LibreOffice

LibreOffice Dokumentationsteam

Kurzanleitung

„Makroprogrammierung“

Copyright

Autoren

Robert Großkopf | Jochen Schiffers | Klaus-Jürgen Weghorn

Personen, die Makros zur Verfügung gestellt haben

Frieder Delor | Thomas Krumbein | Dennis Roczek
Klaus-Jürgen Weghorn

Screenshots

Einige Screenshots wurden aus anderen Dokumenten kopiert. Eine detaillierte Auflistung dieser Dokumente finden Sie im Kapitel Info-Quellen.

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 13.08.2017. Basierend auf der LibreOffice Version 4.3.0

Inhalt

Einführung in diese Kurzanleitung

Allgemeines

Als Grundlage dieser „Kurzanleitung“ dient das Handbuch "Erste Schritte" → Kapitel 13 „Einführung in Makros“.

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. Kommentare senden Sie bitte per eMail an die deutsche discuss-Mailingliste "discuss@de.libreoffice.org".

Was muss bei einer Veröffentlichung von Makros beachtet werden?

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

Deklaration

Es ist empfehlenswert, am Anfang der Prozedur eine Deklaration zu erstellen. Beispiel:

```
' Name der Variable soll "Brutto" sein
Sub Main
DIM oBrutto AS OBJECT
End Sub
```

Option Explicit

Wenn am Anfang des Codes außerhalb der Prozeduren "OPTION EXPLICIT" eingegeben wird, ist es vorbei mit allen Variablen, die nicht vorher definiert wurden. Das kann manchmal ganz heilsam sein, wenn jemand durch Schreibfehler meint, doch überall die gleiche Variable gesetzt zu haben, sich bei der Rechnung aber wundert, warum immer das Zehnfache im Portemonnaie sein müsste, aber leider nur die Rechnung nicht stimmt. Ein kleiner Dreher bei einer Variablen und schon gibt es zwei Variablen mit unterschiedlichen Werten

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 „experimentelle Programmfunktion“ eingestuft, da er in bestimmten Fällen Fehler bzw. Ungereimtheiten verursachen kann. Um den Makrorekorder nutzen zu können, müssen Sie unter **Extras** → **Optionen...** → **LibreOffice** → **Erweitert** die Option *Makroaufzeichnung ermöglichen (eingeschränkt)* aktivieren.

Dispatcher

Der Dispatcher ist ein eigener Service, der die im Core-Code definierten Funktionen aufrufen kann. Dies sind in erster Linie alle Funktionen hinter den Schaltflächen und Icons aller Symbolleisten und noch einige mehr.

Die Funktionsweise des Dispatcher wird ausführlich in dem "Makro-Kochbuch" (Kapitel 4.1) von Thomas Krumbein beschrieben:

www.wollmux.net/wiki/images/f/f9/Makro_Kochbuch.pdf

www.wollmux.net/wiki/Datei:Makro_Kochbuch.odt

Strings, Werte, Arrays und Variablen

Allgemeines

Sehr gute Hinweise zu dem Themenkomplex „Strings, Werten und Arrays“ finden sich in dem "Makro-Kochbuch" (Kapitel 4.2) von Thomas Krumbein:

www.wollmux.net/wiki/images/f/f9/Makro_Kochbuch.pdf

www.wollmux.net/wiki/Datei:Makro_Kochbuch.odt

Variablen-Typen

Zahlen				
Typ	***Entspricht in HSQLDB***	***Startwert***	***Anmerkung***	***Speicherbedarf***
Integer	SMALLINT	0	216 = - 32768 bis + 32767	2 Byte
Long	INTEGER	0	232 = - 2147483648 bis + 2147483647	4 Byte
Single		0.0	Dezimaltrenner: «.»	4 Byte
Double	DOUBLE	0.0	Dezimatrenner: «.»	8 Byte
Currency	Ähnlich DECIMAL, NUMERIC	0.0000	Währung, 4 Dezimalstellen fest	8 Byte
Sonstige Variable-Typen				
Typ	***Entspricht in HSQLDB***	***Startwert***	***Anmerkung***	***Speicherbedarf***
Boolean	BOOLEAN	False	1 = „ja“, alles andere: „nein“	1 Byte
Date	TIMESTAMP	00:00:00	Datum und Zeit	8 Byte
String	VARCHAR	Leerer String	bis 65536 Zeichen	variabel
Object	OTHER	Null		variabel
Variant		Leer	Kann jeden (anderen) Datentyp annehmen	variabel

Variable „ThisComponent“

Was ist „ThisComponent“?

Die Variable „ThisComponent“ ermöglicht einen einfachen Zugang zum aktuellen Dokument. Die (vordefinierte) Variable beinhaltet immer das Objekt des aktuellen Dokumentes – oder genauer: des aktuellen Hauptmoduls und der davon abgeleiteten Komponente, die gerade den Fensterfokus besitzt. Es sind also nur die Hauptkomponenten (Writer, Calc, Draw, Impress oder Base, manchmal jedoch auch die Basic-IDE), deren aktuelles Fenster und das darin befindliche Dokument der Variablen zugewiesen wird.

