```
'------------------------------------------------------------
' Add in Sub Class_Globals()
'------------------------------------------------------------
Dim ActiveEventNotesevID As Int
Dim IsNewEventNotes As Boolean
'------------------------------------------------------------
' Add in BuildPage()
'------------------------------------------------------------
page.AddModalSheetTemplate(ABMGenBuildEventNotes)
page.AddModalSheetTemplate(ABMGenBuildEventNotesDelete)
page.AddModalSheetTemplate(ABMGenBuildEventNotesBadInput)
'------------------------------------------------------------
' Add in the class
'------------------------------------------------------------
#Region EventNotes
Sub ABMGenBuildEventNotes() As ABMModalSheet
	Dim ABMGenEventNotesModal As ABMModalSheet
	ABMGenEventNotesModal.Initialize(page, "EventNotes", True, False, "")
	ABMGenEventNotesModal.IsDismissible = False
	ABMGenEventNotesModal.Size = ABM.MODALSHEET_SIZE_FULL

	ABMGenEventNotesModal.Content.AddRowsM(1, True, 0, 0, "").AddCells12(1, "")
	ABMGenEventNotesModal.Content.AddRowsM(5, True, 0, 0, "").AddCellsOS(1, 0, 0, 0, 12, 12, 12, "")
	ABMGenEventNotesModal.Content.AddRowsM(1, True, 0, 0, "").AddCellsOS(1, 0, 0, 0, 6, 6, 6, "").AddCellsOSMP(1, 0, 0, 0, 6, 6, 6, 14, 0 ,0 ,0 , "") ' lower the switch a bit
	ABMGenEventNotesModal.Content.AddRowsM(1, True, 0, 0, "").AddCellsOS(1, 0, 0, 0, 12, 12, 12, "")
	ABMGenEventNotesModal.Content.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

	Dim ABMGenEventNotesTitleMessage As ABMLabel
	ABMGenEventNotesTitleMessage.Initialize(page, "ABMGenEventNotesTitleMessage", "Vul alle velden van deze event in." , ABM.SIZE_PARAGRAPH, False, "")
	ABMGenEventNotesTitleMessage.IsBlockQuote = True
	ABMGenEventNotesModal.Content.CellR(0,1).AddComponent(ABMGenEventNotesTitleMessage)
	Dim ABMGenEventNotesevStatus As ABMCombo
	ABMGenEventNotesevStatus.Initialize(page, "ABMGenEventNotesevStatus", "Status", 500, "")
	Dim SQL As SQL = DBM.GetSQL
	Dim SQL_Str As String = "SELECT evstatID, evstatOms FROM tEventStatus ORDER BY evstatOms"
	Dim results As List = DBM.ABMGenSQLSelect(SQL, SQL_Str, Null)
	For l = 0 To results.Size - 1
		Dim res As Map = results.Get(l)
		ABMGenEventNotesevStatus.AddItem(res.GetValueAt(0), res.GetValueAt(1), ABMGenBuildEventNotesComboItem("ABMGenEventNotesevStatus" & (l+1), res.GetValueAt(1)))
	Next
	DBM.CloseSQL(SQL)
	ABMGenEventNotesModal.Content.CellR(1,1).AddComponent(ABMGenEventNotesevStatus)
	ABMGenEventNotesevStatus.SetActiveItemId("2")

	Dim ABMGenEventNotesevType As ABMCombo
	ABMGenEventNotesevType.Initialize(page, "ABMGenEventNotesevType", "Actie", 500, "")
	Dim SQL As SQL = DBM.GetSQL
	Dim SQL_Str As String = "SELECT tEventType.evtypID, tEventType.evtypOms FROM tEventType INNER JOIN tEventType_Persoon ON tEventType.evtypID = tEventType_Persoon.evtypprsEvtypID WHERE (((tEventType_Persoon.evtypprsPrsID)=1)) ORDER BY tEventType.evtypOms"
	Dim results As List = DBM.ABMGenSQLSelect(SQL, SQL_Str, Null)
	For l = 0 To results.Size - 1
		Dim res As Map = results.Get(l)
		ABMGenEventNotesevType.AddItem(res.GetValueAt(0), res.GetValueAt(1), ABMGenBuildEventNotesComboItem("ABMGenEventNotesevType" & (l+1), res.GetValueAt(1)))
	Next
	DBM.CloseSQL(SQL)
	ABMGenEventNotesModal.Content.CellR(1,1).AddComponent(ABMGenEventNotesevType)
	ABMGenEventNotesevType.SetActiveItemId("1")

	Dim ABMGenEventNotesevToDoBy As ABMCombo
	ABMGenEventNotesevToDoBy.Initialize(page, "ABMGenEventNotesevToDoBy", "Uit te voeren door", 500, "")
	Dim SQL As SQL = DBM.GetSQL
	Dim SQL_Str As String = "SELECT prsID, prsNaam FROM tPersoon ORDER BY prsNaam"
	Dim results As List = DBM.ABMGenSQLSelect(SQL, SQL_Str, Null)
	For l = 0 To results.Size - 1
		Dim res As Map = results.Get(l)
