LibreOffice Basic Calc

LibreOffice Documents

Current Document

Dim Doc As Object
Doc = ThisComponent

Opening an existing document

In Visible Mode

Dim Doc As Object, DocPath As String
Dim Props() As New com.sun.star.beans.PropertyValue
DocPath = ConvertToURL("C:\Path\To\SomeSpreadsheet.ods")
Props(0).Name = "hidden" 'the document is opened as hidden
Props(0).Value = True
Doc = StarDesktop.loadComponentFromURL(DocPath, "", 0, Props())

In hidden Mode

Dim Doc As Object, DocPath As String
Dim Props() As New com.sun.star.beans.PropertyValue
DocPath = ConvertToURL("C:\Path\To\SomeSpreadsheet.ods")
Props(0).Name = "hidden" 'the document is opened as hidden
Props(0).Value = True
Doc = StarDesktop.loadComponentFromURL(DocPath, ",", 0, Props())

Make visible an hidden document

Doc.CurrentController.Frame.ContainerWindow.toFront()

Creating a new calc document

From (1) the default template or (2) a specified template.

Dim Doc As Object
Doc = ThisComponent

Adding a template (1)

Template = "C:\Path\To\SomeTemplate.ots"

Adding a template (2)

Template = "private:factory/scalc"

Saving a document

The document already exists

(same as File > Save)

Use the document object store method. Ex: ThisComponent.store

The document wasn’t saved yet

(same as File > Save As)

Dim Doc As Object, DocPath As String
Dim Props() As New com.sun.star.beans.PropertyValue
DocPath = ConvertToURL("C:\Path\To\SomeSpreadsheet.ods")
Props(0).Name = "filename" 'the document is saved as new
Props(0).Value = True
Doc = StarDesktop.loadComponentFromURL(DocPath, "", 0, Props())

In case of a duplicate, it becomes the active document.

Saving a copy

As above, using Doc.storeAsURL(DocPath, Props())

The copy won’t become the active document.

Closing a document

Use the document object close method: ThisComponent.close(True)

Getting document information

The document object exposes properties

Location

The document storage directory

Document Properties, File Properties

Author

Author’s name.

Created Date

Creation date.

Modify Date

Last modification date.

Description

Comments.

ModifiedDate

Name of the user who modified the document.

DocumentProperties (Object)

Some more information (see below).

DocumentProperties (File Properties)

ModifyDate Last modification date.

Subject Subj ect (String).

Title String.

UserDefinedProperties Custom properties (Object).

Modifying sheets

Below is the position in the spreadsheet (base 0).

Adding a named sheet name

Doc.Sheets.insertByName(Name, p)

Deleting a sheet

Doc.Sheets.removeByName(FeelsheetName)

Duplicating a sheet

Doc.Sheets.copyByName(SourceName, TargetName, p)

Moving a sheet

Doc.Sheets.moveByName(SourceName, TargetName, p)

Modifying sheets

AsSheet is a sheet object.

Activating a sheet.

Doc.CurrentController.ActiveSheet = AsSheet

Hiding/showing a sheet

AsSheet.Visible = False ' only

Protecting a sheet

protected.kind = AsSheet.protect(Pwd)

Unprotecting a sheet

AsSheet.unprotect(Pwd)

Tab coloring

AsSheet.tabColor = RGB(255, 255, 0)

Linking a sheet

Linking to an existing spreadsheet.

(Excel)

Linking to a file (local lang.)

Filtre, com.sun.star.sheet.SheetLinkMode.VALUE

Destroying a link

AsSheet.unlinkMode(com.sun.star.sheet.SheetLinkMode.NONE)

Finding the last used row/col

AsSheet is a sheet object. Row and Col are the values to retrieve.

Dim Cur As Object

'the cell cursor

Dim Range As Object

'the used range

Dim Row As Long, Col As Long

Cur = AsSheet.getCellByName("A1")

Cur.getCellByName("A1")

Doc.Sheets.copyByName(SrcName, TgtName, p)

Found the last used row

Row = AsSheet.getCellRangeByName("A1")

End If

Cells

Below, Acell is a cell object.

Getting cells

AsSheet is a sheet object. You can get the cell object:

Through its R/C coordinates

AcCell = AsSheet.getcellByRange("A4")

Through its name

AcCell = AsSheet.getCellByName("MYVAL")

Through its X/Y coordinates

AcCell = AsSheet.getCellByPosition(0,3)

with X:0 (col.A) ; Y=3 (row 4)

Knowing the active cell

Doc is a document object and Acell is the active cell we’re looking for.

If Doc.currentSelection.supportsService

AcCell = Doc.currentSelection

End If

Selecting a cell

ThisComponent.CurrentController.select(AcCell)

Cell Coordinates

Coordinates (Object)

Coord = AcCell.cellAddress

Sheet index (Integer)

NumSh = AcCell.cellAddress.Sheet

Column index (Long)

NumCol = AcCell.cellAddress.Column

Row index (Long)

NumRow = AcCell.cellAddress.Row

Container sheet object

AsSheet = AcCell.spreadsheet

Absolute coordinates (String)

Coord = AcCell.AbsoluteName

(Un)Protecting cells

The AxCell.CellProtection property can get the boolean values:

No modifications CellProtection.IsLocked = True

Hidden formula CellProtection.IsFormulaHidden = True

Hide cell CellProtection.IsHidden = True

Don’t print cell CellProtection.IsPrintHidden = True

Getting a cell contents

Properties

Getting text contents

MyText = AcCell.String

Getting numerical contents

MyNumber = AcCell.Value

Getting a formula (US)

MyFormula = AcCell.Formula

Getting a formula (local lang.)

