StarOffice 8 Programmierhandbuch für BASIC

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Teilenr.: 819–1326–01
2005

Copyright 2005 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Alle Rechte vorbehalten.

Dieses Produkt und die Dokumentation sind urheberrechtlich geschützt und werden unter Lizenzen vertrieben, durch die die Verwendung, das Kopieren, Verteilen und Dekompilieren eingeschränkt werden. Ohne vorherige schriftliche Genehmigung durch Sun und gegebenenfalls seiner Lizenzgeber darf kein Teil dieses Produkts oder Dokuments in irgendeiner Form reproduziert werden. Die Software anderer Hersteller, einschließlich der Schriftentechnologie, ist urheberrechtlich geschützt und von Lieferanten von Sun lizenziert.

Teile des Produkts können aus Berkeley BSD-Systemen stammen, die von der University of California lizenziert sind. UNIX ist eine eingetragene Marke in den Vereinigten Staaten und anderen Ländern und wird ausschließlich durch die X/Open Company Ltd. lizenziert.

Sun, Sun Microsystems, das Sun-Logo, docs.sun.com, AnswerBook, AnswerBook2, und Solaris sind in den USA und anderen Ländern Warenzeichen von Sun Microsystems Inc. Sämtliche SPARC-Marken werden unter Lizenz verwendet und sind Marken oder eingetragene Marken von SPARC International Inc. in den Vereinigten Staaten und anderen Ländern. Produkte mit der SPARC-Marke basieren auf einer von Sun Microsystems Inc. entwickelten Architektur.

Die grafischen Benutzeroberflächen von OPEN LOOK und Sun™ wurden von Sun Microsystems Inc. für seine Benutzer und Lizenznehmer entwickelt. Sun erkennt dabei die von Xerox Corporation geleistete Forschungs- und Entwicklungsarbeit auf dem Gebiet der visuellen oder grafischen Benutzeroberflächen für die Computerindustrie an. Sun verfügt über eine nicht-exklusive Lizenz von Xerox über die grafische Benutzeroberfläche von Xerox. Diese Lizenz gilt auch für die Lizenznehmer von Sun, die OPEN LOOK-GUIs implementieren und sich an die schriftlichen Lizenzvereinbarungen mit Sun halten.

U.S. Government Rights – Commercial software. Regierungsbenutzer unterliegen der standardmäßigen Lizenzvereinbarung von Sun Microsystems, Inc. sowie den anwendbaren Bestimmungen der FAR und ihrer Zusätze.

DIE DOKUMENTATION WIRD “AS IS” BEREITGESTELLT, UND JEGLICHE AUSDRÜCKLICHE ODER IMPLIZITE BEDINGUNGEN, DARSTELLUNGEN UND HAFTUNG, EINSCHLIESSLICH JEGLICHER STILLSCHWEIGENDER HAFTUNG FÜR MARKTFÄHIGKEIT, EIGNUNG FÜR EINEN BESTIMMTEN ZWECK ODER NICHTÜBERTRETUNG WERDEN IM GESETZLICH ZULÄSSIGEN RAHMEN AUSDRÜCKLICH AUSGESCHLOSSEN.

Copyright 2005 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, et Solaris sont des marques de fabrique ou des marques déposées, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE “EN L'ETAT” ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

070427@17039

Inhalt

◆ ◆ ◆ KAPITEL 1

1 Einleitung

Dieses Handbuch führt in die Programmierung mit StarOffice™ 8 Basic ein und zeigt, welche Anwendungsmöglichkeiten sich durch die Verwendung von StarOffice Basic in StarOffice eröffnen. Um maximalen Nutzen aus diesem Buch zu ziehen, sollten Sie bereits mit Programmiersprachen vertraut sein.

Sie erhalten umfangreiche Beispiele, damit Sie schnell Ihre eigenen StarOffice Basic-Programme entwickeln können.

Aufbau dieses Buches

Die ersten drei Kapitel bieten eine Einführung in StarOffice Basic:

- Kapitel 2, Die Sprache StarOffice Basic
- Kapitel 3, Die Laufzeitbibliothek von StarOffice Basic
- Kapitel 4, Einführung in die StarOffice API

In diesen Kapiteln erhalten Sie einen Überblick über StarOffice Basic. Deshalb sollte Sie von jedem gelesen werden, der beabsichtigt, StarOffice Basic-Programme zu schreiben.

In den verbleibenden Kapiteln werden die einzelnen Komponenten der StarOffice API eingehender beschrieben. Deshalb können sie eher selektiv nach Bedarf gelesen werden:

- Kapitel 5, Arbeiten mit StarOffice-Dokumenten
- Kapitel 6, Textdokumente
- Kapitel 7, Tabellendokumente
- Kapitel 8, Zeichnungen und Präsentationen
- Kapitel 9, Diagramme (Charts)
- Kapitel 10, Datenbankzugriff
- Kapitel 11, Dialoge
- Kapitel 12, Formulare

Informationen über StarOffice Basic

Die Programmiersprache StarOffice Basic wurde speziell für StarOffice Basic entwickelt und ist fest in die Office Suite integriert.

Wie der Name bereits nahe legt, handelt es sich bei StarOffice Basic um eine Programmiersprache aus der Basic-Familie. Wer bisher mit anderen Basic-Sprachen gearbeitet hat – insbesondere mit Visual Basic oder Visual Basic for Applications (VBA) von Microsoft – wird sich schnell in StarOffice Basic zurecht finden. Die Basiskonstrukte von StarOffice Basic sind über weite Teile kompatibel mit Visual Basic.

Die Programmiersprache StarOffice Basic lässt sich in vier Komponenten unterteilen:

- **Die Sprache StarOffice Basic**: Sie definiert die elementaren Sprachkonstrukte, etwa für Variablendeklarationen, Schleifen und Funktionen.
- **Die Laufzeitbibliothek**: Sie bietet Standardfunktionen, die nicht unmittelbar mit StarOffice in Beziehung stehen, zum Beispiel Funktionen für das Bearbeiten von Zahlen, Zeichenfolgen, Datumswerten und Dateien.
- **Die StarOffice API (Application Programming Interface)**: Sie gestattet den Zugriff auf StarOffice-Dokumente und erlaubt es, diese zu erstellen, zu speichern, zu ändern und zu drucken.
- **Der Dialog-Editor**: Er dient zum Erstellen eigener Dialogfenster und bietet Möglichkeiten, um diese mit Steuerelementen und Ereignis-Handlern zu versehen.

Hinweis – Die Kompatibilität zwischen StarOffice Basic und VBA bezieht sich sowohl auf die Sprache StarOffice Basic als auch auf die Laufzeitbibliothek. Die StarOffice API und der Dialog-Editor sind *nicht* kompatibel mit VBA (durch eine Standardisierung dieser Schnittstellen wären zahlreiche der in StarOffice bereitgestellten Konzepte nicht realisierbar gewesen).

Zielgruppe für die Verwendung von StarOffice Basic

Der Anwendungsbereich von StarOffice Basic beginnt dort, wo die Standardfunktionen von StarOffice enden. So lassen sich in StarOffice Basic Routineaufgaben automatisieren, Verknüpfungen mit anderen Programmen – etwa einem Datenbankserver – herstellen und komplexe Tätigkeiten in Form von vordefinierten Skripten auf Knopfdruck ausführen.

StarOffice Basic bietet vollständigen Zugriff auf alle Funktionen von StarOffice, kann sämtliche unterstützen Dokumenttypen bearbeiten und bietet Möglichkeiten zur Erstellung eigener Dialogfenster.

Verwenden von StarOffice Basic

StarOffice Basic kann ohne zusätzliche Programme oder Hilfsmittel von jedem StarOffice-Anwender verwendet werden. Bereits in der Standardinstallation verfügt StarOffice Basic über alle Komponenten, die für die Erstellung eigener Basic-Makros notwendig sind. Hierzu gehören unter anderem:

- **Die integrierte Entwicklungsumgebung** (Integrated Development Environment, IDE), die einen Editor zum Erstellen und Testen von Makros zur Verfügung stellt.
- **Der Interpreter**, der für das Ausführen von StarOffice Basic-Makros notwendig ist.
- **Die Schnittstellen** zu den verschiedenen StarOffice-Anwendungen, die direkten Zugriff auf Office-Dokumente gestatten.

Weiterführende Informationen

Die in diesem Handbuch behandelten Komponenten der StarOffice API wurden aufgrund der praktischen Vorteile ausgewählt, die sie dem StarOffice Basic-Programmierer bieten können. Generell werden nur Teile der Schnittstellen erläutert. Wer sich ein detaillierteres Bild dazu verschaffen möchte, sei auf die API-Referenz verwiesen, die im Internet unter der folgenden Adresse zu finden ist:

```
http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html
```

Der Developer's Guide beschreibt die StarOffice API detaillierter als das vorliegende Handbuch, wendet sich jedoch primär an Java- und C++-Programmierer. Alle, die bereits mit der StarOffice Basic-Programmierung vertraut sind, finden im Developer's Guide zusätzliche Information über die StarOffice Basic- und StarOffice-Programmierung. Der Developer's Guide kann im Internet unter der folgenden Adresse heruntergeladen werden:

```
http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html
```

Programmierer, die anstelle von StarOffice Basic direkt mit Java oder C++ arbeiten möchten, sollten auf den StarOffice Developer's Guide statt auf dieses Handbuch zurückgreifen. Die StarOffice-Programmierung mit Java oder C++ ist allerdings deutlich komplexer als die mit StarOffice Basic.

◆ ◆ ◆ KAPITEL 2

2 Die Sprache StarOffice Basic

StarOffice Basic gehört zur Familie der Basic-Sprachen. Viele Teile von StarOffice Basic sind mit Microsoft Visual Basic for Applications (VBA) und Microsoft Visual Basic identisch. Wer mit diesen Sprachen bereits einmal gearbeitet hat, findet sich schnell in StarOffice Basic zurecht.

Aber auch Programmierer anderer Sprachen – etwa Java, C++ oder Delphi – dürften keine großen Schwierigkeiten haben, sich in StarOffice Basic einzuarbeiten. StarOffice Basic ist eine vollständig entwickelte prozedurale Programmiersprache, die das Stadium der Verwendung rudimentärer Kontrollstrukturen wie `GoTo` und `GoSub` hinter sich gelassen hat.

Auch von den Vorteilen der objektorientierten Programmierung können Sie profitieren, da Sie über eine in StarOffice Basic vorhandene Schnittstelle externe Objekt-Bibliotheken verwenden können. Die gesamte StarOffice API basiert auf diesen Schnittstellen, die in den folgenden Kapiteln dieses Dokuments detailliert beschrieben werden.

In diesem Kapitel erhalten Sie einen Überblick über die Hauptelemente und -konstrukte der Sprache StarOffice Basic sowie das Rahmenwerk, an dem sich alle Anwendungen und Bibliotheken in StarOffice Basic orientieren.

Übersicht eines StarOffice Basic-Programms

Bei StarOffice Basic handelt es sich um eine Interpreter-Sprache. Im Gegensatz zu klassischen Compiler-Sprachen wie C++ oder Turbo Pascal erzeugt der StarOffice-Compiler jedoch keine ausführbaren oder selbstentpackenden Dateien, die automatisch ausgeführt werden können. Stattdessen kann ein StarOffice Basic-Programm durch Druck auf eine Taste bzw. Klicken auf eine Schaltfläche ausgeführt werden. Der Code wird zuerst auf offensichtliche Fehler überprüft und dann zeilenweise ausgeführt.

Programmzeilen

Der zeilenorientierte Ausführungsvorgang des Basic-Interpreters stellt einen der wesentlichen Unterschiede zwischen Basic und anderen Programmiersprachen dar. Während die Position von Zeilenumbrüchen innerhalb des Quellcodes in Java, C++ oder Delphi bedeutungslos ist, bilden die Zeilen eines Basic-Programms jeweils eine eigene, für sich abgeschlossene Einheit. Funktionsaufrufe, mathematische Ausdrücke und andere Sprachelemente wie Funktions- und Schleifenköpfe müssen stets in der Zeile abgeschlossen werden, in der sie beginnen.

Reicht der Platz dafür nicht aus oder führt dies zu unübersichtlich langen Zeilen, besteht die Möglichkeit, mehrere Zeilen durch das Hinzufügen von Unterstrichen ("_") miteinander zu verbinden. Das folgende Beispiel zeigt, wie vier Zeilen eines mathematischen Ausdrucks verbunden werden können:

```
LongExpression = (Expression1 * Expression2) + _
   (Expression3 * Expression4) + _
   (Expression5 * Expression6) + _
   (Expression7 * Expression8)
```

Hinweis – Der Unterstrich muss immer das letzte Zeichen innerhalb einer verknüpften Zeile sein und darf nie von einem Leerzeichen oder Tabulator gefolgt werden; andernfalls erzeugt der Code einen Fehler.

Neben dem Verknüpfen einzelner Zeilen können Sie bei StarOffice Basic eine Zeile durch Doppelpunkte in mehrere Bereiche unterteilen, so dass ausreichender Platz für mehrere Ausdrücke entsteht. So lassen sich beispielsweise die Zuweisungen

```
a = 1
a = a + 1
a = a + 1
```

wie folgt schreiben:

```
a = 1  :  a = a + 1  :  a = a + 1
```

Kommentare

Neben dem auszuführenden Programmcode kann ein StarOffice Basic-Programm auch Kommentare enthalten, die einzelne Programmteile erklären und wichtige Informationen bereitstellen, die sich zu einem späteren Zeitpunkt als nützlich erweisen können.

StarOffice Basic bietet zwei Methoden zum Einfügen von Kommentaren in den Programmcode:

- Alle Zeichen, die auf ein Apostroph-Zeichen folgen, werden als Kommentar behandelt:

```
Dim A    ' Dies ist ein Kommentar für Variable A
```

- Das Schlüsselwort `Rem`, gefolgt von einem Kommentar:

```
Rem Dieser Kommentar wird durch das Schlüsselwort Rem eingeleitet.
```

Ein Kommentar umfasst normalerweise alle Zeichen bis zum Ende der Zeile. Die Folgezeile interpretiert StarOffice Basic hingegen wieder als reguläre Anweisung. Bei mehrzeiligen Kommentaren muss daher jede Zeile als Kommentar gekennzeichnet werden:

```
Dim B     ' Dieser Kommentar für die Variable B ist relativ lang
          ' und erstreckt sich über mehrere Zeilen. Das
          ' Kommentarzeichen muss daher in jeder Zeile
          ' wiederholt werden.
```

Bezeichner

Ein StarOffice Basic-Programm kann dutzende, hunderte oder gar tausende *Bezeichner* (Marker) enthalten, die Namen für Variablen, Konstanten, Funktionen usw. darstellen. Bei der Wahl eines Namens als Bezeichner sind folgende Regeln zu beachten:

- Bezeichner dürfen nur aus lateinischen Buchstaben, Ziffern und Unterstrichen (_) bestehen.
- Das erste Zeichen eines Bezeichners muss ein Buchstabe oder ein Unterstrich sein.
- Bezeichner dürfen keine Sonderzeichen wie ä â î ß enthalten.
- Die maximal zulässige Länge eines Bezeichners beträgt 255 Zeichen.
- Es findet keine Unterscheidung von Groß- und Kleinschreibung statt. Der Bezeichner `EineTestVariable` definiert beispielsweise dieselbe Variable wie `einetestVariable` und `EINETESTVARIABLE`.

  Von dieser Regel gibt es jedoch eine Ausnahme: Bei Konstanten der UNO-API wird zwischen Groß- und Kleinschreibung unterschieden. Weitere Informationen zu UNO finden Sie in Kapitel 4.

Hinweis – Die Regeln für den Aufbau von Bezeichnern unterscheiden sich in StarOffice Basic gegenüber VBA. So lässt StarOffice Basic beispielsweise Sonderzeichen in Bezeichnern nur zu, wenn "Option Compatible" verwendet wird, da es bei internationalen Projekten hierdurch zu Problemen kommen kann.

Hier einige Beispiele für richtige und falsche Bezeichner:

```
Surname      ' Korrekt
Surname5     ' Korrekt (Zahl 5 steht nicht an erster Stelle)
First Name   ' Falsch (Leerzeichen nicht erlaubt)
DéjàVu       ' Falsch (Buchstaben wie é, à nicht erlaubt)
5Surnames    ' Falsch (erstes Zeichen darf keine Zahl sein)
First,Name   ' Falsch (Komma und Punkt nicht erlaubt)
```

Arbeiten mit Variablen

Implizite Variablendeklaration

Basic-Sprachen sind sehr anwenderfreundlich angelegt. Hieraus resultiert, dass StarOffice Basic die Erstellung von Variablen durch den einfachen Gebrauch ohne explizite Deklaration unterstützt. Mit anderen Worten ist eine Variable von dem Moment an vorhanden, in dem sie im Code verwendet wird. In Abhängigkeit von den bereits vorhandenen Variablen deklariert das folgende Beispiel bis zu drei neue Variablen:

```
a = b + c
```

Das implizite Deklarieren von Variablen ist kein guter Programmierstil, da es dadurch zur versehentlichen Erzeugung neuer Variablen durch beispielsweise Tipp-/Eingabefehler kommen kann. Statt also eine Fehlermeldung auszugeben, initialisiert der Interpreter den Tippfehler als neue Variable mit dem Wert 0. Das Auffinden derartiger Fehler im Code kann sehr schwierig sein.

Explizite Variablendeklaration

Um durch eine implizite Variablendeklaration verursachte Fehler zu vermeiden, bietet StarOffice Basic einen Schalter namens

```
Option Explicit
```

Er muss in der ersten Programmzeile eines jeden Moduls aufgeführt werden und stellt sicher, dass eine Fehlermeldung ausgegeben wird, wenn eine der verwendeten Variablen nicht deklariert ist. Der Schalter `Option Explicit` sollte in allen Basic-Modulen verwendet werden.

In seiner einfachsten Form lautet der Befehl für eine explizite Variablendeklaration:

```
    Dim MyVar
```

Dieses Beispiel deklariert eine Variable mit dem Namen `MyVar` und dem Typ Variant. Ein Variant ist eine Universalvariable, die alle möglichen Werte aufnehmen kann, etwa Zeichenfolgen, ganze Zahlen, Fließkommazahlen und Boolean-Werte. Es folgen einige Beispiele für Variant-Variablen:

```
MyVar = "Hello World"      ' Zuweisung einer Zeichenfolge
MyVar = 1                  ' Zuweisung einer ganzen Zahl
MyVar = 1.0                ' Zuweisung einer Fließkommazahl
MyVar = True               ' Zuweisung eines Boolean-Werts
```

Die im vorangehenden Beispiel deklarierten Variablen können sogar in ein- und demselben Programm für verschiedene Typen verwendet werden. Obgleich hieraus eine beachtliche Flexibilität resultiert, sollte man eine Variable auf nur einen Variablentyp beschränken. Wenn StarOffice Basic einen fehlerhaft definierten Variablentyp in einem bestimmten Kontext erkennt, wird eine Fehlermeldung erzeugt.

Verwenden Sie folgende Syntax, um eine typgebundene Variablendeklaration vorzunehmen:

```
Dim MyVar As Integer   ' Deklaration einer Variablen vom Typ Integer
```

Die Variable wird mit dem Typ Integer deklariert und kann ganzzahlige Werte aufnehmen. Sie können auch mit folgender Syntax eine Variable vom Typ Integer deklarieren:

```
Dim MyVar%          ' Deklaration einer Variablen vom Typ Integer
```

Die Dim-Anweisung kann mehrere Variablendeklarationen aufnehmen:

```
Dim MyVar1, MyVar2
```

Wenn die Variablen einem permanenten Typ zugewiesen werden sollen, müssen Sie für jede Variable eine gesonderte Zuweisung vornehmen:

```
Dim MyVar1 As Integer, MyVar2 As Integer
```

Wird der Typ einer Variable nicht deklariert, weist StarOffice Basic der Variable einen Variant-Typ zu. In der folgenden Variablendeklaration wird beispielsweise `MyVar1` Variant zugewiesen und `MyVar2` Integer:

```
Dim MyVar1, MyVar2 As Integer
```

Im Folgenden werden alle in StarOffice Basic verfügbaren Variablentypen aufgeführt und beschrieben, wie sie eingesetzt und deklariert werden können.

Zeichenfolgen

Zeichenfolgen bilden zusammen mit den Zahlen die wichtigsten Basistypen von StarOffice Basic. Eine Zeichenfolge (String) besteht aus einer Kette aufeinander folgender Einzelzeichen. Intern speichert der Computer die Zeichenfolgen als Zahlenfolge ab, wobei jede Zahl für ein bestimmtes Zeichen steht.

Vom ASCII-Zeichensatz zu Unicode

In Zeichensätzen werden in einer Zeichenfolge enthaltene Zeichen einem entsprechenden Code in einer Tabelle zugeordnet (Zahlen und Buchstaben), die beschreibt, wie der Computer die Zeichenfolge anzeigen soll.

Der ASCII-Zeichensatz

Der ASCII-Zeichensatz ist ein Satz von Codes, der Zahlen, Buchstaben und Sonderzeichen mit jeweils einem Byte darstellt. Die ASCII-Codes 0 bis 127 entsprechen dem Alphabet und gängigen Symbolen (Punkt, Klammern, Komma) sowie einigen speziellen Steuerzeichen für Bildschirm und Drucker. Der ASCII-Zeichensatz wird normalerweise als Standardformat zur Übertragung von Texten zwischen unterschiedlichen Computern verwendet.

Dieser Zeichensatz enthält allerdings keine der in Europa verwendeten Sonderzeichen wie â, ä oder î, noch andere Zeichenformate wie zum Beispiel das kyrillische Alphabet.

Der ANSI-Zeichensatz

Bei der Entwicklung des Produkts Windows verwendete Microsoft den Zeichensatz des American National Standards Institute, kurz ANSI, der schrittweise um zusätzliche Zeichen erweitert wurde, die im ASCII-Zeichensatz nicht enthalten sind.

Codepages

Der ISO 8859-Zeichensatz stellt einen internationalen Standard dar. Die ersten 128 Zeichen des ISO-Zeichensatzes entsprechen dem ASCII-Zeichensatz. Mit dem ISO-Standard wurden neue Zeichensätze eingeführt (*Codepages*), damit weitere Sprachen korrekt dargestellt werden konnten. Hieraus resultierte jedoch auch, dass derselbe Zeichenwert in unterschiedlichen Sprachen verschiedene Zeichen darstellen kann.

Unicode

Unicode erhöht die Länge eines Zeichens auf vier Byte und kombiniert verschiedene Zeichensätze, um einen Standard zu erzeugen, mit dem so viele der Weltsprachen wie möglich dargestellt werden können sollen. Die Version 2.0 von Unicode wird mittlerweile von vielen Programmen unterstützt, wozu auch StarOffice und StarOffice Basic gehören.

String-Variablen

StarOffice Basic speichert Zeichenfolgen als String-Variablen in Unicode. Eine String-Variable kann bis zu 65535 Zeichen aufnehmen. Intern speichert StarOffice Basic für jedes Zeichen den verknüpften Unicode-Wert. Der für eine String-Variable benötigte Arbeitsspeicher hängt davon ab, wie lang die Zeichenfolge ist.

Beispiel-Deklarationen einer String-Variablen:

```
Dim Variable As String
```

Diese Deklaration kann auch wie folgt geschrieben werden:

```
Dim Variable$
```

Hinweis – Achten Sie beim Portieren von VBA-Anwendungen darauf, dass diese die in StarOffice Basic maximal zugelassene Länge für Zeichenfolgen einhalten (65535 Zeichen).

Angeben expliziter Zeichenfolgen

Um einer String-Variable eine explizite Zeichenfolge zuzuweisen, schließen Sie die Zeichenfolge in Anführungszeichen ein (").

```
Dim MyString As String
MyString = "Dies ist ein Test."
```

Um eine Zeichenfolge auf zwei Zeilen aufzuteilen, fügen Sie am Ende der ersten Zeile ein Pluszeichen an:

```
Dim MyString As String
MyString = "Diese Zeichenfolge ist so lang, dass sie " + _
           "auf zwei Zeilen aufgeteilt wurde."
```

Um in einer Zeichenfolge ein Anführungszeichen (") verwenden zu können, müssen Sie es an der gewünschten Stelle doppelt eingeben:

```
Dim MyString As String
MyString = "ein ""-Anführungszeichen."    ' ergibt ein "-Anführungszeichen
```

Zahlen

StarOffice Basic unterstützt fünf Basistypen zur Verarbeitung von Zahlen:

- Integer
- Long Integer
- Float
- Double
- Currency

Integer-Variablen

Integer-Variablen können jede ganze Zahl zwischen -32768 und 32767 aufnehmen. Eine Integer-Variable kann bis zu 2 Byte Speicherplatz belegen. Das Typ-Deklarationszeichen für Integer-Variablen ist %. Berechnungen, die Integer-Variablen verwenden, sind sehr schnell und besonders nützlich für Schleifenzähler. Wenn Sie einer Integer-Variable eine Fließkommazahl zuweisen, wird die Zahl auf die nächste ganze Zahl auf- oder abgerundet.

Beispiel-Deklarationen für Integer-Variablen:

```
Dim Variable As Integer
Dim Variable%
```

Long Integer-Variablen

Long Integer-Variablen können jede ganze Zahl zwischen -2147483648 und 2147483647 aufnehmen. Eine Long Integer-Variable kann bis zu 4 Byte Speicher belegen. Das Typ-Deklarationszeichen für Long Integer-Variablen ist &. Berechnungen, die Long Integer-Variablen verwenden, sind sehr schnell und besonders nützlich für Schleifenzähler. Wenn Sie einer Long Integer-Variable eine Fließkommazahl zuweisen, wird die Zahl auf die nächste ganze Zahl auf- oder abgerundet.

Beispiel-Deklarationen für Long Integer-Variablen:

```
Dim Variable as Long
Dim Variable&
```

Single-Variablen

Single-Variablen können jede positive oder negative Fließkommazahl zwischen $3{,}402823 \times 10^{38}$ und $1{,}401298 \times 10^{-45}$ aufnehmen. Eine Single-Variable kann bis zu 4 Byte Speicherplatz belegen. Das Typ-Deklarationszeichen für Single-Variablen ist !.

Ursprünglich wurden Single-Variablen zur Verringerung der für die präziseren Double-Variablen benötigten Verarbeitungszeit eingesetzt. Dieser Geschwindigkeitsaspekt trifft jedoch nicht mehr zu, so dass Single-Variablen kaum mehr benötigt werden.

Beispiel-Deklarationen für Single-Variablen:

```
Dim Variable as Single
Dim Variable!
```

Double-Variablen

Double-Variablen können jede positive oder negative Fließkommazahl zwischen $1{,}79769313486232 \times 10^{308}$ und $4{,}94065645841247 \times 10^{-324}$ aufnehmen. Eine Double-Variable kann bis zu 8 Byte Speicherplatz belegen. Double-Variablen eignen sich für exakte Berechnungen. Das Typ-Deklarationszeichen ist #.

Beispiel-Deklarationen für Double-Variablen:

```
Dim Variable As Double
Dim Variable#
```

Currency-Variablen (Währung)

Currency-Variablen unterscheiden sich von den anderen Variablentypen in der Art, wie sie Werte verarbeiten. Das Dezimaltrennzeichen ist fest und wird von vier Dezimalstellen gefolgt. Die Variable kann vor dem Dezimaltrennzeichen bis zu 15 Ziffern enthalten. Eine Currency-Variable kann jeden Wert zwischen -922337203685477,5808 und +922337203685477,5807 aufnehmen und bis zu 8 Byte Speicherplatz belegen. Das Typ-Deklarationszeichen für Currency-Variablen ist @.

Currency-Variablen sind hauptsächlich für kaufmännische Berechnungen gedacht, bei denen durch die Verwendung von Fließkommazahlen unvorhersehbare Rundungsfehler auftreten können.

Beispiel-Deklarationen für Currency-Variablen:

```
Dim Variable As Currency
Dim Variable@
```

Angeben expliziter Zahlen

Zahlen können auf zahlreiche Arten dargestellt werden, im Dezimalformat, in wissenschaftlicher Notation oder sogar in einem anderen Zahlensystem als dezimal. Folgende Regeln gelten in StarOffice Basic für Zahlen:

Ganze Zahlen

Am einfachsten ist der Umgang mit ganzen Zahlen. Sie werden im Quellcode ohne Tausendertrennzeichen aufgeführt:

```
Dim A As Integer
Dim B As Float
```

```
A = 1210
B = 2438
```

Den Zahlen kann sowohl ein Plus (+) als auch ein Minus (-) als Vorzeichen vorangestellt werden (mit oder ohne Leerzeichen zwischen Zahl und Vorzeichen):

```
Dim A As Integer
Dim B As Float

A = + 121
B = - 243
```

Dezimalzahlen

Beim Eingeben einer Dezimalzahl verwenden Sie einen Punkt (.) als Dezimaltrennzeichen. Mit dieser Regel wird sichergestellt, dass sich Quellcode ohne Konvertierung von einem Land in ein anderes Land übertragen lässt.

```
Dim A As Integer
Dim B As Integer
Dim C As Float

A = 1223.53      ' wird gerundet
B = - 23446.46   ' wird gerundet
C = + 3532.76323
```

Auch bei Dezimalzahlen ist die Verwendung der Vorzeichen Plus (+) und Minus (-) gestattet (ebenfalls mit und ohne Leerzeichen).

Wird einer Integer-Variable eine Dezimalzahl zugewiesen, rundet StarOffice Basic die Zahl auf oder ab.

Exponentialschreibweise

StarOffice Basic lässt die Angabe von Zahlen in der Exponentialschreibweise zu. So können Sie beispielsweise die Zahl 1.5×10^{-10} (0,00000000015) auch als 1.5e-10 schreiben. Der Buchstabe "e" kann hierbei klein oder groß geschrieben werden sowie mit oder ohne ein vorangestelltes Pluszeichen (+).

Hier einige Beispiele richtiger und falscher Zahlangaben in Exponentialschreibweise:

```
Dim A As Double

A = 1.43E2        ' Korrekt
A = + 1.43E2      ' Korrekt (Leerzeichen zwischen Plus und Basiszahl)
A = - 1.43E2      ' Korrekt (Leerzeichen zwischen Minus und Basiszahl)
A = 1.43E-2       ' Korrekt (Negativer Exponent)
```

```
A = 1.43E -2      ' Falsch (Leerzeichen innerhalb der Zahl nicht erlaubt)
A = 1,43E-2       ' Falsch (Komma als Dezimaltrennzeichen nicht erlaubt)
A = 1.43E2.2      ' Falsch (Exponent muss eine ganze Zahl sein)
```

Beachten Sie, dass im ersten und dritten falschen Beispiel keine Fehlermeldung erzeugt wird, obwohl die Variablen falsche Werte zurückgeben. Der Ausdruck

```
A = 1.43E -2
```

wird interpretiert als 1,43 minus 2, was dem Wert -0,57 entspricht. Gewollt war jedoch der Wert $1{,}43 * 10^2$ (entspricht 0,0143). Bei dem Wert

```
A = 1.43E2.2
```

ignoriert StarOffice Basic den Nachkommateil des Exponenten und interpretiert den Ausdruck als

```
A = 1.43E2
```

Hexadezimalwerte

Das Hexadezimalsystem (System mit Basis 16) 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. In StarOffice Basic können Sie ganzzahlige Hexadezimalwerte verwenden, solange diesen ein &H voransteht.

```
Dim A As Long

A = &HFF   ' Hexadezimalwert FF entspricht dem Dezimalwert 255
A = &H10   ' Hexadezimalwert 10 entspricht dem Dezimalwert 16
```

Oktalwerte

StarOffice Basic kann ebenfalls mit dem Oktalsystem (System mit Basis 8) umgehen, das die Zahlen 0 bis 7 verwendet. Hierbei müssen Sie ganze Zahlen mit vorangestelltem &O verwenden.

```
Dim A As Long

A = &O77   ' Oktalwert 77 entspricht dem Dezimalwert 63
A = &O10   ' Oktalwert 10 entspricht dem Dezimalwert 8
```

True (Wahr) und False (Falsch)

Boolean-Variablen

Boolean-Variablen können nur zwei Werte enthalten: `True` (Wahr) oder `False` (Falsch). Sie eignen sich für binäre Angaben, die nur genau einen der genannten Zustände einnehmen können. Ein Boolean-Wert wird intern als Integer-Wert mit 2 Byte gespeichert, wobei 0 dem `False` und jeder andere Wert dem `True` entspricht. Für Boolean-Variablen existiert kein Typ-Deklarationszeichen. Die Deklaration ist ausschließlich über den Zusatz *As Boolean* möglich.

Beispiel-Deklarationen einer Boolean-Variablen:

```
Dim Variable As Boolean
```

Datums- und Zeitangaben

Date-Variablen

Date-Variablen können Datums- und Zeitwerte enthalten. Bei der Speicherung von Datumswerten verwendet StarOffice Basic ein internes Format, das Vergleiche und mathematische Operationen an den Datums- und Zeitwerten erlaubt. Für Date-Variablen existiert kein Typ-Deklarationszeichen. Die Deklaration ist ausschließlich über den Zusatz *As Date* möglich.

Beispiel-Deklarationen einer Date-Variablen:

```
Dim Variable As Date
```

Datenfelder

Neben einfachen Variablen (*Skalare*) unterstützt StarOffice Basic auch Datenfelder (*Arrays*). Ein Datenfeld enthält mehrere Variablen, die über einen Index adressiert werden.

Einfache Arrays

Die Deklaration eines Arrays ähnelt der einer einfachen Variablendeklaration. Im Gegensatz zur Variablendeklaration folgen dem Array-Namen jedoch Klammern, die Angaben zu der Anzahl der Elemente enthält. Der Ausdruck

```
Dim MyArray(3)
```

deklariert einen Array mit 4 Variablen vom Datentyp Variant, nämlich `MyArray(0)`, `MyArray(1)`, `MyArray(2)` und `MyArray(3)`.

Innerhalb eines Arrays können auch typspezifische Variablen deklariert werden. So deklariert beispielsweise die folgende Zeile einen Array mit 4 Integer-Variablen:

```
Dim MyInteger(3) As Integer
```

In dem vorhergehenden Beispiel beginnt der Index des Arrays immer mit dem Standardstartwert 0. Alternativ kann bei der Datenfelddeklaration aber auch ein Gültigkeitsbereich mit Start- und Endwerten festgelegt werden. Folgendes Beispiel deklariert ein Datenfeld mit 6 Integer-Werten, die über die Indizes 5 bis 10 adressiert werden können:

```
Dim MyInteger(5 To 10)
```

Bei den Indizes muss es sich nicht zwangsläufig um positive Werte handeln. Das folgende Beispiel zeigt eine ebenfalls korrekte Deklaration, jedoch mit negativen Datenfeld-Grenzwerten:

```
Dim MyInteger(-10 To -5)
```

Ein Integer-Datenfeld mit 6 Werten, die mit den Indizes -10 bis -5 adressierbar sind, wird deklariert.

Bei der Definition von Datenfeldindizes müssen Sie drei Grenzwerte beachten::

- Der kleinste mögliche Index beträgt -32768.
- Der größte mögliche Index beträgt 32767.
- Die maximale Anzahl von Elementen (innerhalb einer Datenfelddimension) beträgt 16368.

Hinweis – In VBA gelten teilweise andere Grenzwerte für Datenfeldindizes. Dies gilt ebenso für die je Dimension maximal zugelassene Anzahl von Elementen. Die in VBA gültigen Werte finden Sie in der jeweiligen VBA-Dokumentation.

Vorgabewert für Startindex

Der Startindex eines Datenfeldes beginnt normalerweise mit dem Wert 0. Alternativ können Sie mit folgendem Aufruf den Startindex für alle Datenfelddeklarationen auf den Wert 1 festlegen:

```
Option Base 1
```

Der Aufruf muss im Kopfbereich eines Moduls stehen, damit er auf alle Deklarationen in dem Modul Anwendung findet. Dieser Aufruf hat jedoch keine Auswirkungen auf die mit der StarOffice API definierten UNO-Sequenzen, deren Index *immer* mit 0 beginnt. Um den Code übersichtlich zu halten, sollten Sie von der Verwendung von Option Base 1 Abstand nehmen.

Bei der Verwendung von Option Base 1 bleibt die Anzahl der in einem Array zugelassenen Elemente unberührt, lediglich der Startindex ändert sich. Die Deklaration

```
Option Base 1
' ...
Dim MyInteger(3)
```

erzeugt 4 Integer-Variablen, die sich mit den Ausdrücken `MyInteger(1)`, `MyInteger(2)`, `MyInteger(3)` und `MyInteger(4)` beschreiben lassen.

Hinweis – In StarOffice Basic hat der Ausdruck Option Base 1 im Gegensatz zu VBA keinen Einfluss auf die Anzahl der Elemente eines Arrays. In StarOffice Basic wird lediglich der Startindex verschoben. Während die Deklaration `MyInteger(3)` in VBA drei Integer-Werte mit den Indizes 1 bis 3 erzeugt, erstellt die gleiche Deklaration in StarOffice Basic vier Integer-Werte mit den Indizes 1 bis 4. Durch die Verwendung von Option Compatible verhält sich StarOffice Basic wie VBA.

Mehrdimensionale Datenfelder

Neben eindimensionalen Datenfeldern unterstützt StarOffice Basic auch die Verarbeitung mehrdimensionaler Datenfelder. Die entsprechenden Dimensionen werden durch Kommata voneinander getrennt. So definiert das Beispiel

```
Dim MyIntArray(5, 5)
```

einen Integer-Array mit zwei Dimensionen mit jeweils 6 Indizes (adressierbar über die Indizes 0 bis 5). Der gesamte Array kann somit insgesamt $6 \times 6 = 36$ Integer-Werte aufnehmen.

Obwohl Sie in StarOffice Basic hunderte von Dimensionen in einem Array definieren können, ist die Anzahl tatsächlich verwendbarer Dimensionen durch den zur Verfügung stehenden Arbeitsspeicher begrenzt.

Dynamische Dimensionsänderungen von Datenfeldern

Die vorstehenden Beispiele basieren auf Datenfeldern mit vorgegebener Dimension. Sie können aber auch Arrays definieren, bei denen sich die Dimension der Datenfelder dynamisch ändert. So können Sie beispielsweise einen Array definieren, der alle Begriffe eines Texts

enthalten soll, die mit dem Buchstaben A beginnen. Da die Anzahl dieser Wörter anfangs unbekannt ist, müssen Sie in der Lage sein, die Feldgrenzwerte nachträglich zu ändern. Hierzu verwenden Sie in StarOffice Basic folgenden Aufruf:

```
ReDim MyArray(10)
```

Hinweis – Im Gegensatz zu VBA, wo Sie nur dynamische Arrays mit `Dim MyArray()` dimensionieren können, können Sie in StarOffice Basic sowohl statische als auch dynamische Arrays mit `ReDim` ändern.

Das folgende Beispiel ändert die Dimension des Ausgangs-Arrays so, dass er zwischen 11 und 21 Werte erfassen kann:

```
Dim MyArray(4) As Integer         ' Deklaration mit fünf Elementen

' ...

ReDim MyArray(10) As Integer      ' Erhöhung auf 11 Elemente

' ...

ReDim MyArray(20) As Integer      ' Erhöhung auf 21 Elemente
```

Bei der Redimensionierung eines Arrays lassen sich sämtliche Möglichkeiten verwenden, die in den vorstehenden Abschnitten erläutert wurden. Hierzu gehören auch die Deklaration mehrdimensionaler Datenfelder sowie die Angabe expliziter Start- und Endwerte. Mit einer Dimensionsänderung des Datenfelds geht sein alter Inhalt vollständig verloren. Sollen die ursprünglichen Werte beibehalten werden, müssen Sie den Befehl `Preserve` verwenden:

```
Dim MyArray(10) As Integer            ' Definition der Ausgangs-
                                      ' dimensionen

' ...

ReDim Preserve MyArray(20)   As Integer      ' Erhöhung im
                                             ' Datenfeld unter
                                             ' Beibehaltung des  Inhalts
```

Stellen Sie bei der Verwendung von `Preserve` sicher, dass die Anzahl der Dimensionen und der Typ der Variablen unverändert bleiben.

Hinweis – Im Gegensatz zu VBA, wo lediglich die Obergrenze der letzten Dimension eines Datenfelds mit `Preserve` geändert werden kann, können Sie in StarOffice Basic ebenfalls andere Dimensionen ändern.

Wenn Sie `ReDim` zusammen mit `Preserve` verwenden, müssen Sie denselben Datentyp, wie in der ursprünglichen Datenfelddeklaration angegeben, verwenden.

Gültigkeitsbereich und Lebensdauer von Variablen

Eine Variable hat in StarOffice Basic eine begrenzte Lebensdauer und einen begrenzten Gültigkeitsbereich, innerhalb dessen sie von anderen Programmfragmenten gelesen und verwendet werden kann. Die Zeitspanne, über die eine Variable erhalten bleibt sowie von wo aus auf sie zugegriffen werden kann, hängt von der angegebenen Position und dem Typ ab.

Lokale Variablen

Variablen, die innerhalb einer Funktion oder Prozedur deklariert werden, werden als lokale Variablen bezeichnet:

```
Sub Test
   Dim MyInteger As Integer

   ' ...
End Sub
```

Lokale Variablen behalten ihre Gültigkeit nur so lange, wie die Funktion oder Prozedur ausgeführt wird; danach werden sie auf null zurückgesetzt. Bei jedem Aufruf der Funktion stehen die zuvor erzeugten Werte nicht mehr zur Verfügung.

Um die vorherigen Werte zu erhalten, muss die Variable als `Static` definiert werden:

```
Sub Test
   Static MyInteger As Integer

   ' ...

End Sub
```

Hinweis – Im Gegensatz zu VBA stellt StarOffice Basic sicher, dass der Name einer lokalen Variablen nicht gleichzeitig für eine globale oder private Variable im Kopfbereich des Moduls verwendet wird. Wenn Sie eine VBA-Anwendung nach StarOffice Basic portieren, müssen Sie alle doppelten Variablennamen ändern.

Öffentliche Variablen

Öffentliche Variablen werden im Kopfbereich eines Moduls mit dem Schlüsselwort Dim definiert. Sie stehen allen Modulen in ihrer Bibliothek zur Verfügung:

Modul A:

```
Dim A As Integer

Sub Test
  Flip
  Flop
End Sub

Sub Flip
    A = A + 1
    End Sub
```

Modul B:

```
Sub Flop
   A = A - 1
End Sub
```

Der Wert der Variablen `A` wird von der Funktion `Test` nicht geändert, aber in der Funktion `Flip` um eins erhöht sowie in der Funktion `Flop` um eins verringert. Beide dieser Änderungen an der Variable sind global.

Sie können auch mit dem Schlüsselwort `Public` statt `Dim` eine öffentliche Variable deklarieren:

```
    Public A As Integer
```

Eine öffentliche Variable steht nur so lange zur Verfügung, wie das verknüpfte Makro ausgeführt wird; danach wird die Variable zurückgesetzt.

Globale Variablen

Ihren Funktionen nach sind globale Variablen den öffentlichen Variablen ähnlich. Allerdings bleiben ihre Werte auch dann noch erhalten, wenn die Ausführung des verknüpften Makros beendet wurde. Die Deklaration von globalen Variablen erfolgt im Kopfbereich eines Moduls mit dem Schlüsselwort `Global`:

```
    Global A As Integer
```

Private Variablen

`Private`-Variablen stehen ausschließlich in dem Modul zur Verfügung, in dem sie definiert wurden. Eine solche Variable wird mit dem Schlüsselwort `Private` definiert:

```
    Private MyInteger As Integer
```

Enthalten mehrere Module eine `Private`-Variable mit demselben Namen, so erzeugt StarOffice Basic für jedes Vorkommen des Namens eine unterschiedliche Variable. Im folgenden Beispiel verfügen beide Module `A` und `B` über eine `Private`-Variable namens `C`. Die Funktion `Test` setzt zuerst die `Private`-Variable in Modul `A` und dann die `Private`-Variable in Modul `B`.

Modul A:

```
Private C As Integer

Sub Test
    SetModuleA      ' Setzt die Variable C aus Modul A
    SetModuleB      ' Setzt die Variable C aus Modul B

    ShowVarA        ' Zeigt die Variable C aus Modul A an. (= 10)
    ShowVarB        ' Zeigt die Variable C aus Modul B an. (= 20)
End Sub

Sub SetModuleA
    A = 10
End Sub

Sub ShowVarA
    MsgBox C        ' Zeigt die Variable C aus Modul A an.
End Sub
```

Modul B:

```
Private C As Integer

Sub SetModuleB
```

```
   A = 20
End Sub

Sub ShowVarB
   MsgBox C        ' Zeigt die Variable C aus Modul B an.
End Sub
```

Konstanten

Die Deklaration einer Konstanten erfolgt in StarOffice Basic über das Schlüsselwort `Const`:

```
Const A = 10
```

Sie können auch den Konstantentyp direkt in der Deklaration angeben:

```
Const B As Double = 10
```

Operatoren

StarOffice Basic versteht die gängigen mathematischen, logischen und vergleichenden Operatoren.

Mathematische Operatoren

Mathematische Operatoren können auf alle Zahlentypen angewendet werden, wobei der Operator + auch zum Verknüpfen von Zeichenfolgen verwendet werden kann.

+	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 (gerundet)
^	Potenzieren von Zahlen
MOD	Modulo-Operation (Berechnung des Rests einer Division)

Logische Operatoren

Logische Operatoren gestatten die Verknüpfung von Elementen gemäß den Regeln der booleschen Algebra. Werden die Operatoren auf Boolean-Werte angewendet, liefert die

Verknüpfung sofort 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 wahr, muss auch zweiter wahr sein)

Vergleichsoperatoren

Vergleichsoperatoren können auf alle elementaren Variablentypen (Zahlen, Datumsangaben, Zeichenfolgen und Boolean-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

---

Hinweis – StarOffice Basic unterstützt nicht den VBA-Vergleichsoperator `Like`.

Verzweigungen

Mit Verzweigungs-Anweisungen können Sie die Ausführung eines Codeblocks davon abhängig machen, dass eine bestimmte Bedingung erfüllt wird.

If...Then...Else

Die gängigste Verzweigungs-Anweisung ist die `If`-Anweisung, die im folgenden Beispiel dargestellt ist:

```
If A > 3 Then
   B = 2
End If
```

Die Zuweisung `B = 2` wird nur dann ausgeführt, wenn der Wert der Variablen `A` größer als drei ist. Eine Abwandlung der `If`-Anweisung ist die `If/Else`-Klausel:

```
If A > 3 Then
   B = 2
Else
   B = 0
End If
```

In diesem Beispiel wird der Variable `B` der Wert 2 zugewiesen, wenn `A` größer als 3 ist, andernfalls wird `B` der Wert 0 zugewiesen.

Um komplexere Anweisungen zu erstellen, kann die `If`-Anweisung geschachtelt werden, zum Beispiel:

```
If A = 0 Then
   B = 0
ElseIf A < 3 Then
   B = 1
Else
   B = 2
End If
```

Wenn die Variable `A` den Wert null hat, wird `B` ebenfalls der Wert 0 zugewiesen. Wenn `A` kleiner als 3 ist (aber ungleich Null), wird `B` der Wert 1 zugewiesen. In allen anderen Fällen (d. h., wenn `A` größer oder gleich 3 ist) wird `B` der Wert 2 zugewiesen.

Select...Case

Die `Select...Case`-Anweisung stellt eine Alternative zur geschachtelten `If`-Anweisung dar und wird eingesetzt, wenn ein Wert gegen verschiedene Bedingungen abgeprüft werden soll:

```
Select Case DayOfWeek

Case 1:
   NameOfWeekday = "Sonntag"
Case 2:
   NameOfWeekday = "Montag"
Case 3:
   NameOfWeekday = "Dienstag"
Case 4:
```

```
   NameOfWeekday = "Mittwoch"
Case 5:
   NameOfWeekday = "Donnerstag"
Case 6:
   NameOfWeekday = "Freitag"
Case 7:
   NameOfWeekday = "Samstag"
End Select
```

In diesem Beispiel sind den Wochentagen Nummern zugeordnet, so dass der Variable `DayOfWeek` für `Sonntag` der Wert 1 zugewiesen wird, für `Montag` der Wert 2 usw.

Der Befehl `Select` ist nicht auf einfache 1:1-Zuordnungen beschränkt. Sie können auch Vergleichsoperatoren oder Listen von Ausdrücken in einer `Case`-Verzweigung angeben. Das folgende Beispiel listet die wichtigsten Syntax-Varianten auf:

```
Select Case Var

Case 1 To 5

   ' ... Var liegt zwischen den Zahlen 1 und 5

Case 6, 7, 8

   ' ... Var ist 6, 7 oder 8

Case Var > 8 And Var < 11

   ' ... Var ist größer als 8 und kleiner als 11

Case Else

   ' ... alle anderen Fälle


End Select
```

Schleifen

Eine Schleife führt einen Codeblock für eine angegebene Anzahl von Wiederholungen aus. Die Anzahl der Durchläufe kann bei Schleifen auch unspezifiziert sein.

For...Next

Die `For...Next`-Schleife verfügt über eine festgelegte Anzahl von Durchläufen. Der Schleifenzähler definiert die Anzahl der Ausführungen der Schleife. Im folgenden Beispiel

```
Dim I
For I = 1 To 10
   ' ...  Innenteil der Schleife
Next I
```

ist Variable `I` der Schleifenzähler mit einem Startwert von 1. Der Zähler wird am Ende jedes Durchlaufs der Schleife um 1 erhöht. Wenn die Variable I gleich 10 ist, endet die Schleife.

Wenn der Schleifenzähler am Ende jedes Durchlaufs um einen anderen Wert als 1 erhöht werden soll, verwenden Sie die Funktion `Step`:

```
Dim I

For I = 1 To 10 Step 0.5

   ' ... Innenteil der Schleife

Next I
```

In dem vorangehenden Beispiel wird der Zähler am Ende jedes Durchlaufs um 0,5 erhöht und die Schleife wird insgesamt 19-mal ausgeführt.

Sie können auch negative Iterationswerte verwenden:

```
Dim I

For I = 10 To 1 Step -1

   ' ... Innenteil der Schleife

Next I
```

In diesem Beispiel beginnt der Zähler bei 10 und wird am Ende jedes Durchlaufs um 1 verringert, bis der Zähler gleich 1 ist.

Die Anweisung `Exit For` gestattet das vorzeitige Verlassen einer `For`-Schleife. Im folgenden Beispiel wird die Schleife während des fünften Durchlaufs abgebrochen:

```
Dim I

For I = 1 To 10
```

```
   If I = 5 Then
      Exit For
   End If

   ' ... Innenteil der Schleife

Next I
```

Hinweis – Die in VBA vorhandene Schleifenvariante `For Each...Next` wird in StarOffice Basic nicht unterstützt.

Do...Loop

Die `Do...Loop`-Schleife ist nicht an eine festgelegte Anzahl von Durchläufen gebunden. Stattdessen wird die `Do...Loop`-Schleife solange ausgeführt, bis eine bestimmte Bedingung erfüllt ist. Es gibt vier Varianten der `Do...Loop`-Schleife (in den folgenden Beispielen stellt `A > 10` eine beliebige Bedingung dar):

1. Die `Do While...Loop`-Variante

   ```
   Do While A > 10
      ' ... Schleifenkörper
   Loop
   ```

   prüft vor jedem Durchlauf, ob die Bedingung noch erfüllt ist, und führt nur dann den Schleifenrumpf aus.

2. Die `Do Until...Loop`-Variante

   ```
   Do Until A > 10
      ' ... Schleifenkörper
   Loop
   ```

   führt die Schleife aus, bis die Bedingung *nicht mehr* erfüllt ist.

3. Die `Do...Loop While`-Variante

   ```
   Do
      ' ... Schleifenkörper
   Loop While A > 10
   ```

   prüft die Bedingung nur nach dem ersten Schleifendurchlauf und bricht ab, wenn diese *erfüllt* ist.

4. Die `Do...Loop Until`-Variante

```
Do
   ' ... Schleifenkörper
Loop Until A > 10
```

prüft ebenfalls nach dem ersten Durchlauf ihre Bedingung, führt die Schleife jedoch so lange aus, bis die Bedingung *nicht mehr* erfüllt ist.

Wie die `For...Next`-Schleife verfügt die `Do...Loop`-Schleife auch über einen Befehl zum Abbrechen. Mit dem Befehl `Exit Do` kann eine Schleife an jeder Stelle innerhalb der Schleife verlassen werden.

```
Do
   If A = 4 Then
      Exit Do
   End If

   ' ... Schleifenkörper
While A > 10
```

Programmierbeispiel: Sortieren mit verschachtelten Schleifen

Es gibt zahlreiche Methoden, um Schleifen einzusetzen, beispielsweise zum Durchsuchen von Listen, zum Zurückgeben von Werten oder zum Ausführen komplexer mathematischer Aufgaben. Das folgende Beispiel ist ein Algorithmus, der mit Hilfe von Schleifen eine Liste nach Namen sortiert.

```
Sub Sort
   Dim Entry(1 To 10) As String
   Dim Count As Integer
   Dim Count2 As Integer
   Dim Temp As String

   Entry(1) = "Patty"
   Entry(2) = "Kurt"
   Entry(3) = "Thomas"
   Entry(4) = "Michael"
   Entry(5) = "David"
   Entry(6) = "Cathy"
   Entry(7) = "Susie"
   Entry(8) = "Edward"
   Entry(9) = "Christine"
   Entry(10) = "Jerry"

   For Count = 1 To 10
```

```
        For Count2 = Count + 1 To 10
          If Entry(Count) > Entry(Count2) Then
             Temp = Entry(Count)
             Entry(Count) = Entry(Count2)
             Entry(Count2) = Temp
          End If
        Next Count2
    Next Count

    For Count = 1 To 10
       Print Entry(Count)
    Next Count
End Sub
```

Die Werte werden so lange mehrere Male paarweise vertauscht, bis sie schließlich in aufsteigender Reihenfolge sortiert sind. Wie Blasen wandern die Variablen nach und nach an ihre richtige Position. Aus diesem Grund ist dieser Algorithmus auch als *Bubble Sort* (Blasensortierung) bekannt.

