LibreOffice

LibreOffice Dokumentationsteam

Kurzanleitung

„Makroprogrammierung“

Writer Calc Impress Draw Base Math

LibreOffice ist ein eingetragenes Markenzeichen der The Document Foundation.
Weitere Informationen finden Sie unter http://de.libreoffice.org

Copyright

Dieses Dokument unterliegt dem Copyright © 2010-2014. Die Beitragenden sind unten aufgeführt. Sie dürfen dieses Dokument unter den Bedingungen der GNU General Public License (*http://www.gnu.org/licenses/gpl.html*), Version 3 oder höher, oder der Creative Commons Attribution License (*http://creativecommons.org/licenses/by/3.0/*), Version 3.0 oder höher, verändern und/oder weitergeben.

Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.

Fast alle Hardware- und Softwarebezeichnungen sowie weitere Stichworte und sonstige Angaben, die in diesem Buch verwendet werden, sind als eingetragene Marken geschützt.

Da es nicht möglich ist, in allen Fällen zeitnah zu ermitteln, ob ein Markenschutz besteht, wird das Symbol (R) in diesem Buch nicht verwendet.

Autoren

Jochen Schiffers Klaus-Jürgen Weghorn

Personen, die Makros zur Verfügung gestellt haben

Frieder Delor Thomas Krumbein Dennis Roczek

Rückmeldung (Feedback)

Kommentare oder Vorschläge zu diesem Dokument können Sie in deutscher Sprache an die Adresse *discuss@de.libreoffice.org* senden.

Vorsicht 	Alles, was an eine Mailingliste geschickt wird, inklusive der E-Mail-Adresse und anderer persönlicher Daten, die die E-Mail enthält, wird öffentlich archiviert und kann nicht gelöscht werden. Also, schreiben Sie mit Bedacht!

Datum der Veröffentlichung und Softwareversion

Veröffentlicht am 16.02.2014. Basierend auf der LibreOffice Version 4.1.3.

Inhalt

Anmerkung für Macintosh Nutzer **4**
Einführung in diese Kurzanleitung **4**
Allgemeines 4
Was müssen Sie bei einer Veröffentlichung von Makros beachten? 5
Grundlagen für die Erstellung von Makros **6**
IDE 6
Makrorekorder 6
Komponentenübergreifende Makros **7**
Datei aus dem Internet in ein Verzeichnis auf dem eigenen Rechner herunterladen 7
Datumsfelder 10
Text unformatiert aus der Zwischenablage einfügen 11
Verzeichnissen aktualisieren 12
Makros für Writer **14**
Makro, das den Text "Hallo" an das Ende des Dokuments hängt 14
Tabelle sortieren 14
Text aus der Zwischenablage holen 19
Makros für Calc **21**
Tabelle als PDF-Dokument exportieren 21
Makros für Base **26**
Filter setzen 26
Weitere Makrobeispiele **27**
Tools **28**
X-Ray 28
Info-Quellen **28**
Deutschsprachig 28
Englisch 28

Anmerkung für Macintosh Nutzer

Einige Tastenbelegungen (Tastaturkürzel) und Menüeinträge unterscheiden sich zwischen der Macintosh Version und denen für Windows- und Linux-Rechner. Die unten stehende Tabelle gibt Ihnen einige grundlegende Hinweise dazu. Eine ausführlichere Aufstellung dazu finden Sie in der Hilfedatei des jeweiligen Moduls.

Windows/Linux	***entspricht am Mac***	***Effekt***
Menü **Extras → Optionen...**	**LibreOffice → Einstellungen**	Zugriff auf die Programmoptionen
Rechtsklick	*Control*+Klick	öffnet ein Kontextmenü
Strg (Steuerung)	⌘ (Command)	(Tastaturkürzel in Verbindung mit anderen Tasten)
F5	*Shift*+⌘+*F5*	öffnet den Navigator
F11	⌘+*T*	öffnet den Dialog Formatvorlagen

Einführung in diese Kurzanleitung

Allgemeines

Die Kurzanleitung „Makroprogrammierung“ stellt eine Sammlung verschiedener Makros dar. Dem Anwender soll es durch diese Kurzanleitung ermöglicht werden, zu bestimmten Fragestellungen den Programmiercode nachzulesen bzw. Anregungen für eigene Lösungen zu erhalten.

Die zur Verfügung gestellten Makros sind nicht als benutzbare Programme angedacht und stellen somit primär kein Repository mit Software dar.

Diese Kurzanleitung wird kontinuierlich um weitere Makros ergänzt. Anwender können eigene Makros per Mail an discuss@de.libreoffice.org senden. Das Dokumentations-Team von LibreOffice wird diese Makros dann in dieser Kurzanleitung einarbeiten.

Vorsicht 	Alles, was an eine Mailingliste geschickt wird, inklusive der E-Mail-Adresse und anderer persönlicher Daten, die die E-Mail enthält, wird öffentlich archiviert und kann nicht gelöscht werden. Also, schreiben Sie mit Bedacht!

