This slideshow will show examples of VBA from the Access/VBA application I created for the Minnesota Astronomical Society (MAS). It is a membership management database for the organization's 500+ members. I chose this application as the source of VBA examples to publish because its VBA does not reveal sensitive business information. I also had complete control over standards and conventions, so it shows how I prefer to code. By no means does it represent the typical business processes I automate professionally.

The examples I have chosen require the least amount of explanation of business processes. These consist of code that:

- Tests user-composed SQL, including logic tests on the WHERE clause.
- Centralized error handling.
- Automatic composition of email messages.
- Simulating a mail merge in MS Word from MS Access (the reason for simulating will be explained).

Early binding or late binding?

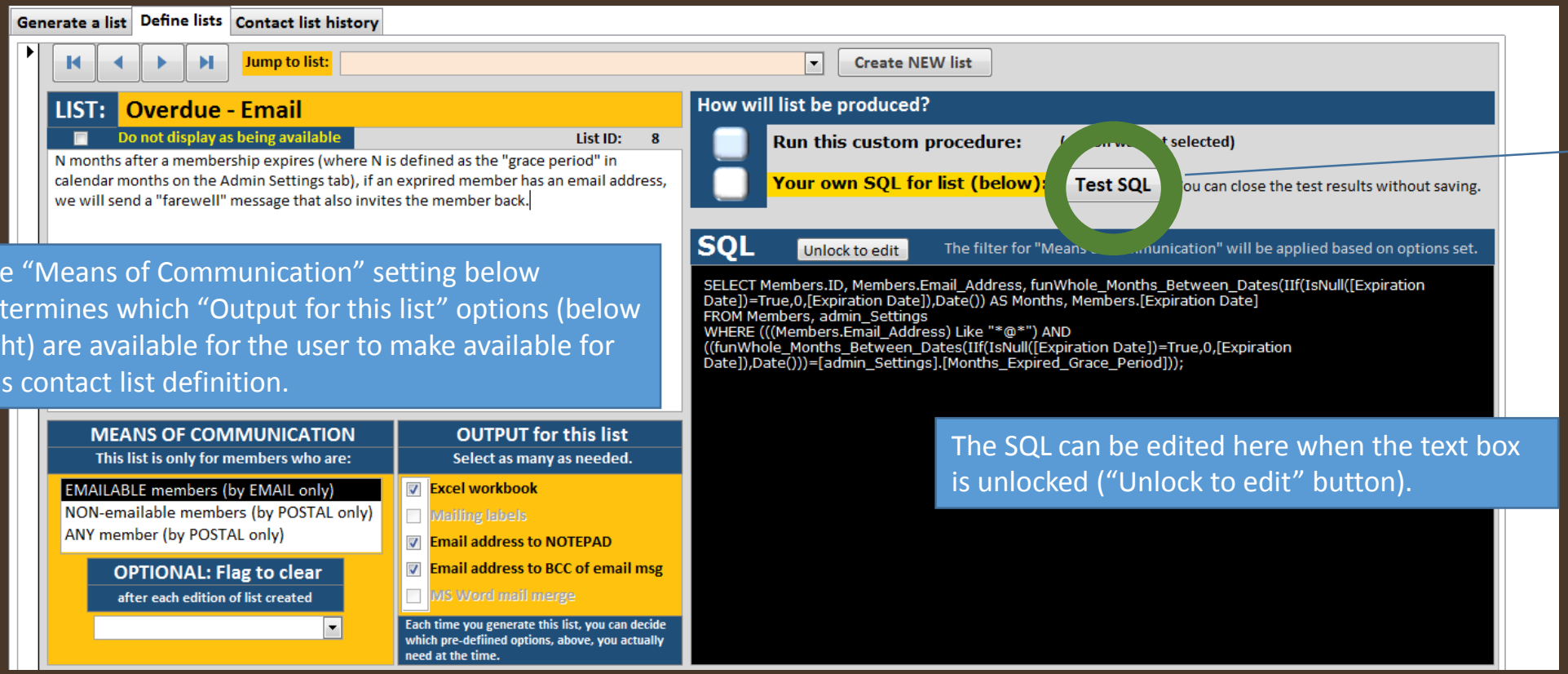
As a rule, like most VBA developers, I use *late* binding for application variables when developing for business clients. It lets the app be used with earlier versions of MS Office. However, for this project I chose to use *early* binding. Why?

