Sue Mosher

Digital Press

Microsoft® Outlook Programming

Jumpstart for Administrators, Developers, and Power Users

COMMUNICATIONS

Microsoft Outlook Programming

This Page Intentionally Left Blank

Microsoft Outlook Programming

Jumpstart for Administrators, Power Users, and Developers

Sue Mosher

Digital Press
An imprint of Elsevier Science
Amsterdam • Boston • London • New York • Oxford • Paris • San Diego
San Francisco • Singapore • Sydney • Tokyo

Digital Press is an imprint of Elsevier Science.

Library of Congress Cataloging-in-Publication Data

A catalog record for this book is available from the Library of Congress.

ISBN: 1-55558-285-9

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library.

The publisher offers special discounts on bulk orders of this book.

For information, please contact:

Manager of Special Sales
Elsevier Science
200 Wheeler Road
Burlington, MA 01803
Tel: 781-313-4700
Fax: 781-313-4882

For information on all Digital Press publications available, contact our World Wide Web home page at: http://www.digitalpress.com or http://www.bh.com/digitalpress

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Contents

Foreword

How many times have you sat in front of your screen, thinking about that one feature you would add to Outlook—if you only had the chance?

One of the benefits of my job is that, in addition to helping to design each new version of Outlook, I meet a lot of people with ideas about how to make Outlook better. Some of these people I run across in the usual places—the seat in front of me on an airplane (inevitably with seatback reclined as far as possible), at the symphony ticket office, or in a local library. Other times, I meet people in places I'd never expect, doing things with Outlook we never imagined.

During a trade show that I used to attend every summer in New York City, I finished up early one afternoon and decided to return to my hotel. As I entered the lobby, I heard unmistakable techno music coming from the back of the hotel. Curious what kind of rave could be going on at 3 P.M., I walked down the hall to see what was going on. As I approached the grand ballroom, the music got louder, and I could see shadows from spinning disco lights under the door. I cracked the door open and peered into the room to find a bustling hive of pre-adolescent activity—a model search for girls 10–14 years old.

The far half of the ballroom was occupied by some sort of contest involving lip syncing and dancing, complete with flashing strobe lights, Britney Spears hits, and anxious mothers petitioning the judges. In another corner, a tropical island set had been constructed; girls in front of *faux* palm trees, intense lights, and painted ocean waves were being photographed by two or three professionals barking "energy, energy!" and "great, great, you're fantastic!"

In the corner nearest to me were several men, outfitted in California chic, sitting behind folding tables interviewing yet another group of would-be models for their agencies. What caught my eye was that in front of the

agent nearest me was a laptop running Outlook. As each girl approached, he created a new contact in Outlook and entered information about her as she talked. When he finished asking his stock questions, he saved the contact, thanked the girl and her mother, and waved on the next person in line.

Intrigued, I came back later after the day's recruiting activities ended. I introduced myself and told the agent that I work on the Outlook team at Microsoft. The agent described to me how he was using Outlook to manage his talent search. He collected information about the girls—height, weight, age, hair color, etc., and typed it into the notes section at the bottom of the contact form. He also entered his own numerical values for more subjective criteria—how pretty the girl's face was, or how outgoing her personality seemed. At the end of the day, the agent went back to his hotel room and manually applied a secret formula to each model's statistical information. He then called the girls with the highest appeal quotient and attempted to sign them to his modeling agency.

"What would be really useful," he told me, "would be if I could have actual fields to enter the information for each girl, and then have Outlook calculate the desirability of the models automatically." He went on to confide: "I bet you could sell a lot more copies of Outlook if you built that right into the product."

Whatever the dubious merits of the agent's suggestion, how many times have you wished that Outlook could do something that there seems to be no way to do? Maybe you've thought about clicking a button to have Outlook automatically file old mail into separate folders for each month in which you received mail. Or maybe, you wish Outlook could validate ZIP codes for your contacts, or assign you a task when you have expense reports to approve, or manage vacation requests from your employees.

The good news is that you can do all of these things and more, and it doesn't take any existing programming expertise. Outlook has an easy-to-use but exceptionally powerful programming model that you can use to customize Outlook for your personal use or for the needs of the organization you administer.

Sue Mosher has been the high priestess of Outlook since before Outlook was even a product. From the first version of Outlook onward, Sue has been at the public forefront of all things Outlook and Exchange—first as an influential voice on the web and on newsgroups, more recently as a valued Microsoft Outlook MVP. Her Web site is so full of reliable information that I'll often try to find the answer I'm looking for there before using Microsoft's internal tools to research an issue.

