Inhaltsverzeichnis

StarBasic / OpenOffice.org Basic FAQ

Version 282 - 06.04.2009

Diese FAQ gibt es auch als StarOffice / OpenOffice.org Dokument oder als PDF- Dokument .
Achtung! Diese beiden Versionen entsprechen nicht immer der aktuellen Online- Version.
Zusätzlich kann man auch die HTML-Version herunterladen.

Stichwortliste ------- Suchen ------- Das Starbasic-Forum zur FAQ

1 StarOffice / OpenOffice - Allgemein

1.1 Welche Tastaturbefehle gibt es?

Es gibt viele Tastaturbefehle um einfache Formatierungen, Markierungen und etc. zu tätigen. Eine Liste mit allen wichtigen Befehlen und dem Vergleich zu MS-Office steht hier als PDF-Datei -> http://www.dannenhoefer.de/down/tastaturbefehle.pdf

1.2 Wie kann man eine Datei mit dem Programmstart öffnen?

Für Windows:
soffice.exe file:///c:/pfad/datei.sxw

oder

soffice.exe c:/pfad/datei.sxw

1.3 Wie installiert man StarOffice unter Windows XP/2000/NT im Mehr-Benutzer-Betrieb?

Für OpenOffice bis 1.1.5/ StarOffice bis 7.0 gilt:

Die Installation erfolgt für den Mehr-Benutzer-Betrieb auf einem System in zwei Schritten:

1.Die Installation der Netzwerkversion auf dem lokalen Rechner.

2.Die Installation der Workstationversion für jeden Benutzer.

1.

Die Installation der Netzwerkversion auf dem lokalen Rechner.

Starten der setup.exe mit dem Parameter -net.

Danach wird die Installation normal durchgeführt. Man sollte nur bei der Auswahl der Installationsart die "Benutzergesteuerte Installation" auswählen und alle Komponenten aktivieren. Dadurch erspart man sich spätere Nachinstallationen, wenn ein User weitere Funktionen als die der Standard-Installation benötigt.

2.Die Installation der Workstationversion für jeden Benutzer.

Man meldet sich unter dem Benutzer am Rechner an und ruft die Setup.exe aus dem Verzeichnis der Serverversion auf.

Bei der Auswahl der Installationsart wählt man jetzt Workstation Installation aus. Ansonsten führt man das Setup normal aus. Auf diese Weise werden dann nur individuelle Daten für Workstation für den Benutzer installiert (ca 1-4 MB).

Detailliertere Infos gibt es hier:

http://de.openoffice.org/doc/setupguide/installations_handbuch.html

1.4 Wie kann man eine andere Standardschrift für neue Tabellendokumente festlegen?

Man muß eine eigene Vorlage als Standardvorlage festlegen.
Siehe -> Wie kann ich eine eigene Standardvorlage festlegen?

1.5 Wie kann ich eine eigene Standardvorlage festlegen?

Es gibt in Staroffice nach der Installation eigentlich keine extra Standardvorlagen für Writer, Calc, Präsentation und Zeichnung. Man kann aber eine eigene Vorlage als Standard festlegen.

Man erstellt eine Datei mit den gewünschten Einstellungen (Seitenrand, Schrift etc.). Diese speichert man dann unter dem Menüpunkt Datei- > Dokumentenvorlage->Speichern in einem der angebotenen Bereiche, am besten in den Bereich "Standard".

Wenn man dann unter Datei-> Dokumentenverwaltung -> Verwalten das Menü aufruft, kann man im gewählten Bereich seine erstellte Vorlage auswählen. Dies kann dann über den Button "Befehle" als Standardvorlage festgelegt werde. StarOffice erkennt selber für welchen Dokumententyp die

Standardvorlage geändert wird. Hier kann man dies auch wieder rückgängig machen ("Standardvorlage zurücksetzen").

1.6 Wie kann man in Calc Zellen mit Zellen aus anderen Tabellen verknüpfen?

Es gibt zwei Möglichkeiten eine Zelle mit einer Zelle einer anderen Tabelle zu verknüpfen.

1.

Man wechselt nach der Eingabe des Leuchtzeichens (=) in die andere Tabelle, wählt die gewünschte Zelle und bestätigt mit der Return-Taste.

Als Ergebnis in der Zelle kommt dann in etwa so etwas heraus:

='File:///c:/test.xlc'#$Tabelle1.A2

2.

Man wechselt in die andere Tabelle und wählt die Zelle aus und kopiert diese. In der Zelle wird dann in der gewünschten Zelle der Menüpunkt Bearbeiten - Inhalte einfügen anklickt und in der Maske unten links "Verknüpfen" angewählt.

Als Ergebnis in der Zelle kommt dann in etwa so etwas heraus:

=DDE("soffice";"C\test1.sxc";"Tabelle1.A2)

Worin liegt der Unterschied?

Die erste Verknüpfung ist manuell bezogen auf den Speicherort der Tabelle. Beim Aktualisieren wird auf den letzten gespeicherten Wert zugegriffen. Sollte die zweite Tabelle offen sein muß diese erst gespeichert werden, damit der richtige Wert aktualisiert wird. Mit dieser Methode kann auch immer nur eine Zelle mit einer Zelle verknüpft werden.

Die Datei die verknüpft worden ist, wird bei der Aktualisierung nicht in Calc geöffnet.

Die zweite Verknüpfung ist eine DDE-Verknüpfung. (DDE = Dynamic Data Exchange). Mit dieser ist ein dynamischer Datenaustausch möglich. Das heißt, wenn beide Tabellen geöffnet sind wird die Änderung sofort sichtbar. Mit dieser Methode kann man auch mehrere Zellen miteinander verknüpfen. Bei einer DDE-Verknüpfung wird die verknüpfte Datei automatisch in einem zweiten Calc-Fenster geöffnet.

Beide Methode bieten je nach Aufgabe Vor- und Nachteile.

1.7 Kann man sich die Tastaturbefehle, die Menüs und die Symbolleisten anpassen?

Ja!

Unter Extras -> Anpassen findet man den Dialog um sich die Menüleisten, Tastaturbefehle, Statusleiste, Symbolleisten und bestimmte Ereignisse anzupassen.

1.8 Kann man die Aufreißleisten noch anders platzieren?

Man kann die Aufreißleisten auch zusätzlich frei auf dem Bildschirm platzieren. Dazu muß man nur die aufgeklappte Leiste mit der Maus in der oberen Leiste anklicken und an die gewünschte Position verschieben. Danach steht sie als freie schwebende Symbolleiste zur Verfügung.

1.9 Wie kann ich die Anzahl der Einträge in der Dokumenten-Liste im Menü "Datei" erweitern?

In Version 6 von StarOffice bzw. in Version 1.0 von OpenOffice.org muss dazu in der Datei

<Office>/share/config/registry/instance/org/openoffice/Office/Common.xll

der Wert des Eintrags "PickListSize" geändert werden.

Für Version 7 von StarOffice bzw. in Version 1.1 von OpenOffice.org kann diese Liste mit dem Makro-Dokument "RecentFilesListChanger" von Danny Brewer verändern. Es kann hier geladen werden:

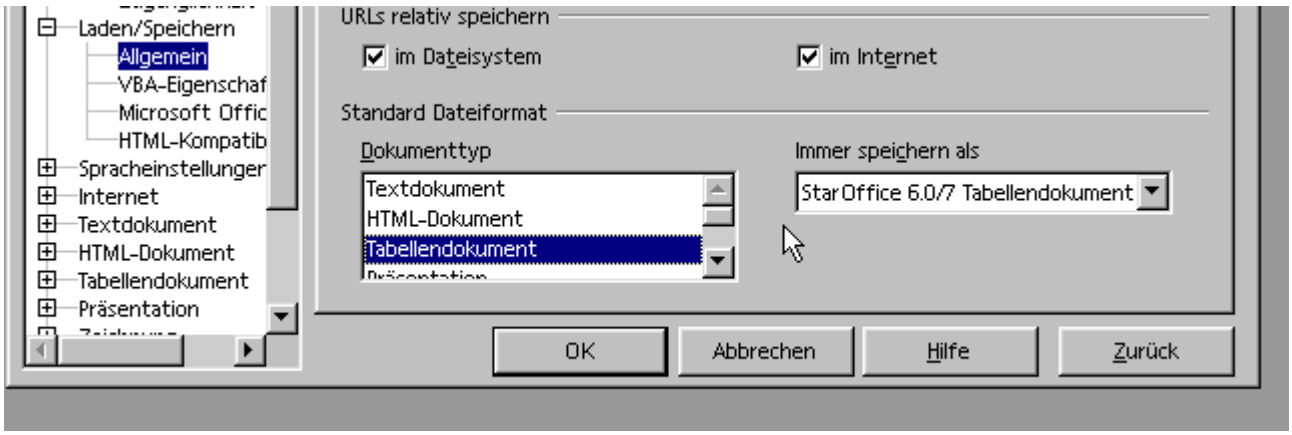
http://kosh.datateamsys.com/~danny/OOo/Examples/ConfigurationMgr/

1.10 Wie kann man Text-Dateien in Calc öffnen?

Bedauerlicherweise öffnet Calc Dateien mit der Endung txt automatisch im Writer.
Um dies zu verhindern muß man die Datei in *.csv umbenennen. Solche Dateien werden automatisch in Calc geladen.

1.11 Wie kann man das Standardformat für das Speichern von Dokumenten festlegen?

Unter Extras >> Optionen >> Laden/Speichern >> Allgemein kann man für jeden Dokumententyp das Standardformat zu Speichern von Dokumenten festlegen.



2 StarBasic - Allgemein

2.1 Informationen

2.2.1 Links zu Starbasic

Links auf einem Blick (die Reihenfolge ist keine Bewertung!):
http://docs.sun.com/app/docs/doc/819-1326?a=load -> Dokumentation von Sun zu StarBasic 8 (deutsch)

http://download.openoffice.org/2.4.0/sdk.html -> Download des SDK 2.4. Entgegen der Seite ist das Developer Guide nicht mehr enthalten. Dieses gibt es zur Zeit nur Online.

http://download.openoffice.org/3.0.0/sdk.html -> Download des SDK 3.0. Entgegen der Seite ist das Developer Guide nicht enthalten. Dieses gibt es zur Zeit nur Online. Der Download steht dann im Ordner Contrib auf den FTP-Servern.

http://documentation.openoffice.org.....macros.sxw -> Eine Einführung (Englisch)

http://kienlein.com/pages/oo.html -> Code zum DeveloperGuide und Datenbankzugriff

http://www.ooowiki.de/-> Wiki zu OpenOffice und StarOffice

http://de.openoffice.info/-> Forum für allen Themen zu OpenOffice (deutsch)

http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/OpenOffice.org_Developers_Guide -> DeveloperGuide

http://www.pitonyak.org/AndrewMacro.odt-> Text mit vielen Beispielen zu Starbasic (englisch)

http://www.pitonyak.org/oo.php -> Startseite von Andrew Pitonyak

http://development.openoffice.org/index.html-> OpenOffice.org-Seite für Entwickler

http://www.oooforum.org/ -> Forum zu OO (englisch)

http://www.ooomacros.org -> Makroseite zu OO

http://sourceforge.net/project/ -> Makrodateien

http://disemia.com/software/openoffice/ -> Makrocollection

Ein kleiner Tipp für die Suche im Netz: Es gibt noch viele Seiten die sich mit Starbasic bis Version 5.2 beschäftigen.
Leider sind diese nicht Informationen nicht immer verwendbar, das sich die Sprache mit der Version 6.0 verändert hat.

2.2.2 Wo finde ich Informationen zu StarBasic und API?

Leider gibt es zur Zeit keine richtig gute Quelle zur Makroprogrammierung mit Starbasic.

Erste Adresse ist natürlich die Online-Hilfe. Dort sind vor allem die Standard Befehle erklärt.

Zweite Adresse ist api.openoffice.org. Dort gibt es das ODK/SDK-Kit (OpenOffice.org.1.0.2_Beta_SDK). In der ist die komplette API dokumentiert. Außerdem gibt es dort einen Developerguide in HTML mit Verlinkung auf die API Dokumentation. Dieser beschreibt aber leider vor allem den Zugriff auf die API über Java und nicht über Starbasic. Aber man kann sich das eine oder andere dort rauslesen. Außerdem sind in diesem SDK auch Beispiele in Starbasic.

Den Developer Guide als PDF-Datei gibt es
http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html.

Es gibt auch einen StarBasic Guide in deutsch von Sun inzwischen für die Version 8.0 : http://docs.sun.com/app/docs/coll/1274.1

Die deutsche Version zur VerSion 7.0 befindet sich hier: http://docs.sun.com/app/docs/doc/817- 3924

Fehler/Ungenauigkeiten im neuen StarbasicGuide:

Bemerkenswerterweise sind alle Fehler der Dokumentation zur Version 7 immer noch in der Version 8.0.

Im gesamten Text wird im Beispielcode statt dem doppelten Anführungszeichen mit dem Einfachen gearbeitet. In StarBasic müssen aber doppelte Anführungszeichen verwendet werden.

Version 8 dt. S. 59

Die Funktion Instr wird bezüglich des Parameters für die Gross- und Kleinschreibung falsch beschrieben. Um den Vergleich inkl. Groß- und Kleinbuchstabe zu verwenden muß auch der
erste Parameter für den Start der Suche auch verwendet werden.

ResultString = InStr(1,MyString, SearchString, 0)

Version 8.0 dt.S.36

Select Case über die Eingrenzung von zwei Werten:

Case Var>8 and Var<11

Geht bis jetzt noch nicht.

Version 8.0 dt. S.65

Die Funktion CurDir steht entgegen der Dokumentation nicht zur Verfügung.

Version 8.0 dt.S.95

Im Beispiel Text steht Doc.storeasUrl(sUrl, mFileproperties()), es muss aber (url,mFileproperties()) heißen.

Version 8.0 dt.S.89

Der Parameter "_hidden" wird bei loadcomponentfromurl nicht unterstützt. Wenn man das erreichen will muß man die Propertie des Mediadescriptors Hidden auf true setzen.

```
Dim Doc As Object
Dim Url As String
Dim Dummy(0) as New com.sun.star.beans.PropertyValue
Dummy(0).name="Hidden"
Dummy(0).value=True
Url = "file:///C:/test.sxw"
Doc = StarDesktop.loadComponentFromURL(Url, "_blank", 0, Dummy())
```

Version 8.0 dt.S.112

In dem Beispiel-Makro muß die zweite Zeile:

Cursor=Doc.Text.createTextCursor()

lauten

Version 8.0 dt.S.123

In dem Code müssen die ersten Zeilen getrennt werden.

```
Dim Doc As Object
Dim Table As Object
Dim Cursor As Object
Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
```

Version 8.0 dt.S.168

Der Code zum zweiten Beispiel ist komplett falsch. Er beschreibt das Einfügen von Tabellenblättern in ein Calc-Dokument.

Man kann in Draw Seiten nicht über einen Namen einfügen. Nur mit insertNewByIndex().

Version 8.0 dt.S.243

Model.EchoChar (String) - echo character for password fields

In der IDE können Buchstaben eingeben werden. In Starbasic muss der Charwert des Zeichen übergeben werden. Zum Beispiel 42 für das Sternchen.

Die Eigenschaft ist entgegen der Dokumentation eine Integer-Variable und keine String-Variable.

Version 8.0 dt.S.245

Model.SelectedItems (Array of Strings) - list of highlighted entries

Die Funktion gibt die Position in der Liste zurück nicht den Inhalt des Eintrags

Was die Funktion jetzt macht? Keine Ahnung.

Version 8.0 dt.S.250

In dem Beispiel-Makro muß die Zeile:

DocCrl = Doc.getCurrentControler()

DocCrl = Doc.getCurrent**Controller**() lauten.

2.2 Basic

2.3.1 Was sind Prozeduren und Funktionen?

Prozeduren und Funktionen sind grundlegende Bestandteile eines Programmes. Mit Hilfe von Prozeduren und Funktionen werden bestimmte Aufgaben in sich abgeschlossen Unterprogrammen gekapselt. Mit der Hilfe von Prozeduren und Funktionen kann man sich einen eigenen Befehlsvorrat für ständig wiederkehrende Aufgaben anlegen. Bei Prozeduren dienen zum Ausführen weiterer Aufgaben, aber man erhält kein direktes Ergebnis zurück.Bei Funktion wird zum Schluß ein Ergebnis zurückgegeben. An beide kann man Parameter übergeben.

Aufbau von Prozeduren

Prozeduren werden mit "Sub Prozedurname" eingeleitet und mit "End Sub" beendet. Dazwischen kommt der Programmcode.

Hier ein Beispiel:

```
Sub TestProzedur
    msgbox "Heute ist der " & Date
End Sub
```

Diese Prozedur zeigt in einer MessageBox das aktuelle Datum an. Das ist natürlich nicht der Idealfall für eine Prozedur, da hier eigentlich außer einer Programmzeile (msgbox "Heute ist der " & Date) drei Zeilen werden. Prozeduren sind dann sinnvoll wenn Aufgaben im Programm mehrfach aufgerufen werden oder wenn man durch Prozeduren das Programm in übersichtliche Teilaufgaben zerlegen kann. Das Hauptprogramm ruft dann nur nach einander die Prozeduren auf oder wiederholt bei Bedarf wieder den Aufruf.

Sie wollen zum Beispiel eine Datei speichern und müssen dazu erst den Dateinamen und dann der Speicherort abfragen. Das ist ja eindeutig eine Aufgabe die öfter vorkommen soll. Also erstes wird nach dem Dateinamen gefragt, danach nach dem Speicherort und zum Schluß wird die Datei gespeichert.

Hier der Code als Aufgabenbeschreibung:

```
Sub Speichern
    Abfrage des Dateiname
    Abfrage des Speicherortes
    Zusammensetzen des Dateinamens und Speicherortes
    Speichern der Datei
    Bestätigung des Speicherns
End Sub
```

Als StarBasiccode

```
Sub Speichern
  Dateiname = InputBox ("Geben Sie bitte den Dateinamen ein:")
  Speicherort = InputBox ("Geben Sie bitte ein Verzeichnis ein:")
  mydoc = ThisComponent
```

```
  myUrl=  Speicherort + Dateiname
  mydoc.storeasurl(myurl,args())
  msgbox "Ihre Datei wurde gespeichert"
End Sub
```

So jetzt haben wir eine Prozedur von sechs Zeilen die man jederzeit mit einer Zeile ("Speichern") im Programm aufrufen kann. Bitte jetzt nicht nach den Details zu Speichern fragen, dazu komme ich weiter hinten im Buch. Und selbstverständlich gibt es für Speichern ja den Menüpunkt "Speichern", der eigentlich auch eine Prozedur ist, aber ich wollte ein Beispiel das man sofort nachvollziehen kann.

Diese Prozedur wollen wir jetzt noch ein wenig ergänzen. Zum Beispiel liegt der Speicherort schon fest. Das heißt diesen müssen sie nicht abfragen. Jetzt haben sie zwei Möglichkeiten. Sie können den Speicherort in der Prozedur festlegen. Statt der Abfrage " Speicherort = InputBox ("Geben Sie bitte den Dateinamen ein:")" kommt dann zum Beispiel "Speicherort = "C:\". Dann müssen sie aber für jeden neuen Speicherort eine neue Prozedur schreiben (SpeichernC, SpeichernD etc). Die zweite Möglichkeit ist, das sie den Speicherort der Prozedur als Paramter übergeben. Sie können an Prozeduren Parameter nach folgendem Syntax übergeben: Sub Prozedurname (Paramater1 as Typ, Parameter2 as Typ,....). In unserem Beispiel übergeben wir den Speicherort als String:

```
Sub Speichern ( Speicherort as String)
  Dateiname = InputBox ("Geben Sie bitte den Dateinamen ein:")
  mydoc = ThisComponent
  myUrl=  Speicherort + Dateiname
  mydoc.storeasurl(myurl,args())
  msgbox "Ihre Datei wurde gespeichert"
End Sub
```

Jetzt können Sie diese Prozedure beliebig oft mit verschiedenen Speicherorten aufrufen. Z.B. speichern ("C:\"), speichern("C:\MeineDaten"), etc.

Beim Aufrufen von Prozeduren mit Paramtern müssen Sie nur darauf achten das sie die gleiche Anzahl an Parametern und die gleichen Typen übergeben, wie Sie in der Deklaration angeben haben. In unserem Beispiel kann ich zum Beispiel als Speicherort keine Zahl übergeben (Speichern(12)), da eine Zahl kein String ist.

Was sind nun Funktionen? Eigentlich auch Prozeduren, nur das diese ein Ergebnis zurück liefern. Eine Procedure führt etwas aus. Eine Funktion führt etwas aus und erzeugt ein Ergebnis. Um dieses Ergebnis richtig zuordnen zu können muss die Art des Ergebnisses vorher festgelegt werden. Daher hat eine Funtion folgenden Aufbau:

```
Function Name ( Parameter1, Paramater2,...) as Typ
  Name=Ergebnis
End Function
```

Sie sieht also der Prozedur sehr ähnlich, nur das statt sub function steht und das der Typ noch festgelegt wird. Und das innerhalb der Funktion das Ergebnis übergeben werden muß.

Machen wir hierzu auch ein Beispiel.

Nehmen wir an wir brauchen für unsere Programm eine bestimmte Berechnung mehrmals. Zum Beispiel die Mehrwertsteuer eines Bruttobetrages abhängig vom Mehrwertsteuersatz.

(Bruttobetrag 116,- Eur Nettobetrag 100,- Eur, Mehrwersteuer 16,- Eur, 16% Mwst).

Ein Weg zum Ergebnis:

Bruttobetrag - Nettobetrag = Mehrwertsteuer

Nettobetrag = Bruttobetrag / (1+ 1/100 Zinsatz)

Bei der Funhtion müssen wir als Parameter den Bruttobetrag und den Zinsatz übergeben, als Ergebnis erhalten wie eine Zahl mit Kommastellen.

```
Function Mehrwertsteuer ( Brutto as Double, Zinssatz as double) as double
Dim Zinsfaktor as double
Zinsfaktor = 1 +  Zinssatz / 100
Netto = Brutto / Zinsfaktor
Mehrwertsteuer = Brutto - Netto
End Function
```

Diese Funktion wird dann im Programnm mit "variable = mehrwertsteuer (Bruttobetrag, Zinssatz)" aufgerufen. Zum Beispiel:

```
steuer = mehrwertsteuer( 1234, 16)
```

Dann bekommt steuer den Wert 170,206.....

Jetzt können Sie jeder Zeit mit einer Programmzeile die Mehrwertsteuer eines Betrages ausrechnen lassen.

Funktion können zum Beispiel für mathematische Aufgaben, Textverarbeitungen und für Wahrheitsprüfungen verwendet werden. Bei eine Wahrheits- Boolschen Funktion wird der Zustand Wahr oder Falsch (true oder false) zurückgeben.

Jetzt kommen noch zusätzliche Eigenschaften beim Aufruf von Prozeduren und Funktionen hinzu. Diese betreffen die Parameter.

a. Optional

Parameter mit der Beigabe "Optional" müssen beim Aufruf nicht übergeben werden. Diese werden, wenn sie nicht gefüllt sind automatisch mit einem Leerwert versehen. Ob der Parameter gefüllt worden ist kann man mit der Funtkion "ismissing" testen. Dieses verhalten gilt sowohl für Prozeduren, wie auch Funktionen.

```
if IsMissing(mytext2) then
    myText2 = "Keine Text"
end if
```

Ist der Parameter jetzt nicht gefüllt wird der Text "Kein Text" eingetragen. Umgekehrt kann man mit "not IsMissing()" auch die positive Fragen stellen.

b. ByVal

Paramater die mit dem Vorsatz "ByVal" übergeben werden, können keinen neuen Wert annehmen. Was bedeutet das?

Normalerweise werden die Parameter an Funktionen als Referenz übergeben. Man könnte es als ausleihen beschreiben. Blöderweise kann aber die Prozedur oder Funktion den geliehenen Parameter auch ändern. Auf diese Weise kann man zum Beispiel auch eine Prozedur als Funktion verwenden.

Diese Prozedur

```
Sub NeuerWert( Zahl as Integer)
  Zahl = Zahl*100
End Sub
```

führt genauso zu einem neuen Wert von "Zahl" wie diese Funktion

```
Function funcNeuerWert(Zahl as Integer)
funcNeuerWert=Zahl*100
End Function
```

Wenn man diese beiden "Funktionen" ein Programm einbindet:

```
Sub TestNeuerWert
   MyWert=10
   NeuerWert(MyWert)
```

```
  ErsteZahl=MyWertl
  MyWert=fNeuerWert(myWert)
  ZweiteZahl=MyZahl
  MsgBox "Erste Zahl: " +ErsteZahl+" Zweite Zahl: "+ZweiteZahl
end sub
```

kommt die vielleicht gar nicht gewollte Vervielfachung des Ausgangswertes heraus.

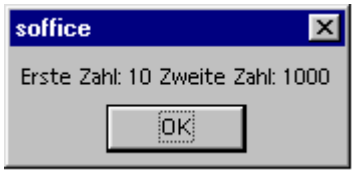
Dies kann man durch zwei Wege verhindern. Der erste ist, wie es sich gehört für einen guten Programmierer, ordentlich programmieren und niemals Parameter einer Prozedur oder Funktion mit neuen Werten belegen, wenn es nicht die Absicht der **Prozedur** ist. Eine Funktion sollte nie einen Parameter als versteckte Rückgabe ändern. Eine Funktion gibt ein Ergebnis zurück und sollte ansonsten auch nichts ändern.

Der zweite Weg ist der Parameter ByVal. Mit diesem wird der Parameter als Wert übergeben und nach dem durchlaufen der Prozedur oder Funktion, wird der alte Wert wieder gesetzt. Wenn man also sicher sein will das sich der Parameter nicht ändern kann, setzt man ByVal davor. Dies ist auch sinnvoll wenn man eine Funktion oder Prozedur schreibt die man in mehreren Programmen verwenden will.

Verwendet man in unserem Beispiel in der Prozedur jetzt ByVal

```
Sub NeuerWert(ByVal Zahl as Integer)
  Zahl = Zahl*100
End Sub
```

Kommen zum Schluß weniger große Zahl heraus.



2.3.2 Welche Operatoren gibt es?

Mathematische Operatoren

Die mathematischen Operatoren sind auf sämtliche Zahlen anwendbar. Der Operatoren + kann darüber hinaus zur Verkettung von Zeichenfolgen verwendet werden.

+ Addition von Zahlen und Datumswerten, Verketten von Zeichenfolgen

- Subtraktion von Zahlen und Datumswerten

* Multiplikation von Zahlen

/ Division von Zahlen

\ Division von Zahlen mit ganzzahligem Ergebnis (wird kaufmännisch gerundet)

^ Potenzieren von Zahlen

MOD Modulo-Operation (Berechnung des Restes einer Division)

Logische Operatoren

Logische Operatoren gestatten eine Verknüpfung nach den Regeln der booleschen Algebra. Werden sie auf boolesche Werte angewendet, so liefert die Verknüpfung direkt das gewünschte Ergebnis. Beim Einsatz im Zusammenhang mit Integer- und Long Integer-Werten erfolgt eine bitweise Verknüpfung.

AND Und-Verknüpfung

OR Oder-Verknüpfung

XOR Exklusive Oder-Verknüpfung

NOT Negation

EQV Equivalent-Test (beide Teile True oder False)

IMP Implikation (falls erster Ausdruck richtig, muss auch zweiter richtig sein)

Vergleichsoperatoren

Die Vergleichsoperatoren können auf sämtliche elementaren Variablentypen (Zahlen, Datumsangaben, Zeichenfolgen und boolsche Werte) angewendet werden.

= Gleichheit von Zahlen, Datumswerten und Zeichenfolgen

<> Ungleichheit von Zahlen, Datumswerten und Zeichenfolgen

> Größer-Prüfung für Zahlen, Datumswerte und Zeichenfolgen

>= Größer-Gleich-Prüfung für Zahlen, Datumswerte und Zeichenfolgen

< Kleiner-Prüfung für Zahlen, Datumswerte und Zeichenfolgen

<= Kleiner-Gleich-Prüfung für Zahlen, Datumswerte und Zeichenfolgen

2.3.3 Wie kann man Fehler abfangen?

Fehlerbehandlung ist eine der wichtigen Aufgaben in einem Programm. Sie dient dazu möglich Fehler während der Laufzeit in der Anwendung und des Anwender abzufangen. Für diese Fehlerbehandlung stellt StarBasic besondere Funktionen zur Verfügung.

On Error

Mit On Error beginnt jede Fehlerroutine. Mit dieser Anweisung wird die mögliche Fehlerbehandlung eingeleitet. Mögliche Fehlerbehandlung? Weil natürlich nicht alle Fehler abgefangen werden können. Es können nur Fehler abgefangen werden für StarBaisc einen Errorhandling bereitstellt. Desweiteren ist die Fehlerbehandlung im Kern nur eine Variante des Goto-Befehls und mit der entsprechenden Vorsicht zu verwenden.

On Error Goto Sprungziel

Um jetzt diese Sprungziel festzulegen wird dieselbe Logik verwendet wie bei Goto.

```
Sub Test
  Anweisung
  Anweisung
On Error Goto Fehlerbehandlung
  Anweisung
Exit Sub
Fehlerbehandlung:
  Anweisung
End Sub
```

Zusätzlich kann man noch festlegen wie das Programm nach der Fehlerbearbeitung weiterarbeiten soll. Dazu wird der Befehl Resume verwendet. Dieser kann auf zwei Arten eingesetzt werden. Die erste ist Resume Next. Dabei wird direkt nach der Zeile weitergearbeitet die den Fehler verursacht. Die andere ist Resume Sprungziel. Das entspricht eigentlich einem Goto. Dann wird das Programm an einem Sprungziel weiter abgearbeitet.

Variante 1:

```
Sub Test
  Anweisung ....
.... Exit Sub
```

Fehlerbehandlung:

Anweisung

Resume Next

End Sub

Variante 2:

Variante 1:

Sub Test

Anweisung ....

.... Exit Sub

Fehlerbehandlung:

Anweisung

Resume Weiter

Weiter:

Anweisung

End Sub

Mit der Kombination von On error und resume Next kann man auf jegliche Fehlerbehandlung verzichten. Dann wird einfach bei einem Fehler die nächste Zeile nach dem Fehler weiter angearbeitet.

Sub Test

Anweisung

On Error Resume Next

Anweisung

Anweisung

End Sub

Beendet wird die Fehlerprüfung mit dem Befehl On Error Goto 0. Damit wird die Fehlerbehandlung wieder ausgeschaltet.

Sub Test

On Error Resume Next

Anweisung

On Error Goto 0

Anweisung

End Sub

Da ein Programmier absichtlich keine Fehler programmiert, ist es mir etwas schwer gefallen ein Beispiel zu finden. Aber ich habe eines gefunden! Wir wollen eine Datei speichern, aber das Verzeichnis existiert nicht, die Datei ist dort schon gespeichert oder der Name macht keinen Sinn.

Also erstmal die Routine ohne Fehlerbehandlung.

```
Sub Speichern
  dim args(0) as new com.sun.star.beans.PropertyValue
  Dateiname = InputBox ("Geben Sie bitte den Dateinamen ein:")
  Speicherort = InputBox ("Geben Sie bitte ein Verzeichnis ein:")
  mydoc = ThisComponent
  myUrl=  Speicherort + Dateiname
  mydoc.storeasurl(myurl,args())
End Sub
```

Die potientelle Problemzeile ist "mydoc.storeasurl.(myurl.args())". Beim Aufruf dieser Zeile kann ein Fehler auftreten wenn zum Beispiel das Verzeichnis zum Speichern nicht existiert. Um diesen Fehler abzufangen machen wir in der einfachsten Version der Fehlerbehandlung nur einen Sprung. Damit ignorien wir den Fehler einfach.

```
Sub Speichern
  dim ........
  ......Dateiname
  On Error Resume Next
  mydoc.storeasurl(myurl,args())
End Sub
```

Diese Art der Fehlerbehandlung hat nur einen Nachtteil: Unser Programm reagiert nicht auf den Fehler. In unserem Beispiel wurde die Datei nicht gespeichert.

Also versuchen wir den Fehler jetzt abzufangen und eine Meldung darüber zu erzeugen. Dazu verwenden jetzt die Funktion goto in Kombination mit on error.

```
Sub SpeichernTest2
  dim ........
  ......Dateiname
  On Error Goto ErrorBehandlung
    mydoc.storeasurl(myurl,args())
    MsgBox "Datei gespeichert"
```

```
  Exit Sub
  ErrorBehandlung:
    MsgBox "Speichern gespeichert"
End Sub
```

So jetzt haben wir schon mal eine Trennung für Anwender. Er erfährt ob das Speichern geklappt oder nicht. Wir können es aber noch besser. Wir verwenden den Errorcode den StarOffice mit drei Funktionen erzeugt und geben eine Meldung über die mögliche Fehlerursache aus.

Die Errorcodes werden von Starbasic nach einem Fehler bereitgestellt und können angezeigt und ausgewertet werden.

Folgende Funktioen werden bereitgestellt:

Die Funktion Err übergibt die Nummer des aufgetretenen Fehlers.

Die Funktion Error$ stellt auch eine Beschreibung des Fehlers zur Verfügung.

Die Funktion Erl übergibt die Zeilennummer eines Fehlers.

Der einfachste Weg diese Meldung anzuzeigen ist eine Msgbox.

```
Sub SpeichernTest4
  dim ....
  ....Sub
  ErrorBehandlung:
  MsgBox "Error " & Err & ": " & Error$ & " (Zeile : " & Erl & ")"
End Sub
```

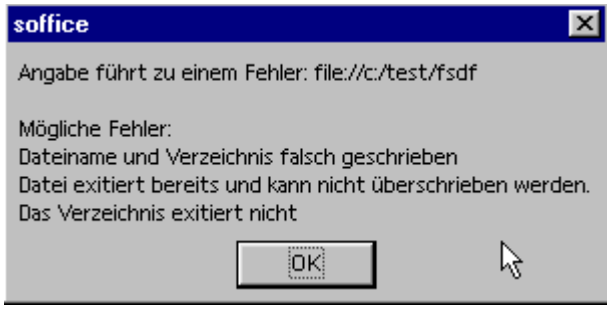
Bei einem ungültigen Dateinamen wird dann folgende Meldung erzeugt:

Das ist ja schon immerhin etwas. Jetzt haben wir eine Meldung. Damit können wir als Programmierer etwas anfangen. Naja zumindestens manchmal. Aber für den Anwender ist uns das zu wenig. Deshalb wollen wir noch zusätzlich die möglichen Fehler mit einen Text der etwas mehr aussagt. Der Error 1 tritt immer dann auf wenn eine Exception ausgelöst wird. Leider kann das bei einem falschen Dateinamen, bei vorhandener Datei mit schreibschutz, oder ungültigen Verzeichnis auftreten. Diese Ursachen können wir aber angeben. Wenn mehrere mögliche Fehlercodes gibt kann man für jeden ein eigene Meldung schreiben.

```
Sub SpeichernTest
  dim ....
```

```
....Sub
ErrorBehandlung:
  MsgBox "Error " & Err & ": " & Error$ & " (Zeile : " & Erl & ")"
  If Err=1 then
    MsgBox "Angabe führt zu einem Fehler" + myUrl  +"Mögliche Fehler:" +  Chr(13)+ _
    "Dateiname und Verzeichnis falsch geschrieben" + Chr(13)+_
    "Datei exitiert bereits und kann nicht überschrieben werden." + Chr(13)+_
    "Das Verzeichnis exitiert nicht"
  end if
End Sub
```

Jetzt bekommt der Anwender eine aussagekräftige Meldung und kann seinen Fehler korregieren.



Man sollte bei der Fehlerbehandlung daruf achten seine Proceduren und Funktioen übersichtlich zu halten. Das ist am einfachsten wenn man die Fehlerbehandlungen an das Ende setzt und vorher seine Routine abschließt. Ebenfalls sollte man auf Rücksprünge in das ursprüngliche Programm verzichten. Dies ist wichtig für eine später Fehlersuche. Besteht wirklich ein Bedarf das unter umständen noch bestimmte Aufrufe erfolgen sollen kann man diese auch am Ende als Extracode für die Fehlerbehandlung einfügen. Der beste Weg dazu ist die Anweisung erstmal ohne Fehlerbehandlung zu schreiben und entsprechend zu erweitern.

```
Sub Test
  Anweisung
  Anweisung
  Anweisung
  Anweisung
End Sub
```

Jetzt kommt die Fehlerbehandlung dazu:

```
Sub Test
  Anweisung
```

```
  On Error Goto Fehlerbehandlung
  Anweisung
  Anweisung
  Anweisung
Exit Sub
  Fehlerbehandlung:
   Anweisung
End Sub
```

Auf diese Weise bleibt der Code übersichtlich und man sieht soforft wo die eigentliche Procedure aufhört und die Fehlerbehandlung anfängt. Dies ist vor allen sinnvoll wenn man auf verscheidene Fehlerreagieren will oder sogar muß.

```
Sub Test
  Anweisung0
  On Error Goto Fehlerbehandlung1
    Anweisung1
  On Error Goto 0
    Anweisung2
  On Error Goto Fehlerbehandlung3
    Anweisung3
  On Error Goto 0
  Anweisung
Exit Sub
  Fehlerbehandlung1:
    Anweisung
Fehlerbehandlung2:
   Anweisung
End Sub
```

2.3.4 Wie kann man beim Anwender Entscheidungen abfragen?

Manchmal ist es nötig den Anwender nach einer Entscheidung zu fragen. Meistens geht es um ja oder nein. Dazu kann die Funktion MsgBox verwendet werden.
Außer das man mit dem Befehl msgbox einen Variable anzeigen kann, kann man auch eine Abfrage über eine Entscheidung damit erstellen.

Das Prinzip ist so einfach wie bei einer Inputbox:

Wert=Msgbox("Abfragetext",2)

Ergänzt werden müssen nur die Parameter für die Art der Abfrage. Die wird als Parameter mit gegeben:

Leider stimmt die Beschreibung in der Online-Hilfe nicht ganz. Zum einen stimmen die Optionen zu anderen die Rückgabewerte.

Hier die Liste für 2.0:

0 : Nur OK-Schaltfläche wird dargestellt.

Rückgabewert : 1

1 : OK- und Abbrechen-Schaltfläche werden dargestellt.

Rückgaberwerte: 1,2

2 : Abbrechen-, Wiederholen, und Ignorieren-Schaltfläche werden dargestellt.

Rückgaberwerte: 3, 4, 5

3 : Ja-, Nein- und Abbrechen-Schaltfläche werden dargestellt.

Rückgabewerte: 6,7,2

4 : Ja- und Nein-Schaltfläche werden dargestellt.

Rückgabewerte: 6,7

5 : Wiederholen- und Abbrechen-Schaltfläche werden dargestellt.

Rückgabewerte: 4,2

Zusätzlich können noch Symbole mit eingeblendet werden, dabei wird der Wert für das Symbol einfach dazu addiert:

Wert=Msgbox("Abfragtext",2+16)

16 : Das Stop-Symbol wird mit in den Dialog aufgenommen.

32 : Das Fragezeichen-Symbol wird mit in den Dialog aufgenommen.

48 : Das Erklärungs-Symbol wird mit in den Dialog aufgenommen.

64 : Das Informations-Symbol wird mit in den Dialog aufgenommen.

Und zu guter letzt kann man noch eine Schaltfläche als Vorgabe einstellen.

Wert=Msgbox("Abfragtext",2+16+128)

128 : Erste Schaltfläche im Dialog ist Standardschaltfläche.

256 : Zweite Schaltfläche des Dialogs ist voreingestellt.

512 : Dritte Schaltfläche des Dialogs ist voreingestellt.

2.3.5 Worin besteht der Unterschied zwischen thisComponent und CurrentComponent?

Es gibt zwei Möglichkeiten die Referenz eines Dokumentes zu erhalten: thisComponent oder currentComponent (oder auch StarDesktop.currentComponent)

Der Unterschied besteht darin, dass sich currentComponent auf das aktive Fenster bezieht. Das schließt z.B. die Basic-IDE ein. Mit thisComponent erfolgt der Zugriff auf das letzte aktive Dokument, aus der Sicht der IDE, oder das gerade aktuelle Dokument.

Gerade wenn man also Programme innerhalb der IDE testen will, muß man thisComponent verwenden.

Seite der Version 2.0 geht der Zugriff auf Draw-, Impress- und Base-Dokumente nicht mit thisComponent. Dies geht nur mit currentComponent.

2.3.6 Welche Runtime-Funktionen gibt es?

StarBasic bietet eine große Zahl von Runtime-Funktionen. Hier folgt eine kurze Auflistung. Details zu den Befehlen stehen in der Onlinehilfe.

Numerische Funktionen

Atn - Arkustangens	Ermittelt den Arkustangens eines numerischen Ausdrucks. Das Ergebnis wird im Bogenmaß (Radiant) zurückgegeben und liegt im Bereich -(Pi/2) und +(Pi/2).
Cos - Cosinus	Ermittelt den Kosinus eines Winkels. Der Winkel wird im Bogenmaß (in Radiant) angegeben. Das Ergebnis liegt im Bereich von -1 bis 1.
Sin - Sinus	Ermittelt den Sinus eines Winkels. Der Winkel wird im Bogenmaß (in Radiant) angegeben. Das Ergebnis liegt im Bereich von -1 bis 1.
Tan - Tangens	Ermittelt den Tangens eines Winkels. Der Winkel wird im Bogenmaß (in Radiant) angegeben.
Exp	Potenziert die Basis des natürlichen Logarithmus (die Eulersche Zahl, e = 2,718282) um die angegebene Potenz.
Log	Liefert den natürlichen Logarithmus eines numerischen Ausdrucks.
Randomize	Initialisiert den Zufallsgenerator.
Rnd	ermittelt einen Zufallswert zwischen 0 und 1.
Sqr	Ermittelt die Quadratwurzel des angegebenen numerischen Ausdrucks.
Fix	Liefert die Vorkommazahl eines numerischen Ausdrucks durch Abschneiden des Nachkommateils zurück.
Int	Liefert die Vorkommazahl eines numerischen Ausdrucks durch Abrunden zurück.
Abs	Ermittelt den Absolutwert eines numerischen Ausdrucks.

Sgn	Liefert einen ganzzahligen Wert zwischen -1 und 1, der anzeigt, ob der der Funktion übergebende Ausdruck positiv oder negativ oder Null ist.
Hex	Liefert das Ergebnis der Umrechnung einer Zahl in das Hexadezimalformat.
Oct	Liefert das Ergebnis der Umrechnung einer Zahl in das Oktalformat.
Fehlerbehandlungsfunktionen	
Erl	Gibt Aufschluß über die Zeilennummer, in der während der Programmausführung ein Fehler auftrat.
Err	Liefert nach dem Auftreten eines Fehlers während der Programmausführung einen Laufzeitfehlercode zurück, der den Fehler kennzeichnet.
Error	Liefert zu einem Fehlercode die Fehlermeldung.
On Error Goto	Verzweigt nach dem Auftreten eines Fehlers während der Programmausführung in eine Fehlerbehandlungsroutine oder setzt die Programmausführung fort.
Zeichenfolgen	
ASC	Gibt den ASCII-Wert (American Standard Code for Information Interchange) eines Zeichenfolgeausdrucks zurück.
InStr	Gibt die Position einer in einem Zeichenfolgenausdruck vorhandenen Zeichenfolge zurück.
Chr	Liefert das Zeichen, das dem angegebenen ASCII-Wert entspricht.
Val	Wandelt eine Zeichenfolge in einen numerischen Ausdruck um.
Cbyte	Wandelt eine Zeichenfolge oder einen numerischen Ausdruck in den Typ Byte um.
Space	Ergibt eine Zeichenfolge, bestehend aus Leerzeichen.
String	Erzeugt eine Zeichenfolge des angegebenen Zeichens oder des ersten Zeichens einer Zeichenkette, die der Funktion übergeben wird.
Format	Wandelt eine Zahl in eine Zeichenkette um, die einer Anweisung entsprechend formatiert wird.
LCase	Wandelt alle Zeichen einer Zeichenfolge in Kleinbuchstaben um. Verändert werden dabei nur die großgeschriebenen Buchstaben des Alphabets, die sich in der Zeichenfolge befinden. Kleinbuchstaben und alle sonstigen Zeichen bleiben unverändert.
Left	Kürzt einen Zeichenfolgeausdruck auf eine angegebene Anzahl von n Zeichen. Die linken n Zeichen werden zurückgegeben.
Lset	Ordnet eine Zeichenkette linksbündig innerhalb einer Zeichenfolgenvariablen an oder kopiert eine Variable eines benutzerdefinierten Datentyps in eine andere Variable eines

	benutzerdefinierten Datentyps.
LTrim	Schneidet alle führenden Leerzeichen eines Zeichenfolgeausdrucks ab.
Mid-Funktion, Mid-Anweisung	Gibt den angegebenen Abschnitt eines Zeichenfolgeausdrucks zurück (Mid-Funktion) oder ersetzt diesen durch eine andere Zeichenfolge (Mid-Anweisung).
Right	Kürzt einen Zeichenfolgeausdruck auf eine angegebene Anzahl von Zeichen. Den Ausgangspunkt hierfür bildet das erste rechte Zeichen der Folge.
RSet	Ordnet eine Zeichenkette rechtsbündig innerhalb einer Zeichenfolgevariablen an oder kopiert einen benutzerdefinierten Variablentyp in einen anderen.
RTrim	Entfernt alle Leerzeichen am Ende eines Zeichenfolgeausdrucks.
Trim	Schneidet alle führenden und nachfolgenden Leerzeichen eines Zeichenfolgeausdrucks ab.
UCase	Wandelt alle Zeichen einer Zeichenfolge in Großbuchstaben um. Umgewandelt werden dabei nur die kleingeschriebenen Buchstaben des Alphabets, die sich in der Zeichenfolge befinden. Großbuchstaben und alle sonstigen Zeichen bleiben unverändert.
ConvertToUrl	Wandelt eine System-Dateiangabe in einen File-URL um.
ConvertFromUrl	Wandelt einen File-URL in eine System-Dateiangabe um.
Join	Gibt aus einer Anzahl von Unterzeichenketten in einem Zeichenketten-Array eine einzige Zeichenkette zurück.
Split	Gibt aus einem Zeichenkettenausdruck ein Array von Unterzeichenketten zurück.
Len	Liefert die Anzahl der Zeichen in einer Zeichenfolge oder, wenn Variablen angegeben werden, die nicht vom Typ String (Zeichenfolge) sind, die Anzahl der Bytes, die benötigt werden, um den Inhalt der Variablen anzuzeigen.
StrComp	Vergleicht zwei Zeichenfolgeausdrücke miteinander und liefert als Ergebnis zurück, ob sie gleich waren, ob ein Leerstring dabei war oder ob der eine Ausdruck länger bzw. kürzer war als der andere.
Variablen	
CBool-Funktion	Konvertiert einen Zeichenfolgenvergleich, respektive einen Vergleich numerischer Werte in einen boolschen Ausdruck oder wandelt einen einzelnen numerischen Ausdruck in einen boolschen Ausdruck um.
CDate-Funktion	Konvertiert einen beliebigen numerischen Ausdruck oder einen Zeichenkettenausdruck in einen Wert vom Datentyp Date.
CDbl-Funktion	Konvertiert einen beliebigen numerischen Ausdruck oder einen String-Ausdruck in einen Double-Typ.
CInt-Funktion	Konvertiert einen beliebigen numerischen Ausdruck oder einen Zeichenkettenausdruck in einen Integer-Typ.
CLng-Funktion	Konvertiert einen beliebigen numerischen Ausdruck oder

	einen Zeichenkettenausdruck in einen Long-Integer-Typ.
Const-Anweisung	Definiert eine Zeichenkette als Konstante.
CSng-Funktion	Konvertiert einen beliebigen numerischen Ausdruck oder einen String-Ausdruck in einen Single-Typ.
CStr-Funktion	Konvertiert einen beliebigen numerischen Ausdruck in einen Zeichenkettenausdruck.
DefBool-Anweisung	Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.
DefDate-Anweisung	Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.
DefDbl-Anweisung	Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.
DefInt-Anweisung	Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.
DefLng-Anweisung	Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.
DefObj-Anweisung	Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.
DefVar-Anweisung	Legt den Vorgabevariablentyp fest. Legt den Vorgabevariablentyp für verschiedene Anfangsbuchstabenbereiche für die Fälle von Variablendeklarationen fest, bei denen kein Typendeklarationszeichen oder Typendeklarationsschlüsselwort angegeben wurde.
Dim-Anweisung	Deklariert eine Variable oder ein Array.
ReDim-Anweisung	Deklariert eine Variable oder ein Array.
IsArray-Funktion	Ermittelt, ob es sich bei der angegebenen Variablen um ein Array, also um ein Datenfeld handelt.

IsDate-Funktion	Überprüft, ob ein angegebener numerischer Ausdruck oder ein Zeichenkettenausdruck in den Variablentyp Date umwandelbar ist oder nicht.
IsEmpty-Funktion	Überprüft, ob eine Variable vom Typ Variant den Wert Empty (Leer) enthält.
IsNull-Funktion	Überprüft, ob eine Variable vom Typ Variant den besonderen Wert Null ("Nichts") enthält. Der Wert Null in einer Variantvariable ist dann vorhanden, wenn diese nicht initialisiert ist.
IsNumeric-Funktion	Ermittelt, ob der angegebene Ausdruck eine Zahl ist. Wird der Ausdruck als Zahl erkannt, wird True zurückgegeben, andernfalls False.
IsObject-Funktion	Stellt fest, ob es sich bei der angegebenen Objektvariablen um ein OLE-Objekt handelt. Ist der angegebene Ausdruck eine Variable vom Typ Object, wird True zurückgegeben, andernfalls False.
LBound-Funktion	Ermittelt die Untergrenze eines Arrays.
UBound-Funktion	Ermittelt die Obergrenze eines Arrays.
Let-Anweisung	Weist einer Variablen einen Wert zu.
Array-Funktion	Gibt den Typ Variant mit einem Datenfeld zurück.
DimArray-Funktion	Gibt ein Variant-Array zurück.
Option Base-Anweisung	Definiert die voreingestellte Untergrenze für Arrays als 0 bzw. 1.
Option Explicit-Anweisung	Bestimmt, daß jede im Programm-Code verwendete Variable explizit mit der Dim-Anweisung deklariert werden muß.
Public-Anweisung	Dimensioniert eine Variable oder ein Array auf Modulebene (also außerhalb einer Sub oder einer Function), damit die Variable und das Array in allen Bibliotheken und Modulen Gültigkeit hat.
Static-Anweisung	Dimensioniert eine Variable oder ein Array auf Prozedurebene (also innerhalb einer Sub oder einer Function), damit die Werte der Variablen oder des Arrays beim Verlassen der Sub oder der Function erhalten bleiben. Es gelten ansonsten die Konventionen der Dim-Anweisung.
TypeName-Funktion; VarType-Funktion	Liefert eine Zeichenkette (TypeName) oder einen numerischen Wert (VarType) zurück, der oder die Informationen über eine Variable gibt.
Set-Anweisung	Setzt einen Objektverweis auf eine Variable oder auf eine Eigenschaft.
FindObject-Funktion	Ermöglicht es, Objekte zur Laufzeit über ihren Namen als String-Parameter anzusprechen.
FindPropertyObject-Funktion	Ermöglicht es, Eigenschaften vom Typ Objekt zur Laufzeit über ihren Namen als String-Parameter anzusprechen.
Optional (in Function-Anweisung)	Ermöglicht es, Argumente, die einer Function übergeben werden, als optionale Argumente zu definieren. Alle weiteren Argumente nach dem ersten optionalen Argument in der Liste müssen ebenfalls als optional definiert sein.
IsMissing-Funktion	Ermöglicht es, festzustellen, ob einer Funktion ein optionales

	Argument übergeben wurde.
Steuern von Programmabläufen	
If...Then...Else-Anweisung	Definiert einen oder mehrere Befehlsblöcke, die jeweils nur dann ausgeführt werden, wenn eine bestimmte Bedingung wahr ist.
Select...Case-Anweisung	Definiert einen oder mehrere Befehlsblöcke, die in Abhängigkeit von dem Ergebnis eines bestimmten Ausdrucks stehen.
IIf-Anweisung	Liefert eines zweier möglicher Funktionsergebnisse zurück, abhängig vom Wahrheitsgehalt eines zu überprüfenden Ausdrucks.
Schleifen	
Do...Loop-Anweisung	Wiederholt die zwischen der Do- und der Loop-Anweisung aufgeführten Anweisungen solange (While) die angegebene Bedingung wahr ist oder bis (Until) die angegebene Bedingung wahr wird.
For..Next-Anweisung	Wiederholt die zwischen der For- und der Next-Anweisung aufgeführten Anweisungen so oft, bis der Schleifenzähler den angegebenen Wert erreicht hat.
While..Wend-Anweisung	Trifft das Programm auf eine While-Anweisung, wird überprüft, ob die Bedingung (Condition) wahr ist. Ist sie falsch, geht es mit der Programmausführung direkt hinter Wend weiter.
Sprünge	
GoSub...Return-Anweisung GoTo-Anweisung	Sorgt dafür, daß das Programm innerhalb einer Unterroutine (Sub) oder einer Funktion (Function) mit der Abarbeitung des Programmteils fortfährt, der durch eine Marke gekennzeichnet ist.
On...GoSub-Anweisung; On...GoTo-Anweisung	Verzweigt an eine von mehreren angegebenen anderen Stellen des Programms in Abhängigkeit eines numerischen Ausdrucks.
Weitere Anweisungen	
Call	Übergibt die Programmsteuerung einer Sub, einer Function oder einer DLL.
Choose	Gibt aus einer Argumentliste den gewählten Wert zurück.
Declare	Deklariert und definiert eine Unterroutine aus einer DLL (Dynamic Link Library) für den Aufruf aus StarOffice Basic heraus.
End	Beendet eine Prozedur oder einen Prozedurabschnitt (Block).
Exit	Beendet einen Block, der mit Do...Loop, For...Next, Function oder Sub definiert wurde.
FreeLibary	Gibt per Declare-Anweisung geladene DLLs frei. Eine freigegebene DLL wird beim Aufruf einer ihrer Funktionen automatisch neu geladen. Siehe auch: Declare
Function	Definiert eine Unterroutine, die als Ausdruck verwendet werden kann, also einen Rückgabetyp liefert.

Rem	Erlaubt die Definition einer Programmzeile oder des auf REM folgenden Teils einer Programmzeile als Kommentar. Der Rest der Programmzeile nach dem REM wird ignoriert.
Stop	Beendet das Programm.
Sub	Definiert eine Unterroutine.
Switch	Wertet eine Argumentliste mit Ausdrücken und Werten aus, in der jeweils ein Wert einem Ausdruck folgt. Ein Ausdruck und ein Wert bilden eine Zuordnungseinheit. Die Switch- Funktion gibt den Wert zurück, der dem Ausdruck zugeordnet ist, der dieser Funktion übergeben wurde.
With	Bestimmt ein Objekt zum Vorgabeobjekt, auf das sich alle Eigenschaften oder Methoden bis zur End With-Anweisung beziehen, solange keine weiteren Objektnamen angegeben werden.
Logische Operatoren	
And-Operator	Führt eine logische UND-Verknüpfung zwischen zwei Ausdrücken durch.
Eqv-Operator	Führt eine logische Äquivalenz-Verknüpfung zwischen zwei Ausdrücken durch.
Imp-Operator	Führt eine logische Implikationsverknüpfung zwischen zwei Ausdrücken durch.
Not-Operator	Negiert einen Ausdruck durch Vertauschen der Bitzustände.
Or-Operator	Führt eine logische ODER-Verknüpfung zwischen zwei Ausdrücken durch.
Xor-Operator	Führt eine logische Exklusiv-Oder-Verknüpfung zwischen zwei Ausdrücken durch.
Bildschirmein-/Ausgabefunktionen	
MsgBox-Anweisung und MsgBox-Funktion	Stellt einen Dialog auf dem Bildschirm dar, der einen Hinweistext enthält. Mit der Anweisung läßt einfach eine Meldung generieren. Mit der Funktion bekommt den gedrückten Button als Rückgabewert.
Print	Gibt die angegebenen Zeichenfolgen oder numerischen Ausdrücke in einem Dialog auf dem Bildschirm aus.
InputBox	Zeigt einen Dialog mit einem Eingabefeld und erlaubt dem Benutzer die Eingabe einer Zeichenfolge bis zur Bestätigung von OK. Die Eingabe wird in einer Variablen aufgenommen.
Farbbrechnungsfunktionen	
Blue	Ermittelt den Blau-Anteil des angegebenen Farbcodes.
Green	Ermittelt den Grün-Anteil des angegebenen Farbcodes.
Red	Ermittelt den Rot-Anteil des angegebenen Farbcodes.
QBColor	Gibt den RGB-Farbcode der Farbe zurück, die der Funktion als Farbnummer eines älteren MS-DOS-Programmiersystems übergeben wurde.
RGB-Funktion	Ermittelt einen Long-Integer-Farbwert aus Rot-, Grün- und Blauanteilen.
Dateiein-/Ausgabefunktionen	

Close	Schließt eine Datei, die mit der Open-Anweisung geöffnet wurde.
FreeFile	Ermittelt die nächste freie Dateinummer zum Öffnen einer Datei. Das ist sinnvoll, wenn Sie bereits mehrere Dateien geöffnet haben, eine weitere Datei öffnen möchten und nicht sicher sind, welche Dateinummer noch nicht belegt ist.
Open	Öffnet einen Datenkanal.
Reset	Schließt alle geöffneten Dateien und schreibt den Inhalt aller Dateipuffer auf den Datenträger.
Get	Liest einen Datensatz aus einer relativen Datei oder eine Folge von Bytes aus einer binären Datei in eine Variable ein.
Input	Erlaubt das Einlesen von Daten aus einer geöffneten sequentiellen Datei.
Line Input	Erlaubt das sequentielle Einlesen von Zeichenfolgen aus einer Datei.
Put	Schreibt einen Datensatz in eine relative Datei oder eine Folge von Bytes in eine binäre Datei.
Write	Schreibt Daten in eine sequentielle Datei.
Loc	Gibt Aufschluß über die gegenwärtige Position innerhalb einer geöffneten Datei.
Seek	Ermittelt die Position für den nächsten Schreib- oder Lesevorgang in einer Datei, welche mit der Open-Anweisung geöffnet wurde. Der Rückgabewert entspricht dem durch die Seek-Anweisung festgelegten Wert:
Eof	Stellt fest, ob der Dateizeiger das letzte Byte einer Datei erreicht hat.
Lof	Liefert die Größe einer geöffneten Datei zurück.
ChDir	Wechselt das aktuelle Verzeichnis oder/und das aktuelle Laufwerk. Achtung diese Funktion geht unter Windows nicht!
ChDrive	Wechselt das aktuelle Laufwerk.
CurDir	Liefert den auf dem angegebenen oder aktuellen Laufwerk derzeit eingestellten Pfad zurück.
Dir	Ermittelt den Namen einer Datei oder eines Verzeichnisses oder alle Dateien und Verzeichnisse eines Laufwerkes oder eines Verzeichnisses, auf die der angegebene Suchpfad paßt.
FileAttr	Liefert den Zugriffsmodus oder die MS-DOS-Dateikennung einer Datei zurück, die mit der Open-Anweisung geöffnet wurde.Die MS-DOS-Dateikennung ist eine Zugriffsnummer des Betriebssystems und daher vom jeweiligen Betriebssystem abhängig (OSH = Operating System Handle).
FileCopy	Kopiert eine Datei.
FileDateTime	Liefert eine Zeichenfolge zurück, die Datum und Uhrzeit der Erstellung oder letzten Modifizierung einer Datei enthält.
FileExists	Ermittelt, ob eine Datei oder ein Verzeichnis auf dem Datenträger vorhanden ist.

FileLen	Gibt die Länge einer Datei in Bytes zurück.
GetAttr	Liefert ein Bitmuster zurück, daß den Dateityp kennzeichnet oder anzeigt, ob es sich um eine Volumebezeichnung oder ein Verzeichnis handelt.
Kill	Löscht eine Datei von einem Datenträger.
MkDir	Erstellt ein Verzeichnis auf einem Datenträger.
Name	Gibt einer bereits vorhandenen Datei oder einem bereits vorhandenen Verzeichnis einen neuen Namen.
RMDir	Löscht ein Verzeichnis auf einem Datenträger.
SetAttr	Setzt die Dateiattribute der angegebenen Datei.

Vergleichsoperatoren

=	gleich
<	kleiner als
>	größer als
<=	kleiner gleich
>=	größer gleich
<>	ungleich

Sonstige Befehle

Beep	Gibt einen Ton über die Computer-Lautsprecher aus. Der ausgegebene Ton ist systemabhängig und in der Regel in Lautstärke und Tonhöhe nicht beeinflußbar.
Shell	Startet ein anderes Anwendungsprogramm und bestimmt bei Bedarf dessen Darstellungsform.
Wait	Unterbricht die Programmausführung für die Dauer der angegebenen Anzahl an Millisekunden.
GetSystemTicks	Liefert die Systemticks die vom verwendeten Betriebssystem bereitgestellt werden
Environ	Gibt den Wert einer Environment-Variablen als String zurück. Environment-Variablen sind Umgebungsvariablen, die vom jeweiligen Betriebssystem abhängig sind.
GetSolarVersion	Ermittelt eine interne Nummer für die aktuell verwendete StarOffice-Version.
TwipsPerPixelX	Ermittelt, wieviele Twips ein Bildschirmpixel in der Breite einnimmt.
TwipsPerPixelY	Ermittelt, wieviele Twips ein Bildschirmpixel in der Höhe einnimmt.

Datum- und Zeitfunktionen

DateSerial	Liefert einen Wert vom Typ Date zurück, der die angegebene Jahres-, Monats- und Tageszahl enthält.
Datevalue	Erzeugt aus einer Datumszeichenfolge eine fortlaufende Nummer, die in einem einzigen numerischen Wert ein komplettes Datum darstellt. Mit Hilfe dieser fortlaufenden Nummer können Sie Differenzen zwischen Daten berechnen.

Day	Liefert aus einer fortlaufenden Datumszahl, die mit DateSerial oder mit DateValue generiert wurde, den Tag des Datums im entsprechenden Monat zurück.
Month	Liefert aus einer fortlaufenden Datumszahl, die mit DateSerial oder mit DateValue generiert wurde, den Monat des Datums im entsprechenden Jahr zurück.
WeekDay	Liefert aus einer fortlaufenden Datumszahl, die mit mit DateSerial oder mit DateValue generiert wurde, die Nummer des Wochentags zurück.
Year	Liefert aus einer fortlaufenden Datumszahl, die mit DateSerial oder mit DateValue generiert wurde, die Jahreszahl zurück.
CDateToIso	Liefert aus einer fortlaufenden Datumszahl, die mit DateSerial oder mit DateValue generiert wurde, das Datum im ISO-Format zurück.
CDateFromIso	Liefert aus einer Zeichenkette, die einem Datum im ISO-Format entspricht, die interne Datumszahl zurück.
Hour	Liefert aus einer fortlaufenden Zeitzahl, die mit TimeSerial oder mit TimeValue generiert wurde, die Stunde der Uhrzeit zurück.
Minute	Liefert aus einer fortlaufenden Zeitzahl, die mit TimeSerial oder mit TimeValue generiert wurde, die Minute der gesamten Uhrzeit zurück.
Now	Gibt das aktuelle Systemdatum und die aktuelle Systemzeit als einen Wert vom Datentyp Date zurück.
Second	Liefert aus einer fortlaufenden Zeitzahl, die mit TimeSerial oder mit TimeValue generiert wurde, die Sekunde der gesamten Uhrzeit zurück.
TimeSerial	Erzeugt aus der Angabe von Stunde, Minute und Sekunde - Parameter, die als numerische Werte übergeben werden - eine fortlaufende Nummer, die in einem einzigen numerischen Wert eine komplette Uhrzeit darstellt. Mit Hilfe dieser fortlaufenden Nummer können Sie Differenzen zwischen Uhrzeiten berechnen.
TimeValue	Erzeugt aus der Angabe von Stunde, Minute und Sekunde - Parameter, die als Zeichenfolge übergeben werden - eine fortlaufende Nummer, die in einem einzigen numerischen Wert eine komplette Uhrzeit darstellt. Mit Hilfe dieser fortlaufenden Nummer können Sie Differenzen zwischen Uhrzeiten berechnen.
Date	Liefert das aktuelle Systemdatum als Zeichenfolge zurück oder setzt das Datum neu. Das Datumsformat ist abhängig von den länderspezifischen Einstellungen in der Systemsteuerung. Wurde hier "Deutschland" gewählt, wird die Zeit im Format "TT.MM.JJJJ" gezeigt.
Time	Diese Funktion liefert die aktuelle Systemzeit als Zeichenfolge im Format "HH:MM:SS" zurück.
Timer	Liefert einen Wert zurück, der die vergangene Zeit seit Mitternacht in Sekunden angibt.