- First, I was developing this on my own time when I was already working 50-60 hours per week. Early binding let me use intellisense to save me from time-consuming debugging later.
- Secondly, as Microsoft says "[e]arly binding is the preferred method. It is the best [sic] performer because… there is no extra overhead in doing a run-time lookup. In terms of overall execution speed, it is at least twice as fast as late binding."* Because this app would be used on someone's home PC, I wanted to avoid memory problems and complaints of slowness. The price of usage for any future user was to possibly upgrade MS Office.

* See https://support.microsoft.com/en-us/help/245115/using-early-binding-and-late-binding-in-automation

VBA SAMPLE: TESTING A USER-COMPOSED/EDITED SQL STRING

In the slides that follow, we will look at the VBA behind the "Test SQL" button (see green circle below). Besides testing the SQL for errors by running it, the VBA will also look for logical errors. For example, if the contact list in question is intended for emailable people, does the SQL's WHERE clause pull only those members with an email address.



The "Means of Communication" setting below determines which "Output for this list" options (below right) are available for the user to make available for this contact list definition.

The SQL can be edited here when the text box is unlocked ("Unlock to edit" button).

The user can test changes to their SQL by clicking this button, which displays a form showing the results.

```
Private Sub cmdTest_SQL_Click()
    '---------------------------------------------------------------------------------------
    ' Procedure : cmdTest_SQL_Click
    ' Author    : Will Beauchemin (Himself@WillBeauchemin.com)
    ' Date      : 8/2/2014
    ' Purpose   : Lets user test the SQL by opening a query to view the results.  Useful for initial verification of this feature and
    '           :   for any manual edits made to the SQL.  Also determines if WHERE clause contains EMAILABILITY test (if applicable).
    ' Editing   :
    '---------------------------------------------------------------------------------------

    Dim db As DAO.Database
    Dim qdf As DAO.QueryDef
    Dim strSQL As String
    Dim lngID As Long
    Dim strCommo As String
    Dim strWHERE As String

5140  If gboolError_Handling = True Then On Error GoTo Err_Handler

5150  Set db = CurrentDb()

    'Save any changes that the user made on the screen (especially in the Me!SQL_For_List text box we are about to test).
5160  If Me.Dirty Then
5170      Me.Dirty = False
5180  End If

    '-------------------------
    ' If something is recorded in the "Procedure_To_Run" control, then this contact list is not meant to use user-defined SQL.
    '-------------------------
5190  If Trim(Me!Procedure_To_Run & "") <> "" Then
5200      MsgBox "This list is generated with VBA and not SQL. Nothing to test."
5210      GoTo Exit_Now
5220  End If
```

Except for simple procedures and functions, all VBA code has a header.

Variable names use prefixes to identify their type.

To disable error handling during debugging and to later enable it, a global variable, **gboolError_Handling**, is used.

