# VBA-Programmierung

## Integrierte Lösungen mit Office 2019

Ricardo Hernández Garcia

1. Ausgabe, Juni 2019

ISBN 978-3-86249-864-2

VBA2019

Plus

HERDT

## Mit VBA programmieren



## Objektmodelle der Office-Anwendungen

## Benutzerdefinierte Dialoge und Datenbankzugriffe



## Integrierte Lösungen

# 1

# Bevor Sie beginnen

## Voraussetzungen und Ziele

### Zielgruppe

Dieses Buch richtet sich hauptsächlich an Office-Anwender und Softwareentwickler, die die Programmierung von integrierten Lösungen in Office 2019 erlernen möchten. Das Buch vermittelt alle erforderlichen Grundkenntnisse, um die Microsoft Office 2019-Anwendungen mittels der Programmiersprache VBA anwendungsübergreifend zu automatisieren und anzupassen.

Der Entwicklungsprozess wird mithilfe der VBA-Entwicklungsumgebung unterstützt und erlernt.

### Empfohlene Vorkenntnisse

Für das Erstellen von integrierten Lösungen mit VBA sind Kenntnisse zu folgenden Themen von Vorteil:

- Grundlagen der Programmierung
- Grundwissen in den benötigten Office-Anwendungen

Verfügen Sie über Kenntnisse in der Programmiersprache Visual Basic 2019 (VB) oder deren Vorgängerversionen, können Sie diese in Visual Basic for Applications (VBA) nutzen. Visual Basic 2019 und Visual Basic for Applications gehören zur gleichen Technologiefamilie. Der Unterschied zwischen beiden besteht darin,

- dass **VB** eine Programmiersprache zur Entwicklung eigenständiger Windows-Anwendungen ist,
- dass **VBA** dagegen ausschließlich der Automatisierung und Anpassung der Anwendungsprogramme dient, in die es eingebettet ist.

### Hinweise zu Soft- und Hardware

Es wird von einer Erstinstallation von Microsoft Office 2019 unter dem Betriebssystem Windows 10 und den Standardeinstellungen für die Office-Anwendungen ausgegangen. Abhängig von der Bildschirmauflösung bzw. der Hardware Ihres Computers kann das Aussehen der Symbole und Schaltflächen in Office 2019 und die Fensterdarstellung unter Windows 10 gegebenenfalls von den Abbildungen im Buch abweichen.

# Aufbau und Konventionen

## Typografische Konventionen

Damit Sie bestimmte Elemente auf einen Blick erkennen und zuordnen können, werden diese im Text durch eine besondere Formatierung hervorgehoben. Bezeichnungen für Programmelemente wie Register oder Schaltflächen werden immer *kursiv* geschrieben und wichtige Begriffe **fett** hervorgehoben.

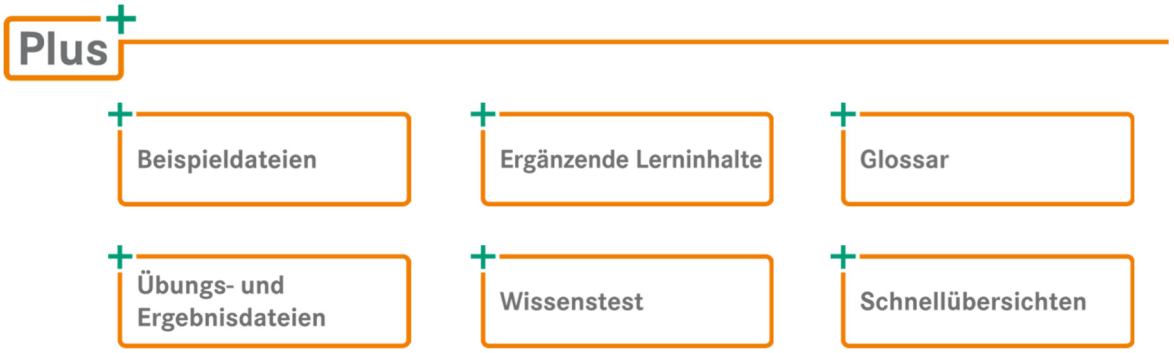
| | |
|---|---|
| `Courier New` | kennzeichnet Programmtext, Programmnamen, Funktionsnamen, Variablennamen, Datentypen, Operatoren etc. |
| *`Courier New Kursiv`* | kennzeichnet Zeichenfolgen, die vom Programm ausgegeben oder ins Programm eingegeben werden. |

