Customize Outlook to Fit Your Needs

by Chris Barlow

Build an app to access and manage data stored in Outlook.

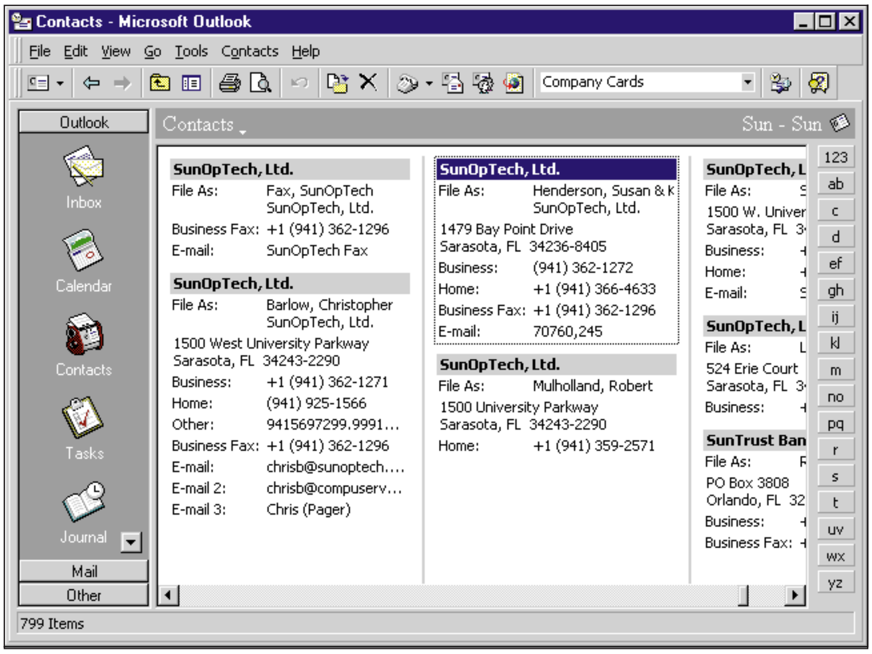
Click & Retrieve Source CODE!

A good way to get started with Visual Basic is to write a simple but useful utility to help you be more productive. If you're interested in learning to program with Visual Basic, you probably already use Microsoft Office 97. It features a word processor, spreadsheet, database, and Personal Information Manager (PIM), all of which you can program with Visual Basic. Office gives you a library of more than 500 programmable objects that you can use to quickly empower your applications. Because so many corporate desktops have standardized with Office, there's a growing need for custom applications to manage the information stored in Office documents.

Outlook, Office's PIM, combines e-mail with a neat task and contact manager. As you can do with the other Office apps, you can easily program Outlook with VB. If you haven't read Tom Campbell's Using VBA column, "Create Outlook Apps in a Snap," in this issue, read it now. Tom gives an excellent overview of how to work with Outlook.

Welcome back . . . now let's write a utility that works with Outlook. When I first bought Office, I fired up Outlook and imported names and addresses from my personal address book, contact manager, and various spreadsheet files. I couldn't wait to begin using a PIM that was integrated with my e-mail. Unfortunately, because the various applications store names differently, some of my contacts displayed the last name first while others were filed under first names. Because I had 823 contacts, cleaning up the data manually would take a while—the perfect opportunity to write a small Visual Basic utility to automate the process.

This utility allows you to scroll through your Contacts folder (see Figure 1) from a Visual Basic form and display your contact's first name, last name, full name, company,



SunOpTech, Ltd.
File As: Fax, SunOpTech
SunOpTech, Ltd.
Business Fax: +1 (941) 362-1296
E-mail: SunOpTech Fax

SunOpTech, Ltd.
File As: Barlow, Christopher
SunOpTech, Ltd.
1500 West University Parkway
Sarasota, FL 34243-2290
Business: +1 (941) 362-1271
Home: (941) 925-1566
Other: 9415697299.9991...
Business Fax: +1 (941) 362-1296
E-mail: chrisb@sunoptech....
E-mail 2: chrisb@compuserv...
E-mail 3: Chris (Pager)

