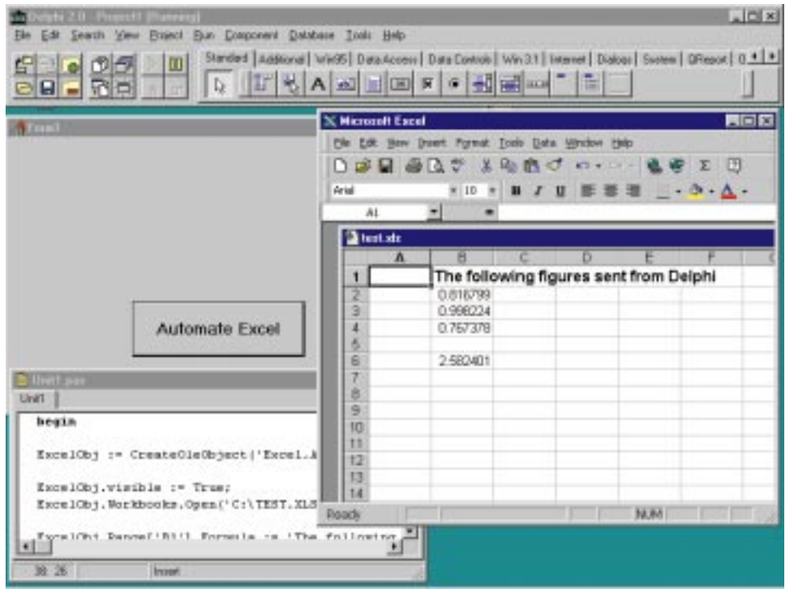
The Outlook is variable

Tim Anderson finds Outlook, the latest Visual Basic-enabled Office 97 application, to be a powerful but frustrating solution. He gives his views about programming and creating forms.

Cedric Maddox writes: "*I'm trying to get to grips with Delphi. Your review of the Appleman Guide was very interesting and I would like a similar guide for Delphi: can you recommend one? If there isn't a Delphi guide available, would the Appleman Guide for VB help with Delphi API calls?*"

Much of Daniel Appleman's guide to the Win32 API (Ziff-Davis) would be useful to Delphi developers since it is, after all, the same API. I hesitate to recommend it though, because large parts are specific to Visual Basic. Calling the Windows API is actually easier in Delphi than in Visual Basic: partly because the language is a better fit, with direct support for pointers and easy handling of Windows messages; and partly because Delphi's developers have helpfully defined the types, function and procedure headers for you.



Get the syntax right and automating Excel is a powerful way of extending a Delphi application (*see "Delphi and Excel 97", below*)

Often, you can use an API function as if it were native Pascal. But although calling the API is easy, understanding how it works is another matter. More advanced Delphi titles like *Delphi 2.0 Unleashed* or *Delphi 2 Developer's Guide*, both from Sams, contain helpful API tips. The official Microsoft reference is essential, and an online version comes with Delphi. Finally, Charles Petzold's *Programming Windows 95* (Microsoft Press) is useful not as a reference title, but as an explanation of how Windows works behind the scenes.

Unfortunately, if you want to get deeply into the Windows API, you have to put up with reference material aimed at C and C++ programmers, since Windows itself is mainly written in these languages.

Delphi and Excel 97

Another reader's problem is: "*According to my testing, there is a problem with Delphi 2 and OLE-automation using the version of Excel delivered with Office 97. Specifically, I can start Excel via OLE-automation and do some things (like ExcelApp.Visible := True), but if I try to access a range, for instance, it just crashes with an EOLESysError. Exactly the same test program works fine if I uninstall Office 97 and use Office 95 instead.*"

I suspect this is a bracket problem. The code in Fig 1, when used with Excel 97, fails with a "Member not found" message. The solution is to replace the last line with:

```
RangeObj := ExcelObj.
Range['B2:B4'];
```

Rich text problems

Gavin Docherty writes: "*For about three months I have been trying to paste bitmaps, metafiles and OLE objects into the standard richtextbox control found in the 32-bit common control DLL but without success. Then, to my amazement, you covered the subject in your May column and gave code examples for VB, but I couldn't get it to work. What have I done wrong?*"

Unfortunately, the news is not as good as I thought. The code printed in May's issue was tried and tested with Visual Basic 4.0. But at some point, some application or other had installed a later version of the RICHTX32.OCX component which makes it work, complete with an updated help file. This later version is required in order to work with pictures.