Reading *Microsoft Outlook Programming: Jumpstart for Administrators, Power Users, and Developers* is a great way to get started with Outlook programming, especially for the beginner. The book is written in plain language, without complicated jargon or convoluted, abstract concepts to learn. Even if you've never written a line of code in your life, you will be able to follow the real-world examples in each chapter to rapidly gain knowledge and confidence. Before you know it, you will be crafting your own programs.

If you are experienced with VBA but have never programmed with Outlook, this book also makes a great starting point. You will apply your expertise to concepts unique to Outlook programming: custom forms, stores, the event model, integrating Outlook data with XML Web services, and more. Even if you are already a seasoned veteran, you will appreciate the technical depth of the information presented and the relevance of the examples therein. No matter what your skill level, you will learn how to program Outlook to work the way your business works—saving you time and helping you to manage your information more effectively.

Alas, I didn't make my usual trip to New York this summer, so I don't know for sure if the gaggle of starry-eyed girls took over the hotel again as in past years. I do know this much, however—when I eventually run into you at the airport or in your office and find out you've customized Outlook to, say, manage turtle breeding or keep track of what flavors of ice cream you own—sigh—at least I'll know who to blame.

Jensen Harris
Lead Program Manager
Microsoft Outlook
July 2002

This Page Intentionally Left Blank

Acknowledgments

More people than I could ever name played a role in this book, but at the top of the list is Digital Press publisher Theron Shreve, who saw how much a new basic Outlook programming book was needed. Pam Chester at DP and my long-time friend and agent, Valda Hilley, also smoothed the way with incredible behind-the-scenes effort.

After answering thousands of Outlook programming questions in the Microsoft newsgroups, I must offer my thanks to all the Outlook administrators, users, and developers who have alternately stimulated and annoyed me. You constantly push me to find new techniques to work around Outlook's limitations and better ways to explain the basics.

A special debt of gratitude goes to all the Outlook MVPs (Most Valuable Professionals recognized by Microsoft for great practical knowledge and grace under fire when helping other users). Ken Slovak played a key role by reading every chapter and testing all the code, but if you find any code that won't run, it's my fault. I also need to single out Randy Byrne, Dmitry Streblechenko (developer of Outlook Spy and Redemption), and Siegfried Weber (an Exchange MVP), who sometimes talk way over my head but always remain a source of inspiration, and Steve Moede and Diane Poremsky, whose great code samples taught me a lot.

At Microsoft, many people contributed to my ability to shed light on Outlook programming, particularly Bill Jacob, Abdias Ruiz, John Eddy, and Paul Cornell. I'd also like to thank Outlook lead program manager Jensen Harris for his support and inspiration, and for writing the foreword to this book. People at other companies who inspired and instructed include Phil Seeman and Sandy Billingsley.

The physical production of this book began with much appreciated formatting assistance from Ann Gosnell. Final production was in the capable hands of Alan Rose, Harlan James, Lauralee Reinke, and Jordan Marx.

The following articles that originally appeared in *Windows & .NET Magazine* (http://www.winnetmag.com) or the *Exchange & Outlook Administrator* newsletter (http://www.exchangeadmin.com <http://www.exchangeadmin.com/>) provided portions of the content for this book. Copyright 2001–2002, Penton Media, Inc. Reused with permission:

"A Primer About Dates" (InstantDoc ID 16455)

"More Fun with Dates" (InstantDoc ID 20078)

"Taking Outlook to Task" (InstantDoc ID 19694)

"Making Contact" (InstantDoc ID 22254)

"Linking Outlook Contacts" (InstantDoc ID 20353)

"More About Links" (InstantDoc ID 20475)

"Improve your Custom Outlook Forms" (InstantDoc ID 23483)

My virtual assistant, Kathleen Redding, was indispensable in keeping the rest of my business organized while I filled my head with Outlook code. You are a treasure, and I am so blessed to have you as a friend!

My family, Robert and Annie, tolerated my long hours at the keyboard and helped me rejoice at the minor triumphs along the way. Those little celebrations really helped. I promise we'll have time to eat something other than pizza. Even Dymka the cat seems more relaxed now that she can stretch out on a desk that's not covered with page proofs and CDs.

Finally, I thank God for the opportunity to share this knowledge and help people connect with each other.

Introduction

It's been more than three years since I first tried to explain how to program Microsoft Outlook in "Teach Yourself Microsoft Outlook 2000 Programming in 24 Hours" (Sams) in a way that would appeal to a wide range of Outlook users. I wanted potential power users to learn how to turn Outlook into a more productive desktop tool. I hoped to help administrators build forms that would enhance collaboration in their organizations and small personal tools to automate repetitive administrative tasks. To experienced developers, I aimed to explain the many quirks that can make programming Outlook difficult.