Intern wechselt somit der Inhalt von „ThisComponent“, sobald man das aktive Fenster wechselt. Das grenzt den Einsatz von ThisComponent deutlich ein. Auf der einen Seite ist es eine bequeme Möglichkeit, Zugang zu dem aktuellen Dokument zu erhalten, auf der anderen Seite ist diese Variable trügerisch, ändert sie doch schnell den Inhalt. Benötigt man also den dauerhaften Zugang zu einem bestimmten Dokument, so muss der Inhalt von ThisComponent einer neuen Variablen zugewiesen und dann nur noch mit dieser gearbeitet werden. Dies geschieht typischerweise ganz am Anfang des Programms – nur so ist sichergestellt, dass das korrekte Dokument gewählt wurde.

Typische Einsatzgebiete von ThisComponent

Die Variable „ThisComponent“ stellt sicher, dass ein Makro immer für die Komponente von LibreOffice abläuft, für die das Makro auch vorgesehen ist.

Dies klingt zwar selbstverständlich. Aber es gibt immer wieder die Situation, dass der/die Benutzer/in zwischenzeitlich ein anderes Dokument aktiviert. Dadurch kann die Ausführung des Makros nicht mehr wie gewünscht ablaufen.

Die Variable „ThisComponent stellt eine direkte Beziehung zum Dokument her, so dass dieses Dokument bearbeitet werden kann bzw. nur auf dieses Dokument zugegriffen wird - auch wenn der/die Benutzer/in zwischenzeitlich ein anderes Dokument aktiviert.

Es folgt ein Beispiel für die korrekte Anwendung der Variablen „ThisComponent“:

```
public oDoc as variant

sub eineWichtigefunktion
  odoc = thisComponent
  oDoc.text.string = "Hallo, hallo"    'ein Text wird eingegeben
  '… hier folgen jetzt jede Menge Anweisungen (intern)
  oDoc.print(args())     'Dokument wird gedruckt
end sub
```

Würde diese Programm wie folgt geschrieben, so könnte es einige „Überraschungen“ geben:

```
sub eineWichtigefunktion
  thisComponent.text.string = "Hallo, hallo"    'ein Text wird eingegeben
  '… hier folgen jetzt jede Menge Anweisungen (intern)
  thisComponent.print(args())     'Dokument wird gedruckt
end sub
```

In der Zeitspanne zwischen dem Schreiben des Textes und dem Drucken könnte der/die Benutzer/in das Dokument auf dem Bildschirm gewechselt haben (durch Klick auf ein anderes

geöffnetes Dokument), in diesem Fall verweist „thisComponent“ nun auf das aktive Dokument und dieses würde gedruckt werden. ThisComponent muss also mit Vorsicht genutzt werden, insbesondere, wenn das Makro nicht direkt aus dem Dokument heraus gestartet wurde (in diesem Fall wäre es nämlich ein Tochterprozess des aktuellen Dokumentes – und der „Rest“ wäre erst einmal blockiert). Insbesondere aber bei Extensions wird der Makro-Prozess unabhängig vom aktuellen Dokument gestartet – und dann kommt es zu den oben beschriebenen Phänomenen.

Praktische Tipps für den Umgang von Variablen

Variablen mit dem Anfangsbuchstaben des jeweiligen Variablen-Typ versehen.

Syntax für Variable-Typen, deren Anfangsbuchstaben nur einmal vorkommt:

Den ersten Buchstaben des Variablen-Typ verwenden.

Beispiel für den Variablen-Typ OBJECT:

Name der Variable soll „Brutto“ sein
Syntax: oBrutto

Syntax für Variable-Typen, deren Anfangsbuchstaben mindestens zweimal vorkommen:

Die ersten zwei Buchstaben des Variablen-Typ verwenden.

Beispiele:

```
DATE → daDatum
DOUBLE → doZah
SINGLE → siZahl
STRING → stText
```

Deklaration s. Kapitel Deklaration

Starten von Makros

Allgemeines

Das Starten eines Makros erfolgt manuell über **Extras → Makros → Makros ausführen...**

Allerdings sollte es Ziel sein, dass das Starten eines Makros intuitiv und kontextabhängig erfolgt. Folgende Möglichkeiten gibt es dafür:

- Eigene Schaltfläche (Icon) in einer Symbolleiste
- Eigene Symbolleiste mit Schaltflächen (Icons)
- Eigener Menüeintrag
- Schaltflächen innerhalb von Dokumenten
- Automatisierung des Programmstartes

Ganz allgemein sollte auch beachtet werden, dass die Startmöglichkeiten eines Makros nur in dem Umfeld angeboten werden, in dem das Makro sinnvollerweise gestartet werden kann. So sind Textverarbeitungsfunktionen in einem Calc-Umfeld nicht sinnvoll und umgekehrt – dies führt nur zu Verwirrungen und zu unnötigen Fehlermeldungen.

Eigene Schaltfläche (Icon) in einer Symbolleiste

Eigene Schaltflächen (Icons) können z.B. in die Symbolleiste „Standard“ eingebunden werden (siehe Abbildung 1).