		ABMGenEventNotesevToDoBy.AddItem(res.GetValueAt(0), res.GetValueAt(1), ABMGenBuildEventNotesComboItem("ABMGenEventNotesevToDoBy" & (l+1), res.GetValueAt(1)))
	Next
	DBM.CloseSQL(SQL)
	ABMGenEventNotesModal.Content.CellR(1,1).AddComponent(ABMGenEventNotesevToDoBy)
	ABMGenEventNotesevToDoBy.SetActiveItemId("1")

	Dim ABMGenEventNotesevToDoDatum As ABMDateTimeScroller
	Dim NewDate As Long = DateTime.Now 'ignore
	ABMGenEventNotesevToDoDatum.Initialize(page, "ABMGenEventNotesevToDoDatum", ABM.DATETIMESCROLLER_TYPE_DATE, ABM.DATETIMESCROLLER_MODE_CLICKPICK, NewDate, "Uitvoer datum", "")
	ABMGenEventNotesevToDoDatum.TitleDateFormat = "DD"
	ABMGenEventNotesevToDoDatum.ReturnDateFormat = "dd/mm/yy"
	ABMGenEventNotesevToDoDatum.DateOrder = "ddMy"
	ABMGenEventNotesevToDoDatum.DateStartYearNowMinus = 100
	ABMGenEventNotesevToDoDatum.DateEndYearNowPlus = 1
	ABMGenEventNotesevToDoDatum.DateMonthNames = "['Januari','Februari','Maart','April','Mei','Juni', 'July','Augustus','September','Oktober','November','December']"
	ABMGenEventNotesevToDoDatum.DateMonthNamesShort = "['Jan', 'Feb', 'Maa', 'Apr', 'Mai', 'Jun', 'Jul', 'Aug', 'Sep', 'Okt', 'Nov', 'Dec']"
	ABMGenEventNotesevToDoDatum.DateDayNames = "['Zondag', 'Maandag', 'Dinsdag', 'Woensdag', 'Donderdag', 'Vrijdag', 'Zaterdag']"
	ABMGenEventNotesevToDoDatum.DateDayNamesShort = "['Zon', 'Maa', 'Din', 'Woe', 'Don', 'Vri', 'Zat']"
	ABMGenEventNotesevToDoDatum.DateMonthText = "Maand"
	ABMGenEventNotesevToDoDatum.DateDayText = "Dag"
	ABMGenEventNotesevToDoDatum.DateYearText = "Jaar"
	ABMGenEventNotesevToDoDatum.DateShortYearCutoff = "+10"
	ABMGenEventNotesevToDoDatum.PickText = "Kies"
	ABMGenEventNotesevToDoDatum.CancelText = "Sluiten"
	ABMGenEventNotesModal.Content.CellR(1,1).AddComponent(ABMGenEventNotesevToDoDatum)

	Dim ABMGenEventNotesevOnderwerp As ABMInput
	ABMGenEventNotesevOnderwerp.Initialize(page, "ABMGenEventNotesevOnderwerp", ABM.INPUT_TEXT, "Onderwerp", False, "")
	ABMGenEventNotesModal.Content.CellR(1,1).AddComponent(ABMGenEventNotesevOnderwerp)
	ABMGenEventNotesevOnderwerp.Text = ""

	Dim ABMGenEventNotesevAandacht As ABMInput
	ABMGenEventNotesevAandacht.Initialize(page, "ABMGenEventNotesevAandacht", ABM.INPUT_TEXT, "Aandacht", False, "")
	ABMGenEventNotesModal.Content.CellR(1,1).AddComponent(ABMGenEventNotesevAandacht)
	ABMGenEventNotesevAandacht.Text = "0"

	Dim ABMGenEventNotesevHot As ABMSwitch
	ABMGenEventNotesevHot.Initialize(page, "ABMGenEventNotesevHot", "Hot", "", False, "")
	ABMGenEventNotesModal.Content.CellR(0,2).AddComponent(ABMGenEventNotesevHot)
	ABMGenEventNotesevHot.State = False

	Dim ABMGenEventNotesevCommentaar As ABMInput
	ABMGenEventNotesevCommentaar.Initialize(page, "ABMGenEventNotesevCommentaar", ABM.INPUT_TEXT, "Commentaar", True, "")
	ABMGenEventNotesModal.Content.CellR(1,1).AddComponent(ABMGenEventNotesevCommentaar)
	ABMGenEventNotesevCommentaar.Text = ""

	ABMGenEventNotesModal.Footer.AddRowsM(1,True,0,0, "").AddCells12(1,"")
	ABMGenEventNotesModal.Footer.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

	' create the buttons for the footer, create in opposite order as aligned right in a footer
	Dim ABMGenEventNotesCancel As ABMButton
	ABMGenEventNotesCancel.InitializeFlat(page, "ABMGenEventNotesCancel", "", "", "Annuleren", "transparent")
	ABMGenEventNotesModal.Footer.Cell(1,1).AddComponent(ABMGenEventNotesCancel)

	Dim ABMGenEventNotesSave As ABMButton
	ABMGenEventNotesSave.InitializeFlat(page, "ABMGenEventNotesSave", "", "", "Bewaren", "transparent")
	ABMGenEventNotesModal.Footer.Cell(1,1).AddComponent(ABMGenEventNotesSave)

	Return ABMGenEventNotesModal
End Sub

' method you can call when the user wants to add a new record
Sub ABMGenEventNotesNew()
	Dim ABMGenEventNotesModal As ABMModalSheet = page.ModalSheet("EventNotes")
```

```
	ActiveEventNotesevID = 0

	Dim NewDate As Long = DateTime.Now 'ignore
