

2 Neuerungen in Excel

Dieses Kapitel vermittelt einen Überblick über die wichtigsten Neuerungen und Änderungen der Excel-Versionen 2003, 2002, 2000, 97, 7 und 5. Um es gleich vorwegzunehmen: Die Unterschiede zwischen Excel 2003, 2002, Excel 2000 und Excel 97 sind gering und es gibt nur relativ wenig Kompatibilitätsprobleme.

Wenn Sie aber einen Versionswechsel von Excel 5 oder 7 auf eine neuere Excel-Version planen, werden Sie überwältigt sein von den zahllosen neuen Bibliotheken und Objekten und von den Umstellungsproblemen.

Kapitelübersicht

2.1	Neu in Excel 2003	62
2.2	Neu in Excel 2002	63
2.3	Neu in Excel 2000	66
2.4	Neu in Excel 97	70
2.5	Neu in Excel 7	74
2.6	Probleme und Inkompatibilitäten	74

2.1 Neu in Excel 2003

Für VBA-Programmierer ändert sich in Excel 2003 wenig. An den VBA-Grundfunktionen gibt es überhaupt keine sichtbaren Neuerungen, und in der Excel-Bibliothek gibt es nur wenige neue Objekte, deren Anwendung sich auf zwei Anwendungsaspekte beschränkt:

- **Listenfunktionen:** Zellbereiche innerhalb eines Tabellenblatts können in eine Liste umgewandelt werden. An den Daten ändert sich dadurch nichts, der Zellbereich wird nun aber blau umrandet und es stehen eine Reihe neuer Funktionen zur Verfügung, die die Bearbeitung der Liste im Vergleich zu früheren Excel-Versionen vereinfachen. Diese Funktionen können auch per VBA-Code gesteuert werden.
- **XML-Funktionen:** Excel 2003 bietet wesentlich mehr und ausgefeiltere Möglichkeiten zum Import, zur Bearbeitung und zum Export von XML-Daten. Diese Funktionen werden in Kapitel 14 ausführlich vorgestellt.

ACHTUNG

Die neuen XML-Funktionen stehen nur zur Verfügung, wenn Sie Excel 2003 als Einzelprogramm gekauft haben bzw. wenn Sie mit Office 2003 Professional arbeiten. In den Excel-Versionen von Office 2003 Small Business Edition und Office 2003 Standard Edition fehlen die XML-Funktionen! (Mit anderen Worten: Nur die teuerste Variante des Office-2003-Pakets enthält eine Excel-Version mit den neuen XML-Funktionen.)

Update-Empfehlung: Vielleicht fragen Sie sich, ob ein Update auf Excel 2003 überhaupt lohnt. Die Antwort hängt davon ab, ob man Excel 2003 isoliert oder als Teil des Office-Pakets betrachtet. In Excel 2003 selbst sind die Veränderungen wie gesagt eher bescheiden; ein Update ist nur sinnvoll, wenn Excel zur Bearbeitung externer XML-Daten eingesetzt werden soll.

Anders sieht es aus, wenn man Office als Ganzes betrachtet: In einigen Office-Komponenten gibt es durchaus fundamentale Verbesserungen und Erweiterungen. Die wichtigsten Punkte sind im Folgenden aufgezählt. (Unzählige weitere Argumente und Werbeversprechungen finden Sie auf der Office-Website von Microsoft.)

- Das E-Mail- und Kommunikationsprogramm Outlook wurde stark verbessert und enthält endlich Schutzmechanismen gegen Spam (also gegen die Zusendung unerwünschter E-Mails).
- Es gibt neue Funktionen zur gemeinsamen Bearbeitung von Office-Dokumenten. (Viele der neuen Funktionen erfordern allerdings, dass ein so genannter Share-Point-Server – ein neues Microsoft-Programm – im lokalen Netzwerk zur Verfügung steht.)
- Die neue Office-Komponente InfoPath erleichtert die Verarbeitung von Formularen.

- Die neue Office-Komponente OneNote ermöglicht ein problemloses Verfassen von Notizen und Ideen.
- Die Integration von XML-Funktionen ist ausgereifter und vielseitiger als in Office 2002.

Insgesamt drängt sich der Eindruck auf, dass Office vor allem im Hinblick auf vernetzte Firmenanwendungen optimiert wurde, während sich für Privatanwender nur wenig ändert. Auch die Tatsache, dass Office 2003 als Betriebssystem Windows 2000/XP voraussetzt, belegt diese Vermutung. (Windows 9x/ME werden also explizit nicht mehr unterstützt!)

Office 2003 und .NET

Hinter dem Kürzel .NET verbirgt sich eine neue Entwicklungsplattform, die sich unter anderem aus neuen Programmiersprachen (Visual Basic .NET und C#) und Bibliotheken zusammensetzt. Office 2003 basiert intern nicht auf .NET und kann daher auch keine .NET-Funktionen nutzen. Die einzigen Ausnahmen stellen so genannte **Web Services** dar, die mit dem in Abschnitt 15.4 beschriebenen, kostenlos zur Verfügung stehenden *Web Services Toolkit* auch unter Excel genutzt werden können.

Die Excel-Makroprogrammierung erfolgt also weiterhin mit VBA (worüber sicher viele Programmierer erleichtert sind). Für alle, die sich schon mit .NET angefreundet haben, gibt es aber immerhin die *Visual Studio Tools for Office* (VSTO). In Kombination mit Visual Studio .NET 2003 können Sie damit Word- und Excel-Programme entwickeln, wobei als Programmiersprachen und deren VB.NET oder C# zum Einsatz kommen. (Andere Office-Komponenten werden allerdings noch nicht unterstützt.) Die *Visual Studio Tools for Office* sind kein Bestandteil von Office 2003, sondern werden getrennt zum Verkauf angeboten. Weitere Informationen finden Sie hier:

<http://msdn.microsoft.com/vstudio/office/officetools.aspx>

HINWIS

Beachten Sie, dass die Office-Programmiersprache VBA zwar weitgehend kompatibel zur Windows-Programmiersprache Visual Basic 6 ist, aber vollkommen inkompatibel zur neuen .NET-Sprache Visual Basic .NET!

2.2 Neu in Excel 2002

Im Vergleich zu Excel 2000 bietet Excel 2002 zwar rund 35 neue Objekte, dabei handelt es sich aber primär um kosmetische Veränderungen. Bei den wenigen wirklichen Neuerungen (z.B. den so genannten Smart Tags) ist eher zweifelhaft, ob sich diese in der Praxis wirklich durchsetzen werden. Zudem ist es problematisch, die neuen Objekte in VBA-Code einzusetzen, weil derartiger Code damit nicht mehr kompatibel zu Excel 2000 ist.