Since that time, of course, Outlook 2002 has been released, adding new programming features, but also new challenges. In the wake of LoveLetter and dozens of other viruses that targeted Outlook, Microsoft dramatically changed the functionality that makes it possible to work with email addresses and send messages programmatically. One of the goals of this book is to explain the new security features thoroughly and lay out your options, including a third-party library, Redemption. I also wanted to explain more of the architecture, especially the elements that tend to be stumbling blocks for new Outlook developers.

I've seen increasing interest in using Outlook forms in non-Exchange environments and expect this trend to continue, especially with recent third-party enhancements for Outlook that allow data sharing without Exchange. At the same time, Microsoft's .NET initiative has made XML Web services an incredibly easy-to-use development strategy for Outlook 2002, bringing new functionality to Outlook with just a couple of lines of code. Therefore, in this new book, I wanted to highlight a broader range of collaboration environments than just Exchange and database connections.

Another goal was to organize the material better so that you can find the building blocks you need to handle everyday Outlook tasks. For example, in

Chapter 14, "Working with Items," you'll find all the techniques for accessing a particular Outlook item. Be sure to check out Appendix C for a list of key procedures you'll want to reuse in your own projects.

I think it's important to say what this book is not: It is not a complete reference to the Outlook object model, nor is it a guide to building complex applications for installation in an enterprise or distribution to the general public. Excellent resources are available on both those topics, and you'll find them listed in Appendix A.

But what you won't find anywhere other than in this book is a balance between deep discussions of the practical aspects of Outlook development and the programming basics that make it possible for someone with no prior code-writing experience to get a great start. Instead of explaining everything there is about Outlook folders, for example, I'll show you the most important techniques—the building blocks that you'll use over and over again—and explain the best practices gleaned from years of listening to other Outlook developers. I have tried throughout the book to provide examples of these techniques that you can use immediately and then combine with other processes into new applications that you build yourself.

Conventions used in this book

This book uses different typefaces to differentiate between code and regular text, and also to help you identify important concepts.

Code appears in `monospace` font:

```
Item.BodyFormat = olFormatRichText
```

Placeholders for various expressions appear in *`monospace italic`* font. You should replace the placeholder with the specific value that it represents.

Text that you type is presented within quotation marks. New terms appear in *italics*.

The Notes, Tips, and Cautions scattered through the book try to call attention to information that will help you become a better Outlook programmer. A Note presents interesting information related to the surrounding discussion. A Tip offers advice or teaches an easier way to do something. A Caution advises you of potential problems and helps you steer clear of disaster.

1

What You Can Do with Outlook

Welcome to Microsoft Outlook's world of programming possibilities! Both Outlook 2000 and Outlook 2002 make it easier than ever to customize the program. This chapter gets you started by introducing the tools for Outlook programming and suggesting ways to start thinking about projects you want to try.

The highlights of this chapter include discussions of the following:

- What kinds of projects are possible in Outlook 2000 and 2002
- What tools you will use the most
- How to decide which tool to use
- How to determine which version of Outlook you have installed
- How to sketch out your plans for Outlook programming projects

1.1 Why program with Outlook?

Whether you have been using Microsoft Outlook for just a few days or more than a year, you've certainly figured out that it does more than e-mail—much, much more. It's not unusual to find people who have organized their entire lives with Outlook.

But if you asked each Outlook user how he or she puts the program to work, you would receive a different answer every time because people have their own ideas on how to organize the critical information in their lives. Wouldn't it be great if Microsoft could make Outlook so customizable that everyone could use it his or her own way? It might not be 100 percent possible, but the programming environment included with Outlook 2000 and 2002 is rich enough to let you make major strides toward bending Outlook to your will—or that of the organization you work for.

This book shows you how to use the programming tools in Outlook to make it your own. It's OK if you have never programmed before—this book shows you how! It's much easier than you might think. If you are an experienced programmer, you will see how to put Outlook's special features to work and how to work around its many quirks.

To help you get excited about the chapters ahead, take a look at this list of things you can do when you learn how to program with Outlook:

- Distribute a list of company holidays to people in the office so that they can instantly add them to their Outlook calendars
- Create your own custom rules to handle incoming e-mail
- Search and replace data, such as telephone area codes
- Create custom reports by integrating Outlook data with Word layouts
- Schedule a follow-up call for a meeting easily
- Create Outlook forms that duplicate the paper phone message, vacation request, and other business forms you use

Are you excited yet?

