

CreatingPpt - 1

Option Explicit

'Makes all text comparisons in this module case inSenSeTive...

Option Compare Text

Private Const m\_MODULE\_NAME As String = "CreatingPpt"

Private Const m\_SLIDE\_CREATION\_HDR As Byte = 1

Private Const m\_SLIDE\_CREATION\_LAYOUT As Byte = 2

Private Const m\_SLIDE\_CREATION\_CHRT1\_POSTN As Byte = 3

Private Const m\_SLIDE\_CREATION\_COPY\_RNG\_1 As Byte = 4

Private Const m\_SLIDE\_CREATION\_COPY\_RNG\_2 As Byte = 5

Private Const m\_FIELDS\_SLIDE\_CREATION As Byte = 5

Private m\_dicPreExistingSlides As New Dictionary

Private m\_rngMppngTblData As Range

Private m\_rgnListWshSlideInfo As Range

Private m\_rngListlInclTF As Range

Private m\_rngMacroInputVals As Range

Private m\_rngListRiskNo As Range

Private m\_ppApp As PowerPoint.Application

Private m\_ppPres As Presentation

Private Enum ModWshType

modDynamic = 1

modStatic = 2

modUnspecified = 3

End Enum

Private m\_arrxlMppngTblData() As Variant

Private m\_arrxlListWshSlideInfo() As Variant

Private m\_arrxlInclTF As Variant

Private m\_arrxlMacroInputs As Variant

Private m\_arrxlRiskNo As Variant

Private m\_mpgcolWshMain As Byte

Private m\_mpgcolWshType As Byte

Private m\_mpgcolDrpDnNm As Byte

Private m\_mpgcolListFillRngNm As Byte

Private m\_mpgcolLnkdCellNm As Byte

Private m\_mpgcolWshList As Byte

Private m\_mpgcolListRngInclTF As Byte

Private m\_mpgcolListRngDpDnVals As Byte

Private m\_mpgcolListRngRskNos As Byte

Private m\_mpgcolMacroNm As Byte

Private m\_mpgcolSlideHdrStat As Byte

Private m\_mpgcolSlideHdrRngDyn As Byte

Private m\_mpgcolPptLyoutTypeRng As Byte

Private m\_mpgcolPostnCopyRng1 As Byte

Private m\_mpgcolCopyRng1 As Byte

Private m\_mpgcolCopyRng2 As Byte

Private m\_wshsMissing As Boolean

'=====

Public Sub Browse()

With Application.FileDialog(msoFileDialogFilePicker)

If .Show = -1 Then WshToDoList.Range("rngSavePath") = .SelectedItems(1)

End With

End Sub

'=====

Public Sub GeneratePptMaestro()

Const METHOD\_NM As String = "GeneratePptMaestro"

Dim rngSheetNmTopDataCell As Range

Dim m\_rngMppngTblData As Range

Dim targetPptFileFullNm As String

Dim calledProcRanOK As Boolean

Dim wshVsbiltyCmlsOutptOrig As XlSheetVisibility

Dim wshVsbiltyRskMtrxOutptOrig As XlSheetVisibility

Dim wshVsbiltyRskFctrOutptOrig As XlSheetVisibility

Dim wshVsbiltyRskRtioSumTblOrig As XlSheetVisibility

Dim wshVsbiltyPeerDataTblOrig As XlSheetVisibility

```

Application.ScreenUpdating = False
Application.DisplayAlerts = False

On Error GoTo ErrHandler

'Capture version...
g_xlVersion = Application.Version

'For debugging...
WshToDoList.Activate

'Set up MsgBoxHandler...
Set g_Mbh = New MsgBoxHandler

On Error Resume Next
Dim ws As Worksheet
For Each ws In ThisWorkbook.Worksheets
    ws.Unprotect g_PWD_GENL
    Debug.Print "Unprotected wsh: " & ws.Name
Next ws

On Error GoTo 0

'Capture state...
wshVsbiltyCmlsOutptOrig = WshCamelsOutpt.Visible
wshVsbiltyRskMtrxOutptOrig = WshRiskMtrxOutpt.Visible
wshVsbiltyRskFctrOutptOrig = WshRiskFctrOutpt.Visible
wshVsbiltyRskRtioSumTblOrig = WshRiskRatioSummTbl.Visible
wshVsbiltyPeerDataTblOrig = WshPeerDataTbl.Visible

'Make select sheets visible...
WshCamelsOutpt.Visible = xlSheetVisible
WshRiskMtrxOutpt.Visible = xlSheetVisible
WshRiskFctrOutpt.Visible = xlSheetVisible
WshRiskRatioSummTbl.Visible = xlSheetVisible
WshPeerDataTbl.Visible = xlSheetVisible

targetPptFileFullNm = Trim(Names.Item("rngSavePath").RefersToRange)

'Call main sub-routine, trapping for invalid PPT file name...
Dim dbgHaveValidPpt As Boolean
dbgHaveValidPpt = _
    targetPptFileFullNm <> "" And InStr(targetPptFileFullNm, ".ppt") > 0
If dbgHaveValidPpt Then
    UpdateExistingPowerPoint _
        pptFileFullNm:=targetPptFileFullNm, _
        subRanOkOUT:=calledProcRanOK "... handled under ExitPoint
Else
    MsgBox "Please select PowerPoint file with browse button.", vbCritical, _
        "Invalid Power Point File"
    GoTo ExitPoint
End If

ExitPoint: '...handle clean-up
Application.StatusBar = vbNullString
'Restore wsh visibility...
WshCamelsOutpt.Visible = wshVsbiltyCmlsOutptOrig
WshRiskMtrxOutpt.Visible = wshVsbiltyRskMtrxOutptOrig
WshRiskFctrOutpt.Visible = wshVsbiltyRskFctrOutptOrig
WshRiskRatioSummTbl.Visible = wshVsbiltyRskRtioSumTblOrig
WshPeerDataTbl.Visible = wshVsbiltyPeerDataTblOrig

If Not m_ppApp Is Nothing Then m_ppApp.Quit '...in case we hit an error
Application.ScreenUpdating = True
Application.DisplayAlerts = True
'So user can see cells highlighted in red...
If m_wshsMissing Or Not calledProcRanOK Then WshMppng.Activate
'Clean-up messages...
If calledProcRanOK Then
    Select Case g_nمبرProbSlides
        Case 0
            m_ppPres.Save
    
```

```

        WshToDoList.Activate
        g_Mbh.SetGoodFinishMsg
        g_Mbh.ShowMsgBoxNoAnswer vbOKOnly
    Case Is < g_MAX_ALLOWED_BAD_SLIDES
        m_ppPres.Save
        WshMppng.Activate
        g_Mbh.SetDoneButSlidesSkippedMsg
        g_Mbh.ShowMsgBoxNoAnswer vbOKOnly
    Case Else
        WshMppng.Activate
        g_Mbh.SetStoppedForTooManyBadSlidesMsg
        g_Mbh.ShowMsgBoxNoAnswer vbCritical
    End Select
End If

Exit Sub

ErrorHandler:
    g_Mbh.SetUnhandledErrMsg Err.Description, METHOD_NM, m_MODULE_NAME
    g_Mbh.ShowMsgBoxNoAnswer vbCritical
    calledProcRanOK = False
    GoTo ExitPoint
End Sub

```

```

=====
Private Sub UpdateExistingPowerPoint( _
    ByVal pptFileFullNm As String, _
    subRanOkOUT As Boolean)
Const METHOD_NM As String = "UpdateExistingPowerPoint"
Dim pptCustLayout As CustomLayout
Dim dicNewAndUpdtdSlides As New Dictionary
Dim pptSlide As PowerPoint.Slide
Dim pptShape As PowerPoint.Shape
Dim enumSlideLayout As ProjSlideLayout
Dim enumWshType As ModWshType
Dim wsMain As Worksheet
Dim wsList As Worksheet
Dim rngCopySrc1 As Range
Dim rngCopySrc2 As Range
Dim cellInclTF As Range
Dim cellProbInput As Range
Dim nmWshMain As String
Dim nmWshList As String
Dim addrCopyRng01FmMpgTbl As String
Dim addrCopyRng02FmMpgTbl As String
Dim addrInclTFRng As String
Dim addrRskNosRng As String
Dim addrCopyRng01FmLstWsh As String
Dim addrCopyRng02FmLstWsh As String
Dim addrDynSlideHdrs As String
Dim nmListFillRng As String
Dim macroNm As String
Dim slideNm As String
Dim sIncl As String
Dim mcroInput As String
Dim riskNmbr As String
Dim nmbrMacroInputs As Integer
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim errMsgPromptGenerated As Boolean
Dim calledProcRanOK As Boolean
Dim dbgIncl As Boolean
Dim dbgSlideExists As Boolean
Dim dbgHaveErr As Boolean

```

```
subRanOkOUT = True '... until proven otherwise
```

```

'Create an instance of PowerPoint...
On Error Resume Next
Set m_ppApp = New PowerPoint.Application
Application.Wait DateAdd("s", 3, Now)
m_ppApp.Visible = msoTrue
'Handle any error...

```

```

dbgHaveErr = Err.number <> 0
If dbgHaveErr Then
    g_Mbh.SetCannotOpenPptMsg
    g_Mbh.ShowMsgBoxNoAnswer vbCritical
    errMsgPromptGenerated = True
    GoTo ErrorHandler
End If

'Open PPT file...
On Error Resume Next
Set m_ppPres = m_ppApp.Presentations.Open(pptFileFullNm)
Application.Wait DateAdd("s", 3, Now)
'Handle any error...
dbgHaveErr = Err.number <> 0
If dbgHaveErr Then
    g_Mbh.SetCannotOpenPptFileMsg pptFileFullNm
    g_Mbh.ShowMsgBoxNoAnswer vbCritical
    errMsgPromptGenerated = True
    GoTo ErrorHandler
End If

On Error GoTo ErrorHandler

'Populate slides dictionary (value immaterial)...
For Each pptSlide In m_ppPres.Slides
    m_dicPreExistingSlides.Item(pptSlide.Name) = 0
Next pptSlide

WshMppng.Activate

'Instantiate mapping data variables...
Set m_rngMppngTblData = Names("rgndoMappingSpecs").RefersToRange
m_arrxlMppngTblData = m_rngMppngTblData.Value

DefineMappingColNosModuleVariables

'Trap for missing worksheets...
FlagMissingWshs
If m_wshsMissing Then
    errMsgPromptGenerated = True
    g_Mbh.SetMissingWshsMsg
    g_Mbh.ShowMsgBoxNoAnswer vbCritical
    Err.Raise g_lERR_HANDLED
End If

'Handle any disparities between the number of rows in the List Wsh _
slide info ranges and the number of rows the SHOULD hold - which we _
determine by checking the number of rows in the relevant _
drop-down's List Fill range...
FlagInconsistentDynamicWshRanges _
subRanOkOUT:=calledProcRanOK _
If Not calledProcRanOK Then
    g_Mbh.SetInconsistentWshListRangesMsg
    g_Mbh.ShowMsgBoxNoAnswer vbCritical
    errMsgPromptGenerated = True
    GoTo ErrorHandler
End If

'Process the mappings...
For i = 1 To m_rngMppngTblData.Rows.Count
    '*****
    '*** Define & instantiate some variables unique to each mapping...
    '*****
    'Set main wsh...
    nmWshMain = Trim(m_arrxlMppngTblData(i, m_mpgcolWshMain))
    Set wsMain = ThisWorkbook.Worksheets(nmWshMain)

    'Wsh type enum...
    enumWshType = GetWshTypeEnum(i)

    '*****
    '*** Now handle matters based on Dynamic vs. Static...

```

```

'*****
Select Case enumWshType
  Case modDynamic
    'Reset variables unique to Dynamic sheets...
    Set wsList = Nothing
    nmWshList = vbNullString
    nmListFillRng = vbNullString
    addrInclTFRng = vbNullString
    addrRskNosRng = vbNullString
    macroNm = vbNullString
    addrCopyRng01FmLstWsh = vbNullString
    addrCopyRng02FmLstWsh = vbNullString
    addrDynSlideHdrs = vbNullString
    nmbrMacroInputs = 0

    'Define List wsh...
    nmWshList = Trim(m_arrxlMppngTblData(i, m_mpgcolWshList))
    Set wsList = ThisWorkbook.Worksheets(nmWshList)

    'Define PPT slide creation range & array...
    'NOTE...
    '*****
    'For Dyanmic slides, the List wsh contains 5 regions of
    data which are the information needed to create a PPT slide:
    1) Slide header
    2) PPT slide layout type code (1-4)
    3) Chart position for chart 1 (up, down, L, R)
    4) Copy range address #1
    5) Copy range address #2
    The address of these 5 regions are individually identified
    in the mapping table.
    'These 5 regions of data are always in the same order (L to R)
    on their appropriate List wsh, are are always in a contiguous
    block. Therefore, rather than define 5 individual ranges
    for the 5 pieces of information, we are:
    1) Identifying one single, module-level, 5-column range
    as the range containing all info we need to populate
    a PPT slide from a dyanmic worksheet.
    2) Specifying module-level constants
    (e.g., m_FIELDS_SLIDE_CREATION) for identifying which
    column in that range contains our desired piece
    of information.
    'As an FYI, we define this range of PPT slide-creation
    in a two-step process:
    1) We first setting this range = the 1-column range of
    slide headers
    2) Then resize the range horizontally by a module-level
    constant (m_FIELDS_SLIDE_CREATION), which, of course, = 5.
    Lastly, of course, we store the values of this range in
    a module-level array.
    'NOTE: This range, and its commensurate array, are
    (of course) redefined for each dynamic wsh in the mapping table.
    '*****
    addrDynSlideHdrs = m_arrxlMppngTblData(i, m_mpgcolSlideHdrRngDyn)
    '2-step definition...
    Set m_rgnListWshSlideInfo = wsList.Range(addrDynSlideHdrs)
    Set m_rgnListWshSlideInfo =
      m_rgnListWshSlideInfo.Resize(
        columnsize:=m_FIELDS_SLIDE_CREATION)
    m_arrxlListWshSlideInfo = m_rgnListWshSlideInfo.Value

    'Instantiate some string variables...
    nmListFillRng = Trim(m_arrxlMppngTblData(i, m_mpgcolListFillRngNm))
    addrInclTFRng = Trim(m_arrxlMppngTblData(i, m_mpgcolListRngInclTF))
    addrRskNosRng =
      Trim(m_arrxlMppngTblData(i, m_mpgcolListRngRskNos))
    macroNm = Trim(m_arrxlMppngTblData(i, m_mpgcolMacroNm))

    'Instantiate remaining variables (we are assuming there will
    be no errors here)...
    Set m_rngMacroInputVals = Names(nmListFillRng).RefersToRange
    Set m_rngListlInclTF = wsList.Range(addrInclTFRng)
    Set m_rngListRiskNo = wsList.Range(addrRskNosRng)

```

```

m_arrxlInclTF = m_rngList1InclTF.Value2
m_arrxlMacroInputs = m_rngMacroInputVals.Value2
m_arrxlRiskNo = m_rngListRiskNo.Value2
nmbrMacroInputs = m_rngMacroInputVals.Rows.Count

Application.EnableEvents = False

'*****
'*** Loop through each drop-down record...
'*****
For j = 1 To nmbrMacroInputs
  On Error GoTo NextListWshRecord
  'Reset variables...
  Set cellInclTF = Nothing
  sIncl = vbNullString
  mcroInput = vbNullString
  riskNmbr = vbNullString

  'If row/record is not flagged as Include then skip _
  to the next drop-down entry...
  sIncl = Trim(m_arrxlInclTF(j, 1))
  If StrComp(sIncl, vbNullString) = 0 _
    Or StrComp(sIncl, "FALSE") = 0 Then
    dbgIncl = False
  Else
    dbgIncl = True
  End If
  If Not dbgIncl Then GoTo NextListWshRecord

  '*** If here then we have an Include value = TRUE...
  'Instantiate some variables (we are assuming there will be _
  no errors here)...
  mcroInput = Trim(m_arrxlMacroInputs(j, 1))
  'Skip if the macro input is blank...
  If StrComp(mcroInput, vbNullString) = 0 Then _
    GoTo NextListWshRecord
  riskNmbr = Trim(m_arrxlRiskNo(j, 1))
  addrCopyRng01FmMpgTbl = _
    Trim(m_arrxlMppngTblData(i, m_mpgcolCopyRng1))

  'Define slide name, add to dictionary...
  'Note: chr(95) = underscore (i.e., "_")
  slideNm = _
    riskNmbr & Chr(95) & _
    nmWshMain & Chr(95) & _
    mcroInput & Chr(95) & _
    addrCopyRng01FmMpgTbl
  dicNewAndUpdtdSlides.Item(slideNm) = 0

  'Error check the slide info range...
  CheckForAndFlagErrorsInSlideCreationInfoRange _
    mappingTblRecNo:=i, _
    infoRngRecNo:=j, _
    subRanOkOUT:=calledProcRanOK
  If Not calledProcRanOK Then
    LogProblemSlide (slideNm)
    Err.Clear
    If g_nmbrProbSlides < g_MAX_ALLOWABLE_BAD_SLIDES Then
      GoTo NextListWshRecord
    Else
      GoTo ExitPoint '...stop updating PPT
    End If
  End If
End If

Set cellProbInput = Nothing '... reset
RunDynamicWshMacro _
  mppngTblRecNo:=i, _
  listWshRecNo:=j, _
  subRanOkOUT:=calledProcRanOK
If Not calledProcRanOK Then
  wsList.Unprotect g_PWD_SPECIAL '... unprotect the worksheet
  'Flag the input cell...
  Set cellProbInput = m_rngMacroInputVals.Cells(j, 1)

```

```

        cellProbInput.Interior.Color = vbRed
        wsList.Protect g_PWD_SPECIAL '...restore protection
        Err.Raise g_lERR_DYN_MACRO '...custom err nmbr
    End If
    Application.Wait DateAdd("s", 1, Now)

    'Get slide layout enum...
    enumSlideLayout = GetPptSlideLayoutEnum( _
        mppngTblRecNo:=i, _
        wshListRecNo:=j)

    PrepExistingSlideOrCreateNewForPastingPic _
        slideNm:=slideNm, _
        mppngTblRecNo:=i, _
        subRanOkOUT:=calledProcRanOK, _
        wshListRecNo:=j
    If Not calledProcRanOK Then Err.Raise g_lERR_HANDLED

    PopulateSlide _
        wshTypeEnum:=enumWshType, _
        sldNm:=slideNm, _
        mppngTblRecNo:=i, _
        subRanOkOUT:=calledProcRanOK, _
        listWshRecNo:=j
    If Not calledProcRanOK Then Err.Raise g_lERR_HANDLED
    Application.Wait DateAdd("s", 1, Now) '... pause 1 second

NextListWshRecord:
If Err.number <> 0 Then
    LogProblemSlide (slideNm)
    Err.Clear
    If g_nمبرProbSlides >= g_MAX_ALLOWABLE_BAD_SLIDES Then GoTo ExitPoint
End If

    Next j

Case modStatic
    On Error GoTo NextMainWsh

    'Define slide name, add to dictionary...
        'Note: chr(124) = pipe (i.e., "|")
    slideNm = "1_" & nmWshMain & Chr(124) & addrCopyRng01FmMpgTbl
    dicNewAndUpdtdSlides.Item(slideNm) = 0

    'Get slide layout enum...
    enumSlideLayout = GetPptSlideLayoutEnum(mppngTblRecNo:=i)

    PrepExistingSlideOrCreateNewForPastingPic _
        slideNm:=slideNm, _
        subRanOkOUT:=calledProcRanOK, _
        mppngTblRecNo:=i

    PopulateSlide _
        wshTypeEnum:=ModWshType.modStatic, _
        sldNm:=slideNm, _
        mppngTblRecNo:=i, _
        subRanOkOUT:=calledProcRanOK
    If Not calledProcRanOK Then GoTo NextMainWsh
End Select
Application.Wait DateAdd("s", 1, Now) '... pause 1 second

NextMainWsh:
If Err.number <> 0 Then
    LogProblemSlide (slideNm)
    Err.Clear
    If g_nمبرProbSlides >= g_MAX_ALLOWABLE_BAD_SLIDES Then GoTo ExitPoint
End If

Next i

On Error GoTo 0

'Delete any slide from PPT that we have NOT updated and which has _

```

```

an underscore in its name...
Dim arrPreviouslyExistingSlideNames() As Variant
Dim prevXstngSldNm As String
Dim dbgSldNmHasUndrscr As Boolean
arrPreviouslyExistingSlideNames = m_dicPreExistingSlides.Keys
For i = 1 To m_dicPreExistingSlides.Count
    dbgSldNmHasUndrscr = False '... reset
    prevXstngSldNm = arrPreviouslyExistingSlideNames(i - 1)
    'See if a previously existing slide has been updated...
    Dim dbgSlideUpdated As Boolean
    dbgSlideUpdated = dicNewAndUpdtdSlides.exists(prevXstngSldNm)
    If Not dbgSlideUpdated Then
        'See if slide name contains underscore...
        'Note: chr(95) = underscore (i.e., "_")
        dbgSldNmHasUndrscr = InStr(1, prevXstngSldNm, Chr(95)) > 0
        If dbgSldNmHasUndrscr Then m_ppPres.Slides(prevXstngSldNm).Delete
    End If
Next i

ExitPoint:
Application.EnableEvents = True
Exit Sub

ErrorHandler:
subRanOkOUT = False
If Not errMsgPromptGenerated Then
    g_Mbh.SetUnhandledErrMsg Err.Description, METHOD_NM, m_MODULE_NAME
    g_Mbh.ShowMsgBoxNoAnswer vbCritical
End If
GoTo ExitPoint

End Sub

'=====
Private Sub DefineMappingColNosModuleVariables()
Dim mppngTblLeftMostColNmbr As Byte

'We do this because the table does not begin in column A...
mppngTblLeftMostColNmbr = m_rngMppngTblData.Column

m_mpgcolWshMain = 1 + WshMppng.Range("colWshNm").Column - _
    mppngTblLeftMostColNmbr
m_mpgcolWshType = _
    1 + WshMppng.Range("colWshType").Column - mppngTblLeftMostColNmbr
m_mpgcolDrpDnNm = _
    1 + WshMppng.Range("colDrpDwnNm").Column - mppngTblLeftMostColNmbr
m_mpgcolListFillRngNm = _
    1 + WshMppng.Range("colListFillRngNm").Column - mppngTblLeftMostColNmbr
m_mpgcolLnkdCellNm = _
    1 + WshMppng.Range("colLnkdCellNm").Column - mppngTblLeftMostColNmbr
m_mpgcolWshList = _
    1 + WshMppng.Range("colListSheetNm").Column - mppngTblLeftMostColNmbr
m_mpgcolListRngInclTF = _
    1 + WshMppng.Range("colListRngTF").Column - mppngTblLeftMostColNmbr
m_mpgcolListRngDpDnVals = _
    1 + WshMppng.Range("colListRngDrpDnVals").Column - mppngTblLeftMostColNmbr
m_mpgcolListRngRskNos = _
    1 + WshMppng.Range("colListRngRiskNmbr").Column - mppngTblLeftMostColNmbr
m_mpgcolMacroNm = _
    1 + WshMppng.Range("colMacro").Column - mppngTblLeftMostColNmbr
m_mpgcolSlideHdrStat = _
    1 + WshMppng.Range("colSlideHdrStatic").Column - mppngTblLeftMostColNmbr
m_mpgcolSlideHdrRngDyn = _
    1 + WshMppng.Range("colSlideHdrRngDynmc").Column - mppngTblLeftMostColNmbr
m_mpgcolPptLayoutTypeRng = _
    1 + WshMppng.Range("colPptLayoutTypeRng").Column - mppngTblLeftMostColNmbr
m_mpgcolPostnCopyRng1 = _
    1 + WshMppng.Range("colPostnCopyRng1").Column - mppngTblLeftMostColNmbr
m_mpgcolCopyRng1 = _
    1 + WshMppng.Range("colCopyRng1").Column - mppngTblLeftMostColNmbr
m_mpgcolCopyRng2 = _
    1 + WshMppng.Range("colCopyRng2").Column - mppngTblLeftMostColNmbr

End Sub

```

```

=====
Private Sub FlagMissingWshs()
Dim cellProb As Range
Dim wsNm As String
Dim dbgWshExists As Boolean
'Required as input for WshExists function, but not subsequently used...
Dim invalidWbkNm As Boolean
Dim i As Integer

'Main wshs...
For i = 1 To m_rngMppngTblData.Rows.Count
wsNm = Trim(m_arrxlMppngTblData(i, m_mpgcolWshMain))
dbgWshExists = Utilities.WshExists(wsNm, invalidWbkNm)
If Not dbgWshExists Then
m_wshsMissing = True
'Flag problem wsh name...
m_rngMppngTblData.Cells(i, m_mpgcolWshMain).Interior.Color = vbRed
End If
Next i

'List wshs...
For i = 1 To m_rngMppngTblData.Rows.Count
wsNm = Trim(m_arrxlMppngTblData(i, m_mpgcolWshList))
'Because there are not always List wshs...
If StrComp(wsNm, vbNullString) = 0 Then GoTo NextListWsh
'*** If here there is a List Wsh identified...
dbgWshExists = Utilities.WshExists(wsNm, invalidWbkNm)
If Not dbgWshExists Then
m_wshsMissing = True
Set cellProb = m_rngMppngTblData.Cells(i, m_mpgcolWshList)
'Flag problem wsh name...
cellProb.Interior.Color = vbRed
End If
NextListWsh:
Next i

End Sub
=====
Private Function GetWshTypeEnum(ByVal mppngTblRecNo As Integer) As ModWshType
Dim wsType As String

wsType = Trim(m_arrxlMppngTblData(mppngTblRecNo, m_mpgcolWshType))
Select Case wsType
Case "Static"
GetWshTypeEnum = modStatic
Case "Dynamic"
GetWshTypeEnum = modDynamic
End Select

End Function
=====
Private Sub CheckForAndFlagErrorsInSlideCreationInfoRange( _
ByVal mappingTblRecNo As Integer, _
ByVal infoRngRecNo As Integer, _
subRanOkOUT As Boolean)

Dim wsMain As Worksheet
Dim wsList As Worksheet
Dim rngCopy01 As Range
Dim cellProb As Range
Dim sldHdr As String
Dim nmWsMain As String
Dim nmWsList As String
Dim layoutCode As String
Dim addrRng01FmLstWsh As String
Dim slideNm As String
Dim layoutEnum As ProjSlideLayout
Dim chrt1PostnEnum As ProjChartPostn
Dim invalidListWshRng01Addr As Boolean
Dim validLayoutCode As Boolean

subRanOkOUT = True '...until we find otherwise

'Instantiate variables...

```

```

nmWsMain = Trim(m_arrxlMppngTblData(mappingTblRecNo, m_mpgcolWshMain))
nmWsList = Trim(m_arrxlMppngTblData(mappingTblRecNo, m_mpgcolWshList))
sldHdr = Trim(m_arrxlListWshSlideInfo(infoRngRecNo, m_SLIDE_CREATION_HDR))
layoutCode =
    Trim(m_arrxlListWshSlideInfo(infoRngRecNo, m_SLIDE_CREATION_LAYOUT))
addrRng01FmLstWsh =
    Trim(m_arrxlListWshSlideInfo(infoRngRecNo, m_SLIDE_CREATION_COPY_RNG_1))

Set wsMain = Worksheets(nmWsMain)
Set wsList = Worksheets(nmWsList)

'***Now trap for and flag errors...

'Slide header...
If StrComp(sldHdr, vbNullString) = 0 Then
    subRanOkOUT = False
    Set cellProb = m_rgnListWshSlideInfo.Cells( _
        infoRngRecNo, m_SLIDE_CREATION_HDR)
    cellProb.Interior.Color = vbRed
End If

'Layout code (entries here are controlled by a drop down, so the only possible _
error would be no entry at all)...
If StrComp(layoutCode, vbNullString) = 0 Then
    subRanOkOUT = False
    Set cellProb = m_rgnListWshSlideInfo.Cells( _
        infoRngRecNo, m_SLIDE_CREATION_LAYOUT)
    cellProb.Interior.Color = vbRed
Else
    validLayoutCode = True
End If

'Copy range 1...
If StrComp(addrRng01FmLstWsh, vbNullString) = 0 Then
    subRanOkOUT = False
    invalidListWshRng01Addr = True
Else '...address exists - confirm that its valid
    On Error Resume Next
    Set rngCopy01 = wsMain.Range(addrRng01FmLstWsh)
    If Err.number <> 0 Then
        subRanOkOUT = False
        invalidListWshRng01Addr = True
        Err.Clear
        On Error GoTo 0
    Else '... no error thrown
        If rngCopy01 Is Nothing Then
            subRanOkOUT = False
            invalidListWshRng01Addr = True
        End If
    End If
End If

If invalidListWshRng01Addr Then
    Set cellProb = m_rgnListWshSlideInfo.Cells( _
        infoRngRecNo, m_SLIDE_CREATION_COPY_RNG_1)
    cellProb.Interior.Color = vbRed
End If

'Now handle double-paste slides...
layoutEnum = GetPptSlideLayoutEnum( _
    mppngTblRecNo:=mappingTblRecNo, _
    wshListRecNo:=infoRngRecNo)

'If we only have one object to paste - or if we've already encountered _
an error - we are done...
If layoutEnum = ProjSlideLayout.projLayoutObj Or _
    layoutEnum = ProjSlideLayout.projLayoutLgObj Or _
    Not subRanOkOUT Then GoTo ExitPoint

'*** If here we have slides requiring 2 objects to be pasted in...
Dim rngCopy02 As Range
Dim positionCode As String

```

```
Dim addrRng02FmLstWsh As String
Dim invalidListWshRng02Addr As Boolean

positionCode = Trim(m_arrxlListWshSlideInfo( _
    infoRngRecNo, m_SLIDE_CREATION_CHRT1_POSTN))
addrRng02FmLstWsh = Trim( _
    m_arrxlListWshSlideInfo(infoRngRecNo, m_SLIDE_CREATION_COPY_RNG_2))

'Chart Position code (entries here are controlled by a drop down, so the _
only possible error would be no entry at all)...
If StrComp(positionCode, vbNullString) = 0 Then
    subRanOkOUT = False
    Set cellProb = _
        m_rgnListWshSlideInfo(infoRngRecNo, m_SLIDE_CREATION_CHRT1_POSTN)
    cellProb.Interior.Color = vbRed
End If

'Copy range 2...
If StrComp(addrRng02FmLstWsh, vbNullString) = 0 Then
    subRanOkOUT = False
    invalidListWshRng02Addr = True
Else '...address exists - confirm that its valid
    On Error Resume Next
    Set rngCopy02 = wsMain.Range(addrRng02FmLstWsh)
    If Err.number <> 0 Then
        subRanOkOUT = False
        invalidListWshRng02Addr = True
        Err.Clear
        On Error GoTo 0
    Else '... no error thrown
        If rngCopy02 Is Nothing Then
            subRanOkOUT = False
            invalidListWshRng02Addr = True
        End If
    End If
End If

If invalidListWshRng02Addr Then
    Set cellProb = m_rgnListWshSlideInfo.Cells( _
        infoRngRecNo, m_SLIDE_CREATION_COPY_RNG_2)
    cellProb.Interior.Color = vbRed
End If

ExitPoint:
    Exit Sub

End Sub

=====
Private Sub PrepExistingSlideOrCreateNewForPastingPic( _
    ByVal slideNm As String, _
    ByVal mppngTblRecNo As Integer, _
    subRanOkOUT As Boolean, _
    Optional ByVal wshListRecNo As Integer = 0)
Dim pptSlide As PowerPoint.Slide
Dim pptShape As PowerPoint.Shape
Dim pptCustLayout As CustomLayout
Dim enumSlideLayout As ProjSlideLayout
Dim i As Integer
Dim calledProcRanOK As Boolean

subRanOkOUT = True '... until proven otherwise

On Error GoTo ErrHandler

m_ppApp.Activate

enumSlideLayout = GetPptSlideLayoutEnum( _
    mppngTblRecNo:=mppngTblRecNo, _
    wshListRecNo:=wshListRecNo)

Dim dbgSlideExists As Boolean
dbgSlideExists = m_dicPreExistingSlides.exists(slideNm)
```

```

If dbgSlideExists Then
    'NOTE: After a shape is "deleted" from a PPT slide _
    there is still a shape left on the slide. The name _
    of that shape is set to "Content Placeholder #", _
    where # is some number (set by PPT). This is how _we _
    are able to go back in later and drop a new table or _
    chart into the same place - we simply look for a shape _
    called "Content Placeholder..."
    Set pptSlide = m_ppPres.Slides(slideNm)
    pptSlide.Select
    Select Case enumSlideLayout
        'Slides with 2 shapes...
        Case ProjSlideLayout.projLayoutTwoObjs, _
            ProjSlideLayout.projLayoutUndefined
            'Build a concatenated string of the 2 shape names _
            to delete from the slide...
            Dim concatShapeNmz As String
            For Each pptShape In pptSlide.Shapes
                If pptShape.Name = slideNm & "_1" Then
                    concatShapeNmz = _
                        concatShapeNmz & "|" & pptShape.Name
                End If
                If pptShape.Name = slideNm & "_2" Then
                    concatShapeNmz = _
                        concatShapeNmz & "|" & pptShape.Name
                End If
            Next pptShape
            Dim dbgShapesFound As Boolean
            dbgShapesFound =
                StrComp(concatShapeNmz, vbNullString) <> 0
            'Delete specified shapes...
            If dbgShapesFound Then
                'NOTE: Must declare as a variant, rather than _
                as a string, in order to use For/Each...
                Dim nmShpToDelete As Variant
                For Each nmShpToDelete In Split(concatShapeNmz, "|")
                    pptSlide.Shapes(nmShpToDelete).Delete
                Next nmShpToDelete
            End If
        'Slides with 1 shape...
        Case ProjSlideLayout.projLayoutObj, _
            ProjSlideLayout.projLayoutLgObj
            'Delete the "..._1" shape...
            For i = pptSlide.Shapes.Count To 1 Step -1
                Set pptShape = pptSlide.Shapes(i)
                If StrComp(pptShape.Name, slideNm & "_1") = 0 Then
                    pptShape.Delete
                    i = 1 '...break For loop
                End If
            Next i
    End Select
Else '... slide does not exist
    'Add a "blank" slide...
    Set pptCustLayout = GetCustomLayout( _
        sldLyoutEnum:=enumSlideLayout, _
        subRanOkOUT:=calledProcRanOK)
    If Not calledProcRanOK Then Err.Raise g_lERR_HANDLED
    Set pptSlide = m_ppPres.Slides.AddSlide( _
        m_ppPres.Slides.Count + 1, pptCustLayout)
    pptSlide.Select
    pptSlide.Name = slideNm
End If

ExitPoint:
Exit Sub

ErrHandler:
subRanOkOUT = False
GoTo ExitPoint

End Sub

```

---

```

Private Sub PopulateSlide(
    ByVal wshTypeEnum As ModWshType, _
    ByVal sldNm As String, _
    ByVal mppngTblRecNo As Integer, _
    subRanOkOUT As Boolean, _
    Optional ByVal listWshRecNo As Integer = 0)
'VERIFY: I strongly suspect that there is A LOT of duplicate code in this _
method - especially within the "If XL2016/Else..." constructs...
Const METHOD_NM As String = "PopulateSlide"
Const TITLE_SHAPE_NM As String = "shp_Title"
Const HDR_LEFT As Integer = -96
Const HDR_HGHT As Integer = 78
Const HDR_TOP As Integer = 18
Const HDR_WIDTH As Integer = 90
Const FONT_SZ As Byte = 11

Dim pptSlide As Slide
Dim pptShape As PowerPoint.Shape
Dim pptPlaceHldr As PowerPoint.Shape
Dim nmbrShapes As Integer
Dim i As Integer
Dim wsSrc As Worksheet
Dim rngCopy01 As Range
Dim rngCopy02 As Range
Dim titleTxt As String
Dim srcWshNm As String
Dim addrCopyRng01 As String
Dim addrCopyRng02 As String
Dim postnCode As String
Dim placeHldrNm As String
Dim pastedShapeSuffix As String
Dim nmbrSlideShpz As Integer
Dim sldLayoutEnum As ProjSlideLayout
Dim postnChrt01Enum As ProjChartPostn
Dim postnChrt02Enum As ProjChartPostn

subRanOkOUT = True '...until proven otherwise

On Error GoTo ErrorHandler

'Define always-needed slide-creation variables...
Select Case wshTypeEnum
    Case ModWshType.modDynamic
        'NOTE: We take all values from List wsh...
        titleTxt = Trim(m_arrxlListWshSlideInfo( _
            listWshRecNo, m_SLIDE_CREATION_HDR))
        addrCopyRng01 = Trim(m_arrxlListWshSlideInfo( _
            listWshRecNo, m_SLIDE_CREATION_COPY_RNG_1))
    Case ModWshType.modStatic
        'NOTE: We take all values from mapping table...
        titleTxt = _
            Trim(m_arrxlMppngTblData(mppngTblRecNo, m_mpgcolSlideHdrStat))
        addrCopyRng01 = _
            Trim(m_arrxlMppngTblData(mppngTblRecNo, m_mpgcolCopyRng1))
End Select
srcWshNm = Trim(m_arrxlMppngTblData(mppngTblRecNo, m_mpgcolWshMain))
Set wsSrc = Worksheets(srcWshNm)
Set rngCopy01 = wsSrc.Range(addrCopyRng01)
sldLayoutEnum = GetPptSlideLayoutEnum( _
    mppngTblRecNo:=mppngTblRecNo, _
    wshListRecNo:=listWshRecNo)

'Define slide-creation variables needed only for 2-image slides...
If sldLayoutEnum = ProjSlideLayout.projLayoutTwoObjs Or _
sldLayoutEnum = ProjSlideLayout.projLayoutUndefined Then
    Select Case wshTypeEnum
        Case ModWshType.modDynamic
            'NOTE: We take all values from List wsh...
            addrCopyRng02 = Trim(m_arrxlListWshSlideInfo( _
                listWshRecNo, m_SLIDE_CREATION_COPY_RNG_2))
            postnCode = Trim(m_arrxlListWshSlideInfo( _
                listWshRecNo, m_SLIDE_CREATION_CHRT1_POSTN))
        Case ModWshType.modStatic

```

```

'NOTE: We take all values from mapping table...
addrCopyRng02 =
    Trim(m_arrxlMppngTblData(mppngTblRecNo, m_mpgcolCopyRng2))
postnCode =
    Trim(m_arrxlMppngTblData(mppngTblRecNo, m_mpgcolPostnCopyRng1))
End Select
Set rngCopy02 = wsSrc.Range(addrCopyRng02)
postnChrt01Enum = GetChartPositionEnumFromString(postnCode)
postnChrt02Enum = GetChart2PositionEnumFmChart1Position(postnChrt01Enum)
End If

Application.StatusBar = "Updating PowerPoint slide: " & sldNm

Set pptSlide = m_ppPres.Slides(sldNm)

'Find the title obj/shape of the slide, and set the slide's title...
nmbrShapes = pptSlide.Shapes.Count
For i = 1 To nmbrShapes
    Set pptShape = pptSlide.Shapes(i)
    If pptShape.Name Like "title*" Or pptShape.Name = TITLE_SHAPE_NM Then
        With pptShape
            .TextFrame.TextRange.Text = titleTxt
            .Name = TITLE_SHAPE_NM
            .Width = HDR_WIDTH
            .Top = HDR_TOP
            .Height = HDR_HGHT
            .Left = HDR_LEFT
            .TextFrame.TextRange.Font.Size = FONT_SZ
            .TextFrame.TextRange.Font.Color = RGB(31, 73, 125)
            .Line.Visible = msoFalse
        End With
        i = nmbrShapes '... break loop
    End If
Next i

'Copy & paste pic into our slide, resizing & renaming along the way...
Select Case sldLayoutEnum
    Case ProjSlideLayout.projLayoutObj, ProjSlideLayout.projLayoutLgObj
        'We are going to be pasting into a one-picture slide...
        placeHldrNm = GetPlaceHldrNameForPastingShape( _
            slideName:=pptSlide.Name, _
            sldLayoutEnum:=sldLayoutEnum, _
            picPositionEnum:=ProjChartPostn.projPostnNotSpecified)
        If StrComp(placeHldrNm, vbNullString) = 0 Then GoTo ErrorHandler
        Set pptPlaceHldr = pptSlide.Shapes(placeHldrNm)

        'Copy our pic...
        ThisWorkbook.Activate
        rngCopy01.CopyPicture
            appearance:=XlPictureAppearance.xlScreen, _
            Format:=XlCopyPictureFormat.xlPicture
        m_ppApp.Activate
        pptPlaceHldr.Select

        'Select & rename placeholder...
        With pptPlaceHldr
            .Select msoTrue '...keeps the previous selection on the slide
            .Name = pptSlide.Name & "_1"
        End With

        'Paste, turn of XL cut/copy mode, reactivate PPT...
        m_ppPres.Windows(1).View.PasteSpecial ppPasteMetafilePicture
        ThisWorkbook.Activate
        Application.CutCopyMode = False '... turn off cut/copy mode
        m_ppApp.Activate

        'As soon as we paste in our picture our Placeholder object is _
        destroyed and a shape object is created which represents our _
        pasted picture. That shape will be called "Picture #" or _
        "Content Placeholder #." (We thought that the renaming convention _
        depended on the Office version we are working with, but now we ain't _
        quite so sure.)
        'In either event, we need to rename that shape: _

```

```

1) For future updates to the presentation.
2) So we can pass a reference to the shape into the _
   ResizePic method...
nmbrSlideShpz = pptSlide.Shapes.Count
For i = 1 To nmbrSlideShpz
    Set pptShape = pptSlide.Shapes(i)
    If pptShape.Name Like "Picture*" Or
        pptShape.Name Like "Content Placeholder*" Then
        pptShape.Name = pptSlide.Name & "_1"
        i = nmbrSlideShpz '...break loop
    End If
Next i

If sldLyoutEnum = ProjSlideLayout.projLayoutObj Then
    ResizePic
        slideNm:=sldNm,
        picName:=pptSlide.Name & "_1"
End If

Case ProjSlideLayout.projLayoutTwoObjs, ProjSlideLayout.projLayoutUndefined
'*****
'Handle pasting the first picture...
'*****
placeHldrNm = GetPlaceHldrNameForPastingShape( _
    slideName:=pptSlide.Name,
    sldLayoutEnum:=sldLyoutEnum,
    picPositionEnum:=postnChrt01Enum)
If StrComp(placeHldrNm, vbNullString) = 0 Then GoTo ErrorHandler
Set pptPlaceHldr = pptSlide.Shapes(placeHldrNm)

'Copy our pic...
ThisWorkbook.Activate
rngCopy01.CopyPicture
    appearance:=XlPictureAppearance.xlScreen,
    Format:=XlCopyPictureFormat.xlPicture
m_ppApp.Activate
pptPlaceHldr.Select

'Select & rename placeholder...
With pptPlaceHldr
    .Select msoTrue '...keeps the previous selection on the slide
    .Name = pptSlide.Name & "_1"
End With

'Paste, turn of XL cut/copy mode, reactivate PPT...
m_ppPres.Windows(1).View.PasteSpecial ppPasteMetafilePicture
ThisWorkbook.Activate
Application.CutCopyMode = False '... turn off cut/copy mode
m_ppApp.Activate

'As soon as we paste in our picture our Placeholder object is _
destroyed and a shape object is created which represents our _
pasted picture. That shape will be called "Picture #". We need to _
rename that shape for future updates to the presentation...
nmbrSlideShpz = pptSlide.Shapes.Count
For i = 1 To nmbrSlideShpz
    Set pptShape = pptSlide.Shapes(i)
    If pptShape.Name Like "Picture*" Then
        pptShape.Name = pptSlide.Name & "_1"
        i = nmbrSlideShpz '...break loop
    End If
Next i

'*****
'Handle pasting the second picture...
'*****
placeHldrNm = GetPlaceHldrNameForPastingShape( _
    slideName:=pptSlide.Name,
    sldLayoutEnum:=sldLyoutEnum,
    picPositionEnum:=postnChrt02Enum)
If StrComp(placeHldrNm, vbNullString) = 0 Then GoTo ErrorHandler
Set pptPlaceHldr = pptSlide.Shapes(placeHldrNm)

```

```

'Copy our pic...
ThisWorkbook.Activate
rngCopy02.CopyPicture _
    appearance:=XlPictureAppearance.xlScreen, _
    Format:=XlCopyPictureFormat.xlPicture
m_ppApp.Activate
pptPlaceHldr.Select

'Select & rename placeholder...
With pptPlaceHldr
    .Select msoTrue '...keeps the previous selection on the slide
    .Name = pptSlide.Name & "_2"
End With

'Paste, turn of XL cut/copy mode, reactivate PPT...
m_ppPres.Windows(1).View.PasteSpecial ppPasteMetafilePicture
ThisWorkbook.Activate
Application.CutCopyMode = False '... turn off cut/copy mode
m_ppApp.Activate

'As soon as we paste in our picture our Placeholder object is _
destroyed and a shape object is created which represents our _
pasted picture. That shape will be called "Picture #". We need to _
rename that shape for future updates to the presentation...
nmbrSlideShpz = pptSlide.Shapes.Count
For i = 1 To nmbrSlideShpz
    Set pptShape = pptSlide.Shapes(i)
    If pptShape.Name Like "Picture*" Then
        pptShape.Name = pptSlide.Name & "_2" '...2nd shape!
        i = nmbrSlideShpz '...break loop
    End If
Next i

```

End Select

```

ExitPoint:
Exit Sub

```

```

ErrorHandler:
subRanOkOUT = False
GoTo ExitPoint
End Sub

```

```

'=====

```

```

Private Function GetPlaceHldrNameForPastingShape( _
    ByVal slideName As String, _
    ByVal sldLayoutEnum As ProjSlideLayout, _
    ByVal picPositionEnum As ProjChartPostn) As String

```

```

Const NUMERICAL_SUFFIX_POSTN As Byte = 2
Const SUFFIX_UPPER_BOUND As Long = 100000
Dim pptSlide As Slide
Dim pptShape As PowerPoint.Shape
Dim concatdPlcHolderNmz As String
Dim plcHolderNmTarget As String
Dim isTwoPictSlide As Boolean
Dim getSecondPlaceHolderNm As Boolean

```

```

Set pptSlide = m_ppPres.Slides(slideName)

```

```

'Set our booleans...
If sldLayoutEnum = ProjSlideLayout.projLayoutTwoObjs Or _
sldLayoutEnum = ProjSlideLayout.projLayoutUndefined _
Then isTwoPictSlide = True
If isTwoPictSlide And _
(picPositionEnum = ProjChartPostn.projPostnDown Or _
picPositionEnum = ProjChartPostn.projPostnRight) _
Then getSecondPlaceHolderNm = True

```

```

'Create a concatenated string of all shapes names where the shape _
is a placeholder...

```

```

For Each pptShape In pptSlide.Shapes
    If pptShape.Name Like "Content Placeholder " & "*" Then
        concatdPlcHolderNmz = _

```

```

        concatdPlcHolderNmz & "|" & pptShape.Name
    End If
Next pptShape

'Handle lack of placeholders...
If StrComp(concatdPlcHolderNmz, vbNullString) = 0 Then GoTo ExitPoint

'Trim leading piping character....
concatdPlcHolderNmz =
    Right(concatdPlcHolderNmz, Len(concatdPlcHolderNmz) - 1)

'If there's only one placeholder we are done...
If Not isTwoPictSlide Then
    plcHolderNmTarget = concatdPlcHolderNmz
    GoTo ExitPoint
End If

'*****
'If here we are dealing with a two-pictures slide...
'*****

'Declare & instantiate our variables...
Dim plcHolder01Nm As String
Dim plcHolder02Nm As String
Dim plcHolder01Suffix As Long
Dim plcHolder02Suffix As Long
Dim numrclSuffixPostn As Byte
plcHolder01Nm = Split(concatdPlcHolderNmz, "|")(0)
plcHolder02Nm = Split(concatdPlcHolderNmz, "|")(1)
plcHolder01Suffix = CLng(Split(plcHolder01Nm, " ")(NUMERICAL_SUFFIX_POSTN))
plcHolder02Suffix = CLng(Split(plcHolder02Nm, " ")(NUMERICAL_SUFFIX_POSTN))

'Determine which placeholder we want...
'NOTE: We assume that the placeholder with the greater numerical suffix
is always the one we want when we are going to paste a second picture
onto a slide. (Etc., etc. for reverse logic.)
If getSecondPlaceholderNm Then
    If plcHolder02Suffix > plcHolder01Suffix Then
        plcHolderNmTarget = plcHolder02Nm
    Else
        plcHolderNmTarget = plcHolder01Nm
    End If
Else '... we are going to paste the 1st of 2 pics onto the slide
    If plcHolder01Suffix < plcHolder02Suffix Then
        plcHolderNmTarget = plcHolder01Nm
    Else
        plcHolderNmTarget = plcHolder02Nm
    End If
End If

ExitPoint:
    GetPlaceHldrNameForPastingShape = plcHolderNmTarget
    Exit Function

End Function

'=====
Private Function GetPptSlideLayoutEnum( _
    ByVal mppngTblRecNo As Integer, _
    Optional ByVal wshListRecNo As Integer = 0) As ProjSlideLayout
Dim wshTypeEnum As ModWshType
Dim loCode As String
Dim loEnum As ProjSlideLayout

If wshListRecNo > 0 Then
    wshTypeEnum = ModWshType.modDynamic
Else
    wshTypeEnum = ModWshType.modStatic
End If

Select Case wshTypeEnum
    Case ModWshType.modDynamic
        loCode =
            Trim(m_arrxllListWshSlideInfo(wshListRecNo, m_SLIDE_CREATION_LAYOUT))

```

```

Case ModWshType.modStatic
    loCode = _
        Trim(m_arrxlMppngTblData(mppngTblRecNo, m_mpgcolPptLayoutTypeRng))
End Select

Select Case loCode
Case "1"
    loEnum = projLayoutObj
Case "2"
    loEnum = projLayoutTwoObjs
Case "3"
    loEnum = projLayoutUndefined
Case "4"
    loEnum = projLayoutLgObj
End Select

GetPptSlideLayoutEnum = loEnum

End Function

'=====
Private Function GetChartPositionEnumFromString(ByVal postn As String) _
    As ProjChartPostn
Dim pstnEnum As ProjChartPostn

Select Case postn
Case "Up"
    pstnEnum = projPostnUp
Case "Down"
    pstnEnum = projPostnDown
Case "Left"
    pstnEnum = projPostnLeft
Case "Right"
    pstnEnum = projPostnRight
Case Else
    pstnEnum = projPostnNotSpecified
End Select

GetChartPositionEnumFromString = pstnEnum

End Function

'=====
Private Function GetChart2PositionEnumFmChart1Position( _
    ByVal chrt1Postn As ProjChartPostn) As ProjChartPostn
Dim chrt2Postn As ProjChartPostn

Select Case chrt1Postn
Case ProjChartPostn.projPostnUp
    chrt2Postn = projPostnDown
Case ProjChartPostn.projPostnDown
    chrt2Postn = projPostnUp
Case ProjChartPostn.projPostnLeft
    chrt2Postn = projPostnRight
Case ProjChartPostn.projPostnRight
    chrt2Postn = projPostnLeft
Case ProjChartPostn.projPostnNotSpecified
    chrt2Postn = projPostnNotSpecified
End Select

GetChart2PositionEnumFmChart1Position = chrt2Postn

End Function

'=====
Private Sub ResizePic( _
    ByVal slideNm As String, _
    ByVal picName As String) _
Const SLIDE_FULL_WIDTH As Long = 720
Const SLIDE_FULL_HT As Long = 540
Const MIN_VERT_OFFSET As Byte = 72
Dim pptSlide As PowerPoint.Slide
Dim pptShapePic As PowerPoint.Shape
Dim topPosition As Long

Set pptSlide = m_ppPres.Slides(slideNm)

```

```

Set pptShapePic = pptSlide.Shapes(picName)

'Center shape horizontally...
pptShapePic.Left = (SLIDE_FULL_WIDTH - pptShapePic.Width) / 2

'Compute position for center the pic vertically...
topPosition = (SLIDE_FULL_HT - pptShapePic.Height) / 2
'However, maintain a minimum vertical offset...
If topPosition < MIN_VERT_OFFSET Then topPosition = MIN_VERT_OFFSET

pptShapePic.Top = topPosition

End Sub

'=====
Private Function GetCustomLayout( _
    ByVal sldLayoutEnum As ProjSlideLayout, _
    subRanOkOUT As Boolean) _
    As PowerPoint.CustomLayout _
Dim custLO As PowerPoint.CustomLayout

subRanOkOUT = True '... until proven otherwise

On Error GoTo ErrHandler

With m_ppPres.Designs(1).SlideMaster
    Select Case sldLayoutEnum
        Case ProjSlideLayout.projLayoutObj
            Set custLO = .CustomLayouts(3)
        Case ProjSlideLayout.projLayoutTwoObjs
            Set custLO = .CustomLayouts(4)
        Case ProjSlideLayout.projLayoutUndefined
            Set custLO = .CustomLayouts(5)
        Case ProjSlideLayout.projLayoutLgObj
            Set custLO = .CustomLayouts(6)
    End Select
End With

ExitPoint:
    Set GetCustomLayout = custLO
    Exit Function

ErrHandler:
    Set custLO = Nothing
    subRanOkOUT = False
    GoTo ExitPoint

End Function

'=====
Private Sub LogProblemSlide(ByVal sldNm As String)
ReDim Preserve g_nmzProbSlides(0 To g_nmbrProbSlides)
g_nmzProbSlides(g_nmbrProbSlides) = sldNm
g_nmbrProbSlides = g_nmbrProbSlides + 1
End Sub

'=====
Private Sub RunDynamicWshMacro( _
    ByVal mppngTblRecNo As Integer, _
    ByVal listWshRecNo As Integer, _
    subRanOkOUT As Boolean)
Const IS_FIRED_BY_DD As Boolean = False
Dim macroNm As String
Dim macroInput As String
Dim calledProcRanOK As Boolean

'NOTE: These macros work by referencing an input cell. In "manual mode" _
the macros are fired via the drop down on their appropriate worksheets. _
The input cell = the linked cell for the drop down. Whenever the value of _
the drop down changes it triggers an even handler which calls the _
appropriate macro.
'We have added a level of indirection to the original code. Now the wsh _
event handler fires a macro in the ComboBoxHandlers module. The macros in _
the ComboBoxHandlers module then fire the desired target macro. _
The purpose of adding this indirection is to enable the code here in the _
Output Model to: _

```

- 1) Update the value of the linked cell/input cell \_
- 2) Fire the target marco directly

'Benefits of this approach: \_

- 1) This code no longer needs to work through updating the drop down \_ object (which was a pretty clumsy architecture) \_
- 2) We were able to add some error-trapping/handling to the final, \_ target macros.

```
subRanOkOUT = True '... until we find otherwise
```

```
macroNm = _
Trim(m_arrxlMppngTblData(mppngTblRecNo, m_mpgcolMacroNm))
macroInput = Trim(m_arrxlMacroInputs(listWshRecNo, 1))
```

```
Select Case macroNm
Case "Load_CAMELS_Factor_Output"
CamelFactor.Load_CAMELS_Factor_Output _
firedByDropDn:=IS_FIRED_BY_DD, _
subRanOkOUT:=calledProcRanOK, _
inputVal:=macroInput
Case "Load_RISKMATRIX_Factor_Output"
RiskMatrix.Load_RISKMATRIX_Factor_Output _
firedByDropDn:=IS_FIRED_BY_DD, _
subRanOkOUT:=calledProcRanOK, _
inputVal:=macroInput
Case "Load_Risk_Factor_Output"
RiskFactor.Load_Risk_Factor_Output _
firedByDropDn:=IS_FIRED_BY_DD, _
subRanOkOUT:=calledProcRanOK, _
inputVal:=macroInput
Case "Load_Risk_Ratio_From_Table"
RiskRatios.Load_Risk_Ratio_From_Table _
firedByDropDn:=IS_FIRED_BY_DD, _
subRanOkOUT:=calledProcRanOK, _
inputVal:=macroInput
Case "Peer_Summary_View"
PeerData.Peer_Summary_View _
firedByDropDn:=IS_FIRED_BY_DD, _
subRanOkOUT:=calledProcRanOK, _
inputVal:=macroInput
Case "RR_Summary_View"
RiskRatioSummary.RR_Summary_View _
firedByDropDn:=IS_FIRED_BY_DD, _
subRanOkOUT:=calledProcRanOK, _
inputVal:=macroInput
End Select
```

```
If Not calledProcRanOK Then GoTo ErrHandler
```

```
ExitPoint:
Exit Sub
```

```
ErrHandler:
Err.Clear
subRanOkOUT = False
GoTo ExitPoint
```

```
End Sub
```

```
Private Sub FlagInconsistentDynamicWshRanges( _
subRanOkOUT As Boolean)
```

```
Dim wsList As Worksheet
Dim rngListFill As Range
Dim rngInclTF As Range
Dim rngMacroInputVals As Range
Dim rngRiskNos As Range
Dim rngSlideHdrs As Range
Dim wsType As String
Dim wsListNm As String
Dim listFillRngXLNm As String
Dim addrInclTF As String
Dim addrMacroInputVals As String
Dim addrRiskNos As String
```

```

Dim addrSlideHdrs As String
Dim listFillRngRecNos As Long
Dim i As Integer

subRanOkOUT = True '... until we find otherwise

On Error GoTo ErrHandler

For i = 1 To m_rngMppngTblData.Rows.Count
    'See if we are even working with a dynamic wsh...
    wsType = Trim(m_arrxlMppngTblData(i, m_mpgcolWshType))
    If StrComp(wsType, "Dynamic") <> 0 Then GoTo NextMappingTblRec

    'Get the # rows each other range SHOULD have...
    listFillRngXLNm = Trim(m_arrxlMppngTblData(i, m_mpgcolListFillRngNm))
    Set rngListFill = Names(listFillRngXLNm).RefersToRange
    listFillRngRecNos = rngListFill.Rows.Count

    'Define List wsh....
    wsListNm = Trim(m_arrxlMppngTblData(i, m_mpgcolWshList))
    Set wsList = Worksheets(wsListNm)

    'Define List wsh range sizes to check...
    addrInclTF = Trim(m_arrxlMppngTblData(i, m_mpgcolListRngInclTF))
    addrMacroInputVals = Trim(m_arrxlMppngTblData(i, m_mpgcolListRngDpDnVals))
    addrRiskNos = Trim(m_arrxlMppngTblData(i, m_mpgcolListRngRskNos))
    addrSlideHdrs = Trim(m_arrxlMppngTblData(i, m_mpgcolSlideHdrRngDyn))
    Set rngInclTF = wsList.Range(addrInclTF)
    Set rngMacroInputVals = wsList.Range(addrMacroInputVals)
    Set rngRiskNos = wsList.Range(addrRiskNos)
    Set rngSlideHdrs = wsList.Range(addrSlideHdrs)

    'Check T/F List range...
    If listFillRngRecNos <> rngInclTF.Rows.Count Then
        ReDim Preserve g_nmzProbListWshs(0 To g_nmbrProbListWshs)
        g_nmzProbListWshs(g_nmbrProbListWshs) = wsListNm
        g_nmbrProbListWshs = g_nmbrProbListWshs + 1
        GoTo NextMappingTblRec '...we already know there are issues
    End If

    'Check macro inputs range...
    If listFillRngRecNos <> rngMacroInputVals.Rows.Count Then
        ReDim Preserve g_nmzProbListWshs(0 To g_nmbrProbListWshs)
        g_nmzProbListWshs(g_nmbrProbListWshs) = wsListNm
        g_nmbrProbListWshs = g_nmbrProbListWshs + 1
        GoTo NextMappingTblRec '...we already know there are issues
    End If

    'Check Risk #s range...
    If listFillRngRecNos <> rngRiskNos.Rows.Count Then
        ReDim Preserve g_nmzProbListWshs(0 To g_nmbrProbListWshs)
        g_nmzProbListWshs(g_nmbrProbListWshs) = wsListNm
        g_nmbrProbListWshs = g_nmbrProbListWshs + 1
        GoTo NextMappingTblRec '...we already know there are issues
    End If

    'Check slide headers range...
    If listFillRngRecNos <> rngSlideHdrs.Rows.Count Then
        ReDim Preserve g_nmzProbListWshs(0 To g_nmbrProbListWshs)
        g_nmzProbListWshs(g_nmbrProbListWshs) = wsListNm
        g_nmbrProbListWshs = g_nmbrProbListWshs + 1
        GoTo NextMappingTblRec '...we already know there are issues
    End If

    'NOTE: We do NOT check the size of the ranges defined for layout type, _
    position copy 1, copy range 1, or copy range 2. This is because our code _
    defines those ranges as an offset from the dynamic slide headers range. _
    Thus, if the dyanamic slide headers range is correctly sized then _
    so are the other 4.

NextMappingTblRec:
Next i

```

```
ExitPoint:  
  If g_nmbrProbListWshs > 0 Then subRanOkOUT = False  
  Exit Sub
```

```
ErrorHandler:  
  subRanOkOUT = False  
  GoTo ExitPoint
```

```
End Sub
```

'=====