	Dim ABMGenEventNotesevStatus As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevStatus")
	Dim ABMGenEventNotesevType As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevType")
	Dim ABMGenEventNotesevToDoBy As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevToDoBy")
	Dim ABMGenEventNotesevToDoDatum As ABMDateTimeScroller = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevToDoDatum")
	Dim ABMGenEventNotesevOnderwerp As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevOnderwerp")
	Dim ABMGenEventNotesevAandacht As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevAandacht")
	Dim ABMGenEventNotesevHot As ABMSwitch = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevHot")
	Dim ABMGenEventNotesevCommentaar As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevCommentaar")

	ABMGenEventNotesevStatus.SetActiveItemId("2")

	' here reload only the event types for this person
	Dim SQL As SQL = DBM.GetSQL
	ABMGenEventNotesevType.Clear
	Dim resultsTypes As List = DBM.ABMGenSQLSelect(SQL, "SELECT tEventType.evtypID, tEventType.evtypOms FROM tEventType INNER JOIN tEventType_Persoon ON tEventType.evtypID = tEventType_Persoon.evtypprsEvtypID WHERE (((tEventType_Persoon.evtypprsPrsID)=" & ActivePersonID & ")) ORDER BY tEventType.evtypOms", Null)
	For l = 0 To resultsTypes.Size - 1
		Dim res As Map = resultsTypes.Get(l)
		ABMGenEventNotesevType.AddItem(res.GetValueAt(0), res.GetValueAt(1),  ABMGenBuildEventNotesComboItem("ABMGenEventNotesevType" & (l+1), res.GetValueAt(1)))
	Next
	DBM.CloseSQL(SQL)

	ABMGenEventNotesevType.SetActiveItemId("1")
	ABMGenEventNotesevToDoBy.SetActiveItemId(ActivePersonID) 'need the active logged in person
	ABMGenEventNotesevToDoDatum.SetDate(NewDate)
	ABMGenEventNotesevOnderwerp.Text = ""
	ABMGenEventNotesevAandacht.Text = "0"
	ABMGenEventNotesevHot.State = False
	ABMGenEventNotesevCommentaar.Text = ""
	IsNewEventNotes=True
	page.ShowModalSheet("EventNotes")
End Sub

' method you can call when the user wants to edit an existing record
Sub ABMGenEventNotesEdit(openId As Int)
	Dim ABMGenEventNotesModal As ABMModalSheet = page.ModalSheet("EventNotes")
	ActiveEventNotesevID = openId
	Dim NewDate As Long = DateTime.Now 'ignore
	Dim SQL As SQL = DBM.GetSQL
	Dim SQL_str As String = "SELECT evID, evCreator, evContact, evDatum, evStatus, evType, evToDoBy, evToDoDatum, evOnderwerp, evAandacht, evHot, evCommentaar, evAfsluitDatum FROM tEvent WHERE evID = ? "
	Dim results As List = DBM.ABMGenSQLSelect(SQL, SQL_str, Array As Int(ActiveEventNotesevID))
	If results.Size>0 Then
		Dim m As Map = results.Get(0)
		Dim ABMGenEventNotesevStatus As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevStatus")
		Dim ABMGenEventNotesevType As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevType")
		Dim ABMGenEventNotesevToDoBy As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevToDoBy")
		Dim ABMGenEventNotesevToDoDatum As ABMDateTimeScroller = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevToDoDatum")
		Dim ABMGenEventNotesevOnderwerp As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevOnderwerp")
		Dim ABMGenEventNotesevAandacht As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevAandacht")
		Dim ABMGenEventNotesevHot As ABMSwitch = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevHot")
		Dim ABMGenEventNotesevCommentaar As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevCommentaar")

		If m.GetDefault("evstatus", Null) = Null Then
			ABMGenEventNotesevStatus.SetActiveItemId("2")
		Else
			ABMGenEventNotesevStatus.SetActiveItemId(m.Get("evstatus"))
		End If

		' here reload all the evTypes
		ABMGenEventNotesevType.Clear
		Dim resultsTypes As List = DBM.ABMGenSQLSelect(SQL, "SELECT evtypId, evtypOms FROM tEventType ORDER BY evtypOms", Null)
		For l = 0 To resultsTypes.Size - 1
