Was ist VBA?

1

In diesem Kapitel

- Ein Überblick über VBA
- Herausfinden, was man mit VBA machen kann
- Vor- und Nachteile, die sich aus der Verwendung von VBA ergeben
- Ein kleiner Rückblick auf die Geschichte von Excel

Dieses Kapitel ist frei von praktischen Übungen. Es enthält aber einige wichtige Hintergrundinformationen, die Ihnen dabei helfen, ein richtiger Excel-Programmierer zu werden. In anderen Worten: Dieses Kapitel ebnet den Weg für alles andere, was folgt, und vermittelt Ihnen ein Gefühl dafür, wie Excel-Programmierung in das Gesamtkonzept des Universums passt.

Was also ist VBA?

VBA, das für *Visual Basic for Applications* – im Deutschen *Visual Basic für Applikationen* – steht, ist eine Programmiersprache, die von Microsoft entwickelt wurde; Sie wissen schon, die Firma, die vom reichsten Mann der Welt geleitet wird. Excel enthält, neben den anderen Mitgliedern der Microsoft Office 2007-Familie, die Sprache VBA (und zwar ohne Zusatzkosten!). Kurzum: VBA ist das Werkzeug, das Menschen wie Sie und ich dazu verwenden, Programme zu schreiben, die Excel steuern.

Stellen Sie sich einen Roboter vor, der alles über Excel weiß. Dieser Roboter kann Befehle lesen und er kann Excel sehr schnell und exakt bedienen. Wenn Sie möchten, dass der Roboter etwas für Sie in Excel erledigt, schreiben Sie ein paar Befehle in einer speziellen Robotersprache auf. Sagen Sie Ihrem Roboter, dass er die Befehle ausführen soll, während Sie sich zurücklehnen und ein Glas Limonade genießen. So funktioniert VBA – eine Spezialsprache für Roboter. Beachten Sie aber bitte, dass Excel ohne Roboter und Limonade ausgeliefert wird ;-)

Verwechseln Sie VBA *nicht* mit VB (was für Visual Basic steht). VB ist eine Programmiersprache, mit der Sie ausführbare Programme (.exe-Dateien) erzeugen können. Obwohl VBA und VB viele Gemeinsamkeiten besitzen, handelt es sich um etwas vollkommen Unterschiedliches.

Ein paar Worte zur Terminologie

Die Excel-Programmierterminologie kann ein wenig verwirrend sein. So ist VBA beispielsweise eine Programmiersprache, wird aber auch als *Makrosprache* bezeichnet. Wie nennen Sie also etwas, das in VBA geschrieben und in Excel ausgeführt wird? Ist das ein Makro oder ist es ein Programm? Die Excel-Hilfe bezeichnet VBA-Prozeduren oft als *Makros*, daher verwende ich ebenfalls diese Terminologie. Ich nenne dieses Zeug aber auch *Programm*. Außerdem verwende ich die Bezeichnung *automatisieren* in diesem Buch. Dies bedeutet, dass eine Reihe von Schritten *automatisch* ausgeführt wird. Wenn Sie beispielsweise ein Makro schreiben, das ein paar Zellen farblich hervorhebt, das Arbeitsblatt ausdruckt und die Farben dann wieder entfernt, haben Sie diese drei Schritte automatisiert.

Übrigens: Makro steht *nicht* für **M**istige, **a**bsolut **k**onfuse, **r**echnergestützte **O**peration. Der Begriff kommt aus dem Griechischen – *Makros*, was groß bedeutet und auch gut Ihren Gehaltsscheck beschreibt, nachdem Sie ein Experte auf dem Gebiet der VBA-Makros geworden sind.

Was kann man mit VBA machen?

Sie gehören möglicherweise zu den Personen, die Excel für unglaublich viele verschiedene Aufgaben verwenden, zum Beispiel um

- ✔ Listen von Dingen, wie Kundennamen, Noten von Schülern oder Weihnachtsgeschenke zu verwalten
- ✔ Budgetierung und Vorhersagen zu erledigen
- ✔ Daten zu analysieren
- ✔ Rechnungen und andere Formulare zu erstellen
- ✔ aus Daten Diagramme zu erstellen und so weiter und so fort

Diese Liste kann man beliebig fortführen, aber ich denke, dass Sie verstanden haben, worum es geht. Ich will einfach nur sagen, dass Excel für extrem viele verschiedene Dinge verwendet wird und dass sicherlich jeder Leser dieses Buches andere Anforderungen und Erwartungen an Excel hat. Eine Sache, die aber fast jeder Leser benötigt, ist, eine bestimmte Funktion von Excel zu automatisieren. Das, liebe Leser, ist das, worum es bei VBA geht.