# HERDT BuchPlus – unser Konzept:

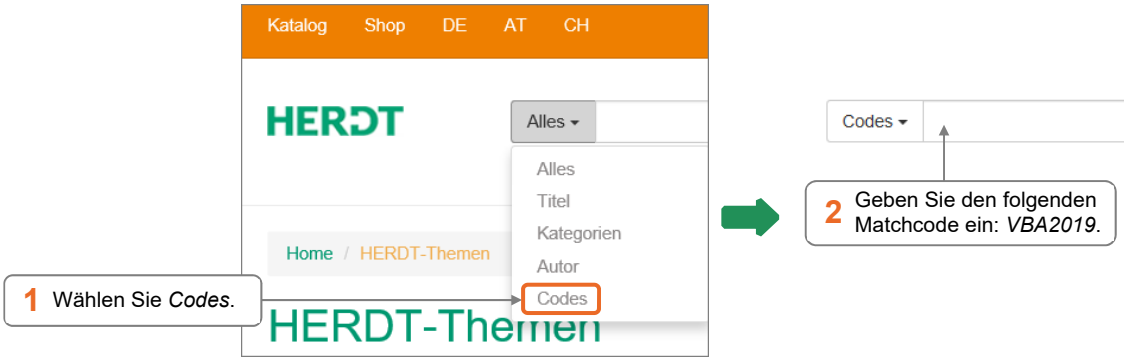
## Problemlos einsteigen – Effizient lernen – Zielgerichtet nachschlagen

(weitere Infos unter *www.herdt.com/BuchPlus*)

Nutzen Sie dabei unsere maßgeschneiderten, im Internet frei verfügbaren Medien:




- Rufen Sie im Browser die Internetadresse *www.herdt.com* auf.



# 2

# Office programmieren

**Beispieldatei:** *Kap02.docm*

## 2.1 Einsatzgebiete von VBA

### Arbeitserleichterung durch Makros

Das Office-Paket enthält Anwendungen, mit denen Sie Aufgaben der Textverarbeitung, Tabellenkalkulation, Datenverarbeitung, Präsentation und Planung erledigen können. Sie können alle diese Anwendungen ohne Programmierkenntnisse einsetzen.

Im Praxiseinsatz der Office-Anwendungen treten häufig immer gleiche Handlungen auf, die auch automatisiert werden könnten, um Zeit zu sparen. Zu diesem Zweck wurden sogenannte Makros eingeführt, die eine Folge von Tastaturanschlägen oder Mausbedienungen zu einer Einheit zusammenfassen konnten. Für die Aufzeichnung vom Makros werden ebenfalls keine Programmierkenntnisse benötigt.

Im Laufe der Zeit wurde die Makro-Sprache der Office-Anwendungen erweitert und immer weiter zu einer relativ umfassenden Programmiersprache entwickelt. Mithilfe dieser Programmiersprache ist es nun auch möglich, die Office-Anwendungen um benutzerdefinierte Funktionen zu erweitern.

### Potenzielle Sicherheitsrisiken und vorbeugende Maßnahmen

Bei der Automatisierung wird beispielsweise mit Makros gearbeitet, die auch sogenannte Makroviren enthalten können. Programmierer solcher Programme nutzen die Fähigkeiten von VBA, auf nahezu alle Informationen des Computers zuzugreifen. Über versteckte Makros sind beim Zugriff auf die Speichermedien auch Verschlüsselungs- und Löschvorgänge möglich, durch die vertrauliche Daten ausgelesen oder der Rechner und die enthaltenen Daten unzugänglich gemacht werden können.

Deshalb sind hier allgemeine vorbeugende Maßnahmen zu beachten, um die Sicherheit der Office-Dokumente und des Anwendungssystems zu gewährleisten.

Umfassende Informationen zu diesen potenziellen Sicherheitsrisiken und den entsprechenden vorbeugenden Office-Einstellungsmöglichkeiten sind in der Office-2019-Online-Hilfe enthalten (F1 und Eingabe von „*Sicherheitseinstellungen*“ im Hilfe-Suchfeld).

### Programmierung mit VBA

VBA (Visual Basic for Applications) bietet über umfassende Objektmodelle Zugriff auf die Office-Anwendungen und erlaubt eine strukturierte und modulorientierte Programmierung.

Um die Programmierung zu erleichtern, verfügt das Office-Paket über eine VBA-Entwicklungsumgebung, die als eigenständige Anwendung ausgeführt wird. Die Entwicklungsumgebung bietet Ihnen eine übersichtliche Arbeitsumgebung zum Programmieren und stellt Ihnen darüber hinaus noch einige Programmierhilfen und Arbeitserleichterungen zur Verfügung. Mit der Entwicklungsumgebung können Sie sowohl aufgezeichnete Makros bearbeiten als auch eigene Automatisierungs- und Erweiterungsroutinen erstellen.

Da VBA Microsoft-Forms und ActiveX-Steuerelemente unterstützt, können Sie eigene Dialogfenster und Benutzeroberflächen erstellen. Damit ist es beispielsweise möglich, die Dateneingabe über einen Dialog zu steuern und Dokumente mit diesen Daten automatisch zu erzeugen.