Falls Sie ein Makro kommentieren möchte, ist dies ausdrücklich erwünscht. Bitte Kommentar auf der de-discuss-ML discuss@de.libreoffice.org posten.

Was müssen Sie bei einer Veröffentlichung von Makros beachten?

Denken Sie beim Veröffentlichen bitte daran, dass alle Makros und Beispieldateien von Anfang an unter einer bestimmten Lizenz veröffentlicht werden müssen. Mehr dazu unter *https://wiki.documentfoundation.org/Macros/de#Lizenzbedingungen*.

Alle Dateien, die in dieser Kurzanleitung eingestellt werden, stehen grundsätzlich unter der Creative Commons Attribution-ShareAlike 3.0 Unported License (CC BY-SA), solange nichts anderes angegeben wird.

Grundlagen für die Erstellung von Makros

Grundlagen s. Handbuch "Erste Schritte" → Kapitel 13 „Einführung in Makros“

ODT-Version

PDF-Version

IDE

Eine integrierte Entwicklungsumgebung (Integrated development environment, kurz IDE) ist eine Zusammenstellung von Programmierwerkzeugen, um die Erstellung von Software zu vereinfachen. LibreOffice enthält sehr leistungsfähige Werkzeuge, mit denen Sie Ihre Makros ausführen, bearbeiten und Fehler darin finden können. Der große Bereich in der Mitte, in dem der Makrocode angezeigt wird, ist das Editor-Fenster. Viele Funktionen wie Stopp, Haltepunkt, Einzelschritt und das Beobachtungsfenster dienen als einfacher und effektiver Debugger für Makrocode.

Makrorekorder

Derzeit wird der Makrorekorder in LibreOffice als „experimentelles“ Programmfeature eingestuft, da er in bestimmten Fällen Fehler bzw. Ungereimheiten verursachen kann. Um den Makrorekorder nutzen zu können, müssen Sie unter {{bc|Extras|Optionen...|LibreOffice|Erweitert}} die Option „Ermöglicht eine Makroaufzeichnung (eingeschränkt)“ aktivieren.

Komponentenübergreifende Makros

Datei aus dem Internet in ein Verzeichnis auf dem eigenen Rechner herunterladen

Beschreibung:

Mit diesem Makro kann man eine Datei aus dem Internet in ein Verzeichnis auf dem eigenen Rechner herunterladen. Es funktioniert sowohl unter Windows, als auch unter Linux. Unter Linux benutzt es die Shell-Funktion "wget" zum herunterladen der Datei. Unter Windows greift es auf die Bibliothek "urlmon" der Windows API zu.

Im Sub "DownloadToFile" müssen noch die Internetadresse und der Name der Datei angepasst werden. Ihr könnt es aber auch gerne mit den angegebenen Daten probieren. Dann wird das "Abrechnungs-Tool" heruntergeladen.

