User forms allow a user to input information into an Excel program. Simple controls include message and input boxes. More complex custom user forms are also available. User forms can be used to facilitate the correct entry of data into a spreadsheet.

**Input Boxes**

Input boxes are built with the `InputBox` function. The input box queries a user for information. Several aspects of the box are configurable. There are two buttons, OK and Cancel. The box returns a string value.

**The `InputBox` Function Syntax**

```
InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])
```

// arguments in square braces are optional, prompt is required
// notice how helpfile and context are enclosed together and need to be used together

**Part Description**

- **prompt**
  - required
  - string message displayed in the dialog box.

- **title**
  - optional
  - string message displayed in the title bar of dialog box.

- **xpos**
  - optional
  - horizontal distance of dialog box left edge from screen left edge
  - by default centered.

- **ypos**
  - optional
  - the vertical distance of dialog box upper edge screen top
Helpfile
  - optional
  - string identifies the Help file to use
  - requires context

Context
  - optional
  - numeric expression that is the Help context number
  - assigned to the appropriate Help topic by the Help author.

Example // from 'VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

AverMos = InputBox(Prompt:="Enter the number " & _
               " of months to average", Title:="Enter Months", _
               Default:="3")

Following is a working example of the function. The value entered into the box is entered into the associated spreadsheet.

Macro Example

Sub InBox()

InvoiceNumber = InputBox(Prompt:="Enter the Invoice Number", Title:="Invoices", Default:="DL0 00000")
  If vbOK Then
    Range("A1").Value = InvoiceNumber
  End If
End Sub

Message Boxes

The message box is used to communicate information to a user and solicit a response corresponding to the combination of buttons that are supplied.

Variation on the Case Statement

We have used Case statements where the value for each case is stated after the Case keyword.

Example

Case vbYes
Mr. Excel examples an equivalent form (on page 184 2010 book, page 176 in 2013 book which adds the 'Is' keyword, more or less to say 'If the case is equal to x'. Either form works. Some programmers may like the read implied by using 'Is' as 'is equal to'.

Example

Case Is = vbYes

Macro Using the MsgBox Function

Sub MessageBox()
    'coding without declarations
    TheTitle = "Purchase Confirmation"
    TheMessage = " Do you wish to proceed with this purchase? "

    Cells(1, 1).Value = "Confirmation Required: "
    Cells(1, 1).Columns.AutoFit

    Response = MsgBox(TheMessage, vbExclamation + vbYesNoCancel, TheTitle)

    Select Case Response
        Case Is = vbYes
            Cells(1, 2).Value = "Positive"
        Case Is = vbNo
            Cells(1, 2).Value = "Negative"
        Case Is = vbCancel
            Cells(1, 2).Value = "Pending"
    End Select
End Sub

Custom User Forms

Excel allows you to create custom use forms as part of the application.

In the VBEditor

- select Insert → Userforms

A blank user form is created and appears in the code window.

- Resize the form via the square 'handles' at it's edges.
- Click desired controls and drop onto form.
- Resize as appropriate
- Right Click and adjust properties as appropriate

To access more controls right-click the toolbox and select Additional Controls. Mr. Excel advises to use the additional controls with discretion as other users may not have them installed.
Properties can be changed manually or by using VBA as we shall see.

To Make Properties Window Visible

- click View → Properties Window

Renaming Controls

You can name your user form in the properties window.

Methods for Showing, Hiding Loading & UnLoading Userforms

- Show
  - displays a user form
- Hide
  - hides a user form
- Load
  - form is active but hidden, controls available to programs
- UnLoad
  - removes form from memory, form not accessible from program

Me

Me is a keyword that can be used in a control to refer to the useform itself, typically the currently active userform.

Example // from VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

Unload Me

Assuming you have a control built called LTSandwich the following would show it over a spreadsheet.

Example

Sub SubShop()
  LTSandwich.Show
End Sub

You can always use Project Explorer code to find user form code. Use View to toggle between Form Code and the Object.
General Events in Excel

The Event Model that is used with UserForms is the same as that used generally in Excel. We looked at the Event model briefly in an earlier talk and surveyed the different sorts of event processing that were possible to use at the following levels.

Excel VBA Event Levels

• Application
• Workbook
• Worksheet
• Chart Sheet

While not included as such in our textbook, we might have added User form events to this list as well.