Programming Outlook 97

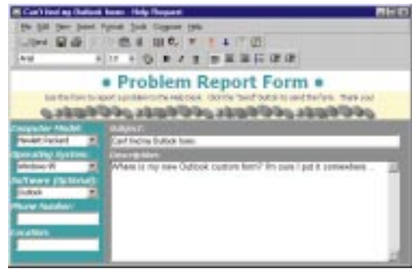
Outlook is an excellent starting point for an Office 97 application. Most people need an email reader and an address book or contact manager, and Outlook does both. The natural next step is to add functionality. For example, Outlook's contact menu already has an option to start a new letter to the current contact.

You might want to add new options, perhaps a choice of several standard letters. Getting more ambitious, you could pull in other information such as account information or product preferences. Another scenario could find you placing an order for a contact you have just called. Many things are possible since Outlook has a forms designer and a scripting language, but this is VB Script and not the powerful Visual Basic for Applications. VB Script is the cut-down version of Visual

p299 ➢

Book Reviews

■ Building Applications with Outlook 97
£37.49 (book and CD)
Microsoft Press



Outlook is a strange combination of elegant simplicity and arcane complexity. The Outlook bar is delightful, with easy access to network and internet mail, appointments, contacts and to-do list. With its own form designer and a version of Visual Basic, it seems the ideal platform for building groupware applications. Start developing, though, and Outlook shows its other face. Want to save a form? There are three ways to do it, claims the online help. Did you want the open folder, the file or the forms library? And if the latter, did you want the personal forms library, the folder forms library, or the organisation forms library?

The other problem is that Microsoft seems determined to disguise the close relationship between Outlook and Exchange. This book is a case in point. The cover wording refers several times to groupware but never to Exchange. Without Exchange, though, Outlook is only suitable for groups of one.

If you do have Exchange Server and want to develop with Outlook, this title is all-but essential. It describes Outlook's main elements and explains how to use the form designer and Visual Basic Script. It concludes with a step-by-step guide to creating three sample applications: the first is a form for requesting business cards, the second a help desk application, and the third tracks The Help Desk application is explained in *Building Outlook Applications* document production. This last is the most interesting since it links to an Access database using Data Access Objects. It's a useful guide, and really should have been part of the Office 97 Developer Edition. The only other caveat is that you will need a lot more help than this when it comes to managing the back-end of an Outlook application, Exchange Server itself.

■ Using Visual Basic 5.0
by Mike McKelvy, Ronald Martinsen and Jeff Webb — £36.99, Que

Smartly published to coincide with the release of Visual Basic 5.0, this title in Que's classic "*Using*" series begins right at the beginning, with topics like what is a program? And what is a variable? By the end of its 950 pages it is tackling API programming, callback functions and remote automation servers. The result is a comprehensive book, but lacking in sparkle.

There is too much here for real beginners, while experienced VB developers migrating to version 5.0 will find themselves skipping large chunks of the material. It has more the style of a manual than a real-world developer's book. Because Visual Basic is now such a large product, this comprehensive approach means little depth on any individual subject.

This is a good buy if you do not have printed documentation, with clear, thorough explanations covering every aspect of Visual Basic. Others will be better served by one of the many more specialist Visual Basic titles now available.

■ Instant Visual Basic 5.0 ActiveX Control Creation — £27.49, Wrox Press

Sporting no less than seven authors, this title covers the hottest new feature of Visual Basic 5.0: the ability to create ActiveX controls. It mainly covers the Control Creation Edition, although it will be equally useful to owners of the full version of Visual Basic. It is aimed at experienced VB developers.

After an introductory section, the main part of the book takes you, blow-by-blow, through developing several example controls, including an aggregate control, a data-bound control, and a user-drawn control. An aggregate control is one that contains several other controls, while the term "user-drawn" describes a control whose visual display is handled entirely by the program.