SunOpTech, Ltd.
File As: Henderson, Susan & K
SunOpTech, Ltd.
1479 Bay Point Drive
Sarasota, FL 34236-8405
Business: (941) 362-1272
Home: +1 (941) 366-4633
Business Fax: +1 (941) 362-1296
E-mail: 70760,245

SunOpTech, Ltd.
File As: Mulholland, Robert
1500 University Parkway
Sarasota, FL 34243-2290
Home: +1 (941) 359-2571

SunOpTech, L
File As: S
1500 W. Univer
Sarasota, FL 3
Business: +
Home: +
E-mail: S

SunOpTech, L
File As: L
524 Erie Court
Sarasota, FL 3
Business: +

SunTrust Ban
File As: F
PO Box 3808
Orlando, FL 32
Business: +
Business Fax: +

FIGURE 1 ***A Look at Outlook.*** *The Contacts folder of Outlook can have many customizable views. This Address Card view displays contacts in order by their name with the frequently used fields.*

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications, including the ObjectBank, ObjectOrder, and ObjectJob Systems. He holds two U.S. patents related to software for decentralized distributed asynchronous object-oriented and scheduling systems. Chris is a frequent speaker at VBITS, Tech•Ed, and DevDays, and has been featured in two Microsoft videos. He holds degrees from Harvard Business School and Dartmouth College, where he worked with Drs. Kemeny and Kurtz on the original Basic language. Reach Chris at ChrisB@SunOpTech.com or through SunOpTech's Web server at www.SunOpTech.com.

and file as fields. You can enter a correction and save your changes or even automate some of the functions to format the names just the way you like.

Launch Visual Basic 5 and select Standard EXE. Draw a TextBox control on the form and place a Label control above it. Select both controls, copy them to the clipboard, and paste them onto the form four more times as control arrays. You will use these controls to display the fields from the Contacts folder. Control arrays are a convenient way to work with input forms because they share the same event procedures. You can easily write code to respond to changes in these controls within a single procedure.

Add a vertical scrollbar control on the right side of the form and a CommandButton control at the bottom. Change the Caption property of the button to Save, and change its name to butSave.

Let VB know you intend to use Outlook objects by selecting the References menu item from the Project menu and selecting the Microsoft Office 8.0 Object Library. Programmers often neglect to reference the objects they intend to use. If you reference them correctly, VB will load the type library and display the properties and methods of all Outlook objects in the Object Browser and in its IntelliSense help popups. Otherwise, VB simply reports a syntax error when you use one of these objects.

SCROLL THROUGH CONTACT FOLDERS

First, set up variables you can use from any event in the form to refer to the folder:

```
Dim ol As New Outlook.Application
Dim ContactFolder As MAPIFolder
Dim IsDirty As Boolean
```

The first line defines the variable ol as the top-level Application object of Outlook. From the Application object, you can refer to any other object within Outlook. The keyword New opens Outlook and creates an instance of Outlook's Application object the first time you use ol. If you leave out the New keyword, you need to use the Set statement with the CreateObject statement to create an instance of the Application object, as Tom Campbell shows in his sample code. The ContactFolder variable is a MAPIFolder object; you use the IsDirty variable to recognize when the user changes a field on the form.

Although this code will launch Outlook if it's not already running, launch Outlook from the Windows Desktop to speed debugging. This way you won't have to wait for Outlook to start up every time you test your new app.

Now double-click on the form to add some code in the Form_Load event to establish a connection with the Contacts folder within Outlook. Use the GetNameSpace method of the Application object to connect to your standard MAPI profile. When you pass the constant olFolderContacts as an argument to the GetDefaultFolder method, you open the Contacts folder. Then you can set the Max property of the vertical scrollbar control to the number of items (contacts) in the Contacts folder. Finally, set the Value property of the vertical scrollbar control to 1 to display the first contact:

```
Private Sub Form_Load()
Set ContactFolder = ol.GetNamespace_
   ("MAPI").GetDefaultFolder _
   (olFolderContacts)
VScroll1.Max = ContactFolder.Items.Count
VScroll1.Value = 1
End Sub
```