Durch den Einsatz von VBA können Sie z. B.:

- benutzerdefinierte Fehlermeldungen anzeigen,
- eigene Funktionen erstellen,
- Daten aus anderen Anwendungen verwenden (z. B. aus Excel oder Access),
- benutzerdefinierte Dialogfenster zur Abfrage verschiedener Eingabewerte oder zur Auswahl von Optionen erstellen,
- Dialogfenster über Steuerelemente steuern,
- System- bzw. Anwendungsinformationen auswerten (Benutzerdaten, Positionen von Grafiken, Dokumentstatistiken usw.).

## 2.2 Integrierte Lösungen

### Was sind integrierte Lösungen?

Jede Office-Anwendung ist auf ein bestimmtes Aufgabengebiet spezialisiert. Bei der praktischen Arbeit kann es immer wieder vorkommen, dass Sie die Funktionen und Möglichkeiten einer anderen Anwendung benötigen. Möglicherweise wollen Sie Daten aus einer Datenbank in Excel grafisch auswerten oder Terminfunktionen von Outlook in Word nutzen. In diesen Fällen können Sie von der Integrationsfähigkeit der Office-Anwendungen profitieren.

Alle Office-Anwendungen stellen ihre eigenen Funktionen als Objekte für andere Office-Anwendungen zur Verfügung. Gleichzeitig erlaubt jede Office-Anwendung den Zugriff auf andere Office-Objekte. Sie können daher beispielsweise in Word auf Excel-Objekte zugreifen und diese bei der Programmierung mit VBA verwenden wie in Excel selbst.

Integrierte Lösungen mit VBA helfen Ihnen, unter Verwendung mehrerer Office-Anwendungen, die Arbeit eines Anwenders zu vereinfachen. Integrierte Lösungen können jedoch auch verwendet werden, um fehlende Funktionen einer Office-Anwendung durch die Funktionalität einer anderen Office-Anwendung nachzurüsten.

## Beispiele für integrierte Lösungen

- Über das Betätigen einer Schaltfläche können Sie eine Adresse aus den Outlook-Kontakten auswählen und in ein Word-Dokument einfügen.
- Sie importieren Umsatz- und Budgetzahlen aus einer Access-Datenbank, um sie in Excel detailliert auszuwerten und um anschauliche Diagramme zu erstellen.
- Sie nutzen die Druckfunktion von Word, um mehrere Seiten einer umfangreichen Excel-Tabelle auf ein einziges Blatt Papier zu drucken.

## Wann ist eine Integration sinnvoll?

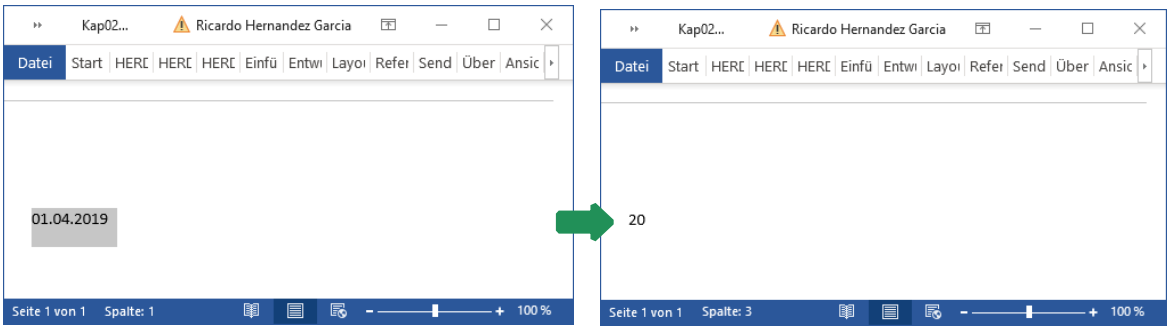
Eine integrierte Lösung ist immer dann sinnvoll und notwendig, wenn innerhalb einer Aufgabe mehrere Teilaufgaben existieren, für die bestimmte Office-Anwendungen durch ihre Spezialisierung besondere Vorteile bieten.

Viele Aufgabenstellungen lassen sich nur mithilfe von integrierten Lösungen mit vertretbarem Aufwand lösen. Sollen beispielsweise komplexe und miteinander verbundene Daten (z. B. Umsatz, Menge, Preis, Kosten) verwaltet und später grafisch aufbereitet werden, ist dies ohne eine Integration von Access und Excel nicht möglich. Access bietet als relationale Datenbank die Möglichkeit, große Mengen anfallender Daten zu speichern, und Excel verfügt über passende Funktionen zur grafischen Aufbereitung der Daten.