Abbildung 1: eingebundene Icons als Startfunktionen

Eigene Symbolleiste mit Schaltflächen (Icons)

Für das Starten von Makros kann auch eine eigene Symbolleiste erstellt werden, die selbst erstellte Schaltflächen (Icons) enthält (siehe Abbildung 2).

Abbildung 2: Eigene Symbolleisten mit (Text-) Startbuttons bzw. mit Icons

Eigener Menüeintrag

Für das Starten von Makros können auch eigene Menüeinträge erstellt werden. Dies kann in Form von zusätzlichen Menüeinträgen (eingebundene Menü-Option) oder als separater Hauptmenü-Punkt (siehe Abbildung 3) erfolgen.

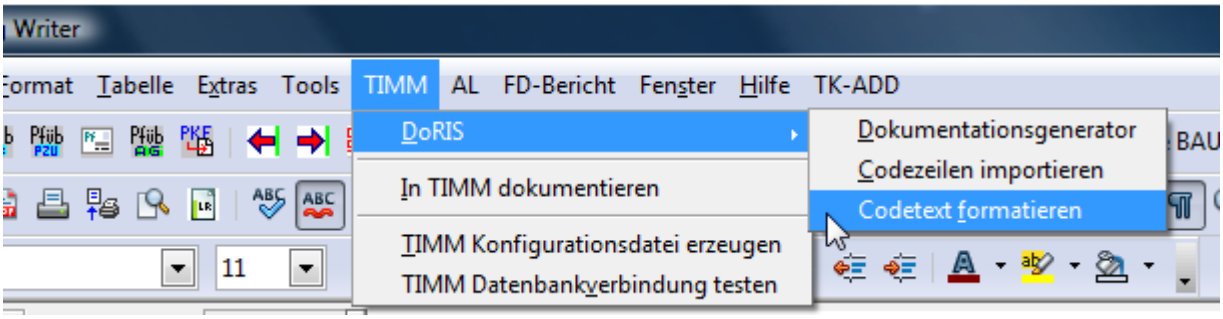

Abbildung 3: Eigener Menü-Eintrag mit Untereinträgen

Automatisierung des Programmstartes

Es ist möglich, den Start eines Makros so zu automatisieren, dass der/die Benutzer/in keine eigene Aktion mehr durchführen muss. So kann z.B. der Start eines Makros an das Ereignis „Öffnen eines Dokuments“ gebunden werden.

Beim automatischen Start eines Makros muss jedoch bedacht werden, dass eine Möglichkeit geschaffen werden muss, den Ablauf des Makros zu unterbrechen bzw. an den Ausgangspunkt wieder zurückkehren zu können.

Fehlerbehandlung

Für die Fehlerbehandlung im Rahmen der Programmierung von Makros gibt es sehr schöne Übersicht in dem "Makro-Kochbuch" (Kapitel 2.5) von Thomas Krumbein:

www.wollmux.net/wiki/images/f/f9/Makro_Kochbuch.pdf

www.wollmux.net/wiki/Datei:Makro_Kochbuch.odt

Verwendung von Makros in Unternehmen oder öffentlichen Einrichtungen

Stadt München

Die Landeshauptstadt München hat sich 2003 dazu entschlossen, auf ihren ca. 14.000 Arbeitsplatzrechnern zukünftig freie Software einzusetzen. Ein Teilprojekt im Rahmen der Migration auf die freie Software war die Umsetzung der bisher verwendeten Makros. Dieses Teilprojekt wird ausführlich in dem "Makro-Kochbuch" von Thomas Krumbein beschrieben:

www.wollmux.net/wiki/images/f/f9/Makro_Kochbuch.pdf

www.wollmux.net/wiki/Datei:Makro_Kochbuch.odt

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 direkter 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