The user navigates between contacts by clicking on the scrollbar. Doing this changes the value of the vertical scrollbar control. The user can even move the thumb of the control to skip through the contacts. This value will always fall between 1 and the maximum number of contacts so you can use it as a direct reference to the collection of contact items in the Contacts folder. In the scrollbar's Change event, add code to call a LoadContact procedure passing the current value of the scrollbar. It's always better to separate these loading and saving procedures from your form's event procedures so you can more easily call them from another procedure in your application:

```
Private Sub VScroll1_Change()
LoadContact VScroll1.Value
End Sub
```

The LoadContacts procedure is simple. You can make it convenient for the user if you display the position of the current contact in the Caption property of the form. This feedback helps the user as he or she scrolls through the contacts. You can use the Items collection of the Contacts folder in a With statement to set the Text property of each TextBox

VB5

```
Function ParseName(s$, n%) As String
Dim Ret$
  If InStr(s$, ",") Then
  'has comma so assume Last,First
     If n% = 1 Then 'first name
        Ret$ = ParseString(s$, ",", 2)
     Else
        Ret$ = ParseString(s$, ",", 1)
     End If
  Else  'assume First Last with space delimited
     If n% = 1 Then 'first name
        Ret$ = ParseString(s$, " ", 1)
     Else
        Ret$ = ParseString(s$, " ", 2)
     End If
  End If
  If Ret$ = "" Then Ret$ = s$ 'if blank set to original
  ParseName = Ret$
End Function

Function ParseString(s$, del$, n) As String
'parse a string by delimiter and return nth value
'CRB 3/27/96 add Trim result
  Dim pos As Long, i As Integer, pos2 As Long
  ParseString = s$
  pos = InStr(s$, del$)
  If pos Then     'if has del$
     If n = 1 Then
        ParseString = Left$(s$, pos - 1)
     Else
        For i = 1 To n - 1      'count items
           pos2 = InStr(pos + 1, s$, del$)
           If pos2 = 0 Then  'end of string
              If i = n - 1 Then
                 ParseString = Trim(Mid$(s$, pos + _
                    Len(del$)))   'len of delimiter
              Else
                 ParseString = ""  'nth item not found
              End If
              Exit Function
           End If
           ParseString = Trim(Mid$_
              (s$, pos + Len(del$), pos2 - pos - Len(del$)))
           pos = pos2
        Next
     End If
  ElseIf n > 1 Then
     ParseString = ""  'nth item not found
  End If
End Function
```

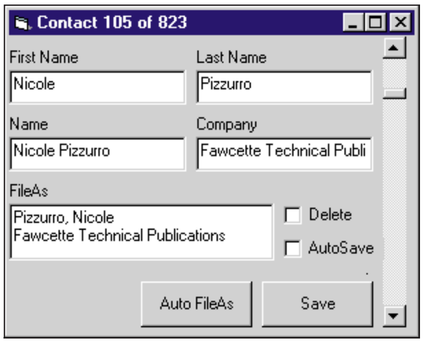
LISTING 1 ***Parse a Name Field.*** *You can use these procedures to parse a name field and return either the first name or last name. Once you have accurate FirstName and LastName fields, you can fill in the FullName and FileAs fields.*

control to one of the fields in this contact item:

```
Private Sub LoadContact(i%)
Me.Caption = 'Contact " & i & " of " & _
   VScroll1.Max
With ContactFolder.Items(i)
   Text1(0) = .FirstName
   Text1(1) = .LastName
   Text1(2) = .FullName
   Text1(3) = .CompanyName
   Text1(4) = .FileAs
End With
IsDirty = False
End Sub
```

I chose to use these five fields because they didn't import correctly from my other contact applications. You can easily change this code to use other fields or add more TextBox controls to your form for additional fields.

Now your application should be able to scroll through your contacts. Press F5 to run your application, and your first contact should appear. Use the scrollbar to move back and forth through your contacts.