User Form Events

The code that is shown associated with a user form in Excel is really the event processing methods associated with each control. It is here where an action can be assigned to respond to a control has been activated.

Add Event Methods From Drop Down List

Events that can used with user forms are listed in the drop down list at the upper right region of the user form code window. Highlight and click on a desired event handler and a sub with the appropriate name is entered into the user form code.

The General Form of Controls

The following is an event handler that shows the standard features of an event handling routine.

Example

Private Sub CommandButton1_Click()
    'more code is executed behind the scenes
End Sub

the subroutine is marked private
the first part of name is the control name
the second part is the action event
some handlers will have additional arguments to the routine
The Event Model Used With UserForms is the One Used Generally in Excel

The User form event model is the same as that which is used generally in Excel. We saw briefly how it worked in an earlier lecture.

Image of Subshop UserForm Together With Worksheet

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Submarine Sandwich Order</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Lettuce</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Tomato</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Macro To Run UserForm

Sub SubShop()
Range("A1").Value = "Submarine Sandwich Order"
Range("A1").Columns.AutoFit
LTSandwich.Show
End Sub

User Form Event Methods Code

Private Sub CheckBox1_Click()
Range("A2").Value = "Lettuce"
End Sub

Private Sub CheckBox2_Click()
Range("A3").Value = "Tomato"
End Sub

Private Sub CheckBox3_Click()
Range("A4").Value = "Mayo"
End Sub

Private Sub CommandButton1_Click()
CheckBox1.Value = False
CheckBox2.Value = False
CheckBox3.Value = False
Range("A2:A4").Value = ""
End Sub
The text lists all the events for each control which is a good for reference however there is a lot of redundancy in the lists so we show only one such example. // see text for many more similar lists

### UserForm Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Action That Triggers the Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate</td>
<td>Triggered by loading or showing</td>
</tr>
<tr>
<td>AddControl</td>
<td>Event when a control is added at runtime.</td>
</tr>
<tr>
<td>BeforeDragOver</td>
<td>When drag-and-drop is done on form.</td>
</tr>
<tr>
<td>BeforeDropOrPaste</td>
<td>On mouse release drop into user form</td>
</tr>
<tr>
<td>Click</td>
<td>On mouse click over form.</td>
</tr>
<tr>
<td>DblClick</td>
<td>On mouse double-click over form.</td>
</tr>
<tr>
<td>Deactivate</td>
<td>On user form deactivation.</td>
</tr>
<tr>
<td>Error</td>
<td>On error whose info can't be sent to app</td>
</tr>
<tr>
<td>Initialize</td>
<td>On first loading of form before activation</td>
</tr>
<tr>
<td>KeyDown</td>
<td>On key press</td>
</tr>
<tr>
<td>KeyPress</td>
<td>On pressing any ANSI key.</td>
</tr>
<tr>
<td>KeyUp</td>
<td>On key release</td>
</tr>
<tr>
<td>Layout</td>
<td>On size change</td>
</tr>
<tr>
<td>MouseDown</td>
<td>On mouse press</td>
</tr>
<tr>
<td>MouseMove</td>
<td>On mouse move</td>
</tr>
<tr>
<td>MouseUp</td>
<td>On mouse button release</td>
</tr>
<tr>
<td>QueryClose</td>
<td>before user form close</td>
</tr>
<tr>
<td>RemoveControl</td>
<td>On removal of a control at runtime</td>
</tr>
<tr>
<td>Resize</td>
<td>On user form resizing</td>
</tr>
<tr>
<td>Scroll</td>
<td>On user form scrolling</td>
</tr>
<tr>
<td>Terminate</td>
<td>On user form termination</td>
</tr>
<tr>
<td>Zoom</td>
<td>On user form being zoomed</td>
</tr>
</tbody>
</table>

In the user form code window in Excel's VB Editor, while in code view for the user form, each control can be selected from the drop down list on the left and it's properties viewed in the drop down list on the right.

**Controls Are Objects With Properties & Methods**

Controls are objects. They have both properties and methods. View the CommandButton information in the Object browser in this context.
The VBA UserForm Controls provide Partial Programming Access

Mr. Excel points out that much of the programming of controls is done behind the scenes so the programmer doesn't have a complete view of what is being done.