Verzeichnisse aktualisieren

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. Es ist für die LibreOffice-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
```

Wird dieses Makro dem Befehl **Dokument speichern** zugeordnet, werden die Verzeichnisse beim Speichern automatisch aktualisiert.

Verwenden Sie hierzu **Extras** → **Anpassen...**, um den Dialog *Anpassen* zu öffnen, und wählen Sie die Registerkarte **Ereignisse** (Abbildung 4). Die Ereignisse im Dialog Anpassen beziehen sich auf die gesamte Anwendung und bestimmte Dokumente. Verwenden Sie das Feld *Speichern in*, um LibreOffice zu wählen. Damit wird das Makro bei jedem Speichern eines Writerdokuments durchgeführt.

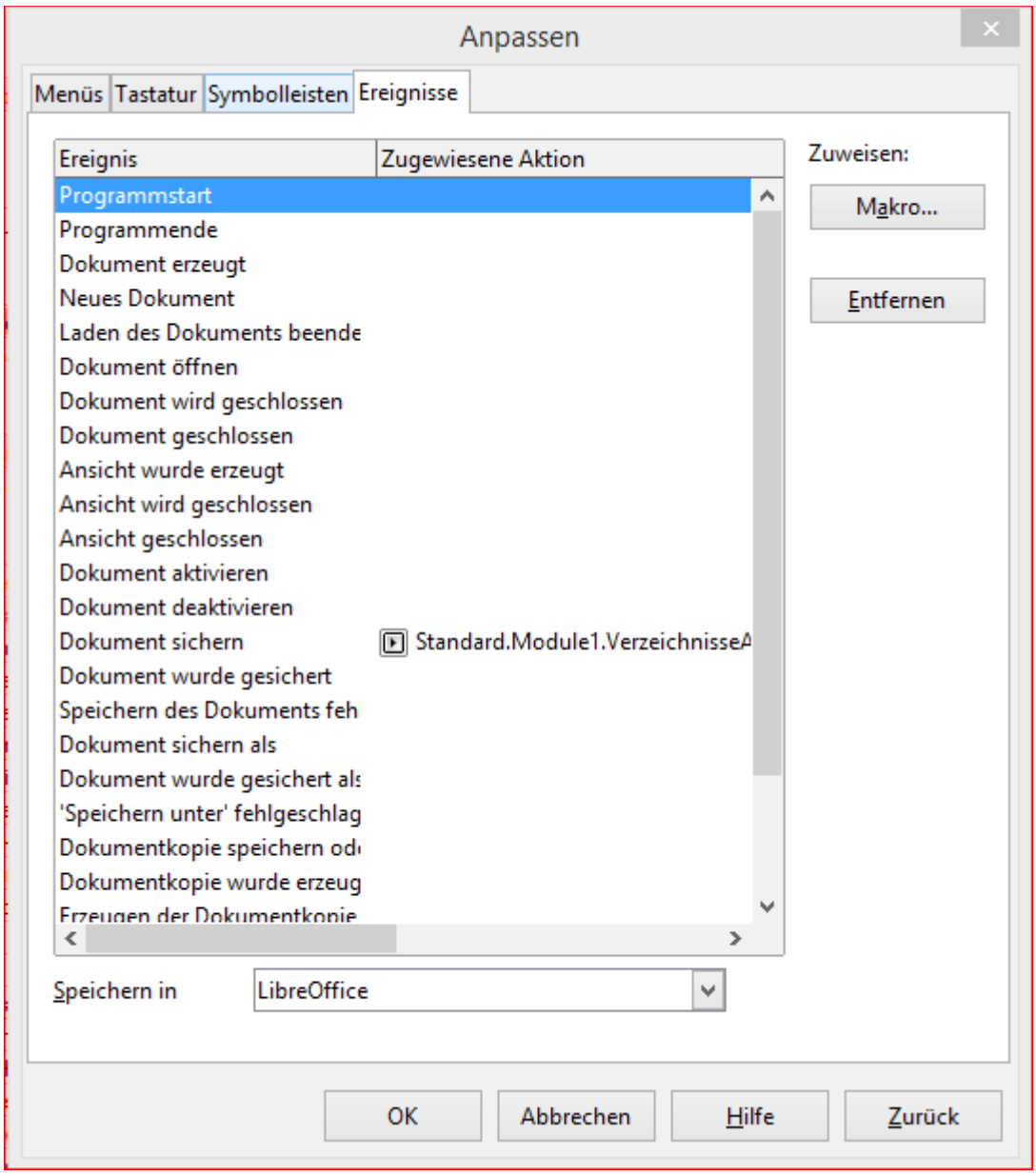

Abbildung 4: Anpassen – Ereignisse I

Wählen Sie das gewünschte Ereignis (*Dokument sichern*) und klicken Sie auf die Makrotaste, um den Makro-Auswahldialog zu öffnen (Abbildung 5).

Wählen Sie das gewünschte Makro und klicken Sie auf **OK**, um das Makro dem Ereignis zuzuordnen. Die Registerkarte Ereignisse zeigt, dass ein Makro einem Ereignis zugeordnet wurde (Abbildung 4).

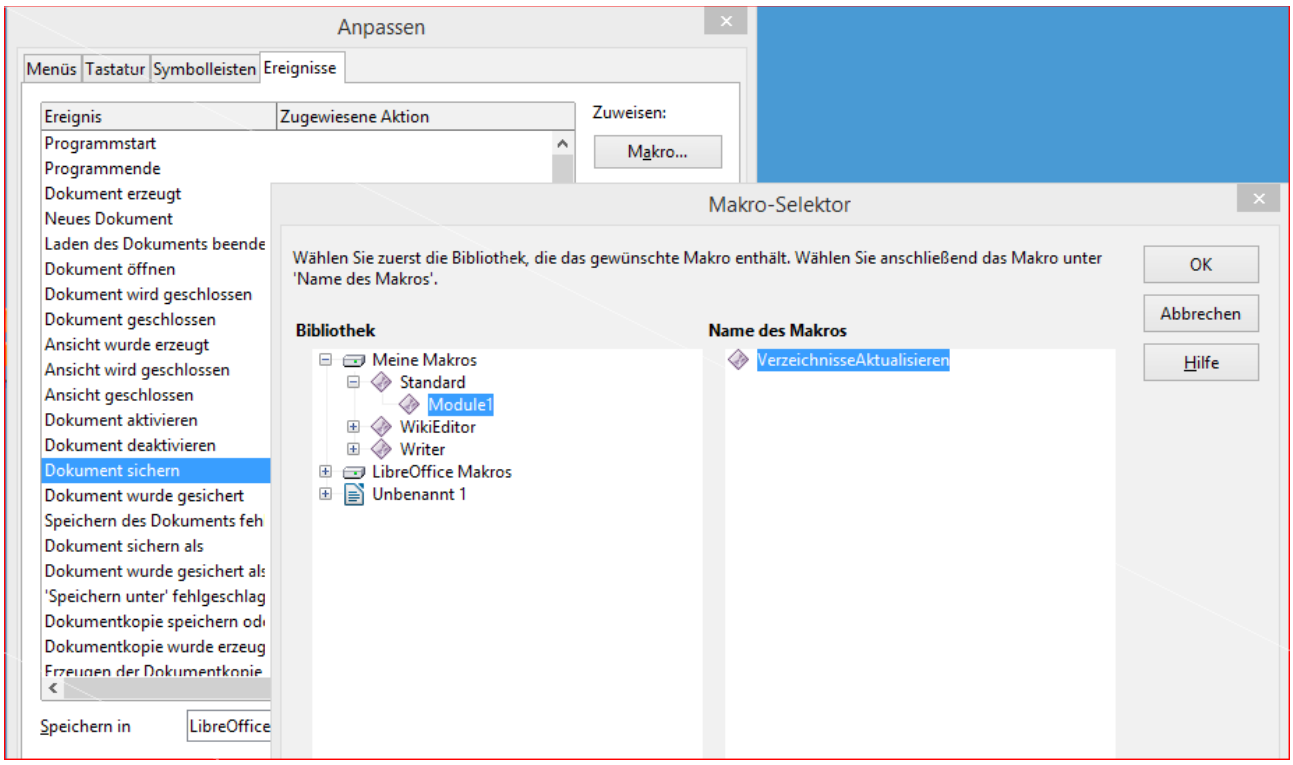

Abbildung 5: Makro auswählen

Makros für Writer

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

Beispiel für ein 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
```