Contact 105 of 823

First Name: Nicole
Last Name: Pizzurro
Name: Nicole Pizzurro
Company: Fawcette Technical Publi
FileAs: Pizzurro, Nicole
Fawcette Technical Publications
☐ Delete
☐ AutoSave

FIGURE 2 ***Enhance Your Form.*** *You can make improvements to your form, such as allowing it to automate more of the conversion functions and to flag a record for deletion.*

SAVE YOUR CHANGES

Now your application lets you view your contacts—but you really want to be able to make changes and save them. First add code to the TextBox control array's Change event to set the IsDirty flag so you will know if the user changes something. Here's the power of the control array—one line of code sets the flag for a change to any of the fields:

```
Private Sub Text1_Change(Index As Integer)
IsDirty = True
End Sub
```

Double-click on the Save button and add this code to call your SaveContact procedure:

```
Private Sub butSave_Click()
SaveContact VScroll1.Value
End Sub
```

Copy the code from your LoadContact procedure and change it to set the field properties of the contact item within the Items collection of the ContactFolder. After you set the fields, use the Save method to write the updated fields to the Contacts folder. Don't forget to set the IsDirty variable to False when the save is complete:

```
Private Sub SaveContact(i%)
With ContactFolder.Items(i)
   .FirstName = Text1(0)
   .LastName = Text1(1)
   .FullName = Text1(2)
   .CompanyName = Text1(3)
   .FileAs = Text1(4)
   .Save
End With
IsDirty = False
End Sub
```

Now run your application, change some contact information, and click on the Save button. Scroll back and forth to refresh the record from the Contacts folder and you should see the changed information.

So far the application works well to update a single contact—but you don't want to scroll through hundreds of contacts and type in corrections for each. In my Contacts folder, I found the FullName field was completed but, for many of the contacts, the FirstName field contained both the FirstName and the LastName while the LastName was blank. I wanted to parse the FullName field and fill in the FirstName and LastName fields. I also wanted to fill in the FileAs field as *LastName, FirstName* and append the Company field if it was present.

Add another CommandButton control to your form and set the Caption property to Auto FileAs and the Name property to butAuto. Double-click on this button and enter this code to use Visual Basic's Instr function to see if the FirstName contains a space and the LastName is blank:

```
Private Sub butAuto_Click()
If InStr(Text1(0), " ") _
   And Text1(1) = "" Then
   Text1(0) = ParseName(Text1(0), 1)
   Text1(1) = ParseName(Text1(0), 2)
End If
```

Call the ParseName function to fill in the FirstName and LastName fields (see Listing 1). Now that you have accurate FirstName and LastName fields, you can fill in the FullName and FileAs fields:

```
Text1(2) = Text1(0) & " " & Text1(1)
Text1(4) = Text1(1) & ", " & Text1(0)
If Len(Text1(3)) Then Text1(4) = _
   Text1(4) & vbCrLf & Text1(3)
End Sub
```

Now when you see a contact that looks wrong, you can click on the Auto FileAs button to automatically adjust the fields and then click on the Save button.

With fewer than 50 lines of code, you can write a utility to access data within an Office application. Now that you have a working application, I'm sure you'll think of a number of other useful functions (see Figure 2). Premier Club members of The Development Exchange can download this more complete implementation (see the Code Online box for details). ☒

Code Online

You can find all the code published in this issue of VBPJ *on The Development Exchange (DevX) at http://www.windx.com. All the listings and associated files essential to the articles are available for free to Registered members of DevX, in one ZIP file. This ZIP file is also posted in the Magazine Library of the* VBPJ *Forum on CompuServe. DevX Premier Club members ($20 for six months) can get each article's listings in a separate file, as well as additional code and utilities for selected articles, plus archives of all code ever published in* VBPJ *and* Microsoft Interactive Developer *magazines.*

Customize Outlook to Fit Your Needs

Locator+ Codes

Listings ZIP file (free Registered Level): VBPJ0697

✪ *Listings for this article plus an enhanced utility that automates more of the conversion functions and flags a record for deletion (subscriber Premier Level): GS0697P*