Ärgerlich ist, dass Excel 2002 weder stabiler noch ausgereifter wirkt als Excel 2000. Fehler aus Excel 2000 finden sich auch in Excel 2002 wieder, Abstürze gibt es nach wie vor. Dafür ist die Zwangsaktivierung bei Hardware-Umbauten oder beim Kauf eines neuen Rechners ärgerlich. Ich habe bei der Arbeit an diesem Buch kein einziges Argument gefunden, das für einen Umstieg von Excel 2000 auf 2002 sprechen würde (sorry, Microsoft).

VBA-Sprachmerkmale

VBA (Visual Basic für Applikationen) stellt die Basis aller Programmiermöglichkeiten innerhalb von Office dar. Mit Office 2002 wird die VBA-Version 6.3 mitgeliefert. Gegenüber VBA 6.0 aus Office 2000 gibt es keine relevanten Änderungen.

Neu in Excel 2002 ist die Möglichkeit, Excel bzw. das ganze Office-Paket explizit ohne VBA-Unterstützung zu installieren. Diese Option bietet einen perfekten Schutz vor VBA-Viren (siehe Abschnitt 4.7), macht aber natürlich jede Anwendung von VBA-Makros unmöglich. Viele Assistenten und Add-Ins können nicht mehr verwendet werden und Access kann überhaupt nicht mehr gestartet werden.

Selbst für sehr sicherheitsbewusste Leser dieses Buchs kommt diese Option daher nicht in Frage. Sie sollten sich als Entwickler aber bewusst sein, dass es diese Möglichkeit gibt und dass von Ihnen entwickelte Makros auf einer anderen Excel-Installation möglicherweise nicht ausgeführt werden können, weil dort VBA gar nicht installiert ist.

VERWEIS

Weitere Informationen zur Deaktivierung bzw. Deinstallation der VBA-Funktionen finden Sie in den Knowledge-Base-Artikeln Q281954 und Q281953, die zuletzt hier zu finden waren:

[http://support.microsoft.com/default.aspx?scid=kb;\[LN\];Q281954](http://support.microsoft.com/default.aspx?scid=kb;[LN];Q281954)

[http://support.microsoft.com/default.aspx?scid=kb;\[LN\];Q281953](http://support.microsoft.com/default.aspx?scid=kb;[LN];Q281953)

Neue bzw. geänderte Objekte, Eigenschaften und Methoden

Automatische Sicherheitskopien: Excel 2002 kann nun endlich, was Word schon immer konnte: nämlich regelmäßig (z.B. alle zehn Minuten) eine Sicherheitskopie der aktuellen Dateien erstellen. Per VBA-Code können Sie diese Funktion über das neue *AutoRecover*-Objekt steuern. Vielleicht kann man das als Eingeständnis Microsofts werten, dass Excel doch nicht immer ganz so stabil läuft, wie es eigentlich sollte ...

Blattschutz: In Excel 2002 ist es möglich, Zellen so zu schützen, dass zwar der Inhalt unveränderlich ist, die Formatierung, Sortierung und andere Gestaltungsdetails aber vom Anwender gesteuert werden können. Zur Steuerung der neuen Schutzoptionen wurde die *Protect*-Methode des *Worksheet*-Objekts erweitert. Außerdem gibt das neue *Protection*-Objekt Auskunft über die aktuellen Schutzoptionen.

Neu ist auch die Möglichkeit, einzelnen Benutzern (mit oder ohne Passwort) Zugriff auf ausgewählte Zellbereiche innerhalb eines geschützten Tabellenblatts zu geben. Das ist dann praktisch, wenn mehrere Benutzer auf dieselbe Excel-Datei zugreifen dürfen, aber nicht jeder alles verändern darf. Zur Verwaltung dieser Einstellungen dienen die neuen Objekte *AllowEditRange[s]* und *UserAccess[List]*.

Dateiauswahldialog: Die *Office*-Bibliothek enthält ein neues *FileDialog*-Objekt zur Durchführung einer Datei- oder Verzeichnisauswahl. Das Objekt kann statt den Methoden *GetOpen*- bzw. *GetSaveAsFilename* eingesetzt werden und steht allen Office-Komponenten (nicht nur Excel) zur Verfügung.

Fehlerüberprüfung: EXTRAS|FEHLERÜBERPRÜFUNG durchsucht das aktive Tabellenblatt nach möglichen Fehlern, z.B. als Text gespeicherte Zahlen, Daten mit zweistelliger Jahreszahl, Formeln, die auf leere Zellen verweisen etc. Ob diese Fehlerüberprüfung automatisch auch im Hintergrund erfolgen soll und welche Fehlerursachen überprüft werden, kann durch das *ErrorCheckingOptions*-Objekt gesteuert werden. Gefundene Fehler können über die *Errors*-Aufzählung ausgewertet werden.

Formatsuche: Bei den Methoden *Find* und *Replace* zum Suchen bzw. Ersetzen von Zellinhalten kann nun auch die Formatierung der Zelle berücksichtigt bzw. verändert werden. Die Formateinstellung erfolgt über zwei neue *CellFormat*-Objekte, die über die Eigenschaften *FindFormat* und *ReplaceFormat* angesprochen werden.

Organigramme: Mit EINFÜGEN|SCHEMATISCHE DARSTELLUNG können einfache Organigramme in das Excel-Tabellenblatt eingefügt und gestaltet werden. Per VBA-Code erfolgt der Zugriff auf diese neuen Objekte durch diverse *DiagramXxx*-Objekte. Getrübt wird die Freude über diese Funktion allerdings dadurch, dass die neuen Objekte schlecht durchdacht sind und noch voller Fehler stecken.

Pivottabellen: Beim Herstellen bzw. Schließen einer Datenbankverbindung zu *PivotTable*-Objekten treten *PivotTableOpen*- und *-Close*-Ereignisse auf. Die Ereignisse stehen unter verschiedenen Namen für die Klassen *Application*, *Workbook* und *Worksheet* zur Verfügung.

Smart Tags: Smart Tags sind kleine, kontextabhängige Menüs, mit denen der Inhalt einer Zelle in besonderer Weise bearbeitet werden kann. Wenn eine Zelle beispielsweise ein Aktienkürzel enthält, können Sie über das Smart-Tag-Menü direkt eine Webseite öffnen, die den aktuellen Kurs enthält. Per VBA können Sie zwar keine neuen Smart Tags erzeugen, Sie können aber immerhin mit sieben neuen *SmartTagXxx*-Objekten auf vorhandene Smart Tags zugreifen.

Sprachausgabe: Nur bei der englischen Version von Excel 2002 können Sie über das neue *Speech*-Objekt die automatische Sprachausgabe steuern bzw. selbst Texte über die Soundkarte ausgeben.

Überwachungsobjekte: Mit dem *Watch*-Objekt und der dazugehörigen *Watches*-Aufzählung können einzelne Zellen aus unterschiedlichen Tabellenblättern in einem Überwachungsfenster angezeigt werden (EXTRAS|FORMELÜBERWACHUNG|ÜBERWACHUNGSFENSTER).

Alle Excel-Objekte sind in der Objektreferenz in Kapitel 16 dokumentiert. Objekte, die in Excel 2002 neu dazugekommen sind, sind dort speziell markiert.