			Dim res As Map = resultsTypes.Get(l)
			ABMGenEventNotesevType.AddItem(res.GetValueAt(0), res.GetValueAt(1),  ABMGenBuildEventNotesComboItem("ABMGenEventNotesevType" & (l+1), res.GetValueAt(1)))
		Next

		If m.GetDefault("evtype", Null) = Null Then
			ABMGenEventNotesevType.SetActiveItemId("1")
		Else
			ABMGenEventNotesevType.SetActiveItemId(m.Get("evtype"))
		End If

		If m.GetDefault("evtodoby", Null) = Null Then
			ABMGenEventNotesevToDoBy.SetActiveItemId(ActivePersonID) ' needs active person
		Else
			ABMGenEventNotesevToDoBy.SetActiveItemId(m.Get("evtodoby"))
		End If

		If m.GetDefault("evtododatum", Null) = Null Then
			ABMGenEventNotesevToDoDatum.SetDate(NewDate)
		Else
			ABMGenEventNotesevToDoDatum.SetDate(ABM.ConvertFromDateTimeString(m.Get("evtododatum"), "yyyy-MM-dd HH:mm:ss"))
		End If

		If m.GetDefault("evonderwerp", Null) = Null Then
			ABMGenEventNotesevOnderwerp.Text = ""
		Else
			ABMGenEventNotesevOnderwerp.Text = m.Get("evonderwerp")
		End If

		If m.GetDefault("evaandacht", Null) = Null Then
			ABMGenEventNotesevAandacht.Text = "0"
		Else
			ABMGenEventNotesevAandacht.Text = m.Get("evaandacht")
		End If

		If m.GetDefault("evhot", Null) = Null Then
			ABMGenEventNotesevHot.State = False
		Else
			ABMGenEventNotesevHot.State = Not(m.Get("evhot")="0") ' to convert int to boolean
		End If

		If m.GetDefault("evcommentaar", Null) = Null Then
			ABMGenEventNotesevCommentaar.Text = ""
		Else
			ABMGenEventNotesevCommentaar.Text = m.Get("evcommentaar")
		End If

		IsNewEventNotes=False
		page.ShowModalSheet("EventNotes")
	Else
		Log("No record found")
	End If
End Sub

' the user clicked save the form
Sub ABMGenEventNotesSave_Clicked(Target As String)
	Dim ABMGenEventNotesModal As ABMModalSheet = page.ModalSheet("EventNotes")

	Dim Variables As List
	Variables.Initialize
	Dim NewDate As Long = DateTime.Now 'ignore

	Dim SQL As SQL = DBM.GetSQL
	Dim valueDouble As Double 'ignore
	Dim valueString As String 'ignore
```

```
	Dim valueInt As Int 'ignore
	Dim valueBoolean As Boolean 'ignore
	Dim valueLong As Long 'ignore
	Dim ret As Int 'ignore

	Dim LastStatus As Int ' needed to do special save
	Dim NeedsSaveAfsluiten As Boolean ' needed to do special save

	Dim SQL_chk As String = "SELECT evID, evStatus FROM tEvent WHERE evID = ? "
	Dim resChk As List = DBM.ABMGenSQLSelect(SQL, SQL_chk, Array As Int(ActiveEventNotesevID))
	If resChk.Size = 0 Then
		IsNewEventNotes =True
	Else
		Dim m As Map = resChk.Get(0)   ' needed to do special save
		LastStatus = m.Get("evstatus") ' needed to do special save
	End If

	If IsNewEventNotes Then
		Dim ABMGenEventNotesevStatus As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevStatus")
		Dim ABMGenEventNotesevType As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevType")
		Dim ABMGenEventNotesevToDoBy As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevToDoBy")
		Dim ABMGenEventNotesevToDoDatum As ABMDateTimeScroller = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevToDoDatum")
		Dim ABMGenEventNotesevOnderwerp As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevOnderwerp")
		Dim ABMGenEventNotesevAandacht As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevAandacht")
		Dim ABMGenEventNotesevHot As ABMSwitch = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevHot")
		Dim ABMGenEventNotesevCommentaar As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevCommentaar")

		valueDouble = ActivePersonID ' needed active person
		Variables.Add(valueDouble)

		valueDouble = ActiveContactId ' needed active contact
		Variables.Add(valueDouble)

		valueLong = NewDate
		Variables.Add(ABM.ConvertToDateTimeString(valueLong, "yyyy-MM-dd HH:mm:ss"))

		valueInt = ABMGenEventNotesevStatus.GetActiveItemId
		If ABMGenNotMin1(valueInt) = False Then