So können Sie beispielsweise ein Programm schreiben, das Ihre monatlichen Verkaufsberichte formatiert und ausdruckt. Nachdem Sie dieses Programm geschrieben und getestet haben, können Sie das Makro mit einem einzigen Befehl ausführen und so veranlassen, dass Excel automatisch viele zeitaufwendige Arbeiten erledigt. Anstatt eine langweilige Sequenz lästiger Befehle auszuführen, können Sie in Ruhe einen Kaffee trinken und den Computer die Arbeit erledigen lassen – so sollte es sein oder?

In den folgenden Abschnitten beschreibe ich kurz einige gebräuchliche Verwendungsmöglichkeiten für VBA-Makros.

Einen Haufen Text einfügen

Wenn Sie oft den Namen Ihres Unternehmens, die Adresse und die Telefonnummer in Ihre Arbeitsmappen eingeben müssen, können Sie ein Makro erstellen, das die Tipparbeit für Sie erledigt. Sie können dieses Konzept nach Ihren Wünschen erweitern. So können Sie beispielsweise ein Makro schreiben, das eine Liste aller Vertriebsmitarbeiter erzeugt, die für Ihr Unternehmen arbeiten.

Eine Arbeit automatisieren, die häufiger ausgeführt werden muss

Angenommen, Sie sind Verkaufsleiter und müssen jeden Monat einen Verkaufsbericht für Ihren Chef erstellen. Wenn diese Arbeit überschaubar ist, können Sie ein VBA-Programm entwickeln, das diese Arbeit für Sie erledigt. Ihr Chef wird von der gleichbleibend hohen Qualität Ihrer Berichte begeistert sein.

Sich wiederholende Operationen automatisieren

Wenn Sie immer dieselben Aktivitäten für zwölf verschiedene Excel-Arbeitsmappen durchführen müssen, können Sie ein Makro aufzeichnen, während Sie die erste Arbeitsmappe bearbeiten. Das Makro kann Ihre Aktivitäten dann in den anderen Arbeitsmappen wiederholen. Das Gute daran ist, dass sich Excel niemals darüber beschwert, dass eine Arbeit langweilig ist. Der Makrorekorder von Excel funktioniert so ähnlich wie ein Kassettenrekorder, mit dem man Musik aufnehmen kann. Allerdings benötigt ein Makrorekorder kein Mikrofon.

Einen benutzerdefinierten Befehl erstellen

Verwenden Sie oft dieselbe Abfolge von Excel-Befehlen? Wenn das so ist, können Sie einige Sekunden sparen, indem Sie ein Makro entwickeln, das diese Befehle in einem einzelnen benutzerdefinierten Befehl vereinigt, den Sie dann mit einem einzigen Tastendruck oder Mausklick ausführen können.

Eine benutzerdefinierte Schaltfläche erstellen

Sie können die Symbolleiste für den Schnellzugriff mit eigenen Schaltflächen versehen, die die Makros ausführen, die Sie schreiben. Office-Anwender sind in der Regel von solchen Dingen stark beeindruckt.

Neue Arbeitsblattfunktionen entwickeln

Obwohl Excel zahlreiche integrierte Funktionen besitzt (wie SUMME und MITTELWERT) können Sie *benutzerdefinierte* Arbeitsblattfunktionen erstellen, die Ihre Formeln stark vereinfachen werden. Ich garantiere Ihnen, dass Sie erstaunt sein werden, wie einfach das ist (ich zeige Ihnen in Kapitel 21, wie das geht). Aber es wird noch besser. Das Dialogfeld Funktion einfügen zeigt Ihre benutzerdefinierten Funktionen so an, als ob es sich um integrierte Excel-Funktionen handeln würde. Ist das nicht klasse?

Vollständige, makrobasierte Anwendungen schreiben

Wenn Sie bereit sind, mehr Zeit zu investieren, können Sie mit VBA sogar ganze Anwendungen mit einer angepassten Multifunktionsleiste, Dialogfeldern, Hilfesystem und vielen weiteren Ausstattungsmerkmalen erstellen. Dieses Buch geht nicht ganz so weit, aber ich erzähle Ihnen das trotzdem, um Sie damit zu beeindrucken, wie mächtig VBA in Wirklichkeit ist.

