by Christine Solomon

MANAGE AND MERGE EXCEL DATA

COUPLE EXCEL AND WORD WITH OLE TO MAKE A USER-FRIENDLY DATABASE DUO.

Microsoft estimates that more than 70 percent of people using Excel use it as a database—and why not? Anyone who can use a spreadsheet can organize data into a simple but effective flat-file format. There's no need for relational database theory and no need to figure out how to construct a query that returns the desired data. Excel 5's List feature treats any contiguous range as a database, making data management especially easy.

I've created an application called BulkMail that demonstrates how to build an Excel database application for managing mailing lists. It is structured so that end users can enter, edit, and delete data through a custom Excel dialog box. Another Excel dialog box is used to select form letter recipients. The mail merge is performed in Word through OLE Automation. Although this version of BulkMail focuses on performing the mail merge with a letter already written in Word, you could expand the application to enable users to control Word through OLE Automation in order to write the letter as well.

BulkMail has three components: BULKMAIL.XLS, an Excel database and mail merge interface that serves as the main application controlling Word; BULKLTTR.DOT, a Word template for creating the form letter; and HEADER.DOC, a Word document that stores the database fields used by the template. You can download these files from the Magazine Library of the *Visual Basic Programmer's Journal* Forum on CompuServe (GO VBPJFO), and find them on the third edition of the VB-CD Quarterly as well.

The first component of my BulkMail sample application, BULKMAIL.XLS, contains the Excel database that stores standard address information for contacts, and a custom menu with two commands: Update Data and Mail Merge. The Update Data command is a placeholder for a dialog box (not included in this sample application) where a user can add records, edit existing records, and delete unwanted records. Whenever you deal with applications for *any* database (from Excel to Access to Oracle), you should always have an interface that hides the actual data from users, and strictly controls the type and format of data entered into the database. This prevents mistakes from corrupting the data.

The Mail Merge command on the BulkMail menu opens a dialog box where users select the recipients of the mailing. Mail Merge then uses OLE Automation to perform the mail merge in Word. BulkMail has four main routines, which are stored in the module modMailMerge: ListDocFiles, cmdChange_Click, ListDistinctEntries, and cmdOK_Click.

The ListDocFiles subroutine uses the Files function to list the Word DOC files in the default directory. The cmdChange_Click subroutine lets users select a mail merge letter in a directory other than the default directory. The ListDistinctEntries subroutine allows users to select the recipients of the mail merge letters. The cmdOK_Click subroutine performs the query that users specified regarding letter recipients, and then performs the mail merge with Word.

To perform the mail merge itself, users write a letter using the BULKLTTR.DOT template. This template includes standard merge fields for letters and uses HEADER.DOC, a table with the column headings from the BulkMail database, as the data source for these fields. Using BULKLTTR.DOT, users can write new text without bothering with fields.

```
Sub ListDocFiles(ByVal strDirectory As String)
      Dim strFileList() As String
      Dim iNumFiles As Integer
      Dim i As Integer

'Clear the drop-down list that lists documents.
   objDocList.RemoveAllItems

'Call the Files function (in modMain) to get the names
'of Word document files in strDirectory.
   iNumFiles = Files(strDirectory & "*.doc", _
      strFileList())
   If iNumFiles > 0 Then

'Fill the document list box with the file names
'retrieved by the Files function.
      For i = 1 To iNumFiles
         objDocList.AddItem strFileList(i)
      Next i
   End If
End Sub
```

LISTING 1 ***What's Up, DOC?*** *The ListDocFiles subroutine retrieves file names that meet a certain criteria (such as *.DOC), along with the number of qualifying files returned by the Files function. The file names then populate the document list box so they can be chosen by the user.*

Christine Solomon has been using Microsoft Office since 1989 when she developed her first Excel application. She is a director of Micro Modeling Associates, Inc., a Windows consulting firm serving Fortune 1000 companies. She is also the author of several books, including the one from which this information is excerpted, Developing Applicatons with Microsoft Office *(published by Microsoft Press). Reach Christine by fax at 212-233-9897, on CompuServe at 74720,3446, or on the Internet at csolomon@panix.com.*

Once users write the form letter for the mail merge, they can perform the merge by choosing the Mail Merge command from the BulkMail menu. The application opens a dialog box that lets users select the form letter and its recipients.

When the BulkMail dialog box first opens, the drop-down list displays the DOC files in the default directory. The default directory is stored in the label above the drop-down list. If this label is empty, the default directory is the Word program directory. Users can select from this list or click the Change button to select a DOC file in a different directory.