			page.ShowModalSheet("EventNotesBadInput")
			Return
		End If
		Variables.Add(valueInt)
		NeedsSaveAfsluiten = True ' needed for special save

		valueInt = ABMGenEventNotesevType.GetActiveItemId
		If ABMGenNotMin1(valueInt) = False Then
			page.ShowModalSheet("EventNotesBadInput")
			Return
		End If
		Variables.Add(valueInt)

		valueInt = ABMGenEventNotesevToDoBy.GetActiveItemId
		If ABMGenNotMin1(valueInt) = False Then
			page.ShowModalSheet("EventNotesBadInput")
			Return
		End If
		Variables.Add(valueInt)

		valueLong = ABMGenEventNotesevToDoDatum.GetDate
		Variables.Add(ABM.ConvertToDateTimeString(valueLong, "yyyy-MM-dd HH:mm:ss"))

		valueString = ABMGenEventNotesevOnderwerp.Text
		If ABMGenNotEmpty(valueString) = False Then
			page.ShowModalSheet("EventNotesBadInput")
			Return
		End If
		Variables.Add(valueString)

		valueDouble = ABMGenEventNotesevAandacht.Text
		Variables.Add(valueDouble)

		valueBoolean = ABMGenEventNotesevHot.State
		Variables.Add(valueBoolean)

		valueString = ABMGenEventNotesevCommentaar.Text
		If ABMGenNotEmpty(valueString) = False Then
			page.ShowModalSheet("EventNotesBadInput")
			Return
		End If
		Variables.Add(valueString)

		valueLong = NewDate
		Variables.Add(ABM.ConvertToDateTimeString(valueLong, "yyyy-MM-dd HH:mm:ss"))

		ret = DBM.ABMGenSQLInsert(SQL, "INSERT INTO tEvent(evCreator, evContact, evDatum, evStatus, evType, evToDoBy, evToDoDatum, evOnderwerp, evAandacht, evHot, evCommentaar, evAfsluitDatum) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", Variables)
	Else
		Dim ABMGenEventNotesevStatus As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevStatus")
		Dim ABMGenEventNotesevType As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevType")
		Dim ABMGenEventNotesevToDoBy As ABMCombo = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevToDoBy")
		Dim ABMGenEventNotesevToDoDatum As ABMDateTimeScroller = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevToDoDatum")
		Dim ABMGenEventNotesevOnderwerp As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevOnderwerp")
		Dim ABMGenEventNotesevAandacht As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevAandacht")
		Dim ABMGenEventNotesevHot As ABMSwitch = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevHot")
		Dim ABMGenEventNotesevCommentaar As ABMInput = ABMGenEventNotesModal.Content.Component("ABMGenEventNotesevCommentaar")

		valueInt = ABMGenEventNotesevStatus.GetActiveItemId
		If ABMGenNotMin1(valueInt) = False Then
			page.ShowModalSheet("EventNotesBadInput")
			Return
		End If
		Variables.Add(valueInt)
		If LastStatus <> 4 And valueInt = 4 Then ' neede for special save
			NeedsSaveAfsluiten = True
		End If

		valueInt = ABMGenEventNotesevType.GetActiveItemId
		If ABMGenNotMin1(valueInt) = False Then
			page.ShowModalSheet("EventNotesBadInput")
			Return
		End If
		Variables.Add(valueInt)

		valueInt = ABMGenEventNotesevToDoBy.GetActiveItemId
		If ABMGenNotMin1(valueInt) = False Then
			page.ShowModalSheet("EventNotesBadInput")
			Return
		End If
		Variables.Add(valueInt)

		valueLong = ABMGenEventNotesevToDoDatum.GetDate
		Variables.Add(ABM.ConvertToDateTimeString(valueLong, "yyyy-MM-dd HH:mm:ss"))

		valueString = ABMGenEventNotesevOnderwerp.Text
		If ABMGenNotEmpty(valueString) = False Then
			page.ShowModalSheet("EventNotesBadInput")
			Return
		End If
		Variables.Add(valueString)
```

```
                valueDouble = ABMGenEventNotesevAandacht.Text
                Variables.Add(valueDouble)

                valueBoolean = ABMGenEventNotesevHot.State
                Variables.Add(valueBoolean)

                valueString = ABMGenEventNotesevCommentaar.Text
                If ABMGenNotEmpty(valueString) = False Then
                        page.ShowModalSheet("EventNotesBadInput")
                        Return
                End If
                Variables.Add(valueString)

                If NeedsSaveAfsluiten Then ' needed for special save
                        valueLong = NewDate
                        Variables.Add(ABM.ConvertToDateTimeString(valueLong, "yyyy-MM-dd HH:mm:ss"))
                        ret = DBM.ABMGenSQLUpdate(SQL, "UPDATE tEvent SET evStatus=?, evType=?, evToDoBy=?, evToDoDatum=?, evOnderwerp=?, evAandacht=?, evHot=?, evCommentaar=?, evAfsluitDatum=? WHERE evID=" & ActiveEventNotesevID, Variables)
                Else
                        ret = DBM.ABMGenSQLUpdate(SQL, "UPDATE tEvent SET evStatus=?, evType=?, evToDoBy=?, evToDoDatum=?, evOnderwerp=?, evAandacht=?, evHot=?, evCommentaar=? WHERE evID=" & ActiveEventNotesevID, Variables)
                End If
        End If
        DBM.CloseSQL(SQL)
        page.CloseModalSheet("EventNotes")

        ActiveEventNotesevID = 0

        ' reload pages with notes
        SearchInfo ' reload the notes on the previous page

        If NeedsSaveAfsluiten Then ' needed for special save
                page.ShowModalSheet("MakeNewToDo")
        End If
End Sub

' the user clicked on cancel on the save form
Sub ABMGenEventNotesCancel_Clicked(Target As String)
        ActiveEventNotesevID = 0
        page.CloseModalSheet("EventNotes")
End Sub

' method to check if a certain field is valid
Sub ABMGenNotEmpty(value As String) As Boolean
        ' TODO by the programmer!
        Return Not(value = "")
End Sub

Sub ABMGenNotMin1(value As Int) As Boolean
        ' TODO by the programmer!
        Return Not(value = -1)
End Sub

Sub ABMGenBuildEventNotesDelete() As ABMModalSheet
        Dim ABMGenEventNotesDeleteModal As ABMModalSheet
        ABMGenEventNotesDeleteModal.Initialize(page, "EventNotesDelete", False, False, "")
        ABMGenEventNotesDeleteModal.IsDismissible = False

        ' Build the content grid
        ABMGenEventNotesDeleteModal.Content.AddRowsM(1, True,0,0, "").AddCells12(1, "")
        ABMGenEventNotesDeleteModal.Content.BuildGrid  'IMPORTANT once you loaded the complete grid AND before you start adding components

        ' add message label
        Dim ABMGenEventNotesDeleteMessage As ABMLabel
        ABMGenEventNotesDeleteMessage.Initialize(page, "ABMGenEventNotesDeleteMessage", "Ben je zeker dat je deze event wilt wissen?" , ABM.SIZE_PARAGRAPH, False, "")
        ABMGenEventNotesDeleteModal.Content.Cell1R(0,1).AddComponent(ABMGenEventNotesDeleteMessage)

        ' Build the footer grid
        ABMGenEventNotesDeleteModal.Footer.AddRowsM(1,True,0,0, "").AddCellsOS(1,6,6,6,3,3,3,"").AddCellsOS(1,0,0,0,3,3,3, "")
        ABMGenEventNotesDeleteModal.Footer.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

        ' create the buttons for the footer
        Dim ABMGenEventNotesDeleteYes As ABMButton
        ABMGenEventNotesDeleteYes.InitializeFlat(page, "ABMGenEventNotesDeleteYes", "", "", "Ja", "transparent")
        ABMGenEventNotesDeleteModal.Footer.Cell(1,1).AddComponent(ABMGenEventNotesDeleteYes)

        Dim ABMGenEventNotesDeleteNo As ABMButton
        ABMGenEventNotesDeleteNo.InitializeFlat(page, "ABMGenEventNotesDeleteNo", "", "", "Nee", "transparent")
        ABMGenEventNotesDeleteModal.Footer.Cell(1,2).AddComponent(ABMGenEventNotesDeleteNo)

        Return ABMGenEventNotesDeleteModal
End Sub

' method you can call when the user wants to delete a record
Sub ABMGenEventNotesDelete(deleteId As Int) 'ignore
        ActiveEventNotesevID = deleteId
        page.ShowModalSheet("EventNotesDelete")
End Sub

' the user clicked yes on the delete messagebox
Sub ABMGenEventNotesDeleteYes_Clicked(Target As String)
        Dim SQL As SQL = DBM.GetSQL
        DBM.ABMGenSQLDelete(SQL, "DELETE FROM tEvent WHERE evID = ? ", Array As Int(ActiveEventNotesevID))
        DBM.CloseSQL(SQL)
        page.CloseModalSheet("EventNotesDelete")
        ActiveEventNotesevID = 0
End Sub

Sub ABMGenEventNotesDeleteNo_Clicked(Target As String)
        page.CloseModalSheet("EventNotesDelete")
End Sub

Sub ABMGenBuildEventNotesBadInput() As ABMModalSheet
        Dim ABMGenEventNotesBadInputModal As ABMModalSheet
        ABMGenEventNotesBadInputModal.Initialize(page, "EventNotesBadInput", False, False, "")
        ABMGenEventNotesBadInputModal.IsDismissible = False

        ' Build the content grid
        ABMGenEventNotesBadInputModal.Content.AddRowsM(1, True,0,0, "").AddCells12(1, "")
        ABMGenEventNotesBadInputModal.Content.BuildGrid  'IMPORTANT once you loaded the complete grid AND before you start adding components

        ' add message label
        Dim ABMGenEventNotesBadInputMessage As ABMLabel
        ABMGenEventNotesBadInputMessage.Initialize(page, "ABMGenEventNotesBadInputMessage", "Gelieve eerst alle velden in te vullen!" , ABM.SIZE_PARAGRAPH, False, "")
        ABMGenEventNotesBadInputModal.Content.Cell1R(0,1).AddComponent(ABMGenEventNotesBadInputMessage)

        ' Build the footer grid
        ABMGenEventNotesBadInputModal.Footer.AddRowsM(1,True,0,0, "").AddCellsOS(1,6,6,6,3,3,3,"").AddCellsOS(1,0,0,0,3,3,3, "")
        ABMGenEventNotesBadInputModal.Footer.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

        ' create the buttons for the footer
        Dim ABMGenEventNotesBadInputCancel As ABMButton
        ABMGenEventNotesBadInputCancel.InitializeFlat(page, "ABMGenEventNotesBadInputCancel", "", "", "Sluiten", "transparent")
        ABMGenEventNotesBadInputModal.Footer.Cell(1,2).AddComponent(ABMGenEventNotesBadInputCancel)

        Return ABMGenEventNotesBadInputModal
End Sub

' the user clicked ok on the Bad Input messagebox
```

```
Sub ABMGenEventNotesBadInputCancel_Clicked(Target As String)
	page.CloseModalSheet("EventNotesBadInput")
End Sub

Sub ABMGenBuildEventNotesComboItem(id As String, title As String) As ABMLabel 'ignore
	Dim ABMGenLabel As ABMLabel
	ABMGenLabel.Initialize(page, id, "(NBSP)" & title, ABM.SIZE_H6, True, "")
	ABMGenLabel.VerticalAlign = True
	Return ABMGenLabel
End Sub
#End Region

#Region MakeNewToDo
Sub ABMGenBuildMakeNewToDo() As ABMModalSheet
	Dim ABMGenMakeNewToDoModal As ABMModalSheet
	ABMGenMakeNewToDoModal.Initialize(page, "MakeNewToDo", False, False, "")
	ABMGenMakeNewToDoModal.IsDismissible = False

	' Build the content grid
	ABMGenMakeNewToDoModal.Content.AddRowsM(1, True,0,0, "").AddCells12(1, "")
	ABMGenMakeNewToDoModal.Content.BuildGrid  'IMPORTANT once you loaded the complete grid AND before you start adding components

	' add message label
	Dim ABMGenMakeNewToDoMessage As ABMLabel
	ABMGenMakeNewToDoMessage.Initialize(page, "ABMGenMakeNewToDoMessage", "Een nieuw event aanmaken?" , ABM.SIZE_PARAGRAPH, False, "")
	ABMGenMakeNewToDoModal.Content.CellR(0,1).AddComponent(ABMGenMakeNewToDoMessage)

	' Build the footer grid
	ABMGenMakeNewToDoModal.Footer.AddRowsM(1,True,0,0, "").AddCellsOS(1,6,6,6,3,3,3,"").AddCellsOS(1,0,0,0,3,3,3, "")
	ABMGenMakeNewToDoModal.Footer.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

	' create the buttons for the footer
	Dim ABMGenMakeNewToDoYes As ABMButton
	ABMGenMakeNewToDoYes.InitializeFlat(page, "ABMGenMakeNewToDoYes", "", "", "Ja", "transparent")
	ABMGenMakeNewToDoModal.Footer.Cell(1,1).AddComponent(ABMGenMakeNewToDoYes)

	Dim ABMGenMakeNewToDoNo As ABMButton
	ABMGenMakeNewToDoNo.InitializeFlat(page, "ABMGenMakeNewToDoNo", "", "", "Nee", "transparent")
	ABMGenMakeNewToDoModal.Footer.Cell(1,2).AddComponent(ABMGenMakeNewToDoNo)

	Return ABMGenMakeNewToDoModal
End Sub

Sub ABMGenMakeNewToDoYes_Clicked(Target As String)
	page.CloseModalSheet("MakeNewToDo")
	ABMGenEventNotesNew
End Sub

Sub ABMGenMakeNewToDoNo_Clicked(Target As String)
	page.CloseModalSheet("MakeNewToDo")
End Sub
#End Region

'------------------------------------------------------------
' Add a module, called DBM
'------------------------------------------------------------
Sub Process_Globals
	Private pool As ConnectionPool
	Private SQLite As SQL
	Private DatabaseType As Int

	Public DBOK As Int = 0
	Public DBERROR As Int = -1
	Public DBNORESULTS As Int = -2
End Sub

Sub InitializeSQLite(Dir As String, fileName As String, createIfNeeded As Boolean) 'ignore
	SQLite.InitializeSQLite(Dir, fileName, createIfNeeded)
	DatabaseType = 0
End Sub

Sub InitializeMySQL(jdbcUrl As String ,login As String, password As String, poolSize As Int) 'ignore
	DatabaseType = 1
	Try
		pool.Initialize("com.mysql.jdbc.Driver", jdbcUrl, login, password)
	Catch
		Log("Last Pool Init Except: " & LastException.Message)
	End Try

	' change pool size...
	Dim jo As JavaObject = pool
	jo.RunMethod("setMaxPoolSize", Array(poolSize))
End Sub

Sub InitializeMSSQL(jdbcUrl As String ,login As String, password As String, poolSize As Int) 'ignore
	DatabaseType = 2
	Try
		pool.Initialize("net.sourceforge.jtds.jdbc.Driver", jdbcUrl, login, password)
	Catch
		Log("Last Pool Init Except: " & LastException.Message)
	End Try

	' change pool size...
	Dim jo As JavaObject = pool
	jo.RunMethod("setMaxPoolSize", Array(poolSize))
End Sub

Sub GetSQL() As SQL
	If DatabaseType = 0 Then
		Return SQLite
	Else
		Return pool.GetConnection
	End If
End Sub

Sub CloseSQL(mySQL As SQL)
	If DatabaseType <> 0 Then
		mySQL.Close
	End If
End Sub

Sub ABMGenSQLSelect(SQL As SQL, Query As String, args As List) As List
	Dim l As List
	l.Initialize
	Dim cur As ResultSet
	Try
		cur = SQL.ExecQuery2(Query, args)
	Catch
		Log(LastException)
		Return l
	End Try
	Do While cur.NextRow
		Dim res As Map
		res.Initialize
		For i = 0 To cur.ColumnCount - 1
			res.Put(cur.GetColumnName(i).ToLowerCase, cur.GetString2(i))
		Next
		l.Add(res)
```

```
        Loop
        cur.Close
        Return l
End Sub

Sub ABMGenSQLDelete(SQL As SQL, Query As String, args As List) As Int
        Dim res As Int
        Try
                SQL.ExecNonQuery2(Query, args)
                res = DBOK
        Catch
                Log(LastException)
                res = DBERROR
        End Try
        Return res
End Sub

Sub ABMGenSQLInsert(SQL As SQL, Query As String, Args As List) As Int
        Dim res As Int
        Try
                Select Case DatabaseType
                Case 0
                        SQL.ExecNonQuery2(Query, Args)
                        res     = ABMGenSQLSelectSingleResult(SQL, "SELECT last_insert_rowid()", Null)
                Case 1
                        SQL.ExecNonQuery2(Query, Args)
                        res = ABMGenSQLSelectSingleResult(SQL, "SELECT LAST_INSERT_ID()", Null)
                Case 2
                        Dim ABSQL As ABSQLExt
                        res = ABSQL.ExecNonQuery3(SQL, Query, Args)
                End Select
        Catch
                Log(LastException)
                res = DBERROR
        End Try
        Return res
End Sub

Sub ABMGenSQLUpdate(SQL As SQL, Query As String, Args As List) As Int
        Dim res As Int
        Try
                SQL.ExecNonQuery2(Query, Args)
                res = DBOK
        Catch
                Log(LastException)
                res = DBERROR
        End Try
        Return res
End Sub

Sub ABMGenSQLSelectSingleResult(SQL As SQL, Query As String, args As List) As String
        Dim res As String
        Try
                res = SQL.ExecQuerySingleResult2(Query, args)
        Catch
                Log(LastException)
                res = DBERROR
        End Try
        If res = Null Then
                res = DBNORESULTS
        End If
        Return res
End Sub
```