. Es wird Ihnen nicht gelingen.

In der Praxis wird diese Technik gern verwendet, um sensible Daten zu schützen. Daher können Sie, bevor Sie diese Eigenschaft verwenden, im Vorfeld Daten in einem entlegenen Teil der Tabelle erfassen und anschließend den Aufenthaltsbereich einschränken. So kommt der Standardanwender von Excel nicht an Ihre Daten und kann sie daher auch nicht ändern.

Leider wird ein geänderter Wert der Eigenschaft *ScrollArea* nicht aufrechterhalten, wenn Sie die Eigenschaft einstellen und danach die Arbeitsmappe speichern und schließen. Nach erneutem Öffnen der Arbeitsmappe ist die vorher gesetzte Eigenschaft wieder weg. Was nun tun, um sicherzustellen, dass diese Eigenschaft nach dem Öffnen der Arbeitsmappe gesetzt ist?

Die Lösung ergibt sich direkt aus dem Verhalten dieser Eigenschaft. Wenn der Eigenschaftswert nach dem Öffnen der Arbeitsmappe nicht mehr vorhanden ist, müssen Sie beim Öffnen dafür sorgen, dass er automatisch gesetzt wird.

Um die ScrollArea dauerhaft einzurichten, greifen Sie in die Trickkiste und richten ein sogenanntes Ereignis ein. Gehen Sie dabei wie folgt vor:

1. Klicken Sie im Projekt-Explorer doppelt auf das Objekt *DieseArbeitsmappe*.
2. Wählen Sie im rechts daneben eingeblendeten Fenster im linken Drop-down-Listenfeld den Eintrag *Workbook*. Daraufhin wird automatisch das Ereignis *Workbook_Open* eingestellt.
3. Ergänzen Sie den noch leeren Rahmen des Ereignisses wie in Abbildung 1.7 gezeigt.

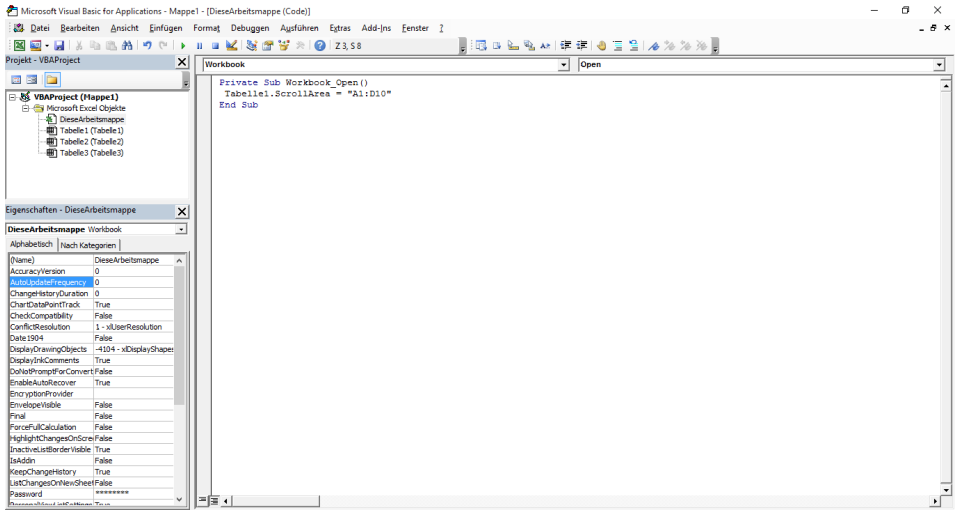


Abbildung 1.7: Das Ereignis Workbook_Open wird automatisch beim Öffnen der Arbeitsmappe ausgeführt.

4. Speichern Sie Ihre Arbeitsmappe. Danach schließen und öffnen Sie die Arbeitsmappe erneut. Sie werden feststellen, dass die ScrollArea jetzt »dauerhaft« eingestellt ist.

Auf diese Art und Weise können Sie zu jeder Tabelle ganz individuell eine ScrollArea einrichten.

Möchten Sie beispielsweise eine Tabelle gänzlich unveränderbar gestalten, ist die kleine ScrollArea durch den Zellenbezug *A1* zu definieren. In einer Tabelle, in der dieser Bezug angegeben wird, ist keinerlei »Bewegung« möglich – auch eine Art von Datenschutz, meinen Sie nicht auch?

Eine ScrollArea ist ein zusammenhängender Bereich. Sie können nicht mehrere ScrollAreas auf einer Tabelle definieren. Wie Sie diese Standardeinstellung umgehen können, verrate ich Ihnen in Kapitel 7 »Ereignisprogrammierung in Excel«.

Den Namen einer Tabelle festlegen

Wenn Sie im Eigenschaftenfenster genau hinsehen, erkennen Sie, dass es zwei Eigenschaften gibt, um den Namen einer Tabelle festzulegen – die beiden Eigenschaften *Name* und *(Name)* im Eigenschaftenfenster. Standardmäßig sind beide Eigenschaften mit dem gleichen Wert belegt. Dieser doppelte Name spiegelt sich auch im Projekt-Explorer wider. Was hat es nun auf sich mit den zwei Namen? Die Beantwortung dieser Frage ist derart wichtig, dass Sie dazu erst einmal ein Beispiel erhalten. Gehen Sie wie folgt vor:

1. Wechseln Sie aus der Entwicklungsumgebung heraus in die Normalansicht von Excel.
2. Klicken Sie direkt im Tabellenregister unten den Namen *Tabelle1* doppelt an.
3. Erfassen Sie den Namen *Test* und bestätigen Sie mit ↵.
4. Wechseln Sie über die Tastenkombination Alt+F11 zurück in die Entwicklungsumgebung.
5. Betrachten Sie das Ergebnis in Abbildung 1.8.

