GemBox Support Center

Range Cell Address Reference Completed

- JG Jan Gunawan
- **Forum name:** #GemBox.Spreadsheet Feature Request

Please implement a new method or constructor on CellRange class or a new GetSubrange method, so it can use cell range address reference of A1 notation, as stated in the following MSDN documentation:

https://msdn.microsoft.com/en-us/library/bb211395%28v=office.12%29.aspx

Comments (2)

RS **Rob Sherratt**
11 years ago
Perhaps these procedures especially Get_CellCollection() will help in the meantime? There is a simple Forms application for testing. I think I tested most of the combinations of cell range references ... seems OK. Imports System Imports System.Collections.Generic Imports System.Text Imports GemBox.Spreadsheet Imports GemBox.Spreadsheet.ConditionalFormatting Imports GemBox.Spreadsheet.PivotTables Class Form1 Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load ' If using Professional version, put your serial key below. SpreadsheetInfo.SetLicense("FREE-LIMITED-KEY") Dim ef As ExcelFile = New ExcelFile Dim ws As ExcelWorksheet = ef.Worksheets.Add("Range Formatting") Dim range As CellRange = Nothing Dim rowCount As Integer = 20 ' Specify data formatting. ws.Columns(0).SetWidth(3, LengthUnit.Centimeter) ws.Columns(1).SetWidth(3, LengthUnit.Centimeter) ws.Columns(2).SetWidth(3, LengthUnit.Centimeter) ws.Columns(3).SetWidth(3, LengthUnit.Centimeter) ws.Columns(3).Style.NumberFormat = "[$$-409]#,##0.00" ws.Columns(4).SetWidth(3, LengthUnit.Centimeter) ws.Columns(4).Style.NumberFormat = "yyyy-mm-dd" ws.Columns(5).SetWidth(3, LengthUnit.Centimeter) Dim cells = ws.Cells ' Specify header row. cells(0, 0).Value = "Departments" cells(0, 1).Value = "Names" cells(0, 2).Value = "Years of Service" cells(0, 3).Value = "Salaries" cells(0, 4).Value = "Deadlines" cells(0, 5).Value = "Comments" ' Insert random data to sheet. Dim random = New Random() Dim departments = New String() {"Legal", "Marketing", "Finance", "Planning", "Purchasing"} Dim names = New String() {"John Doe", "Fred Nurk", "Hans Meier", "Ivan Horvat"} Dim comments = New String() {"", "", "Overdue", ""} For i As Integer = 0 To rowCount - 1 cells(i + 1, 0).Value = departments(random.Next(departments.Length)) cells(i + 1, 1).Value = names(random.Next(names.Length)) + " "c + (i + 1).ToString() cells(i + 1, 2).Value = random.Next(1, 31) cells(i + 1, 3).Value = random.Next(10, 101) * 100 cells(i + 1, 4).Value = DateTime.Now.AddDays(random.Next(-1, 2)) cells(i + 1, 5).Value = comments(random.Next(comments.Length)) Next Get_CellCollection(ws, "D16:F21", range) range.Style.FillPattern.SetSolid(Color.Yellow) Get_CellCollection(ws, "7:8", range) range.Style.Font.Weight = ExcelFont.BoldWeight range.Style.FillPattern.SetSolid(Color.LightGreen) Get_CellCollection(ws, "C4", range) range.Style.FillPattern.SetSolid(Color.LightBlue) Get_CellCollection(ws, "E", range) range.Style.FillPattern.SetSolid(Color.LightSlateGray) Get_CellCollection(ws, "12", range) range.Style.FillPattern.SetSolid(Color.MistyRose) Get_CellCollection(ws, "A17:B", range) range.Style.FillPattern.SetSolid(Color.OrangeRed) System.IO.File.Delete(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) & "\Range Formatting.xlsx") ef.Save(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) & "\Range Formatting.xlsx") Me.Close() End Sub Private Function Get_CellCollection(ws As GemBox.Spreadsheet.ExcelWorksheet, CellRef As String, _ ByRef MyCellRange As GemBox.Spreadsheet.CellRange) As Boolean ' Fetches a CellRange from the Worksheet using Excel-type CellRef string. ' Any "$" absolute references are allowed and stripped out. Dim CellRef1 As String = "" Dim CellRef2 As String = "" Dim DelimIdx As Integer Dim Row1 As Integer = 0 Dim Row2 As Integer = 0 Dim Col1 As Integer = 0 Dim Col2 As Integer = 0 Dim Syntax1 As Integer = 0 Dim Syntax2 As Integer = 0 ' 0 = invalid syntax ' 1 = whole column selection syntax like "AD" - col is valid, row is not modified ' 2 = whole row selection syntax like "3" - row is valid, col is not modified ' 3 = cell reference like "BC3" - both row and col are valid CellRef = Replace(CellRef, "$", "") ' remove any Excel "$" absolute references since these are unsupported in GB DelimIdx = InStr(CellRef, ":") If DelimIdx > 0 Then ' cellrange with two tokens CellRef1 and CellRef2 ' Process the first token, CellRef1 CellRef1 = Mid(CellRef, 1, DelimIdx - 1) If Not Parse_ExcelRefToken(CellRef1, Syntax1, Row1, Col1) Then Return False ' Process the second token, CellRef2 CellRef2 = Mid(CellRef, DelimIdx + 1, Len(CellRef) - DelimIdx) If Not Parse_ExcelRefToken(CellRef2, Syntax2, Row2, Col2) Then Return False Else ' single row or column specified by one token CellRef If Not Parse_ExcelRefToken(CellRef, Syntax1, Row1, Col1) Then Return False End If Select Case Syntax1 Case 1 Select Case Syntax2 Case 0 MyCellRange =