The style of overriding action methods is popular in other programming languages such as Java. Essentially the programmer override event handling subroutines and the internal environment coordinates the application of those directives.

Note:

If you have trouble adding controls to an existing form, Mr. Excel offers a bug fix in the case study on page 189 (2010 book) & 180 (2013 book).

Labels Text Boxes & Command Buttons

Controls form natural groups. For instance, labeled text boxes with buttons that process supplied information will often be found together on a user form.

Mr. Excel's example in the text (p.191 2010 book, page 181, 2013 book) omits showing the subroutines that call the user forms. These calling subs are simple and are found in the resource download.

Image of Mr. Excel's User Form  // generated from Resource Download code

Subs Need To Call Mr. Excel's Example User Forms  // found in resource downloads

Sub AddEmp()
    frm_AddEmp.Show
End Sub
Sub ViewEmp()
    frm_ViewEmp.Show
End Sub

Notice all the action code is put in the user form event handling routines.

The code from the resource download is a little more complicated than that in the book and includes Building 1 and Building 2 information processing.

**Code For Add Form**
<\em>Il from Resource Download, VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad. Page 188</em>

Private Sub btn_EmpCancel_Click()
    Unload Me
End Sub

<\em>Page 190</em>
Private Sub btn_EmpOK_Click()
    Dim LastRow As Long
    LastRow = Worksheets("Employee").Cells(Worksheets("Employee").Rows.Count, 1).End(xlUp).Row + 1
    Cells(LastRow, 1).Value = tb_EmpName.Value
    Cells(LastRow, 2).Value = cb_EmpPosition.Value
    Cells(LastRow, 3).Value = DateSerial(SpBtn_Year.Value, SpBtn_Month.Value, SpBtn_Day.Value)
    If opbtn_Building1.Value Then
        Cells(LastRow, 4).Value = opbtn_Building1.Caption
    Else
        Cells(LastRow, 4).Value = opbtn_Building2.Caption
    End If
    Range("A3:A" & LastRow).Name = "EmpList"
    Unload Me
End Sub

We can create our own form to distill what is in Mr. Excel's example in a simpler form.

**And What Does This UserForm Look Like?**

To get this form we

- open the VB Editor
- Insert UserForm // instead of the usual module to build a Sub
- common elements can be dragged and dropped onto form
- if toolbox is not visible, click View → Toolbox
Three text boxes need to be added to run the following code. Note the default name of the user form is used in the event method code below.

```vba
Sub Code
Sub Baseball()
UserForm1.Show
End Sub

User Form Code
Private Sub CommandButton1_Click()
Dim LastRow As Long
LastRow = Worksheets("Sheet1").Cells(Worksheets("Sheet1").Rows.Count, 1).End(xlUp).Row + 1

Cells(LastRow, 1).Value = TextBox1.Value
Cells(LastRow, 2).Value = TextBox2.Value
Cells(LastRow, 3).Value = TextBox3.Value
End Sub

Lists & Combo Boxes

Lists supply a list of choices. Combo Boxes go one step further supplying a list and the ability to add a choice that is not on the list.

Following is an excellent method for loading a list with an array of fixed values.

The Split Function // routine authored by Gregory K. Maxey,
https://gregmaxey.mvps.org/word_tip_pages/populate_userForm_listbox_or_combobox.html

The following initialization routine supplies a way to add extra features to an array including specifying delimiters. In passing you may wish to check out retired U.S. Navy Commander Gregory Maxey's excellent Excel web site, the link is shown above.

Private Sub UserForm_Initialize()
Dim myArray() As String
'Use Split function to return a zero based one dimensional array.
myArray = Split("AL|AK|AZ|AR|CA|CO|CT|DE|DC|FL|_ & "GA|HI|ID|IL|IN|IA|KS|KY|LA|ME|MD|_ & "MA|MI|MN|MS|MO|MT|NE|NV|NH|NJ|NM|_ & "NY|NC|ND|OH|OK|OR|PA|RI|SC|SD|TN|_ & "TX|UT|VT|VA|WA|WV|WI|WY", "/")
'Use List method to populate listbox.
ListBox1.List = myArray
Lbl_Exit:
Exit Sub
End Sub
```
**Split Function Syntax**

Split(expression[, delimiter[, limit[, compare]]])

- **expression**
  - required
  - expression containing substrings and delimiters
  - If expression is a zero-length string("") , Split returns an empty array
- **delimiter**
  - optional
  - string character used to identify substring limits.
  - if omitted, the space character (" ") is assumed to be the delimiter
- **limit**
  - optional.
  - number of substrings to be returned;
  - –1 indicates that all substrings are returned.
- **compare**
  - optional.
  - numeric value for the comparison method used to evaluate substrings
  - see Settings section for values.