Sonstiges

XML: Microsoft bewirbt Excel 2002 als XML-kompatibel. (XML steht für *Extensible Markup Language* und ist ein Textformat zur Darstellung beliebiger hierarchischer Daten.) Allerdings besteht die einzige XML-Funktion darin, dass Sie Excel-Arbeitsmapen in einem speziellen XML-Format speichern können. Die praktische Bedeutung dieses Formats ist zurzeit gleich Null:

- Es gibt noch keine anderen Programme, die damit umgehen können.
- In XML-Dateien können weder VBA-Code noch eingebettete Objekte oder Diagramme gespeichert werden.
- Die resultierenden *.xml-Dateien sind um ein Vielfaches größer als *.xls-Dateien.

Weitere XML-Funktionen – etwa zur Verarbeitung allgemeiner XML-Dateien – suchen Sie in Excel 2002 leider vergeblich. Die gibt es erst in Excel 2003 (siehe den vorigen Abschnitt).

2.3 Neu in Excel 2000

VBA-Sprachmerkmale

VBA 6.0 (Office 2000) unterscheidet sich von VBA 5.0 (Office 97) durch einige Zusatzfunktionen.

- Es gibt einige neue Funktionen zur Bearbeitung und Formatierung von Zeichenketten: *Join*, *InstrRev*, *Replace*, *Split*, *MonthName*, *WeekdayName*, *FormatCurrency*, *FormatDateTime*, *FormatPercent* und *FormatNumber*.
- Einige kleinere Fortschritte gibt es bei der objektorientierten Programmierung. Auf echte Vererbung (eigentlich *das* entscheidende Merkmal objektorientierter Programmiersprachen) müssen Sie zwar weiterhin verzichten, aber immerhin ermöglicht das Schlüsselwort *Implements* nun eine – wenn auch halbherzige – Weiterverwendung vorhandener Klassen.
- Sie können nun auch eigene Klassen mit Ereignissen ausstatten und diese Ereignisse selbst auslösen (Schlüsselwörter *Event* und *RaiseEvent*).
- Mit *CallByName* können Sie Eigenschaften und Methoden ausführen, wobei Sie den Namen als Zeichenkette übergeben. In manchen Fällen bietet das mehr Flexibilität.

Neue bzw. geänderte Objekte, Eigenschaften und Methoden

Diagramme: Die Beschriftung von Koordinatenachsen kann nun skaliert werden. Statt also Zahlen wie 21.000.000, 22.500.000 etc. anzuzeigen, kann als Skalierungsfaktor 'Millionen' angegeben werden; daraus resultieren dann die Zahlenwerte 21 und 22,5. Für die VBA-Programmierung lässt sich dieses Merkmal über einige neue *DisplayUnit-xxx*-Eigenschaften steuern.

Neu sind auch so genannte Pivotdiagramme: Dabei handelt es sich eigentlich um ganz normale Diagramme (*Chart*-Objekte), deren Inhalt aber durch einige Pivotlistenfelder dynamisch verändert werden kann (wie bei einer Pivottable). Die VBA-Steuerung erfolgt über das neue *PivotLayout*-Objekt, das über die gleichnamige Eigenschaft des *Chart*-Objekts angesprochen wird.

Verzeichnisse: Die neue Eigenschaft *UserLibraryPath* des *Application*-Objekts liefert den Pfad zum persönlichen Verzeichnis mit Add-In-Dateien. Geändert hat sich die Wirkung von zwei anderen *Application*-Eigenschaften: *TemplatesPath* und *StartupPath* verweisen auf die persönlichen Vorlagen- bzw. Xlstart-Verzeichnisse. (In Excel 97 lieferten die beiden Eigenschaften dagegen den Pfad zu den globalen Vorlagen- bzw. Xlstart-Verzeichnissen.) Leider gibt es keine neuen Eigenschaften, um die globalen Vorlagen- oder Xlstart-Verzeichnisse zu ermitteln.

Datenbankanwendungen

Excel ist zwar kein Datenbankprogramm, es wird aber sehr häufig zur Analyse von extern gespeicherten Daten verwendet (etwa in Form von Diagrammen oder Pivottablen). Daher war der Zugriff auf externe Daten schon immer von großer Bedeutung für Excel-Programmierer.

Die einfachste Möglichkeit zur Extraktion von Daten aus einer Datenbank ist zumeist das Programm MS Query, das über DATEN|EXTERNE DATEN|NEUE ABFRAGE gestartet wird. Die in MS Query durchgeführten Einstellungen können per VBA-Code über das *QueryTable*-Objekt verwaltet werden. Dieses Objekt wurde in Excel 2000 mit einer Fülle neuer Eigenschaften ausgestattet.

Erheblich mehr Flexibilität und vor allem die Möglichkeit, Datenbanken auch zu ändern, bietet die neue ADO-Bibliothek (*ActiveX Data Objects*). Die bisher für diesen Zweck verwendete DAO-Bibliothek kann zwar weiterhin eingesetzt werden, wird von Microsoft aber nicht mehr weiterentwickelt. Insofern sollten vor allem neue Datenbankanwendungen die ADO-Bibliothek nutzen. (Es spricht aber selten etwas dagegen, vorhandene Anwendungen bei DAO zu belassen.) Eine Einführung in die ADO-Programmierung finden Sie in Kapitel 12.

Ein beliebtes Hilfsmittel bei der Analyse von Daten innerhalb von Excel – egal ob die Daten nun von herkömmlichen Datenbanksystemen oder aus einem *Data Warehouse* stammen – sind so genannte Pivottabellen. Diese Tabellen gibt es zwar schon seit geraumer Zeit, sie wurden aber in Excel 2000 mit einigen Erweiterungen ausgestattet (etwa 30 neue Eigenschaften und Methoden). Pivottabellen stehen im Mittelpunkt von Kapitel 13.

Internet

Obwohl Office 2000 von Microsoft gleichsam als das Internet-Office angepriesen wurde, gibt es aus der Sicht von VBA-Programmierer nur wenige wirklich relevante Neuerungen:

- **HTML-Export/Import:** Excel bietet neue Funktionen zum Export von Excel-Objekten (z.B. eines Tabellenbereichs) im HTML-Format sowie zum Import von Daten aus HTML-Dateien. Per Programmcode können diese Funktionen über die Objekte *PublishObject* (Export) und *WebQuery* (Import) gesteuert werden.
- **Webkomponenten:** Dabei handelt es sich um internettaugliche Steuerelemente, in denen eine Excel-Tabelle bzw. -Diagramm dargestellt werden kann. In den Steuerelementen stehen zwar nicht alle, aber doch recht viele Funktionen von Excel zur Verfügung. Der Vorteil besteht darin, dass die Daten im HTML-Dokument dynamisch bearbeitet werden können. Das ist aber auch mit Nachteilen verbunden, deren wesentlichster darin besteht, dass Webkomponenten nur von Personen benutzt werden dürfen, die eine Office-2000-Lizenz besitzen. Insofern sind die Webkomponenten für das Internet ungeeignet und eher für den Einsatz in Intranets gedacht (etwa in großen Firmen).
- **E-Mails versenden:** Mit der *Workbook*-Eigenschaft *EnvelopeVisible* können Sie am oberen Rand des Tabellenfensters einige Textfelder zur Eingabe der E-Mail-Adresse und des Betreffs einblenden. Sie erleichtern es damit dem Anwender, die Arbeitsmappe interaktiv zu versenden.