Mathematische Operatoren

-	Subtrahiert zwei Werte.

*	Multipliziert zwei Werte.
+	Addiert zwei Ausdrücke oder fügt diese zusammen.
/	Dividiert zwei Werte.
^	Potenziert einen Wert.
Mod	Liefert den ganzzahligen Rest einer Division.

2.3.7 Wie kann man Bibliotheken kopieren?

Jedes Dokument hat zwei Objekte mit denen auf die Bibliotheken zu gegriffen werden kann:
BasicLibraries und DialogLibraries.

oBasicLibrary = BasicLibraries

oDlgLibrary = DialogLibraries

Zusätzlich gibt es diese beiden Objekte noch global für die installierten Bibliotheken.

oGlobalBasicLibrary = Globalscope.BasicLibraries

oGlobalDlgLibrary = Globalscope.DialogLibraries

Über die Elemente der Bibliothek kann man diese nun löschen und einfügen.

Einfügen oGlobalBasicLibrary.createLibrary("Bibliothek")

Einfügen oGlobalDlgLibrary.createLibrary("Bibliothek")

Wichtig ist hierbei zu wissen, dass Starbasic eine Bibliothek eigentlich zweimal anlegt. Eine als Container für Module und eine als Container für Dialoge.Man muß daher auch beide erzeugen und später auch löschen. Das heißt die folgenden Schritte müssen zweimal ausgeführt werden.

Um die eigentlichen Makros zu kopieren, holt man sich die aus der QuellBibliothek

oquell_Lib = oBasicLibrary.getByName(quelle_lib)

'Diese Elemente müssen kopiert werden.

quellModules = oquell_Lib.getElementNames()

'Wenn alles ok kopieren der Elemente

If oLib.hasByName(quelle_lib) Then

```
            oquell_Lib = oLib.getByName(quelle_lib)
            quellModules = oquell_Lib.getElementNames()
            for Zaehler=0 to ubound( quellModules())
                oZiel_Lib = oGlobalLibrary.getByName(ziel_lib)

                If oZiel_Lib.hasByName( quellModules(zaehler) ) = False Then
                    oZiel_Lib.insertByName( quellModules(zaehler),oquell_Lib.getByName( quellModules(zaehl
er)))
                End If
            Next Zaehler
        End If
```

Das war es schon.

Löschen?

Einfach mit removelibrary

```
    oGlobalLibrary = GlobalScope.DialogLibraries
    oGlobalLibrary.removeLibrary( Libname )
    oGlobalLibrary = GlobalScope.BasicLibraries
    oGlobalLibrary.removeLibrary( Libname )
```

Auch wieder zweimal. Einmal Dialoge und einmal Module.

Siehe auch Tools vom Dannenhöfer.

Dort sind die Makros schon komplett hinterlegt.

2.3.8 Wie kann man Makros aus anderen Bibliotheken aufrufen?

Um ein Makro aus einer anderen Bibliothek aufzurufen, muß man diese Bibliothek in seinem Makro oder Modul öffnen.
Dazu ist die Funktion BasicLibraries.LoadLibrary gedacht.

Hierbei wird zwischen den Bibliotheken in einem Dokument und den Bibliotheken in StarOffice- Anwendung unterschieden.

Wenn man innerhalb eines Dokumentes auf eine andere Bibliothek zugreifen will, langt der Aufruf

BasicLibraries.LoadLibrary("Bibliothek")

Damit stehen alle Module und Dialoge der Bibliothek zur Verfügung.

Möchte man aus einem Dokument auf eine Bibliothek zugreifen, die in StarOffice installiert ist, muß man den Befehl noch "Globalscope" voranstellen.

GlobalScope.BasicLibraries.LoadLibrary("Bibliothek")

Ein Zugriff auf Bibliotheken anderer Dokumente (auch geöffneter) ist nicht möglich.

2.3.9 Wie kann man eine Datei auf einen FTP-Server kopieren?

Dies geht auch mit dem Befehl FileCopy. Nur das man dann die Zugangsdaten und den FTP-Server mit angeben muß.
filecopy "c:\test.ods", "ftp://login:password@ftp-server/meinpfad/test.ods"

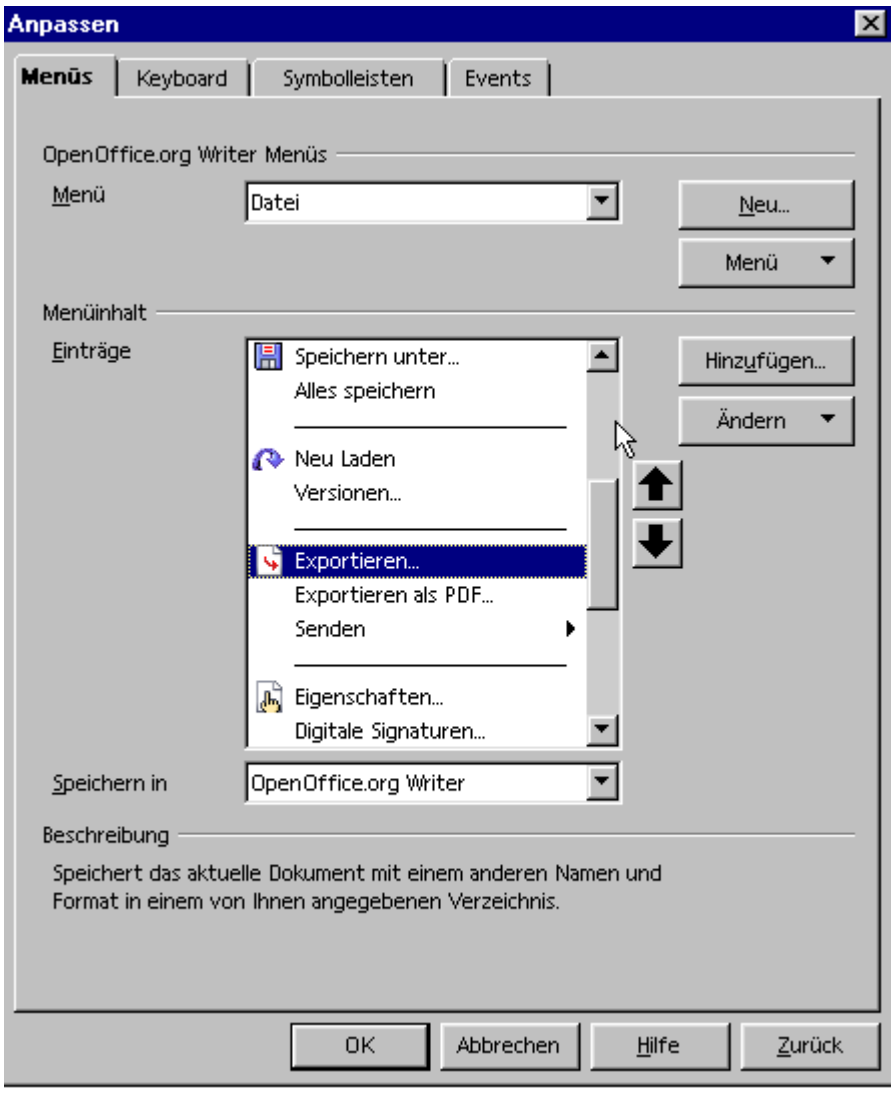
Umgekehrt geht es natürlich auch.

filecopy "ftp://login:password@ftp-server/meinpfad/test.ods","c:\test.ods"

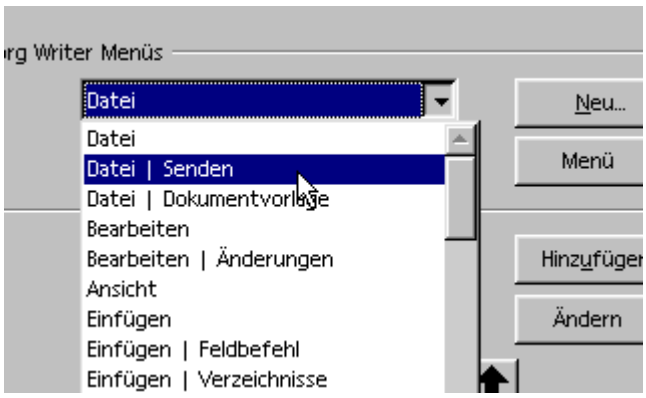
2.3 Sonstiges

2.4.1 Wie kann man Makros in Menüs einfügen -manuell?

Mit OpenOffice kann man sich eigene Makros auch in die Menüs intergrieren. Damit kann man den Aufruf der Makros jeder Zeit ermöglichen. Der Dialog um die Menüs anzupassen und sich auch dort eigene Makros einzufügen geht wieder über Extras ->> Anpassen -> diesmal Register Menü.
Ähnlich wie bei den Ereignissen gibt es hier zwei zusätzliche Möglichkeiten. Man kann die Menüs der Anwendung anpassen oder die Menüs eines Dokumentes. Hier handet es sich um die Anapssung der Anwendung und nicht des Programms. Da jede Anwendung (Writer, Calc, etc.) eigene Menüs hat werden diese hier getrennt verwaltet. Je nach Anwendung hat man unten die Auswahl "Speichern in" OpenOffice.xxx oder Dokument. Wählt man das Dokument aus werden die hinzugeügten Punkte nur in diesem Dokument angezeigt. Wenn es sich um eine Vorlage handelt, werden die Menüpunkte nur in neuen Dokumenten auf Basis dieser Vorlage anzeigt.

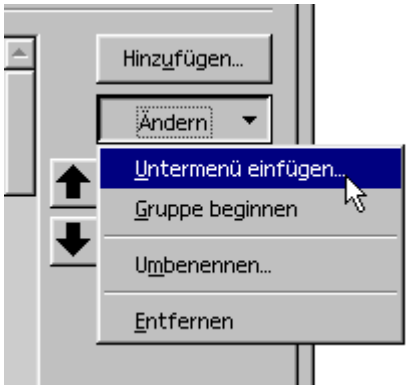


In der oberen Auswahl kann man sich den Menüpunkt und wenn vorhanden die weiteren Unterpunkte auswählen. Mit dem Button "Menü" kann man die vorhanden Menüs verschieben, umbennen oder löschen. Dies geht nicht bei den Standardmenus von OpenOffice (Datei, Bearbeiten, Ansicht, Einfügen, Format, Extras, Fenster, Hilfe, so wie den zusätzlichen Menüs der einzelnen Anwendungen, z.B. Tabelle bei Writer). Diese lassen sich nur verschieben. Unterhalb dieser Menüpunkte sind aber Änderungen möglich.



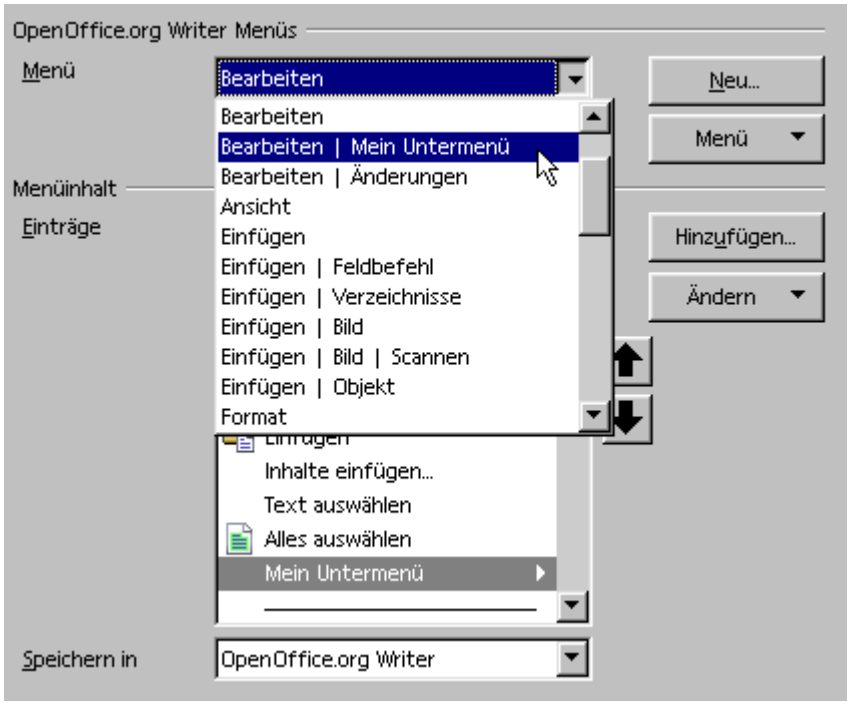
Hat man diese Auswahl getätigt, kann man sich die Menüpunkte anpassen. Auf der rechten sind zwei Buttons. Der Obere davon fügt einen neuen Menüpunkt hinzu. Der Untere öffnet eine Auswahl mit den Möglichkeiten Untermenüs oder Gruppen einzufügen, Menüpunkte und Untermenüs umzubennen oder zu

entfernen.

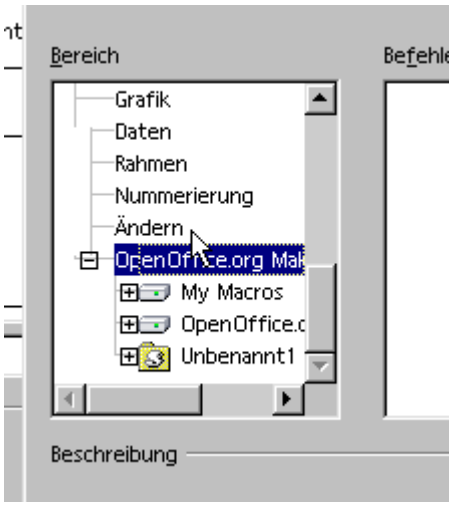


Gruppe einfügen ist nicht gerade die beste Bezeichnung dafür, das ein Trennstrich eingefügt wird. Aber wenn man es weiß, muß man es ja nicht verstehen. Ebenso ist es etwas verwirrend das man Untermenüs unter Ändern hinzufügt. Wenn man ein Untermenü einfügt, kann man dieses erst bearbeiten wenn man es in der oberen Auswahl gewählt hat. Es ist auch Möglich Unternmenüs unter Untermenüs einzufügen. Klingt Toll. Und kann man noch toppen: Unter die Untermenüs von Untermenüs kann man auch wieder Untermenü einfügen. Ich habe es bis zur siebenten Ebene ausprobiert, dann ist mir eingefallen das ich eh nicht mehr wie Eine brauche.

Beim Einfügen von Untermenüs muss man aufpassen. Die Unterpunkte eines Menüs lassen sich nicht in andere Untermenüs verschieben. Man kann nur innerhalb einer Ebene Menüpunkte verschieben.



Einzelne Menüpunkte werden über einen neuen Dialog hinzugefügt. In diesem kann man vorhandene Befehle von OpenOffice hinzufügen und sich auf diese Weise seine eigenen Menüs zusammenstellen. Ganz unten kann man die eigenen Makros hinzufügen. Warum die möglichen Befehle für Makrosmit einem großgeschrieben BASIC angezeigt werden, bleibt irgendeinem sein Geheimnis. Die Makros sind unten! Dort kann man links wieder die Bibliotheken und Module auswählen. Rechts erscheinen dann die Makros. Der Makroname wird dann automatisch als Menüname verwendet. Diesen kann man dann nachträglich über den Button "Ändern" -> Umbenennen anpassen.



Auch hier gibt es wieder eine Falle. Auch wenn sich die Makros geöffeneter Datein in die Menüs der Anwendung einfügen lassen, werden diese nich gestartet wenn die Datei nicht geöffenet ist. Stattdessen kommt eine Fehlermeldung.

Eigene Symbole lassen sich den eigenen Menüpunkten nicht zuordnen.

2.4.2 Wie kann man Makros mit Programm- oder Dokumentereignissen verbinden?

OpenOffice bietet die Möglichkeit Makros mit Ereignisse des Programmes zu verknüpfen. Ein solches Ereignis ist zum Beispiel der Programmstart. Aber es gibt natürlich noch weitere Ereignisse. Mit einigen dieser Ereignisse kann man den Ausruf von Makros verknüpfen.

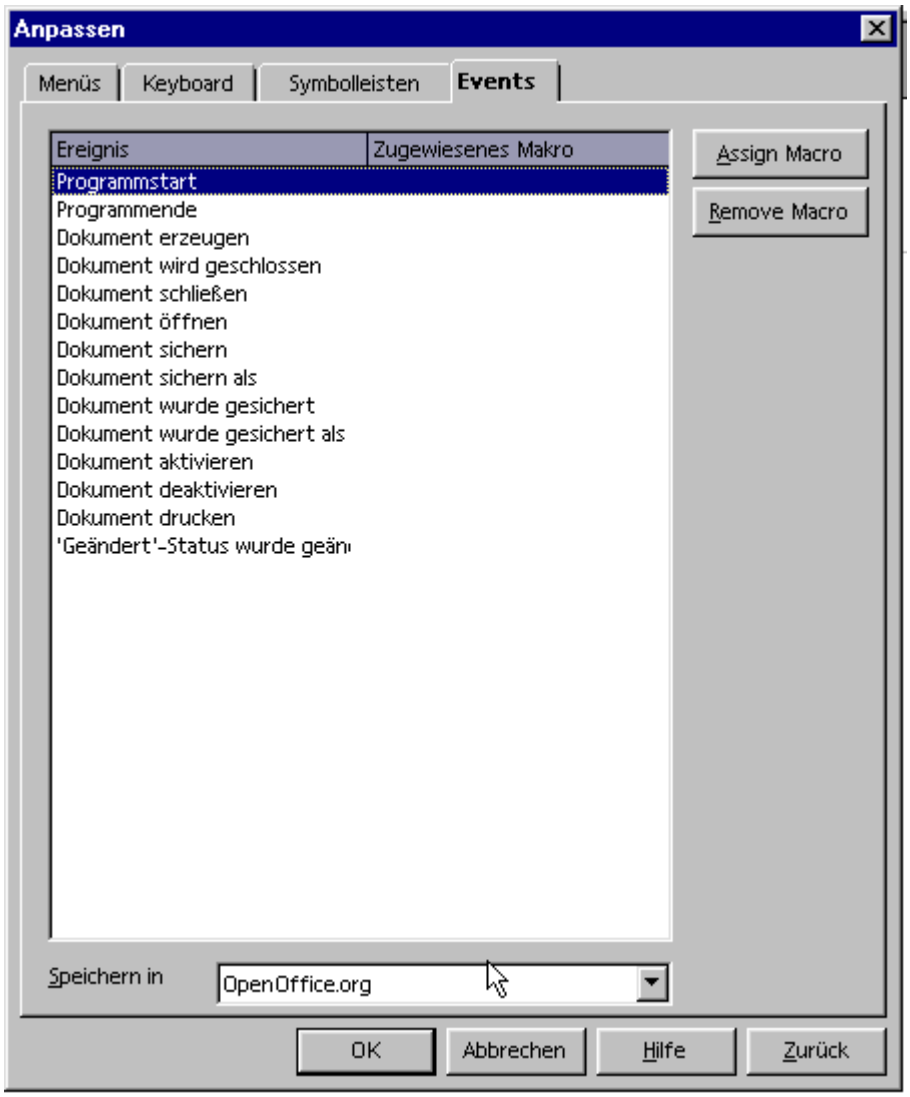
Dabei kann man zwei Arten von Ereignissen unterscheiden: Solche die vom Programm allgemein ausgehen und solche die mit einem Dokument verbunden sind. Das klingt ein wenig undurchsichtig ist aber gar nicht so schwer. Der Programmstart ist ein ziemlich eindeutiges Ereignis des Programmes. Mit diesem Ereignis kann man Makros verknüpfen. Zum Beispiel um Parameter zusetzen. Das Öffnen eines Dokumentes ist auch ein Ereignis. Aber jetzt gibt es zwei Sichtweisen und auch zwei Ereignisse. Erst mal ist es ein Ereignis des Programmes. Dann ist es aber auch ein Ereignis des Dokumentes. Wie? Naja, im ersten Fall denkt sich das Programm: "Juhu ich bekomme arbeit, ich öffne ein Dokument". Im zweiten Fall denkt sich aber das Dokument: "Mist ich habe so schön geschlafen und jetzt werde ich geöffnet". Das erste Ereignis bezieht sich auf jedes Dokument, das geöffnet wird. Der zweite Fall bezieht sich nur auf das geöffnete Dokument. Diese zwei Ereignis treten natürlich zeitgleich auf.

Ereignisse die das Programm allgemein betreffen kann man dazu Nutzen Routinen ablaufenzulassen die immer erfolgen sollen. Zum Beispiel kann man dafür sorgen das ein neues Dokument immer sofort gespeichert werden muß. Dann verbindet man mit dem Erzeugen eines neuen Dokumentes eine Speicher-Routine.

Möchte man aber das ein Makro nur beim Öffnen eines bestimmten Dokumentes erfolgt, verbindet man

diese mit dem Öffnen-Ereignis des Dokumentes. Beziehungsweise bei Dokumentenvorlagen mit den Erzeugen-Ereignis.

Diese Verknüpfung erfolgt im Menü Extras ->> Anpassen -> Register Ereignisse (Events).



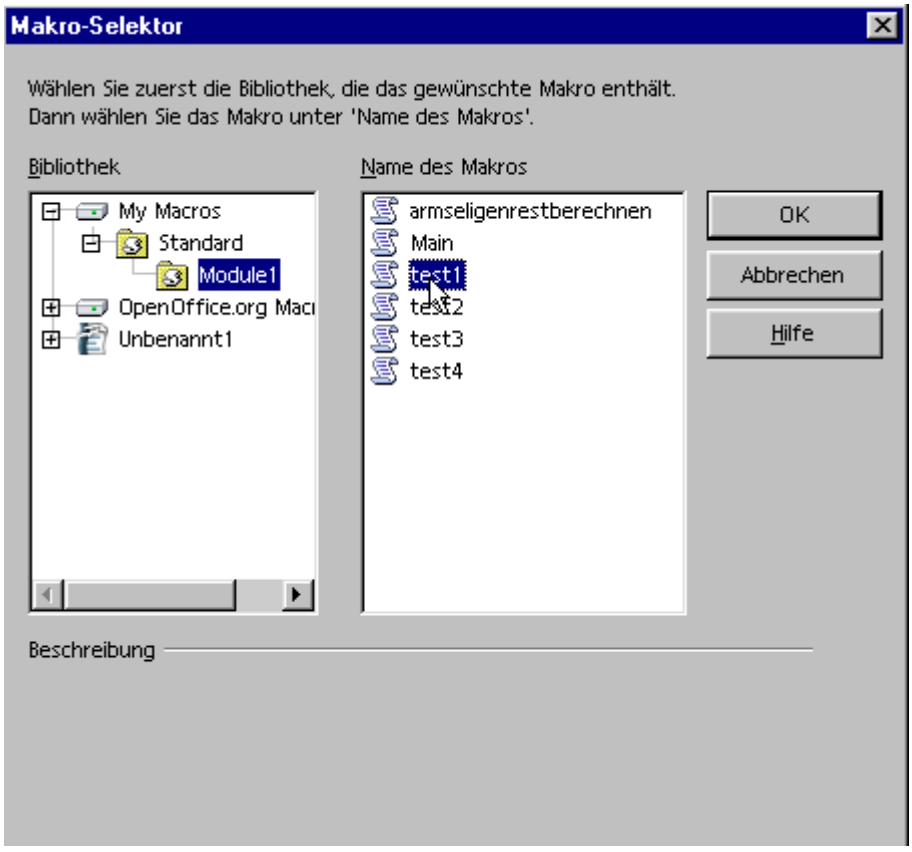
Unten, bei "Speichern in" erfolgt die Zuordnung ob das Ereignis sich auf das Programm bezieht oder auf ein Dokument.

Welche Ereignisse gibt es?

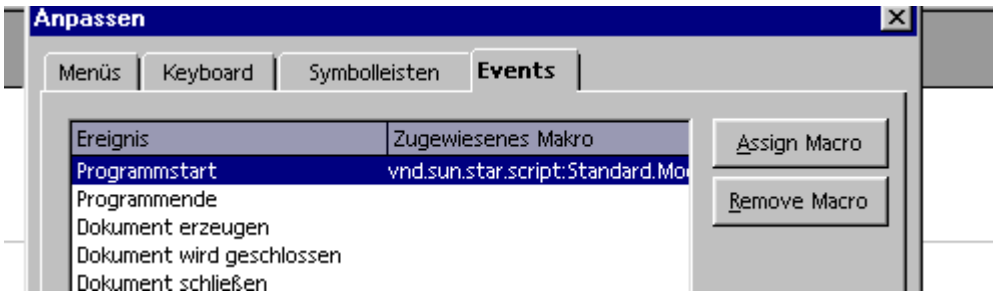
		Programm	Dokument
Programmstart		X	X
Programmende		X	X
Dokument erzeugen	Neues Dokument	X	X
Dokument wird geschlossen	Keine Ahnung worin bei den beiden der	X	X
Dokument schließen	Unterschied besteht	X	X

Dokument gesichert als	Ereignis nach dem "Speichern unter"-Dialog	X	X
Dokument wurde gesichert als	Ereignis nach dem Speichern	X	X
Dokument sichern	Direkt vor dem speichern	X	X
Dokuement wurde gesichert	Direkt nach dem speichern	X	X
	Achtung Falle: Dokument sichern bei einem noch nicht gespeicherten Dokument führt zu dem Ereignis Dokument sichern als		
Dokument drucken	Nach dem Druckdialog vor dem Drucken	X	X
Dokument wurde gedruckt	Nach dem Drucken		X
Dokument aktiviert	Dokument erhält den Fokus der Anwendung	X	X
Dokument deaktiviert	Der Fokus wechselt von dem Dokument weg	X	X
Geändert Status wurde geändert	Dokument wurde geändert	X	X
Serienbrief drucke			Nur Writer
Seitenzahl hat sich geändert			Nur Writer

Die Ereignisse werden im obigen Menü mit "Makro zuweisen ausgewählt. Links wählt man die Bibliothek aus. Zur Auswahl stehen hier "MyMakros", "OpenOffice Makros" und die Bibliotheken der geöffneten Dateien. Rechts kann man dann nach der Auswahl des Moduls das gewünschte Makro auswählen. Durch die Auswahl der Bibliothek in geöffneten Dokumeten darf man sich nicht verwirren lassen. Verknüpft man hier ein Makro mit einem OpenOffice-Ereignis wird dieses nicht ausgeführt.



Um die Verknüpfung wieder aufzuheben, wählt man das Ereignis aus und drückt den Button "Zuweisung aufheben" im Dialog "Anpassen".



2.4.3 Warum gehen die Funktioen ChDir und ChDrive nicht?

Eigentlich die falsche Frage. Richtig wäre, warum sie beschrieben werden. Diese beiden Funktionen sind Placebos. Sie existieren, aber sie haben keinerlei Funktion. Sie führen auch nicht zu einer Fehlermeldung. Warum diese beiden Blindgänger immer noch vorhanden sind, und vor allem warum sie immer noch in der Hilfe beschrieben werden, bleibt eines der vielen Rätsel von OpenSource-Projekten.

2.4.4 Wo werden die Makros gespeichert?

Die Makros werden an zwei Stellen gespeichert:
Im Userverzeichnis: User\Anwendungsdaten\OpenOffice\user\....

und im Share-Verzeichnis von OO/SO

Programme\OpenOffice\Share....

2.4.5 Warum gehen die Schaltflächen in meinen Dialogen nicht mehr?

Wahrscheinlich weil die Dialoge an der "falschen" Stelle stehen. Seite der Version 2.0 wird bei der Zuweisung von Kontrollelementen zusätzlich der Parameter "Document" oder "Application" mitgegeben. Wird nun die Bibliothek mit den Makros kopiert, entweder aus OpenOffice in ein Dokument oder umgekehrt, stimmt diese Verknüpfung nicht mehr.
Mein aktueller Stand ist: Von Hand die Verknüpfungen neu machen.

2.4.6 Wie kann man die Benutzerdaten auslesen?

Bei der Installation hat man die Möglochkeiten die Benutzerdaten einzutragen. Man kann dies auch später unter Extras->Optionen machen. Diese Benutzerdaten lassen sich auch mit Starbasic auslesen. Über die Verwendung des Service com.sun.star.configuration.ConfigurationProvider kann man diese auslesen.

Dim regval(0) as new com.sun.star.beans.PropertyValue

userdaten = createUnoService("com.sun.star.configuration.ConfigurationProvider")

regval(0).Name = "nodepath"

regval(0).Value = "org.openoffice.UserProfile/Data"

benutzerdaten = userdaten.createInstanceWithArguments("com.sun.star.configuration.ConfigurationAccess", regval())

Benutzerdaten hat nun die folgenden Proberties mit den entsprechenden Inhalten.

Firma -> benutzerdaten.o

Vorname ->benutzerdaten.givenname

Nachname -> benutzerdaten.sn

Kürzel -> benutzerdaten.initials

Titel -> benutzerdaten.title

Position > benutzerdaten.position

Telefon geschäftlich > benutzerdaten.telephonenumber

Fax ->benutzerdaten.facsimiletelephonenumber

Telefon privat -> benutzerdaten.homephone

e-mail -> benutzerdaten.mail

Strasse -> benutzerdaten.street

PLZ -> benutzerdaten.postalcode

Land -> benutzerdaten.c

Ort -> benutzerdaten.l

2.4.7 Wie kann man ein Mail verschicken?

Man kann mit StarOffice/OpenOffice.org Dokumente als Anhang an ein e-Mail versenden. Dies erfolgt über einen Service der auf das in den Optionen hinterlegte Mailprogramm zugreift (Extras - Optionen - Gemeinsame Programme) com.sun.star.system.SimpleSystemMail.

Achtung: Bei mir funktioniert es mit Windows XP und Thunderbird/Outlook. Tests mit Windows 98 und Outlook Express 6 und Outlook 2000 führen zum Totalabsturz von OO und SO. Wenn jemand andere Erfahrungen hat bitte melden.

Dieser Service muß neu erzeugt werden. Danach steht ein Objekt für die eigentliche Nachricht zur Verfügung.

Bis OO 1.1.5

MailProgramm = MailProgrammSystem.querySimpleMailClient()

Ab OO 2.0

oMailer = createUnoService("com.sun.star.system.SimpleSystemMail")

MailProgramm = omailer.querySimpleMailClient()

NeueNachricht = MailProgramm.createSimpleMailMessage()

Diese neue Nachricht kann dann mit Inhalt gefüllt werden und abschließend versendet werden. Also an das aktive Mailprogramm übergeben werden. Die wichtigsten Inhalt sind der Empfänger, das Subjekt, und einen Anhang. Anhänge werden als URL-Adresse in einem Array übergeben.

NeueNachricht = MailProgramm.createSimpleMailMessage()

NeueNachricht.setRecipient("empfaenger@domain.de")

NeueNachricht.setSubject("thema des mails")

Dim attachs(0)

attachs(0)="file:///c:/test.sxw"

NeueNachricht.setAttachement(attachs())

Anschließend kann man das Mail versenden.

Mailprogramm.sendSimpleMailMessage(Neuenachricht, 0)

2.4.8 Wie werden Farben verwendet?

Farben werden innerhalb von StarBasic im ARGB-Format verwendet. Dieses Format hat als möglichen Parameter zu RGB noch den Alpha-Kanal.
Dieser kann aber ingnoriert werden.

Die Werte werden als Longinteger-Werte übergeben. Dazu steht in StarBasic dir RGB-Funktion zur Verfügung.

object.color=RGB (255,0,0)

Die RGB-Funktion berechnet den entsprechenden Longinterwert zu den RGB-Farben.

Alternative die Angabe im Hexdezimal-Format. Damit entspricht in der Farbgebung dem bekannten Format für HTML-Seiten.

Für ein helles Grün sieht dann der Code folgendermaßen aus:

object.color=&H00CC00

oder

object.color=&H0000CC00

Man muß daran denken das beim Auslesen von Farben der Integerwert nicht hexaldezimal sondern dezimal übergeben wird. Um diesen als Hexadezimalwert zusehen, muß man den Wert in einen String wandeln. Diese geht mit der HEX-Funktion. Näheres zu dieser Funktion steht in der Online- Hilfe.

myHexZahl = Hex(object.color)

Aber Achtung: Dem Hexwert muß noch "&H" vorangestellt werden.

Mit der Funktion clng wird aus einem Longinteger ein Hex-Wert.

myLongInt = clng("FFFFFF")

Zu der Farbumwandlung gibt es hier ein Datei mit einem Dialog zur Umrechnung der Farbwerte: farben.odt

2.4.9 Wie kann man die Ansicht-Zoom einstellen?

Leider wird die letzte Ansicht eines Dokumentes im Dokument mit abgespeichert. Mich persönlich nervt das vor allem bei Writer- Dokumenten.Man kann aber die Ansicht per Makro auslesen und einstellen.

Bei Calc, Impress und Draw-Dokumenten befindet sich der Parameter direkt unterhalb den Currentcontroller-Objektes:

ThisComponent.CurrentController.ZoomValue

Bei Writer-Dokumenten eine Stufe weiter unten (Wieder eines der Beispiele für die sonderbare Logik der API).

ThisComponent.CurrentController.ViewSettings.ZoomValue

Dieses Makro prüft beim Öffnen eines Dokumentes ob es sich um ein Writer-Dokument handelt und ändert

die Ansicht auf 100%.

```
Sub CheckZoom
  Dim oDoc as Object
  oDoc = ThisComponent
   If oDoc.supportsService("com.sun.star.text.TextDocument" ) then
     ThisComponent.CurrentController.ViewSettings.ZoomValue=100
  end if
end sub
```

Wenn man dieses Makro mit dem Öffnen-Befehl verknüpft, gibt es nie wieder.... (Extras->Anpassen > Ereignisse)

2.4.10 Wie kann man fortlaufende Rechnungsnummern erzeugen?

Nach dem die Frage öfter aufgetaucht ist hier eine mögliche Lösung:

Der Zähler für die Rechnungsnummer wir in eine Datei geschrieben. Diese wird mit einer Funktion ausgelesen und automatisch eine Nummer hochgesetzt.

Diese Funktion kann auch in Calc direkt eingesetzt werden (=getandsetnumber()).

```
function GetAndSetNumber as string
  dim f as Integer
  dim rechnungsdatei as string
  dim renummer as string
 rechnungsdatei="c:/re.txt"
  if FileExists("file:///"&rechnungsdatei) then
      f = FreeFile()
      Open "file:///"&rechnungsdatei for Input as #f
          Line Input #f, renummer
          close #f
          f = FreeFile()
          Open "file:///"&rechnungsdatei for output as #f
          Print #f, val(renummer)+1
          close #f
```

```
        else
            renummer=0
        endif
    GetAndSetNumber=renummer
end function
```

Für einen Mehrplatzbetrieb muß man eventuell noch eine Logdatei hinzufügen die den gleichzeitigen Zugriff verhindert.

2.4.11 Wie kann man andere Programme aus Starbasic aufrufen?

Über die Funktion Shell lassen sich externe Programme starten.

Shell(Pathname, Windowstyle, Param,bSync)

Pathname legt den Pfad sowie den Namen des aufzurufenden Programms fest. Windowstyle

definiert, in welchem Fenster das Programm gestartet wird. Möglich sind die Werte:

0 – Das Programm erhält den Fokus und wird in einem versteckten Fenster gestartet.

1 – Das Programm erhält den Fokus und wird in einem Fenster in Normalgröße gestartet.

2 – Das Programm erhält den Fokus und wird in einem minimierten Fenster gestartet.

3 – Das Programm erhält den Fokus und wird in einem maximierten Fenster gestartet.

4 – Das Programm wird in einem Fenster in Normalgröße gestartet, ohne jedoch den Fokus zu erhalten.

6 – Das Programm wird in einem minimierten Fenster gestartet, der Fokus bleibt beim aktuellen Fenster.

10 – Das Programm wird im Vollbild-Modus gestartet.

Der dritte Parameter Param gestattet es, Kommandozeilenparameter an das zu startende

Programm zu übergeben. Dieser Parameter ist optional.

Beispiel:

shell("D:\STOPEN\stopenst.exe",10,"")

oder

shell("D:\STOPEN\stopenst.exe",10)

Mit dem letzen Parameter, bSync, wird festgelegt wie sich das Makro nach dem Starten der Shell verhält.

Der Standardwert ist false. Dann arbeitet das Makro direkt weiter. Wird der Parameter auf true gesetzt wartet das Makro bis der Shellaufruf beendet worden ist.

2.4.12 Wie kann man den Autor einer Datei ändern?

Man kann bei den Eigenschaften einer Datei nicht den ersten Autor ändern. Man kann ihn löschen, aber keinen neuen einfügen. Mit einem Makro geht es aber doch. Der benötigte Service ist "Documentinfo".

```
sub neuerautor
  oDoc=thiscomponent
  nautor=inputbox("Bitte geben Sie den Autor ein","Neuer Autor")
  oDoc.DocumentInfo.Author=nautor
end sub
```

Damit wird der neue Autor für die Erstellung eingetragen.

2.4.13 Wie kann man die Dokumenteninformationen auslesen und ändern?

Die Dokumenteninformationen, die man unter Datei - Eigenschaften eintragen kann, können mit dem Service "DocumentInfo" gelesen und bearbeitet werden.
Der Service steht direkt unter dem Dokument-Service zur Verfügung.

```
   oDocument=thisComponent
   msgbox oDocument.DocumentInfo.Author
   msgbox oDocument.DocumentInfo.Description
   msgbox oDocument.DocumentInfo.Keywords
   msgbox oDocument.DocumentInfo.Theme
rem bzw. ab 2.0 Theme wurde gelöscht und durch subject ersetzt.
  msgbox oDocument.DocumentInfo.Sunject
   msgbox oDocument.DocumentInfo.Title
```

Ändern geht dann einfach mit:

```
  oDocument.DocumentInfo.Title="Neuer Titel"
```

2.4.14 Wie kann man Grafiken mit Makros verknüpfen?

Es gibt im Writer zwei Möglichkeiten, in Calc geht nur eine.

Grundsätzlich kann man im Writer unter den Eigenschaften (Format -> Grafik) einer Grafik in der Registerkarte "Makro" Verknüpfungen von der Grafik mit Makros erstellen. Dieser Weg geht nicht in Calc.

Der zweite Weg ist die Verwendung eines grafischen Kontrollfeldes. In diesem kann eine Grafik untergebracht werden und dann mit einen Ereignis verknüpft werden.

Dieser Weg geht auch in Calc.

2.4.15 Wie kann man auf die Windows-Registry zugreifen?

Man kann lesend auf die Windows-Registry mit den mitgelieferten Makros aus der Bibliothek "ImportWizard" zugreifen. Dort steht die Funktion "QueryValue" zur Verfügung.

QueryValue braucht folgende Parameter:

BaseKey, den Hauptschlüssel

sKeyName, den Schlüsselnamen

sValueName, den Eintrag

Folgende Konstanten sind für den jeweiligen Hauptschlüssel festgelegt:

Public Const HKEY_CLASSES_ROOT

Public Const HKEY_CURRENT_USER

Public Const HKEY_LOCAL_MACHINE

Public Const HKEY_USERS

Weitere Konstanten stehen im Quellcode von ImportWizard.

Aber Achtung: Wenn man das Makro das erstemal startet nach dem man OpenOffice gestartet hat kommt immer ein leerer String zurück.

Die Ursache für diesen Fehler habe ich leider noch nicht gefunden.

Das folgende Beispiel liest den Produktcode von Windows aus:

```
Sub Main
  BasicLibraries.LoadLibrary("ImportWizard")
'oder wenn Zugriff aus einem Dokument erfolgt: GlobalScope.BasicLibraries.LoadLibrary("ImportWizard")
  sTemplateKeyName = "Software\Microsoft\Windows\CurrentVersion"
```

sTemplateValueName = "ProductID"

sProductID = QueryValue(HKEY_LOCAL_MACHINE, sTemplateKeyName, sTemplateValueName)

msgbox sProductID

End Sub

2.4.16 Wie kann man den Dokumententyp heraus bekommen?

Der Dokumententyp steht als Supportedservices im Opjekt. Diesen kann man auslesen.

If oDocument.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then Calc-Dokument.

Jeder Dokumententyp unterstützt bestimmte Services, die für den Dokumententyp spezifisch sind. Die Unterstützung dieser Services muss man prüfen um den Dokumententyp zu erkenn. Die möglichen Services für die jeweiligen Dokumententypen lauten:

com.sun.star.sheet.SpreadsheetDocument -> Calc-Dokument

com.sun.star.text.TextDocument -> Writer-Dokument

com.sun.star.presentation.PresentationDocument -> Impress-Dokument

com.sun.star.formula.FormulaProperties > Formel

com.sun.star.drawing.DrawingDocument -> Draw-Dokument

com.sun.star.sdb.OfficeDatabaseDocument -> Base-Dokument

/* Ein Sonderfall ist ein Draw-Dokument. Ein Impress-Dokument unterstützt auch die Services einen Draw-Dokumentes. Um also ein Draw-Dokument zu identifizieren muß man noch die Unterstützung des Impress-Dokumentes prüfen.

com.sun.star.presentation.PresentationDocument=nein und com.sun.star.drawing.DrawingDokument -> Draw-Dokument.

Dieses Verhalten wurde in einem Patch umgestellt. Jetzt unterstützt Impress nicht mehr com.sun.star.drawing.DrawingDokument sondern ein GenericDrawingDokument

*/

2.4.17 Wie kann man per Makro die Standardpfade ändern?

Die Standardpfade die man unter Extras - Optionen - Pfade einstellen kann, können auch per Makro ausgelesen und gesetzt werden.
Dies erfolgt mit dem Service c.s.s.util.PathSettings.

Die Properties dieses Services repräsentieren die einzelnen Pfade.

Die Wichtigsten:

work -> Arbeitsverzeichnis

basic -> Die Suchpfade für Basicmodule

template -> Vorlagenverzeichnisse

UserConfig -> Benutzer Konfiguration.

Die Restlichen sind in der Referenz zu finden.

Um zum Beispiel das Arbeitsverzeichnis zu ändern, verwendet man folgendes Makro:

```
Sub Arbeitsverzeichnis
Path=createunoservice("com.sun.star.util.PathSettings")
path.work="file:///d:"
End Sub
```

2.4.18 Wie kann man Makros in Symbolleisten verwenden?

Man kann Makros auch in Symbolleisten einfügen. Hierbei unterscheidet StarOffice zwischen den Symbolleisten der einzelnen Anwendungen.

Mit der Maus auf der Symbolleiste und einem Klick mit der rechten Maustaste öffnet ein Menü in dem man Symbole bearbeiten kann. In dem neuen Fenster kann man rechts auch Makros auswählen und diese in die Symbolleiste einfügen.

Man kann auch eigene Symbolleisten erzeugen und diese mit Makros bestücken.

Eigene Symbolleisten erzeugt man im "Anpassen" - Fenster, Register "Symbolleisten". (Menü Extras -> Anpassen). Diesen neuen Leisten kann man einen neuen Namen geben und findet diese bei "Symbole bearbeiten" in der Liste der Symbolleisten wieder.

2.4.19 Wie kann man ein Makro mit einem Ereignis oder der Tastatur verbinden?

Makros können mit bestimmten Ereignissen innerhalb von StarOffice verknüpft werden. Diese Verknüpfung kann mit StarOffice selbst oder mit einem Dokument erfolgen.

Im Menü "Extras - Anpassen - Register Ereignisse" kann man ein Makro mit vorgebenen Ereignissen verknüpfen. Oben Rechts legt man dabei fest ob eine Verknüpfung innerhalb von StarOffice oder innerhalb eines Dokumentes eingerichtet werden soll. Makros die mit Dokumenten laufen sollen, sollten im Dokument gespeichert werden.

Im Register "Tastatur" kann man Makros auch mit Tastaturbefehlen verknüpfen.

Unten links kann man das Modul auswählen. Im nächsten Fenster dann das Makro. Oben hat man die Auswahl der Tastaturbefehle. Oben rechts kann man wieder auswählen, ob die Verknüpfung in StarOffice gelten soll oder nur in einer Anwendung (Writer, Calc, Impress, Draw, Formel).

2.4.20 Kann man ein Makro über die Kommandozeile mitstarten?

Der Parameter für den Aufruf eines Makros lautet:

soffice.exe macro:///bibliothek.modul.macro

Beispiel:

soffice.exe macro:///standard.module1.macro1

Aber Achtung! Wenn das Makro nichts innerhalb eines Dokumentes macht oder öffnet, wird das Makro ausgeführt und StarOffice wieder geschlossen.

Parameter können wie bei einem normalen Makroaufruf übergeben werden: soffice.exe macro:///standard.module1.macro1(para1,para2).

Diese Parameter müssen natürlich im Aufruf des Makros vorhanden sein.

sub macro1(para1 as string ,para2 as string)

Um zusätzlich eine Datei zu öffnen muss man diese dazwischen setzen.

soffice.exe c:\datei macro:///standard.module1.macro1

Ein Makro aus der Datei zu starten die aufgerufen wird geht nicht. Das sollte man dann dem Ereigniss "Dokument öffnen" verknüpfen.

2.4.21 Kann man ein Makro automatisch beim Programmstart starten?

Unter Extras - Anpassen kann man Makros bestimmten Ereignissen zuordnen:

Unterschieden wird hierbei ob die Ereignisse dem Dokument oder StarOffice zugeordnet werden sollen. Ist der Checkbutton "StarOffice" gewählt, beziehen sich die Ereignisse auf StarOffice. Dann wird zum Beispiel beim Öffnen eines Dokumentes ein Makro ausgeführt.

Wird die Checkbox "Dokument" gewählt, bezieht sich das Ereignis auf das aktuelle Dokument. Damit kann man zum Beispiel ein Makro automatisch mit dem Dokument starten.

2.4.22 Wie kann ich den Start eines Makros mit einem Dokument veranlassen?

Unter Extras - Anpassen kann man Makros bestimmten Ereignissen zu ordnen:

Wird die Checkbox "Dokument" gewählt, bezieht sich das Ereignis auf das aktuelle Dokument. Damit kann man zum Beispiel ein Makro automatisch mit dem Dokument starten.

Siehe auch: Kann man ein Makro automatisch beim Programmstart starten?

2.4.23 Wie kann ich Module und Bibliotheken kopieren und verschieben?

Innerhalb des Makrodialoges gibt es eine Schaltfläche "Verwalten".

Im folgenden Dialog kann man Module zwischen Bibliothek verschieben oder kopieren. Verschieben mit Drag und Drop. Kopieren erfolgt wenn bevor man die Maus los läßt die STRG- Taste drückt.

Auf der zweiten Registerkarte kann man Bibliotheken erzeugen, löschen, erstellen und hinzufügen. Letzteres ermöglicht das kopieren von Bibliotheken aus anderen Dateien.

2.4.24 Wie kann man Makros mit Tastaturbefehlen verknüpfen?

Über den Dialog "Anpassen". Dieser steht im Menü Extras.
Er hat ein Register "Tastatur". In diesem kann man oben in der Liste die Tastenkombination auswählen. Links unten wählt man das Basicmodul aus, dessen Makros dann in der Mitte erscheinen. Um dieses mit dem Tastaturbefehl zu verknüpfen muss dann oben "Ändern" gedrückt werden.

2.4.25 Wie kann man die Aktualisierung des Bildschrims abstellen?

Manchmal möchte man nicht das sich der Bildschirm während des Makros aktualisiert. Das kann auch den Ablauf des Makros beschleunigen.
Das geht mit lockcontrollers() innerhalb des Dokumentes.

odoc = thiscomponent

odoc.lockcontrollers

Die Aktualisierung wird dann mit unlockconrollers wieder aktiviert.

odoc = thiscomponent

odoc.unlockcontrollers

2.4.26 Wie kann man ein ini-Datei auslesen und schreiben?

Innerhalb von Starbasic gbt es dafür keine Funktion. In dem Tools von mir gibt es zwei Funktionen zum Auslesen von ini-Dateien.
ReadIni und WriteIni

2.4.27 Wie kann man die Dateien in einem Ordner und den Unterverzeichnissen auslesen?

Mit den Dir() geht es nicht! Diese Funktion liest nur die Dateien und Ordner in dem ausgewählten Ordner ein, nicht die Inhalte der Unterordner.
Ich habe eine Funktion erstellt die alle Unterordner über eine Rekursion ausliest.

Der Aufruf erfolgt mit

Dim Liste(10000) as string

erg=getDirs(liste(),0,"C:\")

Als Ergebnis stehen alle Dateien und Ordner in liste() und der Rückgabewert entspricht der Anzahl der Einträge.

```
function getdirs( liste(),z, folder) as integer
  sFolderUrl = ConvertToUrl( Folder )
  oSimpleFileAccess = createUnoService( "com.sun.star.ucb.SimpleFileAccess" )
  aFolders = oSimpleFileAccess.getFolderContents( sFolderUrl,true )
  For i = LBound( aFolders ) To UBound( aFolders )
    sFile = aFolders( i )
    If oSimpleFileAccess.isFolder( sFile ) Then
      getdirs( liste(),z, sFile)
    Else
      liste(z)=sfile
      z=z+1
    end if
```

```
  next i
  getdirs=z
end function
```

2.4.28 Wie kann man ein bestimmtes geöffnetes Dokument auswählen?

Man kann geöffnete Dokumente durch ihre URL anwählen. Voraussetzung dafür ist das das Dokument schon gespeichert und ist dadurch eine URL hat.
Nicht gespeicherte Dokumente haben eine leere URL.

Der Weg geht dann über die Enumeration des StarDesktops. Diese Enumeration enthält alle geöffneten Fenster der Anwendung (auch die IDE).

Danach muss man nur noch diese URL mit der gewünschten verglichen werden. Dazu muß man aus der URL den eigentlichen Dateinamen herauslesen.

Dies geht mit FileNameoutofPath aus der Bibliothek Tools.

```
GesuchteDatei="test.odt"
Dim oDesktop As Object, oDocs As Object
Dim oDoc As Object, oComponents As Object
oComponents = StarDesktop.getComponents()
oDocs = oComponents.createEnumeration()
Do While oDocs.hasMoreElements()
  oDoc = oDocs.nextElement()
  datei=odoc.geturl()
  FileN=FileNameoutofPath(datei)
  if FileN=GesuchteDatei then ....myFebster=odoc
Loop
```

In den Tools von mir gibt es eine fertige Funktion dazu von mir: Fensterwaehlen.

2.4.29 Wie kann man Bibliotheken von OO in ein Dokument kopieren?

Worum geht es? Bibliotheken kopiert man innerhalb der Verwaltung bei den Makros. Eine neue Bibliothek kann man mit "Hinzufügen" direkt aus einem anderen Dokument kopieren. Aber man kann nicht sofort eine Bibliothek aus OpenOffice selber in ein Dokument auf diese Weise kopieren.
Dazu muß man auch Hinzufügen wählen. Dann geht man auf das Verzeichnis in dem die User- Daten stehen. Bei Windows ist es in der Regel der Windows-User Pfad mit den Anwendungsdaten. Dort stehen im Basic-Verzeichnis die einzelnen Bibliothek. Diese kann man nun der Datei hinzufügen. Dabei reicht es wenn man ein der beiden Dateien auswählt (Makros oder Dialoge).

2.4.30 Wie kann man sich die sichtbaren Symbolleisten anzeigen lassen?

Man kann sich mit einer Schleife alles sichtbaren Symbolleisten anzeigenlassen.

```
sub mySymbolleisten
layout = ThisComponent.CurrentController.Frame.LayoutManager
elem=layout.getElements()
for i=1  to ubound(elem())
  msgbox  elem(i).ResourceURL
next i
End Sub
```

2.4.31 Wie kann man Bibliotheken erzeugen und löschen?

Man kann innerhalb von OpenOffice oder einzelnen Dokumenten auch selber Bibliotheken und darunter Module mit eigenem Inhalt erzeugen.
Grundsätzlich muß man dazu sich erstmal den Service BasicLibraries holen.

Für OpenOffice allgemein:

BasicLibs = Globalscope.BasicLibraries

Für ein Dokument

BasicLibs = ThisComponent.BasicLibraries

Innerhalb dieses Services stehen für Bibliotheken unter anderem die Funktionen:

hasbyname, createLibary, removeLibary und renameLibrary zur Verfügung.

Mit hasbyname kann man prüfen ob eine Bibliothek schon existiert.

if BasicLibs.hasbyname("NewLib") then ....

Mit CreateLibrary wird eine neue Bibliothek erzeugt.

BasicLibs.createLibrary("NewLib")

oder um gleich das Objekt zu erhalten

NewLibrary = BasicLibs.createLibrary("NewLib")

RemoveLibray löscht die Bibliothek

BasicLibs.removeLibrary("NewLib")

und mit renameLibrary kann sollte man eine Bibliothek umbennen können. Nein der Aufruf

BasicLibs.renameLibrary("NewLib", "Testlib")

löscht die Bibliothek "NewLib"!

2.4.32 Wie kann man Module erzeugen und löschen?

Wenn man sich eine Bibliothek erzeugt hat, oder mit getByname aufgerufen hat, kann man Module einfügen oder löschen.
Man sich den Service der Bibliothek holen und kann dann auf die nötigen Funktionen zugreifen.

BasicLibs = Globalscope.BasicLibraries

Lib = BasicLibs.Getbyname("NewLib")

Grundsätzlich muß man sich erst einen Inhalt für das Modul als reinen Text erstellen oder aus einer Datei auslesen. Hier ein kleines Makro als Beispieltext

moduleText = "REM Test" + CHR(13) + "Sub TestMsg" + CHR(13) + "msgbox ""Hallo Welt"" " + CHR(13) + "End Sub"

Dieser Text wird dann mit insertbyname und dem gewünschten Modulnamen eingefügt.

```
BasicLibs = Globalscope.BasicLibraries
Lib = BasicLibs.Getbyname("NewLib")
moduleText = "REM Test" + CHR(13) + "Sub TestMsg" + CHR(13) + "msgbox ""Hallo Welt"" " + CHR(13) + "End Sub"
Lib.insertByName("Modul1",moduletext)
```

Mit removebyname kann man das Modul wieder löschen.

Lib.removebyname("Modul1")

2.4.33 Wie kann man die installierte Sprache feststellen?

Man die Sprache direkt mit dem Service com.sun.star.configuration.ConfigurationProvider auslesen.
Grundsätzlich ist dieser Service dazu gedacht auf weitere Parameter in der Konfiguration zu zugreifen.

```
Dim oParameter(0) As New com.sun.star.beans.PropertyValue
sProvider = "com.sun.star.configuration.ConfigurationProvider"
oConfigProvider = createUnoService(sProvider)
OSprachversion=oConfigProvider.Locale.Language
msgbox OSprachversion
```

2.4.34 Wie kann man ein Dokument schützen, bzw den Schutz aufheben?

Ein Dokument hat folgeende drei Methoden die sich um den Schutz des Dokuemtes drehen:

isprotected -> Gibt den Schutzstatus zurück. True wenn es geschützt ist, false wenn es nicht geschützt ist.
protect("passwaord") -> Schützt das Dokument
unprotect ("passwaord") -> Hebt den Schutz auf.

```
odoc = thiscomponent
statuss=odoc.isProtected
if status=true then
   odoc.unprotect("password")
  else
   odoc.protect("password")
end if
```

2.4.35 Wie kann man das verwendete Betriebssystem auslesen?

Dies geht mit dem Befehl GetGUIType(). Dieser gibt drei Werte zurück:

1 = Windows

3 = MacOS

4 = Unix

```
Sub ShowBS
OS = GetGUIType()
if OS = 1 then Msgbox "BS ist Windows"
if OS = 3 then Msgbox "BS ist Mac"
if OS = 4 then Msgbox "BS ist Unix"
End Sub
```

2.4.36 Wie kann mein Dokument ausblenden?

Man kann ein Dokument ausblenden mit der Eigenschaft "Visible" des ContainerWindow.

```
odok=ThisComponent
oDokView = oDok.CurrentController.Frame.ContainerWindow
oDokView.Visible = FALSE
```

Einblenden geht dann mit True.

```
oDokView.Visible = TRUE
```

2.4.37 Wie kann man sich alle Fonts in einen Text ausgeben lassen?

Dies geht mit einem kleinen Makro und dem Service FontDescriptor.

```
Sub ShowallFonts

Dim oWindow As Object
Dim aFontArray()
Dim oFontDescriptor As Object
Dim sFontList As String
Dim sTemp As String
Dim I As Integer

  oDoc = ThisComponent
  otext=odoc.text
  ocursor=otext.createtextcursor()
  mytext="abcdefghirklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
  Set oWindow = ThisComponent.CurrentController.Frame.ComponentWindow

  aFontArray() = oWindow.getFontDescriptors
  For I = 0 to UBound(aFontArray)
    oFontDescriptor = aFontArray(I)
```

```
    If oFontDescriptor.Name <> sTemp Then
      sFontList = sFontList & oFontDescriptor.Name & chr(10)
    End If

  if oFontDescriptor.Stylename="Standard" then
  ocursor.CharFontName=oFontDescriptor.Name
  ocursor.String=oFontDescriptor.Name+" --- "+mytext
  otext.insertControlCharacter(ocursor, "com.sun.star.ControlCharacter.PARAGRAPH_BREAK",false)
  ocursor.gotoEndofParagraph(false)

  end if


  Next I


End Sub
```

2.4.38 Wie kann man die installierte Version feststellen?

Die Information steht in den Setupdaten.
Mit einer Funktion von Laurent Godard kann man diese auslesen.

```
Sub ShowVersionOfOOo
  m=OOoVersion()
  msgbox m
End Sub


Function OOoVersion() As String
  'Retreives the running OOO version
  'Author : Laurent Godard
  'e-mail : listes.godard@laposte.net
  Dim aSettings, aConfigProvider
  Dim aParams2(0) As new com.sun.star.beans.PropertyValue
  Dim sProvider$, sAccess$
  sProvider = "com.sun.star.configuration.ConfigurationProvider"
  sAccess   = "com.sun.star.configuration.ConfigurationAccess"
  aConfigProvider = createUnoService(sProvider)
  aParams2(0).Name = "nodepath"
  aParams2(0).Value = "/org.openoffice.Setup/Product"
  aSettings = aConfigProvider.createInstanceWithArguments(sAccess, aParams2())
  OOOVersion=aSettings.getByName("ooSetupVersion")
End Function
```

3 Variablen

3.1 Warum haben manche Variablen anscheinend nicht den richtigen Wert?

Bei der Varibalendeklaration mit DIM muß darauf geachtet werden, dass man für jede Variable den Typ

einzeln deklarieren muß.

Dim i1, i2 as integer

wird als i1 = variant und i2 = integer interpretiert. Das ist auf den ersten Blick nicht schlimm, kann aber bei Programmierfehlern die Fehlersuche erschweren.

Nach dem obigen Beispiel tritt dann folgendes auf:

i2 = 1 wirklicher Wert: 1

i1 = 4 wirklicher Wert: 4

Alles OK

i2 = 1 wirklicher Wert: 1

i1 = 1.33 wirklicher Wert: 1.33

Auch OK aber i1 ist kein Integer!

Wird i1 richtig als Integer deklariert kommt bei 1.33 als Wert 1 heraus.

3.2 Wie kann man die Größe eines unbekannten Arrays auslesen?

Es gibt Methoden die ein Array zurückgeben. Von diesem Array ist dann die Anzahl der Elemente unbekannt. Um diese Anzahl auslesen gibt es die Funktion UBound.

Anzahl = UBound(array())

Dabei daran denken, dass die wirkliche Anzahl der Elemente um eins größer ist. Denn das erste Element ist 0.

Das Gegenstück zu ubound ist lbound. Damit wird der unterste Werte des Arrays bestimmt.

Bei mehrdimensionalen Feldern wird die Dimension bei der Abfrage mit übergeben.

Dim TestArray (2,8)

TestAnzahl = uBound(Testarray(),2)

3.3 Wie kann man in einer Stringvariablen das Hochkomma " verwenden?

In dem man innerhalb des Stringes das Hochkomma doppelt setzt:

MyText = "dies ist ein Test "" mit dem Hochkomma"

ergibt dann:

Dies ist ein Text " mit dem Hochkomma

MyText = "dies ist ein Test """" mit dem Hochkomma"

ergibt dann:

Dies ist ein Text "" mit dem Hochkomma

3.4 Wie geht man mit dem neuen Typ "Struktur" um?

Mit dem Typ "Struktur" kann man Datengruppierungen bilden. Eigene Objekte mit Daten. Diese werden zur Zeit weder in der Online-Hilfe noch im Basic-Handbuch beschrieben.

Eine Datenstruktur wird als Rahmen im Kopf einer Makrobibliothek beschrieben. Die Einleitung erfolgt mit Type .. und schließt mit End Type. Dazwischen folgen die Variablenbezeichungen.

```
Type Adresse
  sName as string
  sVorname as string
  sPlz as string
  sOrt as string
  sTelefon as string
end Type
```

Eine Verwendung von Arrays innerhalb einer solchen Struktur ist nicht möglich.

Innerhalb des Makros wird die Struktur als neues Objekt aufgerufen. Danach können Werte zugewiesen werden.

```
mydata=createObject("Adresse")
with mydata
   .sname="FFF"
   .svorname="ggg"
end with
msgbox mydata.sname
```

Man kann diese Struktur auch an andere Prozeduren oder Funktioen übergeben. Dabei muß die Variable für die Struktur als Objekt deklariert werden

```
sub zeigewert( data as object)
   msgbox data.svorname
end sub
```

Oder als Funktion:

```
Function NeuerWert (data as object) as object
  data.svorname="XXX"
  NeuerWert=Data
end Function
```

3.5 Wie werden Variablen deklariert?

In Starbasic gibt es zwei Arten Variablen zu deklarieren. Implizit und Explizit.

Bei der impliziten Deklaration wird die Variable automatisch mit ihrer ersten Verwendung deklariert. Das macht das Programmieren auf den ersten Blick einfach, weil man einfach mit der Zeile

Eingabe = 2

die Variable Éingabe deklariert. Der Nachteil ist aber das man später bei Tippfehler zum Beispiel auch sofort eine neue Variable "Engabe" generiert. Der Interpreter erzeugt zusätzlich eine neue Variable und die existiert neben der ersten Variable. Man hat also versehentlicvh zwei Variablen. Dies kann zu langwierigen Fehlersuche führen.

Dieses Risko kann man mit der expliziten Deklaration umgehen. Durch den Befehl **Option Explicit** am Anfang des Moduls wird der Interpreter angewiesen nur explizit deklarierte Variablen zu aktzeptieren.

Eine explizite Deklaration der Variablen erfolgt durch den Befehl "Dim".

Dim MyVariable as Integer

Integer ist ein Variablentyp zu dem wir gleich kommen. Wenn jetzt im Programm versehentlichen irgendwo "MVariable" steht, erzeugt der Interpreter eine Fehlermeldung.

Es ist Starbaisc dabei egal an welcher Stelle die Variable deklariert wird. Man kann sie am Anfang der Prozedur alle gemeinsam oder erst in der Zeile vor der ersten Verwendung einzeln deklarien. Guter Programmierstil ist die Variablen am Anfang festzulegen, da damit die Übersichtlichkeit über den Code besser ist. Einige andere Programmiersprachen erzwingen sogar solch ein Vorgehen.