Code:

```
Option Explicit
'-------------------------------------------
Sub DownloadToFile
Dim iSystem%
Dim sURL As String
Dim sPath$
'internetAdresse muss ein direckter Link zum Download sein
sURL="http://wurzelmanager.blogger.de/getfile?name=abrechnungs_tool2.1.1.ods"
'pfad auf dem Rechner
sPath = GetPath
If sPath = "" Then
  MsgBox "Sie haben kein Verzeichnis ausgewählt" _
         ,0 ,"Fehler"
  Exit sub
End if
sPath =  ConvertFromUrl(sPath)
sPath = sPath  & "abrechnungs_tool2.1.1.ods" 'bitte anpassen
iSystem = GetGUIType
  select case iSystem
  Case 1 'Das Betriebssystem ist Windows
       Win_Download (sURL, sPath )
  case 3 'Mac os
```

```
    MsgBox "Leider funktioniert das Makro nicht unter Mac-OS." _
          ,0 ,"Fehler"
  case 4  'Unix oder Linux
       Linux_Download (sURL, sPath )
  case else
     MsgBox "Das Betriebssystem konnte nicht ermittelt werden." _
          ,0 ,"Fehler"
  end select
end sub
'-----------------------------------------------------------------------------------------
Sub Linux_Download (sURL As String, sPath As String )
Dim iVar%,i%
dim dummy()
  if FileExists(sPath)Then
      iVar = MsgBox ("Die Datei " & Chr(10) & sPath & Chr(10) & " existiert bereits." &
    Chr(10) & _
                 "Soll die vorhandene Datei überschrieben werden?",4,"Fehler")
      if iVar =7 Then exit Sub
  End if
'Datei herunterladen
  Shell("wget -q " & sURL &" -O " & """ & sPath & """)
  For i=1 To 10
    Wait 1000
    If FileExists(sPath) Then Exit For 'bis zu 10 sekunden Warten, bis der download
    abgeschlossen ist
  Next
  If Not FileExists(sPath) Then 'Fehler nach 10 sekunden
    MsgBox "Die Datei konnte nicht heruntergeladen werden. " & Chr(10) & _
       "Bitte überprüfen sie die Internetadresse." ,0, "Fehler"
    exit Sub
  Else
    MsgBox "Der Download war erfolgreich. " ,0, "Erfolg"
  End If
End Sub

'-----------------------------------------------------------------
```

```
Sub Win_Download(sURL As String, sPath As String )
Dim iVar%
  if FileExists(sPath)Then
      iVar = MsgBox ("Die Datei " & Chr(10) & sPath & Chr(10) & " existiert bereits." &
    Chr(10) & _
      "Soll die vorhandene Datei überschrieben werden?",4,"Fehler")
      if iVar =7 Then exit Sub
  End if
'Datei herunterladen
  If DownloadFile(sURL, sPath) = False Then
    MsgBox "Die Datei konnte nicht heruntergeladen werden. " & Chr(10) & _
      "Bitte überprüfen sie die Internetadresse." ,0, "Fehler"
    exit Sub
  Else
    MsgBox "Der Download war erfolgreich. " ,0, "Erfolg"
  End If
End Sub
'-------------------------------------------------------------------------------
'Die Funktion "urlmon" aus der Windows API aufrufen
Private Declare Function URLDownloadToFile Lib "urlmon" Alias
    "URLDownloadToFileA" _
                  (ByVal pCaller As Long, _
                   ByVal szURL As String, _
                   ByVal szFileName As String, _
                   ByVal dwReserved As Long, _
                   ByVal lpfnCB As Long) As Long
'-------------------------------------------------------------------------
Public Function DownloadFile(URL As String, LocalFilename As String) As Boolean
  Dim lngRetVal As Long
  lngRetVal = URLDownloadToFile(0, URL, LocalFilename, 0, 0)
  If lngRetVal = 0 Then DownloadFile = True
End Function
'-------------------------------------------------------------------------
'Ordner über den Ordnerauswahl-Dialog holen
Function GetPath() As String
  Dim  oPathSettings, oFolderDialog
```

```
    Dim sPath As String
     oPathSettings = CreateUnoService("com.sun.star.util.PathSettings")
     sPath = oPathSettings.Work
     oFolderDialog = _
          CreateUnoService("com.sun.star.ui.dialogs.FolderPicker")
     oFolderDialog.SetDisplayDirectory(sPath)
     If oFolderDialog.Execute() = _
          com.sun.star.ui.dialogs.ExecutableDialogResults.OK Then
       sPath = oFolderDialog.GetDirectory
     Else
       GetPath = ""
       Exit Function
     End If
   If Right(sPath, 1) <> "/" Then sPath = sPath & "/"
   GetPath = sPath
 End Function
```

Datumsfelder

Datumsfelder ab der LibreOffice-Version 4.1.1

Die Date-Eigenschaft nimmt keine Zahl (long) entgegen, sondern erwartet einen Struct des Typs com.sun.star.util.Date. Um das Datum zu setzen, ist folgender Weg notwendig:

```
dim oDat as new com.sun.star.util.Date
with oDat
.day = Day(now)
.month = Month(now)
.year = Year(now)
end with
oDlg.getControl("meinDatumsKontrollfeld").date = oDat
```

Der Datumswert kann entweder als Struct wieder ausgelesen und entsprechend umgewandelt werden oder man nutzt den angezeigten (Text-) Wert und wandelt diesen in

einen internen Datumswert um.

```
...
dDatum = CDate(oDlg.getControl("meinDatumsKontrollfeld").getAccessibleContext.text)
...
```

Datumsfelder bis zur LibreOffice-Version 4.0

Mit der LibreOffice-Version 4.1.1 wurden bezüglich der Datumsfelder einige entscheidende Änderungen in der API vorgenommen. Eine wesentliche Veränderung betrifft die Datumsfelder in Dialogen (Datums-Kontrollfelder). Die Nutzung von Datumsfelder, die mit Version älter als 4.1.1 erstellt worden sind, führt zu der Fehlermeldung "Objekt nicht definiert".

Was ist geändert worden?

Das Datumskontrollfeld hatte bisher eine Eigenschaft, die das angezeigte Datum repräsentierte. Diese Date-Eigenschaft konnte per Makro gesetzt oder auch ausgelesen werden.

Das Datum wurde dabei im ISO-Format als Zahl (Long) übergeben bzw. ausgelesen.

Bisheriger Weg:

```
oDlg.getControl("meinDatumsKontrollfeld").date = CDatetoIso(now())  REM Setzen des aktuellen Datums
```

Alle bisherigen Programmierungen müssen also nun umgeschrieben und Makros entsprechend angepasst werden.

Das trifft im übrigen auch für das Auslesen des Datumswertes zu. Die bisherige Methode, das ISO-Format einfach wieder zurück zu wandeln, ist nicht mehr möglich.