1.2 Outlook programming tools

Before digging into the details of Outlook programming, here is a preview of the primary tools you will be using:

- Visual Basic for Applications
- Outlook forms
- Visual Basic Scripting Edition (VBScript)
- Object models, not just for Outlook, but also for Collaboration Data Objects, Redemption, Scripting Runtime, Word, and Excel

1.2.1 Visual Basic for Applications

Outlook includes one of the richest available development environments—*Visual Basic for Applications* (or VBA for short). Outlook shares it with not only the other Office programs, but also programs such as AutoCAD that have licensed VBA as their programming environment.

The VBA programming environment is shown in Figure 1.1. (The screen shots in this book were taken using Windows XP. If you use a different operating system, your screen may look slightly different, but will

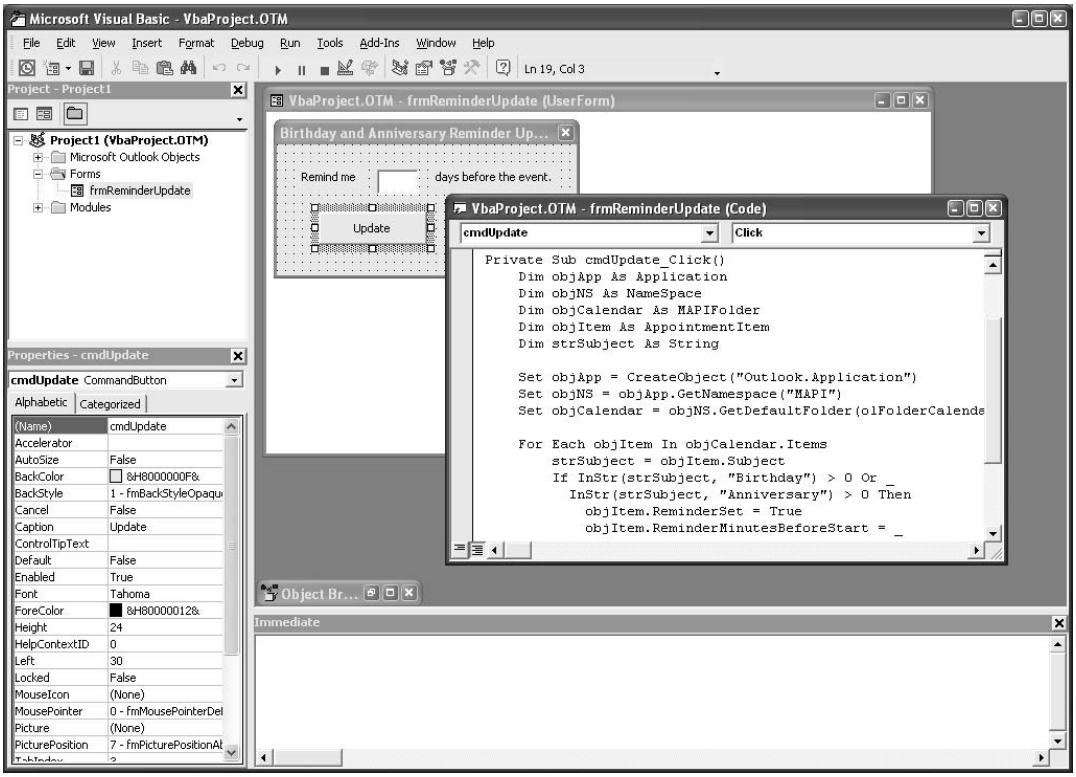

Figure 1.1 *VBA includes a rich form and code environment (compare with Figure 1.4).*

function the same.) Figure 1.1 includes all the tools a professional developer might want:

- Visual forms designer (for Windows dialogs, not for Outlook forms)
- Intelligent editor with color coding and dropdown lists to avoid code errors
- Detailed help on the things you can do with Outlook
- Debugging windows and other tools

VBA allows you to write code that handles the many events that take place when you work with your Outlook information—such as creating new items or switching from one folder to another.

VBA also gives you the ability to design dialog boxes that pop up to get information from the user and windows that stay on the screen to provide information to the user. For example, you might build a VBA form

to display how many vacation days you have used so far this year or the time that you last received messages in your Inbox.

Furthermore, you can use VBA to create macros that you can add to the Outlook toolbar to launch a telephone message form, search for and replace text in all the items in a folder, and expand Outlook's capabilities in many other ways. In Outlook 2002, the Rules Wizard can run a macro in response to a condition that you set in a rule.