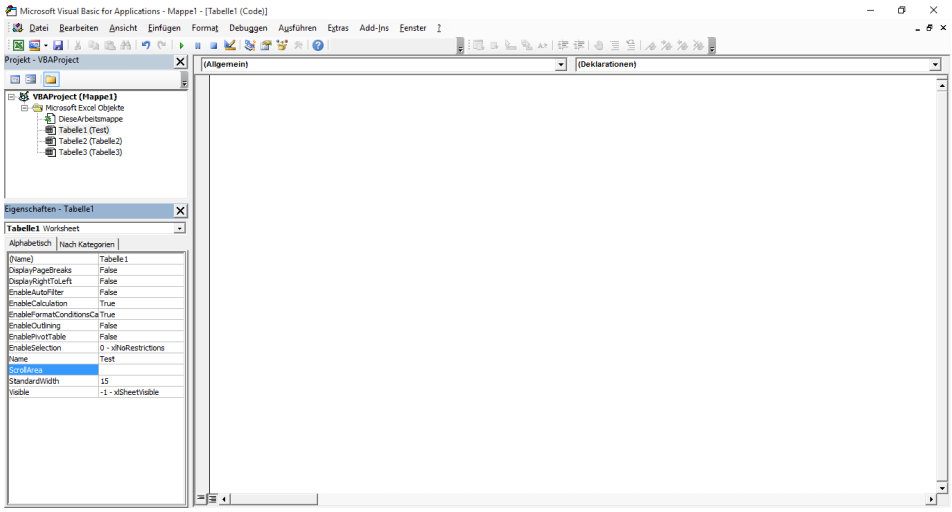


Abbildung 1.8: Wir stellen eine Abweichung der Tabellennamen fest.

Wie Sie sehen können, haben wir nun zwei verschiedene Tabellennamen. Beide Namen können auch jederzeit direkt über das Eigenschaftenfenster geändert werden. Worin besteht nun der Unterschied zwischen den beiden Tabellennamen, und welchen Tabellennamen sollten Sie für die Programmierung verwenden?

Nun, merken Sie sich Folgendes: Benutzen Sie immer den Tabellennamen, der im Projekt-Explorer an erster Stelle steht. Bei diesem Namen spricht man vom sogenannten Codenamen der Tabelle. Durch die Benutzung des Codenamens ersparen Sie sich gleich zu Beginn Ihrer Karriere als Entwickler viel Arbeit und Ärger. Auf die Vor- und Nachteile gehe ich im weiteren Verlauf des Kapitels noch genauer ein.

Bevor Sie überhaupt mit der Programmierung von Makros beginnen, sollten Sie die Tabellen so benennen, dass beide Namen idealerweise wieder gleich lauten, dabei aber auch sprechend sind. In der Praxis hat es sich bewährt, beim Tabellennamen ein Kürzel vorzugeben, damit auch im Makro später direkt erkannt werden kann, dass es sich um eine Tabelle handelt. So könnten Sie beispielsweise die ehemalige *Tabelle1* in beiden Eigenschaftenfeldern mit dem Namen *tbl_ErsteMakros* benennen.

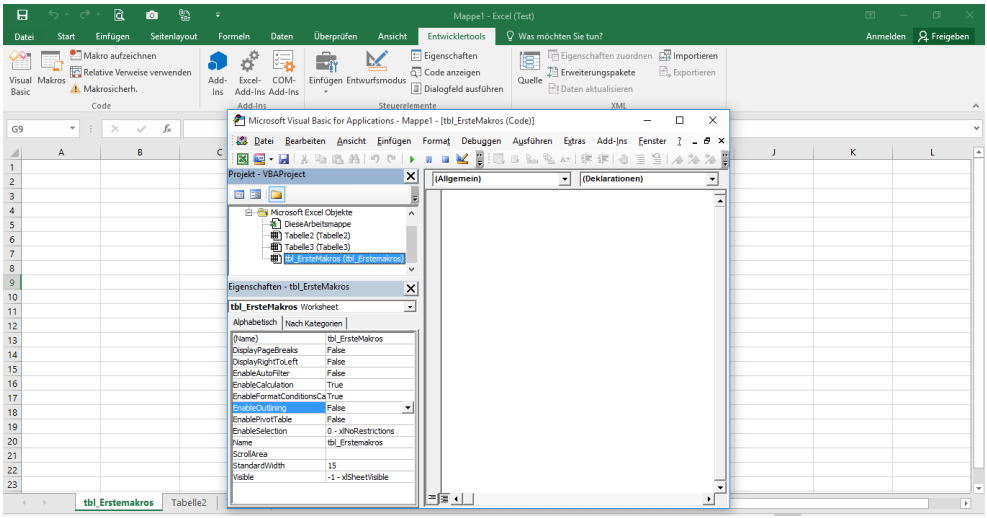


Abbildung 1.9: Sinnvolle Namen für Tabellen vergeben.