Auch entspricht der Date-Wert nicht immer dem angezeigten Zahlenwert - dieser ist ja "nur" die Textdarstellung des Datumswertes - und kann manuell im Feld geändert werden.

Text unformatiert aus der Zwischenablage einfügen

Beschreibung:

Hier eine einfache Funktion, mit der man Text unformatiert aus der Zwischenablage holen kann.

Einfach die Funktion in ein Basic-Modul kopieren, und aus einer beliebigen anderen Funktion oder Sub aufrufen, z.B. so:

```
sub clpboardTest
```

```
  MsgBox (getClipboardText)
end sub
```

Code (Haupt-Funktion):

```
Function getClipboardText () AS String
dim oClip as object ,oConverter as object
dim oClipContents as object ,oTypes as object
dim i%
oClip = createUnoService("com.sun.star.datatransfer.clipboard.SystemClipboard")
oConverter = createUnoService("com.sun.star.script.Converter")
On Error Resume Next
oClipContents = oClip.getContents
oTypes = oClipContents.getTransferDataFlavors
For i=LBound(oTypes) To UBound(oTypes)
  If oTypes(i).MimeType = "text/plain;charset=utf-16" Then
    Exit For
  End If
Next
If (i >= 0) Then
  On Error Resume Next
  getClipboardText = oConverter.convertToSimpleType _
                (oClipContents.getTransferData(oTypes(i)),
  com.sun.star.uno.TypeClass.STRING)
End If
End Function
```

Verzeichnissen aktualisieren

Beschreibung:

Sehr häufig zeigt sich das Problem, dass die Verzeichnisse, insbesondere das Inhaltsverzeichnis, nach Änderungen nicht aktualisiert werden. Es werden dann Dokumente mit fehlerhaften Verzeichnissen übergeben.

Dieses Makro aktualisiert automatisch die Verzeichnisse. Wird dieses Makro dem Befehl Dokument speichern (Extras ► Anpassen ► Ereignisse ► Dokument sichern) zugeordnet, werden die Verzeichnisse beim Speichern automatisch aktualisiert.

Das folgende Makro ist für die LO-Komponente Writer erstellt worden.

Code:

```
Sub VerzeichnisseAktualisieren
  if thisComponent.supportsService ("com.sun.star.text.GenericTextDocument") then
  for i = 0 TO thisComponent.getDocumentIndexes().count - 1
    thisComponent.getDocumentIndexes().getByIndex(I).update()
  NEXT I
else
end if
End Sub
```

Makros für Writer

Makro, das den Text "Hallo" an das Ende des Dokuments hängt

```
Sub HalloalsAnhang
Dim oDoc
Dim sTextService$
Dim oCurs
REM Diese Komponente betrifft das aktuelle Dokument.
oDoc = ThisComponent
REM Überprüfen, ob dies ein Textdokument ist
sTextService = "com.sun.star.text.TextDocument"
If NOT oDoc.supportsService(sTextService) Then
MsgBox "Dieses Makro arbeitet nur mit einem Textdokument"
Exit Sub
End If
REM Abfrage des Zeigers von der aktuellen Steuerung.
oCurs = oDoc.currentController.getViewCursor ()
REM Bewege den Mauszeiger zum Ende des Dokuments.
oCurs.gotoEnd (False)
REM Fügt den Text "Hallo" am Ende des Dokuments ein
oCurs.Text.insertString(oCurs, "Hallo", False)
End Sub
```

Tabelle sortieren

Beschreibung:

Dieses Makro sortiert Writer-Tabellen nach der Spalte, in der sich gerade der Cursor befindet.

Dabei gibt es 4 Möglichkeiten: Aufsteigend und Absteigend, für Tabellen mit oder ohne Kopfzeile.

Dieses Makro ist auch als fertige Extension auf der LibreOffice Extension Seite verfügbar (*Link zu der Seite: Writer Tabellen sortieren*).

Die Extension fügt in Writer eine neue Symbolleiste mit 4 Schaltflächen ein, die mit den entsprechenden Makros verknüpft sind.