Sonstiges

Umgang mit Dateien (FSO-Bibliothek): Wenn Sie auf Dateien oder Verzeichnisse zugreifen möchten oder Textdateien lesen bzw. schreiben müssen, können Sie statt der bisher üblichen Kommandos (*Open*, *Close*, *Print* etc.) die neue FSO-Bibliothek verwenden (*File Scripting Objects*). Die darin definierten Objekte bieten nicht nur mehr Eleganz beim Dateizugriff, sondern erstmals auch Unicode-Unterstützung. Bedauerlicherweise sind die FSO-Methoden nicht zum Umgang mit Binärdateien geeignet – dazu müssen Sie weiterhin die herkömmlichen Funktionen verwenden.

Dialoge (UserForm): Dialoge können nun auch ungebunden geöffnet werden (*Show vbModeless*), d.h., das unter dem Dialog sichtbare Excel kann weiterhin verwendet werden, ohne den Dialog dazu verlassen zu müssen. Verwenden Sie diese neue Funktion aber nicht in Kombination mit dem *RefEdit*-Steuerelement zur Eingabe von Zellbereichen – sonst verliert Excel die Kontrolle über den Tastaturfokus und kann nur noch gewaltsam beendet werden (Task-Manager bzw. Strg+Alt+Entf)!

Textimport: Eine Quelle beständigen Ärgers mit der bisherigen Excel-Version war der Versuch, den Import von ASCII-Dateien zu automatisieren. Dankenswerterweise sind hier Fortschritte zu verzeichnen. Endlich kann das Kommasymbol (, oder .) und das Tausendertrennzeichen explizit durch zusätzliche Parameter der Methode *OpenText* gesteuert werden. Zum Textimport kann auch das überarbeitete *QueryTable*-Objekt eingesetzt werden.

Konfigurationsdateien: Die Orte der Konfigurationsdateien haben sich – wie schon bei allen vorangegangenen Versionen – wieder geändert. Das allseits beliebte Spiel heißt: *Such mich!*

Hilfesystem: Das gesamte Hilfesystem wurde vollständig überarbeitet und basiert intern jetzt auf dem so genannten HTMLHelp-System. Inhaltlich sind die gebotenen Informationen zwar oft in Ordnung, das Problem besteht aber darin, die Informationen auch zu finden. F1 führt durchaus nicht immer zum Ziel, und die aus dem früheren Hilfesystem bekannte Volltextsuche ist aus nicht nachvollziehbaren Gründen verschwunden. Dafür gibt es jetzt einen so genannten Antwortassistenten, der aber keinen vollwertigen Ersatz darstellt. Seine Ergebnisse sind manchmal ganz gut, viel öfter aber schlicht unbrauchbar. (Gibt es denn noch immer nicht genug Assistenten, die ständig im Weg sind?)

Falls Sie eigene Excel-Anwendungen mit einer Hilfedatei ausstatten möchten, können Sie dafür jetzt ebenfalls HTMLHelp verwenden. Zur Entwicklung eigener Hilfedateien benötigen Sie den HTMLHelp-Workshop, den Sie von der Microsoft-Website (kostenlos) herunterladen können.

Eurounterstützung: Bei älteren Office-Versionen konnte das Eurosymbol nur nach der Installation spezieller Updates verwendet werden. Office 2000 ist in dieser Beziehung natürlich schon weiter – aber viel mehr zum Thema Euro hat sich Microsoft nicht einfallen lassen. Die als Excel-Add-In verfügbare Funktion *EuroConvert* ist nicht dokumentiert, andere Hilfsmittel zur Konvertierung vorhandener Tabellen von einer beliebigen europäischen Währungseinheit in Euro sucht man vergeblich. (Diesen Mangel versucht Abschnitt 5.10 so gut es geht zu beheben.)

2.4 Neu in Excel 97

Neu in dieser Version sind die Entwicklungsumgebung und die Konzeption benutzerdefinierter Formulare (MS-Forms-Bibliothek). Außerdem wurden beinahe die Hälfte der rund 120 Excel-7-Objekte durch neue Objekte ersetzt. Darüber hinaus wurden zahllose neue Objekte eingeführt.

Entwicklungsumgebung

Die offensichtlichste Neuerung in Excel 97 war die von Excel getrennte Entwicklungsumgebung. Diese Trennung ist zwar gewöhnungsbedürftig, bringt aber viele Vorteile mit sich. Der wohl größte Fortschritt ist die Erweiterung unvollständiger Schlüsselwörter durch automatisch angezeigte Auswahllisten bzw. durch Strg+Leertaste.

Modulblätter: Die Zerteilung in eine Anwendungs- und eine Programmierkomponente hat auch Auswirkungen auf die Programmierung: Die Aufzählung *Modules* und die Objektklasse *Module* für Modulblätter werden nicht mehr unterstützt und stehen nur noch aus Kompatibilitätsgründen zur Verfügung.

Schutz von Modulblättern: Wegen der Trennung von Excel in eine Anwendungs- und eine VBA-Komponente wurden auch die Schutzfunktionen für Module überarbeitet. Das wäre nicht weiter schlimm, wenn dabei ein Minimum an Kompatibilität gewahrt worden wäre. Das ist leider nicht der Fall: In Excel 7 ausgeblendete und geschützte Module werden in Excel 97 angezeigt, als wären sie ungeschützt. Na ja, Ihre Kunden bzw. Anwender wollten ja schon immer wissen, wie Sie all die Funktionen programmiert haben ...

VBA-Sprachkonzepte

Ereignisse: Die Verwaltung von Ereignissen wurde in Excel 97 vollständig überarbeitet. Während in Excel 5 und 7 eine kleine Zahl vordefinierter Ereignisse über *OnEvent*-Eigenschaften angemeldet wurden, werden Ereignisse jetzt wie in Visual Basic verwaltet: Zu allen möglichen Ereignissen – etwa dem Aktivieren eines Tabellenblatts – sind die Namen von Ereignisprozeduren fix vorgegeben (etwa *Worksheet_Activate*). Wenn Sie diese Prozedur mit Code füllen, wird der Code automatisch jedes Mal ausgeführt, wenn das Ereignis auftritt. Auch Ereignisse zu Objekten, die in der Entwicklungsumgebung nicht in eigenen Modulen angezeigt werden, können über den Umweg eines Klassenmoduls empfangen werden.