Noch eine wichtige Bemerkung zu Variablendeklaration in Starbasic. Jede Variable erhält ohne explezite Deklaration automatisch den Typ Variant. Das ist soweit eigentlich kein Problem, aber Starbasic interpretiert **jede** Variable ohne Typenbezeichnung als Variant. Eine Aufzählung von Variablen mit anschließender Deklaration des Typen, wie in anderen Programmiersprachen üblich geht daher nicht.

Die Zeile:

Dim i1,i2 as Integer,

intepretiert Starbasic als i1 = Variant und i2 = integer.

Das ist auf den ersten Blick nicht schlimm, wenn man aber den Variablen Werte zuweist kann es zu Problemen kommen.

Im obigen Fall kann es dann passieren das die Werte falsch interpretiert werden. Dies kann besonders bei der Verwechslung von Integer- und Double-Variablen passieren. Wenn man den Wert einer Double Variablen einer Integer-Variablen zuweist. Dabei wird die Zahl nämlich zu einer Integer-Zahl gerundet.

Um beide wirklich als Integer zu deklarieren muß man diese einzeln deklarieren.

Dim i1 as integer,i2 as Integer.

Dazu ein Beispiel:

i1= 3

i2= 2.5

Ergebnis=i1*i2

Ohne jede Deklaration ist das Ergebnis 7,5,

Dim i1,i2,i3 as integer -> führt zu dem Ergebnis 8,

Dim i1,i2 as integer, i3 as integer -> führt zu dem Ergebnis 9,

ebenso führt Dim i1 as integer, i2 as integer, i3 as integer zu dem Ergebnis 9.

Ich denke die möglichen Probleme mit falschen Deklarationen sind daraus nachvollziehbar.

Tipp: Jede Variable in einer eigenen Zeile dimensionieren.

3.6 Welche Gültigkeit haben Variablen?

Eine Variable in Starbasic hat eine bestimmte Gültigkeit. Es gibt globale, öffentliche (public), private und "normale" Variablen. Durch diese Unterscheidung wird die Gültigkeit der Variablen für den Interpreter festgelegt.
Erst mal eine Kurzfassung:

Globale Variablen: Sie gelten innerhalb der gesamten Statoffice-Sitzung in allen geladenen Bibliothken, Modulen, Prozeduren und Funktionen.

Public Variablen: Sie gelten innerhalb aller Module einer Bibliothek

Private Variablen: Sie gelten nur innerhalb eines Moduls. Diese Variablennamen können daher auch in mehreren Modulen parellel verwendet werden.

Locale Variablen: Sie gelten nur innerhalb einer Funktion oder Procedure

Jetzt die lange Fassung:

Lokale Variablen werden innerhalb einer Prozedur oder Funktiondeklariert und wird nur innerhalb dieser auch verwendet.

```
Sub MyProzedur
  Dim MyVariable as Integer
end Sub
```

Lokale Variablen haben nur Gültigkeit solange der Interpreter innerhalb der Prozedur oder der Funktion ist. Sobald diese verlassen worden ist stehen die zugewiesen Werte und die Variable nicht mehr zur Verfügung. Auch bei einem neuen Aufruf der Prozedur oder Funktion sind die Variablen nicht mehr belegt. Hierfür gibt es die Ausnahme wenn die Variable als Static deklariert.

```
Sub MyProzedur
Static MyVariable as Integer
end Sub
```

Eine als Static deklarierte Variable behält ihren Wert bis zum nächsten Aufruf der Prozedur bei.

Öffentliche Variablen werden im Kopfbereich eines Moduls deklariert und gelten dann innerhalb aller Prozeduren und Funktionen dieses Moduls.

Alternativ zu dem Befehl "Dim" kann hierfür auch der Befehl "Public" verwendet werden. Wichtig ist aber die Deklaration im Kopfbereich.

```
Dim MyVariable as Integer


Sub MyProzedur
MyVariable = 1
end Sub
```

Diese öffentlichen Variablen haben solange Gültigkeit wie das Modul ausgeführt wird. Hier unterscheidet sich die öffentliche Variable von der globalen Variablen.

Globale Variablen werden ebenfalls im Kopfbereich eines Moduls deklariert, sie haben auch nach beenden des Moduls Gültigkeit.

Erzeugt werden globale Variablen mit dem Befehl "Global"

```
Global MyVariable as Integer
```

Zu guter Letzt kommen noch die privaten Variablen .

```
Private MyVariable as Integer
```

erzeugt eine private Variable. Diese hat nur innerhalb des Moduls Gültigkeit. Wenn in einem zweites Modul ebenfalls eine Variable mit gleichem Namen als privat deklariert wird, erzeugt der Interpretor beim Ausführen auch eine zweite Variable. Private Variablen sollte man einsetzen wenn man Werte wirklich nur innerhalb des Moduls verwenden will und in seinem Code nicht mit öffentlichen Variablen durcheinander kommen will.

3.7 Welche Variablentypen gibt es?

Zeichenfolgen

Zeichenfolgen sind Variablen die Textzeichen beinhalten. Zeichenfolgen, im englischen "Strings" bezeichnet, werden für die Verarbeitung von Text verwendet.

Man kann Inhalte vergleichen, löschen oder kürzen. In Starbasic heißen sie String-Variablen. Um eine Variable als String zu deklarieren verwendet man folgende Zeilen

```
Dim MyText as String
```

oder

```
Dim MyText$
```

Innerhalb einer solchen Variable können sie Zeichenfolgen mit einer maximal Länge von bis zu 65535 Zeichen speichern. Also eigentlich ein ganze Menge.

Um einer String-Variablen einen Wert zu zuordnen muß man den gewünschten Text in Hochkammas einklammern:

MyText = "Dies ist ein Text"

Wenn der Text länger wie eine Zeile wird, muß man in mehrere Zeilen schreiben und jede Zeile mit einen Hochkomma beginnen und beenden, und die einzelnen Zeilen mit einem Plus-Zeichen (+) und dem Verbindungszeichen für Programmzeilen, dem Unterstrich (_), verbinden.

MyText = "Dies ist ein Text um einmal die Länge einer Zeile aus " + _

"zu probieren!"

Wenn man ein Abführungszeichen in seiner Zeichenfolge benötigt wird, muß dieses doppelt eingefügt werden:

MyText = "Dies ist ein Test mit dem "" Zeichen" ergibt dann

Dies ist ein Test mit dem " Zeichen.

Zahlen

Zahlen sind natürlich für fast jedes Programm wichtig. Innerhalb von StarBasic gibt es verschiedene Typen von Zahlenvariablen. Eine der wichtigsten Unterscheidung, in allen Programmiersprachen, ist die Trennung in ganzzahlige Variablentypen und Fließkommatypen. Grob entpricht das der mathematischen Einteilung in ganze und rationale Zahlen.

Es gibt in Starbasic die folgenden Zahlentypen:

Integer

Long Integer

Single

Double

Currency

Integer

Integer Variablen können nur ganze Zahlen zwischen -32768 und 32767 aufnehmen. Fließkommazahlen

werden bei einer Zuweisung automatisch kaufmännisch gerundet bzw. abgerundet.

Die Deklarion erfolgt mit

Dim MyInt as Integer

oder

Dim MyInt%

Long Integer

Integer Variablen können nur ganze Zahlen zwischen -2.147.483.648 und 2.147.483.647 aufnehmen. Fließkommazahlen werden bei einer Zuweisung automatisch kaufmännisch gerundet bzw. abgerundet.

Die Deklarion erfolgt mit

Dim MyLongInt as Long

oder

Dim MyLongInt&

Single

Single-Variablen sind Fließkommazahlen. Sie können positive und negative Werte halten

zwischen 3,402823 * 10 38 und 1,401298 * 10-45.Ursprünglich wurden Single-Variablen konzipiert, um den Rechner bei der

Ausführung mathematischer Operationen nicht so stark wie beim Umgang mit den

genaueren Double-Variablen zu belasten.

Die Deklartion erfolgt über

Dim MySingle as Single

oder

Dim MySingle!

Double

Double-Variablen sind Fließkommazahlen. Sie können positive und negative Werte

zwischen 1,79769313486263 * 10 308 und 4,94065645841247 * 10 -324 aufnehmen. Double-

Variablen sind geeignet für genaue Berechnungen

Die Deklartion erfolgt über

Dim MyDouble as Double

oder

Dim MyDouble#

Currency

Währungsvariablen (Currency-Variablen) sind wie Integer-, Long Integer-, Single- und

Double-Variablen für die Aufnahme von Zahlen gedacht. Die Position des Kommas ist dabei Curruncy festfix, weshalb sie auch keine

Fließkommavariablen sind. Sie haben immer genau vier vier Nachkommastellen.

Vor dem Komma können maximal 15 Ziffern stehen. Daher können Currency Variablen zahlen zwischen

-922.337.203.685.466,5808 und -922.337.203.685.466,5807 enthalten. Diese ungenaue Begrenzung, und nicht -999.999.999.999.999.,9999 bis 999.999.999.999.999,9999 liegt an dem internen Speicherformat in 8 Byte. **????? Anhang ????**

Währungsvariablen sind insbesondere für kaufmännische Kalkulationen gedacht, bei

denen durch die Verwendung von Fließkomma-Zahlen unvorhersehbare Rundungsfehler

entstehen können.

Deklaration einer Currency-Variablen:

Dim MyCurrency as Currency

oder

Dim MyCurrency@

Verwendung von Zahlen im Programmcode

Für die Verwendung von Zahl im Programmcode gelten einige kleine Regeln:

Ganze Zahlen werde so verwendet wie man es sich vorstellt:

A=1,

B=34,

C= -56

Fließkommazahlen müssen als Kommazeichen einen Punkt haben:

D = 12.67

E = 1345.948484

Zahlen können auch in er Expontailschreibweise verwendet werden. Dies ermöglicht

eine einfache Darstellung von sehr großen und sehr kleinen Zahlen. Hier bei gilt dann

Zwischen der Zahl und dem Exponenten darf kein Leerzeichen und der Exponent muß eine ganze Zahl sein.

F = 1.43E10

G = 2.56E-10

H = -3.22E5

Zahlen als Hexadezimalzahl können durch eine vorgestelltes "&H" zugewiesen werden. Das Hexadezimalsystem (16er System) hat den Vorteil, dass zwei Ziffern jeweils exakt einem Byte entsprechen, was einen maschinennahen Umgang mit Zahlen gestattet. Als Ziffern kommen beim Hexadezimalsystem die Ziffern 0 bis 9 sowie die Buchstaben A bis F zum Einsatz. Ein A steht für die Dezimalzahl 10, ein F für die Dezimalzahl 15. StarOffice Basic gestattet die Eingabe ganzzahliger Hexadezimalwerte.

I = &HFF ' FF steht 255 Decimal in Hexadecimal

Zahlen als Oktalzahl (8er-System können mit einem vorgestellten "&O" zugewiesen werden. Diese Zahlenformat wird jedoch sehr selten benötigt.

J= &O377 ' Entspricht Decimal 255.

Logisch

Eine Variable vom Typ Boolean kann nur zwei Werte annehmen Wahr oder Falsch (true oder false). Mit dieser Variable kann man einen Zustand feststellen oder setzen, die nur zwei mögliche Zustände hat. Intern wird dabei mit 0 (für "Falsch") und 1 (für "Wahr") gearbeitet. Im allgemeinen Sprachgebrauch der Programmiersprachen handelt es sich eine boolsche Variable. Die beste deutsche Bezeichnung ist dafür logische Variable. Die Bezeichnung Boolean kommt von dem englischen Mathematikers George Boole. Er hat die nach ihm benannte Bool'sche Algebra entwickelt. Die logischen Operatoren, deren Bescheibung noch folgt, basieren auf der Bool'schen Algebra.

Die Deklaration erfolgt mit

Dim bWahr as Boolean

Setzen und Abfragen einen boolschen Variabe kann über True oder False und "1" oder "0" erfolgen

"bBool = 1" hat die gleiche Wirkung wie "byBool=True".

Datum/Zeit

Date-Variablen können Datums- und Zeitwerte enthalten. Bei der Speicherung von

Datumswerten verwendet StarBasic ein internes Format, das Vergleiche und mathematische

Operationen erlaubt.

Dim MyDate as Date

Datenfelder / Arrays

Eindimensionale Datenfelder

Starbasic unterstützt zusätzlich zu den einfachen Variablen auch Datenfelder. Datenfelder (arrays) können mehrer Variablen eines Type enthalten. Diese werden über den Index des Datenfeldes angesprochen. Mit Datenfeldern kann man zum Beispiel Gruppen von Variablen zusammen fassen.

Einfache Arrays werden mit der Zusatzangabe über die Anzahl der Elemente deklariert:

Dim MyArray(3)

oder

Dim MyArray(3) as Typ

Auch hier wird wieder unterschieden zwischen einer Deklartion im Typ Variant und einer spezifischen Typendeklaration. Hier gilt auch wieder Vorsicht mit dem Typen Variant! Man kann den einzelnen Daten verschiedene Typen zuordnen. In dem Beispiel wird ein Array von vier Elementen erzeugt. Warum es vier Elemente und drei erklären wir gleich.

Der Index eines Datenfeldes beginnt standardmäßig bei Null. Das heißt die eigentlichen Daten lassen sich über den Index von 0 bis Deklaration - 1 ansprechen.

MyArray(0) = 1

MyArray(1) = 1

MyArray(2) = 1

MyArray(3) = 1

Man kann den Index auch selber festlegen in dem man den Bereich einträgt. So kann man mit der Dekalaration

Dim MyArray(1 to 4) as String

ebenfalls einen Array mit vier Elementen festlegen. Dieses wird dann über die Indices 1 bis 4 angesprochen. Diesen Bereisch kann man auch negative festlegen.

Dim MyArray(-3 to -10) as Integer

legt eine Array mit den entsprechenden Indices zwischen -3 und -10 fest.

Wenn man sich die möglichen Bereiche mit der Funktion x to y anschaut wird klar, warum wir oben vier Elemente erhalten haben. Starbasic interpretiert die vorhandene Zahl als y und setzt dann null als Startwert.

Für Datenfelder und Indices gelten folgende Regeln:

Der kleinste erlaubte Index ist -32768.

Der größte erlaubte Index ist 32767.

Die maximale Anzahl an Elementen beträgt 16368. Also nicht die naheliegende Differenz zwischen minimalen und maximalen Index (65535).

Zusätzlich besteht in Starbasic die Möglichkeit den Startindex für Datenfelder dauerhaft auf eins einstellen. Mit dem Befehl

Option Base 1

oberhalb aller Module erfolgt diese Zuweisung. Diese hat jedoch einen Haken. Durch diese Anweisung wird nur der Index eines Datenfeldes auf eins gesetzt. Die Anzahl der Elemente bleibt aber erhalten. In unserem Beispiel von oben Dim MyArray(3) erhält man also wieder vier Elemente, diesmal mit den Indices eins bis vier.

Mehrdimensionale Datenfelder

Zusätzlich zu den eindimensionalen Datenfeldern ist es in Starbasic auch möglich mehrdimensionale Datenfelder anzulegen. Die betreffenden Dimensionen werden dabei durch Kommata getrennt.

Dim MyArray(3,3) as String

Mit einem solchen Array lassen sich dann insgesamt 16 String-Werte zuweisen die über die beiden Indices 0 bis 3 ansprechbar sind. Das beste Beispiel für einen solches Datenfeld ist eine Tabelle. Jedes Tabellenfeld läßt sich genau über zwei Indices ansprechen: Zeile und Spalte.

Theoretisch lassen sich in Starbasic Datenfelder mit mehreren hundert Dimensionen anlegen. In der Praxis sind aber mehr wie drei Dimension schwer pflegbar. Zur Verdeutlichung: Zwei Dimensionen sind eine Tabelle, drei Dimensionen sind eine drei dimensionales Koordinatensystem, vier Dimensionen ..... usw.

Dynamische Änderung von Datenfeldern.

Bisher haben wir nur Datenfelder deklariert deren Größe wir Anfang des Programmes kennen und die sich nicht ändert. Mit Starbasic lassen sich aber auch Datenfelder im laufenden Programm dimensionieren. Nicht immer lassen sich die Dimension vorher festlegen, sondern sie werden im Ablauf des Programmes erst bekannt. Mit dem Befehl Redim ist eine Neudimensionierung möglich. Das einzige Problem dabei ist das eingetragene Wert verloren gehen.

Redim MyArray(5) as String

erzeugt also eigentlich nur im laufenden Programm das Datenfeld mit neuer Dimension neu. Manchmal hat man aber ein Array und muß es nur vergrößern und möchte die Daten nicht verlieren. Dazu gibt es den Befehl "Redim Preserve".

Redim Preserve MyArray(10) as String

vergrößert also das Datenfeld in unserem Beispiel von fünf auf zehn und die vorhandenen Daten bleiben bestehen. Wichtig ist dabei nur darauf zu achten das sich die Dimension und der Typ nicht ändert. Das ist nämlich nicht zulässig.

Zulässig ist aber auch die Verkleinerung eines Datenfeldes. Dabei werden dann die oberen Werte gelöscht.

Wenn man ein Array auswerten muß, von dem man die Größe nicht kennt, zum Beispiel als Ergebnis eine Funktion, kann mit diese mit Ubound erfahren.

anzahlfelder = ubound(array()).

Dabei ist immer daran zu denken das die Null das erste Element ist. Ist die Anzahl zum Beispiel 10 ist die Gesamtanzahlfelder 11.

Das Gegenstück zu ubound ist lbound. Damit wird der unterste Werte des Arrays bestimmt.

Bei mehrdimensionalen Feldern wird die Dimension bei der Abfrage mit übergeben.

Dim TestArray (2,8)

TestAnzahl = uBound(Testarray(),2)

Achtung! Die Zählung der Dimensionen beginnt hier bei 1, und nicht bei 0.

Konstanten

Zusätzlich zu Variablen lassen sich auch Konstanten deklarieren. Konstanten können nicht überschrieben. Das heißt sie bleiben für den gesamten Ablauf des Programmes erhalten. Deklariert werden sie mit const.

const Konstante as Typ = Wert

Wahlweise kann man die Typendeklaration auch weglassen: const Konstante = Wert.

Konstanten machen dann Sinn wenn man weiß das man bestimmte Werte in mehreren Modulen benötigt und diese sich nicht ändern. Ein Beispiel wäre der Euroumrechnungsfaktor. Wenn dieser sich ändern sollte, muß man nur an einer Programmstelle den Wert ändern.

Variant

Der Datentyp Variant wird in Starbasic automatisch erstellt wenn eine Variable ohne Typenzuordnung deklariert wird.

Variable = 1

Eine derartige Varibale kann dann auch verschiedene Typen nacheinnander aufnehmen.

```
sub varianttest
  var1 = "Test"
  msgbox var1
  var1 = 1
  msgbox var1
end sub
```

In diesem Beispiel bekommt Var1 erst eine Zeichenfolge und dann eine ganze Zahl. Der Typ Variant macht sich die Variable bei der Weiterverarbeitung passend. Wenn man zum Beispiel zwei Variantvariablen addiert und die eine hat einen String und die andere eine Zahl, wird das richtige Ergebnis berechnet wenn der String eine Zahl ist. Ansonsten kommt keine Fehlermeldung, aber ein falsches Ergebnis. Daher ist der Einsatz von Variantvariablen mit Vorsicht zu geniessen. Man kann sich damit "schöne" Fallen in seinen Code schreiben.

Tipp: Alle Variablen ordentlich deklarienen und auf den Typ Variant möglichst verzichten.

3.8 Wie kann man Variablen konvertieren?

Starbasic bietet, intern basierend auf den Typ Variant, zu jedem Typ eine passende Konvertierungfuinktion an.

Str(Var) – wandelt einen beliebigen Datentyp in einen String um.

CInt(Var) – wandelt einen beliebigen Datentypen in einen Integer-Wert um.

CLng(Var) – wandelt einen beliebigen Datentypen in einen Long-Wert um.

CSng(Var) – wandelt einen beliebigen Datentypen in einen Single-Wert um.

CDbl(Var) – wandelt einen beliebigen Datentypen in einen Double-Wert um.

CBool(Var) – wandelt einen beliebigen Datentypen in einen Booleschen Wert um.

CDate(Var) – wandelt einen beliebigen Datentypen in einen Date-Wert um.

Mit diesen Funktionen lassen sich die Variablen in verschiedene Datentyp umwandeln und man kann sich sein das wirklich der gewünschte Typ eingesetzt wird.

Bei diesen Funktionen wird bei Dezimalzahlen das in den Ländereinstellungen eingestellte Trennzeichen verwendet: Punkt oder Komma.

Als Gegenstück zu diesen Verwandlungskünslern gibt es noch die Val(Var)-Funktion. Diese wandelt einen String der nur Zahlen enthält in eine Zahl um. Dabei wir daber grundsätzlich nur der Punkt als Trennzeichen für Dezimalzahlen akzepiert.

3.9 Wie kann man den Typ "Struktur" global verwenden?

Man kann Type nicht direkt global verwenden. Das kann man aber mit einem kleinen Trick.
Man muss die Variable die den Typ enthält global erzeugen und in einem Startmakro zuweisen.

Modul1:

```
Global Mydata
Type Adresse
  sName as string
  sVorname as string
  sPlz as string
  sOrt as string
  sTelefon as string
end Type
sub startalles
mydata=createObject("Adresse")
```

end sub

Modul2:

sub test

startalles

with mydata

.sname="FFF"

.svorname="ggg"

end with

msgbox mydata.sname

end sub

4 Dialoge/Formulare

4.1 Dialoge

4.2.1 Wie öffnet man einen Dialog?

Ein Dialog wird über das Laden des Dialoges zu einem Objekt, dass man ausführen kann (execute)

Dim MyDlg as Object

Mit der Funktion Dialoglibrary.Loadlibrary wird die Bibliothek mit den Dialogen geladen

Dialoglibraries.Loadlibrary("Standard")

Nun wird der gewünschte Dialog zugeordnet.

MyDlg= CreateUnoDialog(Dialoglibraries.Standard.Dialog1)

Auf dessen Model kann man zugreifen und Elementwerte auslesen oder ändern.

MyDlg.Model.xxxx

Mit dem Befehl execute wird der Dialog gestartet.

MyDlg.execute()

4.2.2 Wie schließe ich einen Dialog?

Mit dem Aufruf der Methode Dialog.endexecute() und nötigenfalls Dialog.dispose(). Diesen der Routine "Ausglöst" bei dem gewünschten Button (z.B. Schließen) hinterlegen.

Mit endexecute wird der Dialog geschlossen mit dispose wird der Speicher freigegeben.

Beispiel

```
Sub cbSchliessen
  oDialog.endexecute()
  oDialog.dispose()
End Sub
```

Die andere Variante ist einer Schaltfläche bei der Eigenschaft "Art der Schaltfläche" OK oder Abbrechen zu zuweisen. Bei OK wird der Dialog mit dem Rückgabewert 1 geschlossen, bei Abbrechen mit 0. Diesen kann von Dialog.execute abfragen.

```
Rueckwert=Dialog.execute()
```

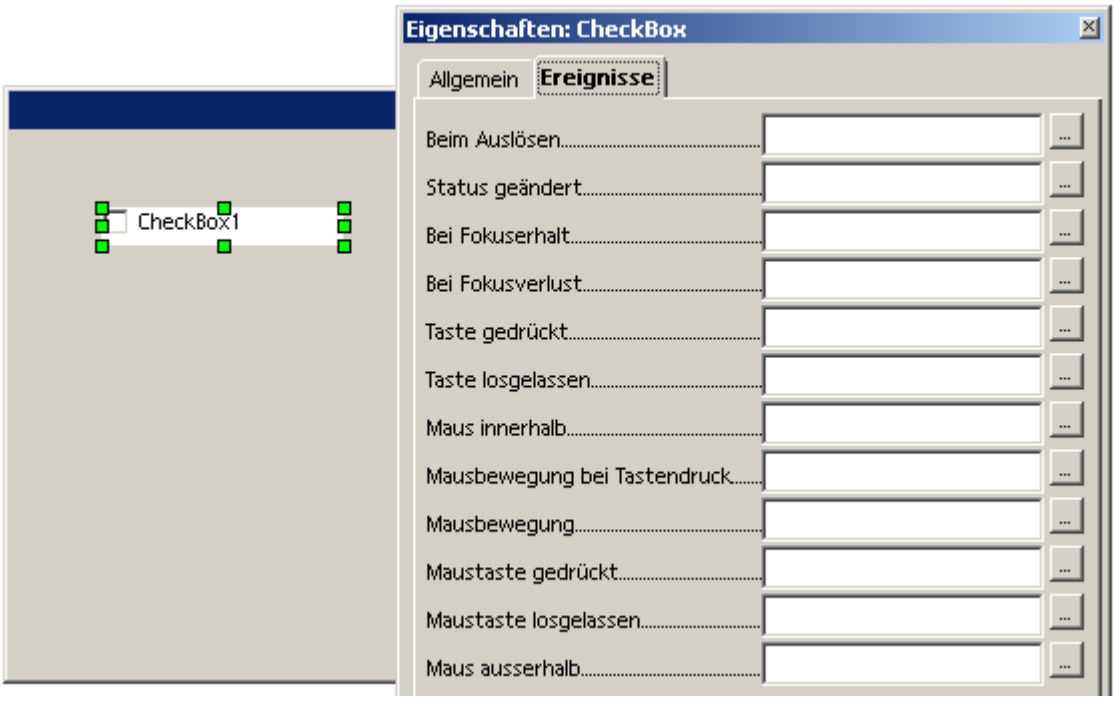
Aber Achtung um den Dialog bei Bedarf wirklich aus dem Speicher zu löschen muß noch Dialog.dispose() ausgeführt werden.

Leider führt der Befehl bei einigen Installationen zu Abstürzen. Daher muß man unter Umständen darauf verzichten.

4.2.3 Wie kann man Dialoge während der Eingabe manipulieren?

Man kann die Ereignisse von Kontrollfeldern abfragen und entsprechende Makros starten. Mit diesen kann man auch den Dialog selber ändern.

Die Ereignisse eines Kontrollfeldes sind bei den Eigenschaften eines Kontrollfeldes zu finden.



Als Beispiel verwende ich hier eine Checkbox. Wir werden jetzt den Status der Checkbox prüfen und dem Status entsprechend Änderungen im Dialog durchführen.

Als Erstes ist es wichtig den Dialog öffentlich zu deklarieren damit alle Routinen auf das Object zugreifen können. Die erste Routine startet den Dialog.

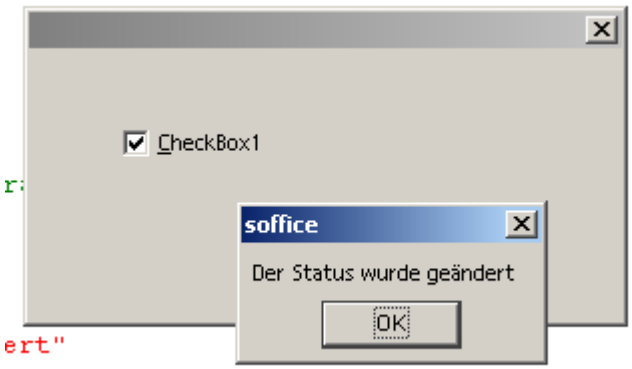
```
Public myDlg as Object

sub test_dialog
  doc = thiscomponent
  dlg = CreateUnoDialog(DialogLibraries.Standard.Dialog1)
  dlg.Execute()
end sub
```

Das folgende Makro ist mit dem Ereignis "Status geändert" der Checkbox verknüpft.

```
sub checkboxchange
   msgbox "Der Status wurde geändert"
end sub
```

Klickt man jetzt die Checkbox an, erscheint die Meldung.



Anstatt der Meldung kann man aber auch andere Kontrollfelder manipulieren. Im Beispiel fügen wir jetzt dem Dialog ein Textfeld hinzu, das nur zu sehen ist und Eingaben zuläßt wenn die Checkbox aktiviert ist. Um das Textfeld nicht sichtbar zu machen muss das Textfeld vor dem Start des Dialoges eingestellt (visible=false) werden.

```
sub test_dialog
  doc = thiscomponent
  dlg = CreateUnoDialog(DialogLibraries.Standard.Dialog1)
  ctl = dlg.getControl("TextField1")
  ctl.visible=true
  dlg.Execute()
end sub
```

Danach kann man diesen Parameter in der Routine zu Checkbox ändern. Dabei wird der Status der Checkbox (0/1) abgefragt und entsprechend reagiert.

```
sub checkboxchange
  ctlcheck=dlg.getControl("CheckBox1")
  ctl = dlg.getControl("TextField1")
  if ctlcheck.model.State=1 then
     ctl.visible=true
   else
     ctl.visible=false
    end if
end sub
```

Dieses Beispiel läßt sich auf alle anderen Manipulationen eines Kontrollfeldes ableiten. Einzig die Größe des Dialoges kann man nicht ändern.

4.2.4 Wie geht man mit Kontrollfeldern in Dialogen um?

Dialoge bestehen aus Kontrollfeldern, die die eigentliche Eingabe unterstützen.
Folgende Kontrollfelder gibt es:

Schaltflächen (CommandButton)

Schaltflächen sind da um einen bestimmten Befehl nach einem Mausklick durch den Anwender auszuführen. Üblicherweise steht auf der Schaltfläche der Befehl.

OptionButtons

Ein OptionButton kann eigentlich immer nur zusammen mit mindestens einem weiteren Optionbutton verwendet werden. Mehrere Optionbuttons werden zusammengefasst, um zwischen ihnen eine eindeutige Auswahl zu ermöglichen.

Checkboxen

Ein Checkbox ist für die Abfrage von Ja/Nein-Werten gedacht. Sie sind damit eine Kurzfassung von zwei Optionbutton.

Listboxen

Eine Listbox ist eine Liste von Einträgen aus denen der Anwender auswählen kann. Zum Einen stehen Eigenschaften in der IDE zur Verfügung zum Anderen noch Methoden des Objektes die den Zugriff auf die Listenelemente ermöglichen.

Textfeld (Textfield)

Eingabefeld für Text/Zahlen

Kombinationsfeld

Das Kombinationsfeld ist der Listbox ähnlich. Das Feld stellt auch eine Liste zur Auswahl zur Verfügung. Aber mit einer entscheidenden Unterschied in dem Textfeld oberhalb der Liste kann der Anwender eine neue Eingabe tätigen. Außerdem ist beim Kombinationsfeld keine Mehrfachauswahl möglich.

Rahmen (FrameControll)

Rahmen in Dialogen haben zwar in Starbasic den Namen FrameControl, was darauf schließen lassen könnte, dass sich mit diesen Objekt zusammenfassen lassen. Dies ist leider nicht möglich. Ein Rahmenobjekt ist in Starbasic-Dialogen einfach nur eine Gestaltungsmittel. Zum Beispiel kann man damit Gruppen von Optionbutton optisch hervorheben.

Beschriftungsfeld (Label)

Dieses ist ein Feld mit dem ein beliebiger Text im Dialog positioniert werden kann. Obwohl in der Maske im Dialogeditior "Titel" steht ist die Property "Label".

Grafisches Kontrollfeld (ImageControl)

Dieses Feld dient dazu um eine Grafik im Dialog anzuzeigen. Diese Grafik muss zur Laufzeit entsprechend der Pfadangabe zur Verfügung stehen.

Kontrollfelder können über ihren Namen angesprochen werden.

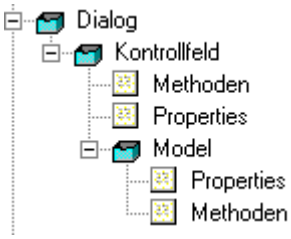
Nachdem der Dialog geladen worden ist, stehen die Kontrollfelder zur Verfügung. Und können mit GetControl zugewiesen werden.

```
DialogLibraries.LoadLibrary( "Standard" )
```

MyDialog = CreateUnoDialog(DialogLibraries.Standard.Dialog1)

MyControl=MyDialog.GetControl("Kontrollfeld")

Danach stehen unter dem Objekt MyControl die Eigenschaften und Methoden zur Verfügung. Viele davon, vor allem die Wichtigsten, befinden sich im Service Model von MyControl.



Objektübersicht eines Kontrollfeldes

Es gibt einige Eigenschaften die alle Kontrollfelder haben. Die Wichtigsten hiervon sind:

Eigenschaften von Kontrollfeldern

Name, Aktiviert,Seite(Step),Höhe,Breite,PositionX,PositionY,Zusatzinformationen,Hilftext,Hilfe URL.

Name

In der Eigenschaft "Name" steht der eindeutige Name des Kontrollfeldes. Mit diesem Namen erfolgte der Zugriff auf dieses Kontrollfeld. Diesen Namen zu ändern macht nur Sinn wenn man ein Kontrollfeld während der Laufzeit neu hinzufügt. Normalerweise wird der Name beim Erstellen des Dialoges im Dialogeditor eingetragen. Der Name eines Kontrollfeldes muß innerhalb eines Dialoges eindeutig sein. Dabei wird Groß- und Kleinschreibung unterschieden.

Die wesentliche Verwendung erfolgt beim Aufruf eines Kontrollfeldes über Dialog.getControl(Name)

Aktiviert

Über den Schalter "Aktiviert", wird entschieden ob das Kontrollfeld aktiviert oder nicht aktiviert ist. Ein nicht aktiviertes Kontrollfeld kann nicht angesprochen und ist dann nur als hellgrauer Schatten zu sehen. Die Eigenschaft im Model ist enabled.

MyControl.Model.enabled = true

Seite(Step)

Mit dieser Eigenschaften kann man Kontrollfelder bestimmten Dialogseiten zuordnen. Dies erfolgt in der Regel auch beim Gestalten eines mehrseitigen Dialoges. Näheres zu Erstellen von mehrseitigen Dialogen steht hier.

Zugriff über

MyControl.Model.Step=0

Reihenfolge

In der Reihenfolge (Tabindex) ist die Stelle des Kontrollfeldes innerhalb des Dialoges hinterlegt. Hat man zum Beispiel drei Eingabefelder, die hintereinander abgefragt werden sollen, bekommen die drei fortlaufende Einträge in der Eigenschaft Reihenfolge. Diese Reihenfolge entspricht auch der Bewegung innerhalb der Maske mit der Tabulator-Taste.

Höhe

Legt die Höhe des Objektes in Pixel fest.

MyControl.Model.Height = 15

Breite

Legt die Breite des Objektes in Pixel fest.

Starbasic: MyControl.Model.Width = 15

PositionX, PositionY

Mit diesen beiden Eigenschaften läßt sich das Kontrollfeld innerhalb des Dialoges positionieren.

StarBasic: MyControl.Model.PositionX und MyControl.Model.PositionY

Zusatzinformation

Hier kann ein zusätzlicher Text hinterlegt werden. Zum Beispiel ein Kommentar zu dem Kontrollfeld.

Starbasic: MyControl.Model.Tag

Hilfetext

Dies ist ein automatisch eingeblendeter Hilfetext, der angezeigt wird,

sofern der Mauszeiger über dem Steuerelement zum Stehen kommt.

HilfeUrl

Legt die ID-Nummer für den Zugriff auf die Online Hilfe fest. Damit wird eine bestimmte Stelle in der Online-Hilfe mit dem Kontrollfeld verbunden, die dann beim Drücken der F1-Taste automatisch aufgerufen wird. Theorie! In der Praxis ist die Funktion nicht hinterlegt.

Zeichensatz

Mit der Eigenschaft Zeichensatz öffnet sich in der IDE ein Dialog mit den Schrifteinstellungen. Innerhalb von Starbasic kann man diese Parameter ebenfalls ansprechen. Dafür steht die Eigenschaft FontDesscriptor als Struktur zur Verfügung. Der Umgang mit Strukten bei Objekten steht im Kapitel "Objekte und Strukturen"

'Anlegen des Strukturobjektes

Dim myFont As New com.sun.star.awt.FontDescriptor

'Zuweisen der Werte

MyFont.Name = "Arial"

MyFont.StyleName = "Fett"

MyFont.Height = 8

'Übergabe an Konrollfeld

MyControl.Model.FontDescriptor=MyFont

Drucken

Entscheidet ob ein Kontrollfeld beim Ausdrucken auf dem Papier mit gedruckt wird. Dies ist eine Eigenschaft die für Basicdialoge nicht wichtig ist. Man kann aber Kontrollfelder auch in Dokumenten verwenden. Dort mach diese Eigenschaft dann einen Sinn. Man kann zu Beispiel ein Dokument mit einer Schaltfläche versehen, die dann nicht mit gedruckt wird.

Starbasic: MyControl.Model.Printable = True

Tapstop

Mit Tapstop wird entschieden ob das Kontrollfeld mit der Tabtaste erreichbar ist. Also ob es innerhalb der Reihenfolge beim Sprung mit Tab- Taste beachtet oder ignoriert werden soll. Es gibt zu jedem Kontrollfeld einen Standardwert. Zum Beispiel bei einer Schaltfläche ja und bei einem Label nein.

Drehfeld

Drehfelder sind Eingabefelder, die sich noch zusätzlich mit Pfeiltasten einstellen lassen. Sie machen nur bei Zahlen und Datums/Zeit- Eingaben einen Sinn.

Starbasic: MyControl.Model.Spin = true / false

Nur Lesen

Der Inhalt des Steuerelementes ist schreibgeschützt. Er kann also nicht geändert werden. Dies kann für Datenfelder, die nicht geändert werden dürfen, interessant sein. Außerdem besteht auf die Daten Zugriff um diese zum Beispiel zu markieren und zu kopieren.

StarBasic: MyControl.Model.Readonly=True / False

4.2.5 Probleme mit dem Gruppen-Steuerelement - FrameControl

Im Gegensatz zu seiner Bezeichnung "Gruppen-Steuerelement" bzw. "Framecontrol" läßt sich mit diesem Objekt keine Gruppe von Objekten zusammenfassen. Es handelt sich hierbei lediglich um eine optische Möglichkeit den Dialog zu strukturieren. Objekte innerhalb eines solchen Rahmens können nicht automatisch gemeinsam mit dem Rahmen verschoben werden.
Warum es den eindeutig nicht korrekt beschreibenden Namen bekommen hat, bleibt in den Köpfen der

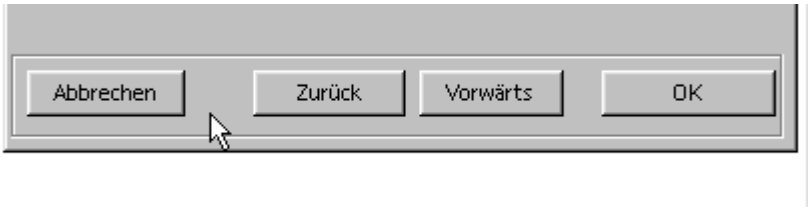
Entwickler.

4.2.6 Wie kann man mehrseitige Dialoge erstellen?

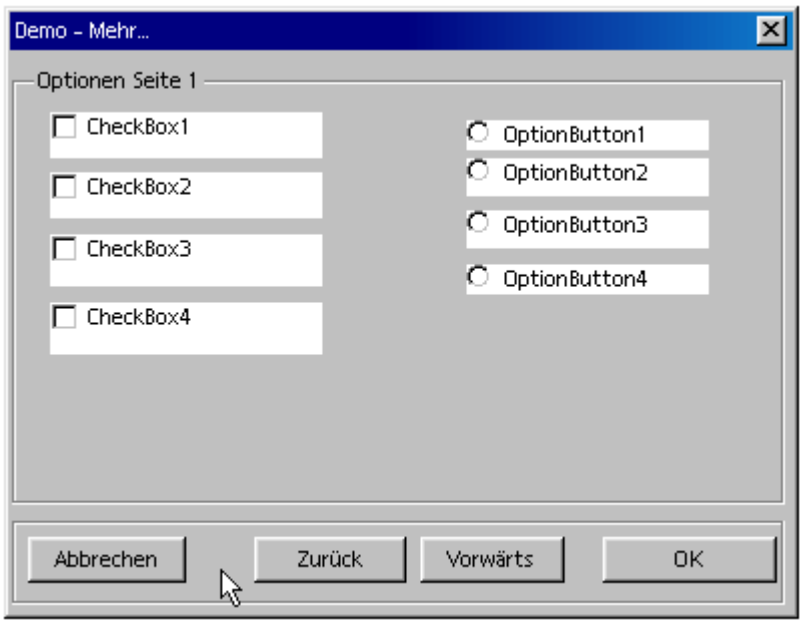
Mit der Eigenschaft "Seite" beim Dialog und den Kontrollfeldern. Mit dieser kann man die einzelnen Objekte Seiten zu ordnen, wenn man dann die Seite des Dialoges aktiviert, erscheinen nur die Objekte mit der gleichen Seitennummer.
Die folgende Anleitung steht auch hier: http://www.dannenhoefer.de/down/mehrseitige_dialoge.sxw

Wir erstellen jetzt einen zweiseitigen Dialog, bei dem zwischen den Seiten gewechselt werden kann. Welche Elemente braucht solch ein Dialog. Erstmal braucht er auf jeden Fall Schaltflächen die immer zu sehen sind und mit denen der Dialog gesteuert werden kann. Also legen wir vier Schaltflächen an: Abbrechen, OK, Vorwärts und Zurück. Diese können wir noch optisch mit einem Rahmen trennen. Damit der Rahmen keinen Text enthält bleibt die Eigenschaft Titel leer.

Das sollte dann in etwa so aussehen:

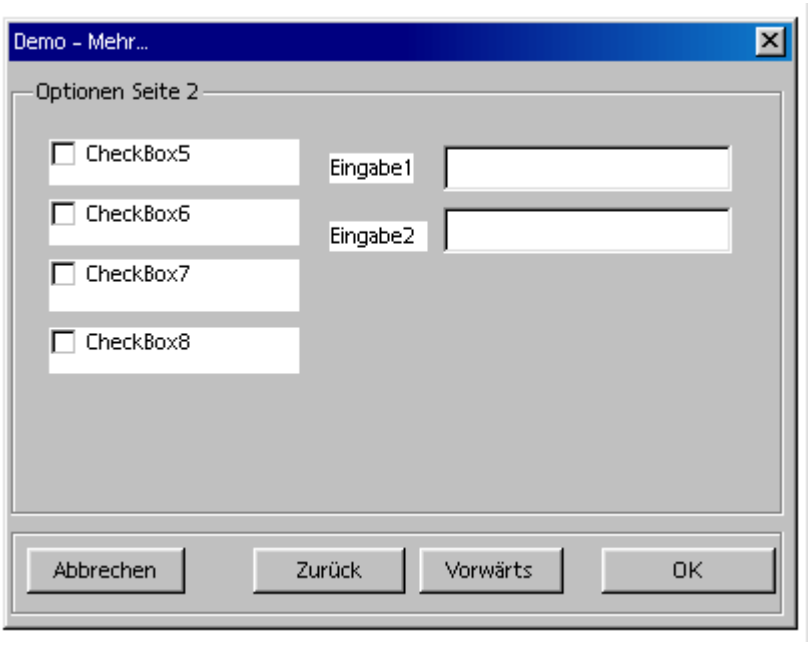


Die Schaltflächen und der Rahmen haben in der Eigenschaft "Seite" die 0 eingetragen. Sie sollen auf allen Dialog-Seiten sichtbar sein. Jetzt legen wir eine zweite Seiten mit einigen Checkboxen an. Dazu wechseln wir in der Eigenschaft "Seite" auf die 1. Wir sehen jetzt natürlich keinen Unterschied, weil ja alle Elemente zu sehen sind. Um jetzt auch dem Anwender die Trennung der Seiten leichter zu machen legen wir um die Checkboxen einen Rahmen mit einem Titel. In unserem Beispiel "Optionen Seite 1".



Jetzt brauchen wir noch die zweite Seite. Dazu markieren wir den Dialog und stellen die Eigenschaft Seite auf 2. Sofort verschwinden Rahmen, Checkboxen und die Optionsbutton. Wenn das nicht passiert, sind sie noch der falschen Seite, 0, zugeordnet.

Auf dieser neuen Seite können wir jetzt den nächsten Rahmen und weitere Checkboxen und Eingabefelder anlegen.



So jetzt haben wir das Grundgerüst unseres Dialoges. Jetzt müssen wir diesen noch mit Leben erfüllen. Die zwei einfachen Teile davon sind hinter der Abbrechen und der OK Schaltfläche verbunden. In beiden Fällen muß sich unser Dialog schließen. Einmal mit der Rückmeldung "Abbruch" und einmal mit der Rückmeldung "OK". Wir erinnern uns, dazu stellen wie die "Art der Schaltfläche" auf Abbrechen bzw. OK.

Das war der einfache Teil. Jetzt wollen wir die Reaktion auf das Drücken der anderen beiden Schaltflächen festlegen. Wenn die Schaltfläche "Zurück" gedrückt wird soll der Dialog eine Seite zurück blättern. Moment mal? Wenn man auf der ersten Seite ist, kann man nicht weiter zurück. Also müssen wir später noch dafür sorgen das die Schaltfläche nur aktiv ist wenn es Sinn macht. Das gilt nachher auch für die andere Schaltfläche "Vorwärts". Kümmern wir uns erstmal um das wechseln der Seiten.

Das erste Wichtige ist, dass das Dialogobjekt öffentlich existiert. Also erstmal eine Routine zum Öffnen des Dialoges:

```
Public MyDlg as object

Sub CallDialog

  DialogLibraries.LoadLibrary( "Standard" )

  MyDlg = CreateUnoDialog( DialogLibraries.Standard.Dialog2 )

  MyDlg.Model.Step=1

  MyDlg.Execute()

  MyDlg.Dispose()

end Sub
```

Man sollte immer die gewünschte Seite beim Aufruf einstellen. Die IDE sorgt dafür, dass immer die letzte in der IDE aktive Seite aufgerufen wird. Man verhindert also mit dem Einstellen vor dem Aufruf Fehler.

Jetzt kommt die Routine die mit der Vorwärts-Schaltfläche verknüpft werden muß. In dieser müssen wir nur die Seite des Dialoges ändern. Um es universal zu verwenden, erhöhen wir die aktuelle Seitenzahl um eins.

```
Sub schflaeche_cbFore
```

```
  MyDlg.Model.Step=   MyDlg.Model.Step+1
End Sub
```

Das war es schon. Jetzt verknüpfen wir diese Routine mit der Schaltfläche und schon können wir den ersten Test durchführen. Das sieht ja nicht schlecht aus aber jetzt sehen wir schon das oben erwähnte Problem. Wenn man nach unseren zweiten Seite nochmal Vorwärts anklickt scheint nichts zu passieren. In Wirklichkeit wird aber die Dialogseite weiter hochgesetzt. Am Besten fügen wir den Befehl für Zurück ein, dann können wir schauen was da passiert.

Für die Schaltfläche "Zurück" können wir fast den Code von "Vorwärts" nehmen. Wir müssen nur den Seitenwert um eins runtersetzen statt rauf.

```
Sub schflaeche_cbBack
  MyDlg.Model.Step=   MyDlg.Model.Step-1
End Sub
```

Wenn wir nun den Dialog starten können wir beliebig oft zurück und vorwärts blättern. Das liegt daran, dass wir dem Dialog keine Einschränkung bezüglich der Seitenanzahl zuordnen können. Also wird der Wert immer weiter rauf oder runter gesetzt. Um dies zu verhindern prüfen wir die Position der Seite und handeln entsprechend nach dem Auslösen der Schaltflächen. Als unteres Ende ist die 0 bekannt. Diese darf nicht eingestellt werden, da sonst alle Kontrollfelder sichtbar sind. Das obere Ende ist die Zwei, die wir als letzte Seite angelegt haben.

Wir prüfen also bei zurück gehen ob wir die 1 erreicht haben und deaktiveren dann die Zurück-Schaltfläche, so daß dann nicht mehr gedrückt werden kann.

```
Sub schflaeche_cbBack
 Back=MyDlg.GetControl("cbBack")
  MyDlg.Model.Step=   MyDlg.Model.Step-1
  If MyDlg.Model.Step=1 then
    back.model.enabled=false
   else
    back.model.enabled=true
  End if
End Sub
```

Natürlich müssen wir auch den möglichen Zustand der anderen Schaltfläche prüfen. Diese könnte ja deaktiviert sein und muß wieder aktiviert werden.

```
Sub schflaeche_cbBack
 Back=MyDlg.GetControl("cbBack")
  MyDlg.Model.Step=   MyDlg.Model.Step-1
  If MyDlg.Model.Step=1 then
    back.model.enabled=false
   else
    back.model.enabled=true
  End if
 Fore=MyDlg.GetControl("cbFore")
  If MyDlg.Model.Step<2 then
    Fore.model.enabled=true
```

```
    else
      Fore.model.enabled=false
    End if
End sub
```

Wir haben jetzt die Routine für die Zurück-Schaltfläche und brauchen das Ganze nochmal für die Vorwärts-Schaltfläche. Aber dann haben wir ja eigentlich viel Code doppelt. Also machen wir es uns etwas einfacher. Wir trennen den Code, der die Aktivierung der Schaltflächen macht, heraus, machen uns daraus eine neue Routine und rufen diese zweimal auf.

```
Sub schflaeche_einstellen
  Back=MyDlg.GetControl("cbBack")
   If MyDlg.Model.Step=1 then
      back.model.enabled=false
    else
      back.model.enabled=true
    End if
  Fore=MyDlg.GetControl("cbFore")
   If MyDlg.Model.Step<2 then
      Fore.model.enabled=true
    else
      Fore.model.enabled=false
    End if
End sub
Sub schflaeche_cbBack
    MyDlg.Model.Step=   MyDlg.Model.Step-1
    schflaeche_einstellen
End sub
Sub schflaeche_cbFore
    MyDlg.Model.Step=   MyDlg.Model.Step+1
    schflaeche_einstellen
End sub
```

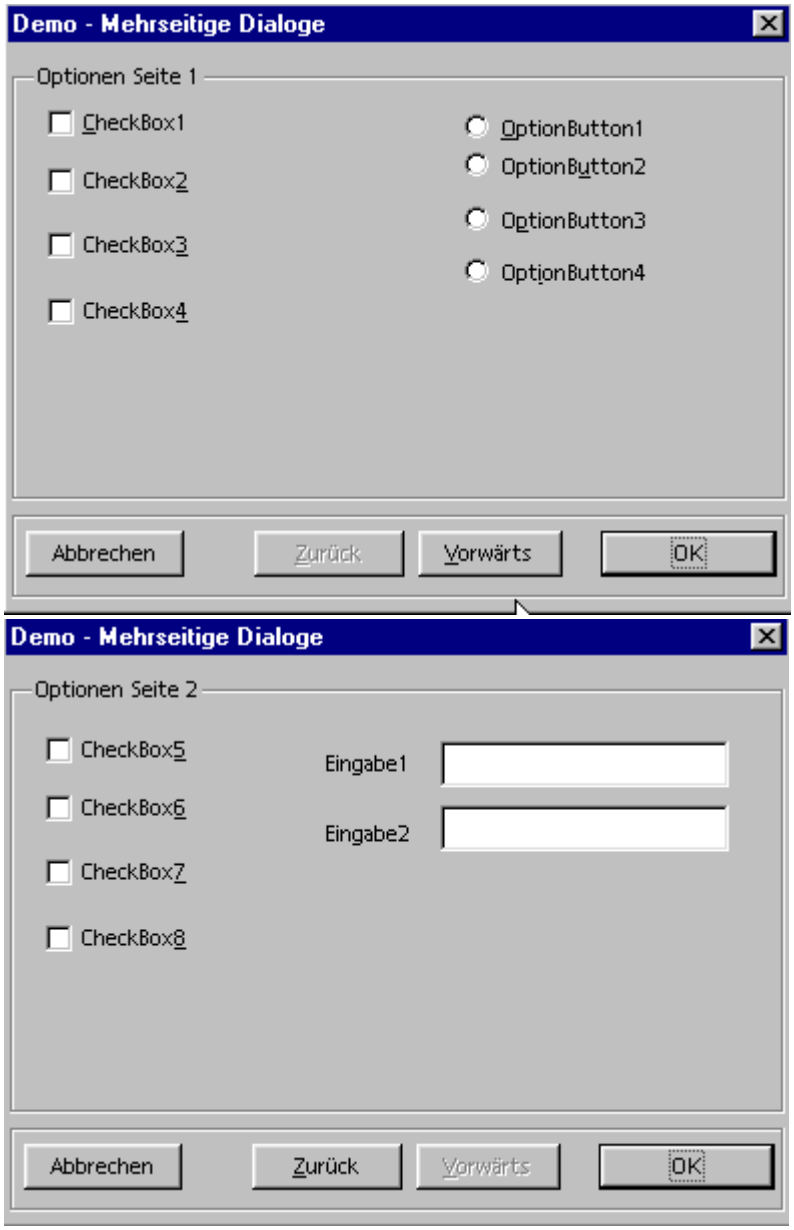
Mit der zusätzlichen Routine haben wir noch einen Vorteil. Mit ihr können wir auch den Dialog beim ersten Start einstellen.

```
Sub CallDialog
  DialogLibraries.LoadLibrary( "Standard" )
  MyDlg = CreateUnoDialog( DialogLibraries.Standard.Dialog2 )
  MyDlg.Model.Step=1
  schflaeche_einstellen
  MyDlg.Execute()
```

MyDlg.Dispose()

end Sub

Und fertig ist unser Dialog mit zwei Seiten und richtig angepaßten Schaltflächen.



4.2.7 Tipp: Dialoge in der IDE verschieben

Wenn mann einen Dialog innerhalb der IDE mit der Maus verschiebt bleiben dabei die Kontrollelemente an ihrer Stelle. Diese werden nicht mitverschoben.

Verwendet man dagegen den Eigenschaftskontrolldialog, x und y Position, werden die Kontrollelemenete ordentlich mit verschoben.

4.2 Formulare

4.3.1 Wie erhält die im Formular ausgewählten Werte eines Datensatzes?

Der Zugriff auf den im Formular aktuell ausgewählten Datensatz erfolgt wie bei einem resultset. Nur das hier das Resultset direkt das Formular ist. Der aktuelle Datensatz ist innerhalb des Formulares eine Zeile mit den entsprechenden Spalten des Datensatzes. Der Zugriff erfolgt dann mit get(typ) und der Spaltenangabe.

```
odoc=thiscomponent
oform=odoc.drawpage.forms.getbyindex(0)
myselect=oform.getstring(1)
```

Die Spaltenzählung beginnt wie bei Resultset mit 1.

4.3.2 Wie kann man auf Unterformulare zugreifen?

Ein Unterformular ist ein Element des Hauptformualars und wird genau wie ein Kontrollelement angesprochen:

```
odoc=thisComponent
odraw1=odoc.drawpage
form1=odraw1.forms.getbyIndex(0)
osubform=form1.getbyname("SubForm")
```

Um die Unterformulare aufzulisten muß die Elemente de Hauptformulares prüfen:

```
form1=odraw1.forms.getbyIndex(0)
for i=0 to form1.count-1
  oform=form1.getbyindex(i)
  if oform.SupportsService("com.sun.star.form.component.DataForm") then msgbox "Subformular"
next i
```

4.3.3 Wie kann man auf Kontrollfelder in Formularen zugreifen?

Nach dem man sich das Objekt des richtigen Formulares geholt hat, kann man mit getByName (nicht wie bei Dialogen mit getControl) auf Kontrollfelder zugreifen.
Danach gelten die normalen Eigenschaften eines Kontrollfeldes, mit Ausnahme der besonderen Eigenschaften für Dialoge (z.B. Position, Seite, Farben usw.).

odoc=thisComponent

oSheet=odoc.sheets(0)

odraw1=oSheet.drawpage

form1=odraw1.forms.getbyIndex(0)

oContr=form1.getbyname("TextBox")

4.3.4 Wie kann man verhindern, dass Formularkontrollfelder mit dem Text wandern?

Formularkontrollfelder werden standardmäßig beim Anlegen mit dem Absatz verbunden. Man kann ein Kontrollfeld auch mit der Seite verankern oder als Zeichen behandeln lassen. Wenn man mit der rechten Maustase auf das Kontrollfeld klickt, kann man im erscheinenden Kontextmenü unter Verankerung auswählen.

Am Absatz: Das Kontrollfeld wird mit den Absatz verbunden und wandert mit diesem mit.

An der Seite: Das Kontrollfeld wir auf der Seite fixiert. Text wird dahinter vesteckt.

Als Zeichen: Das Kontrollfeld wird in die Zeile gesetzt und verhält sich wie ein Buchstabe/Wort. Dabei paßt sich die Zeilenhöhe automatisch an.

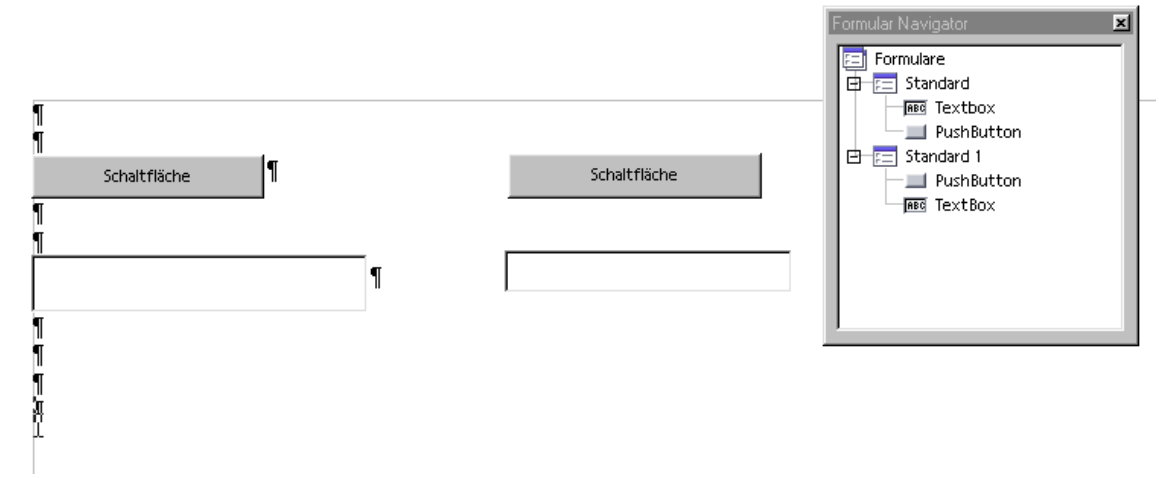
4.3.5 Wie geht man mit Formularen um?

Hier erst mal eine allgemeine Einführung in die Verwendung von Formularen.

Formulare können innerhalb von Writer-, Calc- und Präsentation-Dokumenten verwendet werden. Sie liegen innerhalb des Dokumentes in einer besonderen "Ebene", der Drawpage. Bei Textdokumenten gibt es nur eine, bei Calc- und Präsentations-Dokumenten für jedes Tabellenblatt eine eigene Drawpage.

Alle Formularfunktionen stehen laut Dokumentation nur innerhalb von Text- und Calcdokumenten zur Verfügung. Leider habe ich noch keine Stelle über die Einschränkungen gefunden.

Innerhalb einer Drawpage sind mehrere Formulare möglich. Man sollte hier den Begriff vielleicht auch eher als Gruppe verstehen. Da man auf einer Seite die Formular-Kontrollfelder untereinander mischen kann, lässt sich durch die Gruppierung nicht automatisch ein sichtbares Formular darstellen. Viel mehr können die Kontrolfelder vermischt auf der Seite verwendet werden und verschiedenen Formularen angehören.



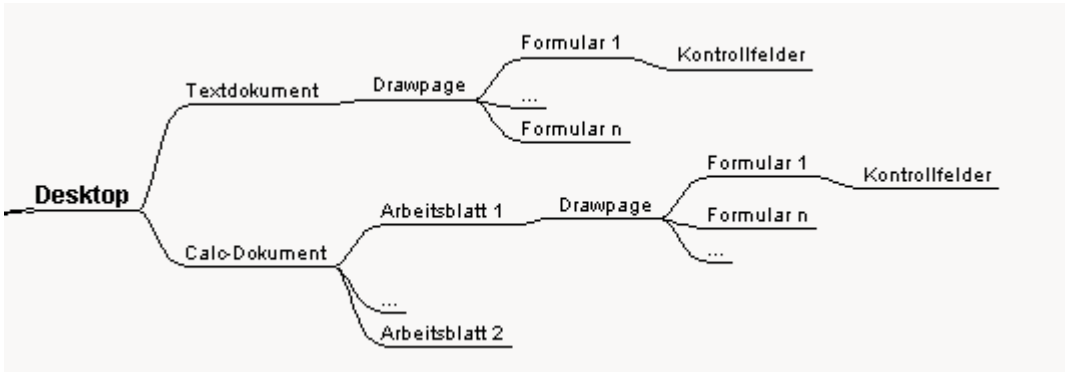
In dem obigen Beispiel sieht man vier Kontrollfelder die zu zwei Formularen gehören. Durch die Verwendung des Formular-Navigators wird dies angezeigt. Diese Kontrollfelder können beliebig plaziert werden. Dadurch kann man bei Verwendung mehrerer Formulare auf einer Seite die Übersicht verlieren oder sich sogar Übersicht schaffen.

Die Kontrollfelder verhalten sich wie die Kontrollfelder für Dialoge und habe ähnliche Eigenschaften und Ereignisse. Um mit diesen also in einem Makro zu arbeiten gelten im wesentlichen die Beispiele für Dialoge und Formulare.

Naja nicht ganz, es gibt noch einen wirklich wesentlichen Unterschied. Kontrollfelder in Formularen können mit Datenbanktabellen verknüpft werden. Dazu steht im Eigenschaften-Dialog ein zusätzliches Register "Daten" zur Verfügung.

Um an die Kontrollfelder von Formularen zu kommen, muß man die Drawpage und das Formular des Dokumentes aufrufen.

Objektstruktur



Zugriff auf die Kontrollfelder:

Der erste wichtige Unterschied ist, dass es die Methode "getControl" nicht gibt. Der Zugriff auf Kontrollfelder erfolgt einfach über den Namen: getByName().

Fangen wir mit einem Textdokument an.

odoc=thisComponent

odrawpage=odoc.drawpage

oform=odrawpage.forms.getbyIndex(0)

oder

oform=odrawpage.forms.getbyName("Standard")

oControl=oform.getByName("TextBox")

Ein Objekt eines Formulares kann man über den Index (getByIndex()) oder über den Namen (getByName()) erhalten. Dies gilt auch für Calc-Dokumenten.

Tabellen:

odoc=thisComponent

oSheet=odoc.sheets(0)

odraw1=oSheet.drawpage

form1=odraw1.forms.getbyIndex(0)

oContr=form1.getbyname("TextBox")

Bei Tabellen gibt es für jedes Sheet eine eigene DrawPage. Dementsprechend muß man das gewünschte Sheet vorher auswählen.

4.3.6 Wie kann man Kontrollfelder unsichtbar schalten?

Kontrollfelder auf auf Formularen können nicht direkt über eine Proberties des Kontrollfeldes sichtbar/Unsichtbar gestellt werden. Bei Formularen erfolgt es über den aktuellen Controllers des Dokumentes. Dessen Object des Kontrollfeldes hat die Eigenschaft View.

oDoc = thisComponent

oController = oDoc.getCurrentController()

oform=odoc.drawpage.forms.getbyindex(0)

oKontroll = oForm.getByName("MeinKontroll")

oKView = oController.getControl(oKontroll)

oKView.visible = false

4.3 Kontrollfelder

4.4.1 Welches sind die Besonderheiten eines Grafikkontrolls?

Grafisches Kontrollfeld

Dies Feld dient dazu eine Grafik im Dialog anzuzeigen. Diese Grafik muss zur Laufzeit entsprechend der Pfadangabe zur Verfügung stehen.

Grafik

Hier wird die Grafik eingetragen.

MyControl.model.imageurl="file:///C:/Programme/StarOffice6.1/share/gallery/bullets/poliball.gif"

Rahmen

Mit dieser Eigenschaft kann man die drei verschiedenen Rahmentypen für eine Grafik festlegen: Ohne Rahmen, 3D und Flach.