Rather than use Exit Sub/Function in the body of my code, my convention is to always jump to an exit routine (Exit_Now) where object closing and other clean-up can be performed.

```
      '-------------------------
      ' QA: If no SQL, let user compose some if they want.
      '-------------------------
5230  If strSQL = "" Then 'Let user design a query.
5240      MsgBox "There's no SQL to preview, but if you are proficient with creating queries in Access, create one and copy-and-paste the SQL back here."
5250      GoTo Exit_Now
5260  End If

      '-------------------------
      ' QA: ID field must always be present, even if not used
      '   (used for recording contact history)
      '-------------------------
5270  If strSQL Like "*Members.ID,*" = False And strSQL Like "*Members.ID *" = False Then
5280      MsgBox "Your SQL's SELECT statement must include the member ID number field, Members.ID. " & _
                 "Even if it is not used in your output, it is used to record contact history for " & _
                 "each recipient."
5290      GoTo Exit_Now
5300  End If

      '-------------------------
      ' QA: Examine WHERE clause in SQL: Is list specific to EMAILABLE members?
      '-------------------------
5310  strCommo = Nz(Me!lstMeans_Of_Communication.Value, "")
5320  strWHERE = funExtract_WHERE_Clause(strSQL) 'return the WHERE clause of the SQL.

      'Just for the comparison we are about to do, remove all parentheses in the stringe (they would complicate our string comparisons).
5330  strWHERE = Replace(strWHERE, "(", "")
5340  strWHERE = Replace(strWHERE, ")", "")

5350  If strCommo Like "EMAIL*" And strWHERE Like "*Email_Address*" = False Then
5360      MsgBox "Your list is for EMAILABLE members, but your WHERE clause does not look for EMAILABLE members. " & _
                 "Add an EMAILABILITY test to your SQL or change the list's Means Of Communication.", _
                 vbCritical, "WHERE clause problem"
5370      GoTo Exit_Now
5380  End If
```

Comments explain what each block does.

```
    '-------------------------
    ' Run test by applying SQL to our "generic" query def.
    '-------------------------
5390  Set qdf = db.QueryDefs("qryOutput_SQL")
5400  qdf.SQL = strSQL
5410  db.QueryDefs.Refresh

5420  DoCmd.OpenForm "popSQL_Preview" 'Open a pop-up form that displays the SQL. We use a pop-up so the user can close it without the Ribbon and "X" buttons visible.

Exit_Now:
5430  On Error Resume Next
5440      Set qdf = Nothing
5450      Set db = Nothing
5460      Err.Clear
5470      Exit Sub

Err_Handler:
5480  If Err.Number = 2501 Or Err.Number = 3420 Then '"RunCommand was cancelled (2501) or Object invalid or no longer set (3420); the latter happens when you test SQL before record saved.
5490      Resume Next
5500  End If
5510  procGeneric_Err_Handler "cmdTest_SQL_Click", Err.Number, Err.Description, Erl  'display an error message; also, turn off hourglass and reset warnings and echo.
5520  Resume Exit_Now


End Sub
```

Exit routine. Rather than using “Exit Sub/Function” in the body of the procedure, we jump to here so that objects can be closed and set to nothing. Of course, an error will occur if we try to close an object that has not been set. We address that with “On Error Resume Next” --- a rare instance where this should be used. We then clear any errors just before we exit the procedure.

Error handling is centralized for the most part. “procGeneric_Err_Handler” is the procedure that does this. We will look at that in the next slide.

VBA SAMPLE: CENTRALIZED ERROR HANDLING

In the slides that follow, we will look at the centralized error handling performed by the procedure "**procGeneric_Err_Handler**", saved as "Public" in a standard module so it is available everywhere in the application. It is called by a procedure/function's error handling routine --- but only if error handling has been turned on in the application (i.e., global variable gboolError_Handling = True).

The procedure below shows how "**procGeneric_Err_Handler**" gets called.

Error handling is centralized for the most part. "procGeneric_Err_Handler" is the procedure that does this. We will look at that in the next slide.

```
Private Sub Form_Current()

5530   If gboolError_Handling = True Then On Error GoTo Err_Handler

       'Run the code the displays or hides the output options based on the means of communication selected.
5540   lstMeans_Of_Communication_Click

       'Run the code that runs when the user decides between running with SQL or procedure.
5550   frameRun_From_Procedure_AfterUpdate

Exit_Now:
5560   Exit Sub

Err_Handler:
5570   procGeneric_Err_Handler "Form_Current", Err.Number, Err.Description, Erl   'display an error message; also, turn off hourglass and reset warnings and echo.
5580   Resume Exit_Now

End Sub
```

The centralized error handler we are about to look at. "procGeneric_Err_Handler" is passed several arguments, including the line number where the error occurred.