Most of the code for handling this list is in the ListDocFiles subroutine in modMailMerge. This routine calls the Files function in modMain to retrieve a list of DOC files in the default directory. The Files function uses the Dir$ function to retrieve files that meet a given specification (such as *.DOC), stores their names in a string array provided by the calling procedure (ListDocFiles), and returns the number of file names retrieved (see Listing 1).

```
Sub cmdChange_Click()
   On Error GoTo ChangeError

   Dim strCurDir As String
   Dim strDocDirectory As String
   Dim strFullPath As String
   Dim strPath As String
   Dim strFileName As String

'Store the current directory in strCurDir and make the
'directory displayed on the dialog box the default
directory.
      strCurDir = CurDir()
      strDocDirectory = objDocDirectory.Caption
      ChDir Left$(strDocDirectory, Len _
         (strDocDirectory) - 1)

'Open the File Open dialog box to allow the user to
'select a mail merge document.
      strFullPath = Application.GetOpenFilename _
            (fileFilter:="Word Docs (*.doc),*.doc", _
            Title:="Select Mail Merge Document")

'When the user clicks the OK button...
      If strFullPath <> "False" Then

'...extract the path and file name from the full path
'for the document chosen by the user.
            strPath = Extract(strFullPath, 0)
            strFileName = Extract(strFullPath, 1)

'Write the path in the default directory label on the
'dialog box and call ListDocFiles to fill document
'list box.
            objDocDirectory.Caption = strPath
            ListDocFiles strPath
            objDocList.Text = strFileName
      End If

RestoreDirectory:
'Restore the original directory.
      ChDir strCurDir
      Exit Sub

ChangeError:
'If the directory displayed on the dialog box isn't a
'valid directory, leave the default directory at its
'original setting.
      If Err = 76 Then
            Resume Next
      ElseIf Err <> 0 Then
            GlobalErrorMsg "cmdChange_Click", Err
            Resume RestoreDirectory
      End If
End Sub
```

LISTING 2 ***The Name Game.*** *Instead of simply returning the name of a file with the GetOpenFilename method, the cmdChange_Click subroutine returns the path name as well as the file name. This information is then used to set the default directory to that currently displayed in the dialog box.*

When users click on the Change button, BulkMail calls the cmdChange_Click subroutine in modMailMerge. This routine uses the GetOpenFilename method to open the standard File Open dialog box—with a twist. Rather than opening the selected file, this method returns the file's full path. The selected directory becomes the default directory, and its DOC files are displayed in the drop-down list (see Listing 2).

WHO GETS LETTERS?

In addition to letting users select a form letter, the BulkMail dialog box lets them select the letter's recipients by clicking on one of four option buttons: All, Last Names, Companies, or States.

When users select Last Name, Companies, or States, they can further target the mailing by clicking on particular names, companies, or states. For example, if users select the States option, they can then select California, Texas, and New York from the multiselect list box.

Here's how this works: The Bulk Mail Database worksheet contains the contact data in a range named Database ("Database" is an Excel reserved name you must use to manipulate data as a database through code). When users select an option such as Companies or States, the ListDistinctEntries subroutine uses the AdvancedFilter method to query the database and select the data specified:

```
objDatabase.Range("Database").Columns(iField)._
   AdvancedFilter Action:=xlFilterCopy, _
   CopyToRange:=.Cells(2, 10), Unique:=True
```

```
Sub ListDistinctEntries(iField As Integer)
      On Error GoTo ListError
   Dim objCriteriaRange As Object

'Run an Advanced Filter to get a list of the distinct
'entries in the column specified by the argument iField.
      With objDatabase
            strField = .Cells(1, iField).Value
            .Range("Database").Columns(iField).Advanced_
               Filter Action:=xl_
                  FilterCopy, CopyToRange:=.Cells_
                     (2, 10), Unique:=True
            Set objCriteriaRange = .Cells(2, 10)._
               CurrentRegion
      End With

'Trim off the column heading and sort the list.
      With objCriteriaRange
            .Rows(1).Delete
            .Sort Key1:=.Cells(1, 1)
      End With

'Fill the criteria list box with the extracted data.
      Dim i As Integer
   For i = 1 To objCriteriaRange.Rows.Count
            objCriteriaList.AddItem
objCriteriaRange.Rows(i).Value
      Next i

'Clear the extracted data range.
      objCriteriaRange.ClearContents
      Set objCriteriaRange = Nothing
      Exit Sub

ListError:
      If Err <> 0 Then
            GlobalErrorMsg "ListDistinctEntries", Err
            Exit Sub
      End If
End Sub
```

LISTING 3 ***Setting Your Targets.*** *The ListDistinctEntries routine fills the multiselect list box in the BulkMail dialog box with the selected mail merge names. After users select the form letter and its recipients, the specified records are merged with the Word document.*

The variable objDatabase in this code represents the Database range on the BulkMail Database worksheet: iField is the number of the column that users select (1 is Last Name, 3 is Company, and 6 is State), and strField is the column heading (Last Name, Company, or State). The Unique argument is set to True to return only nonduplicated entries in the selected column. Also, setting the Action argument to xlFilterCopy copies the entries to the location specified by the CopyToRange argument.