Zugriff Starbasic ?

Skalieren

Mit Skalieren kann man einstellen, ob die Grafik der Rahmengröße angepaßt werden soll. Wenn die Einstellung ja ist, wird eine größere Grafik in dem Rahmen kleiner dargestellt und eine kleinere Grafik größer. Bei Nein wird die Grafik in Originalgröße dargestellt. Bei größeren Grafiken kann man dabei den Ausschnitt nicht bestimmen.

MyCtrl.Model.ScaleImage=false / true

4.4.2 Welches sind die Besonderheiten von Commandbutton?

Schaltflächen sind da, um bestimmte Befehle nach einem Mausklick durch den Anwender auszuführen. Üblicherweise steht auf der Schaltfläche der Befehl.

Neben den meisten Standard-Eigenschaften gibt es noch folgende:

Titel

Diese Eigenschaft legt den Text in der Schaltfläche fest. Leider ist Starbasic hier nicht konsequent. Die Eigenschaft in Starbasic lautet "Label" und nicht wie man aus der IDE schließen könnte "Title", wie beim Dialog zum Beispiel.

MyControl.Model.Label = "OK"

Standardschaltfläche

Mit dieser Eigenschaft kann man eine Schaltfläche als Standard auf die Eingabetaste definieren. Dann reagiert diese Schaltfläche alleine auf die Eingabetaste.

Diese Eigenschaft darf man nur einer Schaltfläche innerhalb eines Dialoges zuweisen. Leider ist es aber auch mehrfach möglich und führt zu keiner Fehlermeldung.

MyControl.Model.DefaultButton= true / false

Art der Schaltfläche

Mit dieser Eigenschaft lässt sich ein Rückgabewert der Dialog.Excute Funktion und das Verhalten der Schaltfläche festlegen. Es sind folgende Einstellungen möglich:

0 = Standard: Die Schaltfläche reagiert nicht mit dem Dialog.

1 = OK: Der Dialog wird geschlossen und der Rückgabewert von Dialog.execute ist 1.

2 = Abbrechen:Der Dialog wird geschlossen und der Rückgabewert von Dialog.execute ist 0.

3 = Hilfe: Die Online Hilfe wird aufgerufen. *Sollte unter HilfeUrl ein Eintrag steht, wird diese Stelle in der Online Hilfe angesprungen.*

Um den Rückgabewert abzufragen, muss der Dialog als Funktion aufgerufen werden.

Wert = Dialog.execute()

Um die Art der Schaltfläche zu ändern, wird die Eigenschaft PushButtonType in Starbasic verwendet.

MyControl.Model.Pushbuttontype = 1

Hat aber keine Auswirkung

Status

Diese Eigenschaft ist in der IDE zwar auswählbar,hat aber keine Auswirkung.

Hintergrundfarbe (2.0)

Hier lässt sich eine andere Hintergrundfarbe einstellen.

Grafik

Mit der Eigenschaft Grafik kann die Schaltfläche mit einer Grafik anzeigen lassen. Es gibt also drei mögliche Schaltflächen: Nur Text, Text und Grafik und nur Grafik.

Die Grafikdatei muß auf dem Computer mit der Pfadangabe existieren. Die Grafik wird nicht mit dem Makro gespeichert. Also ist diese Option mit Vorsicht zu verwenden wenn die Makros verteilt werden müssen.

Der Zugriff über Starbasic erfolgt über myctrl.model.imageurl. Dieser Eigenschaft muß dann der komplette Pfadname übergeben werden. Dabei ist auf die URL-Schreibweise zu achten.

MyControl.model.imageurl="file:///C:/Programme/StarOffice6.1/share/gallery/bullets/poliball.gif"

4.4.3 Welches sind die Besonderheiten von Labels?

Beschriftungsfeld

Dies ist ein Feld mit dem beliebiger Text im Dialog positioniert werden kann. Obwohl in der Maske im Dialogeditior "Titel" steht ist die Property "Label".

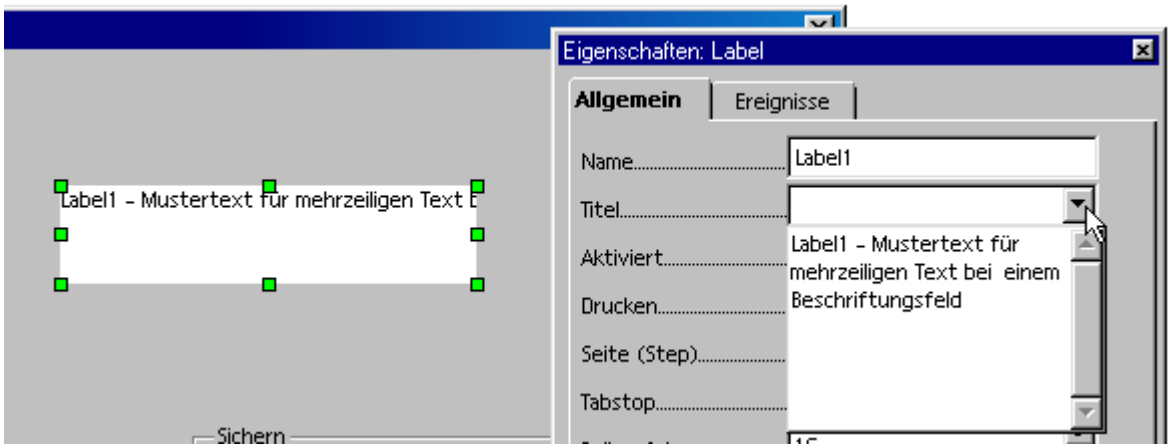
myctrl.model.label="Hallo"

Mehrzeilig

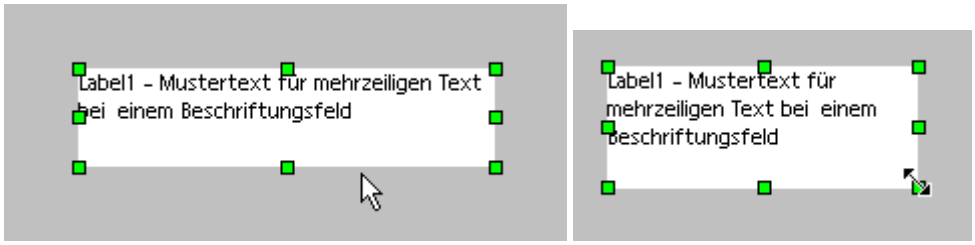
Als Standard werden Texte einzeilig angezeigt. Es gibt zwei Möglichkeiten mehrzeiligen Text zu erzeugen. Innerhalb des Dialogeditors kann man bei Eigenschaften den Schalter "Mehrzeilig" auf "Ja" setzen. Dabei steuert aber StarOffice innerhalb den Dialoges den Umbruch. Entsprechend groß muß dann das Label sein. Oder bei der Eingabe des Textes fügt man einen Zeilenumbruch ein.

Hierzu Beispiele

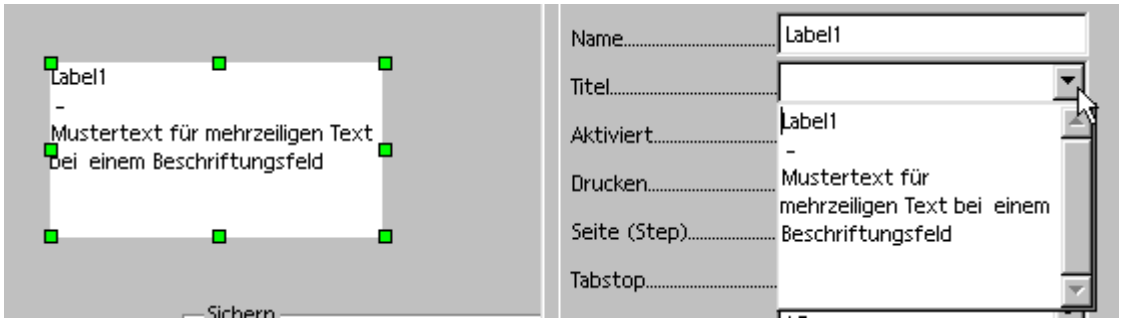
Ein Text enthält einen längeren Satz. Dieser wird in der Eingabebox in den Kontrolleigenschaften als einzeiliger Text eingeben, und erscheint dann im Dialog einzeilig und wird am Rand abgebrochen.



Setzt man nun den Schalter "Mehrzeilig" auf Ja, wird der Text umgebrochen dargestellt. Den Umbruch legt aber Starbasic und die Größe des Feldes fest.



Möchte man einen Umbruch an einer oder mehreren festen Stellen, muß man innerhalb der Eingabebox zwischen den Zeilen einen Umbruch mit Shift-Enter erzeugen.



Die Option "Mehrzeilig bezieht sich immer auf Text der längere wie das Textfeld ist. Um also mehrzeiligen Text zu schreiben ohne den Schalter zu setzten langt auch der harte Umbruch mit Shift-Enter.

Mit Starbasic geht das Einstellen von Mehrzeilig mit der Property "Multline", die auf true oder false gesetzt werden kann.

myctrl.model.Multiline=true / false

Um mehrzeiligen Text zu erzeugen, muß man die einzelnen Zeilen mit dem Umbruchcode Chr$(13)

verbinden.

myctrl.model.label="Hallo" + Chr$(13) + "Welt"

oder auch ohne Dollarzeichen

myctrl.model.label="Hallo" + Chr(13) + "Welt"

4.4.4 Welches sind die Besonderheiten von Checkboxen?

Ein Checkbox ist für die Abfrage von Ja/Nein-Werten gedacht. Sie sind damit eine Kurzfassung von zwei Optionbuttons. In Starbasic ist noch zusätzlicher Modus, Schwebezustand, möglich. Dieser kann zum Einsatz kommen wenn der Ja/Nein-Status nicht eindeutig ist. Von der Anwendung dieser Funktion ist aber abzuraten, da die meisten Anwender diesen Zustand nicht erkennen und zuordnen können.

Titel

Legt den Text der Checkbox fest.

MyControl.Model.Label = "Text"

Dreifacher Status

Legt fest, ob außer Ja/Nein noch unbestimmt als Eingabe zulässig ist. In Starbasic ist der Status mit MyControl.Model.TriState zusetzten. In der API Dokumentation steht an dieser Stelle enableTriState, das ist aber falsch.

MyControl.Model.TriState = true / false

Status

Hier wird der aktive Status ausgewählt. Im Dialogeditor wird ausgewählt, nicht ausgewählt und , wenn

Dreifach Status erlaubt, unbestimmt angeboten.

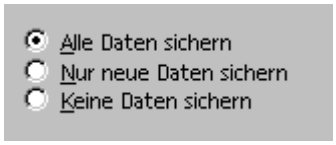
In Starbasic wird die Eigenschaft state verwendet. Hierbei gehen die Werte 0 für nicht ausgewählt, 1 für ausgewählt, oder 2 für unbestimmt.

Myctrl.model.State=1

4.4.5 Welche Besonderheiten gibt es bei Optionbuttons?

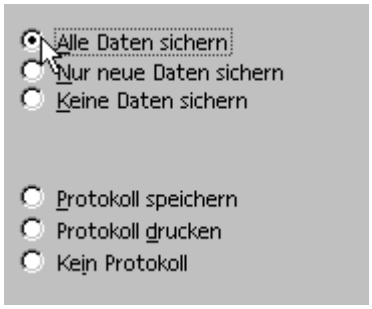
OptionButton

Ein OptionButton kann eigentlich immer nur zusammen mit mindestens einem weiteren Optionbutton verwendet werden. Mehrere Optionbuttons werden zusammengefasst um zwischen ihnen eine eindeutige Auswahl zu ermöglichen.

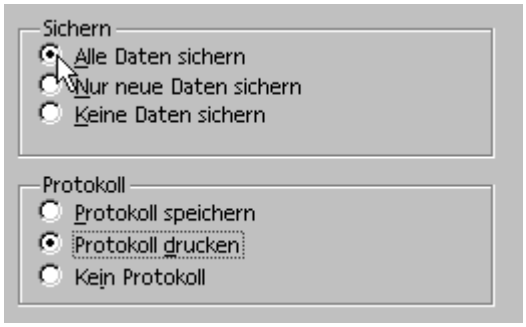


Innerhalb eines Dialoges muss folgendes beachtet werden um Optionbutton zusammen zufassen. Im Dialog gibt es kein Gruppierungselement das eine Gruppe von Kontrollfeldern zusammenfasst. Die Gruppierung erfolgt über die Reihenfolge. Alle in der Reihenfolge aufeinander folgenden Optionbutton werden zusammengruppiert. Durch ein anderes Kontrollfeld kann man diese Gruppierung unterbrechen. Die Reihenfolge der Kontrollfelder kann man in dem Eigenschaft "Aktivierungsreihenfolge" sehen und einstellen.

Hier ein Beispiel. Mit zwei geplanten Gruppen von Optionbutton. Die Reihenfolge der Optionbutton ist eins bis sechs. Dadurch ist die Auswahl nur über eine dieser sechs Optionen möglich.



Um dieser jetzt in zwei Gruppen aufzuteilen muß in der Reihenfolge eine viertes Kontrollfeld eingefügt werden. Hierfür bietet sich der Rahmen an. Dieser stellt so wohl ein zusätzliches Kontrollfeld bereit, wie auch eine optische Trennung der Gruppen. Dieser erhält dann in der Reihenfolge die Position nach dem letzten Optionbutton in der Gruppe.



Achtung: Wenn bei der Dialoggestaltung das Kontrollfeld zur Trennung in Gruppen nach allen Optionbuttons danach eingefügt wird, kann man im Dialogeditor den Status nicht entsprechend der geplanten Gruppierung aktivieren. Dann reagiert der Dialogeditor so, als wären alle Optionbuttons zusammen, obwohl sie es laut Reihenfolge nicht sind.

Titel

Legt den Text des Optionbuttons fest.

MyControl.Model.Label = "Text"

Status

Mit dieser Eigenschaft wird festgelegt ob der Schalter ausgewählt werden kann oder nicht. Da innerhalb einer Gruppe von Optionbutton nur einer den Status ausgewählt haben kann, wird dafür der zuletzt Festgelegte verwendet.

MyControl.State = true/false

Auswahl=MyControl.State

'oder

MyControl.Model.State = true/false

Auswahl=MyControl.Model.State

4.4.6 Kann man Kontrollfelder unsichtbar machen?

Ja, und zwar mit der Eigenschaft visible. Diese muß auf false gesetzt werden. Diese Eigenschaft steht in der IDE nicht zur Verfügung.
Sie steht auch nicht unter Model zur Verfügung, sondern direkt als Property eines Kontrollfeldes.

```
DialogLibraries.LoadLibrary( "Standard" )
MyDlg = CreateUnoDialog( DialogLibraries.Standard.Dialog1 )
MyCtrl=MyDlg.getControl("CommandButton1")
MyCtrl.Visible=false
```

Diese Funktion ist nicht für alle Kontrollfelder von mir geprüft worden. Ich denke aber sie wird immer dabei sein.

Was aber nicht einfach geht ist den Status eines Kontrollfeldes zu prüfen. Weder "Visible" noch "Enabled" kann man direkt abfragen.

Enabled kann man über das Model des Kontrollfeldes abfragen:

```
if ctrl.model.enabled then
```

Falls man diese Möglichkeit unbedingt auch für "Visible" brauchr kann man sich dann eine kleine Umgehung bauen.

```
ctrl=dlg.getControl("ComboBox1")
  if ctrl.model.enabled then
    ctrl.visible=false
    ctrl.model.enabled=false
    else
    ctrl.visible=true
    ctrl.model.enabled=true
   end if
```

4.4.7 Welche Besonderheiten gibt es bei Comboboxen?

Das Kombinationsfeld ist der Listbox ähnlich. Sie stellt auch eine Liste zur Auswahl zur Verfügung. In dem Textfeld oberhalb der Liste kann der Anwender eine neue Eingabe tätigen. Außerdem ist beim Kombinationsfeld keine Mehrfachauswahl möglich.

Auch besteht nicht die Möglichkeit einen Eintrag mit select auszuwählen. Die Auswahl für den ersten Eintrag erfolgt mit setText(string). Das Ergebnis wird mit getText ausgelesen. Man kann den ausgewählten Wert auch mit getSelection auslesen. Dann werden aber freie Eingaben ingnoriert und der Rückgabewert ist leer.

Aufpassen muss man auch mit der Eigenschaft readonly (*myCtrl.model.readonly=true/false).* Wird Readonly auf True gesetzt und es steht ein Text in der Auswahl der ComboBox kann der Anwender keine weitere Auswahl treffen. Bei leere Auswahl kann er eine aus der Liste auswählen, aber keine eigene eingeben.

Eigenschaften und Methoden:

MaxTextLen

Im Gegensatz zu einem Textfeld steht in der IDE nicht die Möglichkeit zur Verfügung die maximale Textlänge für das Eingabefeld der Kombobox festzulegen. Dies muß im Programm erfolgen:

Entweder mit

myCtrl.setMaxTextLen(x)

oder mit

myCtrl.MaxTextLen=x

additem(Item, Pos)

Fügt einen Eintrag an der Position hinzu.

myctrl.additem("Test",1)

oder

strEintrag = "Test"

myctrl.additem(strEintrag,1)

Additems(ItemArray,Pos)

Fügt die Einträge des Array ItemArray ab der Stelle Pos in die Liste ein.

Dim strEintraege(2)

strEintraege(0)="Test0"

strEintraege(1)="Test1"

strEintraege(2)="Test2"

myctrl.additems(strEintraege(),1)

removeitems(Pos,Count)

Entfernt die Anzahl (Count) von Einträgen ab der Position (Pos).

myCtrl.removeitems(0,4)

Achtung! Mit removeitems wird nicht der Eintrag im Textfeld der Combobox gelöscht. Dieser muß extra gelöscht werden.

myCtrl.text=""

itemcount

Liest die Anzahl der Einträge aus.

anzahl = myctrl.itemcount

setText

Mit settext kann ein String in das Eingabefeld eingetragen werden. Dieser erscheint dann nicht in der Liste.

myCtrl.setText("Testeintrag")

getText

Mit GetText wird die Auswahl des Anwenders aus dem Textfeld ausgewählt.

ergebnis = myctrl.getText

Da eine Kombobox auch ohne Auswahl geschlossen werden, kann sollte man daran denken, auf einen leeren Eintrag zu prüfen. Vorher sollte man entscheiden was in einem solchen Fall zu passieren hat. Eventuell ist eine Meldung möglich, dass eine Auswahl getroffen werden muß.

Model.StringItemList

Gibt eine Liste mit allen Einträgen zurück.

```
Dim Ergebnis()
Ergebnis() = myCtrl.model.StringItemList()
for i=0 to ubound(Ergebnis())
  msgbox Ergebnis(i)
next i
  for i=0 to ubound(myCtrl.model.StringItemList())
  msgbox myctrl.model.StringItemList(i)
next i
```

4.4.8 Kann man Listboxeneinträge über die Position selektieren?

Mit dem Befehl selectitempos(integer, boolean), der direkt unter dem Objekt des Kontrolls ist.

```
DialogLibraries.LoadLibrary( "Standard" )
MyDlg = CreateUnoDialog( DialogLibraries.Standard.Dialog1 )
MyCtrl=MyDlg.getControl("ListBox1")
myctrl.selectitempos(0,true)
```

4.4.9 Welches sind die Besonderheiten von Listboxen?

Es folgen die Methoden und Properties die während der Laufzeit des Programms zur Verfügung stehen. Diese sind bei Listboxen viel wichtiger als bei den anderen Kontrollfeldern, weil bei Listboxen ja Informationen ausgewertet werden müssen, die der Anwender in der Listbox ausgewählt hat.

additem(Item, Pos)

Fügt einen Eintrag an der Position hinzu.

myctrl.additem("Test",1)

oder

strEintrag = "Test"

myctrl.additem(strEintrag,1)

Additems(ItemArray,Pos)

Fügt die Einträge des Array ItemArray ab der Stelle Pos in die Liste ein.

Dim strEintraege(2)

strEintraege(0)="Test0"

strEintraege(1)="Test1"

strEintraege(2)="Test2"

myctrl.additems(strEintraege(),1)

removeitems(Pos,Count)

Entfernt die Anzahl (Count) von Einträgen ab der Position (Pos).

myCtrl.removeitems(0,4)

selectitem(Item,SelectMode)

Setzt den Eintrag "Item", das heißt den Eintrag der dem String "Item" entspricht, auf selektiert oder nicht selektiert. Wenn der Befehl mehrmals erfolgt, werden auch mehrere markiert. Wenn der der Modus Mehrfachselektion aus ist, werden alle anderen Markierungen bei einem Klick in die Listbox gelöscht.

Wenn der Eintrag "Item" nicht vorhanden ist erfolgt kein Fehler.

myCtrl.SelectItem("Test2",True)

itemcount

Liest die Anzahl der Einträge aus.

anzahl = myctrl.itemcount

selecteditem

Übergibt den selektierten Text in einem String.

mySelect = myCtrl.SelectedItem

selecteditems

Übergibt, bei aktiver Mehrfachselektion, die ausgewählten Werte als ein Array von Strings. Ausgelesen werden können die Strings mit der Ubound-Funktion.

```
Dim Ergebnis()
Ergebnis = myCtrl.selecteditems
  for i=0 to ubound(Ergebnis())
    msgbox Ergebnis(i)
  next i
```

Man kann auch direkt auf dieses Array zugreifen.

```
  for i=0 to ubound(myCtrl.SelectedItems())
    msgbox myctrl.selecteditems(i)
  next i
```

Achtung: Nicht mit der Methode unter Model verwechseln. Model.selecteditems gibt ein Array der Positionen wieder.

selectitempos(pos, selected))

Markiert einen Eintrag nach der Position.

myctrl.selectitempos(2,true)

selectitemspos(array,selected)

Markiert mehrere Einträge nach der Position. (Nur wenn Multiselection aktiviert ist)

```
dim selectarray(1)
selectarray(0)=3
selectarray(1)=5
myctrl.selectitemspos(selectarray(),true)
```

selecteditempos

Übergibt die Position der Selektion.

```
Position =  myctrl.selecteditempos
```

selecteditemspos

Übergibt bei Mehrfachselektion die ausgewählten Position als Array.Hier wird wieder die uBound-Funktion zum Auslesen verwendet.

```
for i=0 to ubound(myCtrl.SelectedItemspos())
   msgbox myctrl.selecteditemspos(i)
next i
```

Das gleiche Ergebnis liefert Model.SelectedItems.

Model.StringItemList

Gibt eine Liste mit allen Einträgen zurück.

```
Dim Ergebnis()
Ergebnis() = myCtrl.model.StringItemList()
for i=0 to ubound(Ergebnis())
  msgbox Ergebnis(i)
next i
```

Auch auf dieses Array kann man direkt zugreifen.

```
for i=0 to ubound(myCtrl.model.StringItemList())
```

msgbox myctrl.model.StringItemList(i)

next i

4.4.10 Wie kann man die Textlänge des Eingabefeldes einer ComboBox einschränken?

Im Gegensatz zu einem Textfeld, steht in der IDE nicht die Möglichkeit zur Verfügung die maximale Textlänge festzulegen. Dies muß im Programm erfolgen:

Entweder mit

myCtrl.setMaxTextLen(x)

oder mit

myCtrl.MaxTextLen=x

4.4.11 Welches sind Besonderheiten von Textfeldern?

Text

Festlegen des Textes

MyCtrl.Model.Text = "Hallo" oder MyCtrl.Text = "Hallo"

Abfragen des Textes

MyText = MyCtrl.Model.Text oder MyText = MyCtrl.Text

SelectedText

Ermöglicht den Zugriff auf den markierten Text eines Eingabefeldes. Steht direkt unter dem Objekt zur Verfügung.

MyText = MyCtrl.SelectedText

Max. Textlänge

Mit dieser Einstellung kann man die maximale Eingabelänge des Textes einschränken. Die maximal Länge beträgt 65535 Zeichen.

myctrl.model.maxtextlen = 10

Ausrichtung

Bestimmt ob der Text im Eingabefeld links,rechts oder zentriert angezeigt wird.

mycrtl.model.align = 1

Wobei gilt:

0 = linksbündig (Standardvorgabe)

1 = zentriert

2 = rechtsbündig

Rahmen

Legt die Eigenschaft des Rahmens fest. Es gibt drei verschiedene: 3D-Look, Flach oder ohne Rahmen.

Starbasic: myctrl.model.border = 1

Wobei gilt:

0 = Ohne Rahmen

1 = 3D-Look (Standardvorgabe)

2 = Flach

Zeichen bei Passwörtern

Es wird das Zeichen, das im Eingabefeld für eingegebene Zeichen erscheint, eingegeben. Dies dient zum Beispiel der Passwortverschlüsselung.

In der IDE können Buchstaben eingeben werden. In Starbasic muss der Charwert des Zeichens übergeben werden. Zum Beispiel 42 für das Sternchen.

Die Eigenschaft ist entgegen der Dokumentation eine Integer-Variable und keine String-Variable.

MyCrtl.Model.EchoChar = 42

Mehrzeilig

Bestimmt ob der Text mehrzeilig oder einzeilig eingegeben werden kann. Ist das Textfeld auf mehrzeilig nein gesetzt wird der überstehende Teil eines Satzes nur mit dem Cursor sichtbar.

MyCtrl.Model.MultiLine = true / false

Horizontaler Scrollbalken und vertikaler Scrollbalken

Bestimmt ob Scrollalken horizontal und/oder vertikal sichtbar wird, um in Texten die größer als das Textfeld sind zu navigieren.

Diese lassen sich nur aktivieren wenn die Eigenschaft des Textfeldes bei Mehrzeilig auf True steht.

Sollte eigentlich in Starbasic mit folgenden Befehlen gehen:

Für den vertikalen Balken: MyCtrl.Model.VScroll = true / false

Für den horizontalen Balken: MyCtrl.Model.HScroll = true / false

Stimmt aber nicht.

Textcolor

Eine zusätzliche Eigenschaften ist Textcolor, mit der die Textfarbe eingestellt werden kann. Dies geht nicht in der IDE, sondern nur vor dem Öffnen des Dialoges im Programm.

MyCtrl=MyDlg.getControl("Label1")

myctrl.model.textcolor = 255

4.4.12 Kann man bei Kontrollfeldern die Schriftfarbe ändern?

Man kann auch die Schriftfarbe von Labeltext und Textfeldern ändern, nicht nur die Hintergrundfarbe.
Dazu ist die Eigenschaft Textcolor zuständig. Diese kann man innerhalb der IDE über die Eigenschaft "Schrift" und dann auf der zweiten Seite "Zeicheneffekete" einstellen.

MyCtrl=MyDlg.getControl("Label1")

myctrl.model.textcolor = 255 'blau

Es geht auf jeden Fall mit

Text, Textfeldern, Listboxen, CombiBoxen.

Ansonsten muß man nur schauen, ob das Kontrollfeld die Eigenschaft "Textcolor" unterstützt.

4.4.13 Wie kann man Tastenkürzel mit den Kontrollfeldern verbinden?

Innerhalb von Dialogen kann man Kontrollfelder auch mit den Tasten anspringen. Dies geht mit ALT+dem unterstrichenen Buchstaben.
Um den Buchstaben festzulegen muss man den gewünschten Buchstaben die Tilde (~) voranstellen. Z.B. Neue ~Datei.

5 Objekte, Methoden, Eigenschaften

5.1 Wie kann man die Objekte der API verwenden?

Um die angebotenen Objekte (Services) und deren Methoden zu verwenden, müssen diese einem eigenem

Objekt zu gewiesen werden. Dies erfolgt mit createunoservice.

Dim MyObjekt as Object

MyObjekt = createunoservice("com.sun.star.ui.dialogs.FilePicker")

An dieser Stelle steht im Developerguide, dass man den Typ Object nicht angeben soll, da dieser für StarBasic-Objekte gedacht ist.

Also:

Dim MyObjekt

MyObjekt = createunoservice("com.sun.star.ui.dialogs.FilePicker")

In allen mitgelieferten Beispielen wird aber der Typ mit angegeben.

Wichtig ist hierbei darauf zu achten, dass die Groß- und Kleinschreibung stimmt. Hier z.B. FilePicker und nicht filepicker.

Starbasic gibt bei falscher Schreibweise keine Fehlermeldung aus. Das Objekt wird dann nicht erzeugt und die Fehlermeldung kommt dann beim ersten Zugriff auf das Objekt ("Objektvariable nicht belegt").

Danach kann man auf die Methoden und Eigenschaften des Objektes zu greifen.

MyObjekt.Methode(Optionale Parameter)

MyObjekt.Eigenschaft = XXXX

Siehe auch:

Wie kann man sich Methoden und Eigenschaften anzeigen lassen?

Wie geht man mit Eigenschaften des Typ "stuct" um?

5.2 Wie geht man mit Eigenschaften/Properties um, die den Typ "struct" haben?

Properties mit dem Typ "struct" haben ein eigenes Objekt das verwendet und zugewiesen werden muss.

Ein Beispiel:

Die mögliche Property Zeichensatz (fontdescriptor) eines Kontrollfeldes hat den Typ Struct. Der Structtyp dazu ist com.sun.star.awt.fontdescriptor. Die Struktur muss als Objekt vorher erzeugt werden, mit den Parametern versehen werden und dann der eigentlichen Property übergeben werden.

'Anlegen des Strukturobjektes

```
Dim myFont As New com.sun.star.awt.FontDescriptor
'Zuweisen der Werte
MyFont.Name = "Arial"
MyFont.StyleName = "Fett"
MyFont.Height = 8
'Übergabe an Kontrollfeld
MyControl.Model.FontDescriptor=MyFont
```

Die Liste der möglichen Structs und ihrer Eigenschaften findet sich in der API Dokumentation bei dem jeweiligen Service.

5.3 Wie kann man Properties an eine Funktion übergeben?

Starbasic benötigt manchmal bei einem Aufruf einer Funktion ein Array mit bestimmten Properties. Als Beispiel verwende ich hier loadcomponentfromurl.
Der letzte Parameter bei loadcomponentfromurl besteht aus solch einem Array.

Wenn man in diesem Array nichts zu übergeben hat, kann man einen leeren Dummy verwenden. Dieser muß aber unbedingt deklariert werden.

```
Sub NeueDatei
Dim MyProp()
stardesktop.loadcomponentfromurl("private:factory/swriter","_blank",0,myProp())
End Sub
```

Wenn man aber Parameter mit MyProp übergeben muß, kann man das Struct ProbertieValues verwenden.

```
Sub TemplateOeffnen
Dim myProps(0) as New com.sun.star.beans.PropertyValue
  myProps(0).Name="AsTemplate"
  myProps(0).Value=true
  stardesktop.loadcomponentfromurl("file:///f:/wbmakros.sxw","_blank",0,myProps())
End Sub
```

Entsprechend der Anzahl der Parameter wird das Array dimensioniert.

```
Dim myProps(1) as New com.sun.star.beans.PropertyValue
myProps(0).Name="AsTemplate"
myProps(0).Value=false
```

myProps(1).Name="ReadOnly"

myProps(1).Value=true

Die möglichen Properties muß man aus der Dokumentation des Services auslesen.
In diesem Beispiel: com.sun.star.document.MediaDiscriptor.

5.4 Wie kann man die Methoden eines Objektes anzeigen lassen?

Mit der Eigenschaft dbg_methods

Beispiel:

odoc = thiscomponent

msgbox odoc.dbg_methods

Tipp:

Man kann sich diese Infos auch in eine Datei schreiben lassen:

Open "c:\methods.txt" for Output as #1

write #1,odoc.dbg_methods

Close #1

Oder mit der Hilfe eines mitgelierten Makros alles (Eigenschaften, Methoden und Interfaces)
in eine Textdatei schreiben lassen:

GlobalScope.BasicLibraries.LoadLibrary("Tools")

WriteDbgInfo(odoc)

5.5 Wie kann man sich Eigenschaften eines Objektes anzeigen lassen?

Mit der Methode dbg_properties

Beispiel:

odoc = thiscomponent
msgbox odoc.dbg_properties

Tipp:

Man kann sich diese Infos auch in eine Datei schreiben lassen:

Open "c:\properities.txt" for Output as #1
write #1,odoc.dbg_properties
Close #1

Oder mit der Hilfe eines mitgelierten Makros alles (Eigenschaften, Methoden und Interfaces)
in eine Textdatei schreiben lassen:

GlobalScope.BasicLibraries.LoadLibrary("Tools")
WriteDbgInfo(odoc)

5.6 Wie kann man sich die Interfaces eines Objektes anzeigen lassen?

Mit der Eigenschaft dbg_supportedinterfaces
Beispiel:

odoc = thiscomponent
msgbox odoc.dbg_supportedinterfaces

Tipp:

Man kann sich diese Infos auch in eine Datei schreiben lassen:

Open "c:\supportedinterfaces.txt" for Output as #1

write #1,odoc.dbg_supportedinterfaces

Close #1

Oder mit der Hilfe eines mitgelierten Makros alles (Eigenschaften, Methoden und Interfaces)
in eine Textdatei schreiben lassen:

GlobalScope.BasicLibraries.LoadLibrary("Tools")

WriteDbgInfo(odoc)

5.7 Wie kann man sich die unterstützten Services eines Objektes anzeigen lassen?

Mit der Propertie SupportedServiceNames.

Dies erzeugt ein Array das man mit der ubound-Funktion auslesen kann.

for i=0 to ubound(objekt.supportedservicenames())

Msgbox objekt.supportedservicenames(i)

next i

Oder sich diese in eine Datei schreiben.

Open "c:\supportedservices.txt" for Output as #1

for i=0 to ubound(objekt.supportedservicenames())

write #1,objekt.supportedservicenames(i)

next i

Close #1

5.8 Wie kann man überprüfen ob bei einem Service ein Interface zur Verfügung steht?

Dazu gibt es die Funktion hasunointerfaces.

Mit dieser kann man prüfen, ob ein Interface unterstützt wird.

hasunointerfaces(objekt, interface1, interface2)

Der Rückgabewert ist Wahr oder Falsch. Werden mehrere Interfaces abgefragt müssen alle unterstützt werden.

berror= hasunointerfaces(odoc,"com.sun.star.frame.XController")

5.9 Wie kann man bei Objekten Structs manipulieren?

Wenn ein Objekt als Property ein Struct hat kann man dieses nicht direkt ändern. Um es zu ändern muß man es in eine Kopie auslesen und diese Kopie bearbeiten und danach zurückschreiben.

Dazu ein Beispiel:

Siehe auch -> Wie geht man Eigenschaften/Properties um, die den Typ "struct" haben?

```
Dim myFont As New com.sun.star.awt.FontDescriptor
'Auslesen der aktuellen Werte
MyFont =MyControl.Model.FontDescriptor
'Zuweisen der Werte
MyFont.Name = "Arial"
MyFont.StyleName = "Fett"
MyFont.Height = 8
'Übergabe an Kontrollfeld
MyControl.Model.FontDescriptor=MyFont
```

Nur so werden die gewünschten Werte neu gesetzt. Auf diese Weise werden alle vorhandenen Properties, bis auf die geänderten, beibehalten.

MyControl.Model.FontDescriptor.StyleName="Fett" führt zu keiner Änderung.

5.10 Welche Runtime-Funktionen gibt es speziell für den Umgang mit API?

Für den Umgang mit der API werden einige Runtime-Funktionen angeboten. In der Onlinehilfe sind sie nur zu finden, wenn man Ihren Namen kennt. Einen eigenen Bereich dafür gibt es nicht. Die Details sind in der alphabetischen Liste in der Onlinehilfe zu finden.

Im Einzelnen sind es folgende:

CreateUnoDialog

Erzeugt ein Basic-Uno-Objekt, das ein Uno-Dialog-Control zur Laufzeit des Basics repräsentiert.

Siehe -> Wie öffnet man einen Dialog?

CreateUnoListener

Instanziert einen Listener.

Bei vielen Uno-Schnittstellen kann man Listener auf einer speziellen Listener-Schnittstelle registrieren. So können diese den Eintritt bestimmter Ereignisse überwachen und daraufhin die jeweilige Listener-Methode aufrufen. Die Funktion CreateUnoListener wartet auf die aufgerufene Listener-Schnittstelle und übergibt der Schnittstelle dann ein von dieser unterstütztes Objekt. Dieses Objekt wird dann an die Methode zum Registrieren des Listeners übergeben.

CreateUnoService

Instanziert einen Uno service am ProcessServiceManager.

oService = CreateUnoService(Uno service name)

Siehe > Wie kann die Objekte der API (UNO) verwenden?

CreateUnoStruct

Erstellt eine Instanz eines Uno-Strukturtyps.

Zu bevorzugen ist angeblich jedoch eine Anweisung nach folgendem Muster:

Dim oStruct as new com.sun.star.beans.Property

Syntax:

oStruct = CreateUnoStruct(Uno_Typname)

Beispiel:

oStruct = CreateUnoStruct("com.sun.star.beans.Property")

HasUnoInterfaces

Ermittelt, ob ein Basic-Uno-Objekt bestimmte Uno-Interfaces unterstützt.

Gibt "True" zurück, wenn alle angegebenen Uno-Schnittstellen unterstützt werden, ansonsten "False".

Syntax:

HasUnoInterfaces(oTest, Uno-Schnittstellenname 1 [, Uno-Schnittstellenname 2, ...])

Rückgabewert:

Bool

Parameter:

oTest:Das zu prüfende Basic Uno-Objekt.

Uno-Schnittstellenname:Eine Liste der Uno-Schnittstellennamen.

Beispiel:

bHas = HasUnoInterfaces(oTest, "com.sun.star.beans.XIntrospection")

IsUnoStruct

Gibt "True" zurück wenn es sich um ein Struct handelt.

IsUnoStruct(Uno type name)

bIsStruct = IsUnoStruct("com.sun.star.beans.Property")

EqualUnoObjects

Gibt "True" zurück, wenn die beiden angegebenen Basic Uno-Objekte dieselbe Uno-Objektinstanz darstellen.

EqualUnoObjects(oObj1, oObj2)

Beispiel:

```
// Kopie von Objekten -> gleiche Instanz
oIntrospection = CreateUnoService( "com.sun.star.beans.Introspection" )
oIntro2 = oIntrospection
print EqualUnoObjects( oIntrospection, oIntro2 )
// Kopie von Structs als value -> neue Instanz
Dim Struct1 as new com.sun.star.beans.Property
Struct2 = Struct1
print EqualUnoObjects( Struct1, Struct2 )
```

5.11 Wie sieht die Objektstruktur der API-Elemente aus?

Ich verwende hier das Wort Elemente für die Typen/Objekte der API.
Die möglichen Elemente der API unterteilen sich in folgende:

Module:

Hierbei handelt sich um Gruppierungen der anderen Elemente. Diese Gruppierungen sind nur für die bessere Übersicht gedacht und haben für die Programmierung in StarBasic keinerlei Bedeutung. Außer natürlich durch den Namen im Objekt. Achten muß man darauf, dass sich die Module innerhalb der

Dokumentation nicht immer als hilfreich erweisen. Durch die Möglichkeit eines Service eigene Services zu verwenden, kommt es leider auch zu einigen nicht immer sofort ersichtlichen Querverbindungen.

Ein einfaches Beispiel dazu:

Im Modul "Text" befinden sich der Services "TableRow". Dieser hat dann aber ein Interface aus dem Modul "Table".

Services:

Sind die eigentlichen Objekte von Starbasic. Sie können Interfaces, Structs, Enums, Constant Groups, Typdef, Exceptions Properties beinhalten. Und was besonders wichtig ist: Sie können ebenfalls selber Services haben.

Interfaces:

Interfaces sind die Schnittstellen und stellen den Service die Methoden zur Verfügung. Interfaces haben ausschließlich Methoden.

Properties:

Sind die Eigenschaften die Services haben können. Sie kommen auch nur bei den Services vor. Es gibt zwei besonders Typen von Properties: Enum und Struct.

Ansonsten handelt es sich um Typen wie String, Integer (long oder short) oder Boolean.

Methods:

Sind Methoden die ausschließlich von Interfaces bereitgestellt werden. Wenn es sich um Funktionen mit einem Rückgabewert handelt, kann es sich dabei um die Typen: String, Long, Short, Boolean, Void oder um ein Objekt (Service) handeln. Diese Typen können auch als Array übergeben werden.

Structs:

Structs sind gruppierte Properties. Sie können nur gemeinsam an eine Propertie übergeben. Dabei kann man einzelne Werte festlegen oder Standardwerte automatisch verwenden. Siehe auch:

Wie geht man Eigenschaften/Properties um, die den Typ "struct" haben?

Exception:

Ist der Typ der mögliche Fehlermeldungen enthält oder übergibt den betroffenen Service oder das Interface als Meldung.

Enum:

Enums sind Konstanten die für einen bestimmten Kontext gebündelt worden sind. Zum Beispiel: com.sun.star.style.ParagraphAdjust. Unterhalb von ParagraphAdjust werden die Optionen, denen man einen Absatz einstellen kann, gebündelt (Linksbündig, Blocksatz, Rechtsbündig, Zentriert). Der Übergabewert ist nicht dokumentiert, aber eigentlich eine Aufzählung (eng. Enumeration) in fortlaufender Folge. Bei einigen Services kann man daher auch mit dem passenden Integerwert die Einstellung vornehmen. Da die Dokumentation fehlt, muß man den vollen Namen als Konstante verwenden. Hier also "com.sun.star.style.ParagraphAdjust.LEFT" für linksbündig. Achtung: Auf Groß- und Kleinschreibung achten. Einige dieser Enums sind mit großer Vorsicht zu verwenden, da nicht alle dokumentierten Elemente von Enums funktionieren. "com.sun.star.style.ParagraphAdjust.STRETCH" führt zum Beispiel, entgegen der

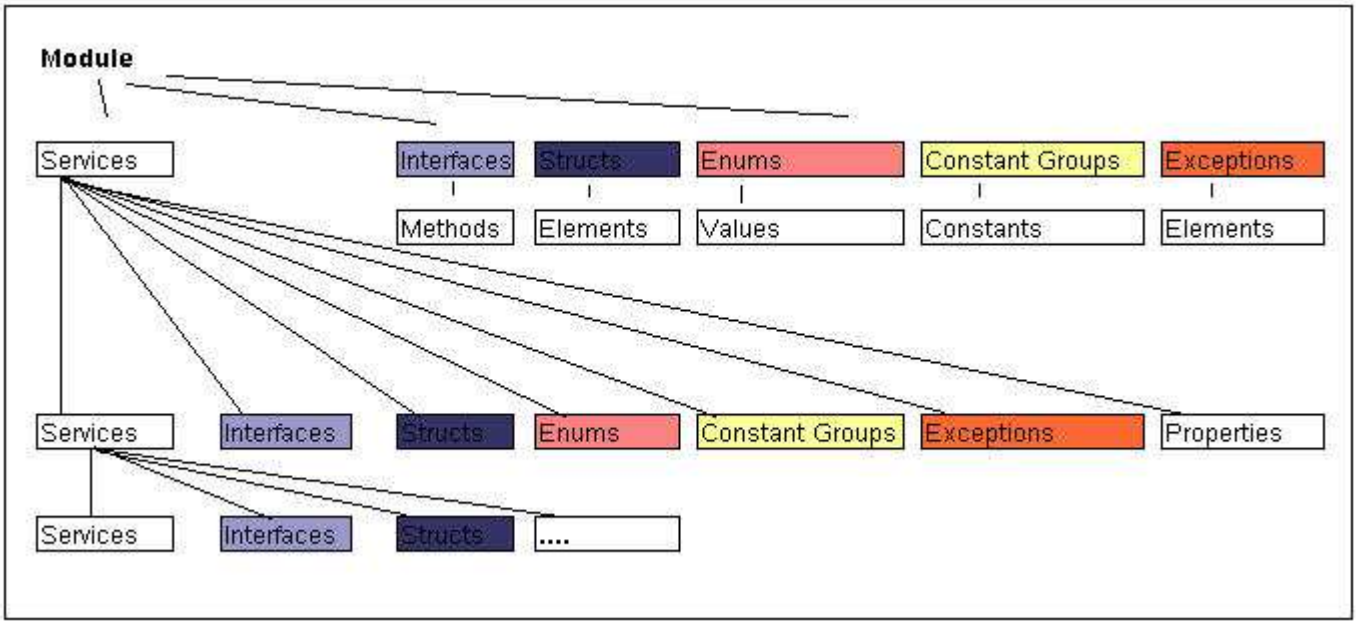
Dokumentation, auch zu einem linksbündigen Absatz.

Sie auch: Wie geht man Properties um die den Typ "Enum" haben?

Constant Groups:

Sind Gruppierungen von Konstanten. Hier kann der Namen auch nur komplett verwendet werden. Aber in der Dokumentation stehen die Werte für diese Konstanten, die man alternativ verwenden kann. Auch hier gilt, nicht alle sind richtig dokumentiert oder funktionieren richtig.

Zum Abschluß eine nicht ganz vollständige Übersicht:



5.12 Wie geht man mit Properties um, die den Typ "Enum" haben?

Properties des Types Enum haben eine Gruppe von Konstanten als mögliche Parameter.
Diese Konstanten müssen mit ihren kompletten Api-Namen verwendet werden. Die Werte zu diesen Konstanten sind nicht dokumentiert.

Als Beispiel verwende ich hier die Textausrichtung eines Absatzes.

Die dazugehörige Propertie ist paraadjust. Das Enum dazu ist com.sun.star.style.ParagraphAdjust, mit den Ausprägungen LEFT, RIGHT, CENTER, BLOCK und STRETCH. Die Übergabe erfolgt durch das Anhängen des gewünschten Parameters.

Zum Beispiel: com.sun.star.style.ParagraphAdjust.LEFT

```
oDocument=thisComponent
oText = oDocument.Text
oCursor = oText.createTextCursor()
ocursor.paraadjust=com.sun.star.style.ParagraphAdjust.LEFT
```

Achtung: Leider sind die Enums innerhalb der API nicht immer der Dokumentation entsprechend. So können zwei Probleme auftreten:

1. Die Konstante führt nicht zu dem gewünschten Ergebnis (Dies ist hier bei STRETCH der Fall)

oder

2. Die Werte lassen sich nicht korrekt auslesen:

```
ocursor.paraadjust=com.sun.star.style.ParagraphAdjust.STRETCH
if ocursor.paraadjust=com.sun.star.style.ParagraphAdjust.STRETCH then msgbox "OK"
```

führt nicht zu der Meldung "OK", obwohl es das müßte.

Dagegen funktioniert:

```
ocursor.paraadjust=com.sun.star.style.ParagraphAdjust.LEFT
if ocursor.paraadjust=com.sun.star.style.ParagraphAdjust.LEFT then msgbox "OK"
```

Sollten es also im Zusammenhang mit Enums zu Fehler kommen, ist eine gründliche Prüfung nötig. Und Geduld!

5.13 Wie geht man mit Property-Arrays um?

Der Titel ist nicht besonders gut, aber ...
Es gibt Eigenschaften die über in einem Array des Typ com.sun.star.beans.PropertyValue festgelegt werden.

Diese können in der Regel mit getXXX und setXXX augelesen und gesetzt werden.

Als Beispiel verwende ich hier "Printer" unterhalb eines Dokumentes.

Es gibt zwei Methoden und eine Property die sich um Printer drehen.

Die Property ist Printer, die Methoden sind getPrinter und setPrinter

Der Zugriff auf die einzelnen Werte der Property geht direkt über Printer oder über ein ausgelesenes Array.

Direkter Zugriff:

```
oDoc = thiscomponent
msgbox oDoc.Printer(0).name
msgbox oDoc.Printer(0).value
```

oder als Schleife

```
for i=1 to ubound(odoc.printer())
  msgbox oDoc.Printer(i).name
  msgbox oDoc.Printer(i).value
next
```

Auf diese Weis können die Werte auch gesetzt werden.

```
  msgbox oDoc.Printer(0).value=1
```

Der Name darf natürlich nicht geändert werden.

Bei dieser Art muß man natürlich den Index der zu ändernden Property kennen. Diese kann man entweder der Dokumentation entnehmen oder mit einer Funktion auslesen. Siehe Tools->GetPropertyValAndInd

Der zweite Weg geht über auslesen und zurückschreiben mit den Methoden.

```
myProps=oDoc.getPrinter
for i=1 to ubound(myProp())
  msgbox myProp(i).name
  msgbox myProp(i).value
next
```

Analog dann einen neuen Wert zuweisen und zurück schreiben.

```
myProp(i).value=0
odoc.setPrinter(myProp())
```

Da es aber Propertys gibt, die sehr umfangreich sind, langt es auch nur die zu ändernden Property neu zu setzen.

Dazu muß man ein Array des Typ com.sun.star.beans.PropertyValue vorher erzeugen. Danach kann man einzelne Propertys neu setzen. Die Anderen bleiben dabei die anderen Propertys mit ihrem alten Wert erhalten.

```
Dim newProps(0) as New com.sun.star.beans.PropertyValue
newProps(0).Name="Name"
newProps(0).value="Neuer Drucker"
odoc.setprinter(newProps())
```

5.14 Was hat es mit createInstance auf sich?

Der Service XMultiServiceFactory ist ein wesentlicher Service für die Arbeit mit API-Objekten. XMultiServiceFactory stellt drei Funktionen zur Verfügung:
createInstance, createInstanceWithArguments() und getAvailableServiceNames().

getAvailableServiceNames() - gibt eine Liste mit den Servicenamen zurück die mit createInstance oder createInstanceWithArguments erzeugt werden können.

createInstance - erzeugt eine neue Instanz eines Services

createInstanceWithArguments - erzeugt eine neue Instanz eines Services mit zusätzlichen Parametern die für den Service gebraucht werden.

Um zum Beispiel ein neues Seitenformat in einem Textdokument zu erzeugen muss eine neue Instanz erzeugen und diese einfügen. Nur das Erzeugen einer Instanz reicht nicht aus. Das neue Objekt muss auch an die passende Stelle eingefügt werden. Die Metoden zum Einfügen des neuen Objektes/Instanz hängen vom Typ ab.

newstyle=doc.createinstance("com.sun.star.style.PageStyle")

pagestyles.insertbyname("MeineSeite",newstyle)

5.15 Wie geht man mit dem Enumeration-Konzept um?

Zugriff auf unbekannte Objekte

In der Regel kann man über den Namen oder den Index auf untergeordnete Objekte zugreifen. Im Beispiel mit der Tabelle geht es mit getByIndex(). Als Alternative kann auch ein Name verwendet, wenn dieser vergeben wurde. Der Befehl lautet getbyName(). Diese beiden Befehle gibt es für viele Objekte, aber nicht für alle.

Es gibt untergeordneten Objekte die weder über den Namen noch über einen Index adressierbar sind. Dafür gibt es aber die Methode createEnumeration, die ein Objekt mit einer Liste aller vorhandenen Objekte bereitstellt. Dazu kommen noch die beiden Methode hasMoreElemenst() und NextElement. Mir diesem Wissen lässt sich eine Schleife über alle Objekte erstellen und diese einzeln Auswerten und gegebenenfalls ändern.

Mit dieser Funktion kann sich zum Beispiel alle geöffneten Dokumente, und damit die Tabellendokumente anzeigen lassen.

Sub doktypetest

Dim myList as Object

Dim myDoc as Object

```
‘Erzeugen der Objektliste
  mylist=stardesktop.components.createenumeration
‘Liste durchlaufen
  while mylist.hasmoreelements()
  myDoc=myList.nextelement()
  on error goto weiter
‘Prüfen ob es ein Tabellendokument ist.
    if myDoc.SupportsService("com.sun.star.sheet.SpreadsheetDocument") then
        msgbox("Ein Calc-Dokument")
         else msgbox ("Kein Calc-Dokument")
         end if
     weiter:
   wend
end sub
```

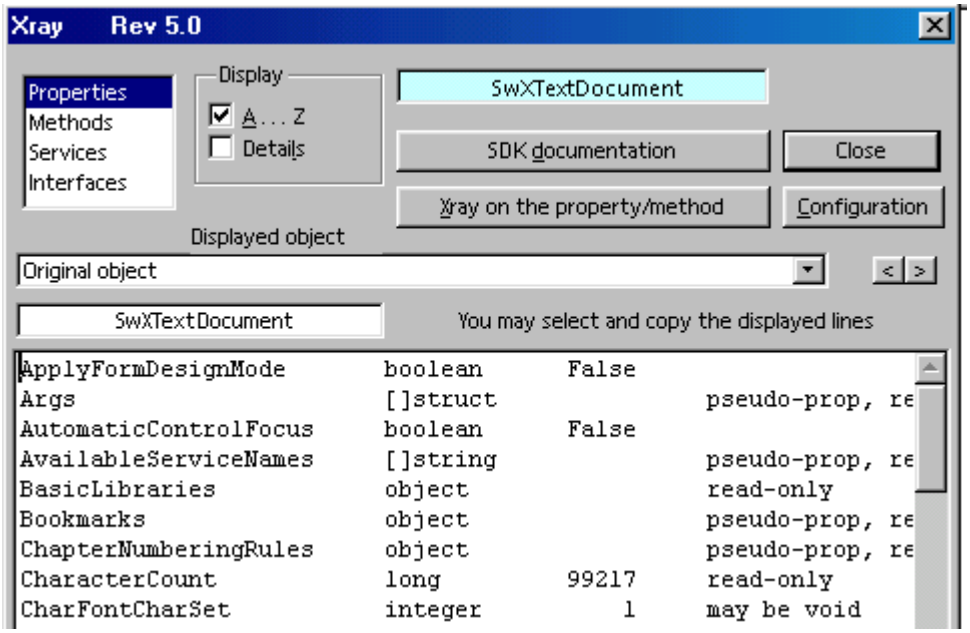
In diesem Beispiel bietet der Service StarDesktop.Comoponents die Schnittstelle com.sun.star.container.XEnumeration. Mit dieser kann man über createEnumeration den Service mit den einzelnen Objekten erzeugen. Dieser Service enthält die einzelnen Objekte, hier die Dokumente, und den Zugriff über die interne Auflistung. Mit den beiden Methoden hasMoreElements und nextElements kann man auf diese Objekte zugreifen.

Die Enumeration-Methode ist ein wesentlicher Bestandteil der API von StarBasic. Es gibt viele Services die sich nur auf diese Weise ansprechen lassen.

5.16 Wie kann man sich Services, Interfaces, Properties etc. anzeigen lassen?

Es gibt ein tolles Tool von Bernard Marcelly, Xray. Mit diesem Tool kann man sich sehr schnell alle Eigenschaften eines Objektes anzeigen lassen.

Es kann nach der Installation (Kopieren der Bibliothek) mit OpenOffice gestartet werden und steht dann immer zur Verfügung. Dann genügt ein einfaches "xray objekt" zum Aufruf. Zu finden ist es hier: http://sourceforge.net/



5.17 Wie kann man einen einzelnen Wert eines Structs erfahren?

Es gibt structs die aus einer Sammlung von PropertyValues bestehen. Um hier schnell einen Wert zu erhalten kann man eine Funktion verwenden.

```
Function GetProperty( searchProperties, cPropertie As String ) As com.sun.star.beans.PropertyValue
  For i = LBound( searchProperties ) To UBound( searchProperties )
    oCheckProp = searchProperties(i)
    If oCheckProp.Name = cPropertie Then
      GetProperty() = oCheckProp
      Exit Function
    EndIf
  Next
End Function
```

Mit dieser Funktion kann man den gesuchten Wert schnelle als eigenes Objekt erhalten.

Als Beispiel dient hier die Abfrage ob der Drucker frei ist.

```
sub main
 myDoc=ThisComponent
  myProperties=mydoc.getprinter()
  oPrinterStatus = GetProperty( myProperties, "IsBusy" )
  if oPrinterStatus.Value=false then msgbox "OK"
End Sub
```

6 Speichern/Öffnen/Drucken von Dateien

6.1 Öffnen/Speichern

6.2.1 Wie kann man eine Datei öffnen?

Eine Datei zu öffnen geht über den StarDesktop und die Methode loadComponentFromURL. Dieser Service (com.sun.star.frame.Desktop) steht automatisch mit dem Start von StarOffice als StarDesktop zur Verfügung. Die zu öffnende Datei ist im Zweifelsfall in das URL-Format zu konvertieren.

```
Sub DateiOeffnen

   url=converttourl("C:\test.sxw")
   dim myFileProp() as new com.sun.star.beans.PropertyValue
   oDocument = StarDesktop.loadComponentFromURL(url, "_blank", 0, myFileProp() )

end sub
```

6.2.2 Wie kann man Dokumente speichern?

Zum Speichern von Dokumenten stehen mehrere Methoden zur Verfügung.

Methode	**Beschreibung**
store()	Speichert das Dokument unter dem Namen mit dem es geöffnet ist. Ein neues Dokument ohne Namen (Unbenannt1) wird nicht gespeichert. Es gibt dann auch keine besondere Meldung.
storeAsUrl(sUrl as String, properties as com.sun.star.bean.PropertyValue)	Speichert das Dokument unter dem angegebenen Namen (sUrl). Dieser Name muß die gesamte Pfadangabe und im richitigen Format sein (convertourl). Das offene Fenster des Dokumentes erhält den Namen von sUrl. Mit den Properties werden mögliche Filtereinstellungen übergeben, z.B. um Dokumente in einem anderen Format zu speichern.
storeToUrl(sUrl as String, properties as com.sun.star.bean.PropertyValue)	Speichert das Dokument unter dem angebeben Namen (sUrl), an einem neuen Ort. Dieser Name muß die gesamte Pfadangabe und im richitigen Format sein (convertourl). Das offene Fenster des Dokumentes behält den alten Namen. Mit den Properties werden mögliche Filtereinstellungen übergeben, z.B. um Dokumente in einem anderen Format zu speichern.

Zusätzlich gibt es noch Methoden um den Zustand eines Dokumentes zu prüfen.

hasLocation()	Gibt True zurück wenn die Datei bereits gespeichert worden ist. Mit der Abfrage getUrl kann dies auch prüfen. Wenn das Dokument noch nicht gespeichert worden ist, ist der Rückgabewert ein leerer String.
getLocation() oder getUrl()	Gibt den kompletten Pfad des Dokumentes zurück. Der

	String ist leer bei einem neuen Dokument.
isModified()	Gibt True zurück, wenn das Dokument geändert wurde.
isReadOnly()	Gibt True zurück, wenn das Dokument nur zum Lesen geöffnet wurde.

Hier ein Beispiel für Speichern ohne besondere Filter. Dabei wird das Dokument im eingestellten Standardformat für den jeweiligen Dokumententyp gespeichert.

```
Sub Speichern
  dim dummy()
  datei="c:\test2.sxw"
  dateiurl=converttourl(datei)
  odoc=thisComponent
  odoc.storeasurl(dateiurl,dummy())
rem oder
rem    odoc.storetourl(dateiurl,dummy())
End Sub
```

Dummy() muß als Array definiert werden, sonst kommt es zu einer Fehlermeldung. Beachten muß man auch das die Endung des Dateitypes mit der URL übergeben werden muß. Steht dort sxc statt sxw wird das Dokument trotzden im Writerformat gespeichert. Auch das Weglassen eines Suffix wird ingnoriert.

Über die Verwendung der Filter stehen hier Informationen:Wie lauten die möglichen Filter zum Speichern und Exportieren von Dokumenten?

6.2.3 Wie kann man Dokumente schließen?

Zum Schließen von Dokumenten steht die Methode close zur Verfügung. Diese Methode hat nur einen Paramter: true/false. Dieser Parameter bestimmt, ob eine vorhandene Datei die Möglichkeit hat den Schließvorgang zu unterbrechen. Das könnte dann nützlich sein wenn das zu schließenende Dokument zum Beispiel noch eigene Prozesse ausführt. Der Paramter true/false muß seit einem Servicepack angegeben werden.
Dier Methode close() kann nicht auf eine Änderung des Dokumentes reagieren. Das Dokument wird ohne Änderung geschlossen. Möchte man dies erreichen, muß man vorher das Dokument auf Änderungen prüfen.

Die Änderung eines Dokumentes wird in der Propertie modified gespeichert. Die Abfrage erfolgt über ismodified().

```
sub closedok
   odoc=thisComponent
   checkclose=odoc.ismodified()
   if checkclose=false then
         odoc.close(false)
     else
         msgbox "Dokument wurde geändert"
    end if
```

end sub

6.2.4 Wie kann man ein Dokument mit Makros öffnen?

Man kann beim öffnen von Dokumenten festlegen wie Makros zu handhaben sind. Damit kann man zum Beispiel bei allen bekannten Dokumenten dafür sorgen das die Makrosicherheitsstufe automatisch übergangen wird. Oder das Gegenteil die Sicherheitsstufe erhöhen.

Verwendet wird dafür ein Parameter des MediaDiscriptors der bei Öffnen mit übergeben wird.

```
Dim myProp(0) as new com.sun.star.beans.PropertyValue
myProp(0).name="MacroExecutionMode"
myProp(0).value = 0
url=converttourl("C:\test.sxw")
oDocument = StarDesktop.loadComponentFromURL(url, "_blank", 0, myFileProp() )
```

Folgende Parameter stehen zur Verfüguung und können als Konstante oder Integerwert verwendet werden:

(Der Vorsatz zu der Konstante ist immer: com.sun.star.document.MacroExecMode.xxx)

NEVER_EXECUTE	0	Makros werden nicht ausgeführt
FROM_LIST	1	Nur Makros aus der hinterlegten List werden ausgeführt
ALWAYS_EXECUTE	2	Makros werden ausgeführt, aber mit Warnung wenn es eingestellt ist
USE_CONFIG	3	Die Konfiguration wird verwendet.
ALWAYS_EXECUTE_NO_WARN	4	Makros werden immer ausgeführt, keine Warnung
USE_CONFIG_REJECT_CONFIRMATION	5	Der Benutzer wird gefragt und bei Ablehnung wird die Konfiguartion verwendet.
USE_CONFIG_APPROVE_CONFIRMATION	6	Der Benutzer wird gefragt und bei Annahme wird die Konfiguartion verwendet.
FROM_LIST_NO_WARN	7	Makros von der Liste werden ausgeführt.
FROM_LIST_AND_SIGNED_WARN	8	Makros aus der Liste und zertifizierte Makros werden ausgeführt, aber mit Warnung.
FROM_LIST_AND_SIGNED_NO_WARN	9	Wie oben, aber ohne Warnung

6.2.5 Was ist der MediaDescriptor?

Um bestimmte Einstellungen bei Öffnen und Speichern zu übergeben wir der MediaDescriptor verwendet.

Dieser besteht aus einem Array von Properties die gesetzt werden können. Ein einzelner oder mehrere.

```
Dim myProp(0) as new com.sun.star.beans.PropertyValue
myProp(0).name="MacroExecutionMode"
myProp(0).value = 0
url=converttourl("C:\test.sxw")
oDocument = StarDesktop.loadComponentFromURL(url, "_blank", 0, myFileProp() )
```

Hier die Liste der wichtigsten Parameter:

AsTemplate	Boolean	True - Das Dokument wird als Vorlage verwendet
Author	String	Autor des Dokumentes wird gesetzt
Comment	String	Bemerkung wird gesetzt
DocumentTitle	String	DokumentenTitel wird gesetzt
FilterName	String	Interner Filtername
FilterOptions	String	Zusätzliche Parameter zum Filter
FilterData	Any	Zusätzliche Parameter zum Filter
Hidden	Boolean	True - Das Dokument wird versteckt geöffnet
ReadOnly	Boolean	True - Das Dokument wird schreibgeschützt geöffnet.
PassWord	String	Password beim Öffnen oder Speichern
OverWrite	Boolean	Eine vorhandene Datei wird beim Speichern überschrieben.
StartPresentation	Boolean	Bei einer Presentation wird diese sofort gestartet

Weitere Parameter stehen in der Referenz unter com.sun.star.document.MediaDesriptor.