Despite the book's multiple authorship, the style is clear and consistent. Creating ActiveX controls presents many new issues for VB developers and this is a helpful and detailed guide. It would be better still if more space were given to the design issues behind component programming as opposed to just the mechanics of how to do it. Other vital topics are version control and web security, neither of which are given sufficient coverage. This does not detract from the high quality of the subject matter included: recommended.

● *The above books are available from Computer Manuals on 0121 706 6000.*

Fig 1: A problem with brackets

```
var
ExcelObj: variant;
RangeObj: variant;

begin

ExcelObj := CreateOleObject('Excel.Application');
ExcelObj.visible := True;
ExcelObj.Workbooks.Open('C:\TEST.XLS');
RangeObj := ExcelObj.Range('B2:B4');
...
```

Basic first used in Internet Explorer.

Outlook is a handy contact manager but its database is a simple flat-file affair which is not suitable as the main data store for a business. The key then is to integrate Outlook with other data sources. Since VB Script has neither built-in database features nor the GetObject or CreateObject functions needed for programming COM objects, it does not, at first, seem promising.

By a roundabout route, though, it does have those functions. Outlook's Application object has a CreateObject method that opens the door to Data Access Objects as well as the automation of applications like Excel and Word. This means you can keep a key field in each contact record that matches the key field in an external database (which might be a desktop database like Access) or an SQL server, and link to this external data as needed (*see "An Outlook example, page 300*). Then you can use Outlook for its address book and calendar features but keep the mission-critical data in a fully-fledged database management system where it belongs.

Embark on Outlook development and you hit the bleeding edge of Office 97. Documentation is scant, some procedures are hopelessly counter-intuitive, the script editor is primitive, and late binding of COM objects means performance is poor compared to real Visual Basic. It is such a

p300 ➢

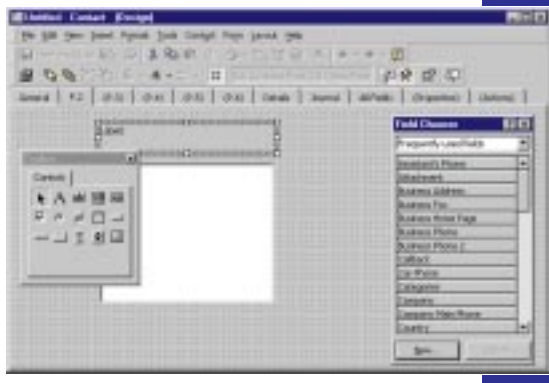


Fig 2 Create a new contact and then choose Design Outlook Form to open the form designer

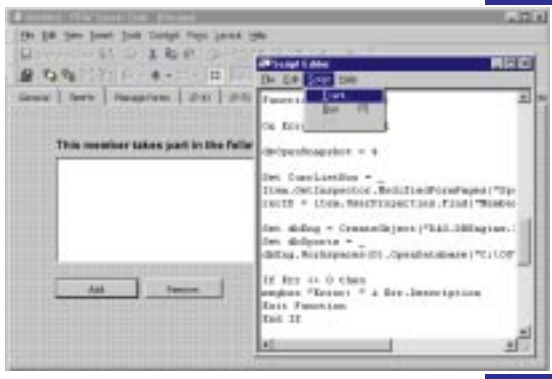


Fig 3 Choose View Code to open the Script Editor. The cunningly-titled Run option does not run the code but merely checks the syntax

Fig 4 When the form opens, Outlook looks up the list of sports from an external database

Creating and saving Outlook forms

Outlook form-handling is perverse. For a start, you cannot just get on and design a new form as you would in Visual Basic. The best way to design a form is to pretend you want to create a new item in the current folder. That opens a blank, default form. Then you can choose Design Outlook Form from the Tools menu. Note that some forms, like the Contacts form, cannot be completely customised. Some parts are read-only. There are plenty of spare tabs, though, which you can use as you want.

A key step is to specify the default form for posting in this dialog

It is when you have designed your form and wrestled with the wretched script editor that the real fun begins. It is no use just saving the form, since all that does is to create one new record with a customised form. You must choose Publish Form As… from the File menu. You can publish to several locations, depending on how widely the form should be available. If you do not have Exchange server, or are working on a test form, the best choice is either your personal forms library or in the specific folder where it is needed. If you choose the latter option, it gets its own entry on the Compose menu.