Benutzerdefinierte Add-Ins für Excel erstellen

Sie sind möglicherweise mit einigen der mit Excel mitgelieferten Add-Ins vertraut. So sind beispielsweise die Analyse-Funktionen ein sehr beliebtes Add-In. Sie können mit VBA eigene Add-Ins erzeugen. Ich habe mein Power Utility Pack-Add-In nur unter Zuhilfenahme von VBA entwickelt und überall auf der Welt wird es verwendet.

Vor- und Nachteile von VBA

In diesem Abschnitt beschreibe ich die guten Dinge, die VBA ausmachen, und ich gehe auch auf die Nachteile ein.

Vorteile von VBA

Sie können fast alles, was Sie in Excel machen können, automatisieren. Dazu schreiben Sie Befehle, die Excel ausführt. Die Automatisierung einer Aufgabe mit VBA hat verschiedene Vorteile:

- ✔ Excel führt die Aufgabe immer genau gleich aus (in den meisten Fällen ist Konsistenz eine gute Sache).
- ✔ Excel führt eine Arbeit viel schneller aus, als Sie das per Hand könnten (außer Sie sind Clark Kent).
- ✔ Wenn Sie ein guter Makroprogrammierer sind, führt Excel die Arbeit fehlerfrei aus.
- ✔ Wenn Sie alles richtig eingerichtet haben, kann jemand, der nichts über Excel weiß, die Arbeit erledigen.

- ✔ Sie können Dinge mit Excel tun, die ansonsten unmöglich wären – das kann Sie zu einer sehr beliebten Person in Ihrer Firma machen.
- ✔ Bei langen, zeitintensiven Aufgaben müssen Sie nicht gelangweilt vor Ihrem Computer sitzen bleiben. Excel erledigt die Arbeit, während Sie sich in der Teeküche aufhalten können.

Nachteile von VBA

Ich verheimliche Ihnen nicht die Nachteile (oder die möglichen Nachteile) von VBA:

- ✔ Sie müssen herausfinden, wie man Programme in VBA schreibt (aber das ist ja der Grund, warum Sie dieses Buch gekauft haben, oder?). Glücklicherweise ist das nicht so schwierig, wie Sie vielleicht denken.
- ✔ Andere Personen, die Ihre VBA-Programme verwenden sollen, müssen ein eigenes Exemplar von Excel besitzen. Es wäre nett, wenn Sie einfach auf einen Knopf drücken könnten und sich Ihre Excel-VBA-Anwendung in ein richtiges Programm verwandelt würde. Aber das ist nicht möglich (und wird möglicherweise auch nie möglich sein).
- ✔ Manchmal geht etwas schief. Mit anderen Worten: Sie können nicht blind darauf vertrauen, dass Ihr VBA-Programm unter allen Umständen richtig arbeiten wird. Willkommen in der Welt der Fehlersuche und – wenn andere Ihr Programm verwenden – in der Welt des technischen Supports.
- ✔ VBA verändert sich ständig. Wie Sie sicherlich wissen, entwickelt Microsoft Excel ständig weiter. Auch wenn Microsoft sehr viel Mühe darauf verwendet, die Kompatibilität zwischen den Excel-Versionen zu erhalten, kann es sein, dass Sie herausfinden, dass Ihr Programm, das Sie mit Excel 2007 entwickelt haben, nicht vernünftig in älteren oder zukünftigen Versionen von Excel funktioniert.

VBA in wenigen Worten

Nur um Ihnen kurz zu zeigen, was Sie zu erwarten haben, habe ich hier eine Zusammenfassung erstellt, worum es bei VBA geht. Natürlich beschreibe ich das Ganze ausführlicher im Verlauf dieses Buches.