Bei der Benennung von Tabellen gilt es, bestimmte Einschränkungen zu beachten. So dürfen Sie beispielsweise bestimmte Zeichenfolgen beim Codenamen der Tabelle (die obere Eigenschaft im Eigenschaftenfenster) nicht verwenden. Unter anderem sind das Sonderzeichen sowie das Leerzeichen. Das erste Zeichen beim Codenamen der Tabelle darf auch keine Zahl sein.

Das Codefenster

Das Codefenster befindet sich standardmäßig rechts neben dem Projekt-Explorer. Dieses Fenster wird dann sichtbar, wenn Sie ein Modul anlegen. Ein Modul ist vergleichbar mit einem Ordner, der im Prinzip unsere zukünftigen Makros beinhaltet.

Gehen Sie wie folgt vor, um ein Modul anzulegen:

1. Wählen Sie aus dem Menü *Einfügen* den Befehl *Modul*.
2. Ersetzen Sie den standardmäßig vorgegebenen Namen *Modul1* durch einen sprechenden Namen, indem Sie das gerade eingefügte Modul im Projekt-Explorer markieren, danach den Namen *mdl_ErsteMakros* im Eigenschaftenfenster eintragen und das Ganze mit ↵ bestätigen.

Ich denke, dass es von Beginn an wichtig ist, Ordnung zu halten und klare Strukturen einzurichten. Daher geben Sie Ihren Tabellen, Modulen und auch Ihren Makros sprechende Namen. Sie machen es sich damit später leichter, Makros zu schreiben. Denken Sie dabei ebenfalls an Kollegen, die Ihre Makros eventuell verstehen und gegebenenfalls auch anpassen müssen, wenn Sie einmal im Urlaub sind. In diesem gerade angelegten Modul werden wir nachher unsere ersten Makros erfassen und starten.

Jedes Makro beginnt übrigens in VBA mit der Anweisung `Sub`. Der Begriff kommt aus dem Englischen und bedeutet so viel wie »Unter-(titel)«, was bedeutet, dass ein Makro unterhalb eines Moduls angesiedelt ist.

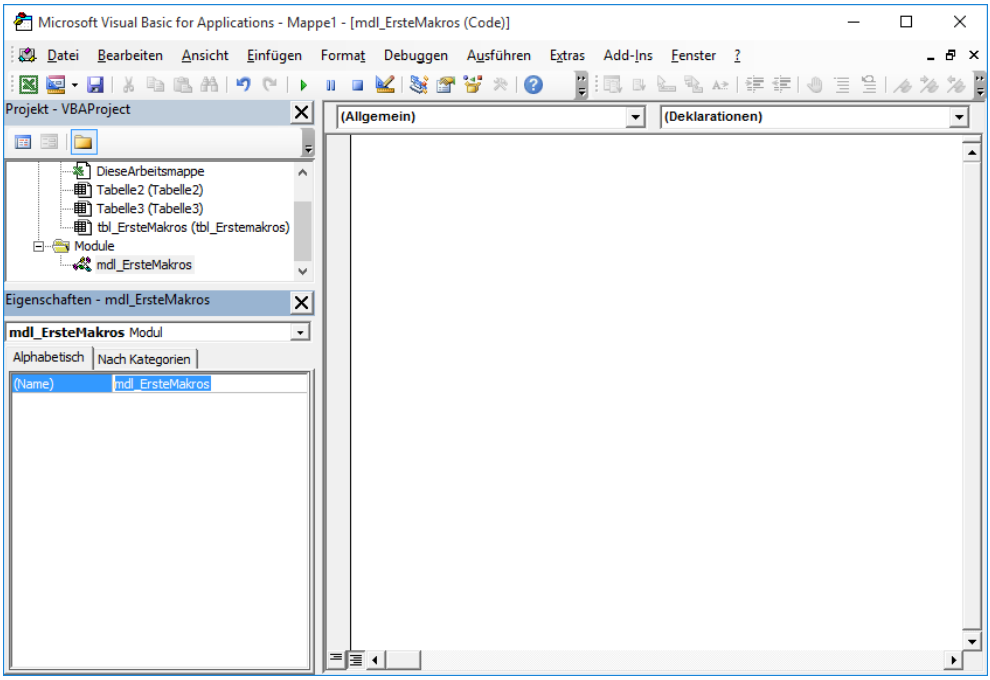


Abbildung 1.10: Auch Module sollten sprechende Namen haben.

Nach dem Schlüsselwort `Sub` folgt ein Leerzeichen. Direkt im Anschluss daran können Sie einen Namen für das Makro angeben. Bedenken Sie dabei, dass für die Benennung von Makros folgende Punkte beachtet werden müssen:

- Das erste Zeichen muss ein alphanumerisches Zeichen sein.
- Der Makroname darf keine Leerzeichen enthalten.
- Es dürfen keine Sonderzeichen wie /, %, -, $, [,], ?, ! oder Ähnliche verwendet werden.