Shown below is the beginning of the standard module that contains our centralized error handler, the procedure "procGeneric_Err_Handler."

```
Option Compare Database
Option Explicit

'@@@@@@@@@@@@@@@@@@@@@@

'The following is used to turn e
'It's value is set when the data
'when the form frm_Admin_Setting
Public gboolError_Handling As Boolean


'@@@@@@@@@@@@@@@@@@@@@@
Public Sub procGeneric_Err_Handler(ByVal strProcedure As String, ByVal intError As Integer, _
                                   ByVal strError As String, ByVal lngLine As Long, _
                                   Optional ByVal strForm As String)
    '---------------------------------------------------

    ' Procedure : procGeneric_Err_Handler
    ' Author    : Will Beauchemin
    ' Date      : 2/22/2012
    ' Purpose   : Prepare an Outlook email to report
    '           :    Also, to turn off annoying sett
    ' Editing   : Made gboolError_Handling and email
    '---------------------------------------------------

    Dim strBody As String, strMsg As String, strTo A
    Dim strSource_Db As String, strSource_DB_Short A
    Dim strVersion As String
    Dim strEmail_DB_Repairer As String
    Dim strLive_Or_Test As String
```

This is the global variable that determines whether or not error handling is on. In the application we are considering, a variable is used rather than a constant: the end user has the ability to set the value using an admin feature in the GUI, letting the end-user do their own debugging. This was necessary since the non-profit that uses this has no on-site maintenance person. In most business applications, however, I would use a constant defined here.

The central error handler. As we will see on the following slides, this performs the following tasks:

- Handling specific, anticipated errors.
- Turns off Hourglass and turns on Echo and SetWarnings.
- Close any open recordsets.
- Prepare an on-screen error message for the user to see.
- Prepare an email message the user can send the off-site database maintainer (me) with details about the error.

In other applications, I might include creating an entry in an error log table. In this instance, with no on-site database maintainer, an error log would be pointless.

```
Public Sub procGeneric_Err_Handler(ByVal strProcedure As String, ByVal intError As Integer, _
                                   ByVal strError As String, ByVal lngLine As Long, _
                                   Optional ByVal strForm As String)
    '---------------------------------------------------------------------------------------

    ' Procedure : procGeneric_Err_Handler
    ' Author    : Will Beauchemin
    ' Date      : 2/22/2012
    ' Purpose   : Prepare an Outlook email to report them, and give user a message.
    '           :    Also, to turn off annoying settings that could be left on when an error occurs (e.g., hourglass)
    ' Editing   : Made gboolError_Handling and email recipient be fed from new table admin_Settings.
    '---------------------------------------------------------------------------------------

    Dim strBody As String, strMsg As String, strTo As String, strSubject As String
    Dim strSource_Db As String, strSource_DB_Short As String
    Dim strVersion As String
    Dim strEmail_DB_Repairer As String
    Dim strLive_Or_Test As String

34960 If Err.Number = 3044 Then 'x "is not a valid path"; will happen the first time a new user opens the db and admin settings point to previous user's folders.
34970     Err.Number = 0
34980     MsgBox "Hello.  You must be new to the MAS Member Database.  We can't link to Member data (LIVE or TEST) until you specify where to find that data." & _
              Chr(13) & Chr(13) & _
              "After you close this message, go the " & Chr(34) & "Admin Settings & Info" & Chr(34) & " tab and specify where to find your LIVE and your TEST data." & _
              Chr(13) & Chr(13) & _
              "You'll also need to specify the folder locations and file names for the Contact Lists on the " & Chr(34) & "Lists - Standard" & Chr(34) & " tab." & _
              Chr(13) & Chr(13) & _
              "Welcome to the MAS Member Database.", vbInformation, "Welcome to MAS Member Database"
34990     End
35000     GoTo Exit_Now
35010 End If

35020 strEmail_DB_Repairer = DLookup("[Email_DB_Repairer]", "[admin_Settings]")

    '---------------------------------------
    ' Reset options that would cause problems if left as-is due to the code having bro
    '---------------------------------------
35030 With DoCmd
35040     .SetWarnings True
35050     .Echo True
35060     .Hourglass False
35070 End With

35080 If IsLoaded("frmMsg_Please_Wait", acForm) = True Then
35090   DoCmd.Close acForm, "frmMsg_Please_Wait"
35100 End If
35110 If IsLoaded("popProgress_Bar", acForm) = True Then
35120     DoCmd.Close acForm, "popProgress_Bar"
35130 End If
```