Die Schattenseite des neuen Ereigniskonzepts besteht darin, dass die Excel-Programmierer vielleicht ein wenig zu überschwänglich waren: Beinahe jedes Objekt ist mit zahllosen Ereignissen ausgestattet. Die Übersichtlichkeit ist dabei auf der Strecke geblieben. Abzuwarten bleibt, ob es wirklich Anwendungen für all diese Ereignisse gibt.

Klassenmodule: In VBA können nun wie in Visual Basic neue Objektklassen mit Methoden und Eigenschaften definiert werden (allerdings ohne eigene Ereignisse und ohne *Enum*-Konstanten). Die Implementierung macht allerdings einen halbfertigen Eindruck. Zudem stellt sich die Frage, ob für selbst definierte Klassenmodule innerhalb von Excel-Anwendungen ein großer Bedarf besteht.

Collection-Objekt: Das *Collection*-Objekt stellt eine komfortable Alternative zu Feldern dar. Der Vorteil gegenüber normalen Feldern besteht darin, dass *Collections* nicht im Voraus in einer bestimmten Größe deklariert werden müssen. Außerdem kann als Index ein beliebiger Text verwendet werden (statt einer fortlaufenden Nummer).

Benutzerdefinierte Funktionen: In Excel 5 und 7 konnten selbst definierte Tabellenfunktionen unterschiedlichen Kategorien zugeordnet werden, die im Dialog EINFÜGEN|FUNKTION (dem ehemaligen Funktionsassistenten) berücksichtigt wurden. Seit Excel 97 gibt es diese Möglichkeit offiziell nicht mehr, alle selbst definierten Funktionen werden einfach der Gruppe *benutzerdefiniert* zugeordnet. (Diese Einschränkung lässt sich umgehen – siehe Abschnitt 5.7.)

Veränderte oder erweiterte Objekte

Eine Menge Excel-Objekte wurden in Version 97 neu eingeführt oder geändert bzw. anderen Bibliotheken zugeordnet (und dabei auch neu benannt). Die wichtigsten Änderungen – soweit sie durch Excel 2000/2002 nicht schon wieder obsolet geworden sind – werden auf den folgenden Seiten zusammengefasst.

Die wichtigsten Erweiterungen beim *Workbook*-Objekt betreffen die gemeinsame Nutzung einer Excel-Datei durch mehrere Anwender (Freigabefunktionen). Um eine Excel-Datei gemeinsam nutzen zu können, muss sie mit *SaveAs* mit *AccessMode:=xl-Shared* als freigegebene Datei gespeichert werden. (Manuell erfolgt die Freigabe von Excel-Dateien übrigens nicht durch SPEICHERN UNTER, sondern durch EXTRAS|ARBEITSMAPPE FREIGEBEN.)

Um die Freigabe wieder aufzuheben, steht die Methode *ExclusiveAccess* zur Verfügung. (Dadurch wird die aktuelle Arbeitsmappe unter dem aktuellen Namen gespeichert.) Der aktuelle Zustand kann der Eigenschaft *MultiUserEditing* entnommen werden. Zur Protokollierung, Verwaltung und Synchronisation gemeinsam genutzter Dateien gibt es eine Menge neuer Eigenschaften und Methoden: *AcceptAllChanges*, *AutoUpdateFrequency*, *AutoUpdateSaveChanges*, *HighlightChangesOnScreen*, *HighlightChangesOptions*, *KeepChangesHistory*, *ListChangesOnNewSheet*, *PersonalViewListSettings*, *PersonalViewPrintSettings*, *ProtectSharing*, *RejectAllChanges* und *UnprotectSharing*.

Mit dem *FormatCondition*-Objekt kann die Formatierung einer Zelle bzw. eines Zellbereichs (*Range*-Objekt) von dessen Inhalt abhängig gemacht werden. Beispielsweise können Sie erreichen, dass sich die Farbe eines Zahlenwerts ändert, wenn dieser größer als der Inhalt einer Vergleichszelle ist. Pro Zelle können maximal drei Bedingungen angegeben werden. Manuell können Sie diese Art der Formatierung mit FORMAT|BEDINGTE FORMATIERUNG durchführen. (Ähnliche Effekte konnten in früheren

Excel-Versionen durch bedingte Zahlenformate erreicht werden. Allerdings gab es dabei viel weniger Gestaltungsmöglichkeiten und keine VBA-Schnittstelle.)

Mit dem *Validation*-Objekt können Validitätskontrollen für die Eingabe in Zellen definiert werden. Damit lassen sich beispielsweise Eingaben auf ein bestimmtes Format (Datum) oder auf einen bestimmten Wertebereich eingrenzen. Manuell können Sie solche Regeln mit DATEN | GÜLTIGKEIT formulieren.

Mit dem *Hyperlink*-Objekt können Verweise zu Excel-Blättern ebenso wie zu anderen Dateien (lokal oder im Internet) hergestellt werden.

Eine ganze Objektfamilie (*Shape*, *ShapeRange*, *ShapeNode*, *GroupShapes* etc.) ersetzt die Zeichnungsobjekte aus Excel 5/7 (*Arc*, *Line* etc.) Außer den Zeichnungsobjekten, die jetzt AutoForm-Objekte heißen und erheblich mehr Gestaltungsmöglichkeiten bieten, werden auch alle anderen Objekte in Tabellen durch *Shape*-Objekte verwaltet: Steuerelemente, OLE-Objekte, Objektgruppen etc.

Neue Eigenschaften und Methoden

Durch die Einstellung *AutoScaleFont=True* kann für diverse Objekte (*AxisTitle*, *LegendEntry* etc.) erreicht werden, dass die Schriftgröße an die Größe des Objekts angepasst wird.

Durch *FormulaLabel=xlColumn/RowLabel* kann der Inhalt einer Zelle als Name für die durch diese Zelle laufende Zeile oder Spalte definiert werden. In der Folge können diese Namen in Formeln verwendet werden. Formeln werden dadurch besser lesbar. Voraussetzung ist, dass *AcceptLabelsInFormulas* auf *True* gestellt ist (Defaulteinstellung). Manuell kann *FormulaLabel* mit EINFÜGEN | NAMEN | BESCHRIFTUNG verändert werden.

In Bild 2.1 gilt für die Zellen A1:C1 *FormulaLabel=xlColumnLabels*. Daher kann in C2 die Formel *=Einnahmen-Ausgaben* verwendet werden. Wenn in A1 ein anderer Text eingegeben wird, wird die Formel automatisch geändert!

	A	B	C
1	Einnahmen	Ausgaben	Gewinn
2	123	97	26
3			

Bild 2.1: Beispiel für *FormulaLabel*

VORSICHT

Das gerade beschriebene Merkmal gibt es zwar in Excel 2000/2002 noch immer, dort ist es aber per Default deaktiviert! Um es auch in Excel 2000/2002 zu nutzen, müssen Sie vorher in EXTRAS | OPTIONEN | BERECHNUNG die Option BESCHRIFTUNGEN IN FORMELN aktivieren.