You might have created macros in Word or Excel by turning on a macro recorder that watches your actions and then builds the appropriate code. Outlook does not include a comparable macro recorder, but the examples in this book show you how to use VBA to perform all the basic actions of creating messages and other items, addressing them, and sending them.

Note that the VBA techniques discussed in this book also apply if you want to move up from Outlook's integrated development environment to building more sophisticated Outlook tools with Microsoft's Visual Basic development program. (Visual Basic .NET requires different techniques that are beyond the scope of this book. We'll stick to VBA and, by extension, Visual Basic.)

1.2.2 Customized forms

The second stop on the road to Outlook programming proficiency is learning how to customize the basic Outlook forms.

Every item that you open in Outlook—whether it's an e-mail message, a contact, or an appointment—uses a particular form to display its data. (If you have programmed in Microsoft Access or Visual Basic, you may already be familiar with using forms as templates to display different data items.) You can customize these forms to show or hide fields, respond to user input and actions, and control other Outlook tasks. Customized forms can be shared with other users, both within your organization and across the Internet.

In many programming environments, you must start from scratch every time you want to create a new window for the user to interact with. Outlook is different in that it presents you with a group of built-in forms. To build a custom form, you start with one of the built-in forms and then add your own special touches.

For example, Figure 1.2 shows the default Contact form as it normally looks when you add a new item to the Contacts folder. Notice that the E-mail field shows only one address and provides no clue that as many as three addresses can be associated with this contact. In Figure 1.3, you see

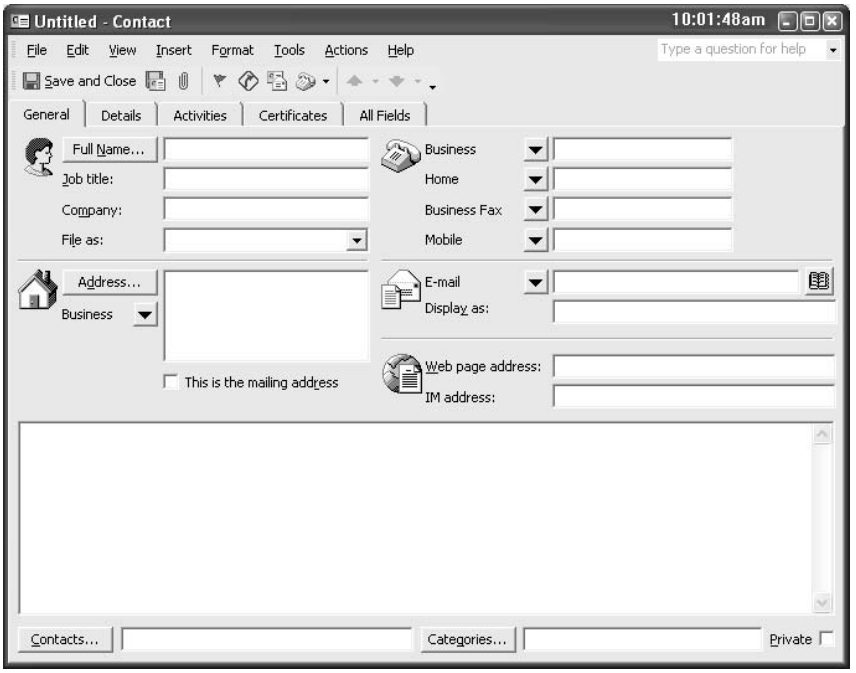

Figure 1.2
This Contact form has not been customized.