Next, you have to tell Outlook to use your new form as the default for new items in this folder. To do this, right-click the folder name in the folder list and choose Properties. On the General tab is an option: "When posting to this folder, use…"; here you can specify the new custom form. Now your custom form is used by default for new entries. But it is not over yet. If you try opening any existing items in the folder, you find the old form still being used. The solution is to run a short VBScript routine. Each item in Outlook has a MessageClass property, a concept alien to VB developers but familiar to experts in Microsoft's Mail API, or MAPI. This property determines the form used to view the item.

This is the code to change it:

```
sub UpdateClass

Set currFolder = Application.ActiveExplorer.CurrentFolder
numItems = currFolder.Items.Count

For countvar = 1 to numItems
Set currItem = currFolder.Items(CInt(countvar))
currItem.MessageClass = "IPM.Contact.PCW Sports Club"
currItem.Save
Next

end sub
```

Fig 5: Accessing additional data from within Outlook

```
Function Item_Open()
On Error Resume Next

dbOpenSnapshot = 4

Set CurrListBox = _
Item.GetInspector.ModifiedFormPages("Sports").LstSports
recID = item.UserProperties.Find("MemberID").Value
Set dbEng = CreateObject("DAO.DBEngine.35")
Set dbSports = _
dbEng.Workspaces(0).OpenDatabase("C:\OUTAPPS\SPORTS.MDB")

If Err <> 0 then
msgbox "Error: " & Err.Description
Exit Function
End If

sql = "Select * from sports, sportlink where sports.ID = sportlink.sportID "
sql = sql & "and sportlink.MemberID = " & recID

Set snSports = dbSports.OpenRecordset(sql, dbOpenSnapshot)

If not (snSports.eof and snSports.bof) then
snSports.movelast
snCount = snSports.Recordcount

snSports.MoveFirst

For countvar = 1 to snCount

currListBox.Additem(snSports.Fields("SPORT"))
snSports.MoveNext

Next

End if

snSports.Close
dbSports.Close

End Function
```

nice component in other ways, though, that it is worth persevering. Expect it to get easier in the next version.

An Outlook example

The following example uses the same Sports Club database as the recent *Hands On Visual Basic Workshop* (*PCW* February-May issues).

It is a relational database with three tables. The Members table includes name and address details, and is easily imported into a new Outlook contact folder. When mapping the fields, it is important to include the key field which uniquely identifies each record. Outlook will not let you map an imported field directly to a custom field, so the solution is use a spare built-in field. For a tidy result, you can then add a custom field to hold the ID number and write some code to copy it across. In the example, this field is called MemberID.

Note that not all the data is imported, but only the Members table. In particular, the information about which sports each member enjoys is not available. The trick now is to access this additional data from within Outlook, without actually importing a copy of the tables.

Here's how to do it (also, see Figs 2-4, page 299):

1. Open the contacts folder and choose New Contact from the Contact menu. This opens the built-in Contacts form.
2. From the Tools menu choose Design Outlook Form. This opens the form in design mode with six spare tabs available. Click the second tab, rename it Sports, and add a list box control. Name the list box lstSports. You can add labels and other decoration as desired.
3. On the Form menu, choose View Code. This opens the Script editor. From the Script menu, choose Event and then add the Open event. Fig 5 is the code. Note that you should replace the database filename with the actual location of the data on your system if you use this example. Note also that you can use the Run option in the Script editor to find syntax errors before you save the code. The Run option does not actually run the code as that would be far too easy. Since the script editor has no syntax highlighting and VB Script has no debugger, it can pay to enter and check chunks of code in Visual Basic or Visual Basic for Applications, beforehand.
4. Publish the form and make it the view form for contacts in this folder. This step is so far from being intuitive that I have devoted a separate panel to it (*page 299*).

Now, when you open a contact in this folder, Outlook looks up the list of sports from the original database. Of course, a real-world system could look up a far greater range of information. The important thing is to realise that this sort of link is possible.

PCW Contacts

Tim Anderson welcomes your Visual Programming tips and queries. He can be contacted at the usual *PCW* address or at visual@pcw.co.uk.

The microsoft.public.outlook97 newsgroup is a valuable source of help for Outlook development, as is www.Microsoft.com. Another useful site is www.Outlook.Useast.com.