# 2.3 Ein einfaches Beispiel für eine integrierte Lösung

In der Buchführung wird häufig von einem Jahr mit 12 Monaten von je 30 Tagen ausgegangen. Beim Schreiben eines kaufmännischen Textes in Word kann es vorkommen, dass Sie die Anzahl von Tagen zwischen zwei Datumswerten basierend auf einem Jahr mit 360 Tagen benötigen. Word verfügt nicht über eine Funktion, mit der Sie diesen Wert direkt ermitteln können. Abhilfe schafft hier eine VBA-Prozedur, die eine geeignete Excel-Funktion aufruft.

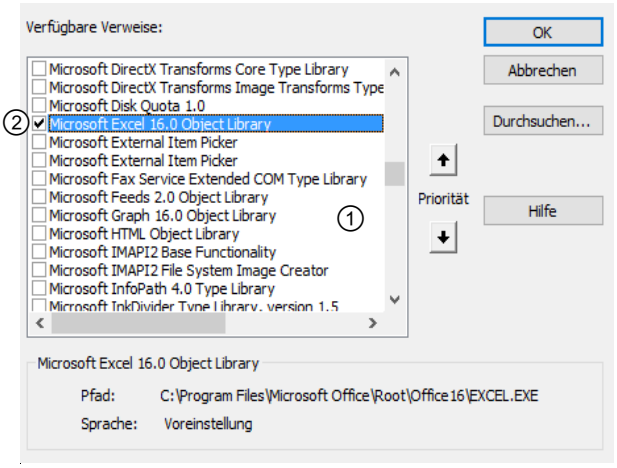
Die Prozedur `AnzahlTage360` liest ein im Word-Dokument ausgewähltes Datum aus und übergibt es zusammen mit dem aktuellen Datum an die Excel-Funktion `Day360`. Die ermittelte Anzahl von Tagen zwischen diesen beiden Daten wird danach in das Word-Dokument eingefügt und ersetzt das ausgewählte Datum.




Die VBA-Prozedur liest ein markiertes Datum (01.04.2019) aus und fügt die Anzahl der Tage bis zum aktuellen Datum (im Beispiel: 21.04.2019) in kaufmännischer Form ein

Für den Zugriff von Word auf die Excel-Objekte benötigt VBA einen Verweis auf die Excel-Objektbibliothek:

- Öffnen Sie in Word die VBA-Entwicklungsumgebung mit Alt F11.
- Wählen Sie den Menüpunkt *Extras - Verweise*.
- Suchen Sie im Listenfeld ① den Eintrag *Microsoft Excel 16.0 Object Library* und aktivieren Sie das Kontrollfeld ② in dieser Zeile.
- Bestätigen Sie mit *OK*.




Der Verweis bezieht sich immer nur auf das aktuelle Dokument, im Beispiel das aktuelle Word-Dokument.

## Beispiel: *Kap02.docm,* Modul **ExcelZugriff**

Die Prozedur ermittelt mithilfe von Excel die Anzahl der Tage, die zwischen einem im Word-Dokument markierten Datum und dem heutigen Datum liegen.

```
   Sub AnzahlTage360()
①    Dim AppXL As Excel.Application
     Dim Datum As Date, Heute As Date, AnzahlTage As Integer
②    Heute = Date
③    Set AppXL = CreateObject("Excel.Application")
④    Datum = DateValue(Selection.Text)
⑤    AnzahlTage = AppXL.WorksheetFunction.Days360(Datum, Heute)
⑥    Selection.TypeText AnzahlTage
⑦    Set AppXL = Nothing
   End Sub
```

① Für den Zugriff auf Excel wird eine Objektvariable benötigt. Über diese Variable kann auf die Excel-Objekte zugegriffen werden wie bei der Programmierung in Excel selbst.

② Die VBA-Funktion `Date` liefert das aktuelle Datum.

③ Mit der `CreateObject`-Funktion wird die Excel-Anwendung im Hintergrund gestartet. Danach sind alle Excel-Objekte verfügbar.

④ Der ausgewählte Text im Word-Dokument wird hier ausgelesen und in ein für die Prozedur verarbeitbares Datumsformat umgewandelt.

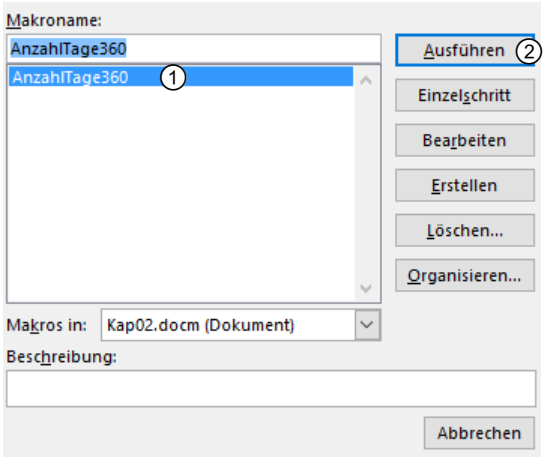
⑤ Die Objektvariable `AppXL` repräsentiert die Excel-Anwendung. Über das `WorksheetFunction`-Objekt kann auf die Excel-Funktionen zugegriffen werden. Hier erfolgt der Aufruf der Funktion `Day360` mit dem im Dokument ausgelesenen Datum und dem aktuellen Datum als Argument.