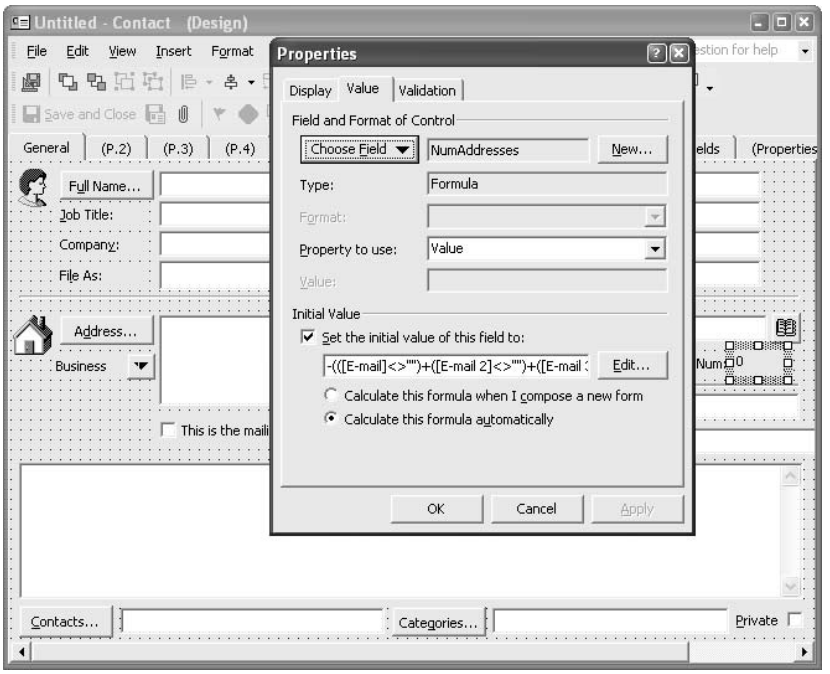

Figure 1.3
This Contact form has been customized to show the number of e-mail addresses for any contact item displayed using this form.

the same form, this time as it's being redesigned and customized with a `NumAddresses` property and text box to indicate how many e-mail addresses the contact has.

Is programming code necessary to do this? Not really. All it takes is a fairly simple formula, no more complicated than those you might have worked with in Microsoft Excel spreadsheets. You will see how it works in Chapter 17, "Extending Form Design with Fields and Controls."

Given that no real code is involved, is this really programming? Sure it is! Designing forms for user interaction is such an important part of programming that whole books have been devoted to that topic alone. Many of the changes you want to make to Outlook might involve nothing more than adding new fields and pages to existing forms to hold the data. Without writing any code at all, you can perform simple validation to make sure that the data meets your criteria for correctness and develop simple formulas such as the one for counting e-mail addresses.

1.2.3 Visual Basic Scripting Edition

A time will come, however, when you want your forms to do more. Maybe you will want to generate a task for a follow-up telephone call from an appointment and have Outlook automatically fill in the contact name for you, or perhaps you would like to be reminded to add a category to each outgoing message you send. When you are ready to go beyond entering data and manipulating it in simple ways, you can move up to *VBScript*, the shorthand name for Visual Basic Scripting Edition, the programming language behind Outlook forms.

You might have heard of VBScript in the context of Web pages. VBScript is one of several languages that can control what you see when you interact with a Web page. It also works with the Windows Script Host that Microsoft has included with Windows since Windows 98 to allow you to write programs that are stored as simple text files.

Scripts don't run as fast as other kinds of programs, but they enjoy the advantages of small size and portability. Having a script associated with an Outlook form hardly increases the size of the form at all.

VBScript is a little scary, though. It's like working without a net, because the built-in editor for building VBScript programs is, well, a text editor. Figure 1.4 shows a sample script for a form to distribute a list of company holidays within an organization. Notice the minimal toolbar. The form

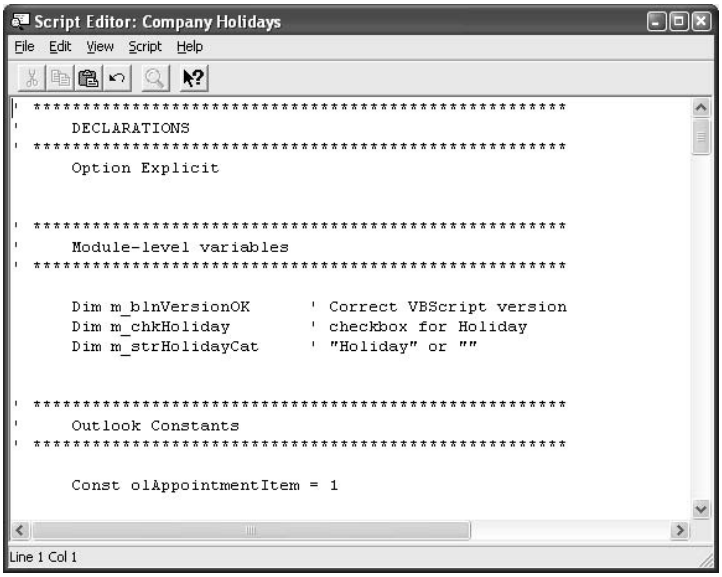

Figure 1.4 *The Outlook form script editor is just a text editor (compare with Figure 1.1).*

script editor also has none of the color-coding or automatic syntax checking found in VBA.

One sneaky technique you will learn in this book is to write and test your Outlook form code in the superior VBA code environment, make a few minor adjustments to adapt it to VBScript, and then copy and paste it into the script window of an Outlook form. This method cuts down on programming time immensely.

