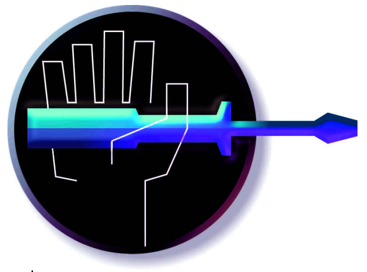


# First class letters

With VB you can use Word to create letters that virtually write themselves. Tim Anderson shows you how. Plus, how to delegate in Visual Basic to achieve the benefits of inheritance.

The "*PCW* Sports Club" is continually sending letters, reminders of coming events, subscription invoices, sympathy for broken bones and the like. The secretary has been running the Visual Basic application to look up the address and then using Alt-Tab to switch back and forth from Word while she copies it across. It is time to make her life a bit easier.

The first thought was to use VB's Clipboard object to copy addresses. This is easy: just add a CopyAddress method to the CPerson class, as in Listing 1.

But Windows can do better than that. It is possible to automate far more of the process of getting addresses into Word. Word has a mail-merge wizard that works fine for bulk mailings, but for *ad hoc* letters a custom solution is needed.

Here, I will show you how to create a Word letter wizard for the sports club (see screenshots, Figs1-4). The wizard is for Word 97, since earlier versions do not support Visual Basic. (As an aside, it is possible to do something similar in earlier versions, using the WODBC.WLL Word add-in and getting at data through ODBC. Another possibility is to automate the WordBasic object from a VB application. But Word 97 makes it easier.)

The plan is to create a Word macro using Visual Basic for Applications, accessing the same SPORTS.MDB database.

Because this tutorial is based on VB 4.0, you cannot import the form in Word. The good news, though, is that the CPerson class module can be reused, as is. The procedure is as follows:

1. In Word, open the Visual Basic editor. Choose Tools, References, and check the Microsoft DAO 3.0 (or higher) object library.

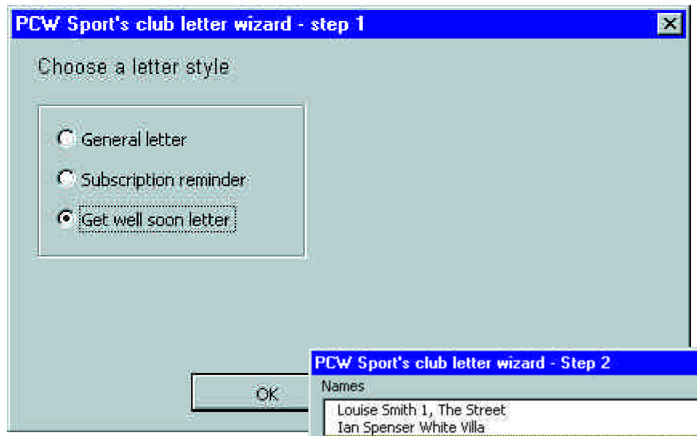
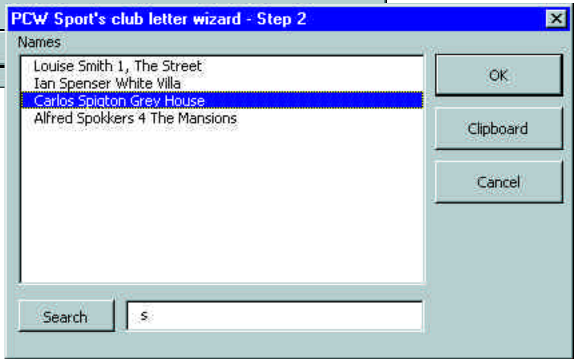


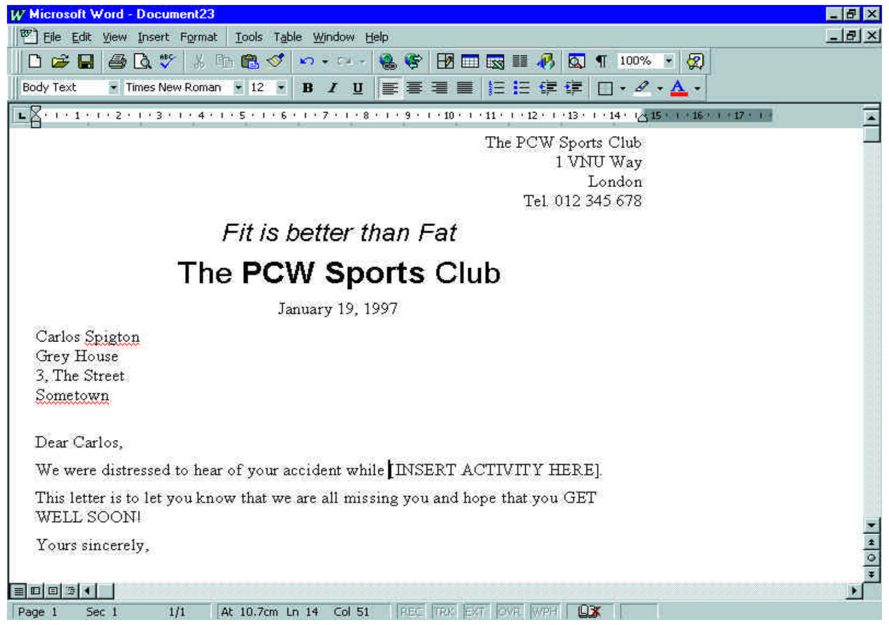
Fig 1 (left) The letter wizard begins by asking which template is needed. You can easily add further options