Gleich eine ganze Reihe neuer Formatierungsmöglichkeiten ergibt sich durch neue Eigenschaften zum *Range*-Objekt: ***IndentLevel*** gibt an, wie weit der Inhalt einer Zelle eingerückt werden soll. (Der zulässige Wertebereich geht von 0 bis 15.) Über ***Borders*** lassen sich jetzt auch diagonale Linien durch eine Zelle zeichnen (bisher konnte die Zelle nur eingerahmt werden). Mit ***Orientation*** kann die Textausrichtung im Bereich zwischen -90 bis 90 Grad beliebig eingestellt werden (bisher waren nur Vielfache von 90 Grad möglich). Die ***Orientation***-Eigenschaft steht auch für zahlreiche andere Objekte zur Verfügung, etwa um die Beschriftung von Diagrammen zu verändern.

Objektbibliotheken

Menüs/Symbolleisten (Office-Bibliothek): Sicher ist Ihnen aufgefallen, dass sich das Aussehen von Menüs und Symbolleisten geändert hat. Die Auswirkung für VBA-Programmierer besteht darin, dass die *Toolbar*- und *Menu*-Objekte durch eine Familie neuer *CommandBar*-Objekte (Office-Bibliothek) ersetzt wurde.

Daraus ergeben sich nicht nur gravierende Änderungen für die Verwaltung eigener Menüs und Symbolleisten, sondern auch eine Reihe von Inkompatibilitäten. Es gibt keinen Menüeditor mehr, stattdessen müssen Sie sich beim Entwurf neuer Menüleisten durch eine Fülle unübersichtlicher Kontextmenüs quälen (Startpunkt ist ANSICHT | SYMBOLLEISTEN | ANPASSEN). Der einzige Vorteil als Gegenleistung für den Adaptions- und Umstellungsaufwand besteht darin, dass die neuen Menü- und Symbolleisten bzw. die zugrunde liegende Office-Objektbibliothek in allen Office-Komponenten gemeinsam genutzt werden kann.

Dialoge (MS-Forms-Bibliothek): Zur Gestaltung von Dialogen steht die neue MS-Forms-Bibliothek zur Auswahl. Damit erstellte Dialoge sehen genauso aus wie Dialoge in Excel 5 und 7, der Dialogeditor ist aber anders zu bedienen und die Verwaltung des Dialogs per Programmcode weist erhebliche Unterschiede auf. Es gibt zwei wesentliche Vorteile: Sie können nun mehrblättrige Dialoge erstellen (in der Art der Optionen-Dialoge) und externe Steuerelemente nutzen (so genannte ActiveX-Steuerelemente).

Im Gegensatz zu herkömmlichen Menüs und Symbolleisten werden Excel-5-/-7-Dialoge auch unter Excel 2000 noch unterstützt, d.h., auch der Dialogeditor wird noch mitgeliefert. Allerdings treten bei der Ausführung von VBA-Code zur Verwaltung alter Dialoge recht häufig Fehlermeldungen auf, es gibt also zum Teil erhebliche Kompatibilitätsprobleme.

VBIDE-Bibliothek: Die Entwicklungsumgebung ist durch eine eigene Bibliothek programmierbar: Microsoft Visual Basic for Applications Extensibility, VBIDE im Objektkatalog, Startobjekt *VBE*. Die Bibliothek ist dann von Nutzen, wenn Sie die Bedienung der Entwicklungsumgebung verbessern möchten oder per Programmcode den Code einer Excel-Datei verändern möchten.

2.5 Neu in Excel 7

In Excel 7 wird VBA-Code per Default nicht mehr in der jeweiligen Landessprache, sondern in Englisch angegeben. Aus der Sprachumstellung können diverse Probleme resultieren.

Neu in Excel 7 ist der Mechanismus ActiveX-Automation (der damals noch Object Automation hieß). Er wird dazu genutzt, um verschiedene Erweiterungen in Excel einzugliedern. Die entsprechenden Bibliotheken müssen vor ihrer Verwendung mit EXTRAS | VERWEISE aktiviert werden.

Office-Bibliothek: Um die Suche nach Office-Dokumenten zu vereinfachen und zu vereinheitlichen, werden jetzt mit jedem Dokument zusätzliche Informationen gespeichert. Die Möglichkeiten, die sich daraus ergeben, erkennen Sie an den Dialogen bei DATEI | ÖFFNEN zum Suchen nach Dateien bzw. bei DATEI | EIGENSCHAFTEN zum Einstellen von Zusatzinformationen. Im VBA-Code können Sie auf die Zusatzinformationen über die Eigenschaften *BuiltinDocumentProperties* und *CustomDocumentProperties* zugreifen.

Office-Binder-Bibliothek: Eine weitere Neuerung des Office-Pakets besteht darin, dass beliebige Office-Dokumente (Texte, Tabellen, Datenbanken etc.) in Form von Sammelmappen zusammengefasst werden können. In der normalen Anwendung lassen sich damit zusammengehörige Dateien übersichtlicher verwalten. Im Programmcode werden Sammelmappen über die neuen Objekte *Binder* und *Section* verwaltet.

2.6 Probleme und Inkompatibilitäten

VORSICHT

In allen Excel-Versionen kann es beim Bearbeiten bzw. Testen von VBA-Code zu Abstürzen kommen. Speichern Sie regelmäßig! Beachten Sie auch, dass Excel nicht immer vollständig abstürzt. Manchmal kommt zwar die obligatorische Systemfehlermeldung, Excel bleibt aber im Speicher und blockiert weiter alle zuletzt geöffneten Dateien (ohne dass aber eine Möglichkeit besteht, diese noch zu speichern). Damit Sie wieder richtig weiterarbeiten können, müssen Sie Excel ganz beenden. Unter Windows NT/2000/XP verwenden Sie dazu den Task-Manager. Unter Windows 9x/ME führen Sie Strg+Alt+Entf aus; es erscheint eine Task-Liste, aus der Sie Excel auswählen und gewaltsam stoppen können.

Allgemeine Probleme

- VBA kennt noch immer keine Optimierungen bei der Auswertung von Bedingungen: Eine Abfrage in der Form *If $x \geq 0$ And $Sqr(x) < 3$* führt bei negativen Zahlen in x zu einem Fehler. (Dieses Problem besteht in Visual Basic schon seit der ersten Version, d.h., ich habe die Hoffnung auf Besserung in diesem Punkt aufgegeben.)

- Bei allen Excel-Versionen gibt es Probleme mit dem Operator *Is* zum Objektvergleich. Dieser Operator sollte feststellen, ob zwei Variablen auf dasselbe Objekt verweisen. Leider funktioniert das nicht immer.
- Der Wechsel von der VBA-Entwicklungsumgebung zu Excel klappt nicht, wenn gerade der Objektkatalog das aktive Fenster ist. Sie müssen in der Entwicklungsumgebung zuerst ein anderes Fenster anklicken.
- Beinahe ebenso lästig ist es, dass Symbolleisten von Excel immer wieder in der Entwicklungsumgebung auftauchen und dort im Weg sind. Sobald sie angeklickt werden, erfolgt ein (meist ungewollter) Wechsel zurück zu Excel.
- Wenn Sie eine Logitech-Radmaus verwenden, funktioniert das Mousrad in der VBA-Entwicklungsumgebung nur, wenn Sie die dazugehörige Maus-Software installieren. Warum das Mousrad bei allen anderen Programmen auf Anhieb (ohne Software-Installation) funktioniert und nur im VBA-Editor den Dienst verweigert, ist rätselhaft geblieben.