Prozeduren und Funktionen

Dreh- und Angelpunkt bei der Strukturierung eines Programms bilden Prozeduren und Funktionen. Sie stellen die Rahmenbedingungen zur Verfügung, um ein komplexes Problem in verschiedene Teilaufgaben untergliedern zu können.

Prozeduren

Eine *Prozedur* führt eine Aktion aus, ohne einen expliziten Wert zurückzugeben. Ihre Syntax lautet

```
Sub Test

   ' ... hier steht der eigentliche Code der Prozedur

End Sub
```

Das Beispiel definiert eine Prozedur namens `Test`, die Code enthält, auf den von jedem Punkt im Programm zugegriffen werden kann. Der Aufruf erfolgt durch Eingabe des Prozedurnamens an der relevanten Stelle im Programm:

```
Test
```

Funktionen

Eine *Funktion* fasst wie eine Prozedur einen auszuführenden Programmblock als eine logische Einheit zusammen, liefert im Gegensatz zu der Prozedur aber einen Rückgabewert.

```
Function Test

   ' ... hier steht der eigentliche Code der Funktion

   Test = 123
End Function
```

Die Zuordnung des Rückgabewerts erfolgt durch eine einfache Zuweisung. Die Zuweisung muss nicht unbedingt am Ende der Funktion stehen, sondern kann an einer beliebigen Stelle innerhalb der Funktion erfolgen.

Die vorangehende Funktion lässt sich innerhalb eines Programms wie folgt aufrufen:

```
Dim A

A = Test
```

Der Code definiert eine Variable `A` und weist ihr das Ergebnis der Funktion `Test` zu.

Innerhalb der Funktion kann der Rückgabewert mehrfach überschrieben werden. Wie bei einer klassischen Variablenzuweisung gibt die Funktion in diesem Beispiel den Wert zurück, der ihr zuletzt zugewiesen wurde.

```
Function Test

   Test = 12

   ' ...

   Test = 123

End Function
```

In diesem Beispiel ist der Rückgabewert der Funktion 123.

Wird eine Zuweisung abgebrochen, gibt die Funktion einen `Null`-Wert zurück (Zahl 0 für numerische Werte, leere Zeichenfolge für Strings).

Der Rückgabewert einer Funktion kann von einem beliebigen Typ sein. Die Deklaration des Typs erfolgt wie eine Variablendeklaration:

```
Function Test As Integer

   ' ... hier steht der eigentliche Code der Funktion

End Function
```

Wird die Angabe eines expliziten Werts abgebrochen, wird dem Rückgabewert der Typ Variant zugewiesen.

Vorzeitiges Abbrechen von Prozeduren und Funktionen

In StarOffice Basic können Sie mit den Befehlen `Exit Sub` und `Exit Function` eine Prozedur oder Funktion vorzeitig abbrechen, beispielsweise zur Fehlerbehandlung. Diese Befehle beenden die Prozedur oder Funktion und bringen das Programm zurück an die Stelle, an der die Prozedur oder Funktion aufgerufen wurde.

Das folgende Beispiel zeigt eine Prozedur, die ihre Ausführung abbricht, wenn die Variable `ErrorOccured` den Wert `True` besitzt.

```
Sub Test
   Dim ErrorOccured As Boolean

   ' ...

   If ErrorOccured Then
      Exit Sub
   End If

   ' ...

End Sub
```

Übergeben von Parametern

Funktionen und Prozeduren können einen oder mehrere Parameter entgegennehmen. Obligatorische Parameter müssen in Klammern nach dem Funktions- oder Prozedurnamen aufgelistet werden. Das Beispiel

```
Sub Test (A As Integer, B As String)
End Sub
```

definiert eine Prozedur, die einen Integer-Wert A und eine Zeichenfolge B als Parameter erwartet.

Parameter werden in StarOffice Basic normalerweise *als Referenz* (by reference) übergeben. Änderungen der Variablen bleiben beim Verlassen der Prozedur oder Funktion erhalten:

```
Sub Test
   Dim A As Integer
   A = 10
   ChangeValue(A)
   ' Der Parameter A hat nun den Wert 20
End Sub
Sub ChangeValue(TheValue As Integer)
    TheValue = 20
End Sub
```

In diesem Beispiel wird der in der Funktion `Test` definierte Wert `A` als Parameter an die Funktion `ChangeValue` übergeben. Der Wert wird dann zu 20 geändert und an `TheValue` übergeben. Dieser Wert bleibt bei Verlassen der Funktion erhalten.

Parameter können auch als *Wert* (by value) übergeben werden, wenn der ursprünglich übergebene Wert von folgenden Änderungen des Parameters unberührt bleiben soll. Um festzulegen, dass ein Parameter als Wert übergeben werden soll, muss der Variablendeklaration im Kopfbereich der Funktion das Schlüsselwort `ByVal` vorangestellt sein.

Tauschen wir im vorangehenden Beispiel die Funktion `ChangeValue` durch die Funktion

```
Sub ChangeValue(ByVal TheValue As Integer)
   TheValue = 20
End Sub
```

aus, bleibt die übergeordnete Variable A von der Änderung unberührt. Nach dem Aufruf der Funktion `ChangeValue` hat die Variable `A` weiterhin den Wert 10.

Hinweis – Die Methode für die Übergabe von Parametern an Prozeduren und Funktionen in StarOffice Basic ist mit der in VBA weitgehend identisch. Standardmäßig werden Parameter als Referenz (by reference) übergeben. Um Parameter als Werte (by value) zu übergeben, verwenden Sie das Schlüsselwort `ByVal`. In VBA können Sie außerdem das Schlüsselwort `ByRef` verwenden, um zu erzwingen, dass ein Parameter als Referenz (by reference) übergeben wird. StarOffice Basic unterstützt dieses Schlüsselwort nicht, weil es sich hierbei bereits um das Standardverfahren in StarOffice Basic handelt.

Optionale Parameter

Funktionen und Prozeduren lassen sich nur dann aufrufen, wenn beim Aufruf sämtliche erforderlichen Parameter übergeben werden.

In StarOffice Basic können Sie Parameter als *optional* definieren, d. h., wenn die entsprechenden Werte in einem Aufruf fehlen, übergibt StarOffice Basic leere Parameter. In dem Beispiel

```
Sub Test(A As Integer, Optional B As Integer)

End Sub
```

ist der Parameter `A` obligatorisch, der Parameter `B` hingegen optional.

Die Funktion `IsMissing` prüft, ob ein Parameter übergeben oder ausgelassen wurde.

```
Sub Test(A As Integer, Optional B As Integer)
   Dim B_Local As Integer

   ' Prüfung, ob Parameter B tatsächlich vorhanden ist
   If Not IsMissing (B) Then
      B_Local = B      ' Parameter B vorhanden
   Else
      B_Local = 0      ' Parameter B fehlt -> Standardwert 0
   End If

   ' ... Start der eigentlichen Funktion

End Sub
```

Das Beispiel testet zunächst, ob der Parameter `B` übergeben wurde, und überträgt exakt diesen Parameter gegebenenfalls in die interne Variable `B_Local`. Fehlt der betreffende Parameter, wird anstelle des übergebenen Parameters ein Standardwert (hier der Wert 0) an `B_Local` übergeben.

Hinweis – Das in VBA vorhandene Schlüsselwort `ParamArray` wird von StarOffice Basic nicht unterstützt.

Rekursion

Rekursion ist jetzt in StarOffice Basic möglich. Unter einer rekursiven Prozedur oder Funktion versteht man, dass diese die Fähigkeit besitzen, sich selbst aufzurufen, bis sie erkennen, dass eine Grundbedingung erfüllt ist. Wenn die Funktion mit der Grundbedingung aufgerufen wird, wird ein Ergebnis zurückgegeben.

Das folgende Beispiel verwendet eine rekursive Funktion zur Berechnung der Fakultät der Zahlen `42`, `-42` und `3.14`:

```
Sub Main
    Msgbox CalculateFactorial(  42 )    ' Zeigt 1,40500611775288E+51 an
    Msgbox CalculateFactorial( -42 )    ' Zeigt "Ungültige Zahl für Fakultät!" an
    Msgbox CalculateFactorial( 3.14 )   ' Zeigt "Ungültige Zahl für Fakultät an!"
End Sub

Function CalculateFactorial( Number )
    If Number < 0 Or Number <> Int( Number ) Then
        CalculateFactorial = "Ungültige Zahl für Fakultät!"
    ElseIf Number = 0 Then
        CalculateFactorial = 1
    Else
        ' Dies ist der rekursive Aufruf:
        CalculateFactorial = Number * CalculateFactorial( Number - 1 )
    Endif
End Function
```

Das Beispiel gibt die Fakultät der Zahl `42` zurück, indem rekursiv die Funktion `CalculateFactorial` aufgerufen wird, bis die Grundbedingung von `0! = 1` erfüllt ist.

Hinweis – Die mögliche Rekursionstiefe ist in Abhängigkeit von der Softwareplattform unterschiedlich. Für Windows liegt die Rekursionstiefe bei 5800. Für Solaris und Linux wird die Größe des Stacks (Stapels) bewertet und dann die Rekursionstiefe berechnet.

Fehlerbehandlung

Eine korrekte Behandlung von Fehlersituationen gehört mit zu den aufwendigsten Aufgaben bei der Programmierung. StarOffice Basic bietet eine ganze Reihe von Hilfsmitteln an, um diese zu vereinfachen.

Die Anweisung "On Error"

Kern einer jeden Fehlerbehandlung ist die Anweisung `On Error`:

```
Sub Test
   On Error Goto ErrorHandler

   ' ... Aufgabe ausführen, bei der ein Fehler passieren kann

Exit Sub

ErrorHandler:

   ' ... individueller Code zur Fehlerbehandlung

End Sub
```

Die Zeile `On Error Goto ErrorHandler` definiert, wie StarOffice Basic im Falle eines Fehlers verfahren soll. Der `Goto ErrorHandler` gewährleistet, dass StarOffice Basic die aktuelle Programmzeile verlässt und dann den `ErrorHandler:`-Code ausführt.

Die Anweisung "Resume"

Der Befehl `Resume Next` führt das Programm nach Ausführung des Fehler-Handler-Codes ab der Zeile weiter aus, die auf die Zeile folgt, in der der Fehler im Programm aufgetreten ist:

```
ErrorHandler:

   ' ... individueller Code zur Fehlerbehandlung

   Resume Next
```

Mit dem Befehl `Resume Proceed` geben Sie eine Sprungmarke zur Fortsetzung des Programms nach der Fehlerbehandlung an:

```
ErrorHandler:

   ' ... individueller Code zur Fehlerbehandlung

   Resume Proceed

Proceed:

   ' ... hier fährt das Programm nach dem Fehler fort
```

Um ein Programm ohne eine Fehlermeldung bei Auftreten eines Fehlers fortzuführen, verwenden Sie folgendes Format:

```
Sub Test
   On Error Resume Next

   ' ... Aufgabe ausführen, bei der ein Fehler auftreten kann

End Sub
```

Verwenden Sie den Befehl `On Error Resume Next` mit großer Vorsicht, da seine Ergebnisse globale Auswirkungen haben. Weitere Informationen finden Sie unter „Tipps für eine strukturierte Fehlerbehandlung" auf Seite 48.

Abfragen von Fehlerinformationen

Bei der Fehlerbehandlung ist es sinnvoll, über eine Beschreibung des Fehlers zu verfügen und zu wissen, an welcher Stelle und aus welchem Grund er aufgetreten ist:

- Die Variable `Err` enthält die Nummer des aufgetretenen Fehlers.
- Die Variable `Error$` enthält eine Beschreibung des Fehlers.
- Die Variable `Erl` enthält die Nummer der Zeile, in der der Fehler aufgetreten ist.

Der Aufruf

```
MsgBox "Error " & Err & ": " & Error$ & " (line : " & Erl & ")"
```

zeigt, wie die Fehlerinformationen in einem Meldungsfenster angezeigt werden können.

Hinweis – Während in VBA die Fehlermeldungen in einem Statistikobjekt namens `Err` zusammengefasst werden, stellt StarOffice Basic die Variablen `Err`, `Error$` und `Erl` zur Verfügung.

Die Statusinformationen sind so lange gültig, bis das Programm auf einen neuen `Resume`- oder `On Error`-Befehl trifft, woraufhin die Informationen zurückgesetzt werden.

Hinweis – In VBA wird der Fehlerstatus nach Auftreten eines Fehlers mit der Methode `Err.Clear` des `Err`-Objekts zurückgesetzt. In StarOffice Basic wird dies mit den Befehlen `On Error` bzw. `Resume` durchgeführt.

Tipps für eine strukturierte Fehlerbehandlung

Sowohl der Definitionsbefehl `On Error` als auch der Rückgabebefehl `Resume` sind Varianten des `Goto`-Konstrukts.

Wenn Sie Ihren Code sauber strukturieren möchten, um die Erzeugung von Fehlern bei der Verwendung dieses Konstrukts zu vermeiden, sollten Sie keine Sprungbefehle benutzen, ohne sie zu überwachen.

Gehen Sie bei der Verwendung des Befehls `On Error Resume Next` mit Sorgfalt vor, da hierdurch alle geöffneten Fehlermeldungen verworfen werden.

Die beste Lösung besteht darin, für die Fehlerbehandlung im Programm nur einen Ansatz zu verwenden. Trennen Sie die Fehlerbehandlung vom eigentlichen Programmcode und springen Sie nicht mehr zurück in den Originalcode, nachdem ein Fehler aufgetreten ist.

Im Folgenden finden Sie ein Beispiel für eine Fehlerbehandlungsprozedur:

```
Sub Example

   ' Fehler-Handler am Anfang der Funktion definieren
   On Error Goto ErrorHandler

   ' ... Hier steht der eigentliche Programmcode

   ' Fehlerbehandlung deaktivieren
   On Error Goto 0

   ' Ende der regulären Programmausführung
Exit Sub

' Startpunkt der Fehlerbehandlung
ErrorHandler:

   ' Prüfen, ob der Fehler erwartet wurde
   If Err = ExpectedErrorNo Then
      ' ... Bearbeitung des Fehlers
   Else
      ' ... Warnhinweis wegen unerwartetem Fehler
   End If

   On Error Goto 0            ' Fehlerbehandlung deaktivieren
End Sub
```

Diese Prozedur beginnt mit der Definition eines Fehler-Handlers, gefolgt von dem eigentlichen Programmcode. Am Ende des Programmcodes wird die Fehlerbehandlung mit dem Aufruf `On Error Goto 0` deaktiviert und die Ausführung der Prozedur mit dem Befehl `Exit Sub` beendet (nicht zu verwechseln mit `End Sub`).

Das Beispiel prüft zuerst, ob die Fehlernummer mit der erwarteten übereinstimmt (wie in der hypothetischen Konstanten `ExpectedErrorNo` gespeichert), und behandelt den Fehler dann entsprechend. Beim Auftreten eines anderen Fehlers gibt das System einen Warnhinweis aus. Die Überprüfung der Fehlernummer ist wichtig, damit unerwartete Fehler erkannt werden können.

Der Aufruf `On Error Goto 0` am Ende des Codes setzt die Statusinformationen für den Fehler (den Fehlercode in der Systemvariablen `Err`) wieder zurück, damit ein später auftretender Fehler eindeutig identifiziert werden kann.

◆ ◆ ◆ KAPITEL 3

Die Laufzeitbibliothek von StarOffice Basic

Die folgenden Abschnitte stellen die zentralen Funktionen der Laufzeitbibliothek vor.

Konvertierungsfunktionen

In vielen Situationen kommt es vor, dass eine Variable eines Typs in eine Variable eines anderen Typs umgewandelt werden muss.

Implizite und explizite Typumwandlungen

Die einfachste Möglichkeit, den Typ einer Variablen zu ändern, besteht in einer Zuweisung.

```
Dim A As String
Dim B As Integer

B = 101
A = B
```

In diesem Beispiel ist Variable `A` vom Typ String (Zeichenfolge) und Variable `B` vom Typ Integer. StarOffice Basic stellt sicher, dass die Variable `B` bei der Zuweisung zu `A` zum Typ String konvertiert wird. Diese Konvertierung ist weit aufwendiger, als es auf den ersten Blick scheint: Die Integer-Variable `B` verbleibt im Arbeitsspeicher in Form einer zwei Byte langen Zahl. `A` ist dagegen vom Typ String und der Computer speichert für jedes Zeichen (jede Ziffer) der Zeichenfolge einen ein oder zwei Byte langen Wert. Aus diesem Grund muss also vor dem Kopieren des Inhalts von `B` nach `A` `B` in das interne Format von `A` konvertiert werden.

Basic führt die Typumwandlung im Gegensatz zu den meisten anderen Programmiersprachen automatisch durch. Dies kann jedoch mitunter folgenschwere Konsequenzen haben. Die zunächst eindeutig scheinende Code-Sequenz

```
Dim A As String
Dim B As Integer
Dim C As Integer

B = 1
C = 1
A = B + C
```

entpuppt sich bei genauerem Hinsehen als Falle. Der Basic-Interpreter berechnet zuerst das Ergebnis der Addition und wandelt es dann in eine Zeichenfolge um, was als Ergebnis die Zeichenfolge 2 ergibt.

Wandelt der Basic-Interpreter hingegen zunächst die Ausgangswerte `B` und `C` in eine Zeichenfolge um und wendet auf diese den Plus-Operator an, so ergibt sich als Ergebnis die Zeichenfolge `11`.

Dasselbe gilt bei der Verwendung von Variant-Variablen:

```
Dim A
Dim B
Dim C

B = 1
C = "1"
A = B + C
```

Da Variant-Variablen sowohl Zahlen als auch Zeichenfolgen enthalten können, bleibt unklar, ob der Variable `A` die Zahl 2 oder die Zeichenfolge 11 zugewiesen wird.

Die genannten Fehlerquellen durch implizite Typumwandlung lassen sich nur durch disziplinierte Programmierung vermeiden, indem beispielsweise auf die Verwendung des Datentyps Variant verzichtet wird.

Um weitere Zweifelsfälle durch implizite Typumwandlung zu vermeiden, bietet StarOffice Basic eine Reihe von Konvertierungsfunktionen an, mit denen Sie definieren können, wann der Datentyp einer Operation umgewandelt werden sollte:

- **CStr(Var)**: konvertiert jeden Datentyp in den Typ String (Zeichenfolge).
- **CInt(Var)**: konvertiert jeden Datentyp in einen Integer-Wert.
- **CLng(Var)**: konvertiert jeden Datentyp in einen Long-Wert.
- **CSng(Var)**: konvertiert jeden Datentyp in einen Single-Wert.
- **CDbl(Var)**: konvertiert jeden Datentyp in einen Double-Wert.
- **CBool(Var)**: konvertiert jeden Datentyp in einen Boolean-Wert.
- **CDate(Var)**: konvertiert jeden Datentyp in einen Date-Wert.

Mit diesen Konvertierungsfunktionen können Sie definieren, wie StarOffice Basic diese Typumwandlungen durchführen soll:

```
Dim A As String
Dim B As Integer
Dim C As Integer

B = 1
C = 1

A = CStr(B + C)        ' B und C werden zuerst addiert und dann
                       ' konvertiert (ergibt die Zahl 2)
A = CStr(B) + CStr(C)  ' B und C werden in eine Zeichenfolge konvertiert
                       ' und dann verkettet (ergibt String "11")
```

Bei der ersten Addition des Beispiels addiert StarOffice Basic zunächst die Integer-Variablen und wandelt anschließend das Ergebnis in eine Zeichenfolge um. `A` wird somit die Zeichenfolge `2` zugewiesen. Im zweiten Fall hingegen werden die Integer-Variablen zuerst in zwei Zeichenfolgen konvertiert und dann über die Zuweisung miteinander verkettet. `A` wird somit die Zeichenfolge `11` zugewiesen.

Die numerischen Konvertierungsfunktionen `CSng` und `CDbl` akzeptieren auch Dezimalzahlen. Als Dezimaltrennzeichen muss dafür das in den entsprechenden Ländereinstellungen definierte Zeichen verwendet werden. Umgekehrt verwenden die Methoden `CStr` beim Formatieren von Zahlen, Datums- und Zeitangaben die aktuell ausgewählten Ländereinstellungen.

Die Funktion `Val` unterscheidet sich von den Methoden `Csng`, `Cdbl` und `Cstr`. Sie wandelt eine Zeichenfolge in eine Zahl um, erwartet darin jedoch immer einen Punkt als Dezimaltrennzeichen.

```
Dim A As String
Dim B As Double

A = "2.22"
B = Val(A)     ' Wird unabhängig von den Ländereinstellungen
               ' korrekt umgewandelt
```

Prüfen des Inhalts von Variablen

In einigen Fällen ist eine Umwandlung der Daten nicht möglich:

```
Dim A As String
Dim B As Date

A = "test"
B = A            ' Erzeugt Fehlermeldung
```

Das Zuweisen der Zeichenfolge `test` zu einer Date-Variable ergibt in dem gezeigten Beispiel keinen Sinn, so dass der Basic-Interpreter einen Fehler meldet. Dasselbe gilt für den Versuch, eine Zeichenfolge einer Boolean-Variable zuzuweisen:

```
Dim A As String
Dim B As Boolean

A = "test"
B = A            ' Erzeugt Fehlermeldung
```

Auch hier meldet der Basic-Interpreter einen Fehler.

Vermeiden lassen sich solche Fehlermeldungen, indem das Programm vor einer Zuweisung prüft, ob der Inhalt der zuzuweisenden Variablen zum Typ der Zielvariablen passt. Hierfür stellt StarOffice Basic folgende Prüffunktionen zur Verfügung:

- **IsNumeric(Value)**: prüft, ob ein Wert eine Zahl ist.
- **IsDate(Value)**: prüft, ob ein Wert ein Datum ist.
- **IsArray(Value)**: prüft, ob ein Wert ein Array ist.

Diese Funktionen sind insbesondere bei der Abfrage von Benutzereingaben sehr nützlich. So lässt sich damit beispielsweise prüfen, ob ein Anwender tatsächlich eine gültige Zahl beziehungsweise ein gültiges Datum eingegeben hat.

```
If IsNumeric(UserInput) Then
   ValidInput = UserInput
Else
   ValidInput = 0
   MsgBox "Fehlermeldung."
End If
```

Enthält die Variable `UserInput` im Beispiel einen gültigen numerischen Wert, wird dieser der Variable `ValidInput` zugewiesen. Enthält `UserInput` keine gültige Zahl, wird `ValidInput` der Wert `0` zugewiesen und eine Fehlermeldung zurückgegeben.

Während zur Überprüfung von Zahlen, Datumsangaben und Arrays in StarOffice Basic die oben genannten Prüffunktionen existieren, ist eine entsprechende Funktion für Boolean-Werte nicht vorhanden. Die Funktionalität kann jedoch zumindest mit Hilfe der Funktion `IsBoolean` nachgebildet werden:

```
Function IsBoolean(Value As Variant) As Boolean
   On Error Goto ErrorIsBoolean:
   Dim Dummy As Boolean

   Dummy = Value

   IsBoolean = True
   On Error Goto 0
```

```
Exit Sub

ErrorIsBoolean:
   IsBoolean = False
   On Error Goto 0
End Function
```

Die Funktion `IsBoolean` definiert eine interne Hilfsvariable `Dummy` vom Typ Boolean und versucht, dieser den übergebenen Wert zuzuweisen. Ist die Zuweisung erfolgreich, gibt die Funktion den Wert `True` zurück. Schlägt sie fehl, wird ein Laufzeitfehler erzeugt, den die Prüffunktion abfängt, um einen Fehler zurückzugeben.

Hinweis – Enthält eine Zeichenfolge in StarOffice Basic einen nicht numerischen Wert und wird dieser einer Zahl zugewiesen, so erzeugt StarOffice Basic keine Fehlermeldung, sondern übergibt den Wert 0 an die Variable. Dieses Verhalten unterscheidet sich von VBA. Dort wird bei einer entsprechenden Zuweisung ein Fehler ausgelöst und die Programmausführung abgebrochen.

Zeichenfolgen

Arbeiten mit Zeichensätzen

StarOffice Basic verwendet bei der Verwaltung von Zeichenfolgen (Strings) den Unicode-Zeichensatz. Die Funktionen `Asc` und `Chr` erlauben es dabei, den zu einem Zeichen gehörenden Unicode-Wert zu ermitteln beziehungsweise für einen Unicode-Wert das entsprechende Zeichen herauszufinden. Die folgenden Ausdrücke weisen der Code-Variable die verschiedenen Unicode-Werte zu:

```
Code = Asc("A")        ' Lateinischer Buchstabe A (Unicode-Wert 65)
Code = Asc("–")        ' Euro-Zeichen (Unicode-Wert 8364)
Code = Asc("??")       ' Kyrillischer Buchstabe ?? (Unicode-Wert 1083)
```

Umgekehrt stellt der Ausdruck

```
MyString = Chr(13)
```

sicher, dass die Zeichenfolge `MyString` mit dem Wert der Zahl `13` (steht für einen Zeilenumbruch) initialisiert wird.

Der Befehl `Chr` wird in Basic-Sprachen häufig dazu verwendet, Steuerzeichen in eine Zeichenfolge einzufügen. So stellt die Zuweisung

```
MyString = Chr(9) + "Dies ist ein Test." + Chr(13)
```

sicher, dass dem Text ein Tabulatorzeichen (Unicode-Wert 9) vorangestellt und ein Zeilenumbruchszeichen (Unicode-Wert 13) angehängt wird.

Zugriff auf Teile einer Zeichenfolge

StarOffice Basic bietet vier Funktionen, die Teilzeichenfolgen zurückgeben:

- **Left(MyString, Length)**: gibt die Length ersten Zeichen von `MyString` zurück.
- **Right(MyString, Length)**: gibt die Length ersten Zeichen von `MyString` zurück.
- **Mid(MyString, Start, Length)**: gibt die Length ersten Zeichen von `MyString` ab der Startposition `Start` zurück.
- **Len(MyString)**: gibt die Anzahl der in `MyString` vorhandenen Zeichen zurück.

Hier einige Beispielaufrufe für die genannten Funktionen:

```
Dim MyString As String
Dim MyResult As String
Dim MyLen As Integer

MyString = "Dies ist ein kleiner Test"

MyResult = Left(MyString,5)      ' Gibt die Zeichenfolge "Dies " zurück
MyResult = Right(MyString, 5)    ' Gibt die Zeichenfolge " Test" zurück
MyResult = Mid(MyString, 8, 5)   ' Gibt die Zeichenfolge "t ein" zurück
MyLen = Len(MyString)            ' Gibt den Wert 25 zurück
```

Suchen und Ersetzen

Für das Suchen einer Teilzeichenfolge innerhalb einer anderen Zeichenfolge bietet StarOffice Basic die Funktion `InStr`:

```
ResultString = InStr (SearchString, MyString)
```

Der Parameter `SearchString` gibt die in `MyString` zu suchende Zeichenfolge an. Als Rückgabewert liefert die Funktion eine Zahl, die die Position beinhaltet, an der `SearchString` das erste Mal innerhalb von `MyString` vorkommt. Möchte man weitere Treffer der Zeichenfolge finden, bietet die Funktion die Möglichkeit, eine optionale Startposition anzugeben, ab der StarOffice Basic mit der Suche beginnen soll. Die Syntax der Funktion lautet in diesem Fall:

```
ResultString = InStr(StartPosition, MyString, SearchString)
```

In den zuvor genannten Beispielen ignoriert `InStr` die Groß- und Kleinschreibung. Um die Suche so abzuwandeln, dass `InStr` Groß- und Kleinschreibung unterscheidet, fügen Sie den Parameter `0` wie im folgenden Beispiel gezeigt hinzu:

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

Mit den vorstehenden Funktionen zur Bearbeitung von Zeichenfolgen können Programmierer Teile einer Zeichenfolge in einer anderen Zeichenfolge suchen und ersetzen:

```
Function Replace(Source As String, Search As String, NewPart As String)
 Dim Result As String
 Dim StartPos As Long
 Dim CurrentPos As Long

 Result = ""
 StartPos = 1
 CurrentPos = 1

 If Search = "" Then
       Result = Source
   Else
       Do While CurrentPos <> 0
           CurrentPos = InStr(StartPos, Search, Source)
           If CurrentPos <> 0 Then
               Result = Result + Mid(Source, StartPos, _
                   CurrentPos - StartPos)
               Result = Result + NewPart
               StartPos = CurrentPos + Len(Search)
           Else
               Result = Result + Mid(Source, StartPos, Len(Source))
           End If ' Position <> 0
       Loop
   End If
   Replace = Result
End Function
```

Die Funktion sucht die übergebene Zeichenfolge `Search` in einer Schleife mittels `InStr` in dem Ausgangsausdruck `Source`. Findet sie den gewünschten Suchausdruck, nimmt sie den vor dem Ausdruck stehenden Teil und schreibt ihn in den Rückgabepuffer `Result`. Diesem fügt sie an der Stelle des Suchausdrucks `Search` den neuen Teil `Part` hinzu. Finden sich keine Treffer mehr für den Suchausdruck, ermittelt die Funktion den übrig gebliebenen Teil der Zeichenfolge und fügt diesen ebenfalls an den Rückgabepuffer an. Die so erzeugte Zeichenfolge wird als Ergebnis des Ersetzungsvorgangs zurückgegeben.

Da das Ersetzen von Teilzeichenfolgen mit zu den am häufigsten benötigten Funktionen gehört, wurde der `Mid`-Funktionsumfang in StarOffice Basic so erweitert, dass diese Aufgabe automatisch ausgeführt wird. Das Beispiel

```
Dim MyString As String

MyString  = "Dies war mein Text."
Mid(MyString, 6, 3, "ist")
```

ersetzt in der Zeichenfolge `MyString` ab der sechsten Position drei Zeichen durch die Zeichenfolge `ist`.

Formatieren von Zeichenfolgen

Die Funktion `Format` formatiert Zahlen als Zeichenfolge. Hierzu erwartet die Funktion die Angabe eines `Format`-Ausdrucks, der dann als Vorlage zur Formatierung der Zahlen verwendet wird. Jeder Platzhalter innerhalb der Vorlage sorgt für eine entsprechende Formatierung dieser Position im Ausgabewert. Die fünf wichtigsten Platzhalter innerhalb einer Vorlage sind die Zeichen *Null* (0), *Doppelkreuz* (#), *Punkt* (.), *Komma* (,) und *Dollar* ($).

Die Ziffer *Null* innerhalb der Vorlage sorgt dafür, dass an der betreffenden Stelle stets eine Zahl ausgegeben wird. Ist keine Zahl vorhanden, wird stattdessen 0 angezeigt.

Ein *Punkt* steht für das in den Ländereinstellungen des Betriebssystems definierte Dezimaltrennzeichen.

Das folgende Beispiel zeigt, wie sich mit den Zeichen `Null` und *Punkt* die Nachkommastellen eines Ausdrucks festlegen lassen:

```
MyFormat = "0.00"

MyString = Format(-1579.8, MyFormat)     ' Gibt "-1579,80" zurück
MyString = Format(1579.8, MyFormat)      ' Gibt "1579,80" zurück
MyString = Format(0.4, MyFormat)         ' Gibt "0,40" zurück
MyString = Format(0.434, MyFormat)       ' Gibt "0,43" zurück
```

Analog ist es möglich, eine Zahl mit führenden Nullen aufzufüllen, um eine gewünschte Länge zu erzielen:

```
MyFormat = "0000.00"

MyString = Format(-1579.8, MyFormat)     ' Gibt "-1579,80" zurück
MyString = Format(1579.8, MyFormat)      ' Gibt "1579,80" zurück
MyString = Format(0.4, MyFormat)         ' Gibt "0000,40" zurück
MyString = Format(0.434, MyFormat)       ' Gibt "0000,43" zurück
```

Ein *Komma* stellt das vom Betriebssystem als Tausendertrennzeichen verwendete Zeichen dar und das *Doppelkreuz* eine Ziffer oder Position, die nur dann angezeigt wird, wenn sie für die Eingabezeichenfolge erforderlich ist.

```
MyFormat = "#,##0.00"

MyString = Format(-1579.8, MyFormat)     ' Gibt "-1.579,80" zurück
MyString = Format(1579.8, MyFormat)      ' Gibt "1.579,80" zurück
MyString = Format(0.4, MyFormat)         ' Gibt "0,40" zurück
MyString = Format(0.434, MyFormat)       ' Gibt "0,43" zurück
```

An Stelle des Platzhalters *Dollar* zeigt die Funktion `Format` das im Betriebssystem definierte Währungszeichen an:

```
MyFormat = "#,##0.00 $"

MyString = Format(-1579.8, MyFormat)     ' Gibt "-1.579,80 –" zurück
MyString = Format(1579.8, MyFormat)      ' Gibt "1.579,80 –" zurück
MyString = Format(0.4, MyFormat)         ' Gibt "0,40 –" zurück
MyString = Format(0.434, MyFormat)       ' Gibt "0,43 –" zurück
```

Hinweis – Die in VBA zur Formatierung von Datums- und Zeitangaben verwendeten Format-Anweisungen werden in StarOffice Basic nicht unterstützt.

Datum und Uhrzeit

StarOffice Basic bietet den Datentyp `Date`, der Datums- und Uhrzeitangaben binär speichert.

Angeben von Datums- und Uhrzeitangaben innerhalb des Programmcodes

Sie können einer Date-Variable ein Datum durch Zuweisung einer einfachen Zeichenfolge zuweisen:

```
Dim MyDate As Date

MyDate = "1.1.2002"
```

Diese Zuweisung könnte tatsächlich funktionieren, da StarOffice Basic den als String definierten Datumswert automatisch in eine Date-Variable konvertiert. Diese Art der Zuweisung kann jedoch zu Fehlern führen, da Datums- und Zeitwerte von Land zu Land unterschiedlich definiert und dargestellt werden.

Da StarOffice Basic bei der Konvertierung einer Zeichenfolge in einen Datumswert die Ländereinstellungen des Betriebssystems verwendet, arbeitet der oben stehende Ausdruck nur so lange korrekt, wie die Ländereinstellungen mit dem Zeichenfolgenausdruck übereinstimmen.

Um dieses Problem zu vermeiden, sollte die Funktion `DateSerial` für die Zuordnung eines festen Werts zu einer Date-Variable verwendet werden:

```
Dim MyVar As Date
MyDate = DateSerial (2001, 1, 1)
```

Der Funktionsparameter muss in der Reihenfolge Jahr, Monat, Tag angegeben werden. Die Funktion stellt sicher, dass der Variable unabhängig von den landesspezifischen Einstellungen tatsächlich der richtige Wert zugewiesen wird.

Die Funktion `TimeSerial` formatiert Zeitangaben auf dieselbe Weise wie die Funktion `DateSerial` Datumsangaben formatiert:

```
Dim MyVar As Date

MyDate =  TimeSerial(11, 23, 45)
```

Die Parameter der Funktion sind in der Reihenfolge Stunden, Minuten, Sekunden anzugeben.

Extrahieren von Datums- und Zeitangaben

Das Gegenstück zu den Funktionen `DateSerial` und `TimeSerial` bilden die folgenden Funktionen:

- **Day(MyDate)**: gibt den Tag des Monats von `MyDate` zurück.
- **Month(MyDate)**: gibt den Monat von `MyDate` zurück.
- **Year(MyDate)**: gibt das Jahr von `MyDate` zurück.
- **Weekday(MyDate)**: gibt die Nummer des Wochentags von `MyDate` zurück.
- **Hour(MyTime)**: gibt die Stunden von `MyTime` zurück.
- **Minute(MyTime)**: gibt die Minuten von `MyTime` zurück.
- **Second(MyTime)**: gibt die Sekunden von `MyTime` zurück.

Diese Funktionen extrahieren die Datums- oder Zeitanteile aus einer angegebenen `Date`-Variable. Das Beispiel

```
Dim MyDate As Date

' ... Initialisierung von MyDate

If Year(MyDate) = 2003 Then

   ' ... Angegebenes Datum liegt im Jahr 2003

End If
```

prüft, ob das in `MyDate` gespeicherte Datum im Jahr 2003 liegt. Hierzu analog prüft das Beispiel

```
Dim MyTime As Date

' ... Initialisierung von MyTime

If Hour(MyTime) >= 12 And Hour(MyTime) < 14 Then
```

```
   ' ... Angegebene Zeit liegt zwischen 12 und 14 Uhr

End If
```

ob `MyTime` zwischen 12 und 14 Uhr liegt.

Die Funktion `Weekday` gibt die Nummer des Wochentags für das übergebene Datum zurück:

```
Dim MyDate As Date
Dim MyWeekday As String

' ... MyDate initialisieren

Select Case WeekDay(MyDate)
case 1
   MyWeekday = "Sonntag"
case 2
   MyWeekday = "Montag"
case 3
   MyWeekday = "Dienstag"
case 4
   MyWeekday = "Mittwoch"
case 5
   MyWeekday = "Donnerstag"
case 6
   MyWeekday = "Freitag"
case 7
   MyWeekday = "Samstag"
End Select
```

Hinweis – Sonntag gilt hierbei als erster Tag der Woche.

Abrufen von Systemdatum und -zeit

Zum Abrufen der Systemzeit und des Systemdatums stehen in StarOffice Basic folgende Funktionen zur Verfügung:

- **Date**: gibt das aktuelle Datum zurück.
- **Time**: gibt die aktuelle Uhrzeit zurück.
- **Now**: gibt den aktuellen Zeitpunkt zurück (Kombinationswert aus Datum und Uhrzeit).

Dateien und Verzeichnisse

Das Arbeiten mit Dateien ist eine der grundlegenden Aufgaben bei einer Anwendung. Die StarOffice API bietet Ihnen eine ganze Reihe von Objekten, mit denen Sie Office-Dokumente erstellen, öffnen und ändern können. Diese werden detailliert in Kapitel 4, vorgestellt. Unabhängig davon ist es in einigen Fällen notwendig, direkt auf das Dateisystem zuzugreifen, Verzeichnisse zu durchsuchen oder Textdateien zu bearbeiten. Für diese Aufgaben stellt die Laufzeitbibliothek von StarOffice Basic einige grundlegende Funktionen zur Verfügung.

Hinweis – Mit StarOffice 8 stehen einige DOS-spezifische Datei- und Verzeichnisfunktionen nicht mehr oder nur noch eingeschränkt zur Verfügung. So fehlt beispielsweise die Unterstützung der Funktionen `ChDir`, `ChDrive` und `CurDir`. In Funktionen, die Dateieigenschaften als Parameter erwarten (etwa zur Differenzierung von versteckten Dateien und Systemdateien), werden einige DOS-spezifische Eigenschaften nicht mehr verwendet. Diese Änderung wurde notwendig, um eine möglichst hohe Plattformunabhängigkeit für StarOffice sicherzustellen.

Verwalten von Dateien

Durchsuchen von Verzeichnissen

Das Durchsuchen von Verzeichnissen nach Dateien und Unterverzeichnissen übernimmt in StarOffice Basic die Funktion `Dir`. Beim ersten Aufruf muss `Dir` als erster Parameter eine Zeichenfolge zugewiesen werden, die den zu durchsuchenden Verzeichnispfad enthält. Der zweite Parameter von `Dir` gibt die Datei oder das Verzeichnis an, nach dem gesucht werden soll. StarOffice Basic gibt den Namen des ersten gefundenen Verzeichniseintrags zurück. Um den nächsten Eintrag abzurufen, muss die Funktion `Dir` ohne Parameter aufgerufen werden. Findet die Funktion `Dir` keine weiteren Einträge mehr, liefert sie eine leere Zeichenfolge zurück.

Das folgende Beispiel zeigt, wie mit der Funktion `Dir` alle Dateien abgerufen werden können, die sich in einem Verzeichnis befinden. Die Prozedur speichert die einzelnen Dateinamen in der Variable `AllFiles` und gibt diese im Anschluss in einem Meldungsfenster aus.

```
Sub ShowFiles
   Dim NextFile As String
   Dim AllFiles As String

   AllFiles = ""
   NextFile = Dir("C:\", 0)

   While NextFile  <> ""
      AllFiles = AllFiles & Chr(13) &  NextFile
      NextFile  = Dir
```

```
   Wend

   MsgBox AllFiles
End Sub
```

Die `0` als zweiter Parameter der Funktion `Dir` sorgt dafür, dass `Dir` ausschließlich die Namen von Dateien zurückgibt, Verzeichnisse hingegen ignoriert. Folgende Parameter können angegeben werden:

- `0`: gibt normale Dateien zurück.
- `16`: Unterverzeichnisse.

Das folgende Beispiel entspricht praktisch dem oben stehenden, übergibt der Funktion `Dir` jedoch den Wert 16 als Parameter, so dass diese anstelle der Dateinamen die Unterverzeichnisse eines Ordners zurückgibt.

```
Sub ShowDirs
   Dim NextDir As String
   Dim AllDirs As String
   AllDirs = ""
   NextDir = Dir("C:\", 16)
   While NextDir <> ""
      AllDirs = AllDirs & Chr(13) &  NextDir
      NextDir  = Dir
   Wend
   MsgBox AllDirs
End Sub
```

Hinweis – Wird die Funktion `Dir` in StarOffice Basic mit dem Parameter 16 aufgerufen, werden nur die Unterverzeichnisse eines Ordners zurückgegeben. In VBA gibt die Funktion zusätzlich die Namen der Standarddateien zurück, so dass eine weitere Überprüfung notwendig ist, um nur die Verzeichnisse zu erhalten. Bei Verwendung der Funktion `CompatibilityMode ( true )` verhält sich StarOffice Basic wie VBA und die Dir-Funktion gibt bei Verwendung des Parameters 16 Unterverzeichnisse und Standarddateien zurück.

Hinweis – Die in VBA vorhandenen Möglichkeiten, Verzeichnisse gezielt nach Dateien mit den Eigenschaften *versteckt*, *Systemdatei*, *archiviert* und *Volumenname* zu durchsuchen, existieren in StarOffice Basic nicht, da die entsprechenden Dateisystemfunktionen nicht unter allen Betriebssystemen zur Verfügung stehen.

Hinweis – Die in `Dir` aufgeführten Pfadangaben dürfen sowohl in VBA als auch in StarOffice Basic die Platzhalter * und ? verwenden. In StarOffice Basic darf der Platzhalter * im Gegensatz zu VBA jedoch nur als letztes Zeichen eines Dateinamens und/oder einer Dateinamenserweiterung verwendet werden.

Erstellen und Löschen von Verzeichnissen

Für das Erstellen von Verzeichnissen bietet StarOffice Basic die Funktion `MkDir`.

```
MkDir ("C:\SubDir1")
```

Diese Funktion erstellt Verzeichnisse und Unterverzeichnisse. Alle innerhalb einer Hierarchie benötigten Verzeichnisse werden bei Bedarf ebenfalls angelegt. Existiert beispielsweise lediglich das Verzeichnis `C:\SubDir1`, so erstellt ein Aufruf

```
MkDir ("C:\SubDir1\SubDir2\SubDir3\")
```

sowohl das Verzeichnis `C:\SubDir1\SubDir2` als auch das Verzeichnis `C:\SubDir1\SubDir2\SubDir3`.

Die Funktion `RmDir` löscht Verzeichnisse.

```
RmDir ("C:\SubDir1\SubDir2\SubDir3\")
```

Enthält das Verzeichnis Unterverzeichnisse oder Dateien, so werden diese *ebenfalls gelöscht*. Aus diesem Grund sollten Sie bei der Verwendung von `RmDir` vorsichtig sein.

Hinweis – In VBA beziehen sich die Funktionen `MkDir` und `RmDir` nur auf das aktuelle Verzeichnis. In StarOffice Basic hingegen ist es möglich, mit `MkDir` und `RmDir` mehrere Ebenen von Verzeichnissen zu erstellen beziehungsweise zu löschen.

Hinweis – In VBA erzeugt `RmDir` eine Fehlermeldung, wenn ein Verzeichnis eine Datei enthält. In StarOffice Basic wird das Verzeichnis *samt allen darin enthaltenen Dateien* gelöscht. Bei Verwendung der Funktion `CompatibilityMode ( true )` verhält sich StarOffice Basic wie VBA.

Kopieren, Umbenennen, Löschen und Prüfen der Existenz von Dateien

Der Aufruf

```
FileCopy(Source, Destination)
```

erstellt eine Kopie der Datei `Source` unter dem Namen `Destination`.

Mit Hilfe der Funktion

```
Name OldName As NewName
```

ist es möglich, die Datei `OldName` in `NewName` umzubenennen. Die Syntax mit dem Schlüsselwort `As` und ohne Verwendung eines Kommas geht auf die Wurzeln von Basic zurück.

Der Aufruf

```
Kill(Filename)
```

löscht die Datei `Filename`. Um ein Verzeichnis (mit darin enthaltenen Dateien) zu löschen, verwenden Sie die Funktion `RmDir`.

Ob eine Datei existiert, lässt sich mit der Funktion `FileExists` prüfen:

```
If FileExists(Filename) Then
   MsgBox "Datei existiert."
End If
```

Lesen und Ändern von Dateieigenschaften

Bei der Arbeit mit Dateien ist es mitunter wichtig, die Dateieigenschaften, den Zeitpunkt der letzten Dateiänderung und die Länge der Datei zu ermitteln.

Der Aufruf

```
Dim Attr As Integer
Attr = GetAttr(Filename)
```

gibt einige Eigenschaften einer Datei zurück. Der Rückgabewert wird als Bit-Maske bereitgestellt, wobei folgende Werte möglich sind:

- 1: schreibgeschützte Datei
- 16: Name eines Verzeichnisses

Das Beispiel

```
Dim FileMask As Integer
Dim FileDescription As String

FileMask = GetAttr("test.txt")
If (FileMask AND 1) > 0 Then
   FileDescription = FileDescription & " read-only "
End IF

If (FileMask AND 16) > 0 Then
   FileDescription = FileDescription & " directory "
```

```
End IF

If FileDescription = "" Then
   FileDescription = " normal "
End IF

MsgBox FileDescription
```

ermittelt die Bit-Maske der Datei `test.txt` und prüft, ob diese schreibgeschützt und/oder ein Verzeichnis ist. Trifft keiner dieser Fälle zu, wird der `FileDescription` die Zeichenfolge "normal" zugewiesen.

Hinweis – Die in VBA zum Abfragen der Dateieigenschaften *versteckt*, *Systemdatei*, *archiviert* und *Volumenname* verwendeten Flags werden in StarOffice Basic nicht unterstützt, da diese Windows-spezifisch sind und unter anderen Betriebssystemen nicht oder nur teilweise zur Verfügung stehen.

Die Funktion `SetAttr` gestattet es, die Eigenschaften einer Datei zu ändern. Der Aufruf

```
SetAttr("test.txt", 1)
```

kann deshalb dazu verwendet werden, eine Datei mit einem Schreibschutz zu versehen. Ein bestehender Schreibschutz kann mit dem folgenden Aufruf gelöscht werden.

```
SetAttr("test.txt", 0)
```

Das Datum und die Uhrzeit der letzten Änderung einer Datei liefert die Funktion `FileDateTime`. Das Datum wird dabei gemäß den im System festgelegten Ländereinstellungen formatiert.

```
FileDateTime("test.txt")    ' Gibt Datum und Uhrzeit der letzten Dateiänderung zurück.
```

Die Funktion `FileLen` bestimmt die Länge einer Datei in Byte (als Long Integer-Wert).

```
FileLen("test.txt")         ' Gibt die Länge der Datei in Byte zurück.
```

Schreiben und Lesen von Textdateien

StarOffice Basic bietet eine ganze Reihe von Methoden zum Lesen und Schreiben von Dateien. Die folgenden Erläuterungen beziehen sich auf das Arbeiten mit Textdateien (*nicht* Textdokumenten).

Schreiben von Textdateien

Vor dem Zugriff auf eine Textdatei muss diese zunächst geöffnet werden. Hierzu wird ein freier *Datei-Handle* benötigt, der die Datei bei allen späteren Dateizugriffen eindeutig identifiziert.

Zum Erstellen eines freien Datei-Handles wird die Funktion `FreeFile` verwendet. Der Handle dient als Parameter der `Open`-Anweisung, mit der die Datei geöffnet wird. Soll die Datei so geöffnet werden, dass sie als Textdatei angegeben werden kann, lautet der `Open`-Aufruf:

```
Open Filename For Output As #FileNo
```

`Filename` ist eine Zeichenfolge (String), die den Namen der Datei enthält. `FileNo` ist der mit der Funktion `FreeFile` erstellte Handle.

Ist die Datei geöffnet, lässt sie sich zeilenweise über die Anweisung `Print` beschreiben:

```
Print #FileNo, "Dies ist eine Testzeile."
```

`FileNo` steht auch hier für den Datei-Handle. Der zweite Parameter gibt den Text an, der als Zeile der Textdatei gespeichert werden soll.

Ist der Schreibvorgang abgeschlossen, so muss die Datei über einen `Close`-Aufruf geschlossen werden:

```
    Close #FileNo
```

Auch hier ist der Datei-Handle anzugeben.

Das folgende Beispiel zeigt, wie eine Textdatei geöffnet, beschrieben und geschlossen wird:

```
Dim FileNo As Integer
Dim CurrentLine As String
Dim Filename As String

Filename = "c:\data.txt"      ' Dateinamen definieren
FileNo = Freefile             ' Freien Datei-Handle ermitteln

Open Filename For Output As #FileNo               ' Datei öffnen (Schreibmodus)
Print #FileNo, "Dies ist eine Zeile Text."        ' Zeile speichern
Print #FileNo, "Dies ist eine weitere Zeile Text." ' Zeile speichern
Close #FileNo                ' Datei schließen
```

Lesen von Textdateien

Das Lesen von Textdateien erfolgt analog zum Schreiben der Datei. Die zum Öffnen der Datei verwendete `Open`-Anweisung enthält anstelle des Ausdrucks `For Output` den Ausdruck `For Input` und anstelle des Print-Befehls zum Schreiben der Daten ist die Line Input-Anweisung zum Lesen der Daten zu verwenden.

Schließlich dient beim Auslesen einer Textdatei die Anweisung

```
eof(FileNo)
```

zur Prüfung, ob das Ende der Datei erreicht wurde.

Das folgende Beispiel zeigt, wie eine Textdatei eingelesen werden kann:

```
Dim FileNo As Integer
Dim CurrentLine As String
Dim File As String
Dim Msg as String

' Dateinamen definieren
Filename = "c:\data.txt"

' Freien Datei-Handle ermitteln
FileNo = Freefile

' Datei öffnen (Lesemodus)
Open Filename For Input As FileNo

' Prüfen, ob Dateiende erreicht ist.

Do While not eof(FileNo)
   ' Zeile lesen
   Line Input #FileNo, CurrentLine
   If CurrentLine <>"" then
      Msg = Msg & CurrentLine & Chr(13)
   end if
Loop

' Datei schließen
Close #FileNo

Msgbox Msg
```

Die einzelnen Zeilen werden in einer `Do While`-Schleife abgerufen, in der Variable `Msg` gespeichert und am Ende in einem Meldungsfenster angezeigt.

Meldungs- und Eingabefenster

Für eine einfache Benutzerkommunikation stellt StarOffice Basic die Funktionen `MsgBox` und `InputBox` bereit.

Anzeigen von Meldungen

`MsgBox` zeigt ein grundlegendes Hinweisfenster an, in dem eine oder mehrere Schaltflächen vorhanden sein können. In der einfachsten Variante

```
MsgBox "Dies ist ein Hinweis!"
```

enthält `MsgBox` lediglich Text und die Schaltfläche "OK".

Das Erscheinungsbild des Hinweisfensters kann über einen Parameter geändert werden. Der Parameter bietet die Möglichkeit, zusätzliche Schaltflächen hinzuzufügen, die vorbelegte Schaltfläche festzulegen und ein Hinweissymbol hinzuzufügen. Die Werte für das Auswählen der Schaltflächen lauten:

- 0: Schaltfläche "OK"
- 1: Schaltflächen "OK" und "Abbrechen"
- 2: Schaltflächen "Abbrechen" und "Wiederholen"
- 3: Schaltflächen "Ja", "Nein" und "Abbrechen"
- 4: Schaltflächen "Ja" und "Nein"
- 5: Schaltflächen "Wiederholen" und "Abbrechen"

Um eine Schaltfläche als Standardschaltfläche festzulegen, addieren Sie dem Parameterwert aus der Liste der Schaltflächenauswahlen einen der folgenden Werte hinzu. Um beispielsweise die Schaltflächen "Ja", "Nein" und "Abbrechen" zu erstellen (Wert 3), mit "Abbrechen" als Standard (Wert 512), lautet der Parameterwert 3 + 512 = 515.