Datenabfrage und numerische Berechnungen

Beschreibung:

Dieses Makro generiert Daten mittels Eingaben aus einem Dialog in Felder eines Textdokuments.

Teilweise werden numerische Berechnungen durchgeführt.

Code:

```
    Dim oDialog as Object
    Dim heute as String


    Sub DialogAusfuehren
       Dim oListBox as Object, aListe()
       DialogLibraries.LoadLibrary("Standard")
       oDialog = createUnoDialog(DialogLibraries.Standard.dialog)
       oListBox = oDialog.getControl("ListBox1")
       aListe = array("Frau","Herr")
       oListBox.removeItems(0,oListBox.getItemCount())
       oListBox.addItems(aListe,0)
       oListBox = oDialog.getControl("ListBox2")
       aListe = array("1","2","3")
       oListBox.removeItems(0,oListBox.getItemCount())
       oListBox.addItems(aListe,0)
       oListBox = oDialog.getControl("ListBox3")
       oListBox.removeItems(0,oListBox.getItemCount())
       oListBox.addItems(aListe,0)
```

```
    oListBox = oDialog.getControl("ListBox4")
    oListBox.removeItems(0,oListBox.getItemCount())
    oListBox.addItems(aListe,0)
    oListBox = oDialog.getControl("ListBox5")
    aListe = array(" ","Dr.","Prof.")
    oListBox.removeItems(0,oListBox.getItemCount())
    oListBox.addItems(aListe,0)
    oDialog.execute()
End Sub

Sub endeDialog
    Dim oDoc as Object
    Dim oFeldName as Object
    Dim titel as String
    Dim summe as Single
    Dim re_nr as String
    Dim msk as String
    Dim kListe(4) as Single
    Dim mkost as Single
    Dim mk as String
    Dim ms as String
    Dim rdat as String
    Dim rbet as String
    Dim rb as String
    Dim faeli as Date
    Dim fg as Date
    Dim heute as Date
    Dim tage as Integer
    Dim bet as Single
    Dim betr as String
    Dim sum as String
    Dim fneu as Date

    oDoc = ThisComponent
```

```
rem Adressfeld und Anrede
aPosList = oDialog.getControl("ListBox1").getSelectedItemsPos()
SetVariable(oDoc,"ku_anrede",aPosList(0))
titel = oDialog.getControl("ListBox5").getSelectedItem()
If titel = "" then
  titel = " "
End If
SetVariable(oDoc,"ku_titel",titel)
vname = oDialog.getControl("TextField1").text
SetVariable(oDoc,"ku_vorname",vname)
name = oDialog.getControl("TextField2").text
SetVariable(oDoc,"ku_name",name)
adzus = oDialog.getControl("TextField19").text
SetVariable(oDoc,"ku_zus",adzus)
strasse = oDialog.getControl("TextField3").text
SetVariable(oDoc,"ku_str",strasse)
hnr = oDialog.getControl("TextField4").text
SetVariable(oDoc,"ku_hnr",hnr)
plz = oDialog.getControl("TextField5").text
SetVariable(oDoc,"ku_plz",plz)
ort = oDialog.getControl("TextField6").text
SetVariable(oDoc,"ku_ort",ort)

rem Gesamtwerte
summe = 0


rem 1. Rechnung
rem Rechnungsnummer
renr = oDialog.getControl("TextField7").text
If renr <> "" then
  SetVariable(oDoc,"re_nr1",renr)

  rem Mahnstufe, Mahnkosten
```

```
  aItemList = oDialog.getControl("ListBox2").getSelectedItems()
  msk = aItemList(0) & " / "
  kListe = array(0.00,5.00,10.00,15.00)
  mkost = kListe(CInt(aItemList(0)))
  mk =  Format(mkost,"############0.00")
  msk = CStr(aItemList(0)) & " / " & mk & " €"
  SetVariable(oDoc,"re_msk1",msk)

  rem Rechnungsdatum, Rechnungsbetrag, Fälligkeit
  rdat = oDialog.getControl("TextField8").text
  SetVariable(oDoc,"re_dat1",rdat)
  rbet = oDialog.getControl("TextField9").text
  rb = Format(rbet,"############0.00")
  SetVariable(oDoc,"re_bet1",rb & " €")
  fael = oDialog.getControl("TextField10").text
  SetVariable(oDoc,"re_f1",fael)

  rem Berechnung der Verzugstage
  faeli = CDate(fael)
  fg = faeli
  heute = CDate(Date())
  tage = Fix(heute - faeli)
  SetVariable(oDoc,"re_verz1",tage)

  rem Gesamtbetrag
  bet = CSng(rbet) + (tage * 0.1) + CSng(mkost)
  betr = Format(bet,"############0.00")
  SetVariable(oDoc,"re_ges1",betr & " €")
  summe = summe + bet
End If

rem 2. Rechnung
rem Rechnungsnummer
renr = oDialog.getControl("TextField11").text
If renr <> "" then
```

```
  SetVariable(oDoc,"re_nr2",renr)

  rem Mahnstufe, Mahnkosten
  aItemList = oDialog.getControl("ListBox3").getSelectedItems()

  msk = aItemList(0) & " / "
  kListe = array(0.00,5.00,10.00,15.00)
  mkost = kListe(CInt(aItemList(0)))
  mk =  Format(mkost,"############0.00")
  msk = CStr(aItemList(0)) & " / " & mk & " €"
  SetVariable(oDoc,"re_msk2",msk)
  rem Rechnungsdatum, Rechnungsbetrag, Fälligkeit
  rdat = oDialog.getControl("TextField12").text
  SetVariable(oDoc,"re_dat2",rdat)
  rbet = oDialog.getControl("TextField13").text
  rb = Format(rbet,"############0.00")
  SetVariable(oDoc,"re_bet2",rb & " €")
  fael = oDialog.getControl("TextField14").text
  SetVariable(oDoc,"re_f2",fael)

  rem Berechnung der Verzugstage
  faeli = CDate(fael)
  If faeli < fg or fg = 0 then
        fg = faeli
  End If
  heute = CDate(Date())
  tage = Fix(heute - faeli)
  SetVariable(oDoc,"re_verz2",tage)

  rem Gesamtbetrag
  bet = CSng(rbet) + (tage * 0.1) + CSng(mkost)
  betr = Format(bet,"############0.00")
  SetVariable(oDoc,"re_ges2",betr & " €")
  summe = summe + bet
End If
```

```
rem 2. Rechnung
rem Rechnungsnummer
renr = oDialog.getControl("TextField15").text
If renr <> "" then
  SetVariable(oDoc,"re_nr3",renr)

  rem Mahnstufe, Mahnkosten