Code:

```
Option Explicit
'----------------------------------------
Sub sortWriterTableDescendingHeader
dim sAktivCell$
Dim sTableName As String
Dim bAscending As boolean
Dim bHeader As boolean
Dim nSortColl As Long
  On Error GoTo ErrorHandler
  sTableName =
    ThisComponent.CurrentController.Selection.getByIndex(0).TextTable.Name
  sAktivCell = ThisComponent.CurrentController.Selection.getByIndex(0).Cell.CellName
  bAscending = False 'Aufsteigend sortiert
  bHeader = True 'Tabelle enthält Kopfzeile
  nSortColl = getColumn(sAktivCell) 'Spalte nach der sortiert werden soll (erste
    Spalte=1)
  sort_Text_Table(sTableName,bAscending,bHeader,nSortColl)
ErrorHandler:
End Sub
'----------------------------
Sub sortWriterTableAscendingHeader
dim sAktivCell$
Dim sTableName As String
Dim bAscending As boolean
Dim bHeader As boolean
Dim nSortColl As Long


  On Error GoTo ErrorHandler
  sTableName =
    ThisComponent.CurrentController.Selection.getByIndex(0).TextTable.Name
  sAktivCell = ThisComponent.CurrentController.Selection.getByIndex(0).Cell.CellName



  bAscending = True 'Aufsteigend sortiert
  bHeader = True 'Tabelle enthält Kopfzeile
```

```
  nSortColl = getColumn(sAktivCell) 'Spalte nach der sortiert werden soll (erste
    Spalte=1)
  sort_Text_Table(sTableName,bAscending,bHeader,nSortColl)
ErrorHandler:
End Sub
'---------------------------------
Sub sortWriterTableAscending
dim sAktivCell$
Dim sTableName As String
Dim bAscending As boolean
Dim bHeader As boolean
Dim nSortColl As Long
  On Error GoTo ErrorHandler
  sTableName =
    ThisComponent.CurrentController.Selection.getByIndex(0).TextTable.Name
  sAktivCell = ThisComponent.CurrentController.Selection.getByIndex(0).Cell.CellName
  bAscending = True 'Aufsteigend sortiert
  bHeader = False 'Tabelle enthält Kopfzeile
  nSortColl = getColumn(sAktivCell) 'Spalte nach der sortiert werden soll (erste
    Spalte=1)
  sort_Text_Table(sTableName,bAscending,bHeader,nSortColl)
ErrorHandler:
End Sub
'-----------------------------------------
Sub sortWriterTableDescending
dim sAktivCell$
Dim sTableName As String
Dim bAscending As boolean
Dim bHeader As boolean
Dim nSortColl As Long
  On Error GoTo ErrorHandler
  sTableName =
    ThisComponent.CurrentController.Selection.getByIndex(0).TextTable.Name
  sAktivCell = ThisComponent.CurrentController.Selection.getByIndex(0).Cell.CellName
  bAscending = False 'Aufsteigend Sortiert
  bHeader = False 'Tabelle enthält kopfzeile
```

```
  nSortColl = getColumn(sAktivCell) 'Spalte nach der sortiert werden soll (erste
    Spalte=1)
  sort_Text_Table(sTableName,bAscending,bHeader,nSortColl)
ErrorHandler:
End Sub
'-----------------------------------------------------------------
sub sort_Text_Table (sTableName As String,bAscending As boolean,bHeader As
    boolean, nSortColl As Long)
Dim oTable as Object
dim oRange as Object
Dim nStartRow As Long
Dim sRange As String
Dim oSortFields(0) as New com.sun.star.table.TableSortField
Dim oDescriptor As Variant
  oTable = ThisComponent.TextTables.getByName(sTableName)
    If bHeader Then
      nStartRow =1
    Else
      nStartRow =0
    End if
  sRange = GetTablerange(oTable,nStartRow)
  oRange = oTable.getCellRangeByName (sRange)
  oSortFields(0).Field = nSortColl
  oSortFields(0).IsAscending = bAscending
  oSortFields(0).FieldType = com.sun.star.util.SortFieldType.ALPHANUMERIC
  oDescriptor = oTable.createSortDescriptor()
  ' set descriptor's properties
  oDescriptor(0).Name = "IsSortInTable"
  oDescriptor(0).Value = True
  'oDescriptor(1).Name = "Delimiter"
  'oDescriptor(1).Value = True
  oDescriptor(2).Name = "IsSortColumns"
  oDescriptor(2).Value = False 'True
  'oDescriptor(3).Name = "MaxSortFieldsCount"
  'oDescriptor(3).Value = 2
  oDescriptor(4).Name = "SortFields"
```

```
  oDescriptor(4).Value = oSortFields()
  oRange.Sort(oDescriptor())
End sub
REM Returns the name of the sortrange
Function GetTablerange (oTable As Object, nStartRow As Long) As String
  Dim sCell As string
  Dim oCursor As Object
  sCell = oTable.getCellByPosition(0, nStartRow).CellName
  oCursor = oTable.createCursorByCellName (sCell)
  oCursor.gotoEnd (True)
  GetTablerange = oCursor.getRangeName
End Function
'-----------------------------------------------------------------
'  Returns the index of the sortcolumn
Function getColumn (sAktivCell As String) As Long
Dim sTableName$
Dim i% ,n%
dim sChar$
Dim nCol as Long
n=0
nCol=0
For i=Len(sAktivCell) to 1 Step -1
 sChar = UCase ( Mid(sAktivCell,i,1))
 If Not IsNumeric(sChar) Then
   nCol = (Asc(sChar)-64)*(26^n) + nCol
   n=n+1
 end if
Next
getColumn = nCol
End Function
```

Text aus der Zwischenablage holen

Beschreibung:

Dieses Makro fügt den Inhalt aus der Zwischenablage als unformatierten und absatzlosen Text ein.