Handle an anticipated error: a new person has assumed the role of end user, but has not configured the admin settings for their computer's folder structure.

Because this non-profit organization does not have an on-site database maintainer, the end user needs to email me with any problems. Rather than hard-code my email address here, I have it updatable in an Admin feature of the GUI. At this point in the VBA, we are looking up that email address.

NOTE: I use a DLOOKUP here rather than a recordset search because the table "admin_Settings" consists of only one record. DLOOKUP is far simpler and has no adverse effect on speed.

Should the error have left Hourglass, Echo, and SetWarnings in a mode that would cause problems, we reset them here. Similarly, we close inconsequential pop-up forms should they be open.

```
      '---------------------------------------
      ' Prepare the details about this error (which we will record, display, and output in various ways)
      '---------------------------------------
35140 strSource_Db = CurrentDb.Name 'Database name

      'Database "Version_ID" is a custom property set in zzzProgrammer_Toolbox module. It is date-time.
35150 strVersion = funDatabase_Version

      'LIVE or TEST mode.
35160 strLive_Or_Test = DLookup("[LIVE_Or_TEST]", "[admin_Settings]")

      'Get the short name of the source db by stripping out the path from the string.
35170 strSource_DB_Short = funCurrent_DB_Location("File")

      '-----------------------------------
      ' Prepare message box text for the user
      '-----------------------------------
35180 strMsg = "An unexpected error occurred:" & Chr(13) & Chr(13) & "Error #:  " & _
              intError & Chr(13) & "Description:  " & strError & Chr(13) & "At line #:  " & _
              lngLine & Chr(13) & "DB version:  " & strVersion

35190 If Nz(strForm, "") <> "" Then
35200     strMsg = strMsg & Chr(13)
35210 End If

      'If a form name was passed, include that in our message.
35220 If Nz(strForm, "") <> "" Then
35230     strBody = strBody & vbCrLf & "   Occured in form:  " & Chr(34) & strForm & _
              Chr(34) & ".  "
35240 End If

      'Display message on the screen about this error.
35250 strMsg = strMsg & Chr(13) & Chr(13) & _
          "An email message with these details has been prepared for
          "Add any comments you want and send it to the Database Admi
          "been importing when this error occurred."
      'Display the Access message box.
35260 MsgBox strMsg, vbCritical, "An error occurred"
```

An on-screen message for the user.

```
      '---------------------------------------
      ' Produce our email message --- but don't send it (so user can address it and add comments)
      '---------------------------------------
35270 strSubject = "ERROR REPORT from " & Chr(34) & strSource_DB_Short & Chr(34)
35280 strTo = strEmail_DB_Repairer
35290 strBody = "From database " & Chr(34) & strSource_Db & Chr(34) & vbCrLf & _
          "DB version:  " & strVersion & vbCrLf & vbCrLf & "Error #" & intError & ":  " & _
          strError & " " & vbCrLf & "   Procedure/function:  " & Chr(34) & strProcedure & _
          Chr(34) & vbCrLf & "   Line #:  " & lngLine & vbCrLf & "   Occurred at:  " & _
          Now() & vbCrLf & vbCrLf & _
          "Data mode was: " & UCase(funLive_Or_Test_Choice)

35300 If funEmail_Compose(strSubject, strBody, strTo, "", "") = False Then
35310     strMsg = "An Outlook email message should have been generated with details about this error, but that attempt failed. " & _
                   "Therefore, take a screenshot of this message you are reading and paste it into an email message." & _
                   Chr(13) & Chr(13) & _
                   strBody
35320 End If
```

…and an email composed that can be sent to the designated database maintainer. Note the details that are included.

