

Re: Class Events

Source:

<http://www.tech-archive.net/Archive/Excel/microsoft.public.excel.programming/2005-10/msg00116.html>

- *From:* "Peter T" <peter_t@discussions>
 - *Date:* Mon, 3 Oct 2005 00:27:20 +0100
-

Hi Gareth,

I ran your code and sort of see what you are doing, though not of course how it relates to your entire project and which parts you want to keep as generic for use in other projects. So the following may not be relevant.

First, I don't see why you need a Withevents class for just your single "large" label. The events already exist in the userform. Could pass the XY coord's of mouse move over the large label to a proc elsewhere, possibly in a non withevents class to do stuff.

But I don't even see why you need the large label at all. Why not dispense with that and set multiple instance's of a withevents class to handle events for each of the grid labels.

In this collection or array of classes you only need to be concerned with label.left, label.width and the Y coordinate to calc' to draw and resize a single red label. Eventually user can click that to create the textbox and remove the temporary red label. Perhaps set an extra instance of the same labels class to handle the red label, thereby avoiding the necessity to "name" its click event in code. (in the class click event – If clsLab.name = varLabelname Then)

Also you could have set whatever unique properties for each label class, as required for other purposes, when these classes were created.

Regards,
Peter T

"Gareth" <nah> wrote in message
news:usWXJK3xFHA.1168@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

> Hi Peter,
>
> Thanks for replying – I think you're right – my posts haven't been that
> clear.
>

Re: Class Events

> I have just one class – and that's all I want to use, for this part at
> least.
>
> The labels hidden under the large label are classless – they have no
> events since they never get clicked (they're always underneath).
>
> I want the logic of the control to follow thus:
>
> When double clicked, tell the parent form that it's been doubleclicked
> and let the parent form decide what to do with it.
>
> I don't want:
> To have the class go off and query the database, populate everything
> etc. because that means the class is no longer generic – it's tied into
> one application and must be modified for use in another.
>
> Since I can't create an event procedure called MyGrid_DoubleClick in the
> userform module I thought I could just set a string in the class called
> OnDoubleClick which was the name of a procedure. This works – but only
> if the procedure is in a standard module. I can't get it to work with
> Userform1.MyProcedureName – whether or not it's Private, not private or
> public. Other than that, this solution is acceptable I guess. I just
> don't like having it in a standard module.
>
> You're right – I could use an If Else construct but again that means the
> Class is not generic.
>
> In case you're still interested (!) I've copied some example code to
> demonstrate the direction I'm heading in. It's crude but it works and
> can just be copied and pasted into a new workbook without any
> modifications.
>
> Just run userform1 and make a selection on the grid using left mouse
> button and moving it left or right and then right click on it. (I'm
> using right click rather than double click for this example.)
>
>
> Many thanks,
> G
>
>
>
> '-----IN USERFORM1-----'
> Option Explicit
> Private Const GRID_START_Y As Integer = 20
> Private Const GRID_START_X As Integer = 50
> Private Const GRID_ROW_HEIGHT As Integer = 20
> Private Const GRID_COL_WIDTH As Integer = 25
> Private Const GRID_NO_OF_ROWS As Integer = 10
> Private Const GRID_NO_OF_COLS As Integer = 24
>

Re: Class Events

Re: Class Events

```
> Private Sub UserForm_Initialize()  
> With Me  
> .Height = 450  
> .Width = 700  
> End With  
> DrawGrid  
> End Sub  
>  
> Sub DrawGrid()  
>  
> Dim lblGrid As MSForms.Label  
>  
> 'Make our main grid label  
> Set lblGrid = Me.Controls.Add("Forms.Label.1", "GRID", True)  
>  
> With lblGrid  
> 'size grid control as desired  
> .Left = GRID_START_X  
> .Top = GRID_START_Y  
> .Height = GRID_NO_OF_ROWS * GRID_ROW_HEIGHT  
> .Width = GRID_NO_OF_COLS * GRID_COL_WIDTH  
> End With  
>  
> 'create the grid control  
> Set GRID.GridControl = lblGrid  
> 'tidy up  
> Set lblGrid = Nothing  
>  
> 'set parameters for the grid  
> With GRID  
> .Start_X = GRID_START_X  
> .Start_Y = GRID_START_Y  
> .RowHeight = GRID_ROW_HEIGHT  
> .ColWidth = GRID_COL_WIDTH  
> .NoOfRows = GRID_NO_OF_ROWS  
> .NoOfCols = GRID_NO_OF_COLS  
> Set .GridParent = Me  
> 'format the grid as per settings  
> .FormatGridControl  
>  
> 'set the procedure to be called in the event _  
> 'of a right click on the grid  
> .OnRightClick = "Event_GridRightClicked"  
> End With  
>  
> End Sub  
>  
> '-----  
>  
> '--IN A STANDARD MODULE-----  
> Option Explicit
```