  aItemList = oDialog.getControl("ListBox4").getSelectedItems()
  msk = aItemList(0) & " / "
  kListe = array(0.00,5.00,10.00,15.00)
  mkost = kListe(CInt(aItemList(0)))
  mk =  Format(mkost,"############0.00")
  msk = CStr(aItemList(0)) & " / " & mk & " €"
  SetVariable(oDoc,"re_msk3",msk)

  rem Rechnungsdatum, Rechnungsbetrag, Fälligkeit
  rdat = oDialog.getControl("TextField16").text
  SetVariable(oDoc,"re_dat3",rdat)
  rbet = oDialog.getControl("TextField17").text
  rb = Format(rbet,"############0.00")
  SetVariable(oDoc,"re_bet3",rb & " €")
  fael = oDialog.getControl("TextField18").text
  SetVariable(oDoc,"re_f3",fael)

  rem Berechnung der Verzugstage
  faeli = CDate(fael)
  If faeli < fg or fg = 0 then
        fg = faeli
  End If
  heute = CDate(Date())
  tage = Fix(heute - faeli)
  SetVariable(oDoc,"re_verz3",tage)

  rem Gesamtbetrag
```

```
      bet = CSng(rbet) + (tage * 0.1) + CSng(mkost)
      betr = Format(bet,"#############0.00")
      SetVariable(oDoc,"re_ges3",betr & " €")
      summe = summe + bet
    End If


    If summe > 0 then
      sum = Format(summe,"#############0.00")
      SetVariable(oDoc,"summe",sum)
      SetVariable(oDoc,"faellig_ges",fg)
      rem neue Fälligkeit
      fneu = DateAdd("d", 14, heute)
      SetVariable(oDoc,"faellig_neu",CStr(fneu))
    End If
    oDoc.TextFields.refresh()

    odialog.endExecute()
End Sub

Function SetVariable(oDocument as object,Variable as string,Inhalt as string) as string
  Dim Var as String
  Dim oTextfieldMaster As Object
  Dim oPropSet as Object
  Dim oDependentTextFields as Object
  Dim oXDependentTextField as Object
  Dim oTextFields as Object
  On Error Resume Next
  Var ="com.sun.star.text.FieldMaster.SetExpression."+Variable
  oTextfieldMasters = oDocument.getTextFieldMasters()
  oPropSet = oTextfieldMasters.getByName(Var)
  oDependentTextFields = oPropSet.DependentTextFields
  oXDependentTextField = oDependentTextFields(0)
  oXDependentTextField.Content=inhalt
  On Error Goto 0
```

```
End Function

Function GetVariable(oDocument as object,Variable) as string
  Dim Var as String
  Dim oTextfieldMaster As Object
  Dim oPropSet as Object
  Dim oDependentTextFields as Object
  Dim oXDependentTextField as Object
  Dim oTextFields as Object
  On Error Resume Next
  Var ="com.sun.star.text.FieldMaster.SetExpression."+Variable
  oTextfieldMasters = oDocument.getTextFieldMasters()
  oPropSet = oTextfieldMasters.getByName(Var)
  oDependentTextFields = oPropSet.DependentTextFields
  oXDependentTextField = oDependentTextFields(0)
  GetVariable= oXDependentTextField.Content
  On Error Goto 0
End Function

Function TestLB2
    Dim txt as String
    txt = oDialog.getControl("ListBox2").getSelectedItem()
    If txt = "" then
      MsgBox ("Bitte wählen Sie eine Mahnstufe!",0,"FEHLER")
      oDialog.getControl("ListBox2").setFocus()
    End If
End Function

Function TestField7
    Dim txt as String
    txt = oDialog.getControl("TextField7").text
    If txt = "" then
      MsgBox ("Bitte tragen Sie die Rechnungsnummer ein!",0,"FEHLER")
      oDialog.getControl("TextField7").setFocus()
    End if
```

```
End Function

Function TestField8
   Dim rdat as String
   Dim fehler as Boolean
   Dim rdat1 as Date
   Dim heute as Date
   rdat = oDialog.getControl("TextField8").text
   If rdat <> "" then
     If isDate(rdat) then
            heute = CDate(Date())
            rdat1 = CDate(rdat)
            If rdat1 > heute then
                  MsgBox ("Das Datum liegt in der Zukunft!",0,"FEHLER")
                  oDialog.getControl("TextField8").setFocus()
            End If
     Else
            MsgBox (rdat &" ist kein gültiges Datum!",0,"FEHLER")
            oDialog.getControl("TextField8").setFocus()
     End If
   Else
     MsgBox ("Bitte tragen Sie das Rechnungsdatum ein!",0,"FEHLER")
     oDialog.getControl("TextField8").setFocus()
   End If
End Function

Function TestField9
   Dim txt as String
   txt = oDialog.getControl("TextField9").text
   If txt = "" then
     MsgBox ("Bitte tragen Sie den Rechnungsbetrag ein!",0,"FEHLER")
     oDialog.getControl("TextField9").setFocus()
   End if
End Function
```

```
Function TestDateDiff10
   Dim rdat as String
   Dim fdat as String
   Dim fehler as Boolean
   Dim rdat1 as Date
   Dim fdat1 as Date
   rdat = oDialog.getControl("TextField8").text
   rdat1 = CDate(rdat)
   fdat = oDialog.getControl("TextField10").text
   If fdat <> "" then
     If isDate(fdat) then
            fdat1 = CDate(fdat)
            if rdat1 > fdat1 then
                  MsgBox ("Das Rechnungsdatum (" & rdat & ") ist größer als das
   Datum der Fälligkeit ("& fdat &")!" ,0,"FEHLER")
                  oDialog.getControl("TextField10").setFocus()
            End If
     Else
            MsgBox (fdat &" ist kein gültiges Datum!",0,"FEHLER")
            oDialog.getControl("TextField10").setFocus()
     End If
   Else
     MsgBox ("Bitte tragen Sie das Fälligkeitsdatum der Rechnung ein!",0,"FEHLER")
     oDialog.getControl("TextField10").setFocus()
   End if
End Function

Function TestLB3
   Dim txt as String
   txt = oDialog.getControl("ListBox3").getSelectedItem()