⑥ Die `Day360`-Funktion liefert die Anzahl der Tage zwischen den beiden Daten. Dieser Wert wird an der aktuellen Cursorposition im Dokument eingefügt. Liegt das ausgelesene Datum in der Zukunft, erhält das Ergebnis ein Minuszeichen vor dem Wert.

⑦ Die Excel-Anwendung wird nicht mehr benötigt. Durch das Zuweisen von `Nothing` wird die Anwendung geschlossen und der benutzte Speicherplatz freigegeben.

Arbeiten Sie häufiger mit Makros, können Sie über das Kontextmenü des Menübands das Register *Entwicklertools* im Menüband dauerhaft einblenden (Kontextmenüpunkt *Menüband anpassen*, Kontrollfeld *Entwicklertools).*

- Markieren Sie im Word-Dokument die Datumsangabe.
- Um die Prozedur in Word zu starten, klicken Sie im Register *Entwicklertools* auf die Schaltfläche *Makros.*
- Alternative: Alt F8
- Markieren Sie im Dialogfenster *Makros* die Prozedur `AnzahlTage360` ①.
- Betätigen Sie *Ausführen* ②.




Makros können Viren enthalten. Deshalb wird die Ausführung von Makros aus nicht vertrauenswürdigen Quellen standardmäßig unterdrückt. Sobald Sie z. B. ein Dokument mit gespeicherten Makros öffnen, erhalten Sie eine Sicherheitswarnung (Benachrichtigung).

Um die Makros zu aktivieren, können Sie den gespeicherten Makros

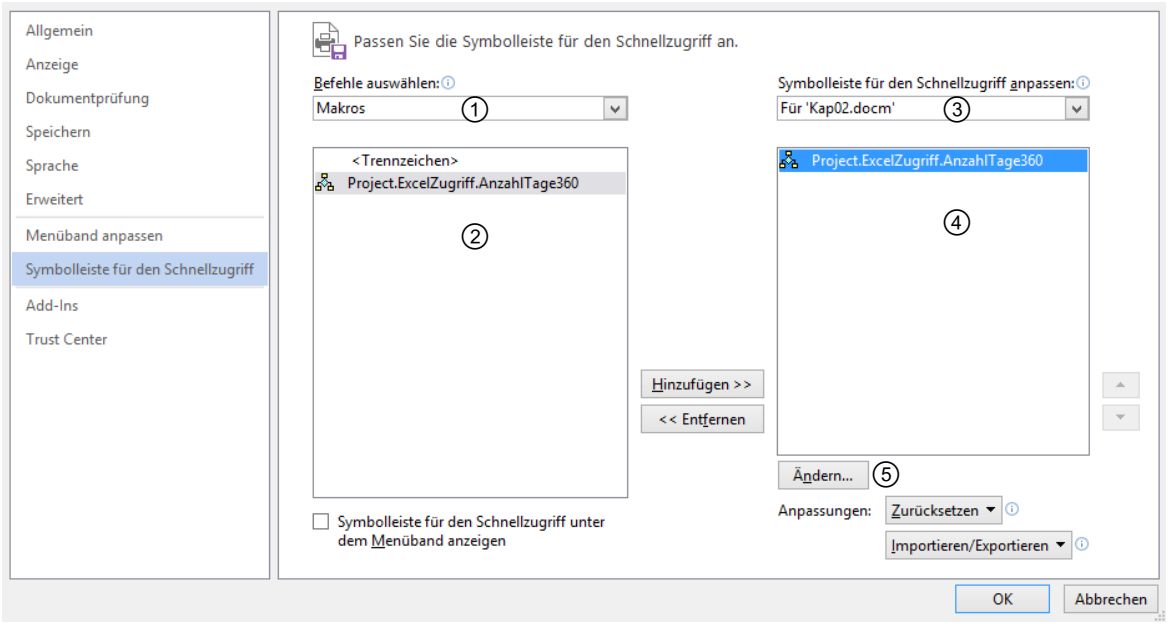
- einmalig vertrauen (Link *Makros wurden deaktiviert* der angezeigten Sicherheitswarnung, Schaltfläche *Inhalt aktivieren*, Auswahl *Erweiterte Optionen*, Optionsfeld *Inhalt für diese Sitzung aktivieren*), indem Sie das Dokument als vertrauenswürdig deklarieren (Schaltfläche *Inhalt aktivieren* der angezeigten Sicherheitswarnung)

  *oder*

- immer vertrauen, indem Sie den Speicherort der Makros als vertrauenswürdigen Speicherort eintragen (Register *Datei,* Schaltfläche *Optionen/Trust Center/Einstellungen für das Trust Center*, Kategorie *Vertrauenswürdige Speicherorte*).