Re: Class Events

```
> Public GRID As New clsGrid
>
> Sub Event_GridRightClicked()
> GRID.CreateBlock "TEST"
> End Sub
> '-----
>
> '--IN A CLASS MODULE NAMED clsGrid-----
> Option Explicit
>
> Public WithEvents GridControl As MSForms.Label
>
> 'Settings for the grid
> Public Start_Y As Integer
> Public Start_X As Integer
> Public RowHeight As Integer
> Public ColWidth As Integer
> Public NoOfRows As Integer
> Public NoOfCols As Integer
>
> Public GridParent As MSForms.UserForm
>
>
> Public btnMouseButtonAlreadyDown As Boolean
>
> Public GridSelection As Collection
> Public SelectionCurrentRow As Integer
> Public SelectionCurrentCol As Integer
> Public SelectionMinCol As Integer
> Public SelectionMaxCol As Integer
>
> Public GridBlocks As Collection
>
> Public OnRightClick As String
>
> Private Sub Class_Initialize()
> Set GridSelection = New Collection
> Set GridBlocks = New Collection
> SelectionCurrentRow = -1
> SelectionCurrentCol = -1
> End Sub
> Sub FormatGridControl()
> Dim iCol As Integer
> Dim myLbl As MSForms.Label
>
> 'draw the back labels for the grid
> For iCol = 0 To NoOfCols - 1
> Set myLbl = GridParent.Controls.Add("Forms.Label.1", _
> "BackDrop_Col" & iCol, True)
> With myLbl
> .Left = Start_X + (ColWidth * iCol)
```

Re: Class Events

```
> .Width = ColWidth
> .Top = Start_Y
> .Height = NoOfRows * RowHeight
> .BorderStyle = fmBorderStyleSingle
> .BorderColor = RGB(0, 0, 180)
> .BackColor = RGB(255, 255, 255)
> .ZOrder = 1
> End With
> Next iCol
>
> 'format the main label as per user settings
> With Me.GridControl
> .BorderStyle = fmBorderStyleSingle
> .BorderColor = RGB(0, 0, 0)
> .SpecialEffect = fmSpecialEffectSunken
> .BackStyle = fmBackStyleTransparent
> .ZOrder 0
> End With
>
>
>
> Set myLbl = Nothing
>
> End Sub
> Private Sub GridControl_Click()
>
> If blnMouseButtonAlreadyDown Then
> blnMouseButtonAlreadyDown = False
> Else
> fcnClearSelection
> End If
> End Sub
>
> Private Sub GridControl_MouseDown(ByVal Button As Integer, _
> ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)
> 'handle right clicking
> If Not Button = 2 Then Exit Sub
>
> If GridSelection.Count = 0 Then
> MsgBox "pls select something"
> Exit Sub
> End If
> Application.Run OnRightClick
>
> End Sub
>
> Private Sub GridControl_MouseMove(ByVal Button As Integer, _
> ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)
>
> Dim newCol As Integer, newRow As Integer
> 'we want to trap when someone holds the mouse button down
```