1.2.4 Office integration, object models, and beyond

Outlook can create Word or Excel documents, and Microsoft Office programs such as Excel and Word can create messages, appointments, and other Outlook items. You can even integrate Outlook with a Microsoft Access database. This integration occurs thanks to something called the *Outlook object model*. All the Office programs and many other Windows components have object models that reveal what these programs can do, the types of items they can work with, and the characteristics of those items.

Besides the Outlook object model, you also work with the Collaboration Data Objects (CDO) model. CDO can reach some corners of e-mail that Outlook by itself can't, such as the fields in the Global Address List on Microsoft Exchange Server. You will see several examples of how CDO delivers details that Outlook can't access.

Table 1.1 *Object Models Commonly Used When Programming with Outlook*

Object Model	Description
ActiveX Data Objects (ADO)	Exchange information with Access and other databases
Active Directory Services Interface (ADSI)	Interact with the Exchange and Windows 2000 directory
Collaboration Data Objects (CDO)	Access MAPI properties of Outlook items, as well as many Exchange Server properties
Outlook object model	Create and manipulate Outlook items, as well as react to application-level events
Other Office application object models	Work with Excel, Word, PowerPoint, and Access objects from within Outlook
Redemption	Access features that secure versions of Outlook may block and use some MAPI features for which the Outlook object model has no equivalent
Scripting Runtime	Working with Windows files and folders

CDO is sometimes thought of as a "light" version of MAPI, which is short for Messaging Application Programming Interface and is the foundation on which Outlook is built. You cannot use VBA to use Extended MAPI directly, but Redemption is an object model that wraps around MAPI, provides access to features that secure versions of Outlook may block, and exposes features for which the Outlook object model has no equivalent.

As you read what other people have done with Outlook programming—in this book and others, on the Web, or in newsgroups—you might hear about other object models relevant to designing with Outlook, such as those listed in Table 1.1.

For example, you will be using the Excel and Word object models to write reports from Outlook objects because Outlook has fairly limited print layout functions. You also will see how to write code that populates a list box from an Access database.

1.3 New programming features in Outlook 2002

The biggest change in Outlook 2002 is the addition of strict security features aimed at preventing a virus from using Outlook to propagate itself. We will cover this topic in depth in Chapter 13, "Understanding Outlook

Table 1.2 *Major Additions to the Outlook Object Model in Outlook 2002*

Component	Description
`ItemProperties` collection, `ItemProperty` object	Provides a single collection that contains all properties on an Outlook item, unlike `UserProperties`, which covers only custom properties
`Search` object, `Results` collection	Allows you to perform complex searches—across multiple folders in a Personal Folders file or Exchange mailbox—using SQL-like syntax
`Views` collection, `View` object	Allows you to create, delete, and modify Outlook folder views
`Reminders` collection, `Reminder` object	Provides objects, events, and methods for building custom handlers for Outlook reminders

Security." In addition, Table 1.2 lists the four main additions to the Outlook object model.

In addition to the objects and collections in Table 1.2, Outlook 2002 also adds a cancelable `BeforeDelete` event for Outlook items; a number of properties and methods to assist in deploying applications based on items in Outlook folders; events triggered by Cut, Copy, and Paste commands on a folder; a property to set the format of a message to plain text, HTML, or rich text; and many more small, but welcome additions. As we encounter them in later chapters, we'll make sure to note that they work only for Outlook 2002.

1.4 How to start

At this point, you might feel the hardest task in Outlook programming is knowing where to start. Do you start with VBScript or VBA? Do you work with the form first and then write the program code, or vice versa?

Start by choosing one or more compelling projects—ideas that will save you time in the long run, make repetitive tasks less burdensome, or perhaps just display information that is hard to get to in the basic Outlook interface. Try to be as specific as possible. Don't decide to build a project to make Outlook work just like GoldMine (a popular sales contact management program). Instead, pick a particular GoldMine feature you want Outlook to duplicate.

When you choose a project, don't start writing code or moving fields around on a form right away! Instead, take some time to outline what you want the project to accomplish, using what programmers call *pseudo code*.

But wait! You say you don't know how to write programming code. (That's why you bought this book, you protest.) No, we're not asking you to write a program (not yet), only to lay the groundwork. When you write pseudocode, you're walking through the logic of what you want to happen, without worrying about the exact language required to make it work.