Probleme mit MS-Forms-Dialogen bzw. -Steuerelementen

Excel und die meisten VBA-Kommandos sind blockiert, solange sich der Eingabecursor in einem MS-Forms-Steuerelement in einem Tabellenblatt befindet. Nur bei Buttons kann das durch *TakeFocusOnClick=False* verhindert werden. (Die Default-einstellung lautet allerdings *True* und ist der Grund, weswegen es mit Buttons in Tabellenblättern oft Probleme gibt. Die auftretenden Fehlermeldungen sind ohne jede Aussagekraft.)

Wenn im Tabellenblatt auch andere Steuerelemente verwendet werden, muss der Eingabecursor per Programmcode (etwa durch *Worksheets(n).[A1].Activate*) in eine Zelle gesetzt werden, um damit sicherzustellen, dass er nicht auf ein Steuerelement gerichtet ist.

Generell bereitet die Verwendung von Steuerelementen in Tabellenblättern (statt in Formularen) enorme Probleme und löst – besonders unter Excel 2002 – nicht nachvollziehbare Fehler und zum Teil sogar Excel-Abstürze aus. Betroffen von diesen Problemen ist insbesondere die Beispieldatei *07\userform.xls*, aus der bei der Neuauflage dieses Buchs einige Beispiele entfernt werden mussten; diese Beispiele funktionierten unter Excel 2000 noch problemlos, verursachten unter Excel 2002 aber Abstürze.

Probleme in Excel 2003, Inkompatibilitäten gegenüber Excel 2002

Erfreulicherweise habe ich beim Test der Beispieldateien zu diesem Buch so gut wie keine Kompatibilitätsprobleme im Vergleich zu Excel 2002 festgestellt. Die einzige Ausnahme betraf das Web-Services-Beispiel aus Kapitel 15. (Microsoft betrachtet das Web Services Toolkit aber ohnedies als ein *not supported product*.)

Probleme in Excel 2002, Inkompatibilitäten gegenüber Excel 2000

Durch den Wechsel von Excel 2000 auf Excel 2002 treten nur relativ wenige VBA-Probleme auf – aber die von Microsoft versprochene vollständige Kompatibilität ist leider nur ein frommer Wunsch. Beim Testen der Beispielprogramme dieses Buchs gab es unter anderem die folgenden Probleme:

- Die Defaultsicherheitseinstellungen für Makros lautet nun HOCH statt MITTEL. Deswegen können nur VBA-Makros erst dann ausgeführt werden, wenn Sie die Einstellung auf MITTEL stellen (siehe Abschnitt 4.7).
- VBE-Code zur dynamischen Veränderung von Code kann nur ausgeführt werden, wenn die neue Option ZUGRIFF AUF VISUAL-BASIC-PROJEKT VERTRAUEN im Dialog EXTRAS | MAKRO | SICHERHEIT | VERTRAUENSWÜRDIGE QUELLEN aktiviert wird. (Per Default ist das nicht der Fall.)
- Die Eigenschaften *Bold*, *Italic* etc. der *Font*-Klasse liefern nicht mehr wie bisher nur *True* und *False*, sondern manchmal auch *Nothing*. Daher können die Eigenschaften nicht mit mehr mit *Boolean*-Variablen verarbeitet werden. Verwenden Sie stattdessen *Variant*-Variablen.
- Manche Assistenten und Add-Ins werden nicht mehr mitgeliefert und sind (wenn überhaupt) nur noch als Download im Internet zugänglich. Das betrifft z.B. den in Kapitel 9 beschriebenen Vorlagenassistenten.
- Pivotfelder werden bisweilen anders benannt als bisher (z.B. "Summe von xy" statt "Summe - xy"). Code, der sich auf die alte Schreibweise verlässt, funktioniert nicht mehr.
- Zuweisungen der Form `QueryTable.Name = "xyz"` werden nicht exakt ausgeführt: Wenn es schon einmal eine `QueryTable` mit dem Namen "xyz" gegeben hat, dann wird die neue `QueryTable` mit "xyz_1", "xyz_2" etc. bezeichnet – auch wenn die alte `QueryTable` längst gelöscht wurde und daher keine Verwechslungsgefahr mehr besteht.
- In Tabellenblättern eingebettete Buttons erscheinen nach dem Loslassen der Maus manchmal weiterhin niedergedrückt (d.h., die Darstellung springt nicht zurück in die Ausgangslage). Daraus ergeben sich zwar keine weiteren Probleme, die Buttons wirken aber optisch falsch.

Probleme in Excel 2000, Inkompatibilitäten gegenüber Excel 97

Auch zwischen Excel 97 und den Nachfolgeversionen 2000 und 2002 gibt es nur wenige Kompatibilitätsprobleme. Am ehesten führen die geänderten Orte von Konfigurationsdateien zu Schwierigkeiten. Neu ist etwa, dass die so genannte »persönliche Makroarbeitsmappe« erstmals wirklich persönlich ist, d.h., dass diese Datei für jeden Benutzer angelegt und an einem eigenen Ort gespeichert wird. Daher gelten darin durchgeführte Änderungen nicht mehr global für alle Excel-Anwender. Grundsätzlich

ist das ein Vorteil, zumal noch immer die Möglichkeit besteht, globale Makros zu definieren. Details zu Excel-Konfigurationsdateien finden Sie in Abschnitt 5.9.3.

Ebenfalls mit den Konfigurationsdateien zu tun haben die *Application*-Eigenschaften *TemplatesPath* und *StartupPath*, deren Wirkung sich mit Excel 2000 geändert hat. Sie verweisen jetzt auf die persönlichen Vorlagen- bzw. Xlstart-Verzeichnisse (statt wie bisher auf die globalen Verzeichnisse). Weitere Informationen zu diesen Eigenschaften gibt Abschnitt 5.6.5.

Probleme kann auch das Löschen von Tabellenblättern bereiten: Werden diese mit *Worksheets(...).Delete* gelöscht, kommt es manchmal vor, dass die resultierende Datei intern defekt ist. Sie kann zwar gespeichert werden, jeder Versuch sie später wieder zu laden, führt aber zu einem Absturz von Excel. Es ließ sich nicht feststellen, unter welchen Umständen dieser Fehler ausgelöst wird. (Er hat aber an sich nichts mit der *Delete*-Methode zu tun. Dasselbe Problem kann auch auftreten, wenn das Blatt manuell gelöscht wird.)