Re: Class Events

```
> If Button <> 1 Then Exit Sub
>
> ' the mouse button isn't already down then this is a new selection
> If Not blnMouseButtonAlreadyDown Then
> 'clear any existing "selections" from our collection
> fcnClearSelection
> End If
>
> 'we want to create a label on the grid to represent a selection
> newRow = fcnCalculateGridRowFromY(Y)
> newCol = fcnCalculateGridColFromX(X)
>
> 'if it's the same cell as last time then exit
> If newRow = SelectionCurrentRow And newCol = SelectionCurrentCol
> Then Exit Sub
>
> 'if this is a new row then set this row as our selection row
> 'clear our selection and exit
> If SelectionCurrentRow = -1 Then SelectionCurrentRow = newRow
>
> 'If this is a different row than last time then
> 'we ignore
> If SelectionCurrentRow <> newRow Then Exit Sub
>
> 'if this isn't the same as the previous column then we want to add a
> label
> If SelectionCurrentCol <> newCol And newCol <= NoOfCols - 1 Then
>
> If SelectionMinCol = -1 Then
> SelectionMinCol = newCol
> ElseIf SelectionCurrentCol < SelectionMinCol Then
> SelectionMinCol = SelectionCurrentCol
> End If
> If SelectionCurrentCol > SelectionMaxCol Then _
> SelectionMaxCol = SelectionCurrentCol
>
> fcnAddNewSelectionLabel newRow
> SelectionCurrentCol = newCol
> blnMouseButtonAlreadyDown = True
>
> End If
>
>
>
>
> End Sub
>
> Function fcnCalculateGridRowFromY(Y As Single) As Integer
> fcnCalculateGridRowFromY = CInt(Y / RowHeight - 0.499999)
> End Function
> Function fcnCalculateGridColFromX(X As Single) As Integer
```

Re: Class Events

```
> fcnCalculateGridColFromX = CInt(X / ColWidth - 0.499999)
> End Function
>
> Sub fcnClearSelection()
> While GridSelection.Count > 0
> GridParent.Controls.Remove GridSelection(1).Name
> GridSelection.Remove 1
> Wend
> SelectionCurrentCol = -1
> SelectionCurrentRow = -1
> SelectionMinCol = -1
> SelectionMaxCol = -1
>
> End Sub
> Sub fcnAddNewSelectionLabel(myRow As Integer)
>
> Dim myLbl As MSForms.Label
> Dim iCol As Integer
>
>
> 'We insert this selection label but also
> 'check that we haven't missed any cells
> '(this occurs when the mouse moves too fast
> 'or the user hits another row while moving the mouse)
> For iCol = SelectionMinCol To SelectionMaxCol
>
> 'check whether this label already exists in our collection
> If Not fcnKeyAlreadyExistsInCollection("R" _
> & myRow & "C" & iCol, GridSelection) Then
>
> 'create the control
> Set myLbl = GridParent.Controls.Add("Forms.Label.1", _
> "R" & myRow & "C" & iCol, True)
> With myLbl
> .Left = Start_X + iCol * ColWidth
> .Top = Start_Y + myRow * RowHeight
> .Height = RowHeight
> .Width = ColWidth
> .BorderStyle = fmBorderStyleSingle
> .BorderColor = RGB(200, 0, 0)
> .BackColor = RGB(255, 0, 0)
> End With
>
> On Error Resume Next
> GridSelection.Add myLbl, "R" & myRow & "C" & iCol
>
> End If
>
> Next iCol
>
> 'bring the main grid label back to the front
```

Re: Class Events

```
> Me.GridControl.ZOrder 0
>
> End Sub
> Function fcnKeyAlreadyExistsInCollection(myKey As String, _
> myColl As Collection) As Boolean
> 'checks a given collection to see if a key already exists in there
>
> On Error Resume Next
> If myColl(myKey).Name = "X" Then
> Exit Function
> End If
> fcnKeyAlreadyExistsInCollection = True
> End Function
> Sub CreateBlock(myCaption As String)
> Dim myTextBox As MSForms.TextBox
>
> Set myTextBox = GridParent.Controls.Add("Forms.Textbox.1", _
> "Block_" & "R" & SelectionCurrentRow & "C" & SelectionMinCol, True)
>
> With myTextBox
> .BackColor = RGB(255, 255, 0)
> .Text = myCaption
> .Left = Start_X + SelectionMinCol * ColWidth
> .Top = Start_Y + SelectionCurrentRow * RowHeight
> .Height = RowHeight
> .Width = (SelectionMaxCol - SelectionMinCol + 1) * ColWidth
> End With
> Set myTextBox = Nothing
>
> 'bring the main grid label back to the front
> Me.GridControl.ZOrder 0
> 'add to my collection
> 'DO THIS LATER'
>
> fcnClearSelection
>
> End Sub
> '-----
```

• **References:**

- ◆ **Class Events**
◇ From: Gareth
- ◆ **Re: Class Events**
◇ From: Gareth

- Prev by Date: **Re: Mailing excel pages**

Re: Class Events

- Next by Date: ***Re: Error 1004***
- Previous by thread: ***Re: Class Events***
- Next by thread: ***Re: Class Events***
- Index(es):
 - ◆ ***Date***
 - ◆ ***Thread***