**Working with Tables in Excel 2010 and 2007 (VBA)**This article has also been published on Microsoft Office Online: [Working with Excel tables in Visual Basic for Applications (VBA)](http://office.microsoft.com/en-us/excel/HA102880241033.aspx)

**Introduction**In [Working with Tables in Excel 2010 and 2007](http://www.jkp-ads.com/articles/Excel2007Tables.asp) I promised to add a page about working with those tables in VBA too. Well, here you go.

**It's a ListObject!**On the VBA side there seems to be nothing new about Tables. They are addressed as ListObjects, a collection that was introduced with Excel 2003. But there are significant changes to this part of the object model and I am only going to touch on the basic parts here.

**This does the same as the Convert to Range function** ActiveSheet.ListObjects("TableName").Unlist

**Creating a table**Converting a range to a table starts with the same code as in Excel 2003:

Sub CreateTable()  
    ActiveSheet.ListObjects.Add(xlSrcRange, Range("$B$1:$D$16"), , xlYes).Name = \_  
        "Table1"  
        'No go in 2003  
    ActiveSheet.ListObjects("Table1").TableStyle = "TableStyleLight2"  
End Sub

But the new stuff is right there already: TableStyles. A collection of objects which are a member of the Workbook object. This gives rise to some oddities. You can change the formatting of a tableStyle, e.g. like this:

Sub ChangeTableStyles()  
    'No Go in Excel 2003  
    ActiveWorkbook.TableStyles(2).TableStyleElements(xlWholeTable) \_  
        .Borders(xlEdgeBottom).LineStyle = xlDash  
End Sub

This changes the linestyle of the bottom of your table. But hold your horses! If you have any other workbook open, all tables with the same tablestyle appear in your changed style! But if you save your file, close Excel and open Excel again with the file, the changes are gone. This is because you've just changed a built-in tablestyle. If you ask me, I find it strange that the Workbook is a tablestyles' parent, whereas built-in table styles behave as if being bound to the Application object.

If you want full control over your table style, you'd better duplicate a built-in style and modify and apply that style to your table.

**Listing the tables**Let's start with finding all tables on the active worksheet:

Sub FindAllTablesOnSheet()  
    Dim oSh As Worksheet  
    Dim oLo As ListObject  
    Set oSh = ActiveSheet  
    For Each oLo In oSh.ListObjects  
        Application.Goto oLo.Range  
        MsgBox "Table found: " & oLo.Name & ", " & oLo.Range.Address  
    Next  
End Sub

This snippet of code works exactly the same in Excel 2003, so nothing new there (well, that is, in 2003 those tables ARE called Lists).

**Selecting parts of tables**You might need to work with specific parts of a table. Here is a couple of examples on how to achieve that. The code comments show you where Excel 2003 differs from 2010 and 2007.

Sub SelectingPartOfTable()  
    Dim oSh As Worksheet  
    Set oSh = ActiveSheet  
    '1: with the listobject  
    With oSh.ListObjects("Table1")  
        MsgBox .Name  
        'Select entire table  
        .Range.Select  
        'Select just the data of the entire table  
        .DataBodyRange.Select  
        'Select third column  
        .ListColumns(3).Range.Select  
        'Select only data of first column  
        'No go in 2003  
        .ListColumns(1).DataBodyRange.Select  
        'Select just row 4 (header row doesn't count!)  
        .ListRows(4).Range.Select  
    End With  
      
    'No go in 2003  
    '2: with the range object  
    'select an entire column (data only)  
    oSh.Range("Table1[Column2]").Select  
    'select an entire column (data plus header)  
    oSh.Range("Table1[[#All],[Column1]]").Select  
    'select entire data section of table  
    oSh.Range("Table1").Select  
    'select entire table  
    oSh.Range("Table1[#All]").Select  
    'Select one row in table  
    oSh.Range("A5:F5").Select  
End Sub

As you may have spotted, Excel 2010 and 2007 handle tables like they are range names. Well, that is exactly what is going on. After inserting a table, a range name is defined automatically. These range names are special though. Excel controls them entirely. You cannot delete them and they get renamed automatically when you change a table's name. Remove a table (convert back to range) and the defined name is removed as well.

**Inserting rows and columns**Another part in which lists already had most of the functionality. Just a few new things have been added, like the "AlwaysInsert" argument to the ListRows.Add method:

Sub TableInsertingExamples()  
'insert at specific position  
    Selection.ListObject.ListColumns.Add Position:=4  
'insert right  
    Selection.ListObject.ListColumns.Add  
'insert above  
    Selection.ListObject.ListRows.Add (11)  
'NoGo in 2003  
'insert below  
    Selection.ListObject.ListRows.Add AlwaysInsert:=True  
End Sub

If you need to do something with a newly inserted row, you can set an object variable to the new row:

     Dim oNewRow As ListRow  
    Set oNewRow = Selection.ListObject.ListRows.Add(AlwaysInsert:=True)

If you then want to write something in the first cell of the new row you can use:

oNewRow .Range.Cells(1,1).Value="Value For New cell"

**Adding a comment to a table**This is something Excel 2003 cannot do and is related to the fact that a table is a range name. Adding a comment to a table through the UI is a challenge, because you have to go to the Name Manager to do that. In VBA the syntax is:

Sub AddComment2Table()  
    Dim oSh As Worksheet  
    Set oSh = ActiveSheet  
    'NoGo in 2003  
    'add a comment to the table (shows as a comment to  
    'the rangename that a table is associated with automatically)  
    'Note that such a range name cannot be deleted!!  
    'The range name is removed as soon as the table is converted to a range  
    oSh.ListObjects("Table1").Comment = "This is a table's comment"  
End Sub

**Convert a table back to a normal range**That is simple and uses the identical syntax as 2003:

Sub RemoveTableStyle()  
    Dim oSh As Worksheet  
    Set oSh = ActiveSheet  
    'remove table or list style  
    oSh.ListObjects("Table1").Unlist  
End Sub

**Special stuff: Sorting and filtering**With Excel 2010 and 2007 we get a whole new set of filtering and sorting options. I'm only showing a tiny bit here, a Sort on cell color (orangish) and a filter on the font color. The code below doesn't work in Excel 2003. A List in 2003 only has the default sort and autofilter possibilities we have known since Excel 5 and which had hardly been expanded at all in the past 12 years or so.