- 0: Erste Schaltfläche ist Standardwert.
- 256: Zweite Schaltfläche ist Standardwert.
- 512: Dritte Schaltfläche ist Standardwert.

Schließlich stehen folgende Hinweissymbole zur Verfügung, die sich ebenfalls durch Addition des jeweiligen Parameterwerts anzeigen lassen:

- 16: Stopp-Schild
- 32: Fragezeichen
- 48: Ausrufezeichen
- 64: Tipp-Symbol

Der Aufruf

```
MsgBox "Möchten Sie fortfahren?",  292
```

gibt ein Hinweisfenster mit den Schaltflächen "Ja" und "Nein" (Wert 4) aus, von denen die zweite Schaltfläche (Nein) als Standardwert festgelegt ist (Wert 256) und das zusätzlich ein Fragezeichen enthält (Wert 32), 4+256+32=292.

Enthält ein Hinweisfenster mehrere Schaltflächen, sollte sein Rückgabewert abgefragt werden, um zu ermitteln, welche Schaltfläche geklickt wurde. In diesem Fall stehen folgende Rückgabewerte zur Verfügung:

- 1: OK
- 2: Abbrechen
- 4: Wiederholen
- 5: Ignorieren
- 6: Ja

- 7: Nein

Im obigen Beispiel könnte eine Prüfung des Rückgabewerts wie folgt aussehen:

```
If MsgBox ("Möchten Sie fortfahren?",  292) = 6 Then
   ' Schaltfläche "Ja" geklickt
Else
   ' Schaltfläche "Nein" geklickt
End IF
```

Neben dem Hinweistext und dem Parameter für die Gestaltung des Hinweisfensters gestattet `MsgBox` die zusätzliche Angabe eines dritten Parameters, der den Text für den Fenstertitel festlegt:

```
MsgBox "Möchten Sie fortfahren?",  292, "Fenstertitel"
```

Wird kein Fenstertitel angegeben, lautet dieser "soffice".

Eingabefenster zur Abfrage einfacher Zeichenfolgen

Die Funktion `InputBox` fragt einfache Zeichenfolgen vom Anwender ab. Sie stellt damit eine einfache Alternative zur Konfiguration von Dialogen dar. `InputBox` nimmt drei Standardparameter entgegen:

- einen Hinweistext,
- einen Fenstertitel,
- einen Vorgabewert, der innerhalb des Eingabereichs angezeigt werden kann.

```
InputVal = InputBox("Bitte Wert eingeben:", "Test", "Standardwert")
```

Als Rückgabewert gibt `InputBox` die vom Anwender eingegebene Zeichenfolge zurück.

Sonstige Funktionen

Beep

Mit der `Beep`-Funktion kann über das System ein Klang ausgegeben werden, mit dem der Anwender vor einer fehlerhaften Eingabe gewarnt werden kann. `Beep` wird ohne jeden Parameter aufgerufen:

```
Beep   ' Erzeugt einen Hinweiston
```

Shell

Mit der Shell-Funktion können Sie externe Programme starten.

```
Shell(Pathname, Windowstyle, Param)
```

`Pathname` definiert den Pfad des auszuführenden Programms. `Windowstyle` definiert das Fenster, in dem das Programm gestartet wird. Folgende Werte sind möglich:

- 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 den Fokus zu erhalten.
- 6: Das Programm wird in einem minimierten Fenster gestartet und der Fokus bleibt im aktuellen Fenster.
- 10: Das Programm wird im Vollbildmodus gestartet.

Der dritte Parameter `Param` gestattet die Übergabe von Kommandozeilenparameter an das zu startende Programm.

Wait

Die Funktion `Wait` unterbricht die Programmausführung für eine vorgegebene Zeit. Die Angabe der Wartezeit erfolgt in Millisekunden. Der Befehl

```
Wait 2000
```

sorgt für eine Unterbrechung von 2 Sekunden (2000 Millisekunden).

Environ

Die Funktion `Environ` gibt die Umgebungsvariablen des Betriebssystems zurück. Je nach System und Konfiguration werden darin unterschiedliche Daten gespeichert. Der Aufruf

```
Dim TempDir

TempDir=Environ ("TEMP")
```

bestimmt beispielsweise die Umgebungsvariable für das temporäre Verzeichnis des Betriebssystems.

◆ ◆ ◆ KAPITEL 4

Einführung in die StarOffice API

Die StarOffice API ist eine universelle Programmierschnittstelle für den Zugriff auf StarOffice. Mit der StarOffice API können Sie StarOffice-Dokumente erstellen, öffnen, bearbeiten und ausdrucken. Sie bietet die Möglichkeit, den Funktionsumfang von StarOffice durch eigene Makros zu erweitern, und gestattet die Erstellung eigener Dialogfelder.

Die StarOffice API kann nicht nur mit StarOffice verwendet werden, sondern auch mit anderen Programmiersprachen wie Java und C++. Möglich macht das eine Technik namens *Universal Network Objects* (UNO), die eine Schnittstelle zu verschiedenen Programmiersprachen bereitstellt.

In diesem Kapitel wird beschrieben, wie sich StarOffice mit Hilfe von UNO in StarOffice nutzen lässt. Behandelt werden die prinzipiellen Konzepte von UNO aus der Sicht eines StarOffice Basic-Programmierers. Details zum Umgang mit den verschiedenen Teilbereichen der StarOffice API finden Sie in den folgenden Kapiteln.

Universal Network Objects (UNO)

StarOffice stellt eine Programmierschnittstelle in Form von Universal Network Objects (UNO) zur Verfügung. Es handelt sich dabei um eine objektorientierte Programmierschnittstelle, die StarOffice in verschiedene Objekte unterteilt, die ihrerseits programmgesteuerten Zugriff auf die Office Suite gewähren.

Da StarOffice Basic an sich eine prozedurale Programmiersprache ist, mussten einige Sprachkonstrukte hinzugefügt werden, um die Nutzung von UNO zu ermöglichen.

Um ein Universal Network Object in StarOffice Basic zu verwenden, bedarf es einer Variablendeklaration für das verknüpfte Objekt. Die Deklaration erfolgt über die `Dim`-Anweisung (siehe Kapitel 2). Für die Deklaration einer Objektvariable ist die Typbezeichnung `Object` zu verwenden:

```
Dim Obj As Object
```

Der Aufruf deklariert eine Objektvariable namens `Obj`.

Die erzeugte Objektvariable muss anschließend initialisiert werden, damit sie genutzt werden kann. Dies ist beispielsweise über die Funktion `createUnoService` möglich:

```
Obj = createUnoService("com.sun.star.frame.Desktop")
```

Dieser Aufruf ordnet der Variable `Obj` einen Verweis auf das neu erzeugte Objekt zu. `com.sun.star.frame.Desktop` ist einem Objekttyp ähnlich; in UNO-Terminologie handelt es sich eher um einen *Dienst* (Service) als um einen Typ. Gemäß der UNO-Philosophie bezeichnet man ein Obj als einen Verweis auf ein Objekt, das den Dienst `com.sun.star.frame.Desktop` unterstützt. Der in StarOffice Basic verwendete Begriff Dienst (Service) entspricht somit den in anderen Programmiersprachen verwendeten Begriffen *Typ* und *Klasse*.

Es existiert allerdings ein wesentlicher Unterschied: Ein Universal Network Object kann mehrere Dienste gleichzeitig unterstützen. Einige UNO-Dienste unterstützen wiederum andere Dienste, so dass Ihnen mit einem Objekt automatisch eine ganze Reihe von Diensten zur Verfügung stehen. So kann das vorgenannte Objekt, das auf dem Dienst `com.sun.star.frame.Desktop` basiert, beispielsweise weitere Dienste zum Laden von Dokumenten sowie zum Beenden des Programms enthalten.

Hinweis – Während der Aufbau eines Objekts in VBA über die Klasse festgelegt wird, der es angehört, wird der Aufbau in StarOffice Basic über die Dienste definiert, die es unterstützt. Ein VBA-Objekt ist immer genau einer einzigen Klasse zugeordnet. Ein StarOffice Basic-Objekt kann hingegen mehrere Dienste unterstützen.

Eigenschaften und Methoden

Ein Objekt stellt in StarOffice Basic eine Reihe von Eigenschaften und Methoden zur Verfügung, die über das Objekt aufgerufen werden können.

Eigenschaften

Eigenschaften (Properties) sind wie die Eigenschaften eines Objekts, z. B. `Filename` und `Title` für ein `Document`- Objekt.

Das Setzen von Eigenschaften erfolgt über eine einfache Zuweisung:

```
Document.Title = "StarOffice 8 Basic Programmer's Guide"
Document.Filename = "progman.sxv"
```

Eine Eigenschaft besitzt wie eine gewöhnliche Variable einen Typ der festlegt, welche Werte sie aufnehmen kann. Die vorgenannten Eigenschaft `Filename` und `Title` sind vom Typ String.

Echte Eigenschaften und nachgebildete Eigenschaften

Die meisten Eigenschaften eines Objekts in StarOffice Basic sind als solche in der UNO-Beschreibung des Dienstes definiert. Neben diesen "echten" Eigenschaften gibt es in StarOffice Basic auch Eigenschaften, die auf UNO-Ebene aus zwei Methoden bestehen. Die eine davon dient dazu, den Wert der Eigenschaft abzufragen, die andere, ihn zu setzen (Methoden `get-` und `set`). Die Eigenschaft wurde gewissermaßen aus zwei Methoden nachgebildet. So stellen Zeichenobjekte in UNO beispielsweise die Methoden `getPosition` und `setPosition` zur Verfügung, über die der verknüpfte Eckpunkt ausgelesen und geändert werden kann. Der StarOffice Basic-Programmierer kann auf die Werte über die Eigenschaft `Position` zugreifen. Unabhängig davon stehen zusätzlich die Original-Methoden zur Verfügung (im Beispiel `getPosition` und `setPosition`).

Methoden

Methoden können als Funktionen aufgefasst werden, die sich direkt auf ein Objekt beziehen, das über diese aufgerufen wird. Das vorstehende Objekt `Document` könnte beispielsweise eine Methode `Save` anbieten, die wie folgt aufgerufen werden kann:

```
Document.Save()
```

Methoden können analog zu Funktionen Parameter und Rückgabewerte enthalten. Die Syntax derartiger Methodenaufrufe orientiert sich an der klassischer Funktionen. Der Aufruf

```
Ok = Document.Save(True)
```

gibt für das Dokumentobjekt beim Aufruf der Methode Save auch den Parameter `True` an. Nach Abschluss der Methode speichert `Save` einen Rückgabewert in der Variable `Ok`.

Module, Dienste und Schnittstellen

StarOffice stellt hunderte von Diensten zur Verfügung. Um einen Überblick dieser Dienste zu bieten, wurden sie in Modulen zusammengefasst. Eine weitere funktionale Bedeutung haben die Module für den StarOffice Basic-Programmierer nicht. Lediglich bei der Angabe eines Dienstnamens ist der Modulname von Bedeutung, da er im Namen mit aufgeführt werden muss. Der vollständige Name eines Dienstes besteht aus dem Ausdruck `com.sun.star`, der angibt, dass es sich um einen StarOffice-Dienst handelt, gefolgt von dem Modulnamen, beispielsweise `frame` und schließlich dem eigentlichen Dienstnamen, etwa `Desktop`. Der vollständige Name würde in dem genannten Beispiel lauten:

```
com.sun.star.frame.Desktop
```

Neben den Begriffen Modul und Dienst (Service) führt UNO den Begriff der *Schnittstelle* (Interface) ein. Während dieser Begriff Java-Programmierern vertraut sein dürfte, wird er in Basic nicht verwendet.

Eine Schnittstelle fasst mehrere Methoden zusammen. Streng genommen unterstützt ein Dienst in UNO keine Methoden, sondern Schnittstellen, die wiederum verschiedene Methoden bereitstellen. Die Methoden werden also mit anderen Worten dem Dienst in Schnittstellen zusammengefasst zugeordnet. Diese Feinheit mag insbesondere den Java- oder C++-Programmierer interessieren, da die Schnittstelle in diesen Sprachen benötigt wird, um eine Methode anzufordern. In StarOffice Basic ist dies irrelevant. Hier werden die Methoden direkt über das jeweils relevante Objekt aufgerufen.

Für das Verständnis der API ist es dennoch hilfreich, sich die Zuordnung von Methoden zu verschiedenen Schnittstellen zu vergegenwärtigen, da viele Schnittstellen in den verschiedenen Diensten benutzt werden. Ist man mit einer Schnittstelle vertraut, so kann man sein Wissen von einem Dienst auf einen anderen übertragen.

Einige zentrale Schnittstellen werden so häufig verwendet, dass sie am Ende dieses Kapitels noch einmal mit den Diensten, von denen Sie ausgelöst werden, aufgeführt werden.

Hilfsmittel für den Umgang mit UNO

Bleibt die Frage, welche Objekte – oder Dienste, um bei der UNO-Terminologie zu bleiben – welche Eigenschaften, Methoden und Schnittstellen unterstützen und wie sich diese ermitteln lassen. Zusätzlich zu diesem Handbuch finden Sie weitere Informationen über Objekte in den folgenden Quellen: in der Methode `supportsService`, in den Debug-Methoden sowie im Developer's Guide und der API-Referenz.

Die Methode "supportsService"

Eine Reihe von UNO-Objekten unterstützt die Methode `supportsService`, mit der sich ermitteln lässt, ob ein Objekt einen bestimmten Dienst unterstützt. Der Aufruf

```
Ok = TextElement.supportsService("com.sun.star.text.Paragraph")
```

bestimmt beispielsweise, ob das Objekt `TextElement` den Dienst `com.sun.star.text.Paragraph` unterstützt.

Debug-Eigenschaften

Jedes UNO-Objekt in StarOffice Basic weiß, welche Eigenschaften, Methoden und Schnittstellen es bereits enthält. Es stellt Eigenschaften bereit, die diese in Form einer Liste zurückliefern. Die entsprechenden Eigenschaften lauten:

DBG_properties: gibt eine Zeichenfolge (String) mit allen Eigenschaften eines Objekts zurück.

DBG_methods: gibt eine Zeichenfolge (String) mit allen Methoden eines Objekts zurück.

DBG_supportetInterfaces: gibt eine Zeichenfolge (String) mit allen Schnittstellen zurück, die ein Objekt unterstützt.

Der folgende Programmcode zeigt, wie sich `DBG_properties` und `DBG_methods` in praktischen Anwendungen verwenden lassen. Er erzeugt zunächst den Dienst `com.sun.star.frame.Desktop` und gibt dann die unterstützten Eigenschaften und Methoden in Meldungsfenstern aus.

```
Dim Obj As Object
Obj = createUnoService("com.sun.star.frame.Desktop")

MsgBox Obj.DBG_Propierties
MsgBox Obj.DBG_methods
```

Bei der Verwendung von `DBG_properties` ist zu beachten, dass die Funktion alle Eigenschaften zurückgibt, die ein bestimmter Dienst theoretisch unterstützen kann. Es ist jedoch nicht sichergestellt, dass diese von dem betreffenden Objekt auch verwendet werden können. Vor dem Auslesen einer Eigenschaften muss daher mit der Funktion `IsEmpty` geprüft werden, ob diese tatsächlich verfügbar ist.

API-Referenz

Weitere Informationen über die verfügbaren Dienste mit ihren Schnittstellen, Methoden und Eigenschaften finden Sie in der API-Referenz für die StarOffice API. Diese steht unter `www.openoffice.org` zur Verfügung:

```
http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html
```

Einige zentrale Schnittstellen im Überblick

Einige Schnittstellen von StarOffice finden sich in vielen Bereichen der StarOffice API. Sie definieren Sätze von Methoden für abstrakte Aufgaben, die sich auf verschiedene Probleme anwenden lassen. Hier finden Sie einen Überblick der gängigsten dieser Schnittstellen.

Der Ursprung der Objekte wird erst später in diesem Handbuch erklärt. An dieser Stelle werden nur einige abstrakte Aspekte der Objekte, für die die StarOffice API einige zentrale Schnittstellen zur Verfügung stellt, diskutiert.

Erzeugen kontextabhängiger Objekte

Die StarOffice API bietet zwei Möglichkeiten zum Erzeugen von Objekten. Die eine besteht in der Verwendung der bereits zu Beginn dieses Kapitels genannten Funktion `createUnoService`. `createUnoService` erzeugt ein Objekt, das universell einsetzbar ist. Solche Objekte und Dienste werden auch als *kontextunabhängige Dienste* bezeichnet.

Neben den kontextunabhängigen Diensten gibt es auch *kontextabhängige Dienste*, deren Objekte nur im Zusammenhang mit einem anderen Objekt sinnvoll eingesetzt werden können. So kann ein Zeichnungsobjekt für ein Tabellendokument beispielsweise nur im Zusammenhang mit eben diesem Dokument existieren.

Schnittstelle `com.sun.star.lang.XMultiServiceFactory`

Die Erzeugung kontextabhängiger Objekte erfolgt in der Regel über eine Methode des Objekts, von der dieses abhängig ist. Dabei kommt insbesondere die Methode `createInstance` in den Dokumentobjekten zum Einsatz, die in der Schnittstelle `XMultiServiceFactory` definiert ist.

Besagtes Zeichnungsobjekt lässt sich beispielsweise wie folgt über ein Tabellendokument-Objekt erzeugen:

```
Dim RectangleShape As Object

RectangleShape = _
   Spreadsheet.createInstance("com.sun.star.drawing.RectangleShape")
```

Analog erfolgt die Erzeugung einer Absatzvorlage in einem Textdokument:

```
Dim Style as Object
Style = Textdocument.createInstance("com.sun.star.style.ParagraphStyle")
```

Benannter Zugriff auf untergeordnete Objekte

Die Schnittstellen `XNameAccess` und `XNameContainer` werden in Objekten verwendet, die untergeordnete Objekte enthalten, die über einen Klartextnamen angesprochen werden können.

Während `XNamedAccess` den Zugriff auf die einzelnen Objekte gestattet, übernimmt `XNameContainer` das Einfügen, Ändern und Löschen von Elementen.

Schnittstelle `com.sun.star.container.XNameAccess`

Ein Beispiel für den Einsatz von `XNameAccess` bietet das Sheet-Objekt eines Tabellendokuments. Es fasst sämtliche Seiten innerhalb des Tabellendokuments zusammen. Der Zugriff auf einzelne Seiten erfolgt über die Methode `getByName` der Schnittstelle `XNameAccess`:

```
Dim Sheets As Object
Dim Sheet As Object

Sheets = Spreadsheet.Sheets
Sheet = Sheets.getByName("Sheet1")
```

Einen Überblick der Namen sämtlicher Elemente bietet die Methode `getElementNames`. Als Ergebnis gibt sie ein Datenfeld mit den Namen zurück. Das folgende Beispiel zeigt, wie sich damit alle Elementnamen eines Tabellendokuments ermitteln und in einer Schleife anzeigen lassen:

```
Dim Sheets As Object
Dim SheetNames
Dim I As Integer

Sheets = Spreadsheet.Sheets
SheetNames = Sheets.getElementNames

For I=LBound(SheetNames) To UBound(SheetNames)
   MsgBox SheetNames(I)
Next I
```

Die Methode `hasByName` der Schnittstelle `XNameAccess` zeigt an, ob innerhalb des Basisobjekts ein untergeordnetes Objekt mit einem bestimmten Namen vorhanden ist. Deshalb gibt das folgende Beispiel eine Meldung aus, ob das Objekt `Spreadsheet` eine Seite mit dem Namen `Sheet1` enthält.

```
Dim Sheets As Object

Sheets = Spreadsheet.Sheets
If Sheets.HasByName("Sheet1") Then
   MsgBox " Sheet1 verfügbar"
Else
   MsgBox "Sheet1 nicht verfügbar"
End If
```

Schnittstelle `com.sun.star.container.XNameContainer`

Die Schnittstelle `XNameContainer` übernimmt das Einfügen, Löschen und Ändern von untergeordneten Elementen in einem Basisobjekt. Die hierfür zuständigen Funktionen lauten `insertByName`, `removeByName` und `replaceByName`.

Im Folgenden finden Sie ein praktisches Beispiel hierfür. Es ruft ein Textdokument auf, das ein Objekt `StyleFamilies` enthält und dieses wiederum verwendet, um die Absatzvorlagen (ParagraphStyles) des Dokuments zur Verfügung zu stellen.

```
Dim StyleFamilies As Objects
Dim ParagraphStyles As Objects
Dim NewStyle As Object

StyleFamilies = Textdoc.StyleFamilies
ParagraphStyles = StyleFamilies.getByName("ParagraphStyles")
```

```
ParagraphStyles.insertByName("NewStyle", NewStyle)
ParagraphStyles.replaceByName("ChangingStyle", NewStyle)
ParagraphStyles.removeByName("OldStyle")
```

Die Zeile `insertByName` fügt die Vorlage `NewStyle` unter dem gleichnamigen Namen in das Objekt `ParagraphStyles` ein. Die Zeile `replaceByName` ändert das hinter `ChangingStyle` liegende Objekt in `NewStyle`. Der Aufruf `removeByName` entfernt schließlich das hinter `OldStyle` stehende Objekt aus `ParagraphStyles`.

Indexbasierter Zugriff auf untergeordnete Objekte

Die Schnittstellen `XIndexAccess` und `XIndexContainer` werden in Objekten verwendet, die untergeordnete Objekte enthalten und die über einen Index adressiert werden können.

`XIndexAccess` bietet die Methoden für den Zugriff auf einzelne Objekte an. `XIndexContainer` stellt Methoden zum Einfügen und Entfernen von Elementen bereit.

Schnittstelle `com.sun.star.container.XIndexAccess`

`XIndexAccess` stellt die Methoden für das Auslesen der untergeordneten Objekte `getByIndex` und `getCount` bereit. `getByIndex` stellt ein Objekt mit einem bestimmten Index zur Verfügung. `getCount` gibt die Anzahl der verfügbaren Objekte zurück.

```
Dim Sheets As Object
Dim Sheet As Object
Dim I As Integer

Sheets = Spreadsheet.Sheets

For I = 0 to Sheets.getCount() - 1
   Sheet = Sheets.getByIndex(I)
   ' Sheet bearbeiten
Next I
```

Das Beispiel zeigt eine Schleife, die sämtliche Tabellenelemente (Sheets) nacheinander durchläuft und in der Objektvariable `Sheet` jeweils einen Verweis darauf ablegt. Beim Umgang mit den Indizes ist darauf zu achten, dass `getCount` die Anzahl der Elemente zurückgibt. Die Elemente in `getByIndex` werden jedoch beginnend mit 0 nummeriert. Die Zählvariable der Schleife läuft daher von 0 bis `getCount()-1`.

Schnittstelle `com.sun.star.container.XIndexContainer`

Die Schnittstelle `XIndexContainer` stellt die Funktionen `insertByIndex` und `removeByIndex` zur Verfügung. Der Aufbau der Parameter erfolgt analog zu den entsprechenden Funktionen in `XNameContainer`.

Iterativer Zugriff auf untergeordnete Objekte

In einigen Fällen kann ein Objekt eine Liste von untergeordneten Objekten enthalten, die weder über einen Namen noch über einen Index adressiert werden können. In diesen Situationen ist die Verwendung der Schnittstellen `XEnumeration` sowie `XEnumerationAccess` angebracht. Sie stellen einen Mechanismus zur Verfügung, über den sämtliche untergeordneten Elemente eines Objekts Schritt für Schritt durchlaufen werden können, ohne dass eine direkte Adressierung erfolgen muss.

Schnittstellen `com.sun.star.container.XEnumeration` und `XEnumerationAccess`

Das Basisobjekt muss die Schnittstelle `XEnumerationAccess` bereitstellen, die nur eine Methode `createEnumeration` enthält. Diese gibt ein Hilfsobjekt zurück, das wiederum die Schnittstelle `XEnumeration` mit den Methoden `hasMoreElements` und `nextElement` bereitstellt. Über diese haben Sie Zugriff auf die untergeordneten Objekte.

Das folgende Beispiel geht schrittweise alle Textabsätze durch:

```
Dim ParagraphEnumeration As Object
Dim Paragraph As Object

ParagraphEnumeration = Textdoc.Text.createEnumeration

While ParagraphEnumeration.hasMoreElements()
   Paragraph = ParagraphElements.nextElement()
Wend
```

Das Beispiel erzeugt zuerst ein Hilfsobjekt `ParagraphEnumeration`. Dieses gibt in einer Schleife nach und nach die einzelnen Absätze des Texts zurück. Die Schleife bricht ab, sobald die Methode `hasMoreElements` den Wert `False` zurückgibt und damit signalisiert, dass das Ende des Texts erreicht ist.

◆ ◆ ◆ KAPITEL 5

Arbeiten mit StarOffice-Dokumenten

Die StarOffice API ist so aufgebaut, dass möglichst viele ihrer Teile universell für verschiedene Aufgaben einsetzbar sind. Hierzu zählen beispielsweise die Schnittstellen und Dienste zum Erstellen, Öffnen, Speichern, Konvertieren und Drucken von Dokumenten und für die Vorlagenverwaltung. Da diese Funktionsbereiche in allen Dokumentarten zur Verfügung stehen, werden sie in diesem Kapitel zuerst erklärt.

Der StarDesktop

Beim Umgang mit Dokumenten werden zwei Dienste am häufigsten verwendet:

- Der Dienst `com.sun.star.frame.Desktop`, der als Kerndienst von StarOffice bezeichnet werden kann. Er stellt die Funktionen für das Rahmenobjekt von StarOffice zur Verfügung, unter dem alle Dokumentfenster klassifiziert sind. Auch das Erstellen, Öffnen und Importieren von Dokumenten erfolgt mit diesem Dienst.
- Die Basisfunktionalität für die einzelnen Dokumentobjekte wird von dem Dienst `com.sun.star.document.OfficeDocument` bereitgestellt. Er stellt die Methoden zum Speichern, Exportieren und Drucken von Dokumenten zur Verfügung.

Der Dienst `com.sun.star.frame.Desktop` wird mit dem Start von StarOffice automatisch geöffnet. Hierzu erstellt StarOffice ein Objekt, auf das über den globalen Namen `StarDesktop` zugegriffen werden kann.

Die wichtigste Schnittstelle des `StarDesktop` ist `com.sun.star.frame.XComponentLoader`. Diese erfasst im Wesentlichen die Methode `loadComponentFromURL`, die für das Erstellen, Importieren und Öffnen von Dokumenten zuständig ist.

Der Name des Objekts `StarDesktop` geht auf StarOffice 5 zurück, bei dem alle Dokumentfenster in eine gemeinsame Anwendung eingebettet waren, die `StarDesktop` hieß. In der aktuellen Version von StarOffice gibt es keinen sichtbaren `StarDesktop` mehr. Der Name `StarDesktop` wurde jedoch für das Rahmenobjekt von StarOffice beibehalten, da er deutlich anzeigt, dass es sich um ein Basisobjekt der gesamten Anwendung handelt.

Das Objekt `StarDesktop` tritt die Nachfolge des `Application`-Objekts aus StarOffice 5 an, das bisher als Stammobjekt verwendet wurde. Im Gegensatz zu dem alten `Application`-Objekt ist dieses jedoch primär für das Öffnen neuer Dokumente verantwortlich. Die im alten `Application`-Objekt angesiedelten Funktionen zur Steuerung der Bildschirmdarstellung von StarOffice (z. B. `FullScreen`, `FunctionBarVisible`, `Height`, `Width`, `Top`, `Visible`) werden nicht mehr verwendet.

Hinweis – Während der Zugriff auf das aktive Dokument in Word über `Application.ActiveDocument` und in Excel über `Application.ActiveWorkbook` erfolgt, ist in StarOffice der `StarDesktop` für diese Aufgabe verantwortlich. Der Zugriff auf das aktive Dokumentobjekt erfolgt in StarOffice 7 über die Eigenschaft `StarDesktop.CurrentComponent`.

Grundlegendes zu Dokumenten in StarOffice

Bei der Arbeit mit StarOffice-Dokumenten ist es hilfreich, sich mit einigen grundlegenden Dingen der Dokumentenverwaltung in StarOffice zu beschäftigen. Hierzu zählen auch die Art und Weise, wie Dateinamen für StarOffice-Dokumente aufgebaut sind sowie das Format, in dem die Dateien gespeichert werden.

Dateinamen in URL-Notation

Da StarOffice als plattformunabhängige Anwendung konzipiert wurde, verwendet es für Dateinamen die betriebssystemunabhängige URL-Notation gemäß dem Internet-Standard RFC 1738. Standarddateinamen beginnen gemäß diesem System mit dem Präfix

```
file:///
```

gefolgt von dem lokalen Pfad. Enthält der Dateiname Unterverzeichnisse, werden diese mit *Schrägstrichen* getrennt, nicht mit dem unter Windows üblichen umgekehrten Schrägstrich. Der folgende Pfad verweist auf die Datei `test.sxw` im Verzeichnis "doc" auf Laufwerk "C:".

```
file:///C:/doc/test.sxw
```

Zur Umwandlung von lokalen Dateinamen in einen URL bietet StarOffice die Funktion `ConvertToUrl`. Zur Umwandlung eines URLs in einen lokalen Dateinamen bietet StarOffice die Funktion `ConvertFromUrl`.

```
MsgBox ConvertToUrl("C:\doc\test.sxw")
      ' Liefert file:///C:/doc/test.sxw
MsgBox ConvertFromUrl("file:///C:/doc/test.sxw")
      ' liefert (unter Windows) c:\doc\test.sxw
```

Das Beispiel konvertiert einen lokalen Dateinamen in einen URL und zeigt diesen in einem Meldungsfenster an. Danach konvertiert es einen URL in einen lokalen Dateinamen und zeigt diesen ebenfalls an.

Der zugrunde liegende Internet-Standard RFC 1738 gestattet die Verwendung der Zeichen `0-9`, `a-z` und `A-Z`. Alle anderen Zeichen werden in einer Escape-Kodierung in die URLs eingefügt. Dazu werden sie in ihren Hexadezimalwert im Zeichensatz ISO 8859-1 (ISO-Latin) umgewandelt und erhalten ein Prozentzeichen vorangestellt. Aus einem Leerzeichen in einem lokalen Dateinamen wird so beispielsweise ein `%20` in einem URL.

XML-Dateiformat

Seit Version 6.0 kommt in StarOffice ein XML-basiertes Dateiformat zum Einsatz. Durch den Einsatz von XML hat der Anwender die Möglichkeit, die Dateien auch mit anderen Programmen zu öffnen und zu bearbeiten.

Komprimierung von Dateien

Da XML auf Standardtextdateien basiert, sind die sich ergebenden Dateien normalerweise sehr groß. StarOffice komprimiert die Dateien daher und speichert sie als ZIP-Datei ab. Mit Hilfe einer Option der `storeAsURL`-Methode kann der Anwender die XML-Dateien direkt im Originalformat speichern. Siehe hierzu „Die Optionen der Methode `storeAsURL`" auf Seite 89.

Erstellen, Öffnen und Importieren von Dokumenten

Das Öffnen, Importieren und Erstellen von Dokumenten erfolgt über die Methode

```
StarDesktop.loadComponentFromURL(URL, Frame, _
            SearchFlags, FileProperties)
```

Der erste Parameter von `loadComponentFromURL` gibt den URL der verknüpften Datei an.

Als zweiten Parameter erwartet `loadComponentFromURL` einen Namen für das Rahmenobjekt des Fensters, das StarOffice intern zu dessen Verwaltung erzeugt. Normalerweise wird hier der vordefinierte Name `_blank` angegeben, der dafür sorgt, dass StarOffice ein neues Fenster erstellt. Alternativ kann auch `_hidden` angegeben werden, wodurch sichergestellt wird, dass das entsprechende Dokument zwar geladen wird, aber unsichtbar bleibt.

Mit diesen Parametern ist es bereits möglich, ein StarOffice-Dokument zu öffnen, da den beiden letzten Parametern Platzhalter (Dummy-Werte) zugewiesen werden können:

```
Dim Doc As Object
Dim Url As String
Dim Dummy()

Url = "file:///C:/test.sxw"

Doc = StarDesktop.loadComponentFromURL(Url, "_blank", 0, Dummy())
```

Der genannte Aufruf öffnet die Datei `text.sxw` und zeigt diese in einem neuen Fenster an.

Auf diese Art lassen sich beliebig viele Dokumente in StarOffice Basic öffnen und anschließend über die jeweils zurückgegebenen Dokumentobjekte bearbeiten.

Hinweis – `StarDesktop.loadComponentFromURL` löst die Methoden `Documents.Add` und `Documents.Open` der alten StarOffice API ab.

Ersetzen des Inhalts des Dokumentenfensters

Die benannten Werte `_blank` und `_hidden` für den Parameter `Frame` stellen sicher, dass StarOffice für jeden Aufruf von `loadComponentFromURL` ein neues Fenster erstellt. In einigen Situationen ist es jedoch sinnvoll, den Inhalt eines vorhandenen Fensters zu ersetzen. In diesem Fall sollte das Rahmenobjekt des Fensters einen expliziten Namen haben. Beachten Sie dabei, dass dieser Name nicht mit einem Unterstrich beginnen darf. Des Weiteren muss der Parameter `SearchFlags` so gesetzt sein, dass der betreffende Rahmen erzeugt wird, wenn er noch nicht vorhanden ist. Die entsprechende Konstante für `SearchFlags` lautet:

```
SearchFlags = com.sun.star.frame.FrameSearchFlag.CREATE + _
         com.sun.star.frame.FrameSearchFlag.ALL
```

Das folgende Beispiel zeigt, wie sich mit Hilfe des Frame-Parameters und `SearchFlags` der Inhalt eines geöffneten Fensters ersetzen lässt:

```
Dim Doc As Object
Dim Dummy()
Dim Url As String
Dim SearchFlags As Long

SearchFlags = com.sun.star.frame.FrameSearchFlag.CREATE + _
         com.sun.star.frame.FrameSearchFlag.ALL

Url = "file:///C:/test.sxw"
Doc = StarDesktop.loadComponentFromURL(Url, "MyFrame", _
      SearchFlags, Dummy)

MsgBox "Klicken Sie auf OK, um das zweite Dokument anzuzeigen."

Url = "file:///C:/test2.sxw"
Doc = StarDesktop.loadComponentFromURL(Url, "MyFrame", _
      SearchFlags, Dummy)
```

Das Beispiel öffnet zuerst die Datei `test.sxw` in einem neuen Fenster mit dem Rahmennamen `MyFrame`. Nach dem Bestätigen des Meldungsfensters ersetzt es den Inhalt des Fensters durch die Datei `test2.sxw`.

Die Optionen der Methode `loadComponentFromURL`

Der vierte Parameter der `loadComponentFromURL`-Funktion ist ein `PropertyValue`-Datenfeld, das StarOffice verschiedene Optionen zum Öffnen und Erstellen von Dokumenten zur Verfügung stellt. Für jede Option muss das Datenfeld eine `PropertyValue`-Struktur bereitstellen, in der der Name der Option als Zeichenfolge mit dem zugehörigen Wert gespeichert wird.

`loadComponentFromURL` unterstützt folgende Optionen:

- **AsTemplate (Boolean)**: wenn Wahr, wird ein neues Dokument ohne Titel aus dem angegebenen URL geladen. Wenn Falsch, werden Vorlagendateien zur Bearbeitung geladen.
- **CharacterSet (String)**: definiert, auf welchem Zeichensatz ein Dokument basiert.
- **FilterName (String)**: gibt einen speziellen Filter für die `loadComponentFromURL`-Funktion an. Die verfügbaren Filternamen sind in der Datei `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml` definiert.
- **FilterOptions (String)**: definiert zusätzliche Optionen für Filter.
- **JumpMark (String)**: springt nach dem Öffnen eines Dokuments an die in JumpMark definierte Position.
- **Password (String)**: übergibt ein Passwort für eine geschützte Datei.
- **ReadOnly (Boolean)**: lädt ein schreibgeschütztes Dokument.

Das folgende Beispiel zeigt, wie sich eine kommagetrennte Textdatei in StarOffice Calc unter Verwendung der Option `FilterName` öffnen lässt.

```
Dim Doc As Object
Dim FileProperties(0) As New com.sun.star.beans.PropertyValue
Dim Url As String

Url = "file:///C:/csv.doc"

FileProperties(0).Name = "FilterName"
FileProperties(0).Value ="scalc: Text - txt - csv (StarOffice Calc)"

Doc = StarDesktop.loadComponentFromURL(Url, "_blank", 0, FileProperties())
```

Das Datenfeld `FileProperties` umfasst genau einen Wert, da es eine Option aufnimmt. Die Eigenschaft `Filtername` definiert hierbei, ob StarOffice einen StarOffice Calc-Textfilter zum Öffnen von Dateien verwendet.

Erstellen neuer Dokumente

StarOffice erstellt automatisch ein neues Dokument, wenn es sich bei dem im URL angegebenen Dokument um eine Vorlage handelt.

Wird lediglich ein leeres Dokument ohne jegliche Anpassungen benötigt, kann alternativ ein `private:factory`-URL angegeben werden:

```
Dim Dummy()
Dim Url As String
Dim Doc As Object

Url = "private:factory/swriter"
Doc = StarDesktop.loadComponentFromURL(Url, "_blank", 0, Dummy())
```

Der Aufruf erstellt ein leeres StarOffice Writer-Dokument.

Dokumentobjekte

Die im vorangehenden Absatz eingeführte Funktion `loadComponentFromURL` gibt ein Dokumentobjekt zurück. Dieses unterstützt den Dienst `com.sun.star.document.OfficeDocument`, der wiederum zwei zentrale Schnittstellen bereitstellt:

- die Schnittstelle `com.sun.star.frame.XStorable`, die für das Speichern von Dokumenten zuständig ist und
- die Schnittstelle `com.sun.star.view.XPrintable`, die Methoden zum Ausdrucken von Dokumenten enthält.

Hinweis – Bei der Umstellung auf StarOffice 8 werden Sie feststellen, dass der Funktionsumfang der Dokumentobjekte weitestgehend unverändert geblieben ist. So stellen die Dokumentobjekte beispielsweise weiterhin Methoden zum Speichern und Drucken von Dokumenten zur Verfügung. Die Namen und Parameter der Methoden haben sich jedoch geändert.

Speichern und Exportieren von Dokumenten

Das Speichern von StarOffice-Dokumenten erfolgt direkt über das Dokumentobjekt. Hierzu steht die Methode `store` der Schnittstelle `com.sun.star.frame.XStorable` zur Verfügung:

```
Doc.store()
```

Damit dieser Aufruf funktioniert, muss dem Dokument bereits ein Speicherplatz zugewiesen worden sein. Bei neuen Dokumenten ist dies nicht der Fall. In diesem Fall wird deshalb die Methode `storeAsURL` verwendet. Sie ist ebenfalls in `com.sun.star.frame.XStorable` definiert und mit ihr kann der Speicherort des Dokuments definiert werden:

```
Dim URL As String
Dim Dummy()
```

```
Url = "file:///C:/test3.sxw"

Doc.storeAsURL(URL, Dummy())
```

Neben den vorgenannten Methoden stellt `com.sun.star.frame.XStorable` auch einige Hilfsmethoden zur Verfügung, die beim Speichern von Dokumenten hilfreich sein können. Hierbei handelt es sich um:

- **hasLocation()**: gibt an, ob dem Dokument bereits ein URL zugewiesen wurde.
- **isReadonly()**: gibt an, ob ein Dokument schreibgeschützt ist.
- **isModified()**: gibt an, ob ein Dokument seit dem letzten Speichervorgang geändert wurde.

Mit diesen Optionen lässt sich der Code zum Speichern eines Dokuments erweitern, so dass die Speicherung nur dann erfolgt, wenn das Objekt tatsächlich geändert wurde, und der Dateiname nur dann abgefragt wird, wenn er auch wirklich benötigt wird:

```
If (Doc.isModified) Then
   If (Doc.hasLocation And (Not Doc.isReadOnly)) Then
      Doc.store()
   Else
      Doc.storeAsURL(URL, Dummy())
   End If
End If
```

Das Beispiel prüft zunächst, ob das betreffende Dokument seit dem letzten Speichervorgang geändert wurde. Nur wenn dies der Fall ist, wird mit dem Speichervorgang fortgefahren. Wurde dem Dokument bereits ein URL zugeordnet und ist es nicht schreibgeschützt, wird es unter dem vorhandenen URL gespeichert. Besitzt es keinen URL oder wurde es mit Schreibschutz geöffnet, wird es unter einem neuen URL gespeichert.

Die Optionen der Methode `storeAsURL`

Wie bei der Methode `loadComponentFromURL` lassen sich auch bei der Methode `storeAsURL` einige Optionen in Form eines `PropertyValue`-Datenfelds angeben. Diese bestimmen die Prozedur, die StarOffice zur Speicherung eines Dokuments verwendet. `storeAsURL` bietet folgende Optionen:

- **CharacterSet (String)**: definiert, auf welchem Zeichensatz ein Dokument basiert.
- **FilterName (String)**: gibt einen speziellen Filter für die `loadComponentFromURL`-Funktion an. Die verfügbaren Filternamen sind in der Datei `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml` definiert.
- **FilterOptions (String)**: definiert zusätzliche Optionen für Filter.
- **Overwrite (Boolean)**: lässt das Überschreiben einer bereits vorhandenen Datei ohne Rückfrage zu.
- **Password (String)**: übergibt das Passwort für eine geschützte Datei.

- **Unpacked (Boolean)**: speichert ein Dokument (nicht komprimiert) in Unterverzeichnissen.

Das folgende Beispiel zeigt, wie die Option `Overwrite` zusammen mit `storeAsURL` verwendet werden kann:

```
Dim Doc As Object
Dim FileProperties(0) As New com.sun.star.beans.PropertyValue
Dim Url As String

' ... Doc initialisieren

Url = "file:///c:/test3.sxw"

FileProperties(0).Name = "Overwrite"
FileProperties(0).Value = True

Doc.storeAsURL(sUrl, mFileProperties())
```

Das Beispiel speichert `Doc` auch dann unter dem angegebenen Dateinamen, wenn bereits eine Datei mit diesem Namen vorhanden ist.

Drucken von Dokumenten

Ähnlich wie das Speichern, erfolgt auch das Ausdrucken von Dokumenten direkt über das Dokumentobjekt. Zu diesem Zweck wird die Methode `Print` der Schnittstelle `com.sun.star.view.XPrintable` bereitgestellt. In seiner einfachsten Form lautet der Print-Aufruf:

```
Dim Dummy()

Doc.print(Dummy())
```

Wie bei der Methode `loadComponentFromURL` handelt es sich bei dem Dummy-Parameter um ein `PropertyValue`-Datenfeld, über das StarOffice einige Optionen zum Drucken festlegen kann.

Die Optionen der Methode "print"

Die Methode `print` erwartet ein `PropertyValue`-Datenfeld als Parameter, das die Einstellungen des Druck-Dialogs von StarOffice widerspiegelt:

- **CopyCount (Integer)**: gibt die Anzahl der zu druckenden Kopien an.
- **FileName (String)**: druckt das in der angegebenen Datei enthaltene Dokument.
- **Collate (Boolean)**: weist den Drucker an, die Seiten der Kopien zusammenzutragen.
- **Sort (Boolean)**: sortiert die Seiten beim Drucken mehrerer Kopien (`CopyCount` > 1).

- **Pages (String)**: enthält die Liste der zu druckenden Seiten (Syntax wie im Druck-Dialog angegeben).

Das folgende Beispiel zeigt, wie sich über die Option `Pages` mehrere Seiten eines Dokuments ausdrucken lassen:

```
Dim Doc As Object
Dim PrintProperties(0) As New com.sun.star.beans.PropertyValue

PrintProperties(0).Name="Pages"
PrintProperties(0).Value="1-3; 7; 9"

Doc.print(PrintProperties())
```

Druckerauswahl- und -einstellungen

Die Schnittstelle `com.sun.star.view.XPrintable` stellt die Eigenschaft `Printer` zur Verfügung, mit der der Drucker ausgewählt wird. Diese Eigenschaft nimmt ein `PropertyValue`-Datenfeld mit folgenden Einstellungen entgegen:

- **Name (String)**: gibt den Namen des Druckers an.
- **PaperOrientation (Enum)**: gibt die Seitenausrichtung an (Wert `com.sun.star.view.PaperOrientation.PORTRAIT` für Hochformat, `com.sun.star.view.PaperOrientation.LANDSCAPE` für Querformat).
- **PaperFormat (Enum)**: gibt das Papierformat an (z. B. `com.sun.star.view.PaperFormat.A4` für DIN A4 oder `com.sun.star.view.PaperFormat.Letter` für US-Letter).
- **PaperSize (Size)**: Papierformat in 100stel Millimeter.

Das folgende Beispiel zeigt, wie mit Hilfe der `Printer`-Eigenschaft der Drucker gewechselt und das Papierformat eingestellt werden können.

```
Dim Doc As Object
Dim PrinterProperties(1) As New com.sun.star.beans.PropertyValue
Dim PaperSize As New com.sun.star.awt.Size

PaperSize.Width = 20000      ' Entspricht 20 cm
PaperSize.Height = 20000     ' Entspricht 20 cm

PrinterProperties (0).Name="Name"
PrinterProperties (0).Value="My HP Laserjet"

PrinterProperties (1).Name="PaperSize"
PrinterProperties (1).Value=PaperSize

Doc.Printer = PrinterProperties()
```

Das Beispiel definiert ein Objekt namens `PaperSize` vom Typ `com.sun.star.awt.Size`. Dieses wird zur Angabe des Papierformats benötigt. Des Weiteren erstellt es ein Datenfeld für zwei `PropertyValue`-Einträge namens `PrinterProperties`. Dieses Datenfeld wird dann mit den Werten für die Eigenschaft `Printer` initialisiert. Im Rahmen von UNO ist der Drucker keine echte Eigenschaft, sondern eine nachgebildete.

Vorlagen

Vorlagen sind benannte Listen mit Formatierungsattributen. Sie finden sie in allen Anwendungen von StarOffice, wo sie zu einer deutlichen Vereinfachung des Formatierungsvorgangs beitragen. Ändert der Anwender eins der Attribute einer Vorlage, so passt StarOffice automatisch alle von diesem Attribut abhängigen Dokumentabschnitte an. Dadurch ist es beispielsweise möglich, die Schriftart sämtlicher Überschriften der Ebene eins durch eine zentrale Änderung im Dokument anzupassen. In Abhängigkeit von den jeweiligen Dokumenttypen erkennt StarOffice eine ganze Reihe verschiedener Vorlagenarten.

StarOffice Writer unterstützt

- Zeichenvorlagen,
- Absatzvorlagen,
- Rahmenvorlagen,
- Seitenvorlagen
- Nummerierungsvorlagen

StarOffice Calc unterstützt

- Zellvorlagen
- Seitenvorlagen

StarOffice Impress unterstützt

- Zeichenelement-Vorlagen
- Präsentationsvorlagen

Die verschiedenen Vorlagenarten werden in der StarOffice-Terminologie gemäß dem zugrunde liegenden Dienst `com.sun.star.style.StyleFamily` auch als `StyleFamilies` bezeichnet. Der Zugriff auf die `StyleFamilies` erfolgt über das Dokumentobjekt:

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim CellStyles As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
CellStyles = StyleFamilies.getByName("CellStyles")
```

Das Beispiel ermittelt über die Eigenschaft `StyleFamilies` eines Tabellendokuments eine Liste mit sämtlichen verfügbaren Zellvorlagen.

Der Zugriff auf die einzelnen Vorlagen ist direkt über einen Index möglich:

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim CellStyles As Object
Dim CellStyle As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
CellStyles = StyleFamilies.getByName("CellStyles")

For I = 0 To CellStyles.Count - 1
   CellStyle = CellStyles(I)
   MsgBox CellStyle.Name
Next I
```

Die gegenüber dem vorigen Beispiel hinzugefügte Schleife zeigt die Namen aller Zellvorlagen nacheinander in einem Meldungsfenster an.

Details zu den verschiedenen Formatierungsmöglichkeiten

Jede Vorlagenart stellt eine ganze Reihe individueller Formatierungseigenschaften zur Verfügung. Hier nun ein Überblick zu den wichtigsten Formatierungseigenschaften und den Stellen, an denen diese erklärt werden:

- Zeicheneigenschaften, Kapitel 6, Textdokumente, Dienst `com.sun.star.style.CharacterProperties`
- Absatzeigenschaften, Kapitel 6, Textdokumente, Dienst `com.sun.star.text.Paragraph`
- Zelleigenschaften, Kapitel 7, Tabellendokumente, Dienst `com.sun.star.table.CellProperties`
- Seiteneigenschaften, Kapitel 7, Tabellendokumente, Dienst `com.sun.star.style.PageStyle`
- Zeichenelement-Eigenschaften, Kapitel 7, Tabellendokumente, verschiedene Dienste

Die Formateigenschaften sind normalerweise keineswegs auf die Anwendungen beschränkt, in denen sie erklärt werden, sondern können vielmehr universell eingesetzt werden. So lassen sich die meisten der in Kapitel 7, beschriebenen Seiteneigenschaften beispielsweise nicht nur in StarOffice Calc einsetzen, sondern auch in StarOffice Writer.

Weitere Informationen zum Arbeiten mit Vorlagen finden Sie im Abschnitt *Standardwerte für Zeichen- und Absatzeigenschaften*, in Kapitel 6.

◆ ◆ ◆ KAPITEL 6

Textdokumente

Neben reinen Zeichenfolgen enthalten Textdokumente ebenfalls Formatierungsinformationen. Diese können an jeder beliebigen Stelle des Texts auftreten. Weiter verkompliziert wird der Aufbau durch Tabellen. Hierzu gehören nicht nur eindimensionale Zeichenfolgen, sondern auch zweidimensionale Felder. Schließlich bieten moderne Textverarbeitungsprogramme die Möglichkeit, Zeichnungsobjekte, Textrahmen und andere Objekte in einem Text zu platzieren. Diese können außerhalb des Textflusses stehen und frei auf einer Seite positioniert werden.

Dieses Kapitel stellt die zentralen Schnittstellen und Dienste von Textdokumenten vor. Der erste Abschnitt behandelt den Aufbau von Textdokumenten und konzentriert sich darauf, wie Sie mit einem StarOffice Basic-Programm schrittweise ein StarOffice-Dokument durchgehen können. Im Mittelpunkt stehen dabei Absätze, Absatzteile und deren Formatierung.

Im zweiten Abschnitt steht die effiziente Arbeit mit Textdokumenten im Vordergrund. Hierzu bietet StarOffice über die im ersten Abschnitt genannten Objekte hinaus einige Hilfsobjekte an (etwa das `TextCursor`-Objekt).

Der dritte Abschnitt führt über das Arbeiten mit Texten hinaus. Er setzt den Schwerpunkt auf Tabellen, Textrahmen, Textfelder, Lesezeichen, Inhaltsverzeichnisse und einiges mehr.

Informationen zum Erstellen, Öffnen, Speichern und Drucken von Dokumenten finden Sie in Kapitel 5, da sie nicht nur für Textdokumente, sondern auch für andere Dokumentarten herangezogen werden können.

Der Aufbau von Textdokumenten

Ein Textdokument kann im Wesentlichen vier Arten von Informationen enthalten:

- den eigentlichen Text
- Vorlagen für die Formatierung von Zeichen, Absätzen und Seiten
- nichttextuelle Elemente wie Tabellen, Grafiken und Zeichnungsobjekte
- globale Einstellungen für das Textdokument

In diesem Abschnitt geht es insbesondere um den Text und die zugehörigen Formatierungsmöglichkeiten.

Hinweis – Die API von StarOffice 8 für StarOffice Writer unterscheidet sich konzeptionell von der der Vorgängerversion. In der alten API stand die Arbeit mit dem Selection-Objekt im Vordergrund, das sich stark an der Idee der Benutzeroberfläche für Endanwender orientiert hat, bei der die mausgesteuerte Arbeit mit Markierungen im Mittelpunkt stand.

Die API von StarOffice 8 hat diese Verknüpfung zwischen Benutzeroberfläche und Programmierschnittstelle aufgehoben. Der Programmierer hat somit parallelen Zugriff auf alle Teile einer Anwendung und kann mit Objekten aus verschiedenen Teilbereichen eines Dokuments gleichzeitig arbeiten. Das alte Selection-Objekt steht nicht mehr zur Verfügung.

Absätze und Absatzteile

Den Kern eines Textdokuments bildet eine Abfolge von Absätzen. Diese sind weder benannt noch indiziert, so dass es keine Möglichkeit zum direkten Zugriff auf einzelne Absätze gibt. Die Absätze lassen sich jedoch mit Hilfe des in Kapitel 4, beschriebenen `Enumeration`-Objekts sequenziell durchlaufen. Auf diese Weise können die Absätze bearbeitet werden.

Bei der Arbeit mit dem `Enumeration`-Objekt ist jedoch ein Sonderfall zu beachten: Es gibt nicht nur Absätze, sondern auch Tabellen zurück (streng genommen handelt es sich in StarOffice Writer bei einer Tabelle um einen Sonderfall eines Absatzes). Daher muss vor einem Zugriff auf das zurückgegebene Objekt geprüft werden, ob das zurückgegebene Objekt den Dienst `com.sun.star.text.Paragraph` für Absätze oder den Dienst `com.sun.star.text.TextTable` für Tabellen unterstützt.