Nach dem Namen geben Sie ein rundes Klammernpaar ein und drücken die Taste ↵. Das Makro wird jetzt automatisch um die Anweisung `End Sub` ergänzt.

```
Sub DasErsteMakro()

End Sub
```

Momentan ist das Makro noch leer. Alle Anweisungen, die Sie innerhalb dieses Rahmens schreiben, werden abgearbeitet und nacheinander ausgeführt. Ergänzen Sie das Makro nun wie folgt, um beispielsweise den Namen des Anwenders auf dem Bildschirm auszugeben:

```
Sub DasErsteMakro()

   MsgBox Environ("Username")

End Sub
```

Listing 1.1: Den angemeldeten Anwender am Bildschirm ausgeben.

Möchten Sie das Makro starten, haben Sie dafür folgende Möglichkeiten:

- Setzen Sie den Mauszeiger auf die erste Zeile des Makros und drücken Sie die Taste F5.
- Setzen Sie den Mauszeiger auf die erste Zeile des Makros und wählen Sie aus dem Menü *Ausführen* den Befehl *Sub/Userform ausführen*.
- Setzen Sie den Mauszeiger auf die erste Zeile des Makros und klicken Sie in der Symbolleiste *Voreinstellung* auf das Symbol *Sub/Userform ausführen*.
- Wechseln Sie auf Ihre Excel-Arbeitsoberfläche und wählen Sie im Menüband *Entwicklertools* (alternativ im Menüband *Ansicht*) in der Gruppe *Code* das Symbol *Makros*. Im nun angezeigten Dialogfeld wählen Sie das Makro aus und klicken auf die Schaltfläche *Ausführen*.

In jeder beschriebenen Variante wird eine Meldung auf dem Bildschirm ausgegeben, in der der Anwendername angezeigt wird. Dies erreichen Sie, indem Sie die Funktion `MsgBox` einsetzen.

Das Direktfenster – die Testhilfe

Vielleicht zu Beginn noch nicht ganz so wesentlich, dafür später aber umso wichtiger ist das sogenannte Direktfenster. Dieses Fenster blenden Sie über das Menü *Ansicht* und den Befehl *Direktfenster* ein. Alternativ dazu können Sie auch die Tastenkombination Strg+G drücken, um das Direktfenster einzublenden.

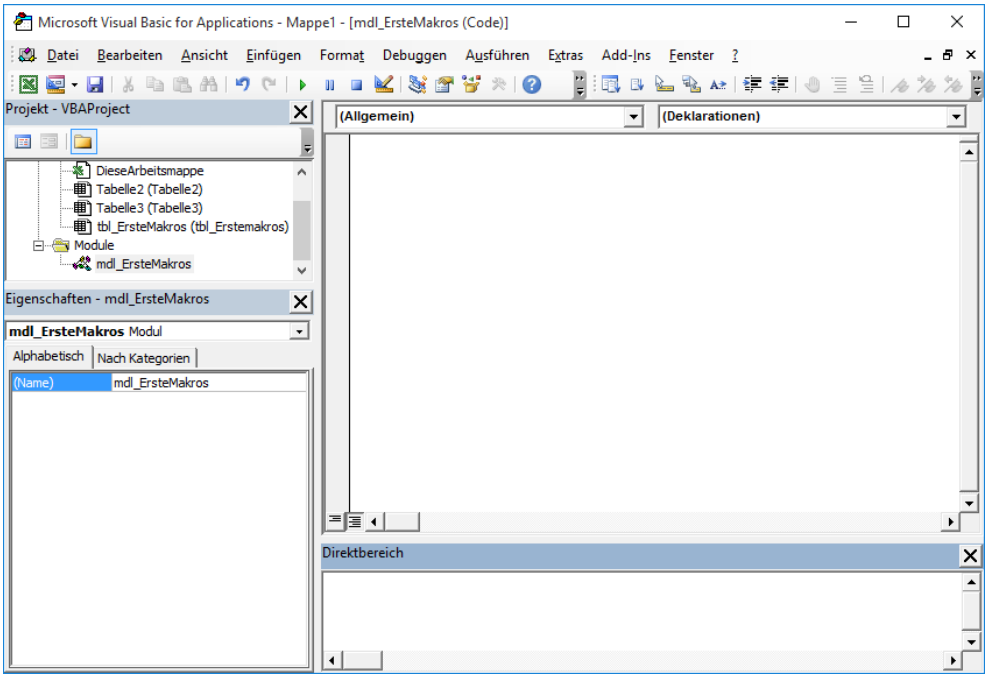


Abbildung 1.11: Das Direktfenster im unteren Bereich der Entwicklungsumgebung.

Über das Direktfenster können unter anderem Fehler in einem Makro gut gefunden und beseitigt werden. Wir werden im Verlauf des Buchs dieses Fenster häufiger einmal nutzen.

In der Praxis wird das Direktfenster auch gern eingesetzt, um Befehle direkt abzusetzen, ohne gleich ein eigenes Makro schreiben zu müssen. Daher folgen an dieser Stelle einmal einige Beispiele, wie Sie das Direktfenster verwenden können, um bestimmte Dinge abzufragen bzw. Aktionen zu starten.