## 2.4 Makros der Symbolleiste für den Schnellzugriff hinzufügen

Haben Sie ein Makro erstellt und getestet, können Sie für dessen Start ein Symbol in der Symbolleiste für den Schnellzugriff hinzufügen.




*Symbole für Makros der Symbolleiste für den Schnellzugriff hinzufügen*

- Klicken Sie mit der rechten Maustaste auf die Symbolleiste für den Schnellzugriff und wählen Sie den Kontextmenüpunkt *Die Symbolleiste für den Schnellzugriff anpassen*.

  Das Dialogfenster *Word-Optionen* wird bei aktivierter Kategorie *Passen Sie die Symbolleiste für den Schnellzugriff an.* geöffnet.
- Markieren Sie im Feld ① den Eintrag *Makros*.

  Im Bereich ② werden die zur Verfügung stehenden Makros angezeigt.
- Markieren Sie das gewünschte Makro.
- Wählen Sie im Feld ③, in welchen Dokumenten die Symbolleiste angepasst werden soll.

  Im Beispiel wird die Symbolleiste nur für das Word-Dokument *Kap02.docm* geändert.

  Standardmäßig wird die Symbolleiste für alle Dokumente geändert.
- Klicken Sie auf die Schaltfläche *Hinzufügen*.

  Das betreffende Element wird in die Symbolleiste für den Schnellzugriff aufgenommen und im Bereich ④ angezeigt.

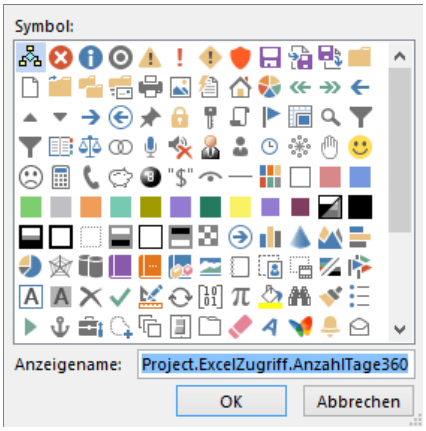
Die Markierung im Bereich ② springt automatisch auf den nächsten Eintrag.

### Standardsymbole ändern

Standardmäßig werden Makros in der Symbolleiste für den Schnellzugriff mit dem Symbol  und dem Namen des Makros in der QuickInfo angezeigt.

- Um das Symbol oder die QuickInfo zu ändern, markieren Sie das Makro im Bereich ④.
- Klicken Sie auf die Schaltfläche *Ändern* ⑤.

  Das Dialogfenster *Schaltfläche 'Ändern'* wird geöffnet.




| Sie möchten … | |
|---|---|
| das Symbol ändern | ▸ Wählen Sie das gewünschte Symbol. |
| die QuickInfo ändern | ▸ Geben Sie im Feld *Anzeigename* eine neue Bezeichnung ein. |

- Bestätigen Sie zweimal mit *OK*.

## Besonderheiten beim Löschen von Makros

Löschen Sie ein Makro, für das ein Symbol in der Symbolleiste für den Schnellzugriff existiert, bleibt das Symbol erhalten. Betätigen Sie das Symbol, wird ein Warnhinweis ausgegeben.

- Klicken Sie mit der rechten Maustaste auf das betreffende Symbol und wählen Sie den Kontextmenüpunkt *Aus Symbolleiste für den Schnellzugriff entfernen*, um das Symbol für das Makro zu löschen.

  Das Symbol wird ohne Rückfrage gelöscht.

Das beschriebene Verfahren, um ein Symbol eines Makros der Symbolleiste für den Schnellzugriff hinzuzufügen, kann in dieser Form in allen Office-Anwendungen verwendet werden.

# 3

# Die VBA-Entwicklungsumgebung

**Beispieldatei:** *Kap03.docm*

## 3.1 VBA-Entwicklungsumgebung verwenden

**Beispieldatei:** *Kap03-Datenbank.accdb*

Die VBA-Programmierung erfolgt in allen Office-Anwendungen im Visual Basic-Editor. Dieser Editor ermöglicht nicht nur die Eingabe des Programmcodes, sondern stellt eine Vielzahl von Hilfsmitteln zur Verfügung (z. B. zum Ermitteln von Fehlern oder zum Finden verschiedener Objekte). Daher wird auch der Begriff VBA-Entwicklungsumgebung verwendet.

### Öffnen der VBA-Entwicklungsumgebung

| Sie möchten die VBA-Entwicklungsumgebung … | |
|---|---|
| in Word, Excel, Outlook oder PowerPoint starten | ▶ Klicken Sie im Register *Entwicklertools* in der Gruppe *Code* auf die Schaltfläche *Visual Basic*. |
| in Access starten | ▶ Klicken Sie im Register *Datenbanktools* in der Gruppe *Makro* auf die Schaltfläche *Visual Basic*. |

In allen Office-Anwendungen können Sie die VBA-Entwicklungsumgebung schnell mit Alt F11 starten.