Um es effizient zu nutzen empfiehlt es sich dem Sub "insert_Clpboard_Text_in_Writer" eine Tastenkombination zuzuweisen (Extras ► Anpassen: Tastatur).

Um nur unformatierten Text aus der Zwischenablage einzufügen kann das Modulübergreifende Makro getClipboardText() verwendet werden.

Code:

```
Option Explicit
sub insert_Clpboard_Text_in_Writer
dim sText As string
  sText= (getClipboardText)
  'Zeilenumbrüche durch Leerzeichen ersetzen.
  sText = Replace (sText,Chr(10)," ")
  'Absatzumbrüche durch Leerzeichen ersetzen.
  sText = Replace (sText,Chr(13)," ")
  'In Writer einfügen
  Write_Cursor_Position (sText)
end sub
'-------------------------
' Das Sub "Write_Cursor_Position" ist von:
   http://www.oooforum.org/forum/viewtopic.phtml?t=75409
'Fügt text in Writer an der cursor Position ein
Sub Write_Cursor_Position (sText as String)
Dim oViewCursor as object
dim oText As Object
  oViewCursor= thiscomponent.GetCurrentController.ViewCursor
  If IsEmpty(oViewCursor.Cell) Then
    oText=thiscomponent.text
  Else
    oText=oViewCursor.Cell.Text
  End If
  oText.insertString(oViewCursor, sText,false)
```

```
end Sub

'holt Text aus zwischenablage, und entfernt formatierungen
Function getClipboardText () AS String
dim oClip as object ,oConverter as object
dim oClipContents as object ,oTypes as object
dim i%
oClip = createUnoService("com.sun.star.datatransfer.clipboard.SystemClipboard")
oConverter = createUnoService("com.sun.star.script.Converter")
On Error Resume Next
oClipContents = oClip.getContents
oTypes = oClipContents.getTransferDataFlavors
For i=LBound(oTypes) To UBound(oTypes)
  If oTypes(i).MimeType = "text/plain;charset=utf-16" Then
    Exit For
  End If
Next
If (i >= 0) Then
  On Error Resume Next
  getClipboardText = oConverter.convertToSimpleType _
                  (oClipContents.getTransferData(oTypes(i)),
    com.sun.star.uno.TypeClass.STRING)
End If
End Function
```

Makros für Calc

Tabelle als PDF-Dokument exportieren

Beschreibung:

Ein Makro, mit dem man eine einzige Tabelle als PDF exportieren kann.

Eine saubere, Betriebssystem-unabhängige Lösung ist: die Tabelle als "Druckbereich" festlegen und dann als PDF exportieren: hier der Code: im ersten "Sub Main" muss der "Deine Tabellen Name" angepasst werden. Ansonsten ist es komplett einsatzbereit. Einfach das Sub Main aus einer Calc Datei ausführen und fertig.