Angemeldeten Nutzer ermitteln

Mit einem einzigen Befehl, den Sie bereits kennengelernt haben, können Sie abfragen, welcher Nutzer gerade in Windows angemeldet ist. Dazu verfahren Sie wie folgt:

1. Setzen Sie den Mauszeiger direkt in das Direktfenster.
2. Geben Sie als erstes Zeichen ein Fragezeichen an. Damit geben Sie bekannt, dass Sie nun eine Information benötigen.
3. Erfassen Sie als kompletten Befehl `?Environ("username")`.
4. Bestätigen Sie diesen Befehl mit ↵.

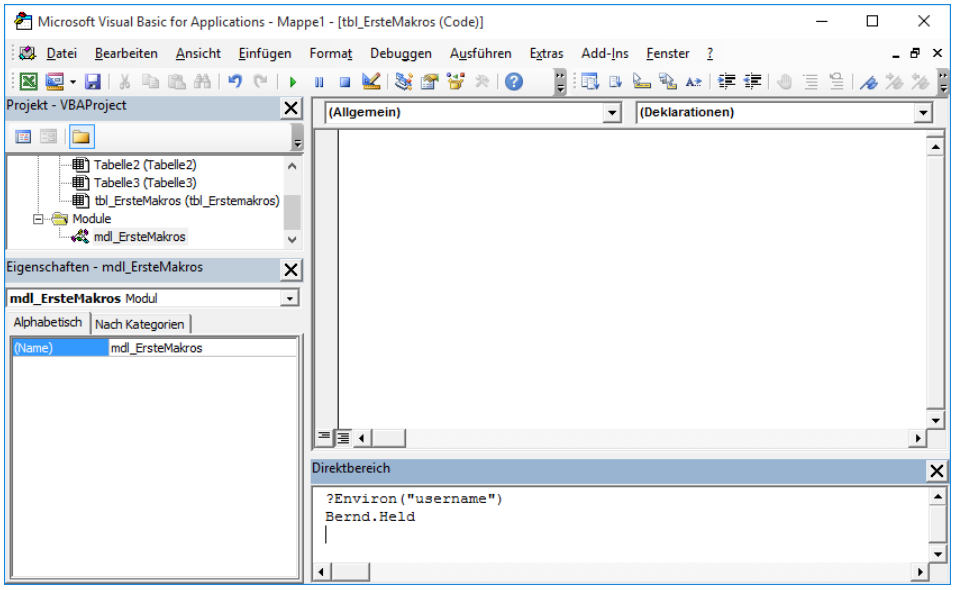


Abbildung 1.12: Das Ergebnis wird direkt unterhalb des eingegebenen Befehls ausgegeben.

Erfassen Sie spaßeshalber noch den Befehl `?Environ("Computername")`, um den Namen des Computers abzufragen, an dem Sie gerade sitzen. Diese beiden Befehle können später sehr gut eingesetzt werden, um eine Benutzerverwaltung in Excel aufzubauen. Diese werden wir gemeinsam in Kapitel 7, »Die Ereignisprogrammierung in Excel« erstellen.

Tabelle ein- und ausblenden

Erinnern Sie sich noch? Wir haben zu Beginn des Kapitels über das Eigenschaftenfenster eine Tabelle aus- und wieder eingeblendet. Diesen Vorgang können Sie auch über das Direktfenster durchführen. Gehen Sie dazu folgende Arbeitsschritte:

1. Setzen Sie den Mauszeiger direkt in das Direktfenster.
2. Geben Sie den Namen der Tabelle ein, die Sie ausblenden möchten.
3. Erfassen Sie direkt nach dem Tabellennamen einen Punkt.

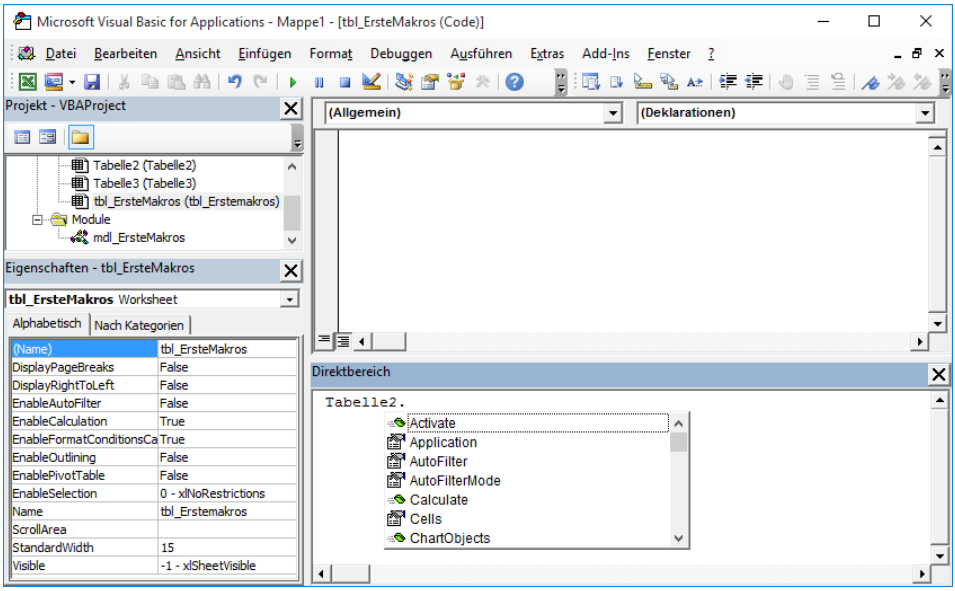


Abbildung 1.13: Excel hilft bei der Eingabe der Befehle.