Das folgende Beispiel durchläuft den Inhalt eines Textdokuments in einer Schleife und teilt jeweils in einer Meldung mit, ob es sich bei dem fraglichen Objekt um einen Absatz oder eine Tabelle handelt.

```
Dim Doc As Object
Dim Enum As Object
Dim TextElement As Object

' Dokumentobjekt erstellen
Doc = StarDesktop.CurrentComponent

' Enumeration-Objekt erstellen
Enum = Doc.Text.createEnumeration

' Schleife über alle Textelemente
While Enum.hasMoreElements
   TextElement = Enum.nextElement
```

```
   If TextElement.supportsService("com.sun.star.text.TextTable") Then
      MsgBox "Der aktuelle Block enthält eine Tabelle."
   End If

   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
      MsgBox "Der aktuelle Block enthält einen Absatz."
   End If
Wend
```

Das Beispiel erstellt ein Dokumentobjekt `Doc`, das auf das aktuelle StarOffice-Dokument verweist. Mit Hilfe von `Doc` erstellt das Beispiel dann ein `Enumeration`-Objekt, das nacheinander die einzelnen Textteile (Absätze und Tabellen) durchläuft und das jeweils aktuelle Element dem Objekt `TextElement` zuweist. Über die Methode `supportsService` prüft das Beispiel, ob es sich bei `TextElement` um einen Absatz oder eine Tabelle handelt.

Absätze

Der Dienst `com.sun.star.text.Paragraph` gestattet Zugriff auf den Inhalt eines Absatzes. Der innerhalb des Absatzes vorhandene Text kann über die Strings-Eigenschaft abgerufen und geändert werden:

```
Dim Doc As Object
Dim Enum As Object
Dim TextElement As Object

Doc = StarDesktop.CurrentComponent
Enum = Doc.Text.createEnumeration

While Enum.hasMoreElements
   TextElement = Enum.nextElement

   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
      TextElement.String = Replace(TextElement.String, "you", "U")
      TextElement.String = Replace(TextElement.String, "too", "2")
      TextElement.String = Replace(TextElement.String, "for", "4")
   End If
Wend
```

Das Beispiel öffnet das aktuelle Textdokument und durchläuft dieses unter Zuhilfenahme des Enumeration-Objekts. In allen Absätzen greift es über die Eigenschaft `TextElement.String` auf die jeweiligen Absätze zu und ersetzt die Zeichenfolgen `you`, `too` und `for` durch die Zeichen `U`, `2` und `4`. Die für das Ersetzen verwendete Funktion `Replace` gehört nicht zum Standardsprachumfang von StarOffice Basic. Es handelt sich hierbei um einen Fall der in Kapitel 3, im Abschnitt „Suchen und Ersetzen” auf Seite 56 beschriebenen Beispielfunktion.

Hinweis – Das hier beschriebene Verfahren für den Zugriff auf die Absätze eines Texts ist inhaltlich vergleichbar mit der in VBA verwendeten Paragraphs-Auflistung, die in den dort bereitgestellten Objekten `Range` und `Document` verfügbar ist. Während in VBA der Zugriff auf die Absätze über ihre Nummer erfolgt (z. B. über den Aufruf `Paragraph(1)`), ist in StarOffice Basic das oben beschriebene `Enumeration`-Objekt zu verwenden.

Für die in VBA vorhandenen Auflistungen `Characters`, `Sentences` und `Words` ist in StarOffice Basic kein direktes Gegenstück vorhanden. Es besteht jedoch die Möglichkeit, zu einem `TextCursor` zu wechseln, der eine Navigation auf der Ebene von Zeichen, Sätzen und Wörtern gestattet (siehe Abschnitt *Der TextCursor*).

Absatzteile

Das oben stehende Beispiel ist zwar in der Lage, den Text wie gewünscht zu ändern, kann dabei jedoch teilweise die Formatierung zerstören.

Dies liegt daran, dass sich ein Absatz wiederum aus einzelnen Unterobjekten zusammensetzt. Jedes dieser Unterobjekte enthält seine eigenen Formatierungsinformationen. Enthält ein Absatz beispielsweise in der Mitte ein fett gedrucktes Wort, so wird er in StarOffice durch drei Absatzteile dargestellt: den Teil vor dem Fettdruck, dann das fettgedruckte Wort und schließlich den Teil nach dem Fettdruck, der wieder normal dargestellt wird.

Wird der Text des Absatzes nun über die `String`-Eigenschaft des Absatzes geändert, so löscht StarOffice zunächst die alten Absatzteile und fügt dann einen neuen Absatzteil ein. Dabei gehen die Formatierungen der vorherigen Teile zwangsläufig verloren.

Um diesen Effekt zu vermeiden, ist es möglich, statt auf den gesamten Absatz nur auf die zugehörigen Absatzteile zuzugreifen. Dazu bieten Absätze ein eigenes `Enumeration`-Objekt. Das folgende Beispiel zeigt eine doppelte Schleife, die sämtliche Absätze und die darin enthaltenen Absatzteile eines Textdokuments durchläuft und die Ersetzungsvorgänge aus dem vorstehenden Beispiel anwendet:

```
Dim Doc As Object
Dim Enum1 As Object
Dim Enum2 As Object
Dim TextElement As Object
Dim TextPortion As Object

Doc = StarDesktop.CurrentComponent
Enum1 = Doc.Text.createEnumeration

' Schleife über alle Absätze
While Enum1.hasMoreElements
   TextElement = Enum1.nextElement
   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
```

```
      Enum2 = TextElement.createEnumeration

      ' Schleife über alle Teilabsätze
      While Enum2.hasMoreElements
         TextPortion = Enum2.nextElement
         MsgBox "'" & TextPortion.String & "'"
         TextPortion.String = Replace(TextPortion.String, "you", "U")
         TextPortion.String = Replace(TextPortion.String, "too", "2")
         TextPortion.String = Replace(TextPortion.String, "for", "4")

      Wend

   End If

Wend
```

Das Beispiel durchläuft ein Textdokument in einer doppelten Schleife. Die äußere Schleife bezieht sich auf die Absätze des Texts. Die innere Schleife bearbeitet die darin enthaltenen Absatzteile. In jedem dieser Absatzteile ändert der Beispielcode den Inhalt über die `String`-Eigenschaft der Zeichenfolge analog zu dem vorstehenden Beispiel für Absätze. Da jedoch die Absatzteile direkt bearbeitet werden, bleiben ihre Formatierungsinformationen beim Ersetzen der Zeichenfolge erhalten.

Formatieren

Es gibt verschiedene Methoden, um einen Text zu formatieren. Der einfachste Weg besteht darin, der betreffenden Textsequenz die gewünschten Formateigenschaften direkt zuzuweisen. Diese Vorgehensweise wird als direkte Formatierung bezeichnet. Direkte Formatierung wird insbesondere bei kurzen Dokumenten angewendet, weil die Formatierungen vom Anwender per Maus zugewiesen werden können. So können Sie beispielsweise ein bestimmtes Wort innerhalb eines Texts mit Fettdruck hervorheben oder eine Zeile zentrieren.

Neben der direkten Formatierung ist eine Formatierung von Text mit Hilfe von Vorlagen möglich. Diese Vorgehensweise wird als indirekte Formatierung bezeichnet. Bei der indirekten Formatierung weist der Anwender dem entsprechenden Textteil eine vordefinierte Vorlage zu. Wird das Layout des Texts dann zu einem späteren Zeitpunkt geändert, muss der Anwender nur die Vorlage ändern. StarOffice ändert dann die Darstellung sämtlicher Textstellen, die diese Vorlage verwenden.

Hinweis – In VBA sind die Formatierungseigenschaften eines Objekts normalerweise auf eine Reihe von Unterobjekten verteilt (z. B. `Range.Font`, `Range.Borders`, `Range.Shading`, `Range.ParagraphFormat`). Der Zugriff auf die Eigenschaften erfolgt über geschachtelte Ausdrücke (z. B. `Range.Font.AllCaps`). In StarOffice Basic stehen die Formatierungseigenschaften hingegen durch Einsatz des jeweiligen Objekts (`TextCursor`, `Paragraph` usw.) direkt zur Verfügung. Einen Überblick zu den in StarOffice verfügbaren Zeichen- und Absatzeigenschaften finden Sie in den folgenden beiden Abschnitten.

Hinweis – In der alten StarOffice API erfolgte die Formatierung eines Texts im Wesentlichen über das `Selection`-Objekt und dessen untergeordnete Objekte (z. B. `Selection.Font`, `Selection.Paragraph` und `Selection.Border`). In der neuen API stehen die Formatierungseigenschaften in jedem Objekt (`Paragraph`, `TextCursor` usw.) zur Verfügung und können direkt angewendet werden. Eine Auflistung der verfügbaren Zeichen- und Absatzeigenschaften finden Sie in den folgenden Absätzen.

Zeicheneigenschaften

Als Zeicheneigenschaften werden alle Formateigenschaften bezeichnet, die sich auf einzelne Zeichen beziehen. Hierzu zählen der Fettdruck und die Schriftart. Objekte, die das Setzen von Zeicheneigenschaften zulassen, müssen den Dienst `com.sun.star.style.CharacterProperties` unterstützen. StarOffice erkennt eine ganze Reihe von Diensten, die diesen Dienst unterstützen. Hierzu zählen unter anderem die zuvor beschriebenen Dienste `com.sun.star.text.Paragraph` für Absätze sowie `com.sun.star.text.TextPortion` für Absatzteile.

Der Dienst `com.sun.star.style.CharacterProperties` stellt keinerlei Schnittstellen zur Verfügung, bietet dafür allerdings eine Reihe von Eigenschaften, über die die Zeicheneigenschaften festgelegt und abgerufen werden können. Eine vollständige Liste sämtlicher Zeicheneigenschaften finden Sie in der StarOffice API-Referenz. In der folgenden Liste werden die wichtigsten Eigenschaften beschrieben:

- **CharFontName (String)**: Name der gewählten Schriftart.
- **CharColor (Long)**: Textfarbe.
- **CharHeight (Float)**: Zeichenhöhe in Punkt (p).
- **CharUnderline (Constant group)**: Art der Unterstreichung (Konstanten gemäß `com.sun.star.awt.FontUnderline`).
- **CharWeight (Constant group)**: Schriftstärke (Konstanten gemäß `com.sun.star.awt.FontWeight`).
- **CharBackColor (Long)**: Hintergrundfarbe.
- **CharKeepTogether (Boolean)**: Unterdrückung des automatischen Zeilenumbruchs.

- **CharStyleName (String)**: Name der Zeichenvorlage.

Absatzeigenschaften

Als Absatzeigenschaften gelten alle Formatierungsinformationen, die sich nicht auf einzelne Zeichen, sondern auf den gesamten Absatz beziehen, Hierzu gehören unter anderem der Abstand des Absatzes zum Seitenrand sowie der Zeilenabstand. Verfügbar sind die Absatzeigenschaften über den Dienst `com.sun.star.style.ParagraphProperties`.

Auch die Absatzeigenschaften stehen in verschiedenen Objekten zur Verfügung. Alle Objekte, die den Dienst `com.sun.star.text.Paragraph` unterstützen, bieten auch Unterstützung für die Absatzeigenschaften in `com.sun.star.style.ParagraphProperties`.

Eine vollständige Liste sämtlicher Absatzeigenschaften finden Sie in der StarOffice API-Referenz. Die gängigsten Absatzeigenschaften lauten:

- **ParaAdjust (Enum)**: vertikale Textausrichtung (Konstanten gemäß `com.sun.star.style.ParagraphAdjust`).
- **ParaLineSpacing (Struct)**: Zeilenabstand (Struktur gemäß `com.sun.star.style.LineSpacing`).
- **ParaBackColor (Long)**: Hintergrundfarbe.
- **ParaLeftMargin (Long)**: linker Rand in 100stel Millimeter.
- **ParaRightMargin (Long)**: rechter Rand in 100stel Millimeter.
- **ParaTopMargin (Long)**: oberer Rand in 100stel Millimeter.
- **ParaBottomMargin (Long)**: unterer Rand in 100stel Millimeter.
- **ParaTabStops (Array of struct)**: Art und Position der Tabulatoren (Array mit Strukturen des Typs `com.sun.star.style.TabStop`).
- **ParaStyleName (String)**: Name der Absatzvorlage.

Beispiel: Einfacher HTML-Export

Das folgende Beispiel demonstriert den Umgang mit Formatierungsinformationen. Es iteriert durch ein Textdokument und erzeugt dabei eine einfache HTML-Datei. Jeder Absatz wird dazu in ein eigenes HTML-Element `<P>` geschrieben. Fett formatierte Absatzteile werden beim Export des Weiteren über ein HTML-Element `<B>` gekennzeichnet.

```
Dim FileNo As Integer, Filename As String, CurLine As String

Dim Doc As Object
Dim Enum1 As Object, Enum2 As Object
Dim TextElement As Object, TextPortion As Object

Filename = "c:\text.html"
FileNo = Freefile
```

```
Open Filename For Output As #FileNo
Print #FileNo, "<HTML><BODY>"

Doc = StarDesktop.CurrentComponent
Enum1 = Doc.Text.createEnumeration

' Schleife über alle Absätze
While Enum1.hasMoreElements
   TextElement = Enum1.nextElement
   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
      Enum2 = TextElement.createEnumeration
      CurLine = "<P>"

      ' Schleife über alle Absatzteile
      While Enum2.hasMoreElements
         TextPortion = Enum2.nextElement
         If TextPortion.CharWeight = com.sun.star.awt.FontWeight.BOLD THEN
            CurLine = CurLine & "<B>" & TextPortion.String & "</B>"
         Else
            CurLine = CurLine & TextPortion.String
         End If
      Wend

      ' Ausgabe der Zeile
      CurLine = CurLine & "</P>"
      Print #FileNo, CurLine

   End If
Wend

' HTML-Fußzeile schreiben
Print #FileNo, "</BODY></HTML>"
Close #FileNo
```

Die Basisstruktur des Beispiels orientiert sich an den weiter oben bereits aufgeführten Beispielen zum Durchlaufen der Absatzteile eines Texts. Hinzugekommen sind die Funktionen zum Schreiben der HTML-Datei sowie ein Prüfcode, der die Schriftstärke der jeweiligen Textabschnitte prüft und fett formatierte Absatzteile mit einem entsprechenden HTML-Tag versieht.

Standardwerte für Zeichen- und Absatzeigenschaften

Direkte Formatierungen haben stets Vorrang gegenüber *indirekten* Formatierungen. Das heißt, durch Vorlagen erzeugte Formatierungen haben eine niedrigere Priorität als direkte Formatierungen in einem Text.

Zu ermitteln, ob ein Abschnitt eines Dokuments direkt oder indirekt formatiert ist, ist keine leichte Aufgabe. In den von StarOffice bereitgestellten Symbolleisten finden sich die

allgemeinen Texteigenschaften wie Schriftart, -stärke und -größe. Ob die betreffenden Einstellungen allerdings auf einer Vorlage basieren oder eine direkte Formatierung im Text sind, bleibt dabei noch unklar.

StarOffice Basic stellt die Methode `getPropertyState` zur Verfügung, mit der Programmierer prüfen können, wie eine bestimmte Formatierungseigenschaft bewirkt wurde. Sie erwartet als Parameter den Namen der Eigenschaft und gibt eine Konstante zurück, die Informationen über die Herkunft der Formatierung enthält. Folgende Antworten, die in der Aufzählung `com.sun.star.beans.PropertyState` definiert sind, sind möglich:

- **com.sun.star.beans.PropertyState.DIRECT_VALUE**: die Eigenschaft wurde im Text direkt definiert (direkte Formatierung),
- **com.sun.star.beans.PropertyState.DEFAULT_VALUE**: die Eigenschaft wurde über eine Vorlage definiert (indirekte Formatierung)
- **com.sun.star.beans.PropertyState.AMBIGUOUS_VALUE**: die Herkunft der Eigenschaft ist unklar. Dieser Zustand tritt beispielsweise beim Abfragen der Eigenschaft Fettdruck eines Absatzes auf, der sowohl fett formatierte als auch normal formatierte Wörter enthält.

Das folgende Beispiel zeigt, wie Formateigenschaften in StarOffice bearbeitet werden können. Es durchsucht einen Text nach Absatzteilen, die mit Hilfe direkter Formatierung fett formatiert wurden. Stößt es auf einen entsprechenden Absatzteil, wird die direkte Formatierung mit der Methode `setPropertyToDefault` gelöscht und dem betreffenden Absatzteil eine Zeichenvorlage `MyBold` zugewiesen.

```
Dim Doc As Object
Dim Enum1 As Object
Dim Enum2 As Object
Dim TextElement As Object
Dim TextPortion As Object

Doc = StarDesktop.CurrentComponent
Enum1 = Doc.Text.createEnumeration

' Schleife über alle Absätze
While Enum1.hasMoreElements
   TextElement = Enum1.nextElement
   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
      Enum2 = TextElement.createEnumeration

      ' Schleife über alle Absatzteile
      While Enum2.hasMoreElements
         TextPortion = Enum2.nextElement
         If TextPortion.CharWeight = _
           com.sun.star.awt.FontWeight.BOLD AND _
           TextPortion.getPropertyState("CharWeight") = _
           com.sun.star.beans.PropertyState.DIRECT_VALUE Then
```

```
            TextPortion.setPropertyToDefault("CharWeight")
            TextPortion.CharStyleName = "MyBold"

         End If

      Wend

   End If

Wend
```

Bearbeiten von Textdokumenten

Im vorigen Abschnitt wurde bereits eine ganze Reihe von Optionen zum Bearbeiten von Textdokumenten besprochen. Dabei lag der Schwerpunkt auf den Diensten `com.sun.star.text.TextPortion` und `com.sun.star.text.Paragraph`, die Zugriff auf Absatzteile und ganze Absätze gewähren. Diese Dienste eignen sich für Anwendungen, in denen der Inhalt eines Texts vom Anfang bis zum Ende in einem einzigen Schleifendurchlauf bearbeitet werden soll. Für viele Probleme reicht diese Vorgehensweise jedoch nicht aus. StarOffice stellt für kompliziertere Aufgaben, einschließlich dem Rückwärtsnavigieren innerhalb eines Dokuments oder dem satz- bzw. wortweisen Navigieren statt mit `TextPortions`, den Dienst `com.sun.star.text.TextCursor` zur Verfügung.

Der TextCursor

Ein `TextCursor` in der StarOffice API ist vergleichbar mit dem sichtbaren Cursor in einem StarOffice-Dokument. Er markiert eine bestimmten Stelle innerhalb eines Textdokuments und lässt sich per Befehl in verschiedene Richtungen navigieren. Dennoch sollte man die in StarOffice Basic verfügbaren `TextCursor`-Objekte nicht mit dem sichtbaren Cursor verwechseln. Es handelt sich dabei um zwei unterschiedliche Dinge.

Hinweis – Die Terminologie weicht gegenüber der in VBA verwendeten ab: Das Range-Objekt von VBA ist vom Funktionsumfang her mit dem TextCursor-Objekt in StarOffice vergleichbar und nicht – wie der Name eventuell vermuten ließe – mit dem Range-Objekt in StarOffice.

Das TextCursor-Objekt in StarOffice stellt beispielsweise Methoden zum Navigieren und Ändern von Text bereit, die in VBA im Range-Objekt enthalten sind (z. B. MoveStart, MoveEnd, InsertBefore, InsertAfter). Die entsprechenden Gegenstücke des TextCursor-Objekts in StarOffice werden in den folgenden Abschnitten beschrieben.

Navigieren innerhalb eines Texts

Das `TextCursor`-Objekt in StarOffice Basic agiert unabhängig von dem sichtbaren Cursor in einem Textdokument. Eine programmgesteuerte Positionsänderung eines `TextCursor`-Objekts hat keinerlei Auswirkungen auf den sichtbaren Cursor. Es ist sogar möglich, für ein und dasselbe Dokument mehrere `TextCursor`-Objekte zu öffnen, die unabhängig voneinander an verschiedenen Positionen eingesetzt werden können.

Ein `TextCursor`-Objekt wird über den Aufruf `createTextCursor` erstellt:

```
Dim Doc As Object
Dim Cursor As Object

Doc = StarDesktop.CurrentComponent
Cursor = TextDocument.Text.createTextCursor()
```

Das so erzeugte Cursor-Objekt unterstützt den Dienst `com.sun.star.text.TextCursor`, der wiederum eine ganze Reihe von Methoden zum Navigieren in Textdokumenten zur Verfügung stellt. Das folgende Beispiel bewegt den `TextCursor` zuerst zehn Zeichen nach links und dann drei Zeichen nach rechts:

```
Cursor.goLeft(10, False)
Cursor.goRight(3, False)
```

Ein `TextCursor` kann einen vollständigen Bereich markieren. Eine solche Markierung ist vergleichbar mit der Markierung einer Textstelle mit der Maus. Der Parameter `False` in den oben stehenden Funktionsaufrufen gibt an, ob der mit der Cursor-Bewegung überquerte Bereich markiert werden soll. So wandert beispielsweise der `TextCursor` in folgendem Beispiel

```
Cursor.goLeft(10, False)
Cursor.goRight(3, True)
```

zuerst zehn Zeichen nach recht, ohne diesen Bereich zu markieren, und geht danach drei Zeichen zurück, wobei diese markiert werden. Der vom `TextCursor` markierte Bereich beginnt somit hinter dem siebten Zeichen im Text und endet hinter dem zehnten Zeichen.

Hier finden Sie nun die zentralen Methoden, die der Dienst `com.sun.star.text.TextCursor` für die Navigation zur Verfügung stellt:

- **goLeft (Count, Expand)**: springt Count Zeichen nach links.
- **goRight (Count, Expand)**: springt Count Zeichen nach rechts.
- **gotoStart (Expand)**: springt an den Anfang des Textdokuments.
- **gotoEnd (Expand)**: springt an das Ende des Textdokuments.
- **gotoRange (TextRange, Expand)**: springt zu dem angegebenen `TextRange`-Objekt.
- **gotoStartOfWord (Expand)**: springt an den Anfang des aktuellen Worts.
- **gotoEndOfWord (Expand)**: springt an das Ende des aktuellen Worts.

- **gotoNextWord (Expand)**: springt an den Anfang des nächsten Worts.
- **gotoPreviousWord (Expand)**: springt an den Anfang des vorangehenden Worts.
- **isStartOfWord ()**: gibt `True` zurück, wenn sich `TextCursor` am Anfang eines Worts befindet.
- **isEndOfWord ()**: gibt `True` zurück, wenn sich `TextCursor` am Ende eines Worts befindet.
- **gotoStartOfSentence (Expand)**: springt an den Anfang des aktuellen Satzes.
- **gotoEndOfSentence (Expand)**: springt an das Ende des aktuellen Satzes.
- **gotoNextSentence (Expand)**: springt an den Anfang des nächsten Satzes.
- **gotoPreviousSentence (Expand)**: springt an den Anfang des vorangehenden Satzes.
- **isStartOfSentence ()**: gibt `True` zurück, wenn sich `TextCursor` am Anfang eines Satzes befindet.
- **isEndOfSentence ()**: gibt `True` zurück, wenn sich `TextCursor` am Ende eines Satzes befindet.
- **gotoStartOfParagraph (Expand)**: springt an den Anfang des aktuellen Absatzes.
- **gotoEndOfParagraph (Expand)**: springt an das Ende des aktuellen Absatzes.
- **gotoNextParagraph (Expand)**: springt an den Anfang des nächsten Absatzes.
- **gotoPreviousParagraph (Expand)**: springt an den Anfang des vorangehenden Absatzes.
- **isStartOfParagraph ()**: gibt `True` zurück, wenn sich `TextCursor` am Anfang eines Absatzes befindet.
- **isEndOfParagraph ()**: gibt `True` zurück, wenn sich `TextCursor` am Ende eines Absatzes befindet.

Die Unterteilung des Texts in Sätze erfolgt auf der Grundlage von Satzzeichen. Punkte werden also beispielsweise Satzendezeichen interpretiert.

Bei dem Parameter `Expand` handelt es sich um einen Boolean-Wert, der angibt, ob der beim Navigieren überquerte Bereich markiert werden soll. Alle Navigationsmethoden geben darüber hinaus einen Parameter zurück, der angibt, ob die Navigation erfolgreich war oder ob die Aktion mangels Text abgebrochen wurde.

Im Folgenden finden Sie eine Liste einiger Methoden zum Bearbeiten der mit einem `TextCursor` markierten Bereiche, die ebenfalls den Dienst `com.sun.star.text.TextCursor` unterstützen:

- **collapseToStart ()**: setzt die Markierung zurück und positioniert den `TextCursor` am Anfang des vorher markierten Bereichs.
- **collapseToEnd ()**: setzt die Markierung zurück und positioniert den `TextCursor` am Ende des vorher markierten Bereichs.
- **isCollapsed ()**: gibt `True` zurück, wenn der `TextCursor` zurzeit keine Markierung enthält.

Formatieren von Text mit dem `TextCursor`

Der Dienst `com.sun.star.text.TextCursor` unterstützt sämtliche Zeichen- und Absatzeigenschaften, die zu Beginn dieses Kapitels bereits vorgestellt wurden.

Das folgende Beispiel verdeutlicht, wie diese zusammen mit einem `TextCursor` verwendet werden können. Es durchläuft ein vollständiges Dokument und formatiert das erste Wort eines jeden Satzes in Fettdruck.

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor

Do

   Cursor.gotoEndOfWord(True)
   Cursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
   Proceed = Cursor.gotoNextSentence(False)
   Cursor.gotoNextWord(False)

Loop While Proceed
```

Das Beispiel erzeugt zuerst ein Dokumentobjekt für den gerade geöffneten Text. Danach iteriert es Satz für Satz durch den gesamten Text, markiert jeweils das erste Wort und formatiert dieses fett.

Abrufen und Ändern von Textinhalten

Enthält ein `TextCursor` einen markierten Bereich, so steht dieser Text über die Eigenschaft `String` des `TextCursor`-Objekts zur Verfügung. Das folgende Beispiel verwendet die Eigenschaft `String` dazu, um jeweils das erste Wort eines Satzes in einem Meldungsfenster anzuzeigen:

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor

Do

   Cursor.gotoEndOfWord(True)
   MsgBox Cursor.String
   Proceed = Cursor.gotoNextSentence(False)
```

```
   Cursor.gotoNextWord(False)
Loop While Proceed
```

Analog kann das erste Wort eines jeden Satzes über die Eigenschaft `String` geändert werden:

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor

Do

   Cursor.gotoEndOfWord(True)
   Cursor.String = "Ups"
   Proceed = Cursor.gotoNextSentence(False)
   Cursor.gotoNextWord(False)

Loop While Proceed
```

Enthält der `TextCursor` einen markierten Bereich, ersetzt eine Zuweisung zur Eigenschaft `String` diesen durch den neuen Text. Ist kein markierter Bereich vorhanden, wird der Text an der aktuellen `TextCursor`-Position eingefügt.

Einfügen von Steuerzeichen

In einigen Situationen ist es notwendig, nicht den eigentlichen Text eines Dokuments zu modifizieren, sondern seine Struktur. Hierfür bietet StarOffice Steuerzeichen an, die in den Text eingefügt werden und seinen Aufbau beeinflussen. Die Steuerzeichen sind in der Konstantengruppe `com.sun.star.text.ControlCharacter` definiert. Folgende Steuerzeichen sind in StarOffice verfügbar:

- **PARAGRAPH_BREAK**: Absatzumbruch.
- **LINE_BREAK**: Zeilenumbruch innerhalb eines Absatzes.
- **SOFT_HYPHEN**: mögliche Position für Silbentrennung.
- **HARD_HYPHEN**: obligatorische Position für Silbentrennung.
- **HARD_SPACE**: geschütztes Leerzeichen, das auch im Blocksatz nicht ausgeschlossen (gespreizt oder gestaucht) wird.

Zum Einfügen der Steuerzeichen bedarf es neben dem Cursor auch des verknüpften Textdokument-Objekts. Das folgende Beispiel fügt hinter dem 20. Zeichen eines Texts einen Absatz ein:

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean
```

```
Doc = StarDesktop.CurrentComponent

Cursor = Doc.Text.createTextCursor
Cursor.goRight(20, False)

Doc.Text.insertControlCharacter(Cursor, _
   com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, False)
```

Der im Aufruf der Methode `insertControlCharacter` verwendete Parameter `False` gewährleistet, dass der aktuell vom `TextCursor` markierte Bereich auch nach der Einfügeoperation bestehen bleibt. Wird hier der Parameter `True` übergeben, so ersetzt `insertControlCharacter` den aktuellen Text.

Suchen von Textteilen

In vielen Fällen gilt es, einen Text nach einen bestimmten Begriff zu durchsuchen und die entsprechende Trefferstelle zu bearbeiten. Hierzu bieten alle StarOffice-Dokumente eine spezielle Schnittstelle, die stets nach dem gleichen Prinzip arbeitet: Vor einem Suchvorgang muss zunächst ein so genannter `SearchDescriptor` erstellt werden, der definiert, was StarOffice in einem Dokument suchen soll. Ein `SearchDescriptor` ist ein Objekt, das den Dienst `com.sun.star.util.SearchDescriptor` unterstützt und über die Methode `createSearchDescriptor` eines Dokuments erzeugt werden kann:

```
Dim SearchDesc As Object
SearchDesc = Doc.createSearchDescriptor
```

Nachdem der `SearchDescriptor` erzeugt wurde, übernimmt er den zu suchenden Text:

```
SearchDesc.searchString="any text"
```

Von seiner Funktion lässt sich der `SearchDescriptor` am ehesten mit dem Such-Dialog von StarOffice vergleichen. Ähnlich wie das Suchfenster lassen sich auch im `SearchDescriptor`-Objekt die für eine Suche notwendigen Einstellungen vornehmen.

Folgende Eigenschaften werden von dem Dienst `com.sun.star.util.SearchDescriptor` zur Verfügung gestellt:

- **SearchBackwards (Boolean)**: durchsucht den Text rückwärts statt vorwärts.
- **SearchCaseSensitive (Boolean)**: unterscheidet bei der Suche Groß- und Kleinschreibung.
- **SearchRegularExpression (Boolean)**: verarbeitet den Suchausdruck wie einen regulären Ausdruck.
- **SearchStyles (Boolean)**: durchsucht den Text nach der angegebenen Absatzvorlage.
- **SearchWords (Boolean)**: sucht nur ganze Wörter.

Auch die StarOffice `SearchSimilarity` (auch "Fuzzy Matching", unscharfe Suche oder Ähnlichkeitssuche) ist in StarOffice Basic verfügbar. Mit dieser Funktion sucht StarOffice nach einem Ausdruck, der dem Suchbegriff zwar ähnlich, aber nicht exakt mit ihm identisch sein muss. Die Anzahl der hinzugekommenen, gelöschten und geänderten Zeichen können für diesen Ausdruck individuell definiert werden. Im Folgenden nun die verknüpften Eigenschaften des Dienstes `com.sun.star.util.SearchDescriptor`:

- **SearchSimilarity (Boolean)**: führt eine Ähnlichkeitssuche durch.
- **SearchSimilarityAdd (Short)**: Anzahl der Zeichen, die bei einer Ähnlichkeitssuche hinzugefügt werden dürfen.
- **SearchSimilarityExchange (Short)**: Anzahl der Zeichen, die im Rahmen einer Ähnlichkeitssuche ersetzt werden dürfen.
- **SearchSimilarityRemove (Short)**: Anzahl der Zeichen, die im Rahmen einer Ähnlichkeitssuche entfernt werden dürfen.
- **SearchSimilarityRelax (Boolean)**: beachtet für den Suchausdruck alle Abweichungsregeln gleichzeitig.

Ist der `SearchDescriptor` wie gewünscht vorbereitet, kann er auf das Textdokument angewendet werden. Für diesen Zweck stellen die StarOffice-Dokumente die Methoden `findFirst` sowie `findNext` zur Verfügung:

```
Found = Doc.findFirst (SearchDesc)

Do While Found
   ' Suchergebnis bearbeiten
   Found = Doc.findNext( Found.End, Search)

Loop
```

Das Beispiel ermittelt in einer Schleife sämtliche Übereinstimmungen und gibt ein `TextRange`-Objekt zurück, das auf die gefundene Textstelle verweist.

Beispiel: Ähnlichkeitssuche

Dieses Beispiel zeigt, wie sich ein Text nach dem Wort "Umsatz" durchsuchen und die gefundenen Stellen fett formatieren lassen. Um neben dem Wort "Umsatz" auch die Pluralform "Umsätze" und Deklinationen wie "Umsatzes" zu finden, kommt eine Ähnlichkeitssuche zum Einsatz. Die gefundenen Ausdrücke weichen um bis zu zwei Buchstaben von dem Suchausdruck ab:

```
Dim SearchDesc As Object
Dim Doc As Object

Doc = StarDesktop.CurrentComponent
```

```
SearchDesc = Doc.createSearchDescriptor
SearchDesc.SearchString="Umsatz"
SearchDesc.SearchSimilarity = True
SearchDesc.SearchSimilarityAdd = 2
SearchDesc.SearchSimilarityExchange = 2
SearchDesc.SearchSimilarityRemove = 2
SearchDesc.SearchSimilarityRelax = False

Found = Doc.findFirst (SearchDesc)

Do While Found
Found.CharWeight = com.sun.star.awt.FontWeight.BOLD
Found = Doc.findNext( Found.End, Search)
Loop
```

Hinweis – Die Grundidee beim Suchen und Ersetzen in StarOffice ist vergleichbar mit der in VBA. Beide Schnittstellen stellen Ihnen ein Objekt zur Verfügung, mit dem die Eigenschaften für das Suchen und Ersetzen definiert werden können. Dieses Objekt wird anschließend auf den gewünschten Textbereich angewendet, um die Aktion auszuführen. Während in VBA über die Find-Eigenschaft des Range-Objekts auf das zuständige Hilfsobjekt zugegriffen wird, wird es in StarOffice Basic über den Aufruf `createSearchDescriptor` oder `createReplaceDescriptor` des Dokumentobjekts erzeugt. Auch die verfügbaren Sucheigenschaften und -methoden unterscheiden sich.

Wie in der alten API von StarOffice erfolgt auch in der neuen API das Suchen und Ersetzen von Text über das Dokumentobjekt. Während es speziell zur Definition der Suchoptionen bisher ein Objekt namens `SearchSettings` gab, erfolgt die Suche im neuen Objekt nun über ein Objekt `SearchDescriptor` oder `ReplaceDescriptor` für das automatische Ersetzen von Text. Diese Objekte umfassen nicht nur die Optionen, sondern auch den aktuellen Suchtext und gegebenenfalls die zugehörige Textersetzung. Die Deskriptor-Objekte werden über das Dokumentobjekt erzeugt, entsprechend den jeweiligen Anforderungen ausgefüllt und im Anschluss daran als Parameter für die Suchmethoden zurück an das Dokumentobjekt übergeben.

Ersetzen von Textteilen

Analog zur Suchfunktion steht auch die Ersetzungsfunktion von StarOffice in StarOffice Basic zur Verfügung. Die Handhabung beider Funktionen ist identisch. Auch für einen Ersetzungsvorgang wird zunächst ein spezielles Objekt benötigt, das die Parameter für den Ersetzungsvorgang aufnimmt. Es wird als `ReplaceDescriptor` bezeichnet und unterstützt den Dienst `com.sun.star.util.ReplaceDescriptor`. Alle der im vorstehenden Absatz beschriebenen Eigenschaften des `SearchDescriptor` werden auch von `ReplaceDescriptor`

unterstützt. So ist es beispielsweise ebenfalls möglich, bei einem Ersetzungsvorgang die Unterscheidung von Groß- und Kleinschreibung ein- und auszuschalten sowie Ähnlichkeitssuchen durchzuführen.

Das folgende Beispiel demonstriert den Einsatz eines `ReplaceDescriptor` für eine Suche innerhalb eines StarOffice-Dokuments.

```
Dim I As Long
Dim Doc As Object
Dim Replace As Object
Dim BritishWords(5) As String
Dim USWords(5) As String

BritishWords() = Array("colour", "neighbour", "centre", "behaviour", _
   "metre", "through")

USWords() = Array("color", "neighbor", "center", "behavior", _
   "meter", "thru")

Doc = StarDesktop.CurrentComponent
Replace = Doc.createReplaceDescriptor

For O = 0 To 5
   Replace.SearchString = BritishWords(I)
   Replace.ReplaceString = USWords(I)
   Doc.replaceAll(Replace)
Next n
```

Die Ausdrücke zum Suchen und Ersetzen werden über die Eigenschaften `SearchString` und `ReplaceString` des `ReplaceDescriptor` gesetzt. Schließlich folgt der eigentliche Ersetzungsvorgang über die Methode `replaceAll` des Dokumentobjekts, die sämtliche Vorkommen des Suchausdrucks ersetzt.

Beispiel: Suchen und Ersetzen von Text mit regulären Ausdrücken

Besonders leistungsfähig ist die Ersetzungsfunktion von StarOffice im Zusammenhang mit regulären Ausdrücken. Diese bieten die Möglichkeit, anstelle eines festen Werts einen variablen Suchausdruck mit Platzhaltern und Sonderzeichen zu definieren.

Die von StarOffice unterstützten regulären Ausdrücke sind in der Online-Hilfe von StarOffice ausführlich beschrieben. Im Folgenden seien einige Beispiele genannt:

- Ein Punkt innerhalb eines Suchausdrucks steht für ein beliebiges Zeichen. So steht der Suchausdruck "sh.rt" sowohl für `shirt` als auch für `short`.
- Das Zeichen ^ markiert den Anfang eines Absatzes. Mit dem Suchausdruck `^Peter` lassen sich so alle Vorkommen des Namens `Peter` finden, die am Anfang eines Absatzes stehen.

- Das Zeichen $ markiert ein Absatzende. Mit dem Suchausdruck `Peter$` lassen sich so alle Vorkommen des Namens `Peter` finden, die am Ende eines Absatzes stehen.
- Ein * bestimmt, dass das vorstehende Zeichen beliebig oft wiederholt vorkommen darf. Mit dem Punkt kann er als Platzhalter für ein beliebiges Zeichen kombiniert werden. Der Ausdruck `temper.*e` steht beispielsweise für die englischen Begriffe `temperance` und `temperature`.

Das folgende Beispiel zeigt, wie mit Hilfe des regulären Ausdrucks "^$" sämtliche Leerzeilen aus einem Textdokument entfernt werden können:

```
Dim Doc As Object
Dim Replace As Object
Dim I As Long

Doc = StarDesktop.CurrentComponent
Replace = Doc.createReplaceDescriptor

Replace.SearchRegularExpression = True
Replace.SearchString = "^$"
Replace.ReplaceString = ""

Doc.replaceAll(Replace)
```

Textdokumente: Mehr als nur Text

Bisher ging es in diesem Kapitel ausschließlich um Textabsätze und deren Teile. Aber Textdokumente können auch andere Objekte enthalten. Dazu zählen Tabellen, Zeichnungen, Textfelder und Verzeichnisse. Diese Objekte können alle an einem beliebigen Punkt innerhalb eines Texts verankert sein.

Aufgrund dieser Gemeinsamkeiten unterstützen alle diese Objekte in StarOffice einen gemeinsamen Basisdienst namens `com.sun.star.text.TextContent`. Er stellt folgende Eigenschaften zur Verfügung:

- **AnchorType (Enum)**: bestimmt den Verankerungstyp eines `TextContent`-Objekts (Standardwerte gemäß Enumeration `com.sun.star.text.TextContentAnchorType`).
- **AnchorTypes (Sequence of Enum)**: Enumeration aller `AnchorTypes`, die ein spezielles `TextContent`-Objekt unterstützen.
- **TextWrap (Enum)**: bestimmt den Textumlauftyp um ein `TextContent`-Objekt (Standardwerte gemäß Enumeration `com.sun.star.text.WrapTextMode`).

Die `TextContent`-Objekte haben auch einige gemeinsame Methoden, insbesondere die zum Erstellen, Einfügen und Löschen von Objekten.

- Ein neues `TextContent`-Objekt wird über die Methode `createInstance` des Dokumentobjekts **erstellt**.

- **Eingefügt** wird ein Objekt mit der `insertTextContent`-Methode des Textobjekts.
- **Gelöscht** werden `TextContent`-Objekte mit der `removeTextContent`-Methode.

In den folgenden Abschnitten finden Sie eine Reihe von Beispielen, die diese Methoden verwenden.

Tabellen

Das folgende Beispiel erstellt eine Tabelle unter Zuhilfenahme der weiter oben beschriebenen Methode createInstance.

```
Dim Doc As Object
Dim Table As Object
Dim Cursor As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

Table = Doc.createInstance("com.sun.star.text.TextTable")
Table.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Table, False)
```

Nach der Erstellung wird die Tabelle über einen `initialize`-Aufruf auf die erforderliche Anzahl von Zeilen und Spalten festgelegt und dann mit `insertTextContent` in das Textdokument eingefügt.

Wie im Beispiel zu sehen, erwartet die Methode `insertTextContent` neben dem einzufügenden `Content`-Objekt noch zwei weitere Parameter:

- ein `Cursor`-Objekt, das die Einfügeposition bestimmt
- eine Boolean-Variable, die angibt, ob das `Content`-Objekt die aktuelle Auswahl des Cursors ersetzen (Wert `True`) oder vor der aktuellen Auswahl in den Text eingefügt werden soll (`False`).

Hinweis – Beim Erstellen und Einfügen von Tabellen in ein Textdokument werden in StarOffice Basic ähnliche Objekte verwendet wie in VBA: das Dokumentobjekt und ein `TextCursor`-Objekt in StarOffice Basic bzw. das `Range`-Objekt als VBA-Gegenstück. Während das Erstellen und Einfügen der Tabelle in VBA die Methode `Document.Tables.Add` übernimmt, wird diese in StarOffice Basic gemäß dem vorstehenden Beispiel über `createInstance` erstellt und initialisiert und mit Hilfe von `insertTextContent` in das Dokument eingefügt.

Die in ein Textdokument eingefügten Tabellen können über eine einfache Schleife ermittelt werden. Zu diesem Zweck dient die Methode des `getTextTables()` des Textdokument-Objekts:

```
Dim Doc As Object
Dim TextTables As Object
Dim Table As Object
Dim I As Integer
Doc = StarDesktop.CurrentComponent
TextTables = Doc.getTextTables()
For I = 0 to TextTables.count - 1

   Table = TextTables(I)
   ' Table bearbeiten
Next I
```

Hinweis – Texttabellen stehen in StarOffice 8 über die Liste `TextTables` des Dokumentobjekts zur Verfügung. Sie löst damit die bisherige Tables-Auflistung im `Selection`-Objekt ab. Das vorstehende Beispiel zeigt, wie eine Texttabelle erstellt werden kann. Im folgenden Abschnitt werden die Zugriffsmöglichkeiten auf Texttabellen beschrieben.

Bearbeiten von Tabellen

Eine Tabelle besteht aus einzelnen Zeilen. Diese wiederum enthalten die verschiedenen Zellen. Streng genommen gibt es in StarOffice keine Tabellenspalten. Diese ergeben sich implizit durch die gleiche Anordnung der untereinander stehenden Zeilen. Um den Zugriff auf die Tabellen zu vereinfachen, bietet StarOffice jedoch einige Methoden an, die spaltenweise arbeiten. Sie sind besonders nützlich, wenn in der Tabelle keine Zellen zusammengeführt wurden.

Wenden wir uns zunächst den Eigenschaften der Tabelle selbst zu. Diese sind im Dienst `com.sun.star.text.TextTable` definiert. Im Folgenden finden Sie eine Liste der wichtigsten Eigenschaften des Tabellenobjekts:

- **BackColor (Long)**: Hintergrundfarbe der Tabelle.
- **BottomMargin (Long)**: unterer Rand in 100stel Millimeter.
- **LeftMargin (Long)**: linker Rand in 100stel Millimeter.
- **RightMargin (Long)**: rechter Rand in 100stel Millimeter.
- **TopMargin (Long)**: oberer Rand in 100stel Millimeter.
- **RepeatHeadline (Boolean)**: Tabellenüberschrift wird auf jeder Seite wiederholt.
- **Width (Long)**: absolute Breite der Tabelle in 100stel Millimeter.

Zeilen

Eine Tabelle besteht aus einer Liste mit Zeilen. Das folgende Beispiel zeigt, wie die Zeilen einer Tabelle abgerufen und formatiert werden können.

```
Dim Doc As Object
Dim Table As Object
Dim Cursor As Object
Dim Rows As Object
Dim Row As Object
Dim I As Integer
Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

Table = Doc.createInstance("com.sun.star.text.TextTable")
Table.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Table, False)
Rows = Table.getRows
For I = 0 To Rows.getCount() - 1
   Row = Rows.getByIndex(I)
   Row.BackColor = &HFF00FF
Next
```

Das Beispiel erstellt zuerst über einen `Table.getRows`-Aufruf eine Liste mit allen Zeilen. Die Methoden `getCount` und `getByIndex` ermöglichen eine Weiterverarbeitung der Liste und gehören zur Schnittstelle `com.sun.star.table.XTableRows`. Die Methode `getByIndex` gibt ein Zeilenobjekt zurück, das den Dienst `com.sun.star.text.TextTableRow` unterstützt.

Im Folgenden finden Sie die zentralen Methoden der Schnittstelle `com.sun.star.table.XTableRows`:

- **getByIndex(Integer)**: gibt ein Zeilenobjekt für den angegebenen Index zurück.
- **getCount()**: gibt die Anzahl der Zeilenobjekte zurück.
- **insertByIndex(Index, Count)**: fügt Count Zeilen ab der Position `Index` in die Tabelle ein.
- **removeByIndex(Index, Count)**: löscht Count Zeilen ab der Position `Index` aus der Tabelle.

Während die Methoden `getByIndex` und `getCount` in allen Tabellen zur Verfügung stehen, können die Methoden `insertByIndex` und `removeByIndex` nur in Tabellen verwendet werden, die keine zusammengeführten Zellen enthalten.

Der Dienst `com.sun.star.text.TextTableRow` bietet folgende Eigenschaften:

- **BackColor (Long)**: Hintergrundfarbe der Zeile.
- **Height (Long)**: Höhe der Zeile in 100stel Millimeter.
- **IsAutoHeight (Boolean)**: Tabellenhöhe wird dynamisch an den Inhalt angepasst.
- **VertOrient (Const)**: vertikale Ausrichtung des Textrahmens – Angaben zur Vertikalen Ausrichtung des Texts innerhalb der Tabelle (Werte gemäß `com.sun.star.text.VertOrientation`).?

Spalten

Der Zugriff auf Spalten erfolgt genauso wie der Zugriff auf Zeilen über die Methoden `getByIndex`, `getCount`, `insertByIndex` und `removeByIndex` am `Column`-Objekt, auf das über `getColumns` zugegriffen wird. Sie können jedoch nur in Tabellen angewendet werden, die keine zusammengeführten Tabellenzellen enthalten. Eine spaltenweise Formatierung von Zellen ist in StarOffice Basic nicht möglich. Um dies zu erreichen, müssen Sie einzelne Tabellenzellen formatieren.

Zellen

Jede Zelle eines StarOffice-Dokuments verfügt über einen eindeutigen Namen. Befindet sich der Cursor von StarOffice in einer Zelle, wird der Name der jeweiligen Zellen in der Statusleiste angezeigt. Die Zelle oben links hat in der Regel den Namen A1, die Zeile unten rechts den Namen `Xn`, wobei `X` für den Buchstaben der höchsten Spalte und `n` für die Nummer der letzten Zeile steht. Die Zellenobjekte sind über die Methode `getCellByName()` des Tabellenobjekts verfügbar. Das folgende Beispiel zeigt eine Schleife, die nacheinander alle Zellen einer Tabelle durchläuft und die jeweilige Zeilen- und Spaltennummer in diese einträgt.

```
Dim Doc As Object
Dim Table As Object
Dim Cursor As Object
Dim Rows As Object
Dim RowIndex As Integer
Dim Cols As Object
Dim ColIndex As Integer
Dim CellName As String
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

Table = Doc.createInstance("com.sun.star.text.TextTable")
Table.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Table, False)

Rows = Table.getRows
Cols = Table.getColumns

For RowIndex = 1 To Rows.getCount()
   For ColIndex = 1 To Cols.getCount()
      CellName = Chr(64 + ColIndex) & RowIndex
      Cell = Table.getCellByName(CellName)
      Cell.String = "Zeile: " & CStr(RowIndex) + ", Spalte: " & CStr(ColIndex)
   Next
Next
```

Eine Tabellenzelle ist mit einem Standardtext vergleichbar. Sie unterstützt die Schnittstelle `createTextCursor` zur Erstellung eines verknüpften `TextCursor`-Objekts.

```
CellCursor = Cell.createTextCursor()
```

Deshalb stehen automatisch alle Formatierungsmöglichkeiten für einzelne Zeichen und Absätze zur Verfügung.

Das folgende Beispiel durchsucht alle Tabellen eines Textdokuments und formatiert alle Zellen mit numerischen Werten über die entsprechende Absatzeigenschaft rechtsbündig.

```
Dim Doc As Object
Dim TextTables As Object
Dim Table As Object
Dim CellNames
Dim Cell As Object
Dim CellCursor As Object
Dim I As Integer
Dim J As Integer

Doc = StarDesktop.CurrentComponent
TextTables = Doc.getTextTables()

For I = 0 to TextTables.count - 1
   Table = TextTables(I)
   CellNames = Table.getCellNames()

   For J = 0 to UBound(CellNames)
      Cell = Table.getCellByName(CellNames(J))
      If IsNumeric(Cell.String) Then
         CellCursor = Cell.createTextCursor()
         CellCursor.paraAdjust = com.sun.star.style.ParagraphAdjust.RIGHT
      End If
   Next
Next
```

Das Beispiel erstellt eine `TextTables`-Liste mit allen Tabellen eines Texts, die in einer Schleife durchlaufen werden. Für jede dieser Tabellen erstellt StarOffice dann wiederum eine Liste der zugehörigen Zellnamen. Auch diese werden in einer Schleife durchlaufen. Enthält eine Zelle einen numerischen Wert, passt das Beispiel die Formatierung entsprechend an. Hierzu erzeugt es zunächst ein `TextCursor`-Objekt, das auf den Inhalt der Tabellenzelle verweist, und passt daraufhin die Absatzeigenschaften der Tabellenzelle an.

Textrahmen

Textrahmen gelten wie Tabellen und Grafiken als `TextContent`-Objekte. Sie bestehen zwar im Wesentlichen aus Standardtext, sind jedoch frei auf einer Seite positionierbar und befinden sich außerhalb des Textflusses.

Wie bei allen `TextContent`-Objekten wird auch bei einem Textrahmen zwischen der eigentlichen Erstellung und dem Einfügen in das Dokument unterschieden.

```
Dim Doc As Object
Dim TextTables As Object
Dim Cursor As Object
Dim Frame As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Frame = Doc.createInstance("com.sun.star.text.TextFrame")
Doc.Text.insertTextContent(Cursor, Frame, False)
```

Das Erstellen eines Textrahmens erfolgt über die Methode `createInstance` des Dokumentobjekts. Der so erstellte Textrahmen kann dann mit der Methode `insertTextContent` des `Text`-Objekts in das Dokument eingefügt werden. Hierbei ist der Name des richtigen Dienstes `com.sun.star.text.TextFrame` anzugeben.

Die Einfügeposition des Textrahmens wird durch ein `Cursor`-Objekt bestimmt, das beim Einfügen mit ausgeführt wird.

Hinweis – Textrahmen bilden das StarOffice-Gegenstück zu den in Word verwendeten Positionsrahmen. Während hierzu in VBA die Methode `Document.Frames.Add` verwendet wird, erfolgt die Erstellung in VBA über oben angeführtes Verfahren unter Zuhilfenahme eines `TextCursor` sowie der Methode `createInstance` des Dokumentobjekts.

Textrahmen-Objekte stellen eine ganze Reihe von Eigenschaften zur Verfügung, mit denen Position und Verhalten des Rahmens beeinflusst werden können. Der Großteil dieser Eigenschaften ist in dem Dienst `com.sun.star.text.BaseFrameProperties` definiert, der auch von jedem `TextFrame`-Dienst unterstützt wird. Die zentralen Eigenschaften sind:

- **BackColor (Long)**: Hintergrundfarbe des Textrahmens.
- **BottomMargin (Long)**: unterer Rand in 100stel Millimeter.
- **LeftMargin (Long)**: linker Rand in 100stel Millimeter.
- **RightMargin (Long)**: rechter Rand in 100stel Millimeter.
- **TopMargin (Long)**: oberer Rand in 100stel Millimeter.
- **Height (Long)**: Höhe des Textrahmens in 100stel Millimeter.

- **Width (Long)**: Breite des Textrahmens in 100stel Millimeter.
- **HoriOrient (Const)**: horizontale Ausrichtung des Textrahmens (gemäß `com.sun.star.text.HoriOrientation`).
- **VertOrient (Const)**: vertikale Ausrichtung des Textrahmens (gemäß `com.sun.star.text.VertOrientation`).