Louise Smith 1, The Street
Ian Spenser White Villa
Carlos Spigton Grey House
Alfred Spokkers 4 The Mansions

Fig 2 (right) The next stage is to choose a name to write to

Fig 3 (below) In the third step, the wizard has placed the address details into a new document based on the chosen template


Microsoft Word - Document23
File Edit View Insert Format Tools Table Window Help
Body Text
Times New Roman
12
100%
Page 1 Sec 1 1/1 At 10.7cm Ln 14 Col 51

The PCW Sports Club
1 VNU Way
London
Tel. 012 345 678

*Fit is better than Fat*

The **PCW Sports** Club

January 19, 1997

Carlos Spigton
Grey House
3, The Street
Sometown

Dear Carlos,

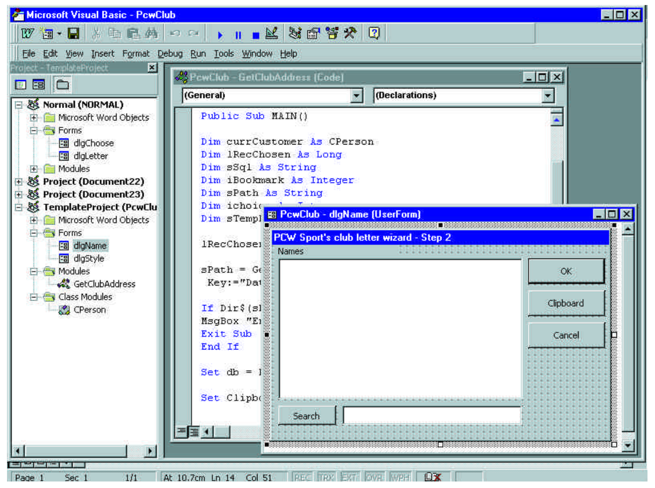
We were distressed to hear of your accident while [INSERT ACTIVITY HERE].

This letter is to let you know that we are all missing you and hope that you GET WELL SOON!

Yours sincerely,

**Listing 1**

```
Sub CopyAddress( )
' copies address to Windows clipboard
Dim sAddress As String
Dim cr As String * 2 ' fixed-length

cr = Chr$(13) & Chr$(10)

If mForename <> "" Then
sAddress = mForename & " " & mSurname & cr
Else
sAddress = mSurname & cr
End If

If mAddress1 <> "" Then
sAddress = sAddress & mAddress1 & cr
End If

...

Clipboard.SetText sAddress, vbCFText
End Sub
```



```
Public Sub MAIN()

Dim currCustomer As CPerson
Dim lRecChosen As Long
Dim sSql As String
Dim iBookmark As Integer
Dim sPath As String
Dim ichoi
Dim sTempl

lRecChosen

sPath = Ge
 Key:="Dat

If Dir$(sl
MsgBox "Er
Exit Sub
End If

Set db = I

Set Clipbc
```

Fig 4 Editing the VBA macro from Word is very like working with standalone VB

This enables Word to use the same data access objects as VB 4.0.

2. Insert a new module into the Normal project. This means the macro will be stored in NORMAL.DOT. Call the macro GetClubAddress and give it a Sub Main.

3. Insert two new userforms. These will be steps one and two of the letter wizard.

4. Name the first userform dlgStyle, and put two or more option buttons on it, along with OK and Cancel buttons. Give the form a Choice property using a memory variable and property procedures. This is for choosing a letter template.

5. Name the second userform dlgName, and put on it a listbox, an editbox and three command buttons. This is for choosing a name for the letter.

6. In the Declarations section of GetClubAddress, declare a public database object. For the example code, I have also declared some convenient constants. Then in Sub Main, open the SPORTS.MDB database using code like:

```
Set db = DAO.OpenDatabase(sPath &
 "\SPORTS.MDB")
```

sPath is a variable to store the path to the database file. (See below for how to get this path from the system registry.) Sub Main also creates a new CPerson object.

The Main procedure continues by opening dlgStyle to obtain a choice of template, and then dlgName to get the ID of name in the Members database. At each point, the user has an option to cancel. The code for the dlgName dialog is almost the same as that used in the main VB 4.0 application, the main difference being that VBA has no data control so you have to create a recordset in code. When a member ID has been retrieved, the record is loaded into the CPerson object.