There is a function for composing an Outlook email message. The function returns a value of True if there are no errors. We will see this function in a later slide.

```
Exit_Now:
35330 On Error Resume Next
35340 Close_Any_Open_Recordsets 'run t                ecordsets and db objects and closes them.
35350 Err.Clear
35360 Exit Sub

Err_Handler:
35370 If Err.Number = 3420 Then 'Object invalid or no longer set (because user decided to close email message without sending).
35380     Resume Exit_Now
35390 ElseIf Err.Number = 0 Then
35400   Err.Clear
35410   Resume Exit_Now
35420 End If

35430 MsgBox "Error " & Err.Number & ": " & Err.Description & _
             "  This occurred in line " & Erl & _
             " of procedure procGeneric_Err_Handler."
35440 Resume Exit_Now

End Sub
```

If a recordset is left open due to an error, it could prevent Access from closing, forcing the user to use Task Manager to do so. To avoid this, we call a procedure that will close any open recordsets.

Even an error handler could have errors. Since this is most likely to happen to me while I develop, I simply report that error with a message box and so avoid a circular error.

```
Public Sub Close_Any_Open_Recordsets()
      '---------------------------------------------------------------------------------------
      ' Procedure : Close_Any_Open_Recordsets
      ' Author    : Will Beauchemin (Himself@WillBeauchemin.com)
      ' Date      : 1/15/2015
      ' Purpose   : Loop through any and all recordsets and close them.  Useful for close-of-db
      '           :    code to prevent application from failing to close due to open recordsets.
      ' Editing   :
      '---------------------------------------------------------------------------------------

      Dim db As DAO.Database
      Dim rst As DAO.Recordset
      Dim wsp As Workspace

35500 If gboolError_Handling = True Then On Error GoTo Err_Handler

35510 For Each wsp In Workspaces

35520     For Each db In wsp.Databases
35530         For Each rst In db.Recordsets
35540             rst.Close
35550             Set rst = Nothing
35560         Next 'rst
35570         db.Close
35580     Next 'db
35590     wsp.Close
35600     Set wsp = Nothing
35610 Next 'wsp

Exit_Now:
35620     Exit Sub

Err_Handler:
35630 procGeneric_Err_Handler Err.Number, Err.Description, Erl, "Close_Any_Open_Recordsets"
35640 Resume Exit_Now

End Sub
```

VBA SAMPLE: GENERATING EMAIL

In the application we are considering, there are several occasions when an email would need to be composed, such as to remind members of the approaching expiration of their membership. We have already seen when email messages are composed to report errors to me.

Regardless of the purpose of the email, I have my VBA call this function, right, to *compose* the email. I never want to automatically send email, however.

Note that the function includes the ability to have attachments added.

If no errors are encountered, this function returns a value of True.