Das folgende Beispiel erstellt einen Textrahmen unter Verwendung der zuvor beschriebenen Eigenschaften.

```
Dim Doc As Object
Dim TextTables As Object
Dim Cursor As Object
Dim Frame As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Cursor.gotoNextWord(False)
Frame = Doc.createInstance("com.sun.star.text.TextFrame")

Frame.Width = 3000
Frame.Height = 1000
Frame.AnchorType = com.sun.star.text.TextContentAnchorType.AS_CHARACTER
Frame.TopMargin = 0
Frame.BottomMargin = 0
Frame.LeftMargin = 0
Frame.RightMargin = 0
Frame.BorderDistance = 0
Frame.HoriOrient = com.sun.star.text.HoriOrientation.NONE
Frame.VertOrient = com.sun.star.text.VertOrientation.LINE_TOP

Doc.Text.insertTextContent(Cursor, Frame, False)
```

Das Beispiel erstellt einen `TextCursor` als Einfügemarke für den Textrahmen. Dieser wird zwischen dem ersten und dem zweiten Wort des Texts positioniert. Der Textrahmen wird über `Doc.createInstance` erstellt. Die Eigenschaften des Textrahmen-Objekts werden auf die erforderlichen Ausgangswerte gesetzt.

Zu beachten ist hierbei das Zusammenspiel der Eigenschaften `AnchorType` (aus dem Dienst `TextContent`) und `VertOrient` (aus dem Dienst `BaseFrameProperties`). `AnchorType` erhält den Wert `AS_CHARACTER`. Der Textrahmen wird somit direkt in den Textfluss eingefügt und verhält sich wie ein Zeichen. So kann er beispielsweise bei einem Zeilenumbruch in die nächste Zeile verschoben werden. Der Wert `LINE_TOP` der Eigenschaft `VertOrient` stellt sicher, dass sich die Oberkante des Textrahmens auf gleicher Höhe mit der Oberkante der Zeichen befindet.

Nach abgeschlossener Initialisierung wird der Textrahmen schließlich über einen Aufruf von `insertTextContent` in das Textdokument eingefügt.

Zum Bearbeiten des Inhalts eines Textrahmens wird der bereits mehrfach erwähnte `TextCursor` verwendet, der auch für Textrahmen verfügbar ist.

```
Dim Doc As Object
Dim TextTables As Object
Dim Cursor As Object
Dim Frame As Object
Dim FrameCursor As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Frame = Doc.createInstance("com.sun.star.text.TextFrame")

Frame.Width = 3000
Frame.Height = 1000

Doc.Text.insertTextContent(Cursor, Frame, False)

FrameCursor = Frame.createTextCursor()
FrameCursor.charWeight = com.sun.star.awt.FontWeight.BOLD
FrameCursor.paraAdjust = com.sun.star.style.ParagraphAdjust.CENTER
FrameCursor.String = "Dies ist ein kleiner Test!"
```

Das Beispiel erstellt einen Textrahmen, fügt diesen in das aktuelle Dokument ein und öffnet einen `TextCursor` für den Textrahmen. Über diesen Cursor wird die Schrift des Rahmens auf Fettdruck und die Absatzausrichtung auf zentriert eingestellt. Schließlich wird dem Textrahmen die Zeichenfolge (String) "Dies ist ein kleiner Test!" zugewiesen.

Textfelder

Textfelder zählen zu den `TextContent`-Objekten, da sie über den reinen Text hinaus eine zusätzliche Logik zur Verfügung stellen. Textfelder lassen sich über die gleichen Methoden in ein Textdokument einfügen wie andere `TextContent`-Objekte:

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

DateTimeField = Doc.createInstance("com.sun.star.text.TextField.DateTime")
DateTimeField.IsFixed = False
DateTimeField.IsDate = True
Doc.Text.insertTextContent(Cursor, DateTimeField, False)
```

Das Beispiel fügt am Anfang des aktuellen Textdokuments ein Textfeld mit dem aktuellen Datum ein. Der Wert `True` der Eigenschaft `IsDate` führt dazu, dass nur das Datum ohne Uhrzeit angezeigt wird. Der Wert `False` für `IsFixed` stellt sicher, dass das Datum beim Öffnen des Dokuments automatisch aktualisiert wird.

Hinweis – Während der Typ eines Felds in VBA durch einen Parameter der Methode `Document.Fields.Add` angegeben wird, erfolgt die Definition in StarOffice Basic durch den Namen des Dienstes, der für den betreffenden Feldtyp verantwortlich ist.

Der Zugriff auf Textfelder erfolgte bisher über eine ganze Reihe von Methoden, die StarOffice im alten `Selection`-Objekt zur Verfügung gestellt hat (z. B. `InsertField`, `DeleteUserField`, `SetCurField`).

In StarOffice 8 werden die Felder gemäß einem objektorientierten Konzept verwaltet. Für die Erstellung eines Textfelds muss zunächst ein Textfeld des erforderlichen Typs erstellt und über die notwendigen Eigenschaften initialisiert werden. Anschließend wird das Textfeld über die Methode `insertTextContent` in das Dokument eingefügt. Ein entsprechender Quelltext wird im vorstehenden Beispiel gezeigt. Die wichtigsten Feldtypen und ihre Eigenschaften werden in den folgenden Abschnitten beschrieben.

Neben dem Einfügen von Textfeldern kann auch das Durchsuchen eines Dokuments nach Feldern eine wichtige Aufgabe sein. Das folgende Beispiel zeigt, wie alle Textfelder eines Textdokuments in einer Schleife durchlaufen und auf ihren jeweiligen Typ hin überprüft werden können.

```
Dim Doc As Object
Dim TextFieldEnum As Object
Dim TextField As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent

TextFieldEnum = Doc.getTextFields.createEnumeration

While TextFieldEnum.hasMoreElements()

   TextField = TextFieldEnum.nextElement()

   If TextField.supportsService("com.sun.star.text.TextField.DateTime") Then
      MsgBox "Datum/Uhrzeit"
   ElseIf TextField.supportsService("com.sun.star.text.TextField.Annotation") Then
      MsgBox "Anmerkung"
   Else
      MsgBox "unbekannt"
   End If
```

```
Wend
```

Ausgangspunkt für die Ermittlung der vorhandenen Textfelder bildet die Liste `TextFields` des Dokumentobjekts. Auf dieser Grundlage erstellt das Beispiel ein `Enumeration`-Objekt, über das wiederum alle Textfelder in einer Schleife abgefragt werden können. Die gefundenen Textfelder werden über die Methode `supportsService` auf den unterstützten Service hin überprüft. Handelt es sich bei dem Feld um ein Datums-/Uhrzeitfeld oder eine Anmerkung, wird die betreffende Feldart in einem Hinweisfenster angezeigt. Stößt das Beispiel hingegen auf ein anderes Feld, so zeigt es den Hinweis "unbekannt" an.

Im Folgenden finden Sie eine Aufstellung der wichtigsten Textfelder sowie der zugehörigen Eigenschaften. Eine vollständige Liste aller Textfelder finden Sie in der API-Referenz im Modul `com.sun.star.text.TextField`. (Beim Aufführen der Dienstnamen eines Textfelds ist in StarOffice Basic die Groß- und Kleinschreibung wie im vorstehenden Beispiel zu verwenden.)

Anzahl der Seiten, Wörter und Zeichen

Die Textfelder

- `com.sun.star.text.TextField.PageCount`
- `com.sun.star.text.TextField.WordCount`
- `com.sun.star.text.TextField.CharacterCount`

geben die Anzahl der Seiten, Wörter und Zeichen eines Texts zurück. Sie unterstützen die folgende Eigenschaft:

- **NumberingType (Const)**: Nummerierungsformat (Richtlinien gemäß der Konstanten aus `com.sun.star.style.NumberingType`).

Aktuelle Seite

Die Nummer der aktuellen Seite lässt sich über das Textfeld `com.sun.star.text.TextField.PageNumber` in ein Dokument einfügen. Folgende Eigenschaften können angegeben werden:

- **NumberingType (Const)**: Zahlenformat (Richtlinien gemäß der Konstanten aus `com.sun.star.style.NumberingType`).
- **Offset (Short)** : Versatz, der der Seitenzahl hinzugefügt wird (negative Angaben sind ebenfalls möglich).

Das folgende Beispiel zeigt, wie die Seitenzahl in die Fußzeile eines Dokuments eingefügt werden kann.

```
Dim Doc As Object
Dim DateTimeField As Object
Dim PageStyles As Object
```

```
Dim StdPage As Object
Dim FooterCursor As Object
Dim PageNumber As Object

Doc = StarDesktop.CurrentComponent

PageNumber = Doc.createInstance("com.sun.star.text.TextField.PageNumber")
PageNumber.NumberingType = com.sun.star.style.NumberingType.ARABIC

PageStyles = Doc.StyleFamilies.getByName("PageStyles")

StdPage = PageStyles("Default")
StdPage.FooterIsOn = True

FooterCursor = StdPage.FooterTextLeft.Text.createTextCursor()
StdPage.FooterTextLeft.Text.insertTextContent(FooterCursor, PageNumber, False)
```

Das Beispiel erstellt zuerst ein Textfeld, das den Dienst `com.sun.star.text.TextField.PageNumber` unterstützt. Da die Kopf- und Fußzeilen als Teil der Seitenvorlagen von StarOffice definiert sind, wird dieses zunächst über die Liste aller `PageStyles` ermittelt.

Um sicherzustellen, dass die Fußzeile tatsächlich angezeigt wird, wird die Eigenschaft `FooterIsOn` auf `True` gesetzt. Anschließend wird das Textfeld über das verknüpfte Textobjekt der linken Fußzeile in das Dokument eingefügt.

Anmerkungen

Anmerkungsfelder (`com.sun.star.text.TextField.Annotation`) sind durch ein kleines gelbes Symbol im Text gekennzeichnet. Durch Klicken auf das Symbol wird ein Textfeld geöffnet, in dem eine Anmerkung zur aktuellen Textstelle erfasst werden kann. Ein Anmerkungsfeld verfügt über folgende Eigenschaften.

- **Author (String)**: Name des Autors.
- **Content (String)**: Anmerkungstext.
- **Date (Date)**: Erstellungsdatum der Anmerkung.

Datum/Uhrzeit

Ein Datums-/Uhrzeit-Feld (`com.sun.star.text.TextField.DateTime`) stellt das aktuelle Datum beziehungsweise die aktuelle Uhrzeit dar. Es unterstützt die folgenden Eigenschaften:

- **IsFixed (Boolean)**: wenn `True`, bleiben die Zeitangaben der Einfügung unverändert, wenn `False`, werden diese bei jedem Öffnen des Dokuments aktualisiert.
- **IsDate (Boolean)**: wenn `True`, zeigt das Feld das aktuelle Datum an, andernfalls die aktuelle Uhrzeit.
- **DateTimeValue (Struct)**: aktueller Inhalt des Felds (Struktur `com.sun.star.util.DateTime`).

- **NumberFormat (Const)**: Format, in dem Uhrzeit und Datum angezeigt werden.

Kapitelname/-nummer

Der Name des aktuellen Kapitels steht über ein Textfeld des Typs `com.sun.star.text.TextField.Chapter` zur Verfügung. Die Form kann über zwei Eigenschaften definiert werden.

- **ChapterFormat (Const)**: bestimmt, ob der Kapitelname oder die Kapitelnummer angezeigt wird (gemäß `com.sun.star.text.ChapterFormat`).
- **Level (Integer)**: bestimmt die Kapitelebene, deren Name und/oder Kapitelnummer angezeigt werden soll. Der Wert 0 steht für die oberste verfügbare Ebene.

Lesezeichen (Bookmarks)

Lesezeichen (Dienst `com.sun.star.text.Bookmark`) gehören zu den `TextContent`-Objekten. Lesezeichen werden gemäß dem bereits zuvor beschriebenen Konzept erstellt und eingefügt:

```
Dim Doc As Object
Dim Bookmark As Object
Dim Cursor As Object

Doc = StarDesktop.CurrentComponent

Cursor = Doc.Text.createTextCursor()

Bookmark = Doc.createInstance("com.sun.star.text.Bookmark")
Bookmark.Name = "My bookmarks"
Doc.Text.insertTextContent(Cursor, Bookmark, True)
```

Das Beispiel erzeugt einen `Cursor`, der die Einfügeposition des Lesezeichens markiert, und dann das eigentliche Lesezeichen-Objekt (`Bookmark`). Dem Lesezeichen wird anschließend ein Name zugewiesen und es wird über `insertTextContent` an der Cursor-Position in das Dokument eingefügt.

Der Zugriff auf die Lesezeichen eines Texts erfolgt über eine Liste namens `Bookmarks`. Auf die Lesezeichen kann entweder über ihre Nummer oder über ihren Namen zugegriffen werden.

Das folgende Beispiel zeigt, wie sich ein Lesezeichen innerhalb eines Texts auffinden und ein Text an seiner Position einfügen lässt.

```
Dim Doc As Object
Dim Bookmark As Object
Dim Cursor As Object

Doc = StarDesktop.CurrentComponent
```

```
Bookmark = Doc.Bookmarks.getByName("My bookmarks")

Cursor = Doc.Text.createTextCursorByRange(Bookmark.Anchor)
Cursor.String = "Hier ist das Lesezeichen."
```

In diesem Beispiel dient die Methode `getByName` dazu, das gewünschte Lesezeichen anhand seines Namens aufzufinden. Danach erzeugt der Aufruf `createTextCursorByRange` dann einen `Cursor`, der an der Anker-Position des Lesezeichens positioniert ist. Dort fügt der Cursor dann den gewünschten Text ein.

◆ ◆ ◆ KAPITEL 7

Tabellendokumente

StarOffice Basic stellt eine umfangreiche Schnittstelle für die programmgesteuerte Erstellung und Bearbeitung von Tabellendokumenten zur Verfügung. In diesem Kapitel wird beschrieben, wie sich die betreffenden Dienste, Methoden und Eigenschaften von Tabellendokumenten steuern lassen.

Im ersten Abschnitt wird der grundlegende Aufbau von Tabellendokumenten behandelt und gezeigt, wie auf den Inhalt einzelner Zellen zugegriffen und dieser bearbeitet werden kann.

Der zweite Abschnitt konzentriert sich auf die effiziente Bearbeitung von Tabellendokumenten. Im Vordergrund stehen dabei der Dienst zum Bearbeiten von Zellbereichen und die Möglichkeiten zum Suchen und Ersetzen von Zellinhalten.

Hinweis – Das `Range`-Objekt ermöglicht Ihnen die Adressierung jedes Tabellenbereichs. Sein Funktionsumfang wurde in der neuen API erweitert.

Der Aufbau von Tabellendokumenten

Die Dokumentobjekte eines Tabellendokuments basieren auf dem Dienst `com.sun.star.sheet.SpreadsheetDocument`. Jedes dieser Dokumente kann mehrere Tabellen enthalten. In diesem Handbuch wird unter einem *Tabellendokument*, manchmal auch als *Spreadsheet-Dokument* bezeichnet, das gesamte Dokument verstanden, während *Tabelle* (oder *Spreadsheet*) eine in diesem Dokument enthaltene einzelne Tabelle bezeichnet.

Hinweis – Die Terminologie für Tabellendokumente und ihren Inhalt unterscheidet sich zwischen VBA und StarOffice Basic. Während das Dokumentobjekt in VBA *Workbook* und seine einzelnen Seiten *Worksheets* heißen, lauten ihre Bezeichnungen in StarOffice Basic *SpreadsheetDocument* und *Sheet*.

Tabellen (Spreadsheets)

Die einzelnen Tabellen eines Tabellendokuments sind über die Liste `Sheets` verfügbar.

Das folgende Beispiel zeigt, wie Sie entweder über die Nummer oder den Namen auf eine Tabelle zugreifen.

Beispiel 1: Zugriff über die Nummer (Nummerierung beginnt mit 0)

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet  = Doc. Sheets (0)
```

Beispiel 2: Zugriff über den Namen

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
```

In Beispiel 1 wird über ihre Nummer auf die Tabelle zugegriffen (die Zählung beginnt hier mit 0). Im zweiten Beispiel erfolgt der Zugriff auf die Tabelle über ihren Namen mit der Methode `getByName`.

Das `Sheet`-Objekt, das mit der Methode `getByName` abgerufen wird, unterstützt den Dienst `com.sun.star.sheet.Spreadsheet`. Neben mehreren Schnittstellen zum Bearbeiten des Inhalts stellt dieser Dienst folgende Eigenschaften zur Verfügung:

- **IsVisible (Boolean)**: die Tabelle wird angezeigt.
- **PageStyle (String)**: Name der Seitenvorlage für die Tabelle.

Erstellen, Löschen und Umbenennen von Tabellen (Sheets)

Auch das Erstellen, Löschen und Umbenennen einzelner Tabellen erfolgt über die `Sheets`-Liste des `Spreadsheet`-Dokuments. Das folgende Beispiel prüft über die Methode `hasByName`, ob eine Tabelle mit dem Namen *MySheet* existiert. Falls ja, ermittelt die Methode eine entsprechende Objektreferenz über die Methode `getByName` und speichert diese dann in einer Variable in `Sheet` ab. Existiert die betreffende Tabelle noch nicht, so wird sie über den Aufruf `createInstance` erzeugt und mit der Methode `insertByName` in das Tabellendokument eingefügt.

```
Dim Doc As Object
Dim Sheet As Object
```

```
Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

If Doc.Sheets.hasByName("MySheet") Then
   Sheet = Doc.Sheets.getByName("MySheet")
Else
   Sheet = Doc.createInstance("com.sun.star.sheet.Spreadsheet")
   Doc.Sheets.insertByName("MySheet", Sheet)
End If
```

Die Methoden `getByName` und `insertByName` stammen aus der Schnittstelle `com.sun.star.container.XnameContainer`, die in Kapitel 4, beschrieben wird.

Zeilen und Spalten

Jede Tabelle enthält eine Liste ihrer Zeilen und Spalten. Sie sind über die Eigenschaften `Rows` beziehungsweise `Columns` des Tabellenobjekts verfügbar und unterstützen die Dienste `com.sun.star.table.TableColumns` beziehungsweise `com.sun.star.table.TableRows`.

Das folgende Beispiel erzeugt zwei Objekte, die auf die erste Zeile und die erste Spalte einer Tabelle verweisen, und speichert diese Referenzen in den Objektvariablen `FirstCol` und `FirstRow`.

```
Dim Doc As Object
Dim Sheet As Object
Dim FirstRow As Object
Dim FirstCol As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

FirstCol = Sheet.Columns(0)
FirstRow = Sheet.Rows(0)
```

Die Spaltenobjekte unterstützen den Dienst `com.sun.star.table.TableColumn`, der folgende Eigenschaften bereitstellt:

- **Width (Long)**: Breite einer Spalte in 100stel Millimeter.
- **OptimalWidth (Boolean)**: setzt eine Spalte auf ihre optimale Breite.
- **IsVisible (Boolean)**: zeigt eine Spalte an.
- **IsStartOfNewPage (Boolean)**: erzeugt beim Drucken vor einer Spalte einen Seitenumbruch.

Die Breite einer Spalte wird nur optimiert, wenn die Eigenschaft `OptimalWidth` auf `True` festgelegt ist. Wird die Breite einer einzelnen Zelle geändert, bleibt die Breite der Spalte, in der sich die Zelle befindet, unverändert. Funktional handelt es sich bei `OptimalWidth` also eher um eine Methode als um eine Eigenschaft.

Die Zeilenobjekte basieren auf dem Dienst `com.sun.star.table.RowColumn`, der folgende Eigenschaften bereitstellt:

- **Height (long)**: Höhe der Zeile in 100stel Millimeter.
- **OptimalHeight (Boolean)**: setzt die Zeile auf ihre optimale Höhe.
- **IsVisible (Boolean)**: zeigt die Zeile an.
- **IsStartOfNewPage (Boolean)**: erzeugt beim Drucken vor einer Zeile einen Seitenumbruch.

Wenn die Eigenschaft `OptimalHeight` einer Zeile auf den Wert `True` festgelegt ist, ändert sich die Zeilenhöhe automatisch, wenn die Höhe einer Zelle in der Zeile geändert wird. Die automatische Optimierung bleibt so lange aktiv, bis der Zeile über die Eigenschaft `Height` eine absolute Höhe zugewiesen wird.

Folgendes Beispiel aktiviert die automatische Höhenoptimierung für die ersten fünf Zeilen in der Tabelle und blendet die zweite Spalte aus.

```
Dim Doc As Object
Dim Sheet As Object
Dim Row As Object
Dim Col As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

For I = 0 To 4
   Row = Sheet.Rows(I)
   Row.OptimalHeight = True
Next I

Col = Sheet.Columns(1)
Col.IsVisible = False
```

Hinweis – Auf die Listen `Rows` und `Columns` kann über einen Index in StarOffice Basic zugegriffen werden. Anders als in VBA hat die erste Spalte jedoch den Index 0 und nicht den Index 1.

Einfügen und Löschen von Zeilen und Spalten

Die Objekte `Rows` und `Columns` einer Tabelle können sowohl auf vorhandene Zeilen und Spalten zugreifen und sie löschen als auch neue einfügen.

```
Dim Doc As Object
Dim Sheet As Object
Dim NewColumn As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

Sheet.Columns.insertByIndex(3, 1)
Sheet.Columns.removeByIndex(5, 1)
```

Das Beispiel fügt über die Methode `insertByIndex` an der vierten Spaltenposition in der Tabelle eine neue Spalte ein (Index 3, da Nummerierung mit 0 beginnt). Der zweite Parameter gibt die Anzahl der einzufügenden Spalten an (im Beispiel eine).

Die Methode `removeByIndex` löscht die sechste Spalte (Index 5). Auch hier gibt der zweite Parameter die Anzahl der zu löschenden Spalten an.

Die Methoden zum Einfügen und Löschen von Zeilen über das `Rows`-Objekt arbeiten auf dieselbe Weise wie die dargestellten Methoden zur Bearbeitung von Spalten über das `Columns`-Objekt.

Zellen

Eine Tabelle setzt sich aus einer zweidimensionalen Liste mit Zellen zusammen. Jede Zelle wird durch ihre X- und Y-Position in Bezug auf die oberste linke Zelle mit der Position (0,0) definiert.

Das folgende Beispiel erzeugt ein Objekt, das auf die oberste linke Zelle verweist und darin Text einfügt:

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

Cell = Sheet.getCellByPosition(0, 0)
Cell.String = "Test"
```

Zusätzlich zu den numerischen Koordinaten hat jede Zelle in einer Tabelle einen Namen, zum Beispiel wird die oberste linke Zelle (0,0) einer Tabelle mit `A1` bezeichnet. Der Buchstabe `A` steht dabei für die Spalte und die Ziffer 1 für die Zeile. Es ist wichtig, den *Namen* und die *Position* einer Zelle nicht zu verwechseln, weil die Zählung der Zeilen für den Namen mit 1 beginnt, die für die Position jedoch mit 0.

Eine Tabellenzelle kann in StarOffice leer sein oder Text, Zahlen oder Formeln enthalten. Der Zellentyp wird dabei nicht über den darin gespeicherten Inhalt bestimmt, sondern über die für den Eintrag verwendete Objekt-Eigenschaft. Zahlen lassen sich über die Eigenschaft `Value` einfügen und auslesen, Text über die Eigenschaft `String` und Formeln über die Eigenschaft `Formula`.

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

Cell = Sheet.getCellByPosition(0, 0)
Cell.Value = 100

Cell = Sheet.getCellByPosition(0, 1)
Cell.String = "Test"

Cell = Sheet.getCellByPosition(0, 2)
Cell.Formula = "=A1"
```

Das Beispiel fügt in die Felder A1 bis A3 eine Zahl, einen Text und eine Formel ein.

Hinweis – Die Eigenschaften `Value`, `String` und `Formula` ersetzen die Methode `PutCell` zum Setzen der Werte einer Tabellenzelle.

StarOffice verarbeitet Zellinhalte, die mit der Eigenschaft `String` eingegeben werden, als Text, sogar wenn es sich bei dem Inhalt um eine Zahl handelt. Zahlen werden in Zellen links- statt rechtsbündig ausgerichtet. Aber auch beim Einsatz von Formeln ist auf den Unterschied zwischen Text und Zahlen zu achten:

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

Cell = Sheet.getCellByPosition(0, 0)
Cell.Value = 100

Cell = Sheet.getCellByPosition(0, 1)
Cell.String = 1000

Cell = Sheet.getCellByPosition(0, 2)
```

```
Cell.Formula = "=A1+A2"

MsgBox Cell.Value
```

Obwohl die Zelle A1 den Wert 100 und die Zelle A2 den Wert 1000 enthält, gibt die Formel A1+A2 den Wert 100 zurück. Ursache hierfür ist, dass der Inhalt von Zelle A2 als String (Zeichenfolge) und nicht als Zahl eingegeben wurde.

Zur Überprüfung, ob eine Zelle eine Zahl oder eine Zeichenfolge enthält, verwenden Sie die Eigenschaft `Type`:

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
Cell = Sheet.getCellByPosition(1,1)

Cell.Value = 1000

Select Case Cell.Type
Case com.sun.star.table.CellContentType.EMPTY
   MsgBox "Content: Empty"
Case com.sun.star.table.CellContentType.VALUE
   MsgBox "Content: Value"
Case com.sun.star.table.CellContentType.TEXT
   MsgBox "Content: Text"
Case com.sun.star.table.CellContentType.FORMULA
   MsgBox "Content: Formula"
End Select
```

Die Eigenschaft `Cell.Type` gibt für die Aufzählung `com.sun.star.table.CellContentType` einen Wert zurück, der den Inhaltstyp einer Zelle identifiziert. Mögliche Werte sind:

- **EMPTY**: kein Wert.
- **VALUE**: Zahl.
- **TEXT**: Zeichenfolge.
- **FORMULA**: Formel.

Einfügen, Löschen, Kopieren und Verschieben von Zellen

Neben der Möglichkeit zum direkten Ändern von Zellinhalten bietet StarOffice Calc außerdem eine Schnittstelle, die das Einfügen, Löschen, Kopieren und Zusammenführen von Zellen gestattet. Die über das Spreadsheet-Objekt verfügbare Schnittstelle (`com.sun.star.sheet.XRangeMovement`) stellt zum Ändern von Zellinhalten vier Methoden zur Verfügung.

Die Methode `insertCell` wird zum Einfügen von Zellen in eine Tabelle verwendet.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

CellRangeAddress.Sheet = 0
CellRangeAddress.StartColumn = 1
CellRangeAddress.StartRow = 1
CellRangeAddress.EndColumn = 2
CellRangeAddress.EndRow = 2

Sheet.insertCells(CellRangeAddress, com.sun.star.sheet.CellInsertMode.DOWN)
```

Das Beispiel fügt in der ersten Tabelle (Nummer 0) des Tabellendokuments in der zweiten Spalte und Zeile (tragen jeweils die Nummer 1) einen zwei Zeilen hohen und zwei Spalten breiten Zellbereich ein. Alle in dem angegebenen Zellbereich bereits vorhandenen Werte werden unter den Bereich verschoben.

Um den einzufügenden Zellbereich zu definieren, verwenden Sie die Struktur `com.sun.star.table.CellRangeAddress`. Diese Struktur enthält folgende Werte:

- **Sheet (Short)**: Nummer der Tabelle (Sheet; Nummerierung beginnt mit 0).
- **StartColumn (Long)**: erste Spalte im Zellbereich (Nummerierung beginnt mit 0).
- **StartRow (Long)**: erste Zeile im Zellbereich (Nummerierung beginnt mit 0).
- **EndColumn (Long)**: letzte Spalte im Zellbereich (Nummerierung beginnt mit 0).
- **EndRow (Long)**: letzte Zeile im Zellbereich (Nummerierung beginnt mit 0).

Die ausgefüllte `CellRangeAddress`-Struktur muss der Methode `insertCells` als erster Parameter übergeben werden. Der zweite Parameter von `insertCells` enthält einen Wert der Aufzählung `com.sun.star.sheet.CellInsertMode` und definiert, wie die Werte zu verarbeiten sind, die sich vor der Einfügeposition befinden. Die Aufzählung `CellInsertMode` erkennt folgende Werte:

- **NONE**: die aktuellen Werte bleiben an ihrer aktuellen Position.
- **DOWN**: die Zellen an und unterhalb der Einfügeposition werden nach unten verschoben.
- **RIGHT**: die Zellen an und rechts der Einfügeposition werden nach rechts verschoben.
- **ROWS**: die unterhalb der Einfügeposition liegenden Zeilen werden nach unten verschoben.
- **COLUMNS**: die rechts von der Einfügeposition liegenden Spalten werden nach rechts verschoben.

Die Methode `removeRange` ist das Gegenstück zur Methode `insertCells`. Diese Methode löscht den in der Struktur `CellRangeAddress` definierten Bereich aus der Tabelle.

```
Dim Doc As Object
Dim Sheet As Object
```

```
Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

CellRangeAddress.Sheet = 0
CellRangeAddress.StartColumn = 1
CellRangeAddress.StartRow = 1
CellRangeAddress.EndColumn = 2
CellRangeAddress.EndRow = 2

Sheet.removeRange(CellRangeAddress, com.sun.star.sheet.CellDeleteMode.UP)
```

Dieses Beispiel entfernt den Zellbereich `B2:C3` aus der Tabelle und verschiebt dann die darunter liegenden Zellen um zwei Zeilen nach oben. Die Art des Löschens wird über einen der folgenden Werte aus der Aufzählung `com.sun.star.sheet.CellDeleteMode` definiert:

- **NONE**: die aktuellen Werte bleiben an ihrer aktuellen Position.
- **UP**: die Zellen an und unterhalb der Einfügeposition werden nach oben verschoben.
- **LEFT**: die Zellen an und rechts der Einfügeposition werden nach links verschoben.
- **ROWS**: die unterhalb der Einfügeposition liegenden Zeilen werden nach oben verschoben.
- **COLUMNS**: die rechts von der Einfügeposition liegenden Spalten werden nach links verschoben.

Die Schnittstelle `XRangeMovement` bietet zwei zusätzliche Methoden zum Verschieben (`moveRange`) oder Kopieren (`copyRange`) von Zellbereichen. Das folgende Beispiel verschiebt den Bereich `B2:C3` so, dass er an der Position `A6` beginnt:

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress
Dim CellAddress As New com.sun.star.table.CellAddress

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

CellRangeAddress.Sheet = 0
CellRangeAddress.StartColumn = 1
CellRangeAddress.StartRow = 1
CellRangeAddress.EndColumn = 2
CellRangeAddress.EndRow = 2

CellAddress.Sheet = 0
CellAddress.Column = 0
CellAddress.Row = 5

Sheet.moveRange(CellAddress, CellRangeAddress)
```

Zusätzlich zu der Struktur `CellRangeAdress` erwartet die Methode `moveRange` eine Struktur `com.sun.star.table.CellAddress`, die den Ursprung des Zielbereichs der Verschiebung definiert. Die Methode `CellAddress` stellt folgende Werte zur Verfügung:

- **Sheet (Short)**: Nummer der Tabelle (Nummerierung beginnt mit 0).
- **Column (Long)**: Nummer der adressierten Spalte (Nummerierung beginnt mit 0).
- **Row (Long)**: Nummer der adressierten Zeile (Nummerierung beginnt mit 0).

Die Zellinhalte im Zielbereich werden immer von der Methode `moveRange` überschrieben. Anders als bei der Methode `InsertCells`, wird in der Methode `removeRange` kein Parameter zur Durchführung automatischer Verschiebungen bereitgestellt.

Die Methode `copyRange` arbeitet auf dieselbe Weise wie die Methode `moveRange`, mit der Ausnahme, dass `copyRange` eine Kopie des Zellbereichs einfügt, statt ihn zu verschieben.

Hinweis – Hinsichtlich ihrer Funktion sind die StarOffice Basic-Methoden `insertCell`, `removeRange` und `copyRange` mit den VBA-Methoden `Range.Insert`, `Range.Delete` und `Range.Copy` vergleichbar. Während sie in VBA auf das entsprechende `Range`-Objekt angewendet werden, werden sie in StarOffice Basic auf das verknüpfte `Sheet`-Objekt angewendet.

Formatieren

Ein Tabellendokument stellt Eigenschaften und Methoden zum Formatieren von Zellen und Seiten zur Verfügung.

Zelleigenschaften

Zum Formatieren von Zellen stehen zahlreiche Optionen zur Verfügung, beispielsweise das Festlegen von Schriftart und -größe für den Text. Jede Zelle unterstützt die Dienste `com.sun.star.style.CharacterProperties` und `com.sun.star.style.ParagraphProperties`, deren Haupteigenschaften in Kapitel 6, beschrieben werden. Spezielle Zellformatierungen werden von dem Dienst `com.sun.star.table.CellProperties` verarbeitet. Die Haupteigenschaften dieses Dienstes werden in den folgenden Abschnitten vorgestellt.

Alle der angeführten Eigenschaften können auf einzelne Zellen sowie auf Zellbereiche angewendet werden.

Hinweis – Das `CellProperties`-Objekt der StarOffice API ist mit dem `Interior`-Objekt in VBA vergleichbar, das ebenfalls zellspezifische Eigenschaften definiert.

Hintergrundfarbe und Schatten

Der Dienst `com.sun.star.table.CellProperties` stellt folgende Eigenschaften zum Definieren von Hintergrundfarben und Schatten bereit:

- **CellBackColor (Long)**: Hintergrundfarbe der Tabellenzelle.
- **IsCellBackgroundTransparent (Boolean)**: setzt die Hintergrundfarbe auf transparent.
- **ShadowFormat (Struct)**: legt den Schatten für Zellen fest (Struktur gemäß `com.sun.star.table.ShadowFormat`).

Die Struktur `com.sun.star.table.ShadowFormat` sowie die Detailangaben für Zellschatten sind wie folgt aufgebaut:

- **Location (Enum)**: Position des Schattens (Wert aus der Struktur `com.sun.star.table.ShadowLocation`).
- **ShadowWidth (Short)**: Größe des Schattens in 100stel Millimeter.
- **IsTransparent (Boolean)**: setzt den Schatten auf transparent.
- **Color (Long)**: Farbe des Schattens.

Das folgende Beispiel schreibt die Zahl 1000 in die Zelle B2, ändert die Hintergrundfarbe über die Eigenschaft `CellBackColor` zu Rot und legt dann einen hellgrauen Schatten für die Zelle an, der um 1 mm nach links unten verschoben ist.

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object
Dim ShadowFormat As New com.sun.star.table.ShadowFormat

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
Cell = Sheet.getCellByPosition(1,1)

Cell.Value = 1000

Cell.CellBackColor = RGB(255, 0, 0)

ShadowFormat.Location = com.sun.star.table.ShadowLocation.BOTTOM_RIGHT
ShadowFormat.ShadowWidth = 100
ShadowFormat.Color = RGB(160, 160, 160)

Cell.ShadowFormat = ShadowFormat
```

Ausrichtung

StarOffice bietet verschiedene Funktionen, mit denen die Ausrichtung von Text in einer Tabellenzelle geändert werden kann.

Zur Definition der horizontalen und vertikalen Ausrichtung eines Texts stehen folgende Eigenschaften zur Verfügung:

- **HoriJustify (Enum)**: horizontale Ausrichtung des Texts (Wert aus `com.sun.star.table.CellHoriJustify`).
- **VertJustify (Enum)**: vertikale Ausrichtung des Texts (Wert aus `com.sun.star.table.CellVertJustify`).
- **Orientation (Enum)**: Laufrichtung des Texts (Wert gemäß `com.sun.star.table.CellOrientation`).
- **IsTextWrapped (Boolean)**: lässt automatische Zeilenumbrüche innerhalb der Zelle zu.
- **RotateAngle (Long)**: Drehwinkel des Texts in 100stel Grad.

Das folgende Beispiel zeigt, wie der Inhalt einer Zelle "gestapelt" werden kann, so dass die einzelnen Zeichen in der linken oberen Ecke der Zelle eins unter dem anderen angezeigt werden. Die Zeichen werden hierbei nicht gedreht.

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
Cell = Sheet.getCellByPosition(1,1)

Cell.Value = 1000

Cell.HoriJustify = com.sun.star.table.CellHoriJustify.LEFT
Cell.VertJustify = com.sun.star.table.CellVertJustify.TOP
Cell.Orientation = com.sun.star.table.CellOrientation.STACKED
```

Zahlen-, Datums- und Textformat

StarOffice bietet eine ganze Reihe von vordefinierten Datums- und Uhrzeitformaten. Jedes dieser Formate verfügt über eine interne Nummer, über die das jeweilige Format mit der Eigenschaft `NumberFormat` den Zellen zugewiesen wird. StarOffice stellt die Methoden `queryKey` und `addNew` zur Verfügung, mit denen Sie auf vorhandene Zahlenformate zugreifen sowie eigene erstellen können. Der Zugriff auf die Methoden erfolgt mit folgendem Objekt-Aufruf:

```
NumberFormats = Doc.NumberFormats
```

Ein Format wird über einen Format-String angegeben, der ähnlich aufgebaut ist wie die Format-Funktion von StarOffice Basic. Es existiert jedoch ein wesentlicher Unterschied: Während der Format-Befehl immer englische Abkürzungen und Dezimal- beziehungsweise Tausendertrennzeichen erwartet, müssen in der Syntax eines Format-Befehls für das `NumberFormats`-Objekt die landesspezifischen Abkürzungen verwendet werden.

Das folgende Beispiel formatiert die Zelle B2 so, dass Zahlen mit drei Dezimalstellen und Kommas als Tausendertrennzeichen angezeigt werden.

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object
Dim NumberFormats As Object
Dim NumberFormatString As String
Dim NumberFormatId As Long
Dim LocalSettings As New com.sun.star.lang.Locale

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
Cell = Sheet.getCellByPosition(1,1)

Cell.Value = 23400.3523565

LocalSettings.Language = "en"
LocalSettings.Country = "us"

NumberFormats = Doc.NumberFormats
NumberFormatString = "#,##0.000"

NumberFormatId = NumberFormats.queryKey(NumberFormatString, LocalSettings, True)
If NumberFormatId = -1 Then
   NumberFormatId = NumberFormats.addNew(NumberFormatString, LocalSettings)
End If

MsgBox NumberFormatId
Cell.NumberFormat = NumberFormatId
```

In dem Dialogfeld **Zellen formatieren** in StarOffice Calc erhalten Sie einen Überblick über die verschiedenen Formatierungsoptionen für Zellen.

Seiteneigenschaften

Seiteneigenschaften sind die Formatierungsoptionen, die sowohl den Dokumentinhalt auf einer Seite positionieren als auch die visuellen Elemente, die auf allen Seiten wiederholt werden. Hierzu zählen unter anderem

- Papierformate
- Seitenränder
- Kopf- und Fußzeilen.

Das Vorgehen bei der Definition von Seitenformaten unterscheidet sich von anderen Arten der Formatierung. Während Zell-, Absatz- und Zeichenelemente direkt formatiert werden können,

können Seitenformatierungen auch definiert und unter Verwendung von Seitenvorlagen indirekt angewendet werden. So werden beispielsweise Kopf- und Fußzeilen in eine Seitenvorlage aufgenommen.

In den folgenden Abschnitten werden die Hauptformatierungsoptionen für Tabellendokumentseiten beschrieben. Viele der beschriebenen Vorlagen stehen auch für Textdokumente zur Verfügung. Die Seiteneigenschaften, die für beide Dokumentarten gültig sind, sind im Dienst `com.sun.star.style.PageProperties` definiert. Die Seiteneigenschaften, die nur für Tabellendokumente gültig sind, sind im Dienst `com.sun.star.sheet.TablePageStyle` definiert.

Hinweis – Die Seiteneigenschaften (Seitenränder, Rahmen usw.) für ein Microsoft Office-Dokument werden auf der Ebene der Objekte `Worksheet` (Excel) beziehungsweise `Document` (Word) über ein `PageSetup`-Objekt definiert. In StarOffice erfolgt die Definition der Seiteneigenschaften hingegen über eine Seitenvorlage, die wiederum mit dem zugehörigen Dokument verknüpft ist.

Seitenhintergrund

Der Dienst `Dcom.sun.star.style.PageProperties` definiert folgende Eigenschaften für einen Seitenhintergrund:

- **BackColor (Long)**: Farbe des Hintergrunds.
- **BackGraphicURL (String)**: URL der zu verwendenden Hintergrundgrafik.
- **BackGraphicFilter (String)**: Name des Filters für die Interpretation von Hintergrundgrafiken.
- **BackGraphicLocation (Enum)**: Position der Hintergrundgrafiken (Wert gemäß der Aufzählung).
- **BackTransparent (Boolean)**: stellt den Hintergrund transparent dar.

Seitenformat

Die Definition des Seitenformats erfolgt über folgende Eigenschaften des Dienstes `com.sun.star.style.PageProperties`:

- **IsLandscape (Boolean)**: Querformat.
- **Width (Long)**: Breite der Seite in 100stel Millimeter.
- **Height (Long)**: Höhe der Seite in 100stel Millimeter.
- **PrinterPaperTray (String)**: Name des zu verwendenden Papierschachts des Druckers.

Das folgende Beispiel setzt die Seitengröße der Seitenvorlage "Default" (Standard) auf das Format DIN A5 quer (Höhe 14,8 cm, Breite 21 cm):

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.IsLandscape = True
DefPage.Width = 21000
DefPage.Height = 14800
```

Seitenrand, -rahmen und -schatten

Der Dienst `com.sun.star.style.PageProperties` stellt folgende Eigenschaften zum Anpassen von Seitenrändern sowie Rahmen und Schatten bereit:

- **LeftMargin (Long)**: Breite des linken Seitenrands in 100stel Millimeter.
- **RightMargin (Long)**: Breite des rechten Seitenrands in 100stel Millimeter.
- **TopMargin (Long)**: Breite des oberen Seitenrands in 100stel Millimeter.
- **BottomMargin (Long)**: Breite des unteren Seitenrands in 100stel Millimeter.
- **LeftBorder (Struct)**: Angaben für die linke Linie des Seitenrahmens (Struktur `com.sun.star.table.BorderLine`).
- **RightBorder (Struct)**: Angaben für die rechte Linie des Seitenrahmens (Struktur `com.sun.star.table.BorderLine`).
- **TopBorder (Struct)**: Angaben für die obere Linie des Seitenrahmens (Struktur `com.sun.star.table.BorderLine`).
- **BottomBorder (Struct)**: Angaben für die untere Linie des Seitenrahmens (Struktur `com.sun.star.table.BorderLine`).
- **LeftBorderDistance (Long)**: Abstand zwischen linkem Seitenrahmen und Seiteninhalt in 100stel Millimeter.
- **RightBorderDistance (Long)**: Abstand zwischen rechtem Seitenrahmen und Seiteninhalt in 100stel Millimeter.
- **TopBorderDistance (Long)**: Abstand zwischen oberem Seitenrahmen und Seiteninhalt in 100stel Millimeter.
- **BottomBorderDistance (Long)**: Abstand zwischen unterem Seitenrahmen und Seiteninhalt in 100stel Millimeter.
- **ShadowFormat (Struct)**: Angaben zum Schatten des Inhaltsbereichs der Seite (Struktur `(com.sun.star.table.ShadowFormat`).

Das folgende Beispiel setzt den linken und rechten Rand der Seitenvorlage "Default" (Standard) auf 1 Zentimeter.

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.LeftMargin = 1000
DefPage.RightMargin = 1000
```

Kopf- und Fußzeilen

Die Kopf- und Fußzeilen eines Dokuments gehören zu den Seiteneigenschaften und werden über den Dienst `com.sun.star.style.PageProperties` definiert. Die Eigenschaften zum Formatieren von Kopfzeilen sind:

- **HeaderIsOn (Boolean)**: Kopfzeile ist aktiviert.
- **HeaderLeftMargin (Long)**: Abstand zwischen Kopfzeile und linkem Seitenrand in 100stel Millimeter.
- **HeaderRightMargin (Long)**: Abstand zwischen Kopfzeile und rechtem Seitenrand in 100stel Millimeter.
- **HeaderBodyDistance (Long)**: Abstand zwischen Kopfzeile und Hauptkörper des Dokuments in 100stel Millimeter.
- **HeaderHeight (Long)**: Höhe der Kopfzeile in 100stel Millimeter.
- **HeaderIsDynamicHeight (Boolean)**: Höhe der Kopfzeile wird automatisch an den Inhalt angepasst.
- **HeaderLeftBorder (Struct)**: Angaben zur linken Linie des Rahmens um die Kopfzeile (Struktur `com.sun.star.table.BorderLine`).
- **HeaderRightBorder (Struct)**: Angaben zur rechten Linie des Rahmens um die Kopfzeile (Struktur `com.sun.star.table.BorderLine`).
- **HeaderTopBorder (Struct)**: Angaben zur oberen Linie des Rahmens um die Kopfzeile (Struktur `com.sun.star.table.BorderLine`).
- **HeaderBottomBorder (Struct)**: Angaben zur unteren Linie des Rahmens um die Kopfzeile (Struktur `com.sun.star.table.BorderLine`).
- **HeaderLeftBorderDistance (Long)**: Abstand zwischen linkem Rahmen und Inhalt der Kopfzeile in 100stel Millimeter.

- **HeaderRightBorderDistance (Long)**: Abstand zwischen rechtem Rahmen und Inhalt der Kopfzeile in 100stel Millimeter.
- **HeaderTopBorderDistance (Long)**: Abstand zwischen oberem Rahmen und Inhalt der Kopfzeile in 100stel Millimeter.
- **HeaderBottomBorderDistance (Long)**: Abstand zwischen unterem Rahmen und Inhalt der Kopfzeile in 100stel Millimeter.
- **HeaderIsShared (Boolean)**: Kopfzeilen auf geraden und ungeraden Seiten haben denselben Inhalt (siehe `HeaderText` , `HeaderTextLeft` und `HeaderTextRight`).
- **HeaderBackColor (Long)**: Hintergrundfarbe der Kopfzeile.
- **HeaderBackGraphicURL (String)**: URL der zu verwendenden Hintergrundgrafik.
- **HeaderBackGraphicFilter (String)**: Name des Filters für die Interpretation von Hintergrundgrafiken für die Kopfzeile.
- **HeaderBackGraphicLocation (Enum)**: Position der Hintergrundgrafiken für die Kopfzeile (Wert gemäß der Aufzählung`com.sun.star.style.GraphicLocation`).
- **HeaderBackTransparent (Boolean)**: zeigt den Hintergrund der Kopfzeile transparent an.
- **HeaderShadowFormat (Struct)**: Angaben zum Schatten der Kopfzeile (Struktur `(com.sun.star.table.ShadowFormat`).

Die Eigenschaften zum Formatieren von Fußzeilen sind:

- **FooterIsOn (Boolean)**: Fußzeile ist aktiviert.
- **FooterLeftMargin (Long)**: Abstand zwischen Fußzeile und linkem Seitenrand in 100stel Millimeter.
- **FooterRightMargin (Long)**: Abstand zwischen Fußzeile und rechtem Seitenrand in 100stel Millimeter.
- **FooterBodyDistance (Long)**: Abstand zwischen Fußzeile und Hauptkörper des Dokuments in 100stel Millimeter.
- **FooterHeight (Long)**: Höhe der Fußzeile in 100stel Millimeter.
- **FooterIsDynamicHeight (Boolean)**: Höhe der Fußzeile wird automatisch an den Inhalt angepasst.
- **FooterLeftBorder (Struct)**: Angaben zur linken Linie des Rahmens um die Fußzeile (Struktur `com.sun.star.table.BorderLine`).
- **FooterRightBorder (Struct)**: Angaben zur rechten Linie des Rahmens um die Fußzeile (Struktur `com.sun.star.table.BorderLine`).
- **FooterTopBorder (Struct)**: Angaben zur oberen Linie des Rahmens um die Fußzeile (Struktur `com.sun.star.table.BorderLine`).
- **FooterBottomBorder (Struct)**: Angaben zur unteren Linie des Rahmens um die Fußzeile (Struktur `com.sun.star.table.BorderLine`).

- **FooterLeftBorderDistance (Long)**: Abstand zwischen linkem Rahmen und Inhalt der Fußzeile in 100stel Millimeter.
- **FooterRightBorderDistance (Long)**: Abstand zwischen rechtem Rahmen und Inhalt der Fußzeile in 100stel Millimeter.
- **FooterTopBorderDistance (Long)**: Abstand zwischen oberem Rahmen und Inhalt der Fußzeile in 100stel Millimeter.
- **FooterBottomBorderDistance (Long)**: Abstand zwischen unterem Rahmen und Inhalt der Fußzeile in 100stel Millimeter.
- **FooterIsShared (Boolean)**: Fußzeilen auf geraden und ungeraden Seiten haben denselben Inhalt (siehe `FooterText` , `FooterTextLeft` und `FooterTextRight`).
- **FooterBackColor (Long)**: Hintergrundfarbe der Fußzeile.
- **FooterBackGraphicURL (String)**: URL der zu verwendenden Hintergrundgrafik.
- **FooterBackGraphicFilter (String)**: Name des Filters für die Interpretation von Hintergrundgrafiken für die Fußzeile.
- **FooterBackGraphicLocation (Enum)**: Position der Hintergrundgrafiken für die Fußzeile (Wert gemäß der Aufzählung`com.sun.star.style.GraphicLocation`).
- **FooterBackTransparent (Boolean)**: zeigt den Hintergrund der Fußzeile transparent an.
- **FooterShadowFormat (Struct)**: Angaben zum Schatten der Fußzeile (Struktur `(com.sun.star.table.ShadowFormat`).

Ändern des Texts in Kopf- und Fußzeilen

Auf den Inhalt von Kopf- und Fußzeilen in einem Tabellendokument kann über folgende Eigenschaften zugegriffen werden:

- **LeftPageHeaderContent (Object)**: Inhalt der Kopfzeilen für gerade Seiten (Dienst `com.sun.star.sheet.HeaderFooterContent`).
- **RightPageHeaderContent (Object)**: Inhalt der Kopfzeilen für ungerade Seiten (Dienst `com.sun.star.sheet.HeaderFooterContent`).
- **LeftPageFooterContent (Object)**: Inhalt der Fußzeilen für gerade Seiten (Dienst `com.sun.star.sheet.HeaderFooterContent`).
- **RightPageFooterContent (Object)**: Inhalt der Fußzeilen für ungerade Seiten (Dienst `com.sun.star.sheet.HeaderFooterContent`).

Wenn Sie keine Unterscheidung zwischen Kopf- und Fußzeilen für gerade und ungerade Seiten benötigen (die Eigenschaft `FooterIsShared` hat den Wert `False`), setzen Sie die Eigenschaften für Kopf- und Fußzeilen auf ungeraden Seiten.

Alle genannten Objekte geben ein Objekt zurück, das den Dienst `com.sun.star.sheet.HeaderFooterContent` unterstützt. Dieser Dienst stellt über die (unechten) Eigenschaften `LeftText`, `CenterText` und `RightText` drei Textelemente für die Kopf- und Fußzeilen von StarOffice Calc zur Verfügung.

Das folgende Beispiel schreibt in das linke Textfeld der Kopfzeile der Vorlage "Default" (Standard) den Wert "Nur ein Test.".

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object
Dim HText As Object
Dim HContent As Object
Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.HeaderIsOn = True
HContent = DefPage.RightPageHeaderContent
HText = HContent.LeftText
HText.String = "Nur ein Test."
DefPage.RightPageHeaderContent = HContent
```

Beachten Sie die letzte Zeile des Beispiels: Nach dem Ändern des Texts muss das `TextContent`-Objekt erneut der Kopfzeile zugewiesen werden, damit die Änderung wirksam wird.

Für Textdokumente (StarOffice Writer) steht ein anderer Mechanismus zum Ändern der Texte in Kopf- und Fußzeilen zur Verfügung, da diese aus einem einzigen Textblock bestehen. Folgende Eigenschaften sind im Dienst `com.sun.star.style.PageProperties` definiert:

- **HeaderText (Object)** : Textobjekt mit dem Inhalt der Kopfzeile (Dienst `com.sun.star.text.XText`).
- **HeaderTextLeft (Object)** : Textobjekt mit dem Inhalt der Kopfzeilen auf linken (geraden) Seiten (Dienst `com.sun.star.text.XText`).
- **HeaderTextRight (Object)** : Textobjekt mit dem Inhalt der Kopfzeilen auf rechten (ungeraden) Seiten (Dienst `com.sun.star.text.XText`).
- **FooterText (Object)** : Textobjekt mit dem Inhalt der Fußzeile (Dienst `com.sun.star.text.XText`).
- **FooterTextLeft (Object)** : Textobjekt mit dem Inhalt der Fußzeilen auf linken (geraden) Seiten (Dienst `com.sun.star.text.XText`).
- **FooterTextRight (Object)** : Textobjekt mit dem Inhalt der Fußzeilen auf rechten (ungeraden) Seiten (Dienst `com.sun.star.text.XText`).

Das folgende Beispiel erstellt in der Vorlage "Default" (Standard) für Textdokumente eine Kopfzeile und fügt dieser den Text "Nur ein Test." hinzu.

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object
Dim HText As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.HeaderIsOn = True
HText = DefPage.HeaderText

HText.String = "Nur ein Test."
```

Der Zugriff erfolgt in diesem Fall statt über das `HeaderFooterContent`-Objekt direkt über die Eigenschaft `HeaderText` der Seitenvorlage.

Zentrieren (nur Tabellendokumente)

Der ausschließlich in den Seitenvorlagen von StarOffice Calc zum Einsatz kommende Dienst `com.sun.star.sheet.TablePageStyle` gestattet es, zu druckende Zellbereiche auf der Seite zu zentrieren. Dieser Dienst stellt folgende Eigenschaften zur Verfügung:

- **CenterHorizontally (Boolean)**: Tabelleninhalt wird horizontal zentriert.
- **CenterVertically (Boolean)**: Tabelleninhalt wird vertikal zentriert.