4. Nach der Eingabe des Tabellennamens und dem Setzen des Punkts klappt automatisch ein Drop-down herunter, in dem nur die Befehle angeboten werden, die für eine Tabelle überhaupt möglich sind.
5. Geben Sie als ersten Buchstaben das *v* ein.

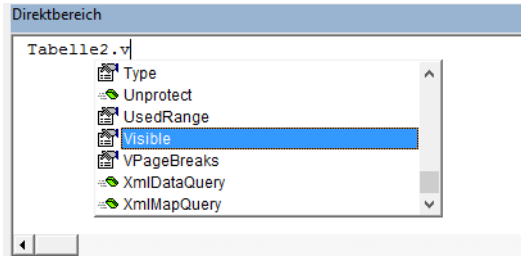


Abbildung 1.14: Der Mauszeiger springt automatisch zum ersten Befehl, der mit dem Buchstaben v beginnt.

6. Drücken Sie die Taste ⇆, um den Befehl zu übernehmen.

7. Nach der Auswahl der Eigenschaft `Visible` erfassen Sie das Gleichheitszeichen. Daraufhin klappt automatisch ein Drop-down herunter, in dem alle möglichen Konstanten zu dieser Eigenschaft auswählbar sind.

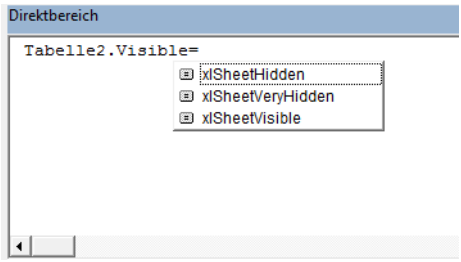


Abbildung 1.15: Übernehmen Sie die gewünschte Konstante.

8. Klicken Sie die Konstante `xlSheetHidden` an, um *Tabelle2* auszublenden.
9. Drücken Sie die Taste ↵, um den Befehl abzusetzen. Die Tabelle wird augenblicklich ausgeblendet.

Vielleicht haben Sie bemerkt, dass, nachdem Sie den Namen der Tabelle erfasst hatten, in dem Drop-down unterschiedliche Symbole bei den angebotenen Befehlen angezeigt wurden, und zwar Handsymbole und grüne Radiergummis. Bei Erstgenannten handelt es sich um Eigenschaften, bei den Radiergummis geht es um Methoden.

Was ist der Unterschied zwischen Eigenschaften und Methoden?

Eigenschaften beschreiben und charakterisieren ein Objekt, Methoden führen ganz konkret Aktionen mit dem Objekt durch. Für unser gerade vorgeführtes Beispiel ist das Ein- und Ausblenden einer Tabelle eine typische Eigenschaft, die den Status einer Tabelle beschreibt.

Eine typische Methode für das Objekt Tabelle wäre beispielsweise das Einfügen (=`Add`) bzw. das Löschen einer Tabelle (=`Delete`). Diese beiden Methoden führen eine Aktion mit der Tabelle durch.

Diese Unterscheidung ist aber nicht das Wesentliche daran, wichtig ist der Unterschied in der Syntax, der es notwendig macht, zu wissen, ob es sich um eine Eigenschaft oder eine Methode handelt.

Dazu ein kleines Beispiel, das die Syntax von Eigenschaften und Methoden näherbringen soll.

Die Syntax einer Eigenschaft am Beispiel des Ausblendens einer Tabelle lautet:

```
Tabelle2.Visible=xlSheetHidden
```

Eine Eigenschaft wird gesetzt, indem mit einem Gleichheitszeichen eine Konstante zugewiesen wird.

Die Syntax einer Methode am Beispiel des Einfügens einer neuen Tabelle lautet:

```
Worksheets.Add Before:=Worksheets(1)
```

Methoden haben sehr oft weitere Parameter, die automatisch angezeigt werden, wenn man die Methode aus dem Drop-down übernimmt und die Leertaste drückt. Dabei wird der Parametername übernommen, gefolgt von der Zeichenfolge `:=`.

In unserem Beispiel wird eine neue Tabelle zu Beginn der Arbeitsmappe eingefügt. Der Begriff `Worksheets` ist ein sogenanntes Auflistungsobjekt, d.h., in der Auflistung `Worksheets` sind automatisch alle Tabellen der Arbeitsmappe enthalten. Dieser Auflistung aller Tabellen fügen wir eine weitere Tabelle mithilfe der Methode `Add` hinzu. Die Methode `Add` enthält weitere Parameter, über die wir die Position der neuen Tabelle in der Mappe sowie die Anzahl der einzufügenden Tabellen festlegen können.

Wir werden im weiteren Verlauf des Buchs noch sehr oft mit diesem Thema in Berührung kommen.