- ✔ **Sie führen Aufgaben in VBA aus, indem Sie Code in ein VBA-Modul schreiben (oder aufzeichnen).** Sie können sich VBA-Module ansehen und bearbeiten, indem Sie den Visual Basic-Editor verwenden.
- ✔ **Ein VBA-Modul besteht aus Unterprozeduren.** Eine Unterprozedur hat nichts mit einer geringerwertigen Prozedur oder einer Prozedur, die irgendwo darunter liegt, zu tun. Stattdessen handelt es sich um Computercode, der Aktionen mit Objekten (hierzu weiter unten mehr) durchführt. Das folgende Beispiel zeigt eine einfache Unterprozedur, die `AddierSie` heißt. Dieses spektakuläre Programm zeigt das Ergebnis der Addition von 1 und 1 an.

```
Sub AddierSie()
   Summe = 1 + 1
   MsgBox "Die Antwort ist " & Summe
End Sub
```

- **Ein VBA-Programm kann auch Funktionen besitzen.** Eine Funktion liefert einen einzelnen Wert zurück. Sie können sie von einer anderen VBA-Prozedur aufrufen oder sie sogar als Funktion in einer Formel im Arbeitsblatt verwenden. Ein Beispiel für eine Funktion sehen Sie im Anschluss (die Funktion heißt `ZweiAddieren`). Diese Funktion nimmt zwei Zahlen entgegen (die auch als Argumente bezeichnet werden) und liefert die Summe der beiden Zahlen zurück.

  ```
  Function ZweiAddieren (arg1, arg2)
     ZweiAddieren = arg1 + arg2
  End Function
  ```

- **VBA verändert Objekte.** Excel bietet Ihnen Dutzende von Objekten, mit denen Sie arbeiten können. Zu diesen Objekten zählen die Arbeitsmappe, das Arbeitsblatt, ein Zellbereich, ein Diagramm und ein Formular. Es gibt natürlich noch viele weitere Objekte, die Sie alle mithilfe von VBA-Code bearbeiten können.

- **Objekte sind in Hierarchien angeordnet.** Objekte können als Behälter für andere Objekte dienen. An der Spitze der Objekthierarchie steht Excel. Excel ist wiederum ein Objekt, das `Application` heißt. Das `Application`-Objekt enthält andere Objekte, zum Beispiel das Objekt `Workbook` oder Add-In-Objekte. Das `Workbook`-Objekt kann wiederum andere Objekte, beispielsweise `Worksheet`- und `Chart`-Objekte enthalten. Ein `Worksheet`-Objekt kann Objekte wie `Range`-Objekte und `PivotTable`-Objekte enthalten. Der Begriff *Objektmodell* bezieht sich auf die Anordnung dieser Objekte. (Zum Objektmodell können Sie mehr in Kapitel 4 erfahren.)

- **Objekte derselben Art bilden eine Auflistung.** So gibt es beispielsweise die Auflistung `Worksheets`, die alle Arbeitsblätter innerhalb einer bestimmten Arbeitsmappe enthält. Die Auflistung `Charts` enthält alle `Chart`-Objekte in einer Arbeitsmappe. Auflistungen sind ebenfalls Objekte.

- **Sie beziehen sich auf ein Objekt, indem Sie seine Position in der Objekthierarchie durch einen Punkt als Trennzeichen angeben.** So können Sie beispielsweise auf die Arbeitsmappe Mappe1.xlsx über

  ```
  Application.Workbooks("Mappe1.xlsx")
  ```

  zugreifen. Damit greifen Sie auf die Arbeitsmappe Mappe1.xlsx in der Auflistung `Workbooks` zu. Die Auflistung `Workbooks` befindet sich innerhalb des `Application`-Objekts (das ist Excel). Wenn Sie das Ganze weitertreiben, können Sie auf Tabelle1 in Mappe1.xlsx zugreifen, indem Sie

  ```
  Application.Workbooks("Mappe1.xlsx").Worksheets("Tabelle1")
  ```

  schreiben.

Das Ganze können Sie sogar, wie im nächsten Beispiel zu sehen ist, noch weitertreiben, indem Sie auf eine bestimmte Zelle (in diesem Fall Zelle A1) zugreifen:

```
Application.Workbooks("Mappe1.xlsx").Worksheets("Tabelle1").Range("A1")
```

- ✔ **Wenn Sie bestimmte Referenzen auslassen, verwendet Excel das aktuelle Objekt.** Wenn Mappe1.xlsx die aktuelle Arbeitsmappe ist, können Sie den obigen Bezug abkürzen, indem Sie

  ```
  Worksheets("Tabelle1").Range("A1")
  ```

  schreiben. Wenn Sie wissen, dass Tabelle1 das aktuelle Arbeitsblatt ist, können Sie die Referenz weiter vereinfachen:

  ```
  Range("A1")
  ```

- ✔ **Objekte besitzen Eigenschaften**. Sie können sich unter einer Eigenschaft so etwas wie eine Einstellung des Objekts vorstellen. So besitzt beispielsweise das Objekt `Range` Eigenschaften wie `Value` (Wert) und `Address` (Adresse). Ein `Chart`-Objekt besitzt Eigenschaften wie `HasTitle` (besitzt Titel) und `Type` (Typ). Mit VBA können Sie die Eigenschaften von Objekten ermitteln und sogar verändern.