The common simple form used in the example above is Split(expression, delimiter)

**UserForm Showing ComboBox with Custom Input and List**

![UserForm Diagram]

**Macro To Run User Form**

```vba
Sub CityProv()
    With Range("A1")
        .Value = "Location Survey"
        .Columns.AutoFit
        .Interior.Color = RGB(244, 176, 190)
    End With
    UserForm1.Show ' show the user form developed below
End Sub
```
User Form Code

Private Sub CommandButton1_Click()
    Dim LastRow As Long
    LastRow = Worksheets("Sheet1").Cells(Worksheets("Sheet1").Rows.Count, 1).End(xlUp).Row + 1
    Cells(LastRow, 1) = " City: ",
    Cells(LastRow, 2).Value = ComboBox1.Value
    Cells(LastRow, 4).Value = "Province"
    Cells(LastRow, 5).Value = ListBox1.Value
End Sub

Private Sub UserForm_Initialize()
    Dim myArray1() As String
    Dim myArray2() As String

    ' default delimiter is a space with the Split Function
    myArray1 = Split("Waterloo Kitchener Cambridge Guelph London Stratford Elmira")
    ComboBox1.List = myArray1

    ' the delimiter selected here is the vertical bar highlighted below
    myArray2 = Split("British_Columbia|N.W.T|Inuvik|Alberta|_
        & "Saskatchewan|Manitoba|Ontario|Quebec|New_Brunswick|_
        & "Nova_Scotia|P.E.|Newfoundland", ")
    ListBox1.List = myArray2
End Sub

// While other controls may be renamed, the UserForm_Initialize( ) method appears
// to need to be used with it's generic name UserForm.

Adding The Capacity to Make Multiple Choices on Lists

The following code allows a List to have multiple choices made on it
as well either by dragging the mouse or by holding down Ctrl while
selecting by pressing 'Enter'.

Some extra logic is added to accommodate the number or rows
needed with respect to the calculation of 'LastRow'.

Fruits UserForm Created In Excel             User Form In Use With Excel Spreadsheet
Macro To Run The Fruits Form

Sub runFruits()
    Range("A1").Value = "Toppings"
    frmFruits.Show
End Sub

Fruits UserForm Event Handler Code

Private Sub cmdSubmit_Click()
    Dim Flag As Boolean
    LastRow = Worksheets("Sheet1").Cells(Worksheets("Sheet1").Rows.Count, 1).End(xlUp).Row + 1
    For j = 0 To lbxFruits.ListCount - 1
        Flag = False
        If lbxFruits.Selected(j) = True Then
            Cells(LastRow, 1).Value = lbxFruits.List(j)
            Flag = True
        End If
        If Flag = True Then  ' accommodating rows to multi-selects
            LastRow = LastRow + 1
        End If
    Next j
End Sub

Private Sub UserForm_Initialize()
    With lbxFruits
        .Font.Bold = True
        .ForeColor = RGB(44, 99, 166)
        .MultiSelect = fmMultiSelectMulti  ' multi-select settings
        .MultiSelect = fmMultiSelectExtended
    End With
    Dim arr() As String
    arr = Split("strawberries pineapple blueberry raspberry banana melon apple orange peach plum")
    lbxFruits.List = arr
End Sub

Option Buttons

Option Buttons allow check boxes to be grouped into checkbox types sometimes referred to as radio buttons. When one bottom is checked the other is unchecked so choices are limited to a single item.