Definieren von auszudruckenden Elementen (nur Tabellendokumente)

Beim Formatieren von Tabellen können Sie definieren, ob Seitenelemente angezeigt werden sollen. Hierzu stellt der Dienst `com.sun.star.sheet.TablePageStyle` folgende Eigenschaften bereit:

- **PrintAnnotations (Boolean)**: druckt Zellkommentare mit aus.
- **PrintGrid (Boolean)**: druckt die Zellgitterlinien mit aus.
- **PrintHeaders (Boolean)**: druckt die Zeilen- und Spaltenüberschriften mit aus.
- **PrintCharts (Boolean)**: druckt in einer Tabelle enthaltene Diagramme mit aus.
- **PrintObjects (Boolean)**: druckt eingebettete Objekte mit aus.
- **PrintDrawing (Boolean)**: druckt Zeichnungsobjekte mit aus.
- **PrintDownFirst (Boolean)**: druckt bei mehrseitigen Tabellen die Seiten zuerst in vertikaler Reihenfolge von oben nach unten und dann die rechts anschließenden Seiten von oben nach unten.
- **PrintFormulas (Boolean)**: druckt statt der berechneten Werte die Formeln aus.

- **PrintZeroValues (Boolean)**: druckt Nullwerte mit aus.

Effizientes Bearbeiten von Tabellendokumenten

Während im vorangegangenen Abschnitt der grundlegende Aufbau von Tabellendokumenten beschrieben wurde, behandelt der vorliegende Abschnitt die Dienste, mit denen Sie unkompliziert auf einzelne Zellen oder Zellbereiche zugreifen können.

Zellbereiche

Neben einem Objekt für Einzelzellen (Dienst `com.sun.star.table.Cell`) stellt StarOffice außerdem Objekte zur Verfügung, die Zellbereiche darstellen. Die Erstellung solcher `CellRange`-Objekte erfolgt über den Aufruf `getCellRangeByName` des Spreadsheet-Objekts:

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
CellRange = Sheet.getCellRangeByName("A1:C15")
```

Mit einem Doppelpunkt (:) wird ein Zellbereich in einem Tabellendokument festgelegt. So stellt beispielsweise A1:C15 alle Zellen in den Zeilen 1 bis 15 und in den Spalten A, B und C dar.

Die Position einzelner Zellen innerhalb eines Zellbereichs lässt sich über die Methode `getCellByPosition` ermitteln, wobei die Koordinaten der obersten linken Zelle des Zellbereichs die Position (0, 0) besitzt. Das folgende Beispiel erstellt mit dieser Methode ein Objekt der Zelle C3.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
CellRange = Sheet.getCellRangeByName("B2:D4")
Cell = CellRange.GetCellByPosition(1, 1)
```

Formatieren von Zellbereichen

Wie bei einzelnen Zellen können Sie Formatierungen mit dem Dienst `com.sun.star.table.CellProperties` auch auf Zellbereiche anwenden. Weitere Informationen und Beispiele zu diesem Dienst finden sich im Abschnitt *Formatieren*.

Rechnen mit Zellbereichen

Mit der Methode `computeFunction` können Sie mathematische Operationen an Zellbereichen vornehmen. `ComputeFunction` erwartet eine Konstante als Parameter, der die anzuwendende mathematische Funktion beschreibt. Die verknüpften Konstanten sind in der Aufzählung `com.sun.star.sheet.GeneralFunction` definiert. Folgende Werte stehen zur Verfügung:

- **SUM**: Summe aller numerischen Werte.
- **COUNT**: Gesamtanzahl aller Werte (einschließlich nicht numerischer Werte).
- **COUNTNUMS**: Gesamtanzahl aller numerischen Werte.
- **AVERAGE**: Durchschnitt aller numerischen Werte.
- **MAX**: größter numerischer Wert.
- **MIN**: kleinster numerischer Wert.
- **PRODUCT**: Produkt aller numerischen Werte.
- **STDEV**: Standardabweichung.
- **VAR**: Varianz.
- **STDEVP**: Standardabweichung auf der Grundlage der Gesamtpopulation.
- **VARP**: Varianz auf der Grundlage der Gesamtpopulation.

Das folgende Beispiel berechnet den Mittelwert des Bereichs `A1:C3` und gibt das Ergebnis in einem Meldungsfenster aus:

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
CellRange = Sheet.getCellRangeByName("A1:C3")

MsgBox CellRange.computeFunction(com.sun.star.sheet.GeneralFunction.AVERAGE)
```

Löschen von Zellinhalten

Die Methode `clearContents` vereinfacht den Vorgang des Löschens von Zellinhalten und Zellbereichen, indem sie einen bestimmten Inhaltstyp aus einem Zellbereich löscht.

Das folgende Beispiel entfernt alle Zeichenfolgen sowie die direkten Formatierungsinformationen aus dem Bereich `B2:C3`.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object
Dim Flags As Long

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
```

```
CellRange = Sheet.getCellRangeByName("B2:C3")

Flags = com.sun.star.sheet.CellFlags.STRING + _
      com.sun.star.sheet.CellFlags.HARDATTR

CellRange.clearContents(Flags)
```

Die in `clearContents` angegebenen Flags stammen aus der Konstantenliste `com.sun.star.sheet.CellFlags`. Diese Liste stellt folgende Elemente bereit:

- **VALUE**: numerische Werte, die nicht als Datum oder Zeit formatiert sind.
- **DATETIME**: numerische Werte, die als Datum oder Zeit formatiert sind.
- **STRING**: Zeichenfolgen.
- **ANNOTATION**: mit Zellen verknüpfte Kommentare.
- **FORMULA**: Formeln.
- **HARDATTR**: direkte Formatierungen von Zellen.
- **STYLES**: indirekte Formatierungen.
- **OBJECTS**: mit Zellen verknüpfte Zeichnungsobjekte.
- **EDITATTR**: Zeichenformatierungen, die nur auf Teile der Zellen Anwendung finden.

Die Konstanten können auch aufaddiert werden, um unter Verwendung eines Aufrufs von `clearContents` verschiedene Informationen zu löschen.

Suchen und Ersetzen von Zellinhalten

Tabellendokumente bieten ähnlich wie Textdokumente eine Funktion zum Suchen und Ersetzen.

Die Deskriptor-Objekte zum Suchen und Ersetzen werden in Tabellendokumenten jedoch nicht direkt über das Dokumentobjekt erzeugt, sondern über die `Sheets`-Liste. Im Folgenden finden Sie ein Beispiel für einen Suchen- und Ersetzen-Vorgang:

```
Dim Doc As Object
Dim Sheet As Object
Dim ReplaceDescriptor As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

ReplaceDescriptor = Sheet.createReplaceDescriptor()
ReplaceDescriptor.SearchString = "is"
ReplaceDescriptor.ReplaceString = "was"

For I = 0 to Doc.Sheets.Count - 1
   Sheet = Doc.Sheets(I)
```

```
   Sheet.ReplaceAll(ReplaceDescriptor)
Next I
```

Dieses Beispiel erstellt mit Hilfe der ersten Seite des Dokuments einen `ReplaceDescriptor` und wendet diesen dann in einer Schleife auf alle Seiten an.

◆ ◆ ◆ KAPITEL 8

8 Zeichnungen und Präsentationen

Dieses Kapitel gibt eine Einführung in die makrogesteuerte Erstellung und Bearbeitung von Zeichnungen. Im ersten Abschnitt wird der Aufbau von Zeichnungen, einschließlich der darin enthaltenen grundlegenden Elemente, beschrieben. Im zweiten Abschnitt werden komplexere Bearbeitungsfunktionen wie das Gruppieren, Drehen und Skalieren von Objekten behandelt.

Informationen zum Erstellen, Öffnen und Speichern von Zeichnungen finden Sie in Kapitel 5, *Arbeiten mit StarOffice-Dokumenten*.

Der Aufbau von Zeichnungen

In StarOffice ist die Seitenanzahl, über die ein Zeichnungsdokument verfügen kann, unbegrenzt. Sie können dabei jede Seite individuell entwerfen. Auch die Anzahl von Zeichnungselementen, die einer Seite hinzugefügt werden können, ist unbegrenzt.

Dieser Eindruck wird durch das Vorhandensein von *Ebenen* (Layer) unwesentlich kompliziert. Standardmäßig enthält jedes Zeichnungsdokument die Ebenen *Layout*, *Steuerelemente* (Controls) und *Maßlinien* (Dimension Lines), wobei alle Zeichnungselemente der Ebene *Layout* hinzugefügt werden. Es besteht auch die Möglichkeit, neue Ebenen hinzuzufügen. Weitere Informationen zu Zeichnungsebenen finden Sie im "StarOffice Developer's Guide".

Seiten

Die Seiten eines Zeichnungsdokuments sind über die Liste `DrawPages` verfügbar. Auf einzelne Seiten können Sie wahlweise über ihre Nummer oder ihren Namen zugreifen. Enthält ein Dokument eine Seite und heißt diese *Slide 1*, so sind die folgenden Beispiele identisch.

Beispiel 1:

```
Dim Doc As Object
Dim Page As Object
```

```
Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
```

Beispiel 2:

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages.getByName("Slide 1")
```

In Beispiel 1 wird die Seite über ihre Nummer adressiert (die Zählung beginnt hier mit 0). Im zweiten Beispiel erfolgt der Zugriff über den Namen mit der Methode `getByName`.

```
Dim sUrl As String, sFilter As String
Dim sOptions As String
Dim oSheets As Object, oSheet As Object

   oSheets = oDocument.Sheets

   If oSheets.hasByName("Link") Then
      oSheet = oSheets.getByName("Link")
   Else
      oSheet = oDocument.createInstance("com.sun.star.sheet.Spreadsheet")
      oSheets.insertByName("Link", oSheet)
      oSheet.IsVisible = False
   End If
```

Der vorangehende Aufruf gibt ein Seitenobjekt zurück, das den Dienst `com.sun.star.drawing.DrawPage` unterstützt. Der Dienst erkennt folgende Eigenschaften:

- **BorderLeft (Long)**: linker Seitenrand in 100stel Millimeter.
- **BorderRight (Long)**: rechter Seitenrand in 100stel Millimeter.
- **BorderTop (Long)**: oberer Seitenrand in 100stel Millimeter.
- **BorderBottom (Long)**: unterer Seitenrand in 100stel Millimeter.
- **Width (Long)**: Seitenbreite in 100stel Millimeter.
- **Height (Long)**: Seitenhöhe in 100stel Millimeter.
- **Number (Short)**: Nummer der Seite (Nummerierung beginnt bei 1); schreibgeschützt (nur Lesen).
- **Orientation (Enum)**: Seitenausrichtung (gemäß der Aufzählung `com.sun.star.view.PaperOrientation`).

Werden diese Einstellungen geändert, so wirkt sich dies auf *alle* Seiten des Dokuments aus.

Das folgende Beispiel setzt die Seitengröße des gerade geöffneten Zeichnungsdokuments auf 20 x 20 cm bei einem Seitenrand von jeweils 0,5 cm:

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

Page.BorderLeft = 500
Page.BorderRight = 500
Page.BorderTop = 500
Page.BorderBottom = 500

Page.Width = 20000
Page.Height = 20000
```

Elementare Eigenschaften von Zeichnungsobjekten

Zu den Zeichnungsobjekte gehören Formen (Rechtecke, Kreise usw.), Linien und Textobjekte. All diese Objekte besitzen eine Reihe gemeinsamer Eigenschaften und unterstützen den Dienst `com.sun.star.drawing.Shape`. Dieser Dienst definiert die Eigenschaften `Size` und `Position` für ein Zeichnungsobjekt.

StarOffice Basic bietet außerdem zahlreiche andere Dienste, mit denen solche Eigenschaften geändert werden können, beispielsweise Formatierung oder Anwenden von Füllungen. Welche Formatierungsoptionen zur Verfügung stehen, hängt vom Typ des Zeichnungsobjekts ab.

Im folgenden Beispiel wird ein Rechteck erstellt und in ein Zeichnungsobjekt eingefügt:

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
```

```
RectangleShape.Position = Point

Page.add(RectangleShape)
```

Das Beispiel ermittelt über den Aufruf `StarDesktop.CurrentComponent` das aktuell geöffnete Dokument. Das so bestimmte Dokumentobjekt gibt über den Aufruf `drawPages(0)` die erste Seite der Zeichnung zurück.

Im Anschluss daran werden die Strukturen `Point` und `Size` mit dem Ursprungspunkt (linke obere Ecke) und der Größe des Zeichnungsobjekts initialisiert. Die Längen werden dabei in 100stel Millimeter angegeben.

Der Programmcode erstellt dann mit dem Aufruf `Doc.createInstance` das Rechteck-Zeichnungsobjekt, wie es vom Dienst `com.sun.star.drawing.RectangleShape` festgelegt ist. Schließlich wird das Zeichnungsobjekt einer Seite mit einem Aufruf `Page.add` zugewiesen.

Fülleigenschaften

In diesem Abschnitt werden vier Dienste beschrieben, wobei der Beispielprogrammcode jedes Mal ein Rechteck-Element (RectangleShape) verwendet, das mehrere Formatierungsarten in sich vereint. Fülleigenschaften sind in dem Dienst `com.sun.star.drawing.FillProperties` zusammengefasst.

StarOffice erkennt vier Hauptarten der Formatierung eines Füllbereichs. Die einfachste Variante stellt eine einfarbige Füllung dar. Die Möglichkeiten zur Definition von Farbverläufen und Schraffuren ermöglichen Ihnen die Erzeugung zusätzlicher Farben. Schließlich besteht als vierte Variante die Möglichkeit, bereits vorhandene Grafiken in den Füllbereich hinein zu projizieren.

Der Füllmodus eines Zeichnungsobjekts wird über die Eigenschaft `FillStyle` definiert. Die zugelassenen Werte sind in `com.sun.star.drawing.FillStyle` definiert.

Einfarbige Füllungen

Die Haupteigenschaft für einfarbige Füllungen lautet

- **FillColor (Long)**: Füllfarbe des Bereichs.

Um den Füllmodus zu verwenden, muss die Eigenschaft `FillStyle` dem Füllmodus `SOLID` zugewiesen werden.

Das folgende Beispiel erzeugt ein Rechteck und füllt es mit Rot (RGB-Wert 255, 0, 0):

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
```

```
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

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.SOLID
RectangleShape.FillColor = RGB(255,0,0)

Page.add(RectangleShape)
```

Farbverläufe

Wenn die Eigenschaft `FillStyle` auf `GRADIENT` gesetzt wird, kann auf jeden Füllbereich eines StarOffice-Dokuments ein Farbverlauf angewendet werden.

Wenn ein vordefinierter Farbverlauf angewendet werden soll, kann der verknüpfte Name der Eigenschaft `FillTransparenceGradientName` zugewiesen werden. Um einen eigenen Farbverlauf zu definieren, müssen Sie eine `com.sun.star.awt.Gradient`-Struktur vervollständigen, um die Eigenschaft `FillGradient` zuzuweisen. Diese Eigenschaft stellt folgende Optionen bereit:

- **Style (Enum)**: Verlaufsart, beispielsweise linear oder radial (Standardwerte gemäß `com.sun.star.awt.GradientStyle`).
- **StartColor (Long)**: Anfangsfarbe des Farbverlaufs.
- **EndColor (Long)**: Endfarbe des Farbverlaufs.
- **Angle (Short)**: Winkel des Farbverlaufs in 10tel Grad.
- **XOffset (Short)**: X-Koordinate, bei der der Farbverlauf beginnt, angegeben in 100stel Millimeter.
- **YOffset (Short)**: Y-Koordinate, bei der der Farbverlauf beginnt, angegeben in 100stel Millimeter.
- **StartIntensity (Short)**: Intensität von `StartColor` als Prozentsatz (in StarOffice Basic können auch Werte angegeben werden, die größer als 100 Prozent sind).
- **EndIntensity (Short)**: Intensität von `EndColor` als Prozentsatz (in StarOffice Basic können auch Werte angegeben werden, die größer als 100 Prozent sind).

- **StepCount (Short)**: Anzahl der Farbabstufungen, die StarOffice für den Farbverlauf berechnen soll.

Das folgende Beispiel demonstriert den Einsatz von Farbverläufen unter Zuhilfenahme der Struktur `com.sun.star.awt.Gradient`:

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
Dim Gradient As New com.sun.star.awt.Gradient

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point
Gradient.Style = com.sun.star.awt.GradientStyle.LINEAR
Gradient.StartColor = RGB(255,0,0)
Gradient.EndColor = RGB(0,255,0)
Gradient.StartIntensity = 150
Gradient.EndIntensity = 150
Gradient.Angle = 450
Gradient.StepCount = 100

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.GRADIENT
RectangleShape.FillGradient = Gradient

Page.add(RectangleShape)
```

In diesem Beispiel wird ein linearer Farbverlauf erstellt (`Style = LINEAR`). Der Farbverlauf beginnt mit Rot (`StartColor`) in der linken oberen Ecke und verläuft in einem Winkel von 45° (`Angle`) nach Grün (`EndColor`) in der rechten unteren Ecke. Die Farbintensität der Start- und Endfarben beträgt jeweils 150 Prozent (`StartIntensity` und `EndIntensity`), was dazu führt, dass die Farben heller erscheinen, als die in den Eigenschaften `StartColor` und `EndColor` angegebenen Werte. Die Darstellung des Farbverlaufs erfolgt über 100 abgestufte Einzelfarben (`StepCount`).

Schraffuren

Um eine Schraffurfüllung zu erzeugen, muss die Eigenschaft `FillStyle` auf `HATCH` gesetzt werden. Der Programmcode zur Definition der Schraffur ähnelt stark dem für Farbverläufe. Wieder wird eine Hilfsstruktur, hier `com.sun.star.drawing.Hatch`, verwendet, um das Aussehen der Schraffuren zu definieren. Die Struktur für Schraffuren verfügt über folgende Eigenschaften:

- **Style (Enum)**: Art der Schraffur: einfach, kariert oder kariert mit Diagonalen (Standardwerte gemäß `com.sun.star.awt.HatchStyle`).
- **Color (Long)**: Farbe der Linien.
- **Distance (Long)**: Abstand zwischen den Linien in 100stel Millimeter.
- **Angle (Short)**: Winkel der Schraffur in 10tel Grad.

Das folgende Beispiel demonstriert den Einsatz einer Schraffurstruktur (Hatch):

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
Dim Hatch As New com.sun.star.drawing.Hatch

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.HATCH

Hatch.Style = com.sun.star.drawing.HatchStyle.SINGLE
Hatch.Color = RGB(64,64,64)
Hatch.Distance = 20
Hatch.Angle = 450

RectangleShape.FillHatch = Hatch

Page.add(RectangleShape)
```

Dieser Code erzeugt eine einfache Schraffur (`HatchStyle = SINGLE`), deren Linien um 45° gedreht sind (`Angle`). Die Linien sind dunkelgrau (`Color`) mit einem Abstand (`Distance`) von 0,2 mm.

Bitmaps

Um eine Bitmap-Projektion als Füllung zu verwenden, muss die Eigenschaft `FillStyle` auf `BITMAP` gesetzt werden. Wenn die Bitmap in StarOffice bereits zur Verfügung steht, genügt es, deren Namen in der Eigenschaft `FillBitMapName` einzugeben und die gewünschte Darstellungsart (einfach, gekachelt oder gestreckt) in der Eigenschaft `FillBitmapMode` festzulegen (Standardwerte gemäß `com.sun.star.drawing.BitmapMode`).

Wenn eine externe Bitmap-Datei verwendet werden soll, können Sie den URL in der Eigenschaft `FillBitmapURL` angeben.

Das folgende Beispiel erzeugt ein Rechteck, dessen Fläche mit der in StarOffice verfügbaren Bitmap "Sky" gekachelt gefüllt wird.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
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

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
RectangleShape.FillBitmapName = "Sky"
RectangleShape.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT

Page.add(RectangleShape)
```

Transparenz

Sie können die Transparenz jeder angewendeten Füllung anpassen. Die einfachste Methode zum Ändern der Transparenz eines Zeichnungselements besteht in der Verwendung der Eigenschaft `FillTransparence`.

Das folgende Beispiel erzeugt ein rotes Rechteck mit einer Transparenz von 50 Prozent.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
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

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.SOLID
RectangleShape.FillTransparence = 50
RectangleShape.FillColor = RGB(255,0,0)

Page.add(RectangleShape)
```

Um die Füllung vollständig transparent zu machen, wird die Eigenschaft `FillTransparence` auf 100 gesetzt.

Über die Eigenschaft `FillTransparence` hinaus bietet der Dienst `com.sun.star.drawing.FillProperties` außerdem die Eigenschaft `FillTransparenceGradient`. Sie wird zum Definieren eines Farbverlaufs verwendet, der die Transparenz eines Füllbereichs festlegt.

Linieneigenschaften

Alle Zeichnungsobjekte, die über eine Umrandungslinie verfügen können, unterstützen den Dienst `com.sun.star.drawing.LineStyle`. Einige der von diesem Dienst bereitgestellten Eigenschaften sind:

- **LineStyle (Enum)**: Linienart (Standardwerte gemäß `com.sun.star.drawing.LineStyle`).
- **LineColor (Long)**: Linienfarbe.
- **LineTransparence (Short)**: Linientransparenz.
- **LineWidth (Long)**: Linienstärke in 100stel Millimeter.
- **LineJoint (Enum)**: Übergänge an Verbindungspunkten (Standardwerte gemäß `com.sun.star.drawing.LineJoint`).

Das folgende Beispiel erzeugt ein Rechteck mit einer durchgezogenen Umrandung (`LineStyle = SOLID`), die eine Stärke (`LineWidth`) von 5 mm aufweist und zu 50 Prozent transparent ist. Der rechte und linke Rand der Linie sind bis zu ihren Schnittpunkten mit der jeweils anderen Linie durchgeführt (`LineJoint = MITER`), so dass ein rechter Winkel gebildet wird.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
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

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.LineColor = RGB(128,128,128)
RectangleShape.LineTransparence = 50
RectangleShape.LineWidth = 500
RectangleShape.LineJoint = com.sun.star.drawing.LineJoint.MITER

RectangleShape.LineStyle = com.sun.star.drawing.LineStyle.SOLID

Page.add(RectangleShape)
```

Neben den aufgeführten Eigenschaften bietet der Dienst `com.sun.star.drawing.LineStyle` Möglichkeiten zum Zeichnen gepunkteter und gestrichelter Linien. Weitere Informationen finden Sie in der StarOffice API-Referenz.

Texteigenschaften (Zeichnungsobjekte)

Mit den Diensten `com.sun.star.style.CharacterProperties` und `com.sun.star.style.ParagraphProperties` kann Text in Zeichnungsobjekten formatiert werden. Diese Dienste wirken sich auf einzelne Zeichen und Absätze aus und werden detailliert in Kapitel 6 (*Textdokumente*) beschrieben.

Im folgenden Beispiel wird Text in ein Rechteck eingefügt und die Schriftart mit dem Dienst `com.sun.star.style.CharacterProperties` formatiert.

```
Dim Doc As Object
Dim Page As Object
```

```
Dim RectangleShape As Object
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
Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

Page.add(RectangleShape)

RectangleShape.String = "This is a test"
RectangleShape.CharWeight = com.sun.star.awt.FontWeight.BOLD
RectangleShape.CharFontName = "Arial"
```

Dieser Code verwendet die `String`-Eigenschaft des Rechtecks, um den Text einzufügen, und die Eigenschaften `CharWeight` und `CharFontName` des Dienstes `com.sun.star.style.CharacterProperties`, um die Textschriftart zu formatieren.

Der Text kann erst nach dem Hinzufügen des Zeichnungsobjekts zu der Zeichenseite eingefügt werden. Zum Positionieren und Formatieren von Text in Zeichnungsobjekten können Sie auch den Dienst `com.sun.star.drawing.Text` verwenden. Im Folgenden werden einige der wichtigsten Eigenschaften dieses Dienstes aufgeführt:

- **TextAutoGrowHeight (Boolean)**: passt die Höhe des Zeichnungselements an den enthaltenen Text an.
- **TextAutoGrowWidth (Boolean)**: passt die Breite des Zeichnungselements an den enthaltenen Text an.
- **TextHorizontalAdjust (Enum)**: horizontale Position des Texts im Zeichnungsobjekt (Standardwerte gemäß `com.sun.star.drawing.TextHorizontalAdjust`).
- **TextVerticalAdjust (Enum)**: vertikale Position des Texts im Zeichnungsobjekt (Standardwerte gemäß `com.sun.star.drawing.TextVerticalAdjust`).
- **TextLeftDistance (Long)**: linker Abstand zwischen Zeichnungselement und Text in 100stel Millimeter.
- **TextRightDistance (Long)**: rechter Abstand zwischen Zeichnungselement und Text in 100stel Millimeter.
- **TextUpperDistance (Long)**: oberer Abstand zwischen Zeichnungselement und Text in 100stel Millimeter.
- **TextLowerDistance (Long)**: unterer Abstand zwischen Zeichnungselement und Text in 100stel Millimeter.

Das folgende Beispiel demonstriert den Einsatz der genannten Eigenschaften.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
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

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

Page.add(RectangleShape)

RectangleShape.String = "Dies ist ein Test"   ' Darf erst nach Page.add erfolgen!

RectangleShape.TextVerticalAdjust = com.sun.star.drawing.TextVerticalAdjust.TOP
RectangleShape.TextHorizontalAdjust = com.sun.star.drawing.TextHorizontalAdjust.LEFT

RectangleShape.TextLeftDistance = 300
RectangleShape.TextRightDistance = 300
RectangleShape.TextUpperDistance = 300
RectangleShape.TextLowerDistance = 300
```

Dieser Code fügt ein Zeichnungselement in eine Seite ein und fügt dann in der linken oberen Ecke des Zeichnungsobjekts über die Eigenschaften `TextVerticalAdjust` und `TextHorizontalAdjust` Text hinzu. Der Mindestabstand zwischen Text und Rand des Zeichnungsobjekts ist auf 3 mm festgelegt.

Schatteneigenschaften

Den meisten Zeichnungsobjekten kann mit dem Dienst `com.sun.star.drawing.ShadowProperties` ein Schatten hinzugefügt werden. Die Eigenschaften des Dienstes sind:

- **Shadow (Boolean)**: aktiviert den Schatten.
- **ShadowColor (Long)**: Schattenfarbe.
- **ShadowTransparence (Short)**: Transparenz des Schattens.

- **ShadowXDistance (Long)**: vertikaler Abstand des Schattens vom Zeichnungsobjekt in 100stel Millimeter.
- **ShadowYDistance (Long)**: horizontaler Abstand des Schattens vom Zeichnungsobjekt in 100stel Millimeter.

Das folgende Beispiel erzeugt ein Rechteck mit einem Schatten, der vertikal und horizontal um 2 mm gegenüber dem Rechteck versetzt ist. Der Schatten wird dunkelgrau und mit 50 Prozent Transparenz angezeigt.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
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

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.Shadow = True
RectangleShape.ShadowColor = RGB(192,192,192)
RectangleShape.ShadowTransparence = 50
RectangleShape.ShadowXDistance = 200
RectangleShape.ShadowYDistance = 200

Page.add(RectangleShape)
```

Verschiedene Zeichnungsobjekte im Überblick

Rechtecke

Rechteck-Objekte (`com.sun.star.drawing.RectangleShape`) unterstützen folgende Dienste zur Formatierung von Objekten:

- **Flächeneigenschaften**: `com.sun.star.drawing.FillProperties`
- **Linieneigenschaften**: `com.sun.star.drawing.LineProperties`

- **Texteigenschaften**: `com.sun.star.drawing.Text` (mit `com.sun.star.style.CharacterProperties` und `com.sun.star.style.ParagraphProperties`)
- **Schatteneigenschaften**: `com.sun.star.drawing.ShadowProperties`
- **CornerRadius (Long)**: Radius zum Abrunden von Ecken in 100stel Millimeter.

Kreise und Ellipsen

Für Kreise und Ellipsen ist der Dienst `com.sun.star.drawing.EllipseShape` verantwortlich, der wiederum folgende Dienste unterstützt:

- **Flächeneigenschaften**: `com.sun.star.drawing.FillProperties`
- **Linieneigenschaften**: `com.sun.star.drawing.LineProperties`
- **Texteigenschaften**: `com.sun.star.drawing.Text` (mit `com.sun.star.style.CharacterProperties` und `com.sun.star.style.ParagraphProperties`)
- **Schatteneigenschaften**: `com.sun.star.drawing.ShadowProperties`

Zusätzlich zu diesen Diensten stellen Kreise und Ellipsen auch diese Eigenschaften bereit:

- **CircleKind (Enum)**: Art des Kreises oder der Ellipse (Standardwerte gemäß `com.sun.star.drawing.CircleKind`).
- **CircleStartAngle (Long)**: Startwinkel in 10tel Grad (nur bei Kreis- bzw. Ellipsensegmenten).
- **CircleEndAngle (Long)**: Endwinkel in 10tel Grad (nur bei Kreis- bzw. Ellipsensegmenten).

Die Eigenschaft `CircleKind` bestimmt, ob ein Objekt ein vollständiger Kreis, ein Kreissegment oder ein Kreisabschnitt ist. Folgende Werte stehen zur Verfügung:

- **com.sun.star.drawing.CircleKind.FULL**: vollständiger Kreis bzw. vollständige Ellipse.
- **com.sun.star.drawing.CircleKind.CUT**: Kreisabschnitt (Teilkreis, dessen Schnittstellen direkt miteinander verbunden sind).
- **com.sun.star.drawing.CircleKind.SECTION**: Kreissegment.
- **com.sun.star.drawing.CircleKind.ARC**: Winkel (ohne Kreislinie).

Das folgende Beispiel erzeugt ein Kreissegment mit einem Winkel von 70° (ergibt sich aus der Differenz zwischen dem Startwinkel von 20° und dem Endwinkel von 90°).

```
Dim Doc As Object
Dim Page As Object
Dim EllipseShape As Object
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

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

EllipseShape = Doc.createInstance("com.sun.star.drawing.EllipseShape")
EllipseShape.Size = Size
EllipseShape.Position = Point

EllipseShape.CircleStartAngle = 2000
EllipseShape.CircleEndAngle = 9000
EllipseShape.CircleKind =  com.sun.star.drawing.CircleKind.SECTION

Page.add(EllipseShape)
```

Linien

Für Linienobjekte hält StarOffice den Dienst `com.sun.star.drawing.LineShape` bereit. Linienobjekte unterstützen alle allgemeinen Formatierungsdienste, mit Ausnahme des Dienstes für Flächen. Im Folgenden finden Sie alle Eigenschaften, die mit dem Dienst `LineShape` verknüpft sind:

- **Linieneigenschaften**: `com.sun.star.drawing.LineProperties`
- **Texteigenschaften**: `com.sun.star.drawing.Text` (mit `com.sun.star.style.CharacterProperties` und `com.sun.star.style.ParagraphProperties`)
- **Schatteneigenschaften**: `com.sun.star.drawing.ShadowProperties`

Das folgende Beispiel erzeugt und formatiert eine Linie unter Zuhilfenahme der genannten Eigenschaften. Der Anfangspunkt der Linie ist in der `Location`-Eigenschaft angegeben, während die in der `Size`-Eigenschaft enthaltenen Koordinaten den Endpunkt der Linie angeben.

```
Dim Doc As Object
Dim Page As Object
Dim LineShape As Object
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
```

```
Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

LineShape = Doc.createInstance("com.sun.star.drawing.LineShape")
LineShape.Size = Size
LineShape.Position = Point

Page.add(LineShape)
```

Vielecke (Polypolygone)

StarOffice unterstützt auch komplexe polygonale Formen durch den Dienst `com.sun.star.drawing.PolyPolygonShape`. Streng genommen handelt es sich bei einem *PolyPolygon* nicht einmal um ein einfaches Vieleck, sondern um ein mehrfaches Vieleck. So lassen sich mehrere unabhängige Listen mit Eckpunkten angeben, die zu einem Gesamtobjekt zusammengefasst werden können.

Analog zu den Rechtecken stehen auch für Vielecke alle Formatierungseigenschaften von Zeichnungsobjekten zur Verfügung:

- **Flächeneigenschaften**: `com.sun.star.drawing.FillProperties`
- **Linieneigenschaften**: `com.sun.star.drawing.LineProperties`
- **Texteigenschaften**: `com.sun.star.drawing.Text` (mit `com.sun.star.style.CharacterProperties` und `com.sun.star.style.ParagraphProperties`)
- **Schatteneigenschaften**: `com.sun.star.drawing.ShadowProperties`

Der Dienst `PolyPolygonShape` verfügt auch über eine Eigenschaft, mit der Sie die Koordinaten eines Polygons definieren können:

- `PolyPolygon (Array)`: Feld mit den Koordinaten des Polygons (Doppel-Array mit Punkten vom Typ `com.sun.star.awt.Point`).

Das folgende Beispiel zeigt, wie sich mit Hilfe des Dienstes `PolyPolygonShape` ein Dreieck definieren lässt.

```
Dim Doc As Object
Dim Page As Object
Dim PolyPolygonShape As Object
Dim PolyPolygon As Variant
Dim Coordinates(2) As New com.sun.star.awt.Point

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

PolyPolygonShape = Doc.createInstance("com.sun.star.drawing.PolyPolygonShape")
Page.add(PolyPolygonShape)   ' Page.add muss vor dem Setzen der Koordinaten erfolgen
```

```
Coordinates(0).x = 1000
Coordinates(1).x = 7500
Coordinates(2).x = 10000
Coordinates(0).y = 1000
Coordinates(1).y = 7500
Coordinates(2).y = 5000

PolyPolygonShape.PolyPolygon = Array(Coordinates())
```

Da die Punkte eines Polygons als Absolutwerte definiert sind, müssen Sie die Größe oder die Startposition eines Polygons nicht angeben. Stattdessen müssen Sie einen Array dieser Punkte erstellen, diesen Array in einen zweiten Array integrieren (mit Hilfe des Aufrufs `Array(Coordinates())()`) und diesen Array dann dem Polygon zuweisen. Bevor der entsprechend Aufruf erfolgen kann, muss das Polygon in das Dokument eingefügt werden.

Der Doppel-Array in der Definition ermöglicht Ihnen die Erstellung komplexer Formen durch das Zusammenführen mehrerer Polygone. So können Sie beispielsweise ein Rechteck erstellen und dann ein weiteres darin einfügen, um in dem Originalrechteck ein Loch zu erzeugen:

```
Dim Doc As Object
Dim Page As Object
Dim PolyPolygonShape As Object
Dim PolyPolygon As Variant
Dim Square1(3) As New com.sun.star.awt.Point
Dim Square2(3) As New com.sun.star.awt.Point
Dim Square3(3) As New com.sun.star.awt.Point

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

PolyPolygonShape = Doc.createInstance("com.sun.star.drawing.PolyPolygonShape")

Page.add(PolyPolygonShape)   ' Page.add muss vor dem Setzen der Koordinaten erfolgen

Square1(0).x = 5000
Square1(1).x = 10000
Square1(2).x = 10000
Square1(3).x = 5000
Square1(0).y = 5000
Square1(1).y = 5000
Square1(2).y = 10000
Square1(3).y = 10000

Square2(0).x = 6500
Square2(1).x = 8500
Square2(2).x = 8500
Square2(3).x = 6500
```

```
Square2(0).y = 6500
Square2(1).y = 6500
Square2(2).y = 8500
Square2(3).y = 8500

Square3(0).x = 6500
Square3(1).x = 8500
Square3(2).x = 8500
Square3(3).x = 6500
Square3(0).y = 9000
Square3(1).y = 9000
Square3(2).y = 9500
Square3(3).y = 9500

PolyPolygonShape.PolyPolygon = Array(Square1(), Square2(), Square3())
```

Im Hinblick darauf, welche Bereiche gefüllt und welche Löcher sind, wendet StarOffice eine einfache Regel an: Der Rand der äußersten Form ist immer die äußere Umrandung des PolyPolygons. Die nächste innen liegende Linie stellt die innere Umrandung der Form dar und markiert somit den Übergang zum ersten Loch. Folgt daraufhin eine weitere Linie nach innen, so markiert diese wiederum den Übergang zu einem gefüllten Bereich.

Grafiken

Das letzte der hier vorgestellten Zeichnungselemente sind Grafik-Objekte auf der Grundlage des Dienstes `com.sun.star.drawing.GraphicObjectShape`. Der Dienst kann auf alle Grafiken innerhalb von StarOffice angewendet werden, deren Aussehen mit einer ganzen Reihe von Eigenschaften angepasst werden kann.

Grafikobjekte unterstützen zwei der allgemeinen Formatierungseigenschaften:

- **Texteigenschaften**: `com.sun.star.drawing.Text` (mit `com.sun.star.style.CharacterProperties` und `com.sun.star.style.ParagraphProperties`)
- **Schatteneigenschaften**: `com.sun.star.drawing.ShadowProperties`

Zusätzliche von Grafikobjekten unterstützte Eigenschaften sind:

- **GraphicURL (String)**: URL der Grafik.
- **AdjustLuminance (Short)**: Leuchtkraft der Farben in Prozent (auch negative Werte zugelassen).
- **AdjustContrast (Short)**: Kontrast in Prozent (auch negative Werte zugelassen).
- **AdjustRed (Short)**: Rotanteil in Prozent (auch negative Werte zugelassen).
- **AdjustGreen (Short)**: Grünanteil in Prozent (auch negative Werte zugelassen).
- **AdjustBlue (Short)**: Blauanteil in Prozent (auch negative Werte zugelassen).
- **Gamma (Short)**: Gamma-Wert einer Grafik.

- **Transparency (Short)**: Transparenz einer Grafik in Prozent.
- **GraphicColorMode (Enum)**: Farbmodus, z. B. Standard, Graustufen, Schwarzweiß (Standardwert gemäß `com.sun.star.drawing.ColorMode`).

Das folgende Beispiel zeigt, wie eine Seite in ein Grafikobjekt eingefügt wird. Dim Doc As Object

```
Dim Page As Object
Dim GraphicObjectShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000         ' Angaben; unerheblich, da spätere
                       ' Koordinaten verbindlich sind
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

GraphicObjectShape = Doc.createInstance("com.sun.star.drawing.GraphicObjectShape")

GraphicObjectShape.Size = Size
GraphicObjectShape.Position = Point

GraphicObjectShape.GraphicURL = "file:///c:/test.jpg"
GraphicObjectShape.AdjustBlue = -50
GraphicObjectShape.AdjustGreen = 5
GraphicObjectShape.AdjustBlue = 10
GraphicObjectShape.AdjustContrast = 20
GraphicObjectShape.AdjustLuminance = 50
GraphicObjectShape.Transparency = 40
GraphicObjectShape.GraphicColorMode = com.sun.star.drawing.ColorMode.STANDARD

Page.add(GraphicObjectShape)
```

Dieser Code fügt die Grafik `test.jpg` ein und passt ihr Aussehen über die `Adjust`-Eigenschaften an. In diesem Beispiel wird die Grafik mit 40% Transparenz und ohne zusätzliche Farbumwandlungen dargestellt (`GraphicColorMode = STANDARD`).

Bearbeiten von Zeichnungsobjekten

Gruppieren von Objekten

In vielen Situationen ist es hilfreich, mehrere einzelne Zeichnungsobjekte zu einem großen Einzelobjekt zusammenzufassen.

Im folgenden Beispiel werden zwei Zeichnungsobjekte zusammengefasst:

```
Dim Doc As Object
Dim Page As Object
Dim Square As Object
Dim Circle As Object
Dim Shapes As Object
Dim Group As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
Dim NewPos As New com.sun.star.awt.Point
Dim Height As Long
Dim Width As Long

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
Point.x = 3000
Point.y = 3000
Size.Width = 3000
Size.Height = 3000
' Rechteckiges Zeichnungselement Square erstellen
Square = Doc.createInstance("com.sun.star.drawing.RectangleShape")
Square.Size = Size
Square.Position = Point
Square.FillColor = RGB(255,128,128)
Page.add(Square)
' Kreisförmiges Zeichnungselement Circle erstellen
Circle = Doc.createInstance("com.sun.star.drawing.EllipseShape")
Circle.Size = Size
Circle.Position = Point
Circle.FillColor = RGB(255,128,128)
Circle.FillColor = RGB(0,255,0)
Page.add(Circle)
' Zeichnungselemente Square und Circlezusammenfassen
Shapes = createUnoService("com.sun.star.drawing.ShapeCollection")
Shapes.add(Square)
Shapes.add(Circle)
Group = Page.group(Shapes)
' Zusammengefasste Zeichnungselemente zentrieren
```

```
Height = Page.Height
Width = Page.Width
NewPos.X = Width / 2
NewPos.Y = Height / 2
Height = Group.Size.Height
Width = Group.Size.Width
NewPos.X = NewPos.X - Width / 2
NewPos.Y = NewPos.Y - Height / 2
Group.Position = NewPos
```

Dieser Code erstellt ein Rechteck und einen Kreis und fügt beide in eine Seite ein. Danach wird ein Objekt erstellt, das den Dienst `com.sun.star.drawing.ShapeCollection` unterstützt und die `Add`-Methode verwendet, um das Rechteck und den Kreis diesem Objekt hinzuzufügen. Die `ShapeCollection` wird der Seite mit der `Group`-Methode hinzugefügt und gibt das eigentliche `Group`-Objekt zurück, das wie ein einzelnes `Shape` bearbeitet werden kann.

Wenn die einzelnen Objekte einer Gruppe formatiert werden sollen, wenden Sie die Formatierung vor dem Hinzufügen zur Gruppe an. Objekte, die sich in der Gruppe befinden, können nicht mehr geändert werden.

Drehen und Scheren von Zeichnungsobjekten

Alle in den vorangehenden Abschnitten beschriebenen Zeichnungsobjekte können mit dem Dienst `com.sun.star.drawing.RotationDescriptor` auch gedreht und geschert werden.

Der Dienst stellt folgende Eigenschaften zur Verfügung:

- **RotateAngle (Long)**: Drehwinkel in 100stel Grad.
- **ShearAngle (Long)**: Scherwinkel in 100stel Grad.

Das folgende Beispiel erstellt ein Rechteck und dreht es über die Eigenschaft `RotateAngle` um 30°.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
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

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
```

```
RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.RotateAngle = 3000

Page.add(RectangleShape)
```

Das nächste Beispiel erstellt dasselbe Rechteck wie in dem vorherigen Beispiel, schert es aber stattdessen über die Eigenschaft `ShearAngle` um 30°.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
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

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.ShearAngle = 3000

Page.add(RectangleShape)
```

Suchen und Ersetzen

Zeichnungsdokumente bieten ähnlich wie Textdokumente eine Funktion zum Suchen und Ersetzen. Diese Funktion ist der in Textdokumenten verwendeten sehr ähnlich. Eine Beschreibung finden Sie in Kapitel 6, *Textdokumente*. In Zeichnungsobjekten werden die Deskriptor-Objekte zum Suchen und Ersetzen jedoch nicht direkt über das Dokumentobjekt, sondern über die verknüpfte Zeichenebene erzeugt. Das folgende Beispiel verdeutlicht einen Ersetzungsvorgang innerhalb einer Zeichnung:

```
Dim Doc As Object
Dim Page As Object
Dim ReplaceDescriptor As Object
Dim I As Integer
```

```
Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

ReplaceDescriptor = Page.createReplaceDescriptor()
ReplaceDescriptor.SearchString = "is"
ReplaceDescriptor.ReplaceString = "was"

For I = 0 to Doc.drawPages.Count - 1
   Page = Doc.drawPages(I)
   Page.ReplaceAll(ReplaceDescriptor)
Next I
```

Dieser Code erzeugt zuerst über die erste `DrawPage` des Dokuments einen `ReplaceDescriptor` und wendet diesen anschließend in einer Schleife auf alle in dem Zeichnungsdokument enthaltenen Seiten an.

Präsentationen

StarOffice-Präsentationen basieren auf Zeichnungsdokumenten. Jede Seite einer Präsentation ist eine Folie. Der Zugriff auf Folien erfolgt wie bei einer Standardzeichnung über die Liste `DrawPages` des Dokumentobjekts. Der für Präsentationsdokumente zuständige Dienst `com.sun.star.presentation.PresentationDocument` stellt dazu auch den vollständigen Dienst `com.sun.star.drawing.DrawingDocument` zur Verfügung.

Arbeiten mit Präsentationen

Über die Zeichnungsfunktionen der Eigenschaft `Presentation` hinaus verfügt das Präsentationsdokument über ein Präsentationsobjekt, das Zugriff auf die Haupteigenschaften und Steuerungsmechanismen für Präsentationen gewährt. Dieses Objekt bietet beispielsweise eine Methode `start`, die zum Starten von Präsentationen dient.

```
Dim Doc As Object
Dim Presentation As Object

Doc = StarDesktop.CurrentComponent
Presentation = Doc.Presentation
Presentation.start()
```

Der Beispielcode erstellt ein `Doc`-Objekt, das auf das aktuelle Präsentationsdokument verweist, und ermittelt das verknüpfte Präsentationsobjekt. Mit der Methode `start()` des Objekts wird das Beispiel anschließend gestartet und die Bildschirmpräsentation vorgeführt.

Das Präsentationsobjekt hält folgende Methoden bereit:

- **start**: startet die Präsentation.

- **end**: beendet die Präsentation.
- **rehearseTimings**: startet die Präsentation vom Anfang und ermittelt ihre Laufzeit.

Darüber hinaus stehen folgende Eigenschaften zur Verfügung:

- **AllowAnimations (Boolean)**: führt Animationen in der Präsentation aus.
- **CustomShow (String)**: gestattet die Angabe des Namens der Präsentation, so dass der Name innerhalb der Präsentation referenziert werden kann.
- **FirstPage (String)**: Name der Folie, mit der die Präsentation beginnen soll.
- **IsAlwaysOnTop (Boolean)**: zeigt das Präsentationsfenster immer als oberstes Fenster am Bildschirm an.
- **IsAutomatic (Boolean)**: führt die Präsentation automatisch vor.
- **IsEndless (Boolean)**: startet die Präsentation nach deren Ende wieder von vorne.
- **IsFullScreen (Boolean)**: startet die Präsentation automatisch im Vollbildmodus.
- **IsMouseVisible (Boolean)**: zeigt den Mauszeiger während der Präsentation an.
- **Pause (long)**: der Zeitraum, für den am Ende der Präsentation ein leerer Bildschirm angezeigt wird.
- **StartWithNavigator (Boolean)**: zeigt bei Beginn der Präsentation das Navigatorfenster an.
- **UsePn (Boolean)**: zeigt den Mauszeiger während der Präsentation als Stift an.

◆ ◆ ◆ KAPITEL 9

9 Diagramme (Charts)

StarOffice kann Daten als Diagramme darstellen und so diese Daten in Form von Balken, Kuchenstücken, Linien oder anderen Elementen grafisch zueinander in Bezug setzen. Die Ausgabe kann wahlweise als 2D- oder 3D-Grafik erfolgen, wobei das Aussehen der Diagrammelemente, ähnlich wie beim Zeichnen von Elementen, individuell angepasst werden kann.

Liegen die darzustellenden Daten als Tabellendokument vor, so lassen sich diese dynamisch mit dem Diagramm verknüpfen. Änderungen der Basisdaten werden in diesem Fall sofort im zugeordneten Diagramm sichtbar. Dieses Kapitel bietet einen Überblick über die Programmierschnittstelle des Diagramm-Moduls von StarOffice, wobei der Schwerpunkt auf dem Einsatz von Diagrammen innerhalb von Tabellendokumenten liegt.

Verwenden von Diagrammen in Tabellendokumenten

Diagramme werden in StarOffice nicht als unabhängige Dokumente behandelt, sondern als Objekte, die in ein existierendes Dokument eingebettet sind.

Während Diagramme in Text- und Zeichnungsdokumenten isoliert vom Inhalt des Dokumentes stehen, steht bei ihrem Einsatz in Tabellendokumenten ein Mechanismus zur Verfügung, der eine Verknüpfung der Dokumentdaten mit den eingebetteten Diagrammen gestattet. Das folgende Beispiel erläutert das Zusammenspiel zwischen Tabellendokument und Diagramm:

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
```

```
Charts = Doc.Sheets(0).Charts

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
```

Obwohl der Beispielcode recht komplex erscheinen mag, beschränken sich die zentralen Vorgänge auf drei Zeilen: In der ersten zentralen Zeile wird die Dokumentvariable `Doc` erzeugt, die auf das aktuelle Tabellendokument verweist (Zeile `Doc = StarDesktop.CurrentComponent`). Im Anschluss daran erstellt der Beispielcode eine Liste mit allen Diagrammen des ersten Tabellendokuments (Zeile `Charts = Doc.Sheets(0).Charts`). Dieser Liste wird schließlich in der letzten Zeile des Beispiels mit der Methode `addNewByName` ein neues Diagramm hinzugefügt, das dann dem Anwender angezeigt wird.

In der letzten Zeile werden die Hilfsstrukturen `Rect` und `RangeAddress` initialisiert, die ebenfalls Parameter der Methode `addNewByName` sind. Durch `Rect` wird die Position des Diagramms innerhalb des Tabellendokuments bestimmt. Mit `RangeAddress` wird der Bereich festgelegt, dessen Daten mit dem Diagramm verknüpft werden sollen.

Das obige Beispiel erzeugt ein Balkendiagramm. Wird ein anderer Diagrammtyp benötigt, so muss das Balkendiagramm explizit ausgetauscht werden:

```
Chart = Charts.getByName("MyChart").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
```

In der ersten Zeile wird das entsprechende Chart-Objekt (Diagramm) definiert. In der zweiten Zeile wird das aktuelle Diagramm durch ein neues ausgetauscht – im Beispiel ein Liniendiagramm.

Hinweis – In Excel wird unterschieden zwischen Diagrammen, die als eigenständige Seite in ein Excel-Dokument eingefügt wurden, und Charts, die in eine Tabellenseite eingebettet sind. Dementsprechend sind dort zwei verschiedene Zugriffsmöglichkeiten auf Diagramme definiert. Diese Unterscheidung entfällt in StarOffice Basic, da Diagramme in StarOffice Calc immer als eingebettete Objekte einer Tabellenseite erzeugt werden. Der Zugriff auf diese Diagramme erfolgt hierbei immer über die Liste `Charts` des zugehörigen `Sheet`-Objekts.

Der Aufbau von Diagrammen

Der Aufbau eines Diagramms – und damit die Liste der von ihm unterstützten Dienste und Schnittstellen – hängt von seinem Typ ab. So stehen die Methoden und Eigenschaften der Z-Achse beispielsweise nur in 3D-Diagrammen zur Verfügung, nicht jedoch in 2D-Diagrammen. In Tortendiagrammen gibt es keine Schnittstellen zum Arbeiten mit Achsen.

Die Einzelelemente eines Diagramms

Titel, Untertitel und Legende

Zu den Basiselementen eines jeden Diagramms gehören ein Titel, ein Untertitel und eine Legende. Diagramme stellen für jedes dieser Elemente eigene Objekte zur Verfügung. Das `Chart`-Objekt bietet zur Verwaltung dieser Elemente folgende Eigenschaften:

- **HasMainTitle (Boolean)**: aktiviert den Titel.
- **Title (Object)**: Objekt mit Detailangaben zum Diagrammtitel (unterstützt den Dienst `com.sun.star.chart.ChartTitle`).
- **HasSubTitle (Boolean)**: aktiviert den Untertitel.
- **Subtitle (Object)**: Objekt mit Detailangaben zum Diagrammuntertitel (unterstützt den Dienst `com.sun.star.chart.ChartTitle`).
- **HasLegend (Boolean)**: aktiviert die Legende.
- **Legend (Object)**: Objekt mit Detailangaben zur Legende des Diagramms (unterstützt den Dienst `com.sun.star.chart.ChartLegendPosition`).

Die genannten Elemente entsprechen in vieler Hinsicht einem Zeichnungselement. Dies liegt daran, dass die beiden Dienste `com.sun.star.chart.ChartTitle` und `com.sun.star.chart.ChartLegendPosition` den Dienst `com.sun.star.drawing.Shape` unterstützen, der die programmtechnische Grundlage für Zeichnungselemente legt.

So besteht die Möglichkeit, die Position und Größe der Elemente über die Eigenschaften `Size` und `Position` zu bestimmen.

Für die Formatierung der Elemente stehen des Weiteren Füll- und Linien-Eigenschaften (Dienste `com.sun.star.drawing.FillProperties` und `com.sun.star.drawing.LineStyle`) sowie Zeichen-Eigenschaften (Dienst `com.sun.star.style.CharacterProperties`) zur Verfügung.

`com.sun.star.chart.ChartTitle` besitzt neben den genannten Format-Eigenschaften zwei weitere Eigenschaften:

- **TextRotation (Long)**: Drehwinkel des Texts in 100stel Grad.
- **String (String)**: Text, der als Titel oder Untertitel angezeigt werden soll.

Die Legende des Diagramms (Dienst `com.sun.star.chart.ChartLegend`) besitzt folgende zusätzliche Eigenschaft:

- **Alignment (Enum)**: Position, an der die Legende angezeigt werden soll (Standardwert gemäß `com.sun.star.chart.ChartLegendPosition`).