The ListDistinctEntries routine then fills the multiselect list box in the BulkMail dialog box with the entries selected (see Listing 3). From here, users can further target recipients. After users select the form letter and its recipients, they click on the OK button to retrieve the records specified and to merge them with the Word document. The code that handles this is in the cmdOK_Click subroutine (see Listing 4).

The cmdOK_Click subroutine uses the AdvancedFilter method to retrieve the specified records. This code is nearly identical to that shown in the Advanced Filter method, except that it requires a CriteriaRange argument, which contains the criteria for selecting records.

BulkMail's criteria range is a single column whose first entry indicates the selected option button—Last Names, Companies, or States. The remaining entries indicate the items that users selected from the list box. The AdvancedFilter method retrieves records that match these criteria and copies them into the CopyToRange.

After selecting the recipients' records, BulkMail uses OLE Automation to perform a mail merge in Word. Although Word normally lets you use an Excel database as the data source for a mail merge, it doesn't let you do this under OLE Automation. Word itself uses Dynamic Data Exchange (DDE) to communicate with Excel during such a mail merge, and you can't communicate with Excel through DDE from inside an OLE Automation operation controlled by Excel.

As a workaround, I've enabled BulkMail to create a temporary Word document (called TMP.DOC), paste the selected records from Excel into this document, perform the mail merge against this document, and then delete the temporary file.

To use BulkMail, follow these steps: First, create the merge letter using BULKLTTR.DOT. This template uses HEADER.DOC as the data source, and already has merge fields. Make sure that both BULKLTTR.DOT and HEADER.DOC are in the Word template directory (if there is a problem with the merge, change the merge field text in BULKLTTR.DOT to conform with that in HEADER.DOC and with the field names in the BulkMail database). Then in Excel, select the Mail Merge command from the BulkMail menu.

In the BulkMail dialog box, select the Word document to use as the form letter. Next, select the criteria for selecting the letter's recipients—All, Last Names, Companies, or States. If you select Last Names, Companies, or States, you should then select the exact records from the list box by clicking on them. Finally, click on OK when you're ready to do the mail merge and let Excel, Word, and OLE Automation do the rest. ■

```
Sub cmdOK_Click()
      On Error GoTo OKError

      Dim i As Integer
      Dim iRow As Integer
      Dim strDocDirectory As String
      Dim strFormLetter As String
      Dim objAddresses As Object

'Read selected names from list box and use these as
'criteria for AdvancedFilter method.
      iRow = 2
      objDatabase.Cells(iRow, 12).Value = strField
      With objCriteriaList
            For i = 1 To .ListCount
               If .Selected(i) Then
                  iRow = iRow + 1
                  objDatabase.Cells(iRow, 12).Value = _
                     .List(i)
               End If
         Next i
      End With

'Run advanced filter.
      With objDatabase
            .Range("Database").AdvancedFilter
Action:=xlFilterCopy, CriteriaRange:=.Cells(2, 12)._
   CurrentRegion, CopyToRange:=.Cells(2, 14), Unique:=False

'Set object variable to extracted data range and trim off
'column headings.
            Set objAddresses = .Cells(2, 14).CurrentRegion
            objAddresses.Rows(1).Delete

'Clear criteria range.
            .Cells(2, 12).CurrentRegion.ClearContents
      End With

'Get path and name of mail merge letters.
      strDocDirectory = objDocDirectory.Caption
      strFormLetter = objDocList.Text

'Copy extracted data to clipboard.
      objAddresses.Copy

      Set objWord = CreateObject("Word.Basic")
      With objWord
            .ScreenUpdating 0

'Create temporary document and paste mail merge data in it.
            .FileNew Template:="Normal"
            .EditPaste
            .FileSaveAs Name:=strDocDirectory _
               & "tmp.doc"

'Open the mail merge letter.
            .FileOpen Name:=strDocDirectory _
               & strFormLetter

'Do mail merge.
            .MailMergeOpenDataSource Name:_
               =strDocDirectory & "tmp.doc"
            .MailMerge CheckErrors:=2, Destination:=0,
MergeRecords:=0, Suppression:=0, MailMerge:=True

'Close the mail merge letter.
            .Activate strFormLetter
            .FileClose 2

'Close the temporary document.
            .Activate "tmp.doc"
            .FileClose 2
            .ScreenUpdating 1
      End With

'Delete the temporary document.
      Kill strDocDirectory & "tmp.doc"

'Clear the range containing the extracted data.
      Application.CutCopyMode = False
      objAddresses.ClearContents
      Set objAddresses = Nothing

CleanUp:
      ClearObjectVariables
      Exit Sub

OKError:
      If Err <> 0 Then
            GlobalErrorMsg "cmdOK_Click", Err
            Resume CleanUp
      End If
End Sub
```

LISTING 4 ***Merging in OLE Requires a Workaround.*** *Normally, using Excel as the source for a Word merge doesn't work under OLE Automation. This workaround copies the records selected for the mail merge into a temporary Word document, runs the merge against this temporary document, and then deletes it.*