- ✔ **Sie beziehen sich auf die Eigenschaft eines Objekts, indem Sie den Objektnamen mit dem Eigenschaftennamen verknüpfen**. **Der Objektname wird von der Eigenschaft durch einen Punkt getrennt.** Sie können sich beispielsweise auf die Eigenschaft `Value` der Zelle A1 in Tabelle1 wie folgt beziehen:

  ```
  Worksheets("Tabelle1").Range("A1").Value
  ```

- ✔ **Sie können Variablen Werte zuweisen.** Eine Variable ist ein benanntes Element, das Informationen speichern kann. Mithilfe von Variablen in Ihrem VBA-Code speichern Sie Dinge wie Werte, Text oder Eigenschaften. Mit dem folgenden VBA-Befehl weisen Sie den Wert, der in Zelle A1 des Arbeitsblatts Tabelle1 steht, der Variablen `interessant` zu.

  ```
  interessant = Worksheets("Tabelle1").Range("A1").Value
  ```

- ✔ **Objekte besitzen Methoden.** Eine *Methode* ist eine Aktion, die Excel mit einem Objekt durchführen kann. So gibt es beispielsweise für das Objekt `Range` die Methode `ClearContents`. Diese Methode löscht den Inhalt des Bereichs, der durch das `Range`-Objekt bezeichnet wird.

- ✔ **Sie beziehen sich auf eine Methode eines Objekts, indem Sie den Objektnamen mit dem Methodennamen verknüpfen**. **Der Objektname wird von der Methode durch einen Punkt getrennt.** So löscht beispielsweise der folgende Befehl den Inhalt der Zelle A1:

  ```
  Worksheets("Tabelle1").Range("A1").ClearContents
  ```

- ✔ **VBA enthält alle Konstrukte einer modernen Programmiersprache, zum Beispiel Felder und Schleifen.** In anderen Worten: Wenn Sie sich Zeit nehmen, um sich mit den Grundlagen zu beschäftigen, können Sie Code schreiben, der ganz unglaubliche Dinge tut.

Die vorhergehende Liste beschreibt VBA in aller Kürze. Nun müssen Sie nur noch etwas über die Details herausfinden. Das ist der Sinn und Zweck dieses Buches.

Ein Ausflug in die Vergangenheit

Wenn Sie planen, VBA-Makros zu entwickeln, sollten Sie ein gewisses Verständnis für die Geschichte von Excel haben. Ich weiß, dass Sie keine Geschichtsstunde erwartet haben, als Sie dieses Buch zur Hand genommen haben, aber vertrauen Sie mir. Das hier ist wichtig.

Es folgt eine Liste aller wichtigen Excel-Versionen, die das Licht dieser Welt erblickt haben. In dieser Liste finden Sie auch ein paar Worte zum Thema Makrounterstützung der jeweiligen Version.