Das folgende Beispiel erzeugt ein Diagramm und weist ihm den Titel "Test", den Untertitel "Test 2" und eine Legende zu. Die Legende erhält eine graue Hintergrundfarbe, wird am unteren Rand des Diagramms platziert und verfügt über eine Schriftgröße von 7 Punkt.

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts
Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MyChart").EmbeddedObject

Chart.HasMainTitle = True
Chart.Title.String = "Test"

Chart.HasSubTitle = True
Chart.Subtitle.String = "Test 2"

Chart.HasLegend = True
Chart.Legend.Alignment = com.sun.star.chart.ChartLegendPosition.BOTTOM
Chart.Legend.FillStyle = com.sun.star.drawing.FillStyle.SOLID
Chart.Legend.FillColor = RGB(210, 210, 210)
Chart.Legend.CharHeight = 7
```

Hintergrund

Jedes Diagramm besitzt eine Hintergrundfläche. Für jede Fläche ist ein Objekt vorhanden, auf das über folgende Eigenschaften des Diagramm-Objekts zugegriffen werden kann:

- **Area (Object)**: Hintergrundfläche des Diagramms (unterstützt den Dienst `com.sun.star.chart.ChartArea`).

Der Hintergrund eines Diagramms umfasst dessen komplette Fläche, einschließlich der Flächen hinter Titel, Untertitel und Diagramm-Legende. Der verknüpfte Dienst `com.sun.star.chart.ChartArea` unterstützt Linien- und Füll-Eigenschaften und stellt keine weitergehenden Eigenschaften zur Verfügung.

Diagrammwände und -böden

Während der Diagrammhintergrund die vollständige Fläche des Diagramms umfasst, ist die Diagrammrückwand auf die Fläche unmittelbar hinter dem Datenbereich beschränkt.

Bei 3D-Diagrammen existieren für gewöhnlich zwei Diagrammwände: eine hinter dem Datenbereich und eine als linke Begrenzung der Y-Achse. Zusätzlich enthalten 3D-Diagramme in der Regel einen Boden.

- **Floor (Object)**: Bodenpanel des Diagramms (nur bei 3D-Diagrammen, unterstützt den Dienst `com.sun.star.chart.ChartArea`).
- **Wall (Object)**: Diagrammwände (nur bei 3D-Diagrammen, unterstützt den Dienst `com.sun.star.chart.ChartArea`).

Die angegebenen Objekte unterstützen den Dienst `com.sun.star.chart.ChartArea`, der wiederum die gängigen Füll- und Linien-Eigenschaften bereitstellt (Dienste `com.sun.star.drawing.FillProperties` und `com.sun.star.drawing.LineStyle`, siehe Kapitel 8).

Der Zugriff auf Diagrammwände und -böden erfolgt über das `Chart`-Objekt, das wiederum Bestandteil des Chart-Objekts ist:

```
Chart.Area.FillBitmapName = "Sky"
```

Das folgende Beispiel zeigt, wie sich eine bereits in StarOffice eingebundene Grafik mit dem Namen „Sky" als Hintergrund eines Diagramms einsetzen lässt.

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000
```

```
RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MyChart").EmbeddedObject

Chart.Area.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
Chart.Area.FillBitmapName = "Sky"
Chart.Area.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT
```

Achsen

StarOffice erkennt fünf verschiedene Achsen, die in einem Diagramm zum Einsatz kommen können. Im einfachsten Falle sind dies die X- und Y-Achse. Bei 3D-Diagrammen steht teilweise zusätzlich eine Z-Achse zur Verfügung. In Diagrammen, bei denen die Werte der verschiedenen Datenzeilen stark voneinander abweichen, bietet StarOffice eine zweite X- und Y-Achse für eine zweite, abweichende Skalierung an.

Erste X-, Y- und Z-Achse

Für jede der ersten X-, Y- und Z-Achse können neben der eigentlichen Achse ein Titel, eine Beschreibung, ein Gitter und ein Hilfsgitter vorhanden sein. Alle dieser Elemente können wahlweise ein- und ausgeblendet werden. Das Diagramm-Objekt bietet zur Verwaltung dieser Elemente folgende Eigenschaften an (im Beispiel die X-Achse; die Eigenschaften für Y- und Z-Achse sind analog aufgebaut):

- **HasXAxis (Boolean)**: aktiviert die X-Achse.
- **XAxis (Object)**: Objekt mit Detailinformationen zur X-Achse (unterstützt den Dienst `com.sun.star.chart.ChartAxis`).
- **HasXAxisDescription (Boolean)**: aktiviert die Beschreibung der X-Achse.
- **HasXAxisGrid (Boolean)**: aktiviert das Hauptgitter für die X-Achse.
- **XMainGrid (Object)**: Objekt mit Detailinformationen zum Hauptgitter der X-Achse (unterstütz den Dienst `com.sun.star.chart.ChartGrid`).
- **HasXAxisHelpGrid (Boolean)**: aktiviert das Hilfsgitter für die X-Achse.
- **XHelpGrid (Object)**: Objekt mit Detailinformationen zum Hilfsgitter der X-Achse (unterstützt den Dienst `com.sun.star.chart.ChartGrid`).
- **HasXAxisTitle (Boolean)**: aktiviert den Titel der X-Achse.

- **XAxisTitle (Object)**: Objekt mit Detailinformationen zum Titel der X-Achse (unterstützt den Dienst `com.sun.star.chart.ChartTitle`).

Zweite X- und Y-Achse

Für die zweite X- und Y-Achse stehen folgende Eigenschaften zur Verfügung (Eigenschaften am Beispiel der zweiten X-Achse):

- **HasSecondaryXAxis (Boolean)**: aktiviert die zweite X-Achse.
- **SecondaryXAxis (Object)**: Objekt mit Detailinformationen zur zweiten X-Achse (unterstützt den Dienst `com.sun.star.chart.ChartAxis`).
- **HasSecondaryXAxisDescription (Boolean)**: aktiviert die Beschreibung der X-Achse.

Eigenschaften der Achsen

Die Achsen-Objekte eines StarOffice-Diagramms unterstützen den Dienst `com.sun.star.chart.ChartAxis`. Er stellt neben den Eigenschaften für Zeichen (Dienst `com.sun.star.style.CharacterProperties`, siehe Kapitel 6) und Linien (Dienst `com.sun.star.drawing.LineStyle`, siehe Kapitel 8) folgende Eigenschaften zur Verfügung:

- **Max (Double)**: Maximalwert für die Achse.
- **Min (Double)**: Minimalwert für die Achse.
- **Origin (Double)**: Schnittpunkt bei sich schneidenden Achsen (Ursprung).
- **StepMain (Double)**: Abstand zwischen zwei Hauptstrichen der Achse.
- **StepHelp (Double)**: Abstand zwischen zwei Hilfsstrichen der Achse.
- **AutoMax (Boolean)**: ermittelt automatisch den Maximalwert für die Achse.
- **AutoMin (Boolean)**: ermittelt automatisch den Minimalwert für die Achse.
- **AutoOrigin (Boolean)**: ermittelt automatisch den Schnittpunkt sich schneidender Achsen (Ursprung).
- **AutoStepMain (Boolean)**: ermittelt automatisch den Abstand zwischen Hauptstrichen einer Achse.
- **AutoStepHelp (Boolean)**: ermittelt automatisch den Abstand zwischen Hilfsstrichen einer Achse.
- **Logarithmic (Boolean)**: skaliert die Achsen in logarithmischem Maßstab (statt linear).
- **DisplayLabels (Boolean)**: aktiviert die Textbeschriftung für Achsen.
- **TextRotation (Long)**: Drehwinkel der Textbeschriftung von Achsen in 100stel Grad.
- **Marks (Const)**: Konstante, die angibt, ob die Hauptstriche der Achse innerhalb oder außerhalb der Diagrammfläche liegen sollen (Standardwerte gemäß `com.sun.star.chart.ChartAxisMarks`).

- **HelpMarks (Const)**: Konstante, die angibt, ob die Hilfsstriche der Achse innerhalb und/oder außerhalb der Diagrammfläche liegen sollen (Standardwerte gemäß `com.sun.star.chart.ChartAxisMarks`).
- **Overlap (Long)**: Prozentwert, der angibt, wie weit sich die zu unterschiedlichen Datensätzen gehörenden Balken überlappen dürfen (bei 100% werden die Balken komplett überlagert dargestellt, bei -100% besteht zwischen ihnen ein Abstand von einer Balkenbreite).
- **GapWidth (Long)**: Prozentwert, der angibt, welcher Abstand zwischen den verschiedenen Balkengruppen eines Diagramms vorhanden sein darf (bei 100% besteht ein Abstand von einer Balkenbreite).
- **ArrangeOrder (Enum)**: Detailangaben zur Position der Beschriftung; neben der Positionierung auf einer Linie besteht die Möglichkeit, die Beschriftung alternierend auf zwei Linien zu verteilen (Standardwert gemäß `com.sun.star.chart.ChartAxisArrangeOrderType`)
- **TextBreak (Boolean)**: ermöglicht Zeilenumbrüche.
- **TextCanOverlap (Boolean)**: ermöglicht Textüberlappungen.
- **NumberFormat (Long)**: Zahlenformat (siehe Abschnitt „Zahlen-, Datums- und Textformat" auf Seite 138)

Eigenschaften des Achsengitters

Das Objekt für das Achsengitter basiert auf dem Dienst `com.sun.star.chart.ChartGrid`, der wiederum die Linien-Eigenschaften des Dienstes `com.sun.star.drawing.LineStyle` unterstützt (siehe Kapitel 8).

Eigenschaften der Achsentitel

Die Objekte zur Formatierung des Achsentitels basieren auf dem Dienst `com.sun.star.chart.ChartTitle`, der auch beim Diagrammtitel zum Einsatz kommt.

Beispiel

Das folgende Beispiel erstellt ein Linien-Diagramm. Die Farbe für die Rückwand des Diagramms wird auf Weiß gesetzt. Sowohl die X- als auch die Y-Achse erhalten ein graues Hilfsgitter zur visuellen Orientierung. Der Minimalwert der Y-Achse wird fest auf 0 und der Maximalwert fest auf 100 gesetzt, so dass die Auflösung des Diagramms auch bei Änderungen der Werte erhalten bleibt.

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object
```

```
Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)

Chart = Charts.getByName("MyChart").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")

Chart.Diagram.Wall.FillColor = RGB(255, 255, 255)

Chart.Diagram.HasXAxisGrid = True
Chart.Diagram.XMainGrid.LineColor = RGB(192, 192, 192)

Chart.Diagram.HasYAxisGrid = True
Chart.Diagram.YMainGrid.LineColor = RGB(192, 192, 192)

Chart.Diagram.YAxis.Min = 0
Chart.Diagram.YAxis.Max = 100
```

3D-Diagramme

Die meisten Diagramme in StarOffice können auch dreidimensional (3D) dargestellt werden. Alle Diagrammtypen, die diese Möglichkeit bieten, unterstützen den Dienst `com.sun.star.chart.Dim3DDiagram`. Der Dienst stellt lediglich eine Eigenschaft zur Verfügung:

- **Dim3D (Boolean)**: aktiviert die 3D-Darstellung.

Gestapelte Diagramme

Unter einer gestapelten Diagrammdarstellung versteht man das Übereinanderordnen mehrerer Einzelwerte zu einem Gesamtwert. Eine solche Darstellung vermittelt damit neben den Einzelwerten einen Überblick zur Gesamtentwicklung.

In StarOffice lassen sich verschiedene Diagrammarten in gestapelter Form darstellen. Alle diese Diagramme unterstützen den Dienst `com.sun.star.chart.StackableDiagram`, der wiederum folgende Eigenschaften bereitstellt:

- **Stacked (Boolean)**: aktiviert die gestapelte Ansicht.
- **Percent (Boolean)**: stellt statt Absolutwerten ihre prozentuale Verteilung dar.

Diagrammarten

Liniendiagramme

Liniendiagramme (Dienst `com.sun.star.chart.LineDiagram`) unterstützen eine X-Achse, zwei Y-Achsen und eine Z-Achse. Sie können als 2D- oder 3D-Grafik angezeigt werden (Dienst `com.sun.star.chart.Dim3DDiagram`). Die Linien sind stapelbar (`com.sun.star.chart.StackableDiagram`).

Liniendiagramme stellen folgende Eigenschaften zur Verfügung:

- **SymbolType (Const)**: Symbol zum Anzeigen von Datenpunkten (Konstante gemäß `com.sun.star.chart.ChartSymbolType`).
- **SymbolSize (Long)**: Größe des zum Anzeigen von Datenpunkten verwendeten Symbols in 100stel Millimeter.
- **SymbolBitmapURL (String)**: Dateiname der Grafik zum Anzeigen der Datenpunkte.
- **Lines (Boolean)**: verbindet die Datenpunkte mit Linien.
- **SplineType (Long)**: Spline-Funktion zum Glätten der Linien (0: keine Spline-Funktion, 1: kubische Splines, 2: B-Splines).
- **SplineOrder (Long)**: Polynomial-Gewicht für Splines (nur für B-Splines).
- **SplineResolution (Long)**: Anzahl der Stützpunkte für die Spline-Berechnung.

Flächendiagramme

Flächendiagramme (Dienst `com.sun.star.chart.AreaDiagram`) unterstützen eine X-Achse, zwei Y-Achsen und eine Z-Achse. Sie können als 2D- oder 3D-Grafik angezeigt werden (Dienst `com.sun.star.chart.Dim3DDiagram`). Die Flächen sind stapelbar (`com.sun.star.chart.StackableDiagram`).

Balkendiagramme

Balkendiagramme (Dienst `com.sun.star.chart.BarDiagram`) unterstützen eine X-Achse, zwei Y-Achsen und eine Z-Achse. Sie können als 2D- oder 3D-Grafik angezeigt werden (Dienst `com.sun.star.chart.Dim3DDiagram`). Die Balken sind stapelbar (`com.sun.star.chart.StackableDiagram`).

Sie stellen folgende Eigenschaften zur Verfügung:

- **Vertical (Boolean)**: stellt die Balken senkrecht (vertikal) dar, ansonsten erfolgt die Darstellung waagerecht (horizontal).
- **Deep (Boolean)**: positioniert die Balken bei 3D-Darstellung hintereinander statt nebeneinander.
- **StackedBarsConnected (Boolean)**: verbindet die verknüpften Balken in einem gestapelten Diagramm mit Linien (nur bei horizontalen Diagrammen verfügbar).
- **NumberOfLines (Long)**: Anzahl der Zeilen, die in einem gestapelten Diagramm als Linien statt als Balken dargestellt werden sollen.

Tortendiagramme

Tortendiagramme (Dienst `com.sun.star.chart.PieDiagram`) enthalten keine Achsen und sind nicht stapelbar. Sie können als 2D- oder 3D-Grafik angezeigt werden (Dienst `com.sun.star.chart.Dim3DDiagram`).

◆ ◆ ◆ KAPITEL 10

10 Datenbankzugriff

StarOffice verfügt über eine integrierte, systemunabhängige Datenbankschnittstelle namens Star Database Connectivity (SDBC). Ziel bei der Entwicklung der Schnittstelle war es, Zugriff auf möglichst viele unterschiedliche Datenquellen zu gewähren.

Um dies zu ermöglichen, erfolgt der Zugriff auf Datenquellen über Treiber. Woher die Treiber ihre Daten beziehen, ist aus Sicht eines SDBC-Anwenders unerheblich. Einige Treiber greifen auf dateibasierte Datenbanken zu und entnehmen ihnen die Daten direkt. Andere verwenden Standardschnittstellen wie JDBC oder ODBC. Es existieren aber auch Spezialtreiber, die beispielsweise auf das MAPI-Adressbuch, LDAP-Verzeichnisse oder StarOffice-Tabellendokumente als Datenquellen zugreifen.

Da die Treiber auf UNO-Komponenten basieren, ist es möglich, weitere Treiber zu entwickeln und so abermals neue Datenquellen zu erschließen. Detailinformationen zu diesem Thema finden Sie im StarOffice Developer's Guide.

Hinweis – SDBC ist vom Konzept her mit den in VBA vorhandenen Bibliotheken ADO beziehungsweise DAO vergleichbar. Es gestattet einen Highlevel-Zugriff auf Datenbanken, unabhängig von den darunter liegenden Datenbank-Backends.

Hinweis – Die Datenbankschnittstelle von StarOffice ist mit der Veröffentlichung von StarOffice 8 umfangreicher geworden. Zwar wurde in der Vergangenheit primär unter Verwendung einer Reihe von Methoden des `Application`-Objekts auf Datenbanken zugegriffen, ist die Schnittstelle von StarOffice 7 in mehrere Objekte untergliedert. Als Stammobjekt für die Datenbankfunktionen dient ein `DatabaseContext`.

SQL als Abfragesprache

Als Abfragesprache steht den Anwendern von SDBC die Sprache SQL zur Verfügung. Um die Unterschiede verschiedener SQL-Dialekte auszugleichen besitzt die SDBC-Komponente von StarOffice einen eigenen SQL-Parser. Dieser prüft die über das Abfragefenster eingegebenen SQL-Befehle und korrigiert einfache Syntaxfehler, etwa im Zusammenhang mit der Groß- und Kleinschreibung.

Ermöglicht ein Treiber Zugriff auf eine Datenquelle, die kein SQL unterstützt, so muss er die übergebenen SQL-Befehle eigenständig auf den notwendigen nativen (systemeigenen) Zugriff umsetzen.

Hinweis – Die SQL-Implementation von SDBC orientiert sich am SQL-ANSI-Standard. Microsoft-spezifische Erweiterungen wie das `INNER JOIN`-Konstrukt werden nicht unterstützt. Diese sind durch Standard-Befehle zu ersetzen (`INNER JOIN` beispielsweise durch eine entsprechende `WHERE`-Klausel).

Arten von Datenbankzugriff

Die Datenbankschnittstelle von StarOffice steht in den Anwendungen StarOffice Writer und StarOffice Calc sowie in den Datenbankformularen zur Verfügung.

In StarOffice Writer ist es möglich, Serienbriefe unter Zuhilfenahme von SDBC-Datenquellen zu erzeugen und diese auszudrucken. Außerdem besteht die Möglichkeit, Daten über Drag & Drop aus dem Datenbankfenster in Textdokumente zu übernehmen.

Zieht der Anwender eine Datenbanktabelle in ein Tabellendokument, so erzeugt StarOffice einen Tabellenbereich, der sich bei Änderungen der Originaldaten per Mausklick aktualisieren lässt. Umgekehrt ist es möglich, Tabellendokumentdaten auf eine Datenbanktabelle zu ziehen und so einen Datenbank-Import durchzuführen.

Schließlich stellt StarOffice einen Mechanismus für datenbankbasierte Formulare zur Verfügung. Hierzu erstellt der Anwender zunächst ein Standardformular in StarOffice Writer oder StarOffice Calc und verknüpft die Felder anschließend mit einer Datenbank.

Alle genannten Möglichkeiten bedienen sich dabei ausschließlich der Benutzeroberfläche von StarOffice. Zur Verwendung der entsprechenden Funktionen sind keinerlei Programmierkenntnisse erforderlich.

In diesem Kapitel geht es jedoch weniger um die genannten Funktionen. Viel mehr steht die Programmierschnittstelle von SDBC im Vordergrund, die eine automatisierte Abfrage von Datenbanken gestattet und so eine noch viel größere Bandbreite an Anwendungen ermöglicht.

Für das Verständnis der folgenden Seiten sind Grundkenntnisse über die Funktionsweise von Datenbanken sowie die Abfragesprache SQL notwendig.

Datenquellen

Die Einbindung einer Datenbank in StarOffice erfolgt über die Erstellung einer sogenannten *Datenquelle*. Die Benutzeroberfläche stellt im Menü **Extras** eine entsprechende Möglichkeit zum Erstellen von Datenquellen zur Verfügung. Sie sind jedoch auch in der Lage, Datenquellen zu erstellen und mit ihnen in StarOffice Basic zu arbeiten.

Ausgangspunkt für den Zugriff auf eine Datenquelle bildet ein Datenbankkontext-Objekt, das mit der Funktion `createUnoService` erstellt wird. Es basiert auf dem Dienst `com.sun.star.sdb.DatabaseContext` und stellt das Stammobjekt für alle Datenbankoperationen dar.

Das folgende Beispiel zeigt, wie sich ein Datenbankkontext erstellen und zur Ermittlung der Namen aller verfügbaren Datenquellen verwenden lässt. Es gibt die Namen in einem Meldungsfenster aus.

```
Dim DatabaseContext As Object
Dim Names
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")

Names = DatabaseContext.getElementNames()
For I = 0 To UBound(Names())
   MsgBox Names(I)
Next I
```

Die einzelnen Datenquellen basieren auf dem Dienst `com.sun.star.sdb.DataSource` und lassen sich aus dem Datenbankkontext mit der Methode `getByName` ermitteln:

```
Dim DatabaseContext As Object
Dim DataSource As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
```

Das Beispiel erstellt ein `DataSource`-Objekt für eine Datenquelle namens *Customers*.

Datenquellen stellen eine Reihe von Eigenschaften zur Verfügung, die allgemeine Informationen über den Ursprung der Daten sowie Informationen über die Zugriffsmöglichkeiten liefern. Die Eigenschaften lauten:

- **Name (String)**: Name der Datenquelle.
- **URL (String)**: URL der Datenquelle in der Form *jdbc: subprotocol : subname* oder *sdbc: subprotocol : subname*.
- **Info (Array)**: ein Array mit `PropertyValue`-Paaren, die die Verbindungsparameter enthalten (normalerweise zumindest Benutzername und Passwort).

- **User (String)**: Name des Benutzers.
- **Password (String)**: Passwort des Benutzers (wird nicht gespeichert).
- **IsPasswordRequired (Boolean)**: das Passwort ist erforderlich und wird interaktiv vom Benutzer abgefragt.
- **IsReadOnly (Boolean)**: ermöglicht schreibgeschützten Zugriff auf die Datenbank (nur Lesen).
- **NumberFormatsSupplier (Object)**: Objekt mit den für die Datenbank verfügbaren Zahlenformaten (unterstützt die Schnittstelle `com.sun.star.util.XNumberFormatsSupplier`, siehe Abschnitt „Zahlen-, Datums- und Textformat" auf Seite 138).
- **TableFilter (Array)**: Liste der anzuzeigenden Tabellennamen.
- **TableTypeFilter (Array)**: Liste der anzuzeigenden Tabellentypen. Mögliche Werte sind `TABLE`, `VIEW` und `SYSTEM TABLE`.
- **SuppressVersionColumns (Boolean)**: unterdrückt die Anzeige von Spalten, die der Versionsverwaltung dienen.

Hinweis – Die Datenquellen von StarOffice sind nicht 1:1 mit den Datenquellen in ODBC vergleichbar. Während eine ODBC-Datenquelle ausschließlich Informationen über den Ursprung der Daten umfasst, enthält eine Datenquelle in StarOffice darüber hinaus eine Reihe von Informationen zur Darstellung der Daten innerhalb des Datenbankfensters von StarOffice.

Abfragen

Einer Datenquelle lassen sich vordefinierte Abfragen zuordnen. StarOffice merkt sich die SQL-Befehle der Abfragen, so dass sie zu jedem Zeitpunkt zur Verfügung stehen. Abfragen dienen zur Vereinfachung der Arbeit mit Datenbanken, da sie sich mit einem einfachen Mausklick öffnen lassen und auch Anwendern ohne SQL-Kenntnisse die Möglichkeit zum Absetzen von SQL-Befehlen ermöglichen.

Hinter einer Abfrage verbirgt sich ein Objekt, das den Dienst `com.sun.star.sdb.QueryDefinition` unterstützt. Der Zugriff auf die Abfragen erfolgt über die Methode `QueryDefinitions` der Datenquelle.

Im folgenden Beispiel werden die Namen der Datenquellenabfragen, die gebildet werden können, in einem Meldungsfenster aufgelistet.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim QueryDefinitions As Object
Dim QueryDefinition As Object
Dim I As Integer
```

```
DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
QueryDefinitions = DataSource.getQueryDefinitions()

For I = 0 To QueryDefinitions.Count() - 1
   QueryDefinition = QueryDefinitions(I)
   MsgBox QueryDefinition.Name
Next I
```

Neben der im Beispiel verwendeten Name-Eigenschaft hält `com.sun.star.sdb.QueryDefinition` eine ganze Reihe weiterer Eigenschaften bereit. Dies sind:

- **Name (String)**: Name der Abfrage.
- **Command (String)**: SQL-Befehl (normalerweise ein `SELECT`-Befehl).
- **UpdateTableName (String)**: für Abfragen, die auf mehreren Tabellen basieren: Name der Tabelle, in der Wertänderungen möglich sind.
- **UpdateCatalogName (String)**: Name des Update-Tabellenkatalogs.
- **UpdateSchemaName (String)**: Name des Update-Tabellenschemas.

Das folgende Beispiel zeigt, wie sich ein Abfrageobjekt programmgesteuert erstellen und einer Datenquelle zuweisen lässt.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim QueryDefinitions As Object
Dim QueryDefinition As Object
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
QueryDefinitions = DataSource.getQueryDefinitions()

QueryDefinition = createUnoService("com.sun.star.sdb.QueryDefinition")
QueryDefinition.Command = "SELECT * FROM Customer"

QueryDefinitions.insertByName("NewQuery", QueryDefinition)
```

Das Abfrageobjekt wird zunächst über den Aufruf `createUnoService` erzeugt, anschließend initialisiert und dann mit `insertByName` in das `QueryDefinitions`-Objekt eingefügt.

Verknüpfungen mit Datenbankformularen

Um die Arbeit mit Datenquellen zu vereinfachen, bietet StarOffice eine Möglichkeit, die Datenquellen mit Datenbankformularen zu verknüpfen. Die Verknüpfungen stehen über die Methode `getBookmarks()` zur Verfügung. Sie gibt einen benannten Container (`com.sun.star.sdb.DefinitionContainer`) zurück, der alle Verknüpfungen der Datenquelle enthält. Ein Zugriff auf die Lesezeichen ist wahlweise über `Name` oder `Index` möglich.

Das folgende Beispiel ermittelt den URL des Lesezeichens *MyBookmark*.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Bookmarks As Object
Dim URL As String
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
Bookmarks = DataSource.Bookmarks()

URL = Bookmarks.getByName("MyBookmark")
MsgBox URL
```

Datenbankzugriff

Für den Zugriff auf eine Datenbank ist eine Datenbankverbindung notwendig. Dabei handelt es sich um einen Übertragungskanal, der eine direkte Kommunikation mit der Datenbank ermöglicht. Im Gegensatz zu den im vorigen Abschnitt vorgestellten Datenquellen muss die Datenbankverbindung daher bei jedem Programmstart neu hergestellt werden.

StarOffice bietet verschiedene Arten, um Datenbankverbindungen herzustellen. Im Folgenden wird eine Methode erklärt, die eine vorhandene Datenquelle voraussetzt.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Connection As Object
Dim InteractionHandler as Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")

If Not DataSource.IsPasswordRequired Then
   Connection = DataSource.GetConnection("","")
Else
   InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
   Connection = DataSource.ConnectWithCompletion(InteractionHandler)
End If
```

Der Beispielcode prüft zunächst, ob die Datenbank passwortgeschützt ist. Wenn nicht, erzeugt er die notwendige Datenbankverbindung mit dem Aufruf `GetConnection`. Die beiden leeren Zeichenfolgen in der Befehlszeile stehen dabei für den Benutzernamen und das Passwort.

Ist die Datenbank passwortgeschützt, so erzeugt das Beispiel einen `InteractionHandler` und öffnet die Datenbankverbindung mit der Methode `ConnectWithCompletion`. Der InteractionHandler sorgt dafür, dass StarOffice den Benutzer nach den erforderlichen Anmeldedaten fragt.

Iterieren von Tabellen

Der Zugriff auf eine Tabelle erfolgt in StarOffice normalerweise über das `ResultSet`-Objekt. Ein `ResultSet` ist eine Art Markierung, die einen aktuellen Satz von Daten innerhalb einer mit dem `SELECT`-Befehl erhaltenen Ergebnismenge kennzeichnet.

Das Beispiel zeigt, wie sich mit einem `ResultSet` Werte aus einer Datenbanktabelle abfragen lassen.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Connection As Object
Dim InteractionHandler as Object
Dim Statement As Object
Dim ResultSet As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")

If Not DataSource.IsPasswordRequired Then
   Connection = DataSource.GetConnection("","")
Else
   InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
   Connection = DataSource.ConnectWithCompletion(InteractionHandler)
End If

Statement = Connection.createStatement()
ResultSet = Statement.executeQuery("SELECT CustomerNumber FROM Customer")

If Not IsNull(ResultSet) Then
   While ResultSet.next
      MsgBox ResultSet.getString(1)
   Wend
End If
```

Nachdem die Datenbankverbindung hergestellt wurde, erzeugt der Beispielcode zunächst über den Aufruf `Connection.createObject` ein `Statement`-Objekt. Dieses `Statement`-Objekt gibt dann über den Aufruf `executeQuery` den eigentlichen `ResultSet` zurück. Der Programmcode

prüft dann, ob der `ResultSet` tatsächlich vorhanden ist, und geht die Datensätze mittels eine Schleife durch. Die erforderlichen Werte (im Beispiel aus dem Feld `CustomerNumber`) werden von dem `ResultSet` mit der Methode `getString` zurückgegeben, wobei der Parameter 1 festlegt, dass sich der Aufruf auf die Werte der ersten Spalte bezieht.

Hinweis – Das `ResultSet`-Objekt von SDBC ist mit dem `Recordset`-Objekt von DAO beziehungsweise ADO vergleichbar, das ebenfalls einen iterativen Zugriff auf eine Datenbank gewährt.

Hinweis – Der eigentliche Datenbankzugriff erfolgt in StarOffice 8 über ein `ResultSet`-Objekt. Es gibt den Inhalt einer Tabelle beziehungsweise das Ergebnis eines SQL-SELECT-Befehls wieder. Das `ResultSet`-Objekt stellte bisher im `Application`-Objekt angesiedelte Methoden zur Navigation innerhalb der Daten zur Verfügung (z. B. `DataNextRecord`).

Typspezifische Methoden zum Abrufen von Werten

Wie aus dem Beispiel des vorangegangenen Abschnitts ersichtlich, stellt StarOffice für den Zugriff auf Tabelleninhalte eine `getString`-Methode zur Verfügung. Die Methode gibt das Ergebnis in Form einer Zeichenfolge zurück. Folgende `get`-Methoden stehen zur Verfügung:

- `getByte()`: unterstützt die SQL-Datentypen für Zahlen, Zeichen und Zeichenfolgen.
- `getShort()`: unterstützt die SQL-Datentypen für Zahlen, Zeichen und Zeichenfolgen.
- `getInt()`: unterstützt die SQL-Datentypen für Zahlen, Zeichen und Zeichenfolgen.
- `getLong()`: unterstützt die SQL-Datentypen für Zahlen, Zeichen und Zeichenfolgen.
- `getFloat()`: unterstützt die SQL-Datentypen für Zahlen, Zeichen und Zeichenfolgen.
- `getDouble()`: unterstützt die SQL-Datentypen für Zahlen, Zeichen und Zeichenfolgen.
- `getBoolean()`: unterstützt die SQL-Datentypen für Zahlen, Zeichen und Zeichenfolgen.
- `getString()`: unterstützt alle SQL-Datentypen.
- `getBytes()`: unterstützt die SQL-Datentypen für Binärwerte.
- `getDate()`: unterstütz die SQL-Datentypen für Zahlen, Zeichenfolgen, Datum und Zeitstempel (Timestamp).
- `getTime()`: unterstütz die SQL-Datentypen für Zahlen, Zeichenfolgen, Datum und Zeitstempel (Timestamp).
- `getTimestamp()`: unterstütz die SQL-Datentypen für Zahlen, Zeichenfolgen, Datum und Zeitstempel (Timestamp).
- `getCharacterStream()`: unterstützt die SQL-Datentypen für Zahlen, Zeichenfolgen und Binärwerte.

- `getUnicodeStream()`: unterstützt die SQL-Datentypen für Zahlen, Zeichenfolgen und Binärwerte.
- `getBinaryStream()`: Binärwerte.
- `getObject()`: unterstützt alle SQL-Datentypen.

In allen Fällen sollte die Anzahl der Spalten, deren Werte abgefragt werden sollen, als Parameter angegeben werden.

Die ResultSet-Varianten

Bei Datenbankzugriffen ist die Geschwindigkeit häufig ein kritischer Faktor. StarOffice bietet daher einige Möglichkeiten, `ResultSets` zu optimieren und so die Zugriffsgeschwindigkeit zu steuern. Je mehr Funktionen von einem `ResultSet` zur Verfügung gestellt werden, um so komplexer ist dessen Implementierung in der Regel und umso langsamer arbeiten aus diesem Grund die Funktionen.

Ein einfacher `ResultSet`, so wie im Abschnitt "Iterieren von Tabellen" vorgestellt, bietet den kleinst möglichen Funktionsumfang. Er gestattet ausschließlich das vorwärts Iterieren sowie das Abfragen von Werten. Umfassendere Navigationsmöglichkeiten, wie beispielsweise das Ändern von Werten, gehören daher nicht zum Funktionsumfang.

Das für die Erzeugung des `ResultSet` verwendete Statement-Objekt bietet einige Eigenschaften, die es ermöglichen, Einfluss auf die Funktionen des `ResultSet` zu nehmen:

- **ResultSetConcurrency (Const)**: Festlegungen, ob Änderungen der Daten möglich sind (Festlegung gemäß `com.sun.star.sdbc.ResultSetConcurrency`).
- **ResultSetType (Const)**: Festlegungen des Typs der `ResultSets` (Festlegung gemäß `com.sun.star.sdbc.ResultSetType`).

Die in `com.sun.star.sdbc.ResultSetConcurrency` definierten Werte lauten:

- **UPDATABLE**: `ResultSet` lässt Änderungen der Werte zu.
- **READ_ONLY**: `ResultSet` lässt keine Änderungen zu.

Die Konstantengruppe `com.sun.star.sdbc.ResultSetConcurrency` stellt folgende Festlegungen zur Verfügung:

- **FORWARD_ONLY**: `ResultSet` lässt nur Vorwärtsnavigation zu.
- **SCROLL_INSENSITIVE**: `ResultSet` lässt jeden Navigationstyp zu, Änderungen der Originaldaten bleiben jedoch ohne Auswirkung.
- **SCROLL_SENSITIVE**: `ResultSet` lässt jeden Navigationstyp zu, Änderungen der Originaldaten wirken sich auf `ResultSet` aus.

Hinweis – Ein `ResultSet` mit den Eigenschaften `READ_ONLY` und `SCROLL_INSENSITIVE` entspricht einem Recordset des Typs `Snapshot` in ADO beziehungsweise DAO.

Bei Verwendung der `ResultSet`-Eigenschaften `UPDATEABLE` und `SCROLL_SENSITIVE` ist der Funktionsumfang eines `ResultSet` mit einem `Recordset` des Typs `Dynaset` aus ADO beziehungsweise DAO vergleichbar.

Methoden zur Navigation in ResultSets

Hat ein `ResultSet` den Typ `SCROLL_INSENSITIVE` oder `SCROLL_SENSITIVE`, so unterstützt er eine ganze Reihe von Methoden zur Navigation im Datenbestand. Die zentralen Methoden lauten:

- **next()**: Navigation zum nächsten Datensatz.
- **previous()**: Navigation zum vorangehenden Datensatz.
- **first()**: Navigation zum ersten Datensatz.
- **last()**: Navigation zum letzten Datensatz.
- **beforeFirst()**: Navigation bis vor den ersten Datensatz.
- **afterLast()**: Navigation bis hinter den letzten Datensatz.

Alle Methoden geben einen booleschen Parameter zurück, der angibt, ob die Navigation erfolgreich war.

Zur Ermittlung der aktuellen Cursor-Position stehen folgende Prüfmethoden zur Verfügung, die alle einen booleschen Wert zurückgeben:

- **isBeforeFirst()**: `ResultSet` steht vor dem ersten Datensatz.
- **isAfterLast()**: `ResultSet` steht hinter dem letzten Datensatz.
- **isFirst()**: `ResultSet` ist der erste Datensatz.
- **isLast()**: `ResultSet` ist der letzte Datensatz.

Ändern von Datensätzen

Wurde ein `ResultSet` mit dem Wert `ResultSetConcurrency` = `UPDATEABLE` erzeugt, so kann sein Inhalt bearbeitet werden. Dies gilt nur so lange, wie der SQL-Befehl prinzipbedingt ein Rückschreiben der Daten in die Datenbank zulässt. Bei komplexen SQL-Befehlen mit verknüpften Spalten oder akkumulierten Werten ist dies beispielsweise nicht möglich.

Für das Ändern von Werten bietet das `ResultSet`-Objekt `Update`-Methoden, die analog zu den `get`-Methoden zum Abrufen von Werten aufgebaut sind. Die `updateString`-Methode gestattet beispielsweise das Schreiben einer Zeichenfolge (String).

Nach dem Ändern müssen die Werte durch einen Aufruf der `updateRow()`-Methode in die Datenbank übertragen werden. Der Aufruf muss vor dem nächsten Navigationsbefehl erfolgen, da die Werte ansonsten verloren gehen.

Unterläuft bei den Änderungen ein Fehler, so lassen sich diese durch den Aufruf der `cancelRowUpdates()`-Methode verwerfen. Dieser Aufruf steht jedoch nur so lange zur Verfügung, wie die Daten nicht mit `updateRow()` in die Datenbank zurück geschrieben wurden.

◆ ◆ ◆ KAPITEL 11

11 Dialoge

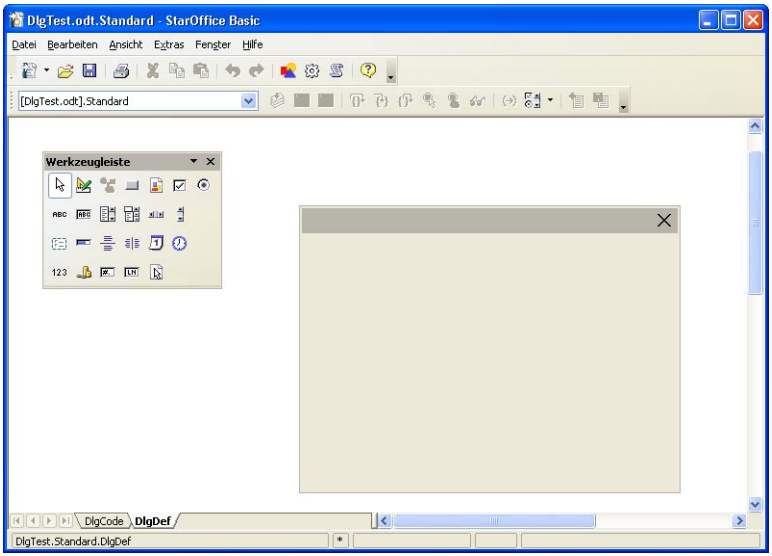
Sie können StarOffice-Dokumente durch benutzerdefinierte Dialogfenster und Formulare erweitern. Diese lassen sich mit StarOffice Basic-Makros verknüpfen, wodurch die Einsatzmöglichkeiten von StarOffice Basic erheblich erweitert werden. In Dialogen können beispielsweise Datenbankinformationen angezeigt oder Anwender schrittweise durch den Erstellungsprozess eines neuen Dokuments in Form eines AutoPiloten geführt werden.

Arbeiten mit Dialogen

StarOffice Basic-Dialoge bestehen aus einem Dialogfenster, das Textfelder, Listenfelder, Optionsfelder und eine Reihe anderer Steuerelemente enthalten kann.

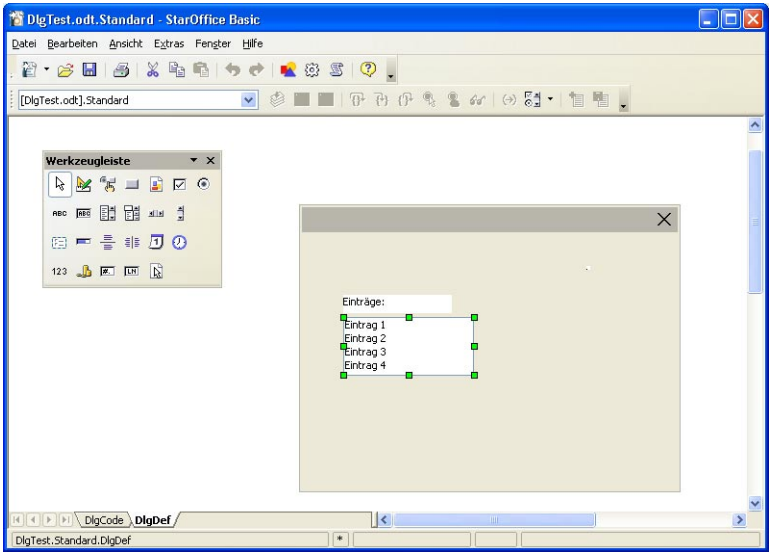
Erstellen von Dialogen

Dialoge können mit Hilfe des StarOffice-Dialog-Editors auf dieselbe Weise erstellt und strukturiert werden, wie mit StarOffice Draw:



Prinzipiell ziehen Sie die gewünschten Steuerelemente aus der Design-Palette (rechts) in den Dialogbereich, wo deren Position und Größe definiert werden können.

Das Beispiel zeigt einen Dialog, der ein Beschriftungs- und ein Listenfeld enthält.



Mit folgendem Code können Sie einen Dialog öffnen:

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.DlgDef)

Dlg.Execute()
Dlg.dispose()
```

`CreateUnoDialog` erzeugt dabei ein Objekt namens `Dlg`, das auf den verknüpften Dialog verweist. Vor dem Erzeugen des Dialogs muss sichergestellt sein, dass die verwendete Bibliothek (im Beispiel `Standard`) geladen ist. Ist dies nicht der Fall, führt die `LoadLibrary`-Methode diese Aufgabe durch.

Ist das Dialog-Objekt `Dlg` initialisiert, so kann der Dialog mit der `Execute`-Methode angezeigt werden. Dialoge wie dieser werden als "modal" bezeichnet, da sie keine weitere Programmaktion zulassen, solange ihr Fenster nicht geschlossen wurde. Während der gesamten Öffnungsphase dieses Dialogs verbleibt das Programm im Aufruf `Execute`.

Die `dispose`-Methode am Ende des Codes gibt die von dem Dialog verwendeten Ressourcen wieder frei, nachdem das Programm beendet wurde.

Schließen von Dialogen

Schließen mit "OK" oder "Abbrechen"

Enthält ein Dialog eine der Schaltflächen **OK** oder **Abbrechen**, wird der Dialog durch Klicken auf eine dieser Schaltflächen automatisch geschlossen. Weitere Informationen zum Arbeiten mit diesen Schaltflächen erhalten Sie in diesem Kapitel unter "Dialog-Steuerelemente im Detail".

Wird ein Dialog unter Verwendung einer Schaltfläche **OK** geschlossen, gibt die `Execute`-Methode einen Rückgabewert von 1 zurück, ansonsten den Wert 0.

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.MyDialog)

Select Case Dlg.Execute()
Case 1
   MsgBox "OK geklickt"
Case 0
   MsgBox "Abbrechen geklickt"
End Select
```

Schließen mit der Schaltfläche "Schließen" in der Titelleiste

Ein Dialog kann gegebenenfalls auch durch Klicken auf die Schaltfläche "Schließen" in der Titelleiste des Dialogfensters geschlossen werden. In diesem Fall gibt die `Execute`-Methode des Dialogs wie beim Klicken auf die Schaltfläche "Abbrechen" den Wert 0 zurück.

Schließen mit einem expliziten Programmaufruf

Ein geöffnetes Dialogfenster kann auch mit der `endExecute`-Methode geschlossen werden:

```
Dlg.endExecute()
```

Zugriff auf einzelne Steuerelemente

Ein Dialog beliebig viele Steuerelemente enthalten. Sie können auf diese Elemente mit der `getControl`-Methode zugreifen, die den Namen des Steuerelements zurückgibt.

```
Dim Ctl As Object

Ctl = Dlg.getControl("MyButton")
Ctl.Label = "New Label"
```

Dieser Code ermittelt das Objekt für das Steuerelement `MyButton` und initialisiert dann die Objektvariable `Ctl` mit einer Referenz auf dieses Element. Anschließend setzt der Code die `Label`-Eigenschaft des Steuerelementes auf den Wert `New Label`.

Hinweis – Bei den Namen von Steuerelementen unterscheidet StarOffice Basic die Groß- und Kleinschreibung.

Arbeiten mit dem *Modell* (Model) von Dialogen und Steuerelementen

Die Unterscheidung zwischen sichtbaren Programmelementen (*Ansicht* (View)) und den dahinter liegenden Daten bzw. Dokumenten (*Modell* (Modell)) findet sich an zahlreichen Stellen in der StarOffice API. Zusätzlich zu den Methoden und Eigenschaften von Steuerelementen verfügen sowohl Dialog- als auch Steuerelementobjekte über ein untergeordnetes `Model`-Objekt. Dieses Objekt gestattet Ihnen den direkten Zugriff auf den Inhalt eines Dialog- oder Steuerelements.

In Dialogen verläuft die Grenze zwischen Daten und Darstellung jedoch nicht immer so klar wie in den anderen API-Bereichen von StarOffice. Elemente der API stehen sowohl in der Ansicht (View) als auch im Modell (Model) zur Verfügung.

Der programmgesteuerte Zugriff auf das Modell von Dialog- und Steuerelementobjekten erfolgt über die Eigenschaft `Model`.

```
Dim cmdNext As Object

cmdNext = Dlg.getControl("cmdNext")
cmdNext.Model.Enabled = False
```

Das Beispiel deaktiviert die Schaltfläche `cmdNtext` innerhalb des Dialogs `Dlg` unter Zuhilfenahme des Model-Objekts von `cmdNtext`.

Eigenschaften

Name und Titel

Jedes Steuerelement verfügt über seinen eigenen Namen, der mit folgender Model-Eigenschaft abgefragt werden kann:

- **Model.Name (String)**: Name des Steuerelements.

Der Titel, der in der Titelleiste angezeigt wird, kann mit folgender Model-Eigenschaft festgelegt werden:

- **Model.Title (String)**: Titel des Dialogs (nur bei Dialogen).

Position und Größe

Größe und Position eines Steuerelements können mit folgenden Eigenschaften des Model-Objekts abgefragt werden:

- **Model.Height (Long)**: Höhe des Steuerelements (in ma-Einheiten).
- **Model.Width (Long)**: Breite des Steuerelements (in ma-Einheiten).
- **Model.PositionX (Long)**: X-Position des Steuerelements, gemessen vom linken Innenrand des Dialogs (in ma-Einheiten).
- **Model.PositionY (Long)**: Y-Position des Steuerelements, gemessen vom oberen Innenrand des Dialogs (in ma-Einheiten).

Damit das Aussehen von Dialogen plattformunabhängig ist, verwendet StarOffice die interne Einheit *Map AppFont (ma)* zur Festlegung von Position und Größe bei Dialogen. Eine ma-Einheit ist definiert als 1/8 der durchschnittlichen Höhe und 1/4 der Breite eines Zeichens der im Betriebssystem definierten Systemschriftart. Durch die Verwendung von ma-Einheiten stellt StarOffice sicher, dass ein Dialog auf unterschiedlichen Systemen und mit verschiedenen Systemeinstellungen stets gleich angezeigt wird.

Sollen Größe oder Position von Steuerelementen zur Laufzeit geändert werden, müssen Sie die Gesamtgröße des Dialogs ermitteln und die Werte für die Steuerelemente an die entsprechenden Teilverhältnisse anpassen.

Hinweis – Die Twips-Einheit wird durch Map AppFont (ma) ersetzt, um eine optimierte Plattformunabhängigkeit zu erreichen.

Fokus und Tabulator-Reihenfolge

Mit der Tabulatortaste können Sie durch die Steuerelemente in einem Dialog navigieren. Folgende Eigenschaften stehen in diesem Zusammenhang im Modell (Model) der Steuerelemente zur Verfügung:

- **Model.Enabled (Boolean)**: aktiviert das Steuerelement.
- **Model.Tabstop (Boolean)**: ermöglicht die Ansteuerung des Steuerelements mit der Tabulatortaste.
- **Model.TabIndex (Long)**: Position des Steuerelements in der Aktivierungsreihenfolge.

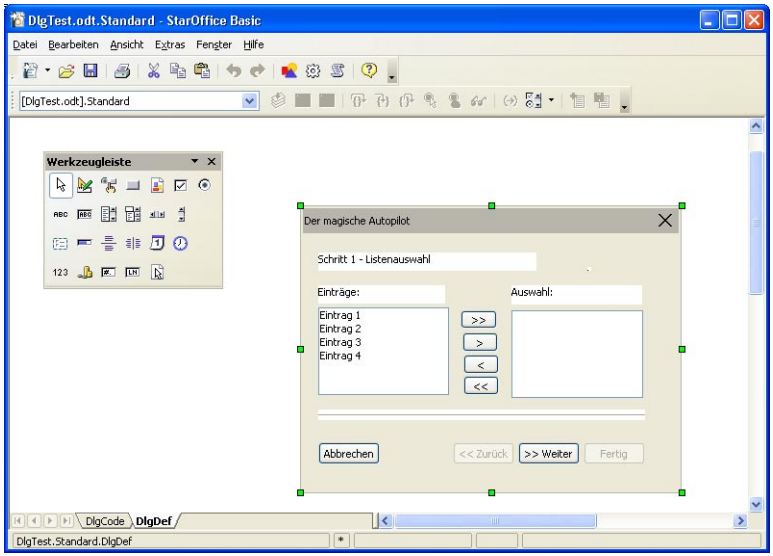
Schließlich bietet das Steuerelement eine `getFocus`-Methode, die sicherstellt, dass das darunter liegende Steuerelement den Fokus erhält:

- **getFocus**: das Steuerelement erhält den Fokus (nur bei Dialogen).

Mehrseitige Dialoge

In StarOffice kann ein Dialog über mehr als eine Registerkarte verfügen. Die `Step`-Eigenschaft eines Dialogs definiert die aktuelle Registerkarte des Dialogs, während die `Step`-Eigenschaft eines Steuerelements die Registerkarte festlegt, auf der das Steuerelement angezeigt werden soll.

Hierbei bildet der `Step`-Wert 0 einen Sonderfall. Wird dieser Wert innerhalb eines Dialogs auf null gesetzt, werden alle Steuerelemente angezeigt, unabhängig von ihrem `Step`-Wert. Ebenso wird ein Steuerelement, wenn dieser Wert für das Element auf null gesetzt wird, auf allen Registerkarten eines Dialogs angezeigt.



Im vorangegangenen Beispiel können Sie also der Trennlinie sowie den Schaltflächen `Abbrechen` (Cancel), `Zurück` (Prev), `Weiter` (Next) und `Fertig stellen` (Done) den `Step`-Wert 0 zuweisen, damit diese Elemente auf allen Seiten angezeigt werden. Die Elemente können auch einer einzelnen Registerkarte zugewiesen werden (z. B. Seite 1).

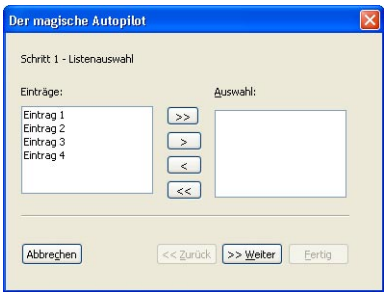
Der folgende Programmcode zeigt, wie der `Step`-Wert in den Ereignis-Handlern der Schaltflächen `Weiter` (Next) und `Zurück` (Prev) erhöht beziehungsweise verringert werden kann, um so den Status der Schaltflächen zu ändern.

```
Sub cmdNext_Initiated
   Dim cmdNext As Object
   Dim cmdPrev As Object

   cmdPrev = Dlg.getControl("cmdPrev")
   cmdNext = Dlg.getControl("cmdNext")

   cmdPrev.Model.Enabled = Not cmdPrev.Model.Enabled
   cmdNext.Model.Enabled = False

   Dlg.Model.Step = Dlg.Model.Step + 1
End Sub

Sub cmdPrev_Initiated
   Dim cmdNext As Object
   Dim cmdPrev As Object

   cmdPrev = Dlg.getControl("cmdPrev")
   cmdNext = Dlg.getControl("cmdNext")
```

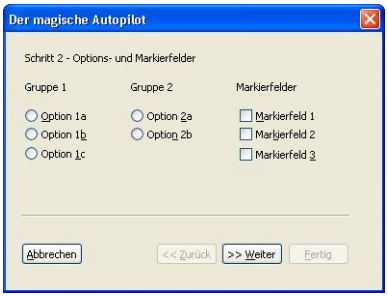
```
   cmdPrev.Model.Enabled = False
   cmdNext.Model.Enabled = True

   Dlg.Model.Step = Dlg.Model.Step - 1
End Sub
```

Das Beispiel setzt voraus, dass eine globale Variable `Dlg` vorhanden ist, die auf einen geöffneten Dialog verweist. Die Anzeige des Dialog ändert sich wie folgt:

Seite 1:



Seite 2:



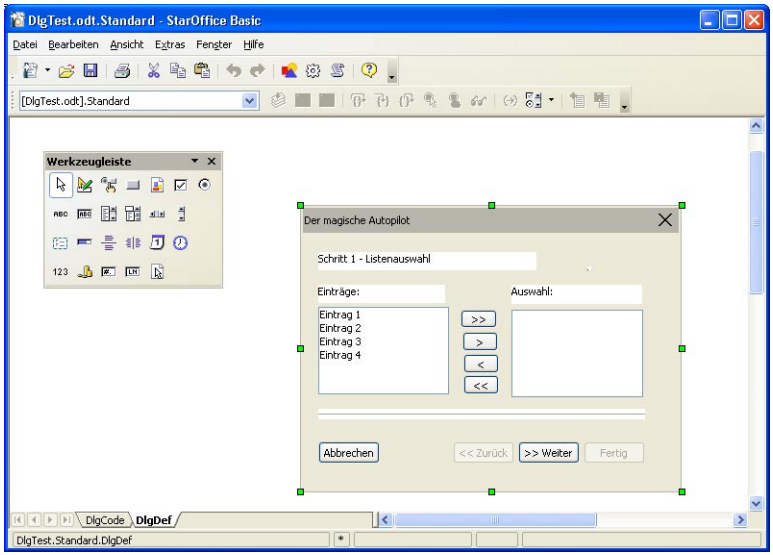
Ereignisse

Dialoge und Formulare basieren in StarOffice auf einem ereignisorientierten Programmiermodell, in dem den Steuerelementen *Ereignis-Handler* zugewiesen werden können. Ein Ereignis-Handler führt bei Eintritt einer bestimmten Aktion eine vordefinierte Prozedur aus, auch wenn es sich bei der Aktion um ein anderes Ereignis handelt. Durch Ereignisbehandlung können sowohl Dokumente oder geöffnete Datenbanken bearbeitet als auch auf andere Steuerelemente zugegriffen werden.