6.2.6 Wie kann man eine Datei schreibgeschützt öffnen?

Der Status "ReadOnly" wird mit dem MediaDiscriptor beim Laden übergeben.

```
Dim myProps(0) as New com.sun.star.beans.PropertyValue
sUrl= "file:///C:/test.sxw"
myProps(0).Name="ReadOnly"
myProps(0).Value = true
oDocument = StarDesktop.loadComponentFromURL(surl, "_blank", 0, myProps() )
```

6.2.7 Wie werden die Filteroptionen bei Dateien gesetzt?

Für bestimmte Dateitypen sind zusätzliche Filteroptionen beim Speichern und beim Öffnen nötig. Diese werden dann mit den Fileproperties übergeben. Der wichtigste Dateityp mit diesen Filteroptioen ist die reine Textdatei (CSV).

```
Sub Speichern
Dim myProps(0) as New com.sun.star.beans.PropertyValue
sUrl= "file:///C:/test.csv"
myProps(0).Name="FilterOptions"
myProps(0).Value ="59/9,34,IBMPC_850,1,1/1/1/1/1/1/1/1" 'string mit den Optionen
myDoc = thisComponent
myDoc.storeAsUrl(sUrl,myProps())
End Sub
```

Manchmal steht in Dokumentationen statt FilterOptions FilterFlags. Dies ist eine ältere Variante die noch unterstützt wird.

Der Filter für Textdateien besteht aus einem String mit Parametern.

6.2.8 Wie kann man eine Vorlage öffnen?

Öffnet man auf normalen Weg eine Vorlage, wird automatisch ein neues Dokument erstellt.
Dies kann man mit dem Paramter AsTemplate verhindern.

```
Dim myFileProp(0) As New com.sun.star.beans.PropertyValue
oDesktop =   createUnoService("com.sun.star.frame.Desktop")
sSourceFile = "file:///c:/vorlage.stw"
myFileProp(0).Name = "AsTemplate"
myFileProp(0).Value = False
```

oDocument = oDesktop.loadComponentFromURL(sSourceFile, "_blank", 0,myFileProp())

6.2.9 Kann man eine Datei ungepackt speichern?

Achtung! Dies geht leider nur bis 00 1.1.2 bzw S0 Service Pack 2. In OO 2.0/SO 8.0 wird die Funktion zur Zeit nicht unterstützt

Man kann! Über ein Makro.

Dazu muß man bei speichern den Parameter "Unpacked" der Methode storeasurl auf True setzen. Dann wird in dem Zielverzeichnis ein Unterverzeichnis mit den Daten angelegt.

```
sub SaveAsUnpacked
  dim mydoc as object
  dim myFileproperties(0) as new com.sun.star.beans.PropertyValue
  dim myurl as string

  mydoc = ThisComponent
  myurl = "file:///d:/test/test3.sxw"
  myFileproperties(0).Name = "Unpacked"
  myFileproperties(0).Value = True

  mydoc.storeasurl(myurl,myfileproperties())
end sub
```

6.2.10 Wie kann man mit Starbasic ein Dokument als PDF speichern?

Nur SO 7.0 oder OO 1.1:

Das geht ähnlich wie mit den Filtern zum Speichern, nur dass man nicht die Methode storeasURL, sondern die Methode storetoURL verwendet, weil es sich um einen Export handelt.

Zusätzlich gibt es noch die Property CompressMode, deren Wirkung konnte ich aber noch nicht nachvollziehen.

```
myProps(1).Name = "CompressMode"
myProps(1).Value = 0
' mögliche Werte sollen 0,-1,-2 sein
```

Es wird immer die Kompressionseinstellung des letzten direkten PDF-Exports aus OO/SO verwendet.

sub SpeichernAlsPDF

Dim myProps(0) as New com.sun.star.beans.PropertyValue

sUrl= "file:///C:/test.pdf"
myProps(0).Name="FilterName"
myProps(0).Value = "writer_pdf_Export"
myDoc = thisComponent

myDoc.storetoUrl(sUrl,myProps())

end Sub

6.2.11 Wie lauten die möglichen Filter zum Speichern und Öffnen von Dokumenten?

Die Wichtigsten:

Für Texte:

StarOffice 6 -> StarOffice XML (Writer)
HTML -> "HTML (StarWriter)"
StarOffice Vorlage -> "writer_StarOffice_XML_Writer_Template"
RTF -> "Rich Text Format"
MS Word 97/2000/XP -> "MS Word 97"

Für Tabellen:

Ms Excel 97/2000/XP -> "MS Excel 97"

Um einen Filter zu verwenden muß die Property "FilterName" gesetzt werden.

Zusätzlich sind noch bei einigen Filter Filteroptionen zu setzen. Näheres dazu hier: Wie werden die Filteroptionen bei Dateien gesetzt?

```
Sub Speichern

Dim myProps(0) as New com.sun.star.beans.PropertyValue


sUrl= "file:///C:/test"

myProps(0).Name="FilterName"

myProps(0).Value = "MS Word 97"


myDoc = thisComponent


myDoc.storeAsUrl(sUrl,myProps())

End Sub
```

Als Alternative kann man auch die Funktion storeToUrl verwenden, dann findet ein Export in ein neues Dokument statt. Das Originaldokument bleibt dann offen. Einige Filter erlauben nur die Verwendung von storToUrl. Zum Beispiel der PDF-Filter.

6.2.12 Wie kann man eine Datei umbenennen?

Die Funktion heißt nicht wie vielen anderen Sprachen Rename, sondern nur Name.
Sinnvollerweise sollte man den Fehler bei einer bereits existierenden Datei abfangen.

Entweder über eine Fehlerroutine oder über eine vorherige Abfrage mit Fileexist.

Beispiel:

```
Sub ReName1 'Fehler prüfen

On Error Goto Fehler

Name "c:\test.sav" as "c:\test.bat"

end

Fehler:

if err = 76 then

msgbox "Datei existiert bereits"

end if
```

```
end
end sub


Sub ReName2 'Vorher prüfen
If Fileexists("c:\test.bat")=false  then
Name "c:\test.sav" as "c:\test.bat"
    else
     msgbox "Datei existiert bereits"
   end if
end sub
```

6.2.13 Wie kann man eine neue Datei aus einer Vorlage öffnen?

```
Sub NewDocFromTemplate
    Dim myFileProp(0) As New com.sun.star.beans.PropertyValue
    oDesktop =   createUnoService("com.sun.star.frame.Desktop")
    sSourceFile = "file:///c:/test.sxw"
    myFileProp(0).Name = "AsTemplate"
    myFileProp(0).Value = True
    oDocument = oDesktop.loadComponentFromURL( sSourceFile, "_blank", 0, myFileProp() )
End Sub
```

6.2.14 Wie kann man einen Dialog zur Verzeichnisauswahl aufrufen?

Mit dem Service FolderPicker von com.sun.star.ui.dialogs

```
Sub TestDialog
MyDialog=createunoservice("com.sun.star.ui.dialogs.FolderPicker")
myDialog.displaydirectory="c:\"
myDialog.execute
msgbox mydialog.directory
End sub
```

Die Propertie Directory enthält das Verzeichnis.

Mit DisplayDirectory kann das Startverzeichnis festgelegt werden.

6.2.15 Wie kann man einen Öffnen- oder Speichern-Dialog aufrufen?

Mit dem Service FilePicker von com.sun.star.ui.dialogs

Sub TestDialog

MyDialog=createunoservice("com.sun.star.ui.dialogs.FilePicker")

myDialog.displaydirectory="c:\"

MyDialog.execute

FileName=MyDialog.Files(0)

End sub

Mit der Propertie Files wird beim Öffnen des Dialoges ein Array bereitgestellt. Als Standard ist die Mehrfachauswahl nicht möglich. Wenn diese gewünscht wird ist dieser Modus zu aktivieren und dann das Array entsprechend auszulesen.

Der Mehrfachauswahlmodus (nur beim Öffnen-Dialog) wird mit:

myDialog.setMultiSelectionMode(true)

aktiviert.

Mit DisplayDirectory kann das Startverzeichnis festgelegt werden.

Um nun verschiedene Dialogtypen für Öffnen und Speichern zu bekommen, muss der Dialog mit einer Konstanten initialisiert werden.

Dim Dialogtyp(0)

DialogTyp(0) =com.sun.star.ui.dialogs.TemplateDescription.FILESAVE_SIMPLE

myDialog.initialize(DialogTyp())

Achtung wenn man den Pfad festlegen will muß dies nach *myDialog.initialize(DialogTyp())* erfolgen.

Mögliche Konstanten sind dabei:

Für den Öffnen Dialog

FILEOPEN_SIMPLE = Standard Öffnen Dialog

FILEOPEN_LINK_PREVIEW_IMAGE_TEMPLATE = Mit Checkbox Vorschau

FILEOPEN_PLAY = Mit "Abspielen" Button

FILEOPEN_READONLY_VERSION = Mit ReadonlyButton

FILEOPEN_LINK_PREVIEW = Verlinken und Vorschau

Für den "Speichern-unter"-Dialog

FILESAVE_SIMPLE = Standard Speichern Dialog

FILESAVE_AUTOEXTENSION_PASSWORD = Speichern mit Automatischer Extension und Password

FILESAVE_AUTOEXTENSION_PASSWORD_FILTEROPTIONS = dito mit Filtern

FILESAVE_AUTOEXTENSION_SELECTION =Speichern mit Automatischer Extension und Selection

FILESAVE_AUTOEXTENSION_TEMPLATE = Mit Listbox Formatvorlage

FILESAVE_AUTOEXTENSION = Automatische Extension

Bei dem "Speichern unter"-Dialog gibt es aber einen Haken: Die möglichen Dateiformate werden nicht angezeigt.

Diese kann man aber mit der Funktion **appendfilter** anhängen. Danach muß man den ausgewählten Filter über die Propertie "currentfilter" abfragen.

Aufgrund des Filters muß dann der entsprechende Filter zum Speichern verwendet werden.

myDoc = thisComponent

DialogTyp(0) = com.sun.star.ui.dialogs.TemplateDescription.FILESAVE_AUTOEXTENSION

MyDialog=createunoservice("com.sun.star.ui.dialogs.FilePicker")

myDialog.initialize(DialogTyp())

myDialog.appendfilter("MS Word 97","Fi")

myDialog.appendfilter("MS Word 95","Filtername")

MyDialog.execute

FileName=MyDialog.Files(0)

Filtername=myDialog.currentfilter

Dim myProps(0) as New com.sun.star.beans.PropertyValue

sUrl= MyDialog.Files(0)

```
myProps(0).Name="FilterName"
myProps(0).Value = myDialog.currentfilter
REM Achtung dann muß der Schreibweise des Filters (siehe Wie lauten die möglichen Filter zum
REM rem Speichern und Exportieren von Dokumenten?entsprechen
myDoc.storeAsUrl(sUrl,myProps())
```

6.2.16 Wie kann man ein Dokument versteckt öffnen?

Man kann ein Dokument auch öffnen, ohne dass es für den Anwender sichtbar ist.
Laut der StarBasic-Dokumentation muss man dafür den Paramter für das Zielframe auf _hidden setzen.

```
oDocument = StarDesktop.loadComponentFromURL(url, "_hidden", 0, myFileProp() )
```

Das geht aber leider nicht.

Der richtige Weg geht über den Mediadescriptor, der auch die Filtereinstellungen festlegt.

```
Sub DateiVerstecktOeffnen
    url=converttourl("C:\test.sxw")
    dim myFileProp(0) as New com.sun.star.beans.PropertyValue
    myFileProp(0).name="Hidden"
    myFileProp(0).value=True
    oDocument = StarDesktop.loadComponentFromURL(url, "_blank", 0, myFileProp() )
end sub
```

Nach dem Öffnen steht das Dokument als Objekt zur Verfügung. Man sollte daran es später auch zu schließen und nötigenfall zu speichern.

Ob ein nicht sichtbares Dokument geöffnet ist kann man über das Enumartion-Objekt der geöffneten Dokumente prüfen.

```
  oComponents = StarDesktop.getComponents()
  oDocs = oComponents.createEnumeration()
  Do While oDocs.hasMoreElements()
```

```
    oDoc = oDocs.nextElement()
  Loop
```

Dazu kann man sich die Funktion fensterwaehlen anpassen.

6.2.17 Wie kann man den Dateinamen, Pfad oder die Extension herausfiltern?

In der mitgelieferten Bibliothek "Tools" Modul "Strings" stehen folgende Befehle zur Verfügung:

Function FileNameoutofPath(ByVal Path as String, Optional Separator as String) as String

Function GetFileNameExtension(ByVal FileName as String)

Function GetFileNameWithoutExtension(ByVal FileName as String, Optional Separator as String)

Function DirectoryNameoutofPath(sPath as String, Separator as String) as String

Näheres dazu ist unter auf dieser Seite unter Makro und Tools -> Welche Tools gibt es zu Stringverarbeitung zu finden.

6.2.18 Wie kann man ein neues leeres Dokument erzeugen?

Um ein neues leeres Dokument zu erzeugen, verwendet man die Methode loadComponentFromURL vom Objekt Desktop.

```
Dim mArgs()
oDocument = StarDesktop.loadComponentFromURL ("private:factory/scalc","_blank", 0, mArgs())
```

Über den Parameter "private:factory/...." wird die Art des Dokumentes festgelegt.

private:factory/swriter = Textdokument

private:factory/scalc = Tabelle

private:factory/sdraw = Zeichnung

private:factory/simpress = Präsentation

private:factory/smath = Formel

6.2.19 Format der Url zum Öffnen und Speichern von Dateien

In Starbasic wird das URL Format gemäß RFC 1738 verwendet:

Zum Beispiel:

file:///c:/test/datei.sxw

Das heißt dem normalen Filenamen wird das Präfix "file:///" vorangestellt.
Außerdem wird unter Windows statt dem Backslash "\" der normale Slash "/" verwendet.

In Starbasic gibt es dazu zwei Funktionen:

ConvertToUrl(sFile as String)
Wandelt einen normalen Dateinamen in das URL-Format

UrlFile = ConvertToUrl("c:\test\datei.sxw")
ergibt dann als String in UrlFile = "file:///c:/test/datei.sxw"

und ConvertFromUrl(sfile as String)
liefert dann das umgekehrte Ergebnis

File = ConvertFromUrl("file:///c:/test/datei.sxw")
ergibt dann als String in File = "c:\test\datei.sxw"

6.2.20 Wie kann man prüfen ob ein Dokument einen Dateinamen hat?

Dokumente erhalten erst nach dem Speichern eine eigen URL-Adresse. Solange ist der Parameter URL eines Dokumentes leer.

Mit der Funktion hasLocation kann man prüfen ob bereits eine Name vorhanden ist.

```
Sub Main
  odoc = thiscomponent
  check = odoc.haslocation()
  if check then
  msgbx "OK - Datei hat einen Namen"
  else msgbox "Nein - Das Dokument ist neu und noch nicht gespeichert"
  end if
End Sub
```

6.2 Drucken

6.3.1 Warum wird der Druckbefehl nicht ausgeführt?

Wenn man den Makrobefehl doc.print verwendet kann es zu Problemen kommen. Die weitere Verarbeitung bricht ab oder wenn man anschließend das Makro weiter verfolgt wird der Druck nicht richtig ausgeführt. Ursache ist ein Bug der den Druck verhindert. Um dieses Problem zu umgehen gibt es den undokumentierten Parameter wait. Die Übergabe erfolgt mit den anderen Druckparametern.

```
    printProp(0).Name = "Wait"
    printProp(0).Value = "true"
```

6.3.2 Wie kann man mehrere Seiten auf einer Seite drucken?

Mit den PagePrintSettings und der Methode printPages kann man bei Textdokumenten mehrere Seiten auf eine Seite drucken. Im Programm geht das nur über die Seitenvoransicht.

Eigentlich gibt es zusätzlich noch die Methoden getPagePrintSettings und setPagePrintsettings. Die Erste geht, da man aber die zweite Methode anscheinend nicht geht, lassen sich nur die Standardwerte auslesen. Um die Einstellungen zu ändern muß man die PageSettings als Parameter bei dem Druckbefehl übergeben.

Eine Wirkung der Einstellungen für die Seitenränder konnte ich nicht feststellen.

```
sub mytest
odoc=thiscomponent
```

```
Dim props(8) as new com.sun.star.beans.PropertyValue
props(0).Name="PageRows"
props(0).Value=0
props(1).Name="PageColumns"
props(1).Value=0
props(2).Name="LeftMargin"
props(2).Value=2
props(3).Name="RightMargin"
props(3).Value=2
props(4).Name="TopMargin"
props(4).Value=2
props(5).Name="BottomMargin"
props(5).Value=2
props(6).Name="HoriMargin"
props(6).Value=2
props(7).Name="VeriMargin"
props(7).Value=2
props(8).Name="IsLandscape"
props(8).Value=False
odoc.printpages(props())
end sub
```

6.3.3 Wie kann man den Drucker wechseln?

Man kann den Drucker auslesen und ändern mit den Methoden getPrinter und setPrinter des Dokumenten-Objektes.
Bei der Eigenschaft handelt es sich um Array aus dem Service com.sun.star.view.PrinterDescriptor. Achtung der Drucker muß genauso geschrieben werden, wie in den Einstellungen des Betriebsystemes

```
myDoc=thiscomponent
myprinter=mydoc.getprinter
msgbox myPrinter(0).value

dim arg(0) as New com.sun.star.beans.PropertyValue
arg(0).name="Name"
```

arg(0).value="Acrobat PDFWriter"

mydoc.setPrinter(arg())

6.3.4 Wie kann man Dokumente drucken?

Eine der wichtigsten Aufgaben einer Office-Anwendung ist das Drucken. Schließlich muß im Zeitalter des papierlosen Büros auch alles für die Akten ausgedruckt werden.

Jeder Dokumententyp unterstützt den Service com.sun.star.view.XPrintable. Dieser steuert den Ausdruck und die Eigenschaften des Drucks.

Die Verwendung ist eigentlich denkbar einfach:

myDoc=ThisComponent

myDoc.Print(args())

Die Parameter die übergeben werden können sind:

Pages	Die Angabe welche(n) Seite(n) gedruckt werden soll.
	Fortlaufend durch Bindestrich oder einzelne getrennt dur ein Semikolon. (z.B.1-4;8;10)
CopyCount	Anzahl der Kopien
Collate	Sortieren der Kopien (True)
Sort	Sortieren der Kopien (True)
Filename	Dateiname, wenn der Ausdruck in eine Datei erfolgen sol
	Mit der Angabe des Dateinamen erfolgt der Ausdruck automatisch in eine Datei.

```
sub testdruck
    Dim printProp(4) as New com.sun.star.beans.PropertyValue
    printProp(0).Name = "Pages"
    printProp(0).Value = "1-3"
    printProp(1).Name = "CopyCount"
    printProp(1).Value = 3
    printProp(2).Name = "Collate"
    printProp(2).Value = False
    printProp(3).Name = "Sort"
    printProp(3).Value = False
   myDoc=ThisComponent
```

```
    myDoc.Print(printProp())
end sub
```

Leider gibt es bei den Parametern ein Problem: Sie funktionieren nicht alle.

Sort und Collate haben keinerlei besondere Auswirkung. Die Sortierung der Kopien erfolgt nicht.

Mit Collate wird nur eine Kopie gedruckt, diese sortiert. Mit Sort passiert nichts.

(Auch 2.0)

Zusätzlich zu den allgemeinen Druckparametern gibt es noch die möglichen Parameter des Druckers. Dies sind in Starbasic acht Parameter.

Der Service dazu ist com.sun.star.view.PrinterDescriptor

Name	Name des Druckers	
PaperOrientation	Horizontal oder Vertikal (Hochformat/Querformat)	
	Einstellung über Enum c.s.s.view.PaperOrientation	PORTAIT (Hochformat)
		LANDSCAPE (Querformat)
PaperFormat	Papier: A4, A5 etc	
	enum com.sun.star.view.PaperFormat	A3,A4,A5,B4,B5,LETTER
		LEGAL, TABLOID
		USER
PaperSize	Eigene Papiergröße (PaperFormat=USER)	Angabe in mm
	struct com.sun.awt.size	WIDTH (Breite)
		HEIGHT (Höhe)
IsBusy	Stellt fest ob der Drucker frei ist. (true)	
CanSetPaperorientation	Erlaubt der Drucker die Änderung	
CanSetPaperFormat	Erlaubt der Drucker die Änderung	
CanSetPaperSize	Erlaubt der Drucker die Änderung	

```
Sub DruckerWahl
    ' Variablen deklarieren
    Dim oODoc as Object
    Dim Dummy()
    Dim printprops(3) as New com.sun.star.beans.PropertyValue

    ' aktuelles Dokument holen
```

```
    ODoc=ThisComponent

    ' Definition des Druckers
    printprops(0).Name = "Name"
    printprops(0).Value="Acrobat PDFWriter"
    printprops(1).Name = "PaperFormat"
    printprops(1).Value = com.sun.star.view.PaperFormat.A4
    printprops(2).Name = "PaperOrientation"
    printprops(2).Value = com.sun.star.view.PaperOrientation.PORTRAIT

    ' Drucker auswählen
    oOfficeDokument.setPrinter( drucker() )

End Sub
```

Um die Papiergröße selber einzugeben:

```
    printprops(2).Name = "PaperFormat"
    printprops(2).Value = com.sun.star.view.PaperFormat.USER
    Dim myPage As New com.sun.star.awt.size
    myPage.WIDTH=100
    myPage.HEIGTH=100
    printprops(3).Name = "PaperSize"
    printprops(3).Value = MyPage
```

6.3.5 Wie kann man die installierten Drucker auslesen?

Die API bietet dafür keine Funktion.
Unter Windows kann man selber die Registry auslesen.

Ich habe hier ein kleines Programm das unter Windows 2000/XP die Drucker mit Hilfe regedit und einem Shellaufruf die Drucker ausliest.

```
sub ZeigeAlleDrucker
  shell("regedit /e c:\printer.txt 'HKEY_CURRENT_USER\Software\Microsoft\Windows
NT\CurrentVersion\Devices'",10)
  wait 2000
```

```
  zaehler=0
  ende=false
  Dim myPrinter(20)
  #iNumber = Freefile
  aFile = "c:\printer.txt"
  Open aFile For Input As #iNumber
  On Error Goto schluss
  While not eof(#iNumber)
  Line Input #iNumber, sZeile
  start=left(sZeile,1)
  if start="""" then
      ipos=InStr(sZeile,"""=")
      myprinter(zaehler)=mid(szeile,2,ipos-2)
      zaehler=zaehler+1
   end if
  wend
  Close #iNumber
  schluss:
  for i=0 to zaehler-1
    msgbox myprinter(i)
  next i
end sub
```

6.3.6 Wie kann man die zusätzlichen Einstellungen einstellen?

Unter Zusätze im Druckerdialog kann man weitere Parameter des Ausdrucks festlegen. Zum Beispiel Prospektdruck, Grafiken drucken, Linke Seiten etc..

Diese Einstellungen werden mit dem Service com.sun.star.text.Printsettings vorgenommen.

Dafür gibt es zwei Wege. Man kann die globalen Einstellungen verwenden oder die Einstellungen des Dokumentes.

Global:

```
Sub setGlobalPrintSettings

oGlobalSettings = createUnoService("com.sun.star.text.GlobalSettings")

oGlobalSettings.PrintSettings.PrintProspect = True

End Sub
```

Dokumentenbezogen:

Dann sind die Einstellungen direkt als Propertie vorzunehmen.

```
Sub setDocPrintSettings

odoc=thisComponent

oDocSettings = oDoc.createInstance("com.sun.star.text.DocumentSettings")

oDocSettings.PrintDrawings=true

end sub
```

6.3.7 Wie kann man den aktuellen Drucker auslesen?

Dies geht über die Eigenschaften eines Dokumentes und getPrinter. Damit wird der aktuell eingestellte Drucker angezeigt.

```
sub GetPrinter
  oDoc = ThisComponent
  aPrinterProperties = oDoc.getPrinter()
  oPrinterName = GetProperty(aPrinterProperties, "Name" )
  If IsNull( oPrinterName ) Then
    MsgBox( "Kein Drucker gefunden" )
  Else
    MsgBox( "Der aktuelle Drucker ist: " & oPrinterName.Value )
  EndIf
end sub
```

Achtung die Subroutine braucht die Funktion GetProperty. Siehe hier

7 Tabellen - Calc

7.1 Zellen

7.2.1 Wie bekomme ich Zugriff zu einer Zelle?

Es gibt zwei Wege:

getCellByPosition und getCellRangeByName

Mit getCellByPosition kommt man an die Zelle über die absolute Positionsangabe, wobei diese über die Spalten- und Zeilenangabe erfolgt.

getCellByPosition(1,1) = B2

Die Zählung fängt dabei bei 0 an A1 ist also 0,0

Mit getCellRangeByName kann man die Zelle direkt mit den Namen ansprechen oder Bereiche auswählen.

getCellRangeByName("$A$1") ergibt dann die Zelle A1

getCellRangeByName("A1:B5") ergibt dann den Bereich A1:B5

Man kann auch gleichzeitig mehrere Bereiche auswählen. Dann müssen diese nur mit Komma getrennt eingegeben werden.

getCellRangeByName("A1:B5","C4:D9") ergibt dann den Bereich A1:B5 und C4:D9

```
sub ZellenInTabellen
' Dieses Makro in einer geöffnetten Tabelle starten
myDoc = thisComponent
mySheet = myDoc.sheets(0)

mycell = mysheet.getCellByPosition(0,0)
mycell.string = "Hallo Welt"

mycell = mysheet.getCellRangeByName("$C$1")
mycell.string = "Hallo Welt"

end sub
```

7.2.2 Wie kann man den Rahmen von Zellen einstellen?

Für Tabellenbereiche und Tabellenzellen erfolgt die Einstellung der Rahmen genau so wie bei Tabellen im TextDateien.

Wie kann man in Tabellen den Rahmen einstellen?

Nur das anstelle der Zelle der Texttabelle die Zelle der Tabelle und statt der Tabelle der Textdatei ein Zellbereich verwendet werden muß.

7.2.3 Wie kann man den Hintergrund einer Zelle/eines Bereiches ändern?

Man kann in Calc den Hintergrund von Zellen, Zeilen, Spalten und Bereichen ändern. Innerhalb von Starbasic steht diese Information in der Properties CellBackColor.

Spalte:

oCalc = ThisComponent

oSpalte = oCalc.Sheets(0).Columns(0)

oSpalte.CellBackColor = &H00CC00

Zeile:

oCalc = ThisComponent

oZeile = oCalc.Sheets(0).Rows(0)

oZeile.CellBackColor = &H00CC00

Bereich:

oCalc = ThisComponent

oBereich=oCalc.Sheets(0).getCellRangeByName("A1:B5")

oBereich.CellBackColor = &H00CC00

Zelle :

oCalc = ThisComponent

*oZelle=oCalc.Sheets(0).*getCellByPosition(1,1)

oZelle.CellBackColor = &H00CC00

7.2.4 Wie kann man den Inhalt von Zellen löschen?

Um den Inhalt und/oder die Formatierung von Zellen oder Zellbereichen zu entfernen, verwendet man die Methode clearContents. Diese Methode steht bei Zellen und Zellbereichen zur Verfügung.

```
oCalc = thisComponent
oSheet = oCalc.sheets(0)
oCellRange = osheet.getCellRangeByName("A2:B5")

ocellRange.clearContents(com.sun.star.sheet.CellFlags.VALUE_
        +com.sun.star.sheet.CellFlags.STRING_
        +com.sun.star.sheet.CellFlags.DATETIME)
ocell=osheet.getcellbyposition(0,0)
ocell.clearContents(com.sun.star.sheet.CellFlags.VALUE_
        +com.sun.star.sheet.CellFlags.STRING_
        +com.sun.star.sheet.CellFlags.DATETIME)
```

ClearContents benötigt Parameter mit den bestimmt werden welche Inhallte oder Formatierungen gelöscht

werden sollen.

Diese Parameter werden durch "+" kombiniert.

VALUE	Nummerische Werte
DATETIME	Datum/Zeit-Werte
STRING	Text
ANNOTATION	Die Notiz
FORMULA	Formeln
HARDATTR	Die harten Formatierungen
STYLES	Styles
OBJECTS	Grafik
EDITATTR	Textformartierungen ?
FORMATTED	?

7.2.5 Wie kann man die Zellen eines Ranges auswerten?

Es gibt zwei Wege. Über die Position im Arbeitsblatt oder die Position innerhalb des Ranges.

Erstmal die Position innerhalb des Arbeitsblattes.

Ein Zellrange besteht aus mehren Zellen in Zeilen und vielleicht Spalten.

Diese lassen sich natürlich auch gezielt ansprechen und auch mit Hilfe von zwei Schleifen einzeln nacheinander anspechen.

Eine Range hat unter der Rangeadresse die Eckpunkte des Ranges. Diese fragt man ab und lässt dann darüber eine verchachtelte For-Schleife laufen.

```
    oCalc = thisComponent
    oSheet = oCalc.sheets(0)
    oCellRange = osheet.getCellRangeByName("A1:B5")
'auslesen der Eckpunkte
    iErsteSpalte = oCellRange.rangeAddress.startColumn
    iErsteZeile = oCellRange.rangeAddress.startRow
    iLetzteSpalte = oCellRange.rangeAddress.EndColumn
    iLetzteZeile = oCellRange.rangeAddress.EndRow

    'Schleife über die Zeilen im Sheet
```

```
For i = iErsteZeile to iLetzteZeile
 For m = iErsteSpalte to iLetzteSpalte
  oCell=osheet.getCellByPosition(m,i)
  oCell.String=i
 next m
next i
```

Oder

```
'Schleife über die Zeilen im Range
For i = 0 to iLetzteZeile-iErsteZeile
 For m = 0 to iLetzteSpalte-iErsteSpalte
  oCell=oCellRange.getCellByPosition(m,i)
  oCell.String=i
 next m
next i
```

In dem Beispiel ist die Range fest angegeben. Dann kann man natürlich die Schleife direkt über die Werte laufen lassen.

Ideal ist die Lösung wenn man auf ein Markierung durch den Anwender reagieren will.

```
oCellRange = osheet.getCurrentSelection()
```

Dann muß aber vorher geprüft werden ob es bei der Selektion eine Zelle, ein Bereich oder mehrere Bereiche sind.

Siehe Wie kann man prüfen ob Zellen oder Bereiche in der aktuellen Selektion sind?

Der zweite Weg geht direkt über die Position innerhalb des Ranges.

Dann erfolgt der Zugriff über Methode getCellByPosition die auch innerhalb eines Ranges zur Verfügung steht. Hier bei gehts es aber nur die relative Position.

```
oCalc = thisComponent
oSheet = oCalc.sheets(0)
oCellRange = osheet.getCellRangeByName("A1:B5")
oCell = oCellRange.getCellByPosition(0,0)
```

Mit 0,0 erreicht man die erste Zelle, mit 0,1 die erste Zelle in der zweite Spalten usw..

Aufpassen muß man hiebei das eine falsche Zeilen oder Spaltenangabe, die außerhalb des gewählten Ranges liegen, kommt es zu einer Fehlermeldung.

7.2.6 Wie kann man die Hintergrundfarbe einer Zelle einstellen?

Man kann mit der Propertie CellBackcolor die Hintergrundfarbe einer Zelle oder eines Zellbereiches einstellen.

```
   oDocument = ThisComponent
   oSheet1 = oDocument.Sheets.getByIndex(0)
'zellbereich
   oZellRange=oSheet1.getCellRangeByPosition(0,0,10,10)
   oZellRange.Cellbackcolor=&H00000
'Zelle
   myZell = osheet1.getCellByPosition(0,0)
   myZell.cellbackcolor=&HCCCCCC
```

7.2.7 Wie kann man den Text von Zellen formatieren?

Sheets, Zellen und Zellbereich unterstützen den Service com.sun.star.style.CharacterProperties. Mit diesem kann man die Einstellungen der Schrift "hart" formatiert werden. Dabei muss man unterscheiden ob man den kompletten Text einer Zelle oder einzelne Zeichen des Textes formatiert.
Zweiteres geht in Calc erst ab 2.x.

Ein Text "hart" formartieren, bedeutet das man den Text selbst formatiert und dies nicht mit der Hilfe von Formatvorlagen macht. Dadurch werden die Formatangaben direkt mit der Zelle verbunden, oder dem Test wenn man diesen extra formatiert.

Formatvorlagen kann man nur für die Zelle verwenden.

Der Service com.sun.star.style.CharacterProperties Zellen und Zellbereichen unterstützt.

Die wichtigsten Properties hierbei sind:

CharFontName - Der Schriftname

CharHeight - Die Schriftgröße

CharWeight und CharPosture - Der Schriftschnitt (fett, kursiv etc.)

CharColor - Schriftfarbe

CharBacKColor -Hintergrundfarbe

CharUnderline - Unterstreichung

Der folgende Code formatiert den Text als ComicSansiff, schwarz, Größe 20, einfach unterstrichen, kursiv und fett.

```
oDoc = thiscomponent
osheet=odoc.sheets(0)
mycell=osheet.getcellbyposition(0,0)
myCell.CharFontName="ComicSansif"
myCell.CharHeight="20"
myCell.charcolor=&H000000
myCell.CharUnderline=com.sun.star.awt.FontUnderline.SINGLE
myCell.CharWeight=com.sun.star.awt.FontWeight.BOLD
myCell.CharPosture=com.sun.star.awt.FontSlant.ITALIC
```

Um einen Text kursiv und fett darzustellen müssen beide Paramter gesetzt werden: CharWeight und CharPosture.

Der folgende Code formatiert die ersten drei Zeichen einer Zelle fett.

```
sheet = ThisComponent.CurrentController.ActiveSheet
textcur=sheet.getcellbyposition(0,0).gettext().createtextcursor()
textcur.gotostart(false)
textcur.goright(3,true)
textcur.CharWeight=com.sun.star.awt.FontWeight.BOLD
```

Achtung nicht jeder Schrifttyp unterstützt alle Fomartierungen!

Die wichtigsten Parameter für CharWeight sind

(weitere stehen in der Referenz)

com.sun.star.awt.FontWeight.NORMAL

com.sun.star.awt.FontWeight.BOLD

Die wichtigsten Parameter für CharPosture sind

(weitere stehen in der Referenz)

com.sun.star.awt.FontSlant.NONE	keine einstellung
com.sun.star.awt.FontSlant.ITALIC	kursiv
com.sun.star.awt.FontSlant.OBLIQUE	Oblique

Die wichtigsten Parameter für CharUnderline sind:

(weitere stehen in der Referenz)

com.sun.star.awt.FontUnderline.None	Keine
com.sun.star.awt.FontUnderline.SINGLE	einfach
com.sun.star.awt.FontUnderline.DOUBLE	doppelt
com.sun.star.awt.FontUnderline.DOTTED	gepunkted
com.sun.star.awt.FontUnderline.DASH	gestrichelt
com.sun.star.awt.FontUnderline.LONGDASH	lang gestrichelt
com.sun.star.awt.FontUnderline.DASHDOT	strich punkt
com.sun.star.awt.FontUnderline.WAVE	Welle
com.sun.star.awt.FontUnderline.DOUBLEWAVE	doppelte Welle
com.sun.star.awt.FontUnderline.BOLD	einfach Fett
com.sun.star.awt.FontUnderline.BOLDDOTTED	gepunkted fett
com.sun.star.awt.FontUnderline.BOLDDASH	gestrichelt fett
com.sun.star.awt.FontUnderline.BOLDLONGDASH	lang gestrichelt
com.sun.star.awt.FontUnderline.BOLDWAVE	Welle fett

7.2.8 Wie kann man auf die aktuelle Zelle oder einen Bereich zugreifen?

Der Zugriff auf die aktuelle Zelle oder einen markierten Bereich erfolgt über die aktuelle Selektion (getCurrentSelection()). Man erhält ein Objekt der Zelle oder des Bereiches zurück. Um zu prüfen ob es such um eine Zelle oder einen Bereich handelt, verwendet man die Abfrage nach dem Interface com.sun.star.table.XCell. Dies wird nur von dem Zellen-Objekt unterstützt.

```
odoc=thisComponent
oZelle=oDoc.getCurrentSelection()
checkzelle=HasUnoInterfaces( oZelle, "com.sun.star.table.XCell" )
if checkzelle then
  msgbox "Sie haben eine Zelle markiert"
else
 msgbox "Sie haben einen Bereich markiert"
end if
```

7.2.9 Wie kann man den aktuellen Cursor auf eine Zelle setzen?

Wenn man den aktuellen Cursor auf eine bestimmte Zelle stellen will, muß man den CurrentController des Dokumenten-Objektes verwenden.

Den aktiven Cursor bekommt man mit

```
myDoc = thisComponent
myView = myDoc.CurrentController
```

Diesem muß man dann die gewünschte Zelle zuweisen

```
mysheet = myDoc.sheets(0)
mycell =  mysheet.getCellByPosition(0,0)
myView.Select(mycell)
```

7.2.10 Wie kann man Werte einer Zelle lesen und schreiben?

Nachdem man sich den Zugriff auf eine Zelle geholt hat (Siehe: Wie bekomme ich Zugriff zu einer Zelle?), kann über drei verschiedene Properties auf den Zellenihnalt zugreifen: String, Value, Formula.

```
myDoc = thisComponent

mySheet = myDoc.sheets(0)

mycell = mysheet.getCellByPosition(0,0)
```

'Lese der Werte

```
myString=mycell.string
```

Ergibt immer den Inhalt der Zelle als String. Also Achtung bei Zahlen.

myvalue=mycell.value

Ergibt den Zahlenwert, bei Text kommt als Standardwert 0

```
myformula=mycell.formula
```

Ergibt eine Formel, wenn eine hinterlegt ist, ansonsten verhält es sich wie bei dem String.

Schreiben der Werte erfolgt gerade umgekehrt:

mycell.string="test"

Auch hier gilt wieder: Achtung bei Zahlen.

mycell.value=5.23

Hier werden die internen Zahlformate verlangt, also Punkt statt Komma

*mycell.formula="=C2*C2"*

Hier ist das Gleichheitszeichen wichtig, ansonsten wird der Wert wie ein String behandelt.

Achtung: Wenn Calc-Funtionen verwendet werden, müssen diese in der englischen Version verwendet werden. Z.B. SUM für SUMME.
Statt formula die Methode formulalocal verwendet werden. Dann kann man die deutschen Funktionbezeichnungen nutzen.

Die Unterscheidung von Formula und Value ist besonders wichtig beim Auslesen von Werten. Man kann sich so die Formel anzeigen lassen und den Wert verarbeiten.

7.2.11 Wie kann man mit einem Makro eine Formel in eine Zelle eintragen?

Mit der Propertie "Formula" oder "FomulaLocal"
Wenn man die Formel als String übergibt wird die Formel nicht interpretiert. Dazu hat Cell die Propertie "Formula" .

myDoc = thisComponent

mySheet = myDoc.sheets(0)

mycell = mysheet.getCellByPosition(0,0)

*mycell.Formula = "=A2*B2"*

Dabei lassen sich auch Werte aus anderen Zellen, Verweise oder eigenen Berechnungen in die Formel einbinden. Man darf sich durch die eigene Eigenschaft "Forumla" nicht verwirren lassen. Es handelt sich trotzdem um einen String.

myWert="A1"

myZelle="$Tabelle1.A1"

mycell.Formula = "=A2*B2"+"+"+myWert+"*"+myZelle

-> In der Zelle steht dann =A2*B2+A1*$Tabelle1.A1

Die Propertie FormulaLocal muß man verwenden wenn man die deutschen Funktionsbezeichnungen verwenden will. Die englischen Funktionsbezeichnungen gehen nur mit Formula.

Daher also entweder:

mycell.Formula = "=SUM(C1:C4)"

oder

mycell.FormulaLocal = "=SUMME(C1:C4)"

7.2.12 Wie kann man eine Formatvorlage zuweisen?

Zellen und Zellbereiche kann man der Hilfe von Formatvorlagen formatieren.
Diese kann man über die Propertie "CellStyle" direkt zuweisen.

```
Sub Main
oDoc = thiscomponent
osheet=odoc.sheets(0)
mycell=osheet.getcellbyposition(0,0)
mycell.cellstyle="Überschrift1"
End Sub
```

Wenn die Vorlage nicht existiert, findet keine Änderung statt.

7.2.13 Wie kann man auf Zellen mit den Namen zugreifen?

Man kann Zellen und Zellbereichen innerhalb einer Tabelle Namen vergeben. (Einfügen -> Namen).
Über diesen Namen kann man auch auf diese Zellen und Bereiche zugreifen.

Dies erfolgt auch über die Funktion getCellRangeByName.

```
oDoc = thiscomponent
oSheet=oDoc.sheets(0)
```

ocell =osheet.getCellRangeByName("Name")

ocell.string = "Test"

Man muß nur aufpassen, das man prüft ob man eine Zelle oder einen Bereich hat.

Sie dazu: Wie kann man prüfen ob Zellen oder Bereiche in der aktuellen Selektion sind?

7.2.14 Wie kann man Zellen ausrichten?

Dies erfolgt mit der Property ParaAdjust.
odoc=thiscomponent

osheet=odoc.sheets(0)

ocell=osheet.getCellByPosition(1,1)

ocell.ParaAdjust=0

Wobei gilt:

0	Linksbündig (LEFT)
1	Rechtsbündig (RIGHT)
2	Blocksatz (BLOCK)
3	Zentriert (CENTER)
4	Gedehnt (STRETCH)

7.2.15 Wie kann man den Zellnamen bestimmen?

Manchmal brauch man den richtigen Namen einer Zelle (z.B. D1).
Dabei gibt es zwei Fälle. Die Zelle wird durch das Makro festgelegt oder es handelt sich um die aktuelle Zelle.

Fall 1:

myDoc = thisComponent

mySheet = myDoc.sheets(0)

orow=0

mycell=mySheet.getCellByPosition(0,orow)

oColumn=mycell.getColumns.getByIndex(0).getName()

cellname=ocolumn+ltrim(str(orow))

msgbox cellname

oder für eine aktuelle Zelle (Fall 2):

```
myDoc = thisComponent
mySheet = myDoc.sheets(0)
oCelle=myDoc.getCurrentSelection().getCellAddress()
oRow=oCelle.Row
oColumn=oCelle.column
oCelle=mySheet.getCellByPosition(ocolumn,orow)
oColumnname=ocelle.getColumns.getByIndex(0).getName()
cellname=ocolumnname+ltrim(str(orow+1))
msgbox cellname
```

7.2.16 Wie kann man diagonale Striche in eine Zelle einfügen?

Man kann einzelne Zellen mit diagonalen Linien versehen. Mit zwei Linen (Kreuz) oder nur einer. Dazu wird eine Rahmenlinie und die Propertie DiagonalXXX der Zelle verwendet.

```
  mLine = createUnoStruct( "com.sun.star.table.BorderLine" )
  mLine.Color = 0
  mLine.InnerLineWidth = 25
  myDoc = thisComponent
  mySheet = myDoc.sheets(0)
  mycell = mysheet.getCellByPosition(0,0)
' TLBR = Top Left to Bottom Right - Links oben nach rechts unten
  mycell.DiagonalTLBR=mLine
' BLTR = Bottom Left to Top Right - Links unten nach rechts oben
  mycell.DiagonalBLTR=mLine
```

Um eine Linie wieder zu löschen muß man die Linienweite auf 0 setzen

```
  mLine.InnerLineWidth = 25
```

Führt man den Befehle über einen Bereich aus werden alle Zellen einzelnen mit Linien versehen.

7.2.17 Wie kann man den Typ einer Zelle bestimmen?

Zellen können verschiedene Inhalte haben: Eine Formel, eine Zahl, einen String oder sie ist leer. Wenn man den Typ nicht kenn kannt man diesen mit der Type-Eigenschaft abfragen.

Es gibt vier mögliche Werte für den Typ

Type	Konstante	Integerwert
Leere Zelle	com.sun.star.table.CellContentType.EMPTY	0
Zahl	com.sun.star.table.CellContentType.VALUE	1
String	com.sun.star.table.CellContentType.STRING	2
Formel	com.sun.star.table.CellContentType.FORMULA	3

```
dim oCalc as Object

dim oSheet as Object

dim oCell as Object
ocalc = thiscomponent
osheet = ocalc.sheets(0)
ocell = osheet.getCellByPosition(0,0)
msgbox ocell.Type
```

Im Falle das eine Formel vorliegt kann man noch prüfen ob das Ergebnis der Formel ein String ist oder eine Zahl.

```
If ocell.type=3 then
  msgbox ocell.FormulaResultType
end if
```

Hier steht 1 wieder für Zahl und 2 für einen String

7.2.18 Wie kann man den Text einer Zelle drehen?

Dies geht mit der Zelleneigenschaft RotateAngle. Mit dieser kann den Rotationswinkel einstellen.
Die Einstellung erfolgt in 100er Schritten. 100 entspricht einem Grad. 9000 steht also für 90 Grad.

```
myDoc = thisComponent
mySheet = myDoc.sheets(0)
mycell = mysheet.getCellByPosition(0,0)
mycell.RotateAngle  =9000
```

7.2.19 Wie kann man den Inhalt einer Zelle löschen?

Den Zellinhalt kann man mit der Methode clearContents löschen. Hierbei wird über die Addierung von Konstanten der zu löschende Inhalt festgelegt. Die Auswahl entspricht dabei der der Auswahl von "Inhalte lösche" über das Menue.
Folgende Konstaneten für ***com.sun.star.sheet.CellFlags*** gibt es:

Konstante	***Inhalt***	***Integerwert***

com.sun.star.sheet.CellFlags.VALUE	***nummerischer Wert***	***1***
com.sun.star.sheet.CellFlags.DATETIME	***Datum / Uhrzeit***	***2***
com.sun.star.sheet.CellFlags.STRING	***Text***	***4***
com.sun.star.sheet.CellFlags.ANNOTATION	***Notizen***	***8***
com.sun.star.sheet.CellFlags.FOMULA	***Funktionen***	***16***
com.sun.star.sheet.CellFlags.HARDATTR	***Harte Formatierung***	***32***
com.sun.star.sheet.CellFlags.STYLES	***Formatvorlagen***	***64***
com.sun.star.sheet.CellFlags.OBJECTS	***Zeichenobjekte***	***128***

Um alles zu löschen setzt man den Wert auf die Summe 255. (Im Beispiel inK=255)

```
oCalc=thisComponent
 oSheet = oCalc.Sheets(0)
' Zelle A1 liegt auf Position 0,0
 oCell = oSheet.getCellByPosition( 0, 0 )
' addieren der Zellinhaltkonstanen, die gelöscht werden sollen
 inK= com.sun.star.sheet.CellFlags.STRING + com.sun.star.sheet.CellFlags.VALUE
 oCell.clearContents( inK )
```

7.2.20 Wie kann man eine bedingte Formatierung einfügen?

Man kann auch mit StarBasic eine oder mehrere bedingte Formatierungen für Zellen einfügen. Dies erfolgt mit der Propertie ConditionalFormat einer einer Zelle.
Dazu muß man eine Objekt des ConditionalFormat holen und diesem neue Eigenschaften für die Formatierung zuweisen. Die Eigenschaften werden in einem Array mit Propertyvalues erzeugt.
Dabei geht es um drei mögliche Parameter: Operator, Formel und Stylename.

Der Operator legt fest wie mit dem Zielwert verglichen wird. Unterschieden werden dabei:

EQUAL	Gleich	Der Wert entspricht dem folgenden Wert
NOT_EQUAL	Nicht gleich	Der Wert entspricht nicht dem folgenden Wert
GREATER	Größer	Der Wert ist größer wie der folgende Wert
GREATER_EQUAL	Größer gleich	Der ist größer oder gleich dem folgenden Wert
LESS	Kleiner	Der Wert ist kleiner wie der folgende Wert
LESS_EQUAL	Kleiner gleich	Der Wert ist kleiner oder gleich dem folgenden Wert
BETWEEN	Zwischen	Der Wert ist zwischen den beiden folgenden Werten
NOT_BETWEEN	Nicht zwischen	Der Wert ist nicht zwischen den beiden folgenden Werten
FORMULA	Formel	Der Wert komm aus einer Formel

Der Parameter wird mit

oCondition(0).Name = "Operator"
oCondition(0).Value = com.sun.star.sheet.ConditionOperator.FORMULA

festgelegt. Hier mit Formula.

Der nächste Parameter ist Formel. Dieser Parameter enthält einen Wert, ein Verweis auf eine Zelle oder eine Formel.
Wenn der Operator "zwischen" oder "nicht zwischen" ist gibt es zwei Formeln für die Werte.

oCondition(0).Name="Formula1"
oCondition(0).Value=100

oder oCondition(0).Value="Formula2"

Der letzte Parameter ist Stylename. In diesem wird das Zellformat in Form der Zellvorlage eingetragen.

oCondition(2).Name = "StyleName"
oCondition(2).Value = "Überschrift1"

Interessanterweise kann man innerhalb eines Makros mehr wie drei Bedingungen festlegen. Über den Dialog in Calc gehen nur drei Bedingungen.

So und jetzt das ganze in einen Beispiel:

```
Sub EinfuegenConditionalFormCell
Dim oCell as Object
Dim oConditionalForm as Object
oCell = ThisComponent.Sheets(0).getCellByPosition( 0, 0 )
oConditionalForm = oCell.ConditionalFormat
Dim oCondition(3) as New com.sun.star.beans.PropertyValue
oCondition(0).Name = "Operator"
oCondition(0).Value = com.sun.star.sheet.ConditionOperator.FORMULA
oCondition(1).Name = "Formula1"
oCondition(1).Value = "SUM(A1:A10) > 100"
oCondition(2).Name = "StyleName"
oCondition(2).Value = "Überschrift1"
oConditionalForm.addNew( oCondition() )
oCondition(0).Value = com.sun.star.sheet.ConditionOperator.BETWEEN
oCondition(1).Name = "Formula1"
oCondition(1).Value = "100"
oCondition(1).Name = "Formula2"
oCondition(1).Value = "200"
oCondition(2).Name = "StyleName"
oCondition(2).Value = "Test"
oConditionalForm.addNew( oCondition() )
oCell.ConditionalFormat = oConditionalForm
End Sub
```

7.2.21 Wie kann die bedingten Formatierungen löschen?

Dies geht über Methode clear.

```
oCell = ThisComponent.Sheets(0).getCellByPosition( 0, 0 )
oConditionalForm = oCell.ConditionalFormat
oConditionalForm.clear()
oCell.ConditionalFormat = oConditionalForm
```

7.2.22 Wie kann man das Zahlenformat auf Standardformate einstellen?

Dies erfolgt über die Eigenschaft Numberformat.

```
myDoc = thisComponent
mySheet = myDoc.sheets(0)
mycell = mysheet.getCellByPosition(0,0)
mycell.NumberFormat=0
```

Leider habe ich keine komplette Liste über die möglichen Einstellungen gefunden. Daher hier eine Liste der wichtigsten Formate.

	Code	**Ausgabe**	**Numberformat**
Zahlen			
Standard	123	123	0
-1234	0	123	1
-1234,00	0,0	123,00	2
-1.234	#.##	1.233	3
-1.234,00	#.##0,00	1.234,00	4
Prozent			
13%	0%	13%	10
13,00%	0,00%	13,00%	11
Währung			
123€	#.##0 [$€-407];-#.##0 [$€-407]	123€	108
123,00€	#.##0,00 [$€-407];-#.##0,00 [$€-407]	123,00€	109
-123€	#.##0 [$€-407];[ROT]-#.##0 [$€-407]	-123€	110
-123,00€	#.##0,00 [$€-407];[ROT]-#.##0,00 [$€-407]	-123,00€	106
Datum			

31. Dez. 2008	T. MMM. JJJJ	31. Dez. 2008	80
31. Dezember 2008	T. MMMM JJJJ	31. Dezember 2008	76
31.12.08	TT.MM.JJ	31.12.08	37
31.12.2008	TT.MM.JJJJ	31.12.2008	36
Uhrzeit			
12:12:12	HH:MM:SS	12:12:12	41
12:12	HH:MM	12:12	40

Um die in der Liste fehlenden Codes zu erfahren muß eine Zelle darauf einstellen und das Format abfragen.

msgbox.NumberFormat=0

7.2.23 Wie kann man eine Notiz an die Zelle einfügen?

Es gibt zwei Wege Notizen einzufügen. Der erste über getAnnotations mit InsertNew. Ist bereits eine vorhanden wird diese überschrieben.

```
oSheet = thisComponent.sheets(0)
oCell = oSheet.getCellRangeByName("F9")
stCellAddress=oCell.CellAddress
oSheet.getAnnotations.insertNew(stCellAddress, "Das ist eine Info")
```

Der zweite geht über die Zelle dirket.

```
oDoc = thisComponent
oSheet = oDoc.sheets(0)
oCell = oSheet.getCellRangeByName("F1")
oCell.annotation.string="Hallo2"
```

7.2.24 Wie kann man mit CellAdress umgehen?

Für manche Methoden wird zur Übergabe das Struct CellAdress (com.sun.star.table.CellAddress) benötigt. Dabei handelt es sich um die Darstellung einer Zelle in den Werten Tabelle, Zeile und Spalte (sheet,row,clommun). Wenn man auf eine Zell zugegriffen hat kann man diesen Wert direkt über CellAdress abfragen.

```
oSheet = thisComponent.sheets(0)
mycell = osheet.getCellByPosition(0,0)
stCellAddress=myCell.CellAddress
```

Auch wenn man sich eine Zell über den Bereich holt kann sich den Wert direkt holen.

```
oSheet = thisComponent.sheets(0)
oCell = oSheet.getCellRangeByName("F9")
stCellAddress = oCell.CellAddress
```

Alternativ kann man sich das Struct auch mit den Werten selber erzeugen, wenn man die Daten für Zelle

weiß.

dim stCellAddress as new com.sun.star.table.CellAddress
stCellAddress.Sheet=0
stCellAddress.row=1
stCellAddress.column=1

7.2.25 Wie kann man die Farbe einer Notiz ändern?

Wenn eine Notiz vorhanden ist kann die Hintegrundfarbe einstellen. Dabei werden die Parameter in RGB übergeben.

oSheet = thisComponent.sheets(0)
oCell = oSheet.getCellRangeByName("C9")
oCell.Annotation.String="Test"
oShape = oCell.Annotation.AnnotationShape
oShape.FillBackground = true
oShape.FillColor = RGB(223,122,0)
oShape.FillStyle = 1
oShape.FillTransparence = 0

7.2 Bereiche

7.3.1 Wie kann man Zellbereiche sortieren?

Um Zellbereich zu sortieren gibt es die Methode "Sort" eines Bereiches.

Um diese Methode zu verwenden muss man aber einige Parameter setzen. Dazu werden zwei neue Structs erzeugt. Einer für bestimmte Properties und einer für die Festlegung der Sortierfeldes. Man kann auch mehrere Sortierfelder verwenden.

Beim Sortierfeld wird die Spalte, die Sortierrichtung (Aufwärts/Abwärts) und der Feldtyp bestimmt.

Mögliche Feldtypen sind:

com.sun.star.util.SortFieldType.ALPHANUMERIC, Sortierung von Text

com.sun.star.util.SortFieldType.NUMERIC, Sortierung von Zahlen

com.sun.star.util.SortFieldType.AUTOMATIC, Typ wird erkannt

Die Sortierrichtung wird mit "isascending" festgelegt.

True = Aufsteigend

False = Absteigend

Das Feld wird durch die Position, beginnend bei 0, festgelegt.

Zusätzlich kann man noch festlegen ob Groß- und Kleinschreibung beachtet wird. Dies geht aber auch über die Sortierkriterien.

Als nächstes müssen die Properties ergänzt werden.

"SortFields", für das neue erzeugt Struct

"SortColumns", false Zeilen werden sortiert, true Spalten.

"ContainsHeader", wenn true wird die erste Zeile nicht mitsortiert

"IsCaseSensitive", Groß-/Kleinschreibung

```
Sub SortiereBereich
   Dim SortProps(2) As new com.sun.star.beans.PropertyValue
   Dim SortFeld(0) As new com.sun.star.table.TableSortField

   oDatei = ThisComponent
   oSheet = oDatei.Sheets(0)
   oBereich = oSheet.getCellRangeByName("A1:D20")

    SortFeld(0).Field = 0
    SortFeld(0).IsAscending = True
    SortFeld(0).FieldType = com.sun.star.util.SortFieldType.ALPHANUMERIC

     SortProps(0).Name = "SortFields"
     SortProps(0).Value = SortFeld()
     SortProps(1).Name = "SortColumns"
     SortProps(1).Value = False
     SortProps(2).Name = "ContainsHeader"
     SortProps(2).Value = true
     oBereich.Sort(SortProps())
End Sub
```

Möchte man über zwei oder drei Felder (mehr lässt OO/SO nicht zu) sortieren muss dies im Struct entsprechend festgelegt werden.

```
Dim SortFeld(1) As new com.sun.star.table.TableSortField
SortFeld(0).Field = 0
SortFeld(0).IsAscending = True
SortFeld(0).FieldType = com.sun.star.util.SortFieldType.ALPHANUMERIC
SortFeld(1).Field = 1
SortFeld(1).IsAscending = True
SortFeld(1).FieldType = com.sun.star.util.SortFieldType.ALPHANUMERIC
```

7.3.2 Wie geht man mit mehreren selektierten Bereichen um?

Mit getCurrentSelection() kann man die aktuelle Selektion des Anwenders abfragen. Dabei ist es auch möglich das es sich um mehrere Bereiche oder Zellen handelt. (Dazu muß man beim Markieren die Strg-Taste gedrückt halten). Bei diesen Bereichen kann es um Zellen oder Bereiche handeln.
Die Anzahl der Selektioen erhält man mit getcount.

```
oCalc=thisComponent
 oSel=oCalc.getCurrentselection()
 iAnzSel=oSel.getCount
```

Die einzelnen Selektionen kann man dann über den Index aufrufen.

```
myRange=oSel.getByIndex
```

Mit einer Schleife kann man dann alle Selektionen aufrufen. Achtung der Index beginnt bei 0,

```
for i=0 to iAnzSel-1
  myRange=getByIndex(i)
next i
```

Hierbei ist daran zu denken, das diese neue Range auch wieder eine Zelle oder ein Bereich sein kann.

7.3.3 Wie kann man prüfen ob Zellen oder Bereiche in der aktuellen Selektion sind?

Wenn man mit getCurrentselction() den vom Anwender markierten Bereich auswählt, kann es sich dabei in einer Tabelle um verschiedende Bereiche handeln:

eine Zelle,

eine Bereich,

oder mehrere Bereiche.

Um nun heraus zu bekommen mit welcher Art man es zu tun hat muss diese Selektion auf die entsprechenden Services prüfen.

```
oCalc = thisComponent
oSel=oCalc.getCurrentselection()
if oSel.supportsService("com.sun.star.sheet.SheetCell") then
  'Der Cursor ist einer Zelle
  MsgBox "Eine Zelle markiert"
 elseif oSel.supportsService("com.sun.star.sheet.SheetCellRange") then
  ' Ein Bereich
  MsgBox "Ein Bereich wurde markiert"
 elseif oSel.supportsService("com.sun.star.sheet.SheetCellRanges") then
  'Mehrere Bereiche
 MsgBox "Mehrere Bereiche wurden markiert"
end if
```

Siehe dazu auch unter Tools die Funktion getSelTyp

7.3.4 Wie kann man auf Zellbereiche zugreifen?

Es gibt zwei Wege:

getRangeByPosition und getCellRangeByName

Mit getRangeByPosition kommt man an die Zelle über die absolute Positionsangabe, wobei diese über die Spalten- und Zeilenangabe, erfolgt.

getCellRangeByPosition(1,1,2,2) = Bereich B1:C3

Die Zählung fängt dabei bei 0 an A1 ist also 0,0

Mit getCellRangeByName kann man die Zelle direkt mit den Namen ansprechen und damit Bereiche auswählen.

getCellRangeByName("A1:B5") ergibt dann den Bereich A1:B5

Außerdem geht natürlich der Zugriff mit einem vergebenen Namen.

getCellRangeByName("Test)

Man kann auch gleichzeitig mehrere Bereiche auswählen. Dann müssen diese nur mit Komma getrennt eingegeben werden.

getCellRangeByName("A1:B5","C4:D9") ergibt dann den Bereich A1:B5 und C4:D9

Bei dem Rahmenangaben für Rangebyname scheint es egal zu sein ob man oben links und unten rechts oder unten links und oben rechts angibt. Dafür lege ich aber meine Hand nicht ins Feuer.

7.3.5 Wie kann man Zellbereiche löschen?

Zellbereiche lassen sich natürlich auch löschen, nicht nur der Inhalt sondern die Zellen selber. Dabei muß mit einem zusätzlichen Parameter die Art des Löschens festgelegt werden.
Dabei können die Zellen von unten nach oben oder von rechts nach links verschoben werden. Oder man löscht die ganze Zeile oder Spalte.

Die Parameter dafür sind

com.sun.star.sheet.CellDeleteMode.UP, com.sun.star.sheet.CellDeleteMode.LEFT, com.sun.star.sheet.CellDeleteMode.ROWS oder com.sun.star.sheet.CellDeleteMode.COLUMNS

```
oDocument = ThisComponent
oSheet1 = oDocument.Sheets.getByIndex(0)
oQuelleRange=oSheet1.getCellRangeByPosition(0,0,10,10)
oQuellRangeAdresse = oQuelleRange.getRangeAddress
oSheet1.removeRange( oQuellRangeAdresse, com.sun.star.sheet.CellDeleteMode.UP)
```

7.3.6 Wie kann man Zellbereiche kopieren?

Zellbereiche können in Calc mit einer Methode des Sheets-Services kopiert werden: copyRange
Die nötigen Parameter sind der Ursprungsbereich und der Zielbereich. Der Zielbereich wird durch die linke obere Ecke festgelegt.

```
   oDocument = ThisComponent
   oSheet1 = oDocument.Sheets.getByIndex(0)
   oSheet2 = oDocument.Sheets.getByIndex(1)
' Quellbereich festlegen
   oQuelleRange=oSheet1.getCellRangeByPosition(0,0,3,4)
   oQuellRangeAddresse = oQuelleRange.getRangeAddress
   oZiel = oSheet2.getCellByPosition(2,5)
   oZielCellAdresse=oZiel.getCellAddress
   oSheet2.copyRange(oZielCellAdresse,oQuellRangeAddresse)
```

Achtung! Dies geht nicht zwischen zwei Dokumenten! Man kann also Quelle und Ziel nicht in zwei Dokumenten verwenden.

7.3.7 Wie kann man Bereiche verschieben?

Bereiche werden auf eine ähnliche Art verschoben wie sie kopiert werden. Nur das der Befehl moveRange lautet.Die nötigen Parameter sind der Ursprungsbereich und der Zielbereich. Der Zielbereich wird durch die linke obere Ecke festgelegt.

```
  oDocument = ThisComponent
   oSheet1 = oDocument.Sheets.getByIndex(0)
   oSheet2 = oDocument.Sheets.getByIndex(1)
' Quellbereich festlegen
   oQuelleRange=oSheet1.getCellRangeByPosition(0,0,3,4)
   oQuellRangeAddresse = oQuelleRange.getRangeAddress
   oZiel = oSheet2.getCellByPosition(2,5)
   oZielCellAdresse=oZiel.getCellAddress
   oSheet2.moveRange(oZielCellAdresse,oQuellRangeAddresse)
```

Auch hier gilt wieder. Es geht nicht zwischen zwei Dokumenten.