- **Excel 2:** Die erste Version von Excel für Windows wurde Version 2 getauft (statt 1), damit sie mit der Versionsnummer der Excel-Version auf dem Macintosh übereinstimmte. Excel 2 wurde 1987 veröffentlicht und mittlerweile wird es von niemandem mehr verwendet. Daher können Sie mehr oder weniger vergessen, dass es diese Version überhaupt gab.
- **Excel 3:** Excel 3 wurde im Herbst 1990 veröffentlicht und beherrschte die XLM-Makrosprache. Diese Version wird mittlerweile auch nicht mehr verwendet.
- **Excel 4:** Diese Version erblickte im Frühjahr 1992 das Tageslicht. Auch sie verwendete die XLM-Makrosprache. Sehr wenige Menschen verwenden diese Version noch immer; sie halten sich an den Leitsatz »Wenn etwas nicht kaputt ist, muss es auch nicht repariert werden«.
- **Excel 5:** Diese Version wurde im Frühjahr 1994 veröffentlicht. Das war auch die erste Version, die VBA verwendete (sie unterstützte aber immer noch XLM). Excel 5-Anwender gehören einer aussterbenden Spezies an.
- **Excel 95:** Technisch ist Excel 95 auch als Excel 7 bekannt (es gab keine Version 6). Diese Version wurde im Sommer 1995 ausgeliefert. Excel 95 ist eine 32-Bit-Anwendung und benötigte Windows 95 oder Windows NT. Im Gegensatz zu Excel 5 gab es ein paar Erweiterungen bei VBA. XLM wurde immer noch unterstützt. Ab und zu trifft man noch jemanden, der Excel 95 verwendet.
- **Excel 97:** Diese Version (auch als Excel 8 bekannt) wurde im Januar 1997 veröffentlicht. Sie besitzt *viele* Erweiterungen und eine vollständig neue Oberfläche für die VBA-Makroprogrammierung. Excel 97 verwendet auch ein neues Dateiformat (das ältere Excel-Versionen nicht öffnen können). Eine überschaubare Zahl von Anwendern verwendet noch Excel 97.
- **Excel 2000:** Ab dieser Version ist die Versionsnummer auf einmal vierstellig. Excel 2000 (auch als Excel 9 bekannt) wurde im Juni 1999 der Öffentlichkeit vorgestellt. Es enthielt aus Programmierersicht nur ein paar Erweiterungen, die meisten Erweiterungen betrafen die Anwender – besonders Anwender mit Onlinezugang. Mit Excel 2000 wurde die digitale Signatur eingeführt, um so den Anwendern zu garantieren, dass der Code wirklich von Ihnen stammt. Excel 2000 wird immer noch von einer mittelgroßen Anzahl von Anwendern eingesetzt.
- **Excel 2002:** Excel 2002 (auch als Excel 10 oder Excel XP bekannt) wurde Ende 2001 veröffentlicht. Eine der Hauptneuerungen ist, dass Excel 2002 dazu in der Lage ist, Ihre Arbeit

wiederherzustellen, wenn das Programm abgestürzt ist. Dies ist auch die erste Version, die einen Kopierschutz (eine Produktaktivierung) besitzt.

- ✔ **Excel 2003:** Von allen neuen Excel-Versionen, die ich je gesehen habe (und ich habe sie alle gesehen), ist dies die Excel-Version mit den wenigsten neuen Funktionen. Die meisten Hardcore-Anwender (was Sie natürlich einschließt) waren von dieser Version sehr enttäuscht. Trotzdem haben viele Anwender Excel 2003 gekauft. Ich denke, das waren all die, die von Excel-Versionen vor 2002 aktualisiert haben.
- ✔ **Excel 2007:** Die neueste und ohne Zweifel auch die beste Excel-Version. Microsoft hat sich mit dieser Version selbst übertroffen. Excel 2007 besitzt ein neues Aussehen, eine neue Benutzeroberfläche und unterstützt nun mehr als eine Million Zeilen. Dieses Buch wurde für Excel 2007 geschrieben, das heißt, Sie lesen gerade das falsche Buch, wenn Sie nicht Excel 2007 besitzen.

Nun, was ist denn jetzt die Moral dieser kurzen Geschichtsstunde? Wenn Sie planen, Ihre Excel-/VBA-Dateien an andere Anwender weiterzugeben, ist es sehr wichtig, dass Sie wissen, welche Excel-Version diese verwenden. Benutzer einer älteren Version können nicht von den Funktionen profitieren, die in neueren Versionen hinzugekommen sind. Wenn Sie beispielsweise Code schreiben, der sich auf die Zelle `XFD1048576` bezieht (die letzte Zelle in einem Arbeitsblatt), bekommen all diejenigen, die eine ältere Version von Excel verwenden, eine Fehlermeldung angezeigt, da Excel-Versionen vor Excel 2007 nur Arbeitsblätter mit 65.536 Zeilen und 255 Spalten unterstützen (hier ist die letzte Zelle `IV65536`). Excel 2007 besitzt auch einige neue Objekte, Methoden und Eigenschaften. Wenn Sie diese in Ihrem Code verwenden, erhalten Anwender älterer Excel-Versionen eine Fehlermeldung, wenn sie Ihr Makro ausführen – und Sie sind schuld.