For example, say you want to enhance Outlook's appointment form with a button that would create a new task for a follow-up telephone call to the person you met with. The pseudocode might look something like this:

```
User clicks button
    Show task form
    Copy details of meeting to task body
    Copy contact from meeting to the task's Contact field
    Set task due date for one week from the meeting date
    If task due date falls on a weekend, holiday, or vacation
    day, then adjust the due date to the next business day
    ...and so forth
```

Nothing in this list looks like programming, but it describes in detail what you want Outlook to do when the user clicks the follow-up call buttons. It won't take much to move from this pseudocode to the program code to implement these steps.

Which tool is appropriate for a particular project? Table 1.3 provides some recommendations of the tools you are most likely to use in particular

Table 1.3 *Choosing Outlook Tools*

If You Want To . . .	You Will Probably Use This Approach
Show additional information on an Outlook form	Modify an Outlook form
Make something happen in response to something the user does with an Outlook item	Modify an Outlook form with VBScript code
Write a macro that can be run from the Outlook toolbar	Write a routine in VBA
Make something happen when the user starts Outlook, switches to a different folder, or performs other actions that don't involve a particular Outlook item	Write a routine in VBA
Display status information as the user performs various Outlook tasks	Create a form in VBA with a routine to show the status information

situations. Don't take these recommendations as hard and fast rules. In many cases, you can approach a project in several ways. As you work through the examples in the chapters that follow, you will develop a better feel for which Outlook tool works best where.

1.5 A word on Outlook versions and setup

This book aims to cover the basics of programming for Outlook 2000, which was the first version to include an integrated VBA environment, and Outlook 2002. Much of the content is also relevant to Outlook 97 and Outlook 98, especially the chapters on designing Outlook forms. You can also run code to automate Outlook from VBA in Word or Excel, from Visual Basic, or even from Windows Script Host scripts, so you shouldn't feel left out if you don't have Outlook 2000 or 2002 yet. You just won't be able to program for those versions' application-level events or use objects, methods, or properties introduced in the newer versions.

Among Microsoft Office programs, Outlook 2000 and Outlook 98 have the peculiar distinction of being two applications in one. Depending on how you install Outlook, you may be working in Internet Mail Only mode or in Corporate or Workgroup mode. Each mode includes features not available to the other. For example, Outlook 2000 includes a Send Plain Text Only check box on the Contact form, but that feature works only in Internet Mail Only mode. To check your version, choose Help, About. In Outlook 2000 or 98, the second line will tell you whether you are in Internet Mail Only or Corporate Workgroup.

Another factor that makes a difference in how Outlook operates is whether you connect to a Microsoft Exchange Server, Microsoft's enterprise

Table 1.4 *Key Outlook Development Components*

Component	Features
Microsoft Outlook	VBScript support
	CDO
Office Shared Features	Digital Signature for VBA Projects
	Office XP Web Components (optional)
	Office 2000 Web Components (optional)
	VBA/Visual Basic Help
Office Tools	HTML Source Editing/Web Scripting/Web Debugging

e-mail and collaboration server. In Chapter 23, "Exchange Server, Databases, and XML Web Services," you will look at some particular issues related to creating applications for Exchange Server users.

The default Office or Outlook installation does not include all the development tools that you might want to be able to use. You can use Control Panel, Add/Remove Programs to modify your installation of Outlook or Office to make sure that the components in Table 1.4 are available. Select Office or Outlook, then Change. When the Installer opens, choose the Add or Remove Features option to add these components.

1.6 Summary

At first, Microsoft Outlook programming can seem complex because there is more than one tool and no clear picture of where to start. However, by modifying Outlook forms and creating custom programs in VBA, you can build powerful applications that use electronic messaging as a vehicle for all kinds of other interactions.

This book is divided into sections that introduce Outlook skills one at a time with examples that you can easily try on your own computer. After an introduction in Part 1 to VBA design, in Part II you learn programming code techniques frequently used in Outlook. Experienced programmers who want to incorporate Outlook automation into their projects may want to skip ahead to Part III, which presents the key techniques related to working with Outlook folders and items. You'll also learn about the security features in Outlook that make some properties and methods difficult to use and about how to use events in VBA to run your Outlook code automatically. Part IV covers Outlook form design with many examples. Finally, in Part V, you'll find out how to integrate Outlook with Word, Excel, Access, and Exchange to print reports, connect to databases, and build collaboration tools. We also discuss how to share your Outlook forms and VBA applications with other users and how to manipulate menus, toolbars, and the Outlook Bar

Code samples specifically related to this book can be found at http://www.outlookcode.com. See Appendix A for other Outlook development resources on the Web and elsewhere.

Outlook VBA Design