7.3.8 Wie kann man einen Zellbereich verbinden?

Dies geht mt der Methode merge. Mit der Funktion getIsMerged kann man prüfen ob der Bereich bereits verbunden ist.

```
oCalc=thisComponent
 oBereich = oCalc.Sheets(0).getCellRangeByPosition( 0, 0, 4, 0 )
 If oBereich.getIsMerged() then
```

```
  oBereich.merge( false )
 Else
  oBereich.merge( true )
End If
```

7.3.9 Wie kann man in Zellbereichen Berechnungen durchführen?

Innerhalb von Zellbereichen stehen einige Grundberechungen zur Verfügung die man auch innerhalb StarBasic ausführen kann.
Folgende Berechnungen stehen zur Verfügung:

com.sun.star.sheet.GeneralFunction.SUM	Summe über alle Zellen
com.sun.star.sheet.GeneralFunction.MAX	Höchster Wert
com.sun.star.sheet.GeneralFunction.MIN	Niedrigster Wert
com.sun.star.sheet.GeneralFunction.AVERAGE	Durchschnittswert
com.sun.star.sheet.GeneralFunction.COUNTNUMS	Anzahl Werte (Nicht nummerische Zellen werden nicht gezählt)
com.sun.star.sheet.GeneralFunction.COUNT	Anzahl aller Zellen mit Inhalt

Weiter Berechnungen stehen unter com.sun.star.sheet.GeneralFunction in der Referenz. Wobei PRODUCT immer den selben Wert erzeugt.

```
oCalc=thisComponent
 oSelektion = oCalc.Sheets(0).getCellRangeByPosition(0,0,2,2)
 sMessage = oSelektion.computeFunction(com.sun.star.sheet.GeneralFunction.SUM)
 msgbox sMessage
```

7.3 Arbeitsblätter

7.4.1 Wie kann man auf Sheets (Arbeitsblätter) zugreifen?

Es gibt zwei Wege um auf Arbeitsblätter zuzugreifen:
Über den Namen oder über die interne Nummer.

Über den Namen:

```
myDoc = thisComponent

mySheet = myDoc.Sheets().getByName("Tabelle2")
```

Über die interne Nummer:

```
myDoc = thisComponent
mySheet = myDoc.Sheets(0)
```

Die Zählung der Sheets beginnt bei 0.

7.4.2 Wie kann man das aktuelle Sheet oder alle ermitteln?

Um einfach Zugriff auf das aktuelle Sheet zu erhalten genügt die Referenzierung von odoc.currentcontroller.activesheet

osheet=odoc.currentcontroller.activesheet

Möchte man die Anzahl aller Sheets und/oder die Position innerhalb der Sheets ermitteln, geht dies mit der Hilfe von zwei Properties, einer von Document und einer von Sheets.

Um die gesamte Anzahl der Sheets zu ermitteln genügt sheets.count:

```
myDoc = thisComponent
Anzahl=myDoc.Sheets.count
```

Um jetzt noch die Namen der einzelnen Sheets auszulesen, muß man nur eine Schleife über die Gesamtzahl der Sheets laufen lassen.

```
myDoc = thisComponent
Anzahl=myDoc.Sheets.count
For i=0 to Anzahl-1
   mySheet = myDoc.Sheets(i)
   msgbox mysheet.name
Next i
```

Möchte man die Position des aktuellen Sheet ermitteln, muß man einen Vergleich der Namen mit dem Namen des aktiven Sheets machen.

Der Name des aktiven Sheets steht in odoc.currentcontroller.activesheet.name.

Ich habe dazu die Funktion GetPosActiveSheet geschrieben. Sie ist bei den Makros und Tools zu finden

Siehe auch bei Makros und Tools nach: Sub GetNameOfAllSheets(NameOfSheets())

7.4.3 Wie kann man per Makro ein Arbeitsblatt aktiveren?

Um ein Arbeitsblatt per Makro zu aktivieren, muß man den CurrentController bemühen.

Dann kann man entweder über den Namen

```
myDoc = ThisComponent
myView = myDoc.CurrentController
mySheet = myDoc.Sheets.getByName(sheetsname)
myView.setActiveSheet(mySheet)
```

oder über den Index darauf zugreifen.

```
myDoc = ThisComponent
myView = myDoc.CurrentController
mySheet = myDoc.Sheets(sheetsnumber)
myView.setActiveSheet(mySheet)
```

Zwei Makros dazu stehen in den Tools.

7.4.4 Wie kann man Arbeitsblätter löschen, erzeugen,kopieren und verschieben?

Löschen:

Löschen kann man ein Sheet über die Funktion removebyname.
Am Besten man prüft vorher mit hasbyname ob das Sheet auch wirklich vorhanden ist.

```
myDoc = thisComponent

If myDoc.Sheets.hasByName("Tabelle3") Then
  mydoc.Sheets.removebyname("Tabelle3")
end if
```

Erzeugen:

Es gibt zwei Wege ein neues Sheet einzufügen

a.

Erzeugt wird ein Sheet zurerst mit createinstance und danach folgendem insertbyname.

myDoc = thisComponent

Sheet=MyDoc.createInstance("com.sun.star.sheet.Spreadsheet")

myDoc.Sheets.insertByName("MySheet", Sheet)

b.

Mit der Methode insertNewByName. Bei dieser kann man zusätzlich noch als Parameter den Index des Arbeitsblattes angeben, vor dem das neue Blatt eingefügt werden soll.

myDoc.Sheets.insertNewByName("MySheet", 1)

Achtung die Zählung der Blätter beginnt bei Null.

Kopieren:

Kopieren erfolgt mit copybyname.

Die Parameter sind: Originalname, neuer Name, Position

myDoc = thisComponent

myDoc.Sheets.CopyByName("Tabelle3","Tabelle10",0)

Verschieben:

Verschoben werden kann ein Sheet mit movebyname.

myDoc = thisComponent

myDoc.Sheets.moveByName("Tabelle3",0)

Der zweite Parameter gibt die Stelle an, auf die das Sheet verschoben werden soll.

7.4.5 Wie kann man auf selektierte Tabellenblätter zugreifen?

Man kann in Calc mit der Maus mehrere Tabellenblätter markieren. Auf diese kann man auch zugreifen. Die aktuelle Selektion kann mit getCurrentSelection() erhalten.

```
oDoc = ThisComponent
oSelect = oDoc.getCurrentSelection()
```

Nun hat man aber eigenltich nicht die Tabellenblätter ausgewählt sondern eine Zelle oder einen Zellenbereich. Das Tabellenblatt steht in diesem Service.

Innerhalb der Selektion stehen nun die ausgewählten Zellen über den Index zu Verfügung.

0 = Zelle/Bereiche des ersten gewählten Blattes

1 = Zelle/Bereiche des zweiten gewählten Blattes

usw.

Über das Zell oder Bereichsobjekt kommt man dann an das eigentliche Ziel: das Tabellenblatt.

```
Zelle=oSelect.getByIndex(0)
oBaltt=Zelle.getSpreadsheet()
```

Nun das ganze in einem und in einer Schleife:

```
oDoc = ThisComponent
oSelect = oDoc.getCurrentSelection()
for i=0 to oSelect.Count-1
 Zelle=oselect.getbyindex(i)
 oBlatt=   Zelle.getspreadsheet()
' ......
' jezt das Blatt bearbeiten
' .....
next i
```

Leider habe ich noch nicht herausbekommen, wie man prüfen kann, das tatsächlich auch Tabellenblätter markiert sind.

Markiert man mehrere Bereiche in einem Arbeitsblatt, wird in der Schleife immer nur dieses bearbeitet.

7.4.6 Wie kann man Grafiken in ein Sheet einbetten?

Also versuche ich's mal.

Grafikobjekte liegen innerhalb eines CALC-Sheets in dem Object DrawPage. Jedes Sheet hat eine eigene Drawpage.

'Aufruf des Sheets und der Drawpage
Doc = thisComponent
mySheet = Doc.Sheets(0)
Page = mySheet.drawPage

Um auf eine einzelne Grafik zuzugreifen, muß man den Index innerhalb der Drawpage wissen. Um die Grafik zuzuweisen wird eine Instanz des Objekts benötigt.

Grafik= doc.createInstance("com.sun.star.drawing.GraphicObjectShape")
Grafik = Page.getbyIndex(0)

Die wichtigste Information dieses Objektes ist die GraphicURL. Diese wird verwendet um ein neues Graphikobjekt zu erzeugen und einzufügen.
Das erste Element kann man nicht in eine zweite Drawpage einfügen.
(Zumindestens habe ich es nicht hinbekommen).

NeueUrl=Grafik.GraphicURL

Jetzt wird ein neues Grafikshape erzeugt.

NewGrafik=
doc.createInstance("com.sun.star.drawing.GraphicObjectShape")

Für diese braucht man noch die Zieladresse und Höhe/Größe

Dim Point As New com.sun.star.awt.Point

Dim Size As New com.sun.star.awt.Size

Point.x = 1000

Point.y = 1000

Size.Width = 10000

Size.Height = 10000

Zuweisen der Werte

NewGrafik.GraphicURL=NeueUrl

NewGrafik.Position=Point

NewGrafik.Size=Size

Neues Sheet aktiveren und einfügen

mySheet = Doc.Sheets(1)

Page = mySheet.drawPage

Page.add(NewGrafik)

So jetzt das Ganze noch mal in einem Stück:

(Voraussetzung: In Sheet 1 ist eine Grafik)

sub GrafikausSheet1nach2

Doc = thisComponent

mySheet = Doc.Sheets(0)

Page = mySheet.drawPage

Grafik= doc.createInstance("com.sun.star.drawing.GraphicObjectShape")

Grafik = Page.getbyIndex(0)

NeueUrl=Grafik.GraphicURL

NewGrafik=

doc.createInstance("com.sun.star.drawing.GraphicObjectShape")

Dim Point As New com.sun.star.awt.Point

Dim Size As New com.sun.star.awt.Size

```
Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000
NewGrafik.GraphicURL=NeueUrl
NewGrafik.Position=Point
NewGrafik.Size=Size
mySheet = Doc.Sheets(1)
Page = mySheet.drawPage
Page.add(NewGrafik)
end sub
```

7.4.7 Wie kann ich die Ausrichtung einer Zelle oder eines ganzen Blattes ändern?

Um die Ausrichtung einer Zelle per Makro zu ändern muß der Wert für die Propertie HoriJustify geändert werden.

Entweder für das ausgewählte Blatt (Sheet) ,für eine einzelne Zelle oder für einen Zellbereich

```
sub ZellenAusrichtung
' Dieses Makro in einer geöffnetten Tabelle starten
myDoc = thisComponent

mySheet = myDoc.sheets(0)
mySheet.HoriJustify=3

mycell = mysheet.getCellByPosition(0,0)
mycell.HoriJustify=1

mycells =mysheet.getCellRangeByName("A1:B5")
mycells.horijustify=0

end Sub
```

Mögliche Werte für HoriJustify sind:

0= Standard

1= Linksbündig

2= Zentriert

3= Rechtsbündig

4=Blocksatz

7.4.8 Wie kann Arbeitsblätter ausblenden?

Mit der Eigenschaft isVisible.

```
myDoc = thisComponent
mySheet = myDoc.Sheets(0)
'oder
mySheet = myDoc.Sheets().getByName("Tabelle1")
mysheet.isvisible=false
```

Einblenden dann mit True.

Die Indexierung bleibt auch bei ausgeblendeten Blättern erhalten.

7.4.9 Wie kann man das Gitter ausblenden?

Das Hintergrundgitter (Grid) lässt sich im Service des Currentcontrollers des Calc-Documentes aus- und einblenden.
Diese Einstellung gilt dann für alle Arbeitsblätter.

```
    oDoc = ThisComponent
    oDoc.CurrentController.ShowGrid = true
'oder   oDoc.CurrentController.ShowGrid = false
```

7.4.10 Wie kann man in Arbeitsblättern suchen?

Jedes Arbeitsblatt unterstützt den Service Searchdescriptor. Mit diesem kann man die Zellen durchsuchen. Dies muß für jedes Arbeitsblatt erfolgen.
Dann am besten in einer Schleife.

```
odoc = thiscomponent
For iSheets = 0 To oDoc.sheets.Count -1
    oSheet = oDoc.Sheets( iSheets )
    oSearchDescriptor = oSheet.createSearchDescriptor
    oSearchDescriptor.SearchString = "XXXX"
    oFound = oSheet.FindFirst( oSearchDescriptor )
    Do Until isNull(oFound)
        ' oFound ist die Zelle auf die die Suche zutrifft
        oFound = oSheet.FindNext( oFound, oSearchDescriptor )
    Loop
Next iSheets
```

7.4.11 Wie kann man alle Seitenumbrüche löschen?

Mit dem Befehl removeAllManualPageBreaks.
Damit werden alle manuell eingefügten Seitenumbrüche eines Tabellenblattes gelöscht.

```
odoc=thiscomponent
mySheet = oDoc.Sheets(0)
mySheet.removeAllManualPageBreaks
```

7.4.12 Wie kann man einen Seitenumbruch einfügen?

Es gibt zwei Arten von Seitenumbrüchen innerhalb eines Arbeitsblattes; vertikal und horizontal.
Ein vertikaler Umbruch wird an einer Zeile eingefügt und ein horizontaler wird an einer Spalte eingefügt.
In beiden Fällen gilt der Umbruch für die ausgewählte Zeile oder Spalte. Daß heißt die Zeile oder Spalte ist die erste Zeile oder Spalte auf der neuen Seite.

Um einen Seitenumbruch einzufügen muss der Paramater IsStartOfNewPage auf Wahr gesetzt werden.

Für eine Zeile geht es mit:

```
odoc=thiscomponent
oSheet = odoc.sheets(0)
```

```
oRow = oSheet.getRows().getByIndex(4)
oRow.IsStartOfNewPage=true
```

Für eine Spalte geht es mit:

```
odoc=thiscomponent
oSheet = odoc.sheets(0)
oRow = oSheet.getRows().getByIndex(4)
oRow.IsStartOfNewPage=true
```

7.4.13 Wie kann man den Namen eines Arbeitsblattes ändern?

Dies geht mit der Eigenschaft Name des Blattes.

```
ocalc=thiscomponent
mySheet=ocalc.sheets(0)
mySheet.name="Neuer Name"
```

7.4.14 Wie kann man die letzte verwendete Zeile/Spalte ermitteln?

Die geht mit der Funktion "GotoEndOfUsedArea" des Cursors. Mit dieser Funktion springt der Cursor an das Ende der Tabelle und man kann die letzte Zeile und Spalte ermitteln.

```
oDoc = thisComponent
Sheets=odoc.sheets(i)
oCellCursor = Sheets.createCursor
oCellCursor.GotoEndOfUsedArea(False)
nRow = oCellCursor.getRangeAddress().endRow
msgbox nRow
nEndColumn = oCellCursor.getRangeAddress().endColumn
msgbox nEndColumn
```

7.4.15 Wie kann man alle Daten eines Blattes löschen?

Dies geht mit clearContents.

```
odoc=thiscomponent
osheet=odoc.sheets(0)
osheet.clearContents(255)
```

osheet.clearContents(***CellFlags***)

Folgende Parameter gelten als Konstanten für ***com.sun.star.sheet.CellFlags*** gibt es,
um sie zu kombineren müssen die Werte addiert werden:

Konstante	***Inhalt***	***Integerwert***
com.sun.star.sheet.CellFlags.VALUE	***nummerischer Wert***	***1***
com.sun.star.sheet.CellFlags.DATETIME	***Datum / Uhrzeit***	***2***
com.sun.star.sheet.CellFlags.STRING	***Text***	***4***
com.sun.star.sheet.CellFlags.ANNOTATION	***Notizen***	***8***
com.sun.star.sheet.CellFlags.FOMULA	***Funktionen***	***16***
com.sun.star.sheet.CellFlags.HARDATTR	***Harte Formatierung***	***32***
com.sun.star.sheet.CellFlags.STYLES	***Formatvorlagen***	***64***
com.sun.star.sheet.CellFlags.OBJECTS	***Zeichenobjekte***	***128***

Um alles zu löschen setzt man den Wert auf die Summe 255. (Im Beispiel 255)

7.4 Spalten/Zeilen

7.5.1 Wie kann man die optimale Breite einer Spalte einstellen?

Gerade wenn man Daten mit einem Makros einträgt kann es sinnvoll sein dafür zu sorgen das die Spalte die optimale Breite hat.
Die erfolgt auf der Arbeitsblattebene.

```
oCalc = ThisComponent
oSheet = oCalc.Sheets(0)
oSheet.Rows(0).OptimalWidth = True
```

7.5.2 Wie kann man die Anzahl Spalten oder Zeilen eines Ranges auslesen?

Der Service Range der eine bestimmte Anzahl von Zeilen und Spalten enthält hat das Interface getRows und getColumns. Mit diesen kann man sich eine Sammlung der Zeilen oder Spalten holen. Diese bietet das Interface getcount.

oCalc = thiscomponent

oSheet=oCalc.Sheets(0)

oRange=oSheet.getCellRangeByName("A1:B5")

oRows=oRange.getRows

AnzahlRows=oRows.getCount()

Analog geht es dann mit getColumns

oCols=oRange.getColumns

AnzahlCols=oCols.getCount()

7.5.3 Wie kann man Zeilen oder Spalten ausblenden?

Der Zugriff auf Zeilen und Spalten erfolgt über den Service com.sun.star.table.TableColumns bzw. com.sun.star.table.TableRows. Mit der Funtktion getRow bzw getcolumn kann man dann über den Namen auf die entsprechende Spalte oder Zeile zugreifen. Dabei ist zu Unterscheiden zwischen dem gezielten Zugriff auf eine Zeile und eine Spalte. Die Zeile kann über den Index angesprochen werden: getbyIndex. Die Spalte kann über den Namen, getbyName oder den Index, getbyindex erreicht werden.
Über die Propertie isvisbile schaltet dann mit true oder false die Zeile ein bzw. aus.

oSheet = thisComponent.sheets(0)

oSpalte1 = oSheet.getColumns().getByName("M")

oSpalte1.isVisible = false

oSpalte2 = oSheet.getColumns().getByName("G")

oSpalte2.isVisible = false

oZeile = oSheet.getrows().getByIndex(8)

oZeile.isVisible = false

7.5.4 Wie kann auf ganze Spalten oder Zeilen zugreifen?

Der Zugriff auf Zeilen und Spalten erfolgt über den Service com.sun.star.table.TableColumns bzw. com.sun.star.table.TableRows.
Der Zugriff kann dann bei der Spalte über den Namen oder den Index erfolgen.

oSheet = thisComponent.sheets(0)

oColumn = oSheet.getColumns().getByName("C")

'oder

oColumn = oSheet.getColumns().getByIndex(2)

Alternative kann man auch direkt über die Spalten des Sheets zugreifen:

oColumn = oSheet.Columns(0)

Der Zugriff kann dann bei der Zeile über den Index erfolgen.

oSheet = thisComponent.sheets(0)

oRow = oSheet.getRows().getByIndex(2)

Auch gibt es die Alternative über die Zeilen des Sheets.

oColumn = oSheet.rows(0)

7.5.5 Wie kann man Zeilen oder Spalten kopieren?

Es scheint in Starbasic keine Funktion dafür zu geben. Zumindestens habe ich bis jetzt keine gefunden. Es geht aber über einen Zellbereich.
Eine Zeile oder Spalte ist ja eigentlich auch nur ein Bereich. Dieser wird erfasst und dann mit der copyRange-Methode kopiert.

Also hier die Anpassung:

```
   oDocument = ThisComponent
   oSheet1 = oDocument.Sheets.getByIndex(0)
   oSheet2 = oDocument.Sheets.getByIndex(1)
' Quellbereich festlegen
' Zeile Format : ( 0,Zeile,255,Zeile)
   oQuelleRange=oSheet1.getCellRangeByPosition(0,0,255,0)
   oQuellRangeAddresse = oQuelleRange.getRangeAddress
   oZiel = oSheet2.getCellByPosition(0,0)
   oZielCellAdresse=oZiel.getCellAddress
   oSheet2.copyRange(oZielCellAdresse,oQuellRangeAddresse)
```

Entsprechend bei einer Spalte

```
' Spalte Format : ( Spalte,0,Spalte,31999)
```

oQuelleRange=oSheet1.getCellRangeByPosition(0,0,0,31999)

Möchte man die Zeile neu einfügen muß man vorher diese noch leer einfügen. Siehe: Wie kann Zeilen und Spalten einfügen oder löschen?

7.5.6 Wie kann man Zeilen und Spalten einfügen oder löschen?

Der Zugriff auf Zeilen und Spalten erfolgt über den Service com.sun.star.table.TableColumns bzw. com.sun.star.table.TableRows.
Mit diesem Serivce erhält man den Zugriff auf alle Zeilen und Spalten. Man kann dabei leere Zeilenund Spalten nach dem Index einfügen oder vorhandene löschen.

Zeilen

```
  odoc=thiscomponent
  mysheet=odoc.sheets(0)
  myrows=mysheet.getrows
' Und jetzt einfügen: Index und Anzahl.
  myrows.insertbyindex(2,4)
'Und wieder löschen
  myrows.removebyindex(2,4)
```

Spalten:

```
  odoc=thiscomponent
  mysheet=odoc.sheets(0)
  myrows=mysheet.getcolumns
' Und jetzt einfügen: Index und Anzahl.
  mycolumns.insertbyindex(2,4)
'Und wieder löschen
  mycolumns.removebyindex(2,4)
```

Bei der Zählung daran denken: der Index beginnt bei 0

7.5.7 Wie bekommt man die aktuelle Zeile und Spalte heraus?

Über die aktuelle Selection und die dann abrufbare Zelladresse:

```
sub getCellPosition
    oDoc=thisComponent
    ' aktive Zelle
    oCelle=oDoc.getCurrentSelection().getCellAddress()
   ' aktuelle Zeile, Index
     oRow=oCelle.Row
     '  aktuelle Spalte, Index
     oColumn=oCelle.column
end sub
```

Achtung die Zählung beginnt eigentlich bei 0. Das Ergebnis für die erste Spalte/Zeile ist 0.

Siehe auch Tools vom Dannenhöfer

7.5.8 Wie kann man die optimale Höhe einer Zeile einstellen?

Die erfolgt auf der Arbeitsblattebene.

```
oCalc = ThisComponent
oSheet = oCalc.Sheets(0)
oSheet.Rows(1).OptimalHeight = True
```

7.5 Allgemein

7.6.1 Wie kann man in Calc-Dokumenten suchen und ersetzen?

Innerhalb eines Calc-Dokumentes muß man vorher festlegen in welchem Bereich (Zellbereich oder Arbeitsblatt) man ersetzen will. Erst dann steht der Replacedescriptor zur Verfügung, der ein Suchen und Ersetzen ermöglicht.

```
sub ersetzen
'Arbeitsblatt
oSheet =ThisComponent.sheets(0)
oSuchen = oSheet.createReplaceDescriptor
oSuchen.setSearchString("41")
oSuchen.setReplaceString("411")
```

```
osheet.replaceAll(oSuchen)


'oder Zellbereich
oSheet =ThisComponent.sheets(0)
oRange = osheet.getCellRangeByName("A1:B5")
oSuchen = oRange.createReplaceDescriptor
oSuchen.setSearchString("41")
oSuchen.setReplaceString("411")
orange.replaceAll(oSuchen)
end sub
```

Möchte man in alle Arbeitsblätter Suchen und Ersetzen muß man eine Schleife über alle Arbeitsblätter durchführen.

```
myCalc = thisComponent
  Anzahl=myCalc.Sheets.count
  For i=0 to Anzahl-1
    oSheet = myCalc.Sheets(i)
    oSuchen = oSheet.createReplaceDescriptor
   oSuchen.setSearchString("41")
   oSuchen.setReplaceString("411")
   osheet.replaceAll(oSuchen)
Next i
```

7.6.2 Wie kann man einen Druckbereich festlegen?

Man kann den Druckbereich über den Adressbereich eines Rangeobjektes zuweisen.

```
oCalc = thiscomponent
oSheet = oCalc.sheets(0)
oZellRange = oSheet.getCellRangeByName("$A$1:$B$5")
oCursor = oSheet.createCursorByRange(oZellRange)
oAdress = oCursor.RangeAddress
Dim aDruckbereich(0)
aDruckbereich(0) = oAdress
```

oSheet.setPrintAreas(aDruckbereich())

7.6.3 Wie kann man Notizen eintragen oder löschen?

Zu jeder Zelle kann man zusätzlich Notizen hinterlegen. Dann het die Zelle einen kleinen roten Punkt.

Dies geht natürlich auch mit Starbasic. Erst mal der Weg aus der Original-Dokumentation.

Notizen sind Objekte des Arbeitsblattes die Zelladressen zugeordnet sind. Über den Container (com.sun.star.sheet.CellAnnotations) der alle Notizen enthält, kann man neue Notizen zu einer Zelle hinzufügen oder Notizen über den Index löschen.

```
Sub InsertNotiz
  oDoc = thisComponent
  oSheet = oDoc.sheets(0)
  oZelle = oSheet.getCellRangeByName("C1")
'oder   oZelle=oSheet.getCellByPosition(0,0)
  oZellAdresse = oZelle.CellAddress
  oNotizen=oSheet.getAnnotations
  oNotizen.insertNew(oZellAdresse, "Eine Notiz")
End Sub
```

Ist bereits eine Notiz zu der Zelle vorhanden, wird der Inhalt überschrieben.

Löschen geht dann mit removebyindex.

```
  oNotizen=oSheet.getAnnotations
  oNotizen.removebyindex(1)
```

Leider kann man nicht unbedingt mit dem Index auf die gewünschte Zelle zugreifen. Daher muß man wenn man den Index nicht kennt, über alle Zellen gehen und damit den richtigen Index herausbekommen. Dies geht über die Enumaration des Containers und ist daher etwas umständlich. Zum Schluß kommt noch eine schnellere Alternative.

```
sub loescheNotiz
' Für Zelle a1
```

```
  zeile=0
  spalte=0
  oDoc = thisComponent
  oSheet = oDoc.sheets(0)
  oNotizen=oSheet.getAnnotations
  listNotizen=oNotizen.createenumeration
  indexzaehler=0
  while listNotizen.hasmoreelements
   oNotiz.nextelement()
   if oNotiz.position.row=zeile and oNotiz.position.column=spalte then index=indexzaehler
   indexzaehler=indexzaehler+1
  wend
  oNotizen.removebyIndex(index)
end sub
```

Jetzt der schnelle Weg.

Der schnellere Weg geht über die Zelle. Dort kann man einfachen den Text der Notiz setzen, damit wird die Notiz eingefügt. Ist die Zeile leer wird die Notiz gelöscht.

```
sub notizaendern
  oDoc = thisComponent
  oSheet = oDoc.sheets(0)
  oZelle = oSheet.getCellRangeByName("F1")
  oNotiz=ozelle.getannotation
  oNotiz.string="text"
' oder zum löschen
'  oNotiz.string=""
end sub
```

7.6.4 Wie kann man auf die Kopf- und Fußzeile zugreifen?

Der Zugriff erfolgt über die Formatierung der Seite. Kopf und Fußzeile sind Bestandteile der Seiteneinstellungen (Format -> Seite)

Jetzt wird es etwas unübersichtlich. Die Kopf- und Fusszeilen sind schon mal zwei Bereiche der Seite. Diese haben selber wieder drei Teile (links, rechts, mitte).

Zusätzlich kann man noch verschiedende Kopf- und Fusszeilen für die rechte und linke Seite einsetzen.

Also erstmal die Seiteneinstellungen holen:

```
Doc = thiscomponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Standard")
```

Kopf/Fusszeile aktivieren

```
DefPage.HeaderIsOn = True
DefPage.FooterIsOn = True
```

Aktivieren wenn linke und rechte Seite verschieden sein sollen

```
DefPage.HeaderIsShared = False
```

Daten für die Kopfzeile eintragen

```
HContent = DefPage.RightPageHeaderContent
HContent.RightText.String = "rechter text"
HContent.LeftText.String = "linker text"
HContent.CenterText.String = "mitte"
DefPage.RightPageHeaderContent = Hcontent
```

Wenn eine linke Seite einen anderen Text bekommen soll:

```
HContent = DefPage.LeftPageHeaderContent
HContent.RightText.String = "rechter text"
HContent.LeftText.String = "linker text"
HContent.CenterText.String = "mitte"
DefPage.LeftPageHeaderContent = Hcontent
```

Für die Fusszeile dann nochmal mit:

RightPageFooterContent und LeftPageFooterContent

7.6.5 Wie kann man vorhandene Formeln ändern?

Man kann auch vorhandene Formeln in Zellen ändern. Dazu muß man diese auslesen.Den String kann man dann normal wie jeden String bearbeiten und zurückschreiben.

7.6.6 Wie lauten die englischen Funktionsbezeichungen?

Übersicht über die Calc-Funktionen in Deutsch und Englisch (Fehler kann ich nicht ausschließen :-))
Diese Liste als calc-dokument oder als pdf-Datei.

Datenbank

Deutsch	Englisch
DBANZAHL	DCOUNT
DBANZAHL2	DCOUNT2
DMAX	DMAX
DMIN	DMIN
DBPRODUKT	DPRODUCT
DBSUMME	DSUM
DBVARIANZ	DVAR
DBVARIANZEN	DVARP
DBSTDABW	DSTEV
DBSTDABWN	DSTEVP
DBAUSZUG	DGET
DBMITTELWERT	DAVERAGE

Datum&Zeit

Deutsch	Englisch
ARBEITSTAG	WORKDAY
BRTEILJAHER	YEARFRAC

DATUM	DATE
DATWERT	DATEVALUE
EDATUM	EDATE
HEUTE	TODAY
ISTSCHALTJAHR	ISLEAPYEAR
JAHR	YEAR
JAHRE	YEARS
JETZT	NOW
KALENDERWOCHE	WEEKNUM
KALENDERWOCHE_ADD	WEEKNUM_ADD
MINUTE	MINUTE
MONAT	MONTH
MONATE	MONTHS
MONATSENDE	EOMONTH
NETTOARBEITSTAGE	NETWORKDAYS
OSTERSONNTAG	EASTERSUNDAY
SEKUNDE	SECOND
STUNDE	HOUR
TAG	DAY
TAGE	DAYS
TAGE360	DAYS360
TAGEIMJAHR	DAYSINMOUNTH
TAHEIMONAT	DAYSINYEAR
WOCHEN	WEEKDAY
WOCHENIMJAHR	WEEKSINYEAR
WOCHENTAG	WEEKDAY
ZEIT	TIME
ZEITWERT	TIMEVALUE

Finanz

Deutsch	Englisch
AMORDEGRK	AMORDEGRC
AMORLINEARK	AMORLINC

AUFGELZINS	ACCRINT
AUFGELZINSF	ACCRINTM
AUSZAHLUNG	RECEIVED
BW	PV
DIA	SYD
DISAGIO	DISC
DURATION_ADD	DURATION_ADD
EFFEKTIV	EFFECT
EFFEKTIV_ADD	EFFECT_ADD
GDA	DB
GDA2	DDB
IKV	IRR
ISPMT	ISPMT
KAPZ	PPMT
KUMKAPITAL	CUMIPT
KUMKAPITAL_ADD	CUMIPMT_ADD
KUMZINSZ	CUMPRINC
KUMZINSZ_ADD	CUMPRINC_ADD
KURS	PRICE
KURSDIAGIO	PRICEDISC
KURSFÄLLIG	PRICEMAT
LAUFZEIT	DURATION
LIA	SLN
MDURATION	MDURATION
NBW	PPMT
NOMINAL	NOMINAL
NOMINAL_ADD	NOMINAL_ADD
NOTIERUNGBRU	DOLLARFR
NOTIERUNGDEZ	DOLLARDE
QIKV	MIRR
RENDITE	YIELD
RENDITEDIS	YIELDISC
RENDITEFÄLL	YLELDMAT
RMZ	PMT
TBILLÄQUIV	TBILLEQ
TBILLKURS	TBILLPRICE
TBILLRENDITE	TBILLYIELD
UNREGERKURS	ODDPRICE

UNREGERREND	ODDYIELD
UNREGLEKURS	ODDLPRICE
UNREGLEREND	ODDLYIELD
VDB	VDB
XINTZINSFUSS	XIRR
XKAPITALWERT	XNPV
ZGZ	RRI
ZINS	RATE
ZINSSATZ	INRATE
ZINSTERMNZ	COUPPCD
ZINSTERMTAGE	COUPDAYS
ZINSTERMTAGNZ	COUPDAYSNC
ZINSTERMTAGVA	COUPDAYBS
ZINSTERMVZ	COUPNCD
ZINSTERMZAHL	COUPNUM
ZINSZ	IPMT
ZW	FV
ZW2	FVSECHEDULE
ZZR	NPER

Information

Deutsch	Englisch
AKTUELL	CURRENT
FORMEL	FORMULA
ISTBEZUG	ISREF
ISTFEHL	ISERR
ISTFEHLER	ISERROR
ISTFORMEL	ISFORMULA
ISTGERADE_ADD	ISEVEN_ADD
ISTKTEXT	ISNONTEXT
ISTLEER	ISBLANK
ISTLOG	ISLOGICAL
ISTNV	ISNA
ISTTEXT	ISTEXT
ISTUNGERADE_ADD	ISODD_ADD
ISTZAHL	ISNUMBER
N	N

NV	NA
TYP	TYPE
ZELLE	CELL

Logisch

Deutsch	Englisch
FALSCH	FALSE
NICHT	NOT
ODER	OR
UND	AND
WAHR	TRUE
WENN	IF

Mathematik

Deutsch	Englisch
ABRUNDEN	ROUNDDOWN
ABS	ABS
ANZAHLLEEREZELLEN	COUNTBLANK
ARCCOS	ACOS
ARCCOSHYP	ACOSH
ARCCOT	ACOT
ARCOTHYP	ACOTH
ARCSIN	ASIN
ARCSINHYP	ASINH
ARCTAN	ATAN
ARCTAN2	ATAN2
ARCTANHYP	ATANH
AUFRUNDEN	ROUNDUP
COS	COS
COSHYP	COSH
COT	COT
COTHYP	COTH
DEG	DEGRES
EXP	EXP
FAKULTÄT	FACT
GANZZAHL	INT

GERADE	EVEN
GGT	GCD
GGT_ADD	GCD_ADD
ISTGERADE	ISEVEN
ISTUNGERADE	ISODD
KGV	LCM
KGV_ADD	LCM_ADD
KOMBINATIONEN	COMBIN
KOMBINATIONEN2	COMBINA
KÜRZEN	TRUNC
LN	LN
LOG	LOG
LOG10	LOG10
OBERGRENZE	CEILING
PI	PI
POLYNOMIAL	MULTINOMIAL
POTENZ	POWER
POTENZREIHE	SERIESSUM
PRODUKT	PRODUCT
QUADRATESUMME	SUMSQ
QUOTIENT	QUOTIENT
RAD	RADIANS
REST	MOD
RUNDEN	ROUND
SIN	SIN
SINHYP	SINH
SUMME	SUM
SUMMEWENN	SUMIF
TAN	TAN
TANHYP	TANH
TEILERGEBNIS	SUBTOTAL
UMRECHNEN	CONVERT
UNGERADE	ODD
UNTERGRENZE	FLOOR
VORZEICHEN	SIGN
VRUNDEN	MROUND
WURZEL	SQRT
WURZELPI	SQRTPI
ZÄHLENWENN	COUNTIF

ZUFALLSBEREICH	RANSBETWEEN
ZUFALLSZAHL	RAND

Matrix

Deutsch	Englisch
EINHEITSMATRIX	MUNIT
HÄUFIGKEIT	FREQUENCY
MDET	MDETERM
MINV	MINVERSE
MMULT	MMULT
MTRANS	TRANSPOSE
RGP	LINEST
RKP	LOGEST
SUMMENPRODUKT	SUMPRODUCT
SUMMEEX2MY2	SUMX2MY2
SUMMEEX2PY2	SUMX2PY2
SUMMEEXMY2	SUMXMY2
TREND	TREND
VARIATION	GROWTH

Statistik

Deutsch	Englisch
ACHSENSABSCHNITT	INTERCEPT()
ANZAHL	COUNT
ANZAHL2	COUNT2
B	B
BESTIMMTHEITSMASS	RSQ
BETAINV	BETAINV

BETAVERT	BETADIST
BINOMVERT	BINOMDIST
CHIINV	CHIINV
CHITEST	CHITEST
CHIVERT	CHIDIST
EXPONVERT	EXPONDIST
FINV	FINV
FISHER	FISHER
FISHERINV	FISHERINV
FTEST	FTEST
FVERT	FDIST
GAMMINV	GAMMINV
GAMMALN	GAMMALN
GAMMAVERT	GAMMADIST
GAUSS	GAUSS
GEOMITTEL	GEOMEAN
GESTUTZTMITTEL	TRIMMEAN
GTEST	ZTEST
HARMITTEL	HARMEAN
HYPGEOMVERT	HYPGEOMDIST
KGRÖSSTE	LARGE
KKLEINSTE	SMALL
KONFINDENZ	CONFIDENCE
KORREL	CORREL
KOVAR	COVAR
KRITBINOM	CRITBINOM
KURT	KURT
LOGINV	LOGINV
LOGNORMVERT	LOGNORMDIST
MAX	MAX
MAXA	MAXA
MEDIAN	MEDIAN
MIN	MIN
MINA	MINA
MITTELABW	AVEDEV
MITTELWERT	AVERAGE
MITTELWERTA	AVERAGE2
MODALWERT	MODE
NEGBINOMVERT	NEGBINOMDIST

NORMINV	NORMINV
NORMVERT	NORMDIST
PEARSON	PEARSON
PHI	PHI
POISSON	POISSON
QUANTIL	PERCENTILE
QUANTILSRANG	PERCENTRANK
QUARTILE	QUARTILE
RANG	RANK
SCHÄTZER	FORECAST
SCHIEFE	SKEWNESS
STABW	STDEV
STABWA	STDEVA
STABWN	STDEVP
STABWNA	STDEVPA
STANSARDISIERUNG	STANDARDIZE
STANDNORMINV	NORMSINV
STANFNORMVERT	NORMSDIST
STEIGUNG	SLOPE
STFEHERYX	STEYX
SUMQUADABW	DEVSQ
TTEST	TTEST
TINV	TINV
TVERT	TDIST
VARIANZ	VAR
VARAINZA	VARA
VARAINZEN	VARP
VARAINZENA	VARPA
VARIATIONEN	PERMUT
VARIATIONEN2	PERMUTATIONA
WAHRSCHBEREICH	PROP
WEIBULL	WEIBULL

Tabelle

Deutsch	Englisch
ADRESSE	ADRESS
BEREICHE	AREAS
DDE	DDE
FEHERLTYP	ERRORTYPE
INDEX	INDEX
INDIREKT	INDIRECT
SPALTE	COLUMN
SPALTEN	COLUMNS
SVERWEIS	VLOOKUP
TABELLE	SHEET
TABELLEN	SHEETS
VERGLEICH	OFFSET
VERSCHIEBUNG	MATCH
VERWEIS	LOOKUP
VORLAGE	STYLE
WAHL	CHOOSE
WVERWEIS	HLOOKUP
ZEILE	ROW
ZEILEN	ROWS

Text

Deutsch	Englisch
ARABISCH	ARABIC

BASIS	BASE
CODE	CODE
DEZIMAL	DECIMAL
DM	DOLLAR
ERSETZEN	REPLACE
FEST	FIXED
FINDEN	FIND
GLÄTTEN	TRIM
GROSS	UPPER
GROSS2	PROPER
IDENTISCH	EXACT
KLEIN	LOWER
LÄNGE	LEN
LINKS	LEFT
RECHTS	RIGHT
ROT13	ROT13
RÖMISCH	ROMAN
SÄUBERN	CLEAN
SUCHEN	SEARCH
T	T
TEIL	MID
TEXT	TEXT
VERKETTEN	CONCATENATE
WECHSELN	SUBSTITUDE
WERT	VALUE
WIEDERHOLEN	REPT
ZEICHEN	CHAR

7.6.7 Wie kann man Funktionen von Calc verwenden?

Man kann innerhalb von StarBasic-Makros auch interne Funktionen von Calc aufrufen. Dazu wird der Service com.sun.star.sheet.FunctionAccess verwendet. Er bietet Zugriff auf die internen Funktionen.

Als Erstes muß man den Service starten.

oFunctionAccess = createUnoService("com.sun.star.sheet.FunctionAccess")

Danach muß man sich ein Array mit der Anzahl der möglichen Parameter erzeugen. Dieses Array am Besten vom Typ Variant, da dieses Array unterschiedliche Typen beinhalten kann.

Dim args(0) As Variant

args(0) = "test"

Jetzt kann man auf die Funktion zugreifen (Hier auf die Text-Funktion Länge)

result = oFunctionAccess.callFunction("Len", args())

Wenn mehr Parameter übergeben werden können oder müssen, passt man das Array entsprechend an.

Hier als Beispiel die Funktion KGV:

Dim args(2) As Variant

args(0) = 8

args(1) = 4

args(2) = 12

result = oFunctionAccess.callFunction("LCM", args())

Achtung:

Wie man in den Beispielen sehen kann, muß man für die Funktion die englische Bezeichnung verwenden. Die deutsche Bezeichnung führt zu einem Fehler. Leider kann man nicht sofort auf die englische Bezeichnung schließen. Hier gibt es eine Liste mit den deutschen und englischen Bezeichnungen: Als calc-dokument oder als pdf-Datei. Oder hier in der FAQ unter: Wie kann man Funktionen von Calc verwenden?

7.6.8 Kann man in StarCalc eigene Funktionen verwenden?

Man kann in StarCalc eigene Funktionen verwenden. Diese kann man in Starbasic schreiben und in Calc aufrufen wie alle anderen Funktionen auch.
Folgende Voraussetzungen müssen zutreffen:

Das Makro muss in der Bibliothek Standard von StarOffice oder dem Dokument stehen. StarCalc durchsucht automatisch dieses Bibliotheken mit ihren Modulen nach der Funktion.

Man kann sich auf diese Weise zum Beispiel eine eigene Funktionssammlung in einem Modul in der Bibliothek Standard zusammenstellen.

7.6.9 Wie kann man den aktuellen Tabellennnamen abfragen?

Der aktuelle Tabellennamen steht im Currentcontroller des Dokumentes.

Function TabellenName as String

odoc=thisComponent

Tabellenname=odoc.currentcontroller.activesheet.name

end Function

Um den Dateinamen abzufragen muß man die Eigenschaft Url des Dokumentes verwenden.

ocalc=thiscomponent
msgbox ocalc.url

7.6.10 Wie kann man Tabellen und Zellen schützen?

Um Tabellen und Zellen per Makro zu schützen muss man einige Besonderheiten beachten.
Tabellen kann man mit Passwort schützen. Dies verhindert aber nur den Eingriff in die Tabelle. Der Inhalt der Tabelle kann trotzdem durch ein einfaches Kopieren der Zellen in eine neue Datei ohne Schutz übertragen werden. Um dies für die Formeln zu verhindern sollte man diese ausblenden. (Format-> Zellen ->Zellschutz -> Formeln ausblenden aktivieren.

Zellen sind nur dann geschützt wenn die Tabelle geschützt ist. Möchte man, dass Zellen ungeschützt sind muß, man diese vor dem Schutz entsperren. Auch per Makro ist dies nur möglich wenn die Tabelle ungeschützt ist.

Man darf auf keinen Fall den Tabellen- und Zellenschutz mit einem echten Schutz der Daten und Formeln verwechseln. Es handelt sich vor allem um einen Schutz vor falschen Eingaben in einem Formular.

Wie werden einzelne Tabellen geschützt?

odoc=thiscomponent
oSheets=odoc.sheets
oSheet = oDoc.Sheets(0)

'oder oSheet = oDoc.Sheets().getByName("Tabelle1")
oSheet.protect("Test")

Passwort : Test

Mit oSheet.unprotect("Test") wird der Schutz aufgehoben.

Um einzelne Zellen zu schützen oder den Schutz zu deaktiveren verwendet man das Struct com.sun.star.util.CellProtection

Dim myProtection As New com.sun.star.util.CellProtection
ocell = mysheet.getCellByPosition(0,0)
myProtection.IsLocked=false
oCell.CellProtection=myProtection

Weitere Parameter in dem Struct sind:

isFormulaHidden -> Die Formel wird bei Tabellenschutz ausgeblendet.

isHidden -> Der Inhalt wird bei Tabellenschutz ausgeblendet.

isPrintHidden -> Der Inhalt der Zelle wird bei Tabellenschutz nicht gedruckt.

Um also eine Zelle in einer geschützten Tabelle zu ändern muß man beides durchführen

odoc=thiscomponent
oSheets=odoc.sheets
oSheet = oDoc.Sheets(0)

oSheet.unprotect("Test")

Dim myProtection As New com.sun.star.util.CellProtection
ocell = mysheet.getCellByPosition(0,0)
myProtection.IsLocked=false
oCell.CellProtection=myProtection

oSheet.protect("Test")

Eine kleine kurze Prozedur für den Schutz von Zellen steht in den Tools

7.6.11 Wie man die Tabellenregister ausblenden?

Die erfolgt über den aktuellen Controller und der Properties ShowTabs.
Diese Einstellung gilt dann für alle Arbeitsblätter.

odoc=ThisComponent

odoc.CurrentController.SheetTabs=false

7.6.12 Wie kann man die Gitterlinien ausblenden?

Die erfolgt über den aktuellen Controller und der Properties ShowGrid.
Diese Einstellung gilt dann für alle Arbeitsblätter.

odoc=ThisComponent

odoc.CurrentController.ShowGrid=false

7.6.13 Wie kann man die Bildlaufleisten ausblenden?

Die erfolgt über den aktuellen Controller und der Properties HorizontalScrollBar und VerticalScrollBar.
Diese Einstellung gilt dann für alle Arbeitsblätter.

odoc=ThisComponent

odoc.CurrentController.HorizontalScrollBar=false

odoc.CurrentController.VerticalScrollBar=false

7.6.14 Wie kann man den Text von Kopf- und Fußzeilen formatieren?

Nach dem man die Kopf- oder Fußzeile als Objekt erhalten hat (Siehe auch hier),kann man auch den jeweiligen Text der Felder formatieren.

Dazu muß für den Text einer Kopfzeile den Cursor aufrufen. Dann stehen die bekannten Formatierungsfunktionen zur Verfügung.

```
sub formattextheader

    Doc = thiscomponent
    StyleFamilies = Doc.StyleFamilies
    PageStyles = StyleFamilies.getByName("PageStyles")
    DefPage = PageStyles.getByName("Standard")
    DefPage.HeaderIsOn = True
    DefPage.FooterIsOn = True
    HContent = DefPage.RightPageHeaderContent
    HContent.RightText.String = "rechter text"
    oCursor = Hcontent.RightText.createTextCursor()
    oCursor.setPropertyValue("CharFontName", "Arial")
    oCursor.setPropertyValue("CharHeight", 16)
    HContent.LeftText.String = "linker text"
    HContent.CenterText.String = "mitte"
    oCursor = Hcontent.LeftText.createTextCursor()
    oCursor.setPropertyValue("CharFontName", "Arial")
    oCursor.setPropertyValue("CharHeight", 16)

    DefPage.RightPageHeaderContent = Hcontent

end sub
```

Entsprechend kann man dann mit CenterText verfahren.

Und natürlich geht das Ganze auch bei den Fußzeilen RightPageFooterContent und LeftPageFooterContent.

7.6.15 Wie kann man das Fenster fixieren?

Unter dem Menüpunkt Fenster gibt es den Befehl "Fixieren". Dies kann man auch per Makro ausführen. Dazu verwendet man den CurrentController des Dokuments. Wenn dabei die aktuelle Position des Cursors verwenden will, setzt man die Paramter für Spalte und Zeile auf 0.
Ansonsten gibt man die gewünschte Spalte und Zeile als Parameter mit.

```
Sub jetztFixieren
    MyDoc=ThisComponent
    MyDoc.getCurrentController().FreezeAtPosition(3,5)
end sub
```

8 Text - Writer

8.1 Allgemein

8.2.1 Wie kann man auf Grafiken im Text zugreifen?

Grafiken innerhalb eines Writer-Dokumentes einem eigenen Container den man mit getGraphicObjects erhält.
Danach kann man über getbyindex oder getbyname die einzelne Grafik holen.

```
odoc=thiscomponent
grafiken=odoc.getGraphicObjects
grafik=grafiken.getbyindex(0)
grafik=grafiken.getbyname("Grafik1")
```

8.2.2 Wie kann man Benutzerfelder einfügen?

Hier erst mal einfach der Code um ein Benuzterfeld einzufügen. Erklärung kommt später.

```
Sub InsertUserField
  oDocument = thiscomponent
   oTextFieldMasters = oDocument.TextFieldMasters
   'Prüfen ob das Feld schon existiert
   if oTextFieldMasters.hasByName("com.sun.star.text.FieldMaster.User.XXXX")=false then
   userField=odocument.createInstance("com.sun.star.text.TextField.User")
   newfield=odocument.createInstance("com.sun.star.text.FieldMaster.User")
   newfield.setPropertyValue ("Name", "XXXX")
   newfield.content="Inhalt"
   userfield.attachTextFieldMaster(newfield)
  'und nun einfügen
   range=odocument.text.createtextcursor()
   odocument.text.inserttextcontent(range,userfield,false)
  end if
end sub
```

8.2.3 Wie kann man eine Textmarke einfügen?

Manchmal ist es nötig ein neue Textmarke einzufügen. Dies geht natürlich auch per Makro.
Dazu muß man eine neue Instanz einer Bookmark (Textmarke) erzeugen und diese an den der gewünschten Stelle einfügen.

Die gewünschte Stelle wählt man mit dem Cursor aus.

oDoc = thisComponent

mytextCursor=odoc.text.createtextcursor()

oNeueTextmarke = oDoc.createInstance("com.sun.star.text.Bookmark")

oNeueTextmarke.setName("NeueMarke")

oDoc.text.insertTextContent(mytextCursor, oNeueTextmarke, false)

8.2.4 Wie kann man einen Hyperlink einfügen?

Ein Hyperlink ist in der Formatierung eines Textes verknüpft. Das ist nicht ganz richtig, beschreibt es aber einigermaßen nachvollziehbar.
Wenn man einen Textteil selektiert hat, steht zusätzlich zu den CharacterProperties auch Propertie HyperlinkUrl zur Verfügung. Mit dieser wird der Hyperlink eingtragen.

doc=thiscomponent

oText = Doc.getText()

oCursor = oText.createTextCursor()

oCursor.gotoEnd(true)

oCursor.hyperlinkurl="http://www.test.de"

8.2.5 Wie kann auf Platzhalter zugreifen?

Man kann auf Platzhalter über die Enumeration der Textfelder eines Dokumentes erreichen. Innerhalb der Numeration muss man nur auf den Service prüfen.
Man kann den Cursor auf den Platzhalter setzen oder mit dem Platzhalterobjekt selber weiterarbeiten.

```
Doc = ThisComponent
mytextCursor=Doc.text.createtextcursor()
myViewCursor=Doc.GetCurrentController.ViewCursor
```

```
  TextFelderEnumration = Doc.getTextFields.createEnumeration
  While TextFelderEnumration.hasMoreElements()

     TextField = TextFelderEnumration.nextElement()

     If TextField.supportsService("com.sun.star.text.TextField.JumpEdit") Then
       If "Test" = TextField.PlaceHolder then
          myViewCursor.goToRange(TextField.getAnchor(), FALSE)
       end if
     end if
Wend
```

Um den Typ des Platzhalters auszulesen, verwendet man die Propertie PlaceHolderType.

```
type=TextField.PlaceHolderType
```

Dabei gilt

Integer	Konstante
0	Text
1	Tabelle
2	Textframe
3	Graphic
4	Object

8.2.6 Warum kann man per Makro geänderte Werte von Feldern nicht sehen?

Die Felder müssen nach dem Ändern noch aktualisert werden.

```
odoc=thisComponent
odoc.textfields.refresh()
```

8.2.7 Wie kann ich an eine Textmarke springen und Text eintragen?

Hier gibt es zwei Fälle:
1. Die Textmarke soll innerhalb eines Makros angesprungen werden (mit dem virtuellen Cursor)

2. Die Textmarke soll mit dem Cursor der Anwendung sichtbar für den Anwender zu der Textmarke springen.

Erster Fall:

Indem man sich den Anchor von der Textmarke holt und über die Propertie String den Eintrag vornimmt.

oDoc=thisComponent

oBookmarks = oDoc.Bookmarks

oBookmark=oBookmarks.getByName("Textmarke")

oAnchor= oBookmark.getAnchor

oAnchor.setString("Mein Text")

In einer Kurzfassung geht es auch:

thisComponent.getBookmarks().getByName("test").getAnchor.setString("Text einfügen")

Zweiter Fall:

In diesem Fall muß erst die aktuelle Cursorposition ermittelt werden. Dies geht mit dem Currentcontroller des Objekts.

oDoc=thiscomponent

oViewCursor = oDoc.CurrentController.getviewCursor()

Dann die Textmarke auswählen

oBookmark = oDoc.Bookmarks.getByName("Textmarke")

oBookmarkAnchor = oBookmark.Anchor

Und diese Range dem Viewcursor übergeben.

oViewCursor.gotorange(oBookmarkAnchor,false)

Siehe auch Tools

8.2.8 Wie kann man Serienbriefe per Makro erstellen? Ab 7.0 / 1.1

Serienbriefe per Makro zu erstellen ist eigentlich nicht so schwer. Für diese Aufgabe steht der Service MailMerge zur Verfügung (ab SO 7.0 und OOo 1.1).

Mailmerge muss nur die passenden Parameter bestückt bekommen und das Quelldokument muß mit der entsprechenden Datenbank verknüpft sein.

Folgende Parameter sind von Bedeutung

MailMerge.DataSourceName

Bekommt als Inhalt den Datenbanknamen der bei den Datenquellen hinterlegt ist. Achtung die genaue Schreibweise ist hier gefordert.

Beispiel Mailmerge.DataSourceName="MyDatenbank"

Mailmerge.DocumentUrl

Enthält das BasisDocument für den Serienbrief.

Diese Datei muß im URL-Format übergeben werden.

Beispiel: MailMerge.DocumentURL=ConvertToUrl("c:\myMail.sxw")

MailMerge.CommandType und MailMerge.Command

Diese Beiden stellen die eigentliche Verbindung zur Datenbank her.

Mit CommandType wird bestimmt wie auf die Datenbank zugegriffen wird.

0 = Tabelle

1 = Abfrage

2 = SQL-Abfrage

Mit Command wird dann die Tabelle, Abfrage oder SQL-Abfrage übergeben.

Beispiel

MailMerge.CommandType=1

MailMerge.Command="Abfrage1"

oder

MailMerge.CommandType=0

MailMerge.Command="Tabelle1"

MailMerge.OutputType

Legt das Ausgabeziel fest.

1=Drucker

2=Datei

Beispiel

MailMerge.OutputType=1

MailMerge.OutputUrl

Legt den Speicherpfad für die neuen Dokumente fest

Beispiel

MailMerge.OutputUrl=ConvertToUrl("C:\")

MailMerge.FileNameFromColumn und

MailMerge.Filenameprefix

Mit diesen Beiden kann man festlegen das der Dateiname aus einem Feld der Tabelle für den Dateinamen verwendet wird.

Diese Funktion ist aktiv wenn MailMerge.FileNameFromColumn true ist.

Der Vorgabewert ist false.

MailMerge.Filenameprefix wird das Feld festgelegt in dem der Dateiname steht.

Beispiel

MailMerge.FileNameFromColumn=True

MailMerge.Filenameprefix="Dateiname"

MailMerge.execute(MyProps())

Mit diesem Befehl wird die Funktion ausgeführt.

Beispiel

Dim MyProps as Object

MailMerge.execute(MyProps())

Das Ganze sieht dann komplett so aus:

MailMerge = createunoservice("com.sun.star.text.MailMerge")

MailMerge.DataSourceName="MyDatenbank"

MailMerge.DocumentURL=ConvertToUrl("c:\myMail.sxw")

MailMerge.CommandType=1

MailMerge.Command="Abfrage1"

MailMerge.OutputType=1

MailMerge.OutputUrl=Converttourl("C:\")

MailMerge.FileNameFromColumn=True

MailMerge.Filenameprefix="Dateiname"

Dim MyProps as Object

MailMerge.execute(MyProps())

8.2.9 Wie kann man auf die Benutzerfelder eines Textes zugreifen?

Innerhalb eines Textdokumentes gibt es unter den möglichen Feldbefehlen "Benutzerfelder".
Diese kann man mit Starbasic ändern und auslesen.

Das dazugehörige Objekt befindet sich im Service "TextFieldMasters" und kann mit Getbyname mit dem kompletten Namen angesprochen werden.

```
Dim oDocu as Object
Dim oTextFieldMasters as Object
Dim TxtUserfield as String
oDocu = thisComponent
strUserfield = "Test"
strContent = "Neuer Text"
 On Error Resume Next
oTextFieldMasters = oDocu.TextFieldMasters
```

TxtUserfield="com.sun.star.text.FieldMaster.User."+StrUserfield

oTextFieldMasters.getByName(TxtUserfield).content = StrContent

Na ja und das Auslesen erfolgt eigentlich auf ähnliche Weise:

TxtUserfield="com.sun.star.text.FieldMaster.User."+StrUserfield

XXXXXX=oTextFieldMasters.getByName(TxtUserfield).content

Siehe auch Tools.

8.2.10 Wie kann man auf die Benutzerfelder der Eigenschaften zugreifen?

Unter Dokument Eigenschaften gibt es eine Registerkarte mit vier Benutzer-Infofelder. Für diese kann man auch eigene Namen festlegen.
Sowohl die Werte, wie auch die Namen lassen sich mit Starbasic lesen und setzen.

Sämtliche Informationen der Dokumenteneigenschaften sind im Service "DocumentInfo" im aktuellen Dokument enthalten.

doc = currentcomponent

doc.DocumentInfo.setUserFieldValue(0 ,"text")

Die Felder werden dabei von 0 bis 3 durchgezählt.

Lesen geht dann mit

xxxx= oDocument.DocumentInfo.getUserFieldValue(0)

Siehe auch Tools

8.2.11 Wie kann man auf Variablen zugreifen?

Ebenso wie es Benutzerfelder bei den Feldvariablen gibt, gibt es auch einfache Variablen. Diese unterschieden sich vor allen in einem Punkt von den Benutzerfeldern: Diese Variablen können mehrere Werte aufnehmen. Eigentlich ist es also ein Array von Werten.

Sie liegen auch unterhalb des "TextFieldMasters"-Service. Der Zugriff erfolgt daher etwas anders als bei Benutzerfeldern. Der Zugriff erfolgt wieder direkt über den Namen. Dann erhält man aber den Container der Variablen. Der eigentliche Zugriff erfolgt dann über "DependentTextFields". Diese ergeben dann das Objekt

mit den Variablen. Hier erfolgt nun der Zugriff über den Index (beginnend bei 0).

```
Dim Var as String

Dim oTextfieldMaster As Object
Dim oPropSet as Object
Dim oDependentTextFields as Object
Dim oXDependentTextField as Object
Dim oTextFields as Object
oDocument = thiscomponent
Var ="com.sun.star.text.FieldMaster.SetExpression."+Variable
oTextfieldMasters = oDocument.getTextFieldMasters()
oPropSet = oTextfieldMasters.getByName(Var)
oDependentTextFields = oPropSet.DependentTextFields
oXDependentTextField = oDependentTextFields(0)
oldValue = oXDependentTextField.Content
oXDependentTextField.setPropertyValue("Content", "Neuer Text")
odocument.textfields.refresh()
```

Um den Wert zu lesen geht man den gleichen Weg.

```
Var ="com.sun.star.text.FieldMaster.SetExpression."+Variable
oTextfieldMasters = oDocument.getTextFieldMasters()
oPropSet = oTextfieldMasters.getByName(Var)
oDependentTextFields = oPropSet.DependentTextFields
oXDependentTextField = oDependentTextFields(0)
XXXX= oXDependentTextField.Content
```

Siehe auch Tools

8.2.12 Welche Formen des Cursors gibt es?

Innerhalb von Starbasic gibt es zwei Arten von Cursor: VisibleCursor und TextCursor.
Bei dem VisibleCursor handelt es sich um den Cursor den der Anwender auf dem Bildschirm sieht. Um diesen zu erhalten verwendet man die Methode myDoc.GetCurrentController.ViewCursor. Diesen kann man dann natürlich auch bewegen.

```
Sub MoveViewCursor
   Dim myDoc as Object
   Dim myViewCursor as Object
   myDoc=thisComponent
   myViewCursor=myDoc.GetCurrentController.ViewCursor
   myViewCursor.goRight(6,false)
End sub
```

Der TextCursor ist virtuell und für den Anwender unsichtbar. Zusätzlich gibt es noch den Vorteil das man innerhalb eines Makros mehrere TextCursor verwenden kann. Hier wird oft etwas zwischen VBA und StarBasic verwechselt. Das Range-Objekt von VBA ist eben mit diesem TextCursor-Objekt vergleichbar und nicht mit dem Range-Objekt von StarBasic.

Ein TextCursor wird mit einer Methode erzeugt und kann dann durch Navigation entsprechend positioniert werden.

```
Sub TextCursorTest
Dim myDoc as object
Dim myTextCursor as object
mDoc = thiscomponent
mytextCursor=mydoc.text.createtextcursor()
mytextcursor.goleft(6,false)
End sub
```

Für die Navigation beider Cursor stehen mehrer Navigationsbefehle zur Verfügung. Die wesentlichen Unterschiede zwischen den beiden Cursor sind:

Den VisibleCursor gibt es nur einmal, beim TextCursor kann man mehrere parallel verwenden.

Die Navigationsmöglichkeiten der beiden Cursor sind verschieden.

Navigation	**ViewCursor**	**TextCursor**
goLeft	X	X
goRight	X	X
gotoRange	X	X
goDown	X	-
goUp	X	-
gotoStart	X	X

gotoEnd	X	X
jumpToFirstPage	X	-
jumpToLastPage	X	-
jumpToNextPage	X	-
jumpToPreviousPage	X	-
jumpToStarofPage	X	-
jumpToEndofPage	X	-
gotoNextWord	-	X
gotoPreviousWord	-	X
gotoStarofWord	-	X
gotoEndofWord	-	X
gotoNextParagraph	-	X
gotoPreviousParagraph	-	X
gotoStartofParagraph	-	X
gotoEndofParagraph	-	X

8.2.13 Wie kann man den ViewCursor dem TextCursor übergeben?

Man kann den ViewCursor an den TextCursor über das Range-Objekt übergeben, und umgekehrt.

```
Sub TextCursorToViewCursor
Dim myDoc as object
Dim myTextCursor as object
myDoc = thiscomponent
mytextCursor=mydoc.text.createtextcursor()
myViewCursor=myDoc.GetCurrentController.ViewCursor
myViewCursor.gotoRange(mytextcursor,false)
' oder umgekehrt
mytextcursor.gotoRange(myViewCursor,false)
End sub
```

8.2.14 Wie geht man mit dem Cursor um?

Es gibt zwei Cursor den sichtbaren und den nichtsichtbaren, VisibleCursor und TextCursor. (Siehe Welche Formen des Cursors gibt es?).
Für Beide gilt im wesentlich das Gleiche. Nur der Funktionsumfang der Cursor ist verschieden.

Bei den Methoden der Navigation gibt es den Parameter "Expand". Dieser bezieht sich auf die Markierung. Er kann die Werte True oder False haben. Bei True wird der Teil zwischen der aktuellen Position und der neuen Position markiert.

Mit diesen Zeilen zum Beispiel werden die nächsten 20 Zeichen ab der Cursorposition markiert:

```
myDoc=thisComponent
myViewCursor=myDoc.GetCurrentController. ViewCursor
myViewCursor.goRight(20,true)
```

Der markierte Text ist über die Eigenschaft "String" erreichbar und kann geändert werden.

```
myViewCursor=myDoc.GetCurrentController. ViewCursor
myViewCursor.goRight(20,true)
myViewCursor.String="Hallo Welt"
```

Es ist eventuell nötig zwischen dem TextCursor und dem ViewCursor zu wechseln, um die verschiedenen Navigationsmöglichkeiten zu verwenden.

(Siehe: Wie kann man den ViewCursor dem TextCursor übergeben?)

8.2.15 Wie kann man das Datum aus einem Datumsfeld auslesen?

Blöderweise hat das Datumsfeld keinen eigenen Namen über den man darauf zugreifen kann. Es geht aber über eine Schleife über alle Felder.
Das Datum ist dann in dem Sruct DateTimeValue abgelegt.