MyFormula = AcCell.FormulaLocal

Knowing content type (below)

TheType = AcCell.Type

Emptying a cell

AcCell.String = ""

Content Types (Type property)

The com.sun.star.sheet.CellContentType.XXX constants can tell the type of contents

(AcCell.Type, above):

EMPTY Empty cell

VALUE Text contents

FORMULA Contents is a formula

Writing in a cell

Replacing existing text

AcCell.String = "Hello!"

Replacing existing value

AcCell.Value = 1,234 'true

Replacing existing formula

AcCell.Formula = "=AND(A1=YES;A2=OK)"

Version Information

LibOBasic_03_Calc_Flat_A4_EN_v115.odt

Written using LibreOffice v.5.3.3 – Platform: All

LibRef nr3
Ranges

Range is a set of cells (one cell is a range): Dim MyRange As Object

Getting Ranges

ASheet is asheet object. You get the ARange range object:

By its coordinates
ARange = ASheet.getCellRangeByName("C2:G14")

By its name
ARange = ASheet.getRangeByName("NomDePlage")

By its coordinates (X1,Y1,X2,Y2)
ARange = ASheet.getRangeByPosition(2, 1, 6, 13)

Arbitration
ARange = ThisComponent.Cells.getRangeByNamePosition(1, 3, 2, 1, 6, 13)

Getting the active range

Like for the active cell, but verify the "com.sun.star.sheet.CellSheetRange" or "[[...], SheetCellRange]" services.

Selecting a range

ThisComponent.CurrentController.getActiveRange (where ARange is an object).

Range Coordinates

<table>
<thead>
<tr>
<th>Coordinates (Object)</th>
<th>Coord = ARange.RangeAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sheet index (Integer)</td>
<td>Rang = ARange.RangeAddress, Sheet</td>
</tr>
<tr>
<td>Column index (Long)</td>
<td>NumCUL = ARange.RangeAddress.StartColumn</td>
</tr>
<tr>
<td>top/left Row index (Long)</td>
<td>NumLTL = ARange.RangeAddress.StartRow</td>
</tr>
<tr>
<td>bottom/right Row index (Long)</td>
<td>NumLBR = ARange.RangeAddress.EndRow</td>
</tr>
<tr>
<td>Containing sheet object</td>
<td>ASheet = ARange.Spreadsheet</td>
</tr>
<tr>
<td>Absolute coordinates (String)</td>
<td>Coord = ARange.AbsoluteName</td>
</tr>
</tbody>
</table>

Named ranges

The Doc object points to the document. Also Ranges As Object

Named ranges

Ranges = Doc.NamedRanges

Ranges (count (long))
Count = Ranges.Count

Getting a range (by index)
ARange = Ranges(Index)

Checking existence by (name)
Existe = Ranges.hasByName(Name)

Getting a range by (name)
ARange = Ranges.getByName(Name)

Adding a range
ARange.addNewByIndex(Name, Coord, CellRef, ARange.Address, Coord = ARange.AbsoluteName)

Removing by (name)
Ranges.removeByName(Name)

Clearing a range

Clearing a Range contents

ARange.ClearContents(DeleteMode)

DeleteMode is a value that specifies the deletion type. com.sun.star.sheet.CellFlags.XXX constants give that choice (combine with +):

- ANNOTATION
- Comments
- FORMULA

Getting a range contents Cells

ARange.DataArray est un tableau des valeurs des cellules pour ARange.

Copying a range contents to another range

Two ranges Source and Target, of the same dimensions.

Copying the contents (values) of Source into Target.DataArray = Source.DataArray Target.

Writing values into a range

ARange is a range object, Table is a table of the same dimensions, which values have to be copied into the range.

Dim Table As Variant
Table = ARange.DataArray 'Table has the same dimensions as the range 
 fête (the array items values)
ARange.DataArray = Table

DataArray is a nested array: use .DataArray(i)(j)

browsing a range cells

From a collection (Ranges, Cells), we create an enumeration. It is then browsed by calling its HasMoreElements and NextElement properties:

Dim Ranges As Object
Ranges = ThisComponent.createInstance("com.sun.star.sheet.CellSheetRanges")
Ranges.insertByName("some_name", ARange)

TheEnum = Ranges.CellCreateEnumeration
While DoWhile TheEnum.hasMoreElements
TheCell = TheEnum.NextElement
Do smthg with the cell object
Loop

Empty cells are not browsed!
Ranges misc.

Merging ARange object cells

ARange.Merge

Ranges types

The range access mode defines which service it implements:

- oCols = com.sun.star.sheet.CellRange
- oCol = com.sun.star.sheet.Cell
- oCell = com.sun.star.sheet.Cell
- oRows = com.sun.star.sheet.TableRange
- oRow = com.sun.star.sheet.TableCell
- oSheet = com.sun.star.sheet.Sheet
- oNamedRange

The implemented service implies how the ranges should be used.
Check using their supportsService() method (rsbellow)

Range or cell?

To know an object type, test supportsService() on a Range or Cell:
If Obj.supportsService(Service_Name) Then ...
Replace Service_Name with:

- com.sun.star.sheet.Sheet
- com.sun.star.sheet.SheetCell
- com.sun.star.sheet.Table
- com.sun.star.sheet.TableCell
- com.sun.star.sheet.TableRange
- com.sun.star.sheet.TableCellRange
- com.sun.star.sheet.TableCell

Always check for a Cell type before a Simple Range as a cell is also a Simple Range!