Der Objektkatalog – das Nachschlagewerk

Innerhalb der Entwicklungsumgebung gibt es einen Katalog, in dem alle VBA-Befehle hinterlegt sind. Schauen wir uns diesen Objektkatalog einmal näher an, indem Sie die Taste F2 drücken.

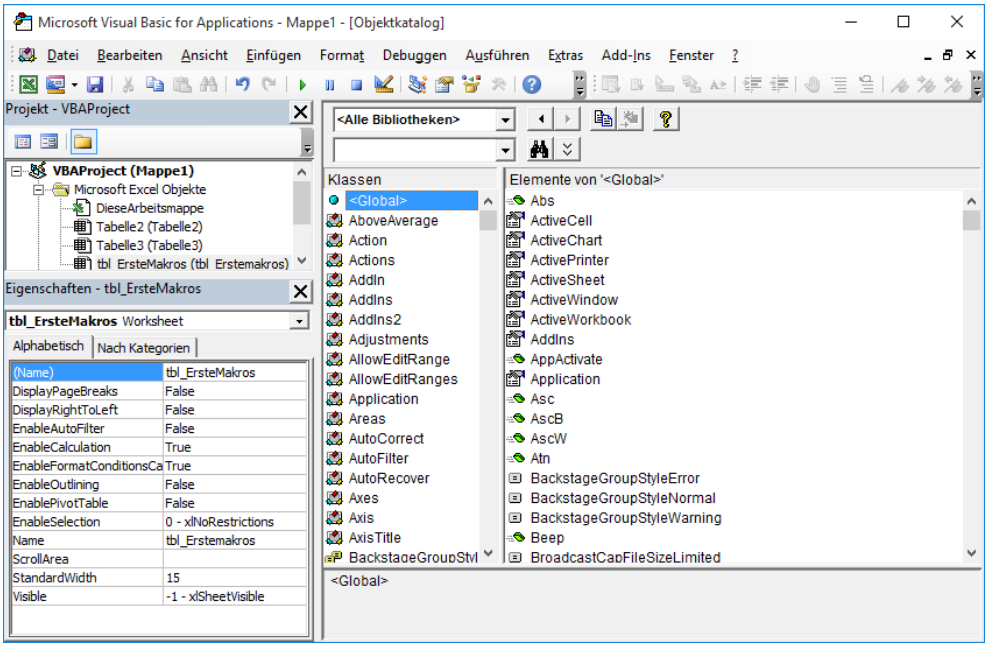


Abbildung 1.16: Der Objektkatalog gibt Auskunft über die verfügbaren Befehle von VBA.

Auf den ersten Blick wirkt dieser Katalog auf den Einsteiger erschlagend, da er Tausende von Befehlen enthält. Lassen Sie sich dadurch nicht abschrecken. Wie schon im Vorwort des Buchs erwähnt, reichen meiner Ansicht nach ca. 40 Befehle aus, um 90 % aller Aufgaben in Excel zu erledigen.

Schauen wir uns einmal das wichtigste Objekt in Excel an, die Zelle (`=Range`). Klicken Sie dazu in das Feld *Klassen* und geben Sie den Buchstaben *r* ein. Es werden jetzt alle Befehle angezeigt, die für das Objekt Zelle verfügbar sind.

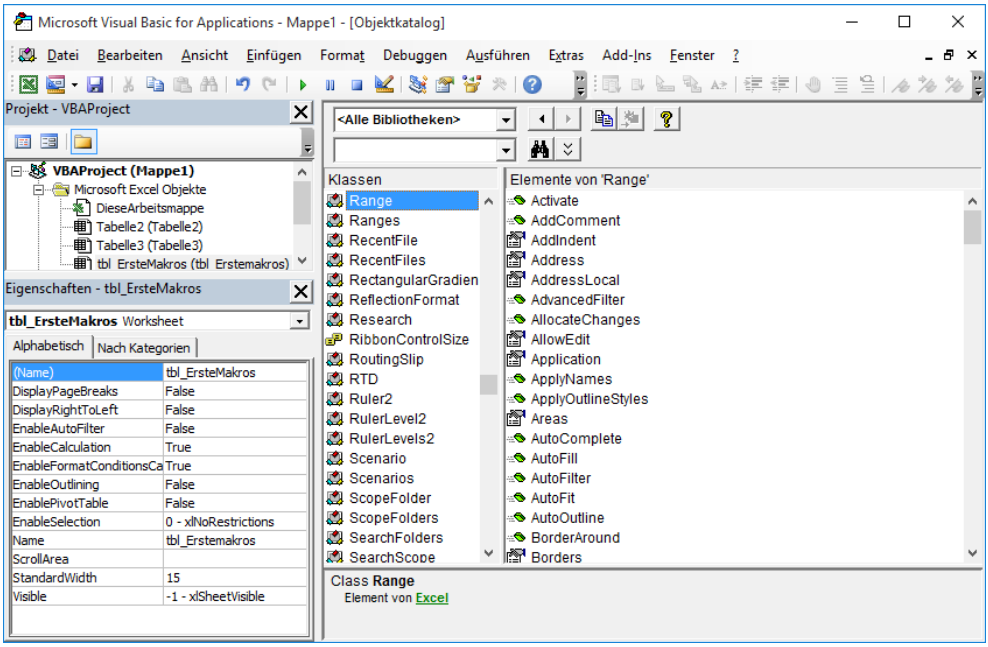


Abbildung 1.17: Alle Methoden und Eigenschaften, die für das Objekt Range verfügbar sind.