Sub SortingAndFiltering()  
'NoGo in 2003  
    With ActiveWorkbook.Worksheets("Sheet1").ListObjects("Table1")  
  
        .Sort.SortFields.Clear  
        .Sort.SortFields.Add( \_  
                Range("Table1[[#All],[Column2]]"), xlSortOnCellColor, xlAscending, , \_  
                xlSortNormal).SortOnValue.Color = RGB(255, 235, 156)  
        With .Sort  
            .Header = xlYes  
            .MatchCase = False  
            .Orientation = xlTopToBottom  
            .SortMethod = xlPinYin  
            .Apply  
        End With  
    End With  
    'Only old autofilter stuff works in 2003  
    ActiveSheet.ListObjects("Table1").Range.AutoFilter Field:=2, \_  
        Criteria1:=RGB(156, 0, 6), Operator:=xlFilterFontColor  
End Sub

**Accessing the formatting of a cell inside a table**You may wonder why this subject is there, why not simply ask for the cell.Interior.ThemeColor if you need the ThemeColor of a cell in a table? Well, because the cell formatting is completely prescribed by the settings of your table and the table style that  has been selected. So in order to get at a formatting element of a cell in your table you need to:

* Find out where in your table the cell is located (on header row, on first column, in the bulk of the table
* Determine the table settings: does it have row striping turned on, does it have a specially formatted first column, ...
* Based on these pieces of information, one can extract the appropriate TableStyleElement from the table style and read its properties.

The function shown here returns the TableStyleElement belonging to a cell oCell inside a table object called oLo:

Function GetStyleElementFromTableCell(oCell As Range, oLo As ListObject) As TableStyleElement  
'-------------------------------------------------------------------------  
' Procedure : GetStyleElementFromTableCell  
' Company   : JKP Application Development Services (c)  
' Author    : Jan Karel Pieterse  
' Created   : 2-6-2009  
' Purpose   : Function to return the proper style element from a cell inside a table  
'-------------------------------------------------------------------------  
    Dim lRow As Long  
    Dim lCol As Long  
    'Determine on what row we are inside the table  
    lRow = oCell.Row - oLo.DataBodyRange.Cells(1, 1).Row  
    lCol = oCell.Column - oLo.DataBodyRange.Cells(1, 1).Column  
  
    With oLo  
        If lRow < 0 And .ShowHeaders Then  
            'on first row and has header  
            Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlHeaderRow)  
        ElseIf .ShowTableStyleFirstColumn And lCol = 0 Then  
            'On first column and has first column style  
            Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlFirstColumn)  
        ElseIf .ShowTableStyleLastColumn And lCol = oLo.Range.Columns.Count - 1 Then  
            'On last column and has last col style  
            Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlLastColumn)  
        ElseIf lRow = .DataBodyRange.Rows.Count And .ShowTotals Then  
            'On last row and has total row  
            Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlTotalRow)  
        Else  
            If .ShowTableStyleColumnStripes And Not .ShowTableStyleRowStripes Then  
                'in table, has column stripes  
                If lCol Mod 2 = 0 Then  
                    Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlColumnStripe1)  
                Else  
                    Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlWholeTable)  
                End If  
            ElseIf .ShowTableStyleRowStripes And Not .ShowTableStyleColumnStripes Then  
                'in table, has column stripes  
                If lRow Mod 2 = 0 Then  
                    Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlRowStripe1)  
                Else  
                    Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlWholeTable)  
                End If  
            ElseIf .ShowTableStyleColumnStripes And .ShowTableStyleRowStripes Then  
                If lRow Mod 2 = 0 And lCol Mod 2 = 0 Then  
                    Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlRowStripe1)  
                ElseIf lRow Mod 2 <> 0 And lCol Mod 2 = 0 Then  
                    Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlColumnStripe1)  
                ElseIf lRow Mod 2 = 0 And lCol Mod 2 <> 0 Then  
                    Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlRowStripe1)  
                Else  
                    Set GetStyleElementFromTableCell = oLo.TableStyle.TableStyleElements(xlWholeTable)  
                End If  
            End If  
        End If  
    End With  
  
End Function

You could use this function like this:

Sub test()  
    Dim oLo As ListObject  
    Dim oTSt As TableStyleElement  
    Set oLo = ActiveSheet.ListObjects(1)  
    Set oTSt = GetStyleElementFromTableCell(ActiveCell, oLo)  
    With ActiveCell.Offset(, 8)  
        .Interior.ThemeColor = oTSt.Interior.ThemeColor  
        .Interior.TintAndShade = oTSt.Interior.TintAndShade  
    End With  
End Sub

Note that the function shown above does not take into account that you can set the width of the stripes, both vertically and horizontally.

**Wrap Up**Of course there is more to learn and know about tables and lists. A good way to come acquainted with the VBA behind them is by recording macro's while fooling around with them. Luckily Microsoft did include the table object if it comes to recording your actions, unlike the omission on the charting side...