// labels can trigger the option button see bottom of page 194

The GroupName Property

Grouping is achieved by using the GroupName property. This is handled for you behind the scenes and the VBA programmer needs only select the round option button from the toolbox.
UserForm In Use With Spreadsheet

UserForm Runner Macro

Sub RunHolidayForm()
    Range("A1").Value = "Holiday Destination"
    Holiday.Show
End Sub

User Form Event Handling Code

Private Sub CommandButton1_Click()
    If OptionButton1.Value = True Then
        Range("A2").Value = OptionButton1.Caption
    ElseIf OptionButton2.Value = True Then
        Range("A2").Value = OptionButton2.Caption
    ElseIf OptionButton3.Value = True Then
        Range("A2").Value = OptionButton3.Caption
    End If
End Sub

Private Sub CommandButton2_Click()
    Unload Me
    Transportation.Show
End Sub

Transportation User Form  // Triggered to Show From Next Command Button
**Images in UserForms**

As is shown above, images can be added to enhance Excel UserForms. In the textbook an image was added to a user form in a change event procedure. The code used was something like the following. Notice the error handling formula was used, presumably to protect against an image not being found.

**Example**

```vba
On Error Resume Next
Image1.Picture = LoadPicture("C:\BermudaBeach.jpg")
On Error GoTo 0
```

**Using Image Properties to Add an Image**

However there is an easier Built-In way to add an image that is more 'fail-safe'. That is once the image control has been added to the user form, double click the picture property and a little icon with three periods will show. Click this icon and select the desired image from a location on the system.

**Clicks to Adding An Image**

- Double click Image Control
- From Image Properties
- Double click Picture property
- Click icon with three periods
- Select image

**Spin Buttons**

Spin buttons are fixed scrolling controls that are used to select numbers. This can ensure a data entry such as a date is formatted uniformly. The SpinButton controls are suited to using the Change event to capture the value that they produce as is shown below.

**UserForm With Spin Buttons Used With A Spreadsheet**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Lucky Lotto 6/49 Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 7 11 13 17 19 23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Macro That Runs the UserForm

Sub RunLotto649()
    Cells(1, 1).Value = "Lucky Lotto 649 Number"
    Cells(1, 1).Columns.AutoFit
    Lotto649.Show
End Sub

UserForm Event Handling Code

Private Sub CommandButton1_Click()
Dim num As String

num = TextBox1.Value
num = num & " " & TextBox2.Value
num = num & " " & TextBox3.Value
num = num & " " & TextBox4.Value
num = num & " " & TextBox5.Value
num = num & " " & TextBox6.Value
Cells(2, 1).Value = num
End Sub

Private Sub SpinButton1_Change()
TextBox1.Value = SpinButton1.Value
End Sub

Private Sub SpinButton2_Change()
TextBox2.Value = SpinButton2.Value
End Sub

Private Sub SpinButton3_Change()
TextBox3.Value = SpinButton3.Value
End Sub

Private Sub SpinButton4_Change()
TextBox4.Value = SpinButton4.Value
End Sub

Private Sub SpinButton5_Change()
TextBox5.Value = SpinButton5.Value
End Sub

Private Sub SpinButton6_Change()
TextBox6.Value = SpinButton6.Value
End Sub

The Multipage Control

The Multipage control is used to combine forms. It is not easy to combine forms after the fact so Mr. 'X' suggests pre-planning such forms.
The Multipage control looks like a folder with tabs.

Click and add it to a form. It starts with two pages. Right-click the second tab and you will see you can add another page, along with other options.

**The MultiPage Control Value Property**

The MultiPage control value property holds an offset number which refers to the control's pages.

**Example** // from VBA & Macros' Microsoft Excel 2010, Bill Jelen & Tracy Syrstad.

```
MultiPage1.Value = 3  // refers to the fourth page, value is offset
```

**General Controls**

To add a control like an Exit or Cancel button that affects all the pages of a multiple page control, add the control to the body of the user form outside the tabs of the MultiPage control.