Aus diesem Struct muß man sich dann wieder das Datum für die Datenbank zusammen setzen.

```
sub zeigedatum
oDoc = thiscomponent
```

```
oTxtFelder = oDoc.getTextFields().createEnumeration()
do while oTxtFelder.hasmoreElements()
  oFeld = oTxtFelder.nextElement

  if ofeld.IsDate=true then
    msgbox ofeld.DateTimeValue.day
   msgbox ofeld.DateTimeValue.Hours
   msgbox ofeld.DateTimeValue.HundredthSeconds
   msgbox ofeld.DateTimeValue.Minutes
   msgbox ofeld.DateTimeValue.Month
   msgbox ofeld.DateTimeValue.Seconds
    msgbox ofeld.DateTimeValue.Year
mydatumstr=ofeld.DateTimeValue.day+"."+ofeld.DateTimeValue.Month+"."+ofeld.DateTimeValue.Year
end if
loop
end sub
```

mfg

Michael

8.2.16 Wie fügt man ein Inhaltsverzeichnis ein?

Ein Inhaltsverzeichnis wird mit dem Service com.sun.star.text.ContentIndex eingefügt.
Vorher sollte man noch prüfden ob nicht bereits ein Verzeichnis vorhanden ist. Ist eines vorhanden wird dieses nur aktualisert.

```
  oDoc = ThisComponent
  oAllIndex = oDoc.getDocumentIndexes()
  bIndex = False

  For i = 0 To oAllIndex.getCount() - 1
    oIndex = oAllIndex.getByIndex(i)
    If oIndex.supportsService("com.sun.star.text.ContentIndex") Then
      bIndex = True
      Exit For
    End If
  Next

  If Not bIndex Then
    oIndex = oDoc.createInstance("com.sun.star.text.ContentIndex")
    oIndex.CreateFromOutline = True
    oCursor = oDoc.getText().createTextCursor()
```

```
    oCursor.gotoStart(False)
    oDoc.getText().insertTextContent(oCursor, oIndex, False)
  End If

  oIndex.update()
```

8.2.17 Wie kann man Dokumente einfügen?

Innerhalb von Writer kann man ganze Dokumente einfügen. Dabei kann es sich um alle Formate handeln die OpenOffice normal auch öffnen kann. Aber es muß sich um Text-Dateien handeln.

```
Sub InsertText
  Dim Dummy(0) as New com.sun.star.beans.PropertyValue
  odoc=thisComponent
  oText = oDoc.getText()
  oCURsor = oDoc.text.createtextcursor
  sInhalt="D:\Test.odt"
  filetoinsert=converttourl(sinhalt)
  oCURsor.InsertDocumentFromUrl(FileToInsert,Dummy())
  oCursor.gotoend(false)
end sub
```

Als Parameter könen die Parameter des MediaDescriptors übergegeben werden. Die meisten davon machen aber beim Einfügen in einen Text keinen Sinn.

8.2.18 Wie kann man eine Textmarke löschen?

Das geht recht einfach mit der Methode Dispose.

```
oDoc=thisComponent
oBookmarks = oDoc.Bookmarks
oBookmark=oBookmarks.getByName( "Zwei" )
obookmark.dispose()
```

8.2.19 Warum macht der ViewCursor nicht was ich möchte?

Der ViewCursor hat einen kleinen Nachteil. Wenn der Cursor gerade auf einen Objekt ist das kein Text oder eine Tabelle im Text ist,
geht er nicht so wie normalerweise. Das beste Beispiel dafür ist wenn der ViewCursor gerade auf einer Grafik liegt. Dann ist das mit dem ViewCursor übergebene
Objekt dieses Grafikobjekt. Diese Objekt hat nicht die Eigenschaften und Methoden des Textobjektes.
Wenn das nicht der Fall ist muss man sich den Viewcursor selber neusetzen.

```
mytextCursor=ThisComponent.text.createtextcursor()
 ThisComponent.getCurrentController().select(mytextCursor)
 ViewCursor = thiscomponent.CurrentController.getviewCursor()
```

8.2 Text

8.3.1 Wie kann man einen Autotext erstellen?

Um mit Basic einen neuen Text als Autotext hinzuzufügen muß dieser erst markiert und dann in den AutotextContainer eingefügt werden.
Genau wie beim Aufruf von Autotexten muss man daran denken das der eigentliche Name des Autotextes das Kürzel ist.

Es ist sinnvoll vorher zu prüfen ob es den Namen schon gibt.Dies erfolgt mit hasbyname.

```
  MyObjekt = createunoservice("com.sun.star.text.AutoTextContainer")
  odoc=thiscomponent
  otext=odoc.text
  ocursor=otext.createtextcursor()
  ocursor.goright(20,true)
  oBereich=MyObjekt.getByName("standard")
if oBereich.hasbyname("NEU") =false then
  oBereich.insertNewbyname("NEU","Titel",ocursor)
end if
```

8.3.2 Wie kann man Autotext verwenden?

Innerhalb von Basic kann man auch auf vorhandene Autotext verwenden. Dazu muss man aber den Bereich und den Autotextnamen kennen.
Der Zugriff erfolgt über den Service "com.sun.star.text.AutoTextContainer".

Über diesen kann man auf die Bereiche und dann auf die einzelnen Autotexte zugreifen.

Die erste Fall sind die englischen Bezeichnungen. Die Bereiche werden im Dialog als "Standard", "Nur für Vorlagen", "Eigene Bausteine" und "Visitenkarten.." angezeigt. Der Aufruf erfolgt aber über die englischen Bezeichnungen: "standard", "template", "crdbus50" und "mytexts".

```
  oAutotexte = createunoservice("com.sun.star.text.AutoTextContainer")
  oBereich=oAutotexte.getByName("standard")
```

Nun kommt die nächste Falle. Es gibt zwar innerhalb des Service com.sun.star.text.AutoTextGroup die

Methode getbyname, mit dieser kann man aber nicht über den Namen des Textbausteines zugreifen. Intern ist der Name nur der Titel und der eigentliche Name das Kürzel.

Der Zugriff erfolgt daher mit getbyname und dem Kürzel. Hier für einen Text innerhalb des Bereiches Standard: Anfrage.

oAutotext=oBereich.getByName("AF")

Den in diesem Autotext gespeicherten Text kann man jetzt an der Cursorstelle einfügen. Dies erfolgt aber nicht über die Methode "insertTextContent" wie bei den meisten anderen Fällen sondern mit einer eigenen Methode des Services: applyto.

odoc=thiscomponent

otext=odoc.text

ocursor=otext.createtextcursor()

oautotext.applyto(ocursor)

8.3.3 Wie kann man einen Textrahmen einfügen?

Einen Textrahmen fügt man mit insertTextContent ein. Dazu muß erst eine neue Instanz eines Textrahmens erzeugt werden. Diesem müssen dann die nötigen Parameter übergeben werden. Dazu gehören vor allem die Größe (width und height mit dem Struct com.sun.star.awt.Size), Position (HoriOrientPosition und VertiOrientPosition) und der Verankerungstyp (com.sun.star.text.TextContentAnchorType.AT_PAGE, AT_PARAGRAPH ,AS_CHARACTER ,AT_PAGE, AT_FRAME,AT_CHARACTER). Alle weiteren Parameter können natürlich auch gesetzt werden.

```
oCursor = oText.createTextCursor()
oFrame = oDocument.createInstance("com.sun.star.text.TextFrame")
Dim aSize As New com.sun.star.awt.Size
aSize.width = 2000
aSize.height = 600
oFrame.Size = aSize
oFrame.AnchorType = com.sun.star.text.TextContentAnchorType.AT_PAGE
oFrame.sizetype=1
oframe.HoriOrientPosition =1000
oFrame.VertOrientPosition = 2000
oText.insertTextContent(oCursor, oFrame, FALSE)
```

8.3.4 Wie kann man Text einfügen?

Text wird über den Befehl insertString an der Cursor-Position eingefügt. (ViewCursor oder TextCursor). Die

Methode steht im Service Text zur Verfügung.

odoc=thiscomponent

otext=odoc.text

ocursor=otext.createtextcursor()

otext.insertString(ocursor, "Die ist der neue Text",false)

Der letzte Parameter bestimmt ob der Cursor ersetzt wird oder nicht. Hat man mit dem Cursor einen Text ausgewählt wird dieser mit dem Parameter True ersetzt.

8.3.5 Wie kann man Bereiche ein- und ausblenden?

Man kann Texte in Bereiche einteilen. (Einfügen -> Bereich)
Diese Bereiche kann über den service Textsections über den Namen des Bereiches auf rufen.

oDocument = ThisComponent

oDocSections = oDocument.TextSections

mysection=oDocSections.getbyname("Bereich1")

Über die Propertie isVisible kann man den Bereich dann ein oder ausblenden.

mysection.isvisible=true 'false

8.3.6 Wie kann auf Textrahmen und den Text zugreifen?

Textrahmen werden im Document in eigenen Container verwaltet.Über diesen hat man über den Index oder den Namen Zugriff auf die einzelnen Textrahmen.
Den Zugriff auf den Container erhält man mit getTextFrames()

odocument=thiscomponent

oText = oDocument.Text

oFrames=odocument.getTextFrames()

oFrame=oFrames.getByIndex(0)

'oder: oFrame=oFrames.getByName("RahmenName")

Innerhalb des Textrahmens kann man sich dann einen Cursor anlegen und mit diesem den Text bearbeiten.

```
oFrameCursor=oFrame.createTextCursor()
oFrameCursor.String="Hier und jezt"
```

8.3.7 Wie kann man einen kompletten Text markieren?

Mit beiden Cursortypen auf mit den Selben Methoden. Nur bei Viewcursor sollte man sicherheitshalber noch an den Start springen.

Alles markieren mit dem normalen Cursor:

```
oDoc=thisComponent
mytextCursor=odoc.text.createtextcursor()
mytextCursor.gotoEnd(true)
```

Und mit dem ViewCursor

```
oDoc=thisComponent
 myViewCursor=oDoc.GetCurrentController.ViewCursor
 myViewCursor.gotoStart(false)
 myViewCursor.gotoEnd(true)
```

8.3.8 Wie kann man Text "hart" formatieren?

Ein Text "hart" formartieren, bedeutet das man den Text selbst formatiert und dies nicht mit der Hilfe von Formatvorlagen macht. Dadurch werden die Formatangaben direkt mitr dem Textverbunden. Vor allem wenn es um Absätze geht sollte man dies vermeiden und Formatvorlagen verwenden. Diese haben den wesentlichen Vorteil das Änderungen an den Vorlagen automatisch von an allen Textteilen durchgeführt, die diese Vorlage verwenden.
Aber trotzdem gibt es natürlich Fälle wo eine harte Formatierung sinn macht. Vor allem wenn es um einzele Textstellen geht, die man zum Beispiel kursiv, fett oder in einer anderen Schrift darstellen möchte.

Um Text zu formatieren muss das Textobjekt den Service com.sun.star.style.CharacterProperties unterstützen. Dieser beihaltet dann die Einstellungen für die Schrift.

Der Service com.sun.star.style.CharacterProperties wird unter anderem von Cursorn, Absätzen, Satzteilen, Zellen und Zellbereichen unterstützt.

Die wichtigsten Properties hierbei sind:

CharFontName - Der Schriftname

CharHeight - Die Schriftgröße

CharWeight und CharPosture - Der Schriftschnitt (fett, kursiv etc.)

CharColor - Schriftfarbe

CharBacKColor -Hintergrundfarbe

CharUnderline - Unterstreichung

Der folgende Code fomratiert den Text als ComicSansiff, schwarz, Größe 20, einfach unterstrichen, kursiv und fett.

```
myDoc=thisComponent
myViewCursor=myDoc.GetCurrentController.ViewCursor
myViewCursor.CharFontName="ComicSansif"
myViewCursor.CharHeight="20"
myViewCursor.charcolor=&H000000
myViewCursor.CharUnderline=com.sun.star.awt.FontUnderline.SINGLE
myViewCursor.CharWeight=com.sun.star.awt.FontWeight.BOLD
myViewCursor.CharPosture=com.sun.star.awt.FontSlant.ITALIC
```

Um einen Text kursiv und fett darzustellen müssen beide Paramter gesetzt werden: CharWeight und CharPosture.

Achtung nicht jeder Schrifttyp unterstützt alle Fomartierungen!

Die wichtigsten Parameter für CharWeight sind

(weitere stehen in der Referenz)

com.sun.star.awt.FontWeight.NORMAL

com.sun.star.awt.FontWeight.BOLD

Die wichtigsten Parameter für CharPosture sind

(weitere stehen in der Referenz)

com.sun.star.awt.FontSlant.NONE	keine einstellung
com.sun.star.awt.FontSlant.ITALIC	kursiv

com.sun.star.awt.FontSlant.OBLIQUE	Oblique

Die wichtigsten Parameter für CharUnderline sind:

(weitere stehen in der Referenz)

com.sun.star.awt.FontUnderline.None	Keine
com.sun.star.awt.FontUnderline.SINGLE	einfach
com.sun.star.awt.FontUnderline.DOUBLE	doppelt
com.sun.star.awt.FontUnderline.DOTTED	gepunkted
com.sun.star.awt.FontUnderline.DASH	gestrichelt
com.sun.star.awt.FontUnderline.LONGDASH	lang gestrichelt
com.sun.star.awt.FontUnderline.DASHDOT	strich punkt
com.sun.star.awt.FontUnderline.WAVE	Welle
com.sun.star.awt.FontUnderline.DOUBLEWAVE	doppelte Welle
com.sun.star.awt.FontUnderline.BOLD	einfach Fett
com.sun.star.awt.FontUnderline.BOLDDOTTED	gepunkted fett
com.sun.star.awt.FontUnderline.BOLDDASH	gestrichelt fett
com.sun.star.awt.FontUnderline.BOLDLONGDASH	lang gestrichelt
com.sun.star.awt.FontUnderline.BOLDWAVE	Welle fett

8.3.9 Wie kann man markierten Text einschwärzen?

Anscheindend besteht im Pentagon bedarf an so einer Funktion :-)
Dies Makro ersetzt den Text mit x und schwärzt ihn.

```
sub cur_selection_machschwarz
  myDoc=thisComponent
  myViewCursor=myDoc.GetCurrentController.ViewCursor
  mytext=myViewCursor.string
  myLen=len(mytext)
  mynewtext="x"
  for i=1 to mylen
    mynewtext=mynewtext+"x"
  next
```

```
   myViewCursor.string=mynewtext
   myViewCursor.charcolor=&H000000
   myViewCursor.charbackcolor=&H000000
   myViewCursor.goright(mylen+1,false)
end sub
```

8.3.10 Wie kann man eine Textrahmen einfügen?

Ein Textrahmen ist ein eigenes Objekt in einem Text.
Die ersten wichtigsten Parameter zum Einfügen des Rahmen sind dabei: Name, Höhe, Breite, Position (horizontal, vertikal).

Um einen Rahmen einzufügen muss einen neues Rahmenobjekt einzufügen, muss man als erstes einen Cursor aufrufen (ViewCursor oder TextCursor dazu weiter unten noch Details). Danach muss man das neue Objekt des Rahmen (com.sun.star.text.TextFrame) erstellen. An diesem Objekt müssen dann die Parameter eingestellt werden. Danach kann man den neuen Rahmen einfügen.

```
odocument=thiscomponent
oText = oDocument.Text
oCursor = oText.createTextCursor()
'Den neune Textrahmen als Objekt erstellen.
oFrame = oDocument.createInstance("com.sun.star.text.TextFrame")
'Weite und Höhe des Rahmen festlegen
oframe.width = 2000
oframe.height = 2000
'Position des Rahmen festlegen in 100/mm
oframe.HoriOrientPosition =2000
oFrame.VertOrientPosition = 2000
'Name festlegen
oframe.setname("RahmenName")
'Positionsvorgaben ausschalten ( siehe unten)
oFrame.HoriOrient = com.sun.star.text.HoriOrientation.NONE
oFrame.VertOrient = com.sun.star.text.VertOrientation.NONE

'Rahmen einfügen
oText.insertTextContent(oCursor, oFrame, FALSE)
```

Zusätzlich müssen noch weitere Einstellungen vorgenommen werden.

Die Art der Verankerung, wird mit den enum com.sun.star.text.TextConetntAnchorType:

oFrame.AnchorType = com.sun.star.text.TextContentAnchorType.AT_PAGE

Mögliche Parameter:

AT_PARAGRAPH, an dem Absatz
AT_PAGE, an der Seite
AT_CHARACTER, am Zeichen
AS_CHARACTER, als Zeichen

Je nach Typ der Verankerung spielt die Position des Cursors eine Rolle. Die Position des Cursors bestimmt auch die Position des neuen Rahmens. Ein neu erstellter TextCursor befindet sich am Anfang des Dokumentes. Dieser muß also erst an die Position für den Rahmen bewegt werden.

Wichtig sind auch die Parameter für vertikale und horizontale Positionierung. Diese müssen angegeben werden, da die Standardwerte eines neuen Objektes die gewünschten Einstellung für die Positionierung überschreiben.

oFrame.HoriOrient = com.sun.star.text.HoriOrientation.NONE

oFrame.VertOrient = com.sun.star.text.VertOrientation.NONE

Die Parameter HoriOrient und VeriOrient bestimmen die Positionierung des Rahmens ohne Angabe von Positionsdaten. Diese werden überschrieben, wenn die Einstellung nicht None ist.

oFrame.HoriOrient = com.sun.star.text.HoriOrientation.CENTER

führt dazu das die Angabe des horizontale Parameters ignoriert wird.

Die Parameter für den horizontalen und vertikalen Wert unterscheiden sich.

Die wichtigsten mögliche Werte für HoriOrient:

CENTER, mittig

LEFT, linksbündig

RIGHT, rechtsbündig

NONE, keine Angabe.

Die wichtigsten mögliche Werte für VeriOrient:

CENTER, mittig

TOP, oben

BOTTOM, unten

NONE, keine Angabe

Zusätzlich gibt es noch Parameter für die Orientierung an Zeichen.

Außerdem gibt es noch Parameter für den Rahmenliniern, Hintergrund usw. Die Beschreibung folgt noch.

8.3.11 Wie kann man Strings bearbeiten?

Abgesehen von den Standard Basic-Befehlen werden noch einige Funktionen mit geliefert. Diese stehen in der Bibliothek "Tools" Modul "Strings". Näheres dazu ist unter auf dieser Seite unter Makro und Tools -> Welche Funktionen gibt es zu Stringverarbeitung? zu finden.

8.3.12 Wie kann man einen Tabulator einfügen?

Ein Tabulator ist ein Sonderzeichen. Man kann ihn aber über den Unicode und der Funktion CHR() einfügen: Chr(9).

odoc=thiscomponent

otext=odoc.text

ocursor=otext.createtextcursor()

otext.insertString(ocursor, "Die ist der neue Text",false)

otext.insertString(ocursor, chr(9),false)

otext.insertString(ocursor, "Die ist der Text nach dem Tabulator",false)

8.3.13 Wie kann man Tabulatoren formatieren?

Tabulatoren sind Bestandteil eines Absatz (Paragraph). Sie stehen dort in einem Array mit Elementen des Structs com.sun.star.style.TabStop.
Dieses Struct besteht aus vier Parametern:

Position - Die Position des Tabulators von links. Die Position wird 1/100 mm angegeben (1000 = 1 cm)

Alignment - Die Ausrichtung:

com.sun.star.style.TabAlign.LEFT - Links

com.sun.star.style.TabAlign.RIGHT - Rechts

com.sun.star.style.TabAlign.CENTER - Zentiert

com.sun.star.style.TabAlign.DECIMAL - Dezimal

DecimalChar - Trennzeichen, wenn Dezimaleinstellungen

FillChar - Füllzeichen

Soweit die Theorie. Die Praxis sieht anders aus. FillChar und Decimalchar haben anscheinend keine Auskwirkung. Prüfung folgt noch.

```
Doc = thisComponent
Enum = Doc.Text.createEnumeration
' Schleife über alle Absätze
While Enum.hasMoreElements
        TextElement = Enum.nextElement
        if TextElement.supportsService("com.sun.star.text.Paragraph") then
'               Msgbox TextElement.string
               m()=Textelement.ParaTabstops
               for i=0 to ubound(m)
                 m(i).position=9500
                 m(i).Alignment=com.sun.star.style.TabAlign.DECIMAL
                Textelement.ParaTabstops=m()
               next i
          end if
Wend
```

8.3.14 Wie kann man den Umlauf eines Textrahmens festlegen?

Der Umlauf wird mit

```
oFrame = oDocument.createInstance("com.sun.star.text.TextFrame")
oFrame.TextWrap = com.sun.star.text.WrapTextMode.NONE
```

festgelegt.

Dabei gilt:

NONE für kein Umlauf,

THROUGHT für Durchlauf,

PARALLEL für Seitenumlauf,

DYNAMIC für dynamischer Seitenumlauf.

8.3.15 Wie kann man eine Grafik löschen?

Dies geht mit der Methode dispose des Grafikobjektes.

```
odoc=thiscomponent
grafiken=odoc.getGraphicObjects
grafik=grafiken.getbyname("Grafik1")
grafik.dispose()
```

8.3 Absätze

8.4.1 Wie kann ein Absatzende oder einen Zeilenumbruch einfügen?

Der Service Text bietet unter anderen Methoden die Methode insertControlCharacter. Mit dieser Methode lassen sich bestimmte Sonderbefehle eintragen.
Die beiden wichtigsten sind Absatzende und Zeilenumbruch. Außerdem gibt es noch harter, weicher Bindestrich, hartes Leerzeichen und Absatz hinzufügen.

Eingefügt wird er zusammen mit dem Cursor. (Viewcursor oder Textcursor)

```
odoc=thiscomponent

otext=odoc.text

ocursor=otext.createtextcursor()

otext.insertControlCharacter(ocursor, "com.sun.star.ControlCharacter.PARAGRAPH_BREAK",false)
```

Anstatt PARAGRAPH_BREAK müsste man dann die anderen festen Werte einsetzen:

LINE_BREAK, HARD_HYPHEN, SOFT_HYPHEN, HARD_SPACE, APPEND_PARAGRAPH

Müsste? Weil es zwar so logisch erscheint und auch dokumentiert ist, aber nicht stimmt.

Die weiteren Konstanten sind nicht hinterlegt, statt deren muß die dazugehörigen Integerwerte direkt verwenden.

```
otext.insertControlCharacter(ocursor, 1,false)
```

Dabei gilt:

LINE_BREAK=1

HARD_HYPHEN=2

SOFT_HYPHEN=3

HARD_SPACE=4

APPEND_PARAGRAPH=5

8.4.2 Wie kann man Absatzformate zuweisen?

Absatzformate können auf zwei Arten zugewiesen werden: Direkt über den Cursor oder über den Absatz. Absatzformate stehen normalerweise im Stylisten zur Verfügung. Hier kann man auch den Namen sehen der innerhalb des Makros verwendet werden muß.

Erstmal über den Cursor. Jeder Cursor hat die Propertie Parastylename. Dieser beinhaltet den Namen des Absatzformates.

```
mydoc=thisComponent
oText = myDoc.getText()
ocursor = MyDoc.text.createtextcursor
ocursor.ParaStyleName = "Überschrift 3"
```

Mit einem Absatz geht es auf ähnlich. Auch dieser hat die benötigte Propertie Parastylename. Nur das man den Absatz erst finden muß. Eventuell geht es auch über die Absatzvorlage. (Wie man auf Absätze zugreifen kann steht hier.)

```
Doc = thisComponent
Enum = Doc.Text.createEnumeration
' Schleife über alle Absätze
While Enum.hasMoreElements
        TextElement = Enum.nextElement
        if TextElement.supportsService("com.sun.star.text.Paragraph") then
          if  Textelement.parastylename="Standard" then
            Textelement.parastylename="Textkörper"
          end if
        end if
Wend
```

8.4.3 Wie kann man auf Absätze zugreifen?

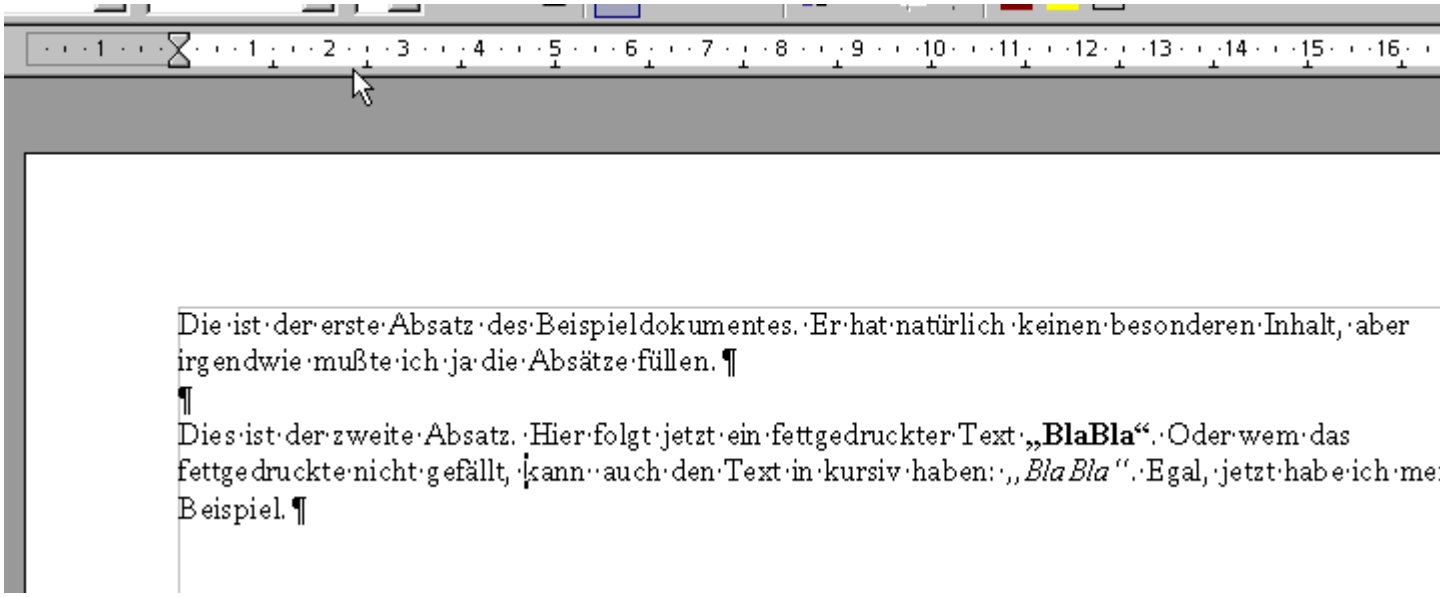
Jedes Textdokument besteht aus Absätzen (engl. Paragraph). Diese Absätze werden im Service "Text" des Dokumentes zusammengefasst. Diese Zusammenfassung schließt auch Tabellen mit ein. StarBasic betrachtet Absätze und Tabellen als ein Objekt unterhalb des Textes. Diese beiden unterscheiden sich in der Unterstützung der entsprechenden Services für Tabellen (com.sun.star.text.Texttable) und Absätze (com.sun.star.text.Paragraph). Es gibt also die Absatztypen "Text" und "Tabelle".

Ein gezielter Zugriff auf einen Textabsatz ist nicht möglich. Absätze werden nicht durchnummeriert, wie zum Beispiel die Arbeitsblätter in Calc, und sie können auch nicht mit einem Namen versehen werden. Tabellen können dagegen einen Namen erhalten und durch diesen kann man auch auf diese zugreifen. Zu dem Tabellen kommt ein eigener Bereich.

Nun zu Absätzen die Text enthalten.

Um auf die einzelnen Absätze zugreifen zu können muß man das Enumeration-Objekt verwenden.

Als Beispiel nehmen wir erstmal ein Dokument mit drei Absätzen:



Wir können die drei Absätze sehen, die durch das Paragraphzeichen getrennt sind.

Tipp: Wenn das Paragraphzeichen nicht sichtbar ist unter Ansicht -> Steuerzeichen anklicken.

Um jetzt auf diese Absätze zu zugreifen benötigen wir eine Enumeration im Service Text, der die Absätze enthält. Zusätzlich müssen wir aber noch prüfen, ob es sich um einen Absatz oder eine Tabelle handelt. Dies erfolgt mit der Runtime Funtion hasunointerfaces. Eine Tabelle unterstützt ein Interface, auf das man prüfen kann (com.sun.star.text.TextTable). Umgekehrt kann man auch das Interface com.sun.star.text.TextRange prüfen. Dieses wird nur durch Absätze unterstützt.

sub absatz1

```
    Doc = thisComponent
    Enum = Doc.Text.createEnumeration
    ' Schleife über alle Absätze
    Doc = ThisComponent
    Enum = Doc.Text.createEnumeration
    ' Schleife über alle Absätze
    While Enum.hasMoreElements
        TextElement = Enum.nextElement
        check=hasunointerfaces(TextElement,"com.sun.star.text.XTextTable")
        if check=false then  Msgbox TextElement.string
    Wend

end sub
```

Als Alternative kann man auch prüfen ob zwei bestimmte Services unterstützt werden: com.sun.star.text.Paragraph für Absätze und com.sun.star.text.TextTable.

```
sub absatz2
    Doc = thisComponent
    Enum = Doc.Text.createEnumeration
    ' Schleife über alle Absätze
    While Enum.hasMoreElements
        TextElement = Enum.nextElement
        if TextElement.supportsService("com.sun.star.text.Paragraph") then
            Msgbox TextElement.string
          end if
    Wend
end sub
```

(Eine Funktion um alle Absätze auszulesen gibt es bei den Tools -> getParagraphs)

Auf beiden Wegen erhalten wir für jeden Absatz eine Messagebox mit dem Text. Beim ersten Absatz sieht das dann so aus:

Das heißt über die Eigenschaft String kann man sich den gesamten Text eines Absatzes auslesen und diesen bearbeiten.

Die Sache hat aber einen Haken. Das Ganze funktioniert mit den ersten Absatz wunderbar, aber beim dritten Absatz haben wir ein Problem. Dieser Absatz enthält unterschiedliche Formatierungen. Und diese unterschiedlichen Formatierungen werden nicht beachtet. Dass heißt wenn der Text in dem String geändert wird, erhält der Text die Formatierung des ersten Teiles.

Wenn wir zum Beispiel die ReplaceString-Funktion aus der mitgelieferten Bibliothek Tools verwenden, um ein oder mehrere Worte auszutauschen erhalten wir eine andere Formatierung.

```
DialogLibraries.LoadLibrary( "Tools" )
While Enum.hasMoreElements
    TextElement = Enum.nextElement
    Textelement.string=ReplaceString(Textelement.string,"BlahBlah","BlaBla")
Wend
```

¶
Dies·ist·der·zweite·Absatz.·Hier·folgt·jetzt·ein·fettgedruckter·Text·„BlaBla".·Oder·wem·das fettgedruckte·nicht·gefällt,·kann··auch·den·Text·in·kursiv·haben:·„BlaBla".·Egal,·jetzt·habe·ich·mein Beispiel.¶

Die beiden Textstellen sind ausgetauscht worden, aber die Formatierung ist leider auch verschwunden.

StarOffice teilt Absätze mit verschiedenen Formatierungen in einzelne Unterteile. Achtung: Nicht die Sätze, sonderen Satzteile mit gleicher Formatierung. Wird der gesamte Text geändert gehen diese einzelnen Teile wieder verloren, da der neue Text wieder ohne unterschiedliche Formatierung als ein Teil übernommen wird.

Um nun auch an diese einzelnen Textteile zu gelangen steht wieder ein Enumeration-Objekt zur Verfügung.

```
enumTextTeile=Textelement.createEnumeration
  while  enumTextTeile.hasMoreElements
   TextTeil=     enumTextTeile.nextelement
   msgbox TextTeil.string
wend
```

Wenn man nun diese einzelnen Teile bearbeitet, bleibt die Formatierung erhalten.

```
enumTextTeile=Textelement.createEnumeration
  while  enumTextTeile.hasMoreElements
```

```
        TextTeil=     enumTextTeile.nextelement
        Textelement.string=ReplaceString(Textelement.string,"BlahBlah","BlaBla")
    wend
```

Beide Services bieten natürlich auch die Möglichkeit Formatierungen vorzunehmen.

8.4.4 Wie kann man mit der Tastatur zwischen den Absätzen wechseln?

Es gibt keinen Tastaturbefehl um zwischen den Absätzen zu springen.

Alternativ kann man diese beiden Makros verwenden und mit einem Tastaturbefehl verknüpfen. Mit diesen Makros springt der Cursor einen Absatz nach oben oder unten und an den Anfang des Absatzes.

```
sub gotoNextParagraph
   Dim myDoc as Object
   Dim myViewCursor as Object
   Dim myTextCursor as object
   myDoc=thisComponent
   myViewCursor=myDoc.GetCurrentController.ViewCursor
   mytextCursor=mydoc.text.createtextcursor()
   mytextCursor.gotoRange(myViewCursor,false)
   mytextCursor.gotoNextParagraph(false)
   mytextCursor.gotoStartOfParagraph(false)
   myViewCursor.gotoRange(myTextCursor,false)
end sub

sub gotoPreviousParagraph
   Dim myDoc as Object
   Dim myViewCursor as Object
   Dim myTextCursor as object
   myDoc=thisComponent
   myViewCursor=myDoc.GetCurrentController.ViewCursor
   mytextCursor=mydoc.text.createtextcursor()
```

```
   mytextCursor.gotoRange(myViewCursor,false)
   mytextCursor.gotoPreviousParagraph(false)
   mytextCursor.gotoStartOfParagraph(false)
   myViewCursor.gotoRange(myTextCursor,false)
end sub
```

8.4 Seiten

8.5.1 Wie kann man das Seitenformat einstellen?

Wenn man die Seitenvorlage aufgerufen hat kann man die Eigenschaften ändern. Dabei sollte man darauf achten das man möglichst die Standardvorlagen nicht ändert. Das ist selbstverständlich möglich, sorgt aber eventuell später für Verwirrung. Da diese Vorlagen in jedem Dokument hinterlegt sind und nicht gelöscht werden können.

Nachdem man sich das Objekt der Seitenvorlage geholt, kann man die Einstellungen vornehmen.

```
Doc = thiscomponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
PageFormat = PageStyles.getByName("MeinStandard")
```

Unter anderen kann man folgende Parameter einstellen:

Den Namen:

```
PageFormat.setname("MeinStandard")
```

Die Hintergrundfarbe

```
Pageformat.backcolor=&H00CC00
```

Den Linken und rechten Seitenrand (in 1/100 mm) :

```
Pageformat.Leftmargin=2000
Pageformat.Rightmargin=2000
```

Oberer und unterer Seitenrand (in 1/100 mm):

```
Pageformat.Topmargin=2000
Pageformat.Bottommargin=2000
```

Kopf- und Fusszeilen aktivieren:

Pageformat.HeaderIsOn=True

Pageformat.FooterIsOn=True

Seitenlayout (für welche Seiten gilt das Layout):

Pageformat.PageStyleFormat=ALL

Mögliche Parameter:

ALL, für beide Seiten
RIGHT, nur für die rechte Seite
LEFT, nur für die linke Seite
MIRRORED, gespiegelt

Hoch- oder Querformat

PageFormat.IsLandscape=true

True für Querformat, False für Hochformat

Nummerierungsformat:

PageFormat.NumberingType=ARABIC

Die Parameter für das Nummerierungsformat finden sich Group der Konstanten zu NumberingType: com.sun.star.style.NumberingType

Seitengröße (in 1/100 mm)

Pageformat.width=14800

Pageformat.height=21000

Weitere Parameter finden sich unter com.sun.star.style.PageStyle in der Referenzdokumentation

8.5.2 Wie kann man die Seitenvorlagen eines Dokumentes ermitteln, löschen oder einfügen?

Um alle Seitenformat eines Dokumentes zu ermitteln muss man über die Elemente der PageStyles gehen. Der Container mit den PageStyles befindet sich im Service com.sun.star.style.StyleFamilies. Dieser Service beinhaltet auch die Formatierungen für Paragraphen, Nummerierung, Zeichen, Rahmen und Seiten.
Um auf die richtige zuzugreifen muss man sich den Container aus den StylesFamilies holen.

Doc = thiscomponent

StyleFamilies = Doc.StyleFamilies

PageStyles = StyleFamilies.getByName("PageStyles")

Innerhalb des PageStyles-Container sind die Seitenformate hinterlegt die in dem Dokument verwendet werden.

Der Aufruf erfolgt über den Namen oder über den Index. getbyName oder getbyIndex.

PageFormat = PageStyles.getByName("Standard")

Oder

PageFormat = PageStyles.getByIndex(0)

An dieser Stelle darf man sich nicht verwirren lassen. Liest man den Namen aus, erscheint bei den Standard-Seitenformaten die englische Bezeichnung, erst ab der Version 2.0 werden die deutschen Bezeichnungen angezeigt. Der Zugriff geht aber auch über die deutsche Bezeichnung. Diese Seitenformate lassen sich auch nicht aus dem Dokument entfernen. removebyname() wird ignoriert.

Englisch	**Deutsch**
Standard	Standard
Left Page	Linke Seite
Right Page	Rechte Seite
First Page	Erste Seite
Envelope	Umschlag
Index	Verzeichnis
Footnote	Fussnote
Endnote	Endnote
HTML	HTML

Um ein Seitenformat zu löschen verwendet man die Methode removebyname().

PageStyles.removeByName("MeineSeite")

Existiert das Format nicht kommt es zu einer Fehlermeldung. Besser man prüft vorher den Namen mit hasbyname(). Bei den Namen kommt es auf die exakte Schreibweise an.

if pagestyles.hasByName("MeineSeite") then pagestyles.removebyname("MeineSeite")

Wie bereits gesagt die Standard-Seitenformate können nicht gelöscht werden und es wird keine Fehlermeldung erzeugt.

Einfügen eines neuen Formates geht über die createinstance-Methode des Dokumentenobjektes und die Einfügung mit insertByName().

newstyle=doc.createinstance("com.sun.star.style.PageStyle")

pagestyles.insertbyname("MeineSeite",newstyle)

8.5.3 Wie kann man auf die Kopf- und Fußzeile zugreifen?

Der Zugriff erfolgt über die Formatierung der Seite. Kopf und Fußzeile sind Bestandteile der Seiteneinstellungen (Format -> Seite)

Also erstmal die Seiteneinstellungen holen:

Doc = thiscomponent

StyleFamilies = Doc.StyleFamilies

PageStyles = StyleFamilies.getByName("PageStyles")

DefPage = PageStyles.getByName("Standard")

Kopf/Fusszeile aktivieren

```
DefPage.HeaderIsOn = True
DefPage.FooterIsOn = True
```

Daten für die Kopfzeile eintragen

```
header=DefPage.headertext
header.setString("Hallo")
```

8.5.4 Wie kann man die aktuelle Seitenvorlage ermitteln?

Das Seitenformat, welches zum Beispiel mit dem Stylisten einer Seiten zuweisen kann, steht in der Gruppe der StyleFamilies innerhalb des Dokumentes.
Um das aktuelle Seitenformat zu ermitteln muss man sich darüber klar sein, das innerhalb eines Dokumentes mehrere Seitenformate möglich sind. Diese werden in einer Gruppe, den "PageStyles", zusammengefasst. Möchte man für die aktuelle Stelle den Namen des Seitenformates erfahren, muss man den Cursor von der aktuelle Stelle verwenden, den ViewCursor. Danach kann man die Seitenvorlage aufrufen.

```
myDoc = thiscomponent
MyCursor=myDoc.GetCurrentController.ViewCursor
mySeitenformat=MyCursor.PageDescName
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
' Die Standardvorlage wird nicht erkannt daher muß dies geprüft werden
if mySeitenformat="" then
PageFormat = PageStyles.getByName("Standard")
else
PageFormat = PageStyles.getByName(mySeitenformat)
end if
```

Siehe auch Wie kann man die Seitenvorlage eines Dokumentes ermitteln, löschen oder einfügen?

8.5.5 Wie kann man Kopf- und Fusszeilen aktivieren?

Kopf- und Fusszeilen sind Bestandteil des Seitenformates.

```
Doc = thiscomponent
```

```
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Standard")
```

Kopf/Fusszeile aktivieren

```
DefPage.HeaderIsOn = True
DefPage.FooterIsOn = True
```

Aktivieren wenn linke und rechte Seite verschieden sein sollen

```
DefPage.HeaderIsShared = False
```

8.5.6 Wie kann man die Seitenzahl ermitteln?

Die aktuelle Seite, auf der sich der Cursor gerade befindet, erhält man mit Hilfe des sichtbaren Cursors (ViewCursors).

```
myDoc=thisComponent
myViewCursor=myDoc.GetCurrentController.ViewCursor
pages=myViewCursor.getPage()
```

Die Seitenanzahl des Dokumentes kann man dem Currentcontroller des Dokumentes auslesen.

```
odoc = thiscomponent
pages = odoc.CurrentController.PageCount
```

8.5.7 Wie kann man die Seitenvorlage ändern?

Die Seitenvorlage kann man mit der Eigenschaft PageDescName des aktuellen Cursors ändern.

```
myDoc = thiscomponent
Dim oCursor As Object
MyCursor = ThisComponent.Text.createTextCursor()
MyCursor.PageDescName = "Vorlage"
```

Siehe auch Makros vom Dannenhöfer

```
Sub SetDocPageStyle(odoc as object,strNew as String)
  Dim oCursor As Object
  MyCursor = ThisComponent.Text.createTextCursor()
  MyCursor.PageDescName = strNew
End Sub
```

8.5 Tabellen

8.6.1 Wie kann man in Tabellen den Rahmen einstellen?

Grundsätzlich kann man Tabellen auf zwei Arten einstellen: Zellenweise oder die Tabelle gesamt. Für alle Linien gilt dabei das Struct "sun.com.star.table.Borderline". Borderline besteht aus folgenden Parametern:

Color -> Farbe der Linie

InnerLineWidth -> Weite der inneren Linie wenn man eine Doppellinie verwendet in 1/100 mm

OuterLineWidth -> Weite der äußeren Linie (einfach oder doppelt) in 1/100 mm

LineDistance > Abstand zwischen den zwei Linien bei einer Doppellinie in 1/100 mm

Um also eine Linie mit einer dicke von 3 mm zu erhalten muss InnerLineWidth und LineDistance auf 0 gesetzt werden und OuterLineWidth auf 300.

Um jetzt bei einer Zelle diesen Rahmen einzustellen muß man die gewünschten Linienwerte nur den einzelnen Linien (Oben,Unten,Links,Rechts) zuweisen.

```
  oDoc = thiscomponent
  Cursor = oDoc.Text.createTextCursor()
  oTables = oDoc.GetTextTables()
  oTable = oTables(0)
  oBorder = oTable.Tableborder
  oBorderline = oBorder.TopLine
```

```
  oBorderline.outerlinewidth = 10
  oBorderline.innerlinewidth = 10
  oBorderline.linedistance = 100
  oBorderline.color = &H00000CCC
  oCell = oTable.getCellByPosition(0,1)
  With oCell
   .topBorder = oBorderline
   .leftBorder = oBorderline
   .rightBorder = oBorderline
   .bottomBorder = oBorderline
 end with
```

Möchte man verschiedene Lienen muß man oBorderline zwischen den einzelnen Zuweisungen anpassen.

Der zweite Weg geht über die gesamte Tabelle. Für die Lieneneinstellung gilt das Selbe wie für Zellen. Nur das es noch zwei weitere Linien gibt die horizontalen und vertikalen. (Horizontalline,Verticalline).

```
  oDoc = thiscomponent
   Cursor = oDoc.Text.createTextCursor()
  oTables = oDoc.GetTextTables()
 oTable = oTables(0)
  oBorder = oTable.Tableborder
 oBorderline = oBorder.TopLine
oBorderline.outerlinewidth = 30
oBorderline.innerlinewidth = 30
oBorderline.linedistance = 100
oBorderline.color = &H00000CCC

oBorder.Topline = oBorderline
oBorder.Bottomline = oBorderline
oBorder.Leftline = oBorderline
oBorder.Rightline = oBorderline
oBorder.Horizontalline = oBorderline
oBorder.Verticalline = oBorderLine
oBorder.Distance = 100
oTable.Tableborder = oBorder
```

8.6.2 Wie kann man Tabellen einfügen?

Wenn man eine Tabelle in einen Text einfügen möchte, mus man vorher einen neues Tabellenobjekt erzeugen und dieses dann an der Cursorposition einfügen.
Nach dem man die neue Tabelle initialisiert hat, muss man noch mit der Methode initialize die Anzahl der Spalten und Zeilen festlegen.

oDoc = thisComponent

oCursor = oDoc.Text.createTextCursor()

newTable = oDoc.createInstance("com.sun.star.text.TextTable")

newTable.initialize(3, 3)

newTable.name="NeueTabelle"

oDoc.Text.insertTextContent(oCursor, newTable, False)

Zusätzlich kann man der Tabelle noch gleich einen Namen mitgeben: newtable.name="NeueTabelle".

Grundsätzlich gilt: Man die Rahmenenstellungen nicht beim Erzeugen vornehmen. Dies muß danach über das neue Tabellenobjekt erflogen. Am Besten einen Namen vergeben und damit auf das neue Tabellenobjekt zugreifen.

myTables = myDoc.getTextTables

myTable=myTables.getByName("NeueTabelle")

Und dann : Wie kann man in Tabellen den Rahmen einstellen?

8.6.3 Wie kann ich auf die Zellen in einer Tabelle zugreifen?

Nachdem man die Tabelle in einem Service erhalten hat:

myDoc = thisComponent

myTables = createunoservice("com.sun.star.text.TextTables")

myTables = myDoc.getTextTables

myTable = createunoservice("com.sun.star.text.TextTable")

myTable=myTables.getByIndex(0)

Kann man auf die einzelnen Zellen oder auf den ganzen Zellbereich zugreifen.

Auf einzelne Zellen kann man mit getCellByPosition zugreifen:

myCell = myTable.getCellByPosition(0,0)

Auf einen Zellbereich greift man mittels zu:

myCellRange = myTable.getCellRangeByPosition(0,0,5,5)

Wenn man die Anzahl der Zeilen und Spalten nicht kennt, kann man mit mytable.rows und .columns die Größe der Tabelle ermitteln

myRows=myTable.rows

myColumns=myTable.columns

AnzahlRows=myRows.Count

AnzahlColumns=myColumns.count

myCellRange=mytable.getCellRangeByPosition(0,0,AnzahlColumns-1,AnzahlRows-1)

8.6.4 Wie kann man einen Textumlauf bei einer Tabelle im Writer erzeugen?

Um Tabellen kann man direkt keinen Textumlauf erzeugen. Man kann aber die Tabelle in einen Rahmen setzen. Der Rahmen gestattet einen Textumlauf.

8.6.5 Wie kann ich auf Tabellen im Text zugreifen?

Tabellen in einem Textdokument sind im Service c.s.s.text.TextTables abgelegt. In diesem sind alle Tabellen eines Textes.
Um diese zu erhalten muß man sich dieses Objekt holen.

myDoc = thisComponent

myTables = myDoc.getTextTables

Je nachdem was man jetzt machen will und weiß kann man jetzt auf die einzelnen Tabellen zugreifen. Die Tabellen werden in TextTables durchgezählt, beginnend mit der 0.

Die einzelne Table wird durch den Service c.s.s.text.TextTable dargestellt.

Um also auf die erste Tabelle zuzugreifen braucht man folgenden Code:

```
myTable=myTables.getByIndex(0)
```

Für andere Tabellen muss die entsprechende Position bekannt sein. Dabei ist es wichtig zu wissen das die Indexierung der Tabellen nicht über die Reihenfolge im Dokument sondern über die Reihenfolge ihrer Erstellung erfolgt.

Um auf alle Tabellen nacheinander zuzugreifen kann man eine Schleife erzeugen die über die Anzahl der Tabellen in myTables geht.

```
AnzahlTables=myTables.count
For i=0 to AnzahlItables
  myTable=myTables.getByIndex(i)
next i
```

Als Alternative kann man auf einzelne Tabellen auch mit den Namen zugreifen. Dieser muß natürlich im Dokument vergeben worden sein (Format -> Tabelle - Eigenschaften -> Name).

```
myTable=myTables.getByName("Tabelle1")
```

8.6.6 Wie kann man eine Zelle teilen?

Dies geht mit dem Cursor der Zelle. Diesen muß man aufrufen und hat dann die Methode zum Teilen.

```
myDoc = thisComponent
myTables = myDoc.getTextTables
myTable = createunoservice("com.sun.star.text.TextTable")
myTable=myTables.getByIndex(0)
myCell = myTable.getCellByPosition(0,0)
oCurs = myTable.createCursorByCellName("A6")
oCurs.splitRange(1, true)
```

Mit dem ersten Parameter wird die Anzahlung der Teilungen festgelegt. Mit dem zweiten (True oder False) ob horizontal oder vertikal geteilt werden soll,

8.6.7 Wie kann man die Schrift einer Zelle senkrecht stellen?

Dies geht mit dem Cursor der Zelle. Diesen muß man aufrufen und hat dann die Eigenschaft CharRotation.
Um die erste Zeile einer Tabelle senkrecht zu stellen geht man folgenderweise vor:

```
myDoc = thisComponent
myTables = myDoc.getTextTables
myTable = createunoservice("com.sun.star.text.TextTable")
myTable=myTables.getByIndex(0)
myCell = myTable.getCellByPosition(0,0)
oCurs = myTable.createCursorByCellName("A1")
```

```
oCurs.CharRotation=900
oCurs = myTable.createCursorByCellName("B1")
oCurs.CharRotation=900
oCurs = myTable.createCursorByCellName("C1")
oCurs.CharRotation=900
```

Mit der Wert 2700 statt 900 dreht sich der Text noch mal um die Achse.

8.6.8 Wie kann man Anzahl der Zeilen einer Tabelle erfahren?

Dies geht mit dem Object der Spalten einer Tabelle und der Funktion getCount.

```
myDoc = thisComponent
myTables = myDoc.getTextTables
myTable=myTables.getByIndex(0)
myrows=myTable.getRows()
anzahl=myrows.getcount()
```

8.6.9 Wie kann man Anzahl der Spalten einer Tabelle erfahren?

Dies geht mit dem Object der Spalten einer Tabelle und der Funktion getCount.

```
myDoc = thisComponent
myTables = myDoc.getTextTables
myTable=myTables.getByIndex(0)
mycols=myTable.getColumns()
anzahl=mycols.getcount()
```

8.6.10 Wie kann man den Cursor in eine Zelle positionieren?

Wenn man den Tabellennamen kennt geht es über die Tabelle und den Cursor (TextCursor oder ViewCursor).

```
odoc=thiscomponent
myTables = oDoc.getTextTables
myTable=myTables.getByName("Tabelle1")
x=myTable.getCellByPosition(0 , 1)
'ViewCursor
myViewCursor=oDoc.GetCurrentController.ViewCursor
myViewCursor.gotoRange(x,false)
'TextCursor
mytextCursor=mydoc.text.createtextcursor()
mytextCursor.gotoRange(x,false)
```

8.6.11 Wie kann man eine Spalte einfügen?

Ein Spalte wird über den Service Columns einfügt. Mit getcolumns erhält man alle Spalten und kann dann mit insertbyindex Spalten einfügen.

```
myDoc = thisComponent
myTables = myDoc.getTextTables
myTable=myTables.getByIndex(0)
columns=myTable.getcolumns()
columns.insertByIndex(0,2)
```

Der erste Parameter gibt die Einfügeposition an, der zweite die Anzahl der Spalten die engefügt werden soll.

8.6.12 Wie kann man eine Zeile einfügen?

Ein Zeile wird über den Service Rows einfügt. Mit getrows erhält man alle Spalten und kann dann mit insertbyindex Spalten einfügen.

```
myDoc = thisComponent
myTables = myDoc.getTextTables
myTable=myTables.getByIndex(0)
rows=myTable.getrows()
rowss.insertByIndex(0,2)
```

Der erste Parameter gibt die Einfügeposition an, der zweite die Anzahl der Zeilen die engefügt werden soll.

9 Datenbanken

9.1 Wie kann man auf Datenbanken zugreifen?

Als Erstes muß man auf den Container zugreifen der alle als Datenquellen hinterlegte Datenbanken enthält. Dies ist der Service "com.sun.star.sdb.DatabaseContext".

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")

Der Zugriff auf eine einzelne Datenbank erfolgt dann über den Namen der Datenbank.

DataSource = DatabaseContext.getByName("myDatenbank")

(Achtung die Datenbank muß so geschrieben werden wie in den Datenquellen hinterlegt).

Eventuell kann man vorher noch überprüfen, ob die Datenbank wirklich vorhanden ist:

if DatabaseContext.hasByName("myDatenbank")=true then .....

Jetzt kann man eine Verbindung (Connection) zu der Datenbank herstellen.

Im einfachsten Fall (Ohne Passwort) mit.

Connection = DataSource.GetConnection("","")

Über den Service "Connection" ist nun eine SQL-Abfrage mit der Übergabe des Ergebnisses in ein ResultSet möglich.

Das ResultSet enthält das Ergebnis der Abfrage zeilenweise.

Für die SQL-Abfrage muß für die Verbindung ein Abfrage-Objekt erzeugt werden.

Dieses ermöglicht erst das Senden der eigentlichen Abfrage.

Statement = Connection.createStatement()

Das Resultset wird dann mit executeQuery(Abfrage) gefüllt.

*ResultSet = Statement.executeQuery("SELECT * FROM Bibliography")*

Die Werte des ResultSet werden dann über die Zeilen und Spalten ausgelesen.

Der zweite Weg

9.2 Wie kann man die Namen aller vorhandenen Datenquellen auslesen?

Man kann sich alle vorhanden Datenbanken über eine Enumaration anzeigen lassen.

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")

Names = DatabaseContext.getElementNames()

For I = 0 To UBound(Names())

MsgBox Names(I)

Next I

Siehe auch bei den Tools: GetListAllDatabases(ListofAllDatabases())

9.3 Wie kann man auf die richtige Spalte des Resultsets zugreifen?

Wenn man den Namen der Spalte kennt kann auf durch die Methode "findColumn" den Index der Spalte ermitteln.

Index=resultset.findcolumn("Spalte")

Bei der Sache sind zwei Dinge zu beachten:

Erstens:

Im Gegensatz zu vielen anderen Bereichen von Starbasic fängt die Zählung hier bei eins an. Erste Spalte hat als den Index 1.

Zweitens:

Kommt der Spaltenname nicht vor, wird kein Fehler erzeugt. Der zurückgegebene Wert entspricht dann der Gesamtanzahl Spalten plus eins.

Man sollte also eventuell eine Prüfung über die Gesamtanzahl durchführen.

Index=resultset.findcolumn("Spalte")

If Index>resultset.columns.count then ... nicht vorhanden.

Nachdem man den Index der Spalte kennt, kann auf diesen mit dem richtigen get-Befehl des Interfaces XRow auf den Inhalt zugreifen.

myWert=resultset.getString(Index)

Siehe auch Wie und auf welche Datentypen kann man zugreifen?

9.4 Wie kann man prüfen ob eine Datenbank auch als Datenquelle zur Verfügung steht?

Mit der Methode "hasbyName".

```
DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
if DatabaseContext.hasByName("test")=true  then
    msgbox "Ja"
     else
   msgbox "Nein"
 end if
```

9.5 Wie kann man die Anzahl der Zeilen des Resultsets ermitteln?

Also ich habe gesucht, aber nichts anderes gefunden als den Weg über resultset.last und getrow

So:

'zum letzen Datensatz springen

resultset.last

anzahl=resultset.getrow

'und nun zurück

resultset.first

9.6 Wie und auf welche Datentypen kann man zugreifen?

Wenn man in der gewünschte Zeile des resultSets ist, kann man mit einer der folgenden Methoden und der Übergabe des Spaltenindexes einen bestimmten Datentyp auslesen.

myWert=resultset.getstring(Index)

getString - Liest den Wert des Type String aus.

getBoolean - Liest den Wert des Type Boolean aus.

getByte - Liest den Wert des Type Byte aus.

getShort - Liest den Wert des Type Short aus.

getInt - Liest den Wert des Type Integer aus.

getLong - Liest den Wert des Type Long aus.

getFloat - Liest den Wert des Type Float aus.

getDouble - Liest den Wert des Type Double aus.

getDate - Liest den Wert des Type Date aus.

getTime - Liest den Wert des Type Time aus.

getTimeStamp - Liest den Wert des Type DateTime aus.

Sonderfälle

(Erklärung folgt noch)

getBytes

getBinaryStream

getCharacterstream

getObject

getRef

getBlob

getClob

getArray

9.7 Wie kann man die Datenbank in Writer austauschen?

Man kann auf Datenfelder über die Enumeration der Textfelder eines Dokumentes erreichen. Innerhalb der Numeration muss man nur auf den Service prüfen.
Bei diesem kann dann mit den Textfieldmasters der Datenbankname geändert werden.

```
Sub databasechange

  Doc = ThisComponent

  TextFelderEnumration = Doc.getTextFields.createEnumeration

    While TextFelderEnumration.hasMoreElements()

      TextField = TextFelderEnumration.nextElement()

      If TextField.supportsService("com.sun.star.text.TextField.Database") Then

          textfield.textfieldmaster.databasename="Neuer Name"

      end if

  Wend

end sub
```

Man darf sich aber nicht verwirren lassen. Nach dem Makro wird beim Überfahren mit der Maus noch die alte Datenbank angezeigt. Geht man aber mit Doppelklick auf das Feld oder noch mal in "Datenbank austauschen" sieht man die Verknüpfung zu der neuen Datenbank. Wenn das Dokument speichert und wieder öffnet wird die neue Datenbank angezeigt.

9.8 Wie kann man eine neue Datenbank erzeugen?

Man kann mit StarBasic auch eine neue Datenbank anlegen. Dies geht aber nicht über ein neues Dokument, wie man nach dem Menü Neu -> Datenbank erwarten könnte, sondern direkt über den Datenbankcontainer (com.sun.star.sdb.DatabaseContext).
Dabei wird eine neue Instanz einer Datenquelle erzeugt und diese mit den nötigen Parameter (Datenbanktyp, Name, Pfad etc.) versehen, danach gespeichert und registriert.

Erstmal holt man sich die vorhandenen Datenquellen:

DatenBanken = createUnoService("com.sun.star.sdb.DatabaseContext")

In diesem Service steht die Methode createinstance zur Verfügung die eine neue Datenquelle erzeugt.

NeueDatenBank= DatenBanken.createinstance()

Der wichtigste Parameter ist "URL" in diesem steht die eigentliche Verknüpfung zu der Datenbank.

Der Syntax für diese Verknüpfung ist sdbc:treiber:datenbank.

Hier Beispiele:

Datenbank	**Verknüpfung**
Adabas	sdbc:adabas::mydb
Dbase	sdbc:dbase:c:\text\daten
Tabelle	sdbc:calc:c:\test.ods
ODBC	sdbc:odbc:Visual FoxPro-Datenbank
Access	sdbc:ado:access:PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA SOURCE=C:\Eigene Dateien\db1.mdb
Outlook	sdbc:address:outlook

(Weitere Beispiele folgen noch)

Hier mit einer Dbase-Datenbank:

NeueDatenBank.url="sdbc:dbase:c:\text\daten"

Weitere wichtige Parameter:

IsPasswordRequired - True, wenn ein Password für die Datenbank benötigt wird.

IsReadOnly - True, wenn die Datenbank nur lesend geöffnet werden kann.

Password - Passwort für die Datenbank.

User - User der Datebanken

Abschließend wird die neue Datenbank gespeichert und im Datenbankcontainer registriert.

NeueDatenBank.DatabaseDocument.storeAsURL("file:///c:/test.odb",getArgs())

DatenBanken.registerObject("NeueDB", NeueDatenBank)

So und jetzt das ganze nochmal hintereinander:

DatenBanken = createUnoService("com.sun.star.sdb.DatabaseContext")

NeueDatenBank= DatenBanken.createinstance()

NeueDatenBank.url="sdbc:dbase:c:\text\daten"

NeueDatenBank.DatabaseDocument.storeAsURL("file:///c:/test.odb",getArgs())

DatenBanken.registerObject("NeueDB", NeueDatenBank)

9.9 Wie kann man eine vorhandene Datenbank-Datei registrieren?

Es ist möglich eine Datenbank-Datei zu erstellen ohne die in OpenOffic anzumelden.
Dies kann mit StarBasic nachgeholt werden. Innerhalb des Datenbankcontainers kann man direkt mit getByname auch Datenbank-Datei laden.

Dieses neue Objekt kann man dann auch registrieren.

DatenBanken = createUnoService("com.sun.star.sdb.DatabaseContext")

NeuDB=DatenBanken.getByname("file:///c:/test5.odb")

DatenBanken.registerobject("NeueDB",NeuDB)

9.10 Wie kann man eine Datenquelle löschen?

Das Löschen einer Datenquelle erfolgt mit revokeObject (aufheben) und nicht wie üblich mit remove.
Sinnvollerweise sollte man vorher prüfen ob die Datenquelle existiert.

```
NameDB="testdb"
oContext=createUnoService("com.sun.star.sdb.DatabaseContext")
if  oContext.hasByName(NameDB) then
   oContext.revokeObject(NameDB)
end if
```

9.11 Wie kann auf die einzelnen Tabellen zugreifen?

Mann kann auf die einzelnen Tabellen einer Datenbank auch zugreifen. Zum Beispiel um die Eigenschaften auszulesen.
Dazu muß man die Verbindung (Connection) herstellen. Damit erhält man den Zugriff.

```
oContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DatenQuellen=oContext.getElementNames()
sNameDB=DatenQuellen(0)
oQuelle=oContext.GetByName(sNameDB)
sLogin=""
sPassword=""
oConnection=oQuelle.getconnection(sLogin,sPassword)

oTables=oConnection.tables
msgbox oTables.count

'Anzeigen der Tabellen
for i = 0 to (oTables.count-1)
    msgbox oTables(i).name
next i
oTable=oTables(1)
```

Dann kann zum Beispiel den Type der Tabelle auslesen.

```
msgbox otTables(1).type
```

9.12 Wie kann auf die einzelnen Abfragen zugreifen?

Mann kann auf die einzelnen Abfragen einer Datenbank auch zugreifen. Zum Beispiel um die Eigenschaften auszulesen. Oder den SQL-Code.
Dazu muß man die Verbindung (Connection) herstellen. Damit erhält man den Zugriff.

```
oContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DatenQuellen=oContext.getElementNames()
sNameDB=DatenQuellen(0)
oQuelle=oContext.GetByName(sNameDB)
sLogin=""
sPassword=""
oConnection=oQuelle.getconnection(sLogin,sPassword)

oAbfragen=oConnection.Queries
msgbox oAbfragen.count

'Anzeigen der Tabellen
for i = 0 to (oAbfragen.count-1)
    msgbox oAbfragen(i).name
next i
oAbfrage=oAbfragen(0)
```

9.13 Wie kann man ein Formular öffnen?

Ein Formular einer Datenbank kann mann innerhalb des geöffneten Datenbank-Dokumentes öffnen. Also ist die Datenbank vor zu öffnen wenn sie nicht aktiv ist.

```
'   dbFrame = StarDeskTop.loadcomponentfromURL( "file:///c:/datenbank.odb","_Default",63, Array())
' oder wenn das Dokument aktiv ist
  dbFrame=StarDesktop.currentComponent
  oDataSource = dbframe.datasource
  oConnection = oDataSource.getConnection("","")
  dim Args(1) as new com.sun.star.beans.PropertyValue
  Args(1).name = "OpenMode"
  Args(1).value = "open"
  Args(0).name = "ActiveConnection"
  Args(0).value = oConnection
```

```
oForms = oDataSource.DataBaseDocument.getFormDocuments
oForms.loadcomponentfromURL( "ADRESSEN", "_Default", 63, Args())
```

9.14 Wie kann man ein Formular aus einem Formular öffnen?

Dies geht in Kombination mit meinen Tool Fensterwaehlen. Mit diesem kann man auf die Datenbankdatei zurückspringen und dann ein anderes Formular öffnen.

```
odoc=fensterwaehlen("Neue Datenbank2.odb")
  formularname = "Tabelle1"
  Dim arg(1) As New com.sun.star.beans.PropertyValue
  x = odoc.getFormDocuments().getElementNames()
  For i = LBOUND(x()) To UBOUND(x())
    if x(i) = formularname Then
      context = CreateUnoService("com.sun.star.sdb.DatabaseContext")
      db = context.getByName(odoc.GetLocation)
      verb = db.getConnection("", "")
      arg(0).Name = "OpenMode"
      arg(0).Value = "open"
      arg(1).Name = "ActiveConnection"
      arg(1).Value = verb
      odoc.getFormDocuments().loadComponentFromURL(x(i), "", 0, arg())
      Exit Sub
    End If
Next i
```

10 Makros und Tools

10.1 Welche Funktionen gibt es zur Stringbearbeitung?

10.2.1 Befehle in Starbasic zur Stringverarbeitung

StarBasic bietet einige Funktionen zum Bearbeiten von Strings. Die Beschreibung dazu steht in der Online-Hilfe Die wichtigsten Funktionen sind:

lcase(str) Gibt den String in kleinen Buchstaben zurück

ucase(str) Gibt den String in großen Buchstaben zurück

len(str) Gibt die Länge eines Strings als Integer zurück. Leerzeichen werden dabei mitgezählt.

ltrim(str) Gibt den String ohne führende Leerzeichen zurück

rtrim(str) Gibt den String ohne nachfolgende Leerzeichen zurück

trim(str) Gibt den String ohne führende und nachfolgende Leerzeichen zurück

left(str,i) Gibt die Anzahl i Zeichen von links des Strings zurück.

right(str,i) Gibt die Anzahl i Zeichen von rechts des Strings zurück.