Die Zelle selbst ist die kleinste Einheit in Excel. Viele Zellen zusammen ergeben einen Bereich, viele Zellen untereinander ergeben eine Spalte, viele Zellen nebeneinander repräsentieren eine Zeile. Alles das ist ein einziges Objekt. Dies bedeutet, dass wir für all diese Dinge die gleichen Befehle verwenden können, die in Abbildung 1.17 gezeigt werden.

Im Objektkatalog werden die gleichen Befehle angeboten wie auch schon vorher im Direktfensterbeispiel, als wir den Unterschied zwischen Methoden und Eigenschaften am Objekt Tabelle erklärten.

Vielleicht sehen wir uns an dieser Stelle bereits ein Beispiel für eine typische Eigenschaft bzw. eine Methode speziell für das Objekt `Range` an. Geben Sie die beiden folgenden Befehle testhalber direkt im Direktfenster der Entwicklungsumgebung ein und drücken Sie danach die Taste ↵.

Die Syntax einer Eigenschaft am Beispiel einer Zellenadresse lautet:

```
?Activecell.Address
```

Die Adresse, also die Zellenkoordinate, ist eine typische Eigenschaft der Zelle, die eben beschreibt, welche Adresse die Zelle hat. Als Ergebnis wird bei Ihnen die Koordinate der aktiven Zelle ausgegeben.

Die Syntax einer Methode am Beispiel eines Zellenkommentars lautet:

```
Range("A1").AddComment Text:="Das ist eine Notiz"
```

Damit fügen Sie eine Zellennotiz in Zelle A1 der aktiven Tabelle ein. Die Methode `AddComment` ist eine typische Methode für das Objekt Zelle. Diese Methode hat den Parameter `Text`, der gefolgt von der Zeichenfolge `:=` und dem eigentlichen Kommentartext angegeben werden muss.

Alternativ und kürzer würde auch folgende Syntax funktionieren:

```
Range("A1").AddComment "Das ist eine Notiz"
```

Gerade bei diesem Beispiel verhält sich Excel etwas sonderbar. Wenn ich versuche, diesen Befehl zweimal hintereinander abzusetzen, erhalte ich beim zweiten Mal eine Fehlermeldung.

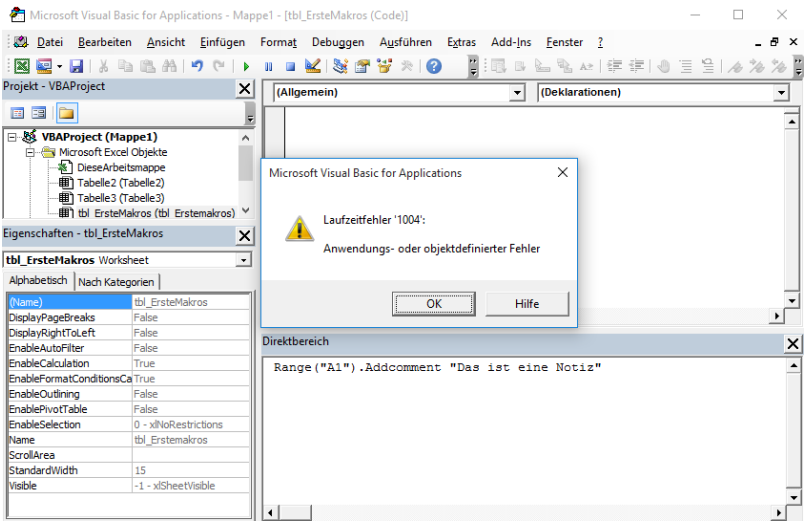


Abbildung 1.18: Eine relativ allgemein gehaltene Fehlermeldung.

Hierbei handelt es sich um einen sogenannten Laufzeitfehler. Da die Fehlermeldungen von Excel eher nicht so aussagekräftig sind, ist diesem Thema ein extra Kapitel im Buch gewidmet. Machen Sie sich also keine Sorgen, wenn Sie gerade zu Beginn Ihres Lernens des Öfteren einmal einen LZF (Laufzeitfehler) erhalten.

Wie Sie mit solchen Fehlern umgehen können, erfahren Sie in Kapitel 9 »Das Fehler-Handling« am Ende des Buchs.

Der Makrorekorder – zu Beginn eine gute Hilfe

Für den Einsteiger in Excel-VBA bietet der eingebaute Makrorekorder eine gute Möglichkeit, sich schnell Befehle anzueignen und ein Gefühl für die VBA-Syntax zu bekommen. Der Makrorekorder ist in der Lage, automatisch die notwendigen Befehle aufzuzeichnen, während Sie händisch eine Aufgabe in Excel erledigen.

Doch eines vorweg. Sie sollten jede Makroaufzeichnung verbessern und ausdünnen. Aufzeichnungen des Makrorekorders produzieren gut drei- bis viermal so viel Quellcode, wie eigentlich notwendig wäre, um die jeweilige Aufgabe zu erledigen. Der Rekorder zeichnet