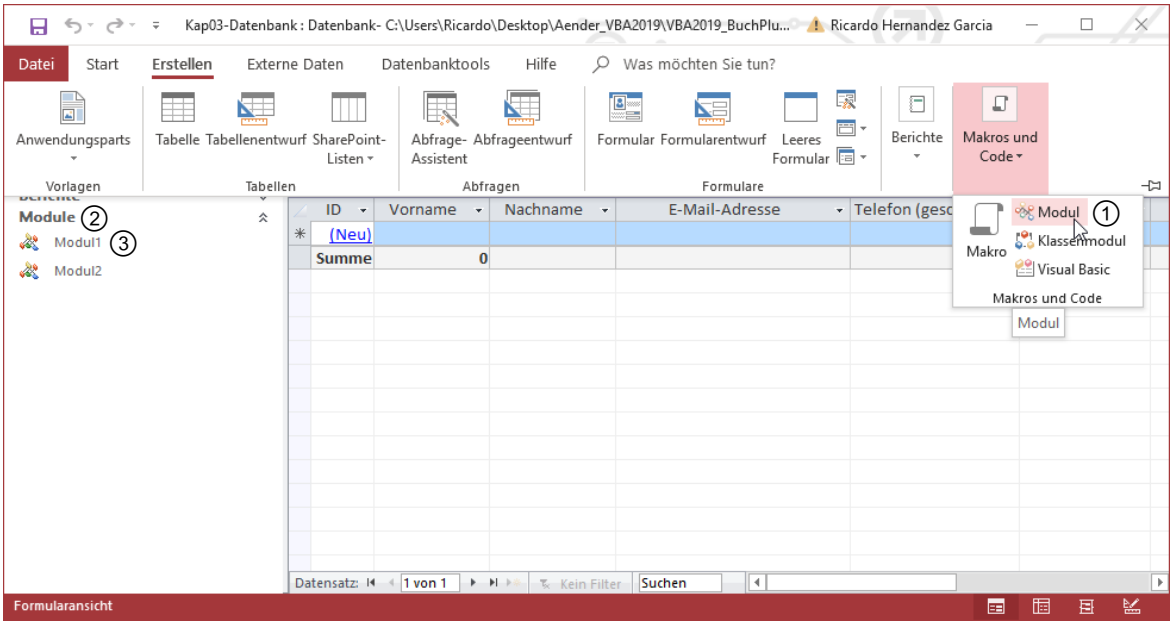
Die VBA-Entwicklungsumgebung wird als eigenständige Anwendung geöffnet und erscheint damit nicht innerhalb der Office-Anwendungen.

### Besonderheiten in Access

Zusätzlich zu den bereits beschriebenen Varianten wird die VBA-Entwicklungsumgebung automatisch geöffnet, wenn Sie im Access-Datenbankfenster ein Modul neu erstellen oder im Entwurfsmodus öffnen.

## Access-Modul erstellen

- Klicken Sie im Register *Erstellen* in der Gruppe *Makros und Code* auf das Modul-Symbol ①.




*Das Datenbankfenster in Access*

Die VBA-Entwicklungsumgebung wird automatisch geöffnet und ein neues Modul erstellt.

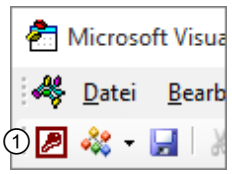
## Access-Modul bearbeiten

- Zeigen Sie im Navigationsbereich des Datenbankfensters die Gruppe *Module* ② an.
- Markieren Sie im Inhaltsbereich das Modul, das Sie bearbeiten möchten ③.
- Um das Modul in der VBA-Entwicklungsumgebung zu öffnen, rufen Sie den Kontextmenüpunkt *Entwurfsansicht* auf.

*oder*

- Klicken Sie doppelt auf das Modul, das Sie bearbeiten möchten.

## Die VBA-Entwicklungsumgebung verlassen




- Wählen Sie in der Symbolleiste *Voreinstellung* das Symbol ①.

  Das Symbol zeigt die Office-Anwendung (hier: Access), aus der Sie die VBA-Entwicklungsumgebung aufgerufen haben.

  Alternative: Alt F11

Mit dem Symbol bzw. der Tastenkombination wechseln Sie zur Ausgangsanwendung. Die VBA-Entwicklungsumgebung bleibt im Hintergrund geöffnet. Schließen Sie die Entwicklungsumgebung wie folgt:

- Wählen Sie in der Entwicklungsumgebung den Menüpunkt *Datei - Schließen und zurück zu Office-Anwendungsname*.