   If txt = "" then
     MsgBox ("Bitte wählen Sie eine Mahnstufe!",0,"FEHLER")
     oDialog.getControl("ListBox3").setFocus()
   End If
End Function
```

```
Function TestField11
   Dim txt as String
   txt = oDialog.getControl("TextField11").text
   If txt = "" then
     MsgBox ("Bitte tragen Sie die Rechnungsnummer ein!",0,"FEHLER")
     oDialog.getControl("TextField11").setFocus()
   End if
End Function

Function TestField12
   Dim rdat as String
   Dim fehler as Boolean
   Dim rdat1 as Date
   Dim heute as Date
   rdat = oDialog.getControl("TextField12").text
   If rdat <> "" then
     If isDate(rdat) then
          heute = CDate(Date())
          rdat1 = CDate(rdat)
          If rdat1 > heute then
               MsgBox ("Das Datum liegt in der Zukunft!",0,"FEHLER")
               oDialog.getControl("TextField12").setFocus()
          End If
     Else
          MsgBox (rdat &" ist kein gültiges Datum!",0,"FEHLER")
          oDialog.getControl("TextField12").setFocus()
     End If
   Else
     MsgBox ("Bitte tragen Sie das Rechnungsdatum ein!",0,"FEHLER")
     oDialog.getControl("TextField12").setFocus()
   End If
End Function

Function TestField13
```

```
    Dim txt as String
    txt = oDialog.getControl("TextField13").text
    If txt = "" then
      MsgBox ("Bitte tragen Sie den Rechnungsbetrag ein!",0,"FEHLER")
      oDialog.getControl("TextField13").setFocus()
    End if
End Function

Function TestDateDiff14
    Dim rdat as String
    Dim fdat as String
    Dim fehler as Boolean
    Dim rdat1 as Date
    Dim fdat1 as Date
    rdat = oDialog.getControl("TextField12").text
    rdat1 = CDate(rdat)
    fdat = oDialog.getControl("TextField14").text
    If fdat <> "" then
      If isDate(fdat) then
            fdat1 = CDate(fdat)
            if rdat1 > fdat1 then
                  MsgBox ("Das Rechnungsdatum (" & rdat & ") ist größer als das
    Datum der Fälligkeit ("& fdat &")!" ,0,"FEHLER")
                  oDialog.getControl("TextField14").setFocus()
            End If
      Else
            MsgBox (fdat &" ist kein gültiges Datum!",0,"FEHLER")
            oDialog.getControl("TextField14").setFocus()
      End If
    Else
      MsgBox ("Bitte tragen Sie das Fälligkeitsdatum der Rechnung ein!",0,"FEHLER")
      oDialog.getControl("TextField14").setFocus()
    End if
End Function
```

```
Function TestLB4
    Dim txt as String
    txt = oDialog.getControl("ListBox3").getSelectedItem()
    If txt = "" then
      MsgBox ("Bitte wählen Sie eine Mahnstufe!",0,"FEHLER")
      oDialog.getControl("ListBox3").setFocus()
    End If
End Function

Function TestField15
    Dim txt as String
    txt = oDialog.getControl("TextField15").text
    If txt = "" then
      MsgBox ("Bitte tragen Sie die Rechnungsnummer ein!",0,"FEHLER")
      oDialog.getControl("TextField15").setFocus()
    End if
End Function

Function TestField16
    Dim rdat as String
    Dim fehler as Boolean
    Dim rdat1 as Date
    Dim heute as Date
    rdat = oDialog.getControl("TextField16").text
    If rdat <> "" then
      If isDate(rdat) then
            heute = CDate(Date())
            rdat1 = CDate(rdat)
            If rdat1 > heute then
                  MsgBox ("Das Datum liegt in der Zukunft!",0,"FEHLER")
                  oDialog.getControl("TextField16").setFocus()
            End If
      Else
            MsgBox (rdat &" ist kein gültiges Datum!",0,"FEHLER")
            oDialog.getControl("TextField16").setFocus()
```

```
    End If
  Else
    MsgBox ("Bitte tragen Sie das Rechnungsdatum ein!",0,"FEHLER")
    oDialog.getControl("TextField16").setFocus()
  End If
End Function

Function TestField17
  Dim txt as String
  txt = oDialog.getControl("TextField17").text
  If txt = "" then
    MsgBox ("Bitte tragen Sie den Rechnungsbetrag ein!",0,"FEHLER")
    oDialog.getControl("TextField17").setFocus()
  End if
End Function

Function TestDateDiff18
  Dim rdat as String
  Dim fdat as String
  Dim fehler as Boolean
  Dim rdat1 as Date
  Dim fdat1 as Date
  rdat = oDialog.getControl("TextField16").text
  rdat1 = CDate(rdat)
  fdat = oDialog.getControl("TextField18").text
  If fdat <> "" then
    If isDate(fdat) then
        fdat1 = CDate(fdat)
        if rdat1 > fdat1 then
            MsgBox ("Das Rechnungsdatum (" & rdat & ") ist größer als das
  Datum der Fälligkeit ("& fdat &")!" ,0,"FEHLER")
            oDialog.getControl("TextField18").setFocus()
        End If
    Else
        MsgBox (fdat &" ist kein gültiges Datum!",0,"FEHLER")
```

```
            oDialog.getControl("TextField18").setFocus()
        End If
    Else
        MsgBox ("Bitte tragen Sie das Fälligkeitsdatum der Rechnung ein!",0,"FEHLER")
        oDialog.getControl("TextField18").setFocus()
    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, die im LibreOffice 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)

OpenOffice.org Makros - „Kochbuch“
www.wollmux.net/wiki/images/f/f9/Makro_Kochbuch.pdf (Stand: 8/17)

www.wollmux.net/wiki/Datei:Makro_Kochbuch.odt

Englisch

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

LibreOffice 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