Code:

```
Sub Main
ExportToPDF ("Deine Tabellen Name")
end sub
Sub ExportToPDF (sTableNam As String)
  sPath = GetPath
  If sPath = "" Then
    MsgBox "Kein gültiges Verzeichnis.", 16, "Fehler"
    goTo Zeile1
  End If
  Zeile0:
  sStandard = "PDF_Export_" & Format(Now, "hh-mm")& "Uhr_" & _
                Format(Now,"dd.mm.yyyy")
  sName=InputBox ("Bitte geben Sie einen Namen" & Chr(10) & _
               "für die PDF-Datei ein" , "PDF Name", sStandard )
  If sName="" Then
    nVar=MsgBox ("Sie haben keinen Name eingegeben. " & Chr(10) & _
              "Bitte geben Sie einen Namen ein. " & Chr(10) & _
              "Wenn Sie auf ""Abbrechen"" klicken, " & Chr(10) & _
              "wird der Export abgebrochen.", 1, "Fehler")
    If nVar=1 then 'OK wurde gedrückt
      goTo Zeile0
    Else
      goTo Zeile1
    End if
```

```
  End if
  sFileName= sName & ".pdf"
  If FileExists(sPath & sFileName) then
       nVar=MsgBox ("Die Datei existiert bereits. " & Chr(10) & _
            "Soll die Datei überschrieben werden?. " & Chr(10) & _
            Chr(10) & _
            "Wenn Sie auf ""Abbrechen"" klicken, " & Chr(10) & _
            "wird der Export abgebrochen." & Chr(10) & _
            "Wenn Sie auf ""Nein"" klicken," & Chr(10) & _
            "können Sie einen anderen Namen wählen.", 3, "Fehler")
    Select Case nVar
    Case 2 'Abbrechen wurde gedrückt
      goTo Zeile1
    Case 6 'Ja wurde gedrückt
    Case 7 'Nein wurde gedrückt
      goTo Zeile0
    End Select
  end if
  ThisComponent.addActionLock
  ThisComponent.LockControllers
  oDoc = ThisComponent
  oSheets = oDoc.Sheets
  oSheet1 =oDoc.Sheets.getByName(sTableNam)
  Delete_PrintAreas
For n = 0 To oSheets.getCount - 1
  oSheet=oDoc.Sheets(n)
  if oSheet.Name= sTableNam Then
  lEndCol= GetLastUsedColumn(oSheet1)
  lEndRow=GetLastUsedRow(oSheet1)
    Set_PrintAreas (n ,0 ,0 ,lEndCol ,lEndRow)
  End if
Next
  export_pdf(sPath & sFileName)
  Delete_PrintAreas
  MsgBox "Das PDF wurden erfolgreich erstellt." , 0, "PDF Export"
```

```
  Zeile1:
  ThisComponent.UnlockControllers
  ThisComponent.removeActionLock
End sub
'----------------------------------------
sub Delete_PrintAreas 'Alle Druckbereiche löschen
  oDoc = ThisComponent
  oSheet = oDoc.Sheets(0)
  for n =0 to oDoc.Sheets.getCount - 1
    oDoc.Sheets(n).setPrintAreas(Array ())
  Next
end sub
'----------------------------------------
Sub Set_PrintAreas (nSheet As Integer,lStartCol As Long,_
lStartRow As Long,lEndCol As Long,lEndRow As Long) 'Druckbereich festlegen
  Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress
  oDoc = ThisComponent
  oSheet = oDoc.Sheets(nSheet)
  CellRangeAddress.Sheet = nSheet
  CellRangeAddress.StartColumn = lStartCol
  CellRangeAddress.StartRow = lStartRow
  CellRangeAddress.EndColumn = lEndCol
  CellRangeAddress.EndRow = lEndRow
  aPrintAreas()=Array (CellRangeAddress)
  oSheet.setPrintAreas(aPrintAreas())

End sub
'-----------------------------------
Function GetLastUsedRow(oSheet as Object) As Integer
  Dim oCell
  Dim oCursor
  Dim aAddress
  oCell = oSheet.getCellByPosition(0, 0)
  oCursor = oSheet.createCursorByRange(oCell)
  oCursor.gotoEndOfUsedArea(True)
```

```
  aAddress = oCursor.RangeAddress
  GetLastUsedRow = aAddress.EndRow
End Function
REM Gibt die Nummer der letzten Spalte von einer
'kontinuierlichen Datenbereich in einer Mappe zurück
Function GetLastUsedColumn(oSheet as Object) As Long
  Dim oCell
  Dim oCursor
  Dim aAddress
  oCell = oSheet.getCellByPosition(0, 0)
  oCursor = oSheet.createCursorByRange(oCell)
  oCursor.gotoEndOfUsedArea(True)
  aAddress = oCursor.RangeAddress
  GetLastUsedColumn = aAddress.EndColumn
End Function
'---------------------------------
Function GetPath() As String
  Dim  oPathSettings, oFolderDialog
  Dim sPath As String
    oPathSettings = CreateUnoService("com.sun.star.util.PathSettings")
    sPath = oPathSettings.Work
    oFolderDialog = _
         CreateUnoService("com.sun.star.ui.dialogs.FolderPicker")
    oFolderDialog.SetDisplayDirectory(sPath)
    If oFolderDialog.Execute() = _
         com.sun.star.ui.dialogs.ExecutableDialogResults.OK Then
      sPath = oFolderDialog.GetDirectory
    Else
      GetPath = ""
      Exit Function
    End If
  If Right(sPath, 1) <> "/" Then sPath = sPath & "/"
  GetPath = sPath
End Function
'-----------------------------------
```

```
sub export_pdf (sFileName AS String)
  dim args1(1) as new com.sun.star.beans.PropertyValue
  args1(0).Name = "ExportFormFields" 'zeige nur den Inhalt von Form.Fields
  args1(0).Value= True
  args1(1).Name = "Printing" "'Du brauchst das nicht!"
  args1(1).Value= 0
'hier können noch weiter Optionen eingegeben werden
  dim args2(2) as new com.sun.star.beans.PropertyValue
  args2(0).Name = "FilterName"
  args2(0).Value = "calc_pdf_Export"
  args2(1).Name = "FilterData"
  args2(1).Value = args1
  args2(2).Name = "SelectionOnly" 'Das bewirkt,
'dass nur der ausgewählte Druckbereich exportiert wird.
  args2(2).Value = true
  ThisComponent.storeToURL(sFileName,args2())
end sub
```

Makros für Base

Filter setzen

Beschreibung:

Das Base-Dokument "Formular_mit_Filter.odb" (Sorry: Dokument ist verschütt gegangen) enthält zwei Makros, mit denen Datensätze, die mittels eines Formulars dargestellt werden, gefiltert werden können.