7. The final step is to start a new document based on the chosen template. The templates must be pre-designed with bookmarks where the name and address information is needed. The wizard finishes by inserting the fields in the bookmark positions and then exits. Controlling Word from VB in this way is not difficult using Word's new object model. For example:

p258 ➢

## Delegating your inheritance

■ A common criticism of Visual Basic is that it doesn't support inheritance. If all your programming has been done in Visual Basic, which is probably true of the majority of VB programmers, this may not mean much to you. Fortunately, it's easy to explain. A class, both in VB and other object-orientated languages, defines an object. In VB, every class starts from scratch without any properties or methods. By contrast, C++, as an example, lets you begin a class definition like this:

```
class monkey : public animal
```

The result is that the monkey class inherits the properties and methods of the animal class. The monkey class just needs to add specialised code that describes monkeys; the generic animal code comes for free.

Although VB does not support inheritance, there are other ways of achieving some of the benefits. It is possible to contain one class within another. Then you can implement properties and methods of the parent class by calling the properties and methods of the contained class. This is called delegation, and the properties and methods of a class are called its interface. For example, the tutorial application has a CPerson class. Imagine you wanted to create a CEmployee class which used the properties and methods of CPerson. Here is how you can do it:

1. Insert a new class module and set its name property to CEmployee.
2. In the declarations section, put:

```
Private m_person As CPerson
Private m_wage As Currency
```

3. In the initialise section put:

```
Set m_person = New CPerson
```

4. Create a CEmployee interface that calls the CPerson interface. For example:

```
Public Property Get surname() As
 String
surname = m_person.surname
End Property
```

5. Add new properties and methods specific to CEmployee. For instance, you must expose the wage property.

The fourth step (*above*) is tedious, but beats re-coding all the functionality of CEmployee in CPerson. It could be automated by a VB Wizard. In Visual Basic 5.0 this approach to object-orientation is built into the language, with a new Implements keyword which guarantees that all the methods of the contained class are implemented by the outer class. You can implement the interface of any ActiveX automation server. Finally, there is nothing to stop you implementing several interfaces in a single class.

Delegation works, but it is neither as intuitive nor as elegant as traditional inheritance. For the moment, though, this is the VB way. It ties in with ActiveX, the component model which is becoming more powerful and pervasive as Windows evolves. VB may not be the fastest or most thoroughly object-orientated language out there, but Microsoft does ensure that it stays up to date with the latest ActiveX developments.

```
Documents.Add (sTempLate) ' starts
a new document based on the given
template

ActiveDocument.Bookmarks("name").
Select ' sets cursor to the "name"
bookmark in the new document

Selection.InsertAfter Trim
(currCustomer.forename & " " _
& currCustomer.surname) ' inserts
text at the cursor position
```

### Problem-solving

There are a few things to notice about this joint Visual Basic and Word project. Although Word VBA is downward compatible with VB 4.0, there are some objects which are available in VB but not VBA. One example is the global App object which, in Word, is the Application object.

The original CPerson class used App.Path to discover the location of SPORTS.MDB. This strategy fails in any case, when the code runs in other applications. A better idea is to use a registry entry, using VB's GetSetting command. The registry entry is created by the main VB 4.0 application when it first runs. This way, the data can easily be found by any Windows application.

Another catch is that VBA has no Clipboard object, so CPerson's CopyAddress method does not compile in Word. The workaround is to declare a public Clipboard variable as a DataObject: VBA's private version of the clipboard. To demonstrate, there is a Clipboard button on the dlgNames form which uses the DataObject's PutInClipboard method to transfer text to the read clipboard.

### Enhancing the wizard

There are plenty of ways you can improve on the Letter Wizard. For instance, you can add database fields for things like job title and salutation. You could increase the range of templates on offer. For the subscription template, you could write code to check a person's outstanding balance and insert the amount into the letter. By adding the bulk of the code to a shared class module like CPerson, you can easily reuse it in VB 4.0 or in other VBA applications such as Excel.

## Installing the example code from the *PCW* CD

When you unpack the tutorial code from our cover-mounted CD, you will find a VB 4.0 project and a Word 97 template. To install the example code, copy PCWCLUB.DOT into your Word templates directory. Then start a new document based on this template. If you then choose Tools, Macro, Visual Basic Editor, you will find the example macros. Choose Tools, Macro, Run, to run the macro. You can also copy the macros into NORMAL.DOT if you want, by using Tools, Templates and Add-ins, Organizer. Finally, the macro will not run without a registry setting for the data path. To create this setting, run PCWCLUB.EXE.

■ *Next month: Back to native Visual Basic for the final stage in the PCW Sports Club application.*

### PCW Contact

Tim Anderson welcomes your comments and queries. Write to the usual *PCW* address, or email pcw@vnu.co.uk.