Daneben sind bei der Arbeit mit Excel 2000 sporadisch Detailprobleme aufgetreten, deren Ursache unklar geblieben ist: Da stürzte Excel ab, bis eine bisher undeklarierte Variable explizit als *Variant* deklariert wurde, dort gab es »Automatisierungs-Fehler«, bis aus einem *Select* ein *Activate* gemacht wurde etc.

Kompatibilitätsprobleme gegenüber Excel 5 und Excel 7

Die folgenden Informationen gelten, wenn Sie Excel-5- oder Excel-7-Programme auf Excel 97, 2000 oder 2002 umstellen möchten. Was das betrifft, sieht es leider wenig rosig aus.

VERWEIS

Selbst Microsoft räumt die Umstellungsprobleme ein und führt auf der Seite <http://support.microsoft.com/default.aspx?scid=KB;EN-US;q162721&> (das ist der Knowledge-Base-Artikel Q162721 in der MSDN-Library) nicht weniger als 75 Einzelprobleme auf.

Die folgende Liste ist daher nicht vollständig, sondern zählt nur die wichtigsten Probleme auf. Die meisten der hier geschilderten Probleme sind zweifellos Kleinigkeiten. Aber es sind solche »Kleinigkeiten«, die oft einen Tag intensiver Fehlersuche kosten.

- Wenn auf Tabellenblätter zugegriffen wird, funktionieren manche Methoden nur mit *Worksheets(n)*, nicht aber mit *Sheets(n)* (auch wenn in beiden Fällen auf dasselbe Tabellenblatt verwiesen wird).
- Die Formatierung von Diagrammen per Programmcode liefert zum Teil andere Ergebnisse als unter Excel 7.
- VBA-Code, der auf *Selection* zurückgreift, bereitet manchmal Schwierigkeiten. Das ist umso ärgerlicher, weil im Regelfall die automatische Makroaufzeichnung die Quelle problematischen Codes ist. Abhilfe: Ändern Sie die beiden folgenden Zeilen

```
object.Select oder object.Activate
Selection.methode
```

zu

```
object.methode.
```

Dieselbe Vorgehensweise gilt natürlich auch, wenn *Selection* durch *With* für mehrere Zeilen verwendet wird. (Geben Sie beim *With*-Kommando statt *Selection* das tatsächliche Objekt an.)

- Zuweisungen an *OnEventXxx*-Eigenschaften werden seit Excel 97 in der Excel-Datei gespeichert, in Excel-5/7 hingegen nicht. Die meisten Excel-5-/7-Anwendungen verlassen sich daher darauf, dass beim Laden einer Datei alle *OnEventXxx*-Eigenschaften leer sind. Dies trifft jetzt nicht mehr zu und kann erhebliche Probleme bereiten.

Probleme gibt es zumeist auch beim Versuch, *OnEventXxx*-Prozeduren durch die in Excel 97 neu eingeführten Ereignisprozeduren zu ersetzen: Excel beklagt sich plötzlich darüber, dass es die in *OnEventXxx*-Eigenschaften eingestellten Prozeduren nicht mehr findet.

Erschwerend kommt hinzu, dass keine Möglichkeit dazu besteht, alle initialisierten *OnEventXxx*-Eigenschaften festzustellen. Sie müssen vielmehr im Direktfenster für jede mögliche *OnEventXxx*-Eigenschaft (für jedes Tabellenblatt!) testen, ob die Eigenschaft belegt ist. Wenn ja, müssen Sie die Eigenschaft durch die Zuweisung einer leeren Zeichenkette "" löschen. Viel Spaß!

- Die Methode *OpenText* lieferte in Excel 7 als Rückgabewert *True* oder *False*, je nachdem, ob der Import der Daten gelungen ist oder nicht. Seit Excel 97 darf die Methode überhaupt nicht mehr als Funktion verwendet werden, Rückgabewert gibt es keinen mehr. Mögliche Fehler müssen mit einer Fehlerbehandlungsroutine abgefangen werden.
- Die Syntax der Parameter *Destination* und *Connection* zur Angabe externer Datenquellen bei der Methode *PivotTableWizard* hat sich geändert. Excel-7-Programmcode läuft im Regelfall nicht mehr.
- MS-Forms-Dialoge (neu seit Excel 97) können nicht mit Zeichenelementen, Textfeldern oder anderen Office-Objekten dekoriert werden. Es gibt daher viel weniger optische Gestaltungsmöglichkeiten als bei Dialogen aus Excel 5/7.
- Im Kompatibilitätsmodus zur Anzeige und Verwaltung von Dialogen aus Excel 5/7 tritt ein Problem auf, wenn Sie einem Listenfeld einen Zellbereich zuzuweisen versuchen:

```
Set listenfeld.List = Sheets("Tabelle1").[a1:a4]
```

Die direkte Zuweisung von Zellbereichen an Listen wird offensichtlich nicht mehr unterstützt. Sie müssen entweder eine Schleife über alle Zellen ausführen und die Einträge einzeln mit *AddItem* hinzufügen oder aber auf das neue MS-Forms-Listenfeld umsteigen.

Ein Kapitel für sich sind die seit Office 97 neuen Menü- und Symbolleisten, die durchaus nicht nur mit Verbesserungen verbunden sind:

- Vorhandenen Menüs, die auf der Basis der *Menu*-Objekte von Excel 5/7 erstellt wurden, können keine neuen Ereignisprozeduren zugewiesen werden. (Genau genommen funktioniert die Zuweisung, nur wird diese nicht gespeichert.) Sie können vorhandene Menüs also weiterverwenden, aber nicht mehr ändern. Die Umwandlung in neue *CommandBar*-Menüs ist nur manuell und mit großem Aufwand möglich.
- Es gibt keinen Menüeditor mehr. Das manuelle Zusammenstellen neuer Menüs erfolgt über den Dialog ANSICHT | SYMBOLLEISTEN | ANPASSEN und ist mit Hunderten von Mausklicks verbunden. Kontextmenüs können überhaupt nur noch per Programmcode verändert werden.
- Veränderungen in vordefinierten Menüs werden nicht mehr in der Excel-Datei gespeichert, sondern separat für jeden Benutzer in einer eigenen Datei. Daher ist zusätzlicher Code notwendig, wenn in Excel-Anwendungen Veränderungen an vorhandenen Menüs oder Symbolleisten durchgeführt werden sollen. (Neue Symbolleisten können wie in Excel 5/7 angebunden werden.)

Kompatibilitätsprobleme gegenüber Excel 5

Der wesentlichste Unterschied zwischen Excel 5 und allen nachfolgenden Versionen besteht darin, dass VBA-Code nicht mehr in der jeweiligen Landessprache formuliert wird, sondern immer auf Englisch. (In der deutschen Version von Excel 5 lautete eine Schleife z.B. *Für i=1 Bis 3: Nächste i!*) Seit Excel 97 wird der Code beim Laden automatisch ins Englische übersetzt, wobei sich aber manchmal kleine Fehler einschleichen.