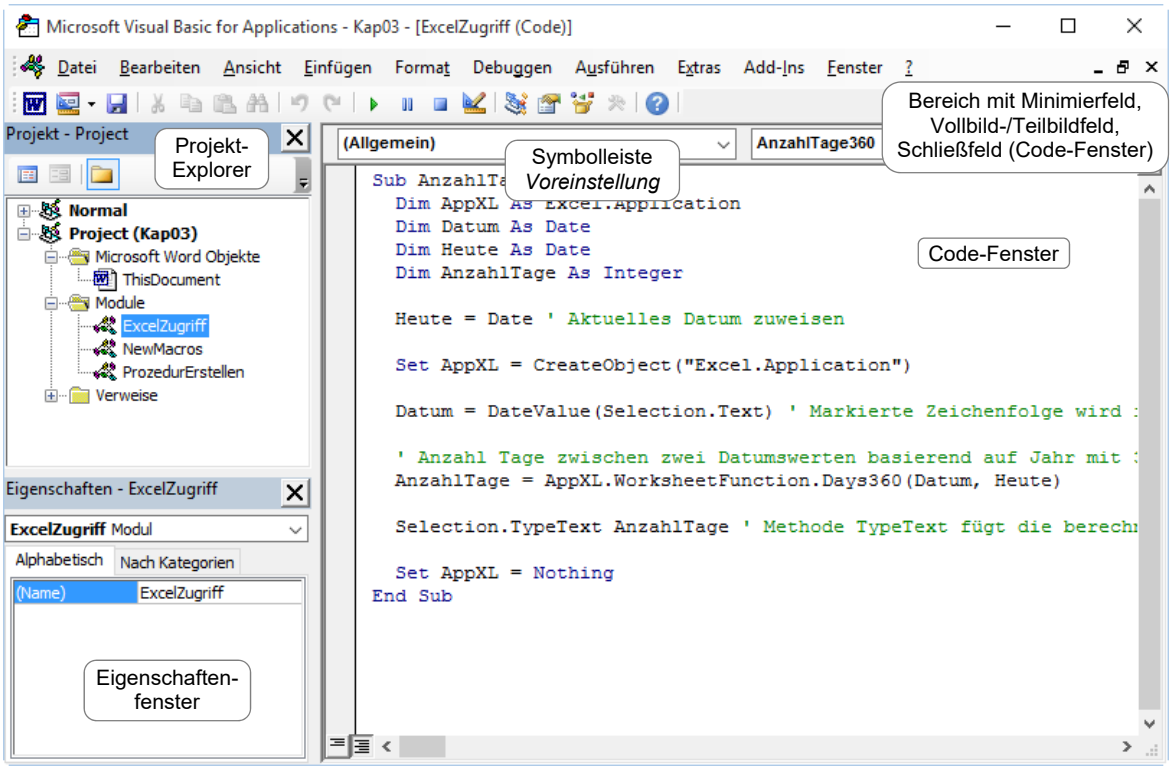
  Alternative: Alt Q

## 3.2 Bestandteile der Entwicklungsumgebung

### Was beinhaltet die VBA-Entwicklungsumgebung?

In der VBA-Entwicklungsumgebung werden die VBA-Programme codiert, getestet und verwaltet. Dafür beinhaltet die VBA-Entwicklungsumgebung unter anderem:

- einen **Editor**, der Sie bei der Eingabe und beim Bearbeiten von Programmcode unterstützt;
- **Werkzeuge** zum Testen und zum schrittweisen Abarbeiten (Debuggen) des VBA-Programms;
- eine **Projektverwaltung**, die abhängig von der jeweiligen Office-Anwendung die entsprechenden Bestandteile des Projektes organisiert.



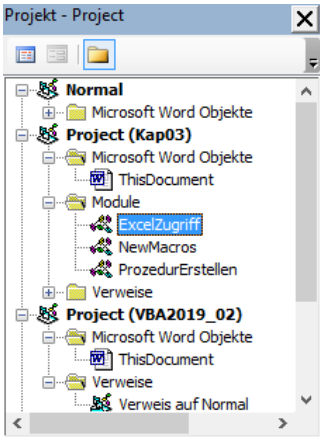

*Die VBA-Entwicklungsumgebung (Visual Basic-Editor)*

## 3.3 Der Projekt-Explorer

### Wozu dient der Projekt-Explorer?

Im Projekt-Explorer werden alle Bestandteile des gerade bearbeiteten Projekts hierarchisch geordnet (in Baumform) dargestellt.

Je nach Office-Anwendung werden unterschiedliche Projektbestandteile angezeigt. Dazu gehören alle geöffneten Dokumente bzw. Arbeitsmappen sowie Module, die Prozeduren aufnehmen. Auch die in Excel und Word automatisch geladenen Add-Ins und Vorlagen gehören zu einem Projekt.




*Anzeige des Projekt-Explorers in Word*