InStr (i,Str1,Str2,ic) Gibt die Position einer in einem Zeichenfolgenausdruck vorhandenen Zeichenfolge zurück.

mid() bietet zwei Ergebnisse:

mid(str,i,n) Gibt den angegebenen Teil einer Zeichenkette (i bis n) zurück. Wird n nicht angegeben wird der String bis zum Ende zurückgegeben.

mid(str,i,n,Strnew) ersetzt die durch i und n eingrenzten Zeichen durch strnew.

Der Unterschied der beiden Befehle ist, dass im ersten Fall ein Funktion vorliegt und im zweiten Fall eine Anweisung.

Also muß man für das Löschen der Zeichen einen neuen String für das Ergebnis verwenden:
NewStr=Mid(OldStr,1,4)

Bei der Anweisung ändert man den vorhandenen String einfach mit dem Aufruf mid(str,1,4,"Test"). Damit ändert sich der Wert von str.

Möchte man dieses auch beim Löschen erreichen, muß man den neuen Wert der alten Variablen zuweisen.

```
newstr=mid(oldstr,1,4)
oldstr=newstr
```

10.2.2 Allgemein

Die Stringfunktionen sind alle in der Bibliothek "Tools" im Modul "Strings".
Leider sind einige Funktionen fehlerhaft. Die Korrekturen stehen am Ende.

Hier stehen noch nicht alle Funktionen.

Grundsätzlich gilt es vor der Benutzung der Funktionen die Bibliothek einzubinden:

```
GlobalScope.BasicLibraries.LoadLibrary("Tools")
```

10.2.3 Deletestr

Function DeleteStr(ByVal BigString,CompString as String) as String

Löscht einen Teilstring (CompString) aus einem String (BigString)

```
NewStr=DeleteStr("Hallo Welt","lo")
```

ergibt: NewStr="Hal Welt"

10.2.4 ReplaceString

Function ReplaceString(ByVal Bigstring, NewReplace, OldReplace as String) as String

Ersetzt einen Teilstring (OldReplace) durch einen Neuen (NewReplace)

OrgStr="Hallo Welt"

OrgStr=ReplaceString(Orgstr,"Erde","Welt")

ergibt Orgstr="Hallo Erde"

10.2.5 PartStringInArray

Function PartStringInArray(BigArray(), SearchString as String, iCompare as Integer) as Integer

Findet das erste Auftreten eines Teiltrings in einem Array von Strings.

dim stringlist(3) as string

stringlist(0)="Hallo Welt"

stringlist(1)="Morgen"

stringlist(2)="Gestern"

stringlist(3)="Heute"

newi=PartStringInArray(stringlist(),"ster",0)

Ergibt newi=2 also stringlist(2)="Gestern"

Der Parameter ist für die Vergleichseinstellung.

Die Voreinstellung 0 führt zu einem binären Vergleich, der Wert 1 zu einem Textvergleich ohne Berücksichtigung der Groß-/Kleinschreibung.

10.2.6 FindPartString

Function FindPartString(BigString, PreString, PostString as String, SearchPos as Integer) as String

Findet einen Teilstring der zwischen zwei Strings steht

NewStr=FindPartString("Hallo Welt", "al","el",1)

ergibt: NewStr="lo w"

Mit Searchpos kann die Stelle angegeben werden, ab der gesucht werden soll.

10.2.7 DirectoryNameoutofPath

Function DirectoryNameoutofPath(sPath as String, Separator as String) as String

Liest den Pfad aus dem kompletten String mit dem Verzeichnis und dem Dateinamen. Achtung hier ist Angabe des Separators erforderlich.

OrgStr="c:\test\test.txt"

newstr=DirectoryNameoutofPath(Orgstr,"\")

Ergibt: NewStr="c:\test"

10.2.8 FileNameoutofPath

Function FileNameoutofPath(ByVal Path as String, Optional Separator as String) as String

Liest aus einem String den Dateinamen aus

Separator kann verwendet werden wenn dieser nicht "\" ist.

Orgstr="C:\test\test.txt"

FileN=FileNameoutofPath(Orgstr)

ergibt FileN="text.txt"

10.2.9 GetFileNameExtension

Function GetFileNameExtension(ByVal FileName as String)

Gibt die Extension eines Dateinamens als Ergebnis zurück. Dabei ist es egal ob ein kompletter Pfad oder nur der Dateinname übergeben wird.

Orgstr="C:\test\test.txt"

FileExt=FileNameoutofPath(Orgstr)

ergibt FileExt="txt"

10.2.10 GetFileNameWithoutExtension

Function GetFileNameWithoutExtension(ByVal FileName as String, Optional Separator as String)

Liest den Dateinamen ohne Extension aus. Wenn aus dem Dateinamen noch das Verzeichnis mit angegeben ist muß der Separator "\" mit angegeben werden.

Orgstr="test.txt"

FileExt=GetFileNameWithoutExtension(Orgstr)

Ergibt FileExt="test"

Orgstr="C:\test\test.txt"

FileExt=GetFileNameWithoutExtension(Orgstr,"\")

Ergibt FileExt="test"

Ohne Separator:

Orgstr="C:\test\test.txt"

FileExt=GetFileNameWithoutExtension(Orgstr)

Ergibt FileExt="C:\test\test"

10.2.11 BubbleSortList

Function BubbleSortList(ByVal SortList(),optional sort2ndValue as Boolean)

Sortiert ein maximal 2 dimensionales Array. Die Funktion hat einen Fehler. Die Schleife geht bis in das letzte Element und erzeugt dabei automatisch ein neues Element. Um dies zu verhindern muß man die beiden Schleifen innerhalb der Select-Anweisung um eine Bedingung ergänzen.

```
Select Case dimensions
   Case 1
     if t<>(i-1) then
      If SortList(t) > SortList(t+1) Then
         DisplayDummy = SortList(t)
         SortList(t) = SortList(t+1)
         SortList(t+1) = DisplayDummy
      End If
     end if
   Case 2
     If t<>(i-1) then
      If SortList(t,sortvalue) > SortList(t+1,sortvalue) Then
         For k = 0 to UBound(SortList(),2)
               DisplayDummy = SortList(t,k)
               SortList(t,k) = SortList(t+1,k)
               SortList(t+1,k) = DisplayDummy
         Next k
      End If
     End if
   End Select
```

Den neuen Code der Funktion muß man dann in sein eigenes Modul einsetzen, da die mitgelieferten Module schreibgeschützt sind.

Der Fehler ist in neueren Versionen inzwischen behoben.

10.2.12 ElimChar

Function ElimChar(ByVal BigString as String, ElimArray() as String)

Mit dieser kann man ein Array aus Strings aus einem String löschen. Man hat ein Array strs() mit verschieden Einträgen und möchte diese alle aus einem String entfernen.

strs(0)="ab"

strs(1)="01"

strs(2)="wieder"

bigstring="ab Montag den 1.01 fahren die Züge wieder pünktlich"

Newstring=Elimchar(Bigstring,strs())

Ergibt: Newstring="Montag den 1. fahren die Züge pünktlich"

Diese Funktion ist leider fehlerhaft. Der korrigierte Code steht unten. Diesen muß man auch in seinem eigenen Modul einsetzen.

```
Function ElimChar(ByVal BigString as String, ElimArray() as String)
Dim i% ,n%
    For i = 0 to Ubound(ElimArray())-1
        BigString = DeleteStr(BigString,ElimArray(i))
    Next
    ElimChar = BigString
End Function
```

10.2.13 ClearArray

Sub ClearArray(BigArray)

Löscht die Werte in einem Array.

Wichtig ist dabei das die Anzahl der Elemente nicht geändert werden, sondern nur alle Elemente leer gemacht werden.

```
dim stringlist(3) as string
stringlist(0)="Hallo Welt"
stringlist(1)="Morgen"
stringlist(2)="Gestern"
stringlist(3)="Heute"
```

ClearArray(stringlist)

Mit "redim" erreicht man auch das Löschen der Werte.

10.2.14 ClearMultiDimArray

Sub ClearMultiDimArray(BigArray,DimCount as integer)

Mit dieser Routine löscht man die Werte in einem zweidimensionalen Array. Als Übergabewert muss noch die Anzahl der zweiten Dimension bekannt sein.

Im Zweifelsfall -> ubound(myArray(),2)

10.2.15 RTrimStr

Function RTrimStr(ByVal BigString, SmallString as String) as String

Löscht einen SmallString aus BigString wenn der zu löschende String am Ende steht, ansonsten wird der String wieder ganz zurückgegeben.

Bigstring = "Hallo Welt"

NewStr=RTrimStr(BigString,"Welt")

Dann -> NewStr="Hallo "

10.2.16 LTrimChar

Function LTRimChar(ByVal BigString as String,CompChar as String) as String

Löscht den ersten Buchstaben eines Wortes, wenn es der Gesuchte ist. Man darf sich den String bei CompChar nicht verwirren lassen.

Bigstring = "Hallo Welt"

NewStr=LTrimChar(BigString,"H")

Dann NewStr="allo Welt"

10.2.17 ArrayOutOfString

Function ArrayOutOfString(BigString, Separator as String, Optional MaxIndex as integer)

Liest aus einem String, der Felder durch feste Separatoren hat, in ein Array ein.

```
Bigstring = "Hallo, Welt, Sonne, Morgen"
BigArray=ArrayOutOfString(BigString,",")
    For i = Lbound(BigArray()) to Ubound(BigArray())
        msgbox Bigarray(i)
    Next
```

Eine besondere Auswirkung von Maxindex kann ich nicht erkennen.

10.2.18 FieldInArray / FieldInList

Function FieldinArray(LocArray(), MaxIndex as integer, LocField as String) As Boolean
Function FieldinList(LocField, BigList()) As Boolean

Beide prüfen ob ein bestimmter String in einem Array vorkommt.
Die Suche läßt sich bei FieldinArray über MaxIndex nach oben abgrenzen.
Das ist auch der einzige Unterschied zu FieldInList.

```
dim stringlist(3) as string
stringlist(0)="Hallo Welt"
stringlist(1)="Morgen"
stringlist(2)="Gestern"
stringlist(3)="Heute"
fall=FieldinArray(Stringlist(),3,"Gestern")
fall2= FieldInList("Gestern",stringlist())
```

10.2.19 IndexinArray

Function IndexinArray(SearchString as String, LocList()) as Integer

Findet den ersten Eintrag eines Strings in einer Liste von Strings. Dabei wird die Groß- und Kleinschreibung ignoriert.

Damit hat sich meine Funktion GetItemPosFromArray erledigt, da in der lediglich noch die Groß- und Kleinschreibung beachtet wird.

10.2.20 GetIndexInMultiArray

Function GetIndexInMultiArray(SearchList(), SearchValue, SearchIndex as Integer) as Integer

Sucht einen String in einem mehrdimensionalen Array, wobei die Suchdimension mit Searchindex angeben und eingeschränkt wird.

Das ist eine etwas komische Funktion, die man eigentlich mehrfach für alle Dimensionen durchlaufen muß, um dann das letzte Auftreten als Array zu erhalten.

Das geht mit der Funktion GetFirstIndex2DimMultiArray, die ein Array mit dem Wertpaar zurück gibt.

```
function GetFirstIndex2DimMultiArray(Searchlist(),SearchValue)
GlobalScope.BasicLibraries.LoadLibrary("Tools")
Dim Dummy(1)
for i=lbound(Searchlist(),2) to ubound(Searchlist(),2)
   fall=GetIndexInMultiArray(Searchlist(),Searchvalue,i)
   if fall<>-1 then
      Dummy(0)=fall
      Dummy(1)=i
      GetFirstIndex2DimMultiArray()=Dummy()
    exit function
   end if
  next
    Dummy(0)=-1
    Dummy(1)=-1
    GetFirstIndex2DimMultiArray()=Dummy()

end function
```

10.2.21 GetIndexForPartStringinMultiArray

Function GetIndexForPartStringinMultiArray(SearchList(), SearchValue, SearchIndex as Integer) as Integer

Führt ebenso eine Suche über ein zweidimensionales Array durch wie die Funktion GetIndexInMultiArray. Nur ist die Suche hier auf einen Teil des Strings eingeschränkt. Auch hier gilt es wird über Searchindex nur eine Dimension durchsucht. Also, selber über die Anzahl iterieren.

10.2.22 MultiArrayInListbox

Sub MultiArrayInListbox(oDialog as Object, ListboxName as String, ValList(), iDim as Integer)

Füllt eine Listbox mit den Werten aus einem zweidimensionalen Arrays auf. Auch hier wird der zweite Index mit übergeben (idem).

In der Routine ist ein kleiner Fehler.

In der Zeile:

```
If ValList(i) <> "" Then
```

wird immer nur die erste Dimension geprüft. Wenn diese einen Wert hat, wird aber der eigentliche Wert (i,idim-1) übergeben.

Korrekt muß die Zeile also

```
If ValList(i,iDim-1) <> "" Then
```

lauten.

10.2.23 StringInMultiArray

Function StringInMultiArray(SearchList(), SearchString as String, SearchIndex as Integer, ReturnIndex as Integer, Optional MaxIndex as Integer) as String

(Okay, ich geb es zu, bei der Funktion habe ich dreimal lesen müssen bis ich den Sinn verstanden habe.)

Die Funktion gibt den zweiten Wert zu einem gesuchten String aus. Alles Klar?

```
dim stringlist(1,3) as string
stringlist(0,0)="Hallo Welt"
stringlist(0,1)="Morgen"
stringlist(0,2)="Gestern"
```

stringlist(0,3)="Heute"

stringlist(1,0)="1Hallo Welt"

stringlist(1,1)="1Morgen"

stringlist(1,2)="1Gestern"

stringlist(1,3)="1Heute"

wert= StringInMultiArray(stringList(), "Gestern",2,1)

Wert ergibt dann Morgen. Man sucht also in der zweiten Dimension nach Gestern und bekommt dann den dazugehörigen Wert der Dimension ReturnIndex zurück.

In dem Beispiel :

stringlist(0,2)="Gestern" ist dann die Rückgabe stringlist(0,1)="Morgen", da ReturnIndex 1 und der gefundene Index 0 ist.

10.2.24 ArrayfromMultiArray

Function ArrayfromMultiArray(MultiArray as String, iDim as Integer)

Liest aus einem zweidimensionalen Array eine Dimension als neues Array aus.

10.2.25 FindSecondValue

Function FindSecondValue(SearchString as String, TwoDimList() as String) as String

Diese Funktion findet den zu dem Suchwert in der zwei splaltigen Liste eingetragen zweiten Wert.

10.2.26 Power/Round

Function Power(Basis as Double, Exponent as Double) as Double
Function Round(BaseValue as Double, Decimals as Integer) as Double

Zwei Funktionen die nichts mit Strings zu tun haben aber hier zu finden sind.

Round rundet eine Zahl auf die gewünschte Anzahl nach dem Komma ab. Achtung es wird wirklich nur der Rest abgeschnitten, es findet kein auf- oder abrunden statt.

10.2.27 CountCharsinString

Function CountCharsinString(BigString, LocChar as String, ByVal StartPos as Integer) as Integer

Zählt die Anzahl von LocChar in einem String. LocChar kann ein einzelner Buchstabe oder eine Wort sein.,

10.2.28 CheckDouble

Function CheckDouble(DoubleString as String)

Prüft ob ein String sich in eine DoubleVariable wandeln lässt. Wenn nicht ist das Ergebnis 0.

10.2.29 AddListtoList

Function AddListtoList(ByVal FirstArray(), ByVal SecondArray(), Optional StartIndex)

Führt zwei Listen zusammen. Wofür der Parameter StartIndex ist bleibt ein Geheimnis.

10.2 Welche Funktionen gibt es für die Bearbeitung von Listboxen?

10.3.1 Allgemein

Funktionen für die Arbeit mit Listboxen sind in der Bibliothek "Tools" Modul "Listbox".

Grundsätzlich gilt es vor der Benutzung der Funktionen die Bibliothek einzubinden:

```
GlobalScope.BasicLibraries.LoadLibrary("Tools")
```

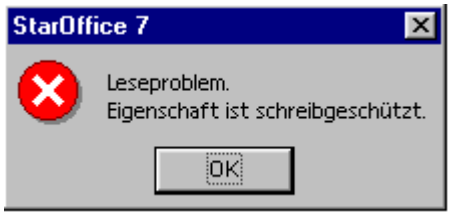
Leider sind einige Funktionen etwas verwirrend deklariert. Es gibt bei einer Listbox zwei Ebenen für Methoden und Properties. Direkt unter Listbox und eine Stufe tiefe unter Listbox.Model. Dies führt bei einigen Funktionen die als Übergabewert anscheinend oListbox erwarten, aber in Wirklichkeit oListbox.Model erhalten müssen.

Beispiel:

```
Sub    SelectListboxItem(oListbox as Object, iSelIndex as Integer)
Dim LocSelList(0) as Integer
    If iSelIndex <> -1 Then
        LocSelList(0) = iSelIndex
        oListbox.SelectedItems() = LocSelList()
    End If
End Sub
```

Wenn man jetzt das Listbox-Objekt übergibt, erhält man eine Fehlermeldung.

```
myControl = myDlg.getControl("ListBox1")
SelectListboxItem(myControl,1)
```



Übergibt man dagegen das Model-Objekt, ist alles in Ordnung.

```
myControl = myDlg.getControl("ListBox1")
SelectListboxItem(myControl.model,1)
```

10.3.2 EmptyListbox

EmptyListbox(oListbox as Object)

Löscht die Einträge einer Listbox.

oListBox = MyDialog.GetControl("ListBox1")

EmptyListbox(oListbox.Model)

10.3.3 GetItemPos

Function GetItemPos(oListBox as Object, sItem as String)

Gibt den ersten Eintrag in der Listbox als Position zurück der sitem entspricht.

Wenn in der Listbox "1","test","drei" steht ergibt.

oListBox1 = MyDialog.GetControl("ListBox1")

pos=GetItemPos(oListBox1.model,"test")

pos=1

10.3 Tools vom Dannenhöfer

Die folgende Funktionen und Prozeduren sind von mir selbst entwickelt oder von mir aus anderen Funktionen abgeleitet. Sie können von jedem weiter verwendet und natürlich auch geändert werden. Bei Veröffentlichungen wäre eine Quellenangabe ganz nett :-).
Manche Funktionen und Prozeduren mögen nicht optimal sein, aber ich bin sehr pragmatisch, wenn so ein Ding das macht was ich will, höre ich meistens auf es weiter zu verbessern.

Es ist durchaus möglich, dass ich bei der einen oder anderen Funktion das Rad neu erfunden habe. Da ich die Originaldokumentation für alles andere als übersichtlich halte, kann mir die eine oder andere Funktion entgangen sein. Ich habe dann trotzdem was gelernt :-)

Natürlich kann ich für die Funktionen keinerlei Haftung übernehmen.

Die Tools gibt es hier zum runterladen:

Als reine Textdatei (bas), zum Einfügen: dtools.bas

Als StarOffice-Dokument: dtools.sxw

10.4.1 GetItemPosFromArray

Function GetItemPosFromArray(sList() as string, sItem as String)

Die Funktion gibt das erste Auftreten eines Strings in einem Array von Strings zurück.

```
Function GetItemPosFromArray(sList() as string, sItem as String) as integer
Dim ItemList()
Dim MaxIndex as Integer
Dim i as Integer
    MaxIndex = Ubound(sList())
    For i = 0 To MaxIndex-1
        If sItem = sList(i) Then
            GetItemPosFromArray() = i
            Exit Function
        End If
    Next i
End Function
```

10.4.2 ListOfAllUrls

ListOfAllUrls(ListOfUrl())
Die Prozedur liest alle geöffneten URL-Pfade aus. Neue Dateien, die noch nicht gespeichert worden sind, werden nicht erkannt. ("UnbenanntX").

Beispiel:

```
Sub BeispielListAll
   dim listall(1)
   listofallurls(listall())
   for i=0 to ubound(listall())-1
     msgbox "URL  "+i+": "+listall(i)
   next
End Sub
```

'Autor: Michael Dannenhöfer

'mail: starbasic@dannenhoefer.de

```
'Version: 19.11.2003
' Die Prozedur liest alle geöffnet URL-Pfade aus.
' Zur Zeit werden neue Dateien "Unbenannt" nicht erkannt
' Der Aufruf erfolgt über
'    dim listall(1)
'    listofallurls(listall())
' Beispiel siehe unten

sub ListOfAllUrls(ListOfUrl())
     'Variablendeklaration
    Dim oComponents as Object
    Dim oDocs as Object
    Dim oDoc as Object
    Dim i as integer
    Dim j as integer
    Dim k as integer
    'Aufruf des Desktops
    oComponents = StarDesktop.getComponents()
    oDocs = oComponents.createEnumeration()
    'Anzahl der offenen Fenster auslesen
    i=0
    Do While oDocs.hasMoreElements()
          oDoc = oDocs.nextElement()
       i=i+1
    Loop
    'Neudimensionierung des arrays
    redim ListOfUrl(i)
    'Audruf der Unterfunktion
    ListOfAllDocs(ListOfUrl())
      'Anzahl der wirkliche Urls auslesen
    j=0
    for k=0 to i
      if ListOfUrl(k)<>"" then j=j+1
    next
    'Neudimensionierung
    Dim TempList(j)
```

```
    l=0
    for k=0 to j
     if ListOfUrl(k)<>"" then
       TempList(l)=ListOfUrl(k)
       l=l+1
     end if
    next
    redim ListOfUrl(j)
    ListOfUrl()=TempList()
end sub


'Autor: Michael Dannenhöfer
'mail: starbasic@dannenhoefer.de
'Version: 19.11.2003
sub ListOfAllDocs(ListAllDocs())
    Dim oComponents as Object
    Dim oDocs as Object
    Dim oDoc as Object
    Dim i as integer
     oComponents = StarDesktop.getComponents()
     oDocs = oComponents.createEnumeration()
    i=0
     Do While oDocs.hasMoreElements()
         oDoc = oDocs.nextElement()
         on error goto nextelement
           ListAllDocs(i)=odoc.url
           i=i+1
         nextelement:
     Loop
end sub
```

10.4.3 GetNameOfAllSheets

Sub GetNameOfAllSheets(NameOfSheets())

Die Prozedur gibt ein Array mit den Namen der Sheets zurück

Der Aufruf erfolgt über:

```
dim listall()
GetNameOfAllSheets(listall())
```

```
Sub GetNameOfAllSheets(NameOfSheets())
  Dim oDoc as Object
  Dim oSheets as Object
  Dim oSheet as Object
  Dim i as integer
  Dim iAnzahl as integer
  oDoc = CurrentComponent
  oSheets=odoc.sheets
  Anzahl = oSheets.count
  Anzahl = Anzahl - 1
  Redim NameOfSheets(Anzahl)
  For i = 0 to Anzahl
    oSheet = oDoc.Sheets(i)
    NameOfSheets(i)=oSheet.name
  Next
End Sub
```

10.4.4 GetPosActiveSheet

Function GetPosActiveSheet(oDoc as object) as Integer

Liest die Position des aktiven Sheets aus. Daran denken das die Zählung bei 0 anfängt.

```
Function GetPosActiveSheet(oDoc as object) as Integer
  Dim MyName as String
  Dim ListOfSheets(1)
  Dim i as Integer
  MyName=odoc.currentcontroller.activesheet.name
```

```
   GetNameOfAllSheets(ListOfSheets(),oDoc)
   For i=0 to ubound(ListOfSheets())
    if ListOfSheets(i)=MyName then GetPosActiveSheet=i
   Next
end Function
```

10.4.5 JumpToSheetsName

Sub JumpToSheet(myDoc as Objekt, sheetsname as String)

Aktiviert das ausgewählte Sheet nach dem Namen.

```
Sub JumpToSheetsName( myDoc as Objekt, sheetsname as String)
    myView = myDoc.CurrentController
    mySheet = myDoc.Sheets.getByName(sheetsname)
    myView.setActiveSheet(mySheet)
End Sub
```

10.4.6 JumpToSheetsNumber

Sub JumpToSheetsNumber(myDoc as Objekt, sheetsnumber as String)

Aktiviert das ausgewählte Sheet nach dem Index

```
Sub JumpToSheetsNumber( myDoc as Objekt, sheetsnumber as String)
    myView = myDoc.CurrentController
    mySheet = myDoc.Sheets(sheetsnumber)
    myView.setActiveSheet(mySheet)
End Sub
```

10.4.7 SetDocPageStyle

Ändert die Seitenvorlage an der aktuellen Cursorposition

```
Sub SetDocPageStyle(odoc as object,strNew as String)
  Dim oCursor As Object
  MyCursor = odoc.GetCurrentController.ViewCursor
  MyCursor.PageDescName = strNew
End Sub
```

10.4.8 MoveCursorToBookmark

Bewegt den sichtbaren Cursor an eine Bookmark

```
Sub MoveCursorToBookmark(oDoc as Objekt, strBookmark as String)
oViewCursor = oDoc.CurrentController.getviewCursor()
oBookmark = oDoc.Bookmarks.getByName(strBookmark)
oBookmarkAnchor = oBookmark.Anchor
oViewCursor.gotorange(oBookmarkAnchor,false)

End Sub
```

10.4.9 AddLibraries

Kopiert die angebene Bilbliothek von der geöffneten Datei in die Standard Bibliothek
Aufruf addLibraries("Bibliothek","Bibliothek")

```
Sub addLibraries(quell_LibName as String, ziel_LibName as String)
    Dim oLibrary As Object
     Dim oGlobalLibrary As Object

'   Erst die mal die Module
     oGlobalLibrary = GlobalScope.BasicLibraries
     oLibrary = BasicLibraries
    addLib(oGlobalLibrary, oLibrary, quell_libname,ziel_libname)
'   Dann die Dialoge
     oLibrary = DialogLibraries
```

```
    oGlobalLibrary = GlobalScope.DialogLibraries
   addLib(oGlobalLibrary, oLibrary, quell_libname,ziel_libname)
End Sub



sub addLib(oGlobalLib as object, oLib as object, quelle_lib as string, ziel_lib as string)
    Dim oquell_Lib As Object
    Dim oZiel_Lib As Object
    Dim Zaehler As Integer
        Dim quellModules() as string

    If oGlobalLib.hasByName(ziel_lib) = False Then
            oGlobalLib.createLibrary( ziel_lib )
      End If
            oquell_Lib = oLib.getByName(quelle_lib)
            quellModules = oquell_Lib.getElementNames()

        If oLib.hasByName(quelle_lib) Then
            oquell_Lib = oLib.getByName(quelle_lib)
            quellModules = oquell_Lib.getElementNames()
            for Zaehler=0 to ubound( quellModules())
                oZiel_Lib = oGlobalLib.getByName(ziel_lib)

                If oZiel_Lib.hasByName( quellModules(zaehler) ) = False Then
                    oZiel_Lib.insertByName( quellModules(zaehler),oquell_Lib.getByName( quellModules(zaehle
r)))
                End If
            Next Zaehler
        End If
end sub
```

10.4.10 GetRow und getColumn

Liest die aktuelle Zeile und Spalte aus.
Achtung der Index beginnt bei 0. A1 = 0,0

```
function getRow as integer
          oDoc=ThisComponent
        If oDoc.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then
      oCelle=oDoc.getCurrentSelection().getCellAddress()
      getRow=oCelle.Row
      end if
end function

function getColumn as integer
      oDoc=ThisComponent
    If oDoc.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then
      oCelle=oDoc.getCurrentSelection().getCellAddress()
      getColumn=oCelle.column
      end if
end function
```

10.4.11 ChangeVariable

Ändert den Wert einer Variablen und aktualisiert die Ansicht

```
Sub ChangeVariable(oDocument as object,Variable,NewContent 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)
  oldValue = oXDependentTextField.Content
  oXDependentTextField.setPropertyValue("Content", Newcontent)
  oTextFields = oDocument.getTextFields()
  oTextFields.refresh()
  On Error Goto 0
End Sub
```

10.4.12 ChangeUserField

Ändert den Wert eines Benutzerfeldes und aktualisert die Ansicht.

```
sub ChangeUserField (oDocument as Object,StrUserField as String, StrContent as String)
     Dim oDocu as Object
    Dim oTextFieldMasters as Object
    Dim TxtUserfield as String
'    oDocu = thisComponent
     On Error Resume Next
    oTextFieldMasters = oDocument.TextFieldMasters
    TxtUserfield="com.sun.star.text.FieldMaster.User."+StrUserfield
    oTextFieldMasters.getByName(TxtUserfield).content = StrContent
    On Error Goto 0
    oTextFields = oDocument.getTextFields()
    oTextFields.refresh()
End Sub
```

10.4.13 ChangeDocUserField

Ändert den Wert des Benutzerfeldes in den Eigenschaften des Dokumentes.
Ist natürlich nur ein Einzeiler, passt als eigene Routine besser zu den Routinen ChangeVariable und ChangeUserField.

x steht für Nummer des Feldes. Die Zählung beginnt bei 0.

```
Sub ChangeDocUserField(odocument as object, x as integer, newt as string)
   oDocument.DocumentInfo.setUserFieldValue(x ,newt)
End Sub
```

10.4.14 GetVariable

Liest den Wert einer Variablen eines Textdokumentes aus.

```
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
```

10.4.15 GetUserField

Liest den Wertes eines Benutzerfeldes aus.

```
Function GetUserField (oDocument as Object,StrUserField as String) as string
     Dim oDocu as Object
    Dim oTextFieldMasters as Object
    Dim TxtUserfield as String
     On Error Resume Next
```

```
    oTextFieldMasters = oDocument.TextFieldMasters
    TxtUserfield="com.sun.star.text.FieldMaster.User."+StrUserfield
    GetUserField=oTextFieldMasters.getByName(TxtUserfield).content
    On Error Goto 0
End Function
```

10.4.16 GetDocUserField

Ist kurz, aber leichter zu merken.
Ist natürlich nur ein Einzeiler, passt als eigene Routine aber besser zu den Routinen GetVariable und GetUserField.

```
Function GetDocUserField(odocument as object,x  as integer) as string
  GetDocUserField= oDocument.DocumentInfo.getUserFieldValue(x)
End Function
```

10.4.17 Zellenschutz

Stellt den Schutzstatus einer Calc-Zelle ein. Achtung die Tabelle selber darf nicht geschützt sein.

```
sub Zellschutz(myCell as Object, gesperrt as boolean, formelhidden as boolean,ausblenden as boolean,druck as boolean)
   Dim myProtection As New com.sun.star.util.CellProtection
   myProtection.IsLocked=gesperrt
   myProtection.IsFormulaHidden=formelhidden
   myProtection.IsHidden=ausblenden
   myProtection.IsPrintHidden=druck
   myCell.CellProtection=myProtection
end sub
```

10.4.18 Tabellenname

Übergibt den aktuellen Tabellennamen in Calc. Kann auch in einer Zelle (=Tabellenname()) verwendet werden.

```
Function TabellenName as String
  odoc=thisComponent
```

```
  Tabellenname=odoc.currentcontroller.activesheet.name
end Function
```

10.4.19 GetCursor

Mit dieser Funktion kann man den sichtbaren Cursor an den leistungsfähigeren nichtsichtbaren Cursor übergeben.

```
Function GetCursor( oDoc As Object ) As Object
' Übernahme des sichtbaren Cursors in eine TextRangeobject

    Dim oRange As Object, oRangeCursor As Object

    oRange = oDoc.CurrentController.Selection.getbyIndex(0)
    oRangeCursor = oDoc.Text.createTextCursorByRange( oRange )
    GetCursor = oRangeCursor

End Function
```

10.4.20 SetViewCursor

Setzt einen nichtsichtbaren Cursor auf den sichtbaren Cursor.

```
Sub SetViewCursor ( oDoc as object, oCursor as object)

  oViewCursor = oDoc.CurrentController.getviewCursor()
  oViewCursor.gotorange(oCursor,false)

end Sub
```

10.4.21 GetListAllDatabases(ListofAllDatabases())

Liest alle vorhandenen Datenbanken in ein array.

```
Function GetListAllDatabases(ListofAllDatabases())
```

```
  Dim AllDatabases As Object
  Dim NameofDB
  Dim i As Integer
  DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
  NameofDB = DatabaseContext.getElementNames()
  ' Zählen der Databases.
  For I = 0 To UBound(NameofDB())
  Next I
  Redim ListofAllDatabases(i-1)
  For i = 0 To UBound(NameofDB())
    ListofAllDatabases(i)=NameofDB(i)
  Next I
end Function
```

10.4.22 SetNeuerAutor

Setzt aus einer Inputbox einen neunen Autor als Ersteller.

```
sub SetNeuerAutor
  oDoc=thiscomponent
  nautor=inputbox("Bitte geben Sie den Autor ein","Neuer Autor")
  oDoc.DocumentInfo.Author=nautor
end sub
```

10.4.23 Fensterwaehlen

function fensterwaehlen(dateiname as string) as object

Mit dieser Funktion kann gezielt ein geöffnetes Fenster als doc-Object aufgerufen werden. Voraussetzung ist, dass der Name bekannt ist.

Ist das Fenster nicht vorhanden wird das Aktuelle übergeben.

```
function fensterwaehlen(dateiname as string) as object
  GlobalScope.BasicLibraries.LoadLibrary("Tools")
```

```
  Dim oDesktop As Object, oDocs As Object
  Dim oDoc As Object, oComponents As Object
  gefunden=false
  oComponents = StarDesktop.getComponents()
  oDocs = oComponents.createEnumeration()
  Do While oDocs.hasMoreElements()
    oDoc = oDocs.nextElement()
    On Error Goto Weiter
    datei=odoc.geturl()
    FileN=FileNameoutofPath(datei)
    if FileN=dateiname then
       fensterwaehlen=odoc
       gefunden=true
      end if
    weiter:
  Loop
  if gefunden=false then fensterwaehlen=stardesktop.currentcomponent
end Function
```

10.4.24 AllSheetsA1

sub AllSheetsA1 (myDoc as Object, row as integer,col as integer)

Diese Routine setzt den aktuellen Cursor von allen Sheets auf eine bestimmte Zelle.

```
sub AllSheetsA1 (myDoc as Object, row as integer,col as integer)
   'Anzahl der Sheets auslesen
   Anzahl=myDoc.Sheets.count
   mySheet = myDoc.Sheets(0)
   'Sichtbaren Cursor holen
   myView = myDoc.CurrentController
   'schleife über die Sheets
   for i=0 to anzahl-1
      mySheet = myDoc.Sheets(i)
      mycell =  mysheet.getCellByPosition(row,col)
      myView.setActiveSheet(mysheet)
```

```
      mydoc.CurrentController.Select(mycell)
   next i
   'erstes Sheet aktiveren
   mySheet = myDoc.Sheets(0)
   myView.setActiveSheet(mysheet)
end sub
```

10.4.25 GetAndSetNumber

Funktion um eine Nummer aus einer Datei zu lesen und die Nummer danach automatisch hochzusetzen. Der Dateiname muß angepaßt werden und der erste Wert muß von Hand erzeugt werden.

```
function GetAndSetNumber as string
   dim f as Integer
   dim rechnungsdatei as string
  dim renummer as string
  rechnungsdatei="c:/re.txt"
   if FileExists("file:///"&rechnungsdatei) then
         f = FreeFile()
         Open "file:///"&rechnungsdatei for Input as #f
              Line Input #f, renummer
              close #f
              f = FreeFile()
              Open "file:///"&rechnungsdatei for output as #f
              Print #f, val(renummer)+1
              close #f
        else
            renummer=0
        endif
    GetAndSetNumber=renummer
end function
```

10.4.26 GotoParagraph

Zwei Routinen die man mit einem Tastaturbefehl verknüpfen kann, um dann zwischen den Absätzen zu springen.

```
sub gotoNextParagraph
   Dim myDoc as Object
   Dim myViewCursor as Object
   Dim myTextCursor as object
   myDoc=thisComponent
   myViewCursor=myDoc.GetCurrentController.ViewCursor
   mytextCursor=mydoc.text.createtextcursor()
   mytextCursor.gotoRange(myViewCursor,false)
   mytextCursor.gotoNextParagraph(false)
   mytextCursor.gotoStartOfParagraph(false)
   myViewCursor.gotoRange(myTextCursor,false)
end sub

sub gotoPreviousParagraph
   Dim myDoc as Object
   Dim myViewCursor as Object
   Dim myTextCursor as object
   myDoc=thisComponent
   myViewCursor=myDoc.GetCurrentController.ViewCursor
   mytextCursor=mydoc.text.createtextcursor()
   mytextCursor.gotoRange(myViewCursor,false)
   mytextCursor.gotoPreviousParagraph(false)
   mytextCursor.gotoStartOfParagraph(false)
   myViewCursor.gotoRange(myTextCursor,false)
end sub
```

10.4.27 GetPropertyValAndInd

Liest zu einer gewünschten Eigenschaft eines Proprtyarrays (com.sun.star.beans.Propertyvalue) den Wert und Index aus. Zusätzlich wir der Name mit zurückgegeben.
Der Rückgabewert ist ein Array mit drei Elementen.

```
function GetPropertyValandInd(aProperty as array,propName as string)
  Dim Ergeb(2) as string
```

```
  for i=0 to ubound(aProperty())
    if aProperty(i).name = propName then
       Ergeb(0)=propName
       Ergeb(1)=aProperty(i).value
       Ergeb(2)=i
     end if
    next
    GetPropertyValueAndIndex()=Ergeb()
end function
```

10.4.28 GetParagraphs

Liest die vorhandenen Absätze aus dem Writerdokument. Mit dem Parameter ignoreEmpty werden leere Absätze ingoriert.

```
function getParagraphs(odoc as object, optional ignoreEmpty as boolean)

     Dim myEnum as object
    Dim isAbsatz as boolean
    Dim myAbsatz as object
    Dim i as Integer
    if IsMissing(ignoreEmpty) then
      ignoreEmpty = false
    end if
    msgbox ignoreEmpty

    'Auslesen der Anzahl
    i=0
     myEnum = oDoc.Text.createEnumeration
    While myEnum.hasMoreElements
          myAbsatz = myEnum.nextElement
            isAbsatz=hasunointerfaces(myAbsatz,"com.sun.star.text.XTextRange")
            if isAbsatz then
               if ignoreEmpty then
                    if myAbsatz.string<>"" then
                       i=i+1
```

```
                end if
              else
              i=i+1
            end if
          end if
    Wend

   Redim allParagraphs(i)

    'Auslesen der Absätze
    i=0
    myEnum = oDoc.Text.createEnumeration
    While myEnum.hasMoreElements
        myAbsatz = myEnum.nextElement
         isAbsatz=hasunointerfaces(myAbsatz,"com.sun.star.text.XTextRange")
         if isAbsatz then
           if ignoreEmpty then
                if myAbsatz.string<>"" then
                  allParagraphs(i)=myAbsatz
                  i=i+1
                end if
              else
              allParagraphs(i)=myAbsatz
              i=i+1
            end if
          end if
       Wend
   getParagraphs()=allParagraphs()
end function
```

10.4.29 HasParagraphFormat

Die Funktion liefert die Anzahl der Textteile eines Absatzes zurück. Jeder Textteil steht für eine eigene Formatierung. Ist der Rückgabewert 1, besteht der Absatz nur aus einer Formatierung.

```
function hasParagraphFormat( oPara as object) as integer
  Dim enumTextTeile as object
```

```
  Dim i as integer
  Dim Dummy as object
  i=0
      enumTextTeile=oPara.createEnumeration
          while  enumTextTeile.hasMoreElements
            Dummy = enumTextTeile.nextElement
             i=i+1
          wend
  hasParagraphFormat=i
end function
```

10.4.30 GetProperty

Die Funktion liest den Wert zu einem durch sName bestimten Property aus. Als Unterstützung zum debuggen kann man sich auch einen Wert übergeben lassen wenn der Eintrag nicht vorhanden ist.

```
Function getProperty( arProps, sName As String, Optional debug as Boolean )
   For i = LBound( arProps ) To uBound( arProps )
      checkProp = arProps(i)
      If checkProp.Name = sName Then
         getProperty() = checkProp.value
         Exit Function
      EndIf
   Next
   getProperty()=""
   if debug then getProperty()="Die Propertie mit dem Namen: "+sName+" ist nicht vorhanden"
End Function
```

10.4.31 GetSelTyp

Liefert für eine aktuell markierten Bereich die Art des Bereiches zurück.

```
Function getSelTyp(oSel as Object) as integer
'  Rückgabe 1 = Zelle
'  Rückgabe 1 = Bereich
'  Rückgabe 1 = Mehrere Bereiche
```

```
    if oSel.supportsService("com.sun.star.sheet.SheetCell") then
      'Der Cursor ist einer Zelle
       getSeltyp=1
      elseif oSel.supportsService("com.sun.star.sheet.SheetCellRange") then
       ' Ein Bereich
       getSeltyp=2
      elseif oSel.supportsService("com.sun.star.sheet.SheetCellRanges") then
       'Mehrere Bereiche
       getSeltyp=3
  end if
end Function
```

10.4.32 GetAlleDrucker

```
'Liefert in einem Array die installierten Drucker unter Windows zurück! Achtung geht nur ab Windows NT!!!

sub GetAlleDrucker(ListOfAllPrinter())
  check=GetGUIType()
  if check=1 then
   shell("regedit /e c:\printer.txt 'HKEY_CURRENT_USER\Software\Microsoft\Windows
NT\CurrentVersion\Devices'",10)
   wait 1000
   zaehler=0
   ende=false
   Dim myPrinter(20)
   #iNumber = Freefile
   aFile = "c:\printer.txt"
   Open aFile For Input As #iNumber
   On Error Goto schluss
   While not eof(#iNumber)
   Line Input #iNumber, sZeile
   start=left(sZeile,1)
   if start="""" then
      ipos=InStr(sZeile,"""=")
      myprinter(zaehler)=mid(szeile,2,ipos-2)
      zaehler=zaehler+1
   end if
```

```
  wend
  Close #iNumber
  schluss:
  redim ListofAllPrinter(zaehler-1)
  for i=0 to zaehler-1
   ListofAllPrinter(i)=myprinter(i)
  next i
 end if
end sub
```

10.4.33 ReadIni

```
'Liest einen Wert aus einer ini-Datei

function readini( inifile as string, bereich as string, param as string, default as string) as string
  Dim inBereich as boolean
  Dim aFile as String
  Dim #inumber
  Dim szeile as String
  Dim para as String
  Dim Start as String

  inBereich=false
  readini=default
  Bereich="["+bereich+"]"
  #iNumber = Freefile
  aFile = inifile
  on error goto ende
  if FileExists(inifile) then
  Open aFile For Input As #iNumber
  While not eof(#iNumber)

  Line Input #iNumber, sZeile
  if szeile=Bereich then inBereich=true
  if inBereich then
    ipos=InStr(sZeile,"=")
```

```
      if ipos>0 then
        para=mid(szeile,1,ipos-1)
        if para = param then
          readini=mid(szeile,ipos+1)
          inBereich=false
        end if
      end if
    end if
    if inBereich then
     start=left(sZeile,1)
     if start="[" then inbereich=false
    end if
    if szeile=bereich then inBereich=true
    wend
    Close #iNumber
   end if
  exit function
  ende:
end function
```

10.4.34 WriteIni

Schreibt einen neuen Wert zu einem Parameter.

```
sub writeini(inifile as string, bereich as string, param as string, wert as string)
   GlobalScope.BasicLibraries.LoadLibrary("Tools")
   Dim inBereich as boolean
   Dim aFile as String
   Dim aFileTemp as String
   Dim #inumber
   Dim #inumber2
   Dim szeile as String
   Dim para as String
   Dim Start as String
   Dim Pfad as String
   Pfad=DirectoryNameoutofPath(inifile,"\")
```

```
  afiletemp=Pfad+"\temp.ini"
  inBereich=false
  Bereich="["+bereich+"]"
  aFile = inifile
  if FileExists(inifile) then
  #iNumber = Freefile
  Open aFile For Input As #iNumber
  #iNumber2 = Freefile
  Open aFiletemp For Output As #iNumber2
  While not eof(#iNumber)
  Line Input #iNumber, sZeile
  if inBereich then
    ipos=InStr(sZeile,"=")
     if ipos>0 then
       para=mid(szeile,1,ipos-1)
       if para = param then
         szeile=para+"="+wert
       end if
     end if
   end if
   if inBereich then
    start=left(sZeile,1)
    if start="[" then
      inbereich=false
    end if
   end if
   if szeile=bereich then inBereich=true
   Print #inumber2, szeile
   wend
   Close #iNumber
   Close #iNumber2
   kill afile
   Filecopy afiletemp, afile
   kill afiletemp
  end if
end sub
```

10.4.35 GetDirs

Liest die Dateien eines Ordners und der Unterordner ein.
Beim Aufruf muß z als 0 übergeben werden.

```
function getdirs( liste(),z, folder) as integer
  sFolderUrl = ConvertToUrl( Folder )
  oSimpleFileAccess = createUnoService( "com.sun.star.ucb.SimpleFileAccess" )
  aFolders = oSimpleFileAccess.getFolderContents( sFolderUrl,true )
  For i = LBound( aFolders ) To UBound( aFolders )
    sFile = aFolders( i )
    If oSimpleFileAccess.isFolder( sFile ) Then
      getdirs( liste(),z, sFile)
     Else
      liste(z)=sfile
      z=z+1
    end if
  next i
  getdirs=z
end function
```

11 Intern

11.1 Letzte Einträge

Letzte Updates

06.04.2009

Wie kann man eine Spalte einfügen? (Writer)
Wie kann man eine Zeile einfügen? (Writer)

11.2 Impressum / Lizenz

Kritik, Korrekturen und sonstiges an starbasic@dannenhoefer.de

Erstellt von Michael Dannenhöfer 2003-2008.

Keine Haftung für die Richtigkeit der Informationen.

Impressum

Michael Dannenhöfer

Waldstraße 98

63263 Neu-Isenburg

Telefon 06102/202626

Telefax 06102/202648

starbasic@dannenhoefer.de

Rechte/Lizenz

Der ausführliche Lizenztext steht hier: CC-Lizenz Info zu CC-Lizenz finden sie im Internet: http://creativecommons.org/

11.3 CC Lizenz

creative commons
LEGAL CODE

Namensnennung – Keine Bearbeitung 2.0

CREATIVE COMMONS IST KEINE RECHTSANWALTSGESELLSCHAFT UND LEISTET KEINE RECHTSBERATUNG. DIE WEITERGABE DIESES LIZENZENTWURFES FÜHRT ZU KEINEM MANDATSVERHÄLTNIS. CREATIVE COMMONS ERBRINGT DIESE INFORMATIONEN OHNE GEWÄHR. CREATIVE COMMONS ÜBERNIMMT KEINE GEWÄHRLEISTUNG FÜR DIE GELIEFERTEN INFORMATIONEN UND SCHLIEßT DIE HAFTUNG FÜR SCHÄDEN AUS, DIE SICH AUS IHREM GEBRAUCH ERGEBEN.

Lizenzvertrag

DAS URHEBERRECHTLICH GESCHÜTZTE WERK ODER DER SONSTIGE SCHUTZGEGENSTAND (WIE UNTEN BESCHRIEBEN) WIRD UNTER DEN BEDINGUNGEN DIESER CREATIVE COMMONS PUBLIC LICENSE („CCPL“ ODER „LIZENZVERTRAG“) ZUR VERFÜGUNG GESTELLT. DER SCHUTZGEGENSTAND IST DURCH DAS URHEBERRECHT UND/ODER EINSCHLÄGIGE GESETZE GESCHÜTZT.

DURCH DIE AUSÜBUNG EINES DURCH DIESEN LIZENZVERTRAG GEWÄHRTEN RECHTS AN DEM SCHUTZGEGENSTAND ERKLÄREN SIE SICH MIT DEN LIZENZBEDINGUNGEN RECHTSVERBINDLICH EINVERSTANDEN. DER LIZENZGEBER RÄUMT IHNEN DIE HIER BESCHRIEBENEN RECHTE UNTER DER VORAUSSETZUNGEIN, DASS SIE SICH MIT DIESEN VERTRAGSBEDINGUNGEN EINVERSTANDEN ERKLÄREN.

1. Definitionen

- Unter einer „Bearbeitung“ wird eine Übersetzung oder andere Bearbeitung des Werkes verstanden, die Ihre persönliche geistige Schöpfung ist. Eine freie Benutzung des Werkes wird nicht als Bearbeitung angesehen.
- Unter den „Lizenzelementen“ werden die folgenden Lizenzcharakteristika verstanden, die vom Lizenzgeber ausgewählt und in der Bezeichnung der Lizenz genannt werden: „Namensnennung“, „Nicht-kommerziell“, „Weitergabe unter gleichen Bedingungen“.
- Unter dem „Lizenzgeber“ wird die natürliche oder juristische Person verstanden, die den Schutzgegenstand unter den Bedingungen dieser Lizenz anbietet.
- Unter einem „Sammelwerk“ wird eine Sammlung von Werken, Daten oder anderen unabhängigen Elementen verstanden, die aufgrund der Auswahl oder Anordnung der Elemente eine persönliche geistige Schöpfung ist. Darunter fallen auch solche Sammelwerke, deren Elemente systematisch oder methodisch angeordnet und einzeln mit Hilfe elektronischer Mittel oder auf andere Weise zugänglich sind (Datenbankwerke). Ein Sammelwerk wird im Zusammenhang mit dieser Lizenz nicht als Bearbeitung (wie oben beschrieben) angesehen.
- Mit „SIE“ und „Ihnen“ ist die natürliche oder juristische Person gemeint, die die durch diese Lizenz gewährten Nutzungsrechte ausübt und die zuvor die Bedingungen dieser Lizenz im Hinblick auf das Werk nicht verletzt hat, oder die die ausdrückliche Erlaubnis des Lizenzgebers erhalten hat, die durch diese Lizenz gewährten Nutzungsrechte trotz einer vorherigen Verletzung auszuüben.
- Unter dem „Schutzgegenstand“wird das Werk oder Sammelwerk oder das Schutzobjekt eines verwandten Schutzrechts, das Ihnen unter den Bedingungen dieser Lizenz angeboten wird, verstanden
- Unter dem „Urheber“ wird die natürliche Person verstanden, die das Werk geschaffen hat.
- Unter einem „verwandten Schutzrecht“ wird das Recht an einem anderen urheberrechtlichen Schutzgegenstand als einem Werk verstanden, zum Beispiel einer wissenschaftlichen Ausgabe, einem nachgelassenen Werk, einem Lichtbild, einer Datenbank, einem Tonträger, einer Funksendung, einem Laufbild oder einer Darbietung eines ausübenden Künstlers.
- Unter dem „Werk“ wird eine persönliche geistige Schöpfung verstanden, die Ihnen unter den Bedingungen dieser Lizenz angeboten wird.

2. Schranken des Urheberrechts. Diese Lizenz lässt sämtliche Befugnisse unberührt, die sich aus den Schranken des Urheberrechts,aus dem Erschöpfungsgrundsatz oder anderen Beschränkungen der Ausschließlichkeitsrechte des Rechtsinhabers ergeben.

3. Lizenzierung. Unter den Bedingungen dieses Lizenzvertrages räumt Ihnen der Lizenzgeber ein lizenzgebührenfreies, räumlich und zeitlich (für die Dauer des Urheberrechts oder verwandten Schutzrechts) unbeschränktes einfaches Nutzungsrecht ein, den Schutzgegenstand in der folgenden Art und Weise zu nutzen:

- den Schutzgegenstand in körperlicher Form zu verwerten, insbesondere zu vervielfältigen, zu verbreiten und auszustellen;
- den Schutzgegenstand in unkörperlicher Form öffentlich wiederzugeben, insbesondere vorzutragen, aufzuführen und vorzuführen, öffentlich zugänglich zu machen, zu senden, durch

Bild- und Tonträger wiederzugeben sowie Funksendungen und öffentliche Zugänglichmachungen wiederzugeben;

- den Schutzgegenstand auf Bild- oder Tonträger aufzunehmen, Lichtbilder davon herzustellen, weiterzusenden und in dem in a. und b. genannten Umfang zu verwerten;

Die genannten Nutzungsrechte können für alle bekannten Nutzungsarten ausgeübt werden. Die genannten Nutzungsrechte beinhalten das Recht, solche Veränderungen an dem Werk vorzunehmen, die technisch erforderlich sind, um die Nutzungsrechte für alle Nutzungsarten wahrzunehmen. Insbesondere sind davon die Anpassung an andere Medien und auf andere Dateiformate umfasst.

4. Beschränkungen. Die Einräumung der Nutzungsrechte gemäß Ziffer 3 erfolgt ausdrücklich nur unter den folgenden Bedingungen:

- Sie dürfen den Schutzgegenstand ausschließlich unter den Bedingungen dieser Lizenz vervielfältigen, verbreiten oder öffentlich wiedergeben, und Sie müssen stets eine Kopie oder die vollständige Internetadresse in Form des Uniform-Resource-Identifier (URI) dieser Lizenz beifügen, wenn Sie den Schutzgegenstandvervielfältigen, verbreiten oder öffentlich wiedergeben. Sie dürfen keine Vertragsbedingungen anbieten oder fordern, die die Bedingungen dieser Lizenz oder die durch sie gewährten Rechte ändern oder beschränken. Sie dürfen den Schutzgegenstand nicht unterlizenzieren. Sie müssen alle Hinweise unverändert lassen, die auf diese Lizenz und den Haftungsausschluss hinweisen. Sie dürfen den Schutzgegenstand mit keinen technischen Schutzmaßnahmen versehen, die den Zugang oder den Gebrauch des Schutzgegenstandes in einer Weise kontrollieren, die mit den Bedingungen dieser Lizenz im Widerspruch stehen. Die genannten Beschränkungen gelten auch für den Fall, dass der Schutzgegenstand einen Bestandteil eines Sammelwerkes bildet; sie verlangen aber nicht, dass das Sammelwerk insgesamt zum Gegenstand dieser Lizenz gemacht wird. Wenn Sie ein Sammelwerk erstellen, müssen Sie - soweit dies praktikabel ist - auf die Mitteilung eines Lizenzgebers oder Urhebers hin aus dem Sammelwerk jeglichen Hinweis auf diesen Lizenzgeber oder diesen Urheber entfernen. Wenn Sie den Schutzgegenstand bearbeiten, müssen Sie - soweit dies praktikabel ist- auf die Aufforderung eines Rechtsinhabers hin von der Bearbeitung jeglichen Hinweis auf diesen Rechtsinhaber entfernen.

- Wenn Sie den Schutzgegenstand oder ein Sammelwerk vervielfältigen, verbreiten oder öffentlich wiedergeben, müssen Sie alle Urhebervermerke für den Schutzgegenstand unverändert lassen und die Urheberschaft oder Rechtsinhaberschaft in einer der von Ihnen vorgenommenen Nutzung angemessenen Form anerkennen, indem Sie den Namen (oder das Pseudonym, falls ein solches verwendet wird) des Urhebers oder Rechteinhabers nennen, wenn dieser angegeben ist. Dies gilt auch für den Titel des Schutzgegenstandes, wenn dieser angeben ist, sowie - in einem vernünftigerweise durchführbaren Umfang - für die mit dem Schutzgegenstand zu verbindende Internetadresse in Form des Uniform- Resource-Identifier (URI), wie sie der Lizenzgeber angegeben hat, sofern dies geschehen ist, es sei denn, diese Internetadresse verweist nicht auf den Urhebervermerk oder die Lizenzinformationen zu dem Schutzgegenstand. Ein solcher Hinweis kann in jeder angemessenen Weise erfolgen, wobei jedoch bei einer Datenbank oder einem Sammelwerk der Hinweis zumindest an gleicher Stelle und in ebenso auffälliger Weise zu erfolgen hat wie vergleichbare Hinweise auf andere Rechtsinhaber.

- Obwohl die gemäss Ziffer 3 gewährten Nutzungsrechte in umfassender Weise ausgeübt werden dürfen, findet diese Erlaubnis ihre gesetzliche Grenze in den Persönlichkeitsrechten der Urheber und ausübenden Künstler, deren berechtigte geistige und persönliche Interessen bzw. deren Ansehen oder Ruf nicht dadurch gefährdet werden dürfen, dass ein Schutzgegenstand über das gesetzlich zulässige Maß hinaus beeinträchtigt wird.

5. Gewährleistung. Sofern dies von den Vertragsparteien nicht anderweitig schriftlich vereinbart,, bietet der Lizenzgeber keine Gewährleistung für die erteilten Rechte, außer für den Fall, dass Mängel arglistig verschwiegen wurden. Für Mängel anderer Art, insbesondere bei der mangelhaften Lieferung von Verkörperungen des Schutzgegenstandes, richtet sich die Gewährleistung nach der Regelung, die die Person, die Ihnen den Schutzgegenstand zur Verfügung stellt, mit Ihnen außerhalb dieser Lizenz vereinbart, oder - wenn eine solche Regelung nicht getroffen wurde - nach den gesetzlichen Vorschriften.

6. Haftung. Über die in Ziffer 5 genannte Gewährleistung hinaus haftet Ihnen der Lizenzgeber nur für Vorsatz und grobe Fahrlässigkeit.

7. Vertragsende

- Dieser Lizenzvertrag und die durch ihn eingeräumten Nutzungsrechte enden automatisch bei jeder Verletzung der Vertragsbedingungen durch Sie. Für natürliche und juristische Personen, die von Ihnen eine Datenbank oder ein Sammelwerk unter diesen Lizenzbedingungen erhalten haben, gilt die Lizenz jedoch weiter, vorausgesetzt, diese natürlichen oder juristischen Personen erfüllen sämtliche Vertragsbedingungen. Die Ziffern 1, 2, 5, 6, 7 und 8 gelten bei einer Vertragsbeendigung fort.

- Unter den oben genannten Bedingungen erfolgt die Lizenz auf unbegrenzte Zeit (für die Dauer des Schutzrechts). Dennoch behält sich der Lizenzgeber das Recht vor, den Schutzgegenstand unter anderen Lizenzbedingungen zu nutzen oder die eigene Weitergabe des Schutzgegenstandes jederzeit zu beenden, vorausgesetzt, dass solche Handlungen nicht dem Widerruf dieser Lizenz dienen (oder jeder anderen Lizenzierung, die auf Grundlage dieser Lizenz erfolgt ist oder erfolgen muss) und diese Lizenz wirksam bleibt, bis Sie unter den oben genannten Voraussetzungen endet.

8. Schlussbestimmungen

Jedes Mal, wenn Sie den Schutzgegenstand vervielfältigen, verbreiten oder öffentlich wiedergeben, bietet der Lizenzgeber dem Erwerber eine Lizenz für den Schutzgegenstand unter denselben Vertragsbedingungen an, unter denen er Ihnen die Lizenz eingeräumt hat.

Sollte eine Bestimmung dieses Lizenzvertrages unwirksam sein, so wird die Wirksamkeit der übrigen Lizenzbestimmungen dadurch nicht berührt, und an die Stelle der unwirksamen Bestimmung tritt eine Ersatzregelung, die dem mit der unwirksamen Bestimmung angestrebten Zweck am nächsten kommt.

Nichts soll dahingehend ausgelegt werden, dass auf eine Bestimmung dieses Lizenzvertrages verzichtet oder einer Vertragsverletzung zugestimmt wird, so lange ein solcher Verzicht oder eine solche Zustimmung nicht schriftlich vorliegen und von der verzichtenden oder zustimmenden Vertragspartei unterschrieben sind

Dieser Lizenzvertrag stellt die vollständige Vereinbarung zwischen den Vertragsparteien hinsichtlich des Schutzgegenstandes dar. Es gibt keine weiteren ergänzenden Vereinbarungen oder mündlichen Abreden im Hinblick auf den Schutzgegenstand. Der Lizenzgeber ist an keine zusätzlichen Abreden gebunden, die aus irgendeiner Absprache mit Ihnen entstehen könnten. Der Lizenzvertrag kann nicht ohne eine übereinstimmende schriftliche Vereinbarung zwischen dem Lizenzgeber und Ihnen abgeändert werden.

Auf diesen Lizenzvertrag findet das Recht der Bundesrepublik Deutschland Anwendung.

CREATIVE COMMONS IST KEINE VERTRAGSPARTEI DIESES LIZENZVERTRAGES UND ÜBERNIMMT KEINERLEI GEWÄHRLEISTUNG FÜR DAS WERK. CREATIVE COMMONS IST IHNEN ODER DRITTEN GEGENÜBER NICHT HAFTBAR FÜR SCHÄDEN JEDWEDER ART. UNGEACHTET DER VORSTEHENDEN ZWEI (2) SÄTZE HAT CREATIVE COMMONS ALL RECHTE UND PFLICHTEN EINES LIZENSGEBERS, WENN SICH CREATIVE COMMONS AUSDRÜCKLICH ALS LIZENZGEBER BEZEICHNET.

AUSSER FÜR DEN BESCHRÄNKTEN ZWECK EINES HINWEISES AN DIE ÖFFENTLICHKEIT, DASS DAS WERK UNTER DER CCPL LIZENSIERT WIRD, DARF KENIE VERTRAGSPARTEI DIE MARKE “CREATIVE COMMONS” ODER EINE ÄHNLICHE MARKE ODER DAS LOGO VON CREATIVE COMMONS OHNE VORHERIGE GENEHMIGUNG VON CREATIVE COMMONS NUTZEN. JEDE GESTATTETE NUTZUNG HAT IN ÜBREEINSTIMMUNG MIT DEN JEWEILS GÜLTIGEN NUTZUNGSBEDINGUNGEN FÜR MARKEN VON CREATIVE COMMONS ZU ERFOLGEN, WIE SIE AUF DER WEBSITE ODER IN ANDERER WEISE AUF ANFRAGE VON ZEIT ZU ZEIT ZUGÄNGLICH GEMACHT WERDEN.

CREATIVE COMMONS KANN UNTER http://creativecommons.org/ KONTAKTIERT WERDEN.

11.4 Spenden

Spenden?

Wenn Sie mich bei meiner Arbeit an der FAQ und dem StopenSuchtool unterstützen wollen, können Sie mir, wenn Sie wollen, einen kleinen Betrag spenden. Da ich zwischenzeitlich auch von der allgemeinen guten Lage der Wirtschaft erreicht worden bin, freue ich mich über jede Bestätigung meiner Arbeit. (Auch ein Dankesmail ist eine nette Bestätigung....)

Sowohl an der StarBasicFAQ wie auch an dem Suchtool wird ständig weiter gearbeitet. Wenn Sie diese Arbeit durch einen kleinen Obulus unterstützen wollen, können Sie dies gerne machen.

Spendenkonto:

Kontonummer: 2547488700

Bankeitzahl: 500 101 11

Bank: SEB Bank