StarOffice-Steuerelemente erkennen verschiedene Arten von Ereignissen, die in jeweils unterschiedlichen Situationen ausgelöst werden können. Diese Ereignisarten lassen sich in vier Gruppen unterteilen:

- **Maussteuerung**: Ereignisse, die Aktionen mit der Maus entsprechen (zum Beispiel einfache Mausbewegungen oder Klicks auf eine bestimmte Stelle des Bildschirms).
- **Tastatursteuerung**: Ereignisse, die durch Betätigung von Tasten auf der Tastatur ausgelöst werden.
- **Fokusänderung**: Ereignisse, die StarOffice beim Aktivieren oder Deaktivieren von Steuerelementen ausführt.
- **Steuerelementspezifische Ereignisse**: Ereignisse, die nur im Zusammenhang mit bestimmten Steuerelementen auftreten.

Bei der Arbeit mit Ereignissen müssen Sie sicherstellen, dass der dazugehörige Dialog in der Entwicklungsumgebung von StarOffice erstellt wird und dass er die erforderlichen Steuerelemente bzw. Dokumente enthält (wenn die Ereignisse auf ein Formular Anwendung finden).



Die vorangehende Abbildung zeigt die Entwicklungsumgebung von StarOffice Basic mit einem Dialogfenster, das zwei Listenfelder enthält. Sie können die Daten mit Hilfe der zwischen den Listenfeldern liegenden Schaltflächen von der einen in die andere Liste verschieben.

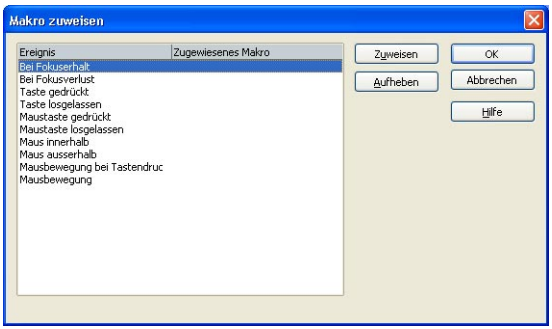
Wenn Sie am Bildschirm das Layout anzeigen möchten, müssen Sie die verknüpften StarOffice Basic-Prozeduren erstellen, damit diese von den Event-Handlern aufgerufen werden können. Auch wenn diese Prozeduren in beliebigen Modulen verwendet werden können, ist es das Beste, ihre Verwendung auf zwei Module zu beschränken. Um die Lesbarkeit des Programmcodes zu verbessern, sollten Sie den Prozeduren "sprechende" Namen geben. Das direkte Anspringen allgemeiner Programm-Prozeduren aus einem Makro heraus führt in der Regel zu unübersichtlichem Programmcode. Sie sollten stattdessen eine weitere Prozedur

erstellen, die als Einstiegspunkt für die Ereignisbehandlung dient, selbst wenn nur ein einziger Aufruf der Zielprozedur erfolgt. Hierdurch werden die Pflege und das Debuggen des Programmcodes deutlich erleichtert.

Der folgende Beispielcode verschiebt einen Eintrag aus dem linken in das rechte Listenfeld eines Dialogs.

```
Sub cmdSelect_Initiated
   Dim objList As Object
   lstEntries = Dlg.getControl("lstEntries")
   lstSelection = Dlg.getControl("lstSelection")

   If lstEntries.SelectedItem > 0 Then
      lstSelection.AddItem(lstEntries.SelectedItem, 0)
      lstEntries.removeItems(lstEntries.SelectItemPos, 1)
   Else
      Beep
   End If
End Sub
```

Wenn diese Prozedur in StarOffice Basic erstellt wurde, können Sie sie über das Eigenschaftenfenster des Dialog-Editors einem erforderlichen Ereignis zuweisen.



Im Zuordnungsdialog werden alle StarOffice Basic-Prozeduren aufgeführt. Um einem Ereignis eine Prozedur zuzuweisen, wählen Sie die Prozedur aus und klicken auf **Assign** (Zuweisen).

Parameter

Das Eintreten eines bestimmten Ereignisses reicht nicht immer aus, um eine angemessene Reaktion auszulösen. Es können zusätzliche Informationen notwendig sein. So wird zur Verarbeitung eines Mausklicks eventuell die Bildschirmposition benötigt, an der die Maustaste gedrückt wurde.

In StarOffice Basic können Sie Objektparameter verwenden, um weitere Informationen über ein Ereignis an eine Prozedur zu übergeben, zum Beispiel:

```
Sub ProcessEvent(Event As Object)

End Sub
```

Wie differenziert das `Event`-Objekt aufgebaut ist und welche Eigenschaften es enthält, hängt von der Art des Ereignisses ab, das den Prozeduraufruf auslöst. In den folgenden Abschnitten werden die Ereignisarten detailliert behandelt.

Unabhängig vom Typ des Ereignisses gewähren alle Objekte Zugriff auf das jeweilige Steuerelement und sein Modell (Model). Auf das Steuerelement greifen Sie mit

```
Event.Source
```

zu und auf sein Modell (Model) mit

```
Event.Source.Model
```

Mit diesen Eigenschaften können Sie in einem Event-Handler ein Ereignis auslösen.

Maus-Ereignisse

StarOffice Basic erkennt folgende Mausereignisse:

- **Mouse moved**: der Anwender bewegt die Maus.
- **Mouse moved while key pressed**: der Anwender zieht die Maus bei gedrückter, beliebiger Maustaste.
- **Mouse button pressed**: der Anwender drückt eine Maustaste.
- **Mouse button released**: der Anwender lässt eine Maustaste los.
- **Mouse outside**: der Anwender bewegt die Maus aus dem aktuellen Fenster heraus.

Der Aufbau des verknüpften Ereignisobjekts wird in der Struktur `com.sun.star.awt.MouseEvent` definiert, die folgende Informationen bereitstellt:

- **Buttons (Short)**: gedrückte Taste (eine oder mehrere Konstanten gemäß `com.sun.star.awt.MouseButton`).
- **X (Long)**: X-Koordinate der Maus, gemessen in Pixel von der linken oberen Ecke des Steuerelements.
- **Y (Long)**: Y-Koordinate der Maus, gemessen in Pixel von der linken oberen Ecke des Steuerelements.
- **ClickCount (Long)**: Anzahl der Klicks, die mit dem Mausereignis verknüpft sind (wenn StarOffice ausreichend schnell reagieren kann, ist `ClickCount` auch bei einem Doppelklick 1, da nur ein Einzelereignis ausgelöst wird).

Die in `com.sun.star.awt.MouseButton` definierten Konstanten für die Maustasten lauten:

- **LEFT**: linke Maustaste.
- **RIGHT**: rechte Maustaste.
- **MIDDLE**: mittlere Maustaste.

Das folgende Beispiel gibt die Position der Maus sowie die Maustaste, die gedrückt wurde, aus:

```
Sub MouseUp(Event As Object)
   Dim Msg As String
   Msg = "Keys: "
   If Event.Buttons AND com.sun.star.awt.MouseButton.LEFT Then
      Msg = Msg & "LEFT "
   End If
   If Event.Buttons AND com.sun.star.awt.MouseButton.RIGHT Then
      Msg = Msg & "RIGHT "
   End If
   If Event.Buttons AND com.sun.star.awt.MouseButton.MIDDLE Then
      Msg = Msg & "MIDDLE "
   End If
   Msg = Msg & Chr(13) & "Position: "
   Msg = Msg & Event.X & "/" & Event.Y
   MsgBox Msg
End Sub
```

Hinweis – Die VBA-Ereignisse `Click` und `Doubleclick` stehen in StarOffice Basic nicht zur Verfügung. Verwenden Sie stattdessen anstelle des `Click`-Ereignisses das StarOffice Basic-Ereignis `MouseUp` und bilden Sie das `Doubleclick`-Ereignis durch eine Änderung der Anwendungslogik nach.

Tastatur-Ereignisse

Folgende Tastaturereignisse stehen in StarOffice Basic zur Verfügung:

- **Key pressed**: der Anwender drückt eine Taste.
- **Key released**: der Anwender lässt eine Taste los.

Beide Ereignisse beziehen sich auf *logische* Tastendrücke und nicht auf *physikalische* Aktionen. Wenn der Anwender mehrere Tasten drückt, um nur ein Zeichen auszugeben (z. B. um ein Akzentzeichen hinzuzufügen), erzeugt StarOffice Basic nur ein Ereignis.

Ein einzelner Druck auf eine Modifizierer-Taste wie die Umschalt- oder Alt-Taste verursacht kein unabhängiges Ereignis.

Informationen über eine gedrückte Taste werden im Event-Objekt bereitgestellt, das StarOffice Basic für die Ereignisbehandlung an die Prozedur übergibt. Es enthält folgende Eigenschaften:

- **KeyCode (Short)**: Code der gedrückten Taste (Standardwerte gemäß `com.sun.star.awt.Key`)

- **KeyChar (String)**: eingegebenes Zeichen (unter Berücksichtigung von Modifizierer-Tasten).

Das folgende Beispiel ermittelt mit Hilfe der Eigenschaft `KeyCode`, ob die Eingabetaste, die Tabulatortaste oder eine der anderen Steuerungstasten gedrückt wurde. Wurde eine dieser Tasten gedrückt, wird der Name der Taste zurückgegeben, andernfalls das eingegebene Zeichen:

```
Sub KeyPressed(Event As Object)
   Dim Msg As String
   Select Case Event.KeyCode
   Case com.sun.star.awt.Key.RETURN
      Msg = "Return pressed"
   Case com.sun.star.awt.Key.TAB
      Msg = "Tab pressed"
   Case com.sun.star.awt.Key.DELETE
      Msg = "Delete pressed"
   Case com.sun.star.awt.Key.ESCAPE
      Msg = "Escape pressed"
   Case com.sun.star.awt.Key.DOWN
      Msg = "Down pressed"
   Case com.sun.star.awt.Key.UP
      Msg = "Up pressed"
   Case com.sun.star.awt.Key.LEFT
      Msg = "Left pressed"
   Case com.sun.star.awt.Key.RIGHT
      Msg = "Right pressed"
   Case Else
      Msg = "Character " & Event.KeyChar & " entered"
   End Select
   MsgBox Msg
End Sub
```

Informationen zu weiteren Tastatur-Konstanten finden Sie in der API-Referenz unter der Konstantengruppe `com.sun.star.awt.Key`.

Fokus-Ereignisse

Fokus-Ereignisse zeigen an, ob ein Steuerelement den Fokus erhält oder verliert. Mit diesen Ereignissen können Sie beispielsweise bestimmen, ob ein Anwender die Bearbeitung eines Steuerelements abgeschlossen hat, damit Sie nun andere Elemente eines Dialogs aktualisieren können. Folgende Fokus-Ereignisse sind verfügbar:

- **When receiving focus**: Element erhält den Fokus.
- **When losing focus**: Element verliert den Fokus.

Die `Event`-Objekte der Fokus-Ereignisse sind wie folgt aufgebaut:

- **FocusFlags (Short)**: Ursache des Fokuswechsels (Standardwert gemäß `com.sun.star.awt.FocusChangeReason`).
- **NextFocus (Object)**: Objekt, das den Fokus erhält (nur für das `When losing focus`-Ereignis).
- **Temporary (Boolean)**: der Fokus wurde vorübergehend verloren.

Steuerelementspezifische Ereignisse

Neben den vorangehenden Ereignissen, die von allen Steuerelementen unterstützt werden, sind einige steuerelementspezifische Ereignisse vorhanden, die nur für bestimmten Steuerelemente definiert sind. Bei den wichtigsten dieser Ereignisse handelt es sich um:

- **When Item Changed**: der Wert eines Steuerelements hat sich geändert.
- **Item Status Changed**: der Status eines Steuerelements hat sich geändert.
- **Text modified**: der Text eines Steuerelements hat sich geändert.
- **When initiating**: eine Aktion, die bei Auslösung des Steuerelements ausgeführt werden kann (z. B. das Drücken einer Taste).

Beachten Sie bei der Arbeit mit Ereignissen, dass manche Ereignisse, wie z. B. das `When initiating`-Ereignis, jedes Mal ausgelöst werden können, wenn mit der Maus auf bestimmte Steuerelemente geklickt wird (z. B. auf Optionsfelder). Dabei wird mit keiner Aktion geprüft, ob sich der Status des Steuerelements tatsächlich geändert hat. Um solche "Blindereignisse" zu vermeiden, speichern Sie den alten Wert des Steuerelements in einer globalen Variable und prüfen dann bei der Ausführung eines Ereignisses, ob er sich tatsächlich geändert hat.

Die Eigenschaften des `Item Status Changed`-Ereignisses lauten:

- **Selected (long)**: aktuell gewählter Eintrag.
- **Highlighted (long)**: aktuell markierter Eintrag.
- **ItemId (long)**: ID des Eintrags.

Dialog-Steuerelemente im Detail

StarOffice Basic erkennt eine Reihe von Steuerelementen, die in folgende Gruppen unterteilt werden können:

Eingabefelder:

- Textfelder
- Datumsfelder
- Zeitfelder
- Zahlenfelder
- Währungsfelder

- Frei formatierbare Felder

Schaltflächen:

- Standard-Schaltflächen
- Kontrollkästchen
- Optionsfelder

Auswahllisten:

- Listenfelder
- Kombinationsfelder

Sonstige Steuerelemente:

- Bildlaufleisten (Scrollbars; horizontal und vertikal)
- Gruppenfelder
- Fortschrittsbalken
- Trennlinien (horizontal und vertikal)
- Grafiken
- Dateiauswahlfelder

Die wichtigsten dieser Steuerelemente werden im Folgenden vorgestellt.

Schaltflächen

Eine Schaltfläche führt eine Aktion aus, wenn darauf geklickt wird.

Im einfachsten Fall löst eine Schaltfläche nur ein `When Initiating`-Ereignis aus, wenn der Anwender darauf klickt. Sie können auch eine andere Aktion mit der Schaltfläche verknüpfen, um einen Dialog zu öffnen, indem Sie die `PushButtonType`-Eigenschaft verwenden. Wenn Sie auf eine Schaltfläche klicken, für die diese Eigenschaft auf den Wert 0 gesetzt ist, wirkt sich dies nicht auf den Dialog aus. Wenn Sie auf eine Schaltfläche klicken, für die diese Eigenschaft auf den Wert 1 gesetzt ist, wird der Dialog geschlossen und die `Execute`-Methode des Dialogs gibt den Wert 1 zurück (die Dialogsequenz wurde korrekt abgeschlossen). Ist die `PushButtonType`-Eigenschaft auf den Wert 2 gesetzt, wird der Dialog geschlossen und die `Execute`-Methode des Dialogs gibt den Wert 0 zurück (Dialog geschlossen).

Im Folgenden finden Sie alle Eigenschaften, die im Schaltflächen-Modell (Model) zur Verfügung stehen:

- **Model.BackgroundColor (Long)**: Hintergrundfarbe.
- **Model.DefaultButton (Boolean)**: Die Schaltfläche wird als Standardwert verwendet und reagiert auf die Eingabetaste, wenn sie nicht den Fokus hat.
- **Model.FontDescriptor (Struct)**: Struktur, die die Details der zu verwendenden Schriftart angibt (gemäß der Struktur `com.sun.star.awt.FontDescriptor`).

- **Model.Label (String)**: Beschriftung, die auf der Schaltfläche angezeigt wird.
- **Model.Printable (Boolean)**: das Steuerelement kann gedruckt werden.
- **Model.TextColor (Long)**: Textfarbe des Steuerelements.
- **Model.HelpText (String)**: Hilfetext, der angezeigt wird, wenn der Mauszeiger über das Steuerelement bewegt wird.
- **Model.HelpURL (String)**: URL der Online-Hilfe für das entsprechende Steuerelement.
- **PushButtonType (Short)**: mit der Schaltfläche verknüpfte Aktion (0: keine Aktion, 1: OK, 2: Abbrechen).

Optionsschaltflächen

Optionsschaltflächen werden gewöhnlich in Gruppen eingesetzt und gestatten es dem Anwender, eine von mehreren Optionen auszuwählen. Bei Auswahl einer Option werden alle anderen in der Gruppe enthaltenen Optionen deaktiviert. Hierdurch wird sichergestellt, dass zu jedem Zeitpunkt immer nur genau eine Optionsschaltfläche aktiviert ist.

Ein Optionsschaltflächen-Steuerelement stellt zwei Eigenschaften zur Verfügung:

- **State (Boolean)**: aktiviert die Schaltfläche.
- **Label (String)**: Beschriftung, die auf der Schaltfläche angezeigt wird.

Sie können auch die folgenden Eigenschaften aus dem Modell (Model) der Optionsschaltflächen verwenden:

- **Model.FontDescriptor (struct)**: Struktur mit Details der zu verwendenden Schriftart (gemäß `com.sun.star.awt.FontDescriptor`).
- **Model.Label (String)**: Beschriftung, die auf dem Steuerelement angezeigt wird.
- **Model.Printable (Boolean)**: Steuerelement kann gedruckt werden.
- **Model.State (Short)**: wenn diese Eigenschaft gleich 1 ist, ist die Option aktiviert, andernfalls ist sie deaktiviert.
- **Model.TextColor (Long)**: Textfarbe des Steuerelements.
- **Model.HelpText (String)**: Hilfetext, der angezeigt wird, wenn der Mauszeiger über dem Steuerelement verharrt.
- **Model.HelpURL (String)**: URL der Online-Hilfe für das entsprechende Steuerelement.

Sollen mehrere Optionsschaltflächen zu einer Gruppe zusammengefasst werden, müssen sie sequenziell und lückenlos in der Aktivierungsreihenfolge positioniert werden (Eigenschaft `Model.TabIndex`, im Dialog-Editor als "Order" (Reihenfolge) bezeichnet). Wird die Aktivierungsreihenfolge durch ein anderes Steuerelement unterbrochen, so beginnt StarOffice automatisch mit einer neuen Steuerelementegruppe, die unabhängig von der ersten Steuerelementegruppe aktiviert werden kann.

Hinweis – Im Gegensatz zu VBA können in StarOffice Basic einer Gruppe von Steuerelementen keine Optionsschaltflächen hinzugefügt werden. In StarOffice Basic werden Steuerelemente nur gruppiert, um eine optische Gliederung bzw. Abgrenzung zu gewährleisten, indem ein Rahmen um die Steuerelemente gezeichnet wird.

Kontrollkästchen

Kontrollkästchen werden zur Erfassung eines Ja- oder Nein-Werts verwendet. In Abhängigkeit von ihrem Modus können sie zwei oder drei Zustände annehmen. Zusätzlich zu den Zuständen "Ja" und "Nein" kann sich ein Kontrollkästchen in einem Zwischenzustand befinden, wenn der entsprechende Ja- oder Nein-Status mehr als eine Bedeutung hat oder unklar ist.

Kontrollkästchen stellen folgende Eigenschaften bereit:

- **State (Short)**: Zustand des Kontrollkästchens (0: Nein, 1: Ja, 2: Zwischenzustand).
- **Label (String)**: Beschriftung des Steuerelements.
- **enableTriState (Boolean)**: zusätzlich zu den Zuständen "aktiviert" und "deaktiviert" kann auch der Zwischenzustand verwendet werden.

Das Model-Objekt eines Kontrollkästchens bietet folgende Eigenschaften:

- **Model.FontDescriptor (struct)**: Struktur mit Details der verwendeten Schriftart (gemäß der Struktur `com.sun.star.awt.FontDescriptor`).
- **Model.Label (String)**: Beschriftung des Steuerelements.
- **Model.Printable (Boolean)**: das Steuerelement kann gedruckt werden.
- **Model.State (Short)**: Zustand des Kontrollkästchens (0: Nein, 1: Ja, 2: Zwischenzustand).
- **Model.Tabstop (Boolean)**: ermöglicht die Ansteuerung des Steuerelements mit der Tabulatortaste.
- **Model.TextColor (Long)**: Textfarbe des Steuerelements.
- **Model.HelpText (String)**: Hilfetext, der angezeigt wird, wenn der Mauszeiger über dem Steuerelement verharrt.
- **Model.HelpURL (String)**: URL der Online-Hilfe für das entsprechende Steuerelement.

Textfelder

Textfelder ermöglichen dem Anwender die Eingabe von Zahlen und Text. Die Grundlage für Textfelder bildet der Dienst `com.sun.star.awt.UnoControlEdit`.

Ein Textfeld kann einzeilig oder mehrzeilig sein, editierbar oder für die Benutzereingabe gesperrt. Textfelder können auch als spezielle Währungs- oder Zahlenfelder sowie als

Bildschirmfelder für bestimmte Aufgaben verwendet werden. Da diese Steuerelemente auf dem Uno-Dienst `UnoControlEdit` basieren, ist ihre programmgesteuerte Handhabung weitgehend identisch.

Textfelder stellen folgende Eigenschaften bereit:

- **Text (String)**: aktueller Text.
- **SelectedText (String)**: aktuell markierter Text.
- **Selection (Struct)**: schreibgeschützte Markierung von Details (Struktur gemäß `com.sun.star.awt.Selection`, wobei die Eigenschaften `Min` und `Max` den Anfang und das Ende der aktuellen Markierung festlegen).
- **MaxTextLen (Short)**: maximale Anzahl von Zeichen, die in das Feld eingegeben werden können.
- **Editable (Boolean)**: `True` aktiviert die Option für Texteingabe, `False` sperrt die Eingabeoption (die Eigenschaft kann nicht direkt aufgerufen werden, nur über `IsEditable`).
- **IsEditable (Boolean)**: der Inhalt des Steuerelements kann geändert werden; schreibgeschützt.

Über das verknüpfte Model-Objekt hinaus stehen folgende Eigenschaften zur Verfügung:

- **Model.Align (Short)**: Ausrichtung des Texts (0: linksbündig, 1: zentriert, 2: rechtsbündig).
- **Model.BackgroundColor (Long)**: Hintergrundfarbe des Steuerelements.
- **Model.Border (Short)**: Art des Rahmens (0: kein Rahmen, 1: 3D-Rahmen, 2: einfacher Rahmen).
- **Model.EchoChar (String)**: Echo-Zeichen für Passwortfelder.
- **Model.FontDescriptor (Struct)**: Struktur mit Details der verwendeten Schriftart (gemäß der Struktur `com.sun.star.awt.FontDescriptor`).
- **Model.HardLineBreaks (Boolean)**: automatische Zeilenumbrüche werden dauerhaft in den Text des Steuerelements eingefügt.
- **Model.HScroll (Boolean)**: der Text verfügt über eine horizontale Bildlaufleiste (Scrollbar).
- **Model.MaxTextLen (Short)**: maximale Textlänge, wobei 0 unbegrenzte Textlänge zulässt.
- **Model.MultiLine (Boolean)**: lässt mehrzeilige Eingabe zu.
- **Model.Printable (Boolean)**: das Steuerelement kann gedruckt werden.
- **Model.ReadOnly (Boolean)**: der Inhalt des Steuerelements ist schreibgeschützt (nur Lesen).
- **Model.Tabstop (Boolean)**: ermöglicht die Ansteuerung des Steuerelements mit der Tabulatortaste.
- **Model.Text (String)**: mit dem Steuerelement verknüpfter Text.
- **Model.TextColor (Long)**: Textfarbe des Steuerelements.

- **Model.VScroll (Boolean)**: der Text verfügt über eine vertikale Bildlaufleiste (Scrollbar).
- **Model.HelpText (String)**: Hilfetext, der angezeigt wird, wenn der Mauszeiger über dem Steuerelement verharrt.
- **Model.HelpURL (String)**: URL der Online-Hilfe für das entsprechende Steuerelement.

Listenfelder

Listenfelder (Dienst `com.sun.star.awt.UnoControlListBox`) unterstützen folgende Eigenschaften:

- **ItemCount (Short)**: Anzahl der Elemente; schreibgeschützt (nur Lesen).
- **SelectedItem (String)**: Text des markierten Eintrags; schreibgeschützt (nur Lesen).
- **SelectedItems (Array Of Strings)**: Datenfeld mit markierten Einträgen; schreibgeschützt (nur Lesen).
- **SelectItemPos (Short)**: Nummer des aktuell markierten Eintrags; schreibgeschützt (nur Lesen).
- **SelectItemsPos (Array of Short)**: Datenfeld mit den Positionsnummern der markierten Einträge (für Listen, die Mehrfachauswahl unterstützen); schreibgeschützt (nur Lesen).
- **MultipleMode (Boolean)**: `True` aktiviert die Option für Mehrfachauswahl von Einträgen, `False` sperrt die Mehrfachauswahl (die Eigenschaft kann nicht direkt aufgerufen werden, nur über `IsMultipleMode`).
- **IsMultipleMode (Boolean)**: lässt Mehrfachauswahl innerhalb von Listen zu; schreibgeschützt (nur Lesen).

Listenfelder stellen folgende Methoden zur Verfügung:

- **addItem (Item, Pos)**: gibt die in `Item` angegebene Zeichenfolge an Position `Pos` in der Liste ein.
- **addItems (ItemArray, Pos)**: gibt die in dem Datenfeld `ItemArray` der Zeichenfolge aufgeführten Einträge an Position `Pos` in der Liste ein.
- **removeItems (Pos, Count)**: entfernt `Count` Einträge ab Position `Pos`.
- **selectItem (Item, SelectMode)**: aktiviert oder deaktiviert die Markierung für das in der Zeichenfolge `Item` angegebene Element, in Abhängigkeit von der booleschen Variable `SelectMode`.
- **makeVisible (Pos)**: führt einen Bildlauf durch das Listenfeld aus, damit der mit `Pos` angegebene Eintrag angezeigt wird.

Das Model-Objekt der Listenfelder hält folgende Eigenschaften bereit:

- **Model.BackgroundColor (Long)**: Hintergrundfarbe des Steuerelements.
- **Model.Border (Short)**: Art des Rahmens (0: kein Rahmen, 1: 3D-Rahmen, 2: einfacher Rahmen).

- **Model.FontDescriptor (Struct)**: Struktur mit Details der verwendeten Schriftart (gemäß der Struktur `com.sun.star.awt.FontDescriptor`).
- **Model.LineCount (Short)**: Anzahl der Zeilen im Steuerelement.
- **Model.MultiSelection (Boolean)**: lässt die Mehrfachauswahl von Einträgen zu.
- **Model.SelectedItems (Array of Strings)**: Liste der markierten Einträge.
- **Model.StringItemList (Array of Strings)**: Liste aller Einträge.
- **Model.Printable (Boolean)**: das Steuerelement kann gedruckt werden.
- **Model.ReadOnly (Boolean)**: der Inhalt des Steuerelements ist schreibgeschützt (nur Lesen).
- **Model.Tabstop (Boolean)**: ermöglicht die Ansteuerung des Steuerelements mit der Tabulatortaste.
- **Model.TextColor (Long)**: Textfarbe des Steuerelements.
- **Model.HelpText (String)**: Hilfetext, der automatisch angezeigt wird, wenn sich der Mauszeiger über dem Steuerelement befindet.
- **Model.HelpURL (String)**: URL der Online-Hilfe für das entsprechende Steuerelement.

Hinweis – Die VBA-Option zum Versehen von Listeneinträgen mit einem numerischen Zusatzwert (`ItemData`) ist in StarOffice Basic nicht vorhanden. Soll neben dem Klartext ein zusätzlicher Zahlenwert (etwa eine Datenbank-Id) mit verwaltet werden, so muss hierfür ein Hilfsdatenfeld erstellt werden, das parallel zum Listenfeld verwaltet wird.

◆ ◆ ◆ KAPITEL 12

12 Formulare

Der Aufbau von StarOffice-Formularen entspricht in vielerlei Hinsicht den im vorangehenden Kapitel besprochenen Dialogen. Es existieren jedoch einige wesentliche Unterschiede:

- Dialoge werden in Form eines einzelnen Dialogfensters angezeigt, das über dem Dokument eingeblendet wird und bis zum Abschluss der Bearbeitung des Dialogs keine anderen Aktionen als die Bearbeitung des Dialogs zulässt. Formulare hingegen werden wie Zeichnungselemente direkt im Dokument angezeigt.
- Für die Erstellung von Dialogen steht ein eigener Dialog-Editor zur Verfügung, der sich in der StarOffice Basic-Entwicklungsumgebung befindet. Formulare werden direkt im Dokument mit der **Formularfunktionen-Symbolleiste** erstellt.
- Während die Dialog-Funktionen in allen StarOffice-Dokumenten zur Verfügung stehen, ist der volle Umfang der Formularfunktionen ausschließlich in Text- und Tabellendokumenten verfügbar.
- Die Steuerelemente eines Formulars lassen sich mit einer externen Datenbanktabelle verknüpfen. In Dialogen steht diese Funktionalität nicht zur Verfügung.
- Auch die Steuerelemente von Dialogen und Formularen unterscheiden sich in einigen Punkten.

Anwender, die ihre Formulare mit eigenen Methoden zur Ereignisbehandlung ausstatten möchten, seien auf das Kapitel 11, Dialoge, verwiesen. Die dort erläuterten Mechanismen sind mit denen für Formulare identisch.

Arbeiten mit Formularen

StarOffice-Formulare können Textfelder, Listenfelder, Optionsfelder und eine Reihe anderer Steuerelemente, die direkt in ein Text- oder Tabellendokument eingefügt werden, enthalten. Für die Bearbeitung von Formularen wird die **Formularfunktionen-Symbolleiste** verwendet.

Ein StarOffice-Formular kann sich in einem von zwei Modi befinden: im Entwurfsmodus und im Anzeigemodus. Im Entwurfsmodus ist es möglich, die Position von Steuerelementen zu verändern und ihre Eigenschaften über ein Eigenschaftenfenster zu bearbeiten.

Der Wechsel zwischen den Modi erfolgt ebenfalls über die **Formularfunktionen-Symbolleiste**.

Ermitteln von Formular-Objekten

StarOffice positioniert die Steuerelemente eines Formulars auf der Zeichnungsobjektebene. Das eigentliche Formular-Objekt ist über die Forms-Auflistung der Zeichnungsebene erreichbar. In Textdokumenten erfolgt der Zugriff auf die Objekte wie folgt:

```
Dim Doc As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
DrawPage = Doc.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

Die Methode `GetByIndex` gibt dabei das Formular mit der Indexnummer 0 zurück.

Bei Tabellendokumenten bedarf es des Zwischenschritts über die Sheets-Auflistung, da sich die Zeichnungsebenen nicht direkt im Dokument, sondern in den einzelnen Tabellen befinden:

```
Dim Doc As Object
Dim Sheet As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.GetByIndex(0)
DrawPage = Sheet.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

Wie der Name der Methode `GetByIndex` bereits nahe legt, kann ein Dokument mehrere Formulare enthalten. Dies ist beispielsweise sinnvoll, wenn die Inhalte verschiedener Datenbanken innerhalb eines Dokuments angezeigt werden sollen oder wenn eine 1:n-Beziehung einer Datenbank innerhalb eines Formulars angezeigt werden soll. Zu diesem Zweck besteht auch die Möglichkeit, Unterformulare zu erstellen.

Die drei Aspekte eines Formular-Steuerelements

Ein Steuerelement eines Formulars verfügt über drei Aspekte:

- Zunächst einmal gibt es das *Model* (Modell) des Steuerelements. Es stellt das Kernobjekt für den StarOffice Basic-Programmierer bei der Arbeit mit Formular-Steuerelementen dar.
- Das Gegenstück dazu bildet die *View* (Ansicht) des Steuerelements, die die Anzeigeinformationen verwaltet.
- Da Formular-Steuerelemente innerhalb der Dokumente wie ein spezielles Zeichnungselement verwaltet werden, ist außerdem ein *Shape-Objekt* vorhanden, das die zeichnungselementspezifischen Eigenschaften des Steuerelements widerspiegelt (insbesondere seine Position und Größe).

Zugriff auf das Modell (Model) von Formular-Steuerelementen

Die Modelle (Models) der Steuerelemente eines Formulars sind über die Methode `GetByName` des Formularobjekts verfügbar:

```
Dim Doc As Object
Dim Form As Object
Dim Ctl As Object

Doc = StarDesktop.CurrentComponent
Form = Doc.DrawPage.Forms.GetByIndex(0)
Ctl = Form.getByName("MyListBox")
```

Das Beispiel ermittelt das Modell (Model) des Steuerelements `MyListBox`, das sich im ersten Formular des aktuell geöffneten Textdokuments befindet.

Ist unklar, in welchem Formular sich ein Steuerelement befindet, so besteht die Möglichkeit, alle Formulare nach dem gewünschten Steuerelement zu durchsuchen:

```
Dim Doc As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim I as Integer

Doc = StarDesktop.CurrentComponent
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
   Form = Forms.GetbyIndex(I)
   If Form.HasByName("MyListBox") Then
      Ctl = Form.GetbyName("MyListBox")
      Exit Function
   End If
Next I
```

Das Beispiel überprüft mit der Methode `HasByName` alle Formulare eines Textdokuments daraufhin, ob sie ein Steuerelement-Modell (Model) namens `MyListBox` enthalten. Wird ein entsprechendes Modell (Model) gefunden, wird eine Referenz darauf in der Variable `Ctl` gespeichert und die Suche wird abgebrochen.

Zugriff auf die Ansicht (View) von Formular-Steuerelementen

Für den Zugriff auf die Ansicht (View) eines Formular-Steuerelements wird zunächst das verknüpfte Modell (Model) benötigt. Mit dessen Hilfe und über den Controller des Dokuments lässt sich anschließend die Ansicht (View) des Steuerelements ermitteln.

```
Dim Doc As Object
Dim DocCrl As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim CtlView As Object
Dim I as Integer

Doc = StarDesktop.CurrentComponent
DocCrl = Doc.getCurrentControler()
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
   Form = Forms.GetbyIndex(I)
   If Form.HasByName("MyListBox") Then
      Ctl = Form.GetbyName("MyListBox")
      CtlView = DocCrl.GetControl(Ctl)
      Exit Function
   End If
Next I
```

Der aufgeführte Beispielcode orientiert sich stark an dem vorhergehenden Beispielcode zur Ermittlung eines Steuerelement-Modells (Model). Er verwendet neben dem Dokumentobjekt `Doc` jedoch auch das Dokument-Controler-Objekt `DocCrl`, das auf das aktuelle Dokumentenfenster verweist. Mit Hilfe dieses Controler-Objekts und dem Modell (Model) des Steuerelements ermittelt es schließlich über die Methode `GetControl` die Ansicht (View; Variable `CtlView`) des Formular-Steuerelements.

Zugriff auf das Shape-Objekt von Formular-Steuerelementen

Auch für den Zugriff auf die Shape-Objekte eines Steuerelements führt der Weg über die betreffende Zeichnungsebene des Dokuments. Zur Ermittlung eines speziellen Steuerelements müssen sämtliche Zeichnungselemente der Zeichnungsebene durchsucht werden.

```
Dim Doc As Object
Dim Shape as Object
Dim I as integer

Doc = StarDesktop.CurrentComponent

For i = 0 to Doc.DrawPage.Count - 1
   Shape = Doc.DrawPage(i)

   If HasUnoInterfaces(Shape, _
      "com.sun.star.drawing.XControlShape") Then
      If Shape.Control.Name = "MyListBox" Then
            Exit Function
      End If
   End If
Next
```

Das Beispiel überprüft alle Zeichnungselemente, ob sie die für Formular-Steuerelemente erforderliche Schnittstelle `com.sun.star.drawing.XControlShape` unterstützen. Ist dies der Fall, so überprüft die Eigenschaft `Control.Name` dann, ob der Name des Steuerelements `MyListBox` lautet. Ist dies der Fall (Wahr), beendet die Funktion die Suche.

Ermitteln der Größe und Position von Steuerelementen

Wie bereits erwähnt, lässt sich die Größe und Position von Steuerelementen über das verknüpfte `Shape`-Objekt ermitteln. Hierzu stellt das Steuerelement-Shape wie alle anderen `Shape`-Objekte die Eigenschaften `Size` und `Position` zur Verfügung:

- **Size (struct)**: Größe des Steuerelements (Datenstruktur `com.sun.star.awt.Size`).
- **Position (struct)**: Position des Steuerelements (Datenstruktur `com.sun.star.awt.Point`).

Das folgende Beispiel zeigt, wie Position und Größe eines Steuerelements über das verknüpfte Shape-Objekt gesetzt werden können:

```
Dim Shape As Object

Point.x = 1000
Point.y = 1000
Size.Width = 10000
```

```
Size.Height = 10000

Shape.Size = Size
Shape.Position = Point
```

Damit der Code funktionsfähig ist, muss das Shape-Objekt des Steuerelements bereits bekannt sein. Ist dies nicht der Fall, muss es über den weiter oben stehenden Code ermittelt werden.

Formular-Steuerelemente im Detail

Die in Formularen verfügbaren Steuerelemente ähneln denen von Dialogen. Die Auswahl reicht von einfachen Textfeldern über Listen- und Kombinationsfelder bis hin zu verschiedenen Schaltflächen.

Im Folgenden finden Sie eine Aufstellung der wichtigsten Eigenschaften für Formular-Steuerelemente. Sämtliche Eigenschaften sind Bestandteil der jeweiligen Model-Objekte.

Über die Standardsteuerelemente hinaus steht für Formulare ein Tabellensteuerelement zur Verfügung, das die vollständige Einbindung von Datenbanktabellen gestattet. Es wird im Abschnitt „Datenbank-Formulare" auf Seite 229 in Kapitel 12, behandelt.

Schaltflächen

Das Model-Objekt einer Formular-Schaltfläche hält folgende Eigenschaften bereit:

- **BackgroundColor (Long)**: Hintergrundfarbe.
- **DefaultButton (Boolean)**: die Schaltfläche dient als Standardwert. Sie reagiert in diesem Fall auch dann auf die Eingabetaste, wenn sie keinen Fokus besitzt.
- **Enabled (Boolean)**: das Steuerelement kann aktiviert werden.
- **Tabstop (Boolean)**: auf das Steuerelement kann mit der Tabulatortaste zugegriffen werden.
- **TabIndex (Long)**: Position des Steuerelements in der Aktivierungsreihenfolge.
- **FontName (String)**: Name der Schriftart.
- **FontHeight (Single)**: Höhe der Zeichen in Punkt (p).
- **Tag (String)**: Zeichenfolge mit zusätzlichen Informationen, die in der Schaltfläche für programmgesteuerten Zugriff gespeichert werden können.
- **TargetURL (String)**: Ziel-URL für Schaltflächen des Typs "URL".
- **TargetFrame (String)**: Name des Fensters (oder Rahmens (Frames)) in dem `TargetURL` geöffnet werden soll, wenn die Schaltfläche aktiviert wird (für Schaltflächen des Typs `URL`).
- **Label (String)**: Beschriftung der Schaltfläche.

- **TextColor (Long)**: Textfarbe des Steuerelements.
- **HelpText (String)**: Hilfetext, der automatisch angezeigt wird, wenn sich der Mauszeiger über dem Steuerelement befindet.
- **HelpURL (String)**: URL der Online-Hilfe für das entsprechende Steuerelement.
- **ButtonType (Enum)**: mit der Schaltfläche verknüpfte Aktion (Standardwert gemäß `com.sun.star.form.FormButtonType`).

Über die Eigenschaft `ButtonType` können Sie eine Aktion definieren, die beim Klicken auf die Schaltfläche automatisch ausgeführt wird. Die verknüpfte Konstantengruppe `com.sun.star.form.FormButtonType` stellt folgende Werte bereit:

- **PUSH**: Standardschaltfläche.
- **SUBMIT**: Ende des Formulareintrags (insbesondere relevant für HTML-Formulare).
- **RESET**: setzt alle Werte innerhalb des Formulars auf ihre Ausgangswerte zurück.
- **URL**: Aufruf des in `TargetURL` definierten URLs (wird innerhalb des Fensters geöffnet, das über `TargetFrame` festgelegt ist).

Die in Dialogen vorhandenen Schaltflächentypen **OK** und **Abbrechen** werden in Formularen nicht unterstützt.

Optionsschaltflächen

Folgende Eigenschaften einer Optionsschaltfläche sind über deren Model-Objekt verfügbar:

- **Enabled (Boolean)**: das Steuerelement kann aktiviert werden.
- **Tabstop (Boolean)**: auf das Steuerelement kann mit der Tabulatortaste zugegriffen werden.
- **TabIndex (Long)**: Position des Steuerelements in der Aktivierungsreihenfolge.
- **FontName (String)**: Name der Schriftart.
- **FontHeight (Single)**: Höhe der Zeichen in Punkt (p).
- **Tag (String)**: Zeichenfolge mit zusätzlichen Informationen, die in der Schaltfläche für programmgesteuerten Zugriff gespeichert werden können.
- **Label (String)**: Beschriftung der Schaltfläche.
- **Printable (Boolean)**: das Steuerelement kann gedruckt werden.
- **State (Short)**: wenn 1, ist die Option aktiviert, andernfalls ist sie deaktiviert.
- **RefValue (String)**: Zeichenfolge zum Speichern zusätzlicher Informationen (z. B. zur Verwaltung von Datensatz-IDs).
- **TextColor (Long)**: Textfarbe des Steuerelements.
- **HelpText (String)**: Hilfetext, der automatisch angezeigt wird, wenn sich der Mauszeiger über dem Steuerelement befindet.

- **HelpURL (String)**: URL der Online-Hilfe für das entsprechende Steuerelement.

Der Mechanismus zum Gruppieren von Optionsschaltflächen unterscheidet zwischen den Steuerelementen für Dialoge und Formulare. Während in Dialogen aufeinander folgende Steuerelemente automatisch zu einer Gruppe zusammengefasst werden, erfolgt die Gruppierung in Formularen aufgrund des Namens. Hierzu müssen alle Optionsschaltflächen einer Gruppe denselben Name enthalten. StarOffice fasst die so gruppierten Steuerelemente in einem Array zusammen, so dass die einzelnen Schaltflächen immer noch von einem StarOffice-Programm aus erreichbar bleiben.

Das folgende Beispiel zeigt, wie sich das Modell (Model) einer Steuerelementegruppe ermitteln lässt.

```
Dim Doc As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim I as Integer

Doc = StarDesktop.CurrentComponent
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
   Form = Forms.GetbyIndex(I)
   If Form.HasByName("MyOptions") Then
      Ctl = Form. GetGroupbyName("MyOptions")
      Exit Function
   End If
Next I
```

Der Code entspricht dem weiter oben aufgeführten Beispiel zur Ermittlung eines einfachen Steuerelement-Modells (Model). In einer Schleife durchsucht er alle Formulare des aktuell geöffneten Textdokuments und prüft über die Methode `HasByName`, ob das entsprechende Formular ein Element mit dem gesuchten Namen `MyOptions` enthält. Ist dies der Fall, erfolgt der Zugriff auf den Model-Array über die Methode `GetGroupByName` (anstelle der Methode `GetByName` zur Ermittlung einfacher Modelle (Models)).

Kontrollkästchen

Das Model-Objekt eines Formular-Kontrollkästchens stellt folgende Eigenschaften zur Verfügung:

- **Enabled (Boolean)**: das Steuerelement kann aktiviert werden.
- **Tabstop (Boolean)**: auf das Steuerelement kann mit der Tabulatortaste zugegriffen werden.
- **TabIndex (Long)**: Position des Steuerelements in der Aktivierungsreihenfolge.

- **FontName (String)**: Name der Schriftart.
- **FontHeight (Single)**: Höhe der Zeichen in Punkt (p).
- **Tag (String)**: Zeichenfolge mit zusätzlichen Informationen, die in der Schaltfläche für programmgesteuerten Zugriff gespeichert werden können.
- **Label (String)**: Beschriftung der Schaltfläche.
- **Printable (Boolean)**: das Steuerelement kann gedruckt werden.
- **State (Short)**: wenn 1, ist die Option aktiviert, andernfalls ist sie deaktiviert.
- **RefValue (String)**: Zeichenfolge zum Speichern zusätzlicher Informationen (z. B. zur Verwaltung von Datensatz-IDs).
- **TextColor (Long)**: Textfarbe des Steuerelements.
- **HelpText (String)**: Hilfetext, der automatisch angezeigt wird, wenn sich der Mauszeiger über dem Steuerelement befindet.
- **HelpURL (String)**: URL der Online-Hilfe für das entsprechende Steuerelement.

Textfelder

Die Model-Objekte von Formular-Textfeldern verfügen über folgende Eigenschaften:

- **Align (Short)**: Ausrichtung des Texts (0: linksbündig, 1: zentriert, 2: rechtsbündig).
- **BackgroundColor (long)**: Hintergrundfarbe des Steuerelements.
- **Border (Short)**: Art des Rahmens (0: kein Rahmen, 1: 3D-Rahmen, 2: einfacher Rahmen).
- **EchoChar (String)**: Echo-Zeichen für Passwortfelder.
- **FontName (String)**: Name der Schriftart.
- **FontHeight (Single)**: Höhe der Zeichen in Punkt (p).
- **HardLineBreaks (Boolean)**: automatische Zeilenumbrüche werden dauerhaft in den Text des Steuerelements eingefügt.
- **HScroll (Boolean)**: der Text verfügt über eine horizontale Bildlaufleiste (Scrollbar).
- **MaxTextLen (Short)**: maximale Länge von Text; bei Festlegung von 0 ist die Länge unbegrenzt.
- **MultiLine (Boolean)**: ermöglicht mehrzeilige Einträge.
- **Printable (Boolean)**: das Steuerelement kann gedruckt werden.
- **ReadOnly (Boolean)**: der Inhalt des Steuerelements ist schreibgeschützt (nur Lesen).
- **Enabled (Boolean)**: das Steuerelement kann aktiviert werden.
- **Tabstop (Boolean)**: auf das Steuerelement kann mit der Tabulatortaste zugegriffen werden.
- **TabIndex (Long)**: Position des Steuerelements in der Aktivierungsreihenfolge.
- **FontName (String)**: Name der Schriftart.

- **FontHeight (Single)**: Höhe der Zeichen in Punkt (p).
- **Text (String)**: Text des Steuerelements.
- **TextColor (Long)**: Textfarbe des Steuerelements.
- **VScroll (Boolean)**: der Text verfügt über eine vertikale Bildlaufleiste (Scrollbar).
- **HelpText (String)**: Hilfetext, der automatisch angezeigt wird, wenn sich der Mauszeiger über dem Steuerelement befindet.
- **HelpURL (String)**: URL der Online-Hilfe für das entsprechende Steuerelement.

Listenfelder

Das Model-Objekt der Formular-Listenfelder stellt folgende Eigenschaften bereit:

- **BackgroundColor (long)**: Hintergrundfarbe des Steuerelements.
- **Border (Short)**: Art des Rahmens (0: kein Rahmen, 1: 3D-Rahmen, 2: einfacher Rahmen).
- **FontDescriptor (struct)**: Struktur mit Details der zu verwendenden Schriftart (gemäß Struktur `com.sun.star.awt.FontDescriptor`).
- **LineCount (Short)**: Anzahl der Zeilen des Steuerelements.
- **MultiSelection (Boolean)**: lässt die Mehrfachauswahl von Einträgen zu.
- **SelectedItems (Array of Strings)**: Liste der markierten Einträge.
- **StringItemList (Array of Strings)**: Liste aller Einträge.
- **ValueItemList (Array of Variant)**: Liste mit zusätzlichen Informationen für jeden Eintrag (z. B. zur Verwaltung von Datensatz-IDs).
- **Printable (Boolean)**: das Steuerelement kann gedruckt werden.
- **ReadOnly (Boolean)**: der Inhalt des Steuerelements ist schreibgeschützt (nur Lesen).
- **Enabled (Boolean)**: das Steuerelement kann aktiviert werden.
- **Tabstop (Boolean)**: auf das Steuerelement kann mit der Tabulatortaste zugegriffen werden.
- **TabIndex (Long)**: Position des Steuerelements in der Aktivierungsreihenfolge.
- **FontName (String)**: Name der Schriftart.
- **FontHeight (Single)**: Höhe der Zeichen in Punkt (p).
- **Tag (String)**: Zeichenfolge mit zusätzlichen Informationen, die in der Schaltfläche für programmgesteuerten Zugriff gespeichert werden können.
- **TextColor (Long)**: Textfarbe des Steuerelements.
- **HelpText (String)**: Hilfetext, der automatisch angezeigt wird, wenn sich der Mauszeiger über dem Steuerelement befindet.
- **HelpURL (String)**: URL der Online-Hilfe für das entsprechende Steuerelement.

Hinweis – Formular-Listenfelder bieten mit der Eigenschaft `ValueItemList` ein Gegenstück zu der VBA-Eigenschaft `ItemData`, über die Sie Zusatzinformationen für einzelne Listeneinträge verwalten können.

Zusätzlich stehen über das View-Objekt des Listenfelds folgende Methoden zur Verfügung:

- **addItem (Item, Pos)**: fügt die in `Item` angegebene Zeichenfolge an Position `Pos` in der Liste ein.
- **addItems (ItemArray, Pos)**: fügt die in dem Datenfeld `ItemArray` der Zeichenfolge aufgeführten Einträge an Position `Pos` in der Liste ein.
- **removeItems (Pos, Count)**: entfernt `Count` Einträge ab Position `Pos`.
- **selectItem (Item, SelectMode)**: aktiviert oder deaktiviert die Markierung für das in der Zeichenfolge `Item` angegebene Element, in Abhängigkeit von der Variable `SelectMode`.
- **makeVisible (Pos)**: führt einen Bildlauf durch das Listenfeld aus, damit der mit `Pos` angegebene Eintrag angezeigt wird.

Datenbank-Formulare

StarOffice-Formulare können direkt mit einer Datenbank verknüpft werden. Die so entstandenen Formulare bieten sämtliche Funktionen eines vollwertigen Datenbank-Frontends, ohne dass dafür eigene Programmierarbeit notwendig wäre.

Der Anwender hat die Möglichkeit, die ausgewählten Tabellen beziehungsweise Abfragen zu durchblättern und zu durchsuchen sowie Datensätze zu ändern und neue Datensätze einzufügen. StarOffice stellt automatisch sicher, dass die jeweils relevanten Daten aus der Datenbank abgerufen und eventuelle Änderungen zurückgeschrieben werden.

Ein Datenbank-Formular entspricht im Wesentlichen einem StarOffice-Standardformular. Über die Standardeigenschaften hinaus müssen im Formular folgende datenbankspezifischen Eigenschaften festgelegt werden:

- **DataSourceName (String)**: Name der Datenquelle (siehe Kapitel 10; die Datenquelle muss in StarOffice global erstellt sein).
- **Command (String)**: Name der Tabelle, Abfrage oder des SQL-Select-Befehls, mit der/dem eine Verknüpfung hergestellt werden soll.
- **CommandType (Const)**: gibt an, ob es sich bei Command um eine Tabelle, eine Abfrage oder einen SQL-Befehl handelt (Wert aus Aufzählung `com.sun.star.sdb.CommandType`).

Die Aufzählung `com.sun.star.sdb.CommandType` umfasst folgende Werte:

- **TABLE**: Tabelle.
- **QUERY**: Abfrage.

- **COMMAND**: SQL-Befehl.

Die Zuordnung der Datenbankfelder zu den einzelnen Steuerelementen erfolgt über diese Eigenschaft:

- **DataField (String)**: Name des verknüpften Datenbankfelds.

Tabellen

Für die Arbeit mit Datenbanken steht noch ein weiteres Steuerelement zur Verfügung: das Tabellen-Steuerelement. Es stellt den Inhalt einer vollständigen Datenbanktabelle oder -abfrage dar. Im einfachsten Fall erfolgt die Verknüpfung eines Tabellen-Steuerelements mit einer Datenbank über den Formular-Autopiloten, der alle Spalten gemäß den Benutzervorgaben mit den entsprechenden Datenbankfeldern verknüpft. Auf eine vollständige Beschreibung der API wird an dieser Stelle aus Gründen der Komplexität verzichtet.

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z