Formular "formular2": Das Beispielformular heißt "formular2" und enthält die Formulare MainForm" und "SubForm".

"MainForm" zeigt die Daten der AbfrageXY und enthält die Schaltelemente "Schaltfläche 1" und das Kombinationsfeld "Kombinationsfeld 1".

"Kombinationsfeld 1" : Ist mit der Spalte der Abfrage verknüpft , nach der gefiltert werden soll.
Es ist über das Ereignis "Text Modifiziert" an das Makro "setFilter2" gebunden.
Dadurch wird das Makro automatisch beim Ändern des Inhaltes ausgeführt.

"Schaltfläche 1" ist über das Ereignis "Aktion ausführen" an das Makro "removeFilter2" gebunden.

"SubForm" zeigt die ursprünglichen Daten der Tabelle, und ist für die Funktion ohne Bedeutung.

Bei der Erstellung des Makros „setFilter2“ kann das Problem auftreten, für den Befehl oForm.filter = " ""Name"" LIKE ''' & sFilterstring & """

den entsprechende SQL-Befehl für den gewünschten Filter zu finden. Dies kann mittels eines kleinen Tricks gelöst werden:
Formular „formular2“ mittels der rechten Maustasten zum bearbeiten öffnen
in das obere Tabellen-Kontrollfeld klicken
mittels der rechten Maustaste die Formular-Eigenschaften aufrufen
dort im Reiter „Daten“ im Feld „SQL-Befehl analysieren “ auf "ja" stellen.
Bei dem Feld "Filter" hinten auf die 3 pünktlichen klicken.
den gewünschten Filter einstellen und mit OK bestätigen
in dem Feld erscheint jetzt der gewünschte SQL-Befehl: "Name" LIKE 'asd'

Code:

```
Sub setFilter2
dim oDoc as object
dim oForm as object
dim oComboBox as object
dim sFilterstring As String
thisComponent.lockControllers  'Geschwindigkeit Modus
  oForm = thisComponent.drawpage.forms.getByName("MainForm") 'Mainform holen
```

```
  oComboBox = oForm.getByName("Kombinationsfeld 1") 'Combobox holen
  sFilterstring = oComboBox.Text 'Text holen
  oForm.filter = " ""Name"" LIKE '" &amp;amp; sFilterstring &amp;amp; "'" 'filter eigenschaften
    definieren (Name ist die Spalte)
  oForm.ApplyFilter = true 'filter status setzen
  oForm.reload() 'filter anwenden
thisComponent.unlockControllers 'Normaler Modus um wieder im formular arbeiten zu
    können.
End Sub

Sub removeFilter2
dim oForm as object
thisComponent.lockControllers
  oForm = thisComponent.DrawPage.Forms.GetByName("MainForm")
  oForm.ApplyFilter=False
  oForm.reload()
thisComponent.unlockControllers
end sub
```

Weitere Makrobeispiele

In diese Kurzanleitung werden schrittweise die Makrobeispiele eingearbeitet, im im LO-Wiki aufgeführt sind (https://wiki.documentfoundation.org/Macros/de#Beispielmakros).

Tools

X-Ray

Xray ist ein Werkzeug für Programmierer von Basic - Makros, die die API benutzen. Das Tool listet Eigenschaften, Methoden, Services und Interfaces auf, die zu einer Objekt-Variable gehören. Xray kann in einem Web Browser die offizielle API Dokumentation über Eigenschaften, Methoden, Services und Interfaces auflisten.

http://bernard.marcelly.perso.sfr.fr/index2.html

Info-Quellen

Deutschsprachig

www.ooowiki.de *www.ooowiki.de*

www.dannenhoefer.de/faqstarbasic/index.html *www.dannenhoefer.de*

https://sites.google.com/site/starbasicmakros/makros-1 sites.google.com/site/starbasicmakros

www.uni-due.de/~abi070/ooo.html

Andrew Pitonyak: OpenOffice.org-Makros Erklärt (ins Deutsche übertragen von Volker Lenhardt)

http://wiki.services.openoffice.org/wiki/DE/Makro_Basic_Tutorial
http://wiki.services.openoffice.org/wiki/DE/Makro_Basic_Tutorial (online)

Englisch

LibreOffice Scripts benutzen die LibreOffice API (Application Programmers Interface - Programmierschnittstelle). Die offizielle Dokumentation ist verfügbar unter: *http://api.libreoffice.org* → API Information

LO-Wiki (international): *https://wiki.documentfoundation.org/Macros*

Andrew's Macros (OpenOffice.org Macros):

www.pitonyak.org/oo.php *www.pitonyak.org*

www.pitonyak.org/OOME_3_0.odt

www.pitonyak.org/OOME_3_0.pdf

Externe Sammlungen:

http://labs.lanedo.com

http://labs.lanedo.com/libreoffice/spreadsheet-population

http://labs.lanedo.com/libreoffice/mailmerge

http://codesnippets.services.openoffice.org