**Image of MultiPage UserForm In Use With Spreadsheet**

```
Sub RunVolunteer()
    Range("A1").Value = "Volunteer Candidate Information"
    Range("A1").Columns.AutoFit
    Range("A8").Value = "Skills Set"
    Range("A12").Value = "Interests"
```

```
Range("A1,A8,A12").Interior.Color = RGB(200, 200, 200)
Volunteer.Show
End Sub

UserForm Event Handling Code

Private Sub CommandButton1_Click( )
'page 1 contact info
Range("A3").Value = "First Name : " & TextBox1.Value
Range("A4").Value = "Last Name : " & TextBox2.Value
Range("A5").Value = "Phone Number: " & TextBox3.Value
Range("A6").Value = "E-Mail : " & TextBox4.Value
'page 2 skill sets
If CheckBox1 = True Then
    Range("A10").Value = CheckBox1.Caption
End If
If CheckBox2 = True Then
    Range("A10").Value = Range("A10").Value & " " & CheckBox2.Caption
End If
If CheckBox3 = True Then
    Range("A10").Value = Range("A10").Value & " " & CheckBox3.Caption
End If
If CheckBox4 = True Then
    Range("A10").Value = Range("A10").Value & " " & CheckBox4.Caption
End If
If CheckBox5 = True Then
    Range("A10").Value = Range("A10").Value & " " & TextBox5.Value
End If
'page 3 Interests textbox
Range("A14").Value = TextBox6.Value
End Sub
'cancel button unloads user form
Private Sub CommandButton2_Click()
Unload Me
End Sub

Field Validation

A field in a user form, particularly a text box, can be tested to see if it is empty with a little boolean logic in which case a message box can be popped up to ask that an entry be made. The user form hide method is handy here.

Example  // would go in Event Handling Code of UserForm

If TestBoxN.Value = "" Then
    theForm.Hide
    MsgBox("An Entry for xxx is required")
    theForm.Show
End If
// see Mr. Excel's example on page 200, discuss if Exit Sub is necessary
Unwanted Window Closing

At times you may not want a user to exit using any closing mechanism except the close methods on the form. There are four conditions in Excel that can close a form. Represented by the following VB constants.

- **vbFormControlMenu**
  - right-click form and click close or
  - click upper right hand corner X
- **vbFormCode**
  - when the Unload statement is invoked
- **vbAppWindows**
  - Windows closes
- **vbAppTaskManager**
  - Task Manager closes the application

The one that will be normally used is the first two on the above list and particularly vbFormControlMenu.

The following example is like the one used in Mr. Excel’s textbook. On the net other examples leave out the VB constant, vbCritical so the message box is a little more subtle.

**UserForms Showing the Message Box That Pops Up on Clicking X to Close**

![UserForm showing message](image1)

**Sub To Run Exits UserForm**

```vba
Sub RunExits()
    Exits.Show
End Sub
```
UserForm Event Handling Code

Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
If CloseMode = vbFormControlMenu Then
MsgBox ("Please Use OK or Cancel to exit this form"), vbCritical
Cancel = True
End If
End Sub

Private Sub CommandButton1_Click()
Range("A1").Value = "Good Job!"
Unload Me
End Sub

Private Sub CommandButton2_Click()
Unload Me
End Sub

Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
If CloseMode = vbFormControlMenu Then
MsgBox ("Please Use OK or Cancel to exit this form"), vbCritical
Cancel = True
End If
End Sub

// See text for a last example on file IO which closes out Chapter 10 (in both 2010 & 2013 books)

Assignment

1 ) Create a Message Box that takes a vote on a topic where there are three choices, 'Yes', 'No' and 'Decline'. Have the result of the vote put into a cell in a spreadsheet following a label captioned, 'Vote Result'. Capture the message box in a screen shot and submit with code.

2 ) Given the following sub and action method, create the appropriate user form to go with the code. Notice you should label the user form 'Baseball Players' or something similar.

The three textfields you add will also be labelled, something like first name, last name and position or name, phone number and position.

Sub Code

Sub Baseball()
UserForm1.Show
End Sub
User Form Code

Private Sub CommandButton1_Click()
Dim LastRow As Long
LastRow = Worksheets("Sheet1").Cells(Worksheets("Sheet1").Rows.Count, 1).End(xlUp).Row + 1

Cells(LastRow, 1).Value = TextBox1.Value
Cells(LastRow, 2).Value = TextBox2.Value
Cells(LastRow, 3).Value = TextBox3.Value
End Sub

3) Create a Sub that uses two page of a multipage form to do a survey on some topic of interest that places the results in a spreadsheet. Label both the user form and the spreadsheet as appropriate.

- textfields
- checkboxes
- option buttons
- spin buttons
- a listbox

You can refer to examples in the note for methods of extracting the information from the user form and putting the information into the spreadsheet.

Optional

For more instruction, step through the following tutorial which provides a good summary of the basic steps involved in developing useful user forms.

UserForm

http://www.excel-easy.com/vba/userform.html