Note that the Err_Handler of this email generator calls the centralized error handling code, which includes generating an email message about the error. This would seem to be inviting a circular error. However, funEmail_Compose is so simple that the most likely cause of errors would be with the arguments that are being passed --- and any error handling-related email messages would have different, proven arguments.

```
Public Function funEmail_Compose(strSubject As String, strBody As String, strTo As String, strCC As String, strBCC As String, Optional strAttachment As String) As Boolean
      '---------------------------------------------------------------------------------------
      ' Procedure : funEmail_Compose
      ' Author    : Will Beauchemin (Himself@WillBeauchemin.com)
      ' Date      : 7/24/2014
      ' Purpose   : Generate an email using the particulars passed as arguments.
      '---------------------------------------------------------------------------------------
      Dim olookApp As Outlook.Application
      Dim olookMsg As Outlook.MailItem
      Dim olookAttachments As Outlook.Attachments

34630 If gboolError_Handling = True Then On Error GoTo Err_Handler

34640 Set olookApp = New Outlook.Application
34650 Set olookMsg = olookApp.CreateItem(olMailItem)

34660 If strAttachment <> "" Then
34670     Set olookAttachments = olookMsg.Attachments
34680     olookAttachments.ADD strAttachment
34690 End If

34700 With olookMsg
34710     .Subject = strSubject
34720     .Body = strBody
34730     .To = strTo
34740     .CC = strCC
34750     .BCC = strBCC
34760 End With

34770 olookMsg.Display
34780 Set olookApp = Nothing

      'If we get this far, there were no errors.
34790 funEmail_Compose = True

Exit_Now:
34800 On Error Resume Next
34810 Set olookMsg = Nothing
34820 Set olookApp = Nothing
34830 Err.Clear
34840 Exit Function

Err_Handler:
34850 Select Case Err.Number
        Case -2147418107 'Automation error due to problems with Outlook "being in filter mode" (related to Voltage Security add-in).
34860     Resume Exit_Now
34870   Case 429, 462 'Outlook was not already running, so we need to create an instance.
34880     Set olookApp = New Outlook.Application
34890     Resume Next
34900   Case 2486 'Can't perform task at this time.
34910     Resume Next
34920   Case Else
34930      procGeneric_Err_Handler "funEmail_Compose", Err.Number, Err.Description, Erl
34940 End Select
34950 Resume Exit_Now

End Function
```

VBA SAMPLE: SIMULATING A MAIL MERGE IN WORD

Beginning with Office 2010, you cannot use VBA to initiate a mail merge in MS Word if that VBA is being run from the same database as the mail merge's record source. Microsoft has not explained this lost functionality, but it would seem to be a security precaution (like so many lost features that MS *does* acknowledge).

In the application we are considering for VBA samples, there are several mail merges the user needs to perform. To run those from this application (which is also the record source for the mail merge), I had to find an alternative to the MS Word mail merge. I did: bookmarks.

Using named bookmarks in an MS Word template (rather than mail merge fields) gives the same results as before --- and actually makes template creation and editing *easier*.

Although it is not necessary, I chose to identify my bookmarks in the template using angle brackets enclosing the bookmark name. The visible name is merely cosmetic; it is the *defined* name that drives my VBA, an excerpt of which is shown right.

Excerpt of Access VBA that runs "mail merge" in Word.

```
47620 Do While Not rst.EOF
47630     lngID = rst!ID
47640     strMembership_Type = rst!Membership_Type
47650     strFNAME = rst!FNAME
47660     strMID_INIT = Nz(rst!MID_INIT, "")
47670     strLNAME = rst!LNAME
47680     strExpiration = Format(rst!Expiration_Date, "mmmm d, yyyy")
47690     strEmail = rst!Email_Address
      '-------------------------------------------------------------------------------
47700     Set wdDoc = wdApp.Documents.ADD(strRenewal_Form_Template)
47710     With wdDoc

      '--- Replace bookmarks with info on this member:
47720         .Bookmarks("Member_ID").Range.Text = lngID
47730         .Bookmarks("FNAME").Range.Text = strFNAME
47740         .Bookmarks("LNAME").Range.Text = strLNAME
47750         .Bookmarks("Organ").Range.Text = Nz(rst!ORGAN, "")
47760         .Bookmarks("Address1").Range.Text = rst!ADDRESS1
47770         .Bookmarks("Address2").Range.Text = Nz(rst!ADDRESS2, "")
47780         .Bookmarks("City").Range.Text = rst!CITY
47790         .Bookmarks("State").Range.Text = rst!STATE
47800         .Bookmarks("ZIP").Range.Text = rst!ZIP
47810         .Bookmarks("Membership_Type").Range.Text = UCase(strMembership_Type)
47820         .Bookmarks("Expiration").Range.Text = strExpiration
              '---
47830         .Bookmarks("NAME_FULL").Range.Text = strFNAME & IIf(strMID_INIT = "", " ", " "
47840         .Bookmarks("Organ_2").Range.Text = Nz(rst!ORGAN, "")
47850         .Bookmarks("Address1_2").Range.Text = rst!ADDRESS1
47860         .Bookmarks("Address2_2").Range.Text = Nz(rst!ADDRESS2, "")
47870         .Bookmarks("City_2").Range.Text = rst!CITY
47880         .Bookmarks("State_2").Range.Text = rst!STATE
47890         .Bookmarks("ZIP_2").Range.Text = rst!ZIP
47900         .Bookmarks("HPHONE").Range.Text = Nz(rst!HPHONE, "")
47910         .Bookmarks("CELL_PHONE").Range.Text = Nz(rst!CELLPHONE, "")
47920         .Bookmarks("Email_Address").Range.Text = strEmail
              '---

              'Remove any empty lines.  We'll do this by replace any double manual line break
47930         With wdDoc.Content.Find
47940             .ClearFormatting
47950             .Replacement.ClearFormatting
47960             .Text = "^l^l"
47970             .Replacement.Text = "^l"
47980             .Forward = True
47990             .Wrap = wdFindContinue
48000             .Format = False
48010             .MatchCase = False
48020             .MatchWholeWord = False
48030             .MatchWildcards = False
48040             .MatchSoundsLike = False
48050             .MatchAllWordForms = False
48060             .Execute Replace:=wdReplaceAll
48070         End With

              'Prepare file name and save this letter.
48080         strRenewal_Form_Merged = strSave_Folder & "RENEWAL FORM " & Format(Date, "YYYY-MM-DD") & " - " & strLNAME & " (" & lngID & ").docx"
```

My "mail merge" template.

<<FNAME>> <<LNAME>>
<<ORGAN>>
<<ADDRESS1>>
<<ADDRESS2>>
<<CITY>>, <<STATE>> <<ZIP>>

4/23/2015

Membership Renewal Notice

Your membership in the Minnesota Astronomical Society will expire on <<Expiration>>. This will be your only notice. To remain a member, please return this *entire* form with your payment to:

[redacted]

If you prefer you may instead renew your membership online, including optional subscriptions, using PayPal at: www.mnastro.org/membership/rates_paypal.htm.

Have any questions? Please contact me at [redacted]

Please return the *entire* form with your payment. If needed, correct any incorrect contact info below.

<<NAME_FULL>> **EMAIL:** <<EMAIL_ADDRESS>>
<<ORGAN_2>>
<<ADDRESS1_2>>
<<ADDRESS2_2>>
<<CITY_2>>, <<STATE_2>> <<ZIP_2>>
Home Phone: <<HPHONE>> **Mobile:** <<CELL_PHONE>>

☐ ALL my information above is correct. --- *OR* --- Changes are noted above.

☐ Yes, please send me confirmation messages by E-mail rather than by postal mail.
☐ Yes, please list me in the annual Club Directory (to members only; ONLY. Name, City, E-mail addr.)
☐ Yes, I want to receive ***Reflector*** (a quarterly publication of the Astronomical League; FREE).

ABOUT GEMINI: All members receive ***Gemini*** as a free PDF download from the MAS web site. You may opt for the paper version of the bimonthly newsletter for an additional fee. Please see below.

Current Membership Type: <<Membership_Type>>	
$65.00 – Patron Membership: includes all household members. $39.00 is tax deductible. $26.00 – Regular Membership: includes all household members. $13.00 – Student Membership: for full time students only, individual only.	$
$9.00 – Optional printed copy of ***Gemini*** newsletter **(Additional fee)**	$
$32.95 – Optional one year ***Sky and Telescope*** subscription **(Additional fee)**	$
Optional subscription to ***Astronomy*** magazine: 1-yr $34.00, 2-yr $60.00 **(Add'l fee)**	$
Optional tax deductible donation:	$

Member ID# <<ID>> Total $_____