ws.Cells.GetSubrangeAbsolute(0, Col1, ws.Rows.Count - 1, Col1) Case 1 MyCellRange = ws.Cells.GetSubrangeAbsolute(0, Col1, ws.Rows.Count - 1, Col2) Case 2 MyCellRange = ws.Cells.GetSubrangeAbsolute(Row2, Col1, Row2, Col1) Case 3 MyCellRange = ws.Cells.GetSubrangeAbsolute(0, Col1, Row2, Col2) End Select Case 2 Select Case Syntax2 Case 0 MyCellRange = ws.Cells.GetSubrangeAbsolute(Row1, 0, Row1, ws.Columns.Count - 1) Case 1 MyCellRange = ws.Cells.GetSubrangeAbsolute(Row1, Col2, Row1, Col2) Case 2 MyCellRange = ws.Cells.GetSubrangeAbsolute(Row1, 0, Row2, ws.Columns.Count - 1) Case 3 MyCellRange = ws.Cells.GetSubrangeAbsolute(Row1, 0, Row2, Col2) End Select Case 3 Select Case Syntax2 Case 0 MyCellRange = ws.Cells.GetSubrangeAbsolute(Row1, Col1, Row1, Col1) Case 1 MyCellRange = ws.Cells.GetSubrangeAbsolute(Row1, Col1, ws.Rows.Count - 1, Col2) Case 2 MyCellRange = ws.Cells.GetSubrangeAbsolute(Row1, Col1, Row2, ws.CalculateMaxUsedColumns - 1) Case 3 MyCellRange = ws.Cells.GetSubrangeAbsolute(Row1, Col1, Row2, Col2) End Select End Select Return True End Function Private Function Parse_ExcelRefToken(CellRef As String, ByRef Syntax As Integer, ByRef Row As Integer, ByRef Col As Integer) As Boolean ' Parses a single Excel Reference token Dim Char1 As String = Mid(CellRef, 1, 1) Dim Char2 As String = Mid(CellRef, Len(CellRef), 1) If Char1 >= "A" And Char1 <= "Z" Then If IsNumeric(Char2) Then CellRange.PositionToRowColumn(CellRef, Row, Col) Syntax = 3 ' CellRef specifies a cell reference like "BC3" Return True Else If Char2 >= "A" And Char2 <= "Z" And Len(CellRef) <= 2 Then Col = ExcelColumnCollection.ColumnNameToIndex(CellRef) Syntax = 1 ' CellRef specifies a whole column selection syntax like "BD" Return True Else Syntax = 0 ' Illegal syntax Return False End If End If Else ' CellRef is either numeric and specifies a single row, or else is invalid If IsNumeric(CellRef) Then Row = CInt(CellRef) - 1 Syntax = 2 ' whole row selection syntax like "321" Return True Else Syntax = 0 ' illegal Syntax Return False End If End If End Function Private Function Get_Excel_Col_By_Index(Col_Index As Integer) As String ' Converts a numeric column index numbered from 0 to 702 into an Excel column index ' with letter references from A ... Z, AA ... AZ, BA ... BZ, , ZA to ZZ If Col_Index < 0 Or Col_Index > 702 Then Return "" Return ExcelColumnCollection.ColumnIndexToName(Col_Index) End Function Private Function Get_Excel_Col_By_Name(ws As ExcelWorksheet, Colname As String, Optional Header_row As Integer = 0) As String ' Converts the first Colname found in the header row number Header_Row into an Excel column index ' with letter references from A ... Z, AA ... AZ, BA ... BZ, , ZA to ZZ Dim headerCells = ws.Rows(Header_row).AllocatedCells Dim colNumber As Integer If String.IsNullOrWhiteSpace(Colname) Then Return "" For colNumber = 0 To headerCells.Count - 1 If (headerCells(colNumber).Value).Equals(Colname) Then Return ExcelColumnCollection.ColumnIndexToName(colNumber) End If Next Return "" End Function End Class

MG **Mario - GemBox**

8 years ago

Hi, This feature request has been implemented and is available in the latest versions of GemBox